МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования «БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет Информационных технологий

Кафедра Программной инженерии

Специальность 1-40 01 01 Программное обеспечение информационных технологий

Направление специальности 1-40 01 01 10 Программное обеспечение информационных технологий (программирование интернет приложений)

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»

Тема «Интернет-магазин сладостей»

Исполнитель

студент 2 курса группы 4         Кравченко Кирилл Владимирович

(Ф.И.О.)

Руководитель работы            ассистент Северинчик Н.А.

(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой

Председатель Северинчик Н.А. (подпись)

Минск 2021

МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ

Учреждение образования   
«БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНОЛОГИЧЕСКИЙ УНИВЕРСИТЕТ»

Факультет информационных технологий   
Кафедра программной инженерии

Утверждаю

Заведующий кафедрой ПИ

\_\_\_\_\_\_\_\_\_\_\_\_ Н.В. Пацей

подпись инициалы и фамилия

“\_\_\_” \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2021г.

**ЗАДАНИЕ**

**к курсовому проектированию**

**по дисциплине** «Объектно-ориентированное программирование»

|  |  |
| --- | --- |
| Специальность: 1-40 01 01 Программное обеспечение информационных технологий Группа: 4  Студент: Кравченко К.В. |  |
| **Тема:** Программное средство «Интернет-магазин сладостей» | |

**1. Срок сдачи студентом законченной работы**: "29 мая 2021 г."

**2. Исходные данные к проекту:**

**2.1**. Функционально ПС поддерживает:

* Функции администратора:
  + Добавлять товары;
  + Отслеживать товары в магазине (их название, стоимость, количество);
  + Осуществлять сортировку товаров (по названию, стоимости (поддерживается поиск по диапазону) и количеству);
  + Удалять товары;
  + Отслеживать заказы;
  + Поддерживать работу c базой данных.
* Функции клиента:
  + Выполнять регистрацию и авторизацию (с помощью имя пользователя и пароля);
  + Отслеживать товары в магазине (их название, стоимость, количество, изображение);
  + Осуществлять покупку товара;
  + Отслеживать и сортировать купленные товары в корзине (по названию, стоимости и количеству).

**2.2.** При выполнении курсового проекта необходимо использовать принципы проектирования ООП. Приложение разрабатывается под ОС Windows и представляет собой настольное приложение (desktop). Отображение, бизнес-логика должны быть максимально независимы друг от друга для возможности расширения. Диаграммы вариантов использования, классов реализации задачи, взаимодействия разработать на основе UML. Язык разработки проекта – C#. Управление программой должно быть интуитивно понятным и удобным. При разработке использовать несколько наиболее подходящих шаблонов проектирования ПО.

**3. Содержание расчетно-пояснительной записки**

(перечень вопросов, подлежащих разработке)

* Введение
* Постановка задачи и обзор литературы (алгоритмы решения, обзор прототипов, актуальность задачи)
* Проектирование архитектуры проекта (структура модулей, классов)
* Разработка функциональной модели и модели данных ПС (выполняемые функции)
* Тестирование
* Заключение
* Список используемых источников
* Приложения

**4. Форма представления выполненной курсовой работы:**

* + Теоретическая часть курсового проекта должны быть представлены в формате docx. Оформление записки должно быть согласно выданным правилам.
  + Листинги программы представляются частично в приложении.
  + Пояснительную записку, листинги, проект (инсталляцию проекта) необходимо загрузить на диск, указанный преподавателем.

#### **Календарный план**

|  |  |  |  |
| --- | --- | --- | --- |
| №  п/п | Наименование этапов курсового проекта | Срок выполнения этапов проекта | Примечание |
| 1 | Введение | 19.02.2021 |  |
| 2 | Аналитический обзор литературы по теме проекта. Изучение требований, определение вариантов использования | 12.03.2021 |  |
| 3 | Анализ и проектирование архитектуры приложения (построение диаграмм, проектирование бизнес-слоя, представления и данных) | 26.03.2021 |  |
| 4 | Проектирование структуры базы данных. Разработка дизайна пользовательского интерфейса | 02.04.2021 |  |
| 5 | Кодирование программного средства | 23.04.2021 |  |
| 6 | Тестирования и отладка программного средства | 30.04.2021 |  |
| 7 | Оформление пояснительной записки | 07.05.2021 |  |
| 8 | Сдача проекта | 29.05.2021 |  |

**5. Дата выдачи задания** 21.02.2021

Руководитель\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ *Н.А. Северинчик*

(подпись)

Задание принял к исполнению

(дата и подпись студента)

Содержание

[Введение 6](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886572)

[1.Аналитический обзор литературы 7](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886577)

[2.Анализ требований к программному средству и разработка функциональных требований 9](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886580)

[2.1 Описание функций приложения 9](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886581)

[2.2 Средства разработки 9](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886582)

[3. Проектирование программного средства 10](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886585)

3[.1 Обобщенная структура 10](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886586)

[3.2 Диаграммы UML 11](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886587)

[3.2.1Диаграмма вариантов использования 11](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886588)

[3.2.1 Диаграмма классов 13](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886588)

[3.3 Проектирование базы данных 16](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886589)

[3.4 MVVM 18](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886589)

[4. Создание (реализация) программного средства 20](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886596)

4[.1 Основные окна 20](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886586)

[4.1.1Окно авторизвции 20](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886588)

[4.1.2 Окно регистрации 21](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886588)

[4.1.3 Окно админа 21](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886588)

[4.1.4 Окно добавления товара 22](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886588)

[4.1.5 Окно наблюдения за заказами 22](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886588)

[4.1.6 Главное окно магазина 23](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886588)

[4.1.7 Окно покупки 23](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886588)

[4.1.8 Окно корзины 24](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886588)

4[.2 Разработка программы 25](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886586)

[5. Тестирование, проверка работоспособности и анализ полученных результатов 26](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886597)

5[.1 Тест авторизации 26](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886586)

5[.2 Тест регистрации 26](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886586)

5[.3 Тест добавления товара 28](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886586)

5[.4 Тест покупки товара 30](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886586)

5[.5 Тест наблюдения за заказами 31](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886586)

5[.6 Тест сортировки товаров 31](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886586)

[6. Руководство по установке и использования 32](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886598)

[Заключение 34](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886599)

[Список используемых источников 35](file:///C:\Users\Asus\Downloads\Poyasnitelnaya_Zapiska_Kravchenko_N_V__kopia.docx#_Toc72886600)

Приложение А…………………………………………………………......36

Приложение Б……………………………...……………………………... 41

Приложение В……………………………………………………………..42

Приложение Г……………………………………………………………..47

ВВЕДЕНИЕ

Мы живем в эпоху современных технологий и интернета. С каждым днем покупка товаров в интернете (не выходя из дома) выходит на новый уровень. Поэтому реализация программного средства “Интернет-магазин” будет как нельзя кстати.

Мое приложение будет процессом организации структуры управления интернет-магазином на основе администратора и пользователя.

Приложение позволяет администратору отслеживать товары, заказы, управлять ими.

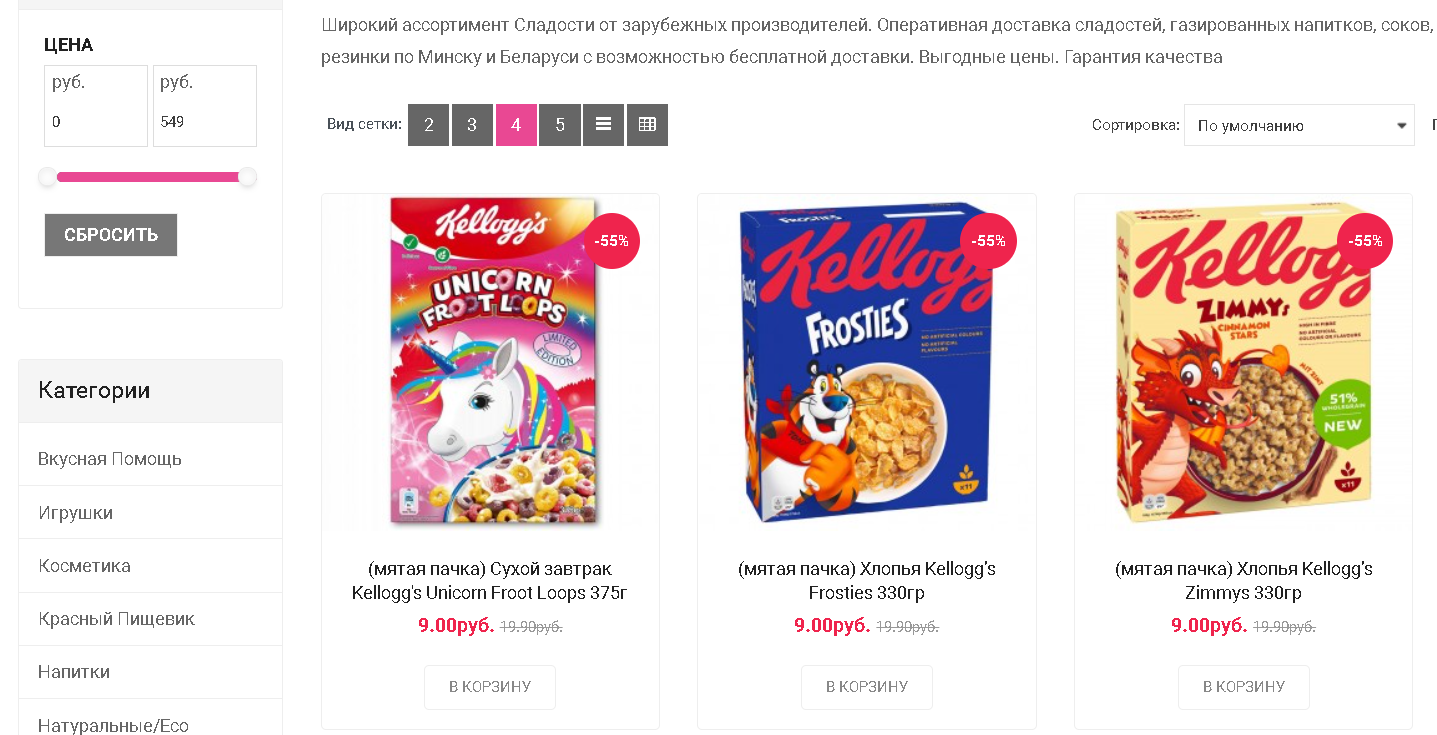
Пользователю же позволяется отслеживать товары в магазине, осуществлять их покупку.

Целью курсового проекта является: развитие практических навыков верстки приложений на базе WPF с использованием языка разметки — XAML, и языка программирования — С#, показать навыки разработки дизайна, макета для приложения и умение грамотно разделить информацию.

# Аналитический обзор литературы

На основе темы своего курсового проекта, я провел анализ достоинств и недостатков известных решений, схожих с задачей моего приложения.

Я рассмотрел популярные реализации интернет-магазинов, их интерфейс и принцип работы функционала.



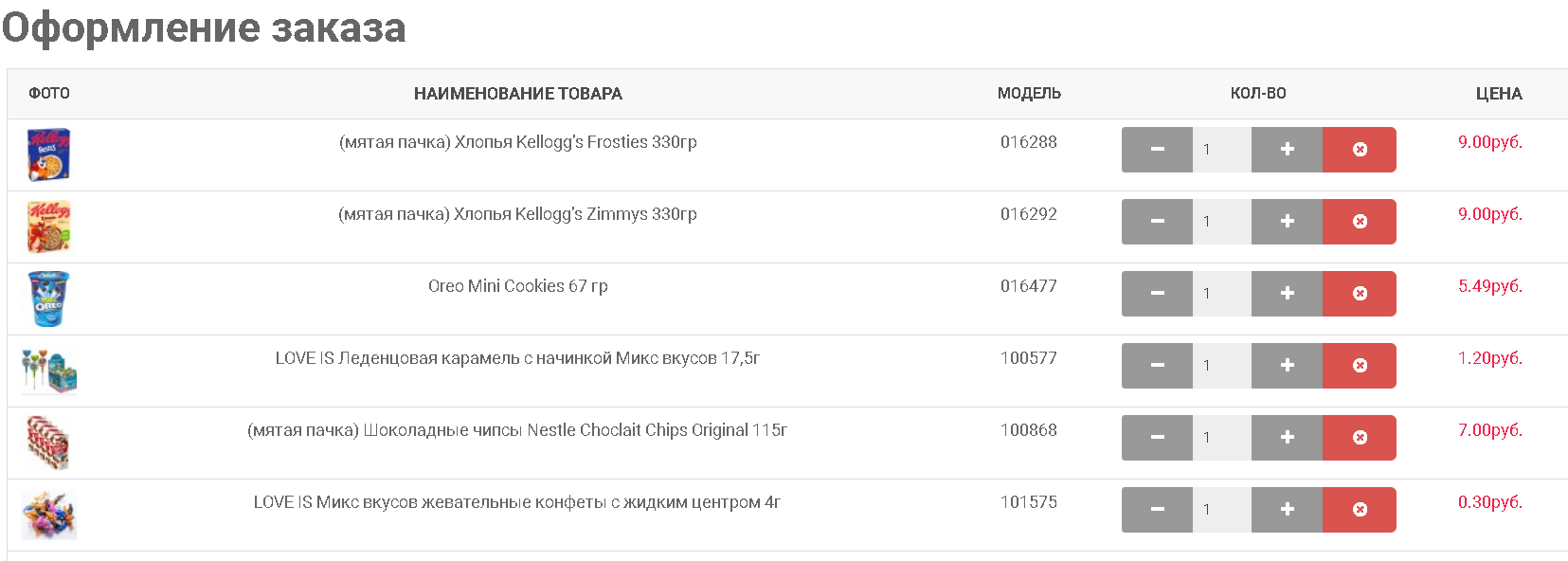
Рисунок 1.1 – Пример окна каталога

Рисунок 1.2 – Пример окна оформления заказа

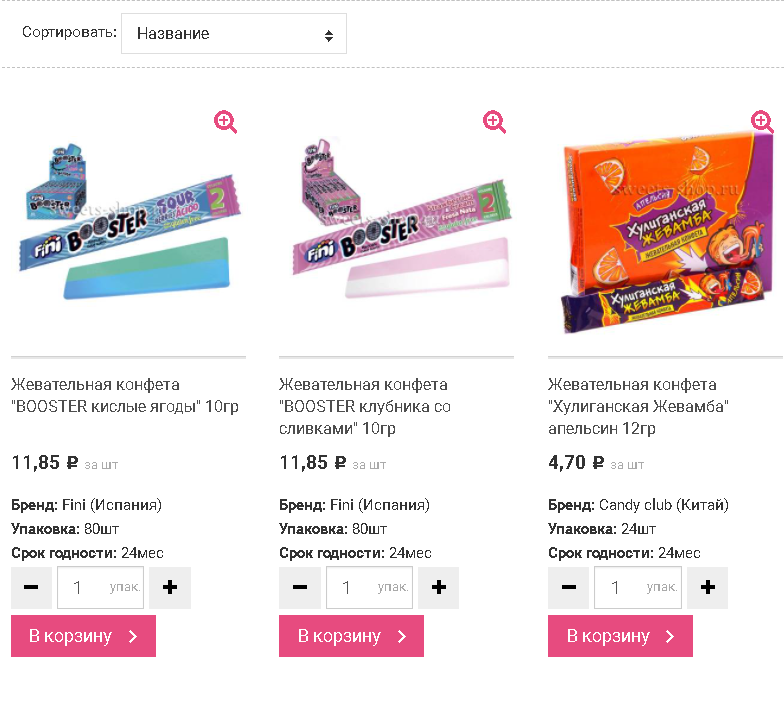


Рисунок 1.3 – Пример окна каталога

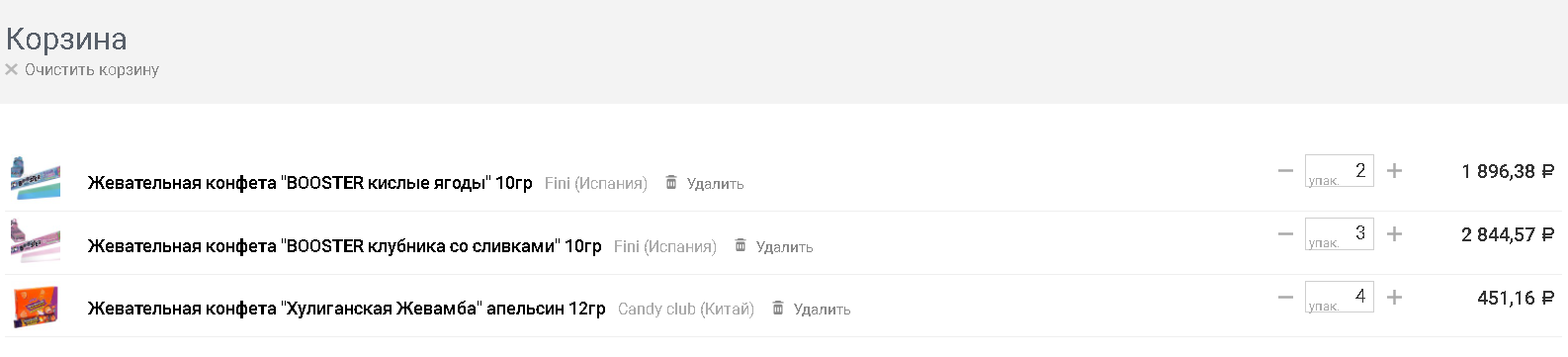


Рисунок 1.4 – Пример окна корзины покупок

На основе проведенного анализа и с учетом требований, указанных в задании на курсовое проектирование, я сформулировал требования (техническое задание) к проектируемому программному средству

# 2. Анализ требований к программному средству и разработка функциональных требований

**2.1 Описание функций приложения**

Основной задачей курсового проекта является разработка приложения с возможностью управлять товарами и их движением, заказами на их покупку. Это поможет пользователям сделать многие из этих процессов автоматическими.

Функционально данное программное средство должно выполнять следующие задачи:

* выполнять регистрацию и авторизацию;
* поддерживать работу с базой данных;
* добавлять товары;
* отслеживать и управлять товарами в магазине;
* удалять товары;
* обновлять товары и их данные;
* осуществлять покупку товара
* отслеживать заказы

## 2.2 Средства разработки

При разработке приложения были использованы:

* среда разработки Microsoft Visual Studio 2019;
* технология WPF;
* объектно-ориентированная технология Entity Framework на базе фреймворка .NET;
* система управления Microsoft SQL Server.

Microsoft Visual Studio — это стартовая площадка для написания, отладки и сборки кода, а также последующей публикации приложений. Интегрированная среда разработки (IDE) представляет собой многофункциональную программу, которую можно использовать для различных аспектов разработки программного обеспечения. Помимо стандартного редактора и отладчика, которые существуют в большинстве сред IDE, Visual Studio включает в себя компиляторы, средства автозавершения кода, графические конструкторы и многие другие функции для упрощения процесса разработки.

Windows Presentation Foundation (WPF) — это система следующего поколения для построения клиентских приложений Windows с визуально привлекательными возможностями взаимодействия с пользователем. C WPF можно создавать широкий спектр как автономных приложений, так и приложений, размещенных в веб-обозревателе. в основе WPF лежит векторная система визуализации, не зависящая от разрешения и созданная с расчетом на возможности современного графического оборудования. WPF расширяет базовую систему полным набором функций разработки приложений, элементами управления, привязкой данных, макетом, графикой, анимацией, стилями, шаблонами, документами, мультимедиа, текстом и оформлением. Именно использование WPF позволило гибко управлять дизайном интерфейса, также стало возможным подключение различных сторонних пакетов и использование паттернов.

Entity Framework представляет специальную объектно-ориентированную технологию на базе фреймворка .NET для работы с данными. Если традиционные средства ADO.NET позволяют создавать подключения, команды и прочие объекты для взаимодействия с базами данных, то Entity Framework представляет собой более высокий уровень абстракции, который позволяет абстрагироваться от самой базы данных и работать с данными независимо от типа хранилища. Если на физическом уровне мы оперируем таблицами, индексами, первичными и внешними ключами, но на концептуальном уровне, который нам предлагает Entity Framework, мы уже работает с объектами.

Microsoft SQL Server — система управления реляционными БД. SQL Server Management Studio – утилита из Microsoft SQL Server, необходимая для управления и администрирования всех компонентов Microsoft SQL Server. Она в себя включает скриптовый редактор и графическую программу, которая работает с объектами и настройками сервера. SQL является, прежде всего, информационно-логическим языком, предназначенным для описания, изменения и извлечения данных, хранимых в реляционных базах данных.

# 3. Проектирование программного средства

**3.1 Обобщенная структура**

Решение представлено проектом «OnlineStore» с имеющей структурой (рисунок 3.1).

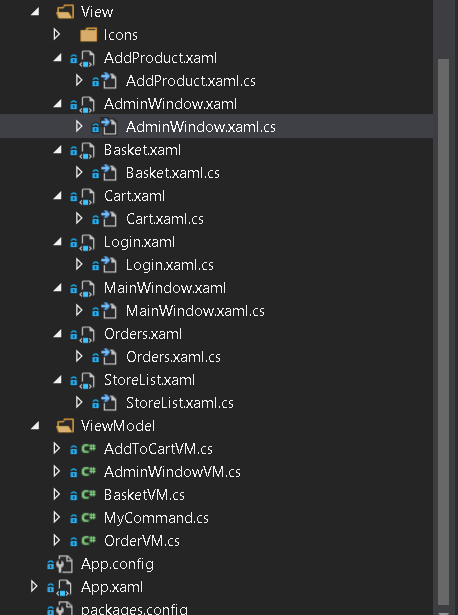
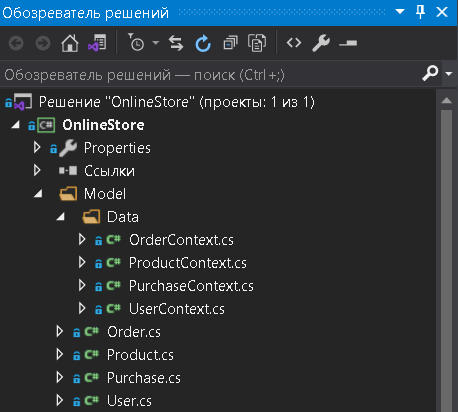


Рисунок 3.1 – Обобщенная структура проекта

Описание обобщенной структуры проекта «OnlineStore» представлено в таблице 3.1.

Таблица 3.1 – Обобщенная структура проекта

|  |  |
| --- | --- |
| Объект | Описание |
| Properties | Cвойства проекта (содержит информацию о сборке, используемых ресурсах и настройках) |
| Ссылки | Перечень сборок, используемых в проекте |
| Папка Model | Описывает используемые в приложении данные |
| Папка View | Определяет визуальный интерфейс, через который пользователь взаимодействует с приложением |
| Папка Icons | Cодержит основные картинки проекта |
| Папка ViewModel | Связывает модель и представление, определяет логику по обновлению данных в модели |
| App.config | Файл с параметрами проекта |
| App.xaml | Класс Application |
| packages.config | Файл для поддержки списка пакетов, на которые ссылается проект |

## 3.2 Диаграммы UML

UML — унифицированный язык моделирования (Unified Modeling Language) — язык графического описания для объектного моделирования в области разработки программного обеспечения, для моделирования системного проектирования и отображения организационных структур.

UML является языком широкого профиля, это — [открытый стандарт](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D1%8B%D0%B9_%D1%81%D1%82%D0%B0%D0%BD%D0%B4%D0%B0%D1%80%D1%82), использующий графические обозначения для создания [абстрактной модели](https://ru.wikipedia.org/wiki/%D0%90%D0%B1%D1%81%D1%82%D1%80%D0%B0%D0%BA%D1%82%D0%BD%D0%B0%D1%8F_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D1%8C) [системы](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D1%81%D1%82%D0%B5%D0%BC%D0%B0), называемой UML-моделью. UML был создан для определения, визуализации, проектирования и документирования, в основном, программных систем.

**3.2.1 Диаграмма вариантов использования**

В данной программе существует две разновидности ролей пользователя. На диаграмме вариантов использования, представлены роли администратора и пользователя (рисунок 3.2).

Администратору предоставляется:

* + Добавлять товары;
  + Отслеживать товары в магазине (их название, стоимость, количество);
  + Осуществлять сортировку товаров (по названию, стоимости (поддерживается поиск по диапазону) и количеству);
  + Удалять товары;
  + Отслеживать заказы;
  + Поддерживать работу c базой данных.

Пользователю предоставляется:

* + Выполнять регистрацию и авторизацию (с помощью имя пользователя и пароля);
  + Отслеживать товары в магазине (их название, стоимость, количество, изображение);
  + Осуществлять покупку товара;
  + Отслеживать и сортировать купленные товары в корзине (по названию, стоимости и количеству).

Добавление товаров

Отслеживание товаров в магазине

Осуществление сортировки товаров

Удаление товаров

Отслеживание заказов

Поддержка работы c базой данных

Выполнение регистрации и авторизации

Отслеживание товаров в магазине

Осуществление покупки товара

Отслеживание и сортировка купленных товаров в корзине

Пользователь

Администратор

Рисунок 3.2 – Диаграмма вариантов использования

## 3.2.2 Диаграмма классов

Структурная диаграмма языка моделирования UML, демонстрирующая общую структуру иерархии классов системы, их коопераций, атрибутов, методов, интерфейсов и взаимосвязей между ними. Широко применяется не только для документирования и визуализации, но также для конструирования посредством прямого или обратного проектирования.

На данной диаграмме классов изображены классы программы, методы, поля и свойства классов. Диаграмма классов служит для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру (поля, методы) и типы отношений (наследование, реализация интерфейсов).

Диаграмма классов моделей (описывает используемые в приложении данные) представлена на рисунке 3.3.

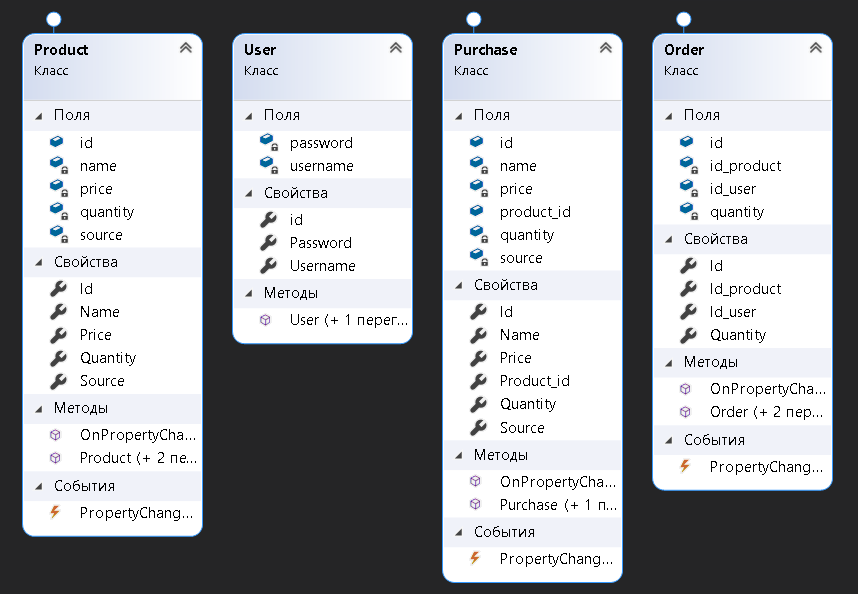


Рисунок 3.3 – Диаграмма классов моделей

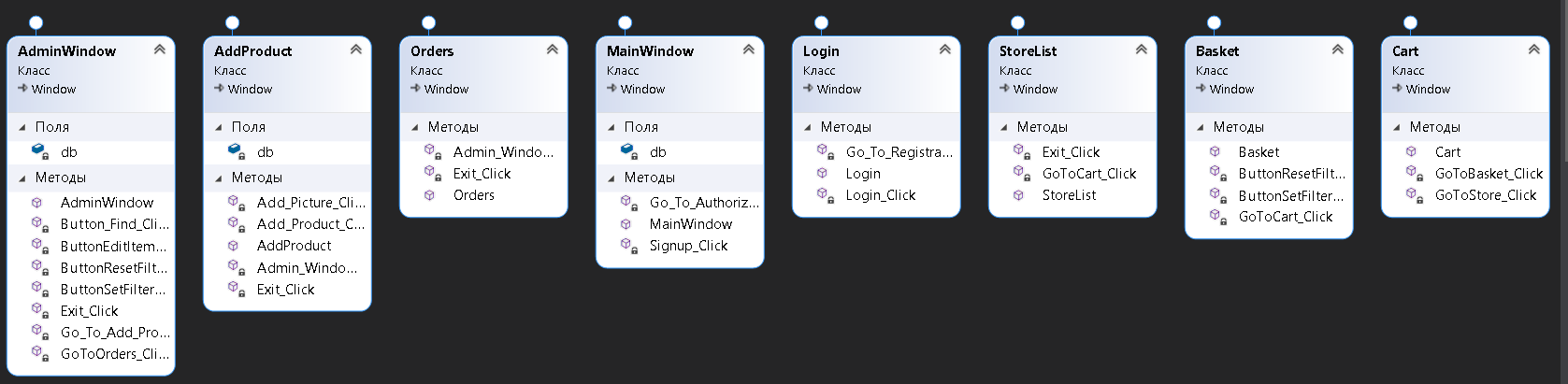
Диаграмма классов представления (определяют визуальный интерфейс) представлена на рисунке 3.4.

Рисунок 3.4 – Диаграмма классов представления

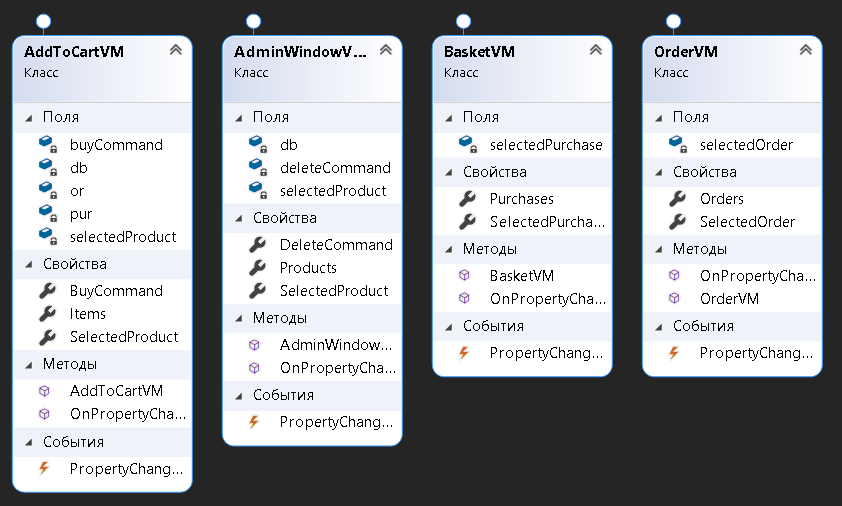
Диаграмма классов моделей представления (связывает модель и представление через механизм привязки данных) представлена на рисунке 3.5.

Рисунок 3.5 – Диаграмма классов модели представления

Диаграмма класса привязки команды с представлена на рисунке 3.6.

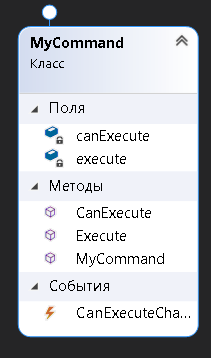


Рисунок 3.6 – Диаграмма класса привязки команд

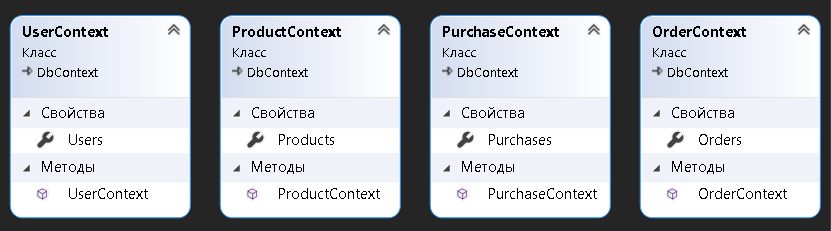
Диаграмма классов контекста базы данных представлена на рисунке 3.7.

Рисунок 3.7 – Диаграмма классов контекста базы данных

Диаграмма классов ресурсов и настроек представлена на рисунке 3.8.

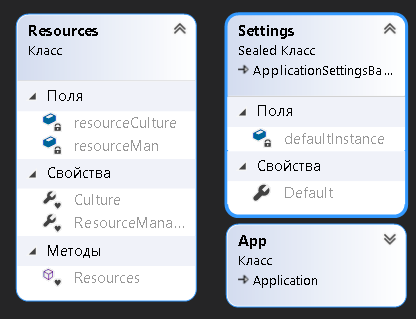


Рисунок 3.8 – Диаграмма классов ресурсов и настроек

## 3.3 Проектирование базы данных

Данные для работы будут храниться в базе данных. Ее создание происходит с помощью Microsoft SQL Server — это система управления реляционными базами данных, которая подключается к приложению.

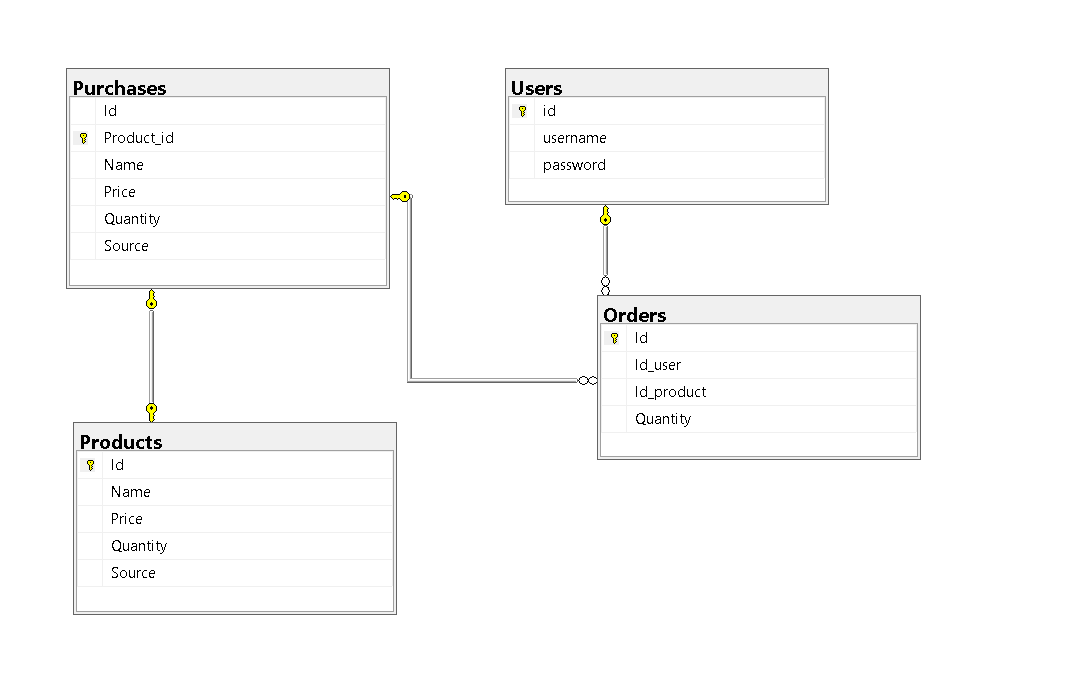
При разработке приложения сначала разрабатывалась база данных, которая должна содержать всю нужную информацию для приложения, после программное обеспечение. База данных состоит из 4 таблиц, между которыми установлены связи один ко многим (рисунок 3.9).

Рисунок 3.9 – Диаграмма базы данных

Таблица Users (рисунок 3.10) предназначена для хранения информации о пользователях, состоит из 3 столбцов:

­– id - уникальный идентификатор пользователя;

– username – имя пользователя;

– password - пароль пользователя.

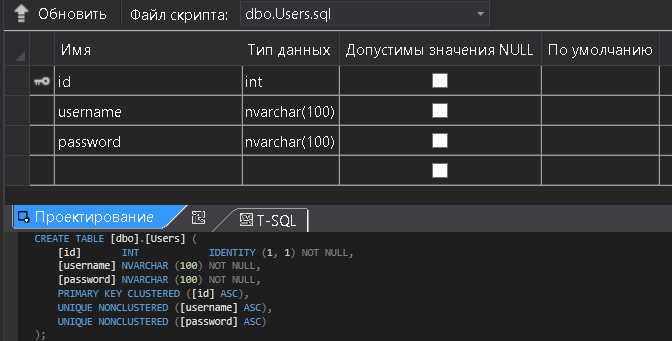


Рисунок 3.10 – Таблица Users

Таблица Products (рисунок 3.11) предназначена для хранения информации о добавленных товарах, состоит из 5 столбцов:

­– Id - уникальный идентификатор товара;

– Name - название товара;

– Price - цена товара;

– Quantity - количество товара;

– Source – изображение товара;

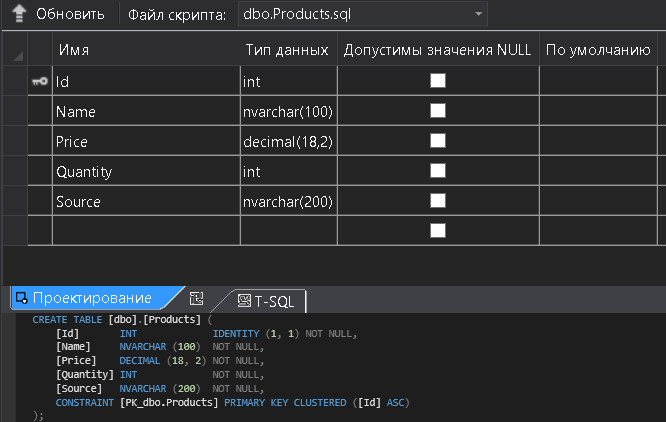


Рисунок 3.11 – Таблица Products

Таблица Purchases (рисунок 3.12) предназначена для хранения информации о покупках, состоит из 6 столбцов:

­– Id - уникальный идентификатор покупки;

– Product\_id - уникальный идентификатор товара;

– Name - название товара;

– Price - цена товара;

– Quantity - количество товара;

– Source – изображение товара;

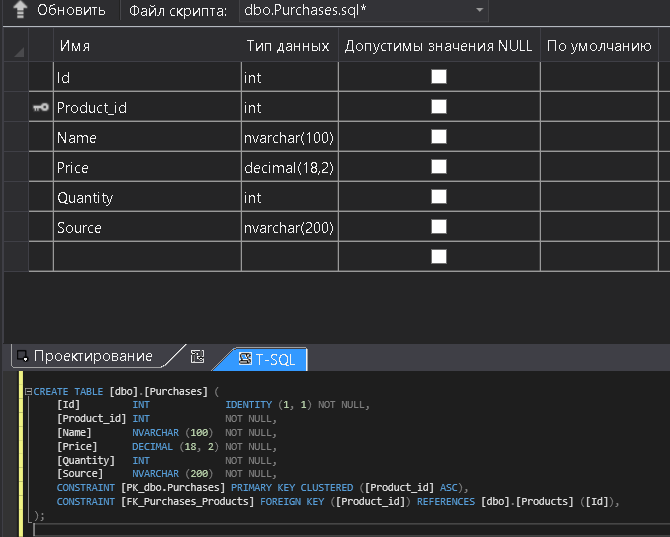


Рисунок 3.12 – Таблица Purchases

Таблица Orders (рисунок 3.13) предназначена для хранения информации о заказах, состоит из 4 столбцов:

­– Id - уникальный идентификатор заказа;

– Id\_user - уникальный идентификатор пользователя;

– Id\_product - уникальный идентификатор товара;

– Quantity - количество товара;

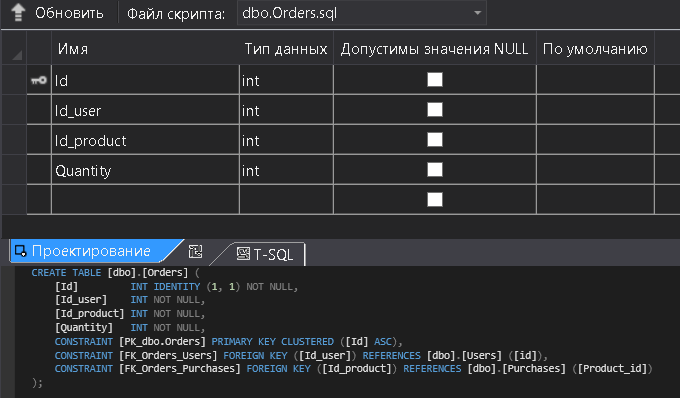


Рисунок 3.13 – Таблица Orders

Также в каждой таблице для поля Id прописан атрибут Identity (1, 1), который позволяет сделать столбец идентификатором, значение которого будет автоматически увеличиваться на единицу для каждой следующей строки.

## 3.4 MVVM

В процессе создания приложения был реализован паттерн MVVM. Паттерн MVVM (Model-View-ViewModel) позволяет отделить логику приложения от визуальной части (представления). Данный паттерн является архитектурным, то есть он задает общую архитектуру приложения.

MVVM состоит из трех компонентов: модели (Model), модели представления (ViewModel) и представления (View).

Паттерн MVVM в WPF:



Рисунок 3.14 – схема паттерна MVVM

**Model**

Модель описывает используемые в приложении данные. Модели могут содержать логику, непосредственно связанную этими данными, например, логику валидации свойств модели. В то же время модель не должна содержать никакой логики, связанной с отображением данных и взаимодействием с визуальными элементами управления.

**View**

View или представление определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Применительно к WPF представление – это код в xaml, который определяет интерфейс в виде кнопок, текстовых полей и прочих визуальных элементов.

Хотя окно (класс Window) в WPF может содержать как интерфейс в xaml, так и привязанный к нему код C#, однако в идеале код C# не должен содержать какой-то логики, кроме разве что конструктора, который вызывает метод InitializeComponent и выполняет начальную инициализацию окна. Вся же основная логика приложения выносится в компонент ViewModel.

Однако иногда в файле связанного кода все может находиться некоторая логика, которую трудно реализовать в рамках паттерна MVVM во ViewModel.

Представление не обрабатывает события за редким исключением, а выполняет действия в основном посредством команд.

**ViewModel**

ViewModel или модель представления связывает модель и представление через механизм привязки данных. Если в модели изменяются значения свойств, при реализации моделью интерфейса INotifyPropertyChanged автоматически идет изменение отображаемых данных в представлении, хотя напрямую модель и представление не связаны.

ViewModel также содержит логику по получению данных из модели, которые потом передаются в представление. И также VewModel определяет логику по обновлению данных в модели.

Поскольку элементы представления, то есть визуальные компоненты типа кнопок, не используют события, то представление взаимодействует с ViewModel посредством команд.

Например, пользователь хочет сохранить введенные в текстовое поле данные. Он нажимает на кнопку и тем самым отправляет команду во ViewModel. А ViewModel уже получает переданные данные и в соответствии с ними обновляет модель.

Итогом применения паттерна MVVM является функциональное разделение приложения на три компонента, которые проще разрабатывать и тестировать, а также в дальнейшем модифицировать и поддерживать.

**4. Создание (реализация) программного средства**

**4.1 Основные окна**

**4.1.1 Окно авторизации**

Окно авторизации содержит поля *ИМЯ ПОЛЬЗОВАТЕЛЯ* и *ПАРОЛЬ*. Вход не осуществиться до тех пор, пока не будут введены корректные данные в поля.

Если при авторизации введены данные админа, то мы попадаем в окно админа AdminWindow (рисунок 4.3).

Иначе же мы можем перейти в окно регистрации пользователя MainWindow (рисунок 4.2).

После успешной регистрации мы можем ввести данные и перейти в главное окно магазина StoreList (рисунок 4.6).

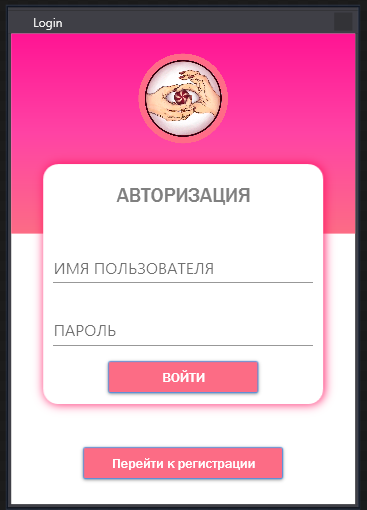
****

Рисунок 4.1 – Окно авторизации Login

**4.1.2 Окно регистрации**

Окно регистрации содержит поля *ИМЯ ПОЛЬЗОВАТЕЛЯ*, *ПАРОЛЬ* и *ПОДТВЕРЖДЕНИЕ ПАРОЛЯ*. Регистрация не осуществиться до тех пор, пока не будут введены корректные данные в поля.

После успешной регистрации нас автоматически перенаправляет в окно авторизации Login (рисунок 4.1).

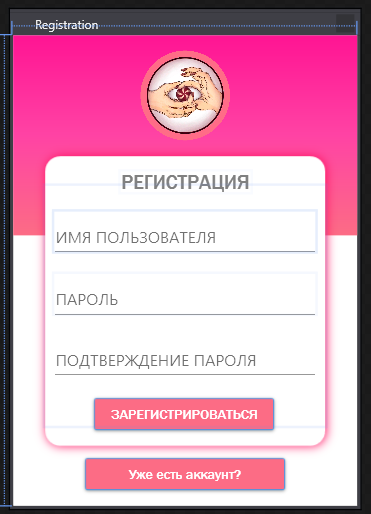
****

Рисунок 4.2 – Окно регистрации MainWindow

**4.1.3 Окно админа**

В окне админа мы можем наблюдать за товарами в нашем магазине, сортировать их, удалять.

Из окна админа мы можем перейти в окно добавления товара AddProduct (рисунок 4.4).

Так же мы можем перейти в окно наблюдения за заказами Orders (рисунок 4.5) и выйти в окно авторизации Login (рисунок 4.1).

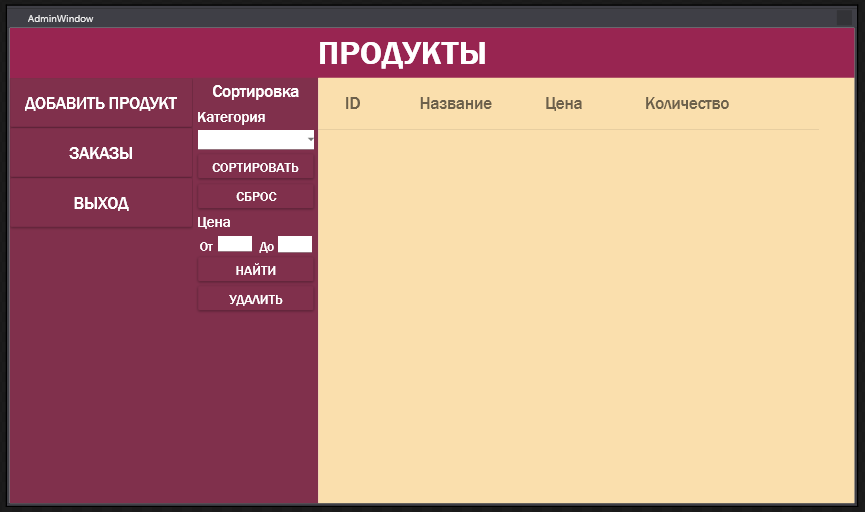
****

Рисунок 4.3 – Окно админа AdminWindow

**4.1.4 Окно добавления товара**

В окне добавления товара мы можем добавлять товары в наш магазин.

Окно добавления товара содержит поля *НАЗВАНИЕ ПРОДУКТА*, *ЦЕНА* ПРОДУКТА, КОЛИЧЕСТВО ПРОДУКТАикнопку добавления *ИЗОБРАЖЕНИЯ ПРОДУКТА*. Добавление товара не осуществиться до тех пор, пока не будут введены корректные данные в поля.

Из окна добавления товара мы можем перейти в окно админа AdminWindow (рисунок 4.3).

Так же мы можем выйти в окно авторизации Login (рисунок 4.1).

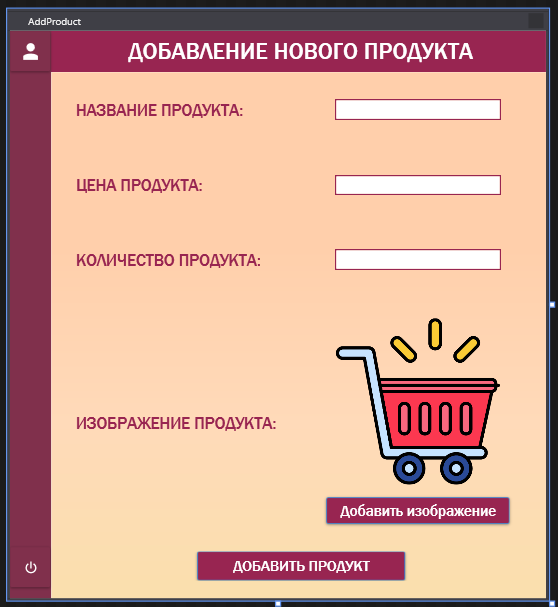


Рисунок 4.4 – Окно добавления товара AddProduct

**4.1.5 Окно наблюдения за заказами**

В окне наблюдения за заказами мы можем наблюдать за заказами, которые были осуществлены.

Из окна наблюдения за заказами мы можем перейти в окно админа AdminWindow (рисунок 4.3).

Так же мы можем выйти в окно авторизации Login (рисунок 4.1).

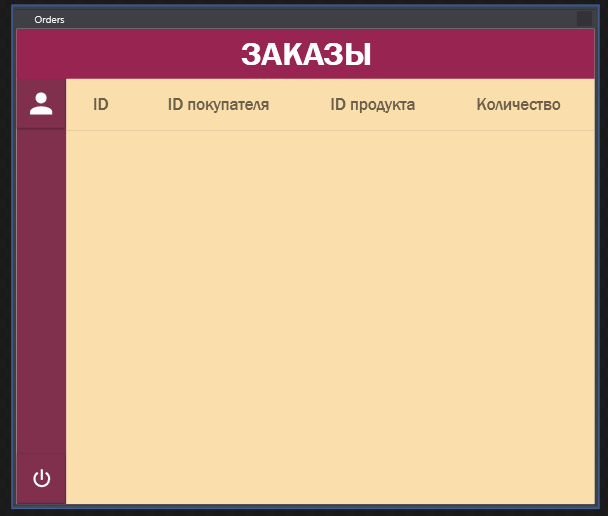
****

Рисунок 4.5 – Окно наблюдения за заказами Orders

**4.1.6 Главное окно магазина**

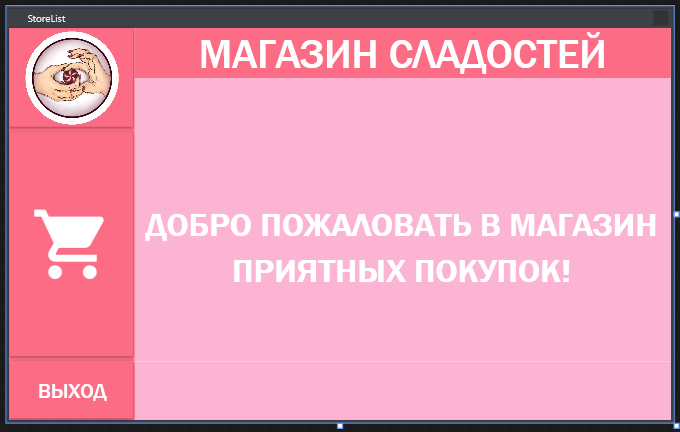
Из главного окна магазина мы можем перейти в окно покупки Cart (рисунок 4.7) или же выйти в окно авторизации Login (рисунок 4.1).

Рисунок 4.6 – Главное окно магазина StoreList

**4.1.7 Окно покупки**

В окне покупки мы можем осуществлять покупку товаров, которые представлены.

После этого мы можем перейти в окно корзины Basket (рисунок 4.8) или же перейти в главное окно магазина StoreList (рисунок 4.6).

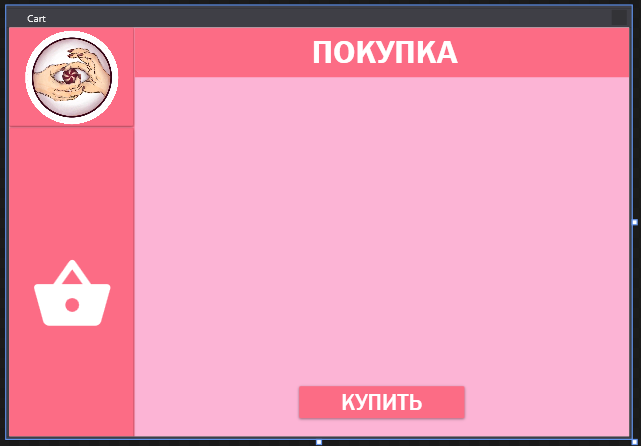
****

Рисунок 4.7 – Окно покупки Cart

**4.1.8 Окно корзины**

В окне корзины мы можем наблюдать за товарами, которые мы приобрели в магазине, а также сортировать их.

Из окна корзины мы можем перейти в окно покупки Cart (рисунок 4.7)

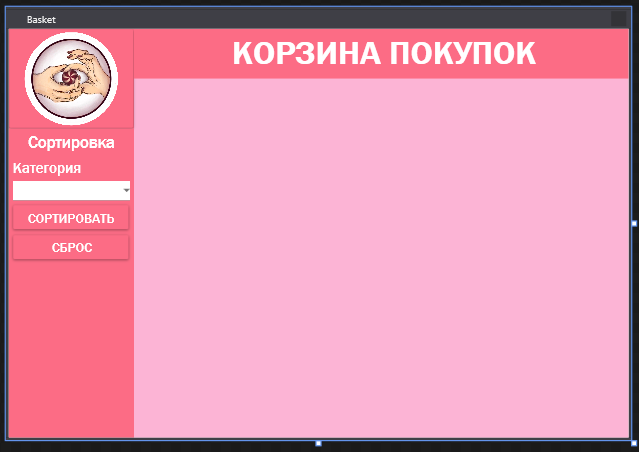
****

Рисунок 4.8 – Окно корзины Basket

**4.2 Разработка программы**

Окна представления содержат как интерфейс в xaml, так и небольшой кусок привязанного к нему кода C#.

Окна AdminWindow, Cart, Orders, Basket связываются с соответствующими классами модели представления AdminWindowVM, AddToCartVM, BasketVM, OrderVM, которые в свою очередь реализует интерфейс *INotifyPropertyChanged*, для того чтобы объект мог полноценно реализовывать механизм привязки. Когда объект класса изменяет значение свойства, то он через событие PropertyChanged извещает систему об изменениях свойства и система обновляет все привязанные объекты.

Код классов AdminWindowVM, AddToCartVM, BasketVM, OrderVM представлен в **приложении А**.

В окнах AdminWindowVM и AddToCartVM для взаимодействия пользователя и приложения используются команды. В WPF команды представлены интерфейсом ICommand.

Для использования команд мы добавили в наше приложение класс MyCommand. Класс реализует два метода:

* CanExecute – определяет, может ли командна выполняться;
* Execute – собственно выполняет логику команды.

Код класса MyCommand представлен в **приложении Б**.

Классы Product, User, Purchase, Order являются классами моделями (описывают используемые в приложении данные).

Код классов Product, User, Purchase, Order представлен в **приложении В**.

Классы ProductContext, UserContext, PurchaseContext, OrderContext представляют собой контекст базы данных.

Код классов ProductContext, UserContext, PurchaseContext, OrderContext представлен в **приложении Г**.

**5. Тестирование, проверка работоспособности и анализ полученных результатов**

**5.1 Тест авторизации**

Ошибка авторизации, если введены данные пользователя, которого не существует (рисунок 5.1).

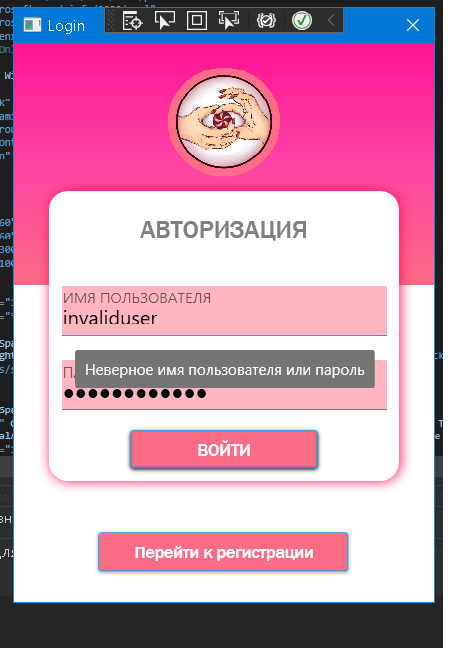
****

Рисунок 5.1 – Тест авторизации

**5.2 Тест регистрации**

Ошибка регистрации, если введенное поле *ИМЯ ПОЛЬЗОВАТЕЛЯ* содержит меньше 4 символов (рисунок 5.2).

Ошибка регистрации, если введенное поле *ПАРОЛЬ* содержит меньше 4 символов (рисунок 5.3).

Ошибка регистрации, если введенное поле *ПАРОЛЬ* не соответствует полю *ПОДТВЕРЖДЕНИЕ* *ПАРОЛЯ* (рисунок 5.4).

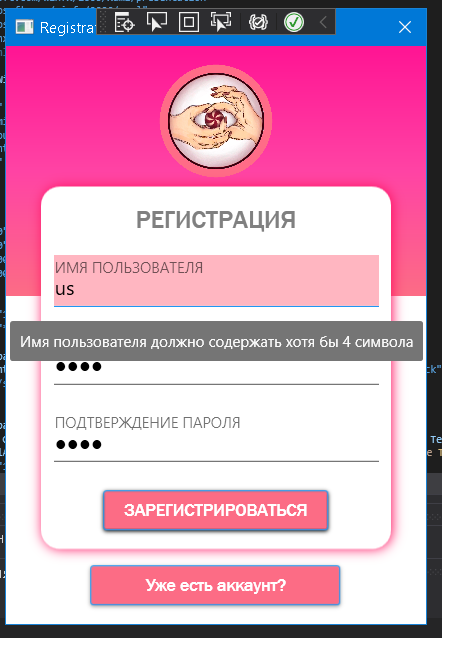
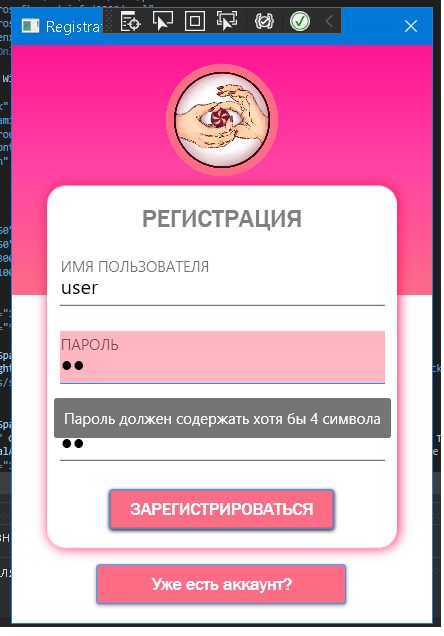


Рисунок 5.2 – Тест регистрации Рисунок 5.3 – Тест регистрации

на пароль на имя пользователя

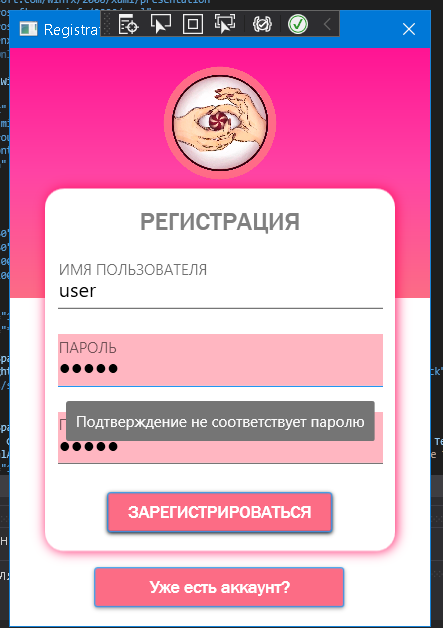


Рисунок 5.4 – Тест регистрации

на совпадение паролей

**5.3 Тест добавления товара**

Ошибка добавления товара, если существуют пустые поля (рисунок 5.5).

Ошибка добавления товара, если поля содержат 0 (рисунок 5.6).

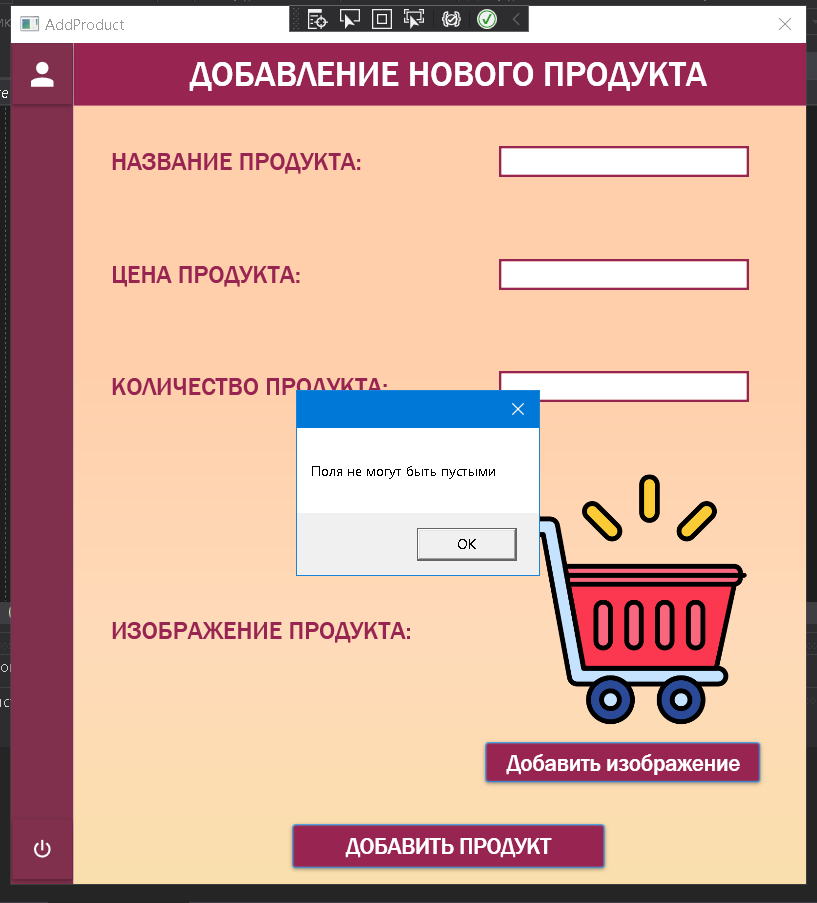
****

Рисунок 5.5 – Тест добавления товара на пустые поля

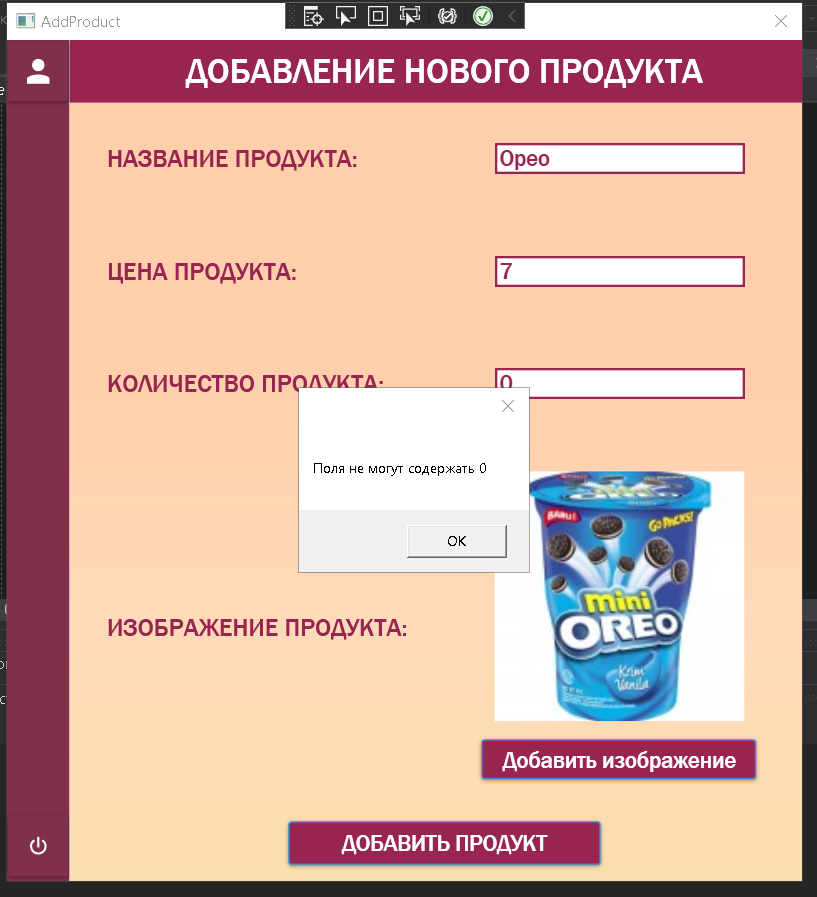
****

Рисунок 5.6 – Тест добавления товара на поля,содержащие 0

Ошибка добавления товара, если поле *ЦЕНА ПРОДУКТА* содержит что-нибудь кроме цифр и запятой (рисунок 5.7).

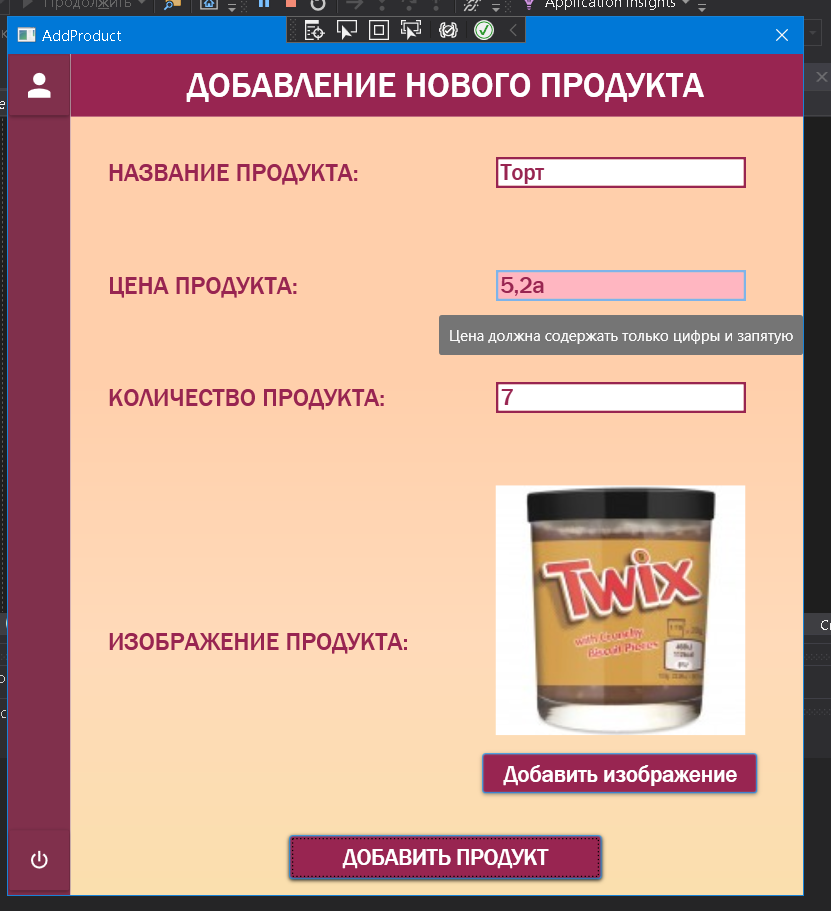


Рисунок 5.7 – Тест добавления товара на буквы в поле цена

Ошибка добавления товара, если поле *КОЛИЧЕСТВО ПРОДУКТА* содержит что-нибудь кроме цифр (рисунок 5.8).

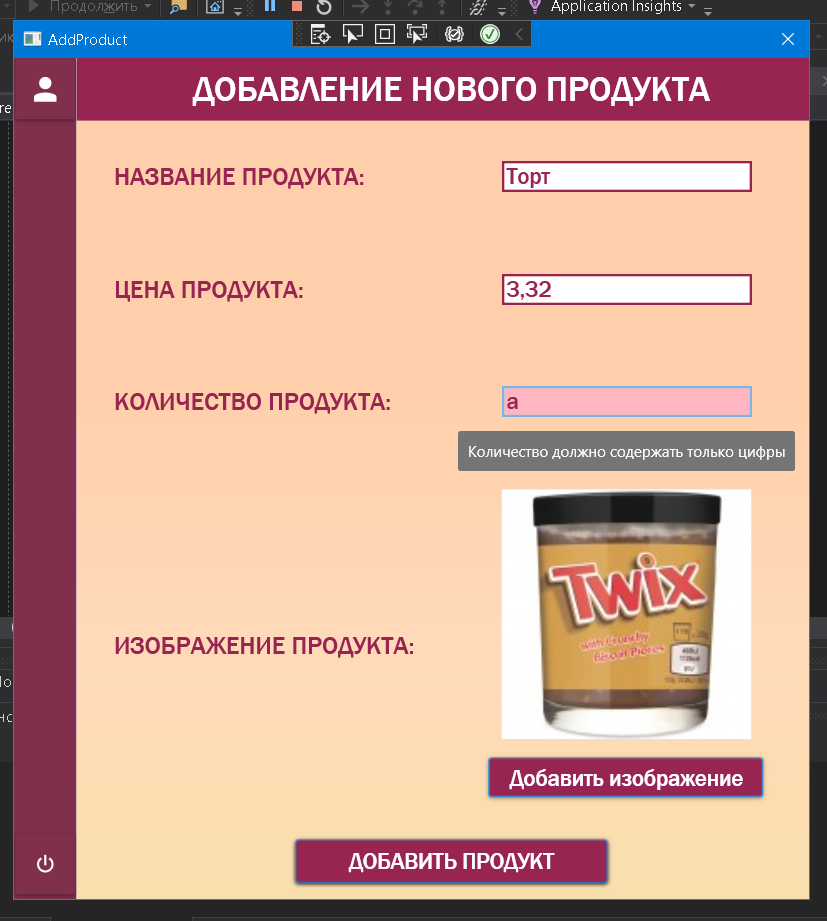


Рисунок 5.8 – Тест добавления товара на буквы в поле количество

Ошибка добавления товара, если поле *НАЗВАНИЕ ПРОДУКТА* содержит цифры (рисунок 5.9).

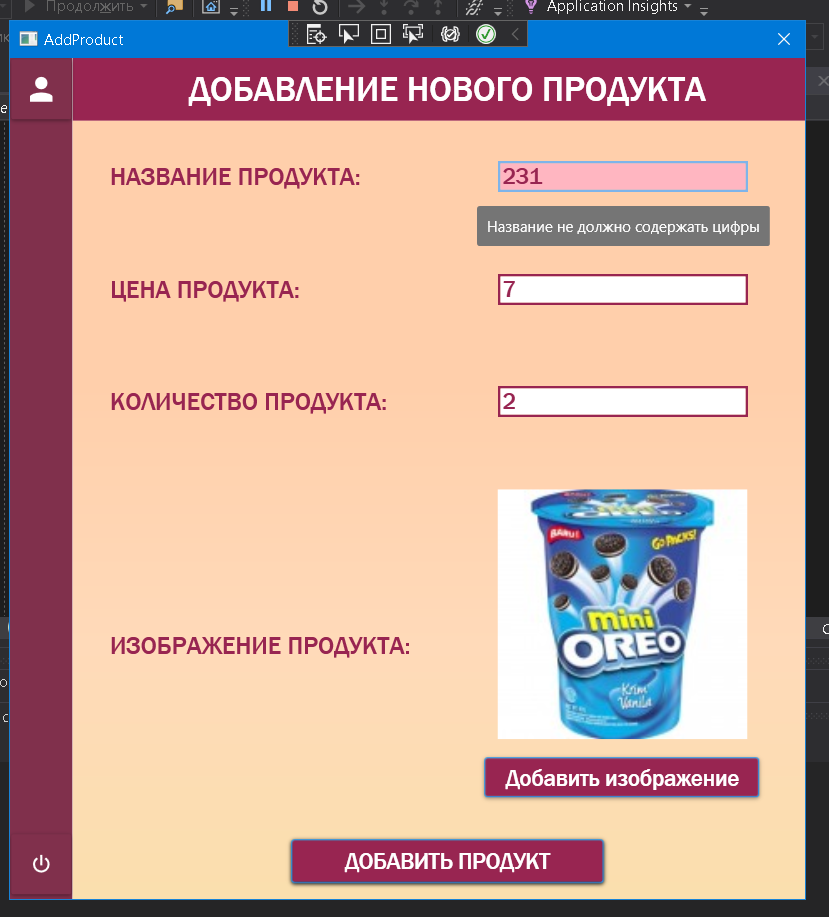


Рисунок 5.9 – Тест добавления товара на цифры в поле название

**5.4 Тест покупки товара**

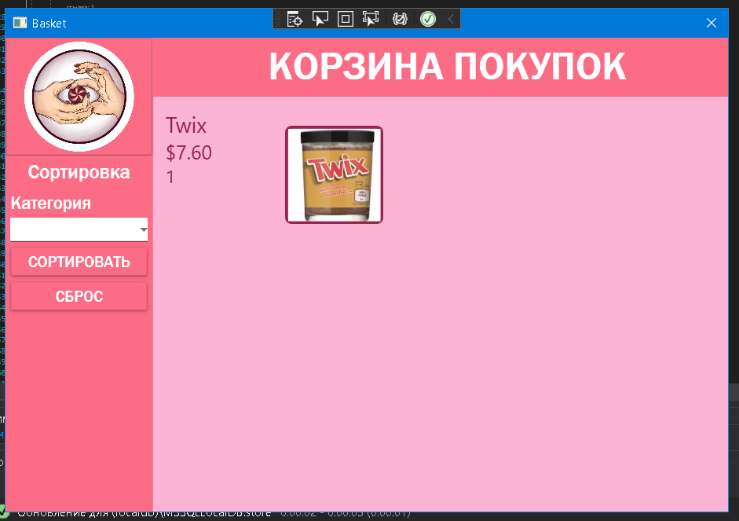
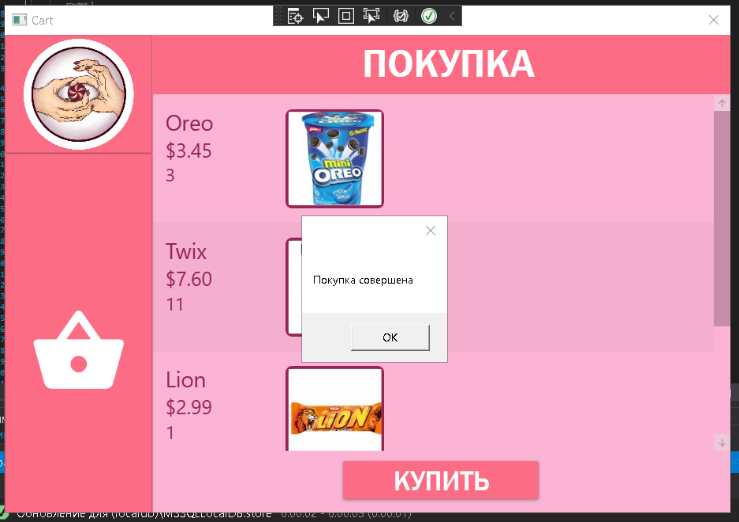
****

Рисунок 5.10 – Тест покупки товара

**5.5 Тест наблюдения за заказами**

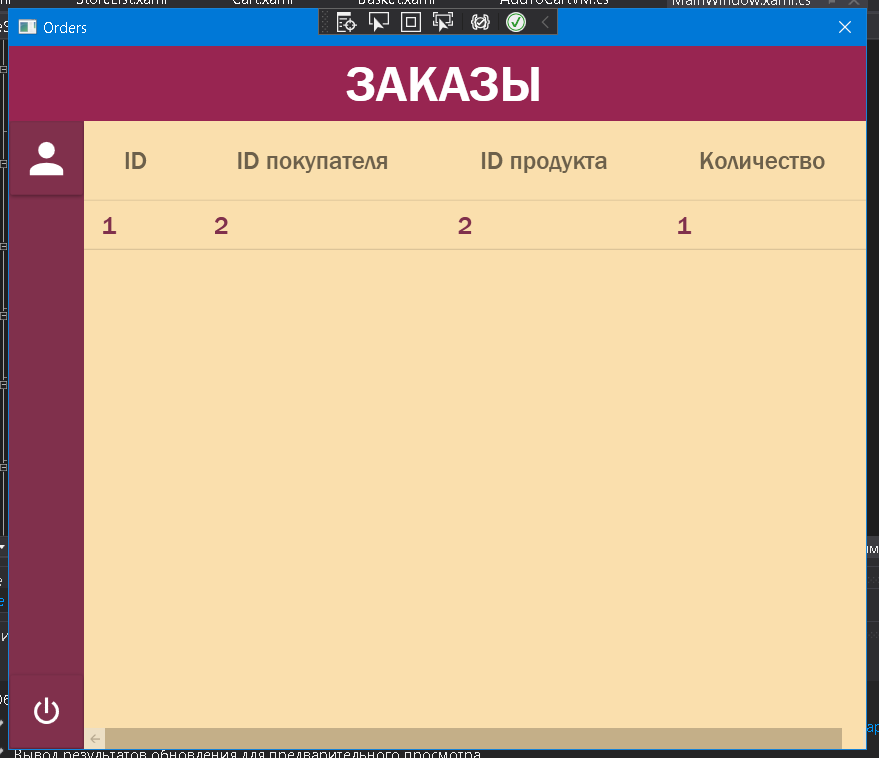
****

Рисунок 5.11 – Тест наблюдения за заказами

**5.6 Тест сортировки товаров**

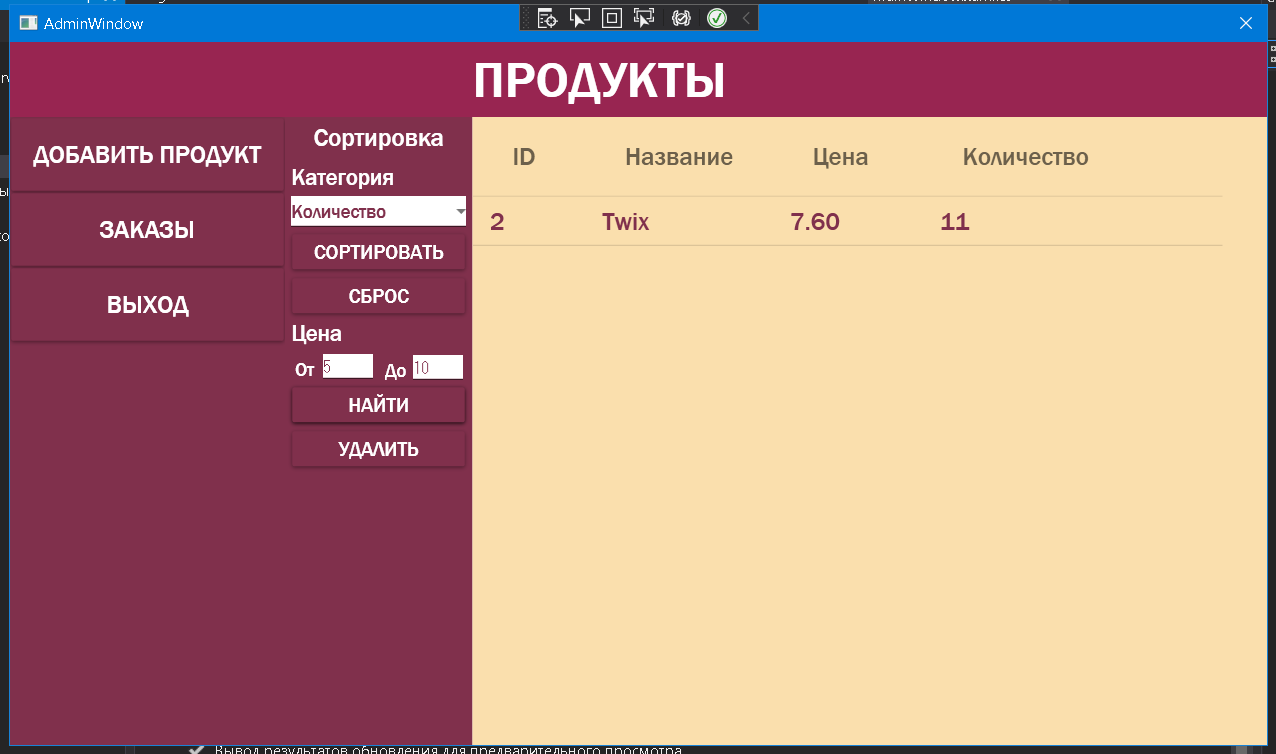
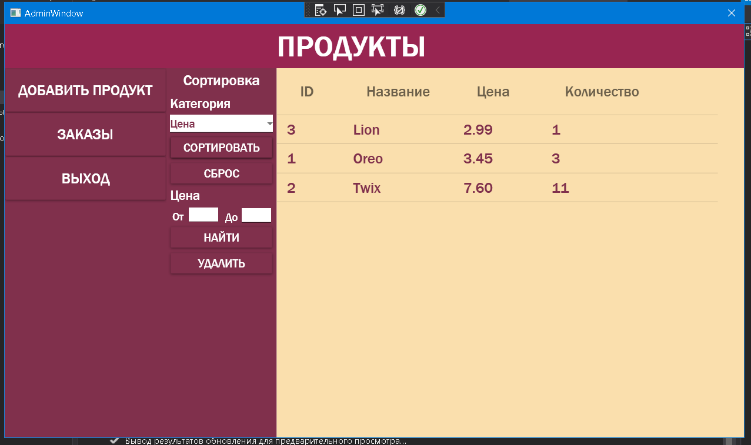
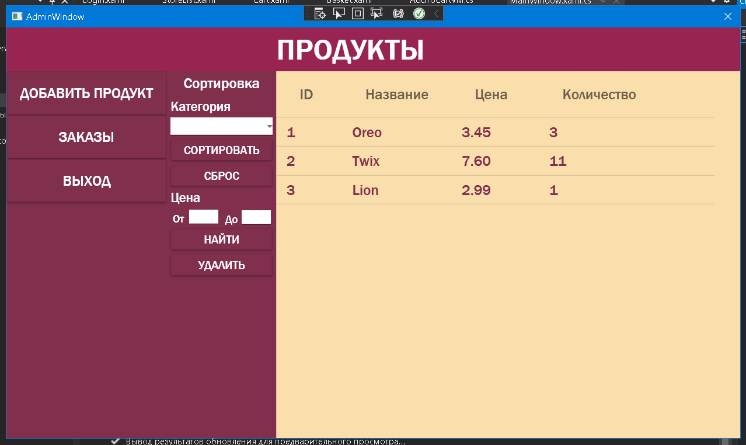
****

Рисунок 5.12 – Тест сортировки товаров (администратор)

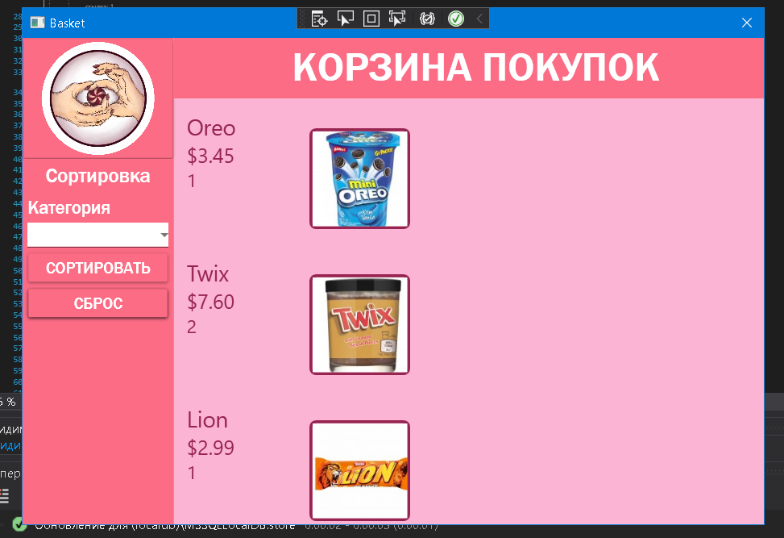
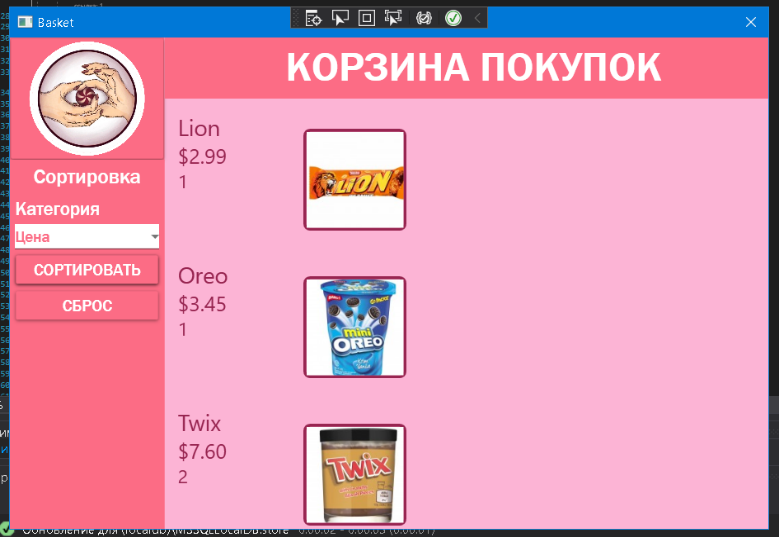


Рисунок 5.13 – Тест сортировки товаров (пользователь)

# 6. Руководство по установке и использования

На начальной странице необходимо ввести учетные данные (или же зарегистрироваться), для дальнейшего входа в программу (рисунок 6.1). В зависимости от введенных данных открывается либо окно администрирования, где администратор может добавлять, удалять, сортировать товары и наблюдать за заказами (рисунок 6.2), либо окно магазина, где пользователь может осуществлять покупку товаров, а также наблюдение за ними в корзине и их сортировку (рисунок 6.3).

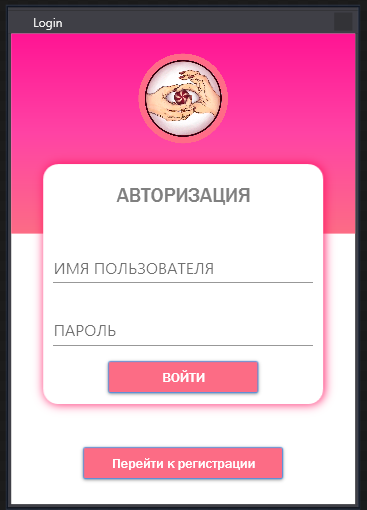
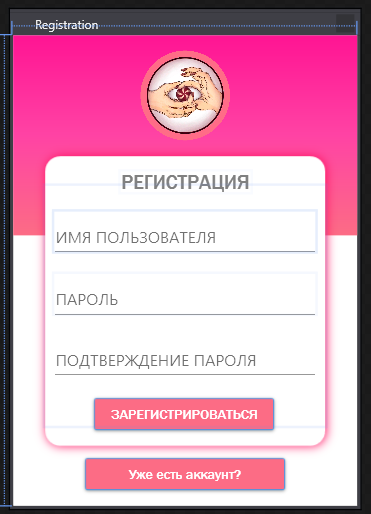
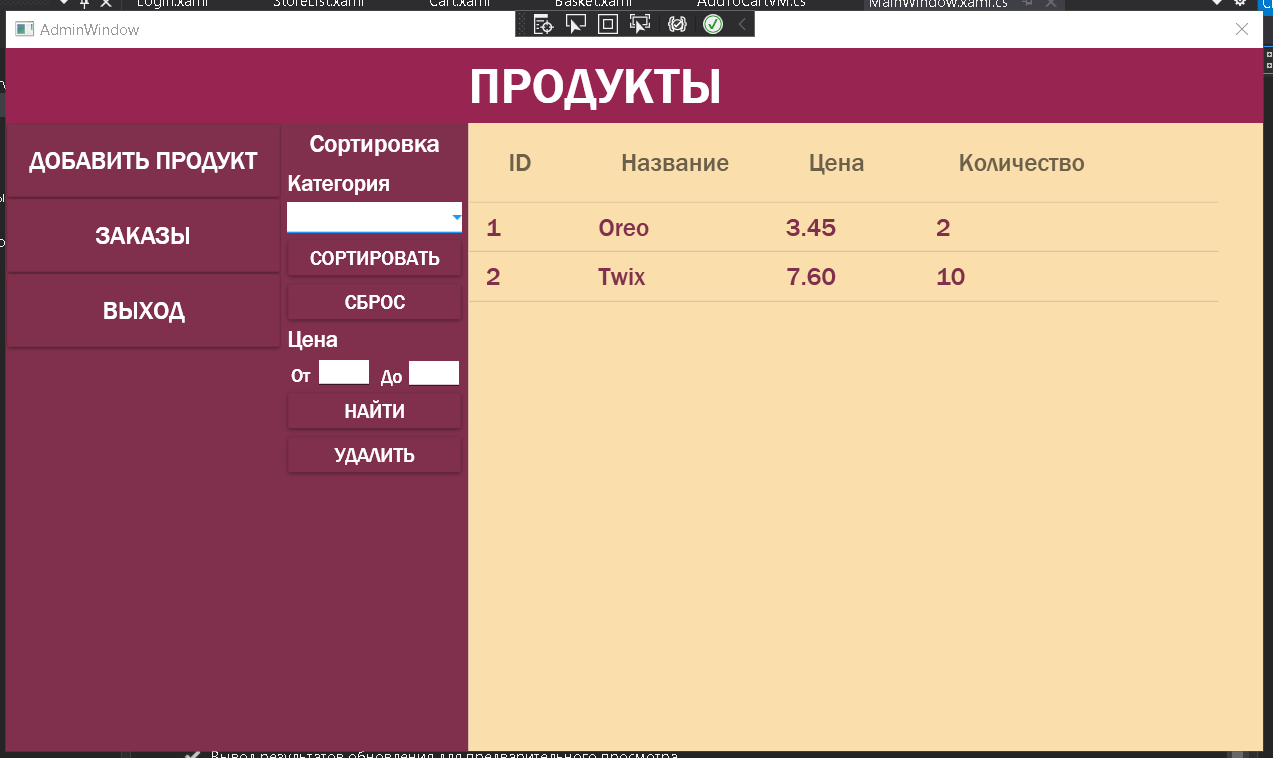


Рисунок 6.1 – Окна авторизации и регистрации

Рисунок 6.2 – Окно администрирования

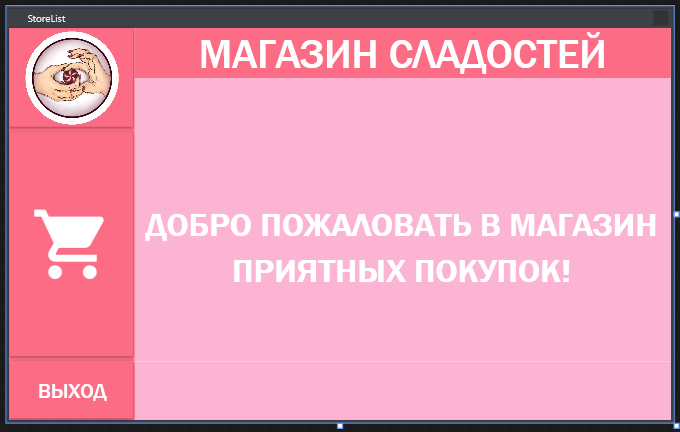


Рисунок 6.3 – Окно магазина

# ЗАКЛЮЧЕНИЕ

Итогом курсового проекта является рабочее приложение типа «Интернет-магазин» с возможностью разделения на администратора и пользователя, где администратор осуществляет добавление, наблюдение, сортировку товара и также наблюдение за заказами, а пользователь осуществляет покупку товаров, наблюдение за ними и сортировку.

В ходе выполнения курсового проекта было выполнено следующее:

* создание базы данных для хранения всей информации;
* разработка архитектуры приложения;
* разработка функциональной части приложения;
* разработка пользовательского интерфейса;
* написание исходного кода приложения;
* тестирование приложения.

В соответствии с полученным результатом работы программы можно сделать вывод, что требования технического задания выполнены, интерфейс получился достаточно простой и понятный, в нем сможет разобраться любой, даже далекий от информационных технологий пользователь.

За время выполнения курсового проекта по созданию приложения были изучены способы работы с «SQL Server», принципы создания приложения с использованием .NET и WPF, изучен новый язык верстки XAML для построения интерфейса и работы с ним. Использовался Entity Framework для работы с базой данный, а само приложение написано на языке С#.

# Список используемых источников

1. Введение в WPF, XAML, Паттерн MVVM [Электронный ресурс]. / Режим доступа <https://metanit.com/>
2. Учебник по языку SQL [Электронный ресурс]. / Режим доступа: <https://habr.com/ru/post/255361/>

**Приложение А**

**AdminWindowVM.cs**

using OnlineStore.Model;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.ComponentModel;

using System.Runtime.CompilerServices;

using OnlineStore.Model.Data;

namespace OnlineStore.ViewModel

{

public class AdminWindowVM : INotifyPropertyChanged

{

ProductContext db = new ProductContext();

MyCommand deleteCommand;

private Product selectedProduct;

public Product SelectedProduct

{

get { return selectedProduct; }

set

{

selectedProduct = value;

OnPropertyChanged("selectedProduct");

}

}

public ObservableCollection<Product> Products { get; set; }

public AdminWindowVM()

{

Products = new ObservableCollection<Product>(db.Products.ToList());

}

public MyCommand DeleteCommand

{

get

{

return deleteCommand ??

(deleteCommand = new MyCommand((selectedItem) =>

{

if (selectedItem == null) return;

Product product = selectedItem as Product;

Products.Remove(product);

db.Products.Remove(product);

db.SaveChanges();

MessageBox.Show("Продукт удалён");

}));

}

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName] string prop = "")

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(prop));

}

}

}

**AddToCartVM.cs**

using OnlineStore.Model;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.ComponentModel;

using System.Runtime.CompilerServices;

using OnlineStore.Model.Data;

using OnlineStore.View;

using System.Data.Entity;

namespace OnlineStore.ViewModel

{

public class AddToCartVM : INotifyPropertyChanged

{

ProductContext db = new ProductContext();

PurchaseContext pur = new PurchaseContext();

OrderContext or = new OrderContext();

public ObservableCollection<Product> Items { get; set; }

MyCommand buyCommand;

public AddToCartVM()

{

Items = new ObservableCollection<Product>(db.Products.ToList());

}

public MyCommand BuyCommand

{

get

{

return buyCommand ??

(buyCommand = new MyCommand((selectedItem) =>

{

if (selectedItem == null) return;

Product product = selectedItem as Product;

Product curPr = db.Products.Find(product.Id);

Purchase p = pur.Purchases.Where(pr => pr.Product\_id == product.Id).FirstOrDefault();

Order o = or.Orders.Where(pr => pr.Id\_product == product.Id).FirstOrDefault();

if (p == null && curPr != null )

{

if (curPr.Quantity >=2)

{

curPr.Quantity--;

db.Entry(curPr).State = EntityState.Modified;

db.SaveChanges();

Items = new ObservableCollection<Product>(db.Products.ToList());

Purchase purchase = new Purchase

{

Product\_id = curPr.Id,

Name = curPr.Name,

Price = curPr.Price,

Source = curPr.Source,

Quantity = 1

};

Order order = new Order

{

Id\_product = curPr.Id,

Id\_user = 2,

Quantity = 1

};

pur.Purchases.Add(purchase);

pur.SaveChanges();

or.Orders.Add(order);

or.SaveChanges();

MessageBox.Show("Покупка совершена");

}

else if (curPr.Quantity == 1)

{

Purchase purchase = new Purchase

{

Product\_id = curPr.Id,

Name = curPr.Name,

Price = curPr.Price,

Source = curPr.Source,

Quantity = 1

};

Order order = new Order

{

Id\_product = curPr.Id,

Id\_user = 2,

Quantity = 1

};

pur.Purchases.Add(purchase);

pur.SaveChanges();

or.Orders.Add(order);

or.SaveChanges();

MessageBox.Show("Покупка совершена");

db.Products.Remove(curPr);

db.SaveChanges();

Items.Remove(product);

MessageBox.Show("Продукт закончился");

}

}

else if (p!= null)

{

if (curPr.Quantity >= 2)

{

curPr.Quantity--;

db.Entry(curPr).State = EntityState.Modified;

db.SaveChanges();

Items = new ObservableCollection<Product>(db.Products.ToList());

p.Quantity++;

pur.Entry(p).State = EntityState.Modified;

pur.SaveChanges();

o.Quantity++;

or.Entry(o).State = EntityState.Modified;

or.SaveChanges();

MessageBox.Show("Покупка совершена");

}

else if (curPr.Quantity ==1 )

{

p.Quantity++;

pur.Entry(p).State = EntityState.Modified;

pur.SaveChanges();

o.Quantity++;

or.Entry(o).State = EntityState.Modified;

or.SaveChanges();

Items.Remove(product);

MessageBox.Show("Покупка совершена");

db.Products.Remove(curPr);

db.SaveChanges();

MessageBox.Show("Продукт закончился");

}

}

}));

}

}

private Product selectedProduct;

public Product SelectedProduct

{

get { return selectedProduct; }

set

{

selectedProduct = value;

OnPropertyChanged("selectedProduct");

}

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName] string prop = "")

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(prop));

}

}

}

**BasketVM.cs**

using OnlineStore.Model;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.ComponentModel;

using System.Runtime.CompilerServices;

using OnlineStore.Model.Data;

namespace OnlineStore.ViewModel

{

public class BasketVM : INotifyPropertyChanged

{

public ObservableCollection<Purchase> Purchases { get; set; }

public BasketVM()

{

using (PurchaseContext db = new PurchaseContext())

{

Purchases = new ObservableCollection<Purchase>(db.Purchases.ToList());

}

}

private Purchase selectedPurchase;

public Purchase SelectedPurchase

{

get { return selectedPurchase; }

set

{

selectedPurchase = value;

OnPropertyChanged("selectedPurchase");

}

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName] string prop = "")

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(prop));

}

}

}

**OrderVM.cs**

using OnlineStore.Model;

using System;

using System.Collections.Generic;

using System.Collections.ObjectModel;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.ComponentModel;

using System.Runtime.CompilerServices;

using OnlineStore.Model.Data;

namespace OnlineStore.ViewModel

{

public class OrderVM : INotifyPropertyChanged

{

public ObservableCollection<Order> Orders { get; set; }

public OrderVM()

{

using (OrderContext db = new OrderContext())

{

Orders = new ObservableCollection<Order>(db.Orders.ToList());

}

}

private Order selectedOrder;

public Order SelectedOrder

{

get { return selectedOrder; }

set

{

selectedOrder = value;

OnPropertyChanged("SelectedOrder");

}

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName] string prop = "")

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(prop));

}

}

}

**Приложение Б**

**MyCommand.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.Windows.Input;

namespace OnlineStore.ViewModel

{

public class MyCommand : ICommand

{

private Action<object> execute;

private Func<object, bool> canExecute;

public event EventHandler CanExecuteChanged

{

add { CommandManager.RequerySuggested += value; }

remove { CommandManager.RequerySuggested -= value; }

}

public MyCommand(Action<object> execute, Func<object, bool> canExecute = null)

{

this.execute = execute;

this.canExecute = canExecute;

}

public bool CanExecute(object parameter)

{

return this.canExecute == null || this.canExecute(parameter);

}

public void Execute(object parameter)

{

this.execute(parameter);

}

}

}

**Приложение В**

**Product.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.ComponentModel;

using System.Runtime.CompilerServices;

namespace OnlineStore.Model

{

public class Product : INotifyPropertyChanged

{

public int id;

private string name;

private decimal price;

private int quantity;

private string source;

public int Id

{

get { return id; }

set

{

id = value;

OnPropertyChanged("Id");

}

}

public string Name

{

get { return name; }

set

{

name = value;

OnPropertyChanged("Name");

}

}

public decimal Price

{

get { return price; }

set

{

price = value;

OnPropertyChanged("Price");

}

}

public int Quantity

{

get { return quantity; }

set

{

quantity = value;

OnPropertyChanged("Quantity");

}

}

public string Source

{

get { return source; }

set

{

source = value;

OnPropertyChanged("Source");

}

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName] string prop = "")

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(prop));

}

public Product() { }

public Product(string name, decimal price, int quantity, string source)

{

this.name = name;

this.price = price;

this.quantity = quantity;

this.source = source;

}

public Product(int id, string name, decimal price, int quantity, string source)

{

this.id = id;

this.name = name;

this.price = price;

this.quantity = quantity;

this.source = source;

}

}

}

**User.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

namespace OnlineStore.Model

{

public class User

{

public int id { get; set; }

private string username;

private string password;

public string Username

{

get { return username; }

set { username = value; }

}

public string Password

{

get { return password; }

set { password = value; }

}

public User() { }

public User(string username, string password) {

this.username = username;

this.password = password;

}

}

}

**Purchase.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.ComponentModel;

using System.Runtime.CompilerServices;

namespace OnlineStore.Model

{

public class Purchase : INotifyPropertyChanged

{

public int id;

public int product\_id;

private string name;

private decimal price;

private int quantity;

private string source;

public int Id

{

get { return id; }

set

{

id = value;

OnPropertyChanged("Id");

}

}

public int Product\_id

{

get { return product\_id; }

set

{

product\_id = value;

OnPropertyChanged("Product\_id");

}

}

public string Name

{

get { return name; }

set

{

name = value;

OnPropertyChanged("Name");

}

}

public decimal Price

{

get { return price; }

set

{

price = value;

OnPropertyChanged("Price");

}

}

public int Quantity

{

get { return quantity; }

set

{

quantity = value;

OnPropertyChanged("Quantity");

}

}

public string Source

{

get { return source; }

set

{

source = value;

OnPropertyChanged("Source");

}

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName] string prop = "")

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(prop));

}

public Purchase() { }

public Purchase(int product\_id, string name, decimal price, int quantity, string source)

{

this.product\_id = product\_id;

this.name = name;

this.price = price;

this.quantity = quantity;

this.source = source;

}

}

}

**Order.cs**

using System;

using System.Collections.Generic;

using System.Linq;

using System.Text;

using System.Threading.Tasks;

using System.ComponentModel;

using System.Runtime.CompilerServices;

namespace OnlineStore.Model

{

public class Order : INotifyPropertyChanged

{

public int id;

private int id\_user;

private int id\_product;

private int quantity;

public int Id

{

get { return id; }

set

{

id = value;

OnPropertyChanged("Id");

}

}

public int Id\_user

{

get { return id\_user; }

set

{

id\_user = value;

OnPropertyChanged("Id\_user");

}

}

public int Id\_product

{

get { return id\_product; }

set

{

id\_product = value;

OnPropertyChanged("Id\_product");

}

}

public int Quantity

{

get { return quantity; }

set

{

quantity = value;

OnPropertyChanged("Quantity");

}

}

public event PropertyChangedEventHandler PropertyChanged;

public void OnPropertyChanged([CallerMemberName] string prop = "")

{

if (PropertyChanged != null)

PropertyChanged(this, new PropertyChangedEventArgs(prop));

}

public Order() { }

public Order (int id\_user, int id\_product, int quantity)

{

this.id\_user = id\_user;

this.id\_product = id\_product;

this.quantity = quantity;

}

public Order(int id, int id\_user, int id\_product, int quantity)

{

this.id = id;

this.id\_user = id\_user;

this.id\_product = id\_product;

this.quantity = quantity;

}

}

}

**Приложение Г**

**ProductContext.cs**

using System.Data.Entity;

namespace OnlineStore.Model.Data

{

class ProductContext : DbContext

{

public ProductContext() : base("DBConnection")

{

}

public DbSet<Product> Products { get; set; }

}

}

**UserContext.cs**

using System.Data.Entity;

namespace OnlineStore.Model.Data

{

class UserContext : DbContext

{

public UserContext() : base("DBConnection")

{

}

public DbSet<User> Users { get; set; }

}

}

**PurchaseContext.cs**

using System.Data.Entity;

namespace OnlineStore.Model.Data

{

class PurchaseContext : DbContext

{

public PurchaseContext() : base("DBConnection")

{

}

public DbSet<Purchase> Purchases { get; set; }

}

}

**OrderContext.cs**

using System.Data.Entity;

namespace OnlineStore.Model.Data

{

class OrderContext : DbContext

{

public OrderContext() : base("DBConnection")

{

}

public DbSet<Order> Orders { get; set; }

}

}