Приложение Г.

Код программы

DB.cs

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows;

using System.Windows.Controls;

namespace BDZarplata.Classes

{

class DB

{

/// <summary>

/// загрузка данных из БД в DataGrid

/// </summary>

/// <param name="dataGrid"></param>

/// <param name="sql">SQL команда для выполнения (типа SELECT) </param>

public static void loadDataGrid(DataGrid dataGrid, string sql)

{

DB\_Connect.OpenConnection();

SqlDataAdapter mysqldataAdapter = new SqlDataAdapter(sql, DB\_Connect.connectionString);

DataTable dataTable = new DataTable();

mysqldataAdapter.Fill(dataTable);

dataGrid.ItemsSource = dataTable.DefaultView;

DB\_Connect.CloseConnection();

}

/// <summary>

/// Загрузка данных из БД в список

/// </summary>

/// <param name="comboBox">Список куда будет загружены данные из БД</param>

/// <param name="sql">SQL команда по которой происходит выборка</param>

/// <param name="numberCol">Номер столбца из выборки (начиная с 0) который присваивается списку </param>

public static void LoadDataComboBox(ComboBox comboBox, string sql, int numberCol)

{

DB\_Connect.OpenConnection();

SqlCommand command = new SqlCommand(sql, DB\_Connect.myConnection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{ comboBox.Items.Add(reader.GetValue(numberCol).ToString()); }

DB\_Connect.CloseConnection();

}

/// <summary>

/// Загрузка данных из БД в список

/// </summary>

/// <param name="listbox">Список куда будет загружены данные из БД</param>

/// <param name="sql">SQL команда по которой происходит выборка</param>

/// <param name="numberCol">Номер столбца из выборки (начиная с 0) который присваивается списку </param>

public static void LoadDataListBox(ListBox listbox, string sql, int numberCol)

{

DB\_Connect.OpenConnection();

SqlCommand command = new SqlCommand(sql, DB\_Connect.myConnection);

SqlDataReader reader = command.ExecuteReader();

while (reader.Read())

{ listbox.Items.Add(reader.GetValue(numberCol).ToString()); }

DB\_Connect.CloseConnection();

}

/// <summary>

/// Выполнение SQL Запроса к БД

/// </summary>

/// <param name="sql">SQL команда для выполнения</param>

/// <returns>Первый столбец первой строки набора результатов или пустая ссылка</returns>

public static object queryScalar(string sql)

{

Manager.UpdateLabel("Выполняю запрос...");

object sqlValue = null;

DB\_Connect.OpenConnection();

try

{

SqlCommand command = new SqlCommand(sql, DB\_Connect.myConnection);

sqlValue = command.ExecuteScalar();

Manager.UpdateLabel("Запрос успешно выполнен!");

}

catch (Exception ex) { MessageBox.Show("Произошла Ошибка при обработке SQL запроса \n" + ex.Message + "\n SQL Запрос: \n" + sql, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); }

finally { DB\_Connect.CloseConnection(); }

if (sqlValue != null)

{

return sqlValue;

}

else

{

sqlValue = "null";

return sqlValue;

}

}

/// <summary>

/// Выполнение запроса к БД C возвратом указанного столбца

/// </summary>

/// <param name="SQL\_Comand">SQL команда для выполнения</param>

/// <param name="Column"> Номер возвращаемого столбца</param>

/// <returns>Указанный столбец первой строки набора результатов или пустая ссылка</returns>

public static string queryScalar(string SQL\_Comand, int Column)

{

try

{

Manager.UpdateLabel("Выполняю запрос...");

DB\_Connect.OpenConnection();

SqlCommand command = new SqlCommand(SQL\_Comand, DB\_Connect.myConnection);

SqlDataReader reader = command.ExecuteReader();

reader.Read();

string temp = reader[Column].ToString();

Manager.UpdateLabel("Запрос успешно выполнен!");

if (temp != null)

{

return temp;

}

else

{

temp = "null";

return temp;

}

}

catch (Exception ex) { MessageBox.Show("Произошла Ошибка \n" + ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); return ""; }

finally { DB\_Connect.CloseConnection(); }

}

/// <summary>

/// Выполнение запроса к БД

/// </summary>

/// <param name="SQL\_Comand">SQL команда для выполнения</param>

/// <param name="Column"> Название возвращаемого столбца</param>

/// <returns>Указанный столбец первой строки набора результатов или пустая ссылка</returns>

public static string queryScalar(string SQL\_Comand, string Column)

{

try

{

Manager.UpdateLabel("Выполняю запрос...");

DB\_Connect.OpenConnection();

SqlCommand command = new SqlCommand(SQL\_Comand, DB\_Connect.myConnection);

SqlDataReader reader = command.ExecuteReader();

reader.Read();

string temp = reader[Column].ToString();

Manager.UpdateLabel("Запрос успешно выполнен!");

if (temp != null)

{

return temp;

}

else

{

temp = "";

return temp;

}

}

catch (Exception ex) { MessageBox.Show("Произошла Ошибка \n" + ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error); return ""; }

finally { DB\_Connect.CloseConnection(); }

}

/// <summary>

/// Выполнение запроса к БД

/// </summary>

/// <param name="SQL\_Comand">SQL команда для выполнения</param>

/// <param name="Columns">Массив индексов возвращаемых столбцов (начиная с 0) </param>

/// <returns> массив значений Указанных столбцов первой строки набора результатов или пустая ссылка</returns>

public static string[] queryScalar(string SQL\_Comand, int[] Columns)

{

string[] results = new string[Columns.Length];

try

{

Manager.UpdateLabel("Выполняю запрос...");

DB\_Connect.OpenConnection();

SqlCommand command = new SqlCommand(SQL\_Comand, DB\_Connect.myConnection);

SqlDataReader reader = command.ExecuteReader();

reader.Read();

for (int i = 0; i < Columns.Length; i++)

{

results[i] = reader[Columns[i]].ToString();

}

Manager.UpdateLabel("Запрос успешно выполнен!");

return results;

}

catch (Exception ex)

{

MessageBox.Show("Произошла Ошибка при чтении данных \n" + ex.Message, "Ошибка\n", MessageBoxButton.OK, MessageBoxImage.Error);

results = new[] { "" };

return results;

}

finally { DB\_Connect.CloseConnection(); }

}

/// <summary>

/// Выполнение запроса к БД

/// </summary>

/// <param name="SQL\_Comand">SQL команда для выполнения(Не рекомендуется использовать эту перегрузку если в запросе содержаться столбцы с одинаковым именем)</param>

/// <param name="Columns">Массив названий возвращаемых столбцов </param>

/// <returns> массив значений Указанных столбцов первой строки набора результатов или пустая ссылка</returns>

public static string[] queryScalar(string SQL\_Comand, string[] Columns)

{

string[] results = new string[Columns.Length];

try

{

Manager.UpdateLabel("Выполняю запрос...");

DB\_Connect.OpenConnection();

SqlCommand command = new SqlCommand(SQL\_Comand, DB\_Connect.myConnection);

SqlDataReader reader = command.ExecuteReader();

reader.Read();

for (int i = 0; i < Columns.Length; i++)

{

results[i] = reader[Columns[i]].ToString();

}

Manager.UpdateLabel("Запрос успешно выполнен!");

return results;

}

catch (Exception ex)

{

MessageBox.Show("Произошла Ошибка при чтении данных \n" + ex.Message, "Ошибка\n", MessageBoxButton.OK, MessageBoxImage.Error);

results = new[] { "" };

return results;

}

finally { DB\_Connect.CloseConnection(); }

}

/// <summary>

/// Запрос к БД с получением данных из таблицы (таблиц)

/// </summary>

/// <param name="SQL\_Comand">SQL команда для выполнения(типа Select)</param>

/// <param name="CountColumn">Количество столбцов для возврата</param>

/// <param name="CountString">Количество строк для возврата(возврат начинается с 1 строки возвращаемым запросом</param>

/// <param name="Results">[c,s]Массив результатов где перебор с - столбца , s- строки</param>

public static void ReturnTable(string SQL\_Comand, out string[,] Results, int CountString = 1, int CountColumn = 1)

{

Results = new string[CountColumn, CountString];

try

{

Manager.UpdateLabel("Выполняю запрос...");

DB\_Connect.OpenConnection();

SqlCommand command = new SqlCommand(SQL\_Comand, DB\_Connect.myConnection);

SqlDataReader reader = command.ExecuteReader();

int s = 0;

while (reader.Read() && s < CountString)

{

for (int с = 0; с < CountColumn; с++)

{

Results[с, s] = reader[с].ToString();

}

s++;

}

Manager.UpdateLabel("Запрос успешно выполнен!");

}

catch (Exception ex)

{

MessageBox.Show("Произошла Ошибка при чтении таблицы \n" + ex.Message, "Ошибка\n", MessageBoxButton.OK, MessageBoxImage.Error);

}

finally { DB\_Connect.CloseConnection(); }

}

/// <summary>

/// Выполнение внешнего SQl Файла или запроса с возвратом количества строк

/// </summary>

/// <param name="sql"></param>

/// <returns> количество задействованных в инструкции строк или -1 если при обработке запроса происходит ошибка </returns>

public static int queryData(string sql)

{

Manager.UpdateLabel("Выполняю запрос...");

DB\_Connect.OpenConnection();

try

{

SqlCommand command = new SqlCommand(sql, DB\_Connect.myConnection);

return command.ExecuteNonQuery();

}

catch (Exception ex)

{

MessageBox.Show("Произошла Ошибка \n" + ex.Message, "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return -1;

}

finally { DB\_Connect.CloseConnection(); }

}

}

}

DB\_Connect.cs

using System;

using System.Data.SqlClient;

using System.Windows;

namespace BDZarplata.Classes

{

public class DB\_Connect

{

public static string connectionString = "Data Source=localhost; Initial Catalog=BD\_Zarplata; Integrated Security=true;";

public static SqlConnection myConnection = new SqlConnection(connectionString);

/// <summary>

/// создание соединения с БД

/// </summary>

public static void OpenConnection()

{

if (myConnection.State == System.Data.ConnectionState.Closed)

{

myConnection.Open();

}

}

/// <summary>

/// закрытие соединения с БД

/// </summary>

public static void CloseConnection()

{

if (myConnection.State == System.Data.ConnectionState.Open)

{

myConnection.Close();

}

}

/// <summary>

/// Проверка на возможность установки соединения с БД

/// </summary>

/// <param name="DataSource">Адрес компьютера на котором храниться БД</param>

/// <param name="InitialCatalog">Название БД</param>

/// <param name="IntegratedSecurity">проверку подлинности</param>

/// <returns></returns>

public static bool OpenClouseConnection(string DataSource = "localhost", string InitialCatalog = "BD\_Zarplata", bool IntegratedSecurity = true)

{

string connectStr = "Data Source=" + DataSource + "; Initial Catalog=" + InitialCatalog + "; Integrated Security=" + IntegratedSecurity + ";";

try

{

SqlConnection userConnection = new SqlConnection(connectStr);

if (userConnection.State == System.Data.ConnectionState.Closed)

{

userConnection.Open();

}

if (userConnection.State == System.Data.ConnectionState.Open)

{

userConnection.Close();

myConnection = userConnection;

}

return true;

}

catch (Exception ex)

{

MessageBox.Show("Произошла ошибка:\n" + ex.Message + "\n Проверьте корректность введенных данных\n" + connectStr);

return false;

}

}

}

}

DG.cs

using System;

using System.Windows.Controls;

namespace BDZarplata.Classes

{

class DG

{

/// <summary>

/// Получение значения выбранной ячейки

/// </summary>

/// <param name="dataGrid1">DataGrid в котором находиться нужная ячейка</param>

/// <param name="selectedColumn">Номер столбца (начиная с 0) в котором находиться ячейка.

/// Присвоить -1 для автоматического определения столбца </param>

/// <returns>значение ячейки или ""</returns>

public static string GetSelectCell(DataGrid dataGrid1, int selectedColumn = -1)

{

try

{

string result = "";

if (selectedColumn == -1) { selectedColumn = dataGrid1.CurrentCell.Column.DisplayIndex; }

var selectedCell = dataGrid1.SelectedCells[selectedColumn];

var cellContent = selectedCell.Column.GetCellContent(selectedCell.Item);

if (cellContent is TextBlock)

{

result = (cellContent as TextBlock).Text;

}

return result;

}

catch (Exception ex) { Manager.UpdateLabel("Ошибка в returnCell, возможно не выбрана ячейка для возврата \n" + ex.Message); return ""; }

}

}

}

Manager.cs

using System.Windows.Controls;

namespace BDZarplata.Classes

{

class Manager

{

public static Frame MainFrame { get; set; }

public static Label LabelStatus { get; set; }

public static ProgressBar MainProgressBar { get; set; }

/// <summary>

/// отладочный вывод сообщений в отдельной Label

/// </summary>

/// <param name="newContent">новое сообщение для вывода</param>

public static void UpdateLabel(string newContent = "")

{

LabelStatus.Content = newContent;

}

}

}

Procedure.cs

using System.Collections.Generic;

using System.Windows;

namespace BDZarplata.Classes

{

class Procedure

{

/// <summary>

/// Обновление значений указанной таблицы в БД

/// </summary>

/// <param name="NameTable">Имя таблицы</param>

/// <param name="ColumnsName">Массив имен столбцов</param>

/// <param name="ColumnsData">Массив значений этих столбцов</param>

/// <param name="Where">Условие отбора</param>

public static void UpdateTable(string NameTable, List<string> ColumnsName, List<string> ColumnsData, string Where)

{

string sqlComand = $"UPDATE {NameTable} SET {ColumnsName[0]} = {ColumnsData[0]}";

if (ColumnsData.Count == ColumnsName.Count)

{

for (int i = 1; i < ColumnsData.Count; i++)

{

sqlComand += $", {ColumnsName[i]} = {ColumnsData[i]}";

}

sqlComand += Where;

Classes.DB.queryScalar(sqlComand);

}

else

{

MessageBox.Show("err UpdateTable \n Количество столбцов не совпадает с количеством значений");

}

}

/// <summary>

/// Добавление значений указанной таблицы в БД

/// </summary>

/// <param name="NameTable">Имя таблицы</param>

/// <param name="ColumnsName">Массив имен столбцов</param>

/// <param name="ColumnsData">Массив значений этих столбцов</param>

public static void InsertTable(string NameTable, List<string> ColumnsName, List<string> ColumnsData)

{

string sqlComand = $"INSERT INTO {NameTable} ( {ColumnsName[0]}";

if (ColumnsData.Count == ColumnsName.Count)

{

for (int i = 1; i < ColumnsName.Count; i++)

{

sqlComand += $", {ColumnsName[i]}";

}

sqlComand += $") VALUES( {ColumnsData[0]} ";

for (int i = 1; i < ColumnsData.Count; i++)

{

sqlComand += $", {ColumnsData[i]}";

}

sqlComand += ")";

DB.queryScalar(sqlComand);

}

else

{

MessageBox.Show("err InsertTable \n Количество столбцов не совпадает с количеством значений");

}

}

}

}

MainWindow.xaml

<Window x:Class="BDZarplata.MainWindow"

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

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

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:local="clr-namespace:BDZarplata"

mc:Ignorable="d"

Title="BDZArlata" Height="600" Width="1500" MinWidth="800" MinHeight="600" FontSize="16">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="1\*"/>

<ColumnDefinition Width="310"/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="50"/>

<RowDefinition/>

<RowDefinition Height="50"/>

</Grid.RowDefinitions>

<!--создание рамок-->

<Rectangle Grid.ColumnSpan="3">

<Rectangle.Fill>

<LinearGradientBrush EndPoint="0.5,1" MappingMode="RelativeToBoundingBox" StartPoint="0.5,0">

<GradientStop Color="White" Offset="1"/>

<GradientStop Color="#FF6666FF"/>

</LinearGradientBrush>

</Rectangle.Fill>

</Rectangle>

<Rectangle Grid.Row="2" Grid.ColumnSpan="3" Fill="#FF6666FF"/>

<!--страницы приложения-->

<Frame

Name="MainFrame"

Grid.Row="1"

Grid.ColumnSpan="3"

NavigationUIVisibility="Hidden"

></Frame>

<!--вспомогательные информационные элементы-->

<Label x:Name="LabelStatus1" Grid.Row="2" Foreground="#FFFFFF66" Content="Готов." />

<ProgressBar x:Name="ProgressBarStatus" Grid.Row="2" Grid.Column="0" Margin="0.4,34.6,-0.4,0" Visibility="Hidden"/>

<Button x:Name="BtnBack" Grid.Row="2" Grid.Column="2" Click="BtnBack\_Click">Назад</Button>

</Grid>

</Window>

Page\_SotrudnikMainInfo.xaml

<Page x:Class="BDZarplata.Pages.Page\_SotrudnikMainInfo"

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

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

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:BDZarplata.Pages"

mc:Ignorable="d"

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

Title="PageSotrudnikMainInfo">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition/>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition/>

<RowDefinition Height="Auto"></RowDefinition>

</Grid.RowDefinitions>

<Menu>

<Button x:Name="Btn\_Add" Cursor="Hand" Content="Добавить" Visibility="Visible" Click="Btn\_Add\_Click"/>

<Button x:Name="Btn\_Redactir" Cursor="Hand" Content="Изменить" Visibility="Collapsed" Click="Btn\_Redactir\_Click"/>

<Button x:Name="Btn\_delete" Cursor="Hand" Content="Удалить" Visibility="Collapsed"/>

<Button x:Name="Btn\_Save" Cursor="Hand" Content="Сохранить" Visibility="Collapsed" Click="Btn\_Save\_Click"/>

</Menu>

<TabControl Grid.Row="1" x:Name="TabC\_Main">

<TabItem x:Name="TabI\_MainData" Header="Основные данные сотрудников" GotFocus="TabI\_MainData\_GotFocus">

<DataGrid x:Name="DG\_Sotrud\_Anketa" Grid.ColumnSpan="2" Grid.RowSpan="2" MouseDoubleClick="DG\_Sotrud\_Anketa\_MouseDoubleClick" AutoGenerateColumns="False">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Path=idSotrudnik}" Header="Табельный Номер" />

<DataGridTextColumn Binding="{Binding Path=full\_name}" Header="Полное Имя"/>

<DataGridTextColumn Binding="{Binding Path=title}" Header="Должность"/>

<DataGridTextColumn Binding="{Binding Path=family\_status}" Header="Семейное Положение"/>

<DataGridTextColumn Binding="{Binding Path=num\_zd\_kids}" Header="Здоровых Детей"/>

<DataGridTextColumn Binding="{Binding Path=num\_invalid\_kids}" Header="Детей Инвалидов"/>

<DataGridTextColumn Binding="{Binding Path=opeka}" Header="Опекаемых Детей инвалидов"/>

<DataGridTextColumn Binding="{Binding Path=SpecStatus}" Header="Специальный статус"/>

<DataGridTextColumn Binding="{Binding Path=Staj}" Header="Стаж"/>

</DataGrid.Columns>

</DataGrid>

</TabItem>

<TabItem Header="Расписание" GotFocus="TabItem\_GotFocus">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"></ColumnDefinition>

<ColumnDefinition/>

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<ListBox x:Name="LB\_Sotrud\_FIO" Grid.Column="0" SelectionChanged="LB\_Sotrud\_FIO\_SelectionChanged" ></ListBox>

<ListBox x:Name="LB\_Sotrud\_id" SelectionChanged="LB\_Sotrud\_id\_SelectionChanged" Visibility="Hidden" ></ListBox>

<DataGrid x:Name="DG\_Raspisnie"

AutoGenerateColumns="False"

Grid.Column="1"

SelectionChanged="DG\_Raspisnie\_SelectionChanged" >

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Path=Date}" Header="Дата" />

<DataGridTextColumn Binding="{Binding Path=StatusSotrud}" Header="Статус сотрудника" />

<DataGridTextColumn Binding="{Binding Path=StatusDay}" Header="Статус дня" />

</DataGrid.Columns>

</DataGrid>

<StackPanel Grid.Column="2">

<Calendar x:Name="calendar\_raspisan"></Calendar>

<Label>Статус Сотрудника</Label>

<ComboBox x:Name="CB\_StatusSotrud" SelectedIndex="1"></ComboBox>

<Label>Статус дня</Label>

<ComboBox x:Name="CB\_StatusDay" SelectedIndex="1"></ComboBox>

<Button x:Name="BTN\_RedRaspisan" IsEnabled="False" Click="BTN\_RedRaspisan\_Click">Сохранить Изменения</Button>

</StackPanel>

</Grid>

</TabItem>

</TabControl>

</Grid>

</Page>

Page\_BughalterInfo.xaml

<Page x:Class="BDZarplata.Pages.Page\_BughalterInfo"

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

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

xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"

xmlns:d="http://schemas.microsoft.com/expression/blend/2008"

xmlns:local="clr-namespace:BDZarplata.Pages"

mc:Ignorable="d"

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

Title="Page\_BughalterInfo">

<Grid>

<Grid.ColumnDefinitions>

<ColumnDefinition Width="\*"></ColumnDefinition>

</Grid.ColumnDefinitions>

<Grid.RowDefinitions>

<RowDefinition Height="Auto"></RowDefinition>

<RowDefinition></RowDefinition>

</Grid.RowDefinitions>

<Menu>

<Button x:Name="Btn\_Add" Cursor="Hand" Content="Добавить" Visibility="Collapsed" Click="Btn\_Add\_Click"/>

<Button x:Name="Btn\_Redactir" Cursor="Hand" Content="Изменить" Visibility="Collapsed" Click="Btn\_Redactir\_Click"/>

<Button x:Name="Btn\_delete" Cursor="Hand" Content="Удалить" Visibility="Collapsed" Click="Btn\_Delete\_Click"/>

<Button x:Name="Btn\_Save" Cursor="Hand" Content="Сохранить" Visibility="Collapsed" Click="Btn\_Save\_Click"/>

<Button x:Name="Btn\_Raschet" Cursor="Hand" Content="Перейти к Созданию Расчётного листа..." Click="Btn\_Raschet\_Click"></Button>

</Menu>

<TabControl Grid.Row="1" x:Name="TabC\_Main">

<TabItem x:Name="TabI\_OkladSotrud" Header="Оклады сотрудников" GotFocus="TabI\_OkladSotrud\_GotFocus" >

<DataGrid x:Name="DG\_SotridnikOklad"

AutoGenerateColumns="False"

SelectionChanged="DG\_SotridnikOklad\_SelectionChanged" MouseDoubleClick="DG\_SotridnikOklad\_MouseDoubleClick" >

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Path=idDoljnost}" Header="№"/>

<DataGridTextColumn Binding="{Binding Path=title}" Header="Название"/>

<DataGridTextColumn Binding="{Binding Path=Oklad}" Header="Оклад"/>

<DataGridTextColumn Binding="{Binding Path=Travmat}" Header="Травматизм"/>

</DataGrid.Columns>

</DataGrid>

</TabItem>

<TabItem x:Name="TabI\_Nadbav" Header="Штрафы и надбавки" GotFocus="TabI\_Nadbav\_GotFocus">

<Grid >

<Grid.ColumnDefinitions>

<ColumnDefinition Width="Auto"></ColumnDefinition>

<ColumnDefinition Width="Auto"/>

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<ListBox x:Name="LB\_Sotrud\_FIO2" Grid.Column="0" SelectionChanged="LB\_Sotrud\_FIO2\_SelectionChanged"></ListBox>

<ListBox x:Name="LB\_Sotrud\_id2" SelectionChanged="LB\_Sotrud\_id2\_SelectionChanged" Visibility="Collapsed"></ListBox>

<DataGrid x:Name="DG\_NadbavShtraf"

AutoGenerateColumns="False"

Grid.Column="1"

SelectionChanged="DG\_NadbavShtraf\_SelectionChanged">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Path=Data}" Header="Дата выплаты"/>

<DataGridTextColumn Binding="{Binding Path=Nadbav}" Header="Сумма надбавок"/>

<DataGridTextColumn Binding="{Binding Path=Vichet}" Header="Сумма штрафов"/>

</DataGrid.Columns>

</DataGrid>

<StackPanel Grid.Column="2">

<Calendar x:Name="Calendar2" FontSize="14"/>

<ComboBox x:Name="CB\_Data\_Nadbav" SelectedIndex="0" Visibility="Hidden">

<ComboBoxItem x:Name="CBI\_New\_Ellement" Background="#FFB1DBF3" Content="&lt;Добавить Новый Эллемент&gt;" FontWeight="Bold" FontStyle="Italic"/>

</ComboBox>

<Label>Надбавка</Label>

<TextBox x:Name="TB\_Nadbav" ></TextBox>

<Label>Вычет</Label>

<TextBox x:Name="TB\_Vichet" ></TextBox>

<Button x:Name="Btn\_Red\_Nadbav"

IsEnabled="False"

Click="Btn\_Red\_Nadbav\_Click">Сохранить изменения

</Button>

</StackPanel>

</Grid>

</TabItem>

<TabItem Header="Льготы и налоги" x:Name="TabI\_LN" GotFocus="TabI\_LN\_GotFocus" Initialized="TabI\_LN\_Initialized" Margin="-1,-2,-3,0">

<Grid >

<Grid.ColumnDefinitions>

<ColumnDefinition Width="62"></ColumnDefinition>

<ColumnDefinition Width="320"/>

<ColumnDefinition></ColumnDefinition>

</Grid.ColumnDefinitions>

<StackPanel Grid.Column="0" Grid.ColumnSpan="2">

<Label>НДФЛ</Label>

<Label>ПФР</Label>

<Label>ФСС</Label>

<Label>ФОМС</Label>

<Label>Размер льготы за 1 и 2 ребенка</Label>

<Label>Размер льготы за 3 и последующих детей</Label>

<Label>Размер льготы за ребенка-инвалида</Label>

<Label>Размер льготы за опеку над ребенком инвалидом</Label>

<Label>МРОТ</Label>

</StackPanel>

<StackPanel Grid.Column="2">

<TextBox x:Name="TB\_NDFL" Height="25" Margin="0,6,0,0" PreviewTextInput="floatOnly\_PreviewTextInput"></TextBox>

<TextBox x:Name="TB\_PFR" Height="25" Margin="0,6,0,0" PreviewTextInput="floatOnly\_PreviewTextInput"></TextBox>

<TextBox x:Name="TB\_FCC" Height="25" Margin="0,6,0,0" PreviewTextInput="floatOnly\_PreviewTextInput"></TextBox>

<TextBox x:Name="TB\_FOMC" Height="25" Margin="0,6,0,0" PreviewTextInput="floatOnly\_PreviewTextInput"></TextBox>

<TextBox x:Name="TB\_Kid1" Height="25" Margin="0,6,0,0" PreviewTextInput="intOnly\_PreviewTextInput"></TextBox>

<TextBox x:Name="TB\_Kid3" Height="25" Margin="0,6,0,0" PreviewTextInput="intOnly\_PreviewTextInput"></TextBox>

<TextBox x:Name="TB\_KidInvalid" Height="25" Margin="0,6,0,0" PreviewTextInput="intOnly\_PreviewTextInput"></TextBox>

<TextBox x:Name="TB\_KidInvalid\_opek" Height="25" Margin="0,6,0,0" PreviewTextInput="intOnly\_PreviewTextInput"></TextBox>

<TextBox x:Name="TB\_Mrot" Height="25" Margin="0,6,0,0" PreviewTextInput="intOnly\_PreviewTextInput" ></TextBox>

</StackPanel>

</Grid>

</TabItem >

<TabItem x:Name="TabI\_Base" Header="Предельная база для исчисления страховых взносов" GotFocus="TabI\_Base\_GotFocus">

<DataGrid x:Name="DG\_Base" AutoGenerateColumns="False">

<DataGrid.Columns>

<DataGridTextColumn Binding="{Binding Path=Y}" Header="Год"/>

<DataGridTextColumn Binding="{Binding Path=Z}" Header="Значение"/>

</DataGrid.Columns>

</DataGrid>

</TabItem>

</TabControl>

</Grid>

</Page>

Page\_AddRedAnketa.xaml.cs

using BDZarplata.Classes;

using System;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace BDZarplata.Pages

{

/// <summary>

/// Логика взаимодействия для Page\_AddRedAnketa.xaml

/// </summary>

public partial class Page\_AddRedAnketa : Page

{

string IDSotrud;

/// <summary>

/// Загрузка формы

/// </summary>

/// <param name="IdSotrud"></param>

public Page\_AddRedAnketa(string IdSotrud = "")

{

IDSotrud = IdSotrud;

InitializeComponent();

DB.LoadDataComboBox(CB\_Doljnost, "SELECT [idDoljnost], [title] FROM [BD\_Zarplata].[bd\_zarplta].[doljnost]", 1);

DB.LoadDataComboBox(CB\_DoljnostID, "SELECT [idDoljnost], [title] FROM [BD\_Zarplata].[bd\_zarplta].[doljnost]", 0);

if (IdSotrud != "")

{

TB\_FIO.Text = DB.queryScalar("SELECT [full\_name] FROM [BD\_Zarplata].[bd\_zarplta].[sotrudnik] WHERE [idSotrudnik]= " + IdSotrud).ToString();

TB\_FamilyStatus.Text = DB.queryScalar("SELECT [family\_status] FROM [BD\_Zarplata].[bd\_zarplta].[sotrudnik] WHERE [idSotrudnik]= " + IdSotrud).ToString();

TB\_KidInvalid.Text = DB.queryScalar("SELECT [num\_invalid\_kids] FROM [BD\_Zarplata].[bd\_zarplta].[sotrudnik] WHERE [idSotrudnik]= " + IdSotrud).ToString();

TB\_KidInvalid\_opek.Text = DB.queryScalar("SELECT [opeka] FROM [BD\_Zarplata].[bd\_zarplta].[sotrudnik] WHERE [idSotrudnik]= " + IdSotrud).ToString();

TB\_kidsZd.Text = DB.queryScalar("SELECT [num\_zd\_kids] FROM [BD\_Zarplata].[bd\_zarplta].[sotrudnik] WHERE [idSotrudnik]= " + IdSotrud).ToString();

TB\_SchetZachisl.Text = DB.queryScalar("SELECT [SchetZachisl] [SchetZachisl] FROM [BD\_Zarplata].[bd\_zarplta].[sotrudnik] WHERE [idSotrudnik]= " + IdSotrud).ToString();

TB\_Staj.Text = DB.queryScalar("SELECT [Staj] FROM [BD\_Zarplata].[bd\_zarplta].[sotrudnik] WHERE [idSotrudnik]= " + IdSotrud).ToString();

CB\_DoljnostID.SelectedItem = DB.queryScalar("SELECT [idDoljnost] FROM [BD\_Zarplata].[bd\_zarplta].[sotrudnik] WHERE [idSotrudnik]= " + IdSotrud).ToString();

CB\_Doljnost.SelectedIndex = CB\_DoljnostID.SelectedIndex;

CB\_SpecStatus.SelectedIndex = Convert.ToInt32(DB.queryScalar("SELECT [SpecStatus] FROM [BD\_Zarplata].[bd\_zarplta].[sotrudnik] WHERE [idSotrudnik]= " + IdSotrud));

}

}

/// <summary>

/// соединение 2 списков для синхронного выбора

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void CB\_Doljnost\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

CB\_DoljnostID.SelectedIndex = CB\_Doljnost.SelectedIndex;

}

/// <summary>

/// Фильтр int значений для TextBox

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void intOnly\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

//узнаем Кто вызвал событие

TextBox textBox = sender as TextBox;

//проверка что введена цифра

if (!Char.IsDigit(e.Text, 0))

{

e.Handled = true;

MessageBox.Show("Только Целые числа", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

}

//проверяем выходит ли за предел int

else if (!int.TryParse(textBox.Text + e.Text, out \_))

{

MessageBox.Show("Максимальный размер числа не может быть больше 2147483647", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

e.Handled = true;

}

}

/// <summary>

/// Проверка и внесение данных в таблицу

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Btn\_Save\_Click(object sender, RoutedEventArgs e)

{

List<string> columnNames = new List<string>() { "[full\_name]", "[idDoljnost]", "[SpecStatus]" };

List<string> RowData = new List<string>() { "'" + TB\_FIO.Text + "'", CB\_DoljnostID.SelectedItem.ToString(), CB\_SpecStatus.SelectedIndex.ToString() };

MessageBoxResult Result\_SaveOrRed;

if (TB\_FIO.Text.Length < 2)

{

MessageBox.Show("Ошибка! Поле ФИО обязательно для заполнения", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);

return;

}

if (TB\_FamilyStatus.Text != "")

{

columnNames.Add("[family\_status]");

RowData.Add("'" + TB\_FamilyStatus.Text + "'");

}

if (TB\_KidInvalid.Text != "")

{

columnNames.Add("[num\_invalid\_kids]");

RowData.Add(TB\_KidInvalid.Text);

}

if (TB\_KidInvalid\_opek.Text != "")

{

columnNames.Add("[opeka]");

RowData.Add(TB\_KidInvalid\_opek.Text);

}

if (TB\_kidsZd.Text != "")

{

columnNames.Add("[num\_zd\_kids]");

RowData.Add(TB\_kidsZd.Text);

}

if (TB\_SchetZachisl.Text.Length == 20)

{

columnNames.Add("[SchetZachisl]");

RowData.Add("'" + TB\_SchetZachisl.Text + "'");

}

else if (TB\_SchetZachisl.Text == "")

{ }

else

{

MessageBox.Show("Счёт Зачисления должен состоять из 20 цифр!");

return;

}

if (TB\_Staj.Text != "")

{

columnNames.Add("[Staj]");

RowData.Add(TB\_Staj.Text);

}

if (IDSotrud == "") { Result\_SaveOrRed = MessageBoxResult.No; }

else

{

Result\_SaveOrRed = MessageBox.Show

(

"Сохранить изменения? \n \n Да - Сохранить изменения существующей записи \n Нет - Создать новую запись \n Отмена - Возврат в окно редактирования",

"Добавить или Изменить?",

MessageBoxButton.YesNoCancel,

MessageBoxImage.Question

);

}

switch (Result\_SaveOrRed)

{

//обновление существующей записи

case MessageBoxResult.Yes:

Procedure.UpdateTable("[BD\_Zarplata].[bd\_zarplta].[sotrudnik]", columnNames, RowData, "WHERE [idSotrudnik]= " + IDSotrud);

MessageBox.Show("Изменения Сохранены!");

break;

case MessageBoxResult.No:

// вставка новой записи

Procedure.InsertTable("[BD\_Zarplata].[bd\_zarplta].[sotrudnik]", columnNames, RowData);

MessageBox.Show("Вставка новой записи завершена!");

break;

default:

Manager.UpdateLabel("Отменено пользователем");

return;

}

Manager.MainFrame.Navigate(new Page\_SotrudnikMainInfo());

}

private void Btn\_Delete\_Click(object sender, RoutedEventArgs e)

{

MessageBoxResult MR;

if (IDSotrud == "")

{

MR = MessageBox.Show("Объект для удаления не выбран. \n хотите выбрать его сейчас?", "Внимание!", MessageBoxButton.YesNo, MessageBoxImage.Warning);

if (MR == MessageBoxResult.Yes)

{

Classes.Manager.MainFrame.Navigate(new Page\_SotrudnikMainInfo());

}

else

{

return;

}

}

else if (MessageBoxResult.Yes == MessageBox.Show("Вы собираетесь безвозвратно удалить\n Вы уверены?", "Внимание!", MessageBoxButton.YesNo, MessageBoxImage.Warning))

{

//ТК в БД прописано каскадное удаление ,удаляем лишь из родительской таблицы

string comand = "DELETE FROM [bd\_zarplta].[sotrudnik] WHERE [idSotrudnik] =" + IDSotrud;

if (Classes.DB.queryData(comand) != -1)

{

Classes.Manager.UpdateLabel("Готово.");

MessageBox.Show("Операция успешна выполнена!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

Classes.Manager.MainFrame.Navigate(new Page\_SotrudnikMainInfo());

}

else

{

MessageBox.Show("Произошла ошибка при удалении");

}

}

}

}

}

Page\_BughalterInfo.xaml.cs

using BDZarplata.Classes;

using System;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace BDZarplata.Pages

{

/// <summary>

/// Логика взаимодействия для Page\_BughalterInfo.xaml

/// </summary>

public partial class Page\_BughalterInfo : Page

{

bool Select0Items = true;

public Page\_BughalterInfo()

{

Manager.MainProgressBar.Visibility = Visibility.Visible;

Manager.MainProgressBar.Maximum = 5;

InitializeComponent();

Manager.MainProgressBar.Value = 1;

DB.loadDataGrid(DG\_SotridnikOklad, "SELECT [idDoljnost], [title] ,[Oklad], [Travmat] FROM [BD\_Zarplata].[bd\_zarplta].[doljnost]");

TabI\_LN\_Initialized(null, null);

DB.loadDataGrid(DG\_Base, "SELECT \* FROM [BD\_Zarplata].[bd\_zarplta].[base]");

Classes.DB.LoadDataListBox(LB\_Sotrud\_FIO2, "SELECT [idSotrudnik],[full\_name] FROM [BD\_Zarplata].[bd\_zarplta].[sotrudnik]", 1);

Classes.DB.LoadDataListBox(LB\_Sotrud\_id2, "SELECT [idSotrudnik] ,[full\_name] FROM [BD\_Zarplata].[bd\_zarplta].[sotrudnik]", 0);

Manager.MainProgressBar.Visibility = Visibility.Collapsed;

}

/// <summary>

/// Подгруздка справочника льгот и налогов

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void TabI\_LN\_Initialized(object sender, EventArgs e)

{

Manager.MainProgressBar.Value = 2;

TB\_FCC.Text = Classes.DB.queryScalar("SELECT FCC FROM [BD\_Zarplata].[bd\_zarplta].[h]").ToString();

TB\_FOMC.Text = Classes.DB.queryScalar("SELECT FOMC FROM [BD\_Zarplata].[bd\_zarplta].[h]").ToString();

Manager.MainProgressBar.Value = 3;

TB\_Kid1.Text = Classes.DB.queryScalar("SELECT kid1 FROM [BD\_Zarplata].[bd\_zarplta].[h]").ToString();

TB\_Kid3.Text = Classes.DB.queryScalar("SELECT Kid3 FROM [BD\_Zarplata].[bd\_zarplta].[h]").ToString();

TB\_KidInvalid.Text = Classes.DB.queryScalar("SELECT invalid FROM [BD\_Zarplata].[bd\_zarplta].[h]").ToString();

TB\_KidInvalid\_opek.Text = Classes.DB.queryScalar("SELECT [invlid\_o] FROM [BD\_Zarplata].[bd\_zarplta].[h]").ToString();

Manager.MainProgressBar.Value = 5;

TB\_Mrot.Text = Classes.DB.queryScalar("SELECT MROT FROM [BD\_Zarplata].[bd\_zarplta].[h]").ToString();

TB\_NDFL.Text = Classes.DB.queryScalar("SELECT NDFL FROM [BD\_Zarplata].[bd\_zarplta].[h]").ToString();

TB\_PFR.Text = Classes.DB.queryScalar("SELECT PFR FROM [BD\_Zarplata].[bd\_zarplta].[h]").ToString();

Manager.MainProgressBar.Value = 5;

}

/// <summary>

/// сохранение измененных полей таблицы БД

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Btn\_Save\_Click(object sender, RoutedEventArgs e)

{

switch (TabC\_Main.SelectedIndex)

{

case 0:

break;

case 2:

Console.WriteLine(2);

if (MessageBoxResult.OK == MessageBox.Show("Сохранить Изменения? \n Ок -сохранить \n Отмена - Отменить все изменения", "Запрос на сохранение", MessageBoxButton.OKCancel))

{

List<string> columnName = new List<string>() { "[NDFL]", "[PFR]", "[FCC]", "[FOMC]", "[kid1]", "[kid3]", "[invalid]", "[invlid\_o]", "[MROT]" };

List<string> MasData = new List<string>() { TB\_NDFL.Text.Replace(',', '.'), TB\_PFR.Text.Replace(',', '.'), TB\_FCC.Text.Replace(',', '.'), TB\_FOMC.Text.Replace(',', '.'), TB\_Kid1.Text, TB\_Kid3.Text, TB\_KidInvalid.Text, TB\_KidInvalid\_opek.Text, TB\_Mrot.Text };

Classes.Procedure.UpdateTable("[BD\_Zarplata].[bd\_zarplta].[h]", columnName, MasData, "");

MessageBox.Show("Изменения успешно сохранены!");

}

TabI\_LN\_Initialized(null, null);

break;

case 1:

Btn\_Red\_Nadbav\_Click(sender, e);

break;

default:

Console.WriteLine("Дефульт");

break;

}

}

/// <summary>

/// Переход на страницу редактирования

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Btn\_Redactir\_Click(object sender, RoutedEventArgs e)

{

switch (TabC\_Main.SelectedIndex)

{

case 0:

Manager.MainFrame.Navigate(new Pages.Page\_BuhgaltAddEdit\_Oklad(CellID));

break;

case 2:

break;

case 3:

string Year = Classes.DG.GetSelectCell(DG\_Base, 0);

string Znach = Classes.DG.GetSelectCell(DG\_Base, 1);

Manager.MainFrame.Navigate(new Pages.Page\_Bughl\_AddEdit\_base(Year, Znach));

break;

default:

break;

}

}

private void Btn\_Add\_Click(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new Pages.Page\_Bughl\_AddEdit\_base());

}

/// <summary>

/// Удаление записи из base

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Btn\_Delete\_Click(object sender, RoutedEventArgs e)

{

string YEAR = Classes.DG.GetSelectCell(DG\_Base, 0);

if (YEAR == "")

{

MessageBox.Show("Объект для удаления не выбран. ", "Внимание!", MessageBoxButton.OK, MessageBoxImage.Warning);

}

else if (MessageBoxResult.Yes == MessageBox.Show("Вы собираетесь безвозвратно удалить запись\n Вы уверены?", "Внимание!", MessageBoxButton.YesNo, MessageBoxImage.Warning))

{

//ТК в БД прописано каскадное удаление ,удаляем лишь из родительской таблицы

string comand = "DELETE FROM [bd\_zarplta].[base] WHERE [Y] =" + YEAR;

if (Classes.DB.queryData(comand) != -1)

{

Classes.Manager.UpdateLabel("Готово.");

MessageBox.Show("Операция успешна выполнена!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

DB.loadDataGrid(DG\_Base, "SELECT \* FROM [BD\_Zarplata].[bd\_zarplta].[base]");

}

else

{

MessageBox.Show("Произошла ошибка при удалении");

}

}

}

private void intOnly\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

//узнаем Кто вызвал событие

TextBox textBox = sender as TextBox;

//проверка что введена цифра

if (!Char.IsDigit(e.Text, 0))

{

e.Handled = true;

MessageBox.Show("Только Целые числа", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

}

//проверяем выходит ли за предел int

else if (!int.TryParse(textBox.Text + e.Text, out \_))

{

MessageBox.Show("Максимальный размер числа не может быть больше 2147483647", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

e.Handled = true;

}

}

/// <summary>

/// Ограничение для TextBox для ввода только Дробных данных

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void floatOnly\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

//узнаем Кто вызвал событие

TextBox textBox = sender as TextBox;

//проверка что в строке лишь 1 .

string fullText = textBox.Text + e.Text;

if (e.Text.Contains(",") && (textBox.Text.Contains(",") || textBox.Text.Length == 0))

{

e.Handled = true;

MessageBox.Show("неверно поставлена точка", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

}

//проверка что введена цифра или .

else if (!Char.IsDigit(e.Text, 0) && e.Text != ",")

{

e.Handled = true;

MessageBox.Show("Только дробные числа", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

}

else if (textBox.Text.Length > 4)

{

e.Handled = true;

MessageBox.Show("Длина строки не может превышать 5 символов", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

/// <summary>

/// запись ID выделенной строки

/// </summary>

string CellID;

private void DG\_SotridnikOklad\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

Select0Items = false;

Btn\_Redactir.Visibility = Visibility.Visible;

CellID = DG.GetSelectCell(DG\_SotridnikOklad, 0);

}

/// <summary>

/// Переход в режим редактирования оклада

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

Manager.MainFrame.Navigate(new Pages.Page\_BuhgaltAddEdit\_Oklad(CellID));

}

/// <summary>

/// Смена видимости кнопок , при выборе вкладки

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void TabI\_OkladSotrud\_GotFocus(object sender, RoutedEventArgs e)

{

Btn\_Add.Visibility = Visibility.Collapsed;

Btn\_delete.Visibility = Visibility.Collapsed;

if (Select0Items)

{

Btn\_Redactir.Visibility = Visibility.Collapsed;

}

Btn\_Save.Visibility = Visibility.Collapsed;

}

/// <summary>

/// Смена видимости кнопок , при выборе вкладки

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void TabI\_Base\_GotFocus(object sender, RoutedEventArgs e)

{

Btn\_Add.Visibility = Visibility.Visible;

Btn\_delete.Visibility = Visibility.Visible;

Btn\_Redactir.Visibility = Visibility.Visible;

Btn\_Save.Visibility = Visibility.Collapsed;

}

/// <summary>

/// Смена видимости кнопок , при выборе вкладки

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void TabI\_LN\_GotFocus(object sender, RoutedEventArgs e)

{

Btn\_Add.Visibility = Visibility.Collapsed;

Btn\_delete.Visibility = Visibility.Collapsed;

Btn\_Redactir.Visibility = Visibility.Collapsed;

Btn\_Save.Visibility = Visibility.Visible;

Btn\_Save.IsEnabled = true;

}

private void TabI\_Nadbav\_GotFocus(object sender, RoutedEventArgs e)

{

Btn\_Add.Visibility = Visibility.Collapsed;

Btn\_delete.Visibility = Visibility.Collapsed;

Btn\_Redactir.Visibility = Visibility.Collapsed;

Btn\_Save.Visibility = Visibility.Visible;

Btn\_Save.IsEnabled = false;

}

private void LB\_Sotrud\_FIO2\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

LB\_Sotrud\_id2.SelectedIndex = LB\_Sotrud\_FIO2.SelectedIndex;

}

/// <summary>

/// Подгрузка данных о надбавках и штрафах выбранного сотрудника в DataGrid

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void LB\_Sotrud\_id2\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

Classes.DB.loadDataGrid(DG\_NadbavShtraf, $"SELECT FORMAT([Data],'d') as 'Data',[Nadbav] ,[Vichet] FROM [BD\_Zarplata].[bd\_zarplta].[zp] Where [Sotrudnik\_idSotrudnik]= {LB\_Sotrud\_id2.SelectedItem}");

CB\_Data\_Nadbav.Items.Clear();

DB.LoadDataComboBox(CB\_Data\_Nadbav, $"SELECT [Data] FROM [BD\_Zarplata].[bd\_zarplta].[zp] Where [Sotrudnik\_idSotrudnik]= {LB\_Sotrud\_id2.SelectedItem}", 0);

}

/// <summary>

/// Вывод в форму даты , надбавки и штрафов

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void DG\_NadbavShtraf\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

try

{

Calendar2.DisplayDate = Convert.ToDateTime(DG.GetSelectCell(DG\_NadbavShtraf, 0));

Calendar2.SelectedDate = Convert.ToDateTime(DG.GetSelectCell(DG\_NadbavShtraf, 0));

TB\_Nadbav.Text = DG.GetSelectCell(DG\_NadbavShtraf, 1);

TB\_Vichet.Text = DG.GetSelectCell(DG\_NadbavShtraf, 2);

Btn\_Red\_Nadbav.IsEnabled = true;

Btn\_Save.IsEnabled = true;

}

catch

{

Calendar2.DisplayDate = DateTime.Now;

Calendar2.SelectedDate = DateTime.Now;

TB\_Nadbav.Text = "";

TB\_Vichet.Text = "";

Btn\_Red\_Nadbav.IsEnabled = false;

Btn\_Save.IsEnabled = false;

}

}

/// <summary>

/// Переход к странице расчета

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Btn\_Raschet\_Click(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new Page\_raschet());

}

/// <summary>

/// Внесение изменений в БД

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Btn\_Red\_Nadbav\_Click(object sender, RoutedEventArgs e)

{

Procedure.UpdateTable

(

"[BD\_Zarplata].[bd\_zarplta].[zp]"

, new List<string>() { "[Vichet]", "[Nadbav]" }

, new List<string>() { TB\_Vichet.Text, TB\_Nadbav.Text }

, $"where[Sotrudnik\_idSotrudnik] = {LB\_Sotrud\_id2.SelectedItem} AND [Data] = '{Calendar2.SelectedDate}'"

);

Classes.DB.loadDataGrid(DG\_NadbavShtraf, $"SELECT FORMAT([Data],'d'),[Nadbav] ,[Vichet] FROM [BD\_Zarplata].[bd\_zarplta].[zp] Where [Sotrudnik\_idSotrudnik]= {LB\_Sotrud\_id2.SelectedItem}");

MessageBox.Show("Изменения сохранены!");

}

}

}

Page\_Bughl\_AddEdit\_base.xaml.cs

using System;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace BDZarplata.Pages

{

/// <summary>

/// Логика взаимодействия для Page\_Bughl\_AddEdit\_base.xaml

/// </summary>

public partial class Page\_Bughl\_AddEdit\_base : Page

{

/// <summary>

/// Режим Добавления новых данных

/// </summary>

bool AddMode;

string YEAR;

public Page\_Bughl\_AddEdit\_base(string year = "", string znach = "")

{

if (year == "")

{

AddMode = true;

}

else

{

AddMode = false;

}

InitializeComponent();

TB\_Year.Text = year;

TB\_Znach.Text = znach;

YEAR = year;

}

/// <summary>

/// Сохранение данных

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

if (TB\_Year.Text == "")

{

MessageBox.Show("Поле год обязательно для заполнения!");

return;

}

if (TB\_Znach.Text == "")

{

MessageBox.Show("Поле значение обязательно для заполнения!");

return;

}

List<string> columnNames = new List<string>() { "[Y]", "[Z]" };

List<string> RowData = new List<string>() { TB\_Year.Text, TB\_Znach.Text };

if (AddMode)

{

Classes.Procedure.InsertTable("[BD\_Zarplata].[bd\_zarplta].[base]", columnNames, RowData);

MessageBox.Show("Вставка новой записи завершена!");

}

else

{

Classes.Procedure.UpdateTable("[BD\_Zarplata].[bd\_zarplta].[base]", columnNames, RowData, "WHERE [Y]= " + TB\_Year.Text);

MessageBox.Show("Обновление записи успешно!");

}

Classes.Manager.MainFrame.Navigate(new Pages.Page\_BughalterInfo());

}

private void Btn\_Delete\_Click(object sender, RoutedEventArgs e)

{

MessageBoxResult MR;

if (YEAR == "")

{

MR = MessageBox.Show("Объект для удаления не выбран. \n хотите выбрать его сейчас?", "Внимание!", MessageBoxButton.YesNo, MessageBoxImage.Warning);

if (MR == MessageBoxResult.Yes)

{

Classes.Manager.MainFrame.Navigate(new Page\_BughalterInfo());

}

else

{

return;

}

}

else if (MessageBoxResult.Yes == MessageBox.Show("Вы собираетесь безвозвратно удалить\n Вы уверены?", "Внимание!", MessageBoxButton.YesNo, MessageBoxImage.Warning))

{

//ТК в БД прописано каскадное удаление ,удаляем лишь из родительской таблицы

string comand = "DELETE FROM [bd\_zarplta].[base] WHERE [Y] =" + YEAR;

if (Classes.DB.queryData(comand) != -1)

{

Classes.Manager.UpdateLabel("Готово.");

MessageBox.Show("Операция успешна выполнена!", "Информация", MessageBoxButton.OK, MessageBoxImage.Information);

Classes.Manager.MainFrame.Navigate(new Page\_BughalterInfo());

}

else

{

MessageBox.Show("Произошла ошибка при удалении");

}

}

}

private void intOnly\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

//узнаем Кто вызвал событие

TextBox textBox = sender as TextBox;

//проверка что введена цифра

if (!Char.IsDigit(e.Text, 0))

{

e.Handled = true;

MessageBox.Show("Только Целые числа", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

}

//проверяем выходит ли за предел int

else if (!int.TryParse(textBox.Text + e.Text, out \_))

{

MessageBox.Show("Максимальный размер числа не может быть больше 2147483647", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

e.Handled = true;

}

}

private void shortOnly\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

//узнаем Кто вызвал событие

TextBox textBox = sender as TextBox;

//проверка что введена цифра

if (!Char.IsDigit(e.Text, 0))

{

e.Handled = true;

MessageBox.Show("Только Целые числа", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

}

//проверяем выходит ли за предел int

else if (!short.TryParse(textBox.Text + e.Text, out \_))

{

MessageBox.Show("Максимальный размер числа не может быть больше 32000", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

e.Handled = true;

}

}

}

}

Page\_BuhgaltAddEdit\_Oklad.xaml.cs

using System;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace BDZarplata.Pages

{

/// <summary>

/// Логика взаимодействия для Page\_BuhgaltAddEdit\_Oklad.xaml

/// </summary>

public partial class Page\_BuhgaltAddEdit\_Oklad : Page

{

public Page\_BuhgaltAddEdit\_Oklad(string IdDoljnost = "NOT")

{

InitializeComponent();

if (IdDoljnost != "NOT")

{

TB\_title.Text = Classes.DB.queryScalar($"Select \* FROM [BD\_Zarplata].[bd\_zarplta].[doljnost] where [idDoljnost]={IdDoljnost}", 1);

TB\_Oklad.Text = Classes.DB.queryScalar($"Select \* FROM [BD\_Zarplata].[bd\_zarplta].[doljnost] where [idDoljnost]={IdDoljnost}", 2);

TB\_Travmat.Text = Classes.DB.queryScalar($"Select \* FROM [BD\_Zarplata].[bd\_zarplta].[doljnost] where [idDoljnost]={IdDoljnost}", 3);

id = IdDoljnost;

}

}

string id;

int mrot = Convert.ToInt32(Classes.DB.queryScalar($"Select [MROT] FROM[BD\_Zarplata].[bd\_zarplta].[h]"));

/// <summary>

/// Проверка и сохранение при их корректности

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

if (Convert.ToInt32(TB\_Oklad.Text) < mrot)

{

MessageBox.Show($"Оклад не может быть меньше МРОТ! ({mrot} руб) ");

return;

}

if (Convert.ToDouble(TB\_Travmat.Text) < 0.2 || Convert.ToDouble(TB\_Travmat.Text) > 8.5)

{

MessageBox.Show("Процент травматизма должен быть в пределах от 0.2 до 8.5");

return;

}

List<string> columnname = new List<string>() { "[Oklad]", "[Travmat]" };

List<string> datacolumn = new List<string>() { TB\_Oklad.Text, TB\_Travmat.Text.Replace(',', '.') };

MessageBoxResult boxResult = MessageBox.Show("Сохранить изменения?", "Запрос на сохранение", MessageBoxButton.YesNo);

if (boxResult == MessageBoxResult.Yes)

{

Classes.Procedure.UpdateTable("[bd\_zarplta].[doljnost]", columnname, datacolumn, $"WHERE [idDoljnost]={id}");

Classes.Manager.MainFrame.Navigate(new Pages.Page\_BughalterInfo());

}

}

/// <summary>

/// Фильтр int значений для TextBox

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void intOnly\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

//узнаем Кто вызвал событие

TextBox textBox = sender as TextBox;

//проверка что введена цифра

if (!Char.IsDigit(e.Text, 0))

{

e.Handled = true;

MessageBox.Show("Только Целые числа", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

}

//проверяем выходит ли за предел int

else if (!int.TryParse(textBox.Text + e.Text, out \_))

{

MessageBox.Show("Максимальный размер числа не может быть больше 2147483647", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

e.Handled = true;

}

}

private void floatOnly\_PreviewTextInput(object sender, TextCompositionEventArgs e)

{

//узнаем Кто вызвал событие

TextBox textBox = sender as TextBox;

//проверка что в строке лишь 1 .

string fullText = textBox.Text + e.Text;

if (e.Text.Contains(",") && (textBox.Text.Contains(",") || textBox.Text.Length == 0))

{

e.Handled = true;

MessageBox.Show("неверно поставлена точка", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

}

//проверка что введена цифра или .

else if (!Char.IsDigit(e.Text, 0) && e.Text != ",")

{

e.Handled = true;

MessageBox.Show("Только дробные числа", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

}

else if (textBox.Text.Length > 2)

{

e.Handled = true;

MessageBox.Show("Длина строки не может превышать 3 символов", "Внимание", MessageBoxButton.OK, MessageBoxImage.Warning);

}

}

}

}

Page\_Login.xaml.cs

using System.Windows;

using System.Windows.Controls;

namespace BDZarplata.Pages

{

/// <summary>

/// Логика взаимодействия для Page\_Login.xaml

/// </summary>

public partial class Page\_Login : Page

{

public Page\_Login()

{

InitializeComponent();

}

/// <summary>

/// Попытка подключения к БД

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

Classes.Manager.UpdateLabel("Попытка подключения К БД");

if (Classes.DB\_Connect.OpenClouseConnection(TB\_IpPc.Text, TB\_DBName.Text))

{

Classes.Manager.UpdateLabel("Подключение прошло успешно");

object DostupLvl =

Classes.DB.queryScalar("SELECT t2.AccessLvl " +

"FROM[bd\_zarplta].[sotrudnik] " +

"t1 LEFT JOIN bd\_zarplta.doljnost t2 ON t1.idDoljnost = t2.idDoljnost " +

"where t1.idSotrudnik=" + TB\_idSotrudnik.Text).ToString();

switch (DostupLvl)

{

case "0":

MessageBox.Show("В Доступе Отказано! Обратитесь в Бухгалтерию для получения выписки");

break;

case "1":

Classes.Manager.MainFrame.Navigate(new Page\_BughalterInfo());

break;

case "2":

Classes.Manager.MainFrame.Navigate(new Pages.Page\_SotrudnikMainInfo());

break;

default:

MessageBox.Show("Неизвестный уровень доступа " + DostupLvl);

break;

}

}

else

{

Classes.Manager.UpdateLabel("Ошибка");

}

}

/// <summary>

/// Переключение поля для ввода адреса

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void CB\_IPPC\_Localhost\_Click(object sender, RoutedEventArgs e)

{

if (CB\_IPPC\_Localhost.IsChecked == true)

{

TB\_IpPc.Text = "localhost";

TB\_IpPc.IsEnabled = false;

}

else

{

TB\_IpPc.Text = "";

TB\_IpPc.IsEnabled = true;

}

}

/// <summary>

/// Переключение поля для ввода названия БД

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void CB\_BD\_NAmeDef\_Click(object sender, RoutedEventArgs e)

{

if (CB\_BD\_NAmeDef.IsChecked == true)

{

TB\_DBName.Text = "BD\_Zarplata";

TB\_DBName.IsEnabled = false;

}

else

{

TB\_DBName.Text = "";

TB\_DBName.IsEnabled = true;

}

}

}

}

Page\_raschet.xaml.cs

using BDZarplata.Classes;

using System;

using System.Windows;

using System.Windows.Controls;

using Excel = Microsoft.Office.Interop.Excel;

namespace BDZarplata.Pages

{

/// <summary>

/// Логика взаимодействия для Page\_raschet.xaml

/// </summary>

public partial class Page\_raschet : Page

{

double NDFL\_stavka, PFR\_stavka, FCC\_stavka, FOMC\_stavka, TRavmat\_stavka, Mrot\_stavka;

int Kid1\_stavka, Kid3\_stavka, Invalid\_stavka, Invalid\_o\_stavka;

int Nadbav, Vichet;

public Page\_raschet()

{

InitializeComponent();

DB.LoadDataComboBox(CB\_FIO, "SELECT [idSotrudnik] ,[full\_name] FROM[BD\_Zarplata].[bd\_zarplta].[sotrudnik]", 1);

DB.LoadDataComboBox(CB\_SotrudID, "SELECT [idSotrudnik] ,[full\_name] FROM[BD\_Zarplata].[bd\_zarplta].[sotrudnik]", 0);

int[] tempColumns = { 0, 1, 2, 3, 4, 5, 6, 7, 8 };

string[] temp = DB.queryScalar("Select \* FROM [BD\_Zarplata].[bd\_zarplta].[h]", tempColumns);

NDFL\_stavka = Convert.ToDouble(temp[0]);

PFR\_stavka = Convert.ToDouble(temp[1]);

FCC\_stavka = Convert.ToDouble(temp[2]);

FOMC\_stavka = Convert.ToDouble(temp[3]);

Kid1\_stavka = Convert.ToInt32(temp[4]);

Kid3\_stavka = Convert.ToInt32(temp[5]);

Invalid\_stavka = Convert.ToInt32(temp[6]);

Invalid\_o\_stavka = Convert.ToInt32(temp[7]);

Mrot\_stavka = Convert.ToInt32(temp[8]);

}

public double itogZarplata = 0;

/// <summary>

/// Расчет зарплаты

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void BTN\_Raschet\_Click(object sender, RoutedEventArgs e)

{

DateTime StrartdateTime;

//ищем ближайшуюю предыдущую выплату ЗП

var StartVar = DB.queryScalar($"SELECT [Data] " +

$"FROM[BD\_Zarplata].[bd\_zarplta].[zp] " +

$"Where[Sotrudnik\_idSotrudnik] = {CB\_SotrudID.SelectedItem} AND[Data] < '{CB\_Date.SelectedItem}' " +

$"order by Data desc");

//Если не находим начинаем считать зарплату сначала месяца

if (StartVar == "null")

{

StrartdateTime = calendar.DisplayDate;

StrartdateTime = StrartdateTime.AddDays(-StrartdateTime.Day + 1);

}

//иначе считаем с него

else { StrartdateTime = Convert.ToDateTime(StartVar); }

//считаем количество рабочих дней

int KolvoRabDney = Convert.ToInt32(DB.queryScalar($"SELECT COUNT(\*) FROM[BD\_Zarplata].[bd\_zarplta].[graphik\_rabot] " +

$"where[Sotrudnik\_idSotrudnik] = {CB\_SotrudID.SelectedItem} " +

$"AND DATE >= '{StrartdateTime}' " +

$"AND DATE < '{CB\_Date.SelectedItem}' " +

$"AND[StatusDay] = 'рабочий' "));

double DnevDohod = Convert.ToInt32(L\_Oklad.Content) / KolvoRabDney;

double OkladMes = 0;

for (DateTime i = StrartdateTime; i < calendar.DisplayDate; i = i.AddDays(1))

{

string statusDay = DB.queryScalar($"SELECT [StatusDay] FROM[BD\_Zarplata].[bd\_zarplta].[graphik\_rabot]" +

$" where[Sotrudnik\_idSotrudnik] = {CB\_SotrudID.SelectedItem} AND DATE = '{i}'").ToString();

string statusSotrud = DB.queryScalar($"SELECT [StatusSotrud] FROM[BD\_Zarplata].[bd\_zarplta].[graphik\_rabot]" +

$" where[Sotrudnik\_idSotrudnik] = {CB\_SotrudID.SelectedItem} AND DATE = '{i}'").ToString();

Manager.UpdateLabel(i.ToString());

//начисление зп в обычный день

if (statusDay.LastIndexOf("рабочий") != -1 && statusSotrud.LastIndexOf("вышел") != -1)

{

OkladMes += DnevDohod;

}

//начисление ЗП за выход выходной

else if (statusDay.LastIndexOf("выходной") != -1 && statusSotrud.LastIndexOf("вышел") != -1)

{

OkladMes += DnevDohod \* 2;

}

else

{

//расчет больничного

}

}

Vichet = Convert.ToInt32(DB.queryScalar($"SELECT [Vichet] FROM[BD\_Zarplata].[bd\_zarplta].[zp] where[Data] = '{CB\_Date.SelectedItem}' AND [Sotrudnik\_idSotrudnik] = {CB\_SotrudID.SelectedItem}"));

Nadbav = Convert.ToInt32(DB.queryScalar($"SELECT [Nadbav] FROM[BD\_Zarplata].[bd\_zarplta].[zp] where[Data] = '{CB\_Date.SelectedItem}' AND [Sotrudnik\_idSotrudnik] = {CB\_SotrudID.SelectedItem}"));

//сумма зарплаты работника до вычета налогов

double FactDohod = OkladMes + Nadbav - Vichet;

double nalogBase = FactDohod;

int CountZdKids = Convert.ToInt32(DB.queryScalar($"SELECT [num\_zd\_kids] FROM [BD\_Zarplata].[bd\_zarplta].[sotrudnik] WHEre [idSotrudnik]={CB\_SotrudID.SelectedItem}"));

int CountInvalidKids = Convert.ToInt32(DB.queryScalar($"SELECT [num\_invalid\_kids] FROM [BD\_Zarplata].[bd\_zarplta].[sotrudnik] WHEre [idSotrudnik]={CB\_SotrudID.SelectedItem}"));

int CountInvalidOpekaKids = Convert.ToInt32(DB.queryScalar($"SELECT [opeka] FROM [BD\_Zarplata].[bd\_zarplta].[sotrudnik] WHEre [idSotrudnik]={CB\_SotrudID.SelectedItem}"));

int SpecStatus = Convert.ToInt32(DB.queryScalar($"SELECT [SpecStatus] FROM [BD\_Zarplata].[bd\_zarplta].[sotrudnik] WHEre [idSotrudnik]={CB\_SotrudID.SelectedItem}"));

if (SpecStatus == 1)

{

nalogBase = nalogBase - 3000;

}

else if (SpecStatus == 2)

{

nalogBase = nalogBase - 500;

}

if (CountZdKids <= 2)

{

nalogBase = nalogBase - (Kid1\_stavka \* CountZdKids);

}

else if (CountZdKids > 2)

{

nalogBase = nalogBase - (Kid1\_stavka \* 2 + Kid3\_stavka \* (CountZdKids - 2));

}

nalogBase = nalogBase - CountInvalidKids \* Invalid\_stavka;

nalogBase = nalogBase - CountInvalidOpekaKids \* Invalid\_o\_stavka;

//Ндфл которое платит работник

double NdFl = (nalogBase / 100 \* NDFL\_stavka);

L\_Ndfl.Content = NdFl;

//Итоговая зарплата на руки работнику

itogZarplata = FactDohod - NdFl;

//налоги уплачиваемые работодателем

L\_FCC.Content = (FactDohod / 100 \* FCC\_stavka);

L\_FOMC.Content = (FactDohod / 100 \* FOMC\_stavka);

L\_pfr.Content = (FactDohod / 100 \* PFR\_stavka);

L\_Travmatizm.Content = (FactDohod / 100 \* TRavmat\_stavka);

BTN\_Export.IsEnabled = true;

}

/// <summary>

/// Экспорт в Эксель

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void BTN\_Export\_Click(object sender, RoutedEventArgs e)

{

Excel.Application ex = new Microsoft.Office.Interop.Excel.Application();

ex.SheetsInNewWorkbook = 1;

Excel.Workbook workBook = ex.Workbooks.Add(Type.Missing);

ex.DisplayAlerts = false;

Excel.Worksheet sheet = (Excel.Worksheet)ex.Worksheets.get\_Item(1);

// sheet.Range["A1", "K1"].Merge(); - объединение ячеек вы заданном диапазоне

// sheet.Cells.Range["A1", "A7"].Font.Bold = true; включение жирного текста

//sheet.Cells[1, 1] = "String" присвоить значение ячейки

//sheet.Cells.WrapText = true; перенос текста в ячейках

// sheet.Range["B13", "B13"].BorderAround2(Excel.XlLineStyle.xlContinuous, Excel.XlBorderWeight.xlThin, Excel.XlColorIndex.xlColorIndexAutomatic);

//рамка для ячейки

sheet.Range["A1", "Q2"].VerticalAlignment = Excel.XlVAlign.xlVAlignCenter;

sheet.Range["A1", "B2"].HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;

sheet.Cells.Font.Name = "Times New Roman"; //задание шрифта

sheet.Cells.Font.Size = "12";

sheet.Columns[1].ColumnWidth = 30;

sheet.Columns[6].ColumnWidth = 30;

sheet.Cells.Range["A1", "F4"].Font.Bold = true;

sheet.Cells[1, 1] = "Расчетный листок за " + Convert.ToDateTime(CB\_Date.SelectedItem).ToShortDateString();

sheet.Cells[2, 1] = "ООО";

sheet.Cells[3, 1] = "Работник: " + CB\_FIO.SelectedItem;

sheet.Cells[4, 1] = "Табельный номер: " + CB\_SotrudID.SelectedItem;

sheet.Cells[4, 6] = "Должность: " + L\_DoljnostTitle.Content;

sheet.Cells.Range["A5", "H6"].Font.Italic = true;

sheet.Cells.Range["A5", "H5"].BorderAround2(Excel.XlLineStyle.xlContinuous, Excel.XlBorderWeight.xlMedium, Excel.XlColorIndex.xlColorIndexAutomatic);

sheet.Cells[5, 1] = "Вид";

sheet.Cells[5, 2] = "Период";

sheet.Cells[5, 3] = "Дни";

sheet.Cells[5, 4] = "Часы";

sheet.Cells[5, 5] = "Сумма";

sheet.Cells[5, 6] = "Вид";

sheet.Cells[5, 7] = "Период";

sheet.Cells[5, 8] = "Сумма";

sheet.Range["A5", "H15"].BorderAround2(Excel.XlLineStyle.xlContinuous, Excel.XlBorderWeight.xlMedium, Excel.XlColorIndex.xlColorIndexAutomatic);

sheet.Range["A1", "H1"].Merge();

sheet.Range["A2", "H2"].Merge();

sheet.Range["A3", "E3"].Merge();

sheet.Range["F4", "H4"].Merge();

sheet.Cells[6, 1] = "1. Начислено";

sheet.Cells[7, 1] = "Оклад";

sheet.Cells[7, 5] = L\_Oklad.Content;

sheet.Cells[8, 1] = "Дополнительные выплаты";

sheet.Cells[8, 5] = Nadbav;

sheet.Cells[9, 1] = "Больничные пособия";

sheet.Cells[11, 1] = "Всего начислено:";

sheet.Cells[11, 5] = "=СУММ(E7:E10)";

sheet.Cells[12, 1] = "3. Взносы в ПФР";

sheet.Cells[13, 1] = "Страховые взносы в ПФР (страховая часть 22%)";

sheet.Cells[13, 5] = L\_pfr.Content;

sheet.Cells[6, 6] = "2.Удержано";

sheet.Cells[7, 6] = "НДФЛ по ставке 13%";

sheet.Cells[7, 8] = L\_Ndfl.Content;

sheet.Cells[8, 6] = "Иные удержания";

sheet.Cells[8, 8] = Vichet;

sheet.Cells[11, 6] = "Всего удержано:";

sheet.Cells[11, 8] = "=СУММ(H7:H10)";

sheet.Cells[12, 6] = "Сумма к выплате";

sheet.Cells[12, 8] = itogZarplata;

sheet.Cells[13, 6] = "Зачислено на счёт№" + DB.queryScalar("SELECT [SchetZachisl] FROM [BD\_Zarplata].[bd\_zarplta].[sotrudnik] WHERE" + CB\_SotrudID.SelectedItem);

sheet.Cells[14, 6] = "Выдано наличными";

sheet.Cells[14, 8] = itogZarplata;

sheet.Cells.WrapText = true;

ex.Visible = true;

}

/// <summary>

/// сопоставление ФИО и ID сотрудника

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void CB\_SotrudID\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

CB\_FIO.SelectedIndex = CB\_SotrudID.SelectedIndex;

}

/// <summary>

/// подгрузка информации о сотруднике

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void CB\_FIO\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

BTN\_Export.IsEnabled = false;

BTN\_Raschet.IsEnabled = false;

CB\_Date.Items.Clear();

CB\_SotrudID.SelectedIndex = CB\_FIO.SelectedIndex;

DB.LoadDataComboBox(CB\_Date, "SELECT [Data] FROM [BD\_Zarplata].[bd\_zarplta].[zp] wHERE [Sotrudnik\_idSotrudnik]= " + CB\_SotrudID.SelectedItem, 0);

L\_IdDoljnost.Content = DB.queryScalar("SELECT [idDoljnost],[full\_name] FROM[BD\_Zarplata].[bd\_zarplta].[sotrudnik] WHERE [idSotrudnik]=" + CB\_SotrudID.SelectedItem);

L\_DoljnostTitle.Content = DB.queryScalar("SELECT [title] FROM[BD\_Zarplata].[bd\_zarplta].[doljnost] WHERE [idDoljnost]=" + L\_IdDoljnost.Content);

L\_Oklad.Content = DB.queryScalar("SELECT [Oklad] FROM [BD\_Zarplata].[bd\_zarplta].[doljnost] where [idDoljnost] =" + L\_IdDoljnost.Content);

TRavmat\_stavka = Convert.ToDouble(DB.queryScalar("SELECT [Travmat] FROM [BD\_Zarplata].[bd\_zarplta].[doljnost] where [idDoljnost] =" + L\_IdDoljnost.Content));

}

/// <summary>

/// Изменение даты на календаре

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void CB\_Date\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

calendar.DisplayDate = Convert.ToDateTime(CB\_Date.SelectedItem);

calendar.SelectedDate = Convert.ToDateTime(CB\_Date.SelectedItem);

BTN\_Raschet.IsEnabled = true;

}

}

}

Page\_SotrudnikMainInfo.xaml.cs

using BDZarplata.Classes;

using System;

using System.Collections.Generic;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Input;

namespace BDZarplata.Pages

{

/// <summary>

/// Логика взаимодействия для Page\_SotrudnikMainInfo.xaml

/// </summary>

public partial class Page\_SotrudnikMainInfo : Page

{

public Page\_SotrudnikMainInfo()

{

InitializeComponent();

Classes.DB.loadDataGrid(DG\_Sotrud\_Anketa, "SELECT [idSotrudnik], full\_name , t2.title, family\_status, num\_zd\_kids,num\_invalid\_kids, opeka, SpecStatus, Staj FROM[bd\_zarplta].[sotrudnik] t1 LEFT JOIN bd\_zarplta.doljnost t2 ON t1.idDoljnost = t2.idDoljnost");

Classes.DB.LoadDataListBox(LB\_Sotrud\_FIO, "SELECT [idSotrudnik],[full\_name] FROM [BD\_Zarplata].[bd\_zarplta].[sotrudnik]", 1);

Classes.DB.LoadDataListBox(LB\_Sotrud\_id, "SELECT [idSotrudnik] ,[full\_name] FROM [BD\_Zarplata].[bd\_zarplta].[sotrudnik]", 0);

DB.LoadDataComboBox(CB\_StatusDay, "SELECT DISTINCT [StatusDay] FROM[BD\_Zarplata].[bd\_zarplta].[graphik\_rabot]", 0);

DB.LoadDataComboBox(CB\_StatusSotrud, "SELECT DISTINCT [StatusSotrud] FROM[BD\_Zarplata].[bd\_zarplta].[graphik\_rabot]", 0);

}

/// <summary>

/// Переход в режим редактирования

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Btn\_Redactir\_Click(object sender, RoutedEventArgs e)

{

string sotrudid = Classes.DG.GetSelectCell(DG\_Sotrud\_Anketa, 0);

switch (TabC\_Main.SelectedIndex)

{

case 0:

Manager.MainFrame.Navigate(new Page\_AddRedAnketa(sotrudid));

break;

}

}

/// <summary>

/// сохранение измененных полей таблицы БД

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Btn\_Save\_Click(object sender, RoutedEventArgs e)

{

}

/// <summary>

/// Сопоставление ФИО и ID сотрудника

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void LB\_Sotrud\_FIO\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

LB\_Sotrud\_id.SelectedIndex = LB\_Sotrud\_FIO.SelectedIndex;

}

/// <summary>

/// Подгрузка данных о Расписании выбранного сотрудника

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void LB\_Sotrud\_id\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

Classes.DB.loadDataGrid(DG\_Raspisnie, $"SELECT FORMAT([DATE],'d') as 'Date' ,[StatusSotrud] ,[StatusDay] FROM [BD\_Zarplata].[bd\_zarplta].[graphik\_rabot] where [Sotrudnik\_idSotrudnik] = {LB\_Sotrud\_id.SelectedItem}");

}

/// <summary>

/// Переход в режим редактирования

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

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

{

Btn\_Redactir\_Click(sender, e);

}

/// <summary>

/// Переход в режим добавления нового сотрудника

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void Btn\_Add\_Click(object sender, RoutedEventArgs e)

{

Manager.MainFrame.Navigate(new Page\_AddRedAnketa());

}

/// <summary>

/// Подгрузка данных о выбранном дне

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void DG\_Raspisnie\_SelectionChanged(object sender, SelectionChangedEventArgs e)

{

try

{

calendar\_raspisan.DisplayDate = Convert.ToDateTime(DG.GetSelectCell(DG\_Raspisnie, 0));

calendar\_raspisan.SelectedDate = Convert.ToDateTime(DG.GetSelectCell(DG\_Raspisnie, 0));

CB\_StatusDay.SelectedItem = DG.GetSelectCell(DG\_Raspisnie, 2);

CB\_StatusSotrud.SelectedItem = DG.GetSelectCell(DG\_Raspisnie, 1);

BTN\_RedRaspisan.IsEnabled = true;

}

catch

{

calendar\_raspisan.DisplayDate = DateTime.Now;

calendar\_raspisan.SelectedDate = DateTime.Now;

CB\_StatusDay.SelectedIndex = 1;

CB\_StatusSotrud.SelectedIndex = 1;

BTN\_RedRaspisan.IsEnabled = false;

}

}

/// <summary>

/// сохранение измененных полей таблицы БД

/// </summary>

/// <param name="sender"></param>

/// <param name="e"></param>

private void BTN\_RedRaspisan\_Click(object sender, RoutedEventArgs e)

{

Procedure.UpdateTable

(

"[BD\_Zarplata].[bd\_zarplta].[graphik\_rabot]"

, new List<string>() { "[StatusSotrud]", "[StatusDay]" }

, new List<string>() { "'" + CB\_StatusSotrud.SelectedItem + "'", "'" + CB\_StatusDay.SelectedItem + "'" }

, $"where[Sotrudnik\_idSotrudnik] = {LB\_Sotrud\_id.SelectedItem} AND[DATE] = '{calendar\_raspisan.SelectedDate}'"

);

Classes.DB.loadDataGrid(DG\_Raspisnie, $"SELECT FORMAT([DATE],'d') ,[StatusSotrud] ,[StatusDay] FROM [BD\_Zarplata].[bd\_zarplta].[graphik\_rabot] where [Sotrudnik\_idSotrudnik] = {LB\_Sotrud\_id.SelectedItem}");

MessageBox.Show("Успешно Обновленно!");

}

private void TabI\_MainData\_GotFocus(object sender, RoutedEventArgs e)

{

Btn\_Add.Visibility = Visibility.Visible;

}

private void TabItem\_GotFocus(object sender, RoutedEventArgs e)

{

Btn\_Add.Visibility = Visibility.Hidden;

}

}

}