**Министерство образования Российской Федерации**

**Пензенский государственный университет**

Кафедра «Математическое обеспечение и применение ЭВМ»

Пояснительная записка к курсовому проекту   
по дисциплине

«Объектно-ориентированное программирование».

Разработка программы с использованием объектно-ориентированного подхода.   
ИС «Мебель»

Автор работы:

студент группы 23ВП2

Степанов Р.Д.

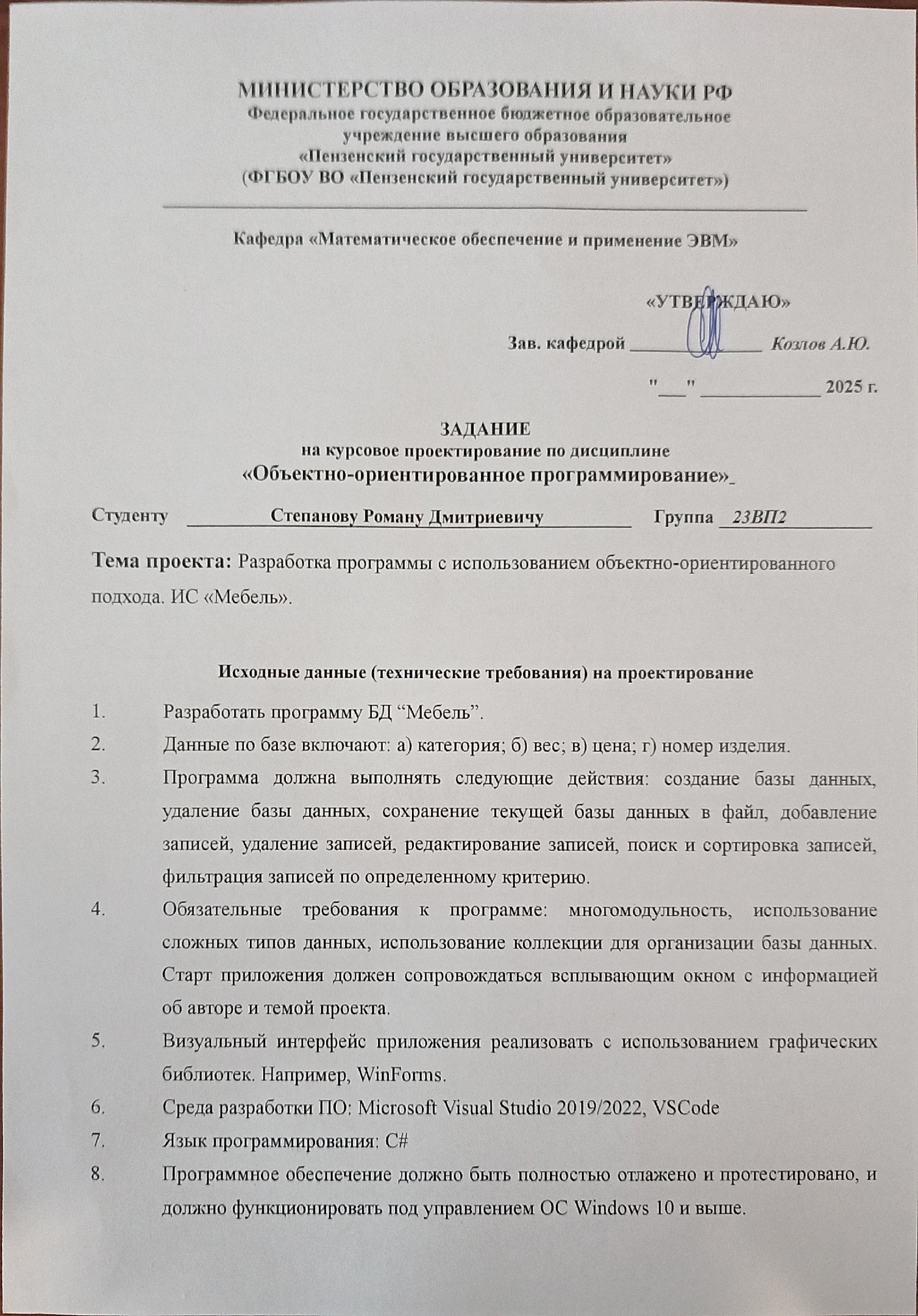
Принял:

к.т.н. Афонин А.Ю.

Работа сдана \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Пенза 2025



ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Кафедра «Математическое обеспечение и применение ЭВМ»

«УТВЕРЖДАЮ»

Зав. кафедрой \_\_\_\_\_\_\_\_\_\_Козлов А.Ю.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_2025 г.

ЗАДАНИЕ

на курсовое проектирование по курсу

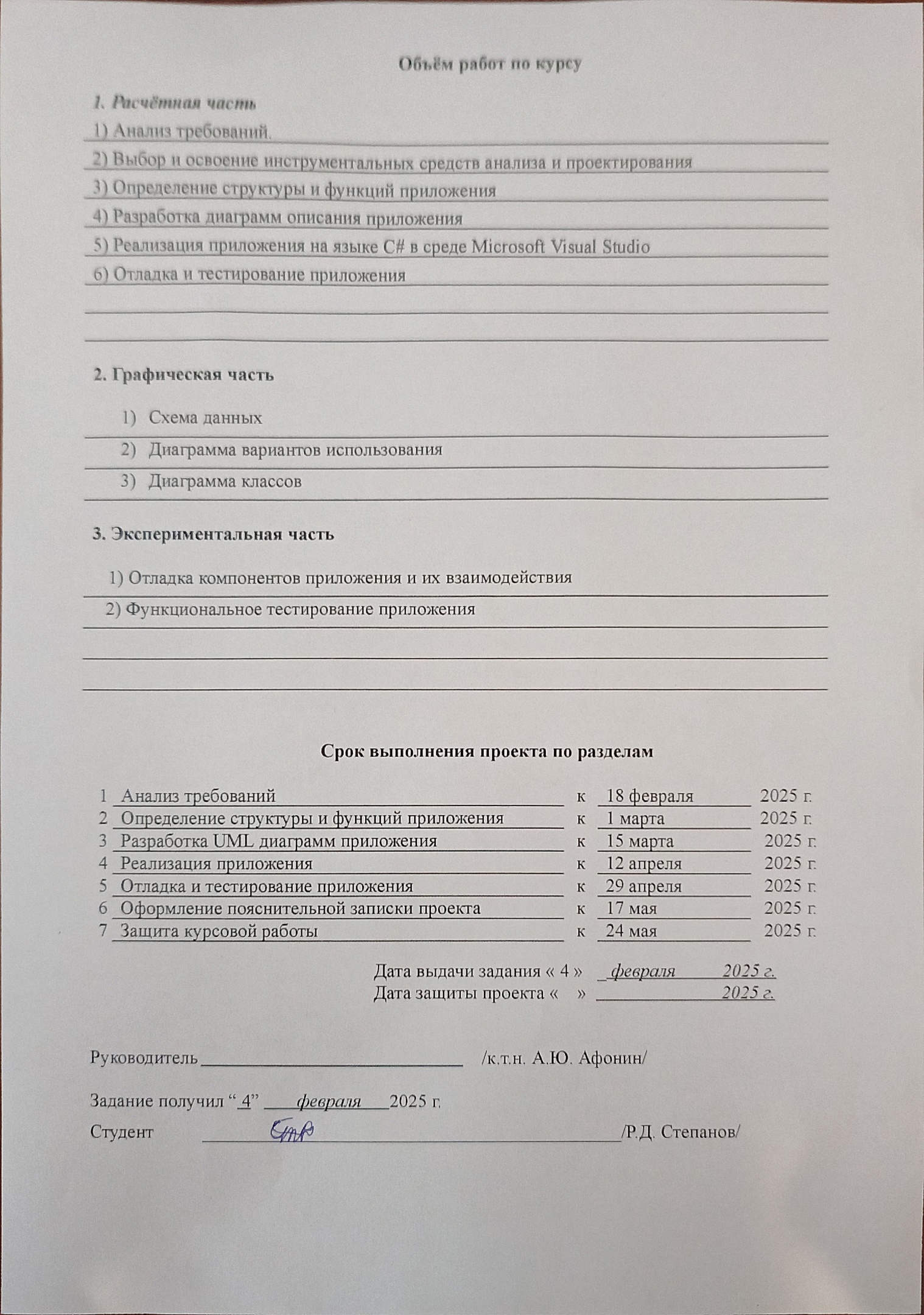
«Программирование динамических структур данных»

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студенту | Степанову Р.Д. | | Группа | 23ВП2 |
| Тема проекта | | Реализация структуры данных «Множество» с использованием контейнерных классов Vector и Map | | |

Исходные данные (технические требования) на проектирование

|  |
| --- |
| Реализовать структуру данных «Множество» и основные операции |
| над множеством (F1-F14) с помощью: односвязного списка, класса |
| «Список», контейнерного класса List, контейнерного класса Set, |
| контейнерного класса «Vector», контейнерного класса «Map». |
| Выполнить сравнение времени выполнения перечисленными выше |
| способами реализации следующих операций: создание множества из N |
| элементов, мощность множества, подмножество А⊂А, подмножество |
| В⊂А, равенство А=А, равенство В=А, объединение множеств А и В, |
| пересечение множеств А и В, разность А-В, разность B-A, симметричная |
| разность А и В. |
| Для разработки приложения использовать по выбору любую среду |
| программирования, которая имеет перечисленные в задании контейнерные |
| классы или аналогичные им. |
| Пользовательский интерфейс должен обеспечивать ввод |
| перечисленных в задании исходных значений, а также удобное |
| представление результирующей информации |
| Программное обеспечение должно быть полностью отлажено и |
| протестировано, функционировать под управлением ОС Windows. |
|  |
|  |
|  |
|  |
|  |

Объем работы по курсу



1. Расчетная часть

|  |
| --- |
| 1. Анализ требований к разработке программного обеспечения |
| 1. Проектирование программы |
| 1. Разработка программного обеспечения |
| 1. Тестирование программного обеспечения |
|  |
|  |
|  |

2. Графическая часть

|  |
| --- |
| 1. Диаграмма вариантов использования |
| 1. Диаграмма классов |
| 1. Диаграмма деятельности |
| 1. Диаграмма компонентов |
|  |

3. Экспериментальная часть

|  |
| --- |
| 1. Подготовка набора тестовых данных |
| 1. Отладка и тестирование программного обеспечения |
|  |
|  |

Срок выполнения проекта по разделам

|  |  |  |
| --- | --- | --- |
| 1. Анализ требований к разработке ПО | к | 20.02.25 |
| 1. Проектирование программы | к | 01.03.25 |
| 1. Разработка программного обеспечения | к | 15.04.25 |
| 1. Тестирование программного обеспечения | к | 15.05.25 |
| 1. Оформление пояснительной записки | к | 28.05.25 |
| 1. Защита курсовой работы | к | 29.05.25 |
|  | к |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Дата выдачи задания | « 11» |  | февраля 2025 |
| Дата защиты проекта | « » |  |  |

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель | | Самуйлов С.В. | |
| Задание получил | | | «11» февраля 2025 г. |
| Студент |  | | |

**Содержание**

[Введение 5](#_Toc197870323)

[1 Постановка задачи и анализ предметной области 6](#_Toc197870324)

[2 Выбор решения 7](#_Toc197870325)

[2.1 Определение необходимых модулей программы 7](#_Toc197870326)

[2.2 Определение структуры файла базы данных 10](#_Toc197870327)

[2.3 Модель интерфейса 10](#_Toc197870328)

[3 Описание разработки программы 11](#_Toc197870329)

[3.1 Кодирование 11](#_Toc197870330)

[3.2 Диаграмма компонентов 12](#_Toc197870331)

[4 Отладка и тестирование 13](#_Toc197870332)

[4.1 Функциональное тестирование 13](#_Toc197870333)

[4.2 Тестирование приложения 14](#_Toc197870334)

[5 Описание программы 19](#_Toc197870335)

[5.1 Разработака приложения «OOP Cursovaya Stepanov.exe» 19](#_Toc197870336)

[Выводы 20](#_Toc197870337)

[Список использованных источников 21](#_Toc197870338)

[Приложение A – Код программы 23](#_Toc197870339)

[Приложение Б – Руководство пользователя 53](#_Toc197870340)

Введение

Исходя из задания на курсовой проект, необходимо разработать информационную систему «Мебель» с использованием объектно-ориентированного программирования. Основная цель проекта — создание базы данных, позволяющей хранить сведения о мебели, а также реализовать основные операции: добавление, удаление, редактирование, поиск, сортировку и фильтрацию записей.

Актуальность разработки данной системы обусловлена широким применением информационных технологий в управлении и учёте товарных позиций в мебельной отрасли. Автоматизация хранения и обработки данных позволяет повысить эффективность работы предприятий, минимизировать ошибки и упростить доступ к информации.

Разработка программы требует использования современных инструментальных средств, таких как язык программирования C# и среда разработки Microsoft Visual Studio. В проекте предусмотрена работа с коллекциями для организации базы данных, а также реализация визуального интерфейса с помощью графических библиотек (например, Windows Forms).

Для успешного выполнения проекта необходимо выполнить следующие задачи:

1. Анализ требований и проектирование структуры приложения.
2. Разработка UML-диаграмм.
3. Реализация приложения с использованием объектно-ориентированного подхода.
4. Отладка и тестирование функционала.
5. Оформление документации и подготовка к защите проекта.

В результате выполнения проекта будет создана информационная система, соответствующая заданным требованиям, обеспечивающая удобный и интуитивно понятный интерфейс для работы с данными о мебели.

1 Постановка задачи и анализ предметной области

Необходимо разработать программу «Мебель». Информация о мебели включает следующие параметры: категория, вес, цена, номер изделия. Для реализации данного пункта необходимо определить структуру базы данных. Это позволит выявить возможные ограничения программы и минимизировать их влияние.

Программа должна предоставлять возможность создания файла базы данных, добавления новых записей, удаления и редактирования существующих записей, а также выполнения поиска, сортировки и фильтрации данных по заданным критериям. Также необходимо реализовать функции сохранения текущей базы данных в файл, сохранении данных в PDF файл, удаления файла базы данных и вывода информации в удобном формате. Визуальный интерфейс должен быть интуитивно понятным, с разделением областей меню и таблицы для отображения данных.

Взаимодействие пользователя с программой представлено на рисунке 1.

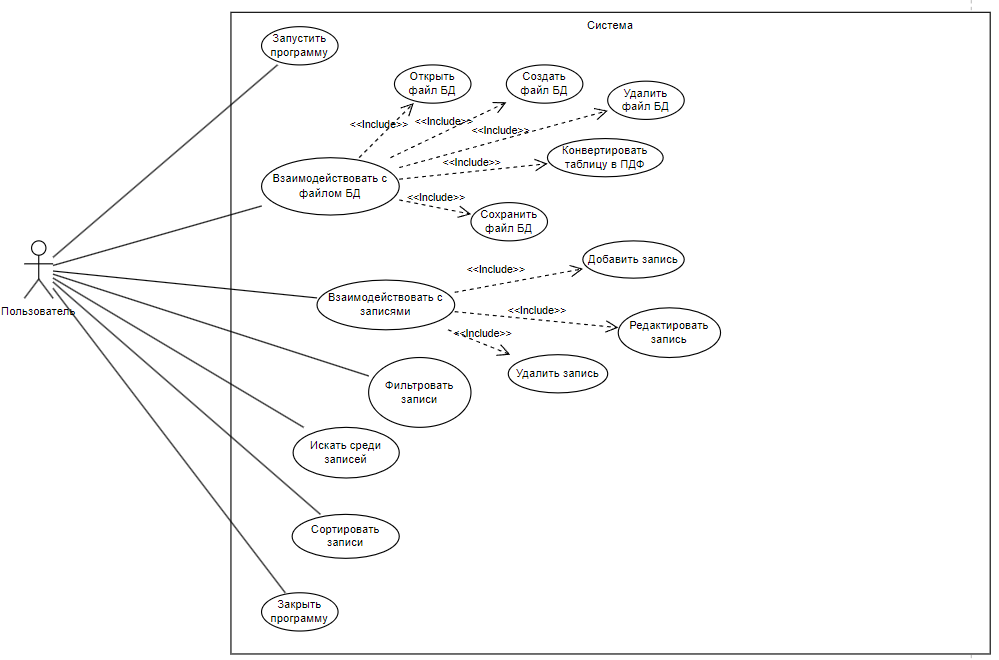


Рисунок 1 – Диаграмма вариантов использования

2 Выбор решения

В качестве решения поставленной задачи выбрано следующее разбиение на программные модули: Furniture – модель данных, представляющая мебель; DatabaseContext – модуль репозитория; ClientForm – модуль пользовательского интерфейса; Program – точка входа в программу; WelcomeForm – приветственная форма. На рисунке 2 представлено взаимодействие классов программы [1].

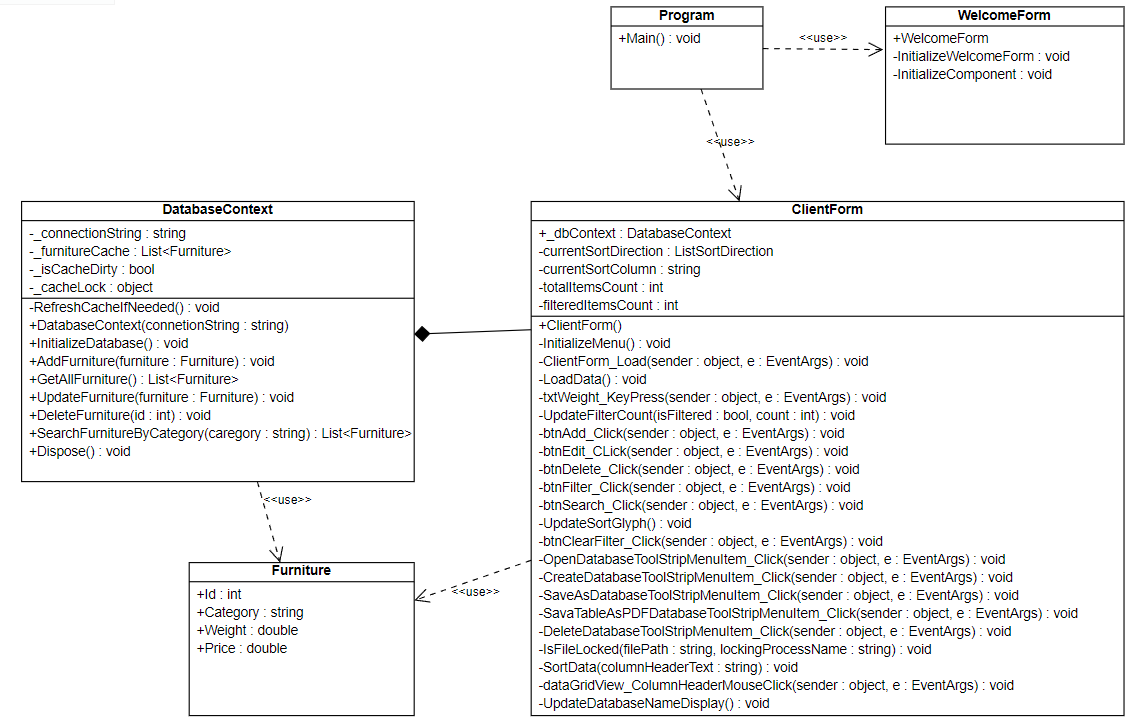


Рисунок 2 – Диаграмма классов

2.1 Определение необходимых модулей программы

Модуль DatabaseContext представляет собой репозиторий, осуществляющий работу с базой данных SQLite, хранящей записи в табличной форме об объектах класса Furniture. Хранит кеш в виде списка объектов Furniture [2]. Содержит методы для добавления, удаления объектов. Код данного модуля приведён на рисунке 3.

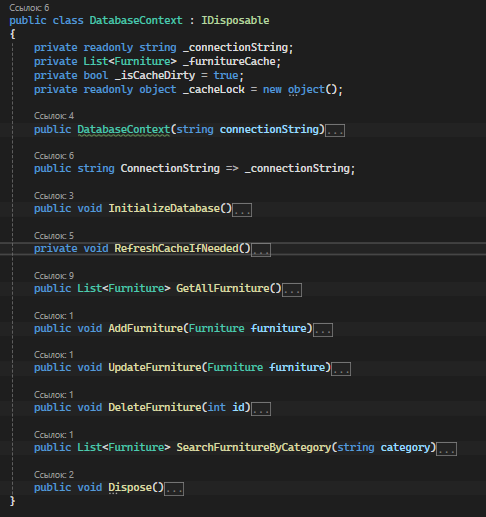


Рисунок 3 – Модуль репозитория

Модуль ClientForm представляет пользовательский интерфейс [3, 4, 5], позволяющий пользователю осуществлять весь задуманный функционал программы. Код данного модуля приведён на рисунке 4.

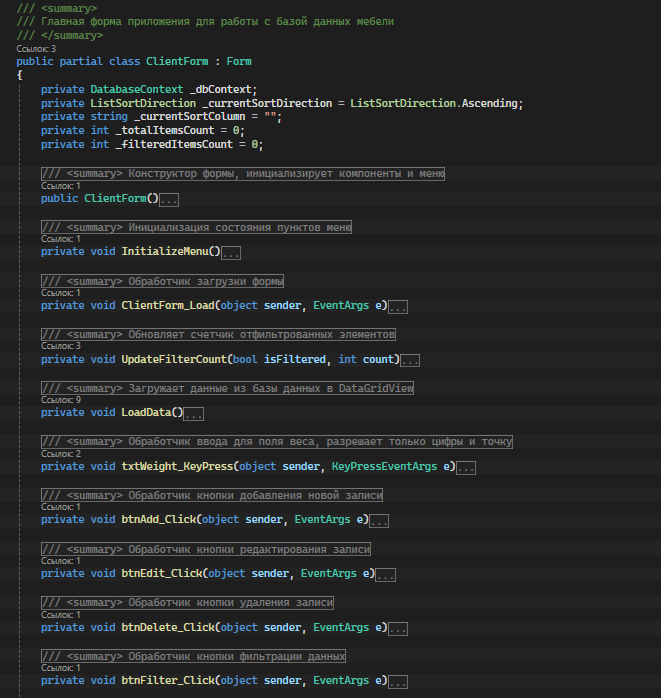


Рисунок 4 – Модуль клиентской формы

Модуль Furniture отвечает за представление данных [6, 7], хранящихся в базе. Содержит поля, характеризующие мебель. Представлен на рисунке 5.

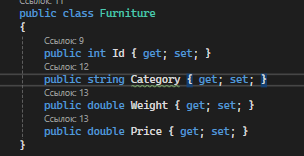


Рисунок 5 – Модуль данных

2.2 Определение структуры файла базы данных

В качестве средства для хранения данных выбрана опосредованная бинарная сериализация в sqlite файл через SQL-команды [8, 9]. Внутри файла формируется таблица, отображающая все записи мебели и их характеристики. Структура файла базы данных представлена на рисунке 6.

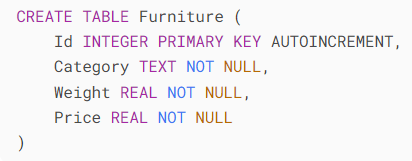
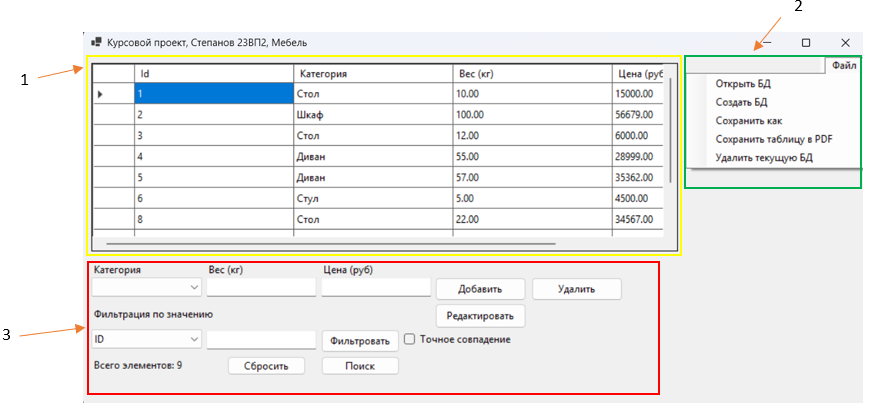


Рисунок 6 – Структура файла базы данных

2.3 Модель интерфейса

Модель интерфейса разрабатываемого приложения представлена на рисунке 7.

Рисунок 7 – Модель интерфейса приложения



Есть три основных блока:

1 – Таблица, отображающая содержимое файла базы данных.

2 – Блок интерфейса для действий с файлами.

3 – Блок интерфейса для действий с записями мебели.

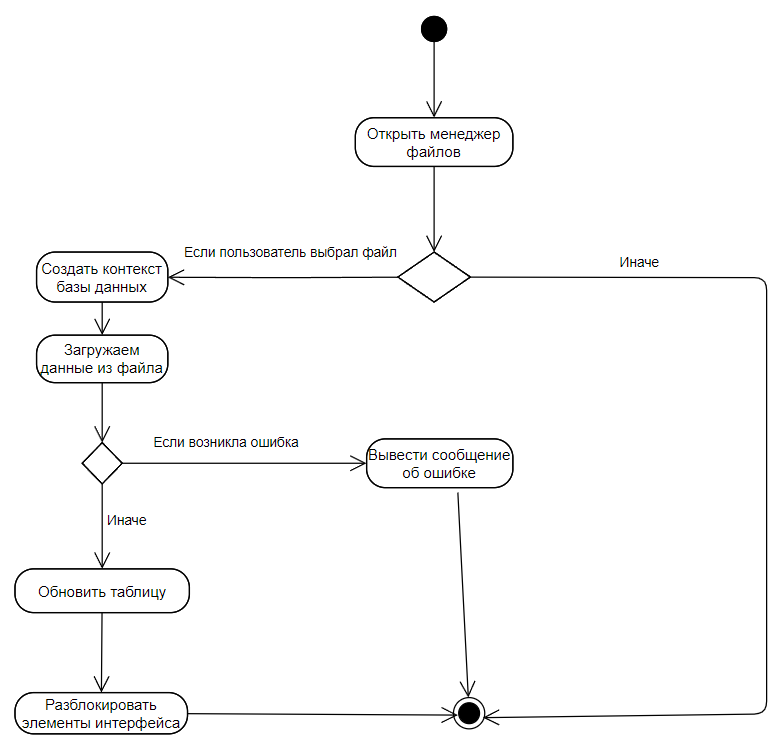
3 Описание разработки программы

3.1 Кодирование

В ходе выполнения курсового проекта было разработано приложение «OOP Cursovaya Stepanov.exe», код которого приведён в приложении А.

Для алгоритма открытия базы данных была разработана диаграмма деятельности [10], представленная на рисунке 8.

Рисунок 8 – Диаграмма деятельности для алгоритма «Открыть базу данных»



Алгоритм, представленный на диаграмме, заключается в том, что пользователь выбирает файл базы данных через диалоговое окно [11, 12]. При успешном выборе создается подключение к SQLite, проверяется структура БД и загружаются данные. Активируются элементы управления интерфейса для работы с данными. В случае ошибки, та логируется и выводится пользователю, подключение сбрасывается.

3.2 Диаграмма компонентов

В процессе выполнения курсового проекта была составлена диаграмма компонентов, которая отображает разбиение программной системы на структурные компоненты и связи между компонентами [13] (рисунок 9).

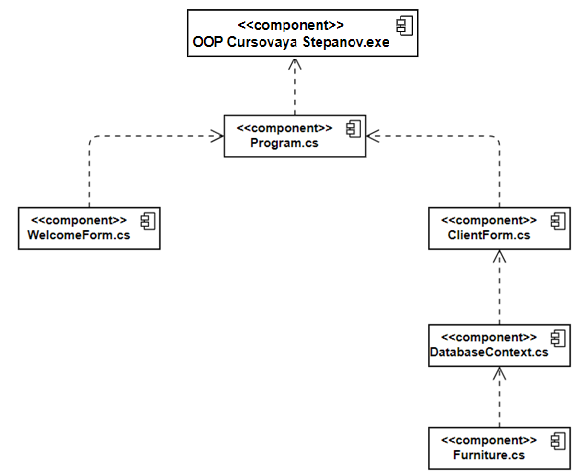


Рисунок 9 – Диаграмма компонентов

Описание компонентов приведено в таблице 1.

Таблица 1 – Компоненты программы

|  |  |
| --- | --- |
| **Название компонента** | **Назначение** |
| OOP Cursovaya Stepanov.exe | Исполняемый файл |
| Program.cs | Точка входа в программу |
| WelcomeForms.cs | Приветственная форма при запуске |
| ClientForm.cs | Форма пользовательского интерфейса |
| DatabaseContext.cs | Файл с классом репозитория |
| Furniture.cs | Файл модели данных |

4 Отладка и тестирование

4.1 Функциональное тестирование

Тестирование программного обеспечения — важный этап разработки, который позволяет выявить ошибки, определить соответствие продукта требованиям и улучшить его качество перед выпуском [14]. Существует множество видов тестирования, каждый из которых выполняет уникальные задачи и применяется на разных этапах жизненного цикла разработки. В рамках курсового проекта решено проводить функциональное тестирование с целью проверки разработанного приложения на соответствие требованиям задания, на правильность реализации функций. Должны быть проверены и продемонстрированы необходимый функционал и обработка исключительных ситуаций.

Функциональное тестирование охватывает не только проверку соответствия системы заявленным требованиям, но и оценку её поведения в реальных условиях эксплуатации [15]. Тестировщики имитируют действия пользователей, проверяя, насколько интуитивно понятным и предсказуемым является интерфейс, а также выявляют возможные несоответствия между ожидаемым и фактическим поведением системы. Особое внимание уделяется граничным условиям и нестандартным сценариям, которые могут привести к сбоям.

Помимо проверки основных функций, функциональное тестирование включает анализ интеграции между различными модулями системы. Важно убедиться, что отдельные компоненты корректно взаимодействуют друг с другом, а данные передаются без искажений.

Ещё одним важным аспектом является проверка обработки ошибок и отказоустойчивости системы. Тестировщики оценивают, насколько понятны и информативны сообщения об ошибках, а также как система восстанавливается после сбоев. Например, корректно ли завершаются транзакции при внезапном прерывании работы или сохраняется ли целостность данных при некорректном вводе.

4.2 Тестирование приложения

В курсовой работе было выполнено функциональное тестирование разработанного программного обеспечения. Результаты тестирования приведены в таблице 2.

Таблица 2 – Проведённые тесты

|  |  |  |
| --- | --- | --- |
| **Состав теста** | **Ожидаемый результат** | **Наблюдаемый результат** |
| 1. Создание файла базы данных. | Система должна создать файл базы данных и открыть его в таблице. | Создан файл базы данных с указанным названием, открыта таблица и интерфейс работы с таблицей разблокирован. (Рисунок 10) |
| 2. Попытка добавить запись в базу данных без заполнения полей. | Система должна распознать пустые поля и вывести пользователю оповещение о необходимости заполнить поля. | Запись не добавлена, так как система распознала пустые поля и вывела пользователю сообщение о том, что необходимо заполнить поля. (Рисунок 11) |
| 3. Добавление записи в базу данных. | Система должна проверить введённые данные и добавить запись. | Введённые данные проверены, запись добавлена в базу данных и выведена в таблицу. (Рисунок 12) |
| 4. Фильтрация по id по значению 5. | Система должна считать выбранную категорию фильтрации и введённое значение, затем провести фильтрацию. | Данные считаны и проведена фильтрация по id по значению 5. (Рисунок 13) |
| 5. Поиск по категории стол. | Система должна считать выбранную категорию поиска и введённое значение, затем провести поиск. | Данные считаны и проведён поиск по категории стол. (Рисунок 14) |
| 6. Уделение позиции №8. | Система должна считать выделенную строку и удалить объект из базы данных и таблицы. | Строка считана верно и удалена из базы данных, таблица обновлена. (Рисунок 15) |
| 7. Редактирование позиции №6. Изменили вес с 5.00 на 5.05. | Система должна распознать выбранную строку, считать новые данные и внести изменения. | Данные изменены в нужной позиции №6. (Рисунок 16) |
| 8. Сохранение текущей таблицы в PDF формат. | Система должна проверить имя PDF файла, создать PDF файл и перенести данные из базы в него. | Создан PDF файл с указанным названием по выбранной базе данных. (Рисунок 17) |
| 9. Сортировка записей по весу по возрастанию. | Система должна считать клик по колонке и отсортировать записи. | Записи корректно отсортированы по весо по возрастанию. (Рисунок 18) |

Ниже на рисунках 10-18 представлены скриншоты проведённых тестов.

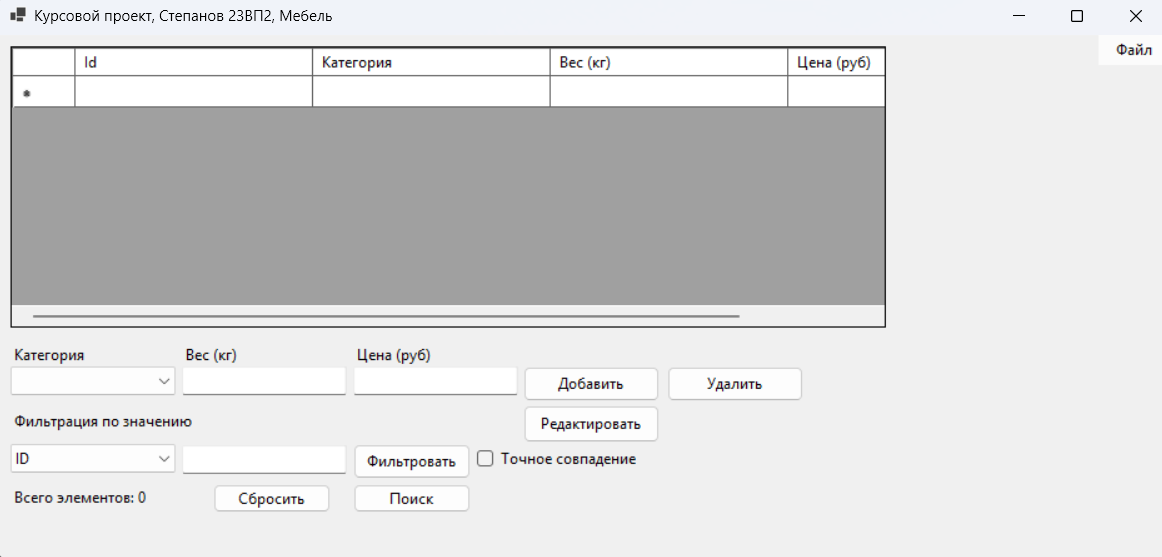


Рисунок 10 – Тест 1, создание файла базы данных

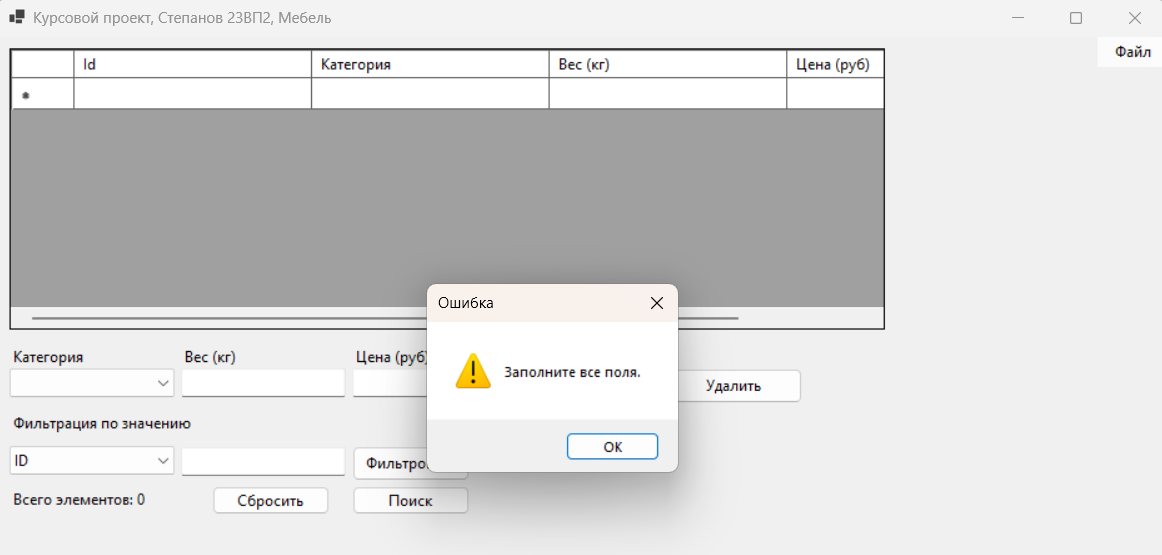


Рисунок 11 – Тест 2, попытка добавления записи без заполнения полей

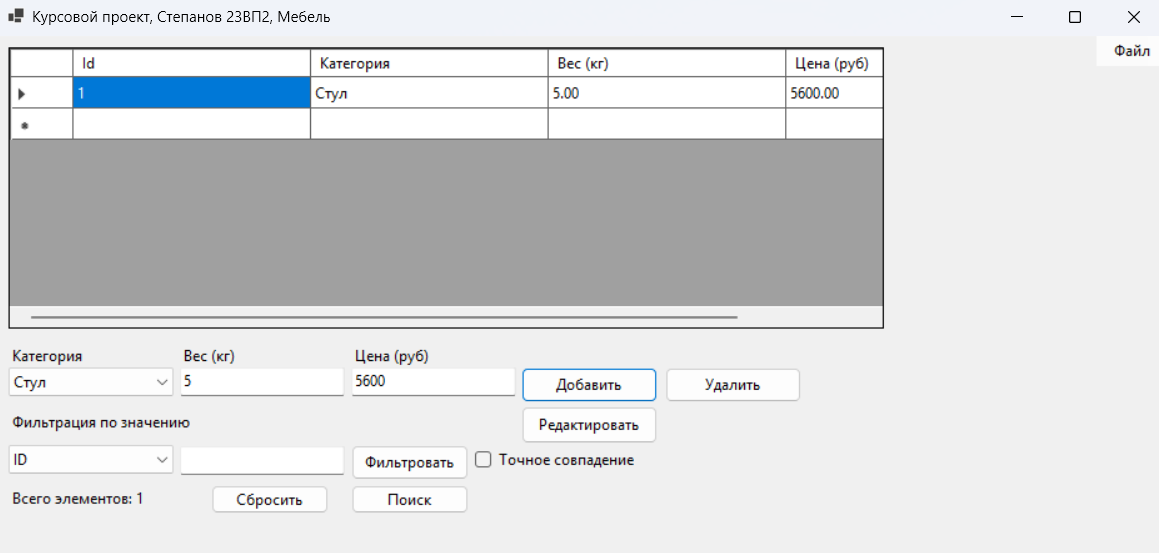


Рисунок 12 – Тест 3, добавление записи в базу данных

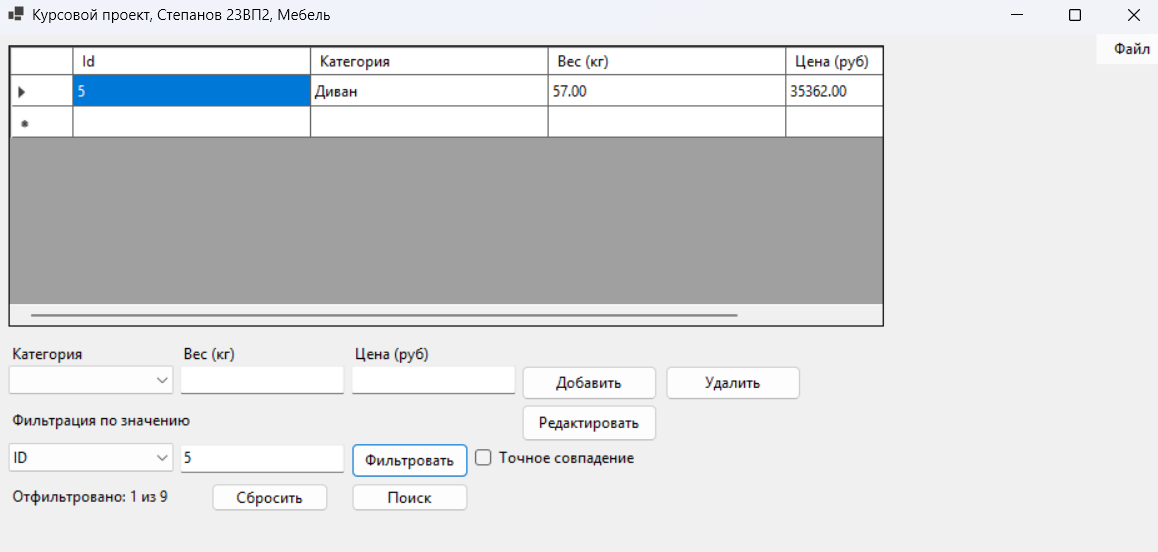


Рисунок 13 – Тест 4, фильтрация по id

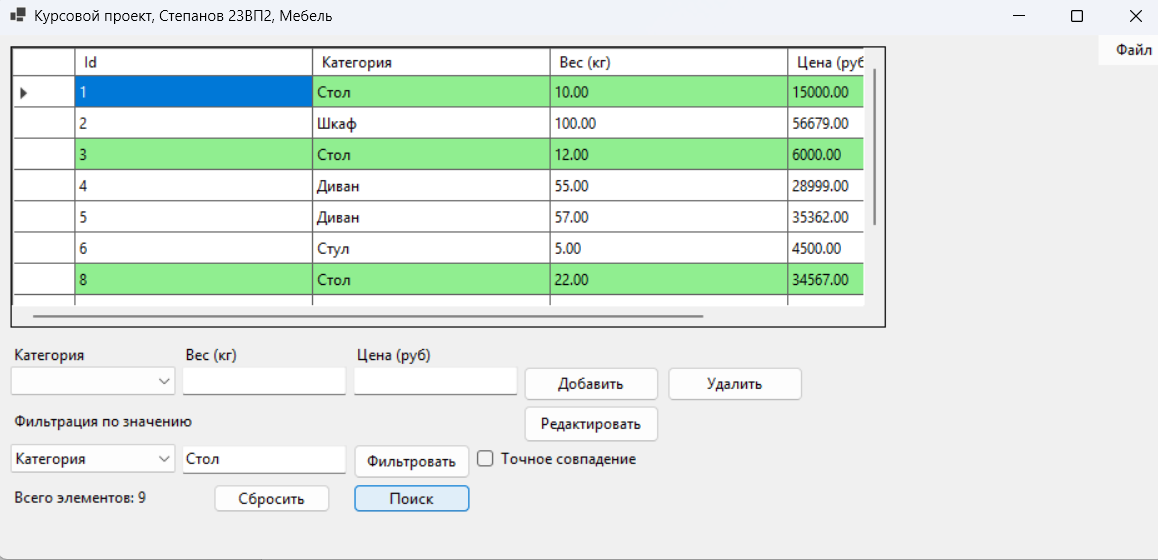


Рисунок 14 – Тест 5, поиск по категории

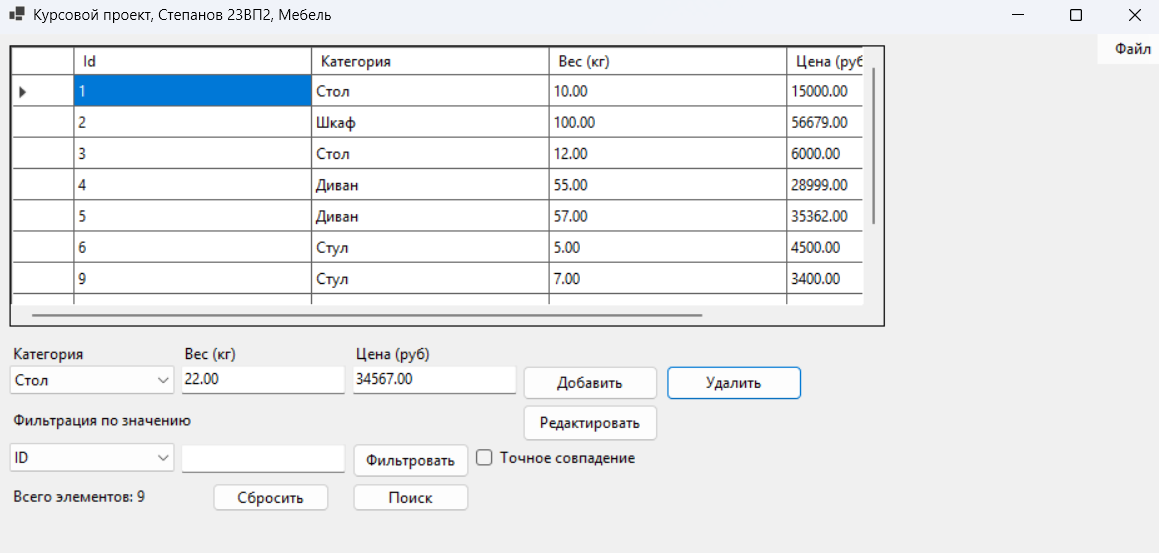


Рисунок 15 – Тест 6, удаление позиции №8

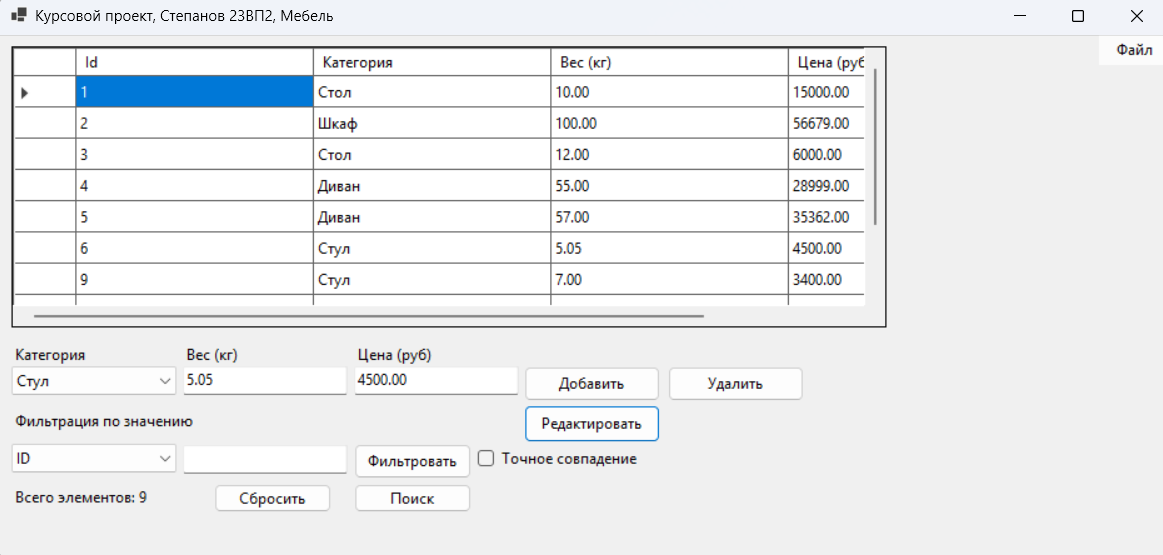


Рисунок 16 – Тест 7, редактирование позиции №6



Рисунок 17 – Тест 8, сохранение таблицы в PDF

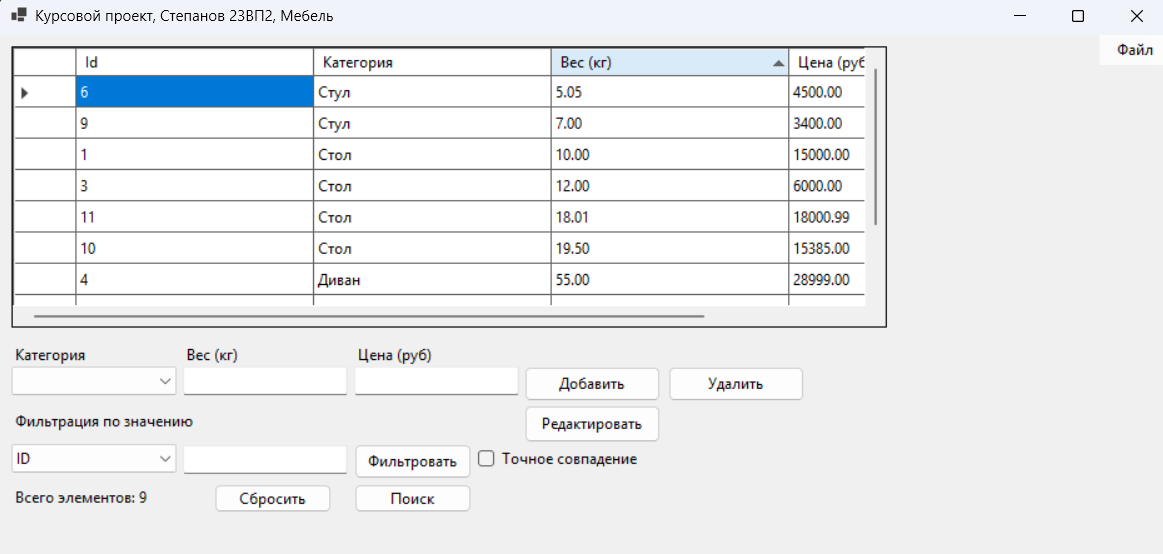


Рисунок 18 – Тест 9, сортировка веса по возрастанию

В ходе выполнения тестирования несовпадения ожидаемого и наблюдаемого результата не выявлены. Следовательно, можно сделать вывод, что программа работает корректно.

5 Описание программы

5.1 Разработака приложения «OOP Cursovaya Stepanov.exe»

Приложение «OOP Cursovaya Stepanov.exe» является основным модулем программы. При запуске программы появляется приветственная форма с информацией об авторе, приложении и кнопкой перехода в основную форму. Если подождать 10 секунд, то также произойдёт переход в основную форму. В основной форме элементы интерфейса скрыты, становятся доступными после открытия файла базы данных. Обработка нажатий на кнопки происходит в файле ClientForm.cs. Описание всех состояний выполнено в руководстве пользователя (Приложение Б).

Выводы

В ходе выполнения курсового проекта была разработана программа «OOP Cursovaya Stepanov.exe», выполняющая поставленную задачу – реализация программы работы с БД «Мебель». Программа соответствует требованиям, поставленным в задании на курсовой проект. Результаты работы программы сохраняются в виде sqlite файлов, содержащих информацию о экземплярах мебели в виде таблицы.

Получен опыт создания проектов в среде Microsoft Visual Studio на Windows Forms на языке C#. Изучен принцип, по которому осуществляется взаимодействие с базой данных.

Список использованных источников

1. Арлоу Дж., Нейштадт А. UML2 и Унифицированный процесс. Практический объектно ориентированный анализ и проектирование. – СПб. : Символ-Плюс, 2007. – 624 с.

2. Программирование динамических структур данных : учеб. пособие / С. В. Самуйлов, С. В. Самуйлова. – Пенза : Изд-во ПГУ, 2024. – 88 с.

3. Васильев А. C#. Объектно-ориентированное программирование: Учебный курс. – СПб.: Питер, 2012. – 320 с.: ил.

4. Герберт Шилдт. С# 3.0. Полное руководство. – Изд. Вильямс, 2010.

5. Троелсен Э. Язык программирования С# 2008 и платформа .NET 3.5 – Изд. Вильямс, 2010.

6. Стиллмен Э., Грин Дж. Изучаем C# [пер. с англ. И. Рузмайкина]. – 2-еизд. - Москва: Питер, 2012. – 694 с. : ил.

7. Гамма Э., Хелм Р., Джонсон Р., Влиссидес Дж. Приемы объектноориентированного программирования. Паттерны проектирования. – СПб: Питер, 2009. – 366 с.: ил.

8. Дейт К. Дж. Введение в системы баз данных. – 8-е изд. – М.: Вильямс, 2006. – 1328 с.

9. Бейли Л. Изучаем SQL. – 3-е изд. – СПб.: БХВ-Петербург, 2020. – 400 с.

10. Фаулер М. UML. Основы. – 3-е изд. – СПб.: Символ-Плюс, 2011. – 192 с.

11. Трей Нэш. C# 2010. Ускоренный курс для профессионалов. – Изд. Вильямс, 2010.

12. Нейгел К., Ивьен Б. и др. С# 2008 и платформа NET 3.5 для профессионалов. – Изд. Диалектика, 2008.

13. Фаулер М. UML. Основы. – 3-е изд. – СПб.: Символ-Плюс, 2011. – 192 с.

14. Куликов С. Тестирование программного обеспечения. Базовый курс. – Минск: Systems, 2020. – 300 с.

15. Куликов С., Александров Л. Тестирование программного обеспечения. Базовый курс. – 3-е изд. – СПб.: БХВ-Петербург, 2022. – 416 с.

Приложение A – Код программы

Код файла Program.cs:

using System;

using System.Windows.Forms;

namespace OOP\_Cursovaya

{

internal static class Program

{

/// <summary>

/// Точка входа в программу

/// </summary>

[STAThread]

static void Main()

{

Application.EnableVisualStyles();

Application.SetCompatibleTextRenderingDefault(false);

// Показываем приветственную форму

using (var welcomeForm = new WelcomeForm())

{

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

{

Application.Run(new ClientForm());

}

}

}

}

}

Код файла Furniture.cs:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OOP\_Cursovaya

{

/// <summary>

/// Класс, представляющий модель мебели в системе.

/// </summary>

public class Furniture

{

/// <summary>

/// Уникальный идентификатор записи о мебели.

/// </summary>

public int Id { get; set; }

/// <summary>

/// Категория мебели (например: "Диван", "Шкаф", "Стол").

/// </summary>

public string? Category { get; set; }

/// <summary>

/// Вес мебели в килограммах.

/// </summary>

public double Weight { get; set; }

/// <summary>

/// Цена мебели в рублях.

/// </summary>

public double Price { get; set; }

}

}

Код файла DatabaseContext.cs:

using System;

using System.Collections.Generic;

using System.Data.SQLite;

using System.Linq;

namespace OOP\_Cursovaya

{

/// <summary>

/// Репозиторий для работы с базой данных мебели, реализующий кэширование данных в памяти

/// </summary>

public class DatabaseContext : IDisposable

{

private readonly string \_connectionString;

private List<Furniture> \_furnitureCache;

private bool \_isCacheDirty = true;

/// <summary>

/// Инициализирует новый экземпляр класса DatabaseContext

/// </summary>

/// <param name="connectionString">Строка подключения к базе данных SQLite</param>

public DatabaseContext(string connectionString)

{

\_connectionString = connectionString;

\_furnitureCache = new List<Furniture>(); // Явная инициализация

InitializeDatabase();

}

/// <summary>

/// Строка подключения к базе данных

/// </summary>

public string ConnectionString => \_connectionString;

/// <summary>

/// Инициализирует базу данных, создавая таблицу Furniture, если она не существует

/// </summary>

/// <exception cref="Exception">Выбрасывается при ошибках инициализации базы данных</exception>

public void InitializeDatabase()

{

try

{

using (var connection = new SQLiteConnection(\_connectionString))

{

connection.Open();

var tableExistsCommand = new SQLiteCommand(

"SELECT name FROM sqlite\_master WHERE type='table' AND name='Furniture';",

connection);

if (tableExistsCommand.ExecuteScalar() == null)

{

var createTableCommand = new SQLiteCommand(

@"CREATE TABLE Furniture (

Id INTEGER PRIMARY KEY AUTOINCREMENT,

Category TEXT NOT NULL,

Weight REAL NOT NULL,

Price REAL NOT NULL

)", connection);

createTableCommand.ExecuteNonQuery();

}

}

\_isCacheDirty = true;

}

catch (Exception ex)

{

throw new Exception("Ошибка инициализации БД", ex);

}

}

/// <summary>

/// Получает все записи о мебели из базы данных

/// </summary>

/// <returns>Список всех объектов мебели</returns>

/// <remarks>Использует кэширование для повышения производительности</remarks>

public List<Furniture> GetAllFurniture()

{

RefreshCacheIfNeeded();

return \_furnitureCache.ToList(); // Возвращаем копию

}

/// <summary>

/// Добавляет новую запись о мебели в базу данных

/// </summary>

/// <param name="furniture">Объект мебели для добавления</param>

/// <remarks>Обновляет кэш после добавления</remarks>

public void AddFurniture(Furniture furniture)

{

using (var connection = new SQLiteConnection(\_connectionString))

using (var command = new SQLiteCommand(

"INSERT INTO Furniture (Category, Weight, Price) VALUES (@Category, @Weight, @Price); " +

"SELECT last\_insert\_rowid();", connection))

{

connection.Open();

command.Parameters.AddWithValue("@Category", furniture.Category);

command.Parameters.AddWithValue("@Weight", furniture.Weight);

command.Parameters.AddWithValue("@Price", furniture.Price);

var newId = Convert.ToInt32(command.ExecuteScalar());

furniture.Id = newId;

// Добавляем в кэш

RefreshCacheIfNeeded();

\_furnitureCache.Add(furniture);

}

}

/// <summary>

/// Обновляет существующую запись о мебели в базе данных

/// </summary>

/// <param name="furniture">Объект мебели с обновленными данными</param>

/// <remarks>Обновляет соответствующую запись в кэше</remarks>

public void UpdateFurniture(Furniture furniture)

{

using (var connection = new SQLiteConnection(\_connectionString))

using (var command = new SQLiteCommand(

"UPDATE Furniture SET Category = @Category, Weight = @Weight, Price = @Price WHERE Id = @Id",

connection))

{

connection.Open();

command.Parameters.AddWithValue("@Category", furniture.Category);

command.Parameters.AddWithValue("@Weight", furniture.Weight);

command.Parameters.AddWithValue("@Price", furniture.Price);

command.Parameters.AddWithValue("@Id", furniture.Id);

command.ExecuteNonQuery();

// Обновляем кэш

RefreshCacheIfNeeded();

var item = \_furnitureCache.FirstOrDefault(f => f.Id == furniture.Id);

if (item != null)

{

item.Category = furniture.Category;

item.Weight = furniture.Weight;

item.Price = furniture.Price;

}

}

}

/// <summary>

/// Удаляет запись о мебели из базы данных по идентификатору

/// </summary>

/// <param name="id">Идентификатор удаляемой записи</param>

/// <remarks>Удаляет соответствующую запись из кэша</remarks>

public void DeleteFurniture(int id)

{

using (var connection = new SQLiteConnection(\_connectionString))

using (var command = new SQLiteCommand(

"DELETE FROM Furniture WHERE Id = @Id",

connection))

{

connection.Open();

command.Parameters.AddWithValue("@Id", id);

command.ExecuteNonQuery();

// Удаляем из кэша

RefreshCacheIfNeeded();

\_furnitureCache.RemoveAll(f => f.Id == id);

}

}

/// <summary>

/// Выполняет поиск мебели по категории

/// </summary>

/// <param name="category">Категория для поиска (регистронезависимо)</param>

/// <returns>Список найденных объектов мебели</returns>

/// <remarks>Поиск выполняется по кэшированным данным</remarks>

public List<Furniture> SearchFurnitureByCategory(string category)

{

RefreshCacheIfNeeded();

return \_furnitureCache

.Where(f => f.Category!.Contains(category, StringComparison.OrdinalIgnoreCase))

.ToList();

}

/// <summary>

/// Освобождает ресурсы, используемые репозиторием

/// </summary>

public void Dispose()

{

try

{

\_furnitureCache?.Clear();

}

catch (Exception ex)

{

throw new Exception("Ошибка при очистке кэша", ex);

}

}

/// <summary>

/// Обновляет кэш данных при необходимости

/// </summary>

private void RefreshCacheIfNeeded()

{

if (!\_isCacheDirty) return;

\_furnitureCache = new List<Furniture>();

using (var connection = new SQLiteConnection(\_connectionString))

using (var command = new SQLiteCommand("SELECT \* FROM Furniture", connection))

{

connection.Open();

using (var reader = command.ExecuteReader())

{

while (reader.Read())

{

\_furnitureCache.Add(new Furniture

{

Id = reader.GetInt32(0),

Category = reader.GetString(1),

Weight = reader.GetDouble(2),

Price = reader.GetDouble(3)

});

}

}

}

\_isCacheDirty = false;

}

}

}

Код файла ClientForm.Designer.cs:

namespace OOP\_Cursovaya

{

partial class ClientForm

{

private System.ComponentModel.IContainer components = null;

private System.Windows.Forms.DataGridView dataGridView;

private System.Windows.Forms.TextBox txtWeight;

private System.Windows.Forms.TextBox txtPrice;

private System.Windows.Forms.Button btnAdd;

private System.Windows.Forms.Button btnEdit;

private System.Windows.Forms.Button btnDelete;

private System.Windows.Forms.ComboBox comboBoxFilter;

private System.Windows.Forms.TextBox txtFilterValue;

private System.Windows.Forms.Button btnFilter;

private System.Windows.Forms.Button btnClearFilter;

private System.Windows.Forms.Label lblCategory;

private System.Windows.Forms.Label lblPrice;

private System.Windows.Forms.Label lblWeight;

private System.Windows.Forms.ComboBox comboBoxCategory;

private System.Windows.Forms.MenuStrip menuStrip;

private System.Windows.Forms.ToolStripMenuItem fileToolStripMenuItem;

private System.Windows.Forms.ToolStripMenuItem openDatabaseToolStripMenuItem;

private System.Windows.Forms.ToolStripMenuItem createDatabaseToolStripMenuItem;

private System.Windows.Forms.ToolStripMenuItem deleteDatabaseToolStripMenuItem;

private System.Windows.Forms.Label lblFilteredCount;

/// <summary>

/// Освобождает неуправляемые ресурсы, используемые формой, а при необходимости — также управляемые ресурсы.

/// </summary>

/// <param name="disposing">

/// Значение <see langword="true"/> указывает, что следует освободить как управляемые, так и неуправляемые ресурсы;

/// Значение <see langword="false"/> означает, что нужно освободить только неуправляемые ресурсы.

/// </param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

private void InitializeComponent()

{

System.ComponentModel.ComponentResourceManager resources = new System.ComponentModel.ComponentResourceManager(typeof(ClientForm));

dataGridView = new DataGridView();

Id = new DataGridViewTextBoxColumn();

Category = new DataGridViewTextBoxColumn();

Weight = new DataGridViewTextBoxColumn();

Price = new DataGridViewTextBoxColumn();

lblCategory = new Label();

lblWeight = new Label();

lblPrice = new Label();

comboBoxCategory = new ComboBox();

txtWeight = new TextBox();

txtPrice = new TextBox();

btnAdd = new Button();

btnEdit = new Button();

btnDelete = new Button();

comboBoxFilter = new ComboBox();

txtFilterValue = new TextBox();

btnFilter = new Button();

btnClearFilter = new Button();

menuStrip = new MenuStrip();

fileToolStripMenuItem = new ToolStripMenuItem();

openDatabaseToolStripMenuItem = new ToolStripMenuItem();

createDatabaseToolStripMenuItem = new ToolStripMenuItem();

saveAsDatabaseToolStripMenuItem = new ToolStripMenuItem();

SaveTableAsPDFToolStripMenuItem = new ToolStripMenuItem();

deleteDatabaseToolStripMenuItem = new ToolStripMenuItem();

labelFilterByValue = new Label();

lblFilteredCount = new Label();

buttonSearch = new Button();

checkBoxExactMatch = new CheckBox();

lblDatabaseName = new Label();

((System.ComponentModel.ISupportInitialize)dataGridView).BeginInit();

menuStrip.SuspendLayout();

SuspendLayout();

//

// dataGridView

//

dataGridView.ColumnHeadersHeightSizeMode = DataGridViewColumnHeadersHeightSizeMode.AutoSize;

dataGridView.Columns.AddRange(new DataGridViewColumn[] { Id, Category, Weight, Price });

dataGridView.Location = new Point(12, 45);

dataGridView.Name = "dataGridView";

dataGridView.ReadOnly = true;

dataGridView.RowHeadersWidth = 51;

dataGridView.Size = new Size(922, 300);

dataGridView.TabIndex = 0;

dataGridView.ColumnHeaderMouseClick += dataGridView\_ColumnHeaderMouseClick;

//

// Id

//

Id.HeaderText = "Id";

Id.MinimumWidth = 6;

Id.Name = "Id";

Id.ReadOnly = true;

Id.Width = 190;

//

// Category

//

Category.HeaderText = "Категория";

Category.MinimumWidth = 6;

Category.Name = "Category";

Category.ReadOnly = true;

Category.Width = 190;

//

// Weight

//

Weight.HeaderText = "Вес (кг)";

Weight.MinimumWidth = 6;

Weight.Name = "Weight";

Weight.ReadOnly = true;

Weight.Width = 190;

//

// Price

//

Price.HeaderText = "Цена (руб)";

Price.MinimumWidth = 6;

Price.Name = "Price";

Price.ReadOnly = true;

Price.Width = 190;

//

// lblCategory

//

lblCategory.AutoSize = true;

lblCategory.Location = new Point(13, 358);

lblCategory.Name = "lblCategory";

lblCategory.Size = new Size(81, 20);

lblCategory.TabIndex = 1;

lblCategory.Text = "Категория";

//

// lblWeight

//

lblWeight.AutoSize = true;

lblWeight.Location = new Point(169, 358);

lblWeight.Name = "lblWeight";

lblWeight.Size = new Size(60, 20);

lblWeight.TabIndex = 3;

lblWeight.Text = "Вес (кг)";

//

// lblPrice

//

lblPrice.AutoSize = true;

lblPrice.Location = new Point(325, 358);

lblPrice.Name = "lblPrice";

lblPrice.Size = new Size(84, 20);

lblPrice.TabIndex = 5;

lblPrice.Text = "Цена (руб)";

//

// comboBoxCategory

//

comboBoxCategory.DropDownStyle = ComboBoxStyle.DropDownList;

comboBoxCategory.Enabled = false;

comboBoxCategory.FormattingEnabled = true;

comboBoxCategory.Items.AddRange(new object[] { "Стул", "Стол", "Шкаф", "Диван" });

comboBoxCategory.SelectedIndex = 0;

comboBoxCategory.Location = new Point(13, 381);

comboBoxCategory.Name = "comboBoxCategory";

comboBoxCategory.Size = new Size(150, 28);

comboBoxCategory.TabIndex = 2;

//

// txtWeight

//

txtWeight.Enabled = false;

txtWeight.Location = new Point(169, 381);

txtWeight.Name = "txtWeight";

txtWeight.Size = new Size(150, 27);

txtWeight.TabIndex = 4;

txtWeight.KeyPress += txtWeight\_KeyPress;

//

// txtPrice

//

txtPrice.Enabled = false;

txtPrice.Location = new Point(325, 381);

txtPrice.Name = "txtPrice";

txtPrice.Size = new Size(150, 27);

txtPrice.TabIndex = 6;

txtPrice.KeyPress += txtWeight\_KeyPress;

//

// btnAdd

//

btnAdd.Enabled = false;

btnAdd.Location = new Point(481, 381);

btnAdd.Name = "btnAdd";

btnAdd.Size = new Size(125, 37);

btnAdd.TabIndex = 7;

btnAdd.Text = "Добавить";

btnAdd.Click += btnAdd\_Click;

//

// btnEdit

//

btnEdit.Enabled = false;

btnEdit.Location = new Point(481, 422);

btnEdit.Name = "btnEdit";

btnEdit.Size = new Size(125, 40);

btnEdit.TabIndex = 8;

btnEdit.Text = "Редактировать";

btnEdit.Click += btnEdit\_Click;

//

// btnDelete

//

btnDelete.Enabled = false;

btnDelete.Location = new Point(612, 381);

btnDelete.Name = "btnDelete";

btnDelete.Size = new Size(125, 37);

btnDelete.TabIndex = 9;

btnDelete.Text = "Удалить";

btnDelete.Click += btnDelete\_Click;

//

// comboBoxFilter

//

comboBoxFilter.DropDownStyle = ComboBoxStyle.DropDownList;

comboBoxFilter.Enabled = false;

comboBoxFilter.FormattingEnabled = true;

comboBoxFilter.Items.AddRange(new object[] { "ID", "Категория", "Вес", "Цена" });

comboBoxFilter.Location = new Point(12, 499);

comboBoxFilter.Name = "comboBoxFilter";

comboBoxFilter.Size = new Size(150, 28);

comboBoxFilter.TabIndex = 10;

//

// txtFilterValue

//

txtFilterValue.Enabled = false;

txtFilterValue.Location = new Point(168, 500);

txtFilterValue.Name = "txtFilterValue";

txtFilterValue.Size = new Size(150, 27);

txtFilterValue.TabIndex = 11;

//

// btnFilter

//

btnFilter.Enabled = false;

btnFilter.Location = new Point(324, 499);

btnFilter.Name = "btnFilter";

btnFilter.Size = new Size(108, 37);

btnFilter.TabIndex = 12;

btnFilter.Text = "Фильтровать";

btnFilter.Click += btnFilter\_Click;

//

// btnClearFilter

//

btnClearFilter.Enabled = false;

btnClearFilter.Location = new Point(196, 542);

btnClearFilter.Name = "btnClearFilter";

btnClearFilter.Size = new Size(108, 31);

btnClearFilter.TabIndex = 13;

btnClearFilter.Text = "Сбросить фильтр";

btnClearFilter.Click += btnClearFilter\_Click;

//

// menuStrip

//

menuStrip.Anchor = AnchorStyles.Top | AnchorStyles.Right;

menuStrip.Dock = DockStyle.None;

menuStrip.ImageScalingSize = new Size(20, 20);

menuStrip.Items.AddRange(new ToolStripItem[] { fileToolStripMenuItem });

menuStrip.LayoutStyle = ToolStripLayoutStyle.HorizontalStackWithOverflow;

menuStrip.Location = new Point(1119, 0);

menuStrip.Name = "menuStrip";

menuStrip.Size = new Size(67, 28);

menuStrip.TabIndex = 14;

menuStrip.Text = "menuStrip";

//

// fileToolStripMenuItem

//

fileToolStripMenuItem.DropDownDirection = ToolStripDropDownDirection.BelowLeft;

fileToolStripMenuItem.DropDownItems.AddRange(new ToolStripItem[] { openDatabaseToolStripMenuItem, createDatabaseToolStripMenuItem, saveAsDatabaseToolStripMenuItem, SaveTableAsPDFToolStripMenuItem, deleteDatabaseToolStripMenuItem });

fileToolStripMenuItem.Name = "fileToolStripMenuItem";

fileToolStripMenuItem.Size = new Size(59, 24);

fileToolStripMenuItem.Text = "Файл";

//

// openDatabaseToolStripMenuItem

//

openDatabaseToolStripMenuItem.Name = "openDatabaseToolStripMenuItem";

openDatabaseToolStripMenuItem.Size = new Size(268, 26);

openDatabaseToolStripMenuItem.Text = "Открыть БД";

openDatabaseToolStripMenuItem.Click += OpenDatabaseToolStripMenuItem\_Click;

//

// createDatabaseToolStripMenuItem

//

createDatabaseToolStripMenuItem.Name = "createDatabaseToolStripMenuItem";

createDatabaseToolStripMenuItem.Size = new Size(268, 26);

createDatabaseToolStripMenuItem.Text = "Создать БД";

createDatabaseToolStripMenuItem.Click += CreateDatabaseToolStripMenuItem\_Click;

//

// saveAsDatabaseToolStripMenuItem

//

saveAsDatabaseToolStripMenuItem.Name = "saveAsDatabaseToolStripMenuItem";

saveAsDatabaseToolStripMenuItem.Size = new Size(268, 26);

saveAsDatabaseToolStripMenuItem.Text = "Сохранить как";

saveAsDatabaseToolStripMenuItem.Click += SaveAsDatabaseToolStripMenuItem\_Click;

//

// SaveTableAsPDFToolStripMenuItem

//

SaveTableAsPDFToolStripMenuItem.Name = "SaveTableAsPDFToolStripMenuItem";

SaveTableAsPDFToolStripMenuItem.Size = new Size(268, 26);

SaveTableAsPDFToolStripMenuItem.Text = "Сохранить таблицу в PDF";

SaveTableAsPDFToolStripMenuItem.Click += SaveTableAsPDFToolStripMenuItem\_Click;

//

// deleteDatabaseToolStripMenuItem

//

deleteDatabaseToolStripMenuItem.Enabled = false;

deleteDatabaseToolStripMenuItem.Name = "deleteDatabaseToolStripMenuItem";

deleteDatabaseToolStripMenuItem.Size = new Size(268, 26);

deleteDatabaseToolStripMenuItem.Text = "Удалить текущую БД";

deleteDatabaseToolStripMenuItem.Click += DeleteDatabaseToolStripMenuItem\_Click;

//

// labelFilterByValue

//

labelFilterByValue.AutoSize = true;

labelFilterByValue.Location = new Point(12, 464);

labelFilterByValue.Name = "labelFilterByValue";

labelFilterByValue.Size = new Size(190, 20);

labelFilterByValue.TabIndex = 17;

labelFilterByValue.Text = "Фильтрация по значению";

//

// lblFilteredCount

//

lblFilteredCount.AutoSize = true;

lblFilteredCount.Location = new Point(12, 545);

lblFilteredCount.Name = "lblFilteredCount";

lblFilteredCount.Size = new Size(141, 20);

lblFilteredCount.TabIndex = 18;

lblFilteredCount.Text = "Всего элементов: 0";

//

// buttonSearch

//

buttonSearch.Enabled = false;

buttonSearch.Location = new Point(324, 542);

buttonSearch.Name = "buttonSearch";

buttonSearch.Size = new Size(108, 31);

buttonSearch.TabIndex = 19;

buttonSearch.Text = "Поиск";

buttonSearch.Click += ButtonSearch\_Click;

//

// checkBoxExactMatch

//

checkBoxExactMatch.AutoSize = true;

checkBoxExactMatch.Enabled = false;

checkBoxExactMatch.Location = new Point(438, 503);

checkBoxExactMatch.Name = "checkBoxExactMatch";

checkBoxExactMatch.Size = new Size(169, 24);

checkBoxExactMatch.TabIndex = 20;

checkBoxExactMatch.Text = "Точное совпадение";

checkBoxExactMatch.UseVisualStyleBackColor = true;

//

// lblDatabaseName

//

lblDatabaseName.AutoSize = true;

lblDatabaseName.Location = new Point(12, 9);

lblDatabaseName.Name = "lblDatabaseName";

lblDatabaseName.Size = new Size(93, 20);

lblDatabaseName.TabIndex = 21;

lblDatabaseName.Text = "Текущая БД:";

//

// ClientForm

//

AutoScaleDimensions = new SizeF(8F, 20F);

AutoScaleMode = AutoScaleMode.Font;

ClientSize = new Size(1186, 618);

Controls.Add(lblDatabaseName);

Controls.Add(checkBoxExactMatch);

Controls.Add(buttonSearch);

Controls.Add(labelFilterByValue);

Controls.Add(btnClearFilter);

Controls.Add(btnFilter);

Controls.Add(txtFilterValue);

Controls.Add(comboBoxFilter);

Controls.Add(btnDelete);

Controls.Add(btnEdit);

Controls.Add(btnAdd);

Controls.Add(txtPrice);

Controls.Add(lblPrice);

Controls.Add(txtWeight);

Controls.Add(lblWeight);

Controls.Add(comboBoxCategory);

Controls.Add(lblCategory);

Controls.Add(dataGridView);

Controls.Add(menuStrip);

Controls.Add(lblFilteredCount);

Icon = (Icon)resources.GetObject("$this.Icon");

MainMenuStrip = menuStrip;

Name = "ClientForm";

Text = "Курсовой проект, Степанов 23ВП2, Мебель";

Load += ClientForm\_Load;

((System.ComponentModel.ISupportInitialize)dataGridView).EndInit();

menuStrip.ResumeLayout(false);

menuStrip.PerformLayout();

ResumeLayout(false);

PerformLayout();

}

private ToolStripMenuItem saveAsDatabaseToolStripMenuItem;

private ToolStripMenuItem SaveTableAsPDFToolStripMenuItem;

private Label labelFilterByValue;

private Button buttonSearch;

private CheckBox checkBoxExactMatch;

private Label lblDatabaseName;

private DataGridViewTextBoxColumn Id;

private DataGridViewTextBoxColumn Category;

private DataGridViewTextBoxColumn Weight;

private DataGridViewTextBoxColumn Price;

}

}

Код файла ClientForm.cs:

using System;

using System.Collections.Generic;

using System.Data.SQLite;

using System.Windows.Forms;

using System.Runtime.InteropServices;

using System.Diagnostics;

using System.IO;

using QuestPDF.Fluent;

using QuestPDF.Helpers;

using QuestPDF.Infrastructure;

using System.Globalization;

using System.ComponentModel;

using System.Linq;

using System.Diagnostics.CodeAnalysis;

namespace OOP\_Cursovaya

{

/// <summary>

/// Главная форма приложения для работы с базой данных мебели

/// </summary>

public partial class ClientForm : Form

{

private DatabaseContext? \_dbContext;

private ListSortDirection \_currentSortDirection = ListSortDirection.Ascending;

private string \_currentSortColumn = string.Empty;

private int \_totalItemsCount = 0;

private int \_filteredItemsCount = 0;

private bool isEnabled = false;

/// <summary>

/// Конструктор формы, инициализирует компоненты и меню

/// </summary>

public ClientForm()

{

InitializeComponent();

InitializeMenu();

dataGridView.SelectionChanged += dataGridView\_SelectionChanged!;

}

/// <summary>

/// Инициализация состояния пунктов меню

/// </summary>

private void InitializeMenu()

{

deleteDatabaseToolStripMenuItem.Enabled = false; // Удаление недоступно

saveTableAsPDFToolStripMenuItem.Enabled = false; // Сохранение в ПДФ недоступно

saveAsDatabaseToolStripMenuItem.Enabled = false; // Сохранение как недоступно

openDatabaseToolStripMenuItem.Enabled = true; // Открытие доступно

createDatabaseToolStripMenuItem.Enabled = true; // Создание доступно

}

/// <summary>

/// Обработчик загрузки формы

/// </summary>

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

{

comboBoxFilter.SelectedIndex = 0;

// Загружаем данные только если база данных открыта

if (\_dbContext != null)

{

LoadData();

}

}

/// <summary>

/// Обновляет счетчик отфильтрованных элементов

/// </summary>

/// <param name="isFiltered">Флаг, указывающий применен ли фильтр</param>

/// <param name="count">Количество элементов</param>

private void UpdateFilterCount(bool isFiltered, int count)

{

if (isFiltered)

{

lblFilteredCount.Text = $"Отфильтровано: {count} из {\_totalItemsCount}";

}

else

{

lblFilteredCount.Text = $"Всего элементов: {count}";

}

}

/// <summary>

/// Загружает данные из базы данных в DataGridView

/// </summary>

private void LoadData()

{

if (\_dbContext == null)

{

dataGridView.Rows.Clear();

UpdateFilterCount(false, 0);

return;

}

try

{

// Получаем все записи из базы данных

var furnitureList = \_dbContext.GetAllFurniture();

\_totalItemsCount = furnitureList.Count;

\_filteredItemsCount = \_totalItemsCount;

dataGridView.Rows.Clear();

// Заполняем DataGridView данными

foreach (var furniture in furnitureList)

{

string weight = furniture.Weight.ToString("F2", CultureInfo.InvariantCulture);

string price = furniture.Price.ToString("F2", CultureInfo.InvariantCulture);

dataGridView.Rows.Add(furniture.Id, furniture.Category, weight, price);

}

UpdateFilterCount(false, \_totalItemsCount);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при загрузке данных: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

/// <summary>

/// Обработчик ввода для поля веса, разрешает только цифры и точку

/// </summary>

private void txtWeight\_KeyPress(object sender, KeyPressEventArgs e)

{

// Проверяем, что sender действительно TextBox

if (sender is not TextBox textBox)

return;

// Комбинированная проверка:

// 1. Если символ не цифра, не точка и не управляющий символ

// 2. ИЛИ если точка, но она уже есть в тексте

// 3. ИЛИ если точка вводится до ввода первой цифры

if ((!char.IsDigit(e.KeyChar) && e.KeyChar != '.' && !char.IsControl(e.KeyChar)) ||

(e.KeyChar == '.' && (textBox.Text.Contains(".") || textBox.SelectionStart == 0)))

{

e.Handled = true;

}

}

/// <summary>

/// Обработчик кнопки добавления новой записи

/// </summary>

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

{

// Проверяем заполнение обязательных полей

if (comboBoxCategory.SelectedItem == null || string.IsNullOrEmpty(txtWeight.Text) || string.IsNullOrEmpty(txtPrice.Text))

{

MessageBox.Show("Заполните все поля.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

try

{

// Создаем новый объект мебели

var furniture = new Furniture

{

Category = comboBoxCategory.SelectedItem?.ToString() ?? string.Empty,

Weight = double.Parse(txtWeight.Text, CultureInfo.InvariantCulture),

Price = double.Parse(txtPrice.Text, CultureInfo.InvariantCulture)

};

// Добавляем запись в базу данных

\_dbContext?.AddFurniture(furniture);

LoadData(); // Обновляем отображение данных

}

catch (FormatException)

{

MessageBox.Show("Некорректный формат числа. Используйте точку как разделитель.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при добавлении записи: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

/// <summary>

/// Обрабатывает изменение выбранной строки в DataGridView, обновляя соответствующие элементы управления.

/// </summary>

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

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

/// <remarks>

/// Заполняет комбо-бокс категории и текстовые поля данными из выбранной строки.

/// </remarks>

private void dataGridView\_SelectionChanged( object sender, EventArgs e)

{

if (dataGridView.SelectedRows.Count == 0)

return;

var row = dataGridView.SelectedRows[0];

comboBoxCategory.SelectedItem = row.Cells["Category"]?.Value?.ToString();

txtWeight.Text = row.Cells["Weight"]?.Value?.ToString();

txtPrice.Text = row.Cells["Price"]?.Value?.ToString();

}

/// <summary>

/// Обрабатывает нажатие кнопки редактирования, обновляя выбранную запись в базе данных.

/// </summary>

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

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

/// <remarks>

/// Проверяет корректность введенных данных перед обновлением записи.

/// В случае ошибки отображает соответствующее сообщение.

/// </remarks>

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

{

if (dataGridView.SelectedRows.Count == 0)

{

MessageBox.Show("Выберите строку для редактирования.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

var selectedRow = dataGridView.SelectedRows[0];

try

{

var furniture = new Furniture

{

Id = (int)selectedRow.Cells["Id"].Value,

Category = comboBoxCategory.SelectedItem?.ToString() ?? string.Empty,

Weight = double.Parse(txtWeight.Text, CultureInfo.InvariantCulture),

Price = double.Parse(txtPrice.Text, CultureInfo.InvariantCulture)

};

\_dbContext?.UpdateFurniture(furniture);

LoadData();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при обновлении записи: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

/// <summary>

/// Обработчик кнопки удаления записи

/// </summary>

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

{

if (dataGridView.SelectedRows.Count > 0)

{

var selectedRow = dataGridView.SelectedRows[0];

// Проверяем, что строка не является пустой или заголовком

if (selectedRow.IsNewRow)

{

MessageBox.Show("Выберите строку для удаления.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

// Получаем ID записи для удаления

int id = (int)selectedRow.Cells[0].Value;

// Удаляем запись из базы данных

\_dbContext?.DeleteFurniture(id);

LoadData(); // Обновляем отображение данных

}

else

{

MessageBox.Show("Выберите строку для удаления.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

}

}

/// <summary>

/// Обработчик кнопки фильтрации данных

/// </summary>

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

{

string filterValue = txtFilterValue.Text;

string filterColumn = comboBoxFilter.SelectedItem?.ToString() ?? string.Empty;

// Инициализируем пустым списком по умолчанию

var filteredList = new List<Furniture>();

try

{

// Проверяем, что контекст БД существует

if (\_dbContext == null)

{

MessageBox.Show("Ошибка: контекст базы данных не инициализирован", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

// Получаем все записи один раз

var allFurniture = \_dbContext.GetAllFurniture() ?? Enumerable.Empty<Furniture>();

// Применяем фильтр в зависимости от выбранного столбца

switch (filterColumn)

{

case "ID":

if (checkBoxExactMatch.Checked)

{

if (int.TryParse(filterValue, out int id))

{

filteredList = allFurniture

.Where(f => f.Id == id)

.ToList();

}

}

else

{

filteredList = allFurniture

.Where(f => f.Id.ToString().Contains(filterValue))

.ToList();

}

break;

case "Категория":

if (checkBoxExactMatch.Checked)

{

filteredList = allFurniture

.Where(f => f.Category!.Equals(filterValue, StringComparison.OrdinalIgnoreCase))

.ToList();

}

else

{

filteredList = \_dbContext.SearchFurnitureByCategory(filterValue) ?? new List<Furniture>();

}

break;

case "Вес":

if (checkBoxExactMatch.Checked)

{

if (double.TryParse(filterValue, NumberStyles.Any, CultureInfo.InvariantCulture, out double weight))

{

filteredList = allFurniture

.Where(f => Math.Abs(f.Weight - weight) < 0.0001)

.ToList();

}

}

else

{

filteredList = allFurniture

.Where(f => f.Weight.ToString(CultureInfo.InvariantCulture).Contains(filterValue))

.ToList();

}

break;

case "Цена":

if (checkBoxExactMatch.Checked)

{

if (double.TryParse(filterValue, NumberStyles.Any, CultureInfo.InvariantCulture, out double price))

{

filteredList = allFurniture

.Where(f => Math.Abs(f.Price - price) < 0.0001)

.ToList();

}

}

else

{

filteredList = allFurniture

.Where(f => f.Price.ToString(CultureInfo.InvariantCulture).Contains(filterValue))

.ToList();

}

break;

default:

MessageBox.Show(

$"Неизвестный столбец для фильтрации: {filterColumn}",

"Ошибка",

MessageBoxButtons.OK,

MessageBoxIcon.Error);

return;

}

// Обновляем DataGridView

dataGridView.Rows.Clear();

foreach (var furniture in filteredList)

{

dataGridView.Rows.Add(

furniture.Id,

furniture.Category,

furniture.Weight.ToString("F2", CultureInfo.InvariantCulture),

furniture.Price.ToString("F2", CultureInfo.InvariantCulture)

);

}

\_filteredItemsCount = filteredList.Count;

UpdateFilterCount(true, \_filteredItemsCount);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при фильтрации: {ex.Message}", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

/// <summary>

/// Обработчик кнопки поиска данных

/// </summary>

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

{

// Получаем параметры поиска

string filterColumnText = comboBoxFilter.SelectedItem?.ToString() ?? string.Empty;

string filterValue = txtFilterValue.Text.Trim();

if (string.IsNullOrEmpty(filterColumnText) || string.IsNullOrEmpty(filterValue))

{

MessageBox.Show("Выберите столбец для фильтрации и введите значение.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

try

{

// Сбрасываем предыдущее выделение

foreach (DataGridViewRow row in dataGridView.Rows)

{

foreach (DataGridViewCell cell in row.Cells)

{

cell.Style.BackColor = System.Drawing.Color.White;

}

}

// Определяем имя столбца для поиска

string filterColumnName = string.Empty;

switch (filterColumnText)

{

case "ID":

filterColumnName = "Id";

break;

case "Категория":

filterColumnName = "Category";

break;

case "Вес":

filterColumnName = "Weight";

break;

case "Цена":

filterColumnName = "Price";

break;

default:

MessageBox.Show($"Столбец '{filterColumnText}' не найден в таблице.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

int foundCount = 0; // Счетчик найденных элементов

// Выполняем поиск и выделяем совпадения

foreach (DataGridViewRow row in dataGridView.Rows)

{

if (row.IsNewRow) continue; // Пропускаем новую строку

var cellValue = row.Cells[filterColumnName]?.Value?.ToString() ?? string.Empty;

bool isMatch = false;

if (checkBoxExactMatch.Checked)

{

// Точное совпадение

if ((filterColumnText == "Вес" || filterColumnText == "Цена") &&

double.TryParse(cellValue, NumberStyles.Any, CultureInfo.InvariantCulture, out double cellDouble) &&

double.TryParse(filterValue, NumberStyles.Any, CultureInfo.InvariantCulture, out double searchDouble))

{

// Сравниваем числа с учетом точности

isMatch = Math.Abs(cellDouble - searchDouble) < 0.0001;

}

else

{

// Для остальных типов (например, ID или Category) обычное строковое сравнение

isMatch = string.Equals(cellValue, filterValue, StringComparison.OrdinalIgnoreCase);

}

}

else

{

// Частичное совпадение

isMatch = cellValue.IndexOf(filterValue, StringComparison.OrdinalIgnoreCase) >= 0;

}

// Выделяем совпадения

if (isMatch)

{

foundCount++; // Увеличиваем счетчик найденных элементов

foreach (DataGridViewCell cell in row.Cells)

{

cell.Style.BackColor = System.Drawing.Color.LightGreen;

}

}

}

// Обновляем информацию о количестве найденных элементов

lblFilteredCount.Text = $"Найдено элементов: {foundCount} из {\_totalItemsCount}";

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при выполнении поиска: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

/// <summary>

/// Обновляет индикаторы сортировки в заголовках столбцов

/// </summary>

private void UpdateSortGlyph()

{

foreach (DataGridViewColumn column in dataGridView.Columns)

{

column.HeaderCell.SortGlyphDirection = SortOrder.None;

}

if (!string.IsNullOrEmpty(\_currentSortColumn))

{

var sortedColumn = dataGridView.Columns

.Cast<DataGridViewColumn>()

.FirstOrDefault(c => c.HeaderText == \_currentSortColumn);

if (sortedColumn != null)

{

sortedColumn.HeaderCell.SortGlyphDirection = \_currentSortDirection == ListSortDirection.Ascending

? SortOrder.Ascending

: SortOrder.Descending;

}

}

}

/// <summary>

/// Обработчик кнопки сброса фильтров

/// </summary>

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

{

// Сбрасываем параметры сортировки

\_currentSortColumn = string.Empty;

\_currentSortDirection = ListSortDirection.Ascending;

// Загружаем исходные данные

LoadData();

UpdateSortGlyph(); // Обновляем индикаторы сортировки

}

/// <summary>

/// Включает или отключает элементы управления интерфейса, связанные с работой с данными.

/// </summary>

/// <param name="isEnabled">Если true - элементы включаются, если false - отключаются.</param>

private void SetControlsEnabled(bool isEnabled)

{

// Основные элементы управления

comboBoxCategory.Enabled = isEnabled;

txtWeight.Enabled = isEnabled;

txtPrice.Enabled = isEnabled;

// Кнопки CRUD операций

btnAdd.Enabled = isEnabled;

btnEdit.Enabled = isEnabled;

btnDelete.Enabled = isEnabled;

// Элементы фильтрации

comboBoxFilter.Enabled = isEnabled;

txtFilterValue.Enabled = isEnabled;

btnFilter.Enabled = isEnabled;

btnClearFilter.Enabled = isEnabled;

checkBoxExactMatch.Enabled = isEnabled;

// Поиск и управление БД

buttonSearch.Enabled = isEnabled;

deleteDatabaseToolStripMenuItem.Enabled = isEnabled;

saveAsDatabaseToolStripMenuItem.Enabled = isEnabled;

saveTableAsPDFToolStripMenuItem.Enabled = isEnabled;

saveAsDatabaseToolStripMenuItem.Enabled = isEnabled;

}

/// <summary>

/// Обработчик пункта меню "Открыть базу данных"

/// </summary>

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

{

using (var openFileDialog = new OpenFileDialog())

{

openFileDialog.Filter = "SQLite Database (\*.sqlite)|\*.sqlite|All Files (\*.\*)|\*.\*";

openFileDialog.Title = "Выберите файл базы данных";

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

{

string selectedFile = openFileDialog.FileName;

try

{

// Создаем новый контекст базы данных

\_dbContext = new DatabaseContext($"Data Source={selectedFile};Version=3;");

\_dbContext.InitializeDatabase(); // Проверяем структуру

LoadData(); // Загружаем данные

UpdateDatabaseNameDisplay();

isEnabled = true;

// Активируем элементы управления

SetControlsEnabled(isEnabled);

}

catch (Exception ex)

{

// Логируем ошибку

string logPath = Path.Combine(Environment.GetFolderPath(Environment.SpecialFolder.ApplicationData), "YourAppName", "error.log");

Directory.CreateDirectory(Path.GetDirectoryName(logPath) ?? string.Empty);

File.AppendAllText(logPath, $"[{DateTime.Now}] Error: {ex}\n\n");

MessageBox.Show($"Ошибка: {ex.Message}\nПодробности в логе: {logPath}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

\_dbContext = null;

}

}

}

}

/// <summary>

/// Обработчик пункта меню "Создать базу данных"

/// </summary>

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

{

using (var saveFileDialog = new SaveFileDialog())

{

saveFileDialog.Filter = "SQLite Database (\*.sqlite)|\*.sqlite|All Files (\*.\*)|\*.\*";

saveFileDialog.Title = "Создайте новый файл базы данных";

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

{

string newDatabasePath = saveFileDialog.FileName;

try

{

// Создаем новый файл базы данных

File.Create(newDatabasePath).Close();

\_dbContext = new DatabaseContext($"Data Source={newDatabasePath};Version=3;");

\_dbContext.InitializeDatabase(); // Инициализируем структуру

LoadData(); // Загружаем данные (пустую таблицу)

UpdateDatabaseNameDisplay();

isEnabled = true;

// Активируем элементы управления

SetControlsEnabled(isEnabled);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при создании базы данных: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

\_dbContext = null;

}

}

}

}

/// <summary>

/// Обработчик пункта меню "Сохранить базу данных как..."

/// </summary>

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

{

if (\_dbContext == null || string.IsNullOrEmpty(\_dbContext.ConnectionString))

{

MessageBox.Show("База данных не открыта.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

try

{

using (var saveFileDialog = new SaveFileDialog())

{

saveFileDialog.Filter = "SQLite Database (\*.sqlite)|\*.sqlite|All Files (\*.\*)|\*.\*";

saveFileDialog.Title = "Сохранить базу данных как";

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

{

string newDatabasePath = saveFileDialog.FileName;

// Получаем путь к текущей базе данных

var connectionStringBuilder = new SQLiteConnectionStringBuilder(\_dbContext.ConnectionString);

string currentDatabasePath = connectionStringBuilder.DataSource;

// Закрываем подключения

\_dbContext.Dispose();

\_dbContext = null;

SQLiteConnection.ClearAllPools();

// Очищаем ресурсы

GC.Collect();

GC.WaitForPendingFinalizers();

System.Threading.Thread.Sleep(500);

// Проверяем блокировку файла

if (IsFileLocked(currentDatabasePath, out string? lockingProcessName))

{

MessageBox.Show($"Файл базы данных используется процессом: {lockingProcessName}.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

// Копируем базу данных

File.Copy(currentDatabasePath, newDatabasePath, overwrite: true);

// Удаляем старый файл, если требуется

if (File.Exists(currentDatabasePath))

{

File.Delete(currentDatabasePath);

}

// Восстанавливаем подключение к новой базе

\_dbContext = new DatabaseContext($"Data Source={newDatabasePath};Version=3;");

LoadData();

MessageBox.Show("База данных успешно сохранена.", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при сохранении базы данных: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

/// <summary>

/// Обработчик пункта меню "Сохранить таблицу как PDF"

/// </summary>

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

{

using (var saveFileDialog = new SaveFileDialog())

{

saveFileDialog.Filter = "PDF Files (\*.pdf)|\*.pdf|All Files (\*.\*)|\*.\*";

saveFileDialog.Title = "Сохранить таблицу как PDF";

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

{

string pdfFilePath = saveFileDialog.FileName;

if (string.IsNullOrEmpty(pdfFilePath) || Path.GetInvalidPathChars().Any(pdfFilePath.Contains))

{

MessageBox.Show("Недопустимый путь к файлу.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

return;

}

try

{

if (File.Exists(pdfFilePath))

{

File.Delete(pdfFilePath);

}

QuestPDF.Settings.License = LicenseType.Community;

string dbName = "Неизвестная база данных";

if (\_dbContext != null && !string.IsNullOrEmpty(\_dbContext.ConnectionString))

{

var builder = new SQLiteConnectionStringBuilder(\_dbContext.ConnectionString);

dbName = Path.GetFileNameWithoutExtension(builder.DataSource) ?? dbName;

}

// Структура документа

Document.Create(container =>

{

container.Page(page =>

{

page.Size(PageSizes.A4);

page.Margin(2, Unit.Centimetre);

page.PageColor(Colors.White);

page.DefaultTextStyle(x => x.FontSize(12));

page.Content().Column(column =>

{

// Cинтаксис для заголовка

column.Item().PaddingBottom(10).AlignCenter().Text(text =>

{

text.Span($"Таблица из базы данных: {dbName}")

.FontSize(20)

.Bold();

});

column.Item().Border(1).Table(table =>

{

table.ColumnsDefinition(columns =>

{

foreach (DataGridViewColumn column in dataGridView.Columns)

{

columns.RelativeColumn();

}

});

var headerStyle = TextStyle.Default.Bold();

var cellStyle = TextStyle.Default;

table.Header(header =>

{

foreach (DataGridViewColumn column in dataGridView.Columns)

{

// Cинтаксис для заголовков столбцов

header.Cell()

.Border(1)

.Background(Colors.Grey.Lighten3)

.Padding(5)

.AlignCenter()

.Text(text =>

{

text.Span(column.HeaderText)

.Style(headerStyle);

});

}

});

// Заполнение данными

foreach (DataGridViewRow row in dataGridView.Rows)

{

if (row.IsNewRow) continue;

foreach (DataGridViewCell cell in row.Cells)

{

string cellValue = cell.Value?.ToString() ?? string.Empty;

table.Cell()

.Border(1)

.Padding(5)

.AlignCenter()

.Text(text =>

{

text.Span(cellValue)

.Style(cellStyle);

});

}

}

});

});

});

}).GeneratePdf(pdfFilePath);

MessageBox.Show("Таблица успешно сохранена в PDF.", "Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при создании PDF: {ex.Message}", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

}

}

/// <summary>

/// Обработчик пункта меню "Удалить базу данных"

/// </summary>

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

{

try

{

if (\_dbContext == null || string.IsNullOrEmpty(\_dbContext.ConnectionString))

{

MessageBox.Show("База данных не открыта.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

// Получаем путь к базе данных

var connectionStringBuilder = new SQLiteConnectionStringBuilder(\_dbContext.ConnectionString);

string databasePath = connectionStringBuilder.DataSource;

if (string.IsNullOrEmpty(databasePath) || !File.Exists(databasePath))

{

MessageBox.Show("Файл базы данных не найден.", "Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

// Запрашиваем подтверждение удаления

if (MessageBox.Show("Вы уверены, что хотите удалить текущую базу данных?",

"Подтверждение", MessageBoxButtons.YesNo, MessageBoxIcon.Question) == DialogResult.Yes)

{

try

{

// Закрываем подключения

\_dbContext.Dispose();

\_dbContext = null;

SQLiteConnection.ClearAllPools();

// Очищаем ресурсы

GC.Collect();

GC.WaitForPendingFinalizers();

System.Threading.Thread.Sleep(500);

// Проверяем блокировку файла

if (IsFileLocked(databasePath, out string? lockingProcessName) && !string.IsNullOrEmpty(lockingProcessName))

{

MessageBox.Show($"Файл базы данных используется процессом: {lockingProcessName}.",

"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Warning);

return;

}

// Перемещаем файл в корзину вместо удаления

Microsoft.VisualBasic.FileIO.FileSystem.DeleteFile(

databasePath,

Microsoft.VisualBasic.FileIO.UIOption.OnlyErrorDialogs,

Microsoft.VisualBasic.FileIO.RecycleOption.SendToRecycleBin);

// Очищаем интерфейс

dataGridView.Rows.Clear();

// Обновляем состояние меню

isEnabled = false;

SetControlsEnabled(isEnabled);

UpdateDatabaseNameDisplay();

MessageBox.Show("База данных перемещена в корзину.",

"Успех", MessageBoxButtons.OK, MessageBoxIcon.Information);

}

catch (IOException ex)

{

MessageBox.Show($"Не удалось удалить файл базы данных: {ex.Message}",

"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

if (!string.IsNullOrEmpty(databasePath))

{

\_dbContext = new DatabaseContext($"Data Source={databasePath};Version=3;");

LoadData();

}

}

}

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при удалении базы данных: {ex.Message}",

"Ошибка", MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

/// <summary>

/// Проверяет, заблокирован ли файл другим процессом

/// </summary>

/// <param name="filePath">Путь к проверяемому файлу</param>

/// <param name="lockingProcessName">Имя процесса, блокирующего файл (если есть)</param>

/// <returns>True, если файл заблокирован, иначе False</returns>

private bool IsFileLocked(string filePath, out string? lockingProcessName)

{

lockingProcessName = null;

if (string.IsNullOrEmpty(filePath)) return false;

try

{

using (FileStream stream = File.Open(filePath, FileMode.Open, FileAccess.ReadWrite, FileShare.None))

{

stream.Close();

}

}

catch (IOException)

{

Process[] processes = Process.GetProcesses();

foreach (Process process in processes)

{

try

{

if (process.MainWindowHandle != IntPtr.Zero && !string.IsNullOrEmpty(process.MainWindowTitle))

{

foreach (ProcessModule module in process.Modules)

{

if (module.FileName?.Equals(filePath, StringComparison.OrdinalIgnoreCase) == true)

{

lockingProcessName = process.ProcessName;

return true;

}

}

}

}

catch

{

continue;

}

}

return true;

}

return false;

}

/// <summary>

/// Сортирует данные в DataGridView по указанному столбцу с учетом текущего направления сортировки.

/// </summary>

/// <param name="columnHeaderText">Название столбца, по которому выполняется сортировка (должно соответствовать именам столбцов DataGridView).</param>

/// <exception cref="ArgumentException">Выбрасывается, если переданное имя столбца не соответствует ожидаемым значениям.</exception>

private void SortData(string columnHeaderText)

{

if (\_dbContext == null) return;

try

{

// Определяем направление сортировки

if (\_currentSortColumn == columnHeaderText)

{

\_currentSortDirection = \_currentSortDirection == ListSortDirection.Ascending

? ListSortDirection.Descending

: ListSortDirection.Ascending;

}

else

{

\_currentSortColumn = columnHeaderText;

\_currentSortDirection = ListSortDirection.Ascending;

}

// Получаем все данные

var furnitureList = \_dbContext.GetAllFurniture();

// Сортируем в зависимости от столбца

switch (columnHeaderText)

{

case "Id":

furnitureList = \_currentSortDirection == ListSortDirection.Ascending

? furnitureList.OrderBy(f => f.Id).ToList()

: furnitureList.OrderByDescending(f => f.Id).ToList();

break;

case "Категория":

furnitureList = \_currentSortDirection == ListSortDirection.Ascending

? furnitureList.OrderBy(f => f.Category).ToList()

: furnitureList.OrderByDescending(f => f.Category).ToList();

break;

case "Вес (кг)":

furnitureList = \_currentSortDirection == ListSortDirection.Ascending

? furnitureList.OrderBy(f => f.Weight).ToList()

: furnitureList.OrderByDescending(f => f.Weight).ToList();

break;

case "Цена (руб)":

furnitureList = \_currentSortDirection == ListSortDirection.Ascending

? furnitureList.OrderBy(f => f.Price).ToList()

: furnitureList.OrderByDescending(f => f.Price).ToList();

break;

default:

throw new ArgumentException("Неверное имя столбца");

}

// Обновляем DataGridView

dataGridView.Rows.Clear();

foreach (var furniture in furnitureList)

{

dataGridView.Rows.Add(

furniture.Id,

furniture.Category,

furniture.Weight.ToString("F2", CultureInfo.InvariantCulture),

furniture.Price.ToString("F2", CultureInfo.InvariantCulture)

);

}

UpdateSortGlyph();

}

catch (Exception ex)

{

MessageBox.Show($"Ошибка при сортировке: {ex.Message}", "Ошибка",

MessageBoxButtons.OK, MessageBoxIcon.Error);

}

}

/// <summary>

/// Обрабатывает клик по заголовку столбца DataGridView и сортирует данные по выбранному столбцу.

/// </summary>

/// <param name="sender">Объект, вызвавший событие (DataGridView).</param>

/// <param name="e">Аргументы события, содержащие информацию о клике (кнопка мыши, индекс столбца).</param>

/// <remarks>

/// Сортировка выполняется только при клике левой кнопкой мыши.

/// </remarks>

private void dataGridView\_ColumnHeaderMouseClick(object sender, DataGridViewCellMouseEventArgs e)

{

if (e.Button == MouseButtons.Left)

{

string columnHeaderText = dataGridView.Columns[e.ColumnIndex].HeaderText;

SortData(columnHeaderText);

}

}

/// <summary>

/// Обновляет отображение имени текущей базы данных в элементе управления Label.

/// </summary>

/// <remarks>

/// Если подключение к БД отсутствует или не установлено, отображается "Текущая БД: не открыта".

/// Для SQLite имя БД извлекается из пути в ConnectionString.

/// </remarks>

private void UpdateDatabaseNameDisplay()

{

if (\_dbContext == null || string.IsNullOrEmpty(\_dbContext.ConnectionString))

{

lblDatabaseName.Text = "Текущая БД: не открыта";

return;

}

var builder = new SQLiteConnectionStringBuilder(\_dbContext.ConnectionString);

string dbName = Path.GetFileName(builder.DataSource);

lblDatabaseName.Text = $"Текущая БД: {dbName}";

}

}

}

Код файла WeclomeForm.Designer.cs:

namespace OOP\_Cursovaya

{

partial class WelcomeForm

{

/// <summary>

/// Обязательная переменная конструктора.

/// </summary>

private System.ComponentModel.IContainer components = null;

/// <summary>

/// Освободить все используемые ресурсы.

/// </summary>

/// <param name="disposing">true, если управляемые ресурсы должны быть освобождены; иначе false.</param>

protected override void Dispose(bool disposing)

{

if (disposing && (components != null))

{

components.Dispose();

}

base.Dispose(disposing);

}

#region Код, автоматически созданный конструктором форм

/// <summary>

/// Метод, необходимый для поддержки конструктора.

/// Не изменяйте содержимое этого метода при помощи редактора кода.

/// </summary>

private void InitializeComponent()

{

SuspendLayout();

//

// WelcomeForm

//

AutoScaleDimensions = new SizeF(8F, 20F);

AutoScaleMode = AutoScaleMode.Font;

ClientSize = new Size(800, 450);

Name = "WelcomeForm";

Text = "WelcomeForm";

ResumeLayout(false);

}

#endregion

}

}

Код файла WelcomeForm.cs:

using System;

using System.Windows.Forms;

namespace OOP\_Cursovaya

{

/// <summary>

/// Форма приветствия, которая появляется при запуске приложения.

/// Предоставляет информацию о программе и автоматически закрывается через 10 секунд.

/// </summary>

public partial class WelcomeForm : Form

{

/// <summary>

/// Инициализирует новый экземпляр формы приветствия.

/// </summary>

public WelcomeForm()

{

InitializeComponent();

InitializeWelcomeForm();

}

/// <summary>

/// Настраивает внешний вид и поведение формы приветствия.

/// Добавляет метку с описанием, кнопку "Далее" и таймер для автоматического закрытия.

/// </summary>

private void InitializeWelcomeForm()

{

// Настройка формы

this.FormBorderStyle = FormBorderStyle.FixedDialog;

this.ControlBox = false;

this.StartPosition = FormStartPosition.CenterScreen;

this.Width = 400;

this.Height = 200;

this.Text = "Степанов Р.Д. 23ВП2, курсовой проект ООП";

// Создание метки с описанием

var label = new Label

{

Text = "Добро пожаловать. Это приложение для работы с базой данных 'Мебель'.\n\nЧтобы продолжить нажмите 'Далее' или подождите 10 секунд",

Dock = DockStyle.Fill,

TextAlign = ContentAlignment.MiddleCenter

};

// Создание кнопки "Далее"

var button = new Button

{

Text = "Далее",

Width = 100,

DialogResult = DialogResult.OK

};

// Панель для размещения кнопки внизу формы

var panel = new Panel

{

Dock = DockStyle.Bottom,

Height = 50

};

panel.Controls.Add(button);

button.Left = (panel.Width - button.Width) / 2;

button.Top = (panel.Height - button.Height) / 2;

// Добавление элементов на форму

this.Controls.Add(label);

this.Controls.Add(panel);

// Таймер для автоматического закрытия через 10 секунд

var timer = new System.Windows.Forms.Timer { Interval = 10000 };

timer.Tick += (sender, e) =>

{

timer.Stop();

this.DialogResult = DialogResult.OK;

this.Close();

};

timer.Start();

// Обработчик нажатия на кнопку "Далее"

button.Click += (sender, e) => this.Close();

}

}

}

Приложение Б – Руководство пользователя

Программа «OOP Cursovaya Stepanov.exe» предназначена для работы с базой данных «Мебель», предоставляя возможность создания, редактирования удалений файлов базы данных и сохранения из в PDF файлы.

Перед началом работы с файлом базы данных, его необходимо создать или открыть уже существующий (рисунок 19).

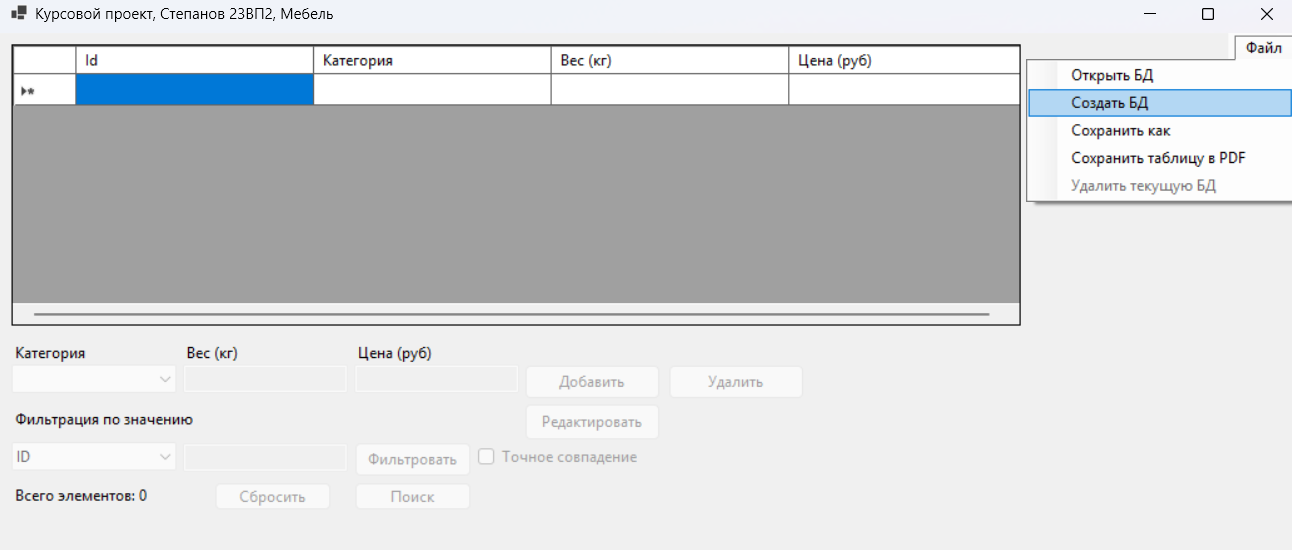


Рисунок 19 – Создание или открытие файла базы данных

При выборе одной их этих опций откроется окно файлового менеджера, где пользователь может создать файл БД или открыть существующий, в зависимости от выбранной функции (рисунок 20).

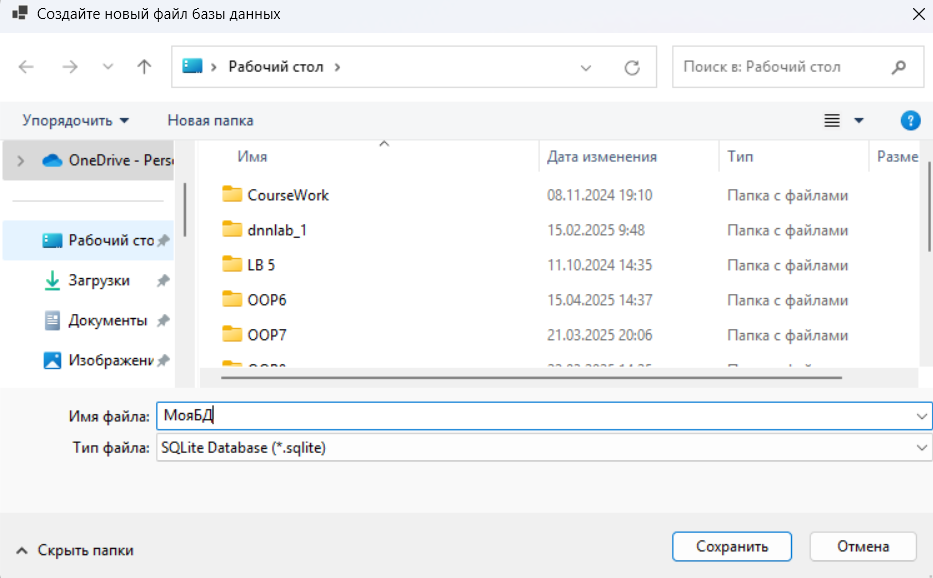


Рисунок 20 – Окно файлового менеджера

После открытия файла будет разблокирован интерфейс работы с содержимым этого файла. Для начала стоит обратить вминание на блок редактирования содержимого (рисунок 21).

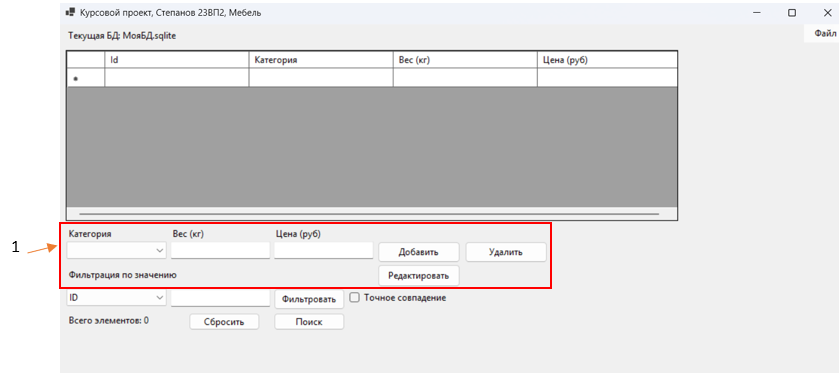


Рисунок 21 – Блок редактирования содержимого файла БД

В этом блоке пользователь указывает данные новой записи: категория, вес, цена, после чего нажимет кнопку «Добавить». После добавления можно выделить запись в таблице, нажав на соответствующую ячейку в самом левом столбце. Произойдёт выделение всей строки и также данные этой записи перенесутся в поля для ввода данных. Это позволит пользователю работать с кнопками «Удалить» и «Редактировать». Для удаления записи после выделения достаточно просто нажать кнопку «Удалить». Для редактирования надо изменить значения в полях и нажать кнопку «Редактировать».

Далее переходим к блоку настройки отображения записей. Это включает функции фильтрации, поиска и сортировки записей (рисунок 22)

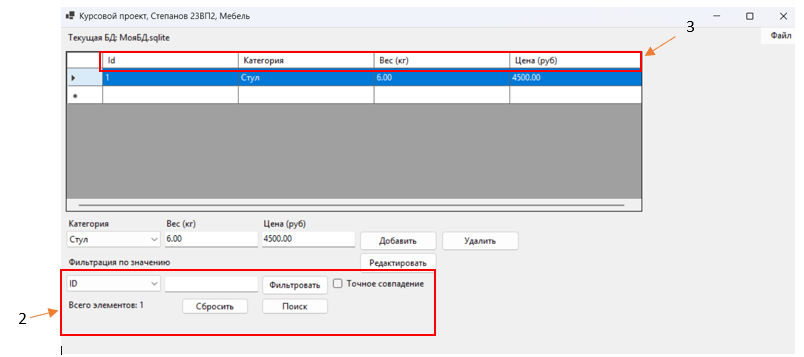


Рисунок 22 – Блок настройки отображения записей

Для функций фильтрации и поиска нужно выбрать категорию в селекторе, по которой будет работать функция, а затем указать значение. Далее нажать кнопку «Фильровать», либо «Поиск». Отличие в том, что при фильтрации все посторонние записи скрываются, а нужные остаются. При поиске остаются все записи, но нужные записи выделяются зелёным цветом. Можно произвести вначале фильтрацию, потом поиск среди отфильтрованных объектов, но не наоборот. Чекбокс «Точное совпадение» позволяет искать записи, соответствующие 1 в 1 указанному значению. Например, при мы фильтруем по категории вес с значением 10 и у нас среди записей есть веса 10, 100, 110. Если мы отфильтруем без точного совпадения, то все эти записи выведутся, так как будут содержать «10» так или иначе. С установленной галочкой выведется только запись с 10, так как будет точно соответствовать.

Для проведения сортировки нужно кликнуть по заголовку столбца, тогда будет произведена сортировка записей. Если записи стороковые то по алфавитному порядку, если числовые, то по убыванию или возрастанию, в записимости от того, сколько раз нажать.

Чтобы сбросить фильтрацию, поиск, сортировку нужно нажать кнопку «Сбросить».

После завершения работы с файлом можно просто закрыть приложение и изменения сохранятся. Также можно воспользоваться функционалом тулбокса «Файл» (рисунок 23).

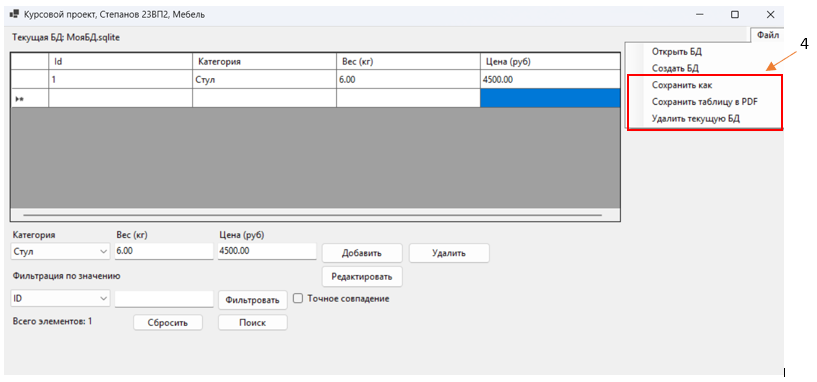


Рисунок 23 – Блок работы с файлом

Кнопка «Сохранить как» позволяет изменить название файла или сменить директорию.

Кнопка «Сохранить таблицу в PDF» позволяется сохранить таблицу в PDF файл, указав название и директорию для файла. Причём будет записана текущая отображённая таблица. То есть если в исходной таблице применена фильтрация или сортировка, то будет сохранена в PDF имеено изменённая таблица. Если это не требуется, то перед сохранением в PDF нажмите кнопку «Сбросить».

Кнопка «Удалить текущую БД» выведет окно с подтверждением удаления, если удаление подтверждено файл базы данных будет помещён в корзину. Приложение при этом останется открытым, но без файла БД.