ПРИЛОЖЕНИЕ А

(ИНФОРМАЦИОННОЕ)

Листинг

using System;

using System.Windows.Forms;

namespace test\_DataBase

{

internal static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

private static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new LogIn());

}

}

}

using System.Data.SqlClient;

namespace test\_DataBase

{

internal class DataBase

{

private readonly SqlConnection sqlConnection = new SqlConnection(@"Data Source=MS-SQL\SQLExpress;Initial Catalog=TourismCompanyDB;Integrated Security=True");

/// <summary>

/// OpenConnection вызывается при открытии соединения с базой данных

/// </summary>

public void OpenConnection()

{

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

{

sqlConnection.Open();

}

}

/// <summary>

/// CloseConnection вызывается при закрытии соединения с базой данных

/// </summary>

public void CloseConnection()

{

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

{

sqlConnection.Close();

}

}

/// <summary>

/// GetConnection вызывается при получении соединения с базой данных

/// </summary>

/// <returns></returns>

public SqlConnection GetConnection()

{

return sqlConnection;

}

}

}

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace test\_DataBase

{

public partial class SignUp : Form

{

private readonly DataBase dataBase = new DataBase();

public SignUp()

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

}

/// <summary>

/// SignUp\_Load вызывается при загрузке формы "SignUp"

/// </summary>

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

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

private void SignUp\_Load(object sender, EventArgs e)

{

textBoxPassword.PasswordChar = '•';

textBoxLogin.MaxLength = 50;

textBoxPassword.MaxLength = 50;

}

/// <summary>

/// ButtonCreate\_Click вызывается при нажатии на кнопку "Войти"

/// </summary>

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

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

private void ButtonCreate\_Click(object sender, EventArgs e)

{

var login = textBoxLogin.Text;

var password = textBoxPassword.Text;

string querystring = $"Insert into Registration(UserLogin, UserPassword, IsAdmin) values('{login}', '{password}', 0)";

SqlCommand sqlCommand = new SqlCommand(querystring, dataBase.GetConnection());

dataBase.OpenConnection();

if (sqlCommand.ExecuteNonQuery() == 1)

{

MessageBox.Show("Аккаунт успешно создан!", "Успех!");

LogIn formLogin = new LogIn();

this.Hide();

formLogin.ShowDialog();

}

else

{

MessageBox.Show("Аккаунт не создан!");

}

dataBase.CloseConnection();

}

/// <summary>

/// ButtonClear\_Click вызывается при нажатии на кнопку очистки

/// </summary>

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

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

private void ButtonClear\_Click(object sender, EventArgs e)

{

textBoxLogin.Text = "";

textBoxPassword.Text = "";

}

}

}

using System;

using System.Data;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace test\_DataBase

{

public partial class LogIn : Form

{

private readonly DataBase dataBase = new DataBase();

public LogIn()

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

}

/// <summary>

/// Form1\_Load вызывается при загрузке формы "Form1"

/// </summary>

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

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

private void Form1\_Load(object sender, EventArgs e)

{

textBoxPassword.PasswordChar = '•';

textBoxLogin.MaxLength = 50;

textBoxPassword.MaxLength = 50;

}

/// <summary>

/// ButtonEnter\_Click вызывается при нажатии на кнопку "Войти"

/// </summary>

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

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

private void ButtonEnter\_Click(object sender, EventArgs e)

{

var loginUser = textBoxLogin.Text;

var passwordUser = textBoxPassword.Text;

SqlDataAdapter sqlDataAdapter = new SqlDataAdapter();

DataTable dataTable = new DataTable();

string querystring = $"select UserID, UserLogin, UserPassword, IsAdmin from Registration where UserLogin = '{loginUser}' and UserPassword = '{passwordUser}'";

SqlCommand sqlCommand = new SqlCommand(querystring, dataBase.GetConnection());

sqlDataAdapter.SelectCommand = sqlCommand;

sqlDataAdapter.Fill(dataTable);

if (dataTable.Rows.Count == 1)

{

bool isAdmin = Convert.ToBoolean(dataTable.Rows[0]["IsAdmin"]);

MessageBox.Show("Вы успешно вошли!", "Успешно!", MessageBoxButtons.OK, MessageBoxIcon.Information);

Form1 form1 = new Form1();

form1.SetAdminStatus(isAdmin);

this.Hide();

form1.ShowDialog();

this.Show();

}

else

{

MessageBox.Show("Такого аккаунта не существует!", "Аккаунта не существует!", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

/// <summary>

/// ButtonClear\_Click вызывается при нажатии на кнопку очистки

/// </summary>

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

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

private void ButtonClear\_Click(object sender, EventArgs e)

{

textBoxLogin.Text = "";

textBoxPassword.Text = "";

}

/// <summary>

/// LabelAuth\_Click вызывается при нажатии на текст "Ещё нет аккаунта?"

/// </summary>

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

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

private void LabelAuth\_Click(object sender, EventArgs e)

{

this.Hide();

SignUp formLogin = new SignUp();

formLogin.ShowDialog();

}

}

}

using iText.IO.Font;

using iText.Kernel.Font;

using iText.Kernel.Pdf;

using iText.Layout.Properties;

using Microsoft.Office.Interop.Word;

using System;

using System.Data;

using System.Data.SqlClient;

using System.IO;

using System.Linq;

using System.Windows.Forms;

using Excel = Microsoft.Office.Interop.Excel;

namespace test\_DataBase

{

internal enum RowState

{

Existed,

New,

Modified,

ModifiedNew,

Deleted

}

public partial class Form1 : Form

{

private readonly DataBase dataBase = new DataBase();

private bool admin;

private int selectedRow;

public Form1()

{

try

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// SetAdminStatus проверяет доступ

/// </summary>

/// <param name="isAdmin"></param>

public void SetAdminStatus(bool isAdmin)

{

admin = isAdmin;

}

/// <summary>

/// CreateColumns вызывается при создании колонок

/// </summary>

private void CreateColumns()

{

try

{

dataGridViewClients.Columns.Add("ClientID", "Номер");

dataGridViewClients.Columns.Add("FirstName", "Имя");

dataGridViewClients.Columns.Add("LastName", "Фамилия");

dataGridViewClients.Columns.Add("Email", "Email");

dataGridViewClients.Columns.Add("Phone", "Телефон");

dataGridViewClients.Columns.Add("IsNew", String.Empty);

dataGridViewTours.Columns.Add("TourID", "Номер");

dataGridViewTours.Columns.Add("TourName", "Название тура");

dataGridViewTours.Columns.Add("Destination", "Назначение");

dataGridViewTours.Columns.Add("StartDate", "Дата начала");

dataGridViewTours.Columns.Add("EndDate", "Дата конца");

dataGridViewTours.Columns.Add("Price", "Цена");

dataGridViewTours.Columns.Add("IsNew", String.Empty);

dataGridViewBookings.Columns.Add("BookingID", "Номер");

dataGridViewBookings.Columns.Add("ClientID", "Номер клиента");

dataGridViewBookings.Columns.Add("TourID", "Номер тура");

dataGridViewBookings.Columns.Add("BookingDate", "Дата бронирования");

dataGridViewBookings.Columns.Add("NumberOfPersons", "Количество человек");

dataGridViewBookings.Columns.Add("TotalAmount", "Общая сумма");

dataGridViewBookings.Columns.Add("IsNew", String.Empty);

dataGridViewPayments.Columns.Add("PaymentID", "Номер");

dataGridViewPayments.Columns.Add("BookingID", "Номер бронирования");

dataGridViewPayments.Columns.Add("PaymentDate", "Дата платежа");

dataGridViewPayments.Columns.Add("Amount", "Сумма");

dataGridViewPayments.Columns.Add("IsNew", String.Empty);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// CreateColumns вызывается при очистке полей

/// </summary>

private void ClearFields()

{

try

{

textBoxClientID.Text = "";

textBoxFirstName.Text = "";

textBoxLastName.Text = "";

textBoxEmail.Text = "";

textBoxPhone.Text = "";

textBoxTourID.Text = "";

textBoxTourName.Text = "";

textBoxDestination.Text = "";

textBoxStartDate.Text = "";

textBoxEndDate.Text = "";

textBoxPrice.Text = "";

textBoxBookingID.Text = "";

textBoxClientIDBookings.Text = "";

textBoxTourIDBookings.Text = "";

textBoxBookingDate.Text = "";

textBoxNumberOfPersons.Text = "";

textBoxTotalAmount.Text = "";

textBoxPaymentID.Text = "";

textBoxBookingIDPayments.Text = "";

textBoxPaymentDate.Text = "";

textBoxAmount.Text = "";

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// ReadSingleRow вызывается при чтении строк

/// </summary>

/// <param name="dataGridView"></param>

/// <param name="iDataRecord"></param>

private void ReadSingleRow(DataGridView dataGridView, IDataRecord iDataRecord)

{

try

{

switch (dataGridView.Name)

{

case "dataGridViewClients":

dataGridView.Rows.Add(iDataRecord.GetInt32(0), iDataRecord.GetString(1), iDataRecord.GetString(2), iDataRecord.GetString(3), iDataRecord.GetString(4), RowState.Modified);

break;

case "dataGridViewTours":

dataGridView.Rows.Add(iDataRecord.GetInt32(0), iDataRecord.GetString(1), iDataRecord.GetString(2), iDataRecord.GetDateTime(3), iDataRecord.GetDateTime(4), iDataRecord.GetInt32(5), RowState.Modified);

break;

case "dataGridViewBookings":

dataGridView.Rows.Add(iDataRecord.GetInt32(0), iDataRecord.GetInt32(1), iDataRecord.GetInt32(2), iDataRecord.GetDateTime(3), iDataRecord.GetInt32(4), iDataRecord.GetInt32(5), RowState.Modified);

break;

case "dataGridViewPayments":

dataGridView.Rows.Add(iDataRecord.GetInt32(0), iDataRecord.GetInt32(1), iDataRecord.GetDateTime(2), iDataRecord.GetInt32(3), RowState.Modified);

break;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// RefreshDataGrid вызывается при обновлении dataGridView

/// </summary>

/// <param name="dataGridView"></param>

/// <param name="tableName"></param>

private void RefreshDataGrid(DataGridView dataGridView, string tableName)

{

try

{

dataGridView.Rows.Clear();

string queryString = $"select \* from {tableName}";

SqlCommand sqlCommand = new SqlCommand(queryString, dataBase.GetConnection());

dataBase.OpenConnection();

SqlDataReader sqlDataReader = sqlCommand.ExecuteReader();

while (sqlDataReader.Read())

{

ReadSingleRow(dataGridView, sqlDataReader);

}

sqlDataReader.Close();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Form1\_Load вызывается при загрузке формы "Form1"

/// </summary>

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

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

private void Form1\_Load(object sender, EventArgs e)

{

try

{

CreateColumns();

RefreshDataGrid(dataGridViewClients, "Clients");

RefreshDataGrid(dataGridViewTours, "Tours");

RefreshDataGrid(dataGridViewBookings, "Bookings");

RefreshDataGrid(dataGridViewPayments, "Payments");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// DataGridView\_CellClick вызывается при нажатии на ячейку в DataGridView

/// </summary>

/// <param name="dataGridView"></param>

/// <param name="selectedRow"></param>

private void DataGridView\_CellClick(DataGridView dataGridView, int selectedRow)

{

try

{

DataGridViewRow dataGridViewRow = dataGridView.Rows[selectedRow];

switch (dataGridView.Name)

{

case "dataGridViewClients":

textBoxClientID.Text = dataGridViewRow.Cells[0].Value.ToString();

textBoxFirstName.Text = dataGridViewRow.Cells[1].Value.ToString();

textBoxLastName.Text = dataGridViewRow.Cells[2].Value.ToString();

textBoxEmail.Text = dataGridViewRow.Cells[3].Value.ToString();

textBoxPhone.Text = dataGridViewRow.Cells[4].Value.ToString();

break;

case "dataGridViewTours":

textBoxTourID.Text = dataGridViewRow.Cells[0].Value.ToString();

textBoxTourName.Text = dataGridViewRow.Cells[1].Value.ToString();

textBoxDestination.Text = dataGridViewRow.Cells[2].Value.ToString();

textBoxStartDate.Text = dataGridViewRow.Cells[3].Value.ToString();

textBoxEndDate.Text = dataGridViewRow.Cells[4].Value.ToString();

textBoxPrice.Text = dataGridViewRow.Cells[5].Value.ToString();

break;

case "dataGridViewBookings":

textBoxBookingID.Text = dataGridViewRow.Cells[0].Value.ToString();

textBoxClientIDBookings.Text = dataGridViewRow.Cells[1].Value.ToString();

textBoxTourIDBookings.Text = dataGridViewRow.Cells[2].Value.ToString();

textBoxBookingDate.Text = dataGridViewRow.Cells[3].Value.ToString();

textBoxNumberOfPersons.Text = dataGridViewRow.Cells[4].Value.ToString();

textBoxTotalAmount.Text = dataGridViewRow.Cells[5].Value.ToString();

break;

case "dataGridViewPayments":

textBoxPaymentID.Text = dataGridViewRow.Cells[0].Value.ToString();

textBoxBookingIDPayments.Text = dataGridViewRow.Cells[1].Value.ToString();

textBoxPaymentDate.Text = dataGridViewRow.Cells[2].Value.ToString();

textBoxAmount.Text = dataGridViewRow.Cells[3].Value.ToString();

break;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// Search вызывается при поиске данных в DataGridView

/// </summary>

/// <param name="dataGridView"></param>

private void Search(DataGridView dataGridView)

{

try

{

dataGridView.Rows.Clear();

switch (dataGridView.Name)

{

case "dataGridViewClients":

string searchStringClients = $"select \* from Clients where concat (ClientID, FirstName, LastName, Email, Phone) like '%" + textBoxSearchClients.Text + "%'";

SqlCommand sqlCommandClients = new SqlCommand(searchStringClients, dataBase.GetConnection());

dataBase.OpenConnection();

SqlDataReader sqlDataReaderClients = sqlCommandClients.ExecuteReader();

while (sqlDataReaderClients.Read())

{

ReadSingleRow(dataGridView, sqlDataReaderClients);

}

sqlDataReaderClients.Close();

break;

case "dataGridViewTours":

string searchStringTours = $"select \* from Tours where concat (TourID, TourName, Destination, StartDate, EndDate, Price) like '%" + textBoxSearchTours.Text + "%'";

SqlCommand sqlCommandTours = new SqlCommand(searchStringTours, dataBase.GetConnection());

dataBase.OpenConnection();

SqlDataReader sqlDataReaderTours = sqlCommandTours.ExecuteReader();

while (sqlDataReaderTours.Read())

{

ReadSingleRow(dataGridView, sqlDataReaderTours);

}

sqlDataReaderTours.Close();

break;

case "dataGridViewBookings":

string searchStringBookings = $"select \* from Bookings where concat (BookingID, ClientID, TourID, BookingDate, NumberOfPersons, TotalAmount) like '%" + textBoxSearchBookings.Text + "%'";

SqlCommand sqlCommandBookings = new SqlCommand(searchStringBookings, dataBase.GetConnection());

dataBase.OpenConnection();

SqlDataReader sqlDataReaderBookings = sqlCommandBookings.ExecuteReader();

while (sqlDataReaderBookings.Read())

{

ReadSingleRow(dataGridView, sqlDataReaderBookings);

}

sqlDataReaderBookings.Close();

break;

case "dataGridViewPayments":

string searchStringPayments = $"select \* from Payments where concat (PaymentID, BookingID, PaymentDate, Amount) like '%" + textBoxSearchPayments.Text + "%'";

SqlCommand sqlCommandPayments = new SqlCommand(searchStringPayments, dataBase.GetConnection());

dataBase.OpenConnection();

SqlDataReader sqlDataReaderPayments = sqlCommandPayments.ExecuteReader();

while (sqlDataReaderPayments.Read())

{

ReadSingleRow(dataGridView, sqlDataReaderPayments);

}

sqlDataReaderPayments.Close();

break;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// DeleteRow вызывается при удалении строки

/// </summary>

/// <param name="dataGridView"></param>

private void DeleteRow(DataGridView dataGridView)

{

try

{

int index = dataGridView.CurrentCell.RowIndex;

dataGridView.Rows[index].Visible = false;

switch (dataGridView.Name)

{

case "dataGridViewClients":

if (dataGridView.Rows[index].Cells[0].Value.ToString() == string.Empty)

{

dataGridView.Rows[index].Cells[5].Value = RowState.Deleted;

return;

}

dataGridView.Rows[index].Cells[5].Value = RowState.Deleted;

break;

case "dataGridViewTours":

if (dataGridView.Rows[index].Cells[0].Value.ToString() == string.Empty)

{

dataGridView.Rows[index].Cells[6].Value = RowState.Deleted;

return;

}

dataGridView.Rows[index].Cells[6].Value = RowState.Deleted;

break;

case "dataGridViewBookings":

if (dataGridView.Rows[index].Cells[0].Value.ToString() == string.Empty)

{

dataGridView.Rows[index].Cells[6].Value = RowState.Deleted;

return;

}

dataGridView.Rows[index].Cells[6].Value = RowState.Deleted;

break;

case "dataGridViewPayments":

if (dataGridView.Rows[index].Cells[0].Value.ToString() == string.Empty)

{

dataGridView.Rows[index].Cells[4].Value = RowState.Deleted;

return;

}

dataGridView.Rows[index].Cells[4].Value = RowState.Deleted;

break;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// UpdateBase вызывается при обновлении базы данных

/// </summary>

/// <param name="dataGridView"></param>

private void UpdateBase(DataGridView dataGridView)

{

try

{

dataBase.OpenConnection();

for (int index = 0; index < dataGridView.Rows.Count; index++)

{

switch (dataGridView.Name)

{

case "dataGridViewClients":

var rowStateClients = (RowState)dataGridView.Rows[index].Cells[5].Value;

if (rowStateClients == RowState.Existed)

{

continue;

}

if (rowStateClients == RowState.Deleted)

{

var clientID = Convert.ToInt32(dataGridView.Rows[index].Cells[0].Value);

var deleteQuery = $"delete from Clients where ClientID = '{clientID}'";

var sqlCommand = new SqlCommand(deleteQuery, dataBase.GetConnection());

sqlCommand.ExecuteNonQuery();

}

if (rowStateClients == RowState.Modified)

{

var clientID = dataGridView.Rows[index].Cells[0].Value.ToString();

var firstName = dataGridView.Rows[index].Cells[1].Value.ToString();

var lastName = dataGridView.Rows[index].Cells[2].Value.ToString();

var email = dataGridView.Rows[index].Cells[3].Value.ToString();

var phone = dataGridView.Rows[index].Cells[4].Value.ToString();

var changeQuery = $"update Clients set FirstName = '{firstName}', LastName = '{lastName}', Email = '{email}', Phone = '{phone}' where ClientID = '{clientID}'";

var sqlCommand = new SqlCommand(changeQuery, dataBase.GetConnection());

sqlCommand.ExecuteNonQuery();

}

break;

case "dataGridViewTours":

var rowStateTours = (RowState)dataGridView.Rows[index].Cells[6].Value;

if (rowStateTours == RowState.Existed)

{

continue;

}

if (rowStateTours == RowState.Deleted)

{

var tourID = Convert.ToInt32(dataGridView.Rows[index].Cells[0].Value);

var deleteQuery = $"delete from Tours where TourID = '{tourID}'";

var sqlCommand = new SqlCommand(deleteQuery, dataBase.GetConnection());

sqlCommand.ExecuteNonQuery();

}

if (rowStateTours == RowState.Modified)

{

var tourID = dataGridView.Rows[index].Cells[0].Value.ToString();

var tourName = dataGridView.Rows[index].Cells[1].Value.ToString();

var destination = dataGridView.Rows[index].Cells[2].Value.ToString();

var startDate = dataGridView.Rows[index].Cells[3].Value.ToString();

var endDate = dataGridView.Rows[index].Cells[4].Value.ToString();

var price = dataGridView.Rows[index].Cells[5].Value.ToString();

var changeQuery = $"update Tours set TourName = '{tourName}', Destination = '{destination}', StartDate = '{startDate}', EndDate = '{endDate}', Price = '{price}' where TourID = '{tourID}'";

var sqlCommand = new SqlCommand(changeQuery, dataBase.GetConnection());

sqlCommand.ExecuteNonQuery();

}

break;

case "dataGridViewBookings":

var rowStateBookings = (RowState)dataGridView.Rows[index].Cells[6].Value;

if (rowStateBookings == RowState.Existed)

{

continue;

}

if (rowStateBookings == RowState.Deleted)

{

var bookingID = Convert.ToInt32(dataGridView.Rows[index].Cells[0].Value);

var deleteQuery = $"delete from Bookings where BookingID = '{bookingID}'";

var sqlCommand = new SqlCommand(deleteQuery, dataBase.GetConnection());

sqlCommand.ExecuteNonQuery();

}

if (rowStateBookings == RowState.Modified)

{

var bookingID = dataGridView.Rows[index].Cells[0].Value.ToString();

var clientID = dataGridView.Rows[index].Cells[1].Value.ToString();

var tourID = dataGridView.Rows[index].Cells[2].Value.ToString();

var bookingDate = dataGridView.Rows[index].Cells[3].Value.ToString();

var numberOfPersons = dataGridView.Rows[index].Cells[4].Value.ToString();

var totalAmount = dataGridView.Rows[index].Cells[5].Value.ToString();

var changeQuery = $"update Bookings set ClientID = '{clientID}', TourID = '{tourID}', BookingDate = '{bookingDate}', NumberOfPersons = '{numberOfPersons}', TotalAmount = '{totalAmount}' where BookingID = '{bookingID}'";

var sqlCommand = new SqlCommand(changeQuery, dataBase.GetConnection());

sqlCommand.ExecuteNonQuery();

}

break;

case "dataGridViewPayments":

var rowStatePayments = (RowState)dataGridView.Rows[index].Cells[4].Value;

if (rowStatePayments == RowState.Existed)

{

continue;

}

if (rowStatePayments == RowState.Deleted)

{

var paymentID = Convert.ToInt32(dataGridView.Rows[index].Cells[0].Value);

var deleteQuery = $"delete from Payments where PaymentID = '{paymentID}'";

var sqlCommand = new SqlCommand(deleteQuery, dataBase.GetConnection());

sqlCommand.ExecuteNonQuery();

}

if (rowStatePayments == RowState.Modified)

{

var paymentID = dataGridView.Rows[index].Cells[0].Value.ToString();

var bookingID = dataGridView.Rows[index].Cells[1].Value.ToString();

var paymentDate = dataGridView.Rows[index].Cells[2].Value.ToString();

var amount = dataGridView.Rows[index].Cells[3].Value.ToString();

var changeQuery = $"update Payments set BookingID = '{bookingID}', PaymentDate = '{paymentDate}', Amount = '{amount}' where PaymentID = '{paymentID}'";

var sqlCommand = new SqlCommand(changeQuery, dataBase.GetConnection());

sqlCommand.ExecuteNonQuery();

}

break;

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

dataBase.CloseConnection();

}

}

/// <summary>

/// Change вызывается при изменении данных в базе данных

/// </summary>

/// <param name="dataGridView"></param>

private void Change(DataGridView dataGridView)

{

try

{

var selectedRowIndex = dataGridView.CurrentCell.RowIndex;

switch (dataGridView.Name)

{

case "dataGridViewClients":

var clientID = textBoxClientID.Text;

var firstName = textBoxFirstName.Text;

var lastName = textBoxLastName.Text;

var email = textBoxEmail.Text;

var phone = textBoxPhone.Text;

dataGridView.Rows[selectedRowIndex].SetValues(clientID, firstName, lastName, email, phone);

dataGridView.Rows[selectedRowIndex].Cells[5].Value = RowState.Modified;

break;

case "dataGridViewTours":

var tourID = textBoxTourID.Text;

var tourName = textBoxTourName.Text;

var destination = textBoxDestination.Text;

var startDate = textBoxStartDate.Value;

var endDate = textBoxEndDate.Value;

var price = textBoxPrice.Text;

dataGridView.Rows[selectedRowIndex].SetValues(tourID, tourName, destination, startDate, endDate, price);

dataGridView.Rows[selectedRowIndex].Cells[6].Value = RowState.Modified;

break;

case "dataGridViewBookings":

var bookingID = textBoxBookingID.Text;

var clientIDBookings = textBoxClientIDBookings.Text;

var tourIDBookings = textBoxTourIDBookings.Text;

var bookingDate = textBoxBookingDate.Value;

var numberOfPersons = textBoxNumberOfPersons.Text;

var totalAmount = textBoxTotalAmount.Text;

dataGridView.Rows[selectedRowIndex].SetValues(bookingID, clientIDBookings, tourIDBookings, bookingDate, numberOfPersons, totalAmount);

dataGridView.Rows[selectedRowIndex].Cells[6].Value = RowState.Modified;

break;

case "dataGridViewPayments":

var paymentID = textBoxPaymentID.Text;

var bookingIDPayments = textBoxBookingIDPayments.Text;

var paymentDate = textBoxPaymentDate.Value;

var amount = textBoxAmount.Text;

dataGridView.Rows[selectedRowIndex].SetValues(paymentID, bookingIDPayments, paymentDate, amount);

dataGridView.Rows[selectedRowIndex].Cells[4].Value = RowState.Modified;

break;

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// ExportToWord вызывается при экспорте данных в Word

/// </summary>

/// <param name="dataGridView"></param>

private void ExportToWord(DataGridView dataGridView)

{

try

{

var wordApp = new Microsoft.Office.Interop.Word.Application();

wordApp.Visible = true;

Microsoft.Office.Interop.Word.Document doc = wordApp.Documents.Add();

Paragraph title = doc.Paragraphs.Add();

switch (dataGridView.Name)

{

case "dataGridViewClients":

title.Range.Text = "Данные клиентов";

break;

case "dataGridViewTours":

title.Range.Text = "Данные туров";

break;

case "dataGridViewBookings":

title.Range.Text = "Данные бронирований";

break;

case "dataGridViewPayments":

title.Range.Text = "Данные выплат";

break;

}

title.Range.Font.Bold = 1;

title.Range.Font.Size = 14;

title.Alignment = WdParagraphAlignment.wdAlignParagraphCenter;

title.Range.InsertParagraphAfter();

Table table = doc.Tables.Add(title.Range, dataGridView.RowCount + 1, dataGridView.ColumnCount - 1);

for (int col = 0; col < dataGridView.ColumnCount - 1; col++)

{

table.Cell(1, col + 1).Range.Text = dataGridView.Columns[col].HeaderText;

}

for (int row = 0; row < dataGridView.RowCount; row++)

{

for (int col = 0; col < dataGridView.ColumnCount - 1; col++)

{

table.Cell(row + 2, col + 1).Range.Text = dataGridView[col, row].Value.ToString();

}

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// ExportToExcel вызывается при экспорте данных в Excel

/// </summary>

/// <param name="dataGridView"></param>

private void ExportToExcel(DataGridView dataGridView)

{

try

{

var excelApp = new Excel.Application();

excelApp.Visible = true;

Excel.Workbook workbook = excelApp.Workbooks.Add();

Excel.Worksheet worksheet = (Excel.Worksheet)workbook.Worksheets[1];

string title = "";

switch (dataGridView.Name)

{

case "dataGridViewClients":

title = "Данные клиентов";

break;

case "dataGridViewTours":

title = "Данные туров";

break;

case "dataGridViewBookings":

title = "Данные бронирований";

break;

case "dataGridViewPayments":

title = "Данные выплат";

break;

}

Excel.Range titleRange = worksheet.Range[worksheet.Cells[1, 1], worksheet.Cells[1, dataGridView.ColumnCount - 1]];

titleRange.Merge();

titleRange.Value = title;

titleRange.Font.Bold = true;

titleRange.Font.Size = 14;

titleRange.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;

for (int col = 0; col < dataGridView.ColumnCount; col++)

{

worksheet.Cells[2, col + 1] = dataGridView.Columns[col].HeaderText;

}

for (int row = 0; row < dataGridView.RowCount; row++)

{

for (int col = 0; col < dataGridView.ColumnCount - 1; col++)

{

worksheet.Cells[row + 3, col + 1] = dataGridView[col, row].Value.ToString();

Excel.Range dataRange = worksheet.Range[worksheet.Cells[2, 1], worksheet.Cells[dataGridView.RowCount + 2, dataGridView.ColumnCount]];

dataRange.HorizontalAlignment = Excel.XlHAlign.xlHAlignCenter;

dataRange.VerticalAlignment = Excel.XlVAlign.xlVAlignCenter;

}

}

worksheet.Columns.AutoFit();

worksheet.Rows.AutoFit();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// ExportToPDF вызывается при экспорте данных в PDF

/// </summary>

/// <param name="dataGridView"></param>

private void ExportToPDF(DataGridView dataGridView)

{

try

{

string filePath = Path.Combine(AppDomain.CurrentDomain.BaseDirectory, "output.pdf");

var pdfWriter = new PdfWriter(filePath);

var pdfDocument = new PdfDocument(pdfWriter);

var pdfDoc = new iText.Layout.Document(pdfDocument);

PdfFont timesFont = PdfFontFactory.CreateFont("c:/windows/fonts/times.ttf", PdfEncodings.IDENTITY\_H, true);

string title = "";

switch (dataGridView.Name)

{

case "dataGridViewClients":

title = "Данные клиентов";

break;

case "dataGridViewTours":

title = "Данные туров";

break;

case "dataGridViewBookings":

title = "Данные бронирований";

break;

case "dataGridViewPayments":

title = "Данные выплат";

break;

}

pdfDoc.Add(new iText.Layout.Element.Paragraph(title).SetFont(timesFont).SetTextAlignment(TextAlignment.CENTER));

iText.Layout.Element.Table table = new iText.Layout.Element.Table(dataGridView.Columns.Count - 1);

table.UseAllAvailableWidth();

var columnsList = dataGridView.Columns.Cast<DataGridViewColumn>().ToList();

foreach (DataGridViewColumn column in columnsList.Take(dataGridView.Columns.Count - 1))

{

iText.Layout.Element.Cell headerCell = new iText.Layout.Element.Cell().Add(new iText.Layout.Element.Paragraph(column.HeaderText).SetFont(timesFont));

table.AddHeaderCell(headerCell);

}

foreach (DataGridViewRow row in dataGridView.Rows)

{

foreach (DataGridViewCell cell in row.Cells.Cast<DataGridViewCell>().Take(dataGridView.Columns.Count - 1))

{

table.AddCell(new iText.Layout.Element.Cell().Add(new iText.Layout.Element.Paragraph(cell.Value.ToString()).SetFont(timesFont)));

}

}

pdfDoc.Add(table);

pdfDoc.Close();

MessageBox.Show("PDF успешно экспортирован.");

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// ButtonRefresh\_Click вызывается при нажатии на кнопку обновления

/// </summary>

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

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

private void ButtonRefresh\_Click(object sender, EventArgs e)

{

try

{

RefreshDataGrid(dataGridViewClients, "Clients");

RefreshDataGrid(dataGridViewTours, "Tours");

RefreshDataGrid(dataGridViewBookings, "Bookings");

RefreshDataGrid(dataGridViewPayments, "Payments");

ClearFields();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

/// <summary>

/// ButtonClear\_Click вызывается при нажатии на кнопку "Изменить"

/// </summary>

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

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

private void ButtonClear\_Click(object sender, EventArgs e)

{

try

{

ClearFields();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonNewClients\_Click(object sender, EventArgs e)

{

try

{

AddFormClients addForm = new AddFormClients();

addForm.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonNewTours\_Click(object sender, EventArgs e)

{

try

{

if (admin)

{

AddFormTours addForm = new AddFormTours();

addForm.Show();

}

else

{

MessageBox.Show("У вас недостаточно прав!");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonNewBookings\_Click(object sender, EventArgs e)

{

try

{

AddFormBookings addForm = new AddFormBookings();

addForm.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonNewPayments\_Click(object sender, EventArgs e)

{

try

{

AddFormPayments addForm = new AddFormPayments();

addForm.Show();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonDeleteClients\_Click(object sender, EventArgs e)

{

try

{

DeleteRow(dataGridViewClients);

ClearFields();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonDeleteTours\_Click(object sender, EventArgs e)

{

try

{

DeleteRow(dataGridViewTours);

ClearFields();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonDeleteBookings\_Click(object sender, EventArgs e)

{

try

{

DeleteRow(dataGridViewBookings);

ClearFields();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonDeletePayments\_Click(object sender, EventArgs e)

{

try

{

DeleteRow(dataGridViewPayments);

ClearFields();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonChangeClients\_Click(object sender, EventArgs e)

{

try

{

Change(dataGridViewClients);

ClearFields();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonChangeTours\_Click(object sender, EventArgs e)

{

try

{

Change(dataGridViewTours);

ClearFields();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonChangeBookings\_Click(object sender, EventArgs e)

{

try

{

Change(dataGridViewBookings);

ClearFields();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonChangePayments\_Click(object sender, EventArgs e)

{

try

{

Change(dataGridViewPayments);

ClearFields();

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonSaveClients\_Click(object sender, EventArgs e)

{

try

{

UpdateBase(dataGridViewClients);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonSaveTours\_Click(object sender, EventArgs e)

{

try

{

if (admin)

{

UpdateBase(dataGridViewTours);

}

else

{

MessageBox.Show("У вас недостаточно прав!");

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonSaveBookings\_Click(object sender, EventArgs e)

{

try

{

UpdateBase(dataGridViewBookings);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonSavePayments\_Click(object sender, EventArgs e)

{

try

{

UpdateBase(dataGridViewPayments);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonWordClients\_Click(object sender, EventArgs e)

{

try

{

ExportToWord(dataGridViewClients);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonWordTours\_Click(object sender, EventArgs e)

{

try

{

ExportToWord(dataGridViewTours);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonWordBookings\_Click(object sender, EventArgs e)

{

try

{

ExportToWord(dataGridViewBookings);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonWordPayments\_Click(object sender, EventArgs e)

{

try

{

ExportToWord(dataGridViewPayments);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonExcelClients\_Click(object sender, EventArgs e)

{

try

{

ExportToExcel(dataGridViewClients);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonExcelTours\_Click(object sender, EventArgs e)

{

try

{

ExportToExcel(dataGridViewTours);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonExcelBookings\_Click(object sender, EventArgs e)

{

try

{

ExportToExcel(dataGridViewBookings);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonExcelPayments\_Click(object sender, EventArgs e)

{

try

{

ExportToExcel(dataGridViewPayments);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonPDFClients\_Click(object sender, EventArgs e)

{

try

{

ExportToPDF(dataGridViewClients);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonPDFTours\_Click(object sender, EventArgs e)

{

try

{

ExportToPDF(dataGridViewTours);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonPDFBookings\_Click(object sender, EventArgs e)

{

try

{

ExportToPDF(dataGridViewBookings);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void ButtonPDFPayments\_Click(object sender, EventArgs e)

{

try

{

ExportToPDF(dataGridViewPayments);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void DataGridViewClients\_CellClick(object sender, DataGridViewCellEventArgs e)

{

try

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridView\_CellClick(dataGridViewClients, selectedRow);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void DataGridViewTours\_CellClick(object sender, DataGridViewCellEventArgs e)

{

try

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridView\_CellClick(dataGridViewTours, selectedRow);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void DataGridViewBookings\_CellClick(object sender, DataGridViewCellEventArgs e)

{

try

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridView\_CellClick(dataGridViewBookings, selectedRow);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void DataGridViewPayments\_CellClick(object sender, DataGridViewCellEventArgs e)

{

try

{

selectedRow = e.RowIndex;

if (e.RowIndex >= 0)

{

DataGridView\_CellClick(dataGridViewPayments, selectedRow);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void TextBoxSearchClients\_TextChanged(object sender, EventArgs e)

{

try

{

Search(dataGridViewClients);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void TextBoxSearchTours\_TextChanged(object sender, EventArgs e)

{

try

{

Search(dataGridViewTours);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void TextBoxSearchBookings\_TextChanged(object sender, EventArgs e)

{

try

{

Search(dataGridViewBookings);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

private void TextBoxSearchPayments\_TextChanged(object sender, EventArgs e)

{

try

{

Search(dataGridViewPayments);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

}

}

}

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace test\_DataBase

{

public partial class AddFormTours : Form

{

private readonly DataBase dataBase = new DataBase();

public AddFormTours()

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

}

/// <summary>

/// ButtonSave\_Click вызывается при нажатии на кнопку "Сохранить"

/// </summary>

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

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

private void ButtonSave\_Click(object sender, EventArgs e)

{

try

{

dataBase.OpenConnection();

var tourName = textBoxTourName.Text;

var destination = textBoxDestination.Text;

var startDate = textBoxStartDate.Value;

var endDate = textBoxEndDate.Value;

var price = textBoxPrice.Text;

var addQuery = $"insert into Tours (TourName, Destination, StartDate, EndDate, Price) values ('{tourName}', '{destination}', '{startDate}', '{endDate}', '{price}')";

var sqlCommand = new SqlCommand(addQuery, dataBase.GetConnection());

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Запись успешно создана!", "Успех!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

dataBase.CloseConnection();

}

}

}

}

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace test\_DataBase

{

public partial class AddFormPayments : Form

{

private readonly DataBase dataBase = new DataBase();

public AddFormPayments()

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

}

/// <summary>

/// ButtonSave\_Click вызывается при нажатии на кнопку "Сохранить"

/// </summary>

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

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

private void ButtonSave\_Click(object sender, EventArgs e)

{

try

{

if (int.TryParse(textBoxBookingIDPayments.Text, out int bookingID))

{

dataBase.OpenConnection();

var paymentDate = textBoxPaymentDate.Value;

var amount = textBoxAmount.Text;

var addQuery = $"insert into Payments (BookingID, PaymentDate, Amount) values ('{bookingID}', '{paymentDate}', '{amount}')";

var sqlCommand = new SqlCommand(addQuery, dataBase.GetConnection());

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Запись успешно создана!", "Успех!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Цена должна иметь числовой формат!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

dataBase.CloseConnection();

}

}

}

}

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace test\_DataBase

{

public partial class AddFormClients : Form

{

private readonly DataBase dataBase = new DataBase();

public AddFormClients()

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

}

/// <summary>

/// ButtonSave\_Click вызывается при нажатии на кнопку "Сохранить"

/// </summary>

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

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

private void ButtonSave\_Click(object sender, EventArgs e)

{

try

{

dataBase.OpenConnection();

var firstName = textBoxFirstName.Text;

var lastName = textBoxLastName.Text;

var email = textBoxEmail.Text;

var phone = textBoxPhone.Text;

var addQuery = $"insert into Clients (FirstName, LastName, Email, Phone) values ('{firstName}', '{lastName}', '{email}', '{phone}')";

var sqlCommand = new SqlCommand(addQuery, dataBase.GetConnection());

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Запись успешно создана!", "Успех!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

dataBase.CloseConnection();

}

}

}

}

using System;

using System.Data.SqlClient;

using System.Windows.Forms;

namespace test\_DataBase

{

public partial class AddFormBookings : Form

{

private readonly DataBase dataBase = new DataBase();

public AddFormBookings()

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

}

/// <summary>

/// ButtonSave\_Click вызывается при нажатии на кнопку "Сохранить"

/// </summary>

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

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

private void ButtonSave\_Click(object sender, EventArgs e)

{

try

{

if (int.TryParse(textBoxClientIDBookings.Text, out int clientID) && int.TryParse(textBoxTourIDBookings.Text, out int tourID) && int.TryParse(textBoxNumberOfPersons.Text, out int numberOfPersons))

{

dataBase.OpenConnection();

var bookingDate = textBoxBookingDate.Value;

var totalAmount = textBoxTotalAmount.Text;

var addQuery = $"insert into Bookings (ClientID, TourID, BookingDate, NumberOfPersons, TotalAmount) values ('{clientID}', '{tourID}', '{bookingDate}', '{numberOfPersons}', '{totalAmount}')";

var sqlCommand = new SqlCommand(addQuery, dataBase.GetConnection());

sqlCommand.ExecuteNonQuery();

MessageBox.Show("Запись успешно создана!", "Успех!", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

else

{

MessageBox.Show("Цена должна иметь числовой формат!", "Ошибка!", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

catch (Exception ex)

{

MessageBox.Show(ex.Message);

}

finally

{

dataBase.CloseConnection();

}

}

}

}