**ВВЕДЕНИЕ**

В данной пояснительной записке рассматривается описание программного модуля «Занятость актёров театра» на основе объектно-ориентированного подхода.

При объектно-ориентированном подходе программа представляет собой описание объектов, их свойств (или атрибутов), совокупностей (или классов), отношений между ними, способов их взаимодействия и операций над объектами (или методов).

Несомненным преимуществом данного подхода является концептуальная близость к предметной области произвольной структуры и назначения. Механизм наследования атрибутов и методов позволяет строить производные понятия на основе базовых и таким образом создавать модель сколь угодно сложной предметной области с заданными свойствами.

Еще одним теоретически интересным и практически важным свойством объектно-ориентированного подхода является поддержка механизма обработки событий, которые изменяют атрибуты объектов и моделируют их взаимодействие в предметной области.

В отличие от предыдущих подходов к программированию, объектно-ориентированный подход требует глубокого понимания основных принципов, или, иначе, концепций, на которых он базируется. К числу основополагающих понятий ООП обычно относят абстракцию данных, наследование, инкапсуляцию и полиморфизм.

Объектно-ориентированное программирование в настоящее время является абсолютным лидером в области прикладного программирования.

В качестве основного инструмента разработки применяется Microsoft Visual Studio 2019. Язык программирования C#.

**ЦЕЛЬ И ЗАДАЧИ ВЫПОЛНЕНИЯ РАБОТЫ**

Целью данного курсового проекта является автоматизация фирм по продаже запчастей на основе объектно-ориентированного подхода, а также получение навыков в реализации этого подхода, проектировании и реализации схемы данных.

Задачи выполнения работы:

* исследование предметной области с целью выявления основных сущностей и их атрибутов;
* проектирование иерархии классов и интерфейсов на основе выделенных сущностей;
* использование наследования и агрегации для структуризации классов;
* применение принципа инкапсуляции к классам;
* использование интерфейсов, описывающих операции в данной предметной области, для обеспечения множественного наследования;
* тестирование созданных классов в клиентском приложении, автоматизирующем деятельность предметной области;
* документация разработанных классов при помощи комментариев.

**1 ВНЕШНЕЕ ОПИСАНИЕ**

**1.1 Анализ предметной области**

Анализ предметной области – важнейший этап разработки программного обеспечения.

Анализ занятости актёров театра позволяет определить ключевые понятия, выяснить задачи, требующие решения в рамках проекта, проанализировать функциональные требования будущего продукта (набор функций, предоставляемых программой) и нефункциональные (надёжность, эффективность, изучаемость, модифицируемость и т.п.).

Компонентами данной предметной области являются поставщики, детали и поставки. Данное приложение должно позволять пользователю создавать записи, удалять, редактировать, сохранять и открывать файлы, создаваемые программой Microsoft SQL Server Management Studio 20.1.

Нефункциональные требования к программному средству:

* надежность: программа должна быть автономной;
* эффективность: программа должна иметь минимальные требования к аппаратному обеспечению. Для использования данного программного средства не должна требоваться установка на компьютер. Не должно требоваться дополнительных периферийных средств (принтер, сканер, дополнительные дисководы и т.д.);
* изучаемость: программа должна быть интуитивно понятна, иметь удобный пользовательский интерфейс;
* модифицируемость: программа должна быть легко модифицируемой вследствие небольшого исходного размера и объектно-ориентированного подхода.

**1.2. Функциональная спецификация**

Минимальные технические характеристики компьютера, на котором гарантируется стабильная работа программы:

* компьютер/процессор: компьютер с процессором класса Pentium II 450 МГц;
* память: 64 МБ ОЗУ;
* монитор: монитор VGA с разрешением 800x600 точек или более высоким, поддерживающий 256 цветов;
* операционная система: операционная система Windows XP с установленным Microsoft .NET Framework 4.5.2;
* наличие свободного дискового пространства на жёстком диске.

Внешние функции:

* внесение данных и редактирование данных о поставщиках, деталях и поставках;
* сохранение данных в mdf -файл;
* считывание данных из mdf -файла;
* проверка вводимых данных и вывод сообщений об ошибках;
* программа должна выполнять все функции за предельно малое количество времени;

Критические случаи:

* отсутствие свободного места на жёстком диске. Любая операция с файловой системой должна быть отменена;
* некорректный mdf-файл. Программа не должна аварийно завершаться;
* завершение работы Windows. В этом случае программа должна завершать свою работу, не применяя выбранные пользователем настройки.

**2 ТЕХНИЧЕСКОЕ ЗАДАНИЕ**

Техническое задание определяет требования к будущему программному модулю со стороны основных процессов.

Система предназначена для решения перечисленных ниже задач:

* изменение списка актёров (добавление, удаление, редактирование);
* изменение списка спектаклей (добавление, удаление, редактирование);
* изменение списка занятости (добавление, удаление, редактирование);
* отображение информации, хранящейся в mdf –файле;
* сохранение информации в mdf –файл.

**3 ПРОГРАММИРОВАНИЕ**

**3.1 Описание классов**

В ООП главным элементом является класс, включающий множество объектов с одинаковыми свойствами, операциями и отношениями. Класс имеет внутреннее (реализацию) и внешнее представление – интерфейс. В данной программе реализованы классы Actors, Performances, Enployment, а также интерфейс MainWindow.

Класс Actor описывает актёра. Содержит такие свойства, как номер, имя, фамилию, отчество, звание, стаж актёра.

public partial class Actor

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Actor()

{

this.Enployments = new HashSet<Enployment>();

}

public int ActorID { get; set; }

public string Name { get; set; }

public string Surname { get; set; }

public string Patronymic { get; set; }

public string Rank { get; set; }

public string Experience { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Enployment> Enployments { get; set; }

}

Класс Performance описывает спектакль. Содержит такие свойства, как номе, название, год, бюджет спектакля.

public partial class Performance

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Performance()

{

this.Enployments = new HashSet<Enployment>();

}

public int PerformanceID { get; set; }

public string Title { get; set; }

public string Year { get; set; }

public string Budget { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Enployment> Enployments { get; set; }

}

Класс Enployment описывает занятость актёра. Содержит такие свойства, как номер актёра, номер спектакля, роль, стоимость годового контракта актёра.

public partial class Enployment

{

public int EnploymentID { get; set; }

public int ActorID { get; set; }

public int PerformanceID { get; set; }

public string Role { get; set; }

public string Cost { get; set; }

public virtual Actor Actor { get; set; }

public virtual Performance Performance { get; set; }

}

Класс DbPartialClass для создания соединения с базой данных

public partial class ACompanySellingSparePartsEntities

{

private static ACompanySellingSparePartsEntities \_context;

public static ACompanySellingSparePartsEntities GetContext()

{

if (\_context == null)

\_context = new ACompanySellingSparePartsEntities();

return \_context;

}

}

**3.2 Схема данных**

На рисунках 1-3 представлены таблицы файла mdf в режиме конструктора.

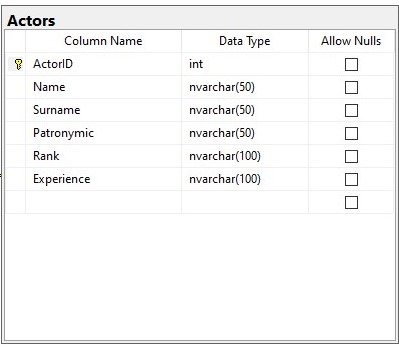


Рисунок 1 – Таблица Actors

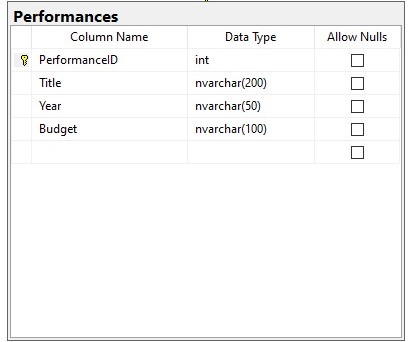


Рисунок 2 – Таблица Performances

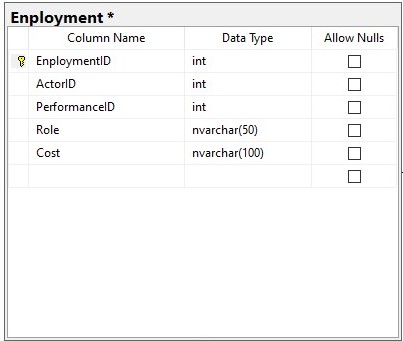


Рисунок 3 – Таблица Enployment

На рисунке 4 представлена схема данных

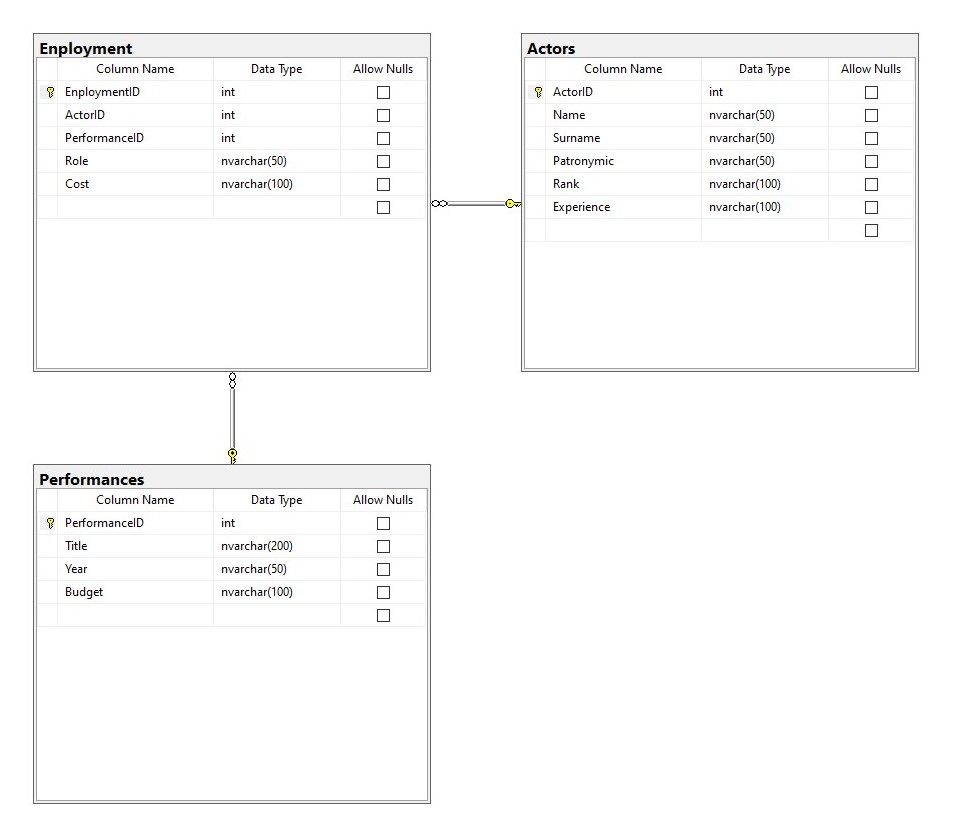


Рисунок 4-Схема данных

Словарь данных представлен в таблицах 1-3.

Таблица 1 – Словарь данных таблицы Actors

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Actors (Актёры) | | | | |
| KEY | FIELD NAME | DATA TYPE | REQURIED? | NOTES |
| PK | ActorID | INT | Y | Autoincrement |
|  | Name | NVARCHAR | Y |  |
|  | Surname | NVARCHAR | Y |  |
|  | Patronymic | NVARCHAR | Y |  |
|  | Rank | NVARCHAR | Y |  |
|  | Experience | NVARCHAR | Y |  |

Таблица 2 – Словарь данных таблицы Performance

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Performance (Спектакль) | | | | |
| KEY | FIELD NAME | DATA TYPE | REQURIED? | NOTES |
| PK | PerformanceID | INT | Y | Autoincrement |
|  | Title | NVARCHAR | Y |  |
|  | Year | NVARCHAR | Y |  |
|  | Budget | NVARCHAR | Y |  |

Таблица 3 – Словарь данных таблицы Enployment

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Enployment (Занятость актёров) | | | | |
| KEY | FIELD NAME | DATA TYPE | REQURIED? | NOTES |
| PK | EnploymentID | INT | Y | Autoincrement |
| FK | ActorID | INT | Y | Внешний ключ на Actors |
| FK | PerformanceID | INT | Y | Внешний ключ на Performance |
|  | Role | NVARCHAR | Y |  |
|  | Cost | NVARCHAR | Y |  |

**4 КЛИЕНТСКОЕ ПРИЛОЖЕНИЕ**

**4.1 Структура**

В данном ПП для продуктивной работы реализован удобный пользовательский интерфейс, похожий на большинство Windows – приложений. Данное программное средство можно разбить на модули, взаимодействующие друг с другом. На рисунке 5 представлена структура приложения:

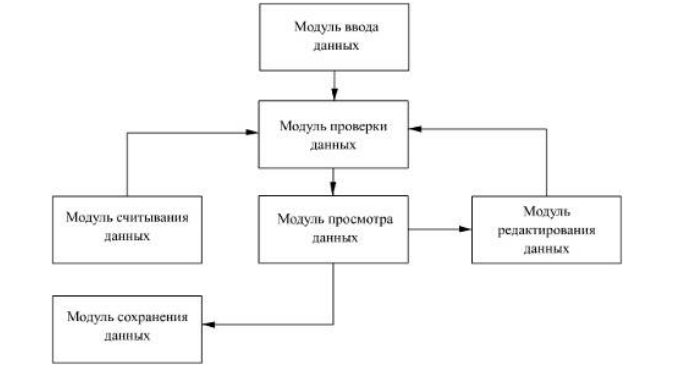
****

Рисунок 5 – Структура Приложения

Модуль просмотра данных предоставляет возможность наглядного отображения данных.

Модуль проверки данных реализует методы:

* корректность вводимых пользователем данных;
* корректность считываемых из файла данных.

Модуль ввода данных, реализует простой графический интерфейс для ввода необходимых значений.

Модуль сохранения данных, формирует все записи в один mdf - файл.

Модуль считывания предоставляет возможность открытия mdf - файла и заполнения списков в программе.

Модуль редактирования записей предоставляет простой интерфейс для изменения ранее внесенных данных, а также добавления и удаления.

**4.2 Проектирование графического интерфейса**

На рисунке 6-8 представлена форма приложения и панель, содержащая таблицы данных:

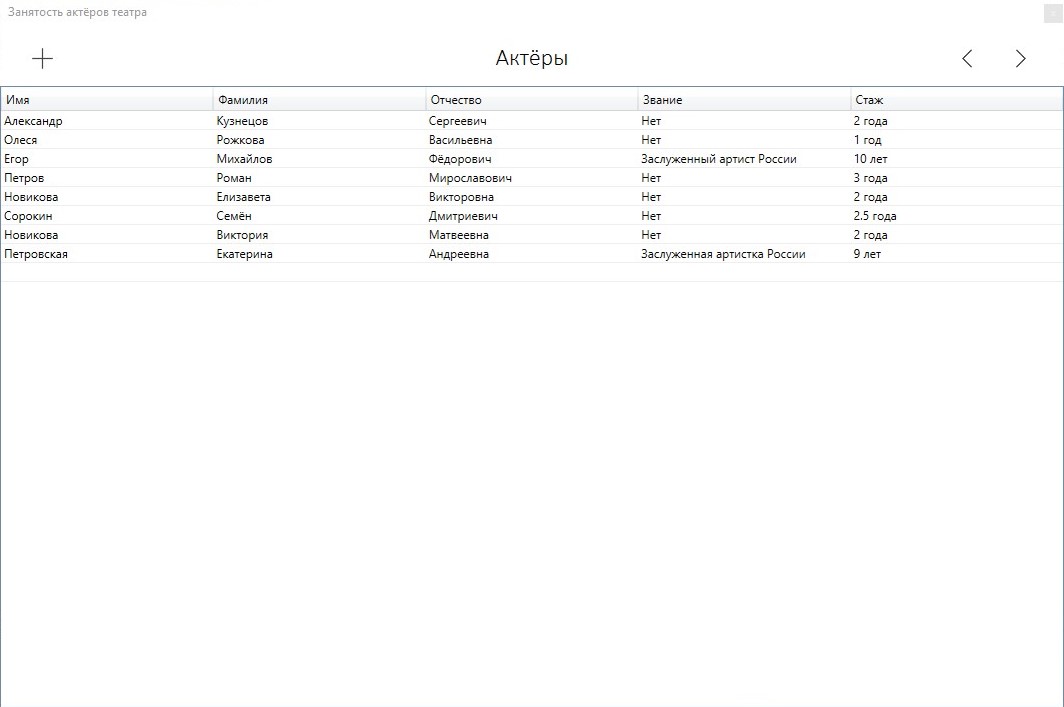


Рисунок 6 - Форма Actors (Актёры)

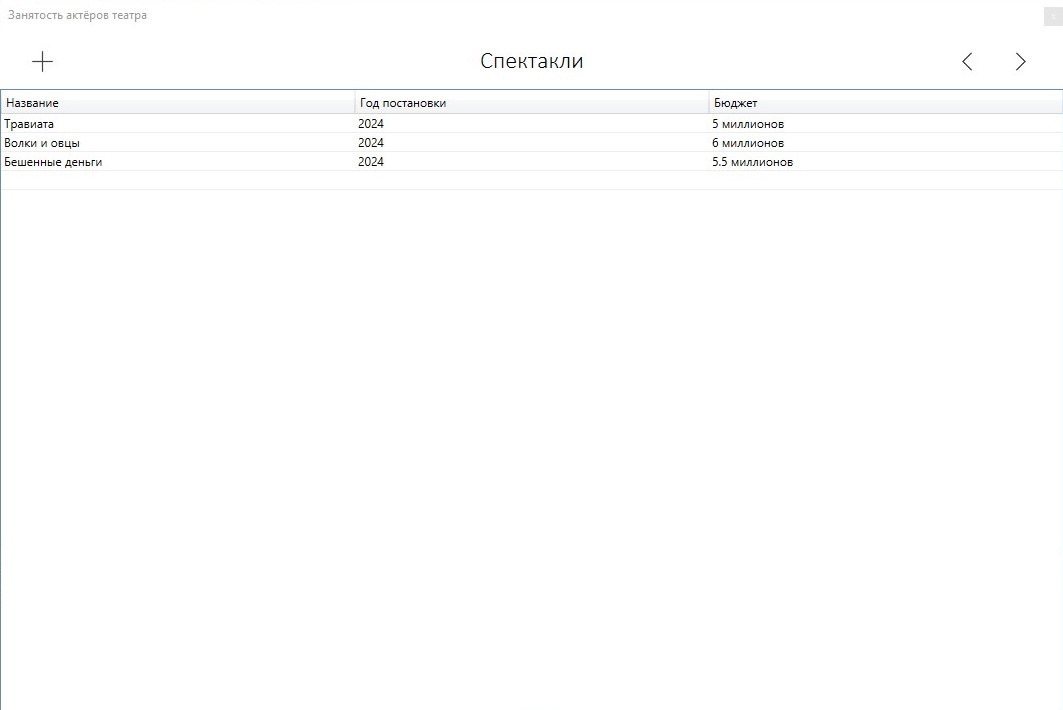


Рисунок 7 - Форма Performances (Спектакли)

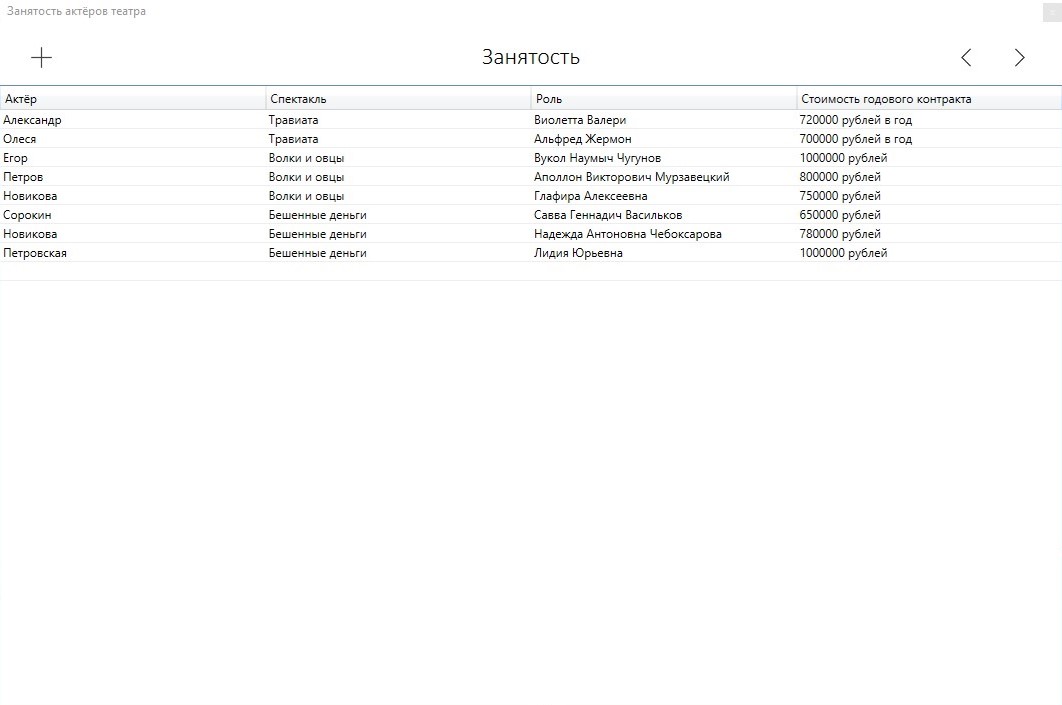


Рисунок 8 - Форма Enployment (Занятость)

Для редактирования записей о поставках, деталях и поставщиках используются отдельные формы приложения и панели для редактирования таблиц в MainWindow. Для редактирования записей в таблице «Актёры» и «Спектакли» используется редактирование напрямую в таблице DataGrid. Для добавления записи в таблицу «Актёры» и «Спектакли» используется форма AddWindow под названием «Добавить», для добавления/изменения таблицы «Занятость», форма AddOrEditEnployentWindow с названием «Добавить/Изменить». Эти формы изображены на рисунках 9-11.

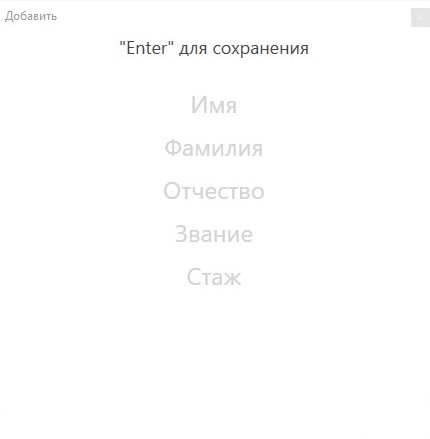


Рисунок 9 - Форма AddWindow (Для актёров)

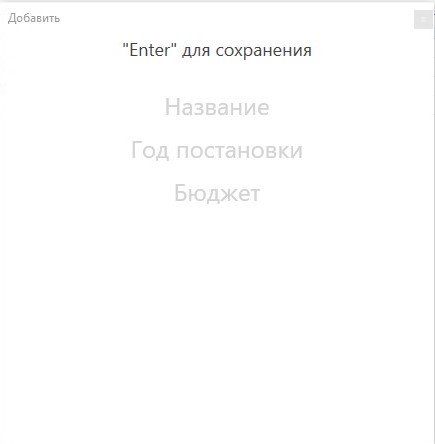


Рисунок 10 - Форма AddWindow (Для спектаклей)

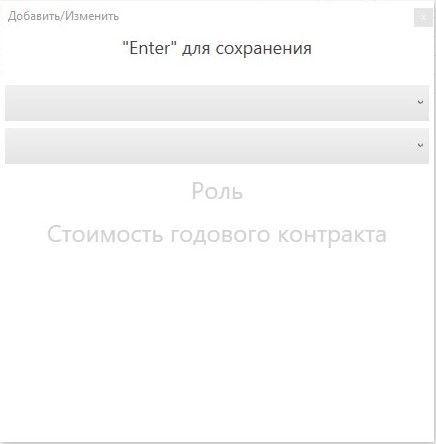


Рисунок 11 - Форма AddOrEditEnploymentWindow

Удаление информации на всех трёх таблицах осуществляется с помощью кнопки клавиатуры «Delete», после выбора строки в таблице.

**5 ТЕСТИРОВАНИЕ**

Для тестирования программы производились различные манипуляции с данными.

Проводились следующие тесты:

Тест 1 – Пустое значение в обязательном поле ввода данных, изображен на рисунке 15:

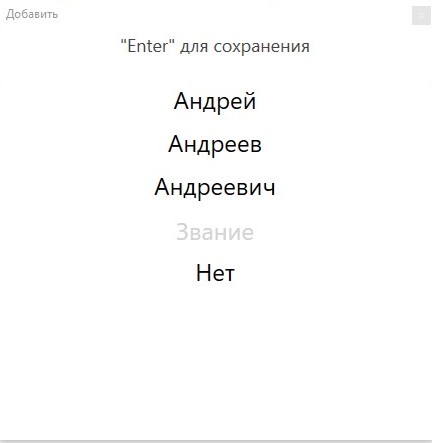


Рисунок 15 - Ввод пустого значения в поле «Звание»

После нажатия «Enter» запись намеренно не добавляется, и таблица не меняется, поля в окне добавления остаются как были.

Тест 2 Пустое значение в обязательном поле ввода данных, изображен на рисунке 17:



Рисунок 16 – Ввод пустого значения в поле «Бюджет»

После нажатия «Enter» запись намеренно не добавляется, и таблица не меняется, поля в окне добавления остаются как были.

А так же были проведены другие операции, призванные выявить ошибки в программе.

Тестирование выявило мелкие недочеты, которые впоследствии были устранены.

**СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ**

1. Основы алгоритмизации и программирования : учеб. пособие / В.Д. Колдаев; под ред. проф. Л.Г. Гагариной. — М. : ИД «ФОРУМ» : ИНФРАМ, 2020. — 414 с. — (Среднее профессиональное образование). - Режим доступа: <http://znanium.com/catalog/product/980416>
2. Программирование графики на С#. Теория и примеры : учеб. пособие / В.И. Корнеев, Л.Г. Гагарина, М.В. Корнеева. — М. : ИД «ФОРУМ» : ИНФРА-М, 2020. — 517 с. + Доп. материалы [Электронный ресурс; Режим доступа <http://www.znanium.com>].
3. Программирование на С# с погружением: практические задания и примеры кода - М.:НИЦ ИНФРА-М, 2020. - 80 с.: 60x90 1/16 - Режим доступа: <http://znanium.com/catalog/product/563294>
4. Программирование на языке высокого уровня. Программирование на языке С#: учеб. пособие / Т.И. Немцова, С.Ю. Голова, А.И. Терентьев ; под ред. Л.Г. Гагариной. — М. : ИД «ФОРУМ» : ИНФРА-М, 2020. — 512 с. — (Среднее профессиональное образование). - Режим доступа: <http://znanium.com/catalog/product/918098>
5. Программирование. Процедурное программирование: Учебное пособие / Кучунова Е.В., Олейников Б.В., Чередниченко О.М. - Краснояр.:СФУ, 2020. - 92 с.: ISBN 978-5-7638-3555-7 - Режим доступа: <http://znanium.com/catalog/product/978627>
6. Введение в программирование на языке Visual C# : учеб. пособие / С.Р. Гуриков. — М. : ФОРУМ : ИНФРА-М, 2020. — 447 с. — (Высшее образование: Бакалавриат). - Режим доступа: <http://znanium.com/catalog/product/967691>
7. Объектно-ориентированное программирование с примерами на С# : учеб. пособие / П.Б. Хорев. — М. : ФОРУМ : ИНФРА-М, 2020. — 200 с. — (Высшее образование: Бакалавриат). - Режим доступа: http://znanium.com/catalog/product/1018034

**ПРИЛОЖЕНИЕ А**

**Листинг программного модуля**

**Форма приложения MainWindow**

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 Theater

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

public MainWindow()

{

InitializeComponent();

ContentFrame.Navigate(new Pages.ActorsPage());

}

}

}

**Форма приложения ActorsPage**

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;

using Theater.Entities;

using Theater.Windows;

namespace Theater.Pages

{

/// <summary>

/// Логика взаимодействия для ActorsPage.xaml

/// </summary>

public partial class ActorsPage : Page

{

public ActorsPage()

{

InitializeComponent();

Update();

}

private void Update()

{

DG.ItemsSource = Core.DB.Actors.ToList();

}

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

{

new AddWindow("Actor", DG).Show();

}

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

{

NavigationService.RemoveBackEntry();

NavigationService.Navigate(new EnploymentPage());

}

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

{

NavigationService.RemoveBackEntry();

NavigationService.Navigate(new PerformancesPage());

}

private void DG\_PreviewKeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Delete)

{

MessageBoxResult result = MessageBox.Show("Вы уверены что хотите удалить элемент?", "Удаление", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

try

{

DG.SelectedItem.GetType();

Core.DB.Actors.Remove((Actor) DG.SelectedItem);

Core.DB.SaveChanges();

}

catch { }

}

Update();

}

}

private void DG\_CellEditEnding(object sender, DataGridCellEditEndingEventArgs e)

{

try

{

Actor Selected = (Actor) DG.SelectedItem;

var cell = (TextBox) e.EditingElement;

string Value = cell.Text;

if (Selected.Name != null && Selected.Surname != null && Selected.Patronymic != null && Selected.Rank != null && Selected.Experience != null && Value != "")

{

if ($"{e.Column.Header}" == "Имя")

Selected.Name = Value;

else if ($"{e.Column.Header}" == "Фамилия")

Selected.Surname = Value;

else if ($"{e.Column.Header}" == "Отчество")

Selected.Patronymic = Value;

else if ($"{e.Column.Header}" == "Звание")

Selected.Rank = Value;

else if ($"{e.Column.Header}" == "Стаж")

Selected.Experience = Value;

Core.DB.SaveChanges();

}

} catch { }

Update();

}

}

}

**Форма приложения PerformancesPage**

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;

using Theater.Entities;

using Theater.Windows;

namespace Theater.Pages

{

/// <summary>

/// Логика взаимодействия для PerformancesPage.xaml

/// </summary>

public partial class PerformancesPage : Page

{

public PerformancesPage()

{

InitializeComponent();

Update();

}

private void Update()

{

DG.ItemsSource = Core.DB.Performances.ToList();

}

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

{

new AddWindow("Performance", DG).Show();

}

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

{

NavigationService.RemoveBackEntry();

NavigationService.Navigate(new ActorsPage());

}

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

{

NavigationService.RemoveBackEntry();

NavigationService.Navigate(new EnploymentPage());

}

private void DG\_PreviewKeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Delete)

{

MessageBoxResult result = MessageBox.Show("Вы уверены что хотите удалить элемент?", "Удаление", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

try

{

DG.SelectedItem.GetType();

Core.DB.Performances.Remove((Performance)DG.SelectedItem);

Core.DB.SaveChanges();

}

catch { }

}

Update();

}

}

private void DG\_CellEditEnding(object sender, DataGridCellEditEndingEventArgs e)

{

try

{

Performance Selected = (Performance)DG.SelectedItem;

var cell = (TextBox)e.EditingElement;

string Value = cell.Text;

if (Selected.Title != null && Selected.Year != null && Selected.Budget != null && Value != "")

{

if ($"{e.Column.Header}" == "Название")

Selected.Title = Value;

else if ($"{e.Column.Header}" == "Год постановки")

Selected.Year = Value;

else if ($"{e.Column.Header}" == "Бюджет")

Selected.Budget = Value;

Core.DB.SaveChanges();

}

}

catch { }

Update();

}

}

}

**Форма приложения EnploymentPage**

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

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 Theater.Entities;

using Theater.Windows;

namespace Theater.Pages

{

/// <summary>

/// Логика взаимодействия для EnploymentPage.xaml

/// </summary>

public partial class EnploymentPage : Page

{

public EnploymentPage()

{

InitializeComponent();

Update();

}

private void Update()

{

DG.ItemsSource = Core.DB.Enployments.ToList();

}

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

{

NavigationService.RemoveBackEntry();

NavigationService.Navigate(new ActorsPage());

}

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

{

NavigationService.RemoveBackEntry();

NavigationService.Navigate(new PerformancesPage());

}

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

{

new AddOrEditEnploymentWindow("add", null, DG).Show();

}

private void DG\_MouseDoubleClick(object sender, MouseButtonEventArgs e)

{

try

{

var item = (Enployment)DG.SelectedItem;

if(Core.DB.Enployments.Find(item.EnploymentID) != null)

new AddOrEditEnploymentWindow("edit", item, DG).Show();

}

catch { }

}

private void DG\_PreviewKeyDown(object sender, KeyEventArgs e)

{

if (e.Key == Key.Delete)

{

MessageBoxResult result = MessageBox.Show("Вы уверены что хотите удалить элемент?", "Удаление", MessageBoxButton.YesNo, MessageBoxImage.Question);

if (result == MessageBoxResult.Yes)

{

try

{

DG.SelectedItem.GetType();

Core.DB.Enployments.Remove((Enployment)DG.SelectedItem);

Core.DB.SaveChanges();

}

catch { }

}

Update();

}

}

private void DG\_CellEditEnding(object sender, DataGridCellEditEndingEventArgs e)

{

Update();

}

}

}

**Форма приложения AddOrEditEnploymentWindow**

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;

using Theater.Entities;

namespace Theater.Windows

{

/// <summary>

/// Логика взаимодействия для AddOrEditEnploymentWindow.xaml

/// </summary>

public partial class AddOrEditEnploymentWindow : Window

{

private static void SetHint(TextBox textBox, string content)

{

var cueBannerBrush = new VisualBrush

{

AlignmentX = AlignmentX.Center,

AlignmentY = AlignmentY.Center,

Stretch = Stretch.None,

Visual = new Label

{

Content = content,

Foreground = Brushes.LightGray,

FontSize = 24,

}

};

textBox.Style = new Style

{

TargetType = typeof(TextBox),

Resources =

{

{ "CueBannerBrush", cueBannerBrush }

},

Triggers =

{

new Trigger

{

Property = TextBox.TextProperty,

Value = string.Empty,

Setters =

{

new Setter { Property = TextBox.BackgroundProperty, Value = cueBannerBrush }

}

},

new Trigger

{

Property = TextBox.TextProperty,

Value = null,

Setters =

{

new Setter { Property = TextBox.BackgroundProperty, Value = cueBannerBrush }

}

},

new Trigger

{

Property = TextBox.IsKeyboardFocusedProperty,

Value = true,

Setters =

{

new Setter { Property = TextBox.BackgroundProperty, Value = Brushes.White }

}

}

}

};

}

private static TextBox CreateTextBox(string content)

{

TextBox tb = new TextBox()

{

Margin = new Thickness(0, 7, 0, 0),

Height = 36,

BorderThickness = new Thickness(0),

FontSize = 18,

TextAlignment = TextAlignment.Center,

};

SetHint(tb, content);

return tb;

}

public AddOrEditEnploymentWindow(string mode, Enployment enployment, DataGrid DG)

{

InitializeComponent();

List<string> values = new List<string> { "Роль", "Стоимость годового контракта" };

List<TextBox> text\_boxes = new List<TextBox>();

values.ForEach((it) => { text\_boxes.Add(CreateTextBox(it)); });

text\_boxes.ForEach((it) => { MainStackPanel.Children.Add(it); });

List<Actor> ActorsList = Core.DB.Actors.ToList();

List<Performance> PerformancesList = Core.DB.Performances.ToList();

ActorCB.ItemsSource = ActorsList;

PerformanceCB.ItemsSource = PerformancesList;

DG.SelectedItem = null;

if (mode == "edit")

{

ActorCB.SelectedItem = enployment.Actor;

PerformanceCB.SelectedItem = enployment.Performance;

text\_boxes[0].Text = enployment.Role;

text\_boxes[1].Text = enployment.Cost;

bool IsNull = false;

MainStackPanel.PreviewKeyDown += (sender, e) =>

{

if (e.Key == Key.Enter)

{

text\_boxes.ForEach((it) => { if (it.Text == null || it.Text == "") IsNull = true; });

if (ActorCB.SelectedItem == null || PerformanceCB.SelectedItem == null) IsNull = true;

if (!IsNull)

{

enployment.ActorID = (ActorCB.SelectedItem as Actor).ActorID;

enployment.PerformanceID = (PerformanceCB.SelectedItem as Performance).PerformanceID;

enployment.Role = text\_boxes[0].Text;

enployment.Cost = text\_boxes[1].Text;

Core.DB.SaveChanges();

DG.ItemsSource = Core.DB.Enployments.ToList();

}

IsNull = false;

Close();

}

};

}

if(mode == "add")

{

bool IsNull = false;

MainStackPanel.KeyDown += (sender, e) =>

{

if (e.Key == Key.Enter)

{

text\_boxes.ForEach((it) => { if (it.Text == null || it.Text == "") IsNull = true; });

if (ActorCB.SelectedItem == null || PerformanceCB.SelectedItem == null) IsNull = true;

if (!IsNull)

{

Core.DB.Enployments.Add(new Enployment()

{

ActorID = (ActorCB.SelectedItem as Actor).ActorID,

PerformanceID = (PerformanceCB.SelectedItem as Performance).PerformanceID,

Role = text\_boxes[0].Text,

Cost = text\_boxes[1].Text

});

Core.DB.SaveChanges();

DG.ItemsSource = Core.DB.Enployments.ToList();

text\_boxes.ForEach((it) => { it.Text = ""; });

ActorCB.SelectedItem = null;

PerformanceCB.SelectedItem = null;

}

IsNull = false;

}

};

}

}

}

}

**Форма приложения AddWindow**

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;

using Theater.Entities;

namespace Theater.Windows

{

/// <summary>

/// Логика взаимодействия для AddWindow.xaml

/// </summary>

public partial class AddWindow : Window

{

// сложни метод, делает подсказку с текстом который задаётся аргументом content

private static void SetHint(TextBox textBox, string content)

{

var cueBannerBrush = new VisualBrush

{

AlignmentX = AlignmentX.Center,

AlignmentY = AlignmentY.Center,

Stretch = Stretch.None,

Visual = new Label

{

Content = content,

Foreground = Brushes.LightGray,

FontSize = 24,

}

};

textBox.Style = new Style

{

TargetType = typeof(TextBox),

Resources =

{

{ "CueBannerBrush", cueBannerBrush }

},

Triggers =

{

new Trigger

{

Property = TextBox.TextProperty,

Value = string.Empty,

Setters =

{

new Setter { Property = TextBox.BackgroundProperty, Value = cueBannerBrush }

}

},

new Trigger

{

Property = TextBox.TextProperty,

Value = null,

Setters =

{

new Setter { Property = TextBox.BackgroundProperty, Value = cueBannerBrush }

}

},

new Trigger

{

Property = TextBox.IsKeyboardFocusedProperty,

Value = true,

Setters =

{

new Setter { Property = TextBox.BackgroundProperty, Value = Brushes.White }

}

}

}

};

}

private static TextBox CreateTextBox(string content)

{

TextBox tb = new TextBox()

{

Margin = new Thickness(0, 7, 0, 0),

Height = 36,

BorderThickness = new Thickness(0),

FontSize = 24,

TextAlignment = TextAlignment.Center,

};

SetHint(tb, content);

return tb;

}

public AddWindow(string type, DataGrid DG)

{

InitializeComponent();

// тут помещены названия столбцов, каких именно определяется в switch дальше

List<string> values = new List<string> { };

List<TextBox> text\_boxes = new List<TextBox> { };

switch (type)

{

case "Actor":

values = new List<string> { "Имя", "Фамилия", "Отчество", "Звание", "Стаж" };

break;

case "Performance":

values = new List<string> { "Название", "Год постановки", "Бюджет" };

break;

}

values.ForEach((it) => { text\_boxes.Add(CreateTextBox(it)); });

text\_boxes.ForEach((it) => { MainStackPanel.Children.Add(it); });

bool IsNull = false;

MainStackPanel.KeyDown += (sender, e) =>

{

if(e.Key == Key.Enter)

{

text\_boxes.ForEach((it) => { if (it.Text == null || it.Text == "") IsNull = true; });

if (!IsNull)

{

switch (type)

{

case "Actor":

Core.DB.Actors.Add(new Actor()

{

Name = text\_boxes[0].Text,

Surname = text\_boxes[1].Text,

Patronymic = text\_boxes[2].Text,

Rank = text\_boxes[3].Text,

Experience = text\_boxes[4].Text,

}); break;

case "Performance":

Core.DB.Performances.Add(new Performance()

{

Title = text\_boxes[0].Text,

Year = text\_boxes[1].Text,

Budget = text\_boxes[2].Text,

}); break;

}

Core.DB.SaveChanges();

switch (type)

{

case "Actor": DG.ItemsSource = Core.DB.Actors.ToList(); break;

case "Performance": DG.ItemsSource = Core.DB.Performances.ToList(); break;

}

text\_boxes.ForEach((it) => { it.Text = ""; });

}

IsNull = false;

}

};

}

}

}

**ПРИЛОЖЕНИЕ Б**

**Диск с исполняемым кодом программного модуля**