

МИНОБРНАУКИ РОССИИ

федеральное государственное бюджетное образовательное учреждение высшего образования

«Санкт-Петербургский государственный технологический институт (технический университет)»

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| УГНС | | 09.00.00 | Информатика и вычислительная техника | | |
| Направление подготовки | | 09.03.01 | Информатика и вычислительная техника | | |
| Направленность (профиль) | |  | Системы автоматизированного проектирования | | |
| Форма обучения | |  | очная | | |
|  | |  |  | | |
| Факультет | |  | Информационных технологий и управления | | |
| Кафедра | |  | Систем автоматизированного проектирования и управления | | |
| Учебная дисциплина | |  | Разработка программных систем | | |
| Курс | II | | | Группа | 423 |

Отчёт по контрольной работе №4

Вариант № 5

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Исполнитель: |  |  |  |  |
| обучающийся группы 423 |  |  |  | Ефремов Иван Андреевич |
|  |  | (дата, подпись) |  |  |
|  |  |  |  |  |
| Проверили: |  |  |  | Корниенко Иван Григорьевич |
|  |  | (дата, подпись) |  | Макарук Роман Валерьевич |
|  |  |  |  | Федин Алексей Константинович |

СОДЕРЖАНИЕ

[1 Постановка задачи 3](#_Toc134647499)

[2 Исходные данные 3](#_Toc134647500)

[3 Особые ситуации 3](#_Toc134647501)

[4 Математические методы и алгоритмы решения задач 4](#_Toc134647502)

[5 Форматы представления данных 4](#_Toc134647503)

[6 Структура программы 5](#_Toc134647504)

[7 Блок-схема алгоритма решения задачи 9](#_Toc134647505)

[8 Описание хода выполнения 13](#_Toc134647506)

[9 Результаты работы программы 14](#_Toc134647507)

[10 Выводы по заданию 16](#_Toc134647508)

[11 Исходный код полученного программного решения 16](#_Toc134647509)

## 1 Постановка задачи

Необходимо написать приложение с использованием технологии WinForms реализующие вариант задания. Программа должна позволять добавлять новые сущности с использованием интерфейса и редактировать существующие. Сущности, добавленные в программу, должны сохраняться между запусками приложения. Для хранения данных необходимо использовать СУБД SqLite. Необходимо предусмотреть возможность сохранения списка существующих сущностей в файл.

Тематика базы данных: хранение списка должников

## 2 Исходные данные

Исходные данные состоят из базы данных списка должников. Пользователь может добавлять новые сущности и редактировать уже существующие.

Поля таблицы базы данных: id, имя, фамилия, сумма займа, дата займа, крайняя дата возврата, статус.

## 3 Особые ситуации

Необходимо рассмотреть следующие особые ситуации:

* программа не дает пользователю вводить никакие символы, кроме цифр, в поля, запрашивающие числа
* программа не дает пользователю вводить никакие символы, кроме букв, в поля, запрашивающие строки
* программа не дает пользователю изменить сущности, с несуществующим в БД id

## 4 Математические методы и алгоритмы решения задач

Согласно постановке задачи, для получения необходимых результатов будут использоваться операторы “>”, “≥”, “<”, “≤”, “=”, “+”.

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 5 Форматы представления данных Таблица 1 - Основные переменные, используемые в программе   |  |  |  | | --- | --- | --- | | Имя | Тип | Описание | | values | DatabaseValues[] | Массив сущностей базы данных | | database | DatabaseManipulation | Объект взаимодействия с базой данных |    6 Структура программы Таблица 2 - Модули программы   |  |  | | --- | --- | | Название модуля | Описание | | Program | Запуск основной формы приложения | | MenuForm | Логика и структура основной формы, являющейся родительской формой, для остальных | | AddForm | Логика и структура формы, для добавления новой сущности в БД | | EditForm | Логика и структура формы, для редактирования существующей сущности в БД | | DatabaseManipulation | Класс для взаимодействия с БД | |  |

Таблица 3 – Основные функции модуля MenuForm

|  |  |
| --- | --- |
| Имя | Описание |
| OpenDebtListForm | Открытие формы со списком сущностей базы данных |
| OpenEditForm | Открытие формы с редактированием существующей сущности |
| menuTransition\_Tick | Запуск анимации открытия бокового меню |

Таблица 4 – Функции модуля AddForm

|  |  |
| --- | --- |
| Имя | Описание |
| addBtn\_Click | Обработка нажатия кнопки и добавление новой сущности в БД |
| ClearFields | Очистка полей формы |
| amountTextBox\_TextChanged | Обработка ввода суммы |

Таблица 5 – Функции модуля EditForm

|  |  |
| --- | --- |
| Имя | Описание |
| LoadDatabaseRecord | Заполнение полей данными соответствующей сущности |
| CheckIfThereAnyChanges | Проверка наличия изменений в данных сущности |
| applyBtn\_Click | Обработка нажатия кнопки и применение изменений в данных сущности |
| deleteBtn\_Click | Обработка нажатия кнопки и удаление сущности из БД |

Таблица 6 – Функции модуля DatabaseManipulations

|  |  |
| --- | --- |
| Имя | Описание |
| ReadTable | Считывание данных сущности и запись их в массив |
| UpdateValue | Изменение данных сущности |
| DeleteValue | Удаление сущности |
| InsertValue | Добавление новой сущности |

## 7 Блок-схема алгоритма решения задачи

На рисунке 1 представлена блок-схема работы программы.



Рисунок 1 – Блок-схема работы программы

## 8 Описание хода выполнения

При запуске программы открывается форма со списком сущностей базы данных и боковым меню. Нажимая на кнопки бокового меню, пользователь может перемещаться между формами добавления и редактирования сущностей. В форме списка пользователь может сохранить сущности из базы данных в формате json. При двойном нажатии на одно из полей сущности открывается форма редактирования сущности. В форме редактирования сущности пользователь может изменить поля, удалить сущность или ничего не делать. При нажатии кнопки «Apply changes» программа изменит сущность в базе данных, после чего откроется форма со списком.

## 9 Результаты работы программы

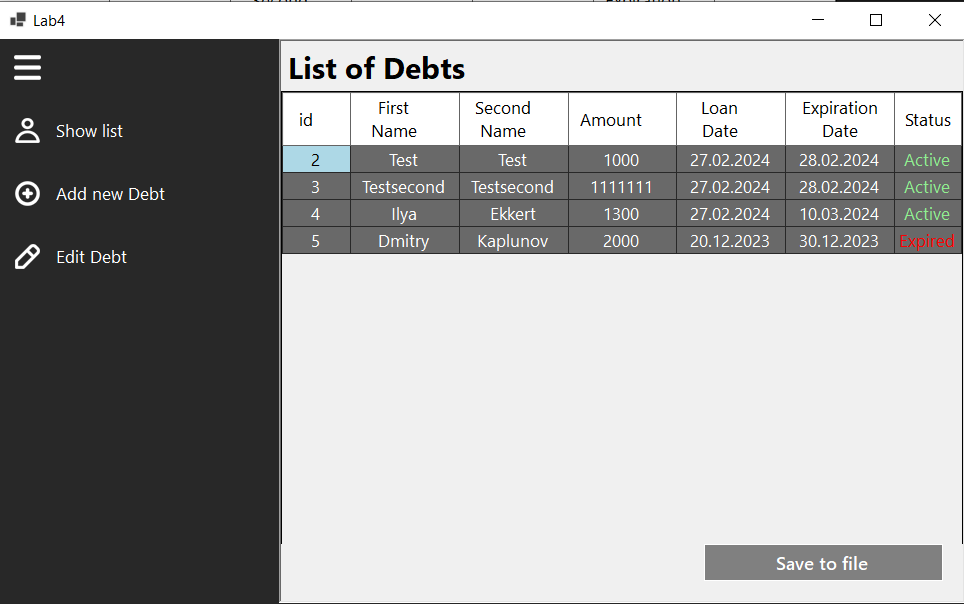


Рисунок 2 – Экранная копия окна программы сразу после запуска

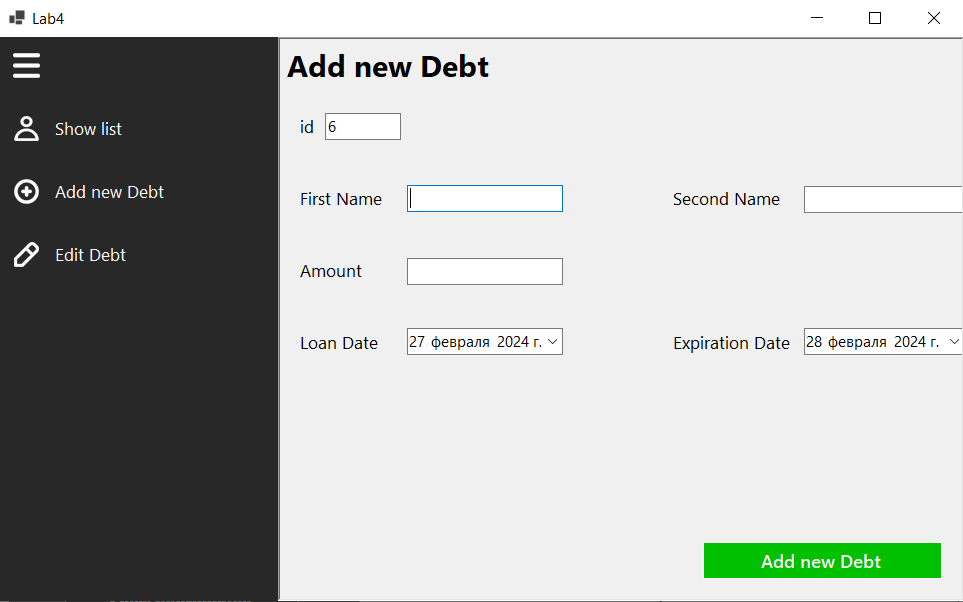


Рисунок 3 – Экранная копия окна программы после открытия формы добавления новой сущности

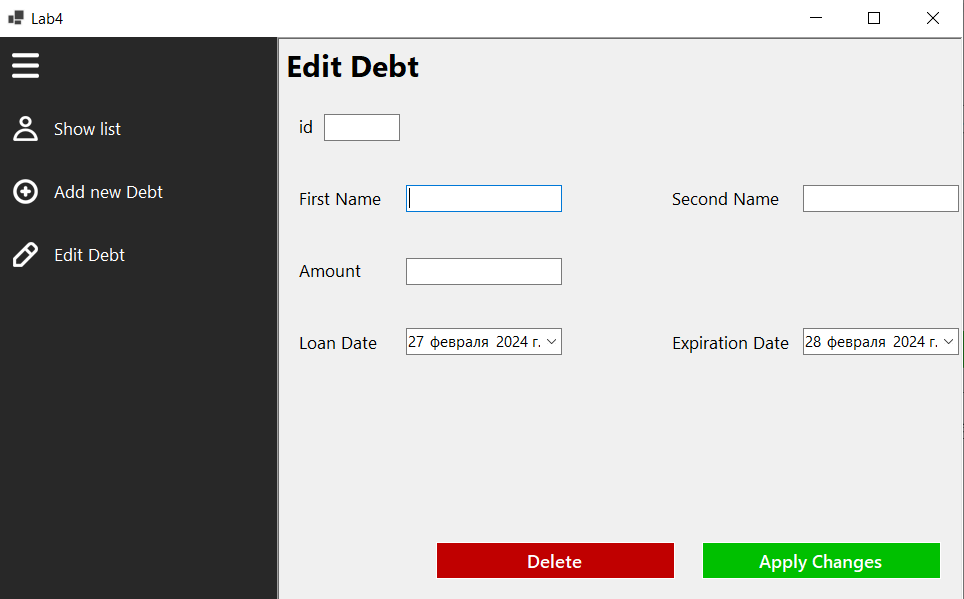


Рисунок 4 – Экранная копия окна программы после открытия формы изменения сущности

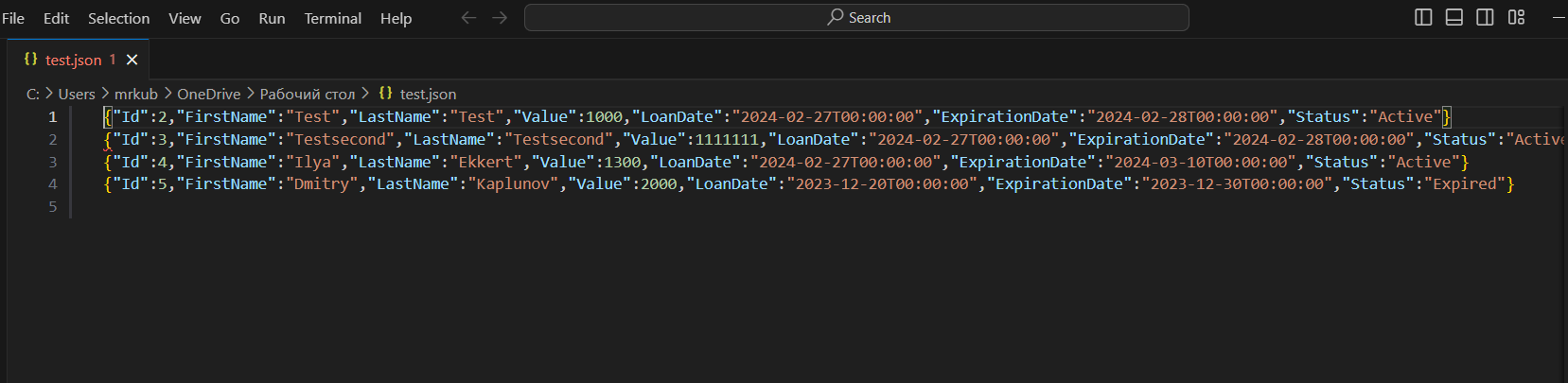


Рисунок 4 – Экранная копия файла с сохраненными сущностями

## 10 Выводы по заданию

В ходе выполнения задания были изучены основы работы с языком С# и работы с системой WinForms. Были получены навыки создания приложений с полноценным графическим интерфейсом, а также навыки взаимодействия с базой данных SQLite.

## 11 Исходный код полученного программного решения

Файл Program.cs:

namespace Lab4

{

internal static class Program

{

/// <summary>

/// The main entry point for the application.

/// </summary>

[STAThread]

static void Main()

{

// To customize application configuration such as set high DPI settings or default font,

// see https://aka.ms/applicationconfiguration.

ApplicationConfiguration.Initialize();

Application.Run(new DBForm());

}

}

}

Файл DatabaseManipulations.cs:

using System;

using System.CodeDom;

using System.Collections.Generic;

using System.Data;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using Microsoft.Data.Sqlite;

namespace Lab4

{

public class DatabaseValues

{

private int id;

public int Id

{

get { return id; }

set { id = value; }

}

private string firstName = "";

public string FirstName

{

get { return firstName; }

set { firstName = value; }

}

private string lastName = "";

public string LastName

{

get { return lastName; }

set { lastName = value; }

}

private int val;

public int Value

{

get { return val; }

set { val = value; }

}

private DateTime loanDate;

public DateTime LoanDate

{

get { return loanDate; }

set { loanDate = value; }

}

private DateTime expirationDate;

public DateTime ExpirationDate

{

get { return expirationDate; }

set { expirationDate = value; }

}

private string status = "";

public string Status

{

get { return status; }

set { status = value; }

}

public DatabaseValues

(int id, string firstName, string lastName, int value,

string loanDate, string expirationDate, string status)

{

Id = id;

FirstName = firstName;

LastName = lastName;

Value = value;

LoanDate = DateTime.Parse(loanDate);

ExpirationDate = DateTime.Parse(expirationDate);

Status = status;

}

public override bool Equals(object? obj)

{

if (obj == null || GetType() != obj.GetType())

{

return false;

}

DatabaseValues other = (DatabaseValues)obj;

return (Id == other.Id &&

FirstName == other.FirstName &&

LastName == other.LastName &&

Value == other.Value &&

LoanDate == other.LoanDate &&

ExpirationDate == other.ExpirationDate &&

Status == other.Status);

}

public override int GetHashCode()

{

return base.GetHashCode() ^ Id;

}

}

public class DatabaseManipulations

{

private SqliteCommand command;

SqliteConnection connection;

public DatabaseManipulations(string path)

{

connection = new SqliteConnection($"Data source={path};Mode=ReadWrite"); ;

connection.Open();

command = new SqliteCommand("",connection);

}

public int GetRecordCount()

{

command.CommandText = "SELECT COUNT(\*) FROM Debts";

return Convert.ToInt32(command.ExecuteScalar());

}

public DatabaseValues[] ReadTable()

{

int size = GetRecordCount();

DatabaseValues[] values = new DatabaseValues[size];

command.CommandText = "SELECT \* FROM Debts";

SqliteDataReader reader = command.ExecuteReader();

int i = 0;

while(reader.Read())

{

int id = Convert.ToInt32(reader.GetValue(0));

string firstName = reader.GetString(1);

string lastName = reader.GetString(2);

int value = Convert.ToInt32(reader.GetValue(3));

string loanDate = reader.GetString(4);

string expirationDate = reader.GetString(5);

string status = reader.GetString(6);

values[i] = new DatabaseValues(id, firstName, lastName, value, loanDate, expirationDate, status);

i++;

}

reader.Close();

return values;

}

public void CloseConnection()

{

connection.Close();

}

public void UpdateValue(DatabaseValues value)

{

command.CommandText = $"UPDATE Debts SET " +

$"firstName = \"{value.FirstName}\", " +

$"secondName = \"{value.LastName}\", " +

$"amount = \"{value.Value}\", " +

$"loanDate = \"{value.LoanDate:dd/MM/yyyy}\", " +

$"expirationDate = \"{value.ExpirationDate:dd/MM/yyyy}\"," +

$"status = \"{value.Status}\"" +

$"WHERE id = \"{value.Id}\"";

command.ExecuteNonQuery();

}

public void UpdateStatus(int id, string stauts)

{

command.CommandText = $"UPDATE Debts SET status = \"{stauts}\" WHERE id = \"{id}\"";

command.ExecuteNonQuery();

}

public void DeleteValue(DatabaseValues value)

{

command.CommandText = $"DELETE FROM Debts WHERE id = {value.Id}";

command.ExecuteNonQuery();

}

public void InsertValue(DatabaseValues value)

{

command.CommandText = $"INSERT INTO Debts (" +

$"firstName, " +

$"secondName, " +

$"amount, " +

$"loanDate, " +

$"expirationDate, " +

$"status) " +

$"VALUES (" +

$"\"{value.FirstName}\", " +

$"\"{value.LastName}\", " +

$"\"{value.Value}\", " +

$"\"{value.LoanDate:dd/MM/yyyy}\", " +

$"\"{value.ExpirationDate:dd/MM/yyyy}\", " +

$"\"{value.Status}\")";

command.ExecuteNonQuery();

}

}

}

Файл MenuForm.cs:

namespace Lab4

{

public partial class DBForm : Form

{

const string dbPath = "C:\\SemesterFourth\\SemesterFourRPS\\Lab4\\bin\\Debug\\net8.0-windows\\debtDatabase.db";

public static DatabaseManipulations database = new DatabaseManipulations(dbPath);

AddForm? addDebt;

EditForm? editDebt;

ListForm? debtList;

public DBForm()

{

InitializeComponent();

}

bool isMenuOpen = true;

private void menuTransition\_Tick(object sender, EventArgs e)

{

if (isMenuOpen)

{

sideBarMenu.Width -= 20;

if (sideBarMenu.Width <= 53)

{

sideBarMenu.Width = 53;

isMenuOpen = false;

menuTransition.Stop();

}

}

else

{

sideBarMenu.Width += 20;

if (sideBarMenu.Width >= 280)

{

sideBarMenu.Width = 280;

isMenuOpen = true;

menuTransition.Stop();

}

}

}

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

{

menuTransition.Start();

}

public void OpenDebtListForm()

{

if (debtList == null)

{

debtList = new ListForm(this);

debtList.FormClosed += DebtList\_FormClosed;

debtList.MdiParent = this;

debtList.Show();

}

else debtList.Activate();

debtList.Dock = DockStyle.Fill;

}

public void OpenEditForm(int id)

{

if (id != 0)

{

editDebt = null;

editDebt = new EditForm(this, id);

editDebt.MdiParent = this;

editDebt.Show();

}

else

{

if (editDebt == null)

{

editDebt = new EditForm(this);

editDebt.FormClosed += EditDebt\_FormClosed;

editDebt.MdiParent = this;

editDebt.Show();

}

else editDebt.Activate();

}

editDebt.Dock = DockStyle.Fill;

}

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

{

OpenDebtListForm();

}

private void DebtList\_FormClosed(object? sender, FormClosedEventArgs e)

=> debtList = null;

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

{

if (addDebt == null)

{

addDebt = new AddForm(this);

addDebt.FormClosed += AddDebt\_FormClosed;

addDebt.MdiParent = this;

addDebt.Show();

}

else addDebt.Activate();

addDebt.Dock = DockStyle.Fill;

}

private void AddDebt\_FormClosed(object? sender, FormClosedEventArgs e)

=> addDebt = null;

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

{

if (editDebt == null)

{

editDebt = new EditForm(this);

editDebt.FormClosed += EditDebt\_FormClosed;

editDebt.MdiParent = this;

editDebt.Show();

}

else editDebt.Activate();

editDebt.Dock = DockStyle.Fill;

}

private void EditDebt\_FormClosed(object? sender, FormClosedEventArgs e)

=> editDebt = null;

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

{

OpenDebtListForm();

}

private void DBForm\_FormClosing(object sender, FormClosingEventArgs e)

{

database.CloseConnection();

}

}

}

Файл ListForm.cs:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Text.Json;

namespace Lab4

{

public partial class ListForm : Form

{

private DBForm main;

public static DatabaseValues[] values = new DatabaseValues[0];

public static DatabaseValues[] oldValues = new DatabaseValues[0];

public ListForm(DBForm main)

{

InitializeComponent();

this.main = main;

}

public void UpdateTableValues()

{

oldValues = new DatabaseValues[values.Length];

Array.Copy(values, oldValues, values.Length);

debtTable.Rows.Clear();

debtTable.RowCount = values.Length;

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

{

debtTable.Rows[i].Cells[0].Value = values[i].Id;

debtTable.Rows[i].Cells[1].Value = values[i].FirstName;

debtTable.Rows[i].Cells[2].Value = values[i].LastName;

debtTable.Rows[i].Cells[3].Value = values[i].Value;

debtTable.Rows[i].Cells[4].Value = values[i].LoanDate.ToString("dd/MM/yyyy");

debtTable.Rows[i].Cells[5].Value = values[i].ExpirationDate.ToString("dd/MM/yyyy");

if (values[i].Status == "Active" && DateTime.Compare(values[i].ExpirationDate, DateTime.Now) < 0)

{

values[i].Status = "Expired";

DBForm.database.UpdateStatus(values[i].Id, "Expired");

}

debtTable.Rows[i].Cells[6].Value = values[i].Status;

if (values[i].Status == "Active")

debtTable.Rows[i].Cells[6].Style.ForeColor = Color.LightGreen;

else

debtTable.Rows[i].Cells[6].Style.ForeColor = Color.Red;

}

}

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

{

values = DBForm.database.ReadTable();

UpdateTableValues();

}

private void ListForm\_Activated(object sender, EventArgs e)

{

values = DBForm.database.ReadTable();

UpdateTableValues();

}

private void debtTable\_CellMouseDoubleClick(object sender, DataGridViewCellMouseEventArgs e)

{

int id = Convert.ToInt32(debtTable.Rows[e.RowIndex].Cells[0].Value);

main.OpenEditForm(id);

}

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

{

if (values.Length == 0)

{

MessageBox.Show("Database contains no data!", "", MessageBoxButtons.OK);

return;

}

string[] output = new string[values.Length];

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

output[i] = JsonSerializer.Serialize(values[i]);

SaveFileDialog saveFileDialog = new SaveFileDialog();

saveFileDialog.Filter = "json (\*json)|\*.json|Все файлы (\*.\*)|\*.\* ";

saveFileDialog.Title = "Save data";

saveFileDialog.InitialDirectory = Environment.GetFolderPath(Environment.SpecialFolder.Desktop);

if(saveFileDialog.ShowDialog() == DialogResult.OK)

{

string filepath = saveFileDialog.FileName;

File.WriteAllLines(filepath, output);

}

}

}

}

Файл EditForm.cs:

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab4

{

public partial class EditForm : Form

{

private bool databaseContainsThisId = false;

private DBForm main;

public EditForm(DBForm main)

{

InitializeComponent();

this.main = main;

}

public EditForm(DBForm main, int id)

{

InitializeComponent();

this.main = main;

this.id = id;

}

private int id=0;

private int index;

private void ClearFields()

{

firstNameTextBox.Clear();

secondNameTextBox.Clear();

amountTextBox.Clear();

loanDatePicker.Value = DateTime.Now;

expirationDatePicker.Value = DateTime.Now.AddDays(1);

}

private void LoadDatabaseRecord()

{

index = Array.FindIndex(ListForm.values, obj => obj.Id == id);

firstNameTextBox.Text = ListForm.values[index].FirstName;

secondNameTextBox.Text = ListForm.values[index].LastName;

amountTextBox.Text = ListForm.values[index].Value.ToString();

loanDatePicker.Value = ListForm.values[index].LoanDate;

expirationDatePicker.Value = ListForm.values[index].ExpirationDate;

}

private void idTextBox\_TextChanged(object? sender, EventArgs? e)

{

databaseContainsThisId = false;

if (!Regex.IsMatch(idTextBox.Text, $"^[1-9][0-9]\*$") && idTextBox.Text != "")

{

idTextBox.Focus();

idTextBox.Text = idTextBox.Text.Substring(0, idTextBox.Text.Length - 1);

idTextBox.Select(idTextBox.Text.Length, 0);

return;

}

if (idTextBox.Text == "")

{

ClearFields();

return;

}

id = Convert.ToInt32(idTextBox.Text);

if (!ListForm.values.Any(obj => obj.Id == id))

{

ClearFields();

idTextBox.Focus();

return;

}

databaseContainsThisId = true;

LoadDatabaseRecord();

}

private bool CheckIfContainsThisId()

{

if (!databaseContainsThisId)

{

MessageBox.Show("Database does not contain any record with this id!", "Warning", MessageBoxButtons.OK, MessageBoxIcon.Warning);

idTextBox.Focus();

idTextBox.SelectAll();

}

return databaseContainsThisId;

}

private void IfDatabaseContainsThisId(object sender, MouseEventArgs e)

{

CheckIfContainsThisId();

}

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

{

if (!Regex.IsMatch(amountTextBox.Text, $"^[1-9]\*[0-9]+$") && amountTextBox.Text != "")

{

amountTextBox.Focus();

amountTextBox.Text = amountTextBox.Text.Substring(0, amountTextBox.Text.Length - 1);

amountTextBox.Select(amountTextBox.Text.Length, 0);

return;

}

}

private void EditForm\_Deactivate(object sender, EventArgs e)

{

ClearFields();

idTextBox.Clear();

}

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

{

firstNameTextBox.Select(0, 0);

}

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

{

secondNameTextBox.Select(0, 0);

}

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

{

ClearFields();

if (id != 0)

{

idTextBox.Text = id.ToString();

}

}

private bool CheckIfThereAnyChanges()

{

int amountOld = ListForm.values[index].Value;

string firstNameOld = ListForm.values[index].FirstName;

string secondNameOld = ListForm.values[index].LastName;

DateTime loanDateOld = ListForm.values[index].LoanDate;

DateTime expirationDateOld = ListForm.values[index].ExpirationDate;

int amount = Convert.ToInt32(amountTextBox.Text);

string firstName = firstNameTextBox.Text;

string secondName = secondNameTextBox.Text;

DateTime loanDate = loanDatePicker.Value;

DateTime expirationDate = expirationDatePicker.Value;

if (!(amount == amountOld && firstNameOld == firstName &&

secondNameOld == secondName && loanDateOld == loanDate &&

expirationDateOld == expirationDate))

return true;

return false;

}

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

{

if(!CheckIfContainsThisId()) return;

if (CheckIfThereAnyChanges())

{

int amount = Convert.ToInt32(amountTextBox.Text);

string firstName = firstNameTextBox.Text;

string secondName = secondNameTextBox.Text;

string loanDate = loanDatePicker.Value.ToString("dd/MM/yyyy");

string expirationDate = expirationDatePicker.Value.ToString("dd/MM/yyyy");

string status;

if (DateTime.Compare(expirationDatePicker.Value, DateTime.Now) > 0) status = "Active";

else status = "Expired";

DatabaseValues \_new = new DatabaseValues(id, firstName, secondName, amount, loanDate, expirationDate, status);

ListForm.values[index] = \_new;

DBForm.database.UpdateValue(\_new);

MessageBox.Show("Database record was successfully updated!", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

main.OpenDebtListForm();

}

else

{

MessageBox.Show("No updates recorded!", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

private DatabaseValues[] RemoveElement()

{

DatabaseValues[] newValues = new DatabaseValues[ListForm.values.Length-1];

for(int i = 0, j = 0; i < ListForm.values.Length; i++)

{

if(i!=index) newValues[j++] = ListForm.values[i];

}

return newValues;

}

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

{

if (!CheckIfContainsThisId()) return;

DialogResult res = MessageBox.Show("Are you sure you want to delete this record?", "", MessageBoxButtons.YesNo);

if (res == DialogResult.Yes)

{

DBForm.database.DeleteValue(ListForm.values[index]);

ListForm.values = RemoveElement();

MessageBox.Show("Database record was successfully deleted!", "", MessageBoxButtons.OK, MessageBoxIcon.Information);

main.OpenDebtListForm();

}

}

}

}

Файл AddForm.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Text.RegularExpressions;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Lab4

{

public partial class AddForm : Form

{

private DBForm main;

public AddForm(DBForm main)

{

this.main = main;

InitializeComponent();

}

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

{

ClearFields();

}

private void ClearFields()

{

if (ListForm.values.Length == 0) idTextBox.Text = "1";

else idTextBox.Text = (ListForm.values.Last().Id + 1).ToString();

firstNameTextBox.Clear();

secondNameTextBox.Clear();

amountTextBox.Clear();

loanDatePicker.Value = DateTime.Now;

expirationDatePicker.Value = DateTime.Now.AddDays(1);

}

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

{

int id = Convert.ToInt32(idTextBox.Text);

string firstName = firstNameTextBox.Text;

string lastName = secondNameTextBox.Text;

string loanDate = loanDatePicker.Text;

string expirationDate = expirationDatePicker.Text;

string status = "Active";

if (string.IsNullOrEmpty(firstName))

{

MessageBox.Show("First name field cannot be empty!", "", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

if (string.IsNullOrEmpty(lastName))

{

MessageBox.Show("Second name field cannot be empty!", "", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

if (!int.TryParse(amountTextBox.Text, out int value))

{

MessageBox.Show("Amount field cannot be empty!", "", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

if (DateTime.Compare(

DateTime.Parse(loanDate),

DateTime.Parse(expirationDate)) >= 0)

{

MessageBox.Show("Expiration date cannot be equal or earlier than loan date!", "", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

Array.Resize(ref ListForm.values, ListForm.values.Length + 1);

DatabaseValues \_new = new DatabaseValues(id, firstName, lastName, value, loanDate, expirationDate, status);

ListForm.values[ListForm.values.Length - 1] = \_new;

DBForm.database.InsertValue(\_new);

ClearFields();

main.OpenDebtListForm();

}

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

{

if (!Regex.IsMatch(amountTextBox.Text, $"^[1-9]\*[0-9]+$") && amountTextBox.Text != "")

{

amountTextBox.Focus();

amountTextBox.Text = amountTextBox.Text.Substring(0, amountTextBox.Text.Length-1);

amountTextBox.Select(amountTextBox.Text.Length, 0);

return;

}

}

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

{

firstNameTextBox.Select(0, 0);

}

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

{

secondNameTextBox.Select(0, 0);

}

private void AddForm\_Deactivate(object sender, EventArgs e)

{

ClearFields();

}

}

}

Файл Lab4Tests.cs

using Lab4;

using Windows.Devices.Lights.Effects;

namespace Lab4\_Tests

{

[TestClass]

public class Lab4Tests

{

private static string testDBPath = "testDatabase.db";

private static DatabaseManipulations database = new DatabaseManipulations(testDBPath);

int previousRecordCount = 0;

[TestMethod]

public void Test1Insertion()

{

DatabaseValues[] records = database.ReadTable();

previousRecordCount = records.Length;

for(int i = 0; i < 100; i++)

{

database.InsertValue(

new DatabaseValues(

previousRecordCount!=0 ? records[records.Length-1].Id + i: i,

"Test", "Test",

i + 1, DateTime.Now.ToString("dd/MM/yyyy"),

DateTime.Now.AddDays(i+1).ToString("dd/MM/yyyy"), "Active")

);

}

records = database.ReadTable();

Assert.IsTrue(records.Length - previousRecordCount == 100);

}

[TestMethod]

public void Test2Update()

{

DatabaseValues[] records = database.ReadTable();

DatabaseValues lastRecord = records[records.Length - 1];

DatabaseValues \_new = records[records.Length-1];

\_new.Value = int.MaxValue;

database.UpdateValue(\_new);

records = database.ReadTable();

Assert.IsTrue(lastRecord != records[records.Length - 1]);

}

[TestMethod]

public void Test3Delete()

{

DatabaseValues[] records = database.ReadTable();

previousRecordCount = records.Length;

for (int i = 1; i <= 100; i++)

{

database.DeleteValue(records[previousRecordCount - i]);

}

records = database.ReadTable();

int newRecordCount = records.Length;

Assert.IsTrue(previousRecordCount - newRecordCount == 100);

}

}

}