

# Лабораторная работа №3

## Содержание

1	КАК ЭТО ЧИТАТЬ? 📖	2
2	ЗАДАНИЕ 🖨	3
2.1	ВАРИАНТЫ ЗАДАНИЙ ☰	4
2.2	ДОПОЛНИТЕЛЬНОЕ ЗАДАНИЕ 💡	8
3	ШАГИ ПО СОЗДАНИЮ ПРИЛОЖЕНИЯ 🛠	9
3.1	ШАГ 1: СОЗДАНИЕ ПРОЕКТА	9
3.2	ШАГ 2: ДОБАВЛЕНИЕ ФАЙЛОВ	11
3.2.1	СОЗДАНИЕ ПАПКИ MODELS	11
3.2.2	КЛАССЫ: PRODUCT.CS, ORDER.CS, ORDERITEM.CS	11
3.3	ШАГ 3: РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА	13
3.4	ШАГ 4: РЕАЛИЗАЦИЯ ЛОГИКИ	16
4	ШАГ 5: ДЕМОНСТРАЦИЯ РАБОТЫ 🧑	31
5	ВОПРОСЫ ДЛЯ ЗАЩИТЫ 🛡	41

## Листинги

1	Класс <code>Order.cs</code>	11
2	Класс <code>OrderItem</code>	11
3	Класс <code>Product.cs</code>	12
4	Класс <code>BooleanToVisibilityConverter</code>	12
5	Пример разметки главного окна <code>MainWindow</code>	13
6	Пример кода для класса <code>Main</code>	16
7	Пример кода для класса <code>Main</code>	17
8	Методы обновления данных	18
9	Метод добавления нового товара	19
10	Метод обновления нового товара	20
11	Метод удаления товара	21
12	Методы редактирования товара и добавления в список выбранных товаров для заказа	22
13	Метод добавления товара в список выбранных товаров для заказа	23
14	Метод обновления заказа	24
15	Метод удаления заказа	26
16	Метод удаления товара из заказа	27
17	Методы управления количеством товара в заказе	28
18	Метод обновления нового товара	29
19	Методы очистки полей и сброса выделения товара	30

# Лабораторная работа №3

## 1 КАК ЭТО ЧИТАТЬ?

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Ut purus elit, vestibulum ut, placerat ac, adipiscing vitae, felis. Curabitur dictum gravida mauris. Nam arcu libero, nonummy eget, consectetur id, vulputate a, magna. Donec vehicula augue eu neque. Pellentesque habitant morbi tristique senectus et netus et malesuada fames ac turpis egestas. Mauris ut leo. Cras viverra metus rhoncus sem. Nulla et lectus vestibulum urna fringilla ultrices. Phasellus eu tellus sit amet tortor gravida placerat. Integer sapien est, iaculis in, pretium quis, viverra ac, nunc. Praesent eget sem vel leo ultrices bibendum. Aenean faucibus. Morbi dolor nulla, malesuada eu, pulvinar at, mollis ac, nulla. Curabitur auctor semper nulla. Donec varius orci eget risus. Duis nibh mi, congue eu, accumsan eleifend, sagittis quis, diam. Duis eget orci sit amet orci dignissim rutrum.

Nam dui ligula, fringilla a, euismod sodales, sollicitudin vel, wisi. Morbi auctor lorem non justo. Nam lacus libero, pretium at, lobortis vitae, ultricies et, tellus. Donec aliquet, tortor sed accumsan bibendum, erat ligula aliquet magna, vitae ornare odio metus a mi. Morbi ac orci et nisl hendrerit mollis. Suspendisse ut massa. Cras nec ante. Pellentesque a nulla. Cum sociis natoque penatibus et magnis dis parturient montes, nascetur ridiculus mus. Aliquam tincidunt urna. Nulla ullamcorper vestibulum turpis. Pellentesque cursus luctus mauris.

Nulla malesuada porttitor diam. Donec felis erat, congue non, volutpat at, tincidunt tristique, libero. Vivamus viverra fermentum felis. Donec nonummy pellentesque ante. Phasellus adipiscing semper elit. Proin fermentum massa ac quam. Sed diam turpis, molestie vitae, placerat a, molestie nec, leo. Maecenas lacinia. Nam ipsum ligula, eleifend at, accumsan nec, suscipit a, ipsum. Morbi blandit ligula feugiat magna. Nunc eleifend consequat lorem. Sed lacinia nulla vitae enim. Pellentesque tincidunt purus vel magna. Integer non enim. Praesent euismod nunc eu purus. Donec bibendum quam in tellus. Nullam cursus pulvinar lectus. Donec et mi. Nam vulputate metus eu enim. Vestibulum pellentesque felis eu massa.

Также [дополнительные задания](#) выполняются по желанию и не подлежат проверке на зачёте.

# Лабораторная работа №3

## 2 ЗАДАНИЕ


Система управления заказами для онлайн-магазина:

1. Реализовать CRUD-приложение (WPF)<sup>1</sup>.
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))

Весь код доступен в [репозитории на GitHub](#) <sup>2</sup>

---

<sup>1</sup>CRUD-приложение — это программа, которая позволяет вам выполнять четыре основных действия с данными (например, товарами или заказами): Создавать новые записи, Читать (просматривать) существующие, Обновлять (редактировать) их и Удалять ненужные.

<sup>2</sup>Ссылки, выделенные [голубым](#) цветом, ведут на внешние ресурсы; ссылки, выделенные [зелёным](#) цветом, указывают на исходный код в репозитории GitHub .

# Лабораторная работа №3

## 2.1 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 «Спортивный инвентарь», 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)

### 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».

#### 8. Магазин бытовой техники

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

#### 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)

#### 12. Магазин канцелярии

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

#### 13. Магазин ювелирных изделий

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

## 14. Магазин зоотоваров

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

## 15. Магазин видеоигр

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

## 16. Магазин садовых товаров

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

## 17. Магазин часов

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

## 18. Магазин обуви

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

## 19. Магазин настольных игр

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

## 20. Магазин сувениров

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

## 21. Магазин парфюмерии

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

## 22. Магазин строительных материалов

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

## 23. Магазин для художников

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

## 24. Магазин чая и кофе

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

## 25. Магазин для рыбалки

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

## 26. Магазин для кемпинга

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

## 27. Магазин медицинских товаров

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


### 28. Магазин фильмов


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

### 29. Магазин горнолыжного оборудования

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

## 2.2 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).



## 3 ШАГИ ПО СОЗДАНИЮ ПРИЛОЖЕНИЯ

### 3.1 ШАГ 1: СОЗДАНИЕ ПРОЕКТА

Для создания проекта необходимо выбрать:

1. Создать проект
2. Приложение WPF (Майкрософт) (Рисунок 1)

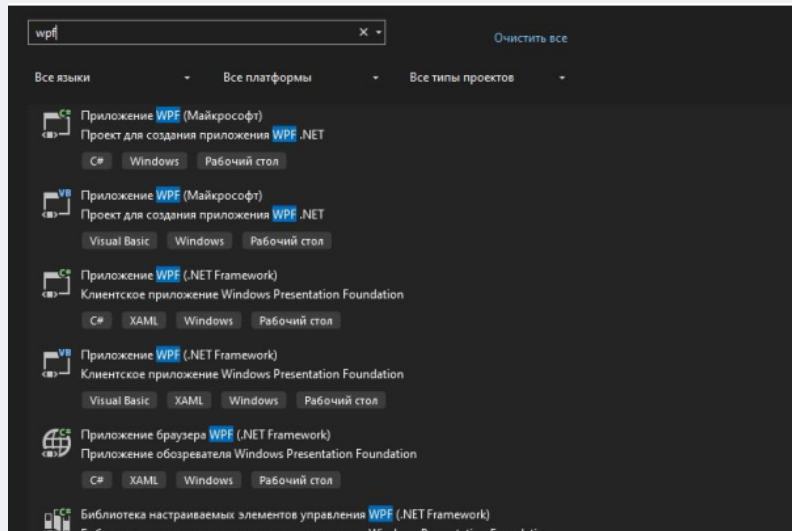


Рис. 1: Создание проекта

3. Далее задать имя проекта: StoreManager (Рисунок 2)

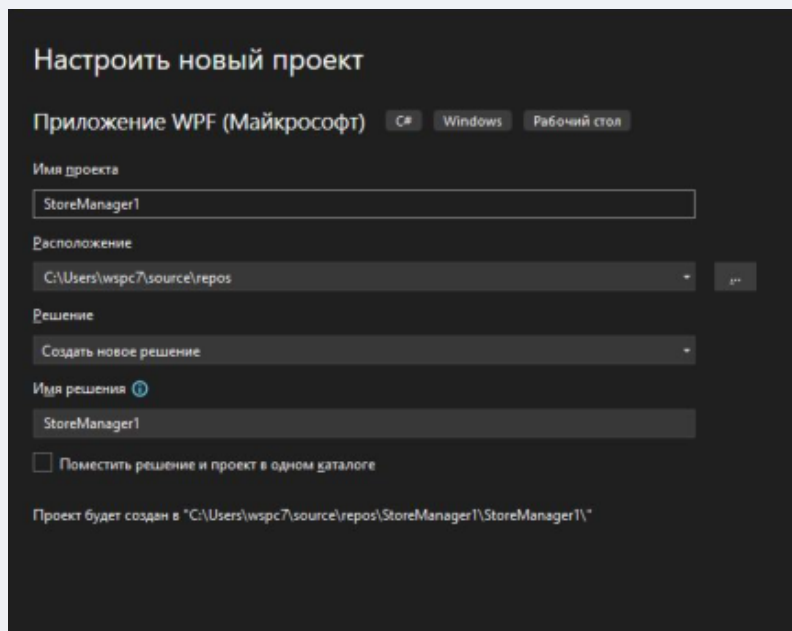


Рис. 2: Задание имени проекта

4. Выбрать платформу (пример выполнен на .net 8.0) (Рисунок 3)

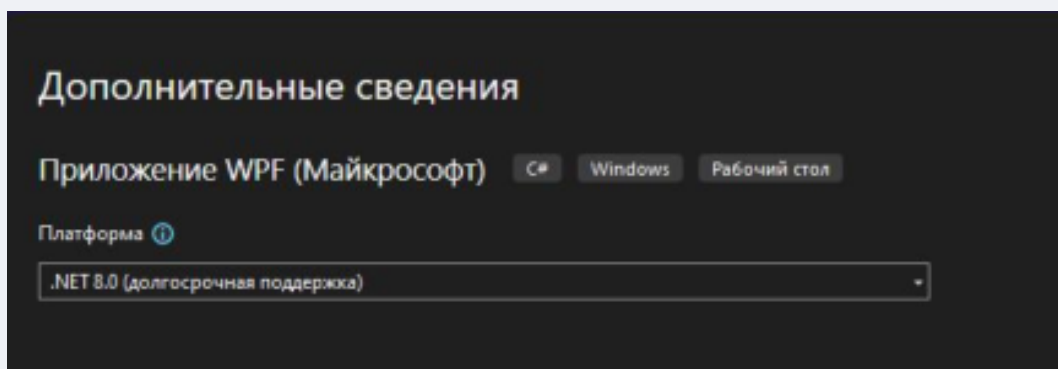


Рис. 3: Задание имени проекта.

Структура проекта состоит из следующих файлов:

- App.xaml — конфигурация приложения. Определяет основной класс приложения, наследуемый от Application. В нём задаются стили, шаблоны приложения и стартовая точка<sup>3</sup>.
- App.xaml.cs — логика приложения. Содержит класс App, который может переопределять методы, такие как OnStartup, для настройки поведения при запуске, или обрабатывать события приложения.
- MainWindow.xaml — разметка и структура UI (интерфейса) главного окна.
- MainWindow.xaml.cs — логика и обработчики событий окна.

<sup>3</sup>Обычно указывается начальное окно в свойстве StartupUri

## 3.2 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 в листинге 1

---

```
namespace StoreManager.Models
{
    // Модель заказа, содержащая информацию о клиенте и товарах
    public class Order
    {
        // Уникальный идентификатор заказа
        public int Id { get; set; }

        // Список элементов заказа (товар + количество)
        public List<OrderItem> Items { get; set; }

        // Имя клиента
        public string CustomerName { get; set; }

        // Дата создания заказа
        public DateTime OrderDate { get; set; }

        // Общая сумма заказа, вычисляемая как сумма цен товаров умноженная на их количество
        public decimal TotalPrice => Items.Sum(item => item.Product.Price * item.Quantity);

        // Конструктор, инициализирующий пустой список товаров
        public Order()
        {
            Items = new List<OrderItem>();
        }
    }
}
```

---

Листинг 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).

Также необходим класс позволяющий связать товар и его количество в заказе. Пример, созданного класса OrderItem И объявления его свойств приведен в листинге 2

---

```
namespace StoreManager.Models
{
    // Модель элемента заказа, связывающая товар и его количество
    public class OrderItem
    {
        // Ссылка на товар
        public Product Product { get; set; }

        // Количество единиц товара в заказе
        public int Quantity { get; set; }
    }
}
```

---

Листинг 2: Класс **OrderItem**

Также необходимо создать файл Product.cs. Данный файл отвечает за создание нового объекта товара. Пример, класса Product приведен в листинге 3.

```
namespace StoreManager.Models
{
    {
        // Модель товара, представляющая продукт в магазине
        public class Product
        {
            // Уникальный идентификатор товара
            public int Id { get; set; }

            // Название товара
            public string Name { get; set; }

            // Цена товара
            public decimal Price { get; set; }

            // Количество товара на складе
            public int Stock { get; set; }
        }
    }
}
```

---

Листинг 3: Класс **Product.cs**

Также необходимо создать папку Converters. Она будет нужна для создания конвертера. В данной папке нужно создать класс BooleanToVisibilityConverter. Пример разметки для данного класса приведен в листинге 4.

```
using System;
using System.Globalization;
using System.Windows;
using System.Windows.Data;

namespace StoreManager.Converters
{
    // Конвертер для отображения placeholder'ов в TextBox
    public class BooleanToVisibilityConverter : IValueConverter
    {
        // Преобразует boolean (IsEmpty) в Visibility для TextBlock
        public object Convert(object value, Type targetType, object parameter, CultureInfo culture)
        {
            return value is bool isEmpty && isEmpty ? Visibility.Visible : Visibility.Collapsed;
        }

        // Обратное преобразование не используется
        public object ConvertBack(object value, Type targetType, object parameter, CultureInfo culture)
        {
            throw new NotImplementedException();
        }
    }
}
```

---

Листинг 4: Класс **BooleanToVisibilityConverter**

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)
- 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.

### 3.3 ШАГ 3: РЕАЛИЗАЦИЯ ИНТЕРФЕЙСА

Для того чтобы реализовать внешний вид приложения необходимо добавить соответствующую разметку в файл MainWindow.xaml. Пример разметки для данного файла будет представлен ниже в листинге 5

```
<Window x:Class="StoreManager.MainWindow"
        xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
        xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
        xmlns:converters="clr-namespace:StoreManager.Converters"
        Title="Управление магазином" Height="650" Width="950"
        WindowStartupLocation="CenterScreen" Background="#F5F5F5">

    <!-- Ресурсы окна: конвертеры и стили -->
    <Window.Resources>
        <!-- Конвертер для преобразования bool в Visibility -->
        <converters:BooleanToVisibilityConverter x:Key="BooleanToVisibilityConverter"/>

        <!-- Общий стиль кнопок -->
        <Style TargetType="Button">
            <Setter Property="Background" Value="#FF6200EE"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="Padding" Value="10,5"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="BorderThickness" Value="0"/>
            <Setter Property="Cursor" Value="Hand"/>
        </Style>

        <!-- Специальный стиль кнопок изменения количества -->
        <Style x:Key="QuantityButtonStyle" TargetType="Button">
            <Setter Property="Background" Value="#FF6200EE"/>
            <Setter Property="Foreground" Value="White"/>
            <Setter Property="FontSize" Value="12"/>
            <Setter Property="Width" Value="25"/>
            <Setter Property="Height" Value="25"/>
            <Setter Property="Margin" Value="5,0"/>
            <Setter Property="BorderThickness" Value="0"/>
            <Setter Property="Cursor" Value="Hand"/>
        </Style>

        <!-- Стиль для полей ввода -->
        <Style TargetType="TextBox">
            <Setter Property="FontSize" Value="14"/>
            <Setter Property="Padding" Value="5"/>
            <Setter Property="Margin" Value="5"/>
            <Setter Property="BorderBrush" Value="#FFCCCC"/>
        </Style>

        <!-- Стиль текста-заглушки (placeholder) -->
        <Style TargetType="TextBlock" x:Key="PlaceholderStyle">
            <Setter Property="Foreground" Value="Gray"/>
            <Setter Property="FontStyle" Value="Italic"/>
            <Setter Property="Margin" Value="8,5,0,0"/>
            <Setter Property="IsHitTestVisible" Value="False"/>
        </Style>
    </Window.Resources>

    <!-- Основная сетка, задаёт структуру окна -->
    <Grid Margin="-16,0,0,-35">
        <Grid.ColumnDefinitions>
            <!-- Колонки для панели товаров и заказов -->
            <ColumnDefinition Width="107*"/>
            <ColumnDefinition Width="344*"/>
            <ColumnDefinition Width="450*"/>
        </Grid.ColumnDefinitions>
```

```
<!-- Панель товаров -->
<Border Grid.Column="0" Margin="10" Background="White" CornerRadius="10" Padding="10" Grid.ColumnSpan="2">
    <StackPanel>
        <TextBlock Text="Товары" FontSize="20" FontWeight="Bold" Margin="0,0,0,10"/>

        <!-- Список товаров -->
        <ListBox x:Name="ProductsList" Height="250" BorderBrush="#FFCCCC"
            SelectionChanged="ProductsList_SelectionChanged"
            MouseLeftButtonDown="ProductsList_MouseLeftButtonDown">
            <ListBox.ItemTemplate>
                <DataTemplate>
                    <StackPanel Orientation="Horizontal">
                        <!-- Название -->
                        <TextBlock Text="{Binding Name}" FontWeight="Bold" Margin="0,0,10,0"/>
                        <!-- Цена -->
                        <TextBlock Text="{Binding Price, StringFormat={}{0:C}}"/>
                        <!-- Остаток -->
                        <TextBlock Text=" Остаток(:" FontStyle="Italic"/>
                        <TextBlock Text="{Binding Stock}">
                            <!-- Подсветка остатка -->
                            <TextBlock.Style>
                                <Style TargetType="TextBlock">
                                    <Style.Triggers>
                                        <!-- Если 0 - красным -->
                                        <DataTrigger Binding="{Binding Stock}" Value="0">
                                            <Setter Property="Foreground" Value="Red"/>
                                        </DataTrigger>
                                        <!-- Если меньше 5 - оранжевым -->
                                        <DataTrigger Binding="{Binding Stock, ConverterParameter=5}" Value="True">
                                            <Setter Property="Foreground" Value="Orange"/>
                                        </DataTrigger>
                                    </Style.Triggers>
                                </Style>
                            </TextBlock.Style>
                        </TextBlock>
                    </StackPanel>
                </DataTemplate>
            </ListBox.ItemTemplate>
        </ListBox>

        <!-- Кнопка добавления в заказ -->
        <Button x:Name="AddToOrderButton" Content="Добавить в заказ" Click="AddToOrder_Click" HorizontalAlignment="Right"/>

        <!-- Поля ввода нового товара -->
        <Grid>
            <TextBox x:Name="ProductName"/>
            <TextBlock Text="Название товара" Style="{StaticResource PlaceholderStyle}"
                Visibility="{Binding ElementName=ProductName, Path=Text.IsEmpty, Converter={StaticResource BooleanToVisibilityConverter}}"/>
        </Grid>
        <Grid>
            <TextBox x:Name="ProductPrice"/>
            <TextBlock Text="Цена" Style="{StaticResource PlaceholderStyle}"
                Visibility="{Binding ElementName=ProductPrice, Path=Text.IsEmpty, Converter={StaticResource BooleanToVisibilityConverter}}"/>
        </Grid>
        <Grid>
            <TextBox x:Name="ProductStock"/>
            <TextBlock Text="Остаток" Style="{StaticResource PlaceholderStyle}"
                Visibility="{Binding ElementName=ProductStock, Path=Text.IsEmpty, Converter={StaticResource BooleanToVisibilityConverter}}"/>
        </Grid>

        <!-- Кнопки управления товарами -->
        <StackPanel HorizontalAlignment="Right" Width="222">
            <Button Content="Добавить товар" Click="AddProduct_Click"/>
            <Button Content="Обновить товар" Click="UpdateProduct_Click"/>
            <Button Content="Удалить товар" Click="DeleteProduct_Click"/>
        </StackPanel>
    </StackPanel>
</Border>

<!-- Панель заказов -->
<Border Grid.Column="2" Margin="10" Background="White" CornerRadius="10" Padding="10">
    <StackPanel>
        <TextBlock Text="Заказы" FontSize="20" FontWeight="Bold" Margin="0,0,0,10"/>

        <!-- Список заказов -->
        <ListBox x:Name="OrdersList" Height="150" BorderBrush="#FFCCCC"
            SelectionChanged="OrdersList_SelectionChanged"
            MouseLeftButtonDown="ProductsList_MouseLeftButtonDown">
            <ListBox.ItemTemplate>
                <DataTemplate>
                    <StackPanel>
                        <TextBlock Text="{Binding CustomerName}" FontWeight="Bold"/>
                        <TextBlock Text="{Binding OrderDate, StringFormat={}{0:dd.MM.yyyy HH:mm}}"/>
                        <TextBlock Text="{Binding TotalPrice, StringFormat={}{0:C}}"/>
                        <!-- Список товаров в заказе -->
                        <ItemsControl ItemsSource="{Binding Items}">
                            <ItemsControl.ItemTemplate>
                                <DataTemplate>
                                    <StackPanel Orientation="Horizontal">
                                        <TextBlock Text="{Binding ProductName}" Margin="0,0,10,0"/>
                                        <TextBlock Text="{Binding ProductPrice, StringFormat={}{0:C}}"/>
                                        <TextBlock Text=" x " FontStyle="Italic"/>
                                        <TextBlock Text="{Binding Quantity}">

```

```

        </StackPanel>
    </DataTemplate>
</ItemsControl.ItemTemplate>
</ItemsControl>
</StackPanel>
</DataTemplate>
</ListBox.ItemTemplate>
</ListBox>

<!-- Список выбранных товаров для нового заказа -->
<TextBlock Text="Выбранные товары для заказа" FontSize="14" FontWeight="Bold" Margin="0,10,0,5"/>
<ListBox x:Name="SelectedProductsList" Height="100" BorderBrush="#FFCCCC" MouseLeftButtonDown="ProductsList_MouseLeftButtonDown">
    <ListBox.ItemTemplate>
        <DataTemplate>
            <StackPanel Orientation="Horizontal">
                <TextBlock Text="{Binding Product.Name}" FontWeight="Bold" Margin="0,0,10,0"/>
                <TextBlock Text="{Binding Product.Price, StringFormat={}{0:C}}"/>
                <TextBlock Text=" x " FontStyle="Italic"/>
                <TextBlock Text="{Binding Quantity}" Margin="0,0,10,0"/>
                <!-- Кнопки изменения количества -->
                <Button Content="+" Style="{StaticResource QuantityButtonStyle}" Click="IncreaseQuantity_Click" Tag="{Binding}"/>
                <Button Content="-" Style="{StaticResource QuantityButtonStyle}" Click="DecreaseQuantity_Click" Tag="{Binding}"/>
            </StackPanel>
        </DataTemplate>
    </ListBox.ItemTemplate>
</ListBox>

<!-- Кнопка удаления товара из заказа -->
<Button Content="Удалить товар из заказа" Click="RemoveProductFromOrder_Click" HorizontalAlignment="Right"/>

<!-- Поле ввода имени клиента -->
<Grid>
    <TextBox x:Name="CustomerName"/>
    <TextBlock Text="Имя клиента" Style="{StaticResource PlaceholderStyle}"
        Visibility="{Binding ElementName=CustomerName, Path=Text.IsEmpty, Converter={StaticResource BooleanToVisibilityConverter}}"/>
</Grid>

<!-- Кнопки управления заказами -->
<StackPanel Orientation="Horizontal" HorizontalAlignment="Right">
    <Button Content="Создать заказ" Click="AddOrder_Click"/>
    <Button Content="Обновить заказ" Click="UpdateOrder_Click"/>
    <Button Content="Удалить заказ" Click="DeleteOrder_Click"/>
</StackPanel>
</StackPanel>
</Border>
</Grid>
</Window>

```

Листинг 5: Пример разметки главного окна **MainWindow**

### 3.4 ШАГ 4: РЕАЛИЗАЦИЯ ЛОГИКИ

Файл `MainWindow.xaml.cs` – это code-behind для главного окна приложения. 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.

Структура кода состоит из следующих элементов

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.

В листинге 6 представлен пример кода для класса `MainWindow`.

```
using StoreManager.Models;
using System.Windows;
using System.Windows.Controls;
using System.Windows.Input;
using System.Windows.Media;

namespace StoreManager
{
    // Главное окно приложения для управления товарами и заказами
    public partial class MainWindow : Window
    {
        // Список всех товаров
        private List<Product> products = new List<Product>();
        // Список всех заказов
        private List<Order> orders = new List<Order>();
        // Следующий ID для нового товара
        private int nextProductId = 1;
        // Следующий ID для нового заказа
        private int nextOrderId = 1;
        // Список выбранных товаров для текущего заказа
        private List<OrderItem> selectedProductsForOrder = new List<OrderItem>();
    }
}
```

Листинг 6: Пример кода для класса `Main`

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`), поэтому код вручную обновляет списки. Далее необходимо объявить основной метод `MainWindow`. Пример кода для данного метода представлен в листинге 7

```
public MainWindow()
{
    InitializeComponent(); // Инициализация UI
    UpdateProductList();   // Обновление списка товаров
    UpdateOrderList();     // Обновление списка заказов
    UpdateSelectedProductsList(); // Обновление списка выбранных товаров
}
```

Листинг 7: Пример кода для класса `Main`

`InitializeComponent()` — метод, автоматически сгенерированный WPF, который загружает разметку из `MainWindow.xaml`, инициализирует элементы интерфейса (такие как кнопки, списки и текстовые поля) и устанавливает необходимые привязки. Без его вызова пользовательский интерфейс не будет отображён.

Вызовы методов обновления:

- `UpdateProductList()` — устанавливает `ItemsSource` для `ListBox` с товарами, 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 подготавливает приложение к работе, загружая UI и синхронизируя данные с элементами управления
- Поскольку списки изначально пусты, вызовы обновления предотвращают ошибки отображения.

Методы для обновления списков товаров, заказов и выбранных товаров для заказа представлены в листинге 8

---

```
// Обновляет отображение списка товаров
private void UpdateProductList()
{
    ProductsList.ItemsSource = null;
    ProductsList.ItemsSource = products;
}

// Обновляет отображение списка заказов
private void UpdateOrderList()
{
    OrdersList.ItemsSource = null;
    OrdersList.ItemsSource = orders;
}

// Обновляет отображение списка выбранных товаров
private void UpdateSelectedProductsList()
{
    SelectedProductsList.ItemsSource = null;
    SelectedProductsList.ItemsSource = selectedProductsForOrder;
}
```

---

Листинг 8: 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.

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 в листинге 9

```
// Обработчик нажатия кнопки "Добавить товар"
private void AddProduct_Click(object sender, RoutedEventArgs e)
{
    // Проверка корректности введенных данных
    if (string.IsNullOrEmpty(ProductName.Text) ||
        !decimal.TryParse(ProductPrice.Text, out decimal price) ||
        !int.TryParse(ProductStock.Text, out int stock) ||
        stock < 0)
    {
        MessageBox.Show(
            """
            Пожалуйста, введите корректные данные о товаре.
            Остаток не может быть отрицательным.
            """
            "Ошибка",
            MessageBoxButton.OK,
            MessageBoxImage.Error
        );

        return;
    }

    // Создание нового товара
    var product = new Product
    {
        Id = nextProductId++,
        Name = ProductName.Text,
        Price = price,
        Stock = stock
    };

    products.Add(product); // Добавление товара в список
    UpdateProductList();   // Обновление UI
    ClearProductFields();   // Очистка полей ввода
}
```

Листинг 9: 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. Создает 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).

Следующий метод позволяет обновить (редактировать) информацию о конкретном товаре. Реализация метода обновления представлен в листинге 10

```
// Обработчик нажатия кнопки "Обновить товар"
private void UpdateProduct_Click(object sender, RoutedEventArgs e)
{
    if (ProductsList.SelectedItem is Product selectedProduct)
    {
        // Проверка корректности введенных данных
        if (string.IsNullOrEmpty(ProductName.Text) ||
            !decimal.TryParse(ProductPrice.Text, out decimal price) ||
            !int.TryParse(ProductStock.Text, out int stock) ||
            stock < 0)
        {
            MessageBox.Show(
                "Пожалуйста, введите корректные данные о товаре. " +
                "Остаток не может быть отрицательным.",
                "Ошибка",
                MessageBoxButton.OK,
                MessageBoxImage.Error
            );
            "Ошибка", MessageBoxButton.OK, MessageBoxImage.Error);
            return;
        }

        // Обновление свойств выбранного товара
        selectedProduct.Name = ProductName.Text;
        selectedProduct.Price = price;
        selectedProduct.Stock = stock;

        UpdateProductList(); // Обновление UI
        ClearProductFields(); // Очистка полей ввода
    }
    else
    {
        MessageBox.Show(
            "Пожалуйста, выберите товар для обновления.",
            "Ошибка",
            MessageBoxButton.OK,
            MessageBoxImage.Warning
        );
    }
}
```

Листинг 10: 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.

Для того, чтобы исключить конкретный товар из общего списка, используется метод DeleteProduct из листинга 11.

```
// Обработчик нажатия кнопки "Удалить товар"
private void DeleteProduct_Click(object sender, RoutedEventArgs e)
{
    if (ProductsList.SelectedItem is Product selectedProduct)
    {
        // Проверка, используется ли товар в заказах
        if (orders.Any(order => order.Items.Any(item => item.Product.Id == selectedProduct.Id)))
        {
            MessageBox.Show(
                "Нельзя удалить товар, используемый в заказах.",
                "Ошибка",
                MessageBoxButton.OK,
                MessageBoxImage.Warning
            );
            return;
        }

        // Подтверждение удаления

        var result = MessageBox.Show($"Вы уверены, что хотите удалить товар '{selectedProduct.Name}'?",
            "Подтверждение удаления", MessageBoxButton.YesNo, MessageBoxImage.Question);

        if (result == MessageBoxResult.Yes)
        {
            selectedProductsForOrder.RemoveAll(item => item.Product.Id == selectedProduct.Id); // Удаление из
            текущего заказа
            products.Remove(selectedProduct); // Удаление из списка товаров
            UpdateProductList();
            UpdateSelectedProductsList();
            ClearProductFields();
        }
    }
    else
    {
        MessageBox.Show(
            "Пожалуйста, выберите товар для удаления.",
            "Ошибка",
            MessageBoxButton.OK,
            MessageBoxImage.Warning
        );
    }
}
```

Листинг 11: 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.

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 (листинг 12)

```
// Обработчик изменения выделения в списке товаров
private void ProductsList_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (ProductsList.SelectedItem is Product selectedProduct)
    {
        // Заполнение полей ввода данными выбранного товара
        ProductName.Text = selectedProduct.Name;
        ProductPrice.Text = selectedProduct.Price.ToString();
        ProductStock.Text = selectedProduct.Stock.ToString();
    }
}

private void AddToOrder_Click(object sender, RoutedEventArgs e)
{
    if (ProductsList.SelectedItem is Product selectedProduct)
    {
        if (selectedProduct.Stock <= 0)
        {
            MessageBox.Show(
                "Товара больше нет на складе.",
                "Ошибка",
                MessageBoxButton.OK,
                MessageBoxImage.Warning
            );
            return;
        }

        var existingItem = selectedProductsForOrder.FirstOrDefault(
            item => item.Product.Id == selectedProduct.Id
        );

        if (existingItem != null)
        {
            existingItem.Quantity++;
        }
        else
        {
            selectedProductsForOrder.Add(
                new OrderItem
                {
                    Product = selectedProduct,
                    Quantity = 1
                }
            );
        }

        selectedProduct.Stock--; // <== уменьшение запаса
        UpdateProductList();
        UpdateSelectedProductsList();
    }
    else
    {
        MessageBox.Show(
            "Товара больше нет на складе.",
            "Ошибка",
            MessageBoxButton.OK,
            MessageBoxImage.Warning
        );
    }
}
```

Листинг 12: Методы редактирования товара и добавления в список выбранных товаров для заказа

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 в листинге 13.

```
// Обработчик нажатия кнопки "Создать заказ"
private void AddOrder_Click(object sender, RoutedEventArgs e)
{
    // Проверка имени клиента
    if (string.IsNullOrWhiteSpace(CustomerName.Text))
    {
        MessageBox.Show(
            "Пожалуйста, введите имя клиента.",
            "Ошибка",
            MessageBoxButton.OK,
            MessageBoxImage.Error
        );
        return;
    }

    // Проверка наличия товаров в заказе
    if (!selectedProductsForOrder.Any())
    {
        MessageBox.Show(
            "Пожалуйста, выберите хотя бы один товар для заказа.",
            "Ошибка",
            MessageBoxButton.OK,
            MessageBoxImage.Warning
        );
        return;
    }

    // Создание нового заказа
    var order = new Order
    {
        Id = nextOrderId++,
        CustomerName = CustomerName.Text,
        OrderDate = DateTime.Now,
        Items = new List<OrderItem>(selectedProductsForOrder)
    };

    orders.Add(order); // Добавление заказа в список
    UpdateProductList();
    UpdateOrderList();
    ClearOrderFields(); // Очистка полей заказа
}
```

Листинг 13: Метод добавления товара в список выбранных товаров для заказа

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 в листинге 14.

```
// Обработчик нажатия кнопки "Обновить заказ"
private void UpdateOrder_Click(object sender, RoutedEventArgs e)
{
    if (OrdersList.SelectedItem is Order selectedOrder)
    {
        // Проверка имени клиента
        if (string.IsNullOrWhiteSpace(CustomerName.Text))
        {
            MessageBox.Show(
                "Пожалуйста, введите имя клиента.",
                "Ошибка",
                MessageBoxButton.OK,
                MessageBoxImage.Error
            );
            return;
        }

        // Проверка наличия товаров
        if (!selectedProductsForOrder.Any())
        {
            MessageBox.Show(
                "Пожалуйста, выберите хотя бы один товар для заказа.",
                "Ошибка",
                MessageBoxButton.OK,
                MessageBoxImage.Warning
            );
            return;
        }

        // Обновление заказа
        selectedOrder.CustomerName = CustomerName.Text;
        selectedOrder.OrderDate = DateTime.Now;
        selectedOrder.Items.Clear();
        selectedOrder.Items.AddRange(selectedProductsForOrder);

        // Проверка доступности перед применением изменений
        foreach (var item in selectedProductsForOrder)
        {
            int availableStock = item.Product.Stock +
                var oldQuantity = selectedOrder.Items
                    .Where(i => i.Product.Id == item.Product.Id)
                    .Sum(i => i.Quantity); // учитываем старый заказ

            if (item.Quantity > availableStock)
            {
                MessageBox.Show(
                    $"Недостаточно товара '{item.Product.Name}' " +
                    "на складе для обновления заказа.",
                    "Ошибка",
                    MessageBoxButton.OK,
                    MessageBoxImage.Warning
                );

                return;
            }
        }

        UpdateProductList();
        UpdateOrderList();
        ClearOrderFields();
    }
    else
    {
        MessageBox.Show(
            "Пожалуйста, выберите заказ для обновления.",
            "Ошибка",
            MessageBoxButton.OK,
        )
    }
}
```



```
        MessageBoxImage.Warning
    );
}
```

---

Листинг 14: Метод обновления заказа

- 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 в листинге 15.

```
// Обработчик нажатия кнопки "Удалить заказ"
private void DeleteOrder_Click(object sender, RoutedEventArgs e)
{
    if (OrdersList.SelectedItem is Order selectedOrder)
    {
        // Подтверждение удаления
        var result = MessageBox.Show(
            $"Вы уверены, что хотите удалить заказ для клиента '{selectedOrder.CustomerName}'?",
            "Подтверждение",
            MessageBoxButton.YesNo,
            MessageBoxImage.Question
        );
        "Подтверждение удаления", MessageBoxButton.YesNo, MessageBoxImage.Question);

        if (result == MessageBoxResult.Yes)
        {
            // Возврат запасов
            foreach (var item in selectedOrder.Items)
            {
                item.Product.Stock += item.Quantity;
            }

            orders.Remove(selectedOrder); // Удаление заказа
            UpdateProductList();
            UpdateOrderList();
            ClearOrderFields();
        }
    }
    else
    {
        MessageBox.Show(
            "Пожалуйста, выберите заказ для удаления.",
            "Ошибка",
            MessageBoxButton.OK,
            MessageBoxImage.Warning
        );
    }
}
```

Листинг 15: Метод удаления заказа

Следующий метод нужен для удаления товара из заказа. Код для данного метода представлен в листинге 16.

```
// Обработчик нажатия кнопки "Удалить товар из заказа"
private void RemoveProductFromOrder_Click(object sender, RoutedEventArgs e)
{
    if (SelectedProductsList.SelectedItem is OrderItem selectedItem)
    {
        // Проверка: это последний товар в заказе?
        if (selectedProductsForOrder.Count == 1 &&
            OrdersList.SelectedItem is Order selectedOrder)
        {
            var result = MessageBox.Show(
                $"Удаление последнего товара приведёт к удалению заказа клиента" +
                $"{selectedOrder.CustomerName}'. Продолжить?",
                "Подтверждение удаления",
                MessageBoxButton.YesNo,
                MessageBoxImage.Warning);

            if (result != MessageBoxResult.Yes)
                return;

            // Удаляем товар и возвращаем остаток
            selectedProductsForOrder.Remove(selectedItem);
            selectedItem.Product.Stock += selectedItem.Quantity;

            // Удаляем сам заказ
            orders.Remove(selectedOrder);
            UpdateProductList();
            UpdateOrderList();
            ClearOrderFields();
        }
        else
        {
            // Обычное удаление товара
            selectedProductsForOrder.Remove(selectedItem);
            selectedItem.Product.Stock += selectedItem.Quantity;

            UpdateProductList();
            UpdateSelectedProductsList();
        }
    }
    else
    {
        MessageBox.Show(
            "Пожалуйста, выберите товар для удаления из заказа.",
            "Ошибка",
            MessageBoxButton.OK,
            MessageBoxImage.Warning
        );
    }
}
```

Листинг 16: 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.

Далее рассмотрим методы, отвечающие за увеличение и уменьшение количества товара в заказе (кнопки «+» и «-»). Пример данных методов представлен в листинге 17.

```
// Обработчик нажатия кнопки "+" для увеличения количества
private void IncreaseQuantity_Click(object sender, RoutedEventArgs e)
{
    if (sender is Button button && button.Tag is OrderItem item)
    {
        if (item.Product.Stock < 1)
        {
            MessageBox.Show(
                "Товара больше нет на складе.",
                "Ошибка",
                MessageBoxButton.OK,
                MessageBoxImage.Warning
            );
            return;
        }

        item.Quantity++;
        item.Product.Stock--; // резерв
        UpdateProductList();
        UpdateSelectedProductsList();
    }
}

// Обработчик нажатия кнопки "-" для уменьшения количества
private void DecreaseQuantity_Click(object sender, RoutedEventArgs e)
{
    if (sender is Button button && button.Tag is OrderItem item)
    {
        if (item.Quantity > 1)
        {
            item.Quantity--; // Уменьшение количества
        }
        else
        {
            selectedProductsForOrder.Remove(item); // Удаление товара
        }
        item.Product.Stock++;
        UpdateProductList();
        UpdateSelectedProductsList();
    }
}
```

Листинг 17: Методы управления количеством товара в заказе

Данный метод позволяет выделить определенный заказ и изменить имя заказчика. Реализация метода представлена в листинге 18.

---

```
// Обработчик изменения выделения в списке заказов
private void OrdersList_SelectionChanged(object sender, SelectionChangedEventArgs e)
{
    if (OrdersList.SelectedItem is Order selectedOrder)
    {
        // Заполнение полей данными выбранного заказа
        CustomerName.Text = selectedOrder.CustomerName;
        selectedProductsForOrder.Clear();
        selectedProductsForOrder.AddRange(selectedOrder.Items);
        UpdateSelectedProductsList();
    }
}
```

---

Листинг 18: 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 и 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 в листинге 19.

```
// Очистка полей ввода для товаров
private void ClearProductFields()
{
    ProductName.Text = "";
    ProductPrice.Text = "";
    ProductStock.Text = "";
    ProductsList.SelectedItem = null; // Сброс выделения
}

// Очистка полей ввода для заказов
private void ClearOrderFields()
{
    CustomerName.Text = "";
    selectedProductsForOrder.Clear();
    UpdateSelectedProductsList();
    OrdersList.SelectedItem = null; // Сброс выделения
    // Выделение в ProductsList не сбрасывается
}

// Обработчик клика мышью по списку товаров
private void ProductsList_MouseLeftButtonDown(object sender, MouseButtonEventArgs e)
{
    var listBox = sender as ListBox;
    var hitTestResult = VisualTreeHelper.HitTest(listBox, e.GetPosition(listBox));
    if (hitTestResult != null)
    {
        // Проверка, попал ли клик на ListBoxItem
        var element = hitTestResult.VisualHit;
        while (element != null && !(element is ListBoxItem))
        {
            element = VisualTreeHelper.GetParent(element);
        }

        if (element == null)
        {
            listBox.SelectedItem = null; // Сброс выделения при клике по пустому месту
        }
    }
}
}
```

Листинг 19: 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.

## 4 ШАГ 5: ДЕМОНСТРАЦИЯ РАБОТЫ 63

В итоговом варианте внешний вид приложения представлен на Рисунке 4

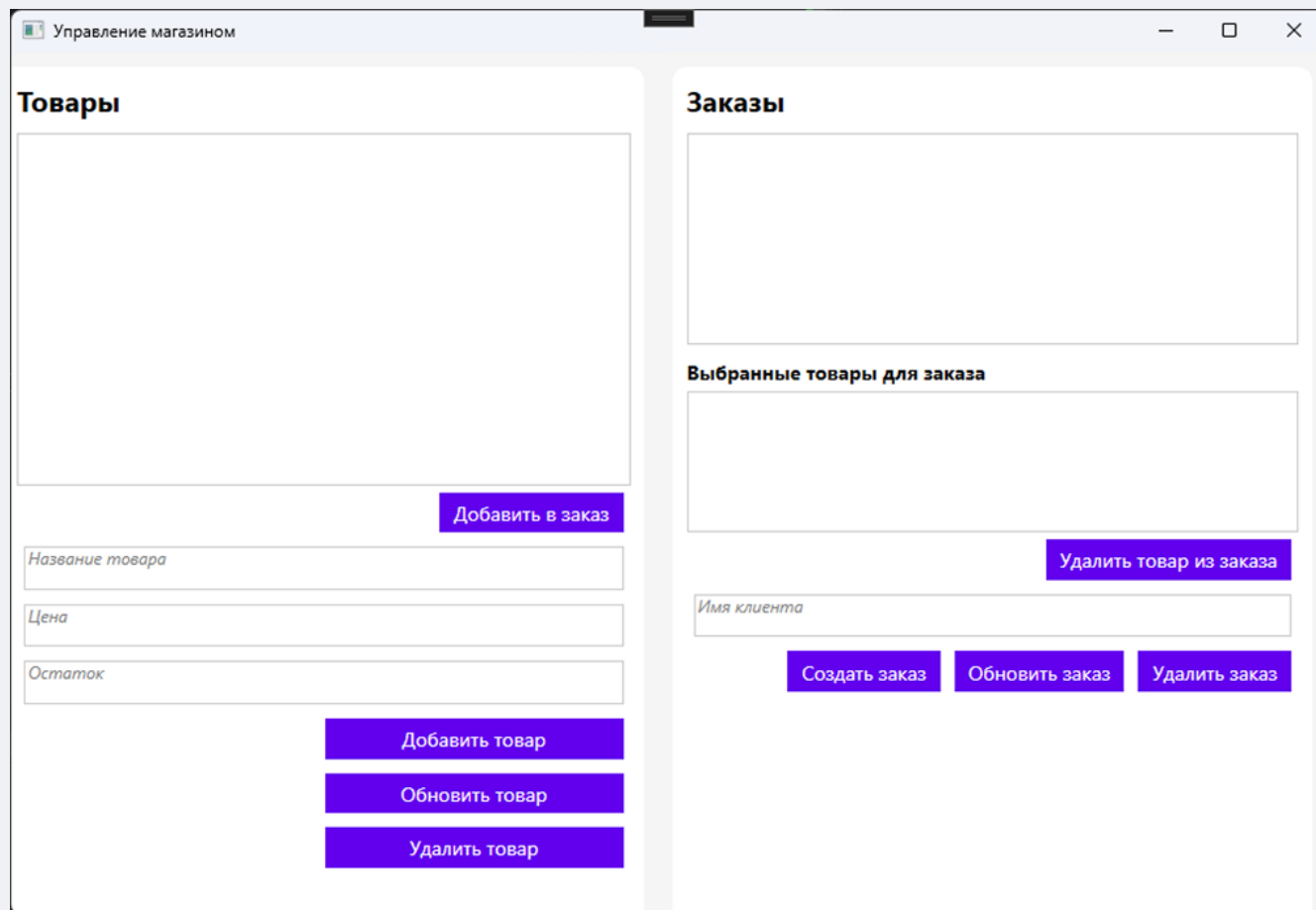


Рис. 4: Интерфейс приложения

В качестве примера отображения информации в приложении, введены данные о товарах и заказе. Пример интерфейса приложения с введенными данными приведен на Рисунке 5

Управление магазином

### Товары

Добавить в заказ

Хлеб

100

20

Добавить товар

Обновить товар

Удалить товар

### Заказы

Удалить товар из заказа

Имя клиента

Создать заказ

Обновить заказ

Удалить заказ

Рис. 5: Пример заполнения информации в приложении



Далее по нажатию кнопки ”Добавить товар”, товар добавляется в общий список. Таким образом может добавит группу товаров (Рисунок 6)

The screenshot displays a web application interface for managing a store. It is divided into two main panels: 'Товары' (Goods) on the left and 'Заказы' (Orders) on the right.

**Товары (Goods) Panel:**

- Товары (Goods):** A list of items with their prices and stock levels:
  - 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)
- Добавить в заказ (Add to order):** A button located below the list.
- Form fields:** Three input fields for adding new goods:
  - Название товара (Goods name)
  - Цена (Price)
  - Остаток (Stock)
- Buttons:** Three buttons for managing the goods list:
  - Добавить товар (Add goods)
  - Обновить товар (Update goods)
  - Удалить товар (Delete goods)

**Заказы (Orders) Panel:**

- Заказы (Orders):** A large empty box for displaying orders.
- Выбранные товары для заказа (Selected goods for order):** A box for displaying selected items.
- Удалить товар из заказа (Delete item from order):** A button located below the selected goods box.
- Form field:** An input field for the client's name: Имя клиента (Client name).
- Buttons:** Three buttons for managing orders:
  - Создать заказ (Create order)
  - Обновить заказ (Update order)
  - Удалить заказ (Delete order)

Рис. 6: Пример добавления группы товаров

Далее необходимо выбрать конкретный товар из списка, по нажатию кнопки доаввить его в список выбранных товаров. В данном списке помощью кнопки «+» и «-» можно отредактировать количество товаров для добавления в заказ (Рисунок 7)

**Управление магазином**

### 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

**Добавить товар**  
**Обновить товар**  
**Удалить товар**

### Заказы

**Выбранные товары для заказа**

Хлеб	\$5.00 x 1	+	-
Торт	\$50.00 x 1	+	-
Творог	\$2.00 x 1	+	-
Яблоки	\$0.40 x 1	+	-

**Удалить товар из заказа**

Имя клиента

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

Рис. 7: Пример добавления группы товаров

Далее необходимо ввести имя клиента (название заказа) и нажать кнопку «Создать заказ» для добавления нового заказа в список заказов (Рисунок 8).

**Управление магазином**

### 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

Рис. 8: Пример добавления группы товаров

При нажатии на конкретный заказ можно вывести список товаров (Рисунок 9).

Управление магазином

### 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

Хлеб \$5.00 x 1 + -  
Торт \$50.00 x 2 + -  
Творог \$2.00 x 6 + -  
Яблоки \$0.40 x 12 + -

Удалить товар из заказа

Иван

Создать заказ Обновить заказ Удалить заказ

Рис. 9: Вывод списка товаров в заказе

При нажатии кнопки «Удалить товар из заказа» можно удалить конкретный товар добавленный в заказ (Рисунок 10).

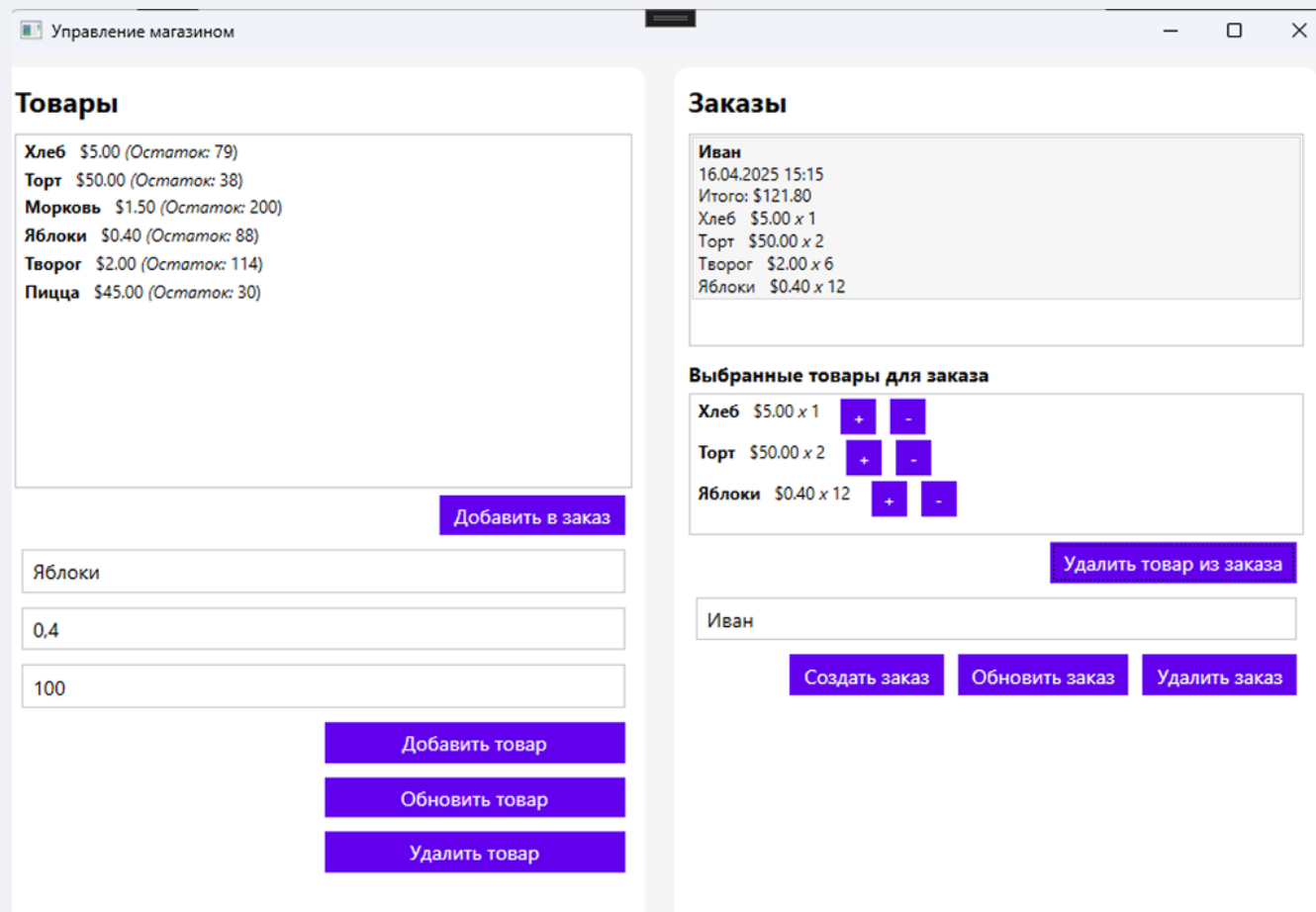


Рис. 10: Удаление товара из заказа

Кнопка «Обновить заказ» позволяет обновить информацию о заказе удалить конкретный товар добавленный в заказ (Рисунок 11).

The screenshot shows a web application titled "Управление магазином" (Store Management). It is divided into two main panels: "Товары" (Goods) and "Заказы" (Orders).

**Товары (Goods):**

- 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)

Below the list, there is a search bar containing "Яблоки", a quantity input field with "0,4", and a stock input field with "100". To the right of the search bar is a blue button "Добавить в заказ". Below the input fields are three stacked blue buttons: "Добавить товар", "Обновить товар", and "Удалить товар".

**Заказы (Orders):**

The order details for "Иван" are shown:

- 16.04.2025 15:20
- Итого: \$109.80
- Хлеб \$5.00 x 1
- Торт \$50.00 x 2
- Яблоки \$0.40 x 12

Below the order details is a section "Выбранные товары для заказа" (Selected goods for order) with an empty box. To the right of this box is a blue button "Удалить товар из заказа".

At the bottom of the "Заказы" panel, there is a text input field for "Имя клиента" (Client name). Below it are three stacked blue buttons: "Создать заказ", "Обновить заказ", and "Удалить заказ".

Рис. 11: Обновление списка товаров в заказе

При нажатии кнопки «Удалить заказ» можно удалить конкретный заказ из списка (Рисунок 12). При удалении на экран выводится сообщение с подтверждением. После удаления количество остатков товара увеличивается в соответствии с тем, сколько товара было в заказе.

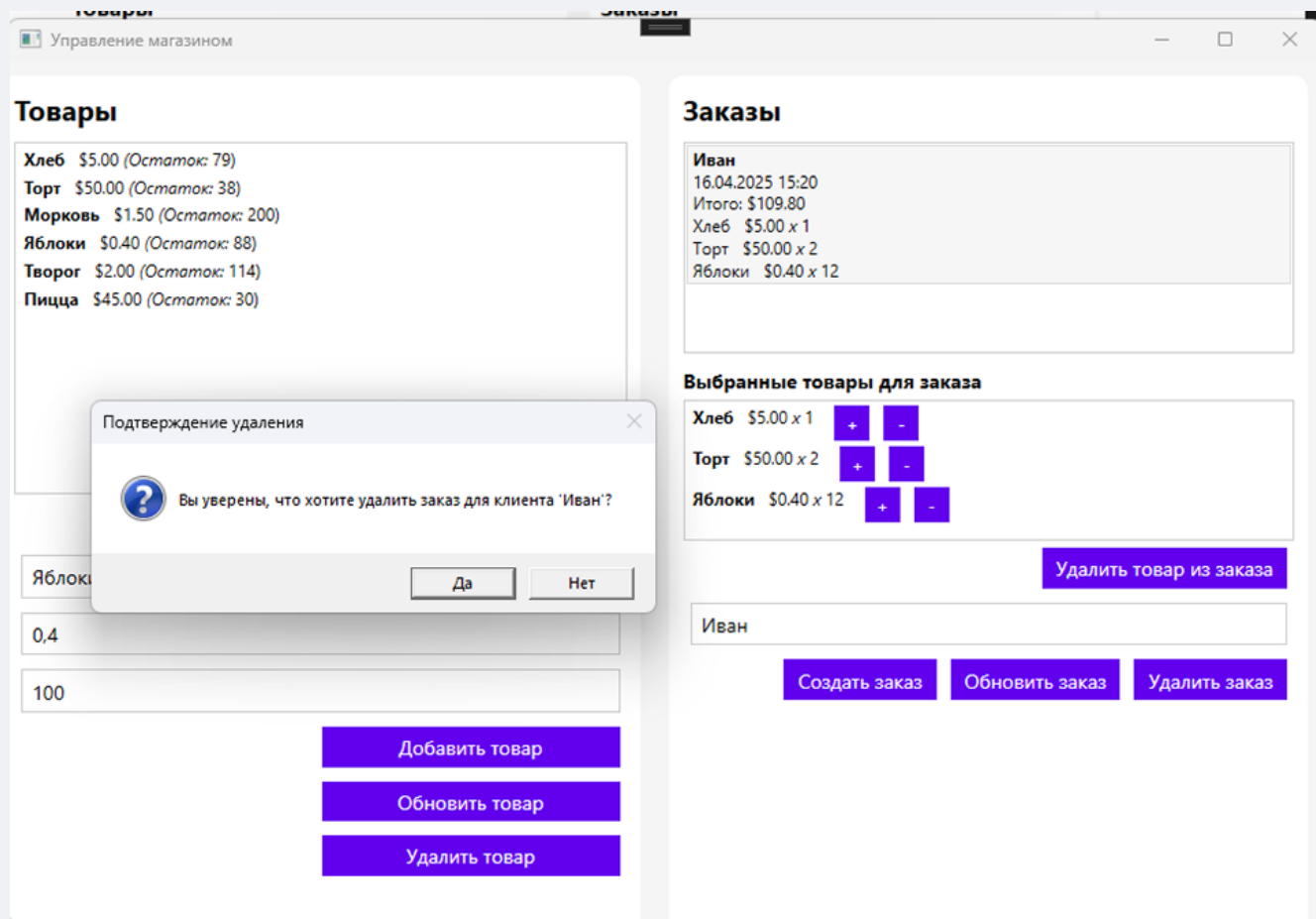


Рис. 12: Удаление товара из заказа

При нажатии кнопки «Удалить товар из заказа» можно удалить конкретный товар добавленный в заказ (Рисунок 13).

The screenshot shows a web application window titled "Управление магазином". It is divided into two main panels: "Товары" (Products) on the left and "Заказы" (Orders) on the right.

**Товары (Products) Panel:**

- A list of products with their names, prices, and stock levels:
  - 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 "Добавить в заказ" (Add to order) button.
- Input fields for "Название товара" (Product name), "Цена" (Price), and "Остаток" (Stock).
- Buttons for "Добавить товар" (Add product), "Обновить товар" (Update product), and "Удалить товар" (Delete product).

**Заказы (Orders) Panel:**

- A large empty box for the order details.
- A section titled "Выбранные товары для заказа" (Selected products for order) with an empty box below it.
- A "Удалить товар из заказа" (Delete product from order) button.
- An input field for "Имя клиента" (Client name).
- Buttons for "Создать заказ" (Create order), "Обновить заказ" (Update order), and "Удалить заказ" (Delete order).

Рис. 13: Вывод списка товаров в заказе



## 5 ВОПРОСЫ ДЛЯ ЗАЩИТЫ

1. Вопрос
2. Вопрос
3. Вопрос
4. Вопрос
5. Вопрос
6. Вопрос
7. Вопрос
8. Вопрос
9. Вопрос
10. Вопрос

Quisque ullamcorper placerat ipsum. Cras nibh. Morbi vel justo vitae lacus tincidunt ultrices. Lorem ipsum dolor sit amet, consectetur adipiscing elit. In hac habitasse platea dictumst. Integer tempus convallis augue. Etiam facilisis. Nunc elementum fermentum wisi. Aenean placerat. Ut imperdiet, enim sed gravida sollicitudin, felis odio placerat quam, ac pulvinar elit purus eget enim. Nunc vitae tortor. Proin tempus nibh sit amet nisl. Vivamus quis tortor vitae risus porta vehicula.

Fusce mauris. Vestibulum luctus nibh at lectus. Sed bibendum, nulla a faucibus semper, leo velit ultricies tellus, ac venenatis arcu wisi vel nisl. Vestibulum diam. Aliquam pellentesque, augue quis sagittis posuere, turpis lacus congue quam, in hendrerit risus eros eget felis. Maecenas eget erat in sapien mattis porttitor. Vestibulum porttitor. Nulla facilisi. Sed a turpis eu lacus commodo facilisis. Morbi fringilla, wisi in dignissim interdum, justo lectus sagittis dui, et vehicula libero dui cursus dui. Mauris tempor ligula sed lacus. Duis cursus enim ut augue. Cras ac magna. Cras nulla. Nulla egestas. Curabitur a leo. Quisque egestas wisi eget nunc. Nam feugiat lacus vel est. Curabitur consectetur.

Suspendisse vel felis. Ut lorem lorem, interdum eu, tincidunt sit amet, laoreet vitae, arcu. Aenean faucibus pede eu ante. Praesent enim elit, rutrum at, molestie non, nonummy vel, nisl. Ut lectus eros, malesuada sit amet, fermentum eu, sodales cursus, magna. Donec eu purus. Quisque vehicula, urna sed ultricies auctor, pede lorem egestas dui, et convallis elit erat sed nulla. Donec luctus. Curabitur et nunc. Aliquam dolor odio, commodo pretium, ultricies non, pharetra in, velit. Integer arcu est, nonummy in, fermentum faucibus, egestas vel, odio.

Sed commodo posuere pede. Mauris ut est. Ut quis purus. Sed ac odio. Sed vehicula hendrerit sem. Duis non odio. Morbi ut dui. Sed accumsan risus eget odio. In hac habitasse platea dictumst. Pellentesque non elit. Fusce sed justo eu urna porta tincidunt. Mauris felis odio, sollicitudin sed, volutpat a, ornare ac, erat. Morbi quis dolor. Donec pellentesque, erat ac sagittis semper, nunc dui lobortis purus, quis congue purus metus ultricies tellus. Proin et quam. Class aptent taciti sociosqu ad litora torquent per conubia nostra, per inceptos hymenaeos. Praesent sapien turpis, fermentum vel, eleifend faucibus, vehicula eu, lacus.