

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 Программное обеспечение информационных технологий

**ПОЯСНИТЕЛЬНАЯ ЗАПИСКА  
КУРСОВОГО ПРОЕКТА:**

по дисциплине «Объектно-ориентированные технологии программирования и стандарты проектирования»  
Тема Программное средство «Магазин игрушек ToyNow»

Исполнитель  
студент (ка) 2 курса группы 4 Гурина Кристина Сергеевна  
(Ф.И.О.)  
Руководитель работы ассистент Чистякова Ю.А  
(учен. степень, звание, должность, подпись, Ф.И.О.)

Курсовой проект защищен с оценкой \_\_\_\_\_

Председатель Пацей Н.В.  
(подпись)

Минск 2023

## Содержание

<b>Введение .....</b>	<b>3</b>
<b>1. Аналитический обзор прототипов и литературных источников .....</b>	<b>4</b>
<b>1.1 Анализ прототипов .....</b>	<b>4</b>
<b>2. Анализ требований к программному средству и разработка функциональных требований .....</b>	<b>8</b>
<b>2.1 Спецификация бизнес-требований к программному средству .....</b>	<b>8</b>
<b>2.2 Спецификация пользовательских требований к программному средству .....</b>	<b>9</b>
<b>2.3 Спецификация функциональных требований к программному средству .....</b>	<b>9</b>
<b>3. Проектирование программного средства .....</b>	<b>11</b>
<b>3.1 Проектирование архитектуры приложения .....</b>	<b>11</b>
<b>4. Проектирование программного средства .....</b>	<b>15</b>
<b>4.1 Реализация архитектуры MVVM .....</b>	<b>15</b>
<b>4.2 Реализация классов Models и ViewModels .....</b>	<b>16</b>
<b>4.3 Реализация паттерна Command .....</b>	<b>17</b>
<b>4.3 Реализация подключения к БД и паттерны Repository и UnitOfWork ....</b>	<b>17</b>
<b>4.5 Реализация вспомогательных классов и методов .....</b>	<b>18</b>
<b>4.6 Реализация Views .....</b>	<b>20</b>
<b>5. Тестирование, проверка работоспособности и анализ полученных результатов .....</b>	<b>21</b>
<b>5.1 Тестирование авторизации и регистрации .....</b>	<b>21</b>
<b>5.2 Тестирование функций администратора .....</b>	<b>23</b>
<b>5.2 Тестирование функций пользователя .....</b>	<b>25</b>
<b>6. Руководство по использованию .....</b>	<b>27</b>
<b>Заключение .....</b>	<b>33</b>
<b>Список литературы .....</b>	<b>34</b>
<b>Приложение А .....</b>	<b>35</b>
<b>Приложение Б .....</b>	<b>36</b>
<b>Приложение В .....</b>	<b>37</b>
<b>Приложение Г .....</b>	<b>38</b>
<b>Приложение Д .....</b>	<b>40</b>
<b>Приложение Е .....</b>	<b>42</b>

## Введение

Основной целью курсового проекта является разработка программного средства для магазина игрушек «Toy Now». Данное программное средство позволит покупателям приобретать игрушки из дома или офиса, экономя тем самым время и деньги на поездки в магазины. Это также дает возможность покупателям иметь доступ к более широкому ассортименту товаров, чем в обычном магазине, и оплачивать товары онлайн. Пользователи могут легко сравнивать цены и характеристики различных товаров, выбирая оптимальное сочетание цены и качества.

В основу проектирования легли следующие принципы:

1) Легкость использования. Приложение должно быть интуитивно понятным и простым в использовании, даже для тех пользователей, которые не имеют опыта покупок в интернете.

2) Безопасность. Приложение должно обеспечивать защиту персональной информации.

3) Оперативность. Приложение должно быстро открываться и быстро реагировать на действия пользователя.

4) Многофункциональность. Приложение должно предоставлять множество возможностей для пользователя, включая поиск, заказ товаров, быструю доставку и удобный способ оплаты, а также возможность отслеживания заказа и возможность связаться со службой поддержки.

5) Надежность. Приложение должно быть надежным и стабильным, чтобы пользователи могли покупать в нем без каких-либо сбоев или проблем.

Программное средство для магазина игрушек «Toy Now» должно реализовывать все вышеперечисленные принципы и решать поставленные задачи. При выполнении курсового проекта будут использованы язык программирования C#, принципы и приемы ООП, база данных MS SQL Server и технология Windows Presentation Foundation (WPF).

Для успешной реализации курсового проекта необходимо:

- провести анализ соответствующей литературы;
- ознакомиться с прототипами программных средств выбранной мной темы;
- определить функциональные требования;
- продумать структуру базы данных;
- продумать структуру проекта;
- реализовать программное средство;
- протестировать программное средство;
- написать руководство пользователя.

Содержание данной пояснительной записки отражает все этапы выполнения моего курсового проекта.

## 1. Аналитический обзор прототипов и литературных источников

### 1.1 Анализ прототипов

Для полноценного определения постановки задачи курсового проекта необходимо провести анализ прототипов программных средств, связанных с выбранной темой.

1) funtastik.by – платформа, на которой предоставлен широкий выбор игрушек разных категории. Содержимое главной страницы представлена на рисунке 1.1. Преимущества данного веб-сайта включают приятный дизайн, который создает привлекательную и уютную атмосферу для пользователей. Удобный интерфейс позволяет легко и интуитивно осуществлять навигацию по платформе. Все товары на платформе разделены на категории, что обеспечивает систематизацию их представления. Кроме того, пользователи имеют возможность фильтровать игрушки по полу, возрасту и цене, а также сортировать их по различным критериям. Это существенно облегчает поиск и позволяет пользователям быстро найти нужный товар. Сервис funtastik.by также предоставляет разнообразные варианты оплаты и доставки товаров. Пользователи могут выбрать наиболее удобный способ оплаты, будь то онлайн-платежи, оплата при получении или другие варианты, а также выбрать предпочитаемый метод доставки. Это позволяет удовлетворить различные потребности и предпочтения клиентов. Пользователи могут прочитать отзывы о конкретном товаре, что помогает им принять взвешенное решение о покупке. Это способствует установлению доверия к платформе и повышает удовлетворенность клиентов.

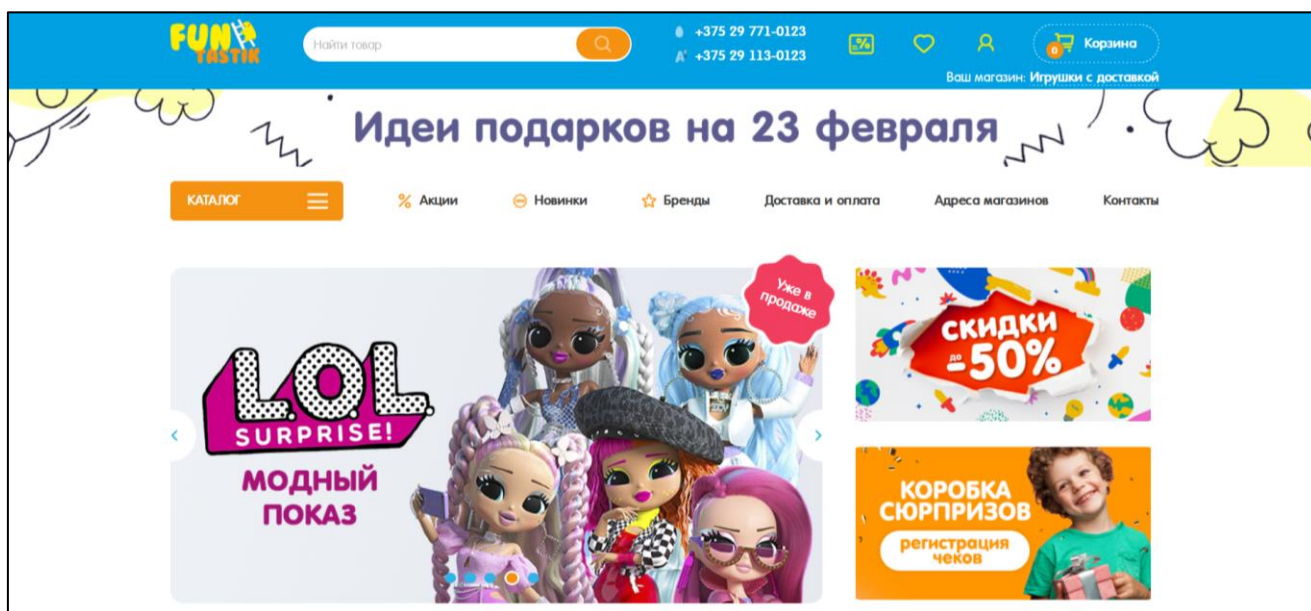


Рисунок 1.1

Отличительной особенностью данной платформы является удобная функция подбора игрушек, основанная на интересах и возрасте ребёнка, доступная прямо на главной странице (Рисунок 1.2). Это позволяет пользователям сразу получить

персонализированные рекомендации и упрощает процесс выбора игрушек, соответствующих потребностям и предпочтениям каждого ребёнка.

Рисунок 1.2

Среди недостатков данного веб-сайта можно выделить поле для выбора способа доставки и условий оплаты, которое не является обязательным и не имеет проверки корректности.

В целом, прототип программного средства *funtastik.by* является хорошим примером успешного веб-магазина, предоставляющего широкий выбор игрушек.

2) *игрушкиоптом.рф* – ещё одна платформа продажи игрушек. Преимуществами данного веб-сайта являются большой выбор товаров, удобный поиск, возможность выбора способа оплаты и доставки товаров, наличие пункта самовывоза. Из недостатков следует отметить устаревший дизайн и слишком нагруженный описанием товаров интерфейс, что затрудняет навигацию и усложняет процесс выбора товаров.

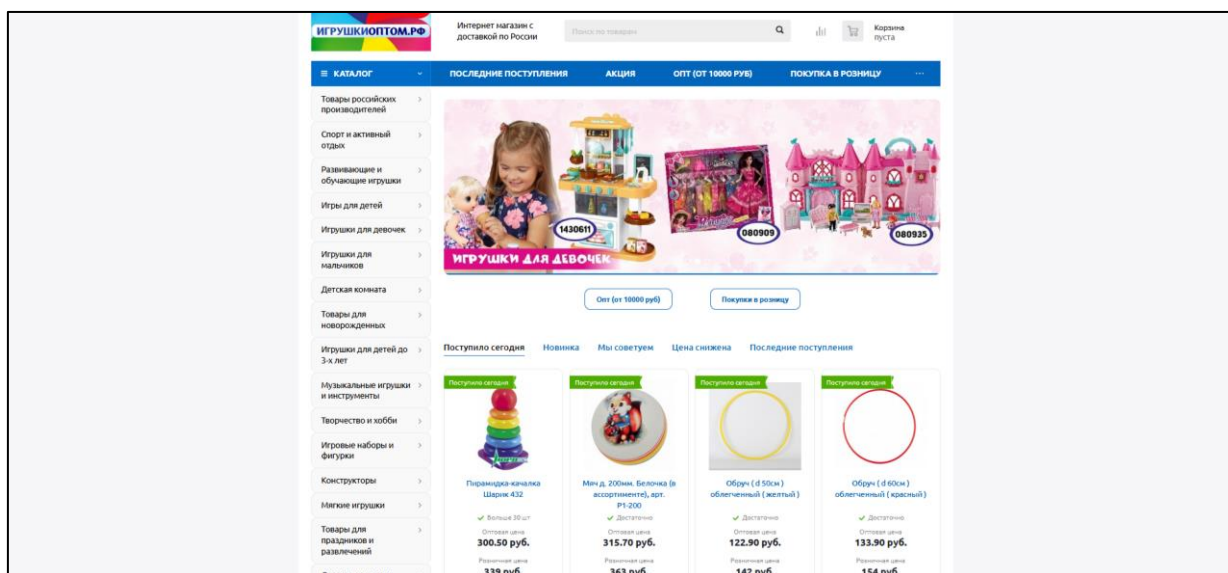


Рисунок 1.3

3) «Детмир» – сеть магазинов игрушек в Минске. Преимуществами платформы *detmir.by* являются большой выбор товаров, приятный дизайн, наличие корзины, бонусной карты, а также возможность обмена и возврата товара. Сервис

предоставляет возможность выбрать удобный способ оплаты и доставки товаров. Главная страница представлена на рисунке 1.4.

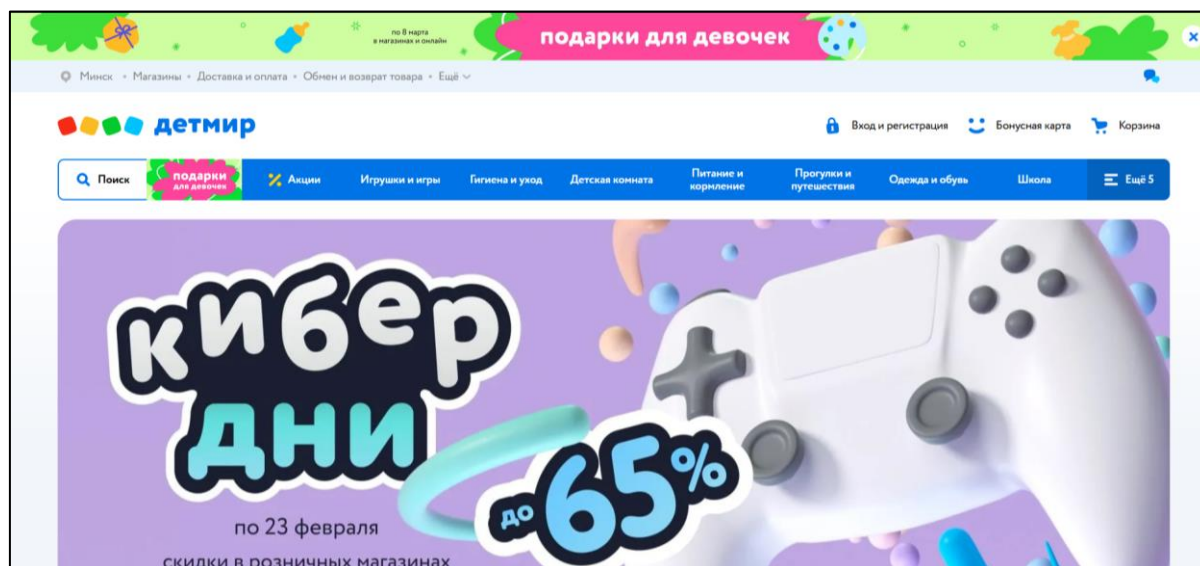


Рисунок 1.4

Главным недостатком веб-сайта является большой размер блоков товаров, который делает их просмотр неудобным.

4) «Мир игрушек» – интернет-магазин игрушек. Преимуществами данного ресурса являются большой выбор товаров, наличие необходимого для интернет-магазина функционала. Также пользователи могут обратиться за технической поддержкой, вернуть товар, ознакомиться с актуальными акциями и предложениями. Сервис предоставляет возможность выбрать удобный способ оплаты и доставки товаров. Главная страница представлена на рисунке 1.5.

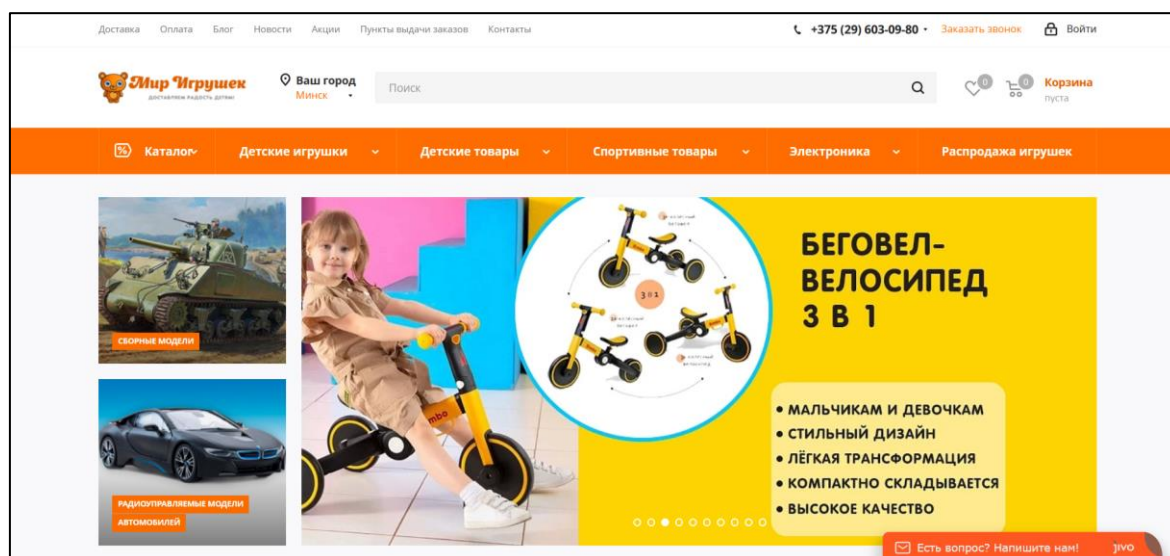


Рисунок 1.5

Описанный на рисунке 1.6 интерфейс каталога товаров может быть воспринят пользователями как избыточный и запутанный из-за большого количества



информации, представленной на одной странице. Это может затруднить навигацию и усложнить процесс поиска нужных товаров. Важно учесть, что слишком много информации на одной странице может вызвать перегрузку пользователя и негативно сказаться на его пользовательском опыте.

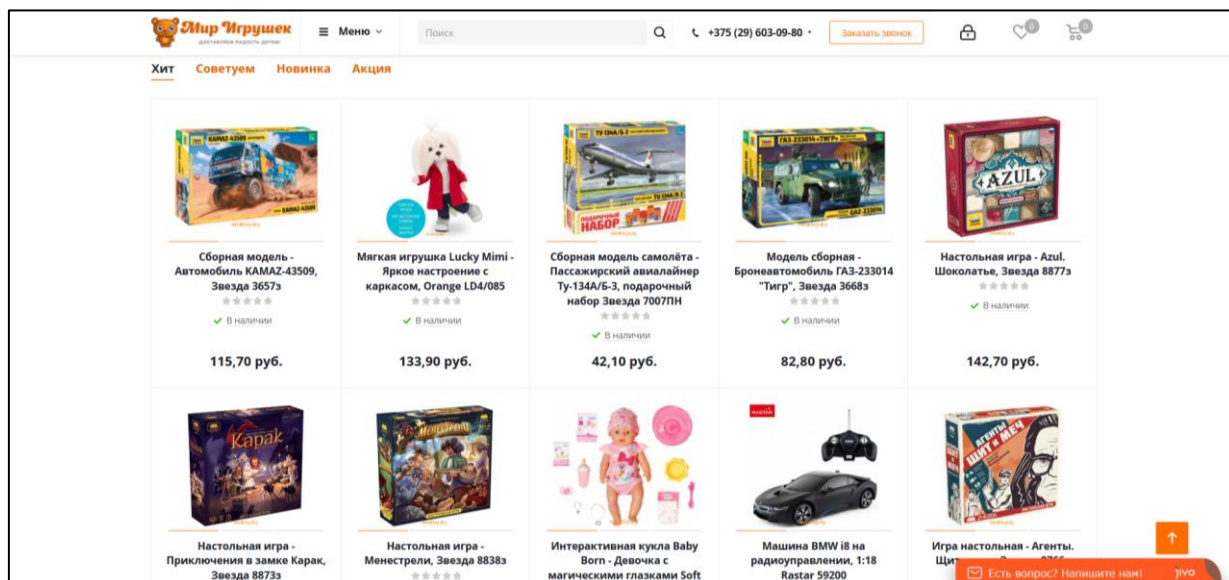


Рисунок 1.6

5) Посещение реального магазина. Преимуществом данного выбора является возможность непосредственно на месте оценить реальный вид продукта, воспользоваться помощью продавца-консультанта, а также простота в оформлении заказа.

Среди недостатков данной альтернативы можно выделить риск отсутствия необходимого товара к моменту прибытия в магазин, сложность в ориентировании при выборе среди множества игрушек. Также временные и денежные затраты на путь до магазина.

Изучение перечисленных аналогов позволяет провести анализ и оценить как положительные, так и отрицательные стороны каждого альтернативного решения. Этот анализ дает возможность сформулировать список требований к разрабатываемому приложению. Определение требований является важным этапом, поскольку оно определяет основные функциональные и нефункциональные характеристики, которые должны быть реализованы в приложении, чтобы соответствовать потребностям пользователей и решать их задачи наиболее эффективным образом.

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

Анализ требований — это процесс сбора требований к программному обеспечению, их систематизации, документирования, анализа, выявления противоречий, неполноты, разрешения конфликтов в процессе разработки программного обеспечения.

Цель анализа требований в проектах — получить максимум информации о заказчике и специфике его задач, уточнить рамки проекта, оценить возможные риски. На этом этапе происходит идентификация принципиальных требований методологического и технологического характера, формулируются цели и задачи проекта, а также определяются критические факторы успеха, которые впоследствии будут использоваться для оценки результатов внедрения. Определение и описание требований – шаги, которые во многом определяют успех всего проекта, поскольку именно они влияют на все остальные этапы.

Различают три уровня требований к проекту:

- бизнес-требования;
- пользовательские требования;
- функциональные требования.

Определение и описание требований являются ключевыми шагами для успешного завершения проекта.

### **2.1 Спецификация бизнес-требований к программному средству**

Бизнес-требования содержат высокоуровневые цели организации или заказчиков системы. Как правило, их высказывают те, кто финансируют проект, покупатели системы, менеджер реальных пользователей, отдел маркетинга. Важно учесть, что бизнес-требования должны быть конкретными, измеримыми и достижимыми. Они должны быть согласованы с интересами и целями организации или заказчика системы, а также учитывать потребности и ожидания пользователей. Анализ и учет бизнес-требований позволяет определить основные задачи, которые должны быть решены программным средством, и направляет весь процесс разработки в соответствии с ожиданиями и потребностями пользователей и заказчика. Для курсового проекта были выдвинуты высокоуровневые требования, которые можно рассматривать как общие требования к разрабатываемому средству. К их числу относятся:

- простота и понятность интерфейса;
- использование системы управления базами данных (СУБД);
- повышение эффективности работы магазина и улучшение процесса управления продажами;
- поддержка нескольких способов оплаты.

Весь дальнейший процесс проектирования и разработки программного средства должен находиться в очерченных бизнес-требованиями границах.



## **2.2 Спецификация пользовательских требований к программному средству**

Пользовательские требования играют важную роль в определении функциональности и возможностей разрабатываемого программного средства. Они описывают, каким образом система будет использоваться пользователями и какие конкретные задачи она поможет им решить. Учет пользовательских требований позволяет сосредоточиться на создании удобного и интуитивно понятного интерфейса, который соответствует потребностям и ожиданиям конечных пользователей. Это способствует повышению удовлетворенности пользователей и успешной адаптации системы в рабочей среде.

Программное средство должно предоставлять следующие функциональные возможности для пользователя

- регистрация
- авторизация;
- выполнение поисковых запросов, фильтрации/сортировки;
- добавление товара в корзину;
- заполнение формы заказа (выбор способа оплаты и доставки);
- возможность просматривать и оставлять отзывы о товаре;
- возможность просмотра личной страницы пользователя с отображением всех заказанных товаров;
- возможность обратиться за поддержкой к администрации.

Администратору должен выполнять следующие функции

- управление базой данных;
- изменение статуса заказов;
- просмотр информации о пользователях и заказах;
- уведомление пользователя о статусе заказа;
- техническая поддержка пользователей.

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список пользовательских требований. Разработка данной программной системы должна включать в себя реализацию функций, соответствующих сформированным спискам.

## **2.3 Спецификация функциональных требований к программному средству**

После проведения анализа были выявлены следующие функциональные требования:

- архитектура приложения должна соответствовать шаблонам проектирования, таким как MVVM, Command;
- вся информация должна храниться в базе данных;
- приложение должно обеспечивать поддержку добавления информации в базу данных;
- приложение должно производить валидацию вводимых пользователем данных;

- приложение должно корректным образом обрабатывать возникающие исключительные ситуации: отображать понятное для пользователя сообщение о возникшей ошибке;

- приложение должно предоставлять пользователям возможность создания нового аккаунта в виде регистрационной формы;

- приложение должно предоставлять возможность пользователям проходить аутентификацию и входить в систему под соответствующим введенным данным пользовательским именем;

- приложение должно предоставлять возможность поиска продуктов по следующим критериям: название, категория, цена;

- приложение должно предоставлять возможность оформления заказа;

- приложение должно предоставлять возможность просмотра информации о всех заказах;

- должна присутствовать средства для связи пользователя с администратором.

Таким образом, был проведен тщательный анализ требований к программному средству, который позволил разработать список функциональных требований. Разработка данной программной системы должна проводиться в соответствии с сформированными списком.

### 3. Проектирование программного средства

#### 3.1 Проектирование архитектуры приложения

Архитектура программного обеспечения — совокупность важнейших решений об организации программной системы. Архитектура включает:

- выбор структурных элементов и их интерфейсов, с помощью которых составлена система, а также их поведения в рамках сотрудничества структурных элементов;

- соединение выбранных элементов структуры и поведения во всё более крупные системы;

- архитектурный стиль, который направляет всю организацию — все элементы, их интерфейсы, их сотрудничество и их соединение.

Для удовлетворения проектируемой системы различным атрибутам качества применяются различные архитектурные шаблоны (паттерны). В разрабатываемом приложении используется архитектурный шаблон Model-View-ViewModel (MVVM).

Шаблон MVVM имеет три основных слоя: модель, которая представляет бизнес-логику приложения, представление пользовательского интерфейса, и представление-модель, в котором содержится вся логика построения графического интерфейса и ссылка на модель, поэтому он выступает в качестве модели для представления.

На рисунке 3.1 представлена диаграмма, которая показывает общую структуру приложения в рамках шаблона MVVM.

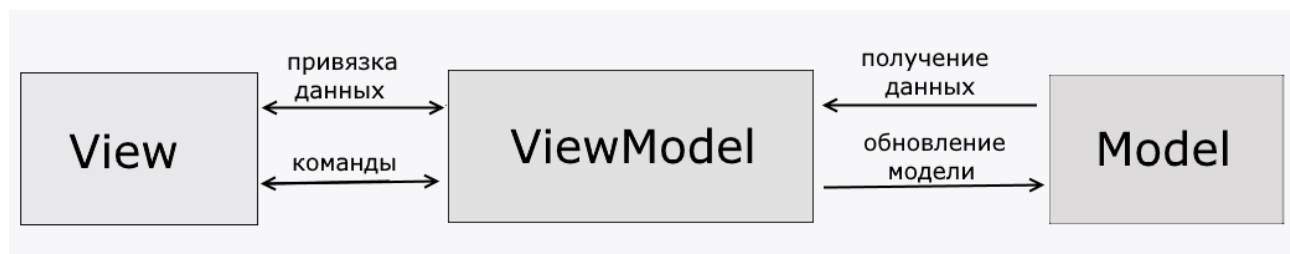


Рисунок 3.1 – Структура шаблона MVVM

View определяет визуальный интерфейс, через который пользователь взаимодействует с приложением. Так как пользовательский интерфейс и качество его реализации играет далеко не последнее место в конечном результате, разработка эффективного интерфейса, приятного и удобного для конечного пользователя, является важной задачей. Поэтому для хорошего проектирования View необходимо понять, как пользователь будет взаимодействовать с приложением. Для этого была составлена диаграмма использования представленная в приложении А, на которой представлен принцип работы приложения с точки зрения пользователя.

На рисунке 3.2 отображена диаграмма последовательности для авторизации. На диаграмме последовательности отображаются только те объекты, которые непосредственно принимают участие во взаимодействии.

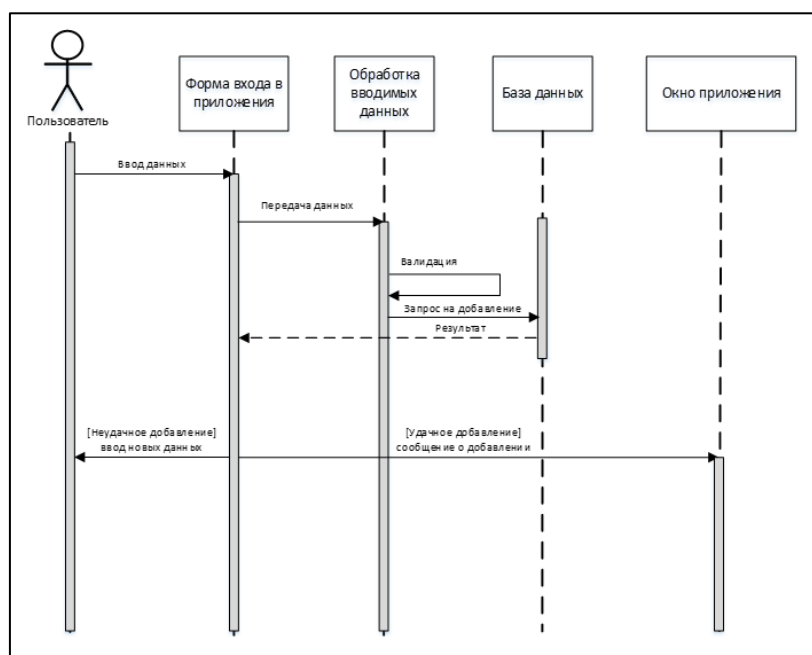


Рисунок 3.2 – Диаграмма последовательности для авторизации

Диаграмма компонентов приложения, представленная на рисунке 3.3, визуально отображает различные компоненты, из которых состоит наше программное средство. Компоненты включают в себя модули, классы и другие структурные элементы, которые взаимодействуют друг с другом для обеспечения функциональности приложения. Эта диаграмма помогает нам понять архитектурную организацию приложения и его основные составляющие. Она также служит важным инструментом для разработчиков, позволяя лучше структурировать и понять внутреннюю логику и взаимодействие компонентов в системе.

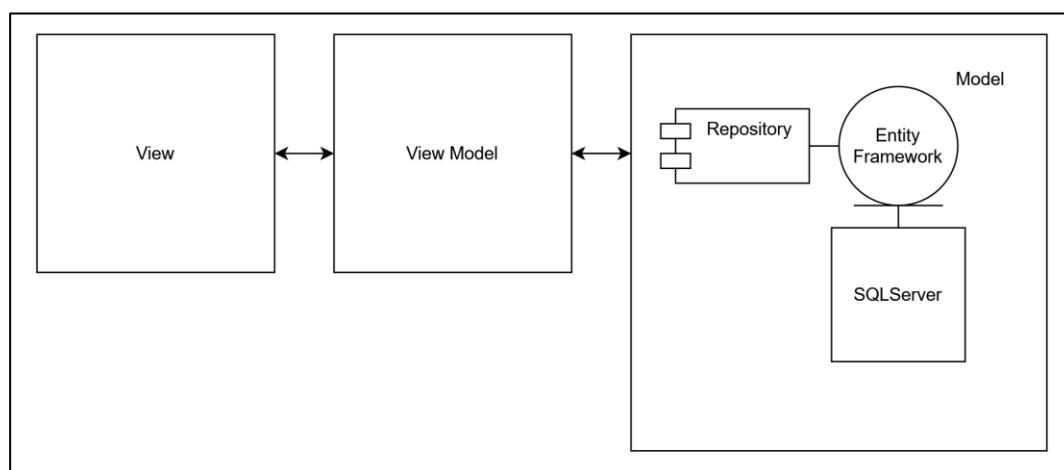


Рисунок 3.3 – Диаграмма последовательности для авторизации

На рисунке 3.4 представлена блок-схема, которая наглядно описывает последовательность шагов и логику, применяемую при регистрации пользователя в системе. Блок-схема помогает визуализировать основные этапы процесса регистрации, включая ввод данных, их валидацию, сохранение в базе данных и уведомление пользователя об успешной регистрации. Такая графическая

представление алгоритма облегчает понимание его структуры и взаимосвязей между различными этапами.

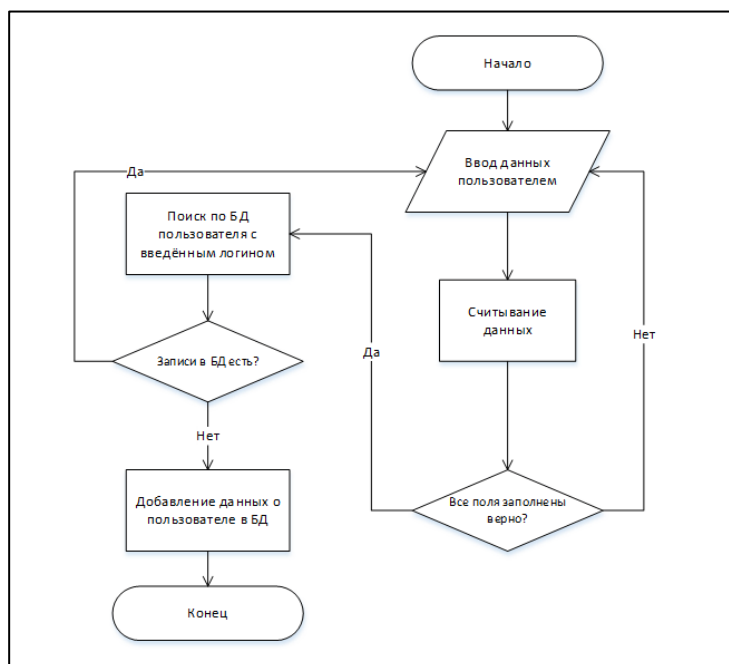


Рисунок 3.5 – Схема алгоритма регистрации

Диаграмма классов представлена в приложении Б.

Логика перехода между страницами пользовательской части реализована в классе `MainWindowViewModel`, а для администраторской части – в классе `AbminViewModel`.

Для получения и записи данных в БД используются классы `AppConection`, `ManagerDBRepository` и `UnitOfWork`. Эти классы предоставляют необходимый уровень абстракции, который облегчает и ускоряет доступ к БД. Они выполняют роль посредников между приложением и базой данных, предоставляя удобные методы для работы с данными.

Для взаимодействия с БД в приложении используется технология `Entity Framework Core`. Она обеспечивает удобный и эффективный способ работы с данными, абстрагируя от особенностей конкретной БД и позволяя использовать объектно-ориентированный подход к работе с данными.

Для администрирования приложения и управления доступом к БД был создан пользователь с логином `admin`, обладающий правами администратора. Это позволяет осуществлять контроль над базой данных, выполнение административных задач и обеспечивает безопасность и конфиденциальность данных приложения

### 3.2. Модель базы данных

Для реализации поставленной задачи была создана база данных «shop». Для ее создания использовалась система управления реляционными базами данных `MSQL Server`. База данных состоит из таблиц, представленных в приложении В.

Таблица «Users» содержит информацию о пользователях. В данной таблице поле ID является первичным ключом. Поле Login хранит логин пользователя, Password – пароль, Email – адрес электронной почты, Phone – номер телефона, Photo – фото, Name и Surname – соответственно имя и фамилию.

Таблица «Categories» содержит информацию о категориях товара. Первичный ключ CategoryID хранит уникальный идентификатор товара, Name – название, ImageLink – путь к изображению соответствующего товара данной категории.

Таблица «Products», содержащая информацию о товарах. Первичный ключ ID хранит уникальный идентификатор товара, Title – название, CategoryId – ID категории, Price – цена, Rating – текущий рейтинг в соответствии с оценками пользователей, ImageLink – путь к изображению соответствующего товара.

Таблица «Carts» содержит данные о корзине пользователя. В данной таблице первичный ключ ID является идентификатором корзины, поле UserID, ссылающееся на поле ID таблицы «Users», содержит ID пользователя, поле ProductID, ссылающееся на поле ID таблицы «Product», поле Count содержит информацию о количестве товара в корзине, поле SumPrice определяет сумму товаров, а поле Cart – карту для оплаты.

Таблица «Orders» содержит данные о заказах. В данной таблице первичный ключ OrderID является идентификатором заказа, поле UserID, ссылающееся на поле ID таблицы «Users», содержит ID пользователя, совершившего заказ, поле ProductID, ссылающееся на поле ID таблицы «Product», поле Count содержит информацию о количестве заказанного товара, а поле OrderState – о текущем статусе заказа (В обработке, отправлен, доставлен). Поле SumPrice определяет сумму заказа, а поле Cart – карту для оплаты.

Таблица «Reviews» содержит данные об отзывах пользователей на товары. Таблица включает в себя: первичный ключ ReviewID, поле UserID, ссылающееся на поле ID таблицы «Users», содержит ID пользователя, поле ProductID, ссылающееся на поле ID таблицы «Product», поле UserRating содержащее оценку пользователя для данного товара и поле ReviewContent определяющее содержание отзыва.



## 4. Проектирование программного средства

### 4.1 Реализация архитектуры MVVM

Для реализации паттерна MVVM файлы программы были распределены по соответствующим директориям и реализовали соответствующие функции. Разделение проекта на логические модули представлено на рисунке 4.1.

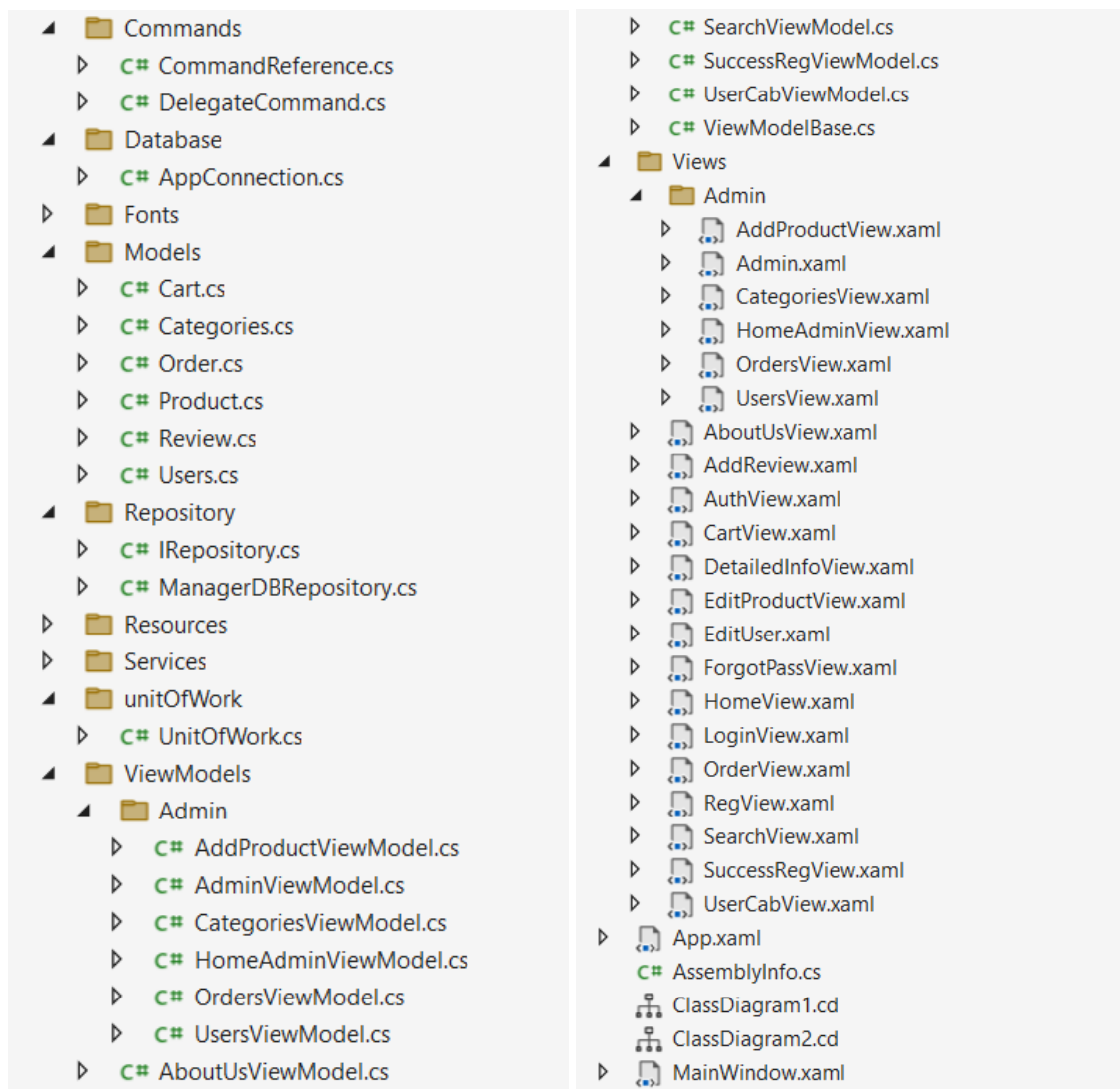


Рисунок 4.1 – Логические модули проекта

Архитектура MVVM (Model-View-ViewModel) является одним из популярных паттернов архитектуры для проектов, связанных с разработкой пользовательских интерфейсов. Модель представляет данные в приложении, Представление отображает данные пользователю, а ViewModel является связующим звеном между Моделью и Представлением, обрабатывая логику, связанную с UI и взаимодействуя с Моделью для запроса и обновления данных.

## 4.2 Реализация классов Models и ViewModels

В папке Models расположены сущностные классы, которые используются для создания БД. Этими классами являются Cart, Categories, Order, Product, User, Review. Пример реализации класса User представлен на рисунке 4.2.

```
namespace shop.Models
{
    Ссылка: 29
    public class Users
    {
        [Key]
        Ссылка: 8
        public int ID { get; set; }
        [StringLength(50, MinimumLength = 2, ErrorMessage = "Имя должно быть не менее 2 и не более 50 символов")]
        [RegularExpression(@"^[A-ЯЁА-Z]{1}[a-яёа-z]+[A-ЯЁА-Z]{1}[a-яёа-z]+$", ErrorMessage = "Имя должно начинаться с большой буквы")]
        Ссылка: 5
        public string Name { get; set; } = string.Empty;
        [StringLength(50, MinimumLength = 2, ErrorMessage = "Фамилия должна быть не менее 2 и не более 50 символов")]
        [RegularExpression(@"^[A-ЯЁА-Z]{1}[a-яёа-z]+[A-ЯЁА-Z]{1}[a-яёа-z]+$", ErrorMessage = "Фамилия должна начинаться с большой буквы")]
        Ссылка: 5
        public string Surname { get; set; } = string.Empty;
        [StringLength(20, MinimumLength = 5, ErrorMessage = "Логин должно быть не менее 5 и не более 20 символов")]
        [RegularExpression(@"^[A-Za-z0-9]+$", ErrorMessage = "Логин должен состоять только из английских символов и цифр")]
        Ссылка: 15
        public string Login { get; set; } = string.Empty;
        Ссылка: 7
        public string Password { get; set; } = string.Empty;
        [RegularExpression(@"^[a-zA-z0-9_-]+\.[a-zA-z0-9_-]+@[a-zA-z0-9_-]+\.[a-zA-z]{2,6}$", ErrorMessage = "Неверный формат email")]
        Ссылка: 9
        public string Email { get; set; } = string.Empty;
        [RegularExpression(@"^\+375\d{9}$", ErrorMessage = "Телефон должен быть в формате +375XXXXXXXXXX")]
        Ссылка: 5
        public string Phone { get; set; } = string.Empty;
        Ссылка: 2
        public string Photo { get; set; } = string.Empty;
    }
}
```

Рисунок 4.2 – Содержимое класса Users

Базовым классом для всех моделей представлений является ViewModelBase, который реализует интерфейс INotifyPropertyChanged. На рисунке 4.3 представлено содержимое класса ViewModelBase.

```
public abstract class ViewModelBase : INotifyPropertyChanged
{
    public event PropertyChangedEventHandler PropertyChanged;

    protected void OnPropertyChanged(string propertyName)
    {
        PropertyChangedEventHandler handler = PropertyChanged;

        if (handler != null)
        {
            handler(this, new
            PropertyChangedEventArgs(propertyName));
        }
    }
}
```

Листинг 4.1 – Класс ViewModelBase

Классы с суффиксом ViewModel содержат: закрытые поля, открытые свойства и команды, привязываемые к элементам управления на страницах View. ViewModel также содержит логику по получению данных из модели, которые потом передаются

в представление. И также ViewModel определяет логику по обновлению данных в модели. Диаграмма классов представлена в приложении Б.

### 4.3 Реализация паттерна Command

В приложении используется паттерн Command который позволяет инкапсулировать запрос на выполнение определенного действия в виде отдельного объекта. В WPF команды представлены интерфейсом ICommand. В приложении он представлен в виде двух классов: CommandReference и DelegateCommand.

Класс DelegateCommand позволяет делегировать командную логику методами, передаваемыми в качестве параметров, и позволяет View связывать команды с объектами, которые не являются частью дерева элементов. Также содержит в себе класс CommandManagerHelper, содержащий методы для CommandManager, которые помогают избежать утечки памяти, используя слабые ссылки.

Класс CommandReference облегчает привязку ключа в разметке XAML к команде. Определяется в модели представления путем предоставления свойства зависимости от команды. Класс является производным от Freezable, чтобы обойти ограничение в WPF, когда привязка данных из XAML.

Внедрение паттерна Command в программное средство позволяет достичь лучшей структурированности и гибкости кода, упрощает поддержку и расширение функциональности, а также облегчает взаимодействие между различными компонентами приложения.

### 4.3 Реализация подключения к БД и паттерны Repository и UnitOfWork

В любом приложении, работающем с БД через Entity Framework Core, необходимо использовать контекст (класс производный от DbContext) и набор данных DbSet, через который можно взаимодействовать с таблицами из БД. В данном случае таким контекстом является класс AppConnection.

В проекте реализовано 2 паттерна для работы с базой данных. В папке Repository расположен класс ManagerDBRepository и интерфейс IRepository. В папке UnitOfWork расположен одноименный класс.

Класс ManagerDBRepository является реализацией паттерна репозиторий. Репозиторий позволяет абстрагироваться от конкретных подключений к источникам данных, с которыми работает программа, и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой. Листинг класса представлен в Приложении Г.

Класс UnitOfWork является реализацией паттерна UnitOfWork для приложения. Паттерн UnitofWork позволяет упростить работу с различными репозиториями и дает уверенность, что все репозитории будут использовать один и тот же контекст данных. Листинг класса представлен в Приложении Д.

Пример создания экземпляра товара с использованием паттернов Command, UnitOfWork и Repository представлен в приложении Е.

## 4.5 Реализация вспомогательных классов и методов

В проекте было реализовано несколько дополнительных вспомогательных классов для расширения функционала приложения.

Для безопасности хранения информации используется хэширование, в данном случае применяется встроенный класс MD5. Код метода хэширования представлен в классе SecurePassService, который продемонстрирован на рисунке 4.5.

```
public class SecurePassService
{
    public static string Hash(string input)
    {
        byte[] hash = Encoding.ASCII.GetBytes(input);
        MD5 md5 = new MD5CryptoServiceProvider();
        byte[] hashenc = md5.ComputeHash(hash);
        string output = "";
        foreach (var b in hashenc)
        {
            output += b.ToString("x2");
        }
        return output;
    }
}
```

Листинг 4.2 – Метод обеспечивающий хэширование пароля

Для отправки сообщений на электронную почту пользователей используется класс EmailSenderService, представленный на листинге 4.3.

```
public static async Task SendEmail(string UserMail, string Subject,
string Text)
{
    MailAddress from = new
MailAddress("rwrkdwywencka2@gmail.com", "Shop ToysNow");
    MailAddress to = new MailAddress(UserMail);
    MailMessage message = new MailMessage(from, to);
    message.Subject = Subject;
    message.Body = Text;
    SmtplibClient smtp = new SmtplibClient("smtp.gmail.com", 587);
    smtp.Credentials = new
NetworkCredential("rwrkdwywencka2@gmail.com", "nyacxvedyinsiczi");
    smtp.EnableSsl = true;
    smtp.DeliveryMethod = SmtplibDeliveryMethod.Network;
    await smtp.SendMailAsync(message);
}
```

Листинг 4.3 – Метод для отправки email

Данный метод использует объекты классов SmtplibClient и MailMessage для настройки и отправки письма. В нем указываются адрес получателя и отправителя, тема письма, текст сообщения и другие необходимые параметры.

Использование классов из пространства имен System.Net.Mail позволяет нам легко интегрировать функциональность отправки электронных писем в наше программное средство, обеспечивая удобство и надежность взаимодействия с пользователями через почтовый сервис.

Для выполнения валидации и получения результатов были использованы классы ValidationResult, Validator и ValidationContext. Класс ValidationResult содержит информацию о результате валидации, включая список ошибок при нарушении правил. Класс Validator предоставляет методы для выполнения валидации объектов, а класс ValidationContext представляет контекст валидации, в котором определены правила и параметры валидации. В листинге 4.4 представлен метод для проверки вводимых значений при создании товара.

```
bool AddProduct(string Title, Category category, string Price,
string Description, string ImageLink)
{
    var product = new Product
    {
        Title = Title,
        Category = Category,
        Price = Convert.ToDouble(Price),
        Description = Description,
        ImageLink = ImageLink,
        Rating = 0
    };
    var results = new
List<System.ComponentModel.DataAnnotations.ValidationResult>();
    var context = new ValidationContext(product);
    if (Validator.TryValidateObject(product, context,
results, true))
    {
        ErrorMessage = String.Empty;
        return true;
    }
    else
    {
        foreach (var error in results)
        {
            ErrorMessage = error.ErrorMessage;
        }
    }
    return false;
}
```

Листинг 4.4 – Метод для проверки корректности введенных данных

В этом методе используются атрибуты валидации, которые определены для свойств модели данных товара. После выполнения валидации метод возвращает результат в виде объекта ValidationResult, который содержит информацию о корректности вводимых значений.

## 4.6 Реализация Views

Для разработки графической части приложения была выбрана технология WPF.

Windows Presentation Foundation (WPF) — это библиотека для создания пользовательских интерфейсов для интеллектуальных клиентских приложений

Одной из важных особенностей WPF является использование языка декларативной разметки интерфейса XAML, основанного на XML. Разработка с использованием XAML позволяет отделить графический интерфейс от логики приложения, а также создавать насыщенный интерфейс, используя или декларативное объявление интерфейса, или код на управляемых языках C#.

В результате применения WPF в приложении были реализованы три основных окна: стартовое окно, главное окно пользователя и окно администратора.

А также одиннадцать страниц:

- 1) страница для авторизации;
- 2) страница для регистрации;
- 3) главная страница с категориями товаров;
- 4) страница с контактной информацией о магазине;
- 5) страница с каталогом товаров;
- 6) страница личного кабинета пользователя;
- 7) страница корзины товаров;
- 8) страница управления категориями товаров;
- 9) страница управления товарами;
- 10) страница для просмотра пользователей;
- 11) страница для просмотра заказов.

В итоге успешно выполненного этапа разработки было создано функционирующее программное средство, обладающее графическим интерфейсом, способным удовлетворить требования пользователя и предоставить удобный опыт использования.



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

Основной целью тестирования приложения было стремление доказать невозможно введения пользователем данных, которые бы могли привести приложение в неработоспособное состояние. Были проведены:

- 1) тестирование авторизации и регистрации;
- 2) тестирование управления товарами и категориями;
- 3) тестирование функциональности поиска, фильтрации, сортировки товаров;
- 4) тестирование корзины и оформления заказа;
- 5) тестирование страницы карточки товара;
- 6) тестирование оплаты и доставки.

Упомянутое тестирование будет подробно описано в последующих подпунктах, где будут представлены результаты и анализ каждого этапа тестирования.

### 5.1 Тестирование авторизации и регистрации

В момент регистрации возможна ситуация, когда пользователь вводит некорректные данные. Предусмотрены ограничения по количеству вводимых символов. Имя и фамилия должны быть в диапазоне от 2 до 50 символов, пароль – от 5 до 20 символов, логин – от 5 до 20 символов. Логин должен быть указан латиницей и может содержать цифры. Также логин должен быть уникальный в системе. При вводе некорректных данных будет сгенерировано исключение. Обработка данного исключения продемонстрирована на рисунке 5.1.

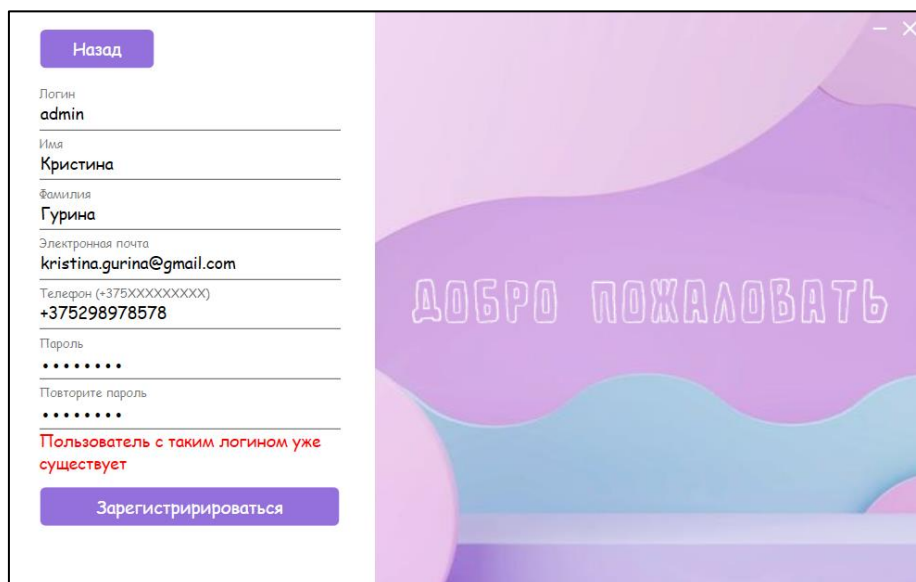
The image shows a web application registration form. On the left, there is a white sidebar with a purple 'Назад' button at the top. Below it are input fields for 'Логин' (containing 'admin'), 'Имя' (containing 'Кристина'), 'Фамилия' (containing 'Гурина'), 'Электронная почта' (containing 'kristina.gurina@gmail.com'), 'Телефон' (containing '+375XXXXXXXXXX'), 'Пароль' (masked with dots), and 'Повторите пароль' (masked with dots). A red error message is displayed below the password fields: 'Пользователь с таким логином уже существует'. At the bottom of the sidebar is a purple button labeled 'Зарегистрироваться'. The main area of the form has a purple and blue wavy background with the text 'ДОБРО ПОЖАЛОВАТЬ' in white, outlined letters.

Рисунок 5.1 – Демонстрация обработки исключения

При регистрации пользователю необходимо ввести повторно пароль, в случае отличия паролей будет выведено сообщение с ошибкой. Номер мобильного телефона и электронная почта должны соответствовать требуемому формату. В

случае некорректного ввода также будет сгенерировано соответствующее исключение.

При попытке ввести логин несуществующего пользователя на странице «Забыли пароль» будет выведено сообщение с ошибкой, представленной на рисунке 5.2.

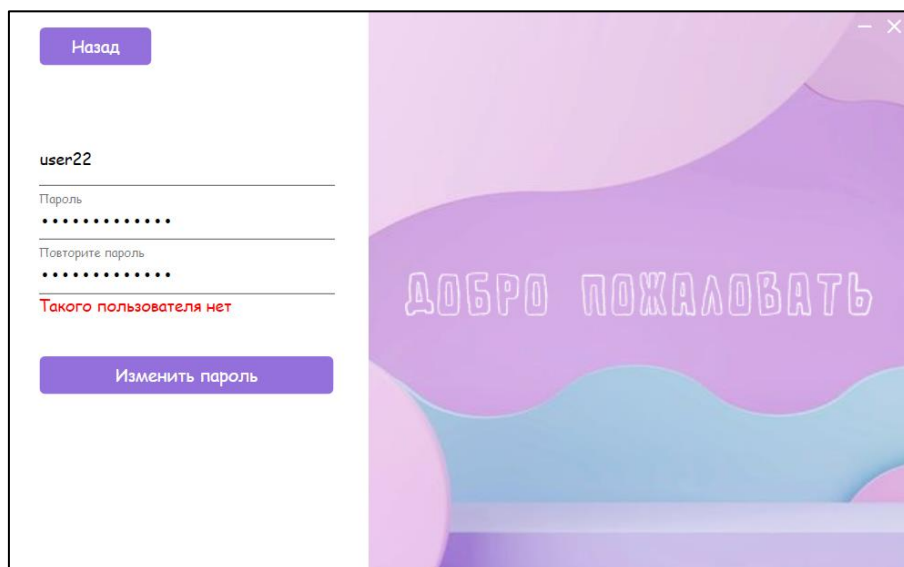


Рисунок 5.2 – Демонстрация обработки исключения

Предупреждение о неверно введённом пароле или логине продемонстрировано на рисунке 5.3.

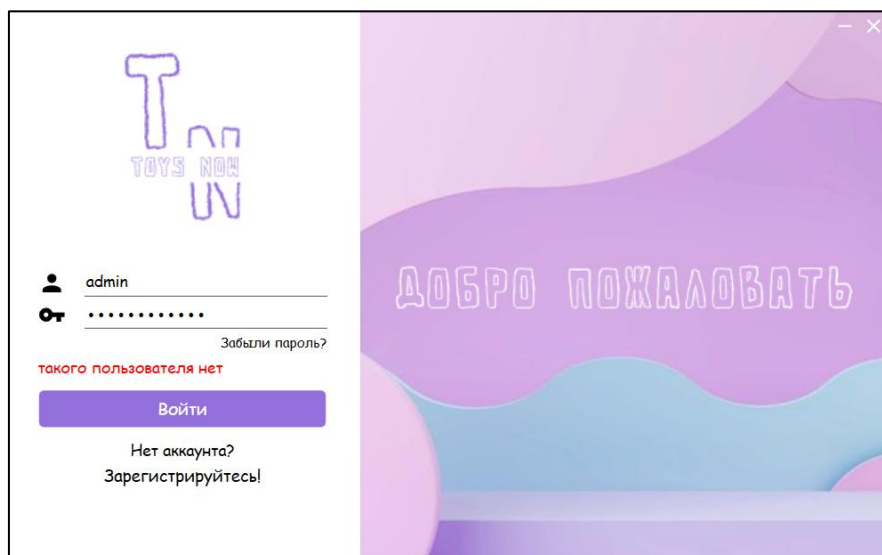


Рисунок 5.3 – Демонстрация обработки исключения

Тестирование авторизации и регистрации подтвердило, что функциональность этих модулей соответствует ожиданиям пользователей. Были проверены различные сценарии, включая успешную авторизацию, регистрацию новых пользователей и обработку ошибок. Тестовые сценарии успешно пройдены без критических проблем или ошибок. Вход в систему и регистрация работают

стабильно и надежно, обеспечивая безопасность данных пользователей. Результаты тестирования гарантируют безопасный и удобный процесс входа и регистрации в приложении.

## 5.2 Тестирование функций администратора

Этап тестирования функций администратора включал проверку основных операций, таких как управление товарами и категориями (добавление, удаление и редактирование), поиск, изменение статуса заказа и просмотр пользователей.

В процессе тестирования функции управления товарами, была проверена возможность успешного добавления новых товаров в систему, их корректное отображение в каталоге и возможность редактирования уже существующих товаров. При вводе некорректных данных будет сгенерировано исключение. Обработка данного исключения продемонстрирована на рисунке 5.4.

ID	TITLE	CATEGORY	PRICE	Rating	DISCRIP...
1	Медведь	Мягие...	155	0	описани...
2	Медведь 2	Мягие...	155	9	описани...
3	Медведь 3	Мягие...	255	0	описани...
4	Медведь 4	Мягие...	150	0	описани...
5	Медведь 5	Мягие...	105	0	описани...
6	Медведь 6	Мягие...	15	0	описани...
7	Медведь 7	Мягие...	19	0	описани...
8	Медведь 8	Мягие...	49	0	описани...
9	Настолка1	Настоль...	155	0	описани...
10	Настолка...	Настоль...	155	0	описани...
11	Настолка...	Настоль...	155	0	описани...
12	Настолка1	Настоль...	155	0	описани...
13	Настолка1	Настоль...	155	0	описани...
14	Настолка1	Настоль...	155	0	описани...
23	dsfdsfds	Настоль...	1000	0	dggfdsfsf
24	тестик	Мягие...	123	0	мприол

Рисунок 5.4 – Демонстрация обработки исключения

Также была проверена функция удаления товаров, удостоверившись в правильном удалении товара из базы данных.

Тестирование функции изменения статуса заказа включало проверку возможности изменения статуса заказа на различные значения («В обработке», «Отправлен» и «Доставлен»). Было также убеждено, что изменение статуса заказа корректно отражается в системе и соответствующая информация видна пользователю. Результат представлен на рисунке 5.5.

<div> Категории Товары Пользователи заказы Пользовательский каталог </div>								
ID	ORDERSTATE	USER	PRODUCT	CO...	Su...	DATE	ADDRESS	PAY
1	В обработке	2	Медведь 2	2	310	5/15/2023...	г Минск, ул...	1111111111111...
2	Доставлен	2	Медведь 3	7	155	5/15/2023...	г Минск, ул...	1111111111111...
3	Доставлен	2	Медведь 4	1	155	5/15/2023...	г Брест, ул...	1111111111111...
4	Отправлен	2	Медведь	1	155	5/15/2023...	г Берёза, ул...	1111111111111...
5	Доставлен	2	Медведь	7	1085	5/17/2023...	Самовывоз	При получении
<div> В обработке Отправлен Доставлен </div>								

Рисунок 5.5 – Демонстрация работоспособности изменения статуса заказа

Функция просмотра и поиска пользователей была протестирована для убеждения в возможности просмотра списка зарегистрированных пользователей, их контактной информации и других соответствующих данных. Результат представлен на рисунке 5.6

<div> Категории Товары Пользователи заказы Пользовательский каталог </div>					
ID	NAME	SURNAME	LOGIN	EMAIL	PHONE
3	user2	user2	user2	user2@gmail.com	+375297723744
<div> <input type="text" value="user"/> <input type="button" value="поиск"/> </div>					

Рисунок 5.6 – Демонстрация работы поиска

В результате тестирования функций администратора было подтверждено, что все указанные функции работают корректно и соответствуют требованиям.

## 5.2 Тестирование функций пользователя

В ходе тестирования функциональности поиска, фильтрации и сортировки товаров была проверена возможность точного поиска товаров по заданным критериям, правильная фильтрация товаров по различным атрибутам (например, цене, категории) и корректная сортировка товаров по заданным параметрам (например, по возрастанию или убыванию цены). Результаты поиска, фильтрации и сортировки, представленные на рисунке 5.7, соответствуют ожиданиям пользователя.

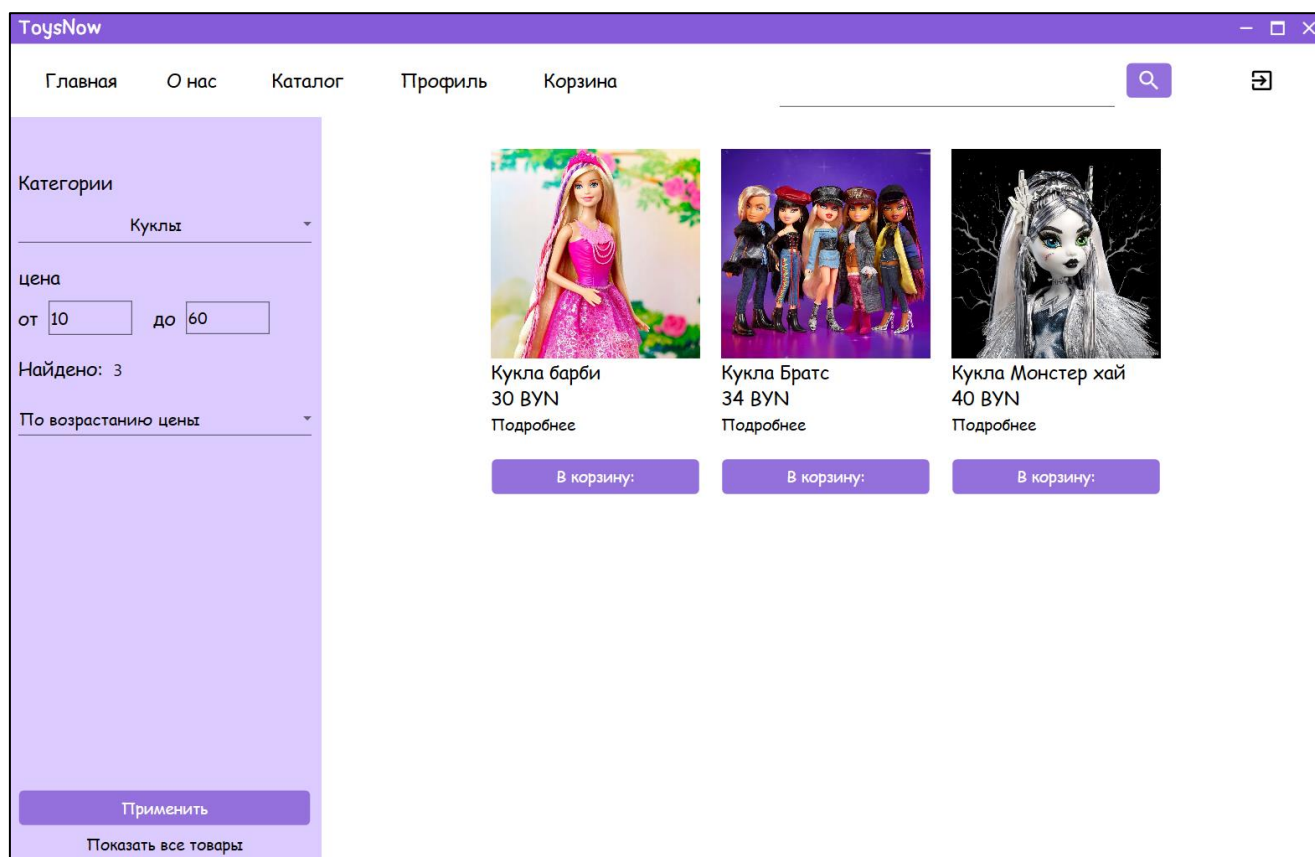


Рисунок 5.7 – Демонстрация корректной работы поиска

Тестирование функциональности корзины и оформления заказа включало проверку возможности добавления товаров в корзину, корректное отображение выбранных товаров и их количества, а также успешное оформление заказа с правильным учетом выбранных товаров, адреса доставки и способа оплаты. Все изменения с товарами происходили корректно, при изменении товара количество и стоимость изменяются корректно, что можно увидеть на рисунке 5.8.

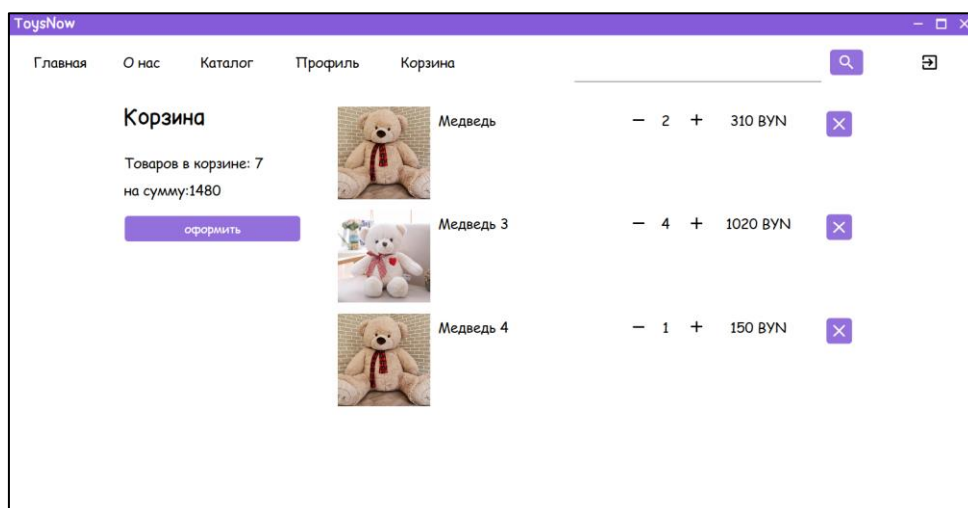


Рисунок 5.8 – Демонстрация корректной работы функций корзины

Тестирование страницы карточки товара включало проверку правильного отображения информации о товаре, его изображения, описания и других атрибутов. Добавления товара в корзину происходит корректно.

Тестирование оплаты и доставки включало проверку возможности успешной оплаты заказа, правильного отображения информации о способах оплаты и доставки. При вводе некорректного номера карты выведено соответствующее сообщение, что отображено на рисунке 5.9.

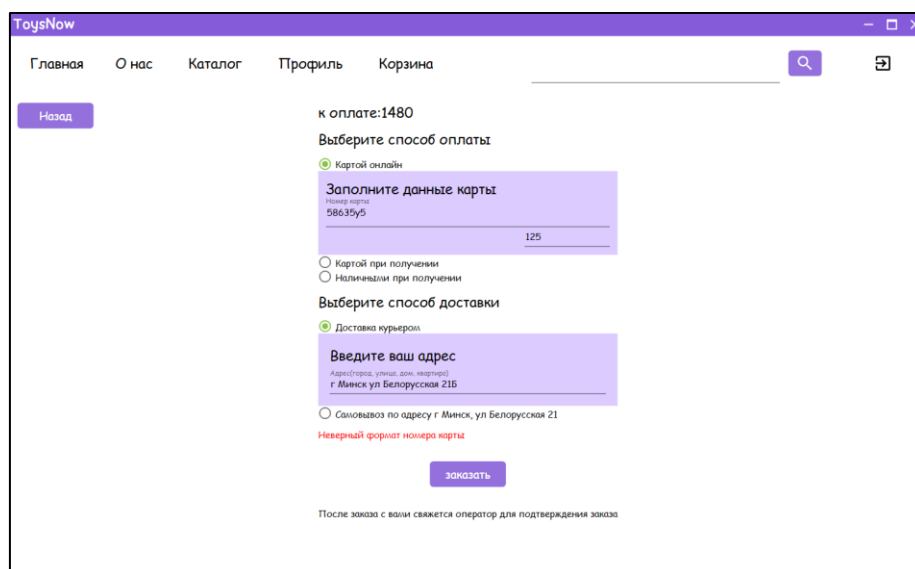


Рисунок 5.9 – Демонстрация обработки исключения

Процесс оплаты и доставки проходит без проблем, и информация сохраняется корректно.

В результате тестирования было подтверждено, что все они работают корректно, соответствуют требованиям и обеспечивают удобство использования для пользователя.



## 6. Руководство по использованию

Окно входа содержит два поля для ввода логина и пароля. После ввода верных данных, необходимо нажать на кнопку вход. Окно входа в аккаунт представлено на рисунке 6.1.

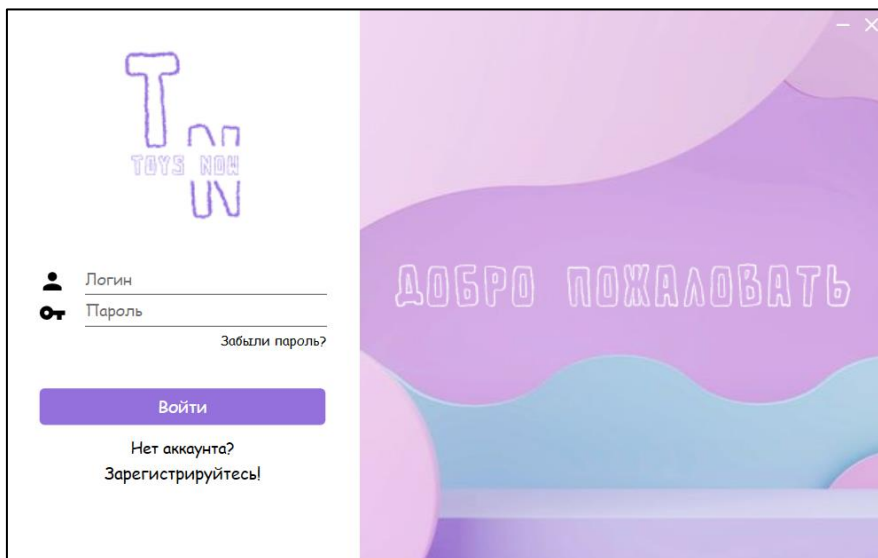


Рисунок 6.1 – Окно входа в аккаунт

Если пользователь забыл пароль, то по нажатию на кнопку «Забыли пароль» откроется страница, представленная на рисунке 6.2. Для смены пароля необходимо ввести логин и новый пароль.

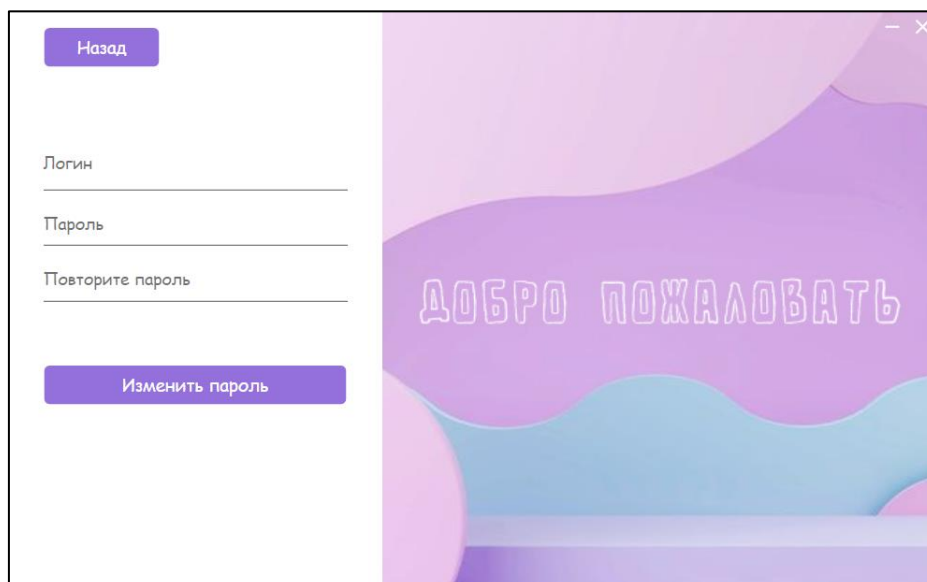


Рисунок 6.2 – Окно входа в аккаунт

Окно регистрации представлено на рисунке 6.3.

Рисунок 6.2 – Окно регистрации

На поля регистрации накладываются следующие ограничения:

- 1) имя и фамилия: должны содержать только буквы, должны начинаться с заглавной буквы, длина должна быть от 2 до 30 символов;
- 2) логин: длина должна быть от 5 до 20 символов, должен быть уникальным в системе, может содержать только английские буквы, цифры, дефисы и подчеркивания;
- 3) номер телефона и почта: должны соответствовать формату;
- 4) пароль: длина должна быть от 5 до 20 символов, может содержать буквы английского алфавита, цифры, нижнее подчеркивание и дефис;

После успешной авторизации открывается главная страница, изображенная на рисунке 6.3.

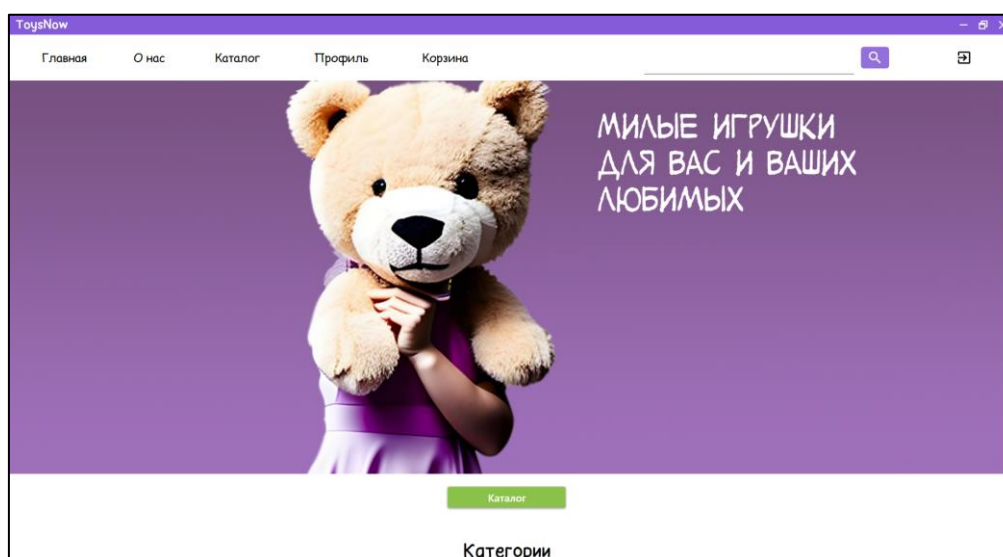


Рисунок 6.3 – Главная страница

Доступна навигация по страницам «О нас», «Каталог», «Профиль», «Корзина». Ниже на главной странице расположены категории товаров, что можно

увидеть на рисунке 6.4. Для отображения товаров определенной категории необходимо нажать на соответствующую категорию.

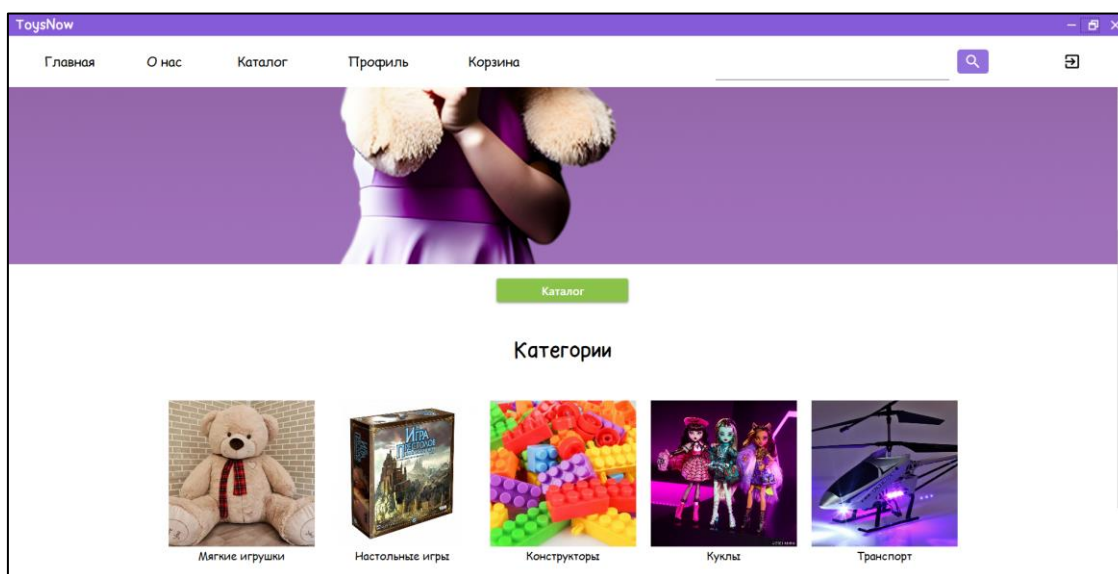


Рисунок 6.4 – Продолжение главной страницы

Страница «О нас» содержит контактную информацию интернет-магазина, ссылки на соцсети, поле для вопроса техподдержки приложения.

После введения поискового запроса пользователь попадает на страницу с результатами поиска, где он может просматривать товары и добавлять их в корзину, применять фильтры и сортировку. Страница с каталогом представлена на рисунке 6.5.

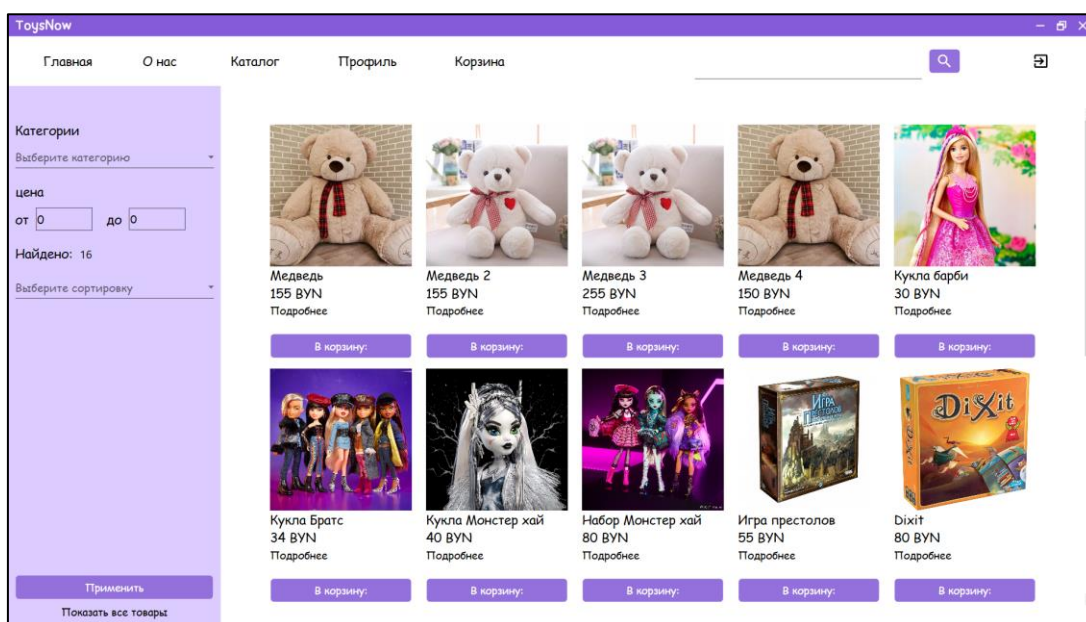


Рисунок 6.5 – Страница каталога товаров

По нажатию на карточку товара открывается персональная страница товара, на которой находятся подробная информация о товаре, представленная на

рисунке 6.6. Также имеется возможность добавить товар в корзину, просматривать комментарии других пользователей.

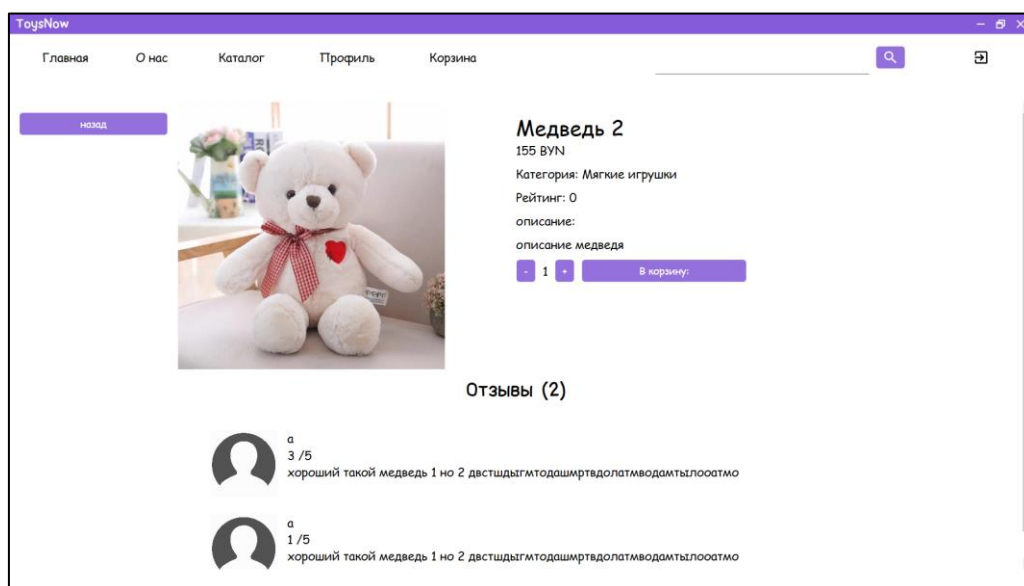


Рисунок 6.6 – Страница товара

Страница «Профиль» содержит персональные данные пользователя, которые он может изменить. На данной странице располагаются активные доставки, статус которых пользователь может просмотреть, а также список ранее заказанных товаров. Для заказанных товаров при нажатии на кнопку «Отзыв» может оставить комментарий и поставить оценку. Длина комментария ограничена 300 символами. На рисунке 6.7 изображена страница «Профиль».

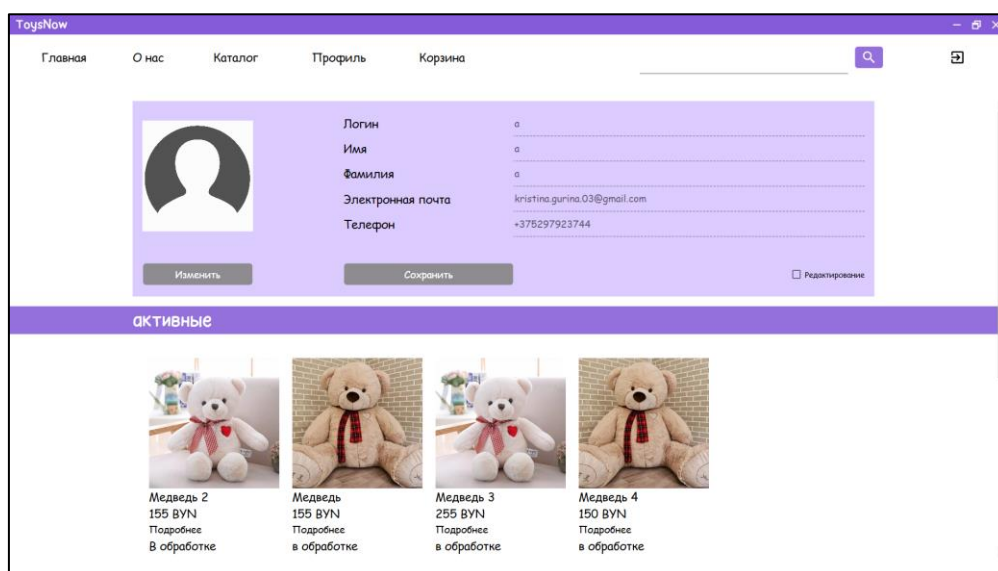


Рисунок 6.7 – Страница «Профиль»

На странице «Корзина» пользователь может изменять количество конкретного товара, удалять его, просмотреть итоговую стоимость покупки. Данная страница изображена на рисунке 6.8.

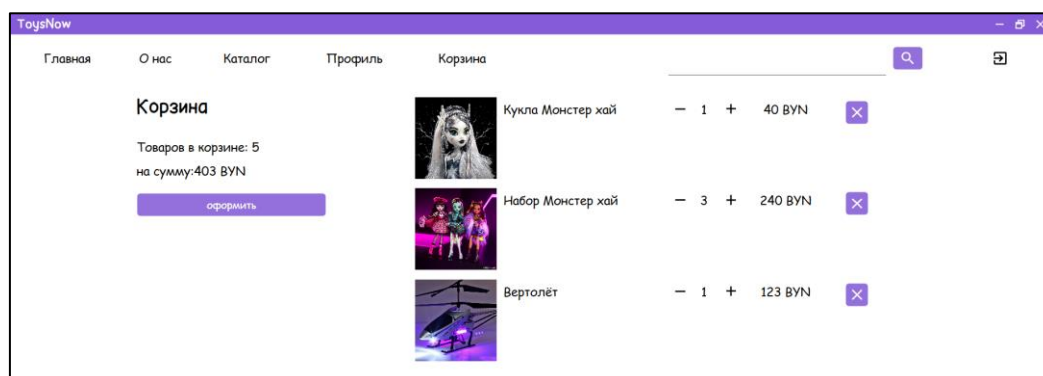


Рисунок 6.8 – Страница «Корзина»

По нажатию на кнопку оформить заказ пользователя перенаправляет на страницу, где он может выбрать удобный для себя способ оплаты и доставки. При выборе способа оплаты «Картой онлайн» откроется окно для привязки карты, в которой нужно будет ввести соответствующие данные. При выборе доставки курьером появится поле для ввода адреса доставки. На рисунке 6.9 продемонстрирована страница оформления заказа.

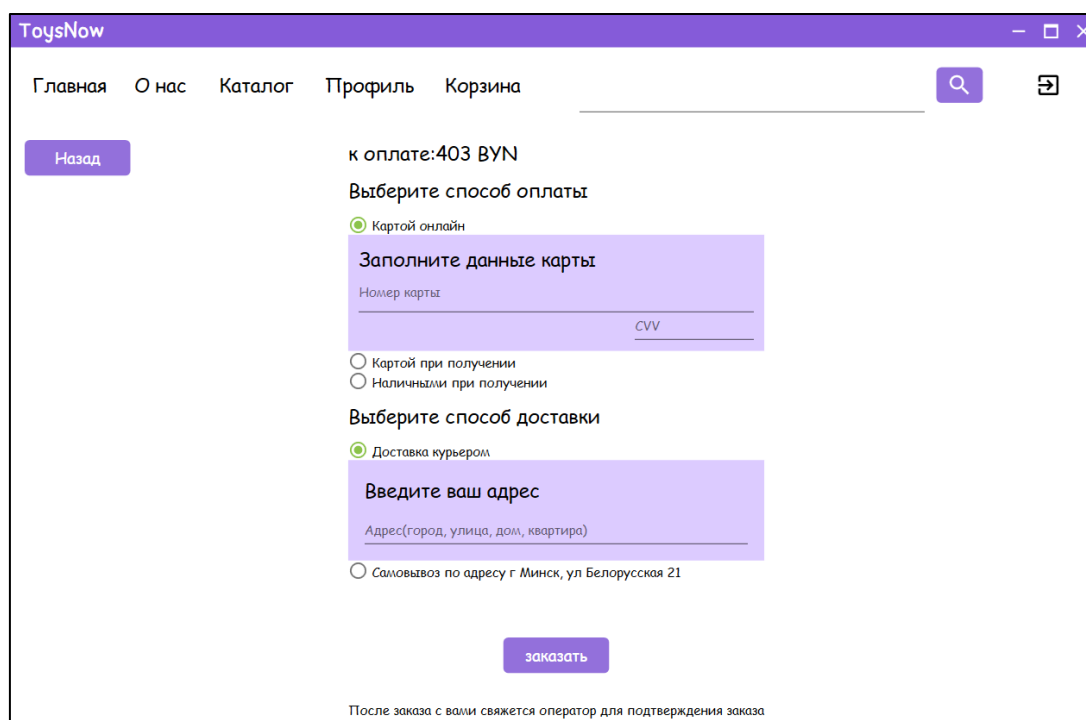


Рисунок 6.9 – Страница «Оформление заказа»

При входе в приложение под аккаунтом администратора присутствует возможность редактирования всех таблиц базы данных приложения, в чём и заключается особенность работы с приложением в роли администратора. Администратору доступны функции добавления, удаления, редактирования товаров и категорий. Страница администрирования представлена на рисунке 6.10.

Категории

Товары

Пользователи

заказы

Пользовательский каталог

Название

\_\_\_\_\_

Категория

\_\_\_\_\_

Цена

\_\_\_\_\_

Описание

\_\_\_\_\_

image link

Фото

Добавить

Удалить

Поиск

☐ Редактирован

Сохранить

ID	TITLE	CATEGORY	PRICE	Rating	DISCRIPTION
1	Медведь	Мякие игрушки	155	0	описание медведя
2	Медведь 2	Мякие игрушки	155	0	описание медведя
3	Медведь 3	Мякие игрушки	255	0	описание медведя
4	Медведь 4	Мякие игрушки	150	0	описание медведя
5	Кукла барби	Куклы	30	0	описание медведя
6	Кукла Братс	Куклы	34	0	описание медведя
7	Кукла Монстер хай	Куклы	40	0	описание медведя
8	Набор Монстер хай	Куклы	80	0	описание медведя
9	Игра престолов	Настольные игры	55	0	описание медведя
10	Dixit	Настольные игры	80	0	описание медведя
11	Монополия	Настольные игры	75	0	описание медведя
12	Монополия плюс	Настольные игры	155	0	описание медведя
13	Лото	Настольные игры	8	0	описание медведя
14	Машинка	Транспорт	145	0	описание медведя
23	Машинка 2	Транспорт	108	0	dggfdfsfsf
24	Вертолёт	Транспорт	123	0	мприол

Рисунок 6.10 – Окно администратора

Также администратор имеет возможность просматривать базу данных пользователей и изменять статус заказа.



## Заключение

Основной задачей данного проекта было усвоение основных технологий разработки десктопных приложений. Разработанное программное средство послужило практической реализацией знаний, полученных в результате изучения теоретических знаний работы с базами данных, технологиями WPF и Entity Framework Core.

В процессе решения поставленной задачи была достигнута поставленная цель по созданию приложения для магазина игрушек «TowsNow». Данное приложение может служить не только примером практического применения полученных знаний, но также представляет из себя готовый продукт, который может иметь практическую пользу при использовании его по назначению.

В программном средстве были реализованы следующие функции:

- 1) авторизация, регистрация и смена пароля;
- 2) выполнение поисковых запросов, фильтрации и сортировки;
- 3) добавление товара в корзину;
- 4) заполнение формы заказа;
- 5) изменение данных пользователя;
- 6) просмотр статуса заказа;
- 7) возможность оставлять отзывы на товары;
- 8) создание, редактирование и удаление товаров и категорий;
- 9) изменение статуса заказа;
- 10) осуществление технической поддержки пользователей;
- 11) оповещение клиента о выполнении заказа.

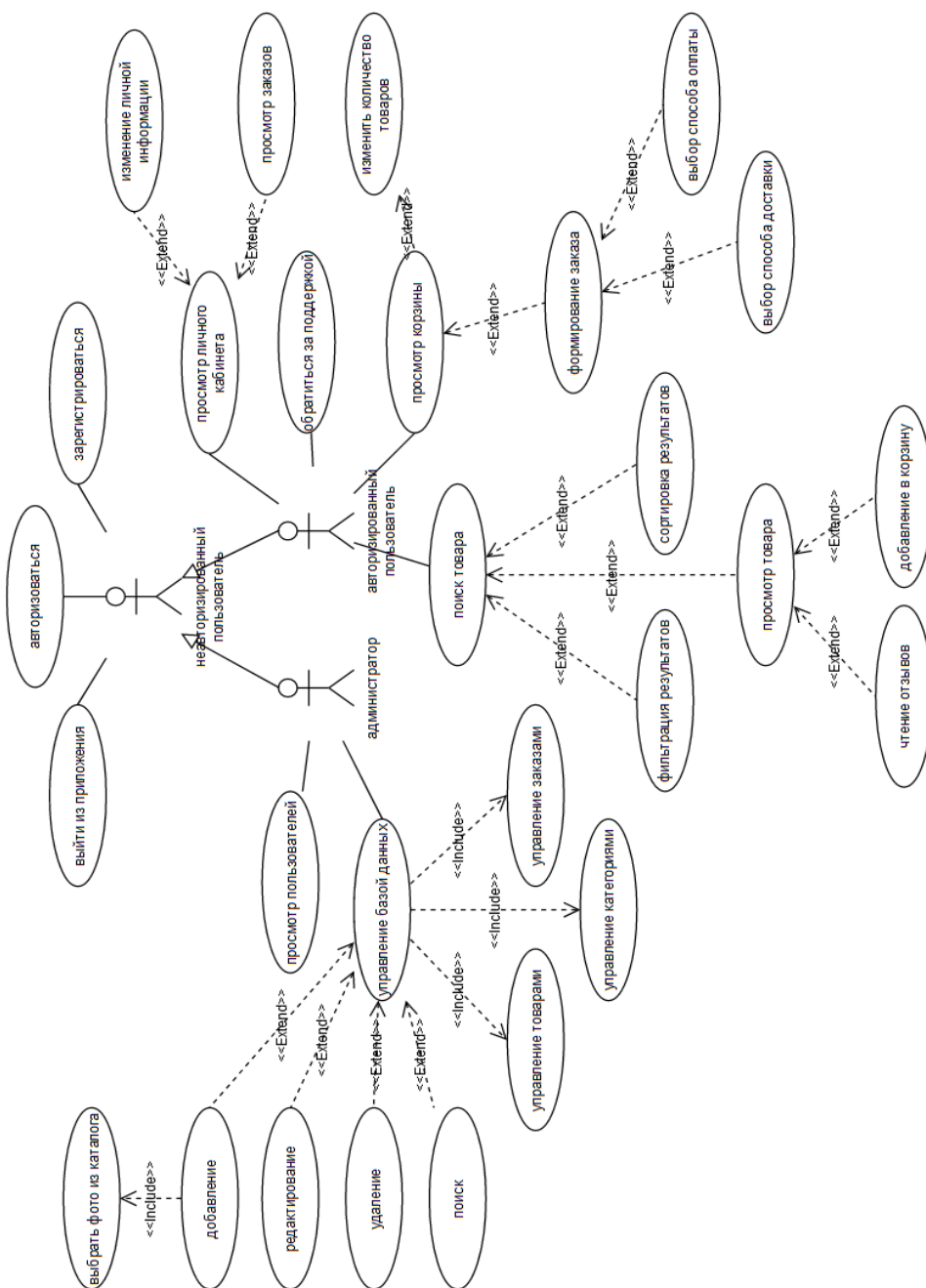
Разработанная программа имеет завершённый вид, работает верно и готова к использованию.

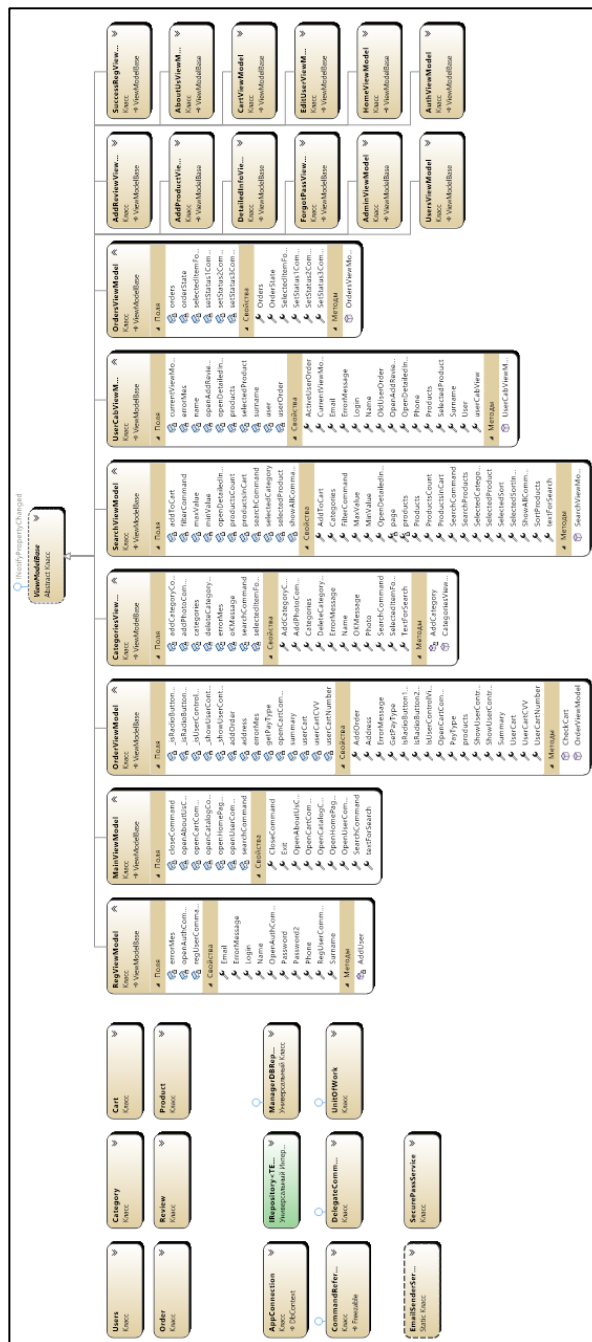
### Список литературы

1. Пацей, Н.В. Курс лекций по языку программирования C# / Н.В. Пацей. – Минск: БГТУ, 2016. – 175 с.
2. MSDN сеть разработчиков в Microsoft [Электронный ресурс] / Режим доступа: <http://msdn.microsoft.com/library/rus/> . Дата доступа: 20.04.2023
3. METANIT.COM Сайт о программировании [Электронный ресурс] / Режим доступа: <https://metanit.com> . Дата доступа: 20.04.2023
4. ProfessorWeb .NET & Web Programming [Электронный ресурс] / Режим доступа: <https://professorweb.ru> Дата доступа: 13.04.2023
- 5 Microsoft Docs Archived Content [Электронный ресурс]/ Режим доступа: <https://docs.microsoft.com/en-us/archive/> Дата доступа: 26.04.2023
- 6 Форум для программистов или разработчиков [Электронный ресурс] – <https://stackoverflow.com/> – Дата доступа: 5.05.2023
7. Блинова, Е.А. Курс лекций по Бадам данным / Е.А. Блинова. – Минск: БГТУ, 2019. – 175 с.

## Приложение А

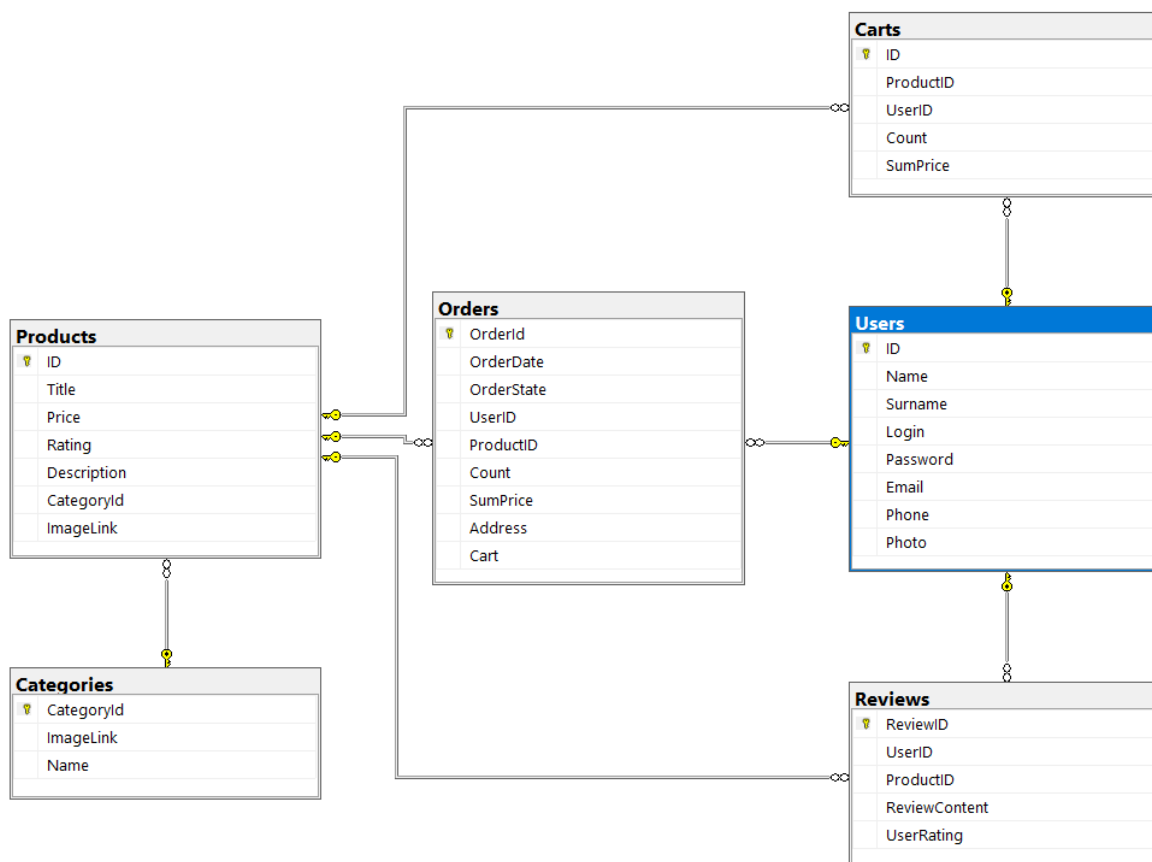
### Диаграмма вариантов использования





## Приложение В

### Структура базы данных



## Приложение Г

### Класс UnitOfWork

```
public class UnitOfWork : IDisposable
{
    private AppConnection db = new AppConnection();
    private ManagerDBRepository<Cart> cartRepository;
    private ManagerDBRepository<Category> categoryRepository;
    private ManagerDBRepository<Order> orderRepository;
    private ManagerDBRepository<Product> productRepository;
    private ManagerDBRepository<Review> reviewRepository;
    private ManagerDBRepository<Users> userRepository;

    public ManagerDBRepository<Cart> CartRepository
    {
        get
        {
            if (cartRepository == null)
                cartRepository = new
ManagerDBRepository<Cart>(db);
            return cartRepository;
        }
    }

    public ManagerDBRepository<Category> CategoryRepository
    {
        get
        {
            if (categoryRepository == null)
                categoryRepository = new
ManagerDBRepository<Category>(db);
            return categoryRepository;
        }
    }

    public ManagerDBRepository<Order> OrderRepository
    {
        get
        {
            if (orderRepository == null)
                orderRepository = new
ManagerDBRepository<Order>(db);
            return orderRepository;
        }
    }

    public ManagerDBRepository<Product> ProductRepository
    {
        get
        {
            if (productRepository == null)
```

```

        productRepository = new
ManagerDBRepository<Product>(db);
        return productRepository;
    }
}

public ManagerDBRepository<Review> ReviewRepository
{
    get
    {
        if (reviewRepository == null)
            reviewRepository = new
ManagerDBRepository<Review>(db);
        return reviewRepository;
    }
}

public ManagerDBRepository<Users> UserRepository
{
    get
    {
        if (userRepository == null)
            userRepository = new
ManagerDBRepository<Users>(db);
        return userRepository;
    }
}

public void Save()
{
    db.SaveChanges();
}

private bool disposed = false;

public virtual void Dispose(bool disposing)
{
    if (!this.disposed)
    {
        if (disposing)
        {
            db.Dispose();
        }
        this.disposed = true;
    }
}

public void Dispose()
{
    Dispose(true);
    GC.SuppressFinalize(this);
}
}

```

## Приложение Д

### Класс **ManagerDBRepository<TEntity>**

```
public class ManagerDBRepository<TEntity> : IRepository<TEntity>
where TEntity : class
{
    DbContext _context;
    DbSet<TEntity> _dbSet;

    public ManagerDBRepository(DbContext context)
    {
        _context = context;
        _dbSet = context.Set<TEntity>();
    }

    public IEnumerable<TEntity> Get()
    {
        return _dbSet.AsNoTracking().ToList();
    }

    public IEnumerable<TEntity> Get(Func<TEntity, bool>
predicate)
    {
        return _dbSet.AsNoTracking().Where(predicate).ToList();
    }

    public TEntity FindById(int id)
    {
        return _dbSet.Find(id);
    }

    public void Create(TEntity item)
    {
        _dbSet.Add(item);
    }
    public void Update(TEntity item)
    {
        _context.Entry(item).State = EntityState.Modified;
    }

    public void Remove(TEntity item)
    {
        _dbSet.Remove(item);
    }

    public IEnumerable<TEntity> GetWithInclude(params
Expression<Func<TEntity, object>>[] includeProperties)
    {
        return Include(includeProperties).ToList();
    }
}
```



```
        public IEnumerable<TEntity> GetWithInclude(Func<TEntity,
bool> predicate,
            params Expression<Func<TEntity, object>>[]
includeProperties)
        {
            var query = Include(includeProperties);
            return query.Where(predicate).ToList();
        }

        private IQueryable<TEntity> Include(params
Expression<Func<TEntity, object>>[] includeProperties)
        {
            IQueryable<TEntity> query = _dbSet.AsNoTracking();
            return includeProperties
                .Aggregate(query, (current, includeProperty) =>
current.Include(includeProperty));
        }

        public void RemoveRange(IEnumerable<TEntity> entities)
        {
            _dbSet.RemoveRange(entities);
        }
    }
```

## Приложение Е

### Команда для добавления товара

```
private DelegateCommand addProduceCommand;
public ICommand AddProductCommand
{
    get
    {
        if (addProduceCommand == null)
        {
            addProduceCommand = new DelegateCommand(() =>
            {
                try
                {
                    ErrorMessage = string.Empty;
                    if (Title != null || SelectedCategory !=
null || Price != null || Description != null || ImageLink != null)
                    {
                        if (AddProduct(Title, SelectedCategory,
Price, Description, ImageLink))
                        {
                            using (UnitOfWork unit = new
UnitOfWork())
                            {
                                Product product = new Product();
                                product.Title = Title;
                                product.Category =
unit.CategoryRepository.FindById(SelectedCategory.CategoryId) ;
                                product.Price =
Convert.ToDouble(Price);

                                product.Rating = 0;
                                product.Description =
Description;

                                product.ImageLink = ImageLink;

                                unit.ProductRepository.Create(product);
                                unit.Save();
                            }
                            ErrorMessage = string.Empty;
                            OKMessage = "товар добавлен";
                        }
                    }
                    else throw new Exception("все поля должны
быть заполнены");
                }
                catch (DbEntityValidationException e)
                {
                    foreach (DbEntityValidationResult
validationRes in e.EntityValidationErrors)
                    {
                        foreach (DbValidationError err in
validationRes.ValidationErrors)

```

```
        {
            OKMessage = string.Empty;
        }
        MessageBox.Show(err.ErrorMessage);
        ErrorMessage = err.ErrorMessage;
    }
}
catch (Exception e)
{
    OKMessage = string.Empty;
    ErrorMessage = e.Message;
}

});
}
return addProduceCommand;
}
}
```