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 «Склад»

Автор работы:	Голосова С.М.
Направление бакалавриата	09.03.04 («Программная инженерия»)
Обозначение курсовой работы	ПГУ 09.03.04 – 04КР211.04 ПЗ
Группа	21ВП1
Руководитель работы	Афонин А.Ю., к.т.н., доцент
Работа защищена «__» ____ 2023 г.	Оценка _____

Пенза 2023

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
Федеральное государственное бюджетное образовательное учреждение высшего образования
«Пензенский государственный университет»
(ФГБОУ ВО «Пензенский государственный университет»)

Кафедра «Математическое обеспечение и применение ЭВМ»

«УТВЕРЖДАЮ»

Зав. кафедрой *Козлов А.Ю.*

"__" _____ 2023 г.

ЗАДАНИЕ
на курсовое проектирование по дисциплине
«Объектно-ориентированное программирование»

Студенту Голосовой Софье Михайловне **Группа** *21ВП1*

Тема проекта: Разработка программы с использованием объектно-ориентированного подхода. ИС «Склад».

Исходные данные (технические требования) на проектирование

1. Разработать программу БД “Склад”.
2. Данные по базе включают: а) учет товаров; б) анализ сроков хранения; в) выдача товара со склада; г) формирование отчетов о хранимых товарах.
3. Программа должна выполнять следующие действия: создание базы данных, удаление базы данных, сохранение текущей базы данных в файл, добавление записей, удаление записей, редактирование записей, поиск и сортировка записей, фильтрация записей по определенному критерию.
4. Обязательные требования к программе: многомодульность, использование сложных типов данных, использование коллекции для организации базы данных. Старт приложения должен сопровождаться всплывающим окном с информацией об авторе и темой проекта.
5. Визуальный интерфейс приложения реализовать с использованием графических библиотек. WinForms.
6. Среда разработки ПО: Microsoft Visual Studio 2019/2022, VSCode
7. Язык программирования: C#
8. Программное обеспечение должно быть полностью отлажено и протестировано, и должно функционировать под управлением ОС Windows 10 и выше.

Объём работ по курсу

1. Расчётная часть

1) Анализ требований.
2) Выбор и освоение инструментальных средств анализа и проектирования
3) Определение структуры и функций приложения
4) Разработка диаграмм описания приложения
5) Реализация приложения на языке C# в среде Microsoft Visual Studio
6) Отладка и тестирование приложения

2. Графическая часть

1) Схема данных
2) Диаграмма вариантов использования
3) Диаграмма классов

3. Экспериментальная часть

1) Отладка компонентов приложения и их взаимодействия
2) Функциональное тестирование приложения

Срок выполнения проекта по разделам

1	Анализ требований	к	21 февраля	2023 г.
2	Определение структуры и функций приложения	к	28 марта	2023 г.
3	Разработка UML диаграмм приложения	к	14 марта	2023 г.
4	Реализация приложения	к	11 апреля	2023 г.
5	Отладка и тестирование приложения	к	28 апреля	2023 г.
6	Оформление пояснительной записки проекта	к	18 мая	2023 г.
7	Защита курсовой работы	к	23 мая	2023 г.

Дата выдачи задания « 7 » *февраля* *2023 г.*
Дата защиты проекта « » *202 г.*

Руководитель _____ /к.т.н. А.Ю. Афонин/

Задание получил "*7*" *февраля* 2023 г.

Студент _____ /С.М.Голосова/

Содержание

Введение

В нашем мире одной из актуальных проблем является проблема хранения, поиска и обработки данных. Почти во всех областях человеку приходится сталкиваться с работой с большим объёмом информации, а информационные системы играют большую роль в структуризации и хранении необходимой информации, схожей по тематике.

Информационные системы позволяют эффективнее организовать деятельность какой-либо организации и полностью, или частично избавиться от бумажного документооборота, который имеет ограниченный срок службы, в отличие от электронных средств хранения данных.

Таким образом информационная система – это рабочая система, деятельность которой направлена на сбор, передачу, хранение, извлечение, обработку и отображение информации.

Целью данной курсовой работы является разработка информационной системы, которая будет хранить данные о контрагентах склада, хранящих свое имущество на складе, а также данные о самом имуществе. Данная платформа поможет сотрудникам склада лучше систематизировать информацию о контрагентах и их имуществе. Разрабатываемое приложение позволяет пользователям подключаться к единой базе данных, добавлять и удалять записи, осуществлять сортировку, фильтрацию и поиск записей по заданным критериям.

Для достижения поставленной цели нужно решить следующие задачи:

- изучить особенности работы с коллекциями и файлами на C#;
- разработать модель пользовательского интерфейса программного средства;
- разработать основной функционал приложения;
- провести тестирование разработанной программы;
- разработать руководство пользователя.

Разработка программы осуществлялась под управлением операционной системы Windows 11 в среде программирования Visual Studio Community 2022 на C#.

1 Постановка задачи

Необходимо разработать программу «Склад». Программа должна содержать два раздела, один из них будет контролировать информацию о контрагентах склада, а другой информацию об имуществе, находящемся на этом складе.

Раздел контрагенты склада содержит следующие информационные поля:

— ID;

— Фамилия;

— Имя;

— Телефон;

— Номер кредитной карты.

Раздел имущество контрагентов содержит следующие информационные поля:

— ID;

— Тип (универсальное, хрупкое, скоропортящееся, острое, тяжелое);

— Количество;

— Срок хранения (в месяцах);

— Стоимость аренды;

— Выдача имущества (выдан, не выдан).

Программа должна предоставить возможности: добавления нового контрагента и имущества, удаления контрагента и имущества, сортировки контрагентов и имущества по различным критериям, фильтрации имущества по стоимости, поиска имущества по типу, удаления данных из файла.

Для реализации этих требований нужно разработать интуитивно понятный интерфейс пользователя.

Разработанное приложение должно содержать окно с информацией об авторе и теме проекта.

2 Выбор решения

2.1 Определение необходимых модулей программы

Разработка программы осуществлялась под управлением операционной системы Windows 11 в среде программирования Visual Studio Community 2022 на основе шаблона приложения Windows Forms (.NET Framework) для языка программирования C#.

Для работы с файлом базы данных была установлена библиотека Newtonsoft.Json.13.0.3.[1]

2.2 Определение структуры файла базы данных

Данные разрабатываемого приложения будут храниться в файле с форматом JSON, так как данный формат имеет структуру, что удобно при хранении записей базы данных.

Файл представляет собой массив записей, заключенных в квадратные скобки []. Одна запись состоит из множества пар ключ: значение, заключенного в фигурные скобки {}, а каждая пара «ключ, значение» отделяется друг от друга запятыми.

В разрабатываемом приложении будет использоваться два файла JSON. Первый JSON файл хранит в себе информацию, представляющую собой контрагентов склада: ID, фамилия, имя, телефон, номер кредитной карты. Второй хранит в себе информацию, представляющую собой имущество контрагентов склада: ID, тип (универсальное, хрупкое, скоропортящееся, острое, тяжелое), количество, срок хранения (в месяцах), стоимость аренды, выдача товара (выдан, не выдан) [2].

3 Описание разработки программы

Разработанное приложение должно выполнить следующие функции:

— Добавить позицию в базу;

— Удалить позицию из базы;

— Сортировать позиции;

— Фильтровать имущество по стоимости;

— Искать имущество по типу;

— Удалить все записи из файла.

В процессе описания разработки программы была составлена диаграмма вариантов использования.

Диаграмма вариантов использования [3], разработанная с помощью средства UML моделирования – Visual Paradigm, приведена на рисунке 1.

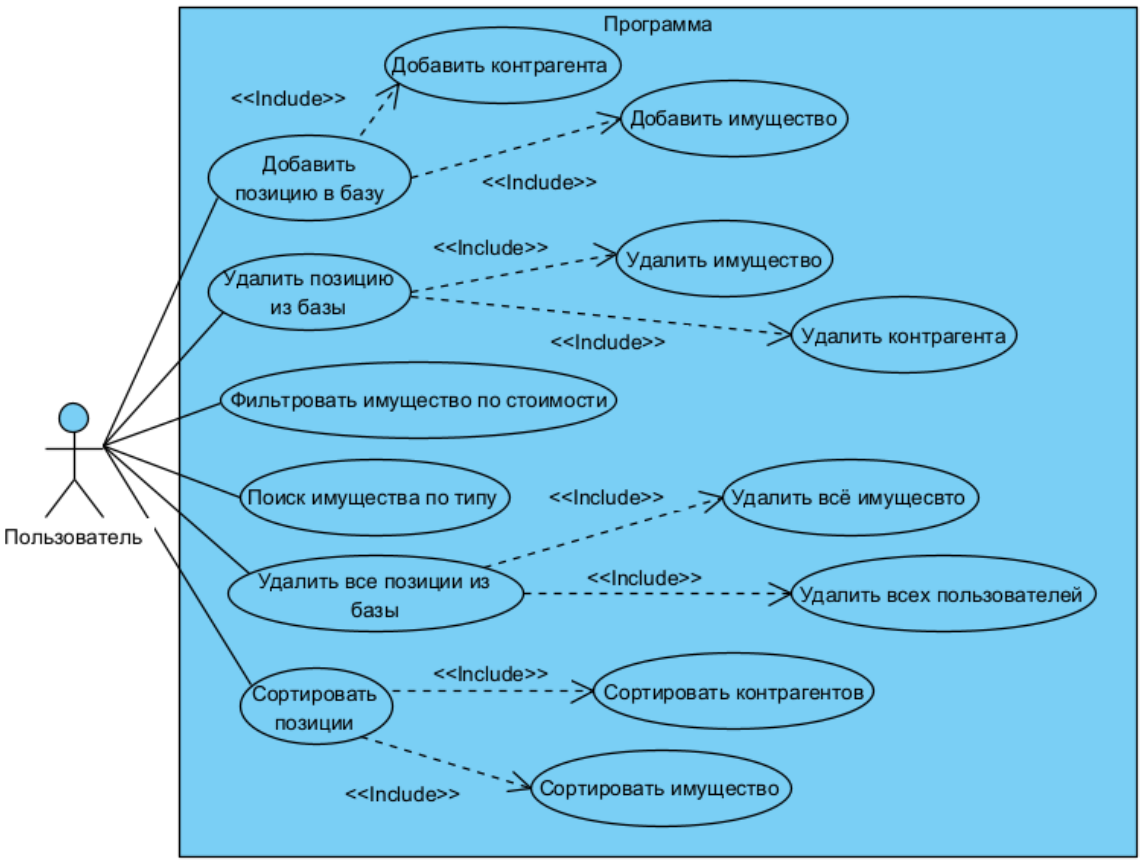

Рисунок 1 – Диаграмма вариантов использования

Описания некоторых спецификаций прецедентов представлены в таблицах 1-3.

Таблица 1 – Спецификация прецедента «Добавить запись в базу»

Наименование: Добавить позицию в базу
ID: 1
Краткое описание: пользователь заполняет информационные поля, программа добавляет запись с этой информацией в базу
Действующие лица: пользователь, программа
Основной поток: 1. Пользователь заполняет информационные поля. 2. Пользователь нажимает на кнопку «Добавить». 3. Программа проверяет корректность введённых данных. 4. Программа добавляет запись в базу.
Постусловие: Запись добавлена в базу

Таблица 2 – Спецификация прецедента «Удалить запись из базы»

Наименование: Удалить запись из базы
ID: 2
Краткое описание: пользователь выбирает запись для удаления, программа удаляет выбранную запись
Действующие лица: программа, пользователь
Основной поток: 1. Пользователь выбирает строку из таблицы с записями. 2. Пользователь нажимает на кнопку «Удалить». 3. Программа удаляет запись.
Постусловие: Запись удалена из базы

Таблица 3 – Спецификация прецедента «Фильтровать мероприятия по стоимости»

Наименование: Фильтровать мероприятия по стоимости
ID: 3
Краткое описание: пользователь вводит текст для фильтрации в текстовое поле, программа фильтрует записи по стоимости
Действующие лица: программа, пользователь
Основной поток: 1. Пользователь вводит информацию в текстовое поле. 2. Программа выводит в таблицу данные, соответствующие выбору пользователя.
Постусловие: данные, подходящие под фильтр, выведены в таблицу

Для описания структуры программного обеспечения была разработана диаграмма классов, представленная на рисунке 2.

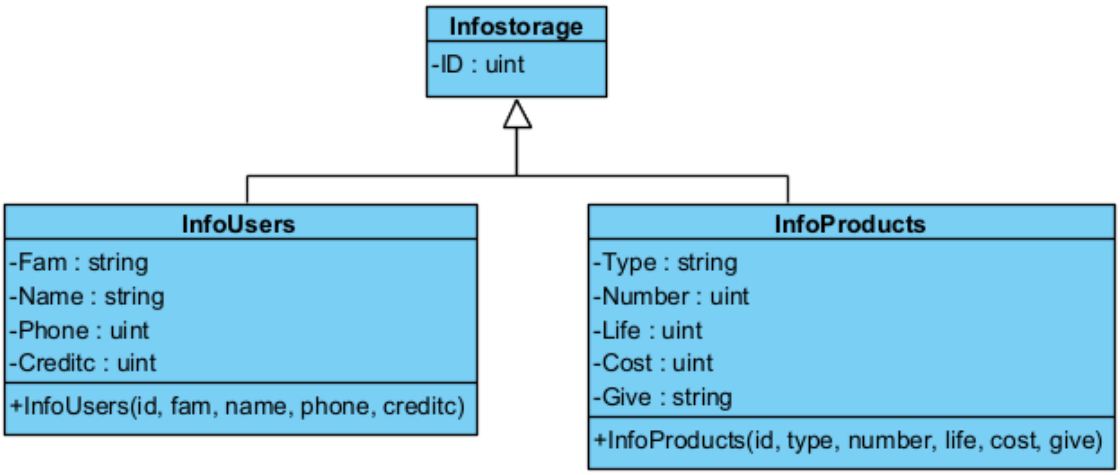

Рисунок 2 – Диаграмма классов

Программа содержит абстрактный класс Infostorage и двух его наследников: класс InfoUsers и класс InfoProducts.

- Класс Infostorage

Назначение класса: абстрактный класс.

Поля класса:

– ID – ID контрагента.

- Класс InfoUsers

Назначение класса: класс, содержащий описание одного контрагента.

Поля класса:

– Fam – фамилия контрагента.

– Name – имя контрагента.

– Phone – номер телефона.

– Creditc – номер кредитной карты.

Методы класса:

public InfoUsers(uint id, string fam, string name, uint phone, uint creditc) – конструктор класса с параметрами.

- Класс InfoProducts

Назначение класса: класс, содержащий описание одной позиции имущества.

Поля класса:

– Type – тип.

– Number – количество.

– Life – срок хранения.

– Cost – стоимость аренды.

– Give – выдача имущества.

Методы класса:

public InfoProducts(uint id, string type, uint number, uint life, uint cost, string give) – конструктор класса с параметрами.

Для упрощения работы с объектами классов InfoUsers и InfoProducts мы храним их в типизированной коллекции List [4][5].

4 Отладка и тестирование

В курсовой работе было выполнено функциональное тестирование разработанного программного обеспечения. План тестирования приведен в таблице 4.

Таблица 4 – План тестирования

№	Состав теста	Ожидаемый результат	Наблюдаемый результат
1	Добавить контрагента в таблицу	Контрагент занесен в таблицу	Контрагент отобразился в таблице (Рисунок 3)
2	Удалить контрагента из таблицы	Контрагент удален из таблицы, сообщение об удалении	Контрагент из таблицы удален, выведено сообщение «Контрагент 214 удален!» (Рисунок 4)
3	Добавить контрагента, который уже есть в базе	Сообщение о наличии такого контрагента в базе	Выведено сообщения о наличии контрагента (Рисунок 5)
4	Добавить имущество существующему контрагенту	Имущество занесено в таблицу	Имущество отобразилось в таблице (Рисунок 6)
5	Ввести некорректные данные при добавлении имущества	Сообщение о некорректных данных.	Выведено сообщение, о том, что данные в текстовых полях некорректны (Рисунок 7)

Таблица 4 – План тестирования

6	Отсортировать контрагентов в порядке возрастания по полю «ID»	Записи отсортированы	Записи в таблице упорядочены в порядке возрастания по полю «ID» (Рисунок 8)
7	Ввести в текстовое поле «13200» и все позиции отфильтровать по стоимости.	Выведены записи с полем стоимости «13200»	Позиции, соответствующие критерию фильтрации выведены в таблицу (Рисунок 9)
8	Сбросить фильтрацию в случае ее применения	В таблице появились все позиции	В таблице появились все существующие позиции (Рисунок 10)
9	Сбросить фильтрацию в случае, когда она не применена	Сообщение о том, что фильтрация не применена	Выведено сообщение «Вы не применяли фильтрацию ранее!» (Рисунок 11)
10	Удалить все позиции имущества	В таблице нет данных, сообщение об удалении данных	Таблица пустая, выведено сообщение «Данные удалены!» (Рисунок 12)
11	Добавить имущество, у которого нет контрагента	Сообщение о том, что нет соответствующего контрагента	Выведено сообщение о том, что нет соответствующего контрагента (Рисунок 13)

Таблица 4 – План тестирования

12	Удалить имущество из таблицы	Имущество удалено из таблицы	Имущество удалено из таблицы (Рисунок 14)
13	Удалить контрагента из таблицы, когда у него есть имущество	Сообщение о том, что у данного контрагента есть имущество	Выведено сообщение о том, что у данного контрагента есть имущество (Рисунок 15)
14	Добавить имущество, которое уже есть в базе	Сообщение о наличии такого имущества	Вывод сообщения о наличии имущества (Рисунок 16)
15	Ввести некорректные данные при добавлении контрагента	Сообщение о некорректных данных и подсвечивание полей.	Выведено сообщение, о том, как данные должны быть записаны в текстовом поле и подсвечиваются поля (Рисунок 17)
16	Удалить всех контрагентов, когда нет имущества	В таблице нет данных	Таблица пустая (Рисунок 18)
17	Удалить всех контрагентов, когда есть имущество.	Сообщение об ошибке.	Выведено сообщение, о том, что удаление невозможно, так как есть имущество (Рисунок 19)

Таблица 4 – План тестирования

18	Ввести в текстовое поле «острое» и найти соответствующую по типу позицию.	Подсвечена позиция с полем типа «острое»	Позиция, соответствующая критерию поиска подсвечена (Рисунок 20)
19	Сбросить поиск в случае его применения	В таблице подсвечена первая позиция	В таблице подсвечена первая позиция (Рисунок 21)

На рисунках 3–21, приведены скриншоты, отражающие результаты работы программы в процессе тестирования.

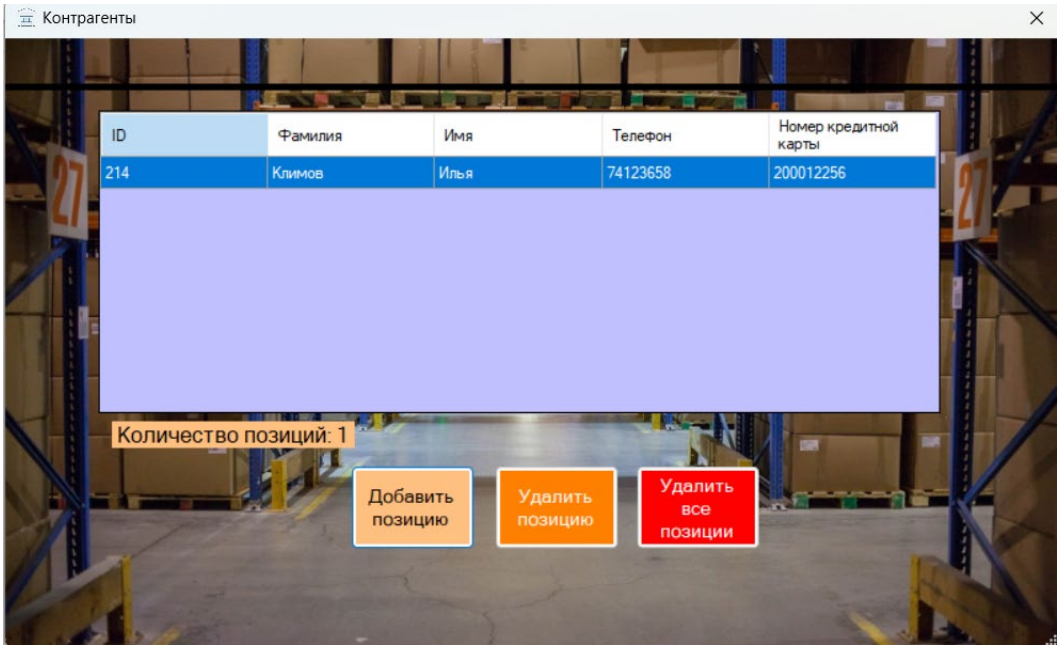

Рисунок 3 – Добавление контрагента в таблицу

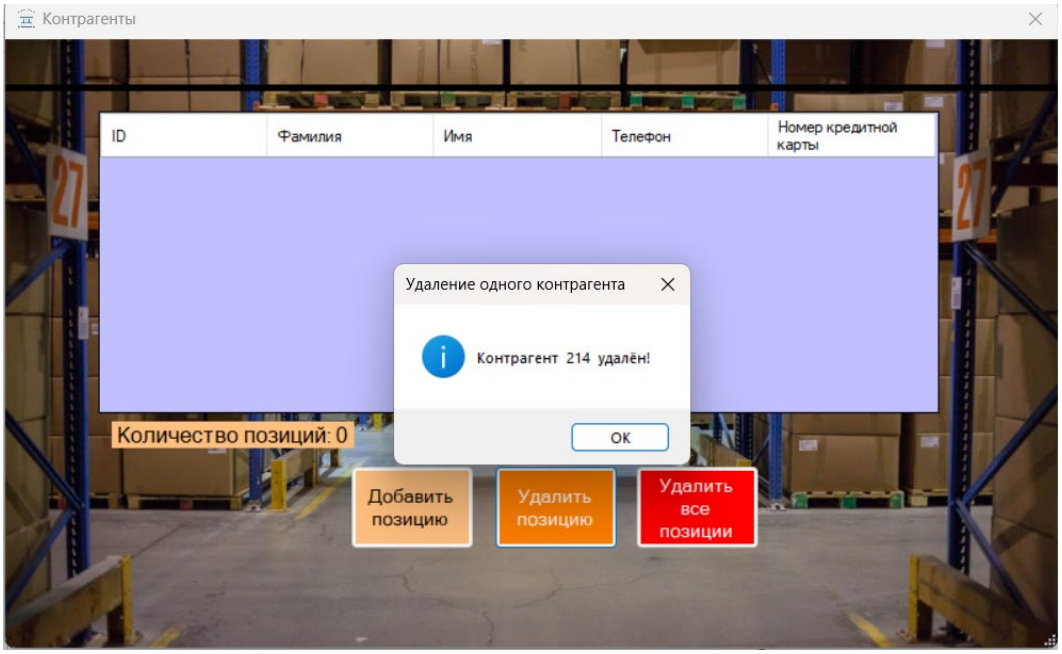

Рисунок 4 – Удаление контрагента из таблицы

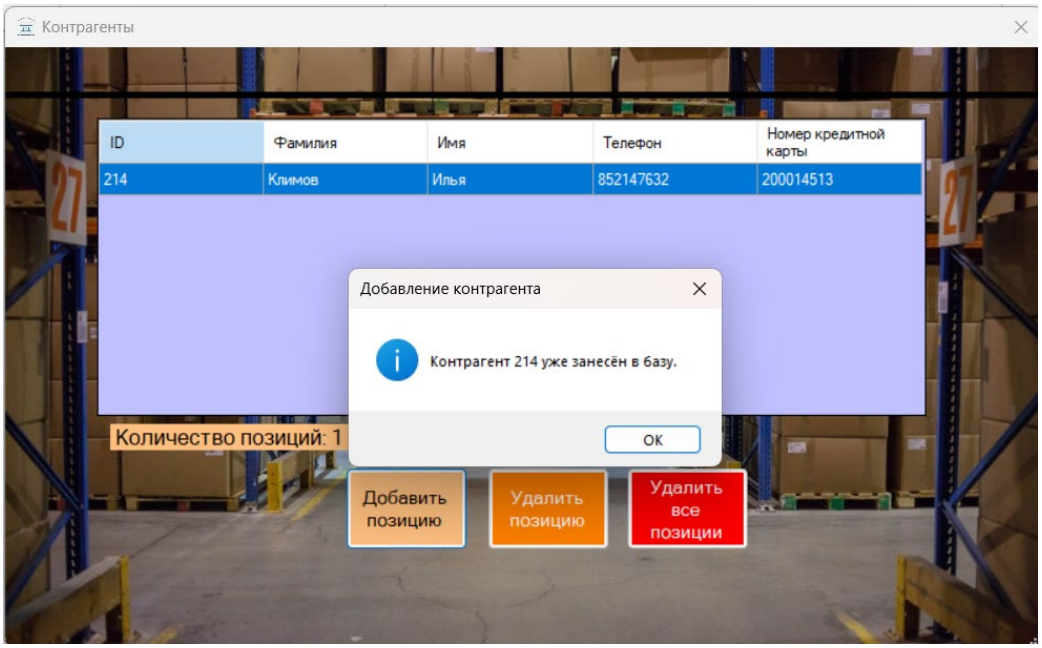

Рисунок 5 – Добавление контрагента повторно

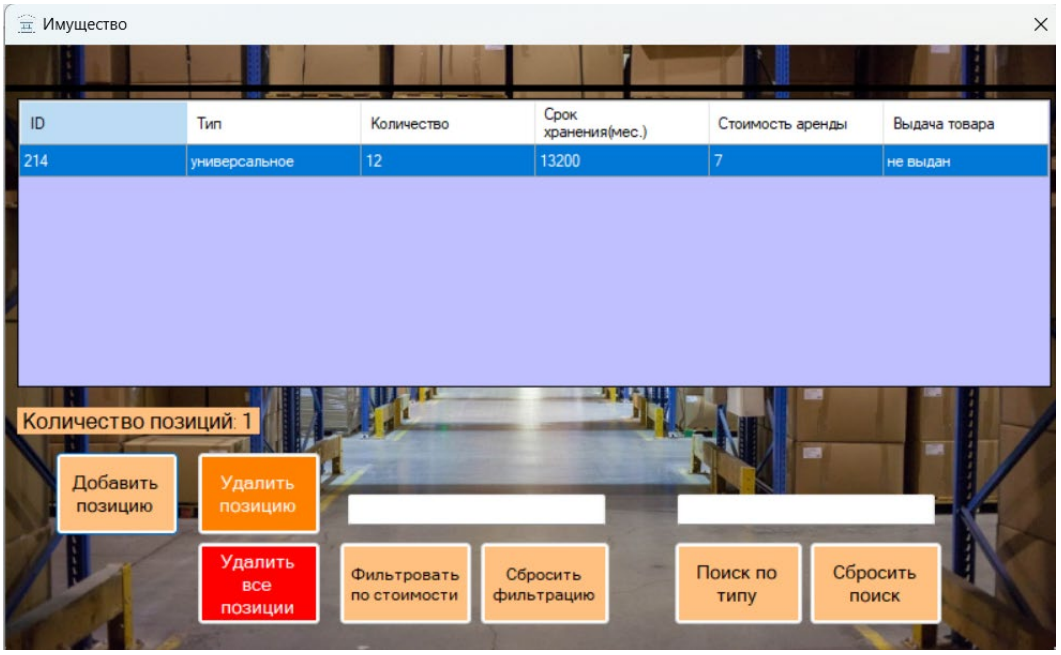

Рисунок 6 – Добавление имущества существующему контрагенту

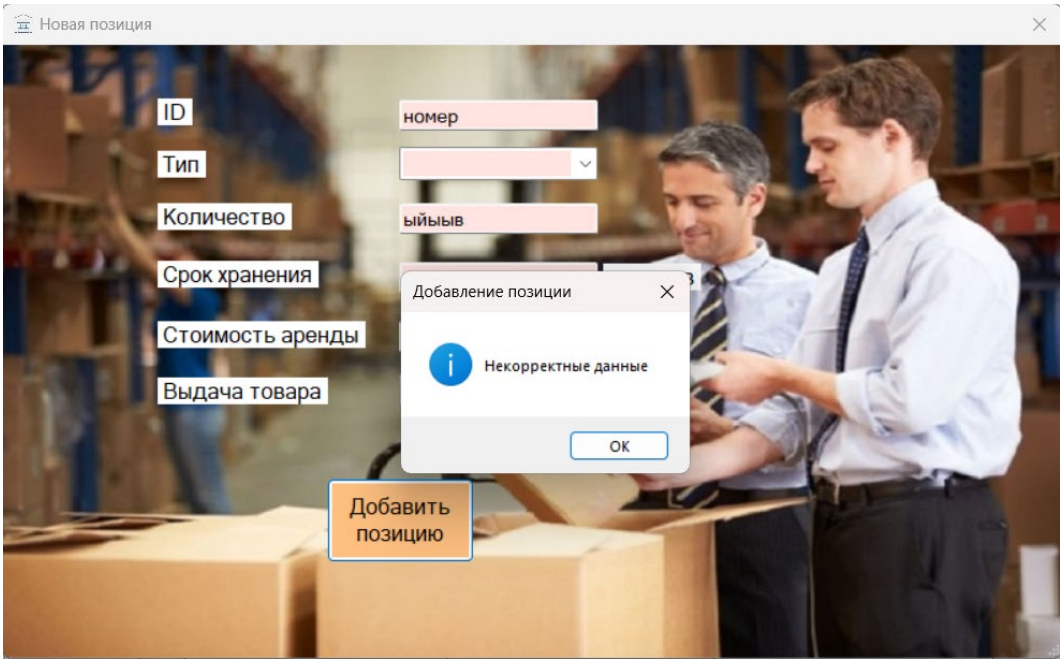

Рисунок 7 – Ввод некорректных данных при добавлении имущества

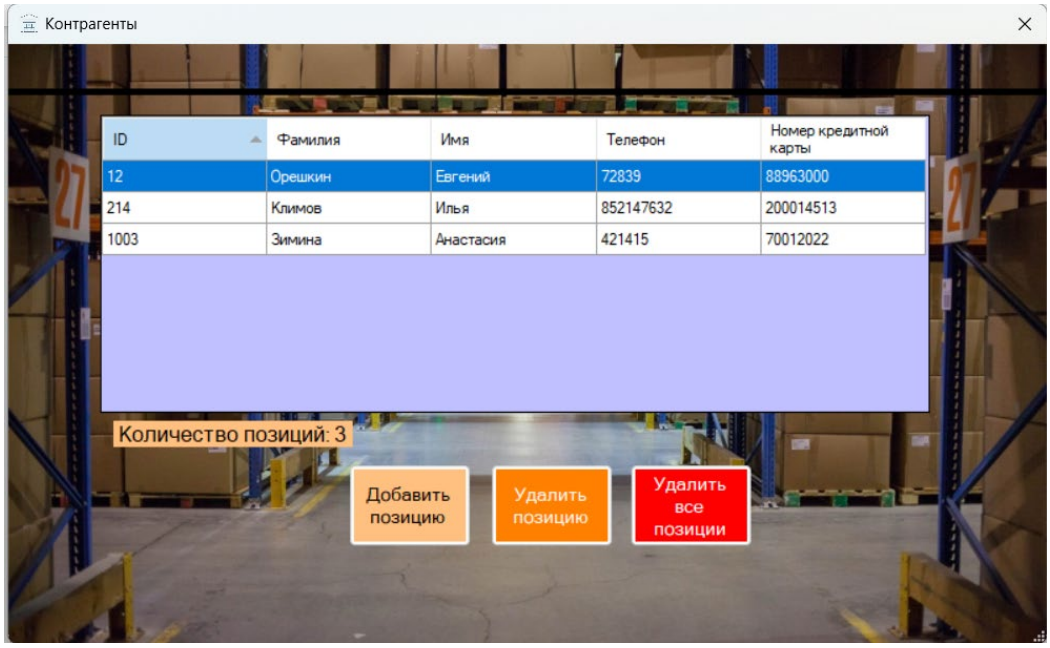

Рисунок 8 – Сортировка контрагентов в порядке возрастания по ID

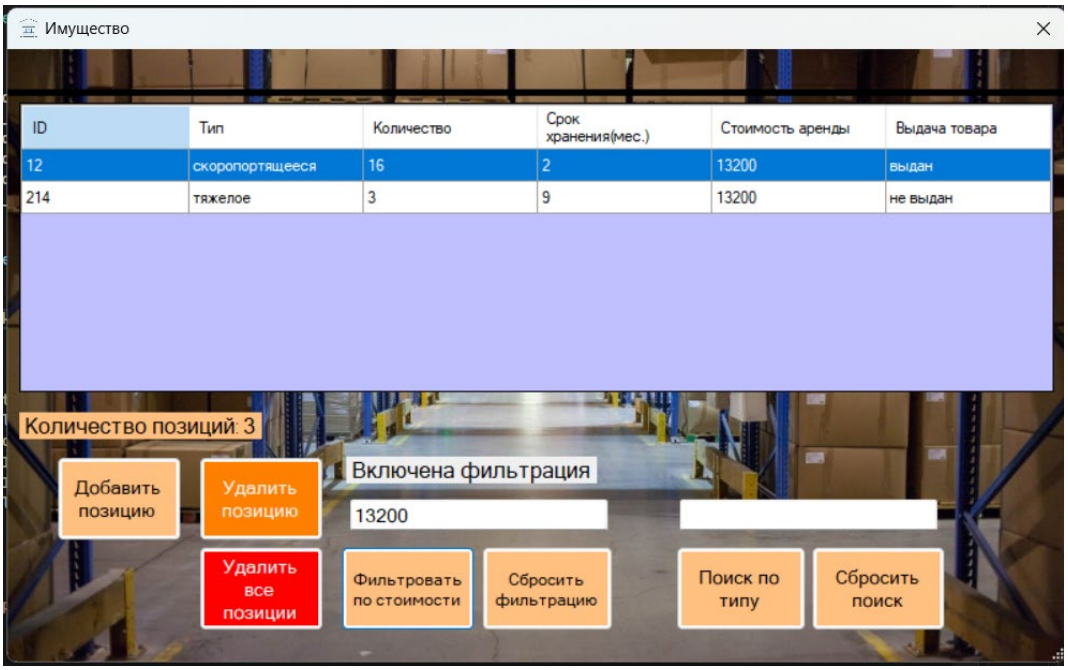

Рисунок 9 – Фильтрация имущества по стоимости «13200»

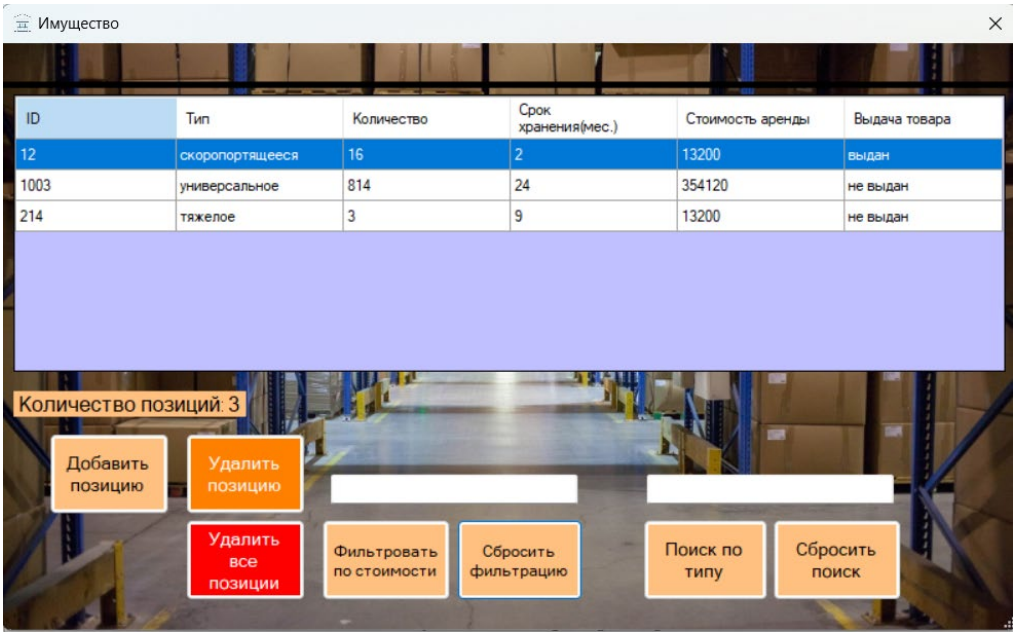

Рисунок 10 – Сброс фильтрации

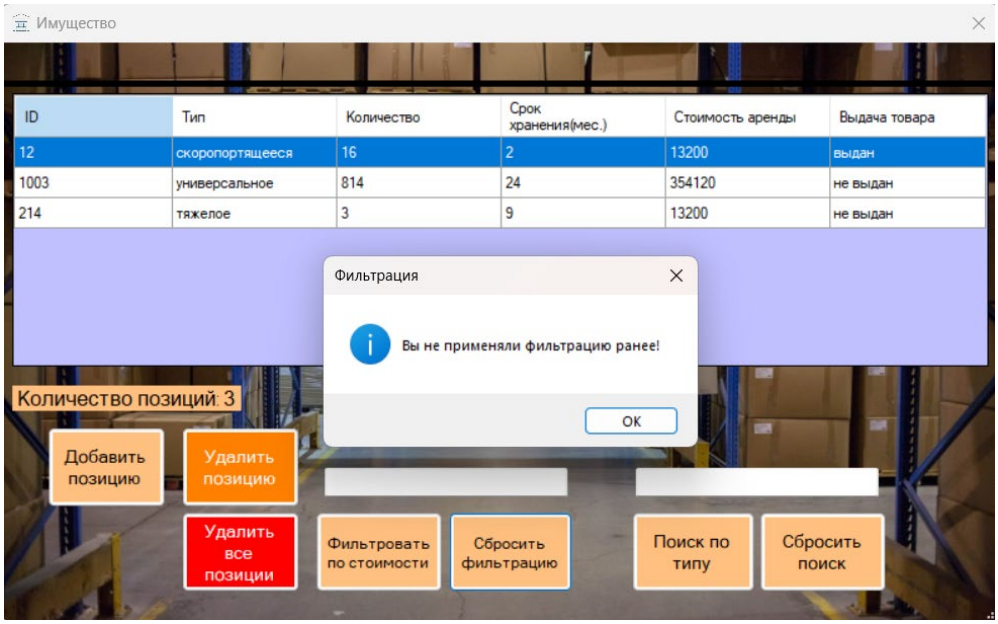

Рисунок 11 – Сброс фильтрации, когда она не применена

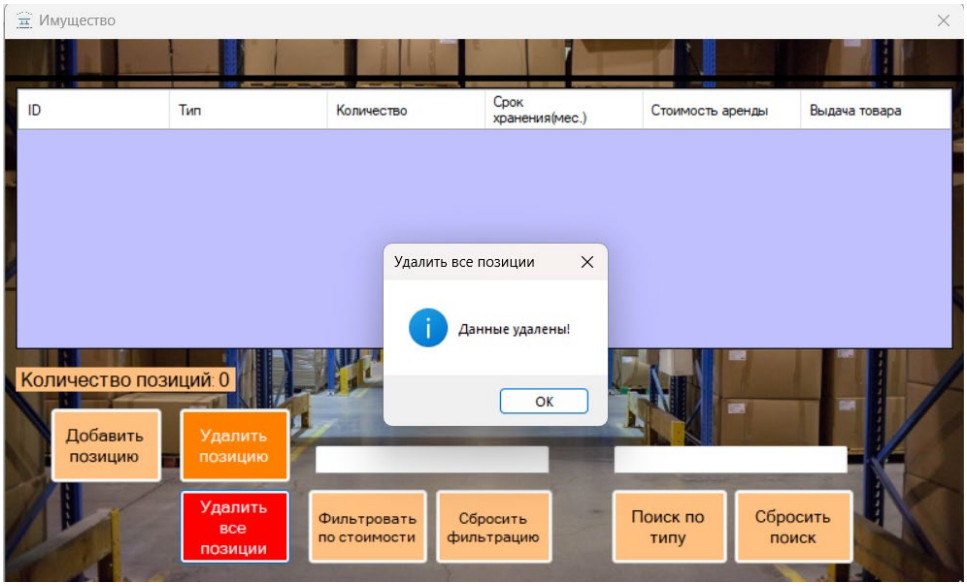

Рисунок 12 – Удаление всех позиций имущества

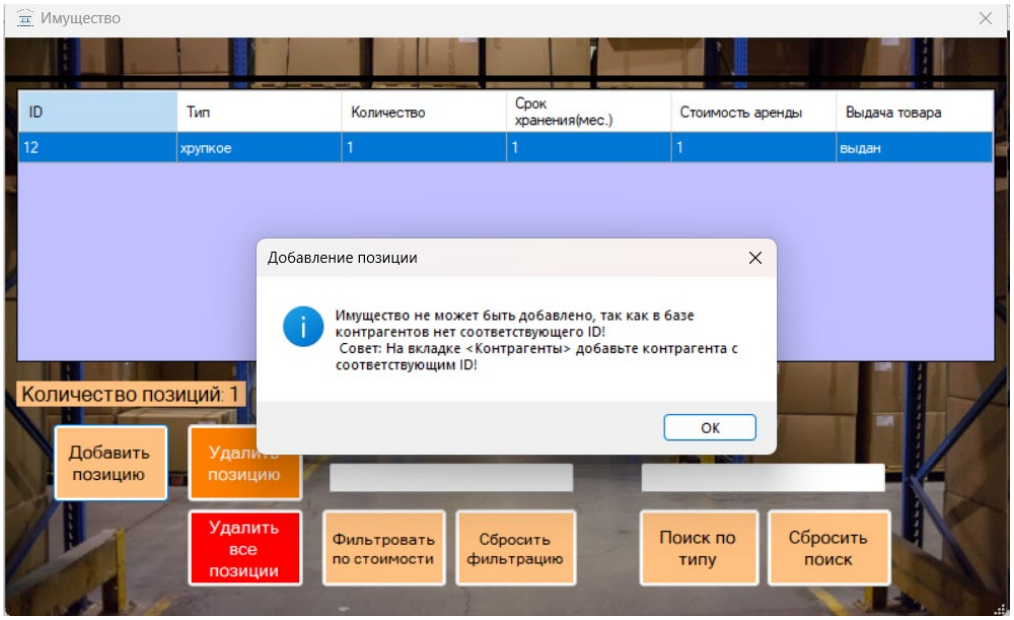

Рисунок 13 – Добавление имущества, у которого нет контрагента

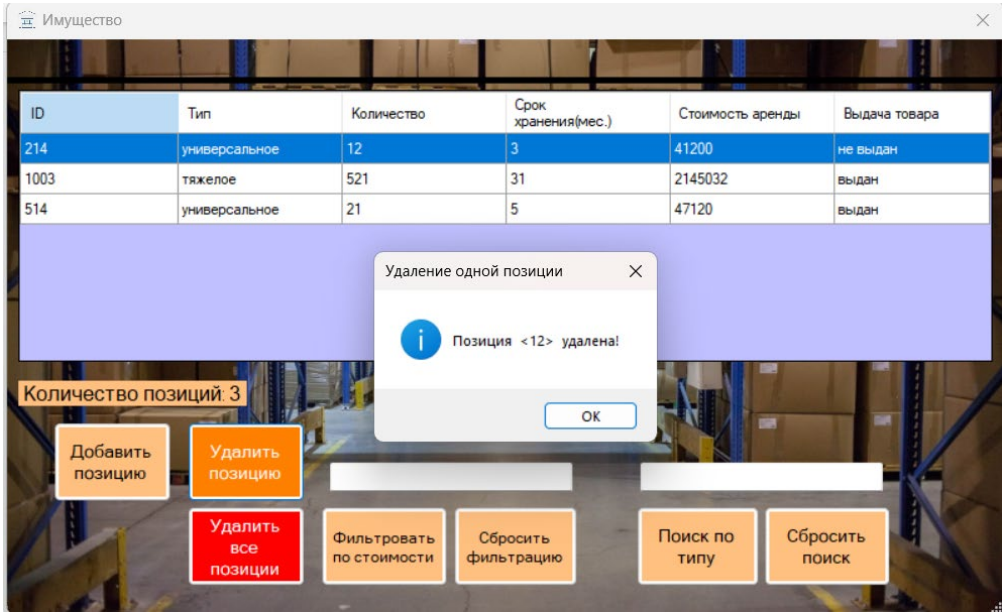

Рисунок 14 – Удаление имущества из таблицы

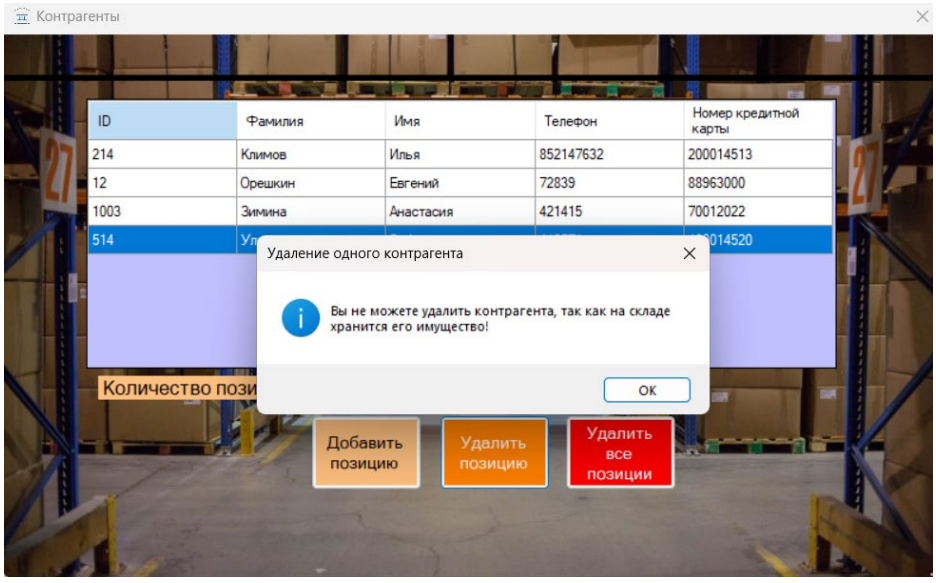

Рисунок 15 – Удаление контрагента, у которого есть имущество

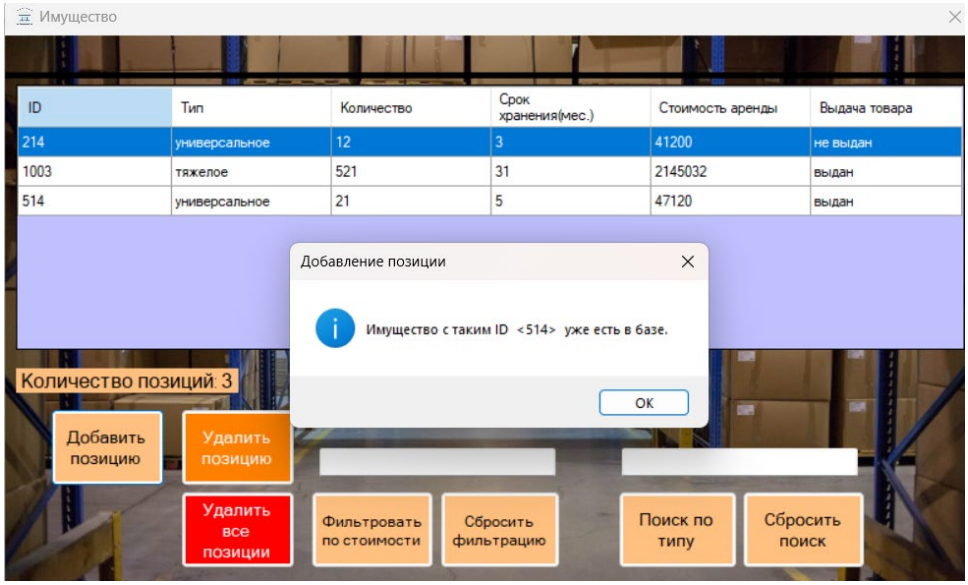

Рисунок 16 – Добавление имущества повторно

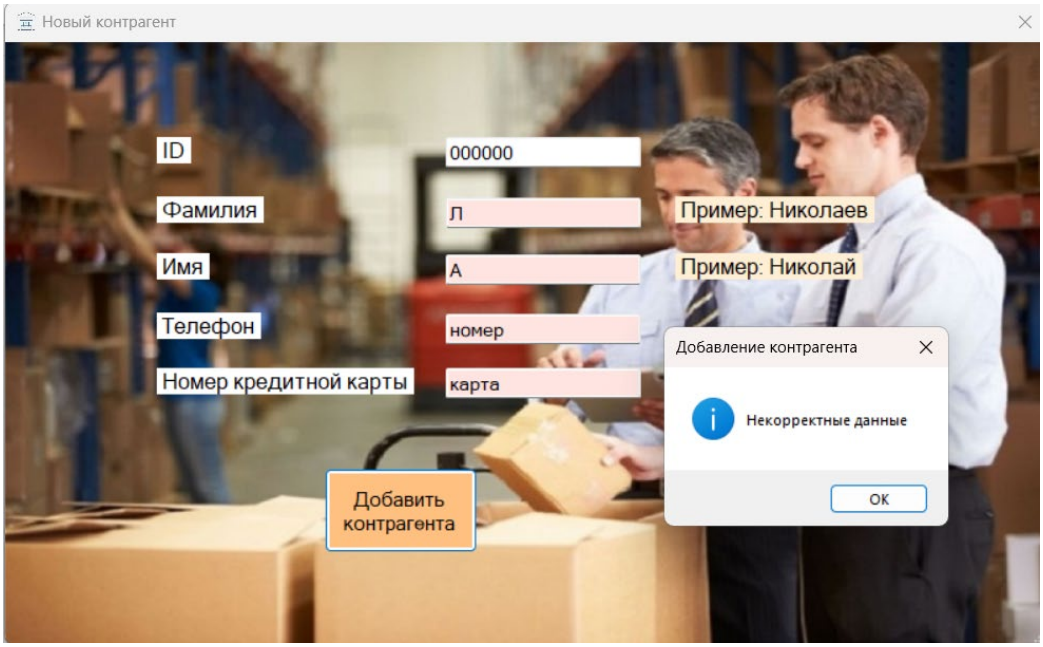

Рисунок 17 – Ввод некорректных данных при добавлении контрагента

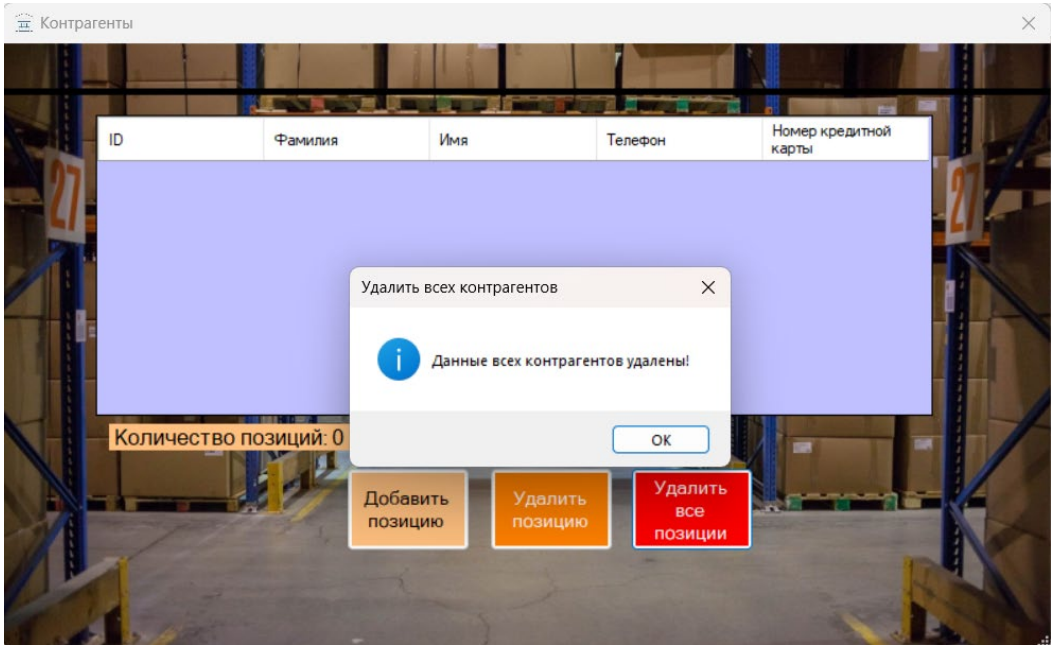

Рисунок 18 – Удаление всех контрагентов

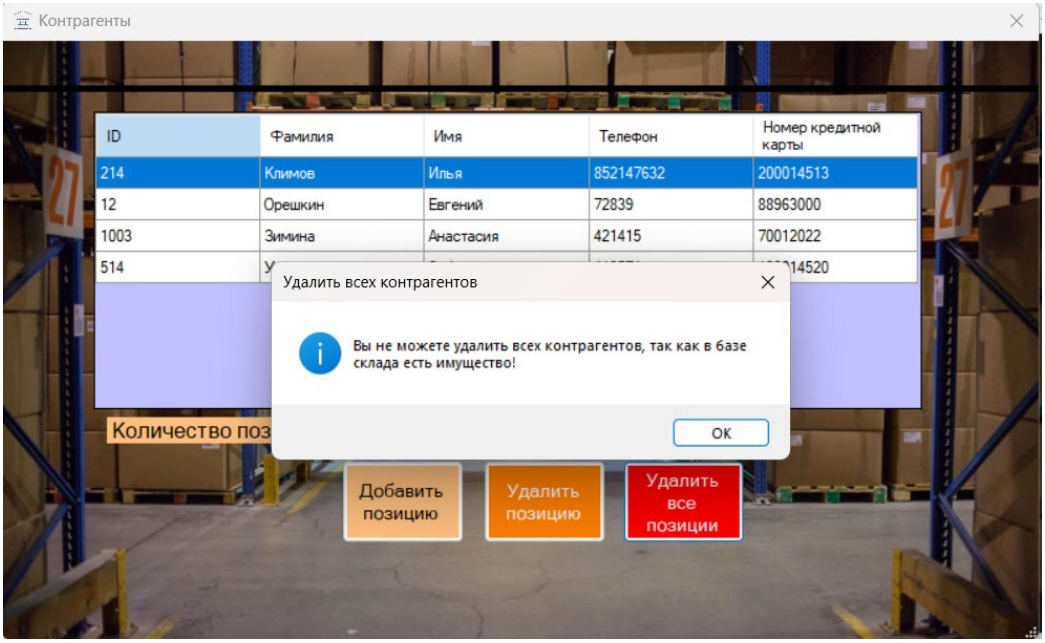

Рисунок 19 – Удаление всех контрагентов при наличии имущества

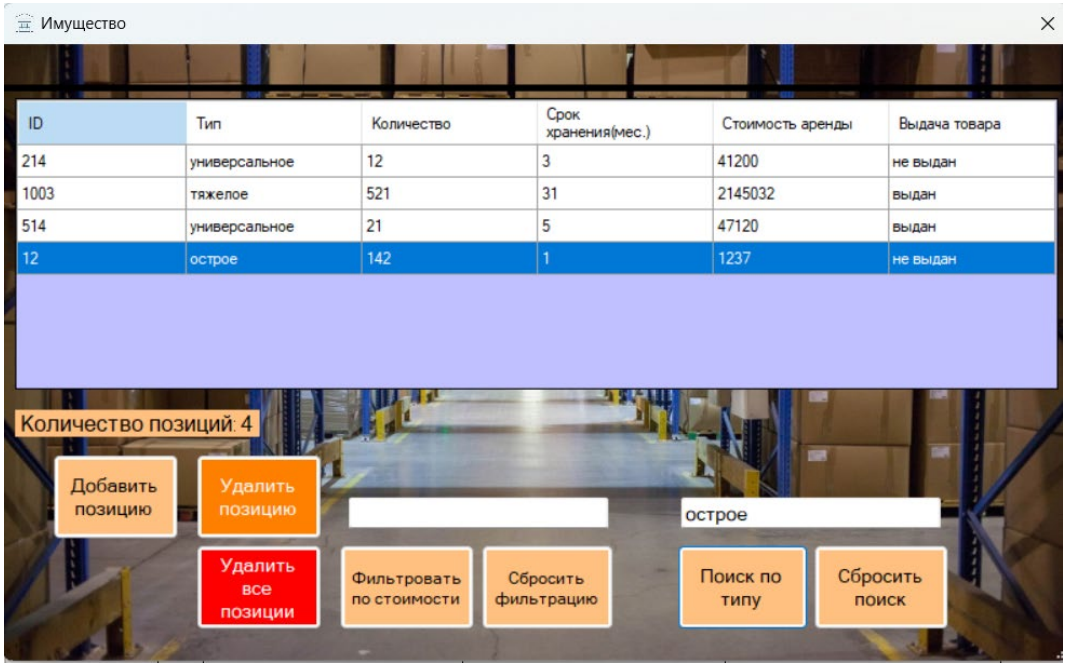

Рисунок 20 – Поиск имущества по типу «острое»

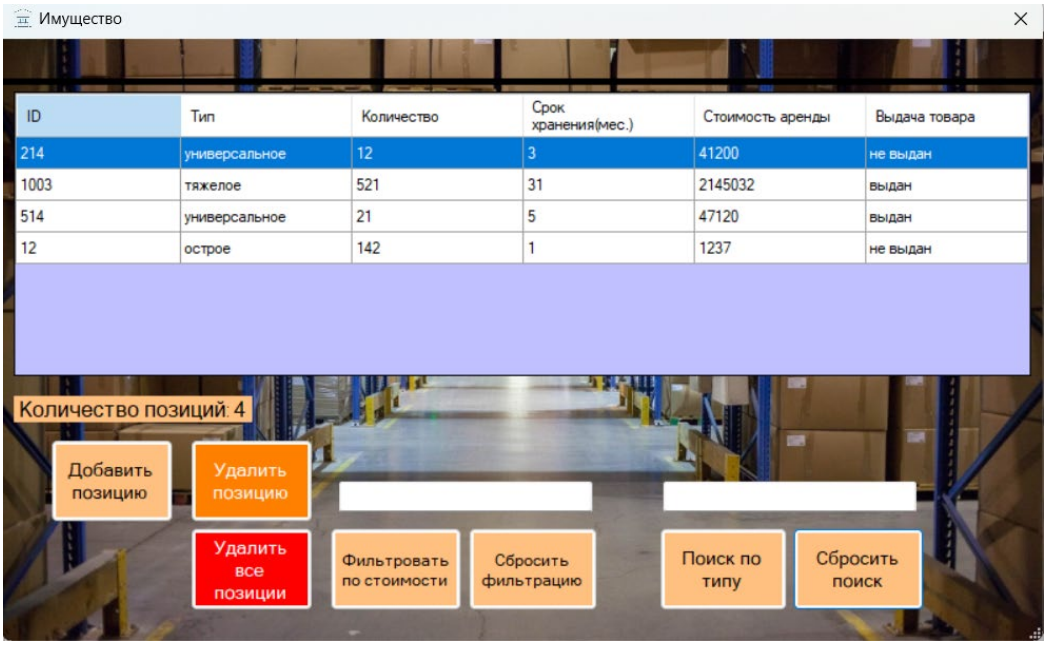

Рисунок 21 – Сброс поиска

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

5 Описание программы

5.1 Разработка приложения

В процессе выполнения курсовой работы была составлена диаграмма компонентов, которая отображает разбиение программной системы на структурные компоненты и связи между ними. Диаграмма компонентов приведена на рисунке 22.

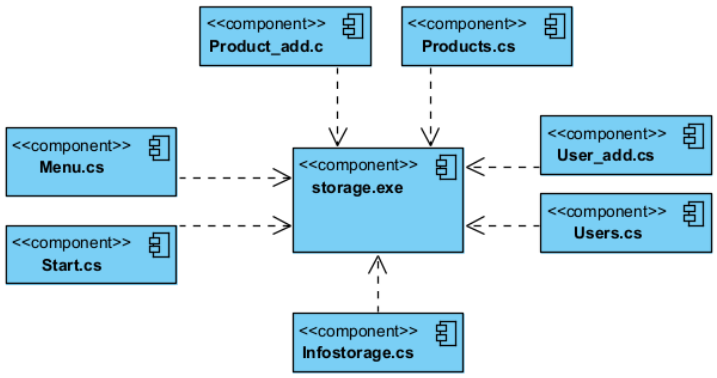

Рисунок 22 – Диаграмма компонентов

Описание компонентов приведено в таблице 5.

Таблица 5 – Описание компонентов

Компонент	Назначение
storage.exe	Исполняемый файл приложения
Infostorage.cs	Исходный файл для класса Infostorage и его наследников: InfoProducts, InfoUsers
Start.cs	Форма с информацией об авторе
Menu.cs	Форма с главным меню
Products.cs	Форма с отображением имущества и функционалом
Users.cs	Форма с отображением контрагентов и функционалом
User_add.cs	Форма добавления нового контрагента
Product_add.cs	Форма добавления нового имущества

5.2 Разработка меню

Приложение состоит из 6 различных форм.

На рисунке 23 представлен интерфейс формы информации об авторе

Рисунок 23 – Интерфейс формы информации об авторе

На рисунке 24 представлен интерфейс формы меню приложения.

Данная форма содержит следующие компоненты:

1 – кнопка для перехода на форму с информацией о контрагентах;

2 – кнопка для перехода на форму с информацией об имуществе.

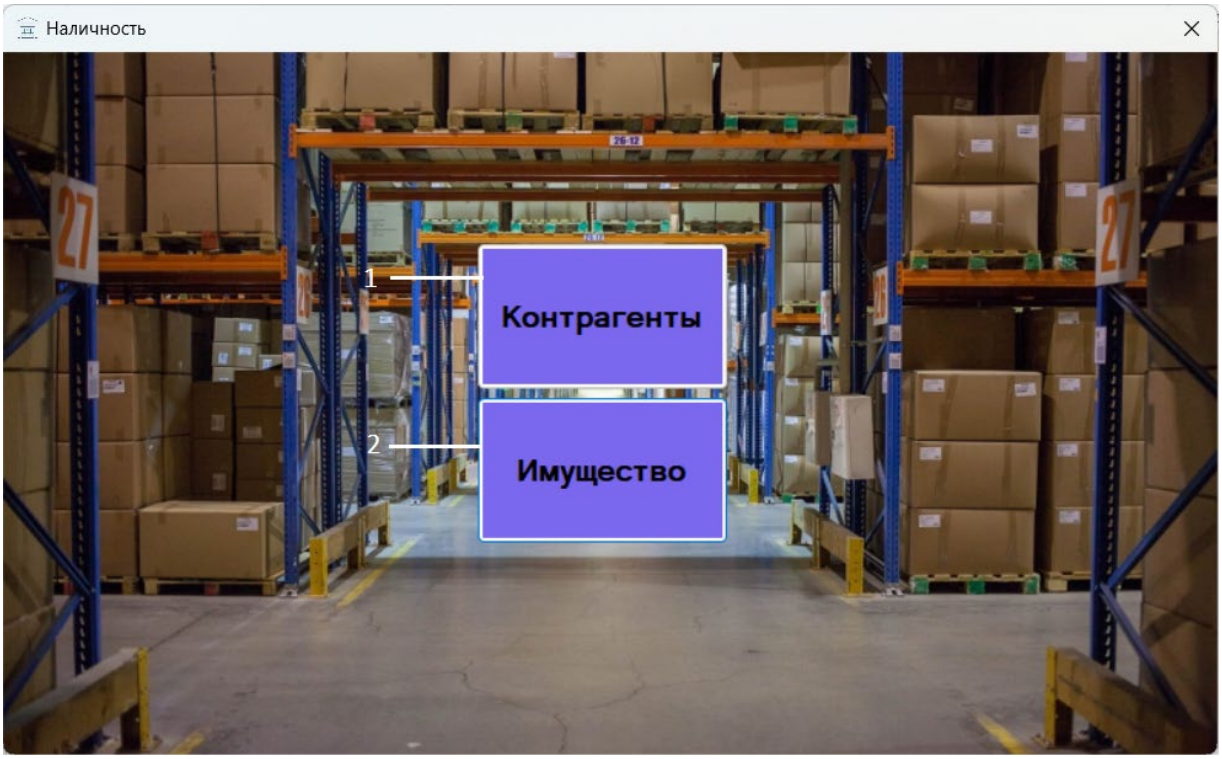

Рисунок 24 – Интерфейс формы меню приложения

На рисунке 25 представлен интерфейс формы информации обимущесвте.

Данная форма содержит следующие компоненты:

1 – таблица, в которой отображается всё имущество;

2 – label, в котором отображается количество позиций;

3 – кнопка для сброса фильтрации;

4 – кнопка для перехода на форму добавление новой позиции;

5 – кнопка для фильтрации позиций по стоимости;

6 – текстовое поле для ввода информации для фильтрации;

7 – кнопка для удаления одной позиции;

8 – кнопка для удаления всех позиций из базы;

9 – кнопка для поиска позиций по типу;

10 – текстовое поле для ввода информации для поиска;

11 – кнопка для сброса поиска.

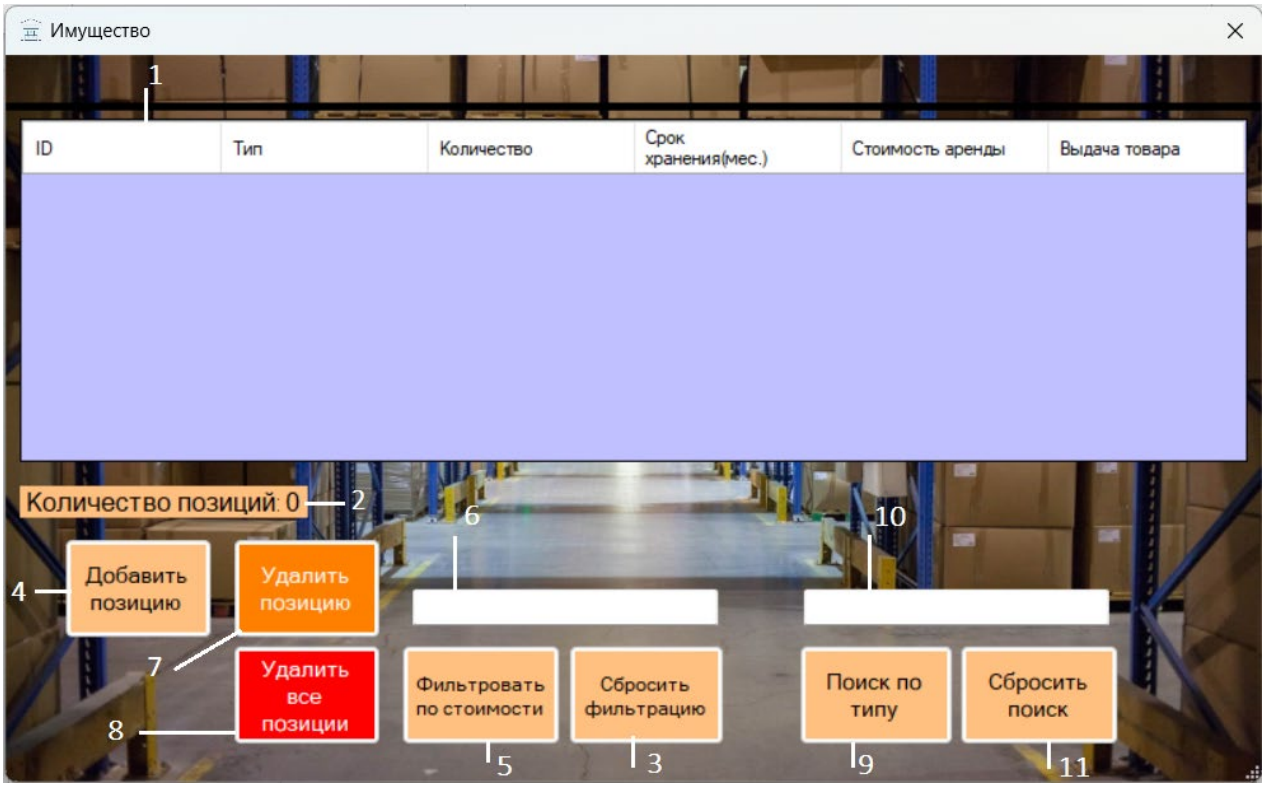

Рисунок 25 – Интерфейс формы информации об имуществе

На рисунке 26 представлен интерфейс формы добавления нового имущества.

Данная форма содержит следующие компоненты:

1 – текстовое поле для ввода ID;

2 – выпадающий список для выбора типа имущества;

3 – текстовое поле для ввода количества;

4 – текстовое поле для ввода срока хранения;

5 – текстовое поле для ввода стоимости аренды;

6 – выпадающий список для выбора состояния выдачи имущества;

7 – кнопка для добавления имущества в базу.

Рисунок 26 – Интерфейс формы добавление нового мероприятия

На рисунке 27 представлен интерфейс формы информации о контрагентах. Данная форма содержит следующие компоненты:

1 – таблица, в которой отображаются все контрагенты;

2 – кнопка для перехода на форму добавление нового контрагента;

3 – кнопка для удаления одного контрагента;

4 – кнопка для удаления всех контрагентов из базы;

5 – label, в котором отображается количество контрагентов.

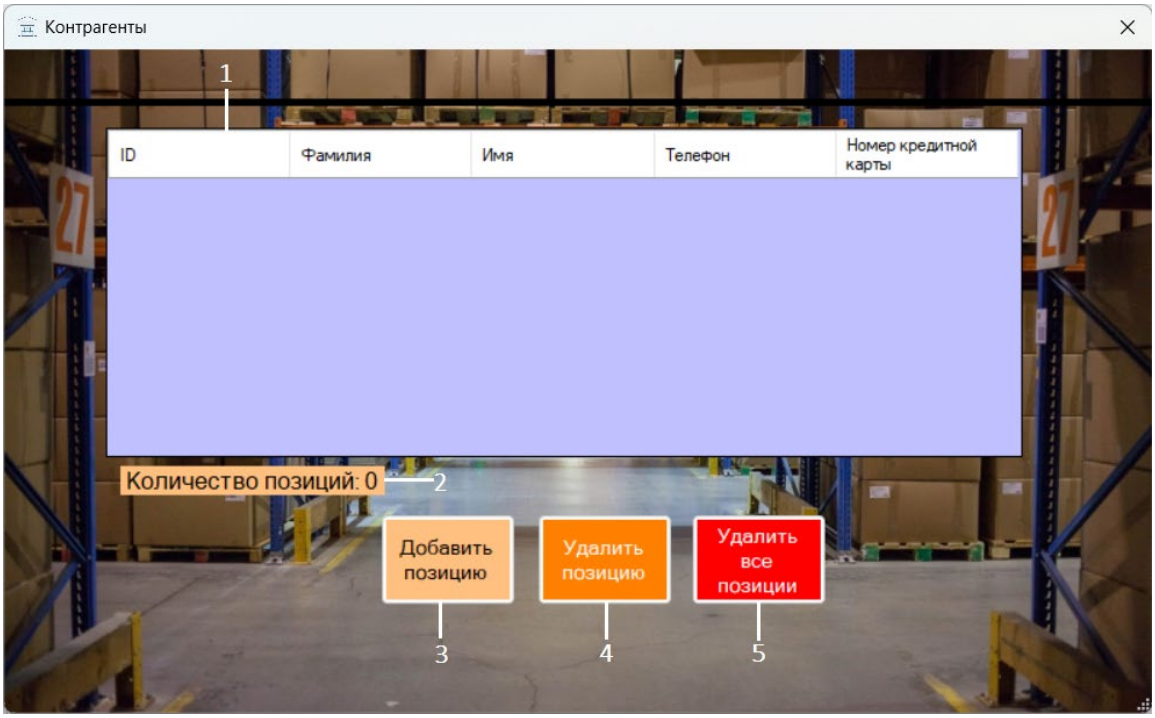

Рисунок 27 – Интерфейс формы информации о контрагентах

На рисунке 28 представлен интерфейс формы добавления нового контрагента. Данная форма содержит следующие компоненты:

1 – текстовое поле для ввода ID контрагента;

2 – текстовое поле для ввода фамилии контрагента;

3 – текстовое поле для ввода имени контрагента;

4 – текстовое поле для ввода номера телефона;

5 – текстовое поле для ввода номера кредитной карты;

6 – кнопка для добавления контрагента в базу.

Рисунок 28 – Интерфейс формы добавление нового контрагента

6 Руководство пользователя

Основная цель руководства пользователя заключается в обеспечении пользователя необходимой информацией для самостоятельной работы с программой, оно содержит описание шагов, которые необходимо выполнить для достижения пользователем конкретной цели. Текст руководства пользователя приведен в приложении А.

Заключение

В ходе выполнения курсового проекта была разработана программа «storage», позволяющая пользователям работать с базой данных, хранящей записи о контрагентах и их хранимом имуществе. Приложение предоставляет основные инструменты для следующих действий: добавление записей в БД, удаление записей, фильтрация, поиск и сортировка записей. Сконструированное программное средство отвечает всем поставленным требованиям. Результаты тестирования показывают корректную работу приложения. Код программы приведен в приложении В.

В процессе выполнения курсового проекта было разработано руководство пользователя, которое позволит быстро разобраться с функциональностью разработанного приложения.

Заключительным этапом разработки приложения стало создание инсталлятора.

Список используемых источников

1. Изучение структуры JSON – Newtonsoft:[сайт] URL: https://www.newtonsoft.com/json (Дата обращения: 18.04.2023)

2. Работа с JSON, URL: https://metanit.com/sharp/tutorial/6.5.php (Дата обращения: 19.04.2023)

3. Балашова, И. Ю. Современные информационные технологии в проектировании программных систем и комплексов: учеб. пособие / И. Ю. Балашова, Д. В. Такташкин ; под ред. П. П. Макарычева. – Пенза : Изд-во ПГУ, 2019. – 106 с. (Дата обращения: 21.04.2023)

4. C# Работа с платформой .Net:[сайт] URL: https://metanit.com/sharp/tutorial/4.5.php (Дата обращения 23.04.2023)

Приложение А - Руководство пользователя

Программа storage.exe предназначена для хранения информации о контрагентах, арендующих склад, а также для хранения информации об имуществе, хранимом на складе. Программа имеет интуитивно понятный интерфейс, оснащена главным меню с выбором необходимого раздела пользователем и поддерживает такие операции как добавление нового контрагента или имущества, удаление контрагента или имущества, удаление всех контрагентов и всего имущества из базы данных, а также несколько второстепенных операций, как сортировка по алфавиту, возрастанию и убыванию, фильтрация по стоимости или поиск по типу.

При запуске программы происходит вывод информационной формы - приветствия, после чего программа готова к выполнению своих функций.

Рисунок А.1 – Информационное окно

Для добавления нового контрагента или же имущества в таблицу необходимо нажать кнопку «Добавить позицию», после чего откроется форма для добавления позиции. После ввода данных в поля формы необходимо нажать «Добавить позицию».

Рисунок А.3 – Форма добавления контрагента

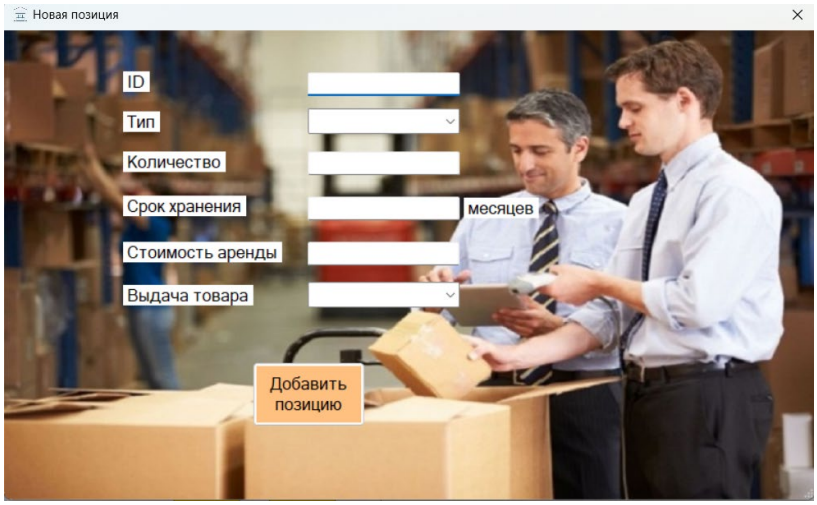

Рисунок А.4 – Форма добавления имущества

При попытке добавить позицию с частично заполненными полями или же некорректными данными, на экран будут выведены надписи с советами, а неправильно заполненные поля будут подсвечены. Все неправильные поля нужно и справить, иначе позиция не будет добавлена.

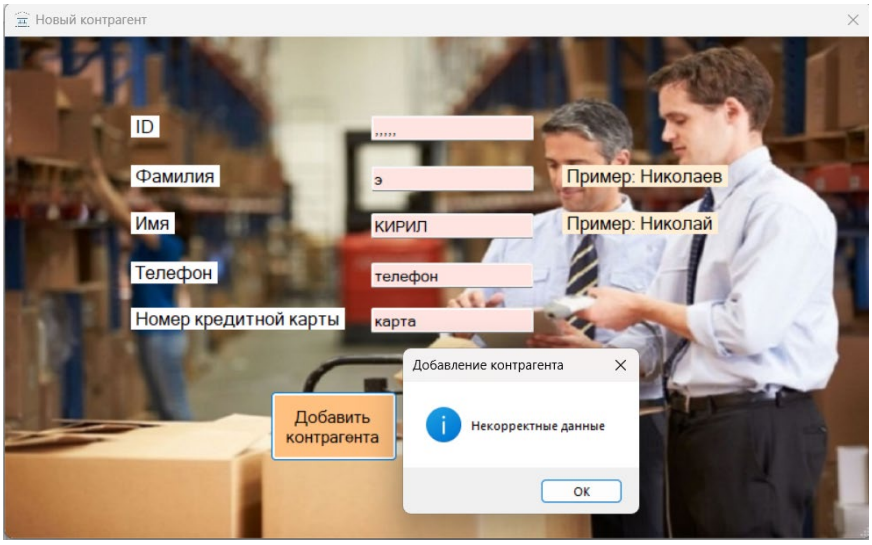

Рисунок А.5 – Форма добавления записи при наличии неверных полей

Если же форма при нажатии на кнопку «Добавить позицию» закрылась, то добавление позиции прошло успешно и запись отобразится в таблице.

Для удаления позиции нужно выбрать строку в таблице, которую необходимо удалить и нажать на кнопку «Удалить позицию». После чего позиция удалится из таблицы.

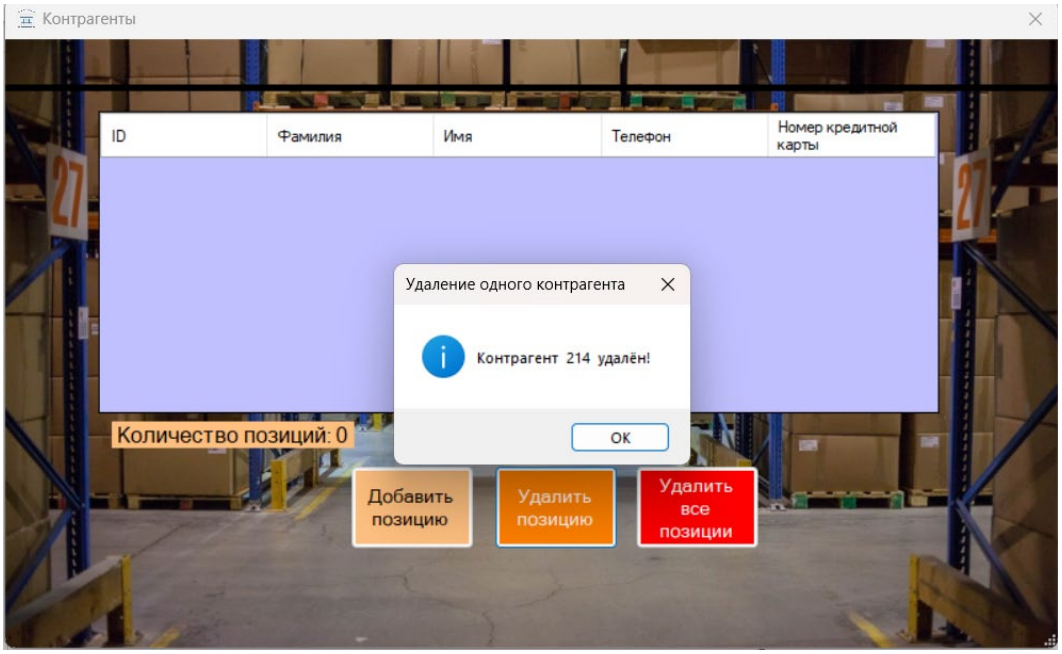

Рисунок А.6 – Удаление контрагента

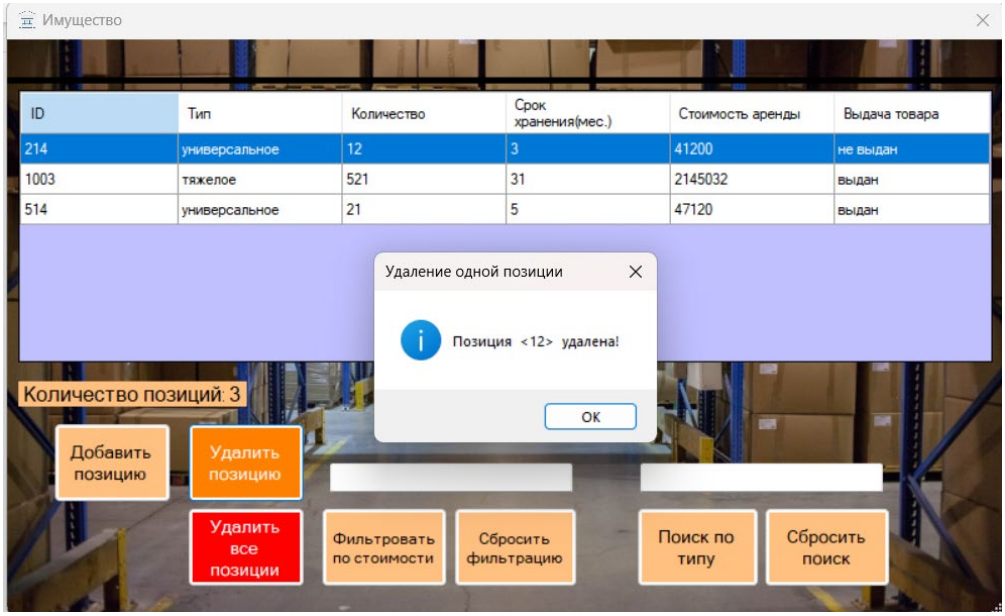

Рисунок А.7 – Удаления имущества

Для фильтрации имущества по стоимости необходимо ввести соответствующее значение в текстовое поле. После нажатия на кнопку «Фильтровать по стоимости» в таблице отобразятся позиции, удовлетворяющие критерию.

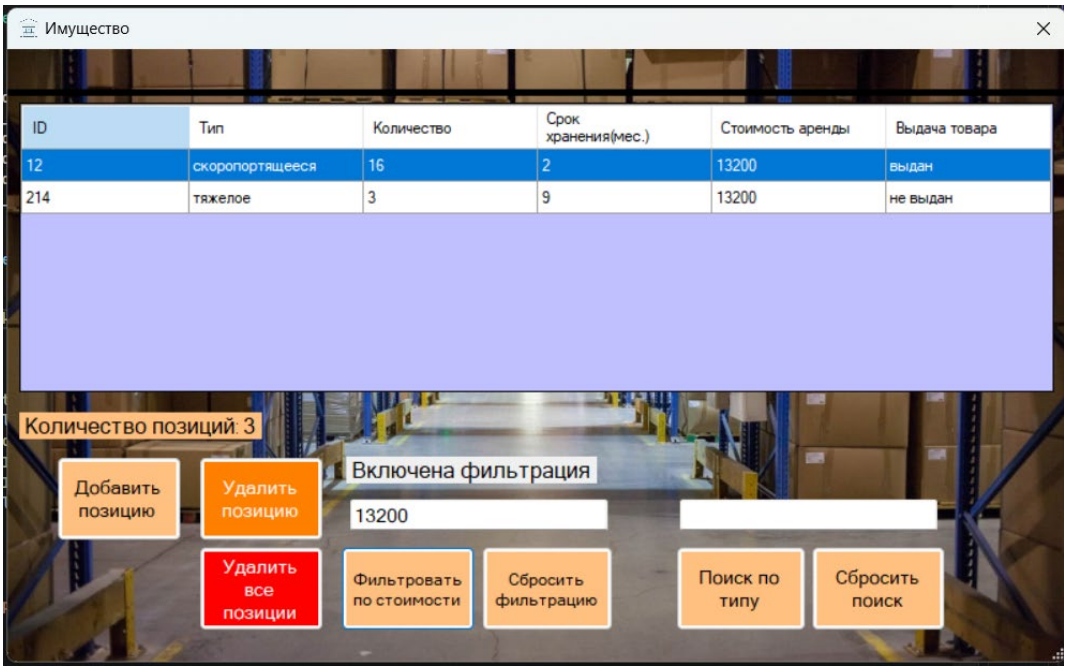

Рисунок А.8 – Применение фильтрации по стоимости

В случае отсутствия текста для фильтрации на экране появится сообщение с просьбой заполнить поле.

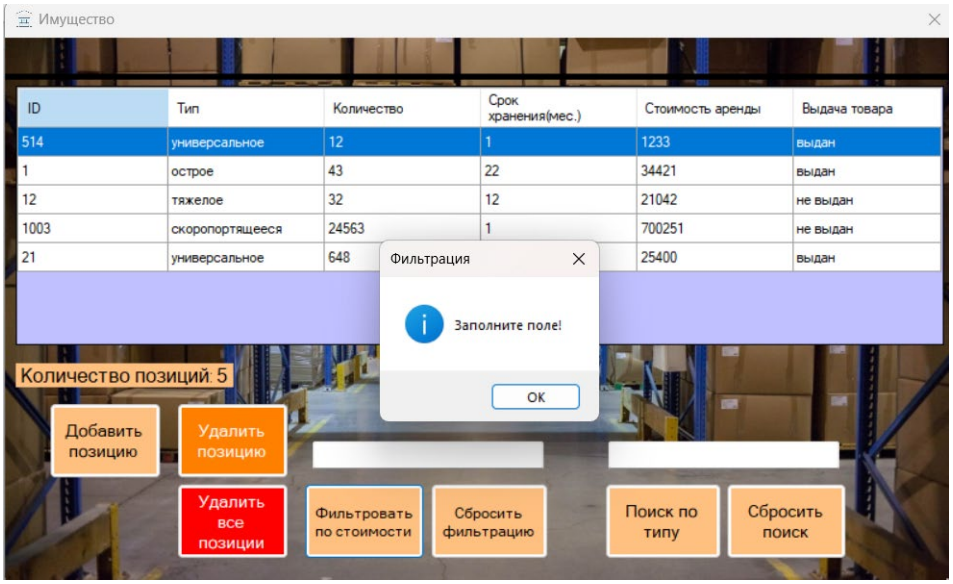

Рисунок А.9 – Сообщение о пустом текстовом поле при фильтрации

Чтобы показать все существующие позиции необходимо нажать кнопку «Сбросить фильтрацию».

Для сортировки позиций необходимо нажать на заголовок той колонки, по которой необходимо отсортировать записи.

При повторном нажатии на заголовки позиции будут отсортированы в обратном порядке.

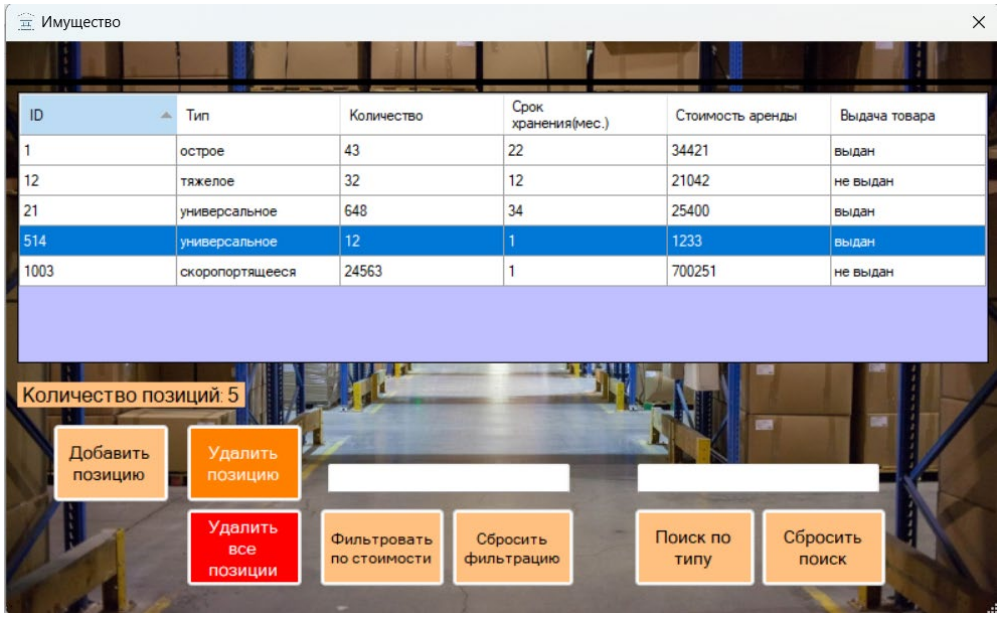

Рисунок А.10 – Первое нажатие на колонку «ID»

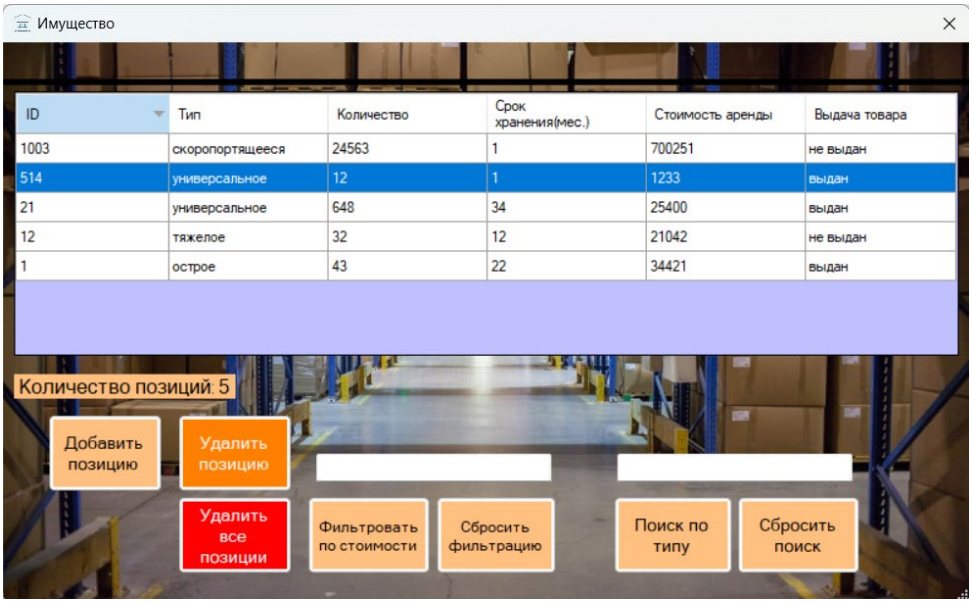

Рисунок А.11 – Повторное нажатие на колонку «ID»

Для поиска имущества по типу необходимо ввести соответствующее значение в текстовое поле. После нажатия на кнопку «Поиск по типу» в таблице подсветится позиция, удовлетворяющая критерию.

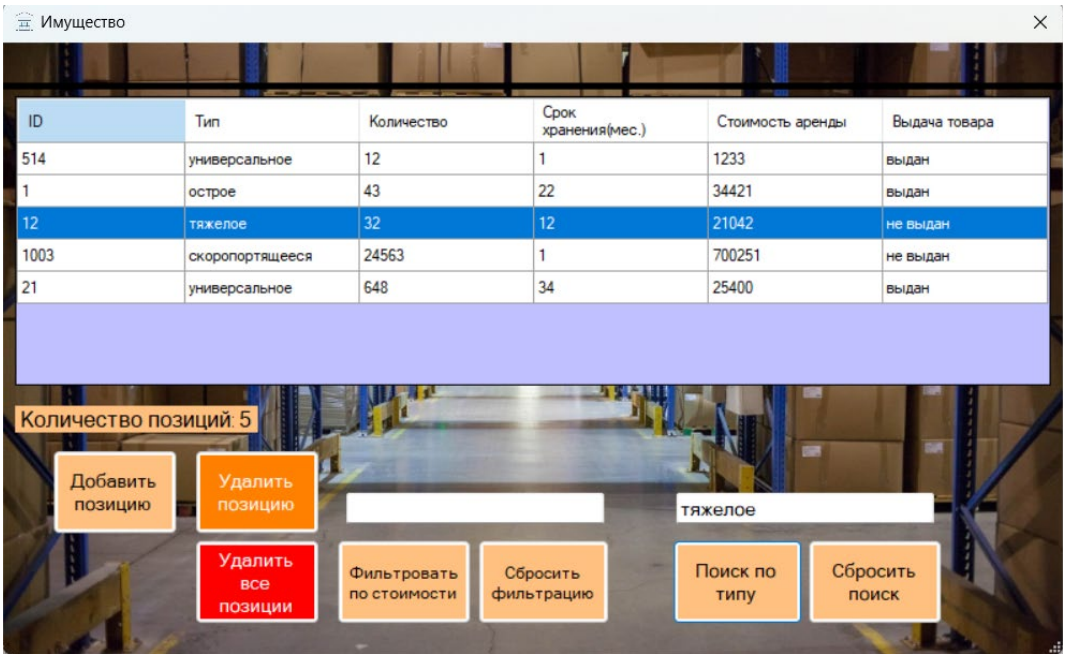

Рисунок А.12 – Применение поиска по типу

В случае отсутствия текста для поиска на экране появится сообщение с просьбой заполнить поле.

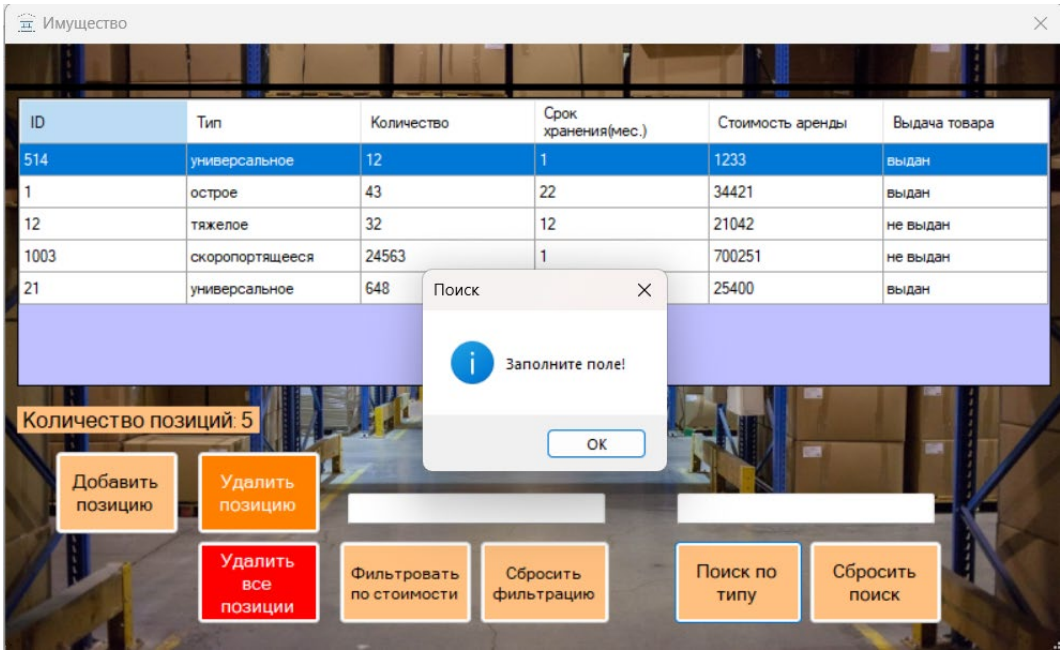

Рисунок А.13 – Сообщение о пустом текстовом поле при поиске

Чтобы переместить выделение на первую позицию необходимо нажать кнопку «Сбросить поиск».

Для удаления всех существующих данных необходимо нажать кнопку «Удалить все позиции». В случае успешного удаления таблица станет пустой.

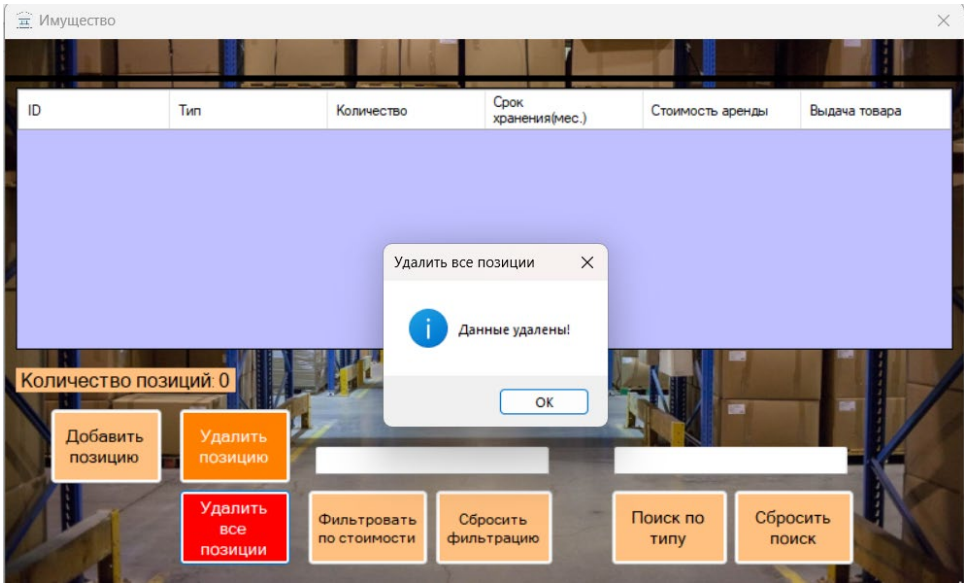

Рисунок А.14 – Вид таблицы после удаления всех позиций имущества

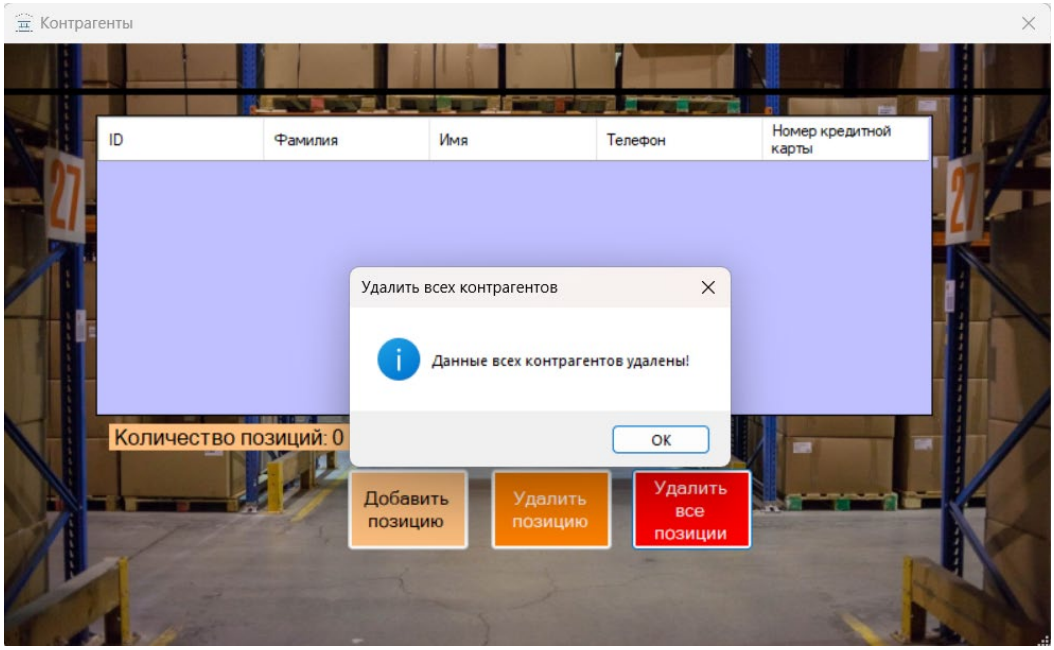

Рисунок А.15 – Вид таблицы после удаления всех контрагентов

Приложение В - Исходные тексты программы

Файл Start.cs

```
using System;
using System.Windows.Forms;

namespace storage
{
    public partial class Start : Form
    {
        private System.Windows.Forms.Timer timer;

        public Start()
        {
            InitializeComponent();
            timer = new System.Windows.Forms.Timer();
            timer.Tick += delegate
            {
                this.Close();
            };
            timer.Interval =
(int)TimeSpan.FromSeconds(2).TotalMilliseconds;
            timer.Start();
        }
    }
}
```

Файл Menu.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 storage
{
    public partial class Menu : Form
    {
        public Menu()
        {
            InitializeComponent();
        }
        private void Users_Click(object sender, EventArgs e)
        {
            Users User = new Users();
            User.ShowDialog();
        }
        private void Products_Click(object sender, EventArgs e)
        {
```

```
            Products Product = new Products();
            Product.ShowDialog();
        }
    }
}
```

Файл Users.cs

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Net.NetworkInformation;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace storage
{
    public partial class Users : Form
    {
        const string FILE_PRODUCTS = "Products.json";
        const string FILE_USERS = "Users.json";

        int n = 0;
        int numbers = 0;
        public Users()
        {
            Task.Run(() => File.Open(FILE_USERS,
FileMode.OpenOrCreate).Close());

            Task.Run(() => File.Open(FILE_PRODUCTS,
FileMode.OpenOrCreate).Close());

            InitializeComponent();
        }

        //запись в файл json
        async Task WriteToFile<T>(List<T> data, string FILE_NAME)
        {
            using (var streamWriter = new StreamWriter(FILE_NAME,
false))
            {
                await streamWriter.WriteAsync(await Task.Run(() =>
JsonConvert.SerializeObject(data)));
            }
        }

        //чтение из файла json
        async Task<List<T>> ReadFromFile<T>(string FILE_NAME)
```

```
        {
            using (var streamReader = new StreamReader(FILE_NAME))
            {
                return await Task.Run(async () =>
JsonConvert.DeserializeObject<List<T>>(await
streamReader.ReadToEndAsync()) ?? new List<T>());
            }
        }

        async void ad_us_Click(object sender, EventArgs e)
        {
            User_add FormAddUser = new User_add();
            FormAddUser.ShowDialog();

            uint idForm = User_add.IdForm;
            string famForm = User_add.FamForm;
            string nameForm = User_add.NameForm;
            uint phoneForm = User_add.PhoneForm;
            uint creditcForm = User_add.CreditcForm;

            InfoUsers newUser = new InfoUsers(idForm, famForm,
nameForm, phoneForm, creditcForm);

            if ((idForm > 0) && !string.IsNullOrEmpty(famForm) &&
!string.IsNullOrEmpty(nameForm) && (phoneForm > 0) && (creditcForm >
0))
            {
                var users = await ReadFromFile<InfoUsers>(FILE_USERS);

                if (!users.Contains(newUser))
                {
                    foreach (var us in users)
                    {
                        if (us.ID == idForm)
                        {
                            MessageBox.Show($"Контрагент {us.ID} уже
занесён в базу. "
                                , "Добавление контрагента", 0,
MessageBoxIcon.Information);
                            return;
                        }
                    }

                    users.Add(newUser);

                    n = users.Count;
                    count_of_users.Text = Convert.ToString(n);

                    await WriteToFile(users, FILE_USERS);

                    dataGridView1.Rows.Add();
                    dataGridView1.Rows[numbers].Cells[0].Value =
idForm;
```

```
                    dataGridView1.Rows[numbers].Cells[1].Value =
famForm;
                    dataGridView1.Rows[numbers].Cells[2].Value =
nameForm;
                    dataGridView1.Rows[numbers].Cells[3].Value =
phoneForm;
                    dataGridView1.Rows[numbers].Cells[4].Value =
creditcForm;
                    numbers++;
                }
                else
                {
                    MessageBox.Show($"Этот контрагент был занесён в
базу ранее", "Добавление контрагента", 0, MessageBoxIcon.Information);
                    return;
                }
            }
        }

        async void del_us_Click(object sender, EventArgs e)
        {
            Int32 selectCount =

dataGridView1.GetCellCount(DataGridViewElementStates.Selected);
            uint idForm = User_add.IdForm;
            if (selectCount > 0)
            {
                var users = await ReadFromFile<InfoUsers>(FILE_USERS);
                var products = await
ReadFromFile<InfoProducts>(FILE_PRODUCTS);
                uint id =
Convert.ToUInt32(dataGridView1.SelectedCells[0].Value.ToString());
                string fam =
dataGridView1.SelectedCells[1].Value.ToString();
                string name =
dataGridView1.SelectedCells[2].Value.ToString();
                uint phone =
Convert.ToUInt32(dataGridView1.SelectedCells[3].Value.ToString());
                uint creditc =
Convert.ToUInt32(dataGridView1.SelectedCells[4].Value.ToString());
                var flag = true;
                bool chet = false;
                foreach (var pr in products)
                {
                    if (id == pr.ID)
                    {
                        chet = true;

                    }

                }
                if (chet)
                {
                    flag = false;
```

```
                    MessageBox.Show($"Вы не можете удалить
контрагента, так как" +
                        $" на складе хранится его имущество!",
"Удаление одного контрагента", 0, MessageBoxIcon.Information);
                    return;
                }
                if (flag)
                {
                    foreach (var us in users)
                    {
                        if (id == us.ID && fam == us.Fam && name ==
us.Name && phone == us.Phone && creditc == us.Creditc)
                        {
                            users.Remove(us);
                            n = users.Count;
                            count_of_users.Text = Convert.ToString(n);

dataGridView1.Rows.Remove(dataGridView1.CurrentRow);
                            numbers--;
                            MessageBox.Show($"Контрагент  {us.ID}
удалён!", "Удаление одного контрагента", 0,
MessageBoxIcon.Information);
                            break;
                        }
                    }

                    await WriteToFile(users, FILE_USERS);
                }

            }
            else
            {
                MessageBox.Show("Нет ни одного контрагента!",
"Удаление", 0, MessageBoxIcon.Information);
                return;
            }
        }

        async void del_allpr_Click(object sender, EventArgs e)
        {
            var users = await ReadFromFile<InfoUsers>(FILE_USERS);

            var products = await
ReadFromFile<InfoProducts>(FILE_PRODUCTS);


            if (dataGridView1.Rows.Count == 0)
            {
                MessageBox.Show("Нет ни одного контрагента!", "Удалить
всех контрагентов", 0, MessageBoxIcon.Information);
            }
            else if (products.Count > 0)
            {
```

```
                MessageBox.Show("Вы не можете удалить всех
контрагентов, так как в базе склада есть имущество!",
                    "Удалить всех контрагентов", 0,
MessageBoxIcon.Information);
            }
            else
            {
                users.Clear();
                n = users.Count;
                count_of_users.Text = Convert.ToString(n);
                dataGridView1.Rows.Clear();
                numbers = 0;
                MessageBox.Show("Данные всех контрагентов удалены!",
"Удалить всех контрагентов", 0, MessageBoxIcon.Information);

            }

            await WriteToFile(users, FILE_USERS);
        }

        async void Users_Load(object sender, EventArgs e)
        {
            if (File.Exists(FILE_USERS))
            {

                var table_of_users = await
ReadFromFile<InfoUsers>(FILE_USERS);

                n = table_of_users.Count;
                count_of_users.Text = Convert.ToString(n);

                if (table_of_users != null)

                    foreach (var user in table_of_users)
                    {
                        dataGridView1.Rows.Add();
                        dataGridView1.Rows[numbers].Cells[0].Value =
user.ID;
                        dataGridView1.Rows[numbers].Cells[1].Value =
user.Fam;
                        dataGridView1.Rows[numbers].Cells[2].Value =
user.Name;
                        dataGridView1.Rows[numbers].Cells[3].Value =
user.Phone;
                        dataGridView1.Rows[numbers].Cells[4].Value =
user.Creditc;
                        numbers++;
                    }
            }
        }
    }
}
```

Файл User_add.cs

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Linq;
using System.Reflection.Emit;
using System.Text;
using System.Text.RegularExpressions;
using System.Threading.Tasks;
using System.Windows.Forms;
using System.Xml.Linq;

namespace storage
{
    public partial class User_add : Form
    {
        public static uint IdForm = 0;
        public static string FamForm = "";
        public static string NameForm = "";
        public static uint PhoneForm = 0;
        public static uint CreditcForm = 0;

        public User_add()
        {
            InitializeComponent();
        }


        bool CheckOnCorrectTextBox(TextBox tb) =>
!Regex.IsMatch(tb.Text, @"^\s*$") &&
            Regex.IsMatch(tb.Text, @"^[А-ЯЁ]+[а-яё]") || (tb.BackColor
= Color.MistyRose) != Color.MistyRose
            || (lb1.Text = "Пример: Николаев") != "Пример: Николаев";

        bool CheckOnCorrectTextBox2(TextBox tb) =>
!Regex.IsMatch(tb.Text, @"^\s*$") &&
           Regex.IsMatch(tb.Text, @"^[А-ЯЁ]+[а-яё]") || (tb.BackColor
= Color.MistyRose) != Color.MistyRose
           || (lb2.Text = "Пример: Николай") != "Пример: Николай";

        bool CheckOnCorrectNumberBox(TextBox nb) =>
uint.TryParse(nb.Text, out _) || (nb.BackColor = Color.MistyRose) !=
Color.MistyRose;

        bool CheckOnCorrectNumberBox2(TextBox nb) =>
uint.TryParse(nb.Text, out _) || (nb.BackColor = Color.MistyRose) !=
Color.MistyRose;

        bool CheckOnCorrectNumberBox3(TextBox nb) =>
uint.TryParse(nb.Text, out _) || (nb.BackColor = Color.MistyRose) !=
Color.MistyRose;
        bool FlagCorrect =>
            CheckOnCorrectNumberBox(add_ID) &
```

```
            CheckOnCorrectTextBox(add_fam) &
            CheckOnCorrectTextBox2(add_name) &
            CheckOnCorrectNumberBox2(add_phone) &
            CheckOnCorrectNumberBox3(add_creditc);

        void Control_Click(object sender, EventArgs e)
        {
            lb1.Text = "";
            lb2.Text = "";
        }

        private void addus_button_Click(object sender, EventArgs e)
        {
            if (FlagCorrect)
            {
                IdForm = Convert.ToUInt32(add_ID.Text);
                FamForm = add_fam.Text;
                NameForm = add_name.Text;
                PhoneForm = Convert.ToUInt32(add_phone.Text);
                CreditcForm = Convert.ToUInt32(add_creditc.Text);

                Close();

            }
            else
                MessageBox.Show($"Некорректные данные", "Добавление
контрагента", 0, MessageBoxIcon.Information);
        }
    }
}
```

Файл Products.cs

```
using Newtonsoft.Json;
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.IO;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Forms;

namespace storage
{
    public partial class Products : Form
    {
        const string FILE_PRODUCTS = "Products.json";
        const string FILE_USERS = "Users.json";

        int n_pr = 0;
        int numberr = 0;
```

```
        public Products()
        {
            Task.Run(() => File.Open(FILE_USERS,
FileMode.OpenOrCreate).Close());

            Task.Run(() => File.Open(FILE_PRODUCTS,
FileMode.OpenOrCreate).Close());

            InitializeComponent();
        }

        //запись в файл json
        async Task WriteToFile<T>(List<T> data, string FILE_NAME)
        {
            using (var streamWriter = new StreamWriter(FILE_NAME,
false))
            {
                await streamWriter.WriteAsync(await Task.Run(() =>
JsonConvert.SerializeObject(data)));
            }
        }

        //чтение из файла json
        async Task<List<T>> ReadFromFile<T>(string FILE_NAME)
        {
            using (var streamReader = new StreamReader(FILE_NAME))
            {
                return await Task.Run(async () =>
JsonConvert.DeserializeObject<List<T>>(await
streamReader.ReadToEndAsync()) ?? new List<T>());
            }
        }


        async void ad_pr_Click(object sender, EventArgs e)
        {
            Product_add FormAdd = new Product_add();
            FormAdd.ShowDialog();

            uint idForm = Product_add.IdForm;
            string typeForm = Product_add.TypeForm;
            uint numberForm = Product_add.NumberForm;
            uint lifeForm = Product_add.LifeForm;
            uint costForm = Product_add.CostForm;
            string giveForm = Product_add.GiveForm;
            bool chet = false;
            var users = await ReadFromFile<InfoUsers>(FILE_USERS);

            InfoProducts newProduct = new InfoProducts(idForm,
typeForm, numberForm, lifeForm, costForm, giveForm);
```

```
            if ((idForm > 0) && !string.IsNullOrEmpty(typeForm) &&
(numberForm > 0) && (lifeForm > 0) && (costForm > 0) &&
!string.IsNullOrEmpty(giveForm))
            {
                var products = await
ReadFromFile<InfoProducts>(FILE_PRODUCTS);


                if (!products.Contains(newProduct))
                {
                    foreach (var us in users)
                    {
                        if (idForm == us.ID)
                        {
                            chet = true;

                        }
                    }
                    if (!chet)
                    {
                    MessageBox.Show($"Имущество не может быть
добавлено, " +
                                $"так как в базе контрагентов нет
соответствующего ID!" +
                                $" {Environment.NewLine} " +
                                $"Совет: На вкладке <Контрагенты> добавьте
контрагента с соответствующим ID!"
                                , "Добавление позиции", 0,
MessageBoxIcon.Information);
                    return;
                    }
                    foreach (var pr in products)
                    {
                        if (pr.ID == idForm)
                        {
                            MessageBox.Show($"Имущество с таким ID
<{pr.ID}>  уже есть в базе. "
                                    , "Добавление позиции", 0,
MessageBoxIcon.Information);
                            return;
                        }
                    }

                    products.Add(newProduct);

                    n_pr = products.Count;
                    count_of_products.Text = Convert.ToString(n_pr);

                    await WriteToFile(products, FILE_PRODUCTS);

                    dataGridView2.Rows.Add();
                    dataGridView2.Rows[numberr].Cells[0].Value =
idForm;
```

```
                    dataGridView2.Rows[numberr].Cells[1].Value =
typeForm;
                    dataGridView2.Rows[numberr].Cells[2].Value =
numberForm;
                    dataGridView2.Rows[numberr].Cells[3].Value =
lifeForm;
                    dataGridView2.Rows[numberr].Cells[4].Value =
costForm;
                    dataGridView2.Rows[numberr].Cells[5].Value =
giveForm;
                    numberr++;
                }
                else
                {
                    MessageBox.Show($"Данное имущество было занесено в
базу ранее", "Добавление позиции", 0, MessageBoxIcon.Information);
                    return;
                }
            }
        }

        async void Products_Load(object sender, EventArgs e)
        {
            if (File.Exists(FILE_PRODUCTS))
            {

                var table_of_products = await
ReadFromFile<InfoProducts>(FILE_PRODUCTS);

                n_pr = table_of_products.Count;
                count_of_products.Text = Convert.ToString(n_pr);

                if (table_of_products != null)

                    foreach (var products in table_of_products)
                    {
                        dataGridView2.Rows.Add();
                        dataGridView2.Rows[numberr].Cells[0].Value =
products.ID;
                        dataGridView2.Rows[numberr].Cells[1].Value =
products.Type;
                        dataGridView2.Rows[numberr].Cells[2].Value =
products.Number;
                        dataGridView2.Rows[numberr].Cells[3].Value =
products.Life;
                        dataGridView2.Rows[numberr].Cells[4].Value =
products.Cost;
                        dataGridView2.Rows[numberr].Cells[5].Value =
products.Give;
                        numberr++;
                    }
            }
        }
```

```
        async void del_pr_Click(object sender, EventArgs e)
        {
            Int32 selectCount =
dataGridView2.GetCellCount(DataGridViewElementStates.Selected);

            if (selectCount > 0)
            {

                var products = await
ReadFromFile<InfoProducts>(FILE_PRODUCTS);

                uint id =
Convert.ToUInt32(dataGridView2.SelectedCells[0].Value.ToString());
                string type =
dataGridView2.SelectedCells[1].Value.ToString();
                uint number =
Convert.ToUInt32(dataGridView2.SelectedCells[2].Value.ToString());
                uint life =
Convert.ToUInt32(dataGridView2.SelectedCells[3].Value.ToString());
                uint cost =
Convert.ToUInt32(dataGridView2.SelectedCells[4].Value.ToString());
                string give =
dataGridView2.SelectedCells[5].Value.ToString();

                foreach (var pr in products)
                {
                    if (id == pr.ID && type == pr.Type && number ==
pr.Number
                        && life == pr.Life && cost == pr.Cost && give
== pr.Give)
                    {
                        products.Remove(pr);
                        n_pr = products.Count;
                        count_of_products.Text =
Convert.ToString(n_pr);

dataGridView2.Rows.Remove(dataGridView2.CurrentRow);
                        numberr--;
                        MessageBox.Show($"Позиция  <{pr.ID}>
удалена!", "Удаление одной позиции", 0, MessageBoxIcon.Information);
                        break;
                    }
                }

                await WriteToFile(products, FILE_PRODUCTS);
            }
            else
            {
                MessageBox.Show("Нет ни одной позиции!", "Удаление
позиции", 0, MessageBoxIcon.Information);
                return;
            }
        }
```

```
        async void del_allpr_Click(object sender, EventArgs e)
        {
            var products = await
ReadFromFile<InfoProducts>(FILE_PRODUCTS);

            if (dataGridView2.Rows.Count != 0)
            {
                products.Clear();
                n_pr = products.Count;
                count_of_products.Text = Convert.ToString(n_pr);
                dataGridView2.Rows.Clear();
                numberr = 0;
                MessageBox.Show("Данные удалены!", "Удалить все
позиции", 0, MessageBoxIcon.Information);
            }
            else
            {
                MessageBox.Show("Нет ни одной позиции!", "Удалить все
позиции", 0, MessageBoxIcon.Information);
            }

            await WriteToFile(products, FILE_PRODUCTS);
        }

        async void filtr_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrEmpty(v_filtr.Text))
            {
                MessageBox.Show("Заполните поле!", "Фильтрация", 0,
MessageBoxIcon.Information);
                v_filtr.BackColor = Color.MistyRose;
            }
            else
            {
                string filtr = v_filtr.Text;

                dataGridView2.Rows.Clear();
                numberr = 0;

                bool flag = false;
                label2.Text = "Включена фильтрация";
                var products = await
ReadFromFile<InfoProducts>(FILE_PRODUCTS);

                foreach (var pr in products)
                {
                    if (pr.Cost == Convert.ToUInt32(filtr))
                    {
                        flag = true;
                        dataGridView2.Rows.Add();
                        dataGridView2.Rows[numberr].Cells[0].Value =
pr.ID;
```

```
                        dataGridView2.Rows[numberr].Cells[1].Value =
pr.Type;
                        dataGridView2.Rows[numberr].Cells[2].Value =
pr.Number;
                        dataGridView2.Rows[numberr].Cells[3].Value =
pr.Life;
                        dataGridView2.Rows[numberr].Cells[4].Value =
pr.Cost;
                        dataGridView2.Rows[numberr].Cells[5].Value =
pr.Give;
                        numberr++;
                    }
                }
                if (flag == false)
                {
                    MessageBox.Show($"Позиции с такой стоимостью
{filtr} не найдены!", "Фильтрация", 0, MessageBoxIcon.Information);
                }

            }
        }

        async void del_f_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrEmpty(v_filtr.Text))
            {
                MessageBox.Show($"Вы не применяли фильтрацию ранее!",
"Фильтрация", 0, MessageBoxIcon.Information);
                return;
            }
            v_filtr.Text = "";
            label2.Text = "";
            dataGridView2.Rows.Clear();
            numberr = 0;

            var products = await
ReadFromFile<InfoProducts>(FILE_PRODUCTS);

            foreach (var pr in products)
            {

                dataGridView2.Rows.Add();
                dataGridView2.Rows[numberr].Cells[0].Value = pr.ID;
                dataGridView2.Rows[numberr].Cells[1].Value = pr.Type;
                dataGridView2.Rows[numberr].Cells[2].Value =
pr.Number;
                dataGridView2.Rows[numberr].Cells[3].Value = pr.Life;
                dataGridView2.Rows[numberr].Cells[4].Value = pr.Cost;
                dataGridView2.Rows[numberr].Cells[5].Value = pr.Give;
                numberr++;
            }
        }

        async void escape_Click(object sender, EventArgs e)
```

```
        {
            if (string.IsNullOrEmpty(v_escape.Text))
            {
                MessageBox.Show("Заполните поле!", "Поиск", 0,
MessageBoxIcon.Information);
                v_escape.BackColor = Color.MistyRose;
            }
            else
            {
                string escape = v_escape.Text;

                numberr = 0;
                int i = 0;

                bool flag = false;

                var products = await
ReadFromFile<InfoProducts>(FILE_PRODUCTS);

                foreach (var pr in products)
                {
                    if (pr.Type == Convert.ToString(escape))
                    {
                        flag = true;
                        dataGridView2.Rows[i].Selected = true;
                        numberr++;
                    }
                    i++;
                }
                if (flag == false)
                {
                    MessageBox.Show($"Позиции с таким типом не
найдены!", "Поиск", 0, MessageBoxIcon.Information);
                }
            }
        }

        async void del_esc_Click(object sender, EventArgs e)
        {
            if (string.IsNullOrEmpty(v_escape.Text))
            {
                MessageBox.Show($"Вы не применяли поиск ранее!",
"Поиск", 0, MessageBoxIcon.Information);
                return;
            }
            v_escape.Text = "";
            dataGridView2.Rows.Clear();
            numberr = 0;

            var products = await
ReadFromFile<InfoProducts>(FILE_PRODUCTS);

            foreach (var pr in products)
            {
```

```
                dataGridView2.Rows.Add();
                dataGridView2.Rows[numberr].Cells[0].Value = pr.ID;
                dataGridView2.Rows[numberr].Cells[1].Value = pr.Type;
                dataGridView2.Rows[numberr].Cells[2].Value =
pr.Number;
                dataGridView2.Rows[numberr].Cells[3].Value = pr.Life;
                dataGridView2.Rows[numberr].Cells[4].Value = pr.Cost;
                dataGridView2.Rows[numberr].Cells[5].Value = pr.Give;
                numberr++;
            }
        }
    }
}
```

Файл Product_add.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.Text.RegularExpressions;
using System.Windows.Forms;

namespace storage
{
    public partial class Product_add : Form
    {
        public static uint IdForm = 0;
        public static string TypeForm = "";
        public static uint NumberForm = 0;
        public static uint LifeForm = 0;
        public static uint CostForm = 0;
        public static string GiveForm = "";

        public Product_add()
        {
            InitializeComponent();
        }
       bool CheckOnCorrectComboBox(ComboBox cb) => !(cb.SelectedItem
is null) || (cb.BackColor = Color.MistyRose) != Color.MistyRose;

        bool CheckOnCorrectNumberBox(TextBox nb) =>
uint.TryParse(nb.Text, out _) || (nb.BackColor = Color.MistyRose) !=
Color.MistyRose;

        bool FlagCorrect =>
            CheckOnCorrectNumberBox(add_id) &
            CheckOnCorrectComboBox(add_type) &
            CheckOnCorrectNumberBox(add_number) &
```

```
            CheckOnCorrectNumberBox(add_life) &
            CheckOnCorrectNumberBox(add_cost) &
            CheckOnCorrectComboBox(add_give);

        void Control_Click(object sender, EventArgs e) => (sender as
Control).BackColor = Color.WhiteSmoke;

        private void addpr_button_Click(object sender, EventArgs e)
        {
            if (FlagCorrect)
            {
                IdForm = Convert.ToUInt32(add_id.Text);
                TypeForm = add_type.SelectedItem as string;
                NumberForm = Convert.ToUInt32(add_number.Text);
                LifeForm = Convert.ToUInt32(add_life.Text);
                CostForm = Convert.ToUInt32(add_cost.Text);
                GiveForm = add_give.SelectedItem as string;

                Close();
            }
            else
                MessageBox.Show($"Некорректные данные", "Добавление
позиции", 0, MessageBoxIcon.Information);


        }
    }
}
```

Файл Infostorage.cs

```
using System.Xml.Linq;

namespace storage
{
    internal class Infostorage
    {
        public uint ID { get; set; }
    }
    class InfoProducts : Infostorage
    {
        public string Type { get; set; }
        public uint Number { get; set; }
        public uint Life { get; set; }
        public uint Cost { get; set; }
        public string Give { get; set; }

        public InfoProducts(uint id, string type, uint number, uint
life, uint cost, string give)
        {
            ID = id;
            Type = type;
            Number = number;
            Life = life;
```

```
            Cost = cost;
            Give = give;
        }
    }

    class InfoUsers : Infostorage
    {
        public string Fam { get; set; }
        public string Name { get; set; }
        public uint Phone { get; set; }
        public uint Creditc { get; set; }

        public InfoUsers(uint id, string fam, string name, uint phone,
uint creditc)
        {
            ID = id;
            Fam = fam;
            Name = name;
            Phone = phone;
            Creditc = creditc;
        }
    }
}
```