

Лабораторная работа №3

СОДЕРЖАНИЕ

1. ЗАДАНИЕ	4
2. ВАРИАНТЫ ЗАДАНИЙ	5
3. ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ*	10
4. ШАГИ ПО СОЗДАНИЮ ПРИЛОЖЕНИЯ	11
4.1 ШАГ 1: СОЗДАНИЕ ПРОЕКТА	11
4.2 ШАГ 2: ДОБАВЛЕНИЕ ФАЙЛОВ	14
4.2.1 СОЗДАНИЕ ПАПКИ MODELS	14
4.2.2 КЛАССЫ: PRODUCT.CS, ORDER.CS, ORDERITEM.CS	15
4.2.3 ДОБАВЛЕНИЕ КОНВЕРТЕРА	16
4.3 ШАГ 3: РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА ПРИЛОЖЕНИЯ	18
4.4 ШАГ 4: РЕАЛИЗАЦИЯ ЛОГИКИ (MAINWINDOW.XAML.CS)	21
4.4.1 КОНСТРУКТОР	23
4.4.2 МЕТОДЫ ОБНОВЛЕНИЯ UI	24
4.4.3 РАБОТА С ТОВАРАМИ (ДОБАВЛЕНИЕ, ОБНОВЛЕНИЕ, УДАЛЕНИЕ)	25

Лабораторная работа №3

СОДЕРЖАНИЕ

4.4.4 РАБОТА С ЗАКАЗАМИ (ДОБАВЛЕНИЕ, ОБНОВЛЕНИЕ, УДАЛЕНИЕ)	29
4.4.5 ИЗМЕНЕНИЕ КОЛИЧЕСТВА ТОВАРОВ (КНОПКИ + И -)	33
4.4.8 СБРОС ВЫДЕЛЕНИЯ В СПИСКАХ	35
5. ШАГ 5: ВНЕШНИЙ ВИД ИНТЕРФЕЙСА	36
6. ШАГ 6: ДЕМОНСТРАЦИЯ РАБОТЫ	37
6.1 ДОБАВЛЕНИЕ ТОВАРОВ.....	38
6.2 ДОБАВЛЕНИЕ ТОВАРОВ В ЗАКАЗ	39
6.3 СОЗДАНИЕ ЗАКАЗА.....	40
6.3 ПРОСМОТР СОДЕРЖИМОГО ЗАКАЗА.....	41
6.3 УДАЛЕНИЕ ТОВАРОВ ИЗ ЗАКАЗА.....	42
6.3 ОБНОВЛЕНИЕ И УДАЛЕНИЕ ЗАКАЗОВ.....	43

Лабораторная работа №3

Система управления заказами для онлайн-магазина

ЗАДАНИЕ

1. Реализовать CRUD-приложение (WPF).
2. Реализовать визуальное отображение списка товаров и списка заказов
3. Реализовать функционал добавления, редактирования, удаления, конкретных товаров, а также учет их количества. Реализовать функции уменьшения количества товаров при добавлении их в заказ.
4. Реализовать функционал добавления, редактирования, удаления, заказов, а также возможность редактирования количества товаров добавленных в заказ.
5. В списке заказов отобразить информацию о названии, дате (времени) заказа, общей сумме заказа и список входящих в него товаров и их количества
6. Реализовать функцию расчета остатков товара (при уменьшении количества товара в заказе, должно увеличиваться количество товара на складе (возвращаться). При удалении товара из заказа, также количество остатков должно увеличиваться (если было куплено 6 единиц товара и заказ удален (отменен) на складе должно стать на 6 единиц товара больше (вернуться))

Лабораторная работа №3

ВАРИАНТЫ ЗАДАНИЙ

1. Музыкальный магазин.

- Товары: гитары, синтезаторы, барабаны (поля: Name, Price, Stock, Id).
- Заказы: имя клиента, дата, список инструментов.
- Создать раздел «Инструменты», добавить поле «Тип» (например, струнные, клавишные).

2. Книжный магазин

- Товары: книги (поля: Name — название, Price, Stock, Id).
- Заказы: имя клиента, дата, список книг.
- Создать раздел «Книги», добавить поле «Автор» в Product.

3. Магазин электроники

- Товары: смартфоны, ноутбуки, наушники.
- Заказы: имя клиента, дата, список гаджетов.
- Создать раздел «Гаджеты», добавить поле «Бренд» (например, Apple, Samsung).

4. Спортивный магазин

- Товары: кроссовки, тренажёры, мячи.
- Заказы: имя клиента, дата, список товаров.
- Создать раздел «Спортивнвентарь», добавить поле «Категория» (обувь, оборудование).

5. Магазин одежды

- Товары: футболки, джинсы, куртки.
- Заказы: имя клиента, дата, список одежды.
- Создать раздел «Одежда», добавить поле «Размер» (S, M, L).

Лабораторная работа №3

ВАРИАНТЫ ЗАДАНИЙ

6. Цветочный магазин

- Товары: розы, тюльпаны, орхидеи.
- Заказы: имя клиента, дата, список букетов.
- Создать раздел «Цветы», добавить поле «Тип» (букет, горшок).

7. Магазин игрушек

- Товары: конструкторы, куклы, машинки.
- Заказы: имя клиента, дата, список игрушек.
- Создать раздел «Игрушки», добавить поле «Возраст» (3+, 6+).

8. Магазин бытовой техники

- Товары: холодильники, пылесосы, микроволновки.
- Заказы: имя клиента, дата, список техники.
- Создать раздел «Техника», добавить поле «Мощность» (Вт).

9. Магазин косметики

- Товары: помады, кремы, духи.
- Заказы: имя клиента, дата, список косметики.
- Создать раздел «Косметика», добавить поле «Тип» (уход, макияж).

10. Магазин автозапчастей

- Товары: фильтры, шины, аккумуляторы.
- Заказы: имя клиента, дата, список запчастей.
- Создать раздел «Запчасти», добавить поле «Марка» (Toyota, BMW).

11. Магазин мебели

- Товары: диваны, столы, шкафы.
- Заказы: имя клиента, дата, список мебели.
- Создать раздел «Мебель», добавить поле «Материал» (дерево, металл).

Лабораторная работа №3

ВАРИАНТЫ ЗАДАНИЙ

13. Магазин канцелярии

- Товары: ручки, тетради, маркеры.
- Заказы: имя клиента, дата, список товаров.
- Создать раздел «Канцелярия», добавить поле «Тип» (письмо, рисование).

14. Магазин ювелирных изделий

- Товары: кольца, серьги, браслеты.
- Заказы: имя клиента, дата, список украшений.
- Создать раздел «Украшения», добавить поле «Материал» (золото, серебро).

15. Магазин зоотоваров

- Товары: корма, игрушки, клетки.
- Заказы: имя клиента, дата, список товаров.
- Создать раздел «Зоотовары», добавить поле «Животное» (кошка, собака).

16. Магазин видеоигр

- Товары: игры для ПК, консолей.
- Заказы: имя клиента, дата, список игр.
- Создать раздел «Игры», добавить поле «Платформа» (PC, PS5).

17. Магазин садовых товаров

- Товары: семена, инструменты, горшки.
- Заказы: имя клиента, дата, список товаров.
- Создать раздел «Садовые товары», добавить поле «Тип» (растения, инструменты).

18. Магазин часов

- Товары: наручные часы, настенные часы.
- Заказы: имя клиента, дата, список часов.
- Создать раздел «Часы», добавить поле «Механизм» (кварцевый, механический).

Лабораторная работа №3

ВАРИАНТЫ ЗАДАНИЙ

19. Магазин обуви

- Товары: кроссовки, ботинки, туфли.
- Заказы: имя клиента, дата, список обуви.
- Создать раздел «Обувь», добавить поле «Размер» (36, 42).

20. Магазин настольных игр

- Товары: шахматы, монополия, карточные игры.
- Заказы: имя клиента, дата, список игр.
- Создать раздел «Игры», добавить поле «Игроков» (2-4, 4-8).

21. Магазин сувениров

- Товары: магниты, статуэтки, открытки.
- Заказы: имя клиента, дата, список сувениров.
- Создать раздел «Сувениры», добавить поле «Страна» (Россия, Италия).

22. Магазин парфюмерии

- Товары: духи, туалетная вода.
- Заказы: имя клиента, дата, список парфюма.
- Изменения: переименовать «Продукты» в «Парфюм», добавить поле «Объём» (50 мл, 100 мл).

23. Магазин строительных материалов

- Товары: краска, гвозди, доски.
- Заказы: имя клиента, дата, список материалов.
- Создать раздел «Материалы», добавить поле «Единица» (литры, кг).

24. Магазин для художников

- Товары: краски, кисти, холсты.
- Заказы: имя клиента, дата, список товаров.
- Создать раздел «Арт-товары», добавить поле «Тип» (масло, акварель).

Лабораторная работа №3

ВАРИАНТЫ ЗАДАНИЙ

25. Магазин чая и кофе

- Товары: чай, кофе, сиропы.
- Заказы: имя клиента, дата, список товаров.
- Создать раздел «Напитки», добавить поле «Вес» (100 г, 500 г).

26. Магазин для рыбалки

- Товары: удочки, приманки, катушки.
- Заказы: имя клиента, дата, список товаров.
- Создать раздел «Снаряжение», добавить поле «Тип» (спиннинг, фидер).

27. Магазин для кемпинга

- Товары: палатки, спальники, горелки.
- Заказы: имя клиента, дата, список товаров.
- Создать раздел «Снаряжение», добавить поле «Вес» (кг).

28. Магазин медицинских товаров

- Товары: тонометры, бинты, маски.
- Заказы: имя клиента, дата, список товаров.
- Создать раздел «Медтовары», добавить поле «Назначение» (диагностика, уход).

29. Магазин фильмов

- Товары: DVD, Blu-ray диски.
- Заказы: имя клиента, дата, список фильмов.
- Создать раздел «Фильмы», добавить поле «Жанр» (драма, комедия).

30. Магазин горнолыжного оборудования

- Товары: сноуборд, горнолыжные костюмы, шлемы.
- Заказы: имя клиента, дата, список оборудования.
- Создать раздел «Снаряжение», добавить поле «Вид» (сноуборд, лыжи).

Лабораторная работа №3

ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ



Реализовать удаление единственного товара в заказе. Если в заказе присутствует только один товар, то при его удалении должен удаляться и сам заказ. Количество данного товара должно возвращаться в остаток.



Добавить функционал снятия выделения товара (в списке выбранных для добавления в заказ) и снятие выделения с заказа при клике по пустому полю (внутри listBox).

Лабораторная работа №2



Шаг 1. Создание проекта

Для создания проекта необходимо выбрать:

- "Создать проект"
- "Приложение WPF (Майкрософт)" (Рисунок 1.1)

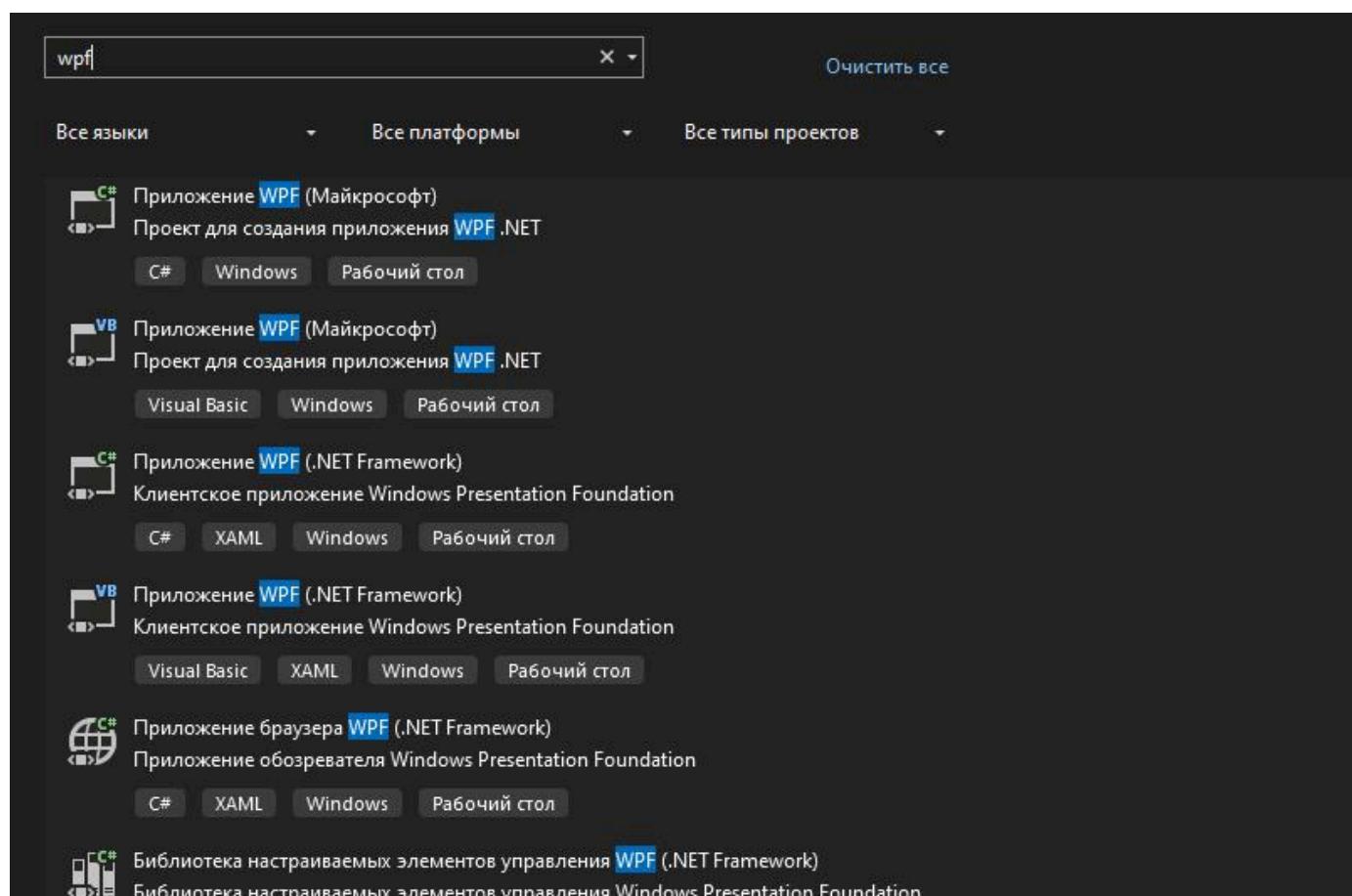


Рисунок 1.1 Создание проекта

Лабораторная работа №2



Шаг 1. Создание проекта

Далее задать имя проекта: StoreManager (Рисунок 1.2)

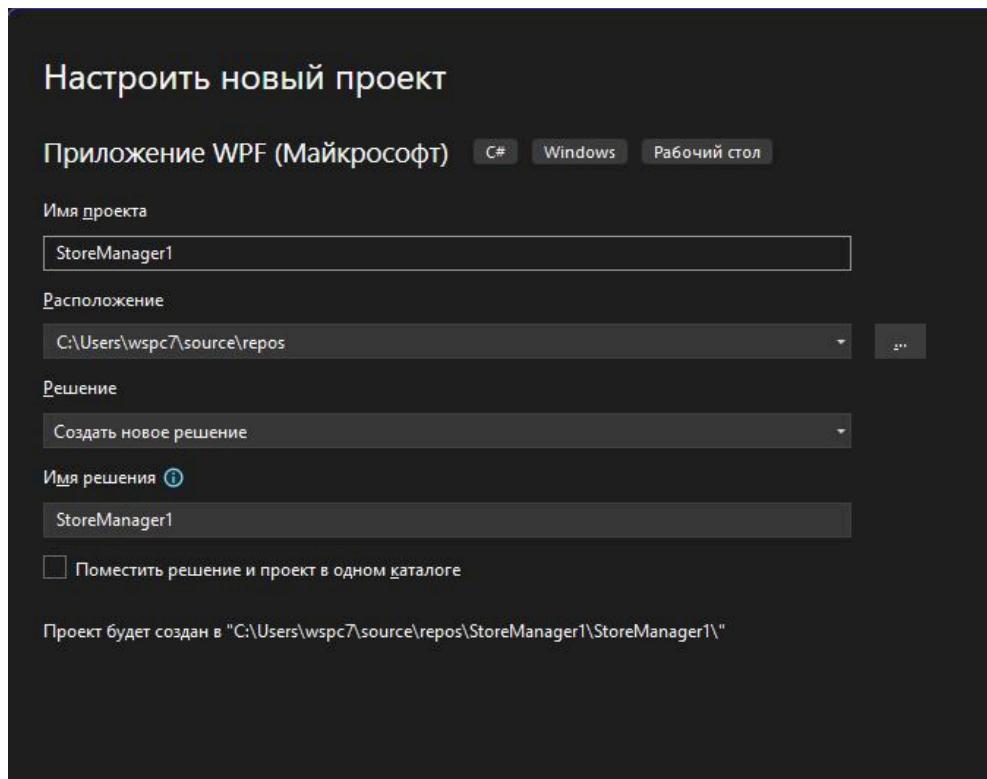


Рисунок 1.2. Задание имени проекта

Лабораторная работа №2



Шаг 1. Создание проекта

Выбрать платформу (пример выполнен на .net 8.0) (Рисунок 1.3)

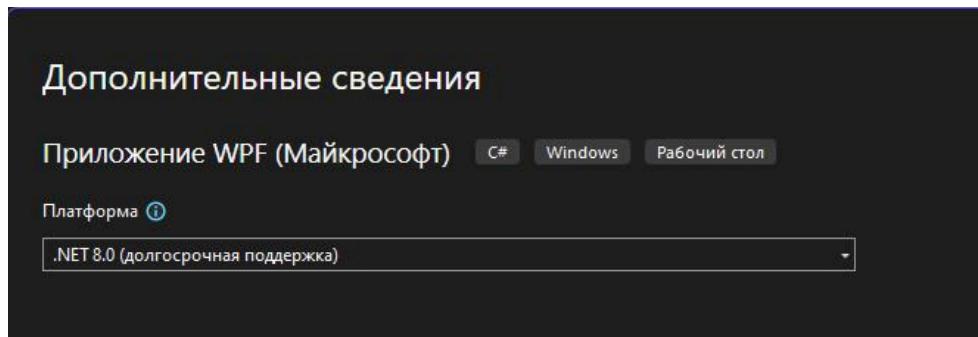


Рисунок 1.3. Задание имени проекта

Структура проекта состоит из следующих файлов:

- App.xaml — конфигурация приложения. Определяет основной класс приложения, наследуемый от Application. В нём задаются ресурсы приложения (например, стили, шаблоны) и стартовая точка (обычно указывается начальное окно в свойстве StartupUri);
- App.xaml.cs — логика приложения. Содержит класс App, который может переопределять методы, такие как OnStartup, для настройки поведения при запуске, или обрабатывать события приложения (например, Exit);
- MainWindow.xaml — разметка и структура UI (интерфейса) главного окна.;
- MainWindow.xaml.cs — логика и обработчики событий окна.

Лабораторная работа №2



Шаг 2. Добавление файлов

Необходимо добавить следующие файлы и папки:

- В корне проекта создать папку Models
- В папке Models добавить два новых файла классов: Product.cs и Order.cs.

Код для файла Order.cs приведен на рисунке 2.1.

```

1  namespace StoreManager.Models
2  {
3      // Модель заказа, содержащая информацию о клиенте и товарах
4      public class Order
5      {
6          // Уникальный идентификатор заказа
7          public int Id { get; set; }
8
9          // Список элементов заказа (товар + количество)
10         public List<OrderItem> Items { get; set; }
11
12         // Имя клиента
13         public string CustomerName { get; set; }
14
15         // Дата создания заказа
16         public DateTime OrderDate { get; set; }
17
18         // Общая сумма заказа, вычисляемая как сумма цен товаров умноженная на их количество
19         public decimal TotalPrice => Items.Sum(item => item.Product.Price * item.Quantity);
20
21         // Конструктор, инициализирующий пустой список товаров
22         public Order()
23         {
24             Items = new List<OrderItem>();
25         }
26     }
27 }
```

Рисунок 2.1 Класс Order.cs

Данный файл отвечает за создание нового объекта заказа. Может содержать уникальные поля не такие как в примере (в соответствии с вариантом).

Лабораторная работа №2



Шаг 2. Добавление файлов

Также необходим класс позволяющий связать товар и его количество в заказе. Пример, созданного класса OrderItem и объявления его свойств приведен на рисунке 2.2.

```
1  namespace StoreManager.Models
2  {
3      // Модель элемента заказа, связывающая товар и его количество
4      public class OrderItem
5      {
6          // Ссылка на товар
7          public Product Product { get; set; }
8
9          // Количество единиц товара в заказе
10         public int Quantity { get; set; }
11     }
12 }
```

Рисунок 2.2 Класс OrderItem

Также необходимо создать файл Product.cs. Данный файл отвечает за создание нового объекта товара. Пример, класса Product приведен на рисунке 2.3.

```
1  namespace StoreManager.Models
2  {
3      // Модель товара, представляющая продукт в магазине
4      public class Product
5      {
6          // Уникальный идентификатор товара
7          public int Id { get; set; }
8
9          // Название товара
10         public string Name { get; set; }
11
12         // Цена товара
13         public decimal Price { get; set; }
14
15         // Количество товара на складе
16         public int Stock { get; set; }
17     }
18 }
```

Рисунок 2.3 Класс Product.cs

Лабораторная работа №2



Шаг 2. Добавление файлов

Также необходимо создать папку Converters. Она будет нужна для создания конвертера. В данной папке нужно создать класс BooleanToVisibilityConverter. Пример разметки для данного класса приведен на рисунке 2.4.

```

1  using System;
2  using System.Globalization;
3  using System.Windows;
4  using System.Windows.Data;
5
6  namespace StoreManager.Converters
7  {
8      Ссылка 0
9      public class BooleanToVisibilityConverter : IValueConverter
10     {
11         Ссылка 0
12         public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
13         {
14             if (value is bool isVisible)
15             {
16                 return isVisible ? Visibility.Visible : Visibility.Collapsed;
17             }
18             return Visibility.Collapsed;
19         }
20
21         Ссылка 0
22         public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
23         {
24             throw new NotImplementedException();
25         }
26     }
}

```

Рисунок 2.4 Разметка для файла BooleanToVisibilityConverter.cs

Этот класс реализует интерфейс **IValueConverter**, который используется для преобразования данных при привязке (Data Binding).

- Метод **Convert** преобразует значение типа **bool** в **Visibility** (перечисление WPF для управления видимостью элементов):
- **true** → **Visibility.Visible** (элемент виден).
- **false** → **Visibility.Collapsed** (элемент скрыт, не занимает место в макете).

Лабораторная работа №2



Шаг 2. Добавление файлов

- Метод ConvertBack не реализован, так как обратное преобразование не требуется. Конвертеры в WPF используются для преобразования данных между источником (моделью) и целью (элементом UI). Например, конвертер позволяет адаптировать значение свойства модели к формату, подходящему для отображения. В данном случае конвертер применяется для реализации эффекта placeholder (подсказки в текстовых полях), показывая текст, когда поле пустое (Text.IsEmpty).

Свойство Visibility в WPF имеет три значения:

- Visible — элемент отображается.
- Collapsed — элемент скрыт и не занимает места.
- Hidden — элемент скрыт, но занимает место в макете.

Лабораторная работа №2



Шаг 3. Реализация интерфейса приложения

Для того чтобы реализовать внешний вид приложения необходимо добавить соответствующую разметку в файл MainWindow.xaml

Пример разметки для данного файла будет представлен ниже на нескольких рисунках (Рисунок 3.1, Рисунок 3.2, Рисунок 3.3).

```

1  <Window x:Class="StoreManager.MainWindow"
2      xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
3      xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
4      xmlns:converters="clr-namespace:StoreManager.Converters"
5      Title="Управление магазином" Height="650" Width="950"
6      WindowStartupLocation="CenterScreen" Background="#F5F5F5">
7
8      <!-- Ресурсы окна: конвертеры и стили --&gt;
9      &lt;Window.Resources&gt;
10         &lt;!-- Конвертер для преобразования bool в Visibility --&gt;
11         &lt;converters:BooleanToVisibilityConverter x:Key="BooleanToVisibilityConverter"/&gt;
12
13         &lt;!-- Общий стиль кнопок --&gt;
14         &lt;Style TargetType="Button"&gt;
15             &lt;Setter Property="Background" Value="#FF6200EE"/&gt;
16             &lt;Setter Property="Foreground" Value="White"/&gt;
17             &lt;Setter Property="FontSize" Value="14"/&gt;
18             &lt;Setter Property="Padding" Value="10,5"/&gt;
19             &lt;Setter Property="Margin" Value="5"/&gt;
20             &lt;Setter Property="BorderThickness" Value="0"/&gt;
21             &lt;Setter Property="Cursor" Value="Hand"/&gt;
22         &lt;/Style&gt;
23
24         &lt;!-- Специальный стиль кнопок изменения количества --&gt;
25         &lt;Style x:Key="QuantityButtonStyle" TargetType="Button"&gt;
26             &lt;Setter Property="Background" Value="#FF6200EE"/&gt;
27             &lt;Setter Property="Foreground" Value="White"/&gt;
28             &lt;Setter Property="FontSize" Value="12"/&gt;
29             &lt;Setter Property="Width" Value="25"/&gt;
30             &lt;Setter Property="Height" Value="25"/&gt;
31             &lt;Setter Property="Margin" Value="5,0"/&gt;
32             &lt;Setter Property="BorderThickness" Value="0"/&gt;
33             &lt;Setter Property="Cursor" Value="Hand"/&gt;
34         &lt;/Style&gt;
35
36         &lt;!-- Стиль для полей ввода --&gt;
37         &lt;Style TargetType="TextBox"&gt;
38             &lt;Setter Property="FontSize" Value="14"/&gt;
39             &lt;Setter Property="Padding" Value="5"/&gt;
40             &lt;Setter Property="Margin" Value="5"/&gt;
41             &lt;Setter Property="BorderBrush" Value="#FFCCCCCC"/&gt;
42         &lt;/Style&gt;
</pre>

```

Рисунок 3.1 Пример разметки главного окна (Часть 1)

Лабораторная работа №2



Шаг 3. Реализация интерфейса приложения

```

43      <!-- Стиль текста-заглушки (placeholder) -->
44      <Style TargetType="TextBlock" x:Key="PlaceholderStyle">
45          <Setter Property="Foreground" Value="#Gray"/>
46          <Setter Property="FontStyle" Value="Italic"/>
47          <Setter Property="Margin" Value="8,5,0,0"/>
48          <Setter Property="IsHitTestVisible" Value="False"/>
49      </Style>
50  </Window.Resources>
51
52
53  <!-- Основная сетка, задаёт структуру окна -->
54  <Grid Margin="-16,0,0,-35">
55      <Grid.ColumnDefinitions>
56          <!-- Колонки для панели товаров и заказов -->
57          <ColumnDefinition Width="107*"/>
58          <ColumnDefinition Width="344*"/>
59          <ColumnDefinition Width="450*"/>
60      </Grid.ColumnDefinitions>
61
62  <!-- Панель товаров -->
63  <Border Grid.Column="0" Margin="10" Background="#White" CornerRadius="10" Padding="10" Grid.ColumnSpan="2">
64      <StackPanel>
65          <TextBlock Text="Товары" FontSize="20" FontWeight="Bold" Margin="0,0,0,10"/>
66
67          <!-- Список товаров -->
68          <ListBox x:Name="ProductsList" Height="250" BorderBrush="#FFCCCCCC"
69              SelectionChanged="ProductsList_SelectionChanged"
70              MouseLeftButtonDown="ProductsList_MouseLeftButtonDown">
71              <ListBox.ItemTemplate>
72                  <DataTemplate>
73                      <StackPanel Orientation="Horizontal">
74                          <!-- Название -->
75                          <TextBlock Text="{Binding Name}" FontWeight="Bold" Margin="0,0,10,0"/>
76                          <!-- Цена -->
77                          <TextBlock Text="{Binding Price, StringFormat={}{{0:C}}}" />
78                          <!-- Остаток -->
79                          <TextBlock Text="({Остаток: " FontStyle="Italic"/>
80                          <TextBlock Text="{Binding Stock}">
81                              <!-- Подсветка остатка -->
82                              <TextBlock.Style>
83                                  <Style TargetType="TextBlock">
84                                      <Style.Triggers>
85                                          <!-- Если 0 - красным -->
86                                          <DataTrigger Binding="{Binding Stock}" Value="0">
87                                              <Setter Property="Foreground" Value="Red"/>
88                                          </DataTrigger>
89                                          <!-- Если меньше 5 - оранжевым -->
90                                          <DataTrigger Binding="{Binding Stock, ConverterParameter=5}" Value="True">
91                                              <Setter Property="Foreground" Value="Orange"/>
92                                          </DataTrigger>
93                                      </Style.Triggers>
94                                  </Style>
95                              </TextBlock.Style>
96                          </TextBlock>
97                          <TextBlock Text="}"/>
98                      </StackPanel>
99                  </DataTemplate>
100             </ListBox.ItemTemplate>
101         </ListBox>
102
103         <!-- Кнопка добавления в заказ -->
104         <Button x:Name="AddToOrderButton" Content="Добавить в заказ" Click="AddToOrder_Click" HorizontalAlignment="Right"/>
105
106
107         <!-- Поля ввода нового товара -->
108         <Grid>
109             <TextBox x:Name="ProductName"/>
110             <TextBlock Text="Название товара" Style="{StaticResource PlaceholderStyle}"
111                 Visibility="{Binding ElementName=ProductName, Path=Text.IsEmpty, Converter={StaticResource BooleanToVisibilityConverter}}"/>
112         </Grid>
113         <Grid>
114             <TextBox x:Name="ProductPrice"/>
115             <TextBlock Text="Цена" Style="{StaticResource PlaceholderStyle}"
116                 Visibility="{Binding ElementName=ProductPrice, Path=Text.IsEmpty, Converter={StaticResource BooleanToVisibilityConverter}}"/>
117         </Grid>
118         <Grid>
119             <TextBox x:Name="ProductStock"/>
120             <TextBlock Text="Остаток" Style="{StaticResource PlaceholderStyle}"
121                 Visibility="{Binding ElementName=ProductStock, Path=Text.IsEmpty, Converter={StaticResource BooleanToVisibilityConverter}}"/>
122         </Grid>

```

Рисунок 3.2 Пример разметки главного окна (Часть 2)

Лабораторная работа №2



Шаг 3. Реализация интерфейса приложения

```

121 </Grid>
122
123 <!-- Кнопки управления товарами -->
124 <StackPanel HorizontalAlignment="Right" Width="222">
125   <Button Content="Добавить товар" Click="AddProduct_Click"/>
126   <Button Content="Обновить товар" Click="UpdateProduct_Click"/>
127   <Button Content="Удалить товар" Click="DeleteProduct_Click"/>
128 </StackPanel>
129 </StackPanel>
130 </Border>
131
132 <!-- Панель заказов -->
133 <Border Grid.Column="2" Margin="10" Background="#FFF" CornerRadius="10" Padding="10">
134   <StackPanel>
135     <TextBlock Text="Заказы" FontSize="20" FontWeight="Bold" Margin="0,0,0,10"/>
136
137   <!-- Список заказов -->
138   <ListBox x:Name="OrdersList" Height="150" BorderBrush="#FFCCCC" SelectionChanged="OrdersList_SelectionChanged">
139     <ListBox.ItemTemplate>
140       <DataTemplate>
141         <StackPanel>
142           <TextBlock Text="{Binding CustomerName}" FontWeight="Bold"/>
143           <TextBlock Text="{Binding OrderDate, StringFormat={} {0:dd.MM.yyyy HH:mm}}"/>
144           <TextBlock Text="{Binding TotalPrice, StringFormat={\{0:C\}}}" />
145           <!-- Список товаров в заказе -->
146           <ItemsControl ItemsSource="{Binding Items}">
147             <ItemsControl.ItemTemplate>
148               <DataTemplate>
149                 <StackPanel Orientation="Horizontal">
150                   <TextBlock Text="{Binding Product.Name}" Margin="0,0,10,0"/>
151                   <TextBlock Text="{Binding Product.Price, StringFormat=\{\{0:C\}\}}"/>
152                   <TextBlock Text="x " FontStyle="Italic"/>
153                   <TextBlock Text="{Binding Quantity}"/>
154                 </StackPanel>
155               </DataTemplate>
156             </ItemsControl.ItemTemplate>
157           </ItemsControl>
158         </StackPanel>
159       </DataTemplate>
160     </ListBox.ItemTemplate>
161   </ListBox>
162
163   <!-- Список выбранных товаров для нового заказа -->
164   <TextBlock Text="Выбранные товары для заказа" FontSize="14" FontWeight="Bold" Margin="0,10,0,5"/>
165   <ListBox x:Name="SelectedProductsList" Height="100" BorderBrush="#FFCCCC">
166     <ListBox.ItemTemplate>
167       <DataTemplate>
168         <StackPanel Orientation="Horizontal">
169           <TextBlock Text="{Binding Product.Name}" FontWeight="Bold" Margin="0,0,10,0"/>
170           <TextBlock Text="{Binding Product.Price, StringFormat=\{\{0:C\}\}}"/>
171           <TextBlock Text="x " FontStyle="Italic"/>
172           <TextBlock Text="{Binding Quantity}" Margin="0,0,10,0"/>
173           <!-- Кнопки изменения количества -->
174           <Button Content="+" Style="{StaticResource QuantityButtonStyle}" Click="IncreaseQuantity_Click" Tag="{Binding}"/>
175           <Button Content="-" Style="{StaticResource QuantityButtonStyle}" Click="DecreaseQuantity_Click" Tag="{Binding}"/>
176         </StackPanel>
177       </DataTemplate>
178     </ListBox.ItemTemplate>
179   </ListBox>
180
181   <!-- Кнопка удаления товара из заказа -->
182   <Button Content="Удалить товар из заказа" Click="RemoveProductFromOrder_Click" HorizontalAlignment="Right"/>
183
184   <!-- Поле ввода имени клиента -->
185   <Grid>
186     <TextBox x:Name="CustomerName"/>
187     <TextBlock Text="Имя клиента" Style="{StaticResource PlaceholderStyle}" />
188     <Visibility="{Binding ElementName=CustomerName, Path=Text.IsEmpty, Converter={StaticResource BooleanToVisibilityConverter}}"/>
189   </Grid>
190
191   <!-- Кнопки управления заказами -->
192   <StackPanel Orientation="Horizontal" HorizontalAlignment="Right">
193     <Button Content="Создать заказ" Click="AddOrder_Click"/>
194     <Button Content="Обновить заказ" Click="UpdateOrder_Click"/>
195     <Button Content="Удалить заказ" Click="DeleteOrder_Click"/>
196   </StackPanel>
197 </StackPanel>
198 </Border>
199 </Grid>
200 </Window>

```

Рисунок 3.3 Пример разметки главного окна (Часть 3)

Лабораторная работа №2



Шаг 4. Реализация логики работы окна приложения

Далее необходимо реализовать логику окна данного приложения.

Файл MainWindow.xaml.cs — это code-behind для главного окна приложения (MainWindow.xaml). Он содержит логику взаимодействия пользовательского интерфейса (UI) с данными, обрабатывает события (например, нажатия кнопок) и управляет состоянием приложения. Приложение представляет собой систему управления магазином, где пользователь может:

- 1. Объявления класса и полей — определение данных, используемых в приложении.
- 2. Конструктора — инициализация окна и начальных данных.
- 3. Методов обновления UI — синхронизация списков товаров и заказов с UI.
- 4. Обработчиков событий — реакция на действия пользователя (нажатия кнопок, выбор элементов).
- 5. Вспомогательных методов — очистка полей ввода.

На рисунке 4.1 представлен пример кода для класса MainWindow.

```

1  using StoreManager.Models;
2  using System.Windows;
3  using System.Windows.Controls;
4  using System.Windows.Input;
5  using System.Windows.Media;
6
7  namespace StoreManager
8  {
9      // Главное окно приложения для управления товарами и заказами
10     public partial class MainWindow : Window
11     {
12         // Список всех товаров
13         private List<Product> products = new List<Product>();
14         // Список всех заказов
15         private List<Order> orders = new List<Order>();
16         // Следующий ID для нового товара
17         private int nextProductId = 1;
18         // Следующий ID для нового заказа
19         private int nextOrderId = 1;
20         // Список выбранных товаров для текущего заказа
21         private List<OrderItem> selectedProductsForOrder = new List<OrderItem>();
22     }

```

Рисунок 4.1 пример кода для класса Main

Лабораторная работа №2



Шаг 4. Реализация логики работы окна приложения

Класс MainWindow:

Наследуется от Window, так как это главное окно WPF-приложения.

Ключевое слово partial указывает, что класс разделён между MainWindow.xaml.cs (логика) и MainWindow.xaml (разметка, сгенерированная часть).

Поля:

- products: Хранит список всех товаров (экземпляры класса Product). Это основная коллекция для управления ассортиментом магазина.
- orders: Хранит список заказов (экземпляры класса Order). Каждый заказ содержит имя клиента, дату и список товаров.
- nextProductId: Счётчик для генерации уникальных идентификаторов товаров. Начинается с 1 и увеличивается при добавлении нового товара.

- nextOrderId: Аналогичный счётчик для заказов.
- selectedProductsForOrder: Временное хранилище товаров (OrderItem), которые пользователь выбрал при формировании нового заказа.

Эти поля представляют состояние приложения. Они хранят данные в памяти (вместо базы данных) и используются для отображения в UI и обработки пользовательских действий.

Использование List<T> означает, что данные не обновляют UI автоматически (в отличие от ObservableCollection<T>), поэтому код вручную обновляет списки.

Далее необходимо объявить основной метод MainWindow. Пример кода для данного метода представлен на рисунке 4.2.

Лабораторная работа №2



Шаг 4. Реализация логики работы окна приложения

```

24    public MainWindow()
25    {
26        InitializeComponent(); // Инициализация UI
27        UpdateProductList(); // Обновление списка товаров
28        UpdateOrderList(); // Обновление списка заказов
29        UpdateSelectedProductsList(); // Обновление списка выбранных товаров
30    }

```

Рисунок 4.2 пример кода для класса Main

- InitializeComponent() - метод, сгенерированный WPF, загружает разметку из MainWindow.xaml, инициализирует элементы управления (кнопки, списки, текстовые поля) и устанавливает привязки. Без данного метода UI не будет отображаться.

Вызовы методов обновления:

- UpdateProductList() — устанавливает ItemsSource для ListBox с товарами (ProductsList), чтобы отобразить начальный список (пустой на старте).
- UpdateOrderList() — аналогично предыдущему методу, только для списка заказов (OrdersList).
- UpdateSelectedProductsList() — обновляет список выбранных товаров для заказа (SelectedProductsList).

Назначение:

- Конструктор подготовливает приложение к работе, загружая UI и синхронизируя данные с элементами управления.
- Поскольку списки изначально пусты, вызовы обновления предотвращают ошибки отображения.

Лабораторная работа №2



Шаг 4. Реализация логики работы окна приложения

Методы для обновления списков товаров, заказов и выбранных товаров для заказа представлены на рисунке 4.3.

```
32      // Обновляет отображение списка товаров
33      Ссылка 10
34      private void UpdateProductList()
35      {
36          ProductsList.ItemsSource = null;
37          ProductsList.ItemsSource = products;
38      }
39
40      // Обновляет отображение списка заказов
41      Ссылка 4
42      private void UpdateOrderList()
43      {
44          OrdersList.ItemsSource = null;
45          OrdersList.ItemsSource = orders;
46      }
47      // Обновляет отображение списка выбранных товаров
48      Ссылка 8
49      private void UpdateSelectedProductsList()
50      {
51          SelectedProductsList.ItemsSource = null;
52          SelectedProductsList.ItemsSource = selectedProductsForOrder;
53      }
```

Рисунок 4.3 Методы обновления данных

Метод UpdateProductList Сбрасывает ItemsSource в null, чтобы очистить привязку. Устанавливает ItemsSource в products, чтобы ListBox (ProductsList) отобразил текущий список товаров. Остальные методы работают аналогично. Null используется для предотвращения проблемы с кэшированием данных в WPF, обеспечивая корректное обновление интерфейса.

Лабораторная работа №2



Шаг 4. Реализация логики работы окна приложения

Далее рассмотрим методы для работы с конкретными товарами. Эти методы реагируют на действия пользователя с товарами (кнопки «Добавить», «Обновить», «Удалить», выбор в списке). Метод добавления нового товара представлен на рисунке 4.4.

```

51
52
53
54 // Обработчик нажатия кнопки "Добавить товар"
Ссылка 1
55 private void AddProduct_Click(object sender, RoutedEventArgs e)
{
56     // Проверка корректности введённых данных
57     if (string.IsNullOrWhiteSpace(ProductName.Text) ||
58         !decimal.TryParse(ProductPrice.Text, out decimal price) ||
59         !int.TryParse(ProductStock.Text, out int stock) ||
60         stock < 0)
61     {
62         MessageBox.Show("Пожалуйста, введите корректные данные о товаре. Остаток не может быть отрицательным.",
63                     "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
64         return;
65     }
66
67     // Создание нового товара
68     var product = new Product
69     {
70         Id = nextProductId++,
71         Name = ProductName.Text,
72         Price = price,
73         Stock = stock
74     };
75
76     products.Add(product); // Добавление товара в список
77     UpdateProductList(); // Обновление UI
78     ClearProductFields(); // Очистка полей ввода
79 }
80
81

```

Рисунок 4.4 Метод добавления нового товара

Данный метод проверяет валидность ввода:

- productName.Text не пустое;
- проверка productPrice.Text (что цена корректна (decimal), количество — неотрицательное число);
- productStock.Text преобразуется в int.

Если валидация не пройдена, показывает сообщение об ошибке. Создаёт новый объект Product с уникальным Id (из nextProductId), именем, ценой и запасом из полей ввода. Далее добавляет товар в список products, обновляет UI (UpdateProductList) и очищает поля ввода (ClearProductFields).

Лабораторная работа №2



Шаг 4. Реализация логики работы окна приложения

Следующий метод позволяет обновить (редактировать) информацию о конкретном товаре. Реализация метода обновления представлена на рисунке 4.5.

```

81 // Обработчик нажатия кнопки "Обновить товар"
82 // Ссылка: 1
83 private void UpdateProduct_Click(object sender, RoutedEventArgs e)
84 {
85     if (ProductsList.SelectedItem is Product selectedProduct)
86     {
87         // Проверка корректности введённых данных
88         if (String.IsNullOrWhiteSpace(ProductName.Text) ||
89             !decimal.TryParse(ProductPrice.Text, out decimal price) ||
90             !int.TryParse(ProductStock.Text, out int stock) ||
91             stock < 0)
92         {
93             MessageBox.Show("Пожалуйста, введите корректные данные о товаре. Остаток не может быть отрицательным.",
94                         "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
95             return;
96         }
97
98         // Обновление свойств выбранного товара
99         selectedProduct.Name = ProductName.Text;
100        selectedProduct.Price = price;
101        selectedProduct.Stock = stock;
102
103        UpdateProductList(); // Обновление UI
104        ClearProductFields(); // Очистка полей ввода
105    }
106    else
107    {
108        MessageBox.Show("Пожалуйста, выберите товар для обновления.", "Ошибка",
109                         MessageBoxButton.OK, MessageBoxImage.Warning);
110    }
111 }

```

Рисунок 4.5 Метод обновления нового товара

Данный метод проверяет, выбран ли товар в списке ProductsList.

- проводит валидацию ввода (аналогично AddProduct_Click);
- если товар выбран и данные валидны, обновляет свойства выбранного объекта Product (Name, Price, Stock);
- обновляет UI и очищает поля;
- если товар не выбран, показывает предупреждение.

Лабораторная работа №2



Шаг 4. Реализация логики работы окна приложения

Для того, чтобы исключить конкретный товар из общего списка, используется метод DeleteProduct (Рисунок 4.6).

```

111 // Обработчик нажатия кнопки "Удалить товар"
112 // Ссылка 1
113 private void DeleteProduct_Click(object sender, RoutedEventArgs e)
114 {
115     if (ProductsList.SelectedItem is Product selectedProduct)
116     {
117         // Проверка, используется ли товар в заказах
118         if (orders.Any(order => order.Items.Any(item => item.Product.Id == selectedProduct.Id)))
119         {
120             MessageBox.Show("Нельзя удалить товар, используемый в заказах.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);
121             return;
122         }
123
124         // Подтверждение удаления
125         var result = MessageBox.Show($"Вы уверены, что хотите удалить товар '{selectedProduct.Name}'?", "Подтверждение удаления", MessageBoxButton.YesNo, MessageBoxImage.Question);
126
127         if (result == MessageBoxResult.Yes)
128         {
129             selectedProductsForOrder.RemoveAll(item => item.Product.Id == selectedProduct.Id); // Удаление из текущего заказа
130             products.Remove(selectedProduct); // Удаление из списка товаров
131             UpdateProductList();
132             UpdateSelectedProductsList();
133             ClearProductFields();
134         }
135     }
136     else
137     {
138         MessageBox.Show("Пожалуйста, выберите товар для удаления.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);
139     }
140 }
141

```

Рисунок 4.6 Метод удаления товара

Данный метод выполняет следующие функции:

- удаляет товар из списка, если он не используется в заказах;
- если товар выбран и не связан с заказами, запрашивает подтверждение и удаляет его;
- также удаляет его из текущего заказа (если он там есть);
- если товар используется в заказах — показывает предупреждение;
- обновляет UI и очищает поля.

Лабораторная работа №2



Шаг 4. Реализация логики работы окна приложения

Далее рассмотрим методы ProductsList_SelectionChanged и AddToOrder. Когда пользователь выбирает товар в списке ProductsList, метод ProductsList_SelectionChanged заполняет поля ProductName, ProductPrice, ProductStock данными выбранного товара. Используется также для редактирования информации о товаре (Рисунок 4.7)

```

141 }
142 }
143 }
144 // Обработчик изменения выделения в списке товаров
Ссылка: 1
145 private void ProductsList_SelectionChanged(object sender, SelectionChangedEventArgs e)
146 {
147     if (ProductsList.SelectedItem is Product selectedProduct)
148     {
149         // Заполнение полей ввода данными выбранного товара
150         ProductName.Text = selectedProduct.Name;
151         ProductPrice.Text = selectedProduct.Price.ToString();
152         ProductStock.Text = selectedProduct.Stock.ToString();
153     }
154 }

Ссылка: 1
155 private void AddToOrder_Click(object sender, RoutedEventArgs e)
156 {
157     if (ProductsList.SelectedItem is Product selectedProduct)
158     {
159         if (selectedProduct.Stock <= 0)
160         {
161             MessageBox.Show("Товара больше нет на складе.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);
162             return;
163         }
164
165         var existingItem = selectedProductsForOrder.FirstOrDefault(item => item.Product.Id == selectedProduct.Id);
166
167         if (existingItem != null)
168         {
169             existingItem.Quantity++;
170         }
171         else
172         {
173             selectedProductsForOrder.Add(new OrderItem { Product = selectedProduct, Quantity = 1 });
174         }
175
176         selectedProduct.Stock--; // <== уменьшение запаса
177         UpdateProductList();
178         UpdateSelectedProductsList();
179     }
180     else
181     {
182         MessageBox.Show("Пожалуйста, выберите товар для добавления в заказ.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);
183     }
184 }
```

Рисунок 4.7 Методы редактирования товара и добавления
в список выбранных товаров для заказа

Лабораторная работа №2



Шаг 4. Реализация логики работы окна приложения

Метод AddToOrder добавляет выбранный товар из списка в текущий заказ.

Если товар уже добавлен — увеличивает его количество.

Если нет — создаёт новую позицию.

Также уменьшает остаток товара на складе.

Обновляет UI.

Если товар не выбран или нет в наличии — показывает предупреждение.

Далее рассмотрим метод создания нового заказа. Пример кода для данного метода представлен на рисунке 4.8.

```

185
186
187 // Обработчик нажатия кнопки "Создать заказ"
188 Ссылка 1
189 private void AddOrder_Click(object sender, RoutedEventArgs e)
190 {
191     // Проверка имени клиента
192     if (string.IsNullOrWhiteSpace(CustomerName.Text))
193     {
194         MessageBox.Show("Пожалуйста, введите имя клиента.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
195         return;
196     }
197
198     // Проверка наличия товаров в заказе
199     if (!selectedProductsForOrder.Any())
200     {
201         MessageBox.Show("Пожалуйста, выберите хотя бы один товар для заказа.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);
202         return;
203     }
204
205     // Создание нового заказа
206     var order = new Order
207     {
208         Id = nextOrderId++,
209         CustomerName = CustomerName.Text,
210         OrderDate = DateTime.Now,
211         Items = new List<OrderItem>(selectedProductsForOrder)
212     };
213
214     orders.Add(order); // Добавление заказа в список
215     UpdateProductList();
216     UpdateOrderList();
217     ClearOrderFields(); // Очистка полей заказа
218 }
```

Рисунок 4.8 Метод добавления товара в список выбранных товаров для заказа

Лабораторная работа №2



Шаг 4. Реализация логики работы окна приложения

Данный метод выполняет следующие функции:

- создаёт новый заказ;
- проверяет, введено ли имя клиента и есть ли хотя бы один товар в заказе;
- если всё в порядке — формирует объект Order, копирует список товаров и добавляет заказ в список orders;
- обновляет UI и очищает поля;
- если что-то не заполнено — показывает ошибку.

Следующий метод позволяет обновить информацию о заказе. Пример кода для данного метода представлен на рисунке 4.9.

```

218 // Обработчик нажатия кнопки "Обновить заказ"
219 private void UpdateOrder_Click(object sender, RoutedEventArgs e)
220 {
221     if (OrdersList.SelectedItem is Order selectedOrder)
222     {
223         // Проверка имени клиента
224         if (string.IsNullOrWhiteSpace(CustomerName.Text))
225         {
226             MessageBox.Show("Пожалуйста, введите имя клиента.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
227             return;
228         }
229
230         // Проверка наличия товаров
231         if (!selectedProductsForOrder.Any())
232         {
233             MessageBox.Show("Пожалуйста, выберите хотя бы один товар для заказа.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);
234             return;
235         }
236
237         // Обновление заказа
238         selectedOrder.CustomerName = CustomerName.Text;
239         selectedOrder.OrderDate = DateTime.Now;
240         selectedOrder.Items.Clear();
241         selectedOrder.Items.AddRange(selectedProductsForOrder);
242
243         // Проверка доступности перед применением изменений
244         foreach (var item in selectedProductsForOrder)
245         {
246             int availableStock = item.Product.Stock +
247                 selectedOrder.Items.Where(i => i.Product.Id == item.Product.Id).Sum(i => i.Quantity); // учитываем старый заказ
248
249             if (item.Quantity > availableStock)
250             {
251                 MessageBox.Show($"Недостаточно товара '{item.Product.Name}' на складе для обновления заказа.",
252                             "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
253
254             }
255         }
256
257         UpdateProductList();
258         UpdateOrderList();
259         ClearOrderFields();
260     }
261     else
262     {
263         MessageBox.Show("Пожалуйста, выберите заказ для обновления.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);
264     }
265 }

```

Рисунок 4.9 Метод обновления заказа

Лабораторная работа №2



Шаг 4. Реализация логики работы окна приложения

- обновляет выбранный заказ;
- проверяет имя клиента и наличие товаров в заказе.
- если данные валидны — обновляет имя клиента, дату и товары в заказе.
- также проверяет, есть ли достаточный остаток на складе для всех товаров.
- обновляет UI и очищает поля.
- если что-то не так — показывает сообщение об ошибке.

Следующий метод позволяет удалить выбранный заказ. Пример кода для данного метода представлен на рисунке 4.10.

```

266
267
268    }
269
270    // Обработчик нажатия кнопки "Удалить заказ"
271    Ссылка: private void DeleteOrder_Click(object sender, RoutedEventArgs e)
272    {
273        if (OrdersList.SelectedItem is Order selectedOrder)
274        {
275            // Подтверждение удаления
276            var result = MessageBox.Show($"Вы уверены, что хотите удалить заказ для клиента '{selectedOrder.CustomerName}'?", "Подтверждение удаления", MessageBoxButton.YesNo, MessageBoxImage.Question);
277
278            if (result == MessageBoxResult.Yes)
279            {
280                // Возврат запасов
281                foreach (var item in selectedOrder.Items)
282                {
283                    item.Product.Stock += item.Quantity;
284                }
285
286                orders.Remove(selectedOrder); // Удаление заказа
287                UpdateProductList();
288                UpdateOrderList();
289                ClearOrderFields();
290            }
291        }
292        else
293        {
294            MessageBox.Show("Пожалуйста, выберите заказ для удаления.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);
295        }
296    }

```

Рисунок 4.10 Метод удаления заказа

Лабораторная работа №2



Шаг 4. Реализация логики работы окна приложения

Следующий метод нужен для удаления товара из заказа. Код для данного метода представлен на рисунке 4.11.

```
297 // Обработчик нажатия кнопки "Удалить товар из заказа"
298 Ссылка:1
299 private void RemoveProductFromOrder_Click(object sender, RoutedEventArgs e)
300 {
301     if (SelectedProductsList.SelectedItem is OrderItem selectedItem)
302     {
303         if (selectedItem.Quantity > 1)
304         {
305             selectedItem.Quantity--; // Уменьшение количества
306         }
307         else
308         {
309             selectedProductsForOrder.Remove(selectedItem); // Удаление товара
310         }
311         UpdateSelectedProductsList();
312     }
313     else
314     {
315         MessageBox.Show("Пожалуйста, выберите товар для удаления из заказа.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);
316     }
317 }
318 // Обработка нажатия клавиши Enter в поле ввода количества
```

Рисунок 4.11 Метод удаления товара из заказа

- метод RemoveProductFromOrder удаляет товар из текущего заказа;
- если количество больше 1 — уменьшает его;
- если 1 — полностью удаляет товар из списка;
- обновляет UI;
- если товар не выбран — показывает предупреждение.

Лабораторная работа №2



Шаг 4. Реализация логики работы окна приложения

Далее рассмотрим методы, отвечающие за увеличение и уменьшение количества товара в заказе (кнопки «+» и «-»). Пример листинга данных методов представлен на рисунке 4.12.

```

318 // Обработчик нажатия кнопки "+" для увеличения количества
319 // Ссылка: 1
320 private void IncreaseQuantity_Click(object sender, RoutedEventArgs e)
321 {
322     if (sender is Button button && button.Tag is OrderItem item)
323     {
324         if (item.Product.Stock < 1)
325         {
326             MessageBox.Show("Товара больше нет на складе.", "Ошибка", MessageBoxButton.OK, MessageBoxImage.Warning);
327             return;
328         }
329
330         item.Quantity++;
331         item.Product.Stock--; // резерв
332         UpdateProductList();
333         UpdateSelectedProductsList();
334     }
335 }
336
337 // Обработчик нажатия кнопки "-" для уменьшения количества
338 // Ссылка: 1
339 private void DecreaseQuantity_Click(object sender, RoutedEventArgs e)
340 {
341     if (sender is Button button && button.Tag is OrderItem item)
342     {
343         if (item.Quantity > 1)
344         {
345             item.Quantity--; // Уменьшение количества
346         }
347         else
348         {
349             selectedProductsForOrder.Remove(item); // Удаление товара
350         }
351         item.Product.Stock++;
352         UpdateProductList();
353         UpdateSelectedProductsList();
354     }
355 }
356 // Обработка нажатия клавиши Esc

```

Рисунок 4.12 Методы управления количеством товаров в заказе

Лабораторная работа №2



Шаг 4. Реализация логики работы окна приложения

Данный метод позволяет выделить определенный заказ и изменить имя заказчика. Реализация метода представлена на рисунке 4.13.

```

354
355
356    // Обработчик изменения выделения в списке заказов
Ссылок: 1
357    private void OrdersList_SelectionChanged(object sender, SelectionChangedEventArgs e)
358    {
359        if (OrdersList.SelectedItem is Order selectedOrder)
360        {
361            // Заполнение полей данными выбранного заказа
362            CustomerName.Text = selectedOrder.CustomerName;
363            selectedProductsForOrder.Clear();
364            selectedProductsForOrder.AddRange(selectedOrder.Items);
365            UpdateSelectedProductsList();
366        }
367    }

```

Рисунок 4.13 Метод обновления нового товара

- срабатывает при выборе заказа из списка;
- заполняет поле имени клиента;
- загружает список товаров из заказа в текущий список selectedProductsForOrder;
- обновляет UI.

Далее рассмотрим методы очищения полей ClearProductFields и ClearOrderFields. Они позволяют очистить текстовые поля для добавления/редактирования товара и снимают выделение в списке товаров. ClearOrderFields очищает поле имени клиента и список выбранных товаров, снимает выделение в списке заказов, при этом не трогает список товаров (оставляет выделение). Листинг данных методов приведен на рисунке 4.14.

Лабораторная работа №2



Шаг 4. Реализация логики работы окна приложения

```

368
369     // Очистка полей ввода для товаров
370     Ссылка 3
371     private void ClearProductFields()
372     {
373         ProductName.Text = "";
374         ProductPrice.Text = "";
375         ProductStock.Text = "";
376         ProductsList.SelectedItem = null; // Сброс выделения
377     }
378
379     // Очистка полей ввода для заказов
380     Ссылка 3
381     private void ClearOrderFields()
382     {
383         CustomerName.Text = "";
384         selectedProductsForOrder.Clear();
385         UpdateSelectedProductsList();
386         OrdersList.SelectedItem = null; // Сброс выделения
387         // Выделение в ProductsList не сбрасывается
388     }
389
390     // Обработчик клика мышью по списку товаров
391     Ссылка 1
392     private void ProductsList_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)
393     {
394         var listBox = sender as ListBox;
395         var hitTestResult = VisualTreeHelper.HitTest(listBox, e.GetPosition(listBox));
396         if (hitTestResult != null)
397         {
398             // Проверка, попал ли клик на ListViewItem
399             var element = hitTestResult.VisualHit;
400             while (element != null && !(element is ListViewItem))
401             {
402                 element = VisualTreeHelper.GetParent(element);
403             }
404             if (element == null)
405             {
406                 listBox.SelectedItem = null; // Сброс выделения при клике по пустому месту
407             }
408         }
409     }

```

Рисунок 4.14 Методы очистки полей и сброса выделения товара

Метод ProductsList_MouseLeftButtonDown используется, если пользователь кликнул мышью по пустому месту в списке товаров. В этом случае метод сбрасывает выделение. Применяется для удобства работы — чтобы можно было "отменить" выбор.

Лабораторная работа №2



Шаг 5. Внешний вид приложения

В итоговом варианте внешний вид приложения представлен на Рисунке 5.1.

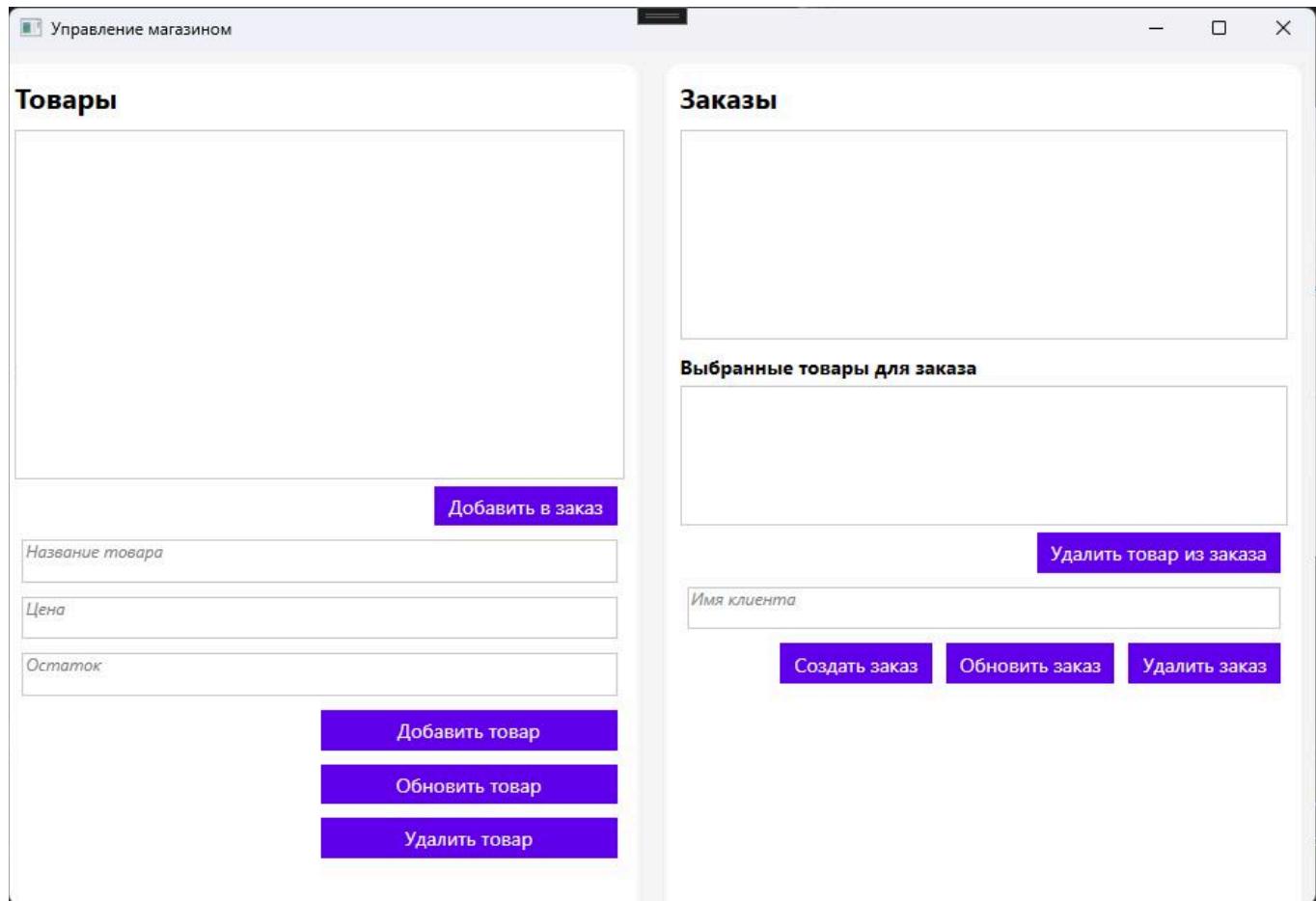


Рисунок 5.1 Интерфейс приложения

Лабораторная работа №2



Шаг 6. Демонстрация работы программы

В качестве примера отображения информации в приложении, введены данные о товарах и заказе. Пример интерфейса приложения с введенными данными приведен на Рисунке 6.1.

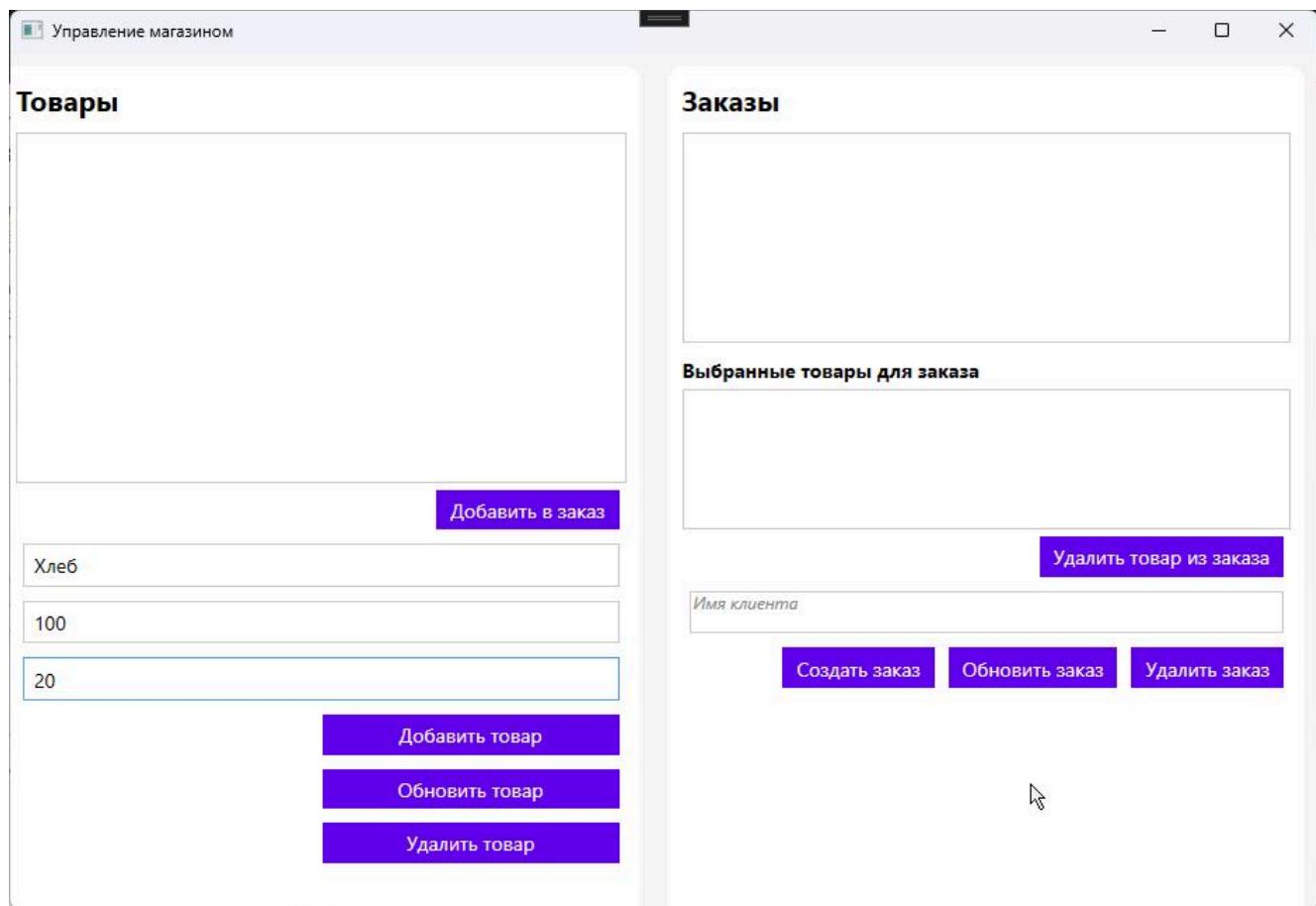


Рисунок 6.1 Пример заполнения информации в приложении

Лабораторная работа №2



Шаг 6. Демонстрация работы программы

Далее по нажатию кнопки “Добавить товар”, товар добавляется в общий список. Таким образом можем добавить группу товаров (Рисунок 6.2).

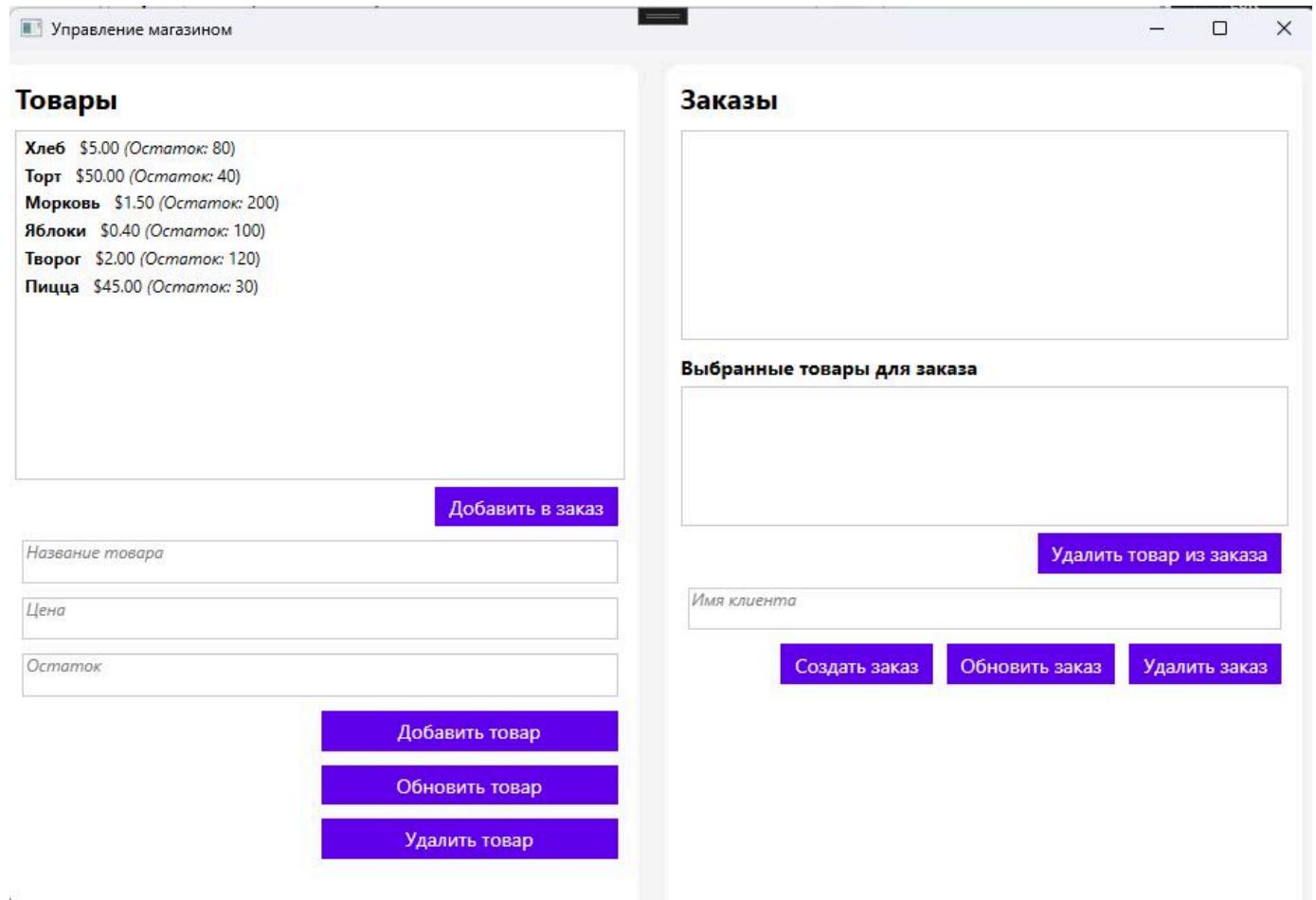


Рисунок 6.2 Пример добавления группы товаров

Лабораторная работа №2



Шаг 6. Демонстрация работы программы

Далее необходимо выбрать конкретный товар из списка, по нажатию кнопки добавить его в список выбранных товаров. В данном списке помошью кнопок «+» и «-» можно отредактировать количество товаров для добавления в заказ (Рисунок 6.3).

Товары

Хлеб	\$5.00	(Остаток: 79)
Торт	\$50.00	(Остаток: 39)
Морковь	\$1.50	(Остаток: 200)
Яблоки	\$0.40	(Остаток: 99)
Творог	\$2.00	(Остаток: 119)
Пицца	\$45.00	(Остаток: 30)

Добавить в заказ

Яблоки
0,4
100

Заказы

Выбранные товары для заказа

Хлеб	\$5.00 x 1	+	-
Торт	\$50.00 x 1	+	-
Творог	\$2.00 x 1	+	-

Удалить товар из заказа

Имя клиента

Создать заказ **Обновить заказ** **Удалить заказ**

Добавить товар
Обновить товар
Удалить товар

Рисунок 6.3 Пример добавления группы товаров

Лабораторная работа №2



Шаг 6. Демонстрация работы программы

Далее необходимо ввести имя клиента (название заказа) и нажать кнопку «Создать заказ» для добавления нового заказа в список заказов (Рисунок 6.4).

Товары

Хлеб	\$5.00	(Остаток: 79)
Торт	\$50.00	(Остаток: 38)
Морковь	\$1.50	(Остаток: 200)
Яблоки	\$0.40	(Остаток: 88)
Творог	\$2.00	(Остаток: 114)
Пицца	\$45.00	(Остаток: 30)

Добавить в заказ

Яблоки
0,4
100

Добавить товар
Обновить товар
Удалить товар

Заказы

Иван	16.04.2025 15:15
Итого: \$121.80	
Хлеб	\$5.00 x 1
Торт	\$50.00 x 2
Творог	\$2.00 x 6
Яблоки	\$0.40 x 12

Выбранные товары для заказа

Удалить товар из заказа

Имя клиента

Создать заказ **Обновить заказ** **Удалить заказ**

Рисунок 6.4 Пример добавления группы товаров

Лабораторная работа №2



Шаг 6. Демонстрация работы программы

При нажатии на конкретный заказ можно вывести список товаров (Рисунок 6.5).

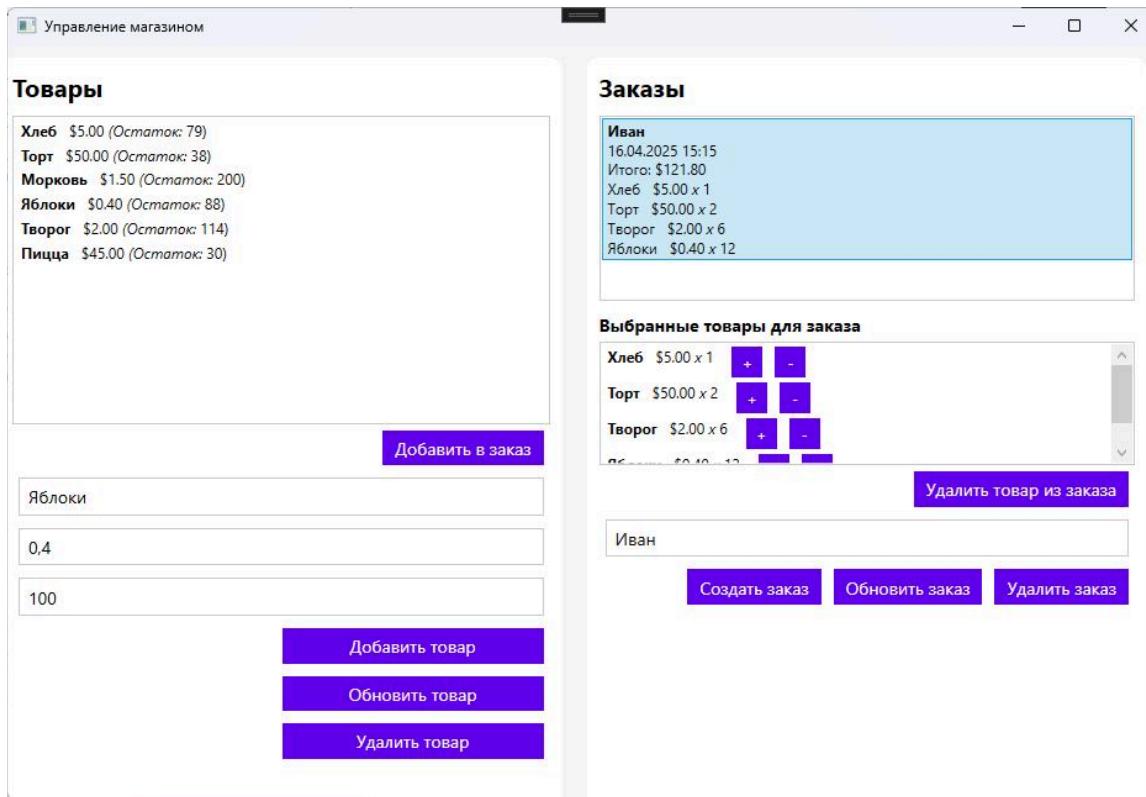


Рисунок 6.5 Вывод списка товаров в заказе

Лабораторная работа №2



Шаг 6. Демонстрация работы программы

При нажатии кнопки «Удалить товар из заказа» можно удалить конкретный товар добавленный в заказ (Рисунок 6.6).

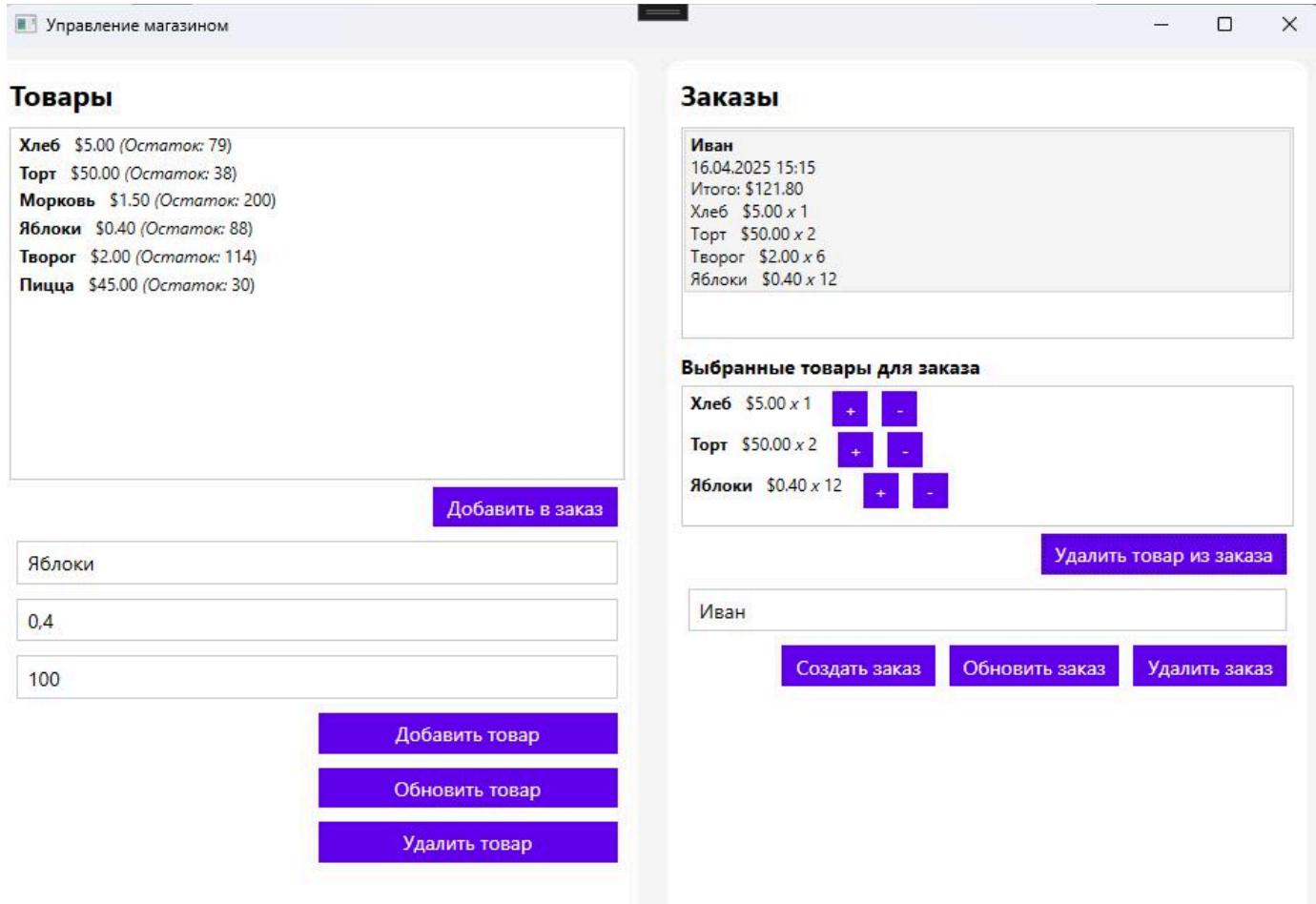


Рисунок 6.6 Удаление товара из заказа

Лабораторная работа №2



Шаг 6. Демонстрация работы программы

Кнопка «Обновить заказ» позволяет обновить информацию о заказе удалить конкретный товар добавленный в заказ (Рисунок 6.7).

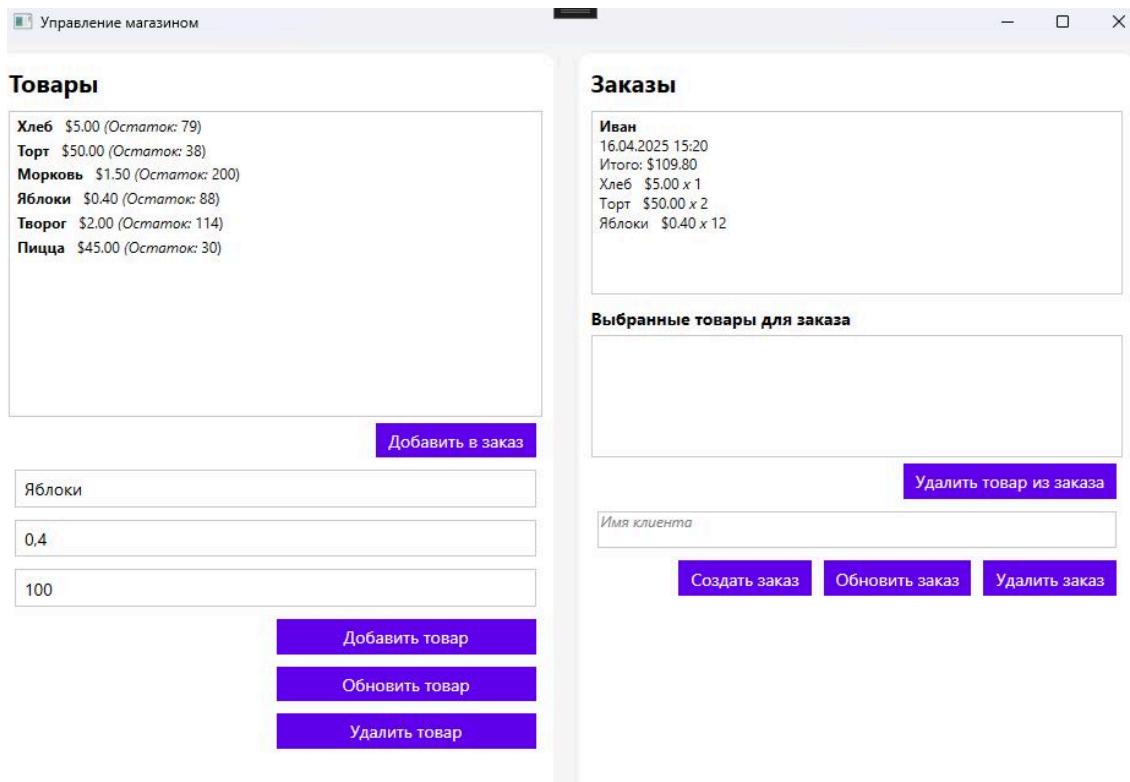


Рисунок 6.7 Обновление списка товаров в заказе

Лабораторная работа №2



Шаг 6. Демонстрация работы программы

При нажатии кнопки «Удалить заказ» можно удалить конкретный заказ из списка (Рисунок 6.8). При удалении на экран выводится сообщение с подтверждением. После удаления количество остатков товара увеличивается в соответствии с тем, сколько товара было в заказе.

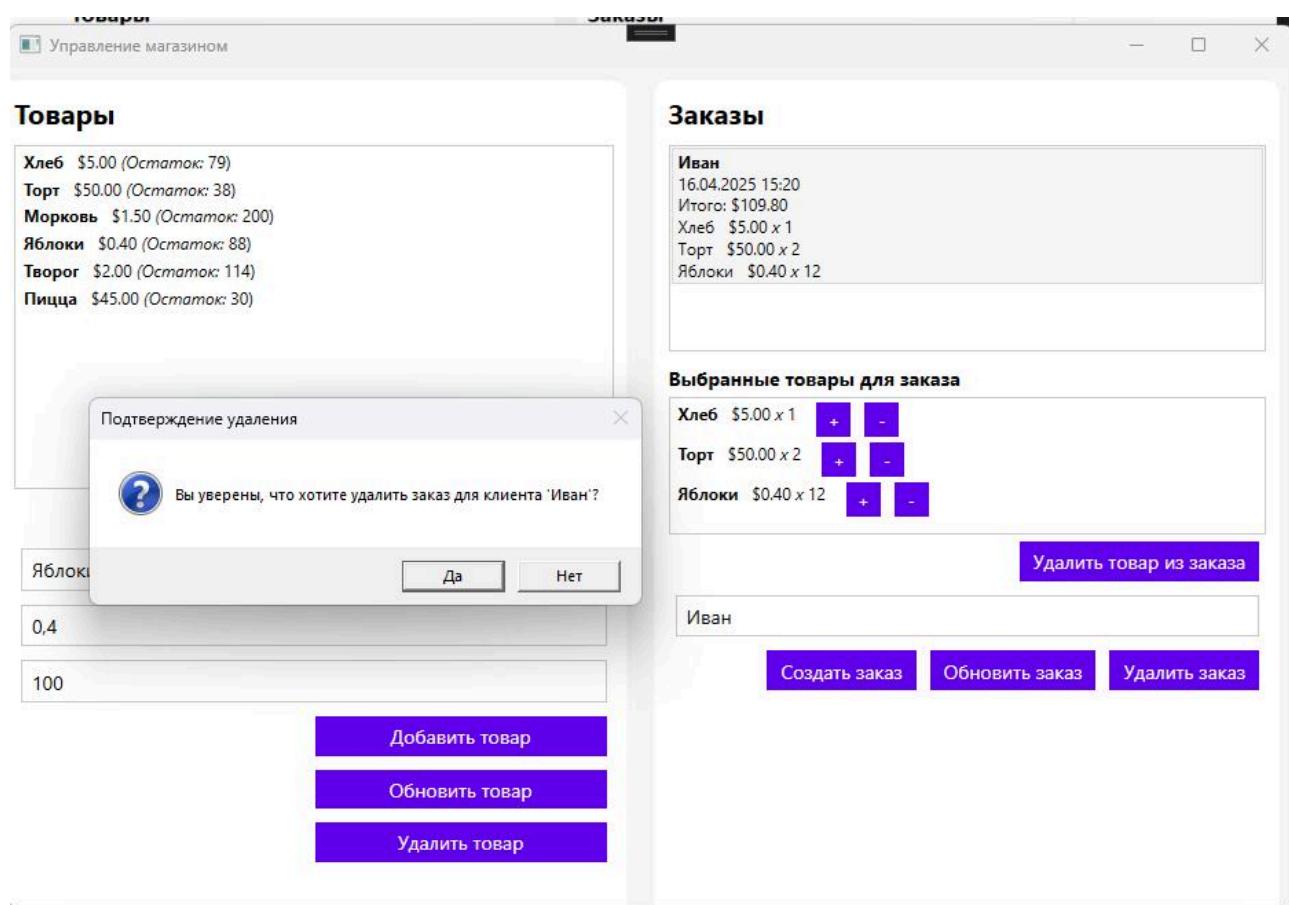


Рисунок 6.8 Удаление товара из заказа

Лабораторная работа №2



Шаг 6. Демонстрация работы программы

При нажатии кнопки «Удалить товар из заказа» можно удалить конкретный товар добавленный в заказ (Рисунок 6.9).

The screenshot shows a Windows application window titled "Управление магазином". It has two main panels:

- Товары (Products) Panel:** Contains a list of items with their prices and remaining stock:
 - Хлеб \$5.00 (Остаток: 80)
 - Торт \$50.00 (Остаток: 40)
 - Морковь \$1.50 (Остаток: 200)
 - Яблоки \$0.40 (Остаток: 100)
 - Творог \$2.00 (Остаток: 120)
 - Пицца \$45.00 (Остаток: 30)A purple button labeled "Добавить в заказ" (Add to Order) is located at the bottom of this panel.
- Заказы (Orders) Panel:** Contains a section titled "Выбранные товары для заказа" (Selected products for order) which is currently empty. Below it are input fields for "Имя клиента" (Client Name) and three buttons: "Создать заказ" (Create Order), "Обновить заказ" (Update Order), and a purple button labeled "Удалить товар из заказа" (Delete item from order).

Рисунок 6.9 Вывод списка товаров в заказе