Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский технологический университет «МИСИС».

Институт компьютерных наук.

Кафедра инженерной кибернетики.

Домашнее задание по курсу «Объектно-ориентированное программирование»

на тему:

«Кулинарная книга»

Выполнила: Игнатова Мария Евгеньевна

Студент группы БИВТ-23-10

Научный руководитель:

Куренкова Татьяна Васильевна

Москва 2024

**Содержание**

1. Цель работы………………………………………………………….3
2. Основная часть………………………………………………...…4-17
   1. Техническое задание………………………………………………4-5
   2. Описание интерфейса……………………………………………..5-9
   3. Техническое описание…………………………………………...9-12
3. Тестирование……………………………………………………….13
4. Документирование…………………………………………………14

Заключение………………………………………………………………..15

Список использованной литературы…………………………………….16

Приложение №1……………………………………………………….17-20

Приложение №2……………………………………………………….21-34

**1 Цель работы**

Цель работы: разработать приложение, позволяющее хранить названия блюд по категориям «Завтрак», «Обед», «Ужин», время их приготовления и калорийность. Приложение позволит пользователю быстрее определяться с желаемым блюдом, что существенно сократит время готовки.

**2 Основная часть**

2.1 Техническое задание

Требования к функциональным характеристикам: Система должна обеспечивать возможность выполнения следующих функций:

1. Выбор нужного приёма пищи.
2. Просмотр списка блюд по выбранному приёму пищи.
3. Получение количества блюд по выбранному приёму пищи.
4. Получение общего количества блюд.
5. Просмотр диаграммы соотношения количества блюд по категориям «Завтрак», «Обед», «Ужин».
6. Добавление новых блюд.
7. Удаление имеющихся блюд.
8. Сортировка списка блюд по выбранному критерию(наименование блюда, время приготовления, калорийность блюда).
9. Получение списка блюд по выбранному критерию в выпадающем списке.

Исходные данные:

1. Список имеющихся блюд.
2. Сведения о времени приготовления и калорийности конкретного блюда.

Результат:

1. Отображённая диаграмма соотношения количества блюд по разным приёмам пищи.
2. Отображение общего числа блюд.
3. Список блюд по выбранному приёму пищи.
4. Отсортированный по выбранному параметру список блюд.
5. Список блюд по выбранной из выпадающего списка категории.
6. Добавление новых блюд.
7. Удаление имеющихся блюд.
8. Отображение числа блюд по выбранному приёму пищи.

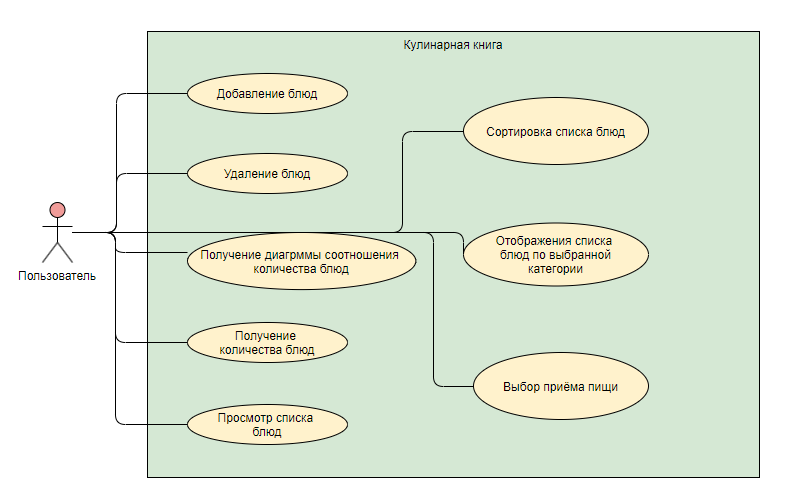
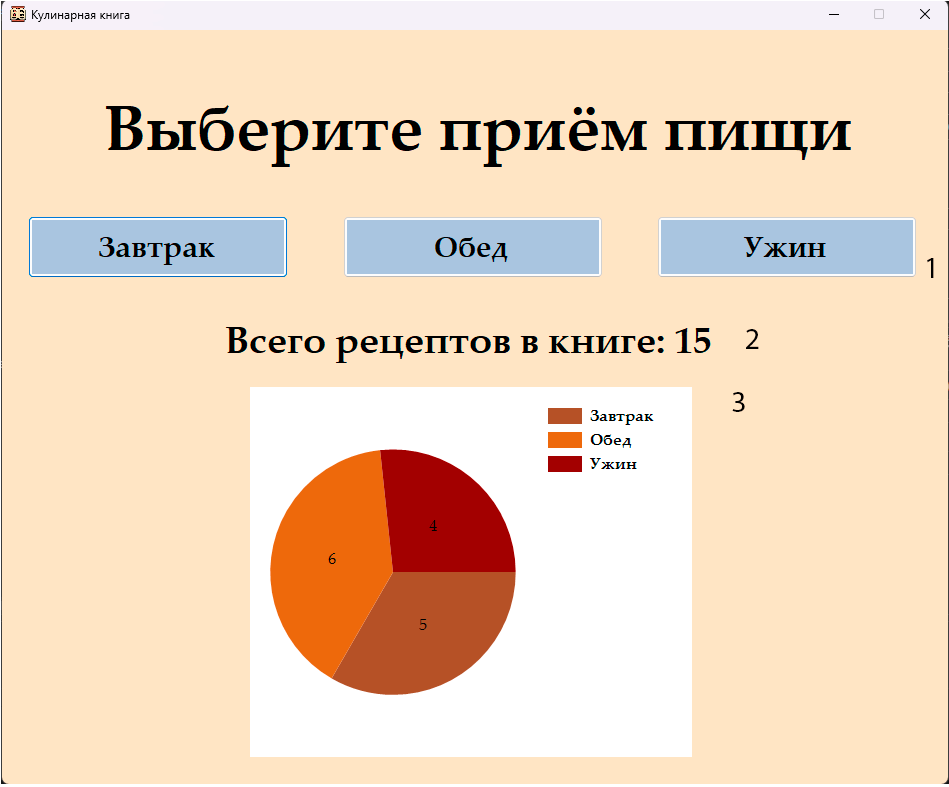
Диаграмма взаимодействия пользователя с системой (рис. 1).

Рис.1

2.2 Описание интерфейса

Пользователю доступно 4 формы для взаимодействия с системой: основная форма «Кулинарная книга» (рис. 2), форма «Завтрак» (рис. 3), форма «Обед» (рис. 4), форма «Ужин» (рис.5).



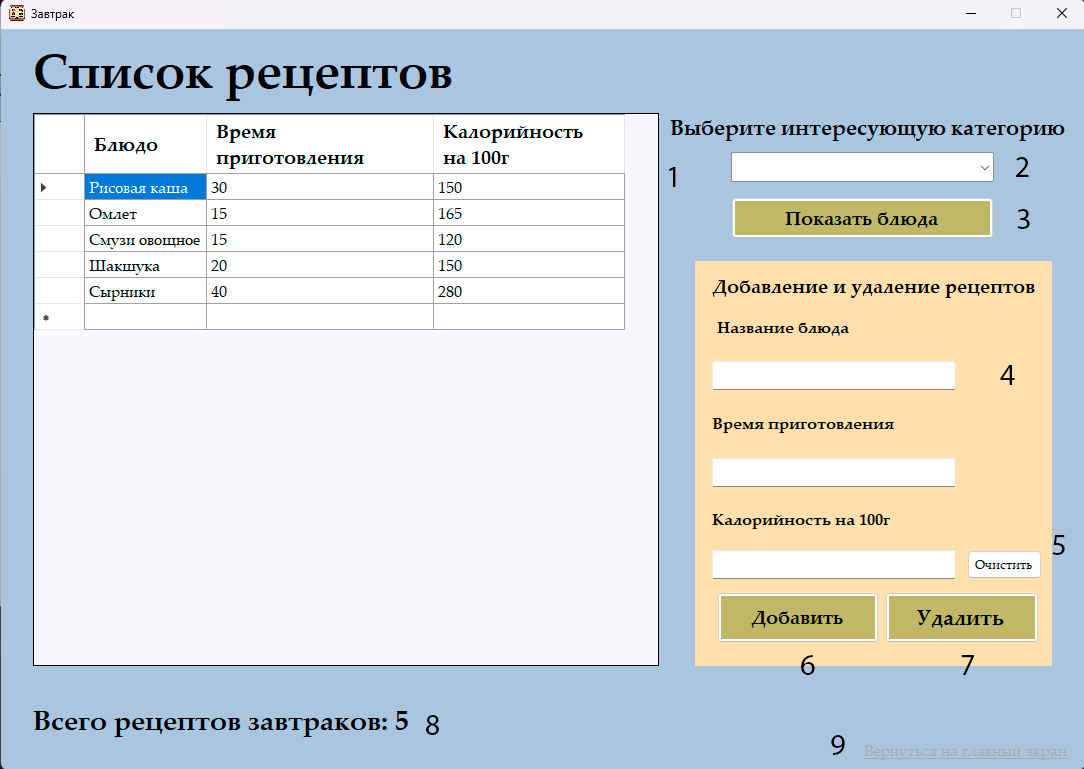
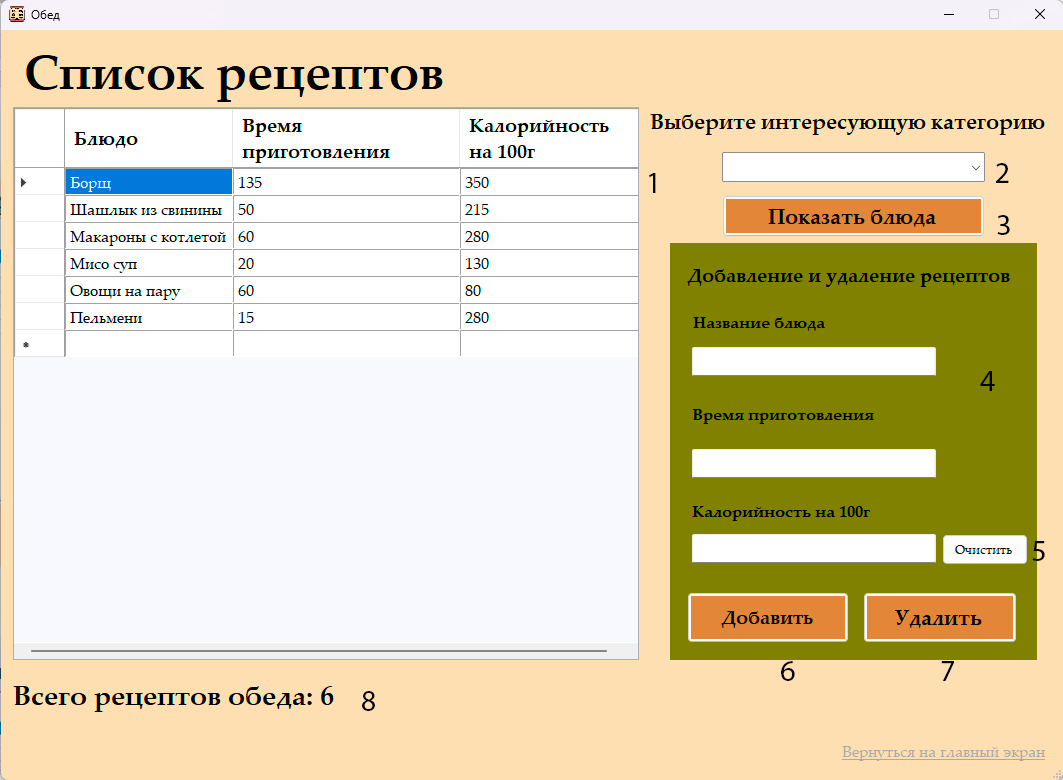
Рис. 2

Рис. 3



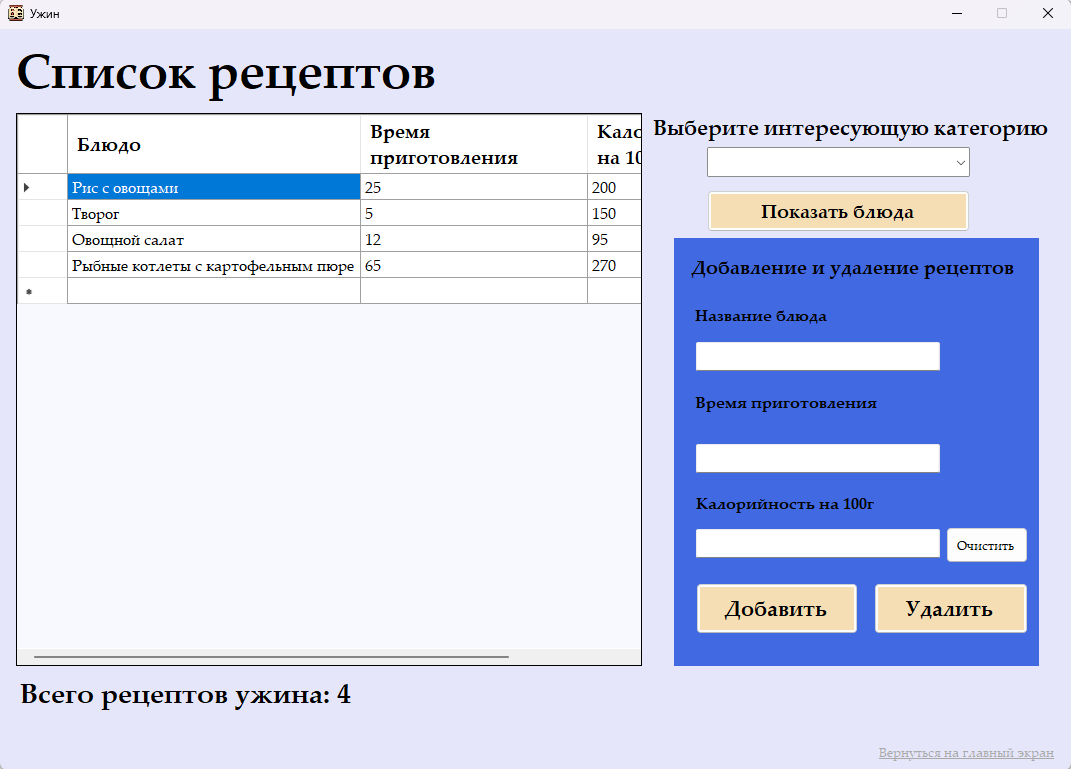
Рис. 4

Рис. 5

При открытии приложения пользователь попадает на основную форму «Кулинарная книга» (рис. 2), на которой расположены:

1. Кнопки выбора приёма пищи «Завтрак», «Обед», «Ужин».
2. Строка, отображающая общее количество блюд.
3. Диаграмма соотношения количества блюд по приёмам пищи.

Для того чтобы посмотреть списки блюд по необходимому приёму пищи, пользователю необходимо нажать на одну из трёх кнопок выбора приёма пищи – кнопка «Завтрак», кнопка «Обед», кнопка «Ужин».

После этого пользователь попадает на форму с выбранным приёмом пищи. Функционал форм «Завтрак», «Обед» и «Ужин» схож, на них расположены одни и те же элементы управления и взаимодействия. Отличаются формы только оформлением.

Рассмотрим функционал форм на примере формы «Завтрак».

На форме «Завтрак» расположены:

1. Список блюд по выбранному приёму пищи.
2. Выпадающий список для выбора интересующей категории блюд.
3. Кнопка «Показать блюда» для отображение списка п.2.
4. Поля для ввода информации о новом блюде.
5. Кнопка «Очистить» для очистки полей ввода.
6. Кнопка «Добавить» для нового блюда.
7. Кнопка «Удалить» для выбранного существующего блюда.
8. Строка для вывода количества блюд по выбранному приёму пищи.
9. Текст-ссылка для возврата на основную форму.

При попадании на форму выбранного приёма пищи список блюд и количество блюд обновляются автоматически.

Для добавления нового блюда пользователю необходима заполнить соответствующие поля с информацией (время следует указывать в минутах, калорийность – в вещественных числах, больших нуля). Затем нажать кнопку «Добавить». При успешном добавлении блюда программа выведет соответствующее сообщение. Список блюд обновиться автоматически.

Для удаления существующего блюда пользователю необходимо нажать на крайнюю левую ячейку в строке выбранного блюда, после чего нажать кнопку «Удалить». При успешном удалении блюда программа выведет соответствующее сообщение и автоматически обновит список.

Пользователь может вывести список блюд по интересующему критерию: «Сытные», «Быстрое приготовление», «Низкокалорийное». Для этого ему необходимо выбрать нужную категорию в выпадающем списке и нажать кнопку «Показать блюда». Для отображения полного списка необходимо выбрать категорию «Все блюда».

Для очистки всех полей ввода пользователю необходимо нажать кнопку «Очистить».

Для сортировки списка блюд пользователю необходимо нажать на заголовок столбца. Список сортируется по возрастанию значения ячеек. При повторном нажатии список сортируется в обратном порядке.

Для возврата на основную форму пользователю необходимо нажать на текст-ссылку «Вернуться на главный экран». Текущая форма закроется.

2.3 Техническое описание

Используемые ресурсы:

1. Язык программирования: C#

2. GUI фреймворк: Windows Forms

3. База данных: SQLite

Структура базы данных SQLite «Book\_of\_recipes» (рис. 6) для хранения данных о задачах:

- Таблица "Recipes"с полями:

- "id" INTEGER

- "name" TEXT,

- "time" INTEGER

- "calories" NUMERIC

- "type" TEXT

- PRIMARY KEY("id" AUTOINCREMENT)

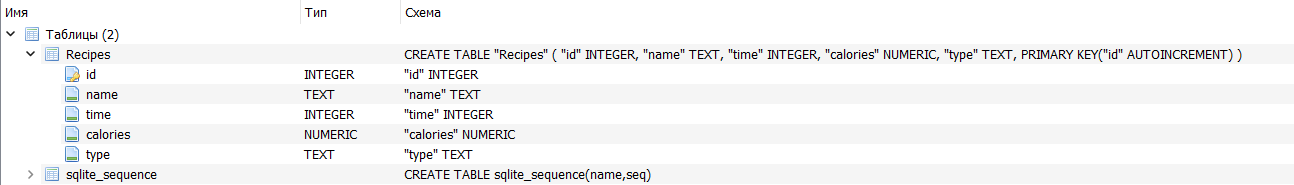


Рис. 6

Таблица «Recipes» хранит сведения о всех блюдах.

Рассмотрим коды классов продукта.

Класс Recipes.cs (см. приложение №1) представляет данные о конкретном блюде (номер блюда, наименование блюда, время приготовления, калорийность, категория приёма пищи), хранимые в полях int id, string name, int time, decimal calories, string type.

Класс позволяет создать экземпляры класса, которые затем помещаются в List<Recipes> recipes (пример), который используется для отображения списка задач и составления запросов для получения необходимых данных.

В классе содержится статический метод ReadFromDataBase () (см. приложение №1), который создаёт экземпляр класса List<Recipes> recipes, вносит в него данные о блюдах из базы данных и возвращает его.

Класс реализуется во всех формах приложения.

Рассмотрим коды форм продукта.

Так как функционал у форм «Завтрак», «Обед», «Ужин» одинаковый, то рассмотрим код форм продукта на примере формы «Завтрак».

При попадании на форму пользователю сразу же доступен список блюд по выбранному приёму пищи.

Это обеспечивает метод private void RefreshData(DataGridView dataGridView) (см. приложение 2 п.1), который использует рассмотренный выше метод ReadFromDataBase () (см. приложение №1), который определяет значение локальной переменной List <Recipes> recipes. Далее создается еще один List < Recipes> breakfasts, в который с помощью запроса к recipes погружаются сведения только о блюдах с категорией «Завтрак». Данные из breakfasts передаются в DataGridView.

В методе также происходит подсчёт количества элементов в списке breakfasts, которое затем записывается в строку вывода количества завтраков.

Метод GetColumns() (см. приложение №2) создает и обозначает столбцы для DataGridView, расположенного на выбранной форме.

Метод void button1\_Click(object sender, EventArgs e) (см. приложение 2) используется для фильтрации списка блюд по выбранной категории. В нём создаётся экземпляр класса List<Recipes> recipes с помощью метода GetFromDataBase() (см. выше), затем List<Recipes> с блюдами, принадлежащих категории «Завтрак», и очищается поле DataGridView. Далее метод получает информацию о индексе выделенной в выпадающем списке (см. выше) категории, по условию индекса выбранной категории переходит в нужную точку кода (case №индекса) и выводит в DataGridView информацию о блюдах, чья категория совпадает с выбранной.

При нажатии на кнопку «Добавить» выполняется метод void button2\_Add\_Click(object sender, EventArgs e) (см. приложение №2), который проверяет корректность введённых данных, а после создаёт запрос к базе данных и добавляет в неё новые данные.

При нажатии на кнопку «Удалить» выполняется метод void button3\_Delete\_Click(object sender, EventArgs e) (см. приложение №2) удаляет выбранную операцию из списка блюд. При этом он проверяет, выбрана ли строка, в противном случае выводит сообщение с просьбой выбрать нужную строку. При корректном выборе строки метод создает запрос к базе данных для удаления строки с нужным значением номера блюда. Затем удаляет выбранную строку из DataGridView и выводит сообщение об успешном удалении операции.

Метод void button2\_Click(object sender, EventArgs e) (см. приложение 2) очищает все поля ввода, присваивая им значение пустой строки.

Метод void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e) (см. приложение 2) возвращает пользователя на основную форму «Кулинарная книга», закрывая при этом текущую форму.

Диаграмма классов приложения «Кулинарная книга» представлена на рисунке 7.

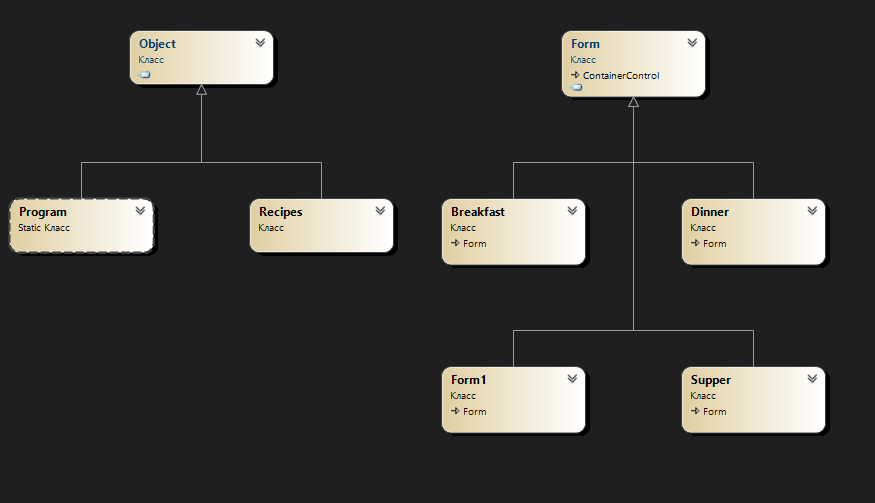
****

Рис. 7

**3 Тестирование**

Проведено тестирование метода ReadFromDataBase () (см. приложение №1), содержащегося в Recipes (см. приложение №1). Для этого создан модульный тест, который инициализирует массив и заполняет его ожидаемыми данными, создает массив с помощью метода ReadFromDataBase() и сравнивает результаты. Также для тестов создана тестовая база данных SQLite TestReadFomDataBase(рис. 8) с заданными исходными данными, равных данным в массиве с ожидаемыми результатами.

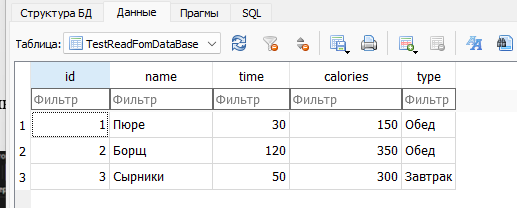


Рис. 8

Результат запуска всех тестов показан на рисунке 9 – успешен.

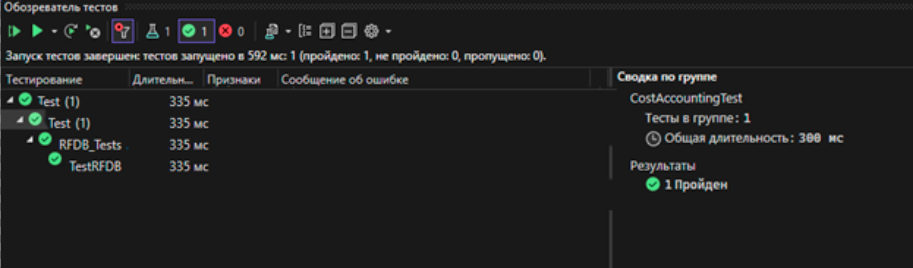


Рис. 9

(Код модульных тестов смотри в приложении №1 п.3 и п.4).

**4 Документирование**

Все программные модули самодокументированы, текст кода смотри в приложении 1 и приложении 2.

**Заключение**

Разработка приложения на C# для системы планирования "Кулинарная книга" является полезным проектом. Приложение позволит пользователям эффективно использовать своё время, так как упрощён выбор блюда, которое необходимо приготовить. Также оно позволяет следить за рационом, так как указана калорийность каждого блюда.

При разработке приложения необходимо уделить особое внимание пользовательскому интерфейсу, чтобы сделать его удобным и интуитивно понятным. Также необходимо было сделать его внешне привлекательным, чтобы использование приложение вызывало у пользователя положительные эмоции.

Разработка приложения «Кулинарная книга» позволяет улучшить свои навыки программирования и знания о языке C#, оформление приложения развивает насмотренность, способность гармонично располагать элементы приложения на форме.

**Список использованной литературы**

1. Документация Microsoft по C#: <https://docs.microsoft.com/en-us/dotnet/csharp/>
2. Справочник по Windows Forms: <https://docs.microsoft.com/en-us/dotnet/desktop/winforms/>
3. Статьи на сайте CodeProject о разработке приложений на C# в Windows Forms: <https://www.codeproject.com/>
4. Лекции и уроки по C# на сайте MSDN: <https://docs.microsoft.com/en-us/learn/>
5. Книга "Programming C# 7.0: Building Windows 10 Applications": <https://www.amazon.com/Programming-C-7-0-Building-Applications/dp/1491987650>

**Приложение №1.**

1. Class Program.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Кулинарная\_книга

{

internal static class Program

{

/// <summary>

/// Главная точка входа для приложения.

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

Application.Run(new Form1());

}

}

}

1. Class Recipes.cs

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Data.SQLite;

namespace Кулинарная\_книга

{

/// <summary>

/// класс, определящий каждый из рецептов

/// </summary>

public class Recipes

{

private int id;

private string name;

private int time;

private decimal calories;

private string type;

public Recipes(int id, string name, int time, decimal calories, string type)

{

this.id = id;

this.name = name;

this.time = time;

this.calories = calories;

this.type = type;

}

public int ID { get { return id; } }

public string Name { get { return name; } }

public int Time { get { return time; } }

public decimal Calories { get { return calories; } }

public string Type { get { return type; } }

/// <summary>

/// Статический метод, который определяет List<Recipes> recipes, содержащий в себе все рецепты из базы данных Book\_of\_Recipes

/// </summary>

/// <returns></returns>

public static List<Recipes> ReadFromDataBase()

{

List<Recipes> recipes = new List<Recipes>();

string connectionString = "Data Source=Book\_of\_recipes.db;";

string query = "SELECT \* FROM Recipes";

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

int id = reader.GetInt32(0);

string name = reader.GetString(1);

int time = reader.GetInt32(2);

decimal calories = reader.GetDecimal(3) ;

string type = reader.GetString(4);

Recipes newTask = new Recipes(id,name,time,calories,type);

recipes.Add(newTask);

}

}

}

}

return recipes;

}

}

}

1. Class RFDB\_Tests.cs

using Microsoft.VisualStudio.TestTools.UnitTesting;

using System;

using System.Collections.Generic;

using ReadFromDataBaseTest;

namespace Test

{

[TestClass]

public class RFDB\_Tests

{

[TestMethod]

public void TestRFDB()

{

List<Recipes> expectedTasks = new List<Recipes>()

{

new Recipes(1, "Пюре", 30, 150, "Обед"),

new Recipes(2, "Борщ", 120, 350, "Обед") ,

new Recipes(3, "Сырники", 50, 300, "Завтрак")

};

List<Recipes> actualRecipes = Recipes.ReadFromDataBase();

Assert.AreEqual(expectedTasks.Count, actualRecipes.Count);

for (int i = 0; i < expectedTasks.Count; i++)

{

Assert.AreEqual(expectedTasks[i].ID, actualRecipes[i].ID);

Assert.AreEqual(expectedTasks[i].Name, actualRecipes[i].Name);

Assert.AreEqual(expectedTasks[i].Time, actualRecipes[i].Time);

Assert.AreEqual(expectedTasks[i].Calories, actualRecipes[i].Calories);

Assert.AreEqual(expectedTasks[i].Type, actualRecipes[i].Type);

}

}

}

}

1. Class Recipes.cs

using System;

using System.Collections.Generic;

using System.Data.SQLite;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace ReadFromDataBaseTest

{

public class Recipes

{

private int id;

private string name;

private int time;

private decimal calories;

private string type;

public Recipes(int id, string name, int time, decimal calories, string type)

{

this.id = id;

this.name = name;

this.time = time;

this.calories = calories;

this.type = type;

}

public int ID { get { return id; } }

public string Name { get { return name; } }

public int Time { get { return time; } }

public decimal Calories { get { return calories; } }

public string Type { get { return type; } }

/// <summary>

/// Статический метод, который определяет List<Recipes> recipes, содержащий в себе все рецепты из базы данных Book\_of\_Recipes

/// </summary>

/// <returns></returns>

public static List<Recipes> ReadFromDataBase()

{

List<Recipes> recipes = new List<Recipes>();

string connectionString = "Data Source=TestReadFomDataBase.db;";

string query = "SELECT \* FROM TestReadFomDataBase";

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(query, connection))

{

using (SQLiteDataReader reader = command.ExecuteReader())

{

while (reader.Read())

{

int id = reader.GetInt32(0);

string name = reader.GetString(1);

int time = reader.GetInt32(2);

decimal calories = reader.GetDecimal(3);

string type = reader.GetString(4);

Recipes newTask = new Recipes(id, name, time, calories, type);

recipes.Add(newTask);

}

}

}

}

return recipes;

}

static void Main(string[] args)

{

}

}

}

**Приложение №2.**

1. Код формы Form1.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;

namespace Кулинарная\_книга

{

public partial class Form1 : Form

{

public Form1()

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

}

/// <summary>

/// Переход на форму "Завтрак"

/// </summary>

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

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

private void button1\_breakfast\_Click(object sender, EventArgs e)

{

Breakfast breakfast = new Breakfast();

breakfast.Show();

this.Hide();

}

/// <summary>

/// Переход на форму "Обед"

/// </summary>

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

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

private void button2\_dinner\_Click(object sender, EventArgs e)

{

Dinner dinner = new Dinner();

dinner.ShowDialog();

this.Hide();

}

/// <summary>

/// Переход на форму "Ужин"

/// </summary>

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

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

private void button3\_supper\_Click(object sender, EventArgs e)

{

Supper supper = new Supper();

supper.Show();

this.Hide();

}

/// <summary>

/// Загрузка основной формы

/// </summary>

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

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

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

{

RefreshMainForm();

}

/// <summary>

/// Метод, обновляющий строку с выводом количества рецептов и диаграмму, отображающую соотношение количества рецептов в категориях

/// </summary>

private void RefreshMainForm()

{

List<Recipes> recipes = Recipes.ReadFromDataBase();

int count = recipes.Count();

label2\_countRecipes.Text = "Всего рецептов в книге: " + count.ToString();

List<string> categories = recipes.Select(r => r.Type).Distinct().ToList();

chart1.Series[0].Points.Clear();

foreach (string ctg in categories)

{

decimal total = recipes.Where(q => q.Type == ctg).Count();

if (total != 0)

{

chart1.Series[0].Points.AddXY(ctg, total);

}

}

}

}

}

1. Код формы Breakfast.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SQLite;

using System.Drawing;

using System.Globalization;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

using System.Windows.Forms.DataVisualization.Charting;

using System.Windows.Forms.VisualStyles;

namespace Кулинарная\_книга

{

public partial class Breakfast : Form

{

/// <summary>

/// Форма взаимодействия пользователя с разделом "Завтраки"

/// </summary>

public Breakfast()

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

}

/// <summary>

/// Загрузка формы "Завтрак"

/// </summary>

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

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

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

{

GetColumns();

RefreshData(dataGridView1);

}

/// <summary>

/// Создание колонок в DataGridView с нужными заголовками

/// </summary>

public void GetColumns()

{

dataGridView1.Columns.Add("id", "Номер");

dataGridView1.Columns.Add("name", "Блюдо");

dataGridView1.Columns.Add("time", "Время приготовления");

dataGridView1.Columns.Add("calories", "Калорийность на 100г");

dataGridView1.Columns.Add("type", "Приём пищи");

dataGridView1.Columns[0].Visible = false;

dataGridView1.Columns[4].Visible = false;

}

/// <summary>

/// Метод, обновляющий информацию в DataGridView и строке вывода количества рецептов

/// </summary>

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

private void RefreshData(DataGridView dataGridView)

{

List<Recipes> recipes = Recipes.ReadFromDataBase();

List < Recipes> breakfasts = (from p in recipes where p.Type == "Завтрак" select p).ToList();

dataGridView.Rows.Clear();

foreach (Recipes br in breakfasts)

{

dataGridView.Rows.Add(br.ID, br.Name, br.Time, br.Calories, br.Type);

}

int countbr = breakfasts.Count();

label2\_countBreakfast.Text = "Всего рецептов завтраков: " + countbr.ToString(); // Отображаем количество завтраков

}

/// <summary>

/// Метод, отображающий в dataGridView рецепты по выбранной категории, заданной в ComboBox

/// </summary>

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

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

private void button1\_Click(object sender, EventArgs e)

{

List<Recipes> recipes = Recipes.ReadFromDataBase();

List<Recipes> breakfasts = (from p in recipes where p.Type == "Завтрак" select p).ToList();

dataGridView1.Rows.Clear();

switch (comboBox1.SelectedIndex)

{

case 0:

foreach (Recipes br in breakfasts) { dataGridView1.Rows.Add(br.ID, br.Name, br.Time, br.Calories, br.Type); }

break;

case 1:

foreach (Recipes br in (from p in breakfasts where p.Calories >= 225 select p).ToList()) { dataGridView1.Rows.Add(br.ID, br.Name, br.Time, br.Calories, br.Type); }

break;

case 2:

foreach (Recipes br in (from p in breakfasts where p.Time <= 25 select p).ToList()) { dataGridView1.Rows.Add(br.ID, br.Name, br.Time, br.Calories, br.Type); }

break;

case 3:

foreach (Recipes br in (from p in breakfasts where p.Calories <= 130 select p).ToList()) { dataGridView1.Rows.Add(br.ID, br.Name, br.Time, br.Calories, br.Type); }

break;

}

}

/// <summary>

/// Метод, добавляющий в базу данных новый рецепт

/// </summary>

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

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

private void button2\_Add\_Click(object sender, EventArgs e)

{

var name = textBox1\_Name.Text;

var time = textBox2\_Time.Text;

var calories = textBox3\_Calories.Text;

string type = "Завтрак";

if (name == "" || time == "" || calories == "")

{

MessageBox.Show("Введиите данные");

return;

}

decimal Calories;

if (!decimal.TryParse(calories, out Calories))

{

MessageBox.Show("Введите корректное значение для калорийности блюда!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else if (Calories < 0)

{

MessageBox.Show("Введите корректное значение для калорийности блюда!" + "Вводимая калорийность блюда должна быть больше 0!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

int Time;

if(!int.TryParse(time, out Time))

{

MessageBox.Show("Введите корректное значение для времени приготовления!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else if(Time < 0)

{

MessageBox.Show("Введите корректное значение для времени приготовления!" + "Вводимое время в минутах должно быть больше 0!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

string connectionString = "Data Source=Book\_of\_recipes.db;";

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(connection))

{

string insertingquery = $"insert into Recipes(name, time, calories, type) values('{name}','{Time}','{Calories}', '{type}')";

command.CommandText = insertingquery;

command.ExecuteNonQuery();

}

connection.Close();

}

MessageBox.Show("Рецепт добавлен!", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

RefreshData(dataGridView1);

}

/// <summary>

/// Удаление выбранного рецепта из базы данных Book\_of\_recipes

/// </summary>

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

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

private void button3\_Delete\_Click(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count == 0)

{

MessageBox.Show("Пожалуйста, выберите задачу для удаления! \n" +

"Выберите для этого крайнюю левую ячейку в строке нужной задачи!", "Укажите строку!", MessageBoxButtons.OK, MessageBoxIcon.Question);

return;

}

string connectionString = "Data Source=Book\_of\_recipes.db;";

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(connection))

{

for (int i = 0; i < dataGridView1.SelectedRows.Count; i++)

{

int selectedId = Convert.ToInt32(dataGridView1.SelectedRows[i].Cells["id"].Value);

string deletingstring = $"DELETE FROM Recipes WHERE id = '{selectedId}'";

command.CommandText = deletingstring;

command.ExecuteNonQuery();

}

}

connection.Close();

}

dataGridView1.Rows.Remove(dataGridView1.SelectedRows[0]);

RefreshData(dataGridView1);

}

/// <summary>

/// Метод, который очищает все поля ввода при нажатии на кнопку "Очистить"

/// </summary>

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

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

private void button2\_Click(object sender, EventArgs e)

{

textBox1\_Name.Text = "";

textBox2\_Time.Text = "";

textBox3\_Calories.Text = "";

}

/// <summary>

/// Метод, возврающий на главную форму при нажатии на текст-ссылку

/// </summary>

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

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

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Form1 form = new Form1();

form.Show();

this.Close();

}

}

}

1. Код формы Dinner.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SQLite;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Кулинарная\_книга

{

public partial class Dinner : Form

{

public Dinner()

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

}

/// <summary>

/// Загрузка формы "Обед"

/// </summary>

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

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

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

{

GetColumns();

RefreshData(dataGridView1);

}

/// <summary>

/// Создание колонок в DataGridView с нужными заголовками

/// </summary>

public void GetColumns()

{

dataGridView1.Columns.Add("id", "Номер");

dataGridView1.Columns.Add("name", "Блюдо");

dataGridView1.Columns.Add("time", "Время приготовления");

dataGridView1.Columns.Add("calories", "Калорийность на 100г");

dataGridView1.Columns.Add("type", "Приём пищи");

dataGridView1.Columns[0].Visible = false;

dataGridView1.Columns[4].Visible = false;

}

/// <summary>

/// Метод, обновляющий информацию в DataGridView и строке вывода количества рецептов

/// </summary>

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

private void RefreshData(DataGridView dataGridView)

{

List<Recipes> recipes = Recipes.ReadFromDataBase();

List<Recipes> dinners = (from p in recipes where p.Type == "Обед" select p).ToList();

dataGridView.Rows.Clear();

foreach (Recipes br in dinners)

{

dataGridView.Rows.Add(br.ID, br.Name, br.Time, br.Calories, br.Type);

}

int countbr = dinners.Count();

label2\_countBreakfast.Text = "Всего рецептов обеда: " + countbr.ToString(); // Отображаем количество завтраков

}

/// <summary>

/// Метод, возврающий на главную форму при нажатии на текст-ссылку

/// </summary>

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

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

private void linkLabel1\_LinkClicked(object sender, LinkLabelLinkClickedEventArgs e)

{

Form1 form = new Form1();

form.Show();

this.Close();

}

/// <summary>

/// Метод, добавляющий в базу данных новый рецепт

/// </summary>

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

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

private void button2\_Add\_Click\_1(object sender, EventArgs e)

{

var name = textBox1\_Name.Text;

var time = textBox2\_Time.Text;

var calories = textBox3\_Calories.Text;

string type = "Обед";

if (name == "" || time == "" || calories == "")

{

MessageBox.Show("Введиите данные");

return;

}

decimal Calories;

if (!decimal.TryParse(calories, out Calories))

{

MessageBox.Show("Введите корректное значение для калорийности блюда!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else if (Calories < 0)

{

MessageBox.Show("Введите корректное значение для калорийности блюда!" + "Вводимая калорийность блюда должна быть больше 0!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

int Time;

if (!int.TryParse(time, out Time))

{

MessageBox.Show("Введите корректное значение для времени приготовления!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else if (Time < 0)

{

MessageBox.Show("Введите корректное значение для времени приготовления!" + "Вводимое время в минутах должно быть больше 0!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

string connectionString = "Data Source=Book\_of\_recipes.db;";

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(connection))

{

string insertingquery = $"insert into Recipes(name, time, calories, type) values('{name}','{Time}','{Calories}', '{type}')";

command.CommandText = insertingquery;

command.ExecuteNonQuery();

}

connection.Close();

}

MessageBox.Show("Рецепт добавлен!", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

RefreshData(dataGridView1);

}

/// <summary>

/// Удаление выбранного рецепта из базы данных Book\_of\_recipes

/// </summary>

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

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

private void button3\_Delete\_Click\_1(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count == 0)

{

MessageBox.Show("Пожалуйста, выберите задачу для удаления! \n" +

"Выберите для этого крайнюю левую ячейку в строке нужной задачи!", "Укажите строку!", MessageBoxButtons.OK, MessageBoxIcon.Question);

return;

}

string connectionString = "Data Source=Book\_of\_recipes.db;";

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(connection))

{

for (int i = 0; i < dataGridView1.SelectedRows.Count; i++)

{

int selectedId = Convert.ToInt32(dataGridView1.SelectedRows[i].Cells["id"].Value);

string deletingstring = $"DELETE FROM Recipes WHERE id = '{selectedId}'";

command.CommandText = deletingstring;

command.ExecuteNonQuery();

}

}

connection.Close();

}

dataGridView1.Rows.Remove(dataGridView1.SelectedRows[0]);

RefreshData(dataGridView1);

}

/// <summary>

/// Метод, который очищает все поля ввода при нажатии на кнопку "Очистить"

/// </summary>

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

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

private void button2\_Click\_1(object sender, EventArgs e)

{

textBox1\_Name.Text = "";

textBox2\_Time.Text = "";

textBox3\_Calories.Text = "";

}

/// <summary>

/// Метод, отображающий в dataGridView рецепты по выбранной категории, заданной в ComboBox

/// </summary>

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

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

private void button1\_Click\_1(object sender, EventArgs e)

{

List<Recipes> recipes = Recipes.ReadFromDataBase();

List<Recipes> dinners = (from p in recipes where p.Type == "Обед" select p).ToList();

dataGridView1.Rows.Clear();

switch (comboBox1.SelectedIndex)

{

case 0:

foreach (Recipes dn in dinners) { dataGridView1.Rows.Add(dn.ID, dn.Name, dn.Time, dn.Calories, dn.Type); }

break;

case 1:

foreach (Recipes dn in (from p in dinners where p.Calories >= 225 select p).ToList()) { dataGridView1.Rows.Add(dn.ID, dn.Name, dn.Time, dn.Calories, dn.Type); }

break;

case 2:

foreach (Recipes dn in (from p in dinners where p.Time <= 25 select p).ToList()) { dataGridView1.Rows.Add(dn.ID, dn.Name, dn.Time, dn.Calories, dn.Type); }

break;

case 3:

foreach (Recipes dn in (from p in dinners where p.Calories <= 130 select p).ToList()) { dataGridView1.Rows.Add(dn.ID, dn.Name, dn.Time, dn.Calories, dn.Type); }

break;

}

}

}

}

1. Код формы Supper.cs

using System;

using System.Collections.Generic;

using System.ComponentModel;

using System.Data;

using System.Data.SQLite;

using System.Drawing;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Forms;

namespace Кулинарная\_книга

{

public partial class Supper : Form

{

public Supper()

{

InitializeComponent();

StartPosition = FormStartPosition.CenterScreen;

}

/// <summary>

/// загрузка формы "Ужин"

/// </summary>

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

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

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

{

GetColumns();

RefreshData(dataGridView1);

}

/// <summary>

/// Создание колонок в DataGridView с нужными заголовками

/// </summary>

public void GetColumns()

{

dataGridView1.Columns.Add("id", "Номер");

dataGridView1.Columns.Add("name", "Блюдо");

dataGridView1.Columns.Add("time", "Время приготовления");

dataGridView1.Columns.Add("calories", "Калорийность на 100г");

dataGridView1.Columns.Add("type", "Приём пищи");

dataGridView1.Columns[0].Visible = false;

dataGridView1.Columns[4].Visible = false;

}

/// <summary>

/// Метод, обновляющий информацию в DataGridView и строке вывода количества рецептов

/// </summary>

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

private void RefreshData(DataGridView dataGridView)

{

List<Recipes> recipes = Recipes.ReadFromDataBase();

List<Recipes> dinners = (from p in recipes where p.Type == "Ужин" select p).ToList();

dataGridView.Rows.Clear();

foreach (Recipes br in dinners)

{

dataGridView.Rows.Add(br.ID, br.Name, br.Time, br.Calories, br.Type);

}

int countbr = dinners.Count();

label2\_countBreakfast.Text = "Всего рецептов ужина: " + countbr.ToString(); // Отображаем количество завтраков

}

/// <summary>

/// Метод, отображающий в dataGridView рецепты по выбранной категории, заданной в ComboBox

/// </summary>

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

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

private void button1\_Click\_1(object sender, EventArgs e)

{

List<Recipes> recipes = Recipes.ReadFromDataBase();

List<Recipes> dinners = (from p in recipes where p.Type == "Ужин" select p).ToList();

dataGridView1.Rows.Clear();

switch (comboBox1.SelectedIndex)

{

case 0:

foreach (Recipes dn in dinners) { dataGridView1.Rows.Add(dn.ID, dn.Name, dn.Time, dn.Calories, dn.Type); }

break;

case 1:

foreach (Recipes dn in (from p in dinners where p.Calories >= 225 select p).ToList()) { dataGridView1.Rows.Add(dn.ID, dn.Name, dn.Time, dn.Calories, dn.Type); }

break;

case 2:

foreach (Recipes dn in (from p in dinners where p.Time <= 25 select p).ToList()) { dataGridView1.Rows.Add(dn.ID, dn.Name, dn.Time, dn.Calories, dn.Type); }

break;

case 3:

foreach (Recipes dn in (from p in dinners where p.Calories <= 130 select p).ToList()) { dataGridView1.Rows.Add(dn.ID, dn.Name, dn.Time, dn.Calories, dn.Type); }

break;

}

}

/// <summary>

/// Метод, добавляющий в базу данных новый рецепт

/// </summary>

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

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

private void button2\_Add\_Click\_1(object sender, EventArgs e)

{

var name = textBox1\_Name.Text;

var time = textBox2\_Time.Text;

var calories = textBox3\_Calories.Text;

string type = "Ужин";

if (name == "" || time == "" || calories == "")

{

MessageBox.Show("Введиите данные");

return;

}

decimal Calories;

if (!decimal.TryParse(calories, out Calories))

{

MessageBox.Show("Введите корректное значение для калорийности блюда!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else if (Calories < 0)

{

MessageBox.Show("Введите корректное значение для калорийности блюда!" + "Вводимая калорийность блюда должна быть больше 0!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

int Time;

if (!int.TryParse(time, out Time))

{

MessageBox.Show("Введите корректное значение для времени приготовления!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

else if (Time < 0)

{

MessageBox.Show("Введите корректное значение для времени приготовления!" + "Вводимое время в минутах должно быть больше 0!", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

string connectionString = "Data Source=Book\_of\_recipes.db;";

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(connection))

{

string insertingquery = $"insert into Recipes(name, time, calories, type) values('{name}','{Time}','{Calories}', '{type}')";

command.CommandText = insertingquery;

command.ExecuteNonQuery();

}

connection.Close();

}

MessageBox.Show("Рецепт добавлен!", "Успешно", MessageBoxButtons.OK, MessageBoxIcon.Information);

RefreshData(dataGridView1);

}

/// <summary>

/// Удаление выбранного рецепта из базы данных Book\_of\_recipes

/// </summary>

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

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

private void button3\_Delete\_Click\_1(object sender, EventArgs e)

{

if (dataGridView1.SelectedRows.Count == 0)

{

MessageBox.Show("Пожалуйста, выберите задачу для удаления! \n" +

"Выберите для этого крайнюю левую ячейку в строке нужной задачи!", "Укажите строку!", MessageBoxButtons.OK, MessageBoxIcon.Question);

return;

}

string connectionString = "Data Source=Book\_of\_recipes.db;";

using (SQLiteConnection connection = new SQLiteConnection(connectionString))

{

connection.Open();

using (SQLiteCommand command = new SQLiteCommand(connection))

{

for (int i = 0; i < dataGridView1.SelectedRows.Count; i++)

{

int selectedId = Convert.ToInt32(dataGridView1.SelectedRows[i].Cells["id"].Value);

string deletingstring = $"DELETE FROM Recipes WHERE id = '{selectedId}'";

command.CommandText = deletingstring;

command.ExecuteNonQuery();

}

}

connection.Close();

}

dataGridView1.Rows.Remove(dataGridView1.SelectedRows[0]);

RefreshData(dataGridView1);

}

/// <summary>

/// Метод, который очищает все поля ввода при нажатии на кнопку "Очистить"

/// </summary>

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

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

private void button2\_Click\_1(object sender, EventArgs e)

{

textBox1\_Name.Text = "";

textBox2\_Time.Text = "";

textBox3\_Calories.Text = "";

}

/// <summary>

/// Метод, возврающий на главную форму при нажатии на текст-ссылку

/// </summary>

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

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

private void linkLabel1\_LinkClicked\_1(object sender, LinkLabelLinkClickedEventArgs e)

{

Form1 form = new Form1();

form.Show();

this.Close();

}

}

}