

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение
высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ УТ

Факультет

вычислительной техники

Кафедра

МОиПЭВМ

Направление

09.03.02 «Информационные системы и технологии»

подготовки

БАКАЛАВРСКАЯ РАБОТА

на тему

**«Реализация клиент-серверного приложения учета и
анализа закупок товаров для информационной системы
магазина косметики»**

Студент

(подпись, дата)

Семенова Наталья Сергеевна

(ФИО полностью)

Руководитель

(подпись, дата)

Кольчугина Е.А.

(фамилия, инициалы)

Нормоконтролёр

(подпись, дата)

Такташкин Д.В.

(фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от _____ № _____)

Заведующий кафедрой

(подпись)

Козлов А.Ю.

(фамилия, инициалы)

Работа защищена с отметкой _____ (протокол заседания ГЭК от _____ № _____)

Секретарь ГЭК

(подпись)

Попова Н.А.

(фамилия, инициалы)

Пенза, 2024

Федеральное государственное бюджетное образовательное учреждение
высшего образования

ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет

Кафедра

Вычислительной техники

МО и ПЭВМ

«Утверждаю»

Заведующий кафедрой

А.Ю. Козлов

«25» декабря 2023 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Семенова Наталья Сергеевна гр. 20ВИ2 факультета вт
<Фамилия, имя, отчество полностью>

2. Тема работы «Реализация клиент-серверного приложения учета и анализа закупок товаров для информационной системы магазина косметики».

Тема утверждена приказом ПГУ № _____ от " _____ " _____ 20 _____

2. Руководитель работы Кольчугина Е.А.

3. Задание на работу (назначение разработки, исходные данные и т.п.)

Разработать клиент-серверное приложение учета и анализа закупок товаров для информационной системы магазина косметики. Среда и средства разработки: C#, ASP.NET, .NET Framework 4.6, MySqlLeech, MySql + phpMyAdmin. Среда функционирования: Windows 7 и выше. Основные функции: - учет поставки товаров, - учет реализации товаров, - прогноз продажи товаров, - ведение аналитики по реализации товаров. Особенности (особые функции, особые требования): Бесплатное

ПО, наличие API для получение информации об остатках, изменении количества товаров, добавления новых товаров; многопользовательское приложение

4. Перечень подлежащих разработке вопросов.

1. Анализ предметной области и постановка задачи. _____
2. Анализ требований на разработку. _____
3. Разработка программных средств. _____
4. Тестирование программных средств. _____
5. Оформление ВКР. _____

5. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области и постановка задачи	15%	08.04.2024-13.04.2024	
2.	Анализ требований на разработку	20%	15.04.2024-25.04.2024	
3.	Разработка программных средств	30%	26.04.2024-12.05.2024	
4.	Тестирование программных средств	20%	13.05.2024-22.05.2024	
5.	Оформление ВКР	15%	23.05.2024-03.06.2024	

Дата выдачи задания "21" декабря 2023г.

Руководитель бакалаврской работы _____

(подпись, дата)

Задание к исполнению принял студент _____

(подпись, дата)

Работу к защите допустить

Декан факультета _____

(подпись, дата)

Реферат

Пояснительная записка содержит 86 листов, 29 рисунков, 11 таблиц, 15 использованных источников, 7 листингов и 3 приложения.

АИС, ASP.NET, VISUAL STUDIO 2022, C#, ЗАКУПКИ, МАГАЗИН.

Предмет работы: разработка информационной системы автоматизации деятельности закупок магазина косметики.

Объект исследования: клиент серверное приложение, обеспечивающее деятельность закупок магазина косметики.

Цель работы: разработка информационной системы, которая автоматизирует деятельность закупок магазина косметики.

Средства разработки: C#, ASP.NET, .NET Framework 4.8, MySql.

Результаты работы: спроектировано, разработано, отлажено и протестировано клиент-серверное приложение учета и анализа закупок товаров для информационной системы магазина косметики.

Область применения: магазины, занимающиеся розничной реализацией косметики.

					ПГУ 09.03.02 – 08ВР202.12 ПЗ			
Изм.	Лист	№ докум.	Подпись	Дата	«Реализация клиент-серверного приложения учета и анализа закупок товаров для информационной системы магазина косметики» Пояснительная записка	Лит.	Лист	Листов
Разраб.		Семенова Н.С.						
Провер.		Кольчугина Е.А.					4	86
Реценз.						группа 20ВИ2		
Н. Контр.		Такташкин Д.В.						
Утверд.		Козлов А.Ю.						

Содержание

Введение.....	11
1 Аналитическая часть.....	12
1.1 Анализ предметной области	12
1.1.1 Требования к составу и параметрам технических средств ..	13
1.1.2 Описание функциональности контроля закупок магазина ..	13
1.2 Анализ информационных систем аналогичного назначения	14
1.2.1 «Подружка»	14
1.2.2 «Летуаль».....	14
1.2.3 «Золотое Яблоко»	17
1.2.4 Обоснование для разработки клиент серверного приложения системы закупок магазина косметики	19
1.3 Анализ требований к программной реализации.....	22
1.3.1 Список функциональных требований	22
1.3.2 Список нефункциональных требований	28
1.3.3 Требования к надежности	28
1.3.4 Условия эксплуатации.....	29
2 Проектирование программного обеспечения	30
2.1 Проектирование пользовательского интерфейса	30
2.2 Анализ бизнес процессов организации в нотации IDEF0	32
2.3 Проектирование модели базы данных.....	34
2.3.1 Проектирование физической и инфологической моделей базы данных с помощью Erwin	34
2.4 Проектирование поведения ИС	36
2.4.1 Диаграмма состояний.....	36
2.4.2 Диаграмма деятельности.....	37
2.4.3 Диаграмма последовательности.....	37
2.4.4 Диаграмма коммуникаций	38

2.4.5	Диаграмма компонентов	39
2.4.6	Диаграмма развертывания	39
3	Разработка приложения	41
3.1	Программные средства	41
4	Отладка и тестирование	44
4.1	Тестирование программного обеспечения	44
4.1.1	Метрики программного средства	48
4.1.2	Расчет метрик кода с помощью встроенного средства Visual Studio	48
4.1.3	Определение метрик компонента серверной части приложения	49
	Заключение	52
	Список использованных источников	53
	Приложение А (рекомендуемое) Диаграммы UML	55
	Приложение Б (рекомендуемое) Листинг программы	62
	Приложение В (рекомендуемое) Результаты тестирования	82

Введение

Успех любого бизнеса во многом зависит от того, насколько эффективно компания доводит свою информацию до клиентов и партнеров.

Создание клиент серверного – один из альтернативных методов позиционирования компании и информирования покупателя.

На сегодняшний день, Интернет является неотъемлемой частью нашей жизни. Именно в Интернете многие ищут информацию, на основе которой можно получить представление об интересующей компании, товарах и услугах, и многом другом.

Приложение анализа и контроля закупок товаров магазина – часть торгового предприятия торговой организации, предназначенная для предоставления покупателю посредством сети Интернет сведений, необходимых при совершении покупки.

С точки зрения функционирования или структуры закупок магазина можно сказать, что в сервисе размещены товары, обладающие такими характеристиками как: цена, описание, категория, изображение.

Добавление, удаление и изменение товара ведется в админ-панели с учетной записи администратора.

Все клиенты, желающие приобрести товар, должны просмотреть ассортимент, сформировать корзину и отправить заявку, после рассмотрения которой товар будет отправлен покупателю, а магазин закупит товар.

Таким образом, в функционирование закупок магазина входит:

- просмотр каталога товаров;
- формирование пользователем корзины товаров;
- отправка заявки на покупку товаров;
- добавление, удаление, изменение параметров товара

администратором, заказ нового товара.

1 Аналитическая часть

1.1 Анализ предметной области

Администратор системы контроля закупок магазина представляет собой лицо, занимающееся управлением закупками магазина. Он отвечает за добавление/удаление/изменение ассортимента товаров.

Проектируемая система должна использоваться для хранения информации о товарах, формированием пользовательской корзины, а также отправкой заявки на e-mail администратора. Система должна быть предназначена для пользователей и администраторов.

Система создается для обслуживания следующих групп пользователей:

- покупатель магазина – просматривает каталог товаров, формирует корзину и отправляет заявку на почту администратора;
- администратор контроля закупок магазина – отвечает за добавление/удаление/изменение ассортимента товаров.

Информационная система должна обеспечивать:

- ведение базы данных (запись, чтение, модификация, удаление);
- хранение информации о товарах, продающихся в магазине;
- временное хранение информации о содержимом корзины пользователя системы.

Цель разработки заключается в создании базы данных и клиентского приложения для хранения информации о товарах, пользовательской корзины и отправки заявок на покупку. Внедрение информационной системы призвано автоматизировать процесс обработки информации, что позволит увеличить число продаж товаров магазина, а также добавит удобство администратору для контроля закупками товаров магазина, учета количества товаров и их реализации.

1.1.1 Требования к составу и параметрам технических средств

Для работы с Web-приложением рекомендуются следующие программные и технические средства:

- операционная система семейства Windows/Linux;
- современный браузер (рекомендуются Google Chrome, Firefox);
- одноплатный процессор Intel или совместимый с частотой ядра не менее 500МГц;
- объем свободной оперативной памяти 512 Мб;
- объем необходимой памяти на жестком диске 500 Мб;
- стандартный VGA-монитор или совместимый, с поддерживаемым разрешением от 1024x768;
- 3D-ускоритель с объемом оперативной памяти больше 32 Мб;
- стандартная клавиатура;
- манипулятор «мышь».

1.1.2 Описание функциональности контроля закупок магазина

В настоящее время существует много информационных систем контроля закупок магазинов косметики.

Внедрение автоматизированной информационной системы контроля закупок магазина косметики позволит:

- снизить случайные ошибки связанных с ручным оформлением заказов;
- автоматизировать обработку информации в системе;
- получить доступ к каталогу продукции из любой точки, где есть компьютер, имеющий выход в Интернет;
- автоматизировать процесс учета и анализа продаж магазина косметики.

1.2 Анализ информационных систем аналогичного назначения

1.2.1 «Подружка»

Подружка – это система контроля закупок, разработанная специально для магазинов косметики и парфюмерии. Она предлагает широкий спектр функций, которые позволяют эффективно управлять ассортиментом, оптимизировать закупки и контролировать расходы. Система «Подружка» помогает магазинам, подобным «Подружке», удерживать лидирующие позиции на рынке и обеспечивать высокий уровень обслуживания клиентов. Система обладает широким функционалом для выполнения своих ключевых задач.

Функции управления ассортиментом:

- добавление и редактирование информации о товарах (название, описание, бренд, цена, остатки);
- категоризация товаров по брендам, типу продукта, назначению;
- установка минимального уровня запасов для автоматического формирования заказов поставщикам;
- анализ продаж и популярности товаров для оптимизации ассортимента.

Функции управления закупками:

- создание и отправка заказов поставщикам;
- отслеживание статуса заказов и сроков поставки;
- ведение базы данных поставщиков с контактной информацией и условиями сотрудничества;
- сравнение цен от разных поставщиков для выбора наиболее выгодных предложений.

Функции контроля расходов:

- установка бюджетов на закупки;

- мониторинг расходов по категориям товаров, поставщикам и периодам времени;
- анализ эффективности закупок и поиск возможностей для экономии.

Дополнительные функции:

- интеграция с кассовыми системами и системами учета товаров;
- генерация отчетов о продажах, закупках, остатках товаров;
- возможность работы с несколькими магазинами или складами.

Преимущества использования системы «Подружка»:

- оптимизация закупок: система помогает избежать избыточных запасов и дефицита товаров, что способствует снижению затрат и повышению эффективности бизнеса;
- контроль расходов: постоянный мониторинг расходов позволяет эффективно управлять бюджетом и принимать обоснованные финансовые решения;
- улучшение ассортимента: анализ продаж и спроса помогает сформировать оптимальный ассортимент, отвечающий потребностям клиентов;
- повышение эффективности: автоматизация рутинных задач высвобождает время сотрудников для решения более важных вопросов.

Система контроля закупок «Подружка» – это незаменимый инструмент для магазинов косметики и парфюмерии, которые стремятся к успеху и развитию. Благодаря широкому спектру функций и удобному интерфейсу, система «Подружка» помогает оптимизировать бизнес-процессы, повысить эффективность и увеличить прибыль.

1.2.2 «Летуаль»

Л'Этуаль – известная сеть магазинов парфюмерии и косметики, которая также использует собственную систему контроля закупок. Эта система

обладает рядом преимуществ, позволяющих управлять обширным ассортиментом и многочисленными магазинами по всей стране. Однако, как и любая другая система, она имеет свои недостатки. Система обладает широким функционалом для выполнения своих ключевых задач.

Функции управления ассортиментом:

- добавление и редактирование информации о товарах;
- категоризация товаров по брендам, типу, назначению;
- анализ популярности товаров и сезонных трендов;
- установка минимального уровня запасов для автоматического заказа.

Функции управления закупками:

- формирование заказов поставщикам с учетом потребностей магазинов и центрального склада;
- отслеживание статуса заказов и сроков поставки;
- ведение базы данных поставщиков;
- сравнение цен от разных поставщиков.

Функции контроля расходов:

- установка бюджетов на закупки для каждого магазина и сети в целом;
- мониторинг расходов по категориям товаров, поставщикам и периодам;
- анализ эффективности закупок и поиск возможностей для оптимизации.

Дополнительные функции:

- интеграция с кассовыми системами и программами лояльности;
- генерация отчетов о продажах, закупках, остатках товаров;
- управление доставкой товаров в магазины.

Недостатки системы контроля закупок Л'Этуаль:

- сложность системы: большое количество функций и настроек может затруднить работу с системой для новых сотрудников и потребовать дополнительного обучения;
- недостаточная гибкость: система может быть не достаточно гибкой для адаптации к быстро меняющимся условиям рынка и потребностям клиентов;
- интеграция с другими системами: могут возникать сложности с интеграцией системы контроля закупок с другими системами, используемыми в компании (например, с системами бухгалтерского учета);
- стоимость: внедрение и обслуживание системы контроля закупок может быть затратным для компании, особенно если речь идет о сети магазинов.

Несмотря на недостатки, система контроля закупок Л'Этуаль позволяет эффективно управлять закупками для крупной сети магазинов. Однако, компании стоит обратить внимание на упрощение системы, повышение ее гибкости и улучшение интеграции с другими системам.

1.2.3 «Золотое Яблоко»

Золотое Яблоко – сеть магазинов парфюмерии и косметики класса люкс, которая использует систему контроля закупок, адаптированную к специфике работы с премиальными брендами. Система обладает широким функционалом для выполнения своих ключевых задач.

Функции управления ассортиментом:

- фокус на селективный ассортимент премиальных и нишевых брендов;
- отслеживание новинок и лимитированных коллекций;
- работа с эксклюзивными контрактами и ограниченными партиями товаров.

Функции управления закупками:

- прямое сотрудничество с производителями и официальными дистрибьюторами;
- строгий контроль качества продукции;
- учет специфических требований брендов к хранению и транспортировке товаров.

Функции контроля расходов:

- управление бюджетом с учетом высокой стоимости товаров;
- анализ эффективности закупок с фокусом на рентабельность и оборачиваемость товаров.

Дополнительные функции:

- интеграция с CRM-системой для персонализированного обслуживания клиентов;
- организация мероприятий и презентаций новых продуктов;
- обучение персонала по особенностям премиальных брендов.

Преимущества системы контроля закупок Золотое Яблоко:

- экспертиза в премиальном сегменте: система учитывает специфику работы с люксовыми брендами и обеспечивает высокое качество продукции;
- эксклюзивный ассортимент: позволяет предложить клиентам уникальные товары, недоступные в других магазинах;
- персонализированный подход: интеграция с CRM-системой способствует построению долгосрочных отношений с клиентами.

Недостатки системы контроля закупок Золотое Яблоко:

- высокая стоимость закупок: работа с премиальными брендами связана с высокими затратами, что требует эффективного управления бюджетом;
- сложность логистики: учет специфических требований брендов к хранению и транспортировке товаров усложняет логистические процессы;

– риск неликвидов: некоторые нишевые продукты могут иметь низкий спрос, что приводит к замораживанию средств в неликвидных товарах.

Система контроля закупок Золотое Яблоко отвечает специфическим потребностям сети, работающей в премиальном сегменте.

1.2.4 Обоснование для разработки клиент серверного приложения системы закупок магазина косметики

В таблице 1 представлена сравнительная характеристика рассмотренных систем.

Таблица 1 – Сравнительная характеристика рассмотренных систем

Характеристика	Подружка	Л'Этуаль	Золотое Яблоко
1	2	3	4
Целевая аудитория	Магазины косметики и парфюмерии среднего уровня	Крупная сеть магазинов косметики и парфюмерии	Сеть магазинов премиальной косметики и парфюмерии
Управление ассортиментом	Базовое управление ассортиментом, категоризация товаров, установка минимальных остатков, анализ продаж	Расширенное управление ассортиментом, анализ сезонных трендов, фокус на популярность товаров	Фокус на селективный ассортимент премиальных и нишевых брендов, отслеживание новинок и лимитированных коллекций, работа с эксклюзивными контрактами
Управление закупками	Создание и отправка заказов, отслеживание статуса, ведение базы поставщиков	Формирование заказов с учетом потребностей магазинов	Прямое сотрудничество с производителями и дистрибьюторами

Продолжение таблицы 1

1	2	3	4
Контроль расходов	Установка бюджетов, мониторинг расходов по категориям, анализ эффективности закупок	Установка бюджетов для каждого магазина и сети, мониторинг расходов, анализ эффективности закупок	Управление бюджетом с учетом высокой стоимости товаров, анализ эффективности закупок с фокусом на рентабельность и оборачиваемость
Дополнительные функции	Интеграция с кассовыми системами, генерация отчетов, работа с несколькими магазинами	Интеграция с кассовыми системами и программами лояльности, генерация отчетов, управление доставкой товаров	Интеграция с CRM-системой, организация мероприятий и презентаций, обучение персонала по особенностям премиальных брендов
Преимущества	Оптимизация закупок, контроль расходов, улучшение ассортимента, повышение эффективности	Эффективное управление закупками для большой сети	Экспертиза в премиальном сегменте, эксклюзивный ассортимент, персонализированный подход
Недостатки	-	Сложность системы, недостаточная гибкость, интеграция с другими системами	Высокая стоимость закупок, сложность логистики, риск неликвидов
Стоимость	-	Высокая	Очень высокая
Гибкость	-	Невысокая	Высокая
Интеграция	-	Проблемы с интеграцией	-

В рассмотренных аналогичных системах после анализа выявлено следующее:

- внешний вид и устройство сайтов не позволяют пользователю сходу освоиться и начать совершать покупки или редактирование товаров;
- для созданий заказа зачастую нужна регистрация пользователя и создание личного кабинета;
- устаревшая система контроля над закупками.

Лучшим решением выше выявленных сложностей является разработка автоматизированной информационной системы с использованием технологии ASP.NET [1], потому что ASP.NET – это один из самых популярных фреймворков для разработки сайтов. Инфраструктура ASP.NET MVC [5] представляет собой последнюю версию веб-платформы ASP.NET от Microsoft. У нее есть множество преимуществ, которых нет в классической платформе для веб-разработки ASP.NET Web Forms.

Встроенные методы представлений генерируют ясный и соответствующий стандартам код разметки в формате HTML. Таким образом ASP.NET MVC [5] предлагает мощную систему маршрутизации URL, позволяющую создавать удобные и приятные в смысловом плане URL-адреса. Компоненты ASP.NET MVC расширяемы и тестируемы, она обеспечивает расширенную поддержку разработки через тестирование. Огромным преимуществом является также мощная полнофункциональная среда разработки Microsoft Visual Studio.

Для платформы .NET на сегодняшний день существуют сотни тысяч бесплатных библиотек и компонентов, расширяющих функционал базового языка, доступных для скачивания. Также присутствует удобная система документации и база данных чуть ли не по всем вопросам, которые могут возникнуть в процессе разработки, что так же является огромным плюсом в сторону данного инструмента разработки.

Ожидаемый результат разработки системы:

- современность – Web-приложение, разработанное с помощью современных инструментов разработки;
- понятный интерфейс – имеется ввиду то, что пользователь сможет без особых усилий понять, как функционируют элементы системы и для чего они нужны;
- комфорт – пользователь сможет просматривать каталог товаров и оформлять заказ без необходимости совершения лишних действий, в том числе и регистрации;
- непротиворечивость и достоверность данных;
- низкая стоимость – так же является неоспоримым преимуществом.

1.3 Анализ требований к программной реализации

Спроектированная и полностью разработанная АИС [2] является целью выпускной квалификационной работы.

После разработки и последующего внедрения АИС предполагаются следующие достоинства:

- улучшение эффективности выполнения функций;
- неизменный контроль над процессом формирования и управление каталогом товаров;
- увеличение количества категорий товаров;
- добавление информационных и справочных разделов.

1.3.1 Список функциональных требований

Автоматизированная информационная система учета деятельности магазина косметики должна выполнять необходимый функционал:

- ведение базы данных (запись, чтение, модификация, удаление);
- хранение информации о товарах, продающихся в магазине;

- временное хранение информации о содержимом корзины пользователя системы;
- формирование заказа и отправку заявки на электронную почту администратора.

Для описания поведения и определения функциональных требований разрабатываемой системы существует UML диаграмма прецедентов, состоящая из актеров (действующих лиц), которые взаимодействуют с системой по средствам вариантов использования.

Диаграмма вариантов использования представлена на рисунке 1.



Рисунок 1 – Диаграмма вариантов использования

На диаграмме представлено 2 действующих лица – покупатель и администратор, и 6 вариантов использования – просмотр каталога товаров, формирование корзины, отправка заявки, изменение данных товаров, добавление новых товаров, удаление товаров. Описание действующих лиц представлено на таблице 2, а краткое описание вариантов использования на таблице 3. Также в приложении А приведена диаграмма с использованием включений, обобщений.

Таблица 2 – Описание действующих лиц

Название	Профиль, подготовка и навыки
Администратор	Лицо, занимающееся администрированием интернет-магазина. Может вносить изменения в работу приложения
Покупатель	Лицо, имеющее или не имеющее опыт работы с данным приложением, ознакомлен с узким функционалом программы. Вносить изменения и корректировать работу приложения не может.

Таблица 3 – Описание вариантов использования

Действующее лицо	Цель	Краткое описание
Администратор	Изменение данных товаров	Администратор может изменять данные существующих в каталоге товаров
Администратор	Добавление новых товаров	Администратор может добавлять товары в каталог
Администратор	Удаление товаров	Администратор может удалять товары из каталога
Покупатель	Просмотр каталога товаров	Покупатель может посмотреть интересующую его информацию о товарах
Покупатель	Формирование корзины	Покупатель может отправить заявку при завершении формирования корзины

Описание прецедентов представлено в таблицах 4 – 9.

В таблице 4 описывается прецедент «Просмотр каталога товаров», главный актер в котором – Покупатель.

Таблица 4 – Описание прецедента «Просмотр каталога товаров»

Название прецедента	Просмотр каталога товаров
ID	1
Краткое описание	Покупатель просматривает каталог товаров
Главные актёры	Покупатель
Второстепенные актёры	-
Предусловие	-
Основной поток событий	<ol style="list-style-type: none"> 1. Покупатель выбирает категорию товара для сортировки; 2. Покупатель просматривает характеристики товаров из интересующей его категории.
Постусловие	-
Альтернативные потоки	-

В таблице 5 описывается прецедент «Формирование корзины», главный актер в котором – Покупатель.

Таблица 5 – Описание прецедента «Формирование корзины»

Название прецедента	Формирование корзины
ID	2
Краткое описание	Покупатель формирует корзину
Главные актёры	Покупатель
Второстепенные актёры	Администратор
Предусловие	-
Основной поток событий	<ol style="list-style-type: none"> 1. Покупатель выбирает интересующий его товар; 2. Покупатель выбирает пункт добавления товара в корзину.
Постусловие	-
Альтернативные потоки	-

В таблице 6 описывается прецедент «Отправка заявки», главный актер в котором – Покупатель.

Таблица 6 – Описание прецедента «Отправка заявки»

Название прецедента	Отправка заявки
ID	3
Краткое описание	Покупатель отправляет заявку
Главные актёры	Покупатель
Второстепенные актёры	Администратор
Предусловие	Корзина сформирована
Основной поток событий	<ol style="list-style-type: none"> 1. Покупатель переходит в корзину; 2. Покупатель выбирает пункт «Отправить заявку»; 3. Покупатель заполняет контактные данные; 4. Заявка приходит на почту администратору для обработки заказа.
Постусловие	Заявка отправлена
Альтернативные потоки	Если контактные данные заполнены неверно, система уведомит пользователя и попросит исправить допущенные ошибки.

В таблице 7 описывается прецедент «Изменение данных товаров», главный актер в котором – Администратор.

Таблица 7 – Описание прецедента «Изменение данных товаров»

Название прецедента	Изменение данных товаров
1	2
ID	4
Краткое описание	Администратор изменяет данные товаров
Главные актёры	Администратор

Продолжение таблицы 7

1	2
Второстепенные актёры	-
Предусловие	В базе есть хотя бы 1 товар
Основной поток событий	1. Администратор входит в систему под своим логином и паролем; 2. Администратор открывает страницу с данными о товаре; 3. Администратор вносит изменения и сохраняет информацию.
Постусловие	-
Альтернативные потоки	Если информацию о товаре невозможно добавить, выведется сообщение об ошибке

В таблице 8 описывается прецедент «Добавление новых товаров», главный актер в котором – Администратор.

Таблица 8 – Описание прецедента «Добавление новых товаров»

Название прецедента	Добавление новых товаров
ID	5
Краткое описание	Администратор добавляет новые товары
Главные актёры	Администратор
Второстепенные актёры	-
Предусловие	-
Основной поток событий	1. Администратор входит в систему под своим логином и паролем; 2. Администратор открывает страницу добавления товара; 3. Администратор вносит данные о товаре и добавляет товар.
Постусловие	-
Альтернативные потоки	Если товар невозможно добавить, выведется сообщение об ошибке

В таблице 9 описывается прецедент «Удаление товаров», главный актер в котором – Администратор.

Таблица 9 – Описание прецедента «Удаление товаров»

Название прецедента	Удаление товаров
ID	6
Краткое описание	Администратор удаляет товары
Главные актёры	Администратор
Второстепенные актёры	-
Предусловие	В базе есть хотя бы 1 товар
Основной поток событий	<ol style="list-style-type: none"> 1. Администратор входит в систему под своим логином и паролем; 2. Администратор выбирает товар из списка и удаляет его.
Постусловие	-
Альтернативные потоки	-

Более подробно некоторые из прецедентов описаны в диаграммах UML в приложении А.

1.3.2 Список нефункциональных требований

Требования к временным характеристикам:

- время открытия главной страницы не должно быть более 20 с;
- время перехода между страницами не должно быть более 10 с.

1.3.3 Требования к надежности

Администрирование системы.

Доступ к админ-панели может быть получен только администратором путем его перехода к админ-панели по специальной ссылке и авторизации с помощью личного логина и пароля;

Необходимое время для восстановительных мер после отключения.

При отключении сервера, необходимо обеспечивать восстановление информации утраченной в результате сбоя в работе.

1.3.4 Условия эксплуатации

Для успешного функционирования данного клиент-серверного приложения необходима операционная система, современный web-браузер, СУБД MySql, подключение к глобальной сети Интернет. Система должна быть ориентирована на простого пользователя с навыками работы в web-браузерах. Администрирование системы должно осуществляться лицом, ответственным за контроль закупок магазина косметики, обладающими специальными знаниями в сфере администрирования информационных систем, с навыками работы в web-браузерах, знающим структуру и все функциональные возможности проектируемой и разрабатываемой информационной системы контроля закупок магазина косметики.

2 Проектирование программного обеспечения

2.1 Проектирование пользовательского интерфейса

Пользовательский интерфейс – это совокупность программных средств, обеспечивающих взаимодействие пользователя с программным продуктом. Основу такого взаимодействия составляют диалоги. Под диалогом в данном случае можно понимать обмен информацией между человеком и программным средством, осуществляемый в реальном масштабе времени и направленный на совместное решение конкретной задачи.

Пользовательский интерфейс информационной системы контроля закупок магазина косметики представлен ниже на рисунке 2.

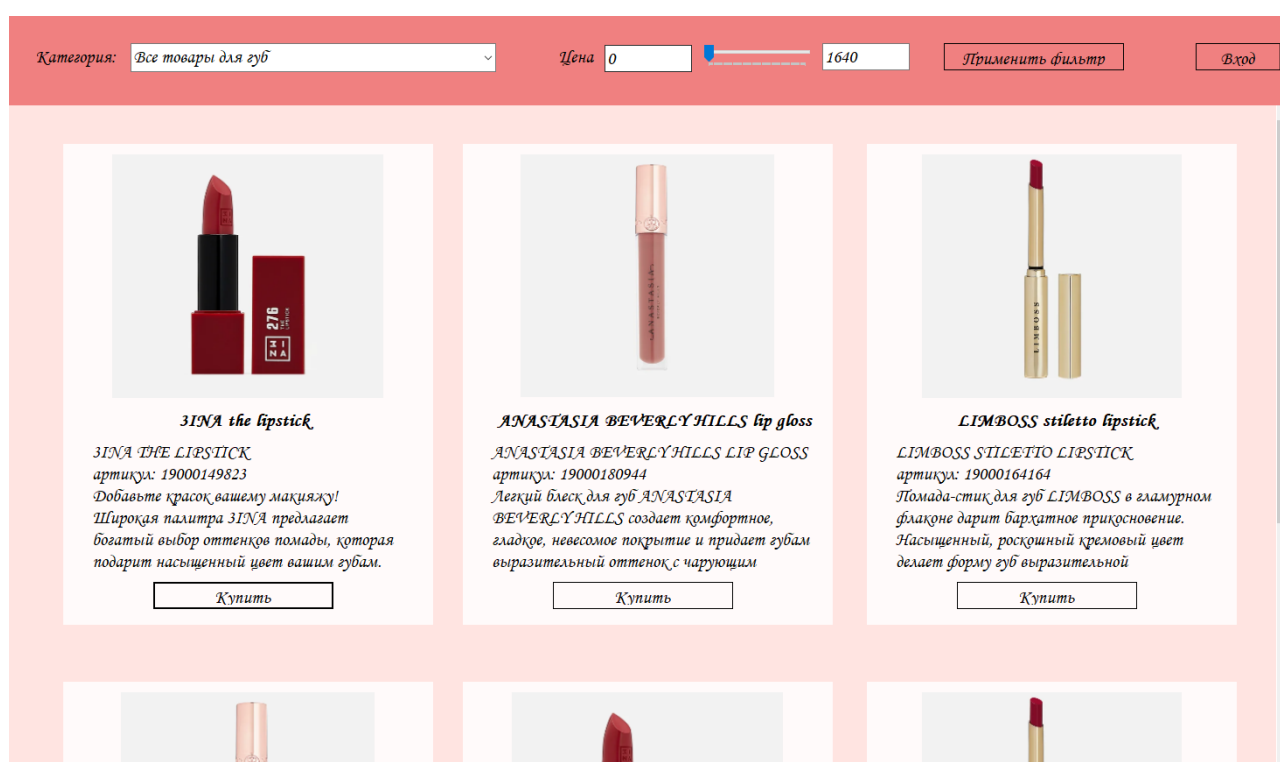


Рисунок 2 – Стартовая страница системы

Для администратора имеется собственная форма авторизации, позволяющая ему получить доступ к операциям создания/изменения/удаления товаров. Она включает в себя два поля, заполнив которые администратор может получить доступ к админ панели.

Ее внешний вид представлен на рисунке 3.

Рисунок 3 – Форма входа администратора

После успешной авторизации администратор попадает в админ панель, внешний вид которой представлен на рисунке 4.

	Название товара	Описание	Категория	Стоимость
▶	Розовый Металлик 781	Помада с Матовый фин...	Объемная	2126
	Фиолетовый Глянцевый...	Помада с Глянцевый фи...	Увлажняющая	570
	Фиолетовый Глянцевый...	Помада с Металлик фин...	Объемная	1321
	Персиковый Матовый 646	Помада с Шиммерный ...	Вечерняя	2222
	Нюдový Сатиновый 931	Помада с Шиммерный ...	Объемная	1298
	Коралловый Глянцевый ...	Помада с Глянцевый фи...	Вечерняя	628
	Бордовый Металлик 114	Помада с Матовый фин...	Увлажняющая	2522
	Фиолетовый Матовый ...	Помада с Матовый фин...	Повседневная	1261
	Бордовый Матовый 425	Помада с Металлик фин...	Объемная	2820
	Ягодный Металлик 449	Помада с Матовый фин...	Стойкая	799
	Коричневый Металлик ...	Помада с Матовый фин...	Вечерняя	1234
	Бордовый Сатиновый 725	Помада с Металлик фин...	Объемная	689
	Нюдový Глянцевый 744	Помада с Глянцевый фи...	Повседневная	2163
	Розовый Металлик 560	Помада с Сатиновый фи...	Повседневная	2467
	Красный Металлик 199	Помада с Сатиновый фи...	Стойкая	1538
	Фиолетовый Матовый ...	Помада с Шиммерный ...	Повседневная	518

Рисунок 4 – Интерфейс админ панели

Из данной панели администратор может добавлять/удалять товары, нажимая на соответствующие кнопки, либо же изменять содержимое товара, путем нажатия на название товара.

2.2 Анализ бизнес процессов организации в нотации IDEF0

IDEF0 – методология функционального моделирования (англ. function modeling) и графическая нотация, предназначенная для формализации и описания бизнес-процессов. Отличительной особенностью IDEF0 является ее акцент на соподчиненность объектов. В IDEF0 рассматриваются логические отношения между работами, а не их временная последовательность. Логические отношения представляют собой определенные действия, направленные на отображение работы деятельности компании, в частности контроля закупок магазина косметики.

Бизнес-процесс включает в себя взаимосвязанные действия, которые реализуют одну (или несколько) из бизнес-целей компании в информационной системе компании.

За основу взят бизнес-процесс «Деятельность магазина косметики», для отображения логических отношений бизнес процесса. На контекстной диаграмме организации работы закупок магазина видно функционирование основной бизнес-функции:

- новые товары (Вход);
- прибыль от продажи товаров (Выход);
- нормативные документы (Управление);
- покупатель (Механизм);
- компания-поставщик (Механизм);
- отдел маркетинга (Механизм);
- руководитель (Механизм);
- бухгалтерский отдел (Механизм);
- отдел продаж (Механизм);
- склад готовой продукции (Механизм).

Диаграмма бизнес-процессов в нотации IDEF0 [3] представлена на рисунке 5.

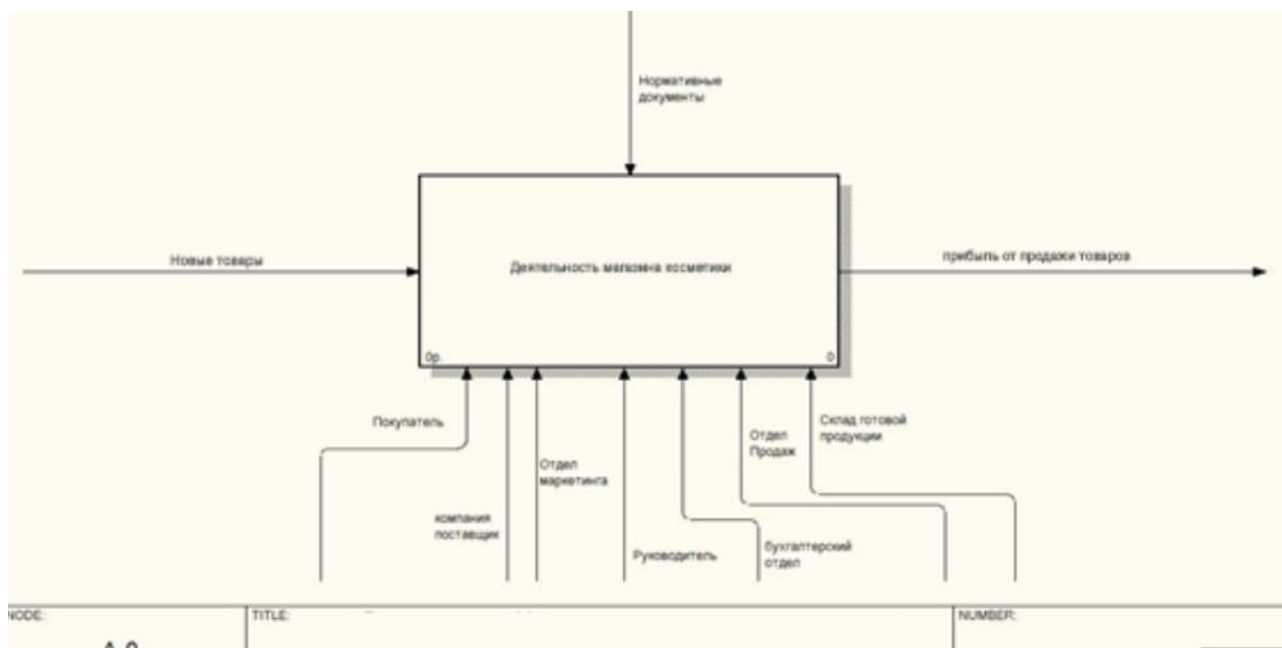


Рисунок 5 – IDEF0 модель работы системы закупок магазина косметики

Декомпозиция представлена на рисунке 6.

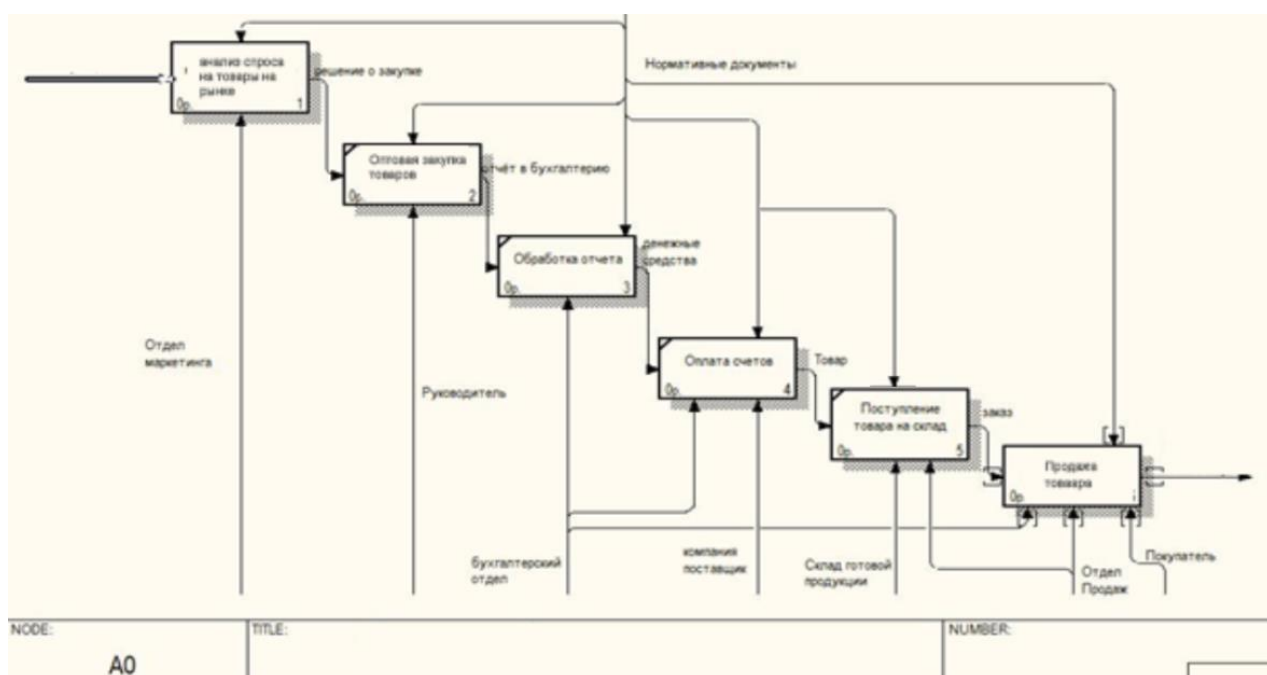


Рисунок 6 – Контекстная диаграмма деятельности по организации работы системы закупок магазина косметики

При декомпозиции главной бизнес-функции выделяются следующие функциональные блоки: анализ спроса на товары на рынке, анализ спроса на

товары на рынке, обработка отчета, оплата счетов, поступление товара на склад, продажа товара.

2.3 Проектирование модели базы данных

Для разработки моделей базы данных используется Case-средство ERwin и его методологию IDEF1X.

В ERwin существуют два уровня представления и моделирования – инфологический и физический.

2.3.1 Проектирование физической и инфологической моделей базы данных с помощью Erwin

Инфологический уровень – это общий взгляд на данные, на нем данные представляются так, как выглядят в реальном мире, и могут называться так, как они называются в реальном мире. Объекты модели, представляемые на инфологическом уровне, называются сущностями и атрибутами. Инфологическая модель данных является универсальной и никак не связана с конкретной реализацией СУБД.

Создание модели данных, как правило, начинается с создания инфологической модели. После описания инфологической модели, человек, занимающийся проектированием может выбрать необходимую СУБД и ERwin [4] автоматически создаст соответствующую физическую модель. На основе физической модели ERwin может сгенерировать системный каталог СУБД или соответствующий SQL-скрипт [5], необходимый для создания базы данных.

В Case средстве ERwin [6] разработана инфологическая модель базы данных для хранения информации охранной организации с учетом всех нюансов и специфики данной предметной области. Данная модель содержит семь таблиц, связанных между собой логическими связями. Инфологическая модель представлена на рисунке 7.

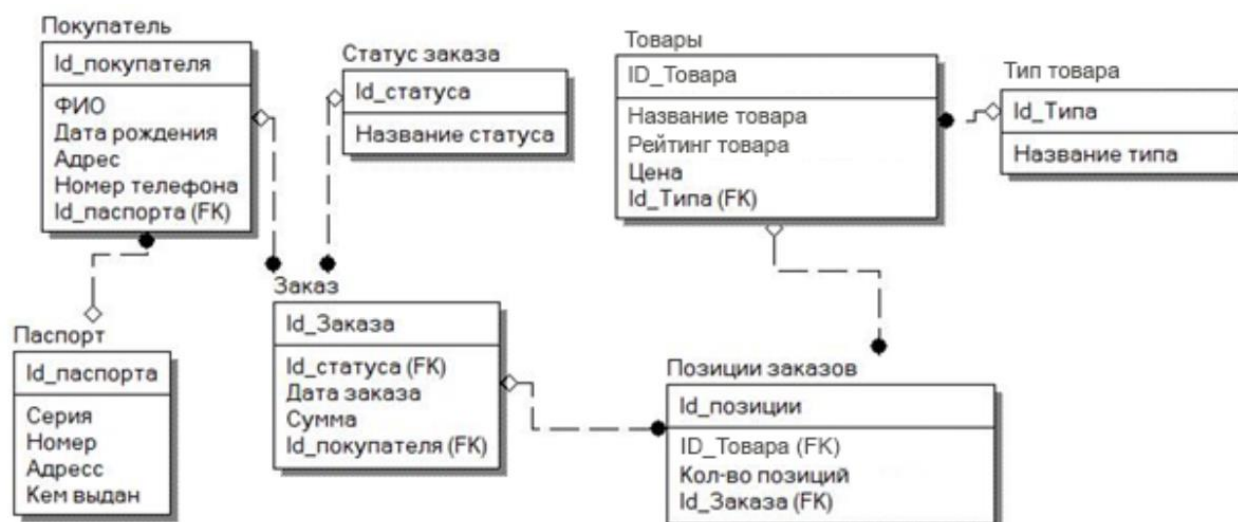


Рисунок 7 – Диаграмма инфологической модели

Физическая модель данных зависит от конкретной СУБД, фактически являясь отображением системного каталога. В физической модели содержится информация о всех объектах БД. Поскольку стандартов на объекты БД не существует (например, нет стандарта на типы данных), физическая модель зависит от конкретной реализации СУБД. Следовательно, одной и той же инфологической модели могут соответствовать несколько разных физических моделей.

Физическая модель представлена на рисунке 8.

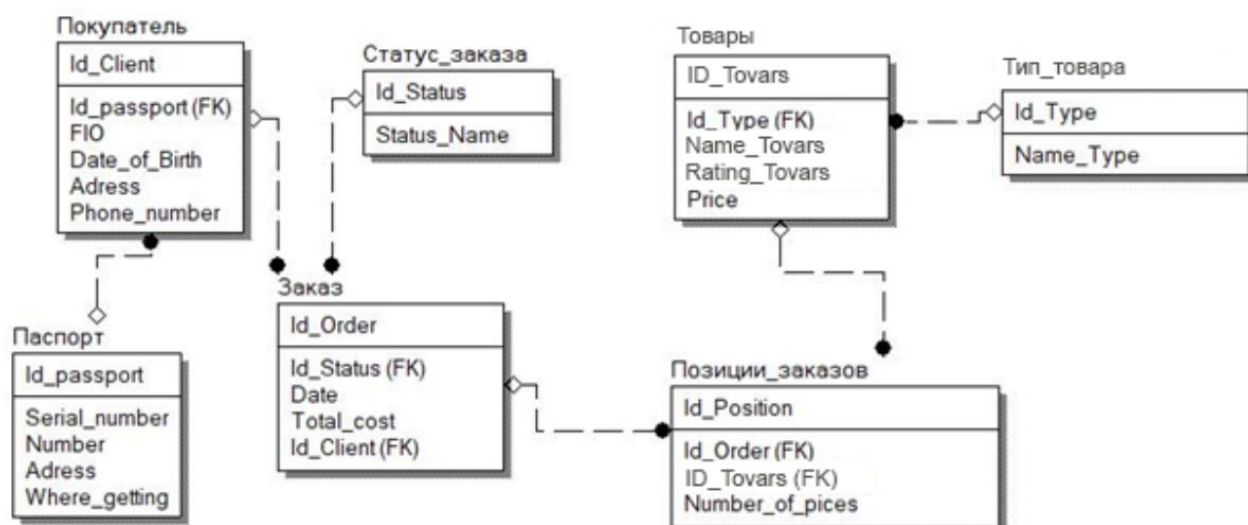


Рисунок 8 – Диаграмма физической модели

Если в инфологической модели не имеет значения, какой конкретно тип данных имеет атрибут, то в физической модели дело обстоит иначе – важно описать всю информацию о конкретных физических объектах, таких как таблицы, колонки, индексы, первичные и внешние ключи.

На диаграмме IDEF1X [7] физического уровня представлены таблицы и связи между ними, которые будут реализованы в разрабатываемой базе данных.

2.4 Проектирование поведения ИС

Для отражения взаимодействия информационной системы закупок магазина косметики и ее компонентов построены поведенческие диаграммы: диаграмма состояния, диаграмма деятельности, диаграмма последовательности, диаграмма коммуникации, диаграмма компонентов и диаграмма развертывания.

2.4.1 Диаграмма состояний

Объекты характеризуются поведением и состоянием, в котором находятся. Диаграмма состояний показывает, как объект переходит из одного состояния в другое. В данном случае диаграмма состояний отражает состояние объекта класса Корзина, и его возможные состояния.

Класс **Корзина**, помимо начального и конечного состояния может, находиться в следующих состояниях:

- cart is Empty (корзина пуста) – корзина может быть пуста в начале покупки, после удаления из не всех товаров или после совершения заказа;
- cart contains products (корзина содержит товар) – корзина может содержать товар, если пользователь добавил в нее товар, но еще не совершил заказ.

Переходы в указанные состояния на диаграмме имеют сторожевые условия (условия перехода).

2.4.2 Диаграмма деятельности

Диаграмма деятельности показывает разложение некоторой деятельности на ее составные части. Под деятельностью понимается спецификация исполняемого поведения в виде координированного последовательного и параллельного выполнения подчиненных элементов – вложенных видов деятельности и отдельных действий, соединенных между собой потоками, которые идут от выходов одного узла ко входам другого.

Диаграмма деятельности построена для прецедента «Совершить заказ».

На диаграмме деятельности отражено выполнение действий:

Сначала необходимо просмотреть каталог товаров, затем добавить товар в корзину. Если пользователь добавил все что хотел в корзину, пользователь переходит к оформлению заказа. Если нет, пользователь возвращается к выбору товаров. На диаграмме это отражено с помощью ветвления (ромб). Затем, пользователь вводит контактные данные в заявку.

Если данные введены верно, оформление заявки завершается и деятельность заканчивается. Если ввел данные неверно – возвращается к этапу ввода данных.

Диаграмма представлена в приложении А.

2.4.3 Диаграмма последовательности

Диаграмма последовательности – диаграмма, на которой для некоторого набора объектов на единой временной оси показан жизненный цикл какого-либо определенного объекта и взаимодействие актеров ИС в рамках какого-либо определенного прецедента (отправка запросов и получение ответов). Для разрабатываемой ИС диаграмма последовательности построена для прецедента «Добавление нового сотрудника». На ней «Пользователь» актер взаимодействует с объектами системы: формой заполнения данных, модулем работы с данными и базой данных – отправляя и получая сообщения.

Объект «Покупатель» выбирает интересующую его категорию товара, затем «Система» возвращает товары только запрошенной категории. Объект «Покупатель» добавляет товар в корзину. Объект «Система» Возвращает содержимое корзины на экран. «Пользователь» заполняет и отправляет заявку их «Системе», она, в свою очередь, отправляет заявку на почту администратора и возвращает объекту «Пользователь» сообщение, что заявка принята.

Диаграмма представлена в приложении А.

2.4.4 Диаграмма коммуникаций

Диаграмма коммуникаций – диаграмма, на которой изображаются взаимодействия между частями композитной структуры или ролями кооперации. В отличие от диаграммы последовательности, на диаграмме кооперации явно указываются отношения между объектами, а время как отдельное измерение не используется (применяются порядковые номера вызовов). Для информационной системы контроля рабочего времени сотрудников построена диаграмма коопераций для процесса «администрирование каталога».

На диаграмме коммуникаций представлены следующие объекты:

- администратор;
- система;
- админ панель;
- база данных.

Администратор переходит в админ панель, система возвращает ему форму авторизации, администратор вводит свои логин и пароль. После этого она попадает в админ-панель, где создает/изменяет/удаляет товары, и отправляет запрос в БД. После успешного проведения операции БД возвращает измененную таблицу в админ панель.

Диаграмма представлена в приложении А.

2.4.5 Диаграмма компонентов

Диаграмма компонентов – статическая структурная диаграмма, показывает разбиение программной системы на структурные компоненты [8] и связи (зависимости) между компонентами. В качестве физических компонентов могут выступать файлы, библиотеки, модули, исполняемые файлы, пакеты.

Автоматизированная информационная система контроля закупок магазина содержит в себе следующие структурные компоненты:

- loginViewModel.cs, CartIndexModel.cs, PagingInfo.cs, TovarListViewModel.cs – модели;
- login.chnml, admin.Edit.chnml, cart.Checkout.chnml, cart.Completed.chnml, cart.Index.chnml, cart.Summary.chnml – представления;
- accountController.cs, AdminController.cs, CartController.cs, NavController.cs, ShopController.cs – контролеры.

Компоненты связаны между собой отношениями зависимости.

Диаграмма представлена в приложении А.

2.4.6 Диаграмма развертывания

Диаграмма развертывания – это диаграмма, на которой представлены узлы выполнения программных компонентов реального времени, а также процессов и объектов.

Диаграмма развертывания применяется для представления общей конфигурации и топологии распределенной программной системы и содержит изображение размещения компонентов по отдельным узлам системы. Кроме того, диаграмма развертывания показывает наличие физических соединений-маршрутов передачи информации между аппаратными устройствами, задействованными в реализации системы.

Диаграмма развертывания предназначена для визуализации элементов и компонентов программы, существующих только на этапе ее исполнения.

Выделим основные компоненты web-приложения информационной системы контроля закупок магазина косметики. В компоненте SQL Server DB находится база данных нашей информационной системы. На компоненте **ISS** **11** web-сервер находится начальная страница web-приложения ИС <http://localhost/port>. Компонент браузер служит для подключения к стартовой странице приложения.

Компоненты связаны между собой отношениями зависимости и интерфейсами.

Диаграмма представлена в приложении А.

3 Разработка приложения

3.1 Программные средства

После ознакомления с основными существующими на данный момент типами баз данных, выбор пал на реляционную базу данных на основе SQL Server 2022. Microsoft SQL Server – система управления реляционными базами данных (РСУБД), разработанная корпорацией Microsoft. Основным используемый язык запросов – Transact-SQL, создан совместно Microsoft и Sybase. Transact-SQL является реализацией стандарта ANSI/ISO по структурированному языку запросов (SQL) с расширениями.

SQL (Structured Query Language) [9] – стандартный язык запросов по работе с реляционными базами данных. SQL в чистом (базовом) виде является информационно-логическим языком, а не языком программирования. Однако стандарт языка спецификацией SQL/PSM предусматривает возможность его процедурных расширений, с учетом которых язык уже может рассматриваться в качестве языка программирования. SQL предоставляет пользователю достаточно простой и понятный механизм доступа к данным, не связанный с конструированием алгоритма и его описанием на языке программирования высокого уровня. Так, вместо указания того, как необходимо действовать, пользователь с помощью операторов SQL объясняет СУБД, что нужно сделать. Далее СУБД сама анализирует текст запроса и определяет, как именно его выполнять, и после этого выдает результат запроса.

Соединение с базой данных [10] и клиентского приложения осуществляется с помощью языковых средств C# [11,12], а именно его фреймворка ASP.NET в среде разработки Visual Studio 2022.

C# – это язык программирования, предназначенный для разработки самых разнообразных приложений, предназначенных для выполнения в среде .NET Framework. Язык C# прост, безопасен и объектно-ориентирован. Благодаря множеству нововведений C# обеспечивает возможность быстрой

разработки приложений, но при этом сохраняет выразительность и элегантность, присущую языкам С. Так же весомым преимуществом языка С# является то, что он постоянно обновляется, благодаря чему пользователю не приходится прибегать к сторонним средствам для выполнения поставленных задач. Достаточно загрузить нужные библиотеки, добавляющие необходимый функционал.

Платформа ASP.NET MVC [13,14] представляет технологию от компании Microsoft, предназначенную для создания различного рода Web-приложений: от небольших web-сайтов до крупных web-порталов и web-сервисов.

С одной стороны, ASP.NET MVC является продолжением развития платформы ASP.NET. Но с другой стороны, это не просто очередной релиз. Выход ASP.NET MVC фактически означает революцию всей платформы, ее качественное изменение.

С помощью ASP.NET MVC разработчик может создавать кроссплатформенные приложения. И хотя Windows в качестве среды для разработки и развертывания приложения до сих пор превалирует, но теперь он уже не ограничен только этой операционной системой. То есть разработчик может запускать web-приложения не только на ОС Windows, но и на Linux и Mac OS. А для развертывания web-приложения можно использовать традиционный IIS, либо кросс-платформенный web-сервер.

Microsoft Visual Studio – линейка продуктов компании Microsoft, включающих интегрированную среду разработки программного обеспечения и ряд других инструментальных средств. Данные продукты позволяют разрабатывать как консольные приложения, так и приложения с графическим интерфейсом, в том числе с поддержкой технологии как устаревшей технологии Windows Forms, так и более новой Windows Presentation Foundation. Так же эти продукты позволяют разрабатывать веб-сайты,

Web-приложения, web-службы как в родном, так и в управляемом кодах для всех платформ.

Преимущества ASP.NET MVC:

- архитектурный шаблон MVC;
- расширяемость;
- жесткий контроль над HTML и HTTP;
- тестируемость.

В результате, с использованием данных средств приложение было реализовано и запущено. Текст приложения приведен в приложении Б.

4 Отладка и тестирование

4.1 Тестирование программного обеспечения

Тестирование [15] – очень важный этап разработки как приложений. По способам, целям и глубине тестирования оно делится на следующие виды:

- функциональное тестирование (functional testing) [16];
- системное тестирование (system testing);
- тестирование производительности (performance testing);
- регрессионное тестирование (regression testing);
- модульное тестирование (unit testing);
- тестирование безопасности (security testing);
- тестирование локализации (localization testing);
- юзабилити тестирование (usability testing).

Модульное тестирование (Unit Testing) – это тип тестирования программного обеспечения, при котором тестируются отдельные модули или компоненты программного обеспечения. Его цель заключается в том, чтобы проверить, что каждая единица программного кода работает должным образом. Данный вид тестирования выполняется разработчиками на этапе кодирования приложения. Модульные тесты изолируют часть кода и проверяют его работоспособность. Единицей для измерения может служить отдельная функция, метод, процедура, модуль или объект. Отсутствие модульного тестирования при написании кода значительно увеличивает уровень дефектов при дальнейшем (интеграционном, системном, и приемочном) тестировании. Качественное модульное тестирование на этапе разработки экономит время, а следовательно, в конечном итоге, и бюджет проекта.

Проведем модульное тестирование [17] работы отдельных функций приложения. Таблица с описанием тестов представлена ниже (таблица 10).

Результаты тестирования приведены в приложении В.

Таблица 10 – Модульное тестирование работы системы

№	Название теста	Предполагаемый результат	Полученный результат	Результат тестирования
1	2	3	4	5
1	Adding_Tovar_To_Cart_Goes_To_Cart_Screen	При добавлении товара в корзину открывается модуль	Соответствует ожидаемым результатам	Тест пройден
2	Calculate_Cart_Total	Подсчет итоговой стоимости товаров	Соответствует ожидаемым результатам	Тест пройден
3	Can_Add_New_Lines	Добавление новых товаров в каталог из админ-панели работает верно	Соответствует ожидаемым результатам	Тест пройден
4	Can_Add_Quantity_For_Existing_Lines	Повторное добавление товара в корзину приводит к увеличению счетчика	Соответствует ожидаемым результатам	Тест пройден
5	Can_Add_To_Cart	Добавление товара в базу данных работает верно	Соответствует ожидаемым результатам	Тест пройден
6	Can_Checkout_And_Submit_Order	Формирование и отправка запроса работает верно	Соответствует ожидаемым результатам	Тест пройден
7	Can_Clear_Contents	Функция очистки корзины работает верно	Соответствует ожидаемым результатам	Тест пройден
8	Can_Create_Users	Добавление новых пользователей работает верно	Соответствует ожидаемым результатам	Тест пройден

Продолжение таблицы 10

1	2	3	4	5
9	Can_Delete_Valid_Tovars	Удаление товаров из админ панели работает верно	Соответствует ожидаемым результатам	Тест пройден
10	Can_Edit_Tovar	Изменение товаров из админ панели работает верно	Соответствует ожидаемым результатам	Тест пройден
11	Can_Generate_Page_Links	Генерация ссылок на страницы работает верно	Соответствует ожидаемым результатам	Тест пройден
12	Can_Login_With_Valid_Credits	Авторизация администратора для доступа к админ-панели проходит успешно	Соответствует ожидаемым результатам	Тест пройден
13	Can_Paginate	Разбиение на страницы работает верно	Соответствует ожидаемым результатам	Тест пройден
14	Can_Remove_Line	Удаление отдельных товаров из каталога работает верно	Соответствует ожидаемым результатам	Тест пройден
15	Can_Retrieve_Info_Data	Добавление информации пользователей работает верно	Соответствует ожидаемым результатам	Тест пройден
16	Can_Save_Valid_Changes	Изменения, соответствующие требованиям, успешно сохраняются	Соответствует ожидаемым результатам	Тест пройден

Окончание таблицы 10

1	2	3	4	5
17	Can_View_Cart_Contents	Пользователь может просматривать корзину	Соответствует ожидаемым результатам	Тест пройден
18	Cannot_Checkout_Empty_Cart	Невозможно отправить заявку при пустой корзине	Соответствует ожидаемым результатам	Тест пройден
19	Cannot_Edit_Nonexist_Tovar	Невозможно редактировать несуществующий товар	Соответствует ожидаемым результатам	Тест пройден
20	Cannot_Login_Without_Valid	Невозможно войти по не верным данным аккаунта	Соответствует ожидаемым результатам	Тест пройден
21	Cannot_Retrive_Info_Data	Невозможно присвоить сведения несуществующему товару	Соответствует ожидаемым результатам	Тест пройден
22	Cannot_Save_Invalid_Changes	Невозможно сохранить изменения, если они не соответствуют требования	Соответствует ожидаемым результатам	Тест пройден
23	Generate_Day_Specific_Tovars	Количество товаров подсчитывается верно	Соответствует ожидаемым результатам	Тест пройден
24	Index_Contains_All_Tovars	Каталог содержит все товары, содержащиеся в базе данных	Соответствует ожидаемым результатам	Тест пройден

Модульное тестирование [18] показало, что система функционирует корректно и не имеет явных ошибок и сбоев в работе, что является положительным результатом проведения тестирования. Результаты тестирования представлены в приложении В.

4.1.1 Метрики программного средства

Метрика программного обеспечения – мера, позволяющая получить численное значение некоторого свойства программного обеспечения или его спецификаций.

Метрики используются в следующих целях:

- для контроля над процессом разработки в целом, а в тестировании в частности;
- для оценки прогресса выполнения работ по тестированию ПО;
- для последующего планирования процессов.

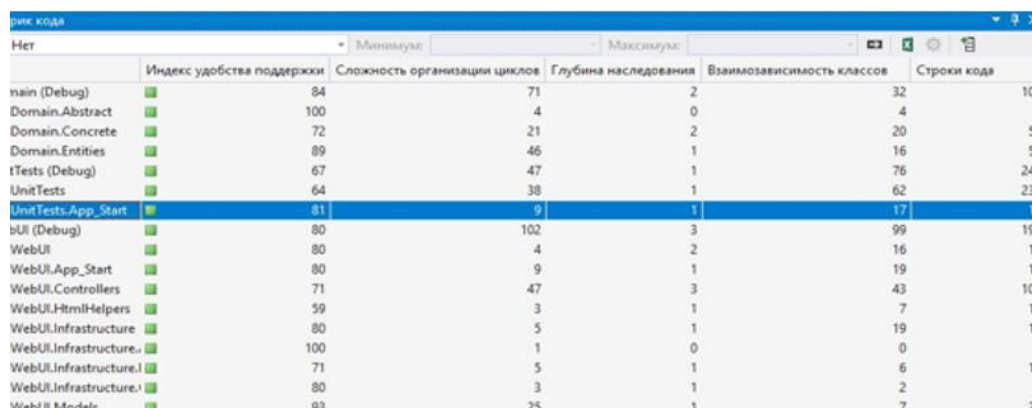
4.1.2 Расчет метрик кода с помощью встроенного средства Visual Studio

Среда разработки Microsoft Visual Studio 2022 имеет встроенное средство для выполнения автоматического расчета метрик кода. К таковым относятся:

- индекс удобства поддержки;
- сложность организации циклов;
- глубина наследования;
- объединение классов;
- строки кода.

Для расчета метрик кода средствами Microsoft Visual Studio 2022 необходимо выбрать пункт контекстного меню «Анализ», далее – «Рассчитать метрики кода».

На рисунке 9 приведен результат расчета метрик кода в среде Microsoft Visual Studio 2022.



Имя	Индекс удобства поддержки	Сложность организации циклов	Глубина наследования	Взаимозависимость классов	Строки кода
main (Debug)	84	71	2	32	108
Domain.Abstract	100	4	0	4	0
Domain.Concrete	72	21	2	20	54
Domain.Entities	89	46	1	16	54
tTests (Debug)	67	47	1	76	245
UnitTests	64	38	1	62	232
UnitTests.App_Start	81	9	1	17	11
bUI (Debug)	80	102	3	99	193
WebUI	80	4	2	16	11
WebUI.App_Start	80	9	1	19	14
WebUI.Controllers	71	47	3	43	104
WebUI.HtmlHelpers	59	3	1	7	12
WebUI.Infrastructure	80	5	1	19	11
WebUI.Infrastructure.I	100	1	0	0	0
WebUI.Infrastructure.I	71	5	1	6	10
WebUI.Infrastructure.I	80	3	1	2	6
WebUI.Models	93	25	1	7	25

Рисунок 9 – Метрики кода в Visual Studio

Для наглядности среда Microsoft Visual Studio отображает цветовую индикацию уровня качества кода. Зеленая цвет свидетельствует о высоком качестве кода, желтая означает, что качество кода ниже оптимального. В данном случае видно, что качество кода всех компонентов имеет высокое качество.

4.1.3 Определение метрик компонента серверной части приложения

Далее рассмотрен исходный код метода Login класса AccountController.cs (рисунок 10), для которого будут рассчитаны метрики Холстеда и выполнен комплексный анализ кода. Анализ кода будет выполнен согласно формулам метрики Холстеда.

```
public ActionResult Login(LoginViewModel model, string returnUrl)
{
    if (ModelState.IsValid)
    {
        if (authProvider.Authenticate(model.UserName, model.Password))
        {
            return Redirect(returnUrl ?? Url.Action("Index", "Admin"));
        }
        else
        {
            ModelState.AddModelError("", "Неправильный логин или пароль");
            return View();
        }
    }
    else
    {
        return View();
    }
}
```

Рисунок 10 – Код метода Login(LoginViewModel, string)

Словарь для расчета метрик вышеуказанного блока исходного кода приведен в таблице 11.

Таблица 11 – Словарь

Переменная	№	Встречается	Оператор	№	Встречается
model	1	1	if	1	2
returnUrl	2	1	return	2	3
ModelState	3	2	else	3	2
authProvider	4	1	??	4	1
UserName	5	1			
Password	6	1			
Итого:	6	7		4	8
	n1	N1		n2	N2

Расчет метрик Холстеда выполнен с использованием математического пакета Mathcad. Результат расчета приведен ниже.

Исходные данные:

$$mn1 := 6, mN1 := 7, mn2 := 4, mN2 := 8.$$

Расчет словаря, длины и объема программы производится по формуле:

$$n := mn1 + mn2 = 10;$$

$$N := mN1 + mN2 = 15;$$

$$V := N * \log(2, n) = 49,83.$$

Расчет уровня качества программирования производится по формуле:

$$L := 2mn2 / (mn1 * mN2) = 0,17.$$

Расчет сложности понимания программы производится по формуле:

$$EC := V/2L = 146,55.$$

Расчет уровня языка производится по формуле:

$$Y := V/D2 = 1,38.$$

Расчет оценки необходимых интеллектуальных усилий производится по формуле:

$$E := N * \log(2, n/L) = 87,87.$$

Расчет трудоемкости кодирования производится по формуле:

$$D := 1/L = 6.$$

Расчет информационного содержания программы производится по формуле:

$$I := V/D = 8,31.$$

Подход, приведенный в расчете метрик Холстеда, значительно усложняется при увеличении объема кода проекта. В таком случае целесообразно использовать встроенные в среду разработки или отдельно поставляемые инструменты для автоматического анализа качества кода.

Заключение

С помощью языка C#, базы данных SQL SERVER 2022 и среды разработки Visual Studio 2022 от компании Microsoft были реализованы база данных и клиент-серверное приложение информационной системы закупок магазина косметики. По результатам функционального тестирования автоматизированная информационная система работает корректно и выполняет поставленные задачи.

Применение данной системы позволит организовать деятельность магазина, занимающегося продажей косметики:

- быстро и удобно вести администрирование имеющейся продукции;
- снизить случайные ошибки связанных с ручным оформлением заказов;
- автоматизировать обработку информации в системе;
- получить доступ к каталогу продукции из любой точки, где есть компьютер, имеющий выход в Интернет;
- получить на выходе систему с низкой стоимостью разработки, но имеющую весь основной функционал, необходимый для комфортного совершения заказов покупателями.

В дальнейшем систему планируется дорабатывать, добавлять новый функционал, такой как составление счетов а закупку материалов, автоматический документооборот, формирование чеков, интеграция с сервисами складского учета, такими как МойСклад или Контур. Также планируется внедрять систему в работу уже существующих компаний и предлагать подключение только открывающимся компаниям, являющимися магазинами косметики.

Список использованных источников

1. Информационная система – это //Что такое информационные системы и зачем они нужны – классификация и использование [Электронный ресурс] URL: <https://ktonanovenkogo.ru/voprosy-i-otvety/informacionnayasisistema-cto-eh-to-takoe.html?ysclid=l4hevni6mn440545180> (дата обращения 07.05.2024).
2. Вендров А.М. Структурный подход к проектированию ИС // Портал научно-практических публикаций [Электронный ресурс]. URL: http://citforum.ru/database/case/glava2_1.shtml (дата обращения: 07.05.2024).
3. Приложение с открытым исходным кодом для персональных компьютеров // ArgoUML [Электронный ресурс]. URL: <https://argouml.softonic.ru> (Дата обращения: 15.05.2024).
4. Построение UML-модели в среде Visual Paradigm 16.1. Система обработки заказов [Электронный ресурс]. URL: <http://sp.cs.msu.ru/ooap/exerb2020.html> (Дата обращения: 18.05.2024).
5. Маклаков, С. В. BPwin и ERwin: CASE-средства для разработки информационных систем // Портал научно-практических публикаций [Электронный ресурс]. URL: <https://elib.pnzgu.ru/files/eb/doc/DSA5gziAMfNS.pdf> (дата обращения: 18.05.2024).
6. Проектирование информационных систем [Электронный ресурс]. URL: <https://moodle.kstu.ru/course/view.php?id=4638&> (Дата обращения: 26.05.2024).
7. Варианты использования // Диаграмма вариантов использования [Электронный ресурс]. URL: http://www.telenir.net/uchebniki/samouchitel_uml/p4.php?ysclid=l4hg07qpf139464158 (Дата обращения: 28.05.2024).
8. Варианты использования // Диаграмма вариантов использования [Электронный ресурс]. URL:

http://www.telenir.net/uchebniki/samouchitel_uml/p4.php?ysclid=l4hg07qpf139464158 (Дата обращения: 28.05.2024).

9. Что такое интерфейс пользователя? [Электронный ресурс]. URL: <https://www.internet-technologies.ru/articles/newbie/polzovatelskiy-interfeys.html> (Дата обращения: 01.06.2024).

10. Простое руководство по диаграмме компонентов: [Электронный ресурс] // So creately URL: <https://creately.com/blog/ru/uncategorized-ru/учебное-пособие-по-компонентной-диаг/>. (Дата обращения: 04.06.2024).

11. Об одном подходе к разбиению на классы эквивалентности: [Электронный ресурс] // cyberleninka.ru. Теория. URL: <https://cyberleninka.ru/article/n/ob-odnom-podhode-k-razbieniyu-na-klassy-ekvivalentnosti> (дата обращения: 26.03.2024).

12. Граничные условия в неизолированных системах: [Электронный ресурс] // cyberleninka.ru. Теория. URL: <https://cyberleninka.ru/article/n/granichnye-usloviya-v-neizolirovannyh-sistemah>. (Дата обращения: 26.03.2024).

13. Простое сложное программирование: [Электронный ресурс] // <http://habr.com>. Теория. URL: <https://habr.com/ru/companies/timeweb/articles/551754/>. (Дата обращения: 26.03.2024).

14. Применение метрик Холстеда для количественного оценивания характеристик программ ЭВМ: [Электронный ресурс] // cyberleninka.ru. Теория. URL: <https://cyberleninka.ru/article/n/primenenie-metrik-holsteda-dlya-kolichestvennogo-otsenivaniya-harakteristik-programm-evm>. (Дата обращения: 26.03.2024).

Приложение А
(рекомендуемое)
Диаграммы UML

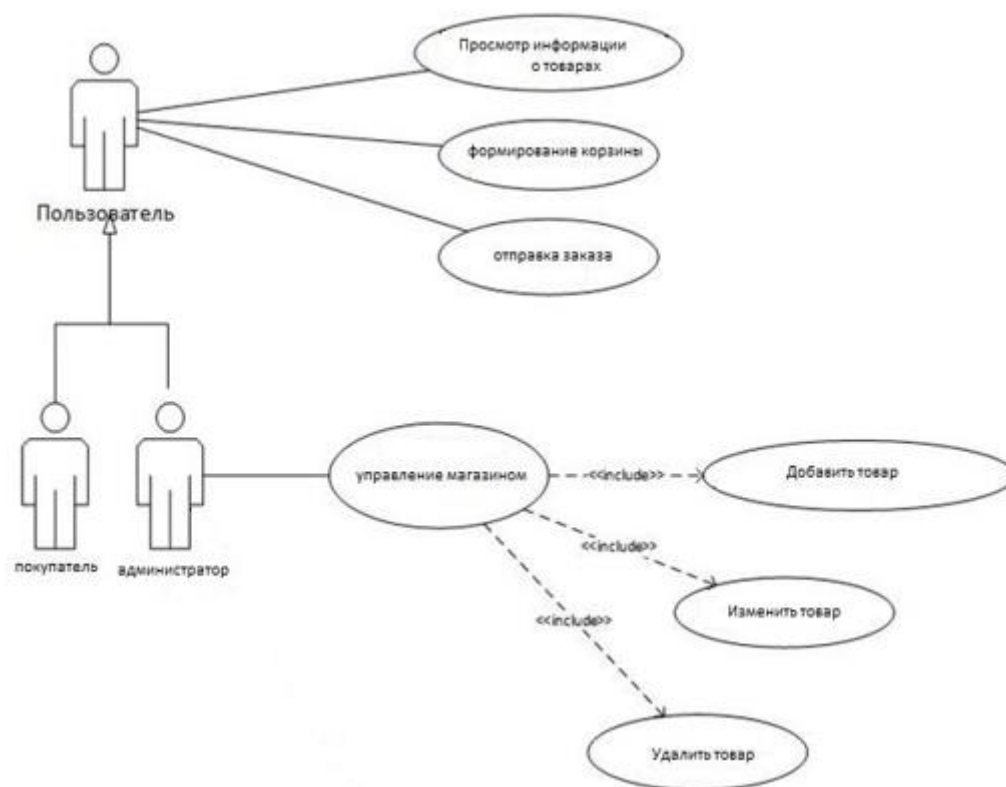


Рисунок А.1 – Диаграмма вариантов использования

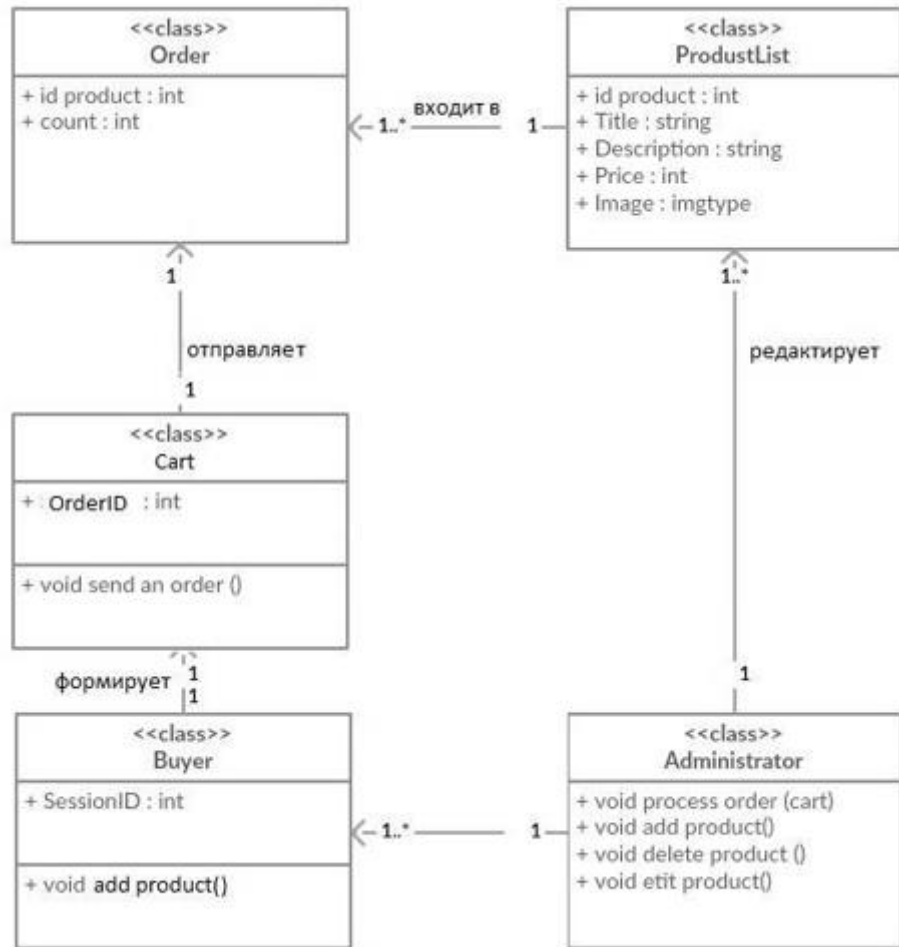


Рисунок А.2 – Диаграмма классов

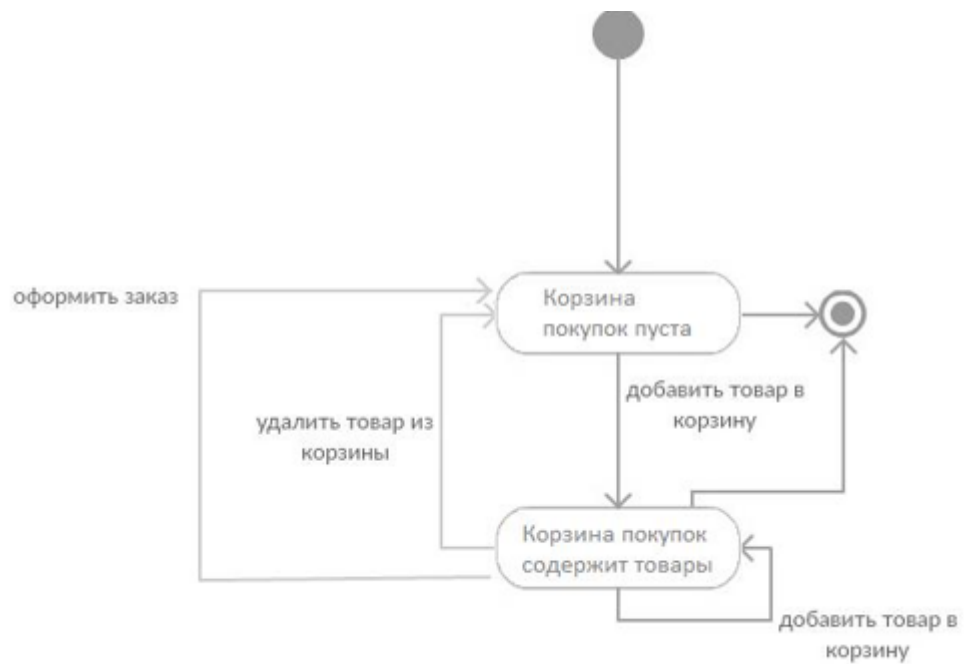


Рисунок А.3 – Диаграмма состояний

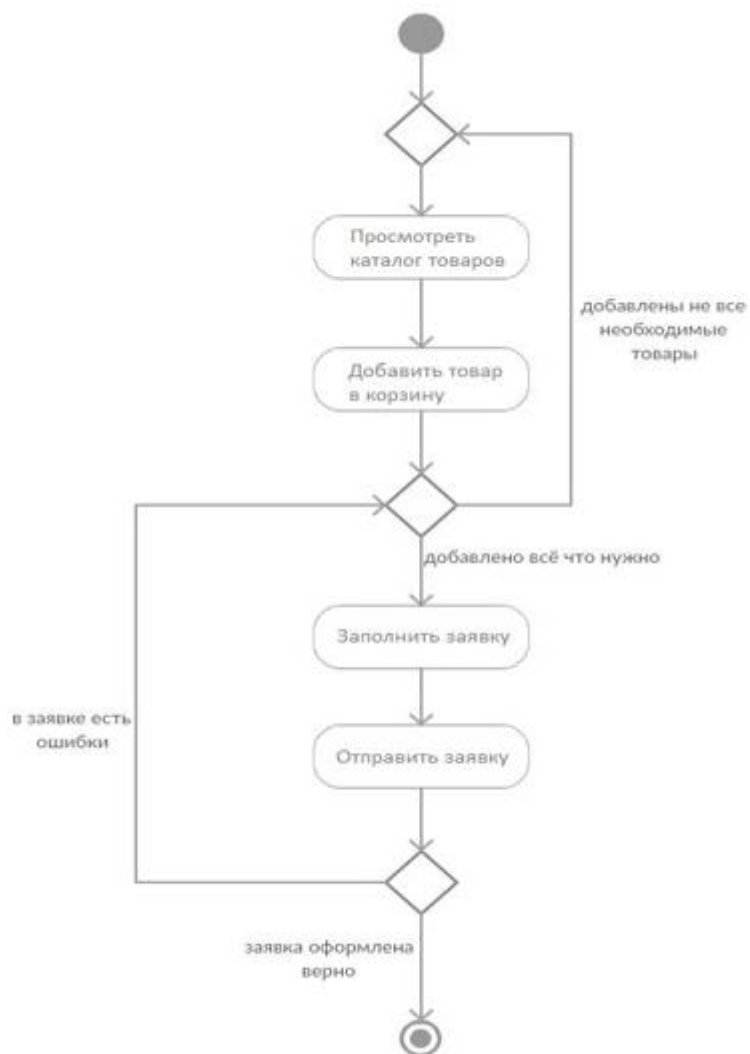


Рисунок А.4 – Диаграмма деятельности

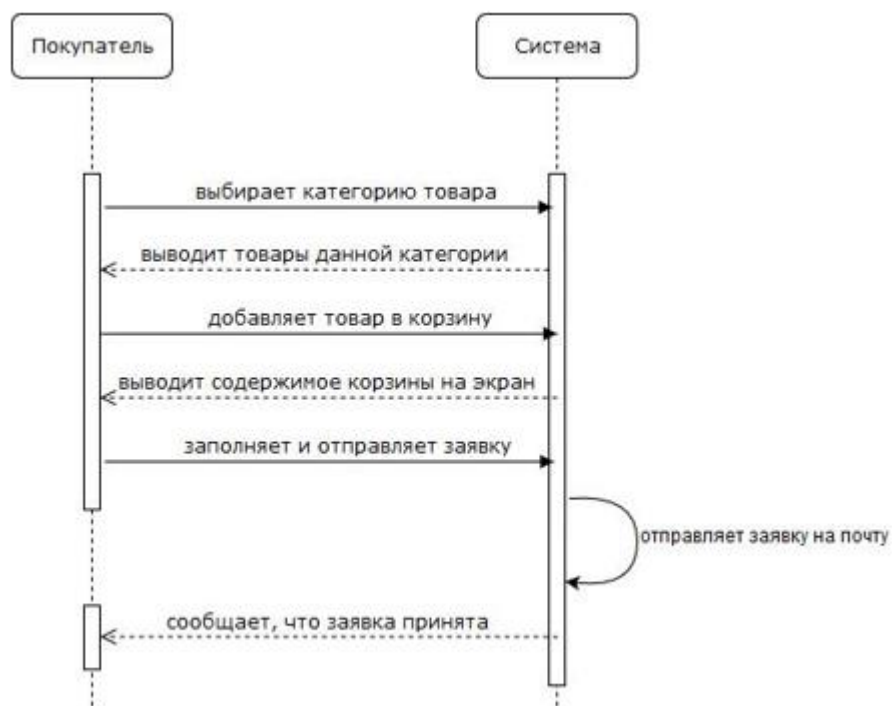


Рисунок А.5 – Диаграмма последовательности



Рисунок А.6 – Диаграмма коммуникаций

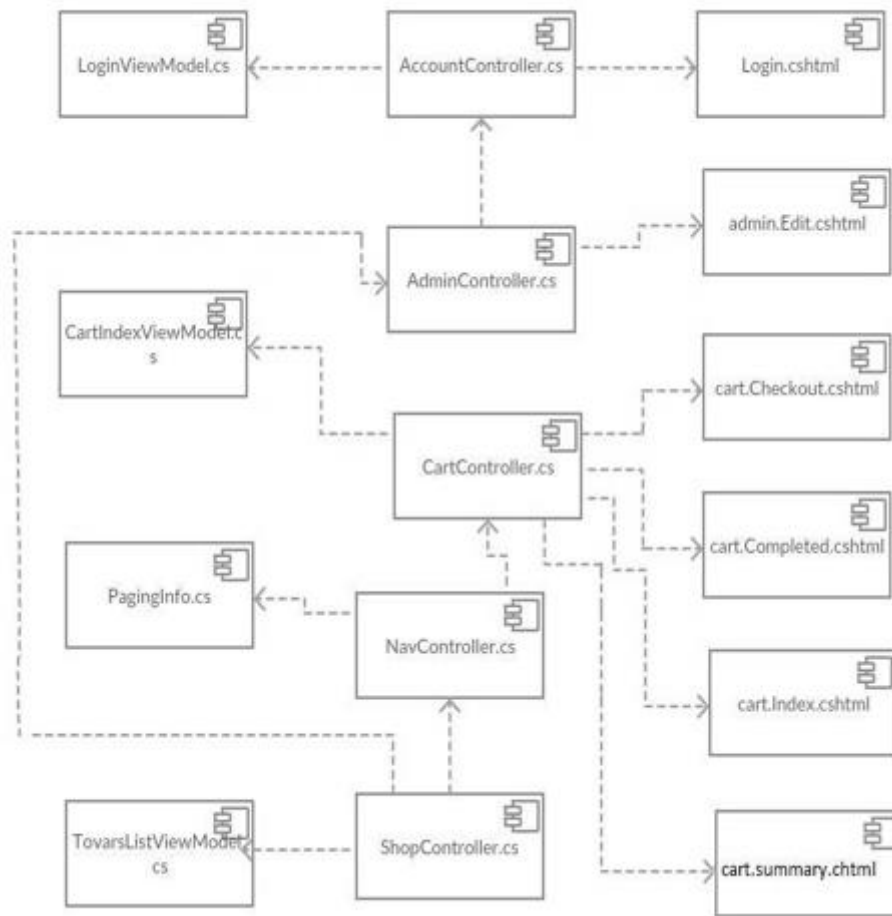


Рисунок А.7 – Диаграмма компонентов

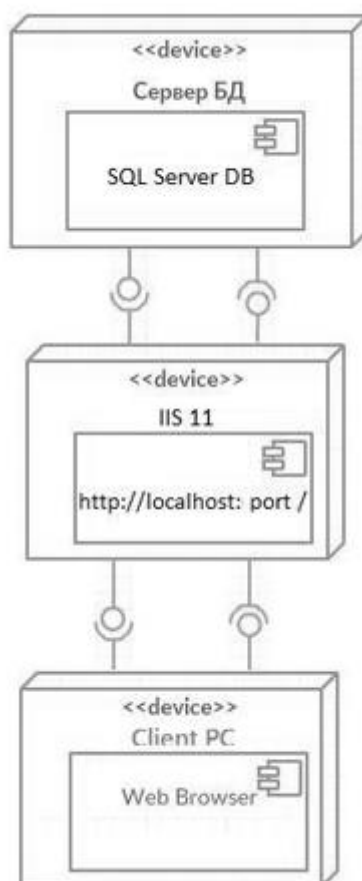


Рисунок А.8 – Диаграмма развертывания

Приложение Б
(рекомендуемое)
Листинг программы

Листинг Б.1 – Код модуля «Shop.Domain.Concrete.EFShopRepository.cs»

```

using System.Collections.Generic;

using Shop.Domain.Entities;
using Shop.Domain.Abstract;
using System.Linq;
using System.Text;

namespace Shop.Domain.Concrete
{
    public class EFShopRepository : IShopRepository
    {
        EFDbContext ccontext = new EFDbContext();

        public IEnumerable<Tovar> Tovar
        {
            get { return ccontext.Tovars; }
        }

        public void SaveTovar(Tovar tovar)
        {
            if (tovar.TovarId == 0)
                ccontext.Tovars.Add(tovar);

            else
            {
                Tovar dbEntry = ccontext.Tovars.Find(tovar.TovarId);
                if (dbEntry != null)
                {
                    dbEntry.Name = tovar.Name; dbEntry.Description =
                    tovar.Description; dbEntry.Price = tovar.Price; dbEntry.Category =
                    tovar.Category; dbEntry.ImageData = tovar.ImageData;

                    dbEntry.ImageMimeType = tovar.ImageMimeType;
                }
            }

            ccontext.SaveChanges();
        }

        public Tovar DeleteTovar(int tovarId)
        {
            Tovar dbEntry = ccontext.Tovars.Find(tovarId);
            if (dbEntry != null)

```

```

{
    ccontext.Tovars.Remove(dbEntry);
    ccontext.SaveChanges();
}

return dbEntry;
}
}

}

```

Листинг Б.2 – Код модуля «Shop.UnitTests. AdminSecurityTests.cs»

```

using System.Web.Mvc;

using Microsoft.VisualStudio.TestTools.UnitTesting; using Moq;
using Shop.WebUI.Controllers;

using Shop.WebUI.Infrastructure.Abstract; using Shop.WebUI.Models; using System.Linq;
using System.Text;

namespace Shop.UnitTests
{
    [TestClass]
    public class AdminSecurityTests
    {
        [TestMethod]

        public void Can_Login_With_Valid_Credentials()
        {
            // Организация - создание имитации поставщика аутентификации

            Mock<IAuthProvider>newmock = new Mock<IAuthProvider>();
            newmock.Setup(m => m.Authenticate("admin",
                "12345")).Returns(true);

            // Организация - создание модели представления
            // с правильными учетными данными
            LoginViewModel model = new LoginViewModel
            {
                UserName = "admin", Password = "12345" };

            // Организация - создание контроллера
            AccountController target = new
            AccountController(newmock.Object);

            // Действие - аутентификация
            ActionResult result = target.Login(model, "/MyURL");

```

```

//      Утверждение

Assert.IsInstanceOfType(result,      typeof(RedirectResult));
Assert.AreEqual("/MyURL", ((RedirectResult)result).Url); }

[TestMethod]
public void Cannot_Login_With_Invalid_Credentials()

{
//      Организация - созданиеимитациипоставщикааутентификации

Mock<IAuthProvider>newmock      =      new      Mock<IAuthProvider>();
newmock.Setup(m => m.Authenticate("badUser",
    "badPass")).Returns(false);

//      Организация - создание модели представления

//      с некорректными учетными данными
LoginViewModel model = new LoginViewModel
{

    UserName = "badUser", Password = "badPass" };

//      Организация - созданиеконтроллера
AccountController target = new
AccountController(newmock.Object);

//      Действие - аутентификация
ActionResult result = target.Login(model, "/MyURL");

//      Утверждение

Assert.IsInstanceOfType(result,      typeof(ViewResult));
Assert.IsFalse(((ViewResult)result).ViewData.ModelState.IsValid);

}
}
}

```

Листинг Б.3 – Код модуля «Shop.Domain.Concrete.EFDbContext.cs»

```

using System;
using System.Collections.Generic;
using System.Linq;

using System.Text;
using System.Threading.Tasks;
using Shop.Domain.Entities;

namespace Shop.Domain.Concrete
{

```

```

public class EFDbContext : DbContext
{
    public DbSet<Tovar> Tovars { get; set; }

}

}

```

Листинг Б.4 – Код модуля «Shop.UnitTests.CartTests.cs»

```

using System;

using Microsoft.VisualStudio.TestTools.UnitTesting;      using
System.Linq;
using System.Collections.Generic;
using Shop.Domain.Entities;
using Shop.Domain.Abstract;

using Moq;
using Shop.WebUI.Controllers;
using System.Web.Mvc;

using Shop.WebUI.Models;

namespace Shop.UnitTests
{
    [TestClass]
    public class CartTests
    {
        [TestMethod]
        public void Can_Add_New_Lines()
        {
            //      Организация - создание нескольких тестовых товаров

            Tovar1 = new Tovar { TovarId = 1, Name = "Товар1" }; Tovar
            2 = new Tovar { TovarId = 2, Name = "Товар2" };

            //      Организация - создание корзины

            Cart cart = new Cart();

            //      Действие cart.AddItem(tovar1, 1);

            cart.AddItem(tovar2, 1);
            List<CartLine> results = cart.Lines.ToList();

            //      Утверждение

```

```

        Assert.AreEqual(results.Count(), 2);
        Assert.AreEqual(results[0].Tovar, tovar1);
        Assert.AreEqual(results[1].Tovar, tovar2);
    }

    [TestMethod]
    public void Can_Add_Quantity_For_Existing_Lines()
    {
        // Организация - создание нескольких тестовых товаров

        Tovar tovar1 = new Tovar { TovarId = 1, Name = "Товар1" }; Tovar
        tovar2 = new Tovar { TovarId = 2, Name = "Товар2" };

        // Организация - создание корзины

        Cart cart = new Cart();

        // Действие cart.AddItem(tovar1, 1); cart.AddItem(tovar2, 1);
        cart.AddItem(tovar1, 5);
        List<CartLine> results = cart.Lines.OrderBy(c =>
            c.Tovar.TovarId).ToList();

        // Утверждение
        Assert.AreEqual(results.Count(), 2);

        Assert.AreEqual(results[0].Quantity, 6);
        // экземпляр добавлен в корзину

        Assert.AreEqual(results[1].Quantity, 1);

    }
    // 6
    [TestMethod]
    public void Can_Remove_Line()
    {
        // Организация - создание нескольких тестовых товаров

        Tovar tovar1 = new Tovar { TovarId = 1, Name = "Товар1" }; Tovar
        tovar2 = new Tovar { TovarId = 2, Name = "Товар2" }; Tovar tovar3 =
        new Tovar { TovarId = 3, Name = "Товар3" };

        // Организация - создание корзины

        Cart cart = new Cart();

        // Организация - добавление нескольких товаров в корзину
        cart.AddItem(tovar1, 1);

        cart.AddItem(tovar2, 4); cart.AddItem(tovar3, 2);
        cart.AddItem(tovar2, 1);

        // Действие

```

```

        cart.RemoveLine(tovar2);

//          Утверждение
Assert.AreEqual(cart.Lines.Where(c => c.Tovar ==
    tovar2).Count(), 0);
Assert.AreEqual(cart.Lines.Count(), 2);
}

[TestMethod]
public void Calculate_Cart_Total()
{
//          Организация - создание нескольких тестовых товаров

    Tovar tovar1 = new Tovar { TovarId = 1, Name = "Товар1", Price=
100 };

    Tovar tovar2 = new Tovar { TovarId = 2, Name = "Товар2",
Price = 55 };

//          Организация - создание корзины

    Cart cart = new Cart();

//          Действие
    cart.AddItem(tovar1, 1);

    cart.AddItem(tovar2, 1);
    cart.AddItem(tovar1, 5);
    decimal result = cart.ComputeTotalValue();

//          Утверждение
Assert.AreEqual(result, 655);
}

[TestMethod]
public void Can_Clear_Contents()
{
//          Организация - создание нескольких тестовых товаров

    Tovar tovar1 = new Tovar { TovarId = 1, Name = "Товар1", Price =
100 };

    Tovar tovar2 = new Tovar { TovarId = 2, Name = "Товар2",
Price = 55 };

//          Организация - создание корзины

    Cart cart = new Cart();

//          Действие

```

```

        cart.AddItem(tovar1, 1);
        cart.AddItem(tovar2, 1);
        cart.AddItem(tovar1, 5);
        cart.Clear();

//          Утверждение
Assert.AreEqual(cart.Lines.Count(), 0);

    }

    [TestMethod]

    public void Can_Add_To_Cart()
    {
//          Организация - созданиеимитированногохранилища

        Mock<IShopRepository>newmock = new Mock<IShopRepository>();
        newmock.Setup(m => m.Tovar).Returns(new List<Tovar> {

            new Tovar {TovarId = 1, Name = "Товар1", Category = "Кат1"},
        }.AsQueryable());

//          Организация - созданиекорзины
        Cartcart = newCart();

//          Организация - созданиеконтроллера
        CartController controller = new
            CartController(newmock.Object, null);

//          Действие - добавить товар в корзину
        controller.AddToCart(cart, 1, null);

//          Утверждение

        Assert.AreEqual(cart.Lines.Count(), 1);
        Assert.AreEqual(cart.Lines.ToList()[0].Tovar.TovarId, 1);

    }

    ///          <summary>

    ///          После добавления товара в корзину, должно быть
    перенаправление на страницу корзины
    ///          </summary>
    [TestMethod]

    public void Adding_Tovar_To_Cart_Goes_To_Cart_Screen()
    {
//          Организация - созданиеимитированногохранилища

        Mock<IShopRepository>newmock = new Mock<IShopRepository>();
        newmock.Setup(m => m.Tovar).Returns(new List<Tovar> {

```

```

        new Tovar {TovarId = 1, Name = "Товар1", Category = "Кат1"},
    }.AsQueryable());

//      Организация - создание корзины
    Cart cart = new Cart();

//      Организация - создание контроллера
    CartController controller = new
    CartController(newmock.Object, null);

//      Действие - добавить товар в корзину

    RedirectToRouteResult result = controller.AddToCart(cart, 2,
    "myUrl");

//      Утверждение

    Assert.AreEqual(result.RouteValues["action"], "Index");
    Assert.AreEqual(result.RouteValues["returnUrl"], "myUrl");
    }

//      Проверяем URL [TestMethod]
    public void Can_View_Cart_Contents()

    {
//      Организация - создание корзины
        Cart cart = new Cart();

//      Организация - создание контроллера
        CartController target = new CartController(null, null);

//      Действие - вызов метода действия Index()
        CartIndexViewModel result
        = (CartIndexViewModel)target.Index(cart,
        "myUrl").ViewData.Model;

//      Утверждение

        Assert.AreSame(result.Cart, cart);
        Assert.AreEqual(result.ReturnUrl, "myUrl");
    }

    [TestMethod]

    public void Cannot_Checkout_Empty_Cart()
    {
//      Организация - создание имитированного обработчика заказов

        Mock<IOrderProcessor>newmock = new Mock<IOrderProcessor>();

//      Организация - создание пустой корзины
        Cart cart = new Cart();

//      Организация - создание деталей доставки

```

```

        ShippingDetailsshoppingDetails = newShippingDetails();

//          Организация - созданиеконтроллера

        CartController controller = new CartController(null,
newmock.Object);

//          Действие

        ViewResult result = controller.Checkout(cart,
shippingDetails);

//          Утверждение - проверка, что заказ не был передан
обработчику

        newmock.Verify(m => m.ProcessOrder(It.IsAny<Cart>(),
It.IsAny<ShippingDetails>()),
        Times.Never());

//          Утверждение - проверка, что метод вернул стандартное
представление
        Assert.AreEqual("", result.ViewName);

//          Утверждение - проверка, что-представлению передана
неверная модель

        Assert.AreEqual(false, result.ViewData.ModelState.IsValid); }

[TestMethod]
public void Can_Checkout_And_Submit_Order()
{
//          Организация - созданиеимитированногообработчиказаказов

        Mock<IOrderProcessor>newmock = new Mock<IOrderProcessor>();

//          Организация - создание корзины с элементом

        Cart cart = new Cart();
        cart.AddItem(new Tovar(), 1);

//          Организация - создание контроллера

        CartController controller = new CartController(null,
newmock.Object);

//          Действие - попыткапереходакоплате

        ViewResult result = controller.Checkout(cart, new
ShippingDetails());

//          Утверждение - проверка, что заказ передан обработчику
newmock.Verify(m => m.ProcessOrder(It.IsAny<Cart>(),
        It.IsAny<ShippingDetails>()),

```

```

        Times.Once());

//      Утверждение - проверка, что метод возвращает
представление

        Assert.AreEqual("Completed", result.ViewName);

//      Утверждение - проверка, что представлению передается
допустимая модель
        Assert.AreEqual(true, result.ViewData.ModelState.IsValid);
    }
}

}

```

Листинг Б.5 – Код модуля Shop.UnitTests. AdminTests.cs

```

using Shop.WebUI.Controllers;

using Microsoft.VisualStudio.TestTools.UnitTesting; using Moq;

using Shop.Domain.Abstract;
using Shop.Domain.Entities;
using System;

using System.Collections.Generic;
using System.Linq;
using System.Web.Mvc;
using System.Text;

namespace Shop.UnitTests
{
    [TestClass]

    public class AdminTests
    {
        [TestMethod]

        public void Index_Contains_All_Tovarss()
        {
//      Организация - созданиеимитированногохранилищаданных

            Mock<IShopRepository>newmock = new Mock<IShopRepository>();
            newmock.Setup(m => m.Tovar).Returns(new List<Tovar>
            {

                new Tovar { TovarId = 1, Name = "Товар1"}, new Tovar {
TovarId = 2, Name = "Товар2"}, new Tovar { TovarId = 3, Name =
"Товар3"}, new Tovar { TovarId = 4, Name = "Товар4"}, new Tovar {
TovarId = 5, Name = "Товар5"} });

//      Организация - созданиеконтроллера

```

```

        AdminController controller = new
AdminController(newmock.Object);

//        Действие

        List<Tovar> result =
            ((IEnumerable<Tovar>)controller.Index().
            ViewData.Model).ToList();

//        Утверждение

        Assert.AreEqual(result.Count(), 5); Assert.AreEqual("Товар1",
result[0].Name); Assert.AreEqual("Товар2", result[1].Name);
Assert.AreEqual("Товар3", result[2].Name);
    }
    [TestMethod]

    public void Can_Edit_Tovar()
    {
//        Организация - созданиеимитированногохранилищаданных

        Mock<IShopRepository>newmock = new Mock<IShopRepository>();
newmock.Setup(m => m.Tovar).Returns(new List<Tovar>
    {

        new Tovar { TovarId = 1, Name = "Товар1"}, new Tovar {
TovarId = 2, Name = "Товар2"}, new Tovar { TovarId = 3, Name =
"Товар3"}, new Tovar { TovarId = 4, Name = "Товар4"}, new Tovar {
TovarId = 5, Name = "Товар5"} });
//        Организация - созданиеконтроллера
        AdminController controller = new
AdminController(newmock.Object);

//        Действие

        Tovar tovar1 = controller.Edit(1).ViewData.Model as Tovar; Tovar
tovar2 = controller.Edit(2).ViewData.Model as Tovar; Tovar tovar3 =
controller.Edit(3).ViewData.Model as Tovar;

        // Assert

        Assert.AreEqual(1, tovar1.TovarId);
        Assert.AreEqual(2, tovar2.TovarId);
        Assert.AreEqual(3, tovar3.TovarId);
    }

    [TestMethod]
    public void Cannot_Edit_Nonexistent_Tovar()

    {
//        Организация - созданиеимитированногохранилищаданных

```

```

    Mock<IShopRepository>newmock = new Mock<IShopRepository>();
    newmock.Setup(m => m.Tovar).Returns(new List<Tovar>

        {

            new Tovar { TovarId = 1, Name = "Товар1"}, new Tovar {
TovarId = 2, Name = "Товар2"}, new Tovar { TovarId = 3, Name =
"Товар3"}, new Tovar { TovarId = 4, Name = "Товар4"}, new Tovar {
TovarId = 5, Name = "Товар5"} });

//      Организация - создание контроллера

    AdminController controller = new
AdminController(newmock.Object);

//      Действие
    Tovar result = controller.Edit(6).ViewData.Model as Tovar;

//      Assert
    }

    [TestMethod]
    public void Can_Save_Valid_Changes()
    {

//      Организация - создание имитированного хранилища данных
        Mock<IShopRepository>newmock = new Mock<IShopRepository>();

//      Организация - создание контроллера

        AdminController controller = new
AdminController(newmock.Object);

//      Организация - создание объекта Tovar
        Tovar tovar = new Tovar
{ Name = "Test" };

//      Действие - попытка сохранения товара
        ActionResult result = controller.Edit(tovar);

//      Утверждение - проверка того, что к хранилищу производится
        обращение
        newmock.Verify(m => m.SaveTovar(tovar));

//      Утверждение - проверка типа результата метода
        Assert.IsNotInstanceOfType(result, typeof(ViewResult));
    }

    [TestMethod]
    public void Cannot_Save_Invalid_Changes()
    {

//      Организация - создание имитированного хранилища данных
        Mock<IShopRepository>newmock = new Mock<IShopRepository>();

```

```

//      Организация - создание контроллера
AdminController controller = new
AdminController(newmock.Object);

//      Организация - создание объекта Tovar Tovar tovar = new Tovar
{ Name = "Test" };

//      Организация - добавление ошибки в состояние модели
controller.ModelState.AddModelError("error", "error");

//      Действие - попытка сохранения товара
ActionResult result = controller.Edit(tovar);

//      Утверждение - проверка того, что обращение к хранилищу НЕ
производится

newmock.Verify(m => m.SaveTovar(It.IsAny<Tovar>()),
Times.Never());

//      Утверждение - проверка типа результата метода
Assert.IsInstanceOfType(result, typeof(ViewResult));
}

[TestMethod]
public void Can_Delete_Valid_Tovars()
{

//      Организация - создание объекта Tovar
Tovar tovar = new Tovar { TovarId = 2, Name = "Товар2" };

//      Организация - создание имитированного хранилища данных
Mock<IShopRepository>newmock = new Mock<IShopRepository>();
newmock.Setup(m => m.Tovar).Returns(new List<Tovar>
{

    new Tovar { TovarId = 1, Name = "Товар1"}, new Tovar { TovarId =
2, Name = "Товар2"}, new Tovar { TovarId = 3, Name = "Товар3"},

    new Tovar { TovarId = 4, Name = "Товар4"}, new Tovar {
TovarId = 5, Name = "Товар5" } });

//      Организация - создание контроллера
AdminController controller = new
AdminController(newmock.Object);

//      Действие - удаление товара controller.Delete(tovar.TovarId);

//      Утверждение - проверка того, что метод удаления в
хранилище

```

```
//      вызывается для корректного объекта Товар newmock.Verify(m
=>m.DeleteTovar(tovar.TovarId));
    }
}

}
```

Листинг Б.6 –Код модуля «Shop.UnitTests.UnitTest1.cs»

```
using System;
using System.Collections.Generic;
using System.Linq;

using System.Web.Mvc;

using Microsoft.VisualStudio.TestTools.UnitTesting; using Moq;
using Shop.Domain.Abstract;

using Shop.Domain.Entities;
using Shop.WebUI.Controllers;
using Shop.WebUI.Models;

using Shop.WebUI.HtmlHelpers;

namespace Shop.UnitTests
{
    [TestClass]
    public class UnitTest1
    {
        [TestMethod]
        public void Can_Paginate()
        {
            //      Организация (arrange)

            Mock<IShopRepository>newmock = new Mock<IShopRepository>();
            newmock.Setup(m => m.Tovar).Returns(new List<Tovar> {

                new Tovar { TovarId = 1, Name = "Товар1"}, new Tovar {
TovarId = 2, Name = "Товар2"}, new Tovar { TovarId = 3, Name =
"Товар3"}, new Tovar { TovarId = 4, Name = "Товар4"}, new Tovar {
TovarId = 5, Name = "Товар5"} });

            ShopController controller = new
ShopController(newmock.Object);

            controller.pageSize = 3;

            //      Действие (act) TovarsListViewModel result =
(TovarsListViewModel)controller.List(null, 2).Model;

            //      Утверждение
```

```

        List<Tovar>        tovars        =        result.Tovars.ToList();
Assert.IsTrue(tovars.Count == 2); Assert.AreEqual(tovars[0].Name,
"Товар4"); Assert.AreEqual(tovars[1].Name, "Товар5");
    }
    [TestMethod]
    public void Can_Generate_Page_Links()

    {

//        Организация - определение вспомогательного метода HTML -
это необходимо

//        для применения расширяющего метода
HtmlHelper myHelper = null;

//        Организация - создание объекта PagingInfo
PagingInfo pagingInfo = new PagingInfo
{

    CurrentPage = 2, TotalItems = 28, ItemsPerPage = 10 };

//        Организация - настройка делегата с помощью лямбда-
выражения
Func<int, string> pageUrlDelegate = i => "Page" + i;

//        Действие

MvcHtmlString    result    =    myHelper.PageLinks(pagingInfo,
pageUrlDelegate);

//        Утверждение
Assert.AreEqual(@"<a class=""btn btn-default""
href=""Page1"">1</a>"
+ @"<a class=""btn btn-default btn-primary selected""
href=""Page2"">2</a>"
+ @"<a class=""btn btn-default"" href=""Page3"">3</a>",
result.ToString());
    }
    [TestMethod]

    public void Can_Send_Pagination_View_Model()
    {
// Организация (arrange)

Mock<IShopRepository>newmock = new Mock<IShopRepository>();
newmock.Setup(m => m.Tovar).Returns(new List<Tovar>
{

    new Tovar { TovarId = 1, Name = "Товар1"}, new Tovar {
TovarId = 2, Name = "Товар2"}, new Tovar { TovarId = 3, Name =

```

```

"Товар3"}, new Tovar { TovarId = 4, Name = "Товар4"}, new Tovar {
TovarId = 5, Name = "Товар5"} });
    ShopController controller = new
    ShopController(newmock.Object);

    controller.pageSize = 3;

    // Act

    TovarListViewModel result
    = (TovarListViewModel)controller.List(null, 2).Model;

    // Assert

    PagingInfo pageInfo = result.PagingInfo;
    Assert.AreEqual(pageInfo.CurrentPage, 2);
    Assert.AreEqual(pageInfo.ItemsPerPage, 3);
    Assert.AreEqual(pageInfo.TotalItems, 5);
    Assert.AreEqual(pageInfo.TotalPages, 2); }

[TestMethod]
public void Can_Create_Categories()

{
//      Организация - созданиеимитированногохранилища

    Mock<IShopRepository>newmock = new Mock<IShopRepository>();
    newmock.Setup(m => m.Tovar).Returns(new List<Tovar> {
        new Tovar { TovarId = 1, Name = "Товар1",
        Category="Косметика3"},

        new Tovar { TovarId = 2, Name = "Товар2",
        Category="Косметика2"},

        new Tovar { TovarId = 3, Name = "Товар3",
        Category="КОСМЕТИКА1"}, new Tovar { TovarId = 4, Name = "Товар4",
        Category="КОСМЕТИКА1"}, });

//      Организация - созданиеконтроллера
    NavController target = new NavController(newmock.Object);

//      Действие - получениенаборакатегорий
    List<string> results =

        ((IEnumerable<string>)target.Menu().Model).ToList();

//      Утверждение

    Assert.AreEqual(results.Count(), 3); Assert.AreEqual(results[1],
    "Косметика3"); Assert.AreEqual(results[0], "Косметика2");
    Assert.AreEqual(results[2], "КОСМЕТИКА1");
}

[TestMethod]

```

```

    public void Indicates_Selected_Category()
    {
//        Организация - созданиеимитированногохранилища

        Mock<IShopRepository>newmock = new Mock<IShopRepository>();
newmock.Setup(m => m.Tovar).Returns(new Tovar[] {

            new Tovar { TovarId = 1, Name = "Товар1",
            Category="Косметика2"},

            new Tovar { TovarId = 2, Name = "Товар2",
            Category="Кометика4"} });

// Организация - созданиеконтроллера

        NavController target = new NavController(newmock.Object);

//        Организация - определение выбранной категории string
categoryToSelect = "Косметика2";

//        Действие
string result =

        target.Menu(categoryToSelect).ViewBag.SelectedCategory;

//        Утверждение

        Assert.AreEqual(categoryToSelect, result);
    }

[TestMethod]

    public void Generate_Category_Specific_Tovar_Count()
    {
///        Организация (arrange)

        Mock<IShopRepository>newmock = new Mock<IShopRepository>();
newmock.Setup(m => m.Tovar).Returns(new List<Tovar> {

            new Tovar { TovarId = 1, Name = "Товар1", Category="Cat2"},
            new Tovar { TovarId = 2, Name = "Товар2", Category="Cat3"}, new
            Tovar { TovarId = 3, Name = "Товар3", Category="Cat2"}, new Tovar
            { TovarId = 4, Name = "Товар4", Category="Cat1"}, new Tovar {
            TovarId = 5, Name = "Товар5", Category="Cat1"} });

        ShopController controller = new ShopController(mock.Object);
        controller.pageSize = 3;

//        Действие - тестирование счетчиков товаров для различных
        категорий

```

```

        int                                res1                                =
        ((TovarsListViewModel)controller.List("Cat1").Model).PagingInfo.
        TotalItems;

        int                                res2                                =
        ((TovarsListViewModel)controller.List("Cat2").Model).PagingInfo.
        TotalItems;

        int                                res3                                =
        ((TovarsListViewModel)controller.List("Cat3").Model).PagingInfo.
        TotalItems;

        int                                resAll                             =
        ((TovarsListViewModel)controller.List(null).Model).PagingInfo.To
        talItems;

//      Утверждение
        Assert.AreEqual(res1, 2);

        Assert.AreEqual(res2, 2);
        Assert.AreEqual(res3, 1);
        Assert.AreEqual(resAll, 5);

```

Листинг Б.7 –Код модуля «Shop.UnitTests.ImageTests.cs»

```

using Microsoft.VisualStudio.TestTools.UnitTesting; using Moq;

using Shop.Domain.Abstract;
using Shop.Domain.Entities;
using Shop.WebUI.Controllers;

using System.Linq;
using System.Web.Mvc;
using System.Collections.Generic;

namespace Shop.UnitTests
{
    [TestClass]

    public class ImageTests

    {
        [TestMethod]
        public void Can_Retrieve_Image_Data()

        {

//      Организация - создание объекта Товар с данными
//      изображения
            Товар товар = new Товар

            {

```

```

        TovarId = 2, Name = "Товар2",

        ImageData = new byte[] { }, ImageMimeType = "image/png" };

//          Организация - создание имитированного хранилища

        Mock<IShopRepository>newmock = new Mock<IShopRepository>();
newmock.Setup(m => m.Tovar).Returns(new List<Tovar> {

        new Tovar {TovarId = 1, Name = "Товар1"}, tovar,

        new Tovar {TovarId = 3, Name = "товар3"} }.AsQueryable());

//          Организация - создание контроллера

        ShopController controller = new
ShopController(newmock.Object);

//          Действие - вызов метода действия GetImage()
        ActionResult result = controller.GetImage(2);

//          Утверждение
        Assert.IsNotNull(result);

        Assert.IsInstanceOfType(result, typeof(FileResult));
        Assert.AreEqual(tovar.ImageMimeType,
        ((FileResult)result).ContentType);
    }

    [TestMethod]
    public void Cannot_Retrieve_Image_Data_For_Invalid_ID()
    {

//          Организация - создание имитированного хранилища

        Mock<IShopRepository>newmock = new Mock<IShopRepository>();
newmock.Setup(m => m.Tovar).Returns(new List<Tovar> {

        new Tovar {TovarId = 1, Name = "Товар1"}, new Tovar {TovarId
= 2, Name = "Товар2"} }.AsQueryable());

//          Организация - создание контроллера

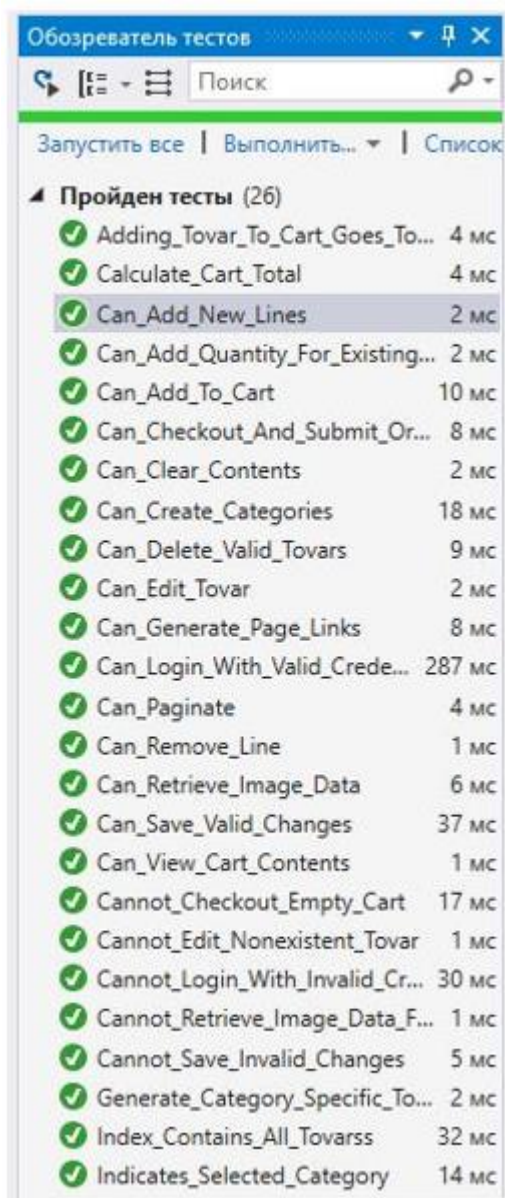
        ShopController controller = new
ShopController(newmock.Object);

//          Действие - вызов метода действия GetImage()
        ActionResult result = controller.GetImage(10);

//          Утверждение
        Assert.IsNull(result);

```

Приложение В
(рекомендуемое)
Результаты тестирования



Обозреватель тестов

Поиск

Запустить все | Выполнить... | Список

▲ Пройден тесты (26)

✓ Adding_Tovar_To_Cart_Goes_To...	4 мс
✓ Calculate_Cart_Total	4 мс
✓ Can_Add_New_Lines	2 мс
✓ Can_Add_Quantity_For_Existing...	2 мс
✓ Can_Add_To_Cart	10 мс
✓ Can_Checkout_And_Submit_Or...	8 мс
✓ Can_Clear_Contents	2 мс
✓ Can_Create_Categories	18 мс
✓ Can_Delete_Valid_Tovars	9 мс
✓ Can_Edit_Tovar	2 мс
✓ Can_Generate_Page_Links	8 мс
✓ Can_Login_With_Valid_Crede...	287 мс
✓ Can_Paginate	4 мс
✓ Can_Remove_Line	1 мс
✓ Can_Retrieve_Image_Data	6 мс
✓ Can_Save_Valid_Changes	37 мс
✓ Can_View_Cart_Contents	1 мс
✓ Cannot_Checkout_Empty_Cart	17 мс
✓ Cannot_Edit_Nonexistent_Tovar	1 мс
✓ Cannot_Login_With_Invalid_Cr...	30 мс
✓ Cannot_Retrieve_Image_Data_F...	1 мс
✓ Cannot_Save_Invalid_Changes	5 мс
✓ Generate_Category_Specific_To...	2 мс
✓ Index_Contains_All_Tovarss	32 мс
✓ Indicates_Selected_Category	14 мс

Рисунок В.1 – Результаты модульного тестирования

Корзина:

Тестовый товар 1 - 1шт - 500 руб. - 500 руб.	X
Тестовый товар 2 - 1шт - 400 руб. - 400 руб.	X

Итого: 900 руб.

Оформить заказ

Рисунок В.2 – Результаты тестирования работы корзины

Корзина:

Тестовый товар 1 - 1шт - 500 руб. - 500 руб.	X
Тестовый товар 2 - 2шт - 400 руб. - 800 руб.	X
Тестовый товар 3 - 1шт - 100 руб. - 100 руб.	X

Итого: 1400 руб.

Оформить заказ

Рисунок В.3 – Результаты тестирования добавления нового товара в корзину

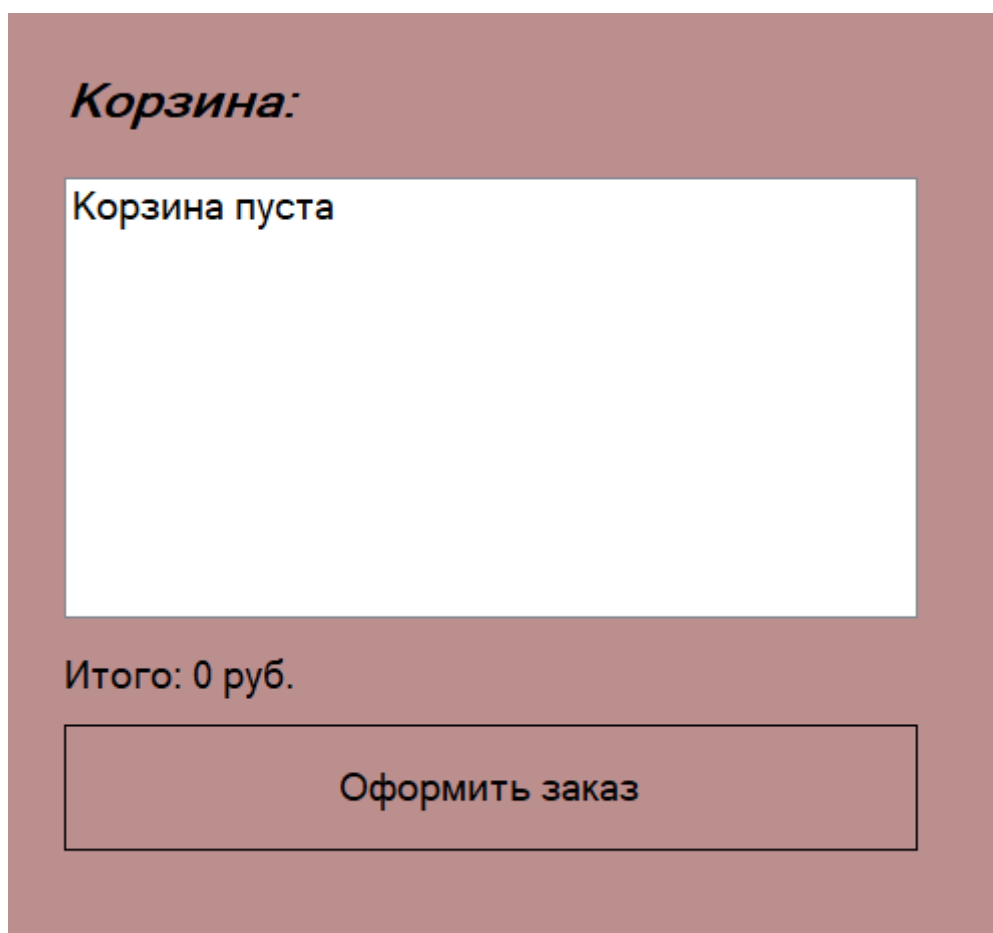


Рисунок В.4 – Результаты тестирования очистки корзины

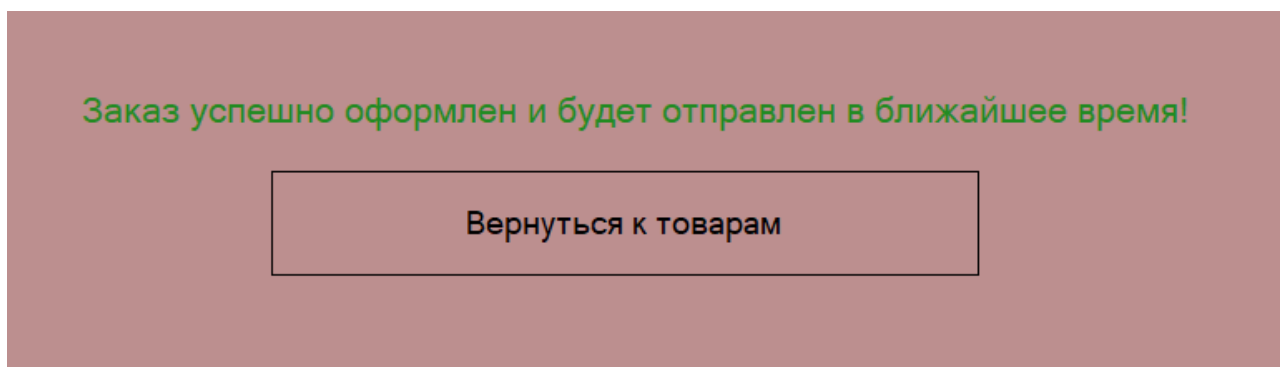


Рисунок В.5 – Результаты тестирования оформления заказа

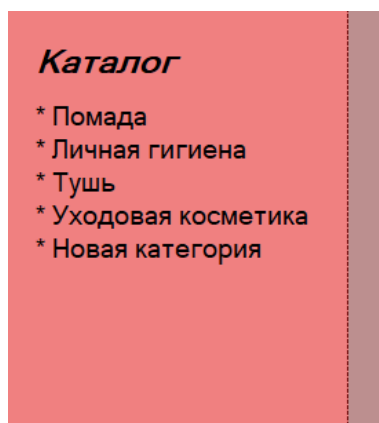


Рисунок В.6 – Результаты тестирования просмотра каталога

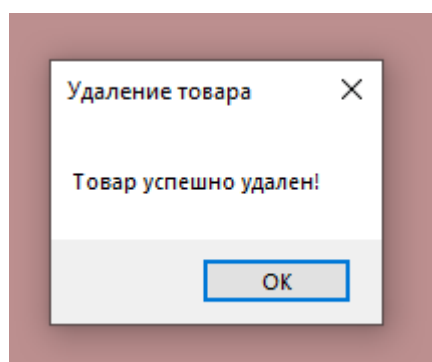


Рисунок В.7 – Результаты тестирования удаления товара

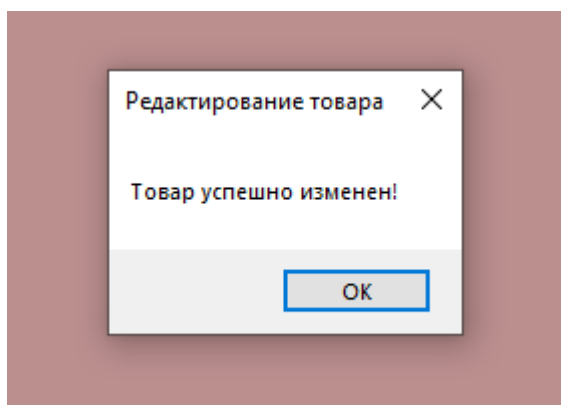


Рисунок В.8 – Результаты тестирования редактирования товара

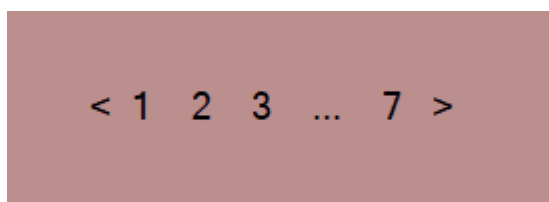


Рисунок В.9 – Результаты тестирования разбиения на страницы

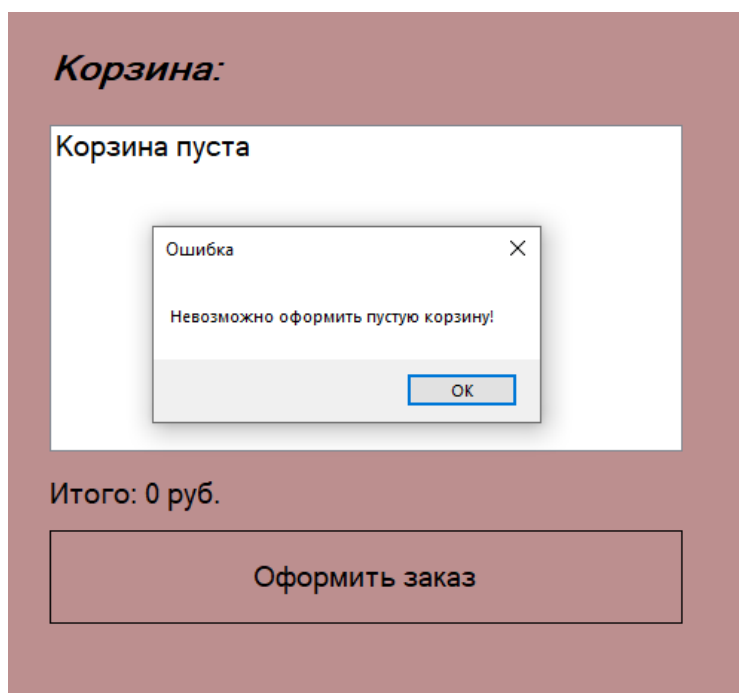


Рисунок В.10 – Результаты тестирования проверки на полноту корзины

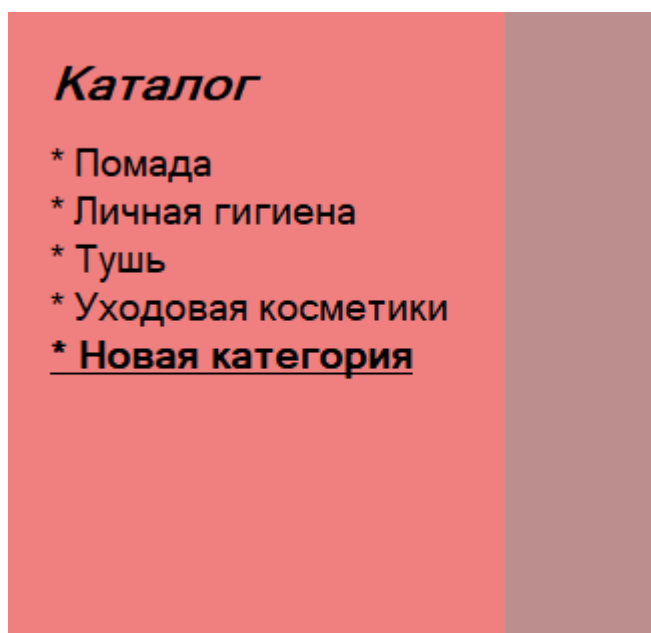


Рисунок В.11 – Результаты тестирования подсветки категории



Рисунок В.12 – Результаты тестирования изменения изображения