Содержание

ВВЕДЕНИЕ

Приложения для хранения и обмена информации в данный момент имеют невероятный спрос. Благодаря своему удобству, защищенности и постоянному доступу к данным. Это необходимы инструмент в работе любой команды. Возможность делиться информацией одна из ключевых возможностей таких приложений.

MS SQL Server — это система управления реляционными базами данных, работающая по клиент-серверной модели. Она была создана компанией Microsoft и первая версия вышла в 1987 году. MS SQL Server подходит для самых различных проектов: от небольших приложений до больших высоконагруженных проектов.

Реляционные базы данных хранят все данные в виде таблиц, и эти таблицы связаны между собой. Для взаимодействия с базой данных применяется язык SQL (Structured Query Language).

Вот некоторые основные особенности MS SQL Server:

- Производительность: SQL Server работает очень быстро.
- Надежность и безопасность: SQL Server предоставляет шифрование данных.
- Простота: С данной СУБД относительно легко работать и вести администрирование.

C# — это один из самых мощных, быстро развивающихся и востребованных языков программирования в ИТ-отрасли. Он был создан компанией Microsoft и используется для разработки самых разнообразных приложений: от небольших десктопных программ до крупных веб-порталов и веб-сервисов.

C# является объектно-ориентированным языком с Си-подобным синтаксисом, близким к C++ и Java. Он поддерживает полиморфизм, наследование, перегрузку операторов, статическую типизацию. Это позволяет строить крупные, гибкие, масштабируемые и расширяемые приложения.

C# активно развивается, и с каждой новой версией появляется все больше интересных функциональностей[1]. Текущей версией языка является версия C# 12, которая вышла 14 ноября 2023 года вместе с релизом .NET 8.

C# тесно связан с платформой .NET, которая представляет собой мощную платформу для создания приложений[1]. Она поддерживает несколько языков, является кроссплатформенной и имеет мощную библиотеку классов.

Курсовой проект посвящен изучению и применению технологии ADO.NET, которая является ключевым компонентом платформы .NET и служит для взаимодействия с данными, хранящимися в различных источниках информации, в основном в базах данных.

ADO.NET предоставляет гибкий и эффективный способ для взаимодействия с данными любого типа: от настольных баз данных Microsoft Access до корпоративных баз данных SQL Server и Oracle. Благодаря своей гибкости и мощности, ADO.NET стала неотъемлемой частью разработки приложений на платформе .NET.

В рамках данного проекта будет проведен анализ основных возможностей и преимуществ ADO.NET, а также разработано приложение, демонстрирующее практическое применение этой технологии.

Основной целью проекта является показать, как ADO.NET может быть использована для эффективного управления данными в современных приложениях.

Microsoft Excel - это мощный инструмент для работы с электронными таблицами, который используется во всем мире для обработки данных, анализа информации и визуализации результатов.

Создание аналога такого сложного и функционального приложения, как MS Excel, является большим вызовом. Это требует глубокого понимания принципов работы с данными, а также навыков программирования и проектирования пользовательского интерфейса.

В рамках данного проекта будет разработано приложение, которое будет включать в себя основные функции MS Excel, такие как работа с ячейками, формулами, графиками и диаграммами. Основная цель проекта - создать эффективный и удобный инструмент для работы с электронными таблицами, который может быть использован в различных областях, от образования до бизнеса.

Windows Presentation Foundation (WPF) — аналог WinForms, система для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем, графическая (презентационная) подсистема в составе .NET Framework (начиная с версии 3.0), использующая язык XAML.

WPF предустановлена в Windows Vista (.NET Framework 3.0), Windows 7 (.NET Framework 3.5 SP1), Windows 8 (.NET Framework 4.0 и 4.5), Windows 8.1 (.NET Framework 4.5.1) и Windows 10 (.NET Framework 4.7). С помощью WPF можно создавать широкий спектр как автономных, так и запускаемых в браузере приложений.

В основе WPF лежит векторная система визуализации, не зависящая от разрешения устройства вывода и созданная с учётом возможностей современного графического оборудования. WPF предоставляет средства для создания визуального интерфейса, включая язык XAML (eXtensible Application Markup Language), элементы управления, привязку данных, макеты, двухмерную и трёхмерную графику, анимацию, стили, шаблоны, документы, текст, мультимедиа и оформление.

Графической технологией, лежащей в основе WPF, является DirectX, в отличие от Windows Forms, где используется GDI/GDI+. Производительность WPF выше, чем у GDI+ за счёт использования аппаратного ускорения графики через DirectX.

Также существует урезанная версия CLR, называющаяся WPF/E, она же известна как Silverlight.

XAML представляет собой язык декларативного описания интерфейса, основанный на XML. Также реализована модель разделения кода и дизайна, позволяющая кооперироваться программисту и дизайнеру. Кроме того, есть встроенная поддержка стилей элементов, а сами элементы легко разделить на элементы управления второго уровня, которые, в свою очередь, разделяются до уровня векторных фигур и свойств/действий. Это позволяет легко задать стиль для любого элемента, например, Button (кнопка).

WPF предоставляет широкий спектр возможностей по созданию интерактивных настольных приложений:

1. Привязка данных. Это гибкий механизм, который позволяет через расширения разметки XAML связывать различные данные (от значений свойств элементов управления до общедоступных свойств, реализующих поля базы данных через Entity Framework). Привязка данных представлена классом Binding, который в свою очередь унаследован от MarkupExtension, что позволяет использовать привязки не только в коде, но и в разметке;
2. Стили. Позволяют создавать стилевое оформление элементов и, как правило, используются только в разметке;
3. Шаблоны элементов управления. Позволяют менять графическое оформление элементов и представлены классом ControlTemplate. В отличие от стилей, можно менять не только графическое представление элемента, но и его структуру. При этом шаблон элемента управления задается через свойство Template.
4. Шаблоны данных. В отличие от шаблонов элементов управления, задаются для определенного контекста данных (который в блочных элементах управления задается через свойство DataContext, а в списковых через ItemsSource). Сам шаблон данных представлен классом DataTemplate. Для обозначения типа данных, к которому необходимо применить шаблон, используется свойство DataType.
5. Ресурсы. Система ресурсов позволяет объединять шаблоны, стили, кисти, анимацию и многие другие интерактивные элементы, что существенно упрощает работу с ними. Ресурсы задаются в свойстве Resources класса FrameworkElement, от которого унаследованы все элементы управления, панели компоновки и даже класс Application. Это позволяет создавать многоуровневую систему ресурсов:
    a. ресурсы внутри объекта — действительны только для этого объекта
    b. ресурсы внутри панели компоновки (например Grid) — позволяет задать границу контекста ресурсов на уровне этой панели
    c. ресурсы внутри окна Window — если в приложении используется несколько окон, то ресурсы одного окна не будут доступны ресурсам другого окна

6. Графика. WPF представляет обширный, масштабируемый и гибкий набор графических возможностей:
    a. Графика, не зависящая от разрешения и устройства. Основной единицей измерения в графической системе WPF является аппаратно-независимый пиксель, который составляет 1/96 часть дюйма независимо от фактического разрешения экрана.
    b. Дополнительная поддержка графики и анимации. WPF упрощает программирование графики за счет автоматического управления анимацией. Разработчик не должен заниматься обработкой сцен анимации, циклами отрисовки и билинейной интерполяцией
    c. Аппаратное ускорение. Графическая система WPF использует преимущества графического оборудования, чтобы уменьшить использование ЦП.

Двухмерная графика. WPF предоставляет библиотеку общих двухмерных фигур, нарисованных с помощью векторов, таких, как прямоугольники и эллипсы, а также графические пути. И в своей функциональности фигуры реализуют многие возможности, которые доступны обычным элементам управления.Двухмерная графика в WPF включает визуальные эффекты, такие как градиенты, точечные рисунки, чертежи, рисунки с видео, поворот, масштабирование и наклон.

Трехмерная графика. WPF также включает возможности трехмерной отрисовки, интегрированные с двухмерной графикой, что позволяет создавать более яркий и интересный пользовательский интерфейс.

1. ПОСТАНОВКА ЗАДАЧИ

Цель данного курсового проекта – разработка графического приложения, аналога MS Excel.

Основные задачи проекта включают в себя:

1. Создание приложения
2. Тестирование
3. Интеграция

Исходные данные к проекту:

1. Библиотек Microsoft.Data.SqlClient. Пространство имен Microsoft.Data.SqlClient по сути является новой версией пространства имен System.Data.SqlClient. Microsoft.Data.SqlClient обычно поддерживает те же API и обратную совместимость, что и System.Data.SqlClient. Для большинства приложений переход с System.Data.SqlClient на Microsoft.Data.SqlClient не составляет проблем. Добавьте зависимость NuGet в Microsoft.Data.SqlClient, после чего обновите ссылки и инструкции `using` в Microsoft.Data.SqlClient.
2. Система создания оконных приложений WPF. Windows Presentation Foundation (WPF) – это система для построения клиентских приложений Windows, которая используется для создания графических пользовательских интерфейсов (GUI). WPF использует язык разметки XAML для создания пользовательского интерфейса и позволяет разработчикам создавать богатые и интерактивные пользовательские интерфейсы. WPF включает в себя множество инструментов и библиотек, которые облегчают разработку приложений.
3. Спецификация компилятора. C# – это язык программирования, который разработала компания Microsoft. Он используется для создания приложений для Windows, веб-приложений, игр и многого другого. C# является объектно-ориентированным языком программирования, который использует синтаксис, похожий на язык программирования Java. Он также включает в себя множество библиотек, которые облегчают разработку приложений.

2. АНАЛИЗ ЗАДАЧИ

Основная идея – создание аналога MS Excel, ключевое отличие будет в возможности интеграции и управление БД, прямо из программы.

Пользователь должен иметь возможность выполнения следующих действий:

1. Добавлять новые таблицы в базу данных
2. Создавать базу данных в случае ее отсутствия
3. Редактировать таблицы
4. Удаление таблиц

База данных – MS SQL.

Библиотека для взаимодействия с базой данных – Microsoft.Data.SqlClient.

Для создания клиентского графического приложения был выбран фреймворк Windows Presentation Foundation (WPF).

Основной элемент управления – DataGrid. Связь DataGrid с БД, будет осуществляться с помощью класса посредника DataTable.

3. РАЗРАБОТКА АЛГОРИТМОВ

Создание базы данных:

Создаю подключение и передаю в него SQL команду для создания базы данных:

```
public static int CreateDataBase()
{
    using (SqlConnection connection = new
SqlConnection(_conForCreate))
    {
        connection.Open();
        string sqlExpression = "CREATE DATABASE CourseWork";
        SqlCommand command = new SqlCommand(sqlExpression,
connection);
        return command.ExecuteNonQuery();
    }
}
```

Создание таблиц:

Создаю подключение и передаю в него SQL команду с ключевым словом CREATE TABLE.

Создаю 2 таблицы:

1. Таблица для хранения дней занятий
2. Таблица для контроля успеваемости студента

```
public static void CreateNewTable(string name)
{
    using (SqlConnection connection = new
SqlConnection(_connectionString))
    {
        connection.Open();
        string sqlExpression = $"CREATE TABLE {name}_d (Id INT
PRIMARY KEY IDENTITY, Date NVARCHAR(100) NOT NULL );" +
                               $"CREATE TABLE {name}_a (Id INT
PRIMARY KEY identity);";
        SqlCommand command = new SqlCommand(sqlExpression,
connection);
        _tables.Add(name);
        command.ExecuteNonQuery();
    }
}
```

Создание метода для удаления таблиц, получает на вход имя группы, после чего удаляет 2 таблицы из базы данных:

```
public static void DeleteTable(string name)
{
    using (SqlConnection connection = new
SqlConnection(_connectionString))
    {
        connection.Open();
        string sqlExpression = $"DROP TABLE {name}_d;" +
                               $"DROP TABLE {name}_a;";
        SqlCommand command = new SqlCommand(sqlExpression,
connection);
        _tables.Remove(name);
        command.ExecuteNonQuery();
    }
}
```

Создание метода для вывода данных хранящихся в таблицы. Получает на вход имя таблицы, возвращает элемент DataTable, для последующего вывода в DataGrid:

```
public static DataTable ShowTable(string name)
{

    SqlConnection connection = new
SqlConnection(_connectionString);
    SqlCommand command = new SqlCommand($"SELECT * FROM
{name};",connection);
    SqlDataAdapter sqlDataAdapter = new
SqlDataAdapter(command);
    DataTable dataTable = new DataTable();
    sqlDataAdapter.Fill(dataTable);
    return dataTable;
}
```

Создание метода для сохранения данных, после их изменения в DataGrid, в базу данных, на вход получает имя базы данных и DataTable:

```
public static void SaveTable(string name,DataTable dataTable)
{
    SqlConnection connection = new
SqlConnection(_connectionString);
    SqlDataAdapter sqlDataAdapter = new SqlDataAdapter($"SELECT
* FROM {name}",connection);
    new SqlCommandBuilder(sqlDataAdapter);
    sqlDataAdapter.Update(dataTable);
}
```

Создание метода для генерации основной таблицы после добавления или удаления даты занятия, на основе данных из таблицы с датами генерируется новая основная таблицв:

```
public static void GenTableA(string name)
{
    DropTableA(name);
    List<string> list = new List<string>();
    using (SqlConnection connection = new
SqlConnection(_connectionString))
    {
        connection.Open();
        string sqlExpression = $"SELECT * FROM {name}_d";
        SqlCommand command = new SqlCommand(sqlExpression,
connection);
        SqlDataReader reader = command.ExecuteReader();
        if (reader.HasRows)
        {
            while (reader.Read()) // построчно считываем данные
            {
                string date = reader.GetString(1);
                list.Add(date);
            }
        }
    }

    string str = $"CREATE TABLE {name}_a (Id INT PRIMARY KEY
IDENTITY, Name NVARCHAR(100)";
    foreach (var l in list)
    {
        str += $", d{l} NVARCHAR(100)";
    }
```

```
str += ");";
    using (SqlConnection connection = new
SqlConnection(_connectionString))
    {
        connection.Open();
        SqlCommand command = new SqlCommand(str, connection);
        _tables.Add(name);
        command.ExecuteNonQuery();
    }

}
```

Создание метода для записи списка таблиц в бд:

```
public static void AddTables(string name)
{
    using (SqlConnection connection = new
SqlConnection(_connectionString))
    {
        connection.Open();
        SqlCommand command = new SqlCommand($"INSERT datatables
VALUES ('{name}');",connection);
        command.ExecuteNonQuery();
    }
}
```

Создание метода для чтения списка таблиц из бд:

```
public static void GetTables()
{
    _tables = new List<string>();
    using (SqlConnection connection = new
SqlConnection(_connectionString))
    {
        connection.Open();
        SqlCommand command = new SqlCommand("SELECT
datatables.Name FROM datatables",connection);
        SqlDataReader reader = command.ExecuteReader();
        if (reader.HasRows)
        {
            while (reader.Read())
            {
                _tables.Add(reader.GetString(0));
            }
        }
    }
}
```

Создание метода для удаления таблицы из бд:

```
public static void DeleteTables(string name)
{
    using (SqlConnection connection = new
SqlConnection(_connectionString))
    {
        connection.Open();
        SqlCommand command = new SqlCommand($"DELETE datatables
WHERE Name = '{name}';",connection);
        command.ExecuteNonQuery();
    }
}
```

Создание обработчика событие нажатия на кнопку добавления новой таблицы:

```
private void AddNewButton_OnClick(object sender,
RoutedEventArgs e)
{
    AddNewDb addNewDb = new AddNewDb();
    addNewDb.ShowDialog();
    DbComboBox.ItemsSource = null;
    DbComboBox.ItemsSource = DataBase._tables;
}
```

Создание обработчика событие нажатия на кнопку удаление таблицы:

```
private void DeleteButton_OnClick(object sender,
RoutedEventArgs e)
{
    if (DbComboBox.Text != "")
    {
        switch (MessageBox.Show($"Would you like to delete
group {DbComboBox.Text}? ", "",
                    MessageBoxButton.YesNo))
        {
            case MessageBoxResult.Yes:
                DataBase.DeleteTable(DbComboBox.Text);
                MessageBox.Show("Success");
                break;
            case MessageBoxResult.No:
                return;
        }
        DbComboBox.ItemsSource = null;
        DbComboBox.ItemsSource = DataBase._tables;
        DbComboBox.SelectedIndex = 0;
    }
}
```

Создание обработчика событие нажатия на кнопку изменение таблицы:

```
private void EditTablesButton_OnClick(object sender,
RoutedEventArgs e)
{
    if (DbComboBox.Text!="")
    {
        EditWindow editWindow = new
EditWindow(DbComboBox.Text);
        editWindow.ShowDialog();
        Update(prevStr);
    }

}
```

Создание обработчика событие нажатия на кнопку сохранения изменений:

```
private void SaveButton_OnClick(object sender, RoutedEventArgs
e)
{
    if (DbComboBox.Text != "")
    {
        DataBase.SaveTable($"{DbComboBox.Text}_a",_dt);
        MessageBox.Show("Success");
    }

}
```

Создание обработчика событие изменение выбора в выпадающем списке:

```
private void DbComboBox_OnSelectionChanged(object sender,
SelectionChangedEventArgs e)
{
    if (isEdit)
    {
        if (MessageBox.Show("Would u like to save
changed?","",MessageBoxButton.YesNo) == MessageBoxResult.Yes)
        {
            DataBase.SaveTable($"{prevStr}_a",_dt);
            MessageBox.Show("Success");
        }

    }
    if (DbComboBox.ItemsSource != null)
    {
        Update(DbComboBox.SelectedItem.ToString());
        prevStr = DbComboBox.SelectedItem.ToString();
    }


}
```

Создание метода Update для вывода таблицы в DataGrid:

```
private void Update(string? s)
{
    var str = s;
    if (str != "")
    {
        _dt = DataBase.ShowTable(str += "_a");
        DataGridVeiw.ItemsSource = _dt.DefaultView;
    }

    isEdit = false;
}
```

Создание окна для изменения таблиц, таблица для изменения выбирается в зависимости от переданного имени в конструктор окна. Создание обработчика события нажатия на кнопку сохранения изменений:

```
using System.Data;
using System.Windows;
using Microsoft.Data.SqlClient;

namespace CourseProject;

public partial class EditWindow : Window
{
    public EditWindow(string name)
    {
        InitializeComponent();



        _name = name;
        _name2 = name;
        _dataTableDate = DataBase.ShowTable(_name2 += "_d");
        DataGridDate.ItemsSource = _dataTableDate.DefaultView;
    }
```

```
    private string _name2;
    private string _name;
    private DataTable _dataTableDate;
    private void Save_OnClick(object sender, RoutedEventArgs e)
    {
        DataBase.SaveTable(_name2, _dataTableDate);
        DataBase.GenTableA(_name);
        MessageBox.Show("Success");
        Close();
    }
}
```

Создание окна для добавления таблиц, имя для новой таблицы вводится в TextBox. Создание обработчика события нажатия на кнопку добавления новой таблицы. Создание обработчика события нажатия на кнопку добавления новой базы данных:

```
public partial class AddNewDb : Window
{
    public AddNewDb()
    {
        InitializeComponent();
    }



    private void ButtonAdd_OnClick(object sender,
RoutedEventArgs e)
    {
        DataBase.CreateNewTable(TextBoxNameOfDb.Text);
        MessageBox.Show("Success");
        Close();
    }



    private void ButtonCreateNewDb_OnClick(object sender,
RoutedEventArgs e)
    {
        DataBase.CreateDataBase();
        MessageBox.Show("Success");
    }
}
```

Создание интерфейса главного окна, основной элемент DataGrid, один ComboBox для выбора таблиц, 4 кнопки для управления

```
<Window x:Class="CourseProject.MainWindow"

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

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"
        xmlns:local="clr-namespace:CourseProject"
        mc:Ignorable="d"
        Title="MainWindow" Height="450" Width="800">
```

Создание сетки для разметки приложения:

```
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition></RowDefinition>
            <RowDefinition Height="Auto"></RowDefinition>
        </Grid.RowDefinitions>
        <Grid.ColumnDefinitions>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
            <ColumnDefinition></ColumnDefinition>
        </Grid.ColumnDefinitions>
```

Создание таблицы для вывода данных:

```
        <DataGrid Name="DataGridVeiw"
RowEditEnding="DataGridVeiw_OnRowEditEnding" Grid.Column="0"
Grid.Row="0" Grid.ColumnSpan="5" Margin="5"></DataGrid>
        <ComboBox Name="DbComboBox"
SelectionChanged="DbComboBox_OnSelectionChanged" Grid.Row="1"
Grid.Column="0" Margin="5,0,5,5"></ComboBox>
        <Button Name="AddNewButton" Grid.Column="1"
Grid.Row="1" Click="AddNewButton_OnClick" Content="Add new"
Margin="0,0,5,5"></Button>
        <Button Name="DeleteButton" Grid.Column="2"
Grid.Row="1" Click="DeleteButton_OnClick" Content="Delete"
Margin="0,0,5,5"></Button>
        <Button Name="EditTablesButton" Grid.Column="3"
Grid.Row="1" Click="EditTablesButton_OnClick" Content="Edit
Table" Margin="0,0,5,5"></Button>
        <Button Name="SaveButton" Grid.Column="4" Grid.Row="1"
```

```
Click="SaveButton_OnClick" Content="Save"
Margin="0,0,5,5"></Button>
    </Grid>
</Window>
```

Создание интерфейса окна для изменения таблиц, основной элемент DataGrid, кнопка для сохранения изменение:

```
<Window x:Class="CourseProject.EditWindow"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentatio
n"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:local="clr-namespace:CourseProject"
        mc:Ignorable="d"
        Title="EditWindow" Height="250" Width="400">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition></RowDefinition>
            <RowDefinition Height="Auto"></RowDefinition>
        </Grid.RowDefinitions>

        <DataGrid Name="DataGridDate" Grid.Row="0"
Margin="5"></DataGrid>
        <Button Name="Save" Click="Save_OnClick" Grid.Row="1"
Grid.Column="0"  Content="Save" Margin="5,0,5,5"></Button>

    </Grid>
</Window>
```

Создание интерфейса окна для добавления таблиц, TextBox для ввода имения новой таблиц, 2 кнопки:

```
<Window x:Class="CourseProject.AddNewDb"

xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentatio
n"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:mc="http://schemas.openxmlformats.org/markup-
compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
        xmlns:local="clr-namespace:CourseProject"
```

```
        mc:Ignorable="d"
        Title="AddNewDb" SizeToContent="Height" Width="200">
    <Grid>
        <Grid.RowDefinitions>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
            <RowDefinition></RowDefinition>
        </Grid.RowDefinitions>
        <Label Name="Label" Content="Name of group:"></Label>
        <TextBox Grid.Row="1" Name="TextBoxNameOfDb"
Margin="5,0,5,5"></TextBox>
        <StackPanel Grid.Row="2" Orientation="Horizontal">
            <Button Name="ButtonAdd" Content="Add"
Click="ButtonAdd_OnClick" Margin="5,0,5,5"></Button>
            <Button Name="ButtonCreateNewDb" Content="Create
New DB" Click="ButtonCreateNewDb_OnClick"
Margin="0,0,5,5"></Button>
        </StackPanel>

    </Grid>
</Window>
```

4. ТЕСТИРОВАНИЕ И ОТЛАДКА

При запуске приложения мы видим начальный экран приложения, с одним выпадающем списком и 4 кнопками:

1. Add new – создание новой таблицы.
2. Delete – удаление выбранной таблицы.
3. Edit – изменение выбранной таблицы.
4. Save – сохранение изменений.

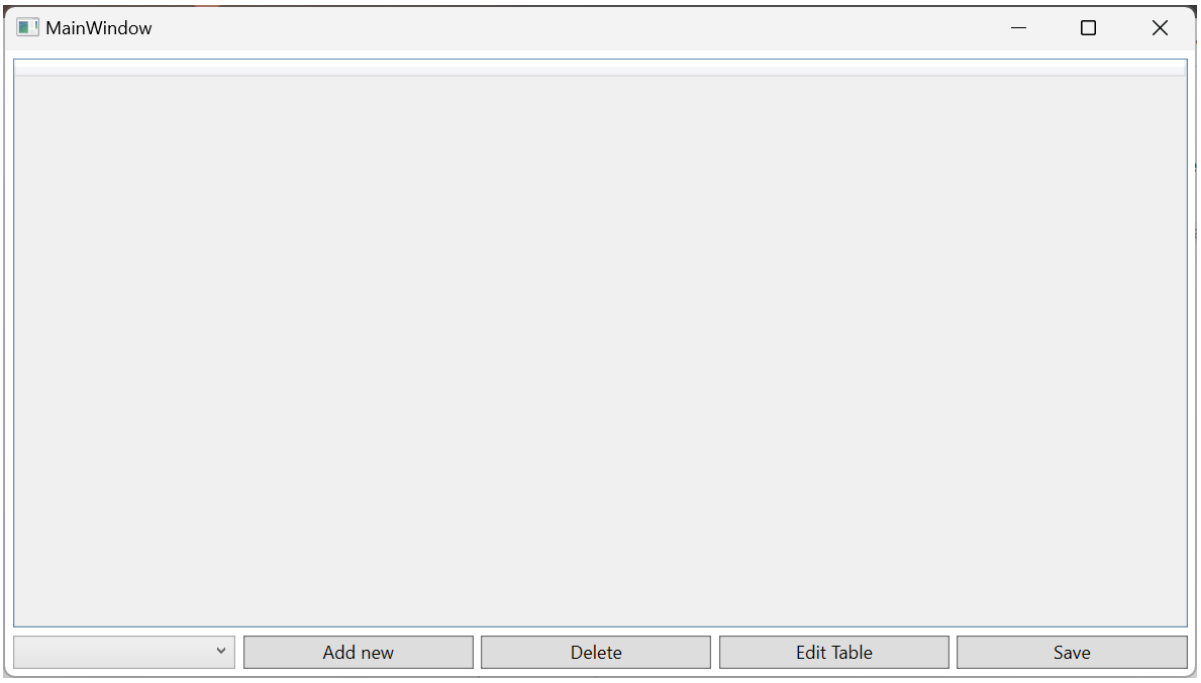
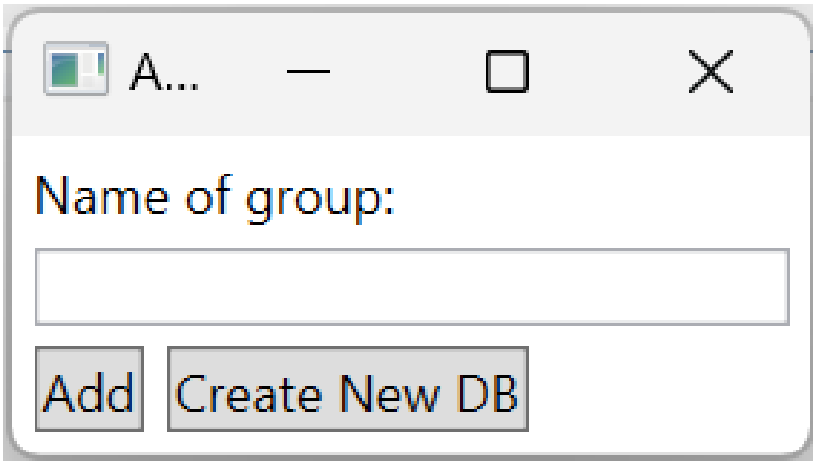


Рис 1. Начальный экран приложения

При нажатии на кнопку Add new открывается новое окно:

1. Add – добавляет таблицы в базу данных.
2. Create New DB – создает новую базу данных.



Рису 2. Создание новой группы

При вводе имени (Рисунок 3.) и последующем нажатии на кнопку Add, при успешном выполнении добавления появится окно об успехе операции (Рисунок 4.)

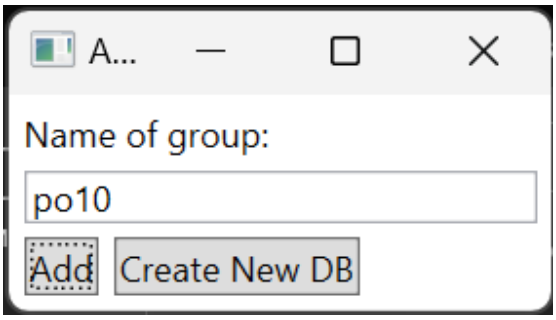


Рис 3. Вывод данных

Сообщение об удачном создании таблицы:

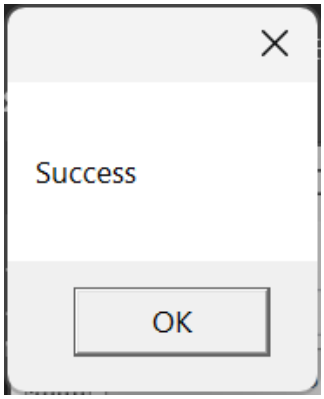


Рис 4. MessageBox (Success)

При нажатии на кнопку Create New DB будет создана новая база данных если она отсутствует, иначе появится уведомление об ошибке.

При нажатии на кнопку удалить будет показано окно с уточнением выбранного действия:

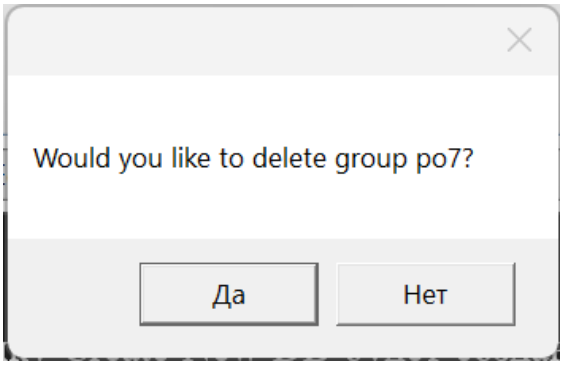


Рис 5. Окно с подтверждением действия

При нажатии на кнопку Edit Table, появляется окно EditWindow с возможность редактирования таблицы, при нажатии на кнопку Save происходит перегенерация основной таблицы

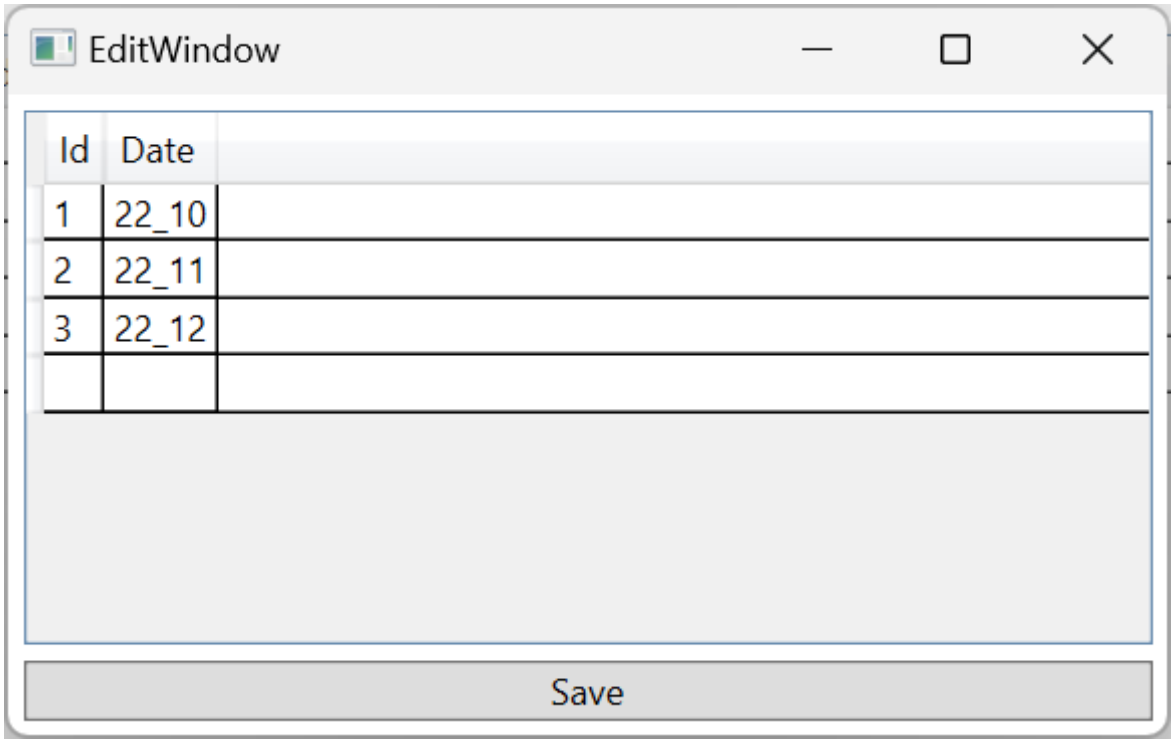


Рисунок 6. Окно Edit Table

Пример работы программы, выбранная группа выводится в DataGrid, где может быть отредактирована пользователем:

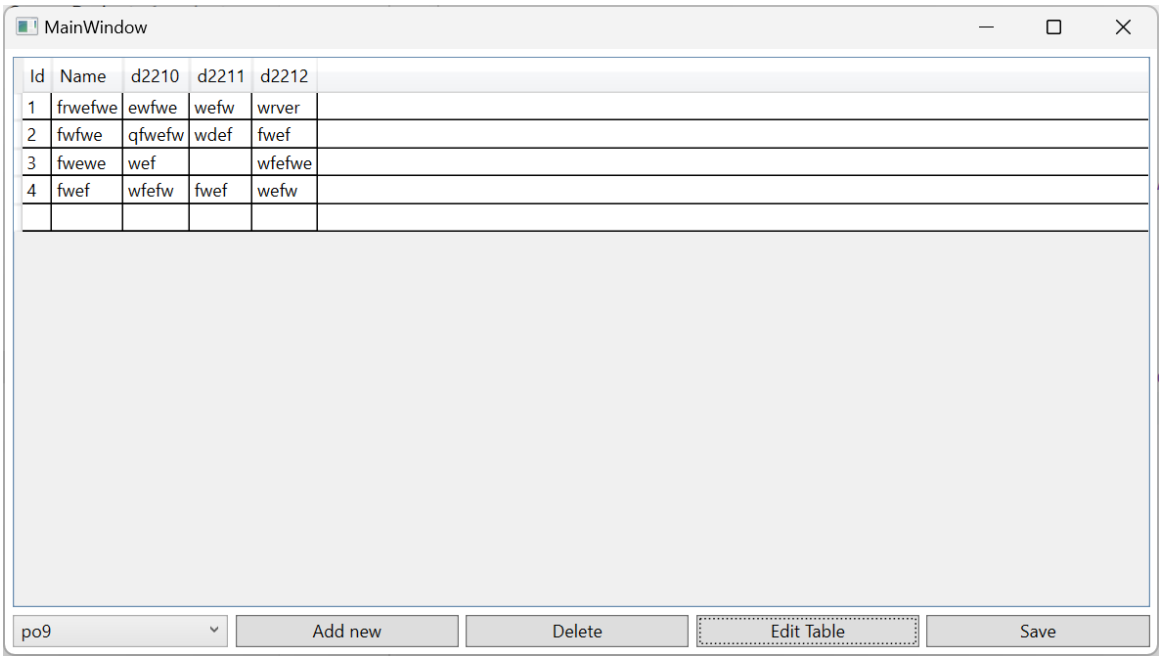


Рисунок 7. Пример работы приложения

5. ЗАКЛЮЧЕНИЕ

Согласно условиям задания, было разработано графическое приложение, ориентированное на ввод, хранение и отображение информации. Осуществлена связь с базой данных для управления данными, а также предусмотрены:

- Обработка исключительных ситуаций (ввод некорректных данных и т.д.)
- Возможность возврата (навигация);
- Запрос на подтверждение удаления вида «Вы действительно хотите удалить эту запись?»

Было реализовано индивидуальное задание и требуемые функциональные возможности GUI.

Полученная в итоге программа адекватно отвечает требованиям предметной области и пригодна для повседневного использования. Однако, ее функционал весьма серьезно ограничен по сравнению с, например, MS Excel, что не позволяет ей захватить широкую аудиторию. Таким образом, была получена узкоспециализированная программа-аналог MS Excel, заточенная под определенную предметную область.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

- Извлечение данных из DataReader URL: https://learn.microsoft.com/ru-ru/dotnet/framework/data/adonet/retrieving-data-using-a-datareader (дата обращения: 24.09.2023);
- Класс DataAdapter в .Net 8.0: https://learn.microsoft.com/ru-ru/dotnet/api/system.data.datatable?view=net-8.0 (дата обращения: 26.09.2023);
- Руководство по WPF // URL: https://metanit.com/sharp/wpf/ (дата обращения: 17.10.2023).
- Руководство по C# // URL: https://metanit.com/sharp/tutorial/ (дата обращения: 17.10.2023).
- Руководство по MS SQL Server // URL: https://metanit.com/sql/sqlserver/ (дата обращения: 17.10.2023).