Учреждение образования  
 «Полоцкий государственный экономический колледж»

**ДНЕВНИК ОТЧЕТА  
по учебной практике**

**«Разработка и сопровождение программного обеспечения»**

Тема: «Каталог канцелярских товаров»

за период с 20.10.2025 г. по 17.11.2025 г.

|  |  |  |
| --- | --- | --- |
| Выполнил: |  |  |
| учащийся 3 курса группы П33  специальности 5-04-0612-02 «Разработка и сопровождение программного обеспечения информационных систем» | \_\_\_\_\_\_\_\_\_\_\_  (подпись) | А.В. Гайдель |
| Руководитель: |  |  |
| преподаватель | \_\_\_\_\_\_\_\_\_\_\_\_  (подпись) | Ю.Д. Симанович |

Дата сдачи дневника отчета \_\_\_.\_\_.\_\_\_

Дата защиты практики \_\_\_.\_\_.\_\_\_

Отметка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Полоцк 2025

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 9](#_Toc214030186)

[1 АНАЛИЗ ИСХОДНЫХ ДАННЫХ И ПОСТАНОВКА ЗАДАЧИ 10](#_Toc214030187)

[1.1 Описание предметной области 10](#_Toc214030188)

[1.2 Сравнительный анализ аналогов и прототипов 12](#_Toc214030189)

[1.3 Разработка технического задания 17](#_Toc214030190)

[1.4 Выводы по первому разделу 21](#_Toc214030191)

[2 СОЗДАНИЕ ИНФОРМАЦИОННОЙ БАЗЫ ПРОЕКТА 23](#_Toc214030192)

[2.1 Разработка концептуальной модели базы данных 23](#_Toc214030193)

[2.2 Разработка модели «сущность-связь» 23](#_Toc214030194)

[2.3 Разработка физической модели базы данных 27](#_Toc214030195)

[2.4 Выводы по второму разделу 31](#_Toc214030196)

[3 СТРУКТУРНЫЙ ПОДХОД К ПРОЕКТИРОВАНИЮ ПО 32](#_Toc214030197)

[3.1 Функциональное моделирование 32](#_Toc214030198)

[3.2 Моделирование бизнес-процессов 34](#_Toc214030199)

[4 ОБЪЕКТНЫЙ ПОДХОД К ПРОЕКТИРОВАНИЮ И РАЗРАБОТКЕ ПО. ЯЗЫКУ UML 37](#_Toc214030200)

[4.1 Разработка диаграммы вариантов использования 37](#_Toc214030201)

[4.2 Разработка диаграммы классов 39](#_Toc214030202)

[4.3 Разработка диаграммы последовательности 45](#_Toc214030203)

[4.4 Разработка диаграммы деятельности 46](#_Toc214030204)

[4.5 Разработка диаграммы состояний 47](#_Toc214030205)

[5 ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСА 51](#_Toc214030206)

[5.1 Проектирование интерфейса ПО 51](#_Toc214030207)

[6 ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА 54](#_Toc214030208)

[7 РЕАЛИЗАЦИЯ ПРОГРАММНОГО СРЕДСТВА 56](#_Toc214030209)

[7.1 Структура программного средства 56](#_Toc214030210)

[7.2 Диаграмма развертывания 56](#_Toc214030211)

[7.3 Установка программного средства 57](#_Toc214030212)

[8 СОПРОВОЖДЕНИЕ ПРОГРАММНОГО СРЕДСТВА 58](#_Toc214030213)

[ЗАКЛЮЧЕНИЕ 59](#_Toc214030214)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 60](#_Toc214030215)

[ПРИЛОЖЕНИЕ A Техническое задание 61](#_Toc214030216)

[ПРИЛОЖЕНИЕ Б SQL-сценарий базы данных 64](#_Toc214030226)

[ПРИЛОЖЕНИЕ В Диаграмма модели IDEF-0 86](#_Toc214030227)

[ПРИЛОЖЕНИЕ Г Диаграмма модели DFD 89](#_Toc214030228)

[ПРИЛОЖЕНИЕ Д Диаграмма вариантов использования 92](#_Toc214030229)

[ПРИЛОЖЕНИЕ Е Диаграмма классов 93](#_Toc214030230)

*8*

*8*

ПРИЛОЖЕНИЕ Ж Диаграмма последовательности 94

ПРИЛОЖЕНИЕ И Диаграмма деятельности 96

ПРИЛОЖЕНИЕ К Диаграмма состояний 98

ПРИЛОЖЕНИЕ Л Проект экранных форм 99

ПРИЛОЖЕНИЕ М Диаграмма развертывания 103

ПРИЛОЖЕНИЕ Н Программа и методика испытаний 104

ПРИЛОЖЕНИЕ П Руководство оператора 120

# ВВЕДЕНИЕ

Информационные технологии – представляют собой применение вычислительной техники и программных средств для обработки, хранения и передачи информации. В современном мире ИТ играют ключевую роль, так как значительная часть процессов в различных сферах деятельности осуществляется в цифровой форме. Использование информационных технологий позволяет повышать скорость обработки данных, улучшать качество результатов и снижать трудозатраты.

Одним из основных преимуществ информационных технологий является возможность аккумулировать большие объёмы информации и обеспечивать быстрый доступ к ней. Это облегчает поиск и систематизацию данных, повышает эффективность их обработки и анализа. Кроме того, применение информационных технологий способствует сокращению использования бумажных носителей, что снижает затраты на печать и пересылку документов и способствует охране окружающей среды.

Целью настоящей учебной практики является проектирование каталога канцелярских товаров, которое предусматривает проектирование решения для структуризации ассортимента, систематизации товарных позиций и организации взаимодействия с пользователями. В рамках проекта предполагается проектирование концепции удобного интерфейса для администраторов и пользователей, а также обеспечение логики эффективного хранения и обработки данных.

Объектом учебной практики является проектируемый каталог канцелярских товаров.

Предметом учебной практики является каталог канцелярских товаров как проектное решение, включающее структуру данных, функциональные возможности и взаимодействие с пользователями.

# АНАЛИЗ ИСХОДНЫХ ДАННЫХ И ПОСТАНОВКА ЗАДАЧИ

## Описание предметной области

В соответствии с заданием на учебную практику темой является проектирование информационной системы «Каталог канцелярских товаров».

Каталог канцелярских товаров представляет собой розничную торговую точку, осуществляющую продажу канцелярской продукции конечным потребителям. Для автоматизации работы магазина требуется веб-приложение, позволяющее вести учет товаров, обрабатывать продажи, управлять складскими запасами и формировать необходимую отчетность.

Информационная система – это программное обеспечение, устанавливаемое на персональный компьютер продавца в магазине и обеспечивающее автоматизацию процессов продажи товаров, учета и формирования отчетов.

Основные термины и понятия предметной области

Товар – единица продукции канцелярского назначения, характеризующаяся наименованием, описанием, ценой, количеством на складе и категорией.

Категория товаров – классификационная группа, объединяющая товары по функциональному назначению или типу использования.

Продажа – операция реализации товара покупателю с оформлением чека и уменьшением остатка товара на складе.

Заказ – документ, подтверждающий факт продажи товара с указанием наименования, количества, цены и общей суммы покупки.

Основные категории канцелярских товаров

Ассортимент магазина канцелярских товаров включает следующие основные категории:

Товары для письма: ручки шариковые, гелевые, перьевые; карандаши простые и цветные; маркеры; фломастеры; текстовыделители; стержни для ручек.

Бумажная продукция: бумага для печати различных форматов; тетради школьные и общие; блокноты; ежедневники; альбомы для рисования; конверты; стикеры.

Офисные принадлежности: степлеры и скобы; дыроколы; ножницы; канцелярские ножи; клей различных видов; скотч; скрепки; кнопки; зажимы для бумаг; папки и файлы; лотки для документов; настольные органайзеры.

Товары для творчества: краски (акварель, гуашь, акрил); кисти художественные; цветная бумага и картон; пластилин; глина для лепки.

Школьные товары: рюкзаки и ранцы; пеналы; обложки для тетрадей и учебников; линейки; угольники; циркули; транспортиры.

Предметная область охватывает процессы управления товарами и обработки заказов в розничной торговле. Система предусматривает две роли: администратор, управляющий каталогом товаров и обработкой заказов, и покупатель, осуществляющий просмотр товаров и оформление заказов.

Основными сущностями являются товар, категория, заказ и позиция заказа. Товар характеризуется артикулом (уникальный идентификатор), названием, ценой, количеством на складе, категорией и изображением. Категория объединяет товары по типам и имеет идентификатор и название. Заказ содержит дату оформления, статус (новый или выполненный) и набор товаров с их количеством.

В системе реализованы четыре основных процесса. Добавление товара позволяет администратору вносить новые позиции в каталог с указанием необходимых атрибутов. Управление товарами включает просмотр каталога, редактирование, удаление товаров и работу с файлами. Оформление заказа позволяет покупателю выбрать товары и подтвердить заказ, после чего система регистрирует его и автоматически уменьшает остатки на складе. Обработка заказа позволяет администратору управлять статусами и выполнением заказов. Поиск и сортировка доступны обеим ролям для удобства работы с каталогом.

Процесс оформления заказа инициируется покупателем через клиент-панель. Пользователь просматривает таблицу товаров, выбирает требуемые товары и добавляет их в корзину. Система поддерживает возможность изменения содержимого корзины (добавление, удаление товаров, очистка). После фиксации заказа система регистрирует его в базе данных и производит автоматическое уменьшение остатков товаров на складе.

Обе роли пользователя имеют доступ к функциям поиска и сортировки товаров. Пользователь может вводить поисковый запрос, выбирать критерий поиска, и система выдает отфильтрованные результаты. Отсортированные по различным параметрам товары представляются в порядке возрастания или убывания выбранного критерия.

Процессы системы управляются двумя ролями: администратор, ответственный за управление товарами и заказами, и покупатель, осуществляющий просмотр товаров и оформление заказов.

Администратор – работает с админ-панелью: добавляет товары, управляет категориями, просматривает и выполняет заказы, импортирует/экспортирует данные о товарах, контролирует остатки на складе, удаляет устаревшие товары.

Клиент – работает с клиент-панелью: просматривает каталог товаров, использует поиск и сортировку, добавляет товары в корзину, формирует и сохраняет заказ.

В настоящее время процесс продажи канцелярских товаров в небольших магазинах характеризуется следующими проблемами:

* ручной учет товаров в тетрадях или таблицах Excel;
* отсутствие автоматизации процесса продажи;
* сложность контроля остатков товаров на складе;
* отсутствие быстрого доступа к информации о товарах;
* затруднения при поиске нужного товара в большом ассортименте;
* риск ошибок при ручном расчете стоимости покупки.

Разработка информационной системы позволит решить ряд критических проблем, с которыми сталкиваются малые торговые предприятия. Система обеспечит быстрый и точный учет товаров, исключив ошибки, связанные с ручным ведением реестра, а также позволит оперативно отслеживать изменения в каталоге товаров. Автоматизация процесса продажи упростит оформление заказов, снизит время обработки каждой покупки и повысит пропускную способность торговой точки. Система будет формировать необходимую отчетность о продажах, остатках товаров и финансовых показателях, предоставляя владельцу бизнеса достоверную информацию для принятия управленческих решений.

## Сравнительный анализ аналогов и прототипов

Для формирования требований к системе был проведен анализ трех популярных решений для розничной торговли: «1С:Розница», «БелБланк» и «Торговый склад». Эти системы выбраны как референтные благодаря их широкому применению на рынке.

«1С:Розница» обеспечивает комплексный функционал управления продажами, складским учетом и интеграцией с оборудованием, однако имеет высокую стоимость и сложность в освоении.

«БелБланк» отличается простым интерфейсом и низкими требованиями к ресурсам, но ограничена базовыми операциями без расширенной аналитики.

«Торговый склад» ориентирована на микробизнес с минимальными требованиями к ресурсам, однако обладает примитивным функционалом.

Анализ показал, что существует ниша для разработки системы, объединяющей надежность и полноту функционала с простотой и доступностью.

«1С:Розница»  –   программа для управления операционной деятельностью магазинов и небольших сетей.

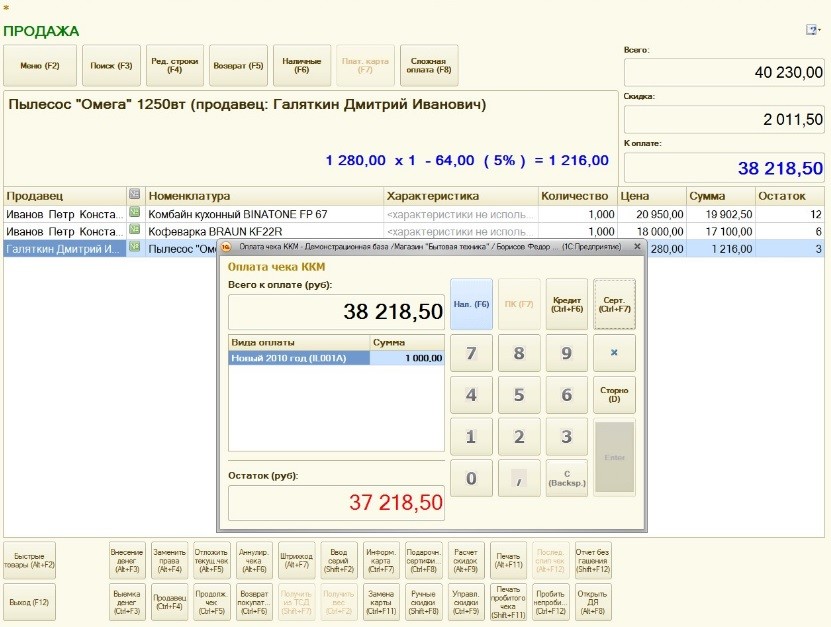


Рисунок 1.1 – Интерфейс системы «1С:Розница»

Функциональные возможности:

* учет товаров с возможностью добавления, редактирования и удаления;
* управление категориями товаров;
* обработка продаж с автоматическим формированием чеков;
* учет остатков товаров на складе;
* формирование отчетов о продажах и остатках;
* работа со штрих-кодами;
* интеграция с фискальными регистраторами.

Программно-аппаратная платформа:

* платформа: 1С:Предприятие 8.3;
* язык программирования: Встроенный язык 1С;
* база данных: Microsoft SQL Server или встроенная файловая база данных;
* операционная система: Windows 7/8/10/11;
* требования к ПК: Процессор 2 ГГц, 4 ГБ RAM, 10 ГБ свободного места.

Преимущества:

* широкая функциональность;
* возможность интеграции с другими продуктами 1С;
* надежность и стабильность работы;
* регулярные обновления и техническая поддержка;
* соответствие требованиям российского законодательства;
* развитая система отчетности.

Недостатки:

* высокая стоимость лицензии (от 12 000 рублей);
* сложный интерфейс, требующий обучения персонала;
* избыточная функциональность для небольших магазинов;
* требовательность к ресурсам компьютера.

«БелБланк» – белорусский онлайн-магазин канцелярских товаров с собственной информационной системой управления, ориентированный как на розничных, так и на корпоративных клиентов.

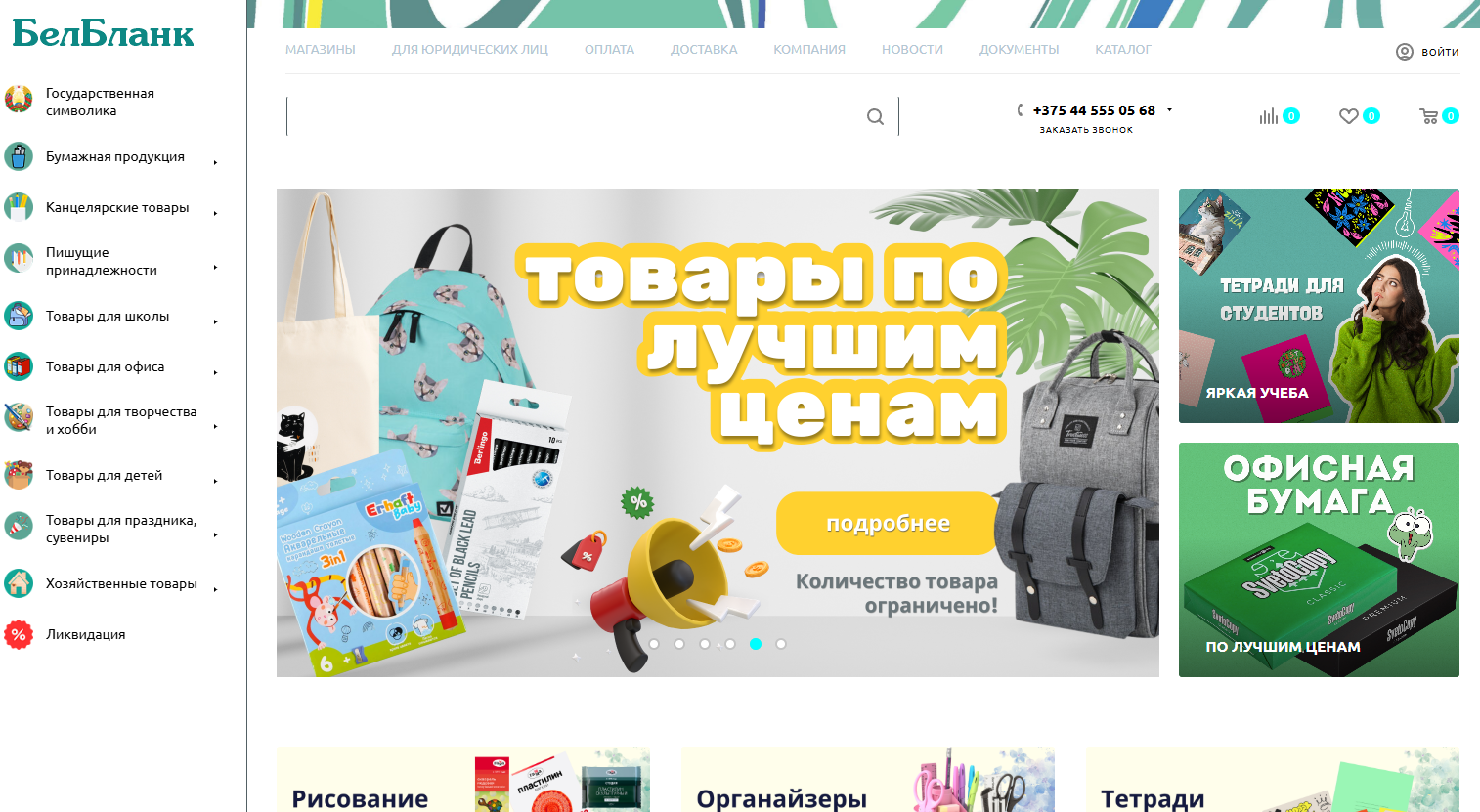


Рисунок 1.2 – Интерфейс сайта «БелБланк»

Функциональные возможности:

* структурированный каталог с разделением на бизнес-сегменты;
* система быстрого заказа для постоянных клиентов по артикулам;
* модуль сравнения характеристик товаров;
* функционал "отложенной покупки";
* интегрированная система бонусов и накопительных скидок;
* модуль учета товарных остатков с привязкой к торговым точкам.

Преимущества:

* адаптированность к особенностям местного рынка;
* оптимальный баланс между функциональностью и простотой использования;
* интеграция с локальными платежными системами;
* эффективная организация многоуровневого категорийного меню.

Недостатки:

* ограниченные возможности международной доставки;
* менее развитая мобильная версия системы;
* отсутствие продвинутых аналитических инструментов;
* недостаточная оптимизация для высоких нагрузок.

Программно-аппаратная платформа:

* платформа: Веб-приложение (React), десктоп (Electron);
* язык программирования: JavaScript (Node.js, React);
* база данных: SQLite (локальная), PostgreSQL (облачная);
* операционная система: Windows 7/8/10/11, macOS, Linux;
* требования к ПК: Процессор 1.5 ГГц, 2 ГБ RAM, 2 ГБ на диске.

«Торговый склад» – это бюджетное приложение для автоматизации небольших магазинов розничной торговли.

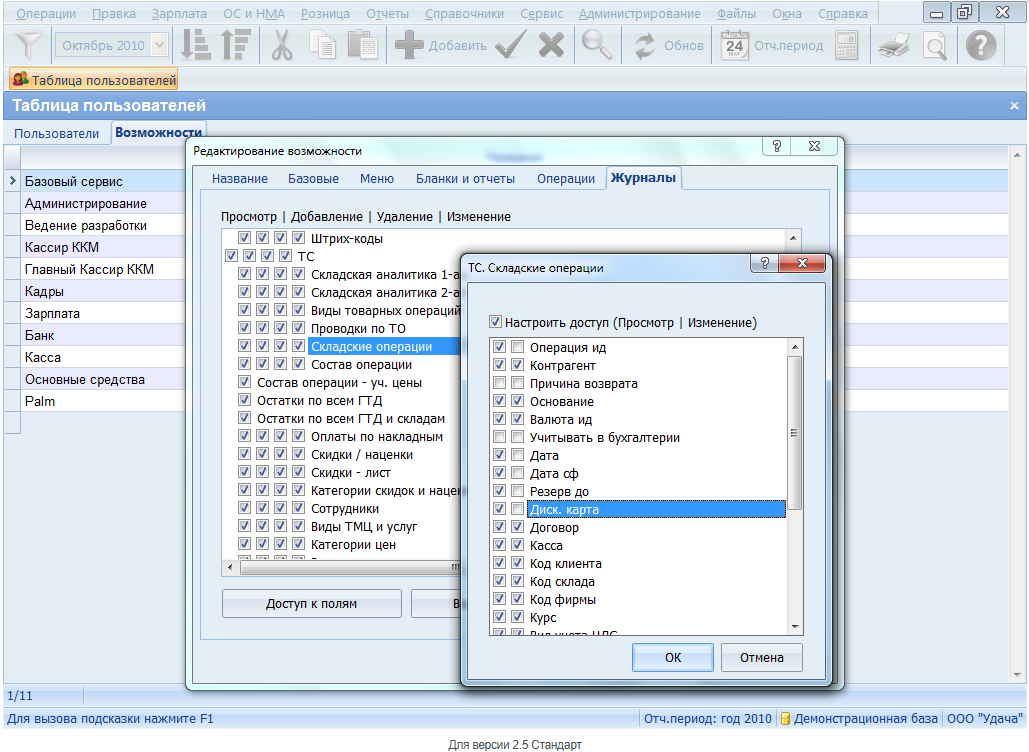


Рисунок 1.3 – Интерфейс системы «Торговый склад»

Функциональные возможности:

* справочник товаров с возможностью добавления и редактирования;
* управление категориями товаров;
* оформление продаж с формированием чеков;
* учет остатков товаров;
* простой поиск товаров;
* отчет о продажах за период;
* работа со штрих-кодами;
* печать ценников.

Программно-аппаратная платформа:

* платформа: Windows Forms (.NET Framework);
* язык программирования: C#;
* база данных: Microsoft Access или SQL Server Express;
* операционная система: Windows 7/8/10/11;
* требования к ПК: Процессор 1 ГГц, 1 ГБ RAM, 1 ГБ свободного места.

Преимущества:

* очень простой интерфейс, не требующий обучения;
* низкая стоимость лицензии (от 3 000 рублей);
* минимальные требования к компьютеру;
* быстрая установка и настройка;
* автономная работа без интернета.

Недостатки:

* устаревший дизайн интерфейса;
* ограниченная функциональность;
* нет интеграции с современными системами;
* отсутствие облачной синхронизации;
* работает только на Windows;
* ограниченные возможности отчетности.

По результатам проведенного анализа были выявлены следующие ключевые аспекты разработки системы управления товарами и заказами.

Рынок демонстрирует наличие двух крайностей: дорогие сложные решения с избыточной функциональностью, такие как 1С:Розница, и простые бюджетные приложения с ограниченными возможностями, такие как Торговый склад. Это указывает на наличие четкой рыночной ниши для разработки оптимального решения, которое балансирует между функциональностью и простотой использования.

На основе анализа определены приоритетные требования к разрабатываемой системе. Система должна обладать простым и интуитивно понятным интерфейсом, позволяющим пользователям быстро освоить функциональность без дополнительного обучения. Необходимо реализовать управление каталогом товаров с организацией по категориям, обеспечить быстрое оформление продаж с автоматическим расчетом сумм, реализовать автоматический контроль остатков товаров на складе, включить функционал поиска товаров по названию и сортировки результатов. Система должна формировать отчеты о продажах и остатках, иметь низкие требования к ресурсам компьютера и поддерживать автономную работу без подключения к интернету.

Для реализации этих требований рекомендуется использовать технологический стек, включающий язык программирования C# или Python, графический интерфейс на базе Windows Forms, WPF или PyQt/Tkinter, базу данных SQL Server или PostgreSQL и операционную систему Windows. Такой выбор обеспечивает баланс между производительностью, кроссплатформенностью и простотой разработки.

Разрабатываемая система должна объединить положительные качества проанализированных решений. От 1С:Розница система должна унаследовать надежность, полноту функционала учета и качественную отчетность. От БелБланк – простоту интерфейса, быстроту работы и легкость освоения. От Торгового склада – низкие требования к вычислительным ресурсам, доступность и простоту установки.

Тема разработки актуальна по совокупности следующих факторов. Существует постоянный спрос на доступные решения со стороны небольших магазинов, нуждающихся в автоматизации своих процессов. На рынке наблюдается отсутствие оптимального решения с хорошим балансом между ценой, функциональностью и простотой использования. Проект предполагает разработку с использованием современных технологий, что обеспечивает его технологическую актуальность. Система обладает практической ценностью и может быть реально использована небольшими магазинами и торговыми точками. Проект имеет высокую образовательную значимость, так как охватывает ключевые аспекты разработки приложений, включая проектирование пользовательского интерфейса, работу с базами данных и организацию бизнес-логики. Кроме того, система характеризуется доступностью внедрения, не требуя серверной инфраструктуры, что значительно снижает порог входа для малого бизнеса и делает решение доступным для широкого круга потенциальных пользователей.

Таким образом, разработка информационной системы «Каталог канцелярских товаров» является целесообразной задачей, позволяющей создать востребованный программный продукт с оптимальным соотношением функциональности, удобства использования и доступности для небольших торговых точек.

## Разработка технического задания

Техническое задание – это документ, который фиксирует все требования к разрабатываемому ПО и служит договором между заказчиком и разработчиком. ТЗ предотвращает недоразумения и становится основанием для оценки сроков и приемки продукта.

Спецификация функций описывает, какие операции должно выполнять ПО и какие бизнес-процессы оно автоматизирует. Это нужно, чтобы разработчики точно понимали, что реализовывать.

Архитектура и организация данных объясняют структурное устройство системы и схему базы данных. Правильная архитектура обеспечивает масштабируемость, безопасность и удобство поддержки.

Инструменты реализации – это технологии и языки программирования, которые будут использованы. Выбор обосновывается требованиями проекта и возможностями команды.

Эксплуатационные требования определяют условия работы системы: производительность, надежность, безопасность и требования к оборудованию. Это гарантирует стабильную работу ПО в реальной среде.

### Анализ входных и выходных данных

В ходе анализа предметной области были выделены объекты, значимые характеристики объектов, и определены входные и выходные данные для разрабатываемой системы.

Для автоматизации учета заказов и управления каталогом канцелярских товаров разработана информационная система, которая позволит в удобной форме следить за внесёнными данными в таблицы, учитывать правильность вводимых данных, организовывать поиск необходимой информации и т.д.

На основании анализа предметной области и выявленных функциональных требований была разработана диаграмма вариантов использования, которая описывает функциональное назначение системы в самом общем виде с точки зрения всех её пользователей и заинтересованных лиц.

Перечень входных и выходных данных приведен в таблице 1.1.

### **Таблица 1.1 – Перечень входных и выходных данных**

|  |  |  |
| --- | --- | --- |
| Объект | Входные данные  (Характеристика объекта) | Выходные данные |
| 1 | 2 | 3 |
| Администратор | Логин, Пароль | Сообщение об ошибке |
| Клиент | Логин, Пароль | Сообщение об ошибке |

Продолжение табл. 1.1

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| Товары | Данные о товарах (название, цена, изображение, кол-во на складе, артикул) | Список товаров |
| Заказ | Данные о заказе от клиента (выбранные товары клиентом) | Отчёт о выполненном заказе |
| Поиск | Критерии поиска | Результаты поиска |
| Сортировка | Критерии сортировки | Результаты сортировки |
| Корзина | Товары в корзине | Сумма заказа |
| Отчёты | Данные о выполненном заказе (название товара, дата выполнения, общая сумма выполненного заказа, артикул товара, оставшееся кол-во на складе) | Отчёт о выполненном заказе |

### Спецификация функций программного средства

Программное средство должно обеспечить разграничение прав доступа для двух категорий пользователей: **Администратор** и **Клиент**. Каждая категория обладает специфическим набором функциональных возможностей, соответствующих её роли в системе управления каталогом канцелярских товаров.

**Администратор** – пользователь с расширенными правами доступа, отвечающий за управление каталогом канцелярских товаров, добавление и редактирование товарных позиций, контроль наличия, управление категориями товаров и обеспечение корректной работы программного средства. Администратор имеет полный доступ ко всем модулям системы.

**Клиент** – конечный пользователь системы с ограниченными правами доступа, который использует каталог для просмотра ассортимента канцелярских товаров, поиска необходимых позиций, формирования заказов.

В таблице 1.2 приведен перечень функциональных возможностей для каждой группы пользователей.

Таблица 1.2 – Спецификация функций программного средства

|  |  |  |
| --- | --- | --- |
| № функции | Название функции | Роль |
| 1 | 2 | 3 |
| 1 | Регистрация нового администратора | Администратор |
| 2 | Регистрация нового клиента | Клиент |

Продолжение табл. 1.2

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| 3 | Аутентификация | Администратор, Клиент |
| 4 | Формирование списка товаров (просмотр, добавление, редактирование, удаление) | Администратор |
| 5 | Просмотр списка товаров | Клиент |
| 6 | Добавление товара в корзину | Клиент |
| 7 | Формирование заказов от клиента (выполнение заказов) | Администратор |
| 8 | Формирование отчётов (данные о заказе) | Администратор |
| 9 | Поиск по заданным критериям | Администратор, Клиент |
| 10 | Сортировка по заданным критериям | Администратор, Клиент |

### Обоснование архитектуры программного средства и способа организации данных

Для создания информационной системы управления каталогом канцелярских товаров можно использовать различные архитектуры. Системы управления базами данных (СУБД) – это программные средства, предназначенные для создания, наполнения, обновления и удаления баз данных.

По архитектуре СУБД делятся на одно-, двух- и трехзвенные. В двухзвенной архитектуре значительную часть логики управления данными берет на себя сервер БД, в то время как клиент занят отображением данных в удобном для пользователя виде.​

Для разрабатываемой информационной системы выбрана клиент-серверная архитектура СУБД на базе MySQL. При выборе подхода учитывались следующие критерии: высокое быстродействие, одновременная работа всех пользователей и защита данных от несанкционированного доступа.​

Рекомендуется использовать клиент-серверную архитектуру с MySQL в качестве сервера базы данных, при которой данные о товарах, заказах и клиентах хранятся и обрабатываются централизованно на сервере. Это обеспечивает централизованную базу данных с единым управлением. Клиент-серверный подход на основе MySQL позволяет пользователям получать доступ к системе с любого рабочего места, обеспечивает стабильную работу за счёт резервного копирования, повышает безопасность за счёт разграничения прав доступа пользователей, а также предоставляет возможность масштабирования системы при расширении функциональности.

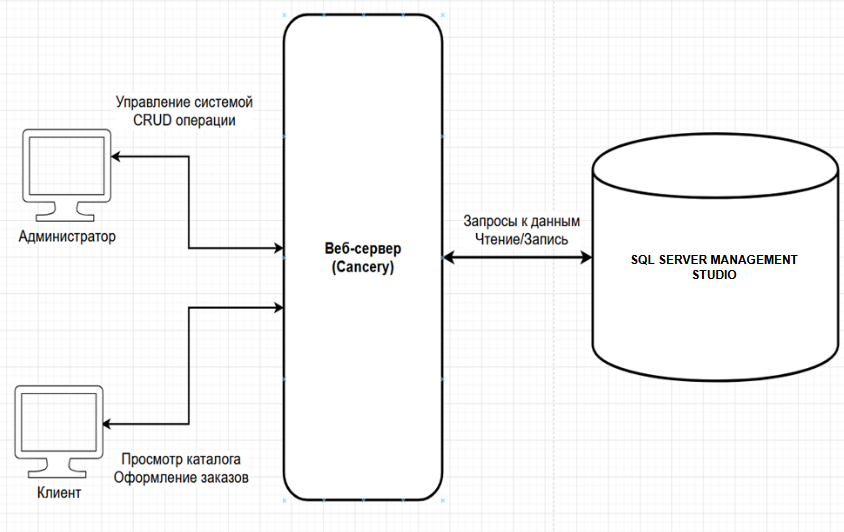


Рисунок 1.4 – Клиент-серверная архитектура

Такая структура позволяет эффективно организовать работу с каталогом, обеспечивая необходимый уровень доступа для различных категорий пользователей и централизованное хранение всех данных.

### Характеристика инструментов реализации

В соответствии с заданием на учебную практику для разработки каталога канцелярских товаров был использован язык программирования C# в сочетании со средой разработки CLI/CLR. Такой выбор обусловлен высокой производительностью языка, его универсальностью и поддержкой различных платформ.

Для реализации проекта применяются следующие инструменты и технологии:

* NET Framework: платформа, необходимая для функционирования приложений, созданных с использованием CLI/CLR;
* WPF: библиотека классов NET Framework, предназначенная для разработки графического интерфейса пользователя (GUI) в Windows-приложениях;​
* MySQL Connector/NET (MySql.Data):    полностью управляемый ADO.NET драйвер для подключения к базе данных MySQL, обеспечивающий взаимодействие между приложением на C# и сервером баз данных. Connector/NET позволяет выполнять SQL-запросы (SELECT, INSERT, UPDATE, DELETE) и управлять данными о товарах, заказах и клиентах непосредственно из программного кода с использованием классов MySqlConnection, MySqlCommand и MySqlDataReader;​
* ADO.NET: технология доступа к данным в .NET Framework, используемая для установления соединения с базой данных MySQL, выполнения команд и обработки результатов запросов.​

Использование C# и CLI/CLR в связке с MySQL Connector/NET позволяет разработать эффективное и надежное приложение с интуитивно понятным интерфейсом и централизованным хранением данных в СУБД MySQL, отвечающим всем требованиям заказчика.

### Обоснование эксплуатационных требований

Эксплуатационные требования к техническим и программным средствам каталога канцелярских товаров формируются следующим образом: на рабочих станциях администратора и клиента должно быть установлено программное обеспечение, включающее операционную систему Windows 10 с последними обновлениями, что обеспечивает выполнение задач, необходимых для реализации проекта.

На основании этого минимальные требования к конфигурации аппаратных средств выглядят следующим образом.

Рабочая станция администратора:

− процессор: 1 гигагерц (ГГц) или более быстрый процессор или SoC;

− оперативная память: 1 гигабайт (ГБ) для 32 – разрядных или 2 ГБ для

64 – разрядных;

− место на жестком диске: 4 ГБ для 64 – битной и 32 – битной ОС;

− видеокарта: с поддержкой DirectX 9;

− разрешение экрана: 800 x 600, минимальный размер диагонали для основного дисплея 13 дюймов и больше.

Рабочая станция клиента:

− процессор: 1 гигагерц (ГГц) или более быстрый процессор или SoC;

− оперативная память: 1 гигабайт (ГБ) для 32 – разрядных или 2 ГБ для

64 – разрядных;

− место на жестком диске: 4 ГБ для 64 – битной и 32 – битной ОС;

− видеокарта: с поддержкой DirectX 9;

− разрешение экрана: 800 x 600, минимальный размер диагонали для основного дисплея 13 дюймов и больше.

Данные технические требования обеспечивают стабильную работу программного обеспечения и эффективное взаимодействие всех компонентов каталога для обработки информации о товарах, платежах и формирования отчетов.

## Выводы по первому разделу

В ходе выполнения этапа анализа исходных данных был проведен анализ предметной области, включающий исследование входных и выходных данных каталога канцелярских товаров, существующих прототипов и эксплуатационных требований.

Анализ показал, что существующие каталоги учета канцелярских товаров имеют определенные ограничения – от высокой стоимости внедрения до недостаточной гибкости. Это подтвердило необходимость разработки собственного каталога, объединяющего преимущества существующих решений и устраняющего их недостатки.​

Каталог канцелярских товаров предусматривает разделение прав доступа для пользователей (Администратор и Клиент), а также выполнение задач, связанных с учетом канцелярских товаров, их сортировкой, поиском и оформлением заказов.​

Для разрабатываемого каталога были определены эксплуатационные требования, предусматривающие стабильную работу, снижение вероятности сбоев, создание удобного и интуитивно понятного интерфейса, а также возможность регулярных обновлений для соответствия изменяющимся условиям и требованиям рынка канцелярских товаров.

Была определена архитектура каталога на основе клиент-серверной модели с использованием СУБД SQL Server Management Studio. Структура хранения данных реализована через реляционную базу данных с централизованным доступом, что обеспечивает удобство работы, целостность информации и многопользовательский режим. Основными компонентами системы являются: сервер приложений с модулями авторизации, каталога и заказов, а также база данных SQL Server, включающая следующие таблицы:​

* пользователь – таблица пользователей (идентификатор, логин, пароль, роль, дата регистрации);
* Категория – таблица категорий (идентификатор, наименование, описание);
* Товар – таблица товаров (идентификатор, категория, пользователь, наименование, артикул, цена, количество на складе, описание, дата добавления);
* Фотография – таблица фотографий (идентификатор, пользователь, товар, путь файла, дата загрузки);
* Корзина – таблица корзины (идентификатор, пользователь, товар, количество);
* Заказ – таблица заказов (идентификатор, пользователь, дата заказа, статус, сумма, адрес доставки);
* Заказ\_товар – таблица товаров в заказе (идентификатор заказа, товар, количество, цена на момент заказа).

Для реализации проекта выбран язык программирования C# в сочетании с WPF, SQL Server для подключения к базе данных и средой разработки Visual Studio, что обеспечивает высокую производительность, надежность каталога канцелярских товаров и эффективное взаимодействие с СУБД SQL Server.

Разработано техническое задание, которое представлено в ПРИЛОЖЕНИИ А.

# СОЗДАНИЕ ИНФОРМАЦИОННОЙ БАЗЫ ПРОЕКТА

## Разработка концептуальной модели базы данных

На данном этапе проектирования была разработана концептуальная модель базы данных информационной системы «Каталог канцелярских товаров».

Модель определяет основные объекты предметной области и взаимосвязи между ними, обеспечивая логическую основу для дальнейшего проектирования структуры данных.

Концептуальная модель отражает ключевые сущности, такие как пользователи системы (администраторы и покупатели), категории товаров, товары, поставщики, заказы, элементы заказов и отзывы покупателей.  
Каждая сущность содержит набор атрибутов, характеризующих хранимую информацию, а связи между сущностями определяют логику взаимодействия участников торгового процесса и элементов каталога.

Разработанная концептуальная модель демонстрирует структуру базы данных и принципы организации информации в системе, обеспечивая целостность, логичность и удобство обработки данных.

Концептуальная модель предметной области представлена на рисунке 2.1.

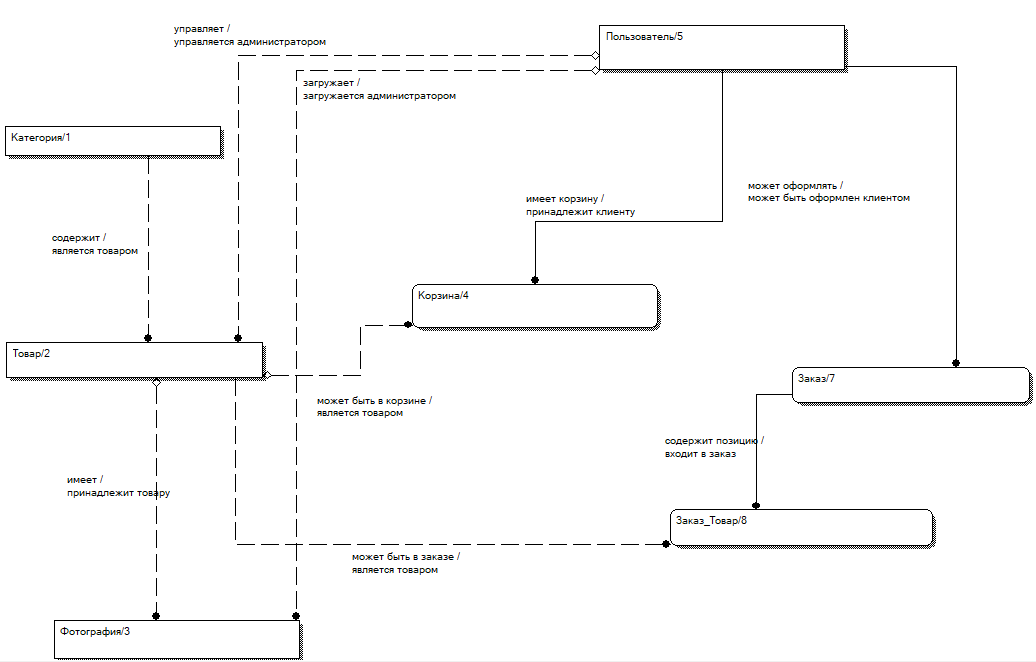


Рисунок 2.1 – Концептуальная модель предметной области

## Разработка модели «сущность-связь»

Для информационной системы каталога канцелярских товаров была разработана диаграмма «сущность-связь» представленная в ПРИЛОЖЕНИИ Б. В данной диаграмме были выделены следующие сущности:

* «Пользователь»;
* «Категория»;
* «Товар»;
* «Фотография»;
* «Корзина»;
* «Заказ»;
* «Заказ\_Товар».

Сущность «Пользователь» предназначена для хранения данных о зарегистрированных в системе лицах, включая администраторов и клиентов. Подробное описание данной сущности приведено в таблице 2.1.

Таблица 2.1 - Структура сущности «Пользователь»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип атрибута | Пояснение |
| ID\_Пользователя | Обязательный | Первичный ключ, характеризующий определённого пользователя |
| Логин | Обязательный | Уникальный логин для аутентификации |
| Пароль | Обязательный | Пароль пользователя |
| Email | Обязательный | Электронная почта пользователя |
| Роль | Обязательный | Роль пользователя (дискриминатор: Администратор/Клиент) |
| Дата\_регистрации | Обязательный | Дата регистрации пользователя |

Сущность «Категория» предназначена для хранения данных о группах канцелярских товаров, представленных в системе. Категории позволяют структурировать ассортимент, облегчая поиск и навигацию для пользователей. Подробное описание данной сущности представлено в таблице 2.2.

Таблица 2.2 - Структура сущности «Категория»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип атрибута | Пояснение |
| ID\_Категории | Обязательный | Первичный ключ, характеризующий категорию |
| Название | Обязательный | Название категории товаров |
| Описание | Необязательный | Описание категории |

Сущность «Товар» предназначена для хранения данных о канцелярских товарах, доступных в каталоге системы. Она содержит сведения о наименовании, количеству на складе товара, цене и принадлежности товара к определённой категории. Подробное описание данной сущности приведено в таблице 2.3.

Таблица 2.3 - Структура сущности «Товар»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип атрибута | Пояснение |
| ID\_Товара | Обязательный | Первичный ключ, характеризующий товар |
| Название | Обязательный | Название товара |
| Артикул | Обязательный | Уникальный артикул товара |
| Цена | Обязательный | Цена товара |
| Количество\_на\_складе | Обязательный | Количество товара на складе |
| Описание | Необязательный | Описание товара |
| ID\_Категории | Обязательный | Внешний ключ. Ссылка на категорию |
| ID\_Пользователя | Обязательный | Внешний ключ. Ссылка на пользователя |
| Дата\_добавления | Обязательный | Дата добавления товара в каталог |

Сущность «Фотография» предназначена для хранения данных об изображениях, связанных с канцелярскими товарами. Она обеспечивает визуальное представление продукции в каталоге, что помогает пользователям при выборе и оформлении заказов. Подробное описание данной сущности представлено в таблице 2.4.

Таблица 2.4 - Структура сущности «Фотография»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип атрибута | Пояснение |
| ID\_Фотографии | Обязательный | Первичный ключ, характеризующий фотографию |
| ID\_Товара | Обязательный | Внешний ключ. Ссылка на товар |
| ID\_Пользователя | Обязательный | Внешний ключ. Ссылка на пользователя |
| Путь\_файла | Обязательный | Путь к файлу фотографии |
| Дата\_загрузки | Обязательный | Дата загрузки фотографии |

Сущность «Корзина» предназначена для хранения данных о товарах, добавленных клиентами перед оформлением заказа. Она обеспечивает возможность временного сохранения выбранных позиций, изменения их количества и последующего оформления покупки. Подробное описание данной сущности представлено в таблице 2.5.

Таблица 2.5 - Структура сущности «Корзина»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип атрибута | Пояснение |
| 1 | 2 | 3 |
| ID\_Корзины | Обязательный | Первичный ключ, характеризующий корзину |
| ID\_Пользователя | Обязательный | Внешний ключ. Ссылка на пользователя |

Продолжение табл. 2.5

|  |  |  |
| --- | --- | --- |
| 1 | 2 | 3 |
| ID\_Товара | Обязательный | Внешний ключ. Ссылка на товар |
| Количество | Обязательный | Количество товаров в корзине |

Сущность «Заказ» предназначена для хранения данных о сформированных клиентами заказах в информационной системе каталога канцелярских товаров. Она содержит сведения о дате оформления, составе заказа, его стоимости и текущем статусе выполнения. Подробное описание данной сущности представлено в таблице 2.6.

Таблица 2.6 - Структура сущности «Заказ»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип атрибута | Пояснение |
| ID\_Заказа | Обязательный | Первичный ключ, характеризующий заказ |
| ID\_Пользователя | Обязательный | Внешний ключ. Ссылка на пользователя |
| Дата\_заказа | Обязательный | Дата оформления заказа |
| Статус | Обязательный | Статус заказа (новый, обработан, отправлен, доставлен) |
| Сумма | Обязательный | Сумма заказа |
| Адрес\_доставки | Обязательный | Адрес доставки заказа |

Сущность «Заказ\_Товар» предназначена для хранения данных о составе заказов клиентов, включая перечень товаров, их количество и стоимость на момент оформления. Она обеспечивает связь между сущностями «Заказ» и «Товар», позволяя формировать детализированную структуру каждого заказа. Подробное описание данной сущности представлено в таблице 2.7.

Таблица 2.7 - Структура сущности «Заказ\_Товар»

|  |  |  |
| --- | --- | --- |
| Название атрибута | Тип атрибута | Пояснение |
| ID\_Заказа | Обязательный | Внешний ключ. Ссылка на заказ |
| ID\_Товара | Обязательный | Внешний ключ. Ссылка на товар |
| Количество | Обязательный | Количество товара в заказе |
| Цена\_на\_момент\_заказа | Обязательный | Цена товара на момент оформления заказа |

Сущности, представленные в таблицах 2.1–2.7, имеют различные идентифицирующие связи, которые приведены в таблице 2.8. Эти связи позволяют точно определить взаимосвязи между объектами системы и обеспечивают целостность данных при их обработке.

Таблица 2.8 - Связи между сущностями

|  |  |  |  |
| --- | --- | --- | --- |
| Сущность 1 | Сущность 2 | Вид связи | Пояснение |
| «Пользователь» | «Товар» | 1 : М (FK – NOT NULL) | Один администратор управляет несколькими товарами |
| «Пользователь» | «Корзина» | 1 : М (FK – NOT NULL) | Клиент может иметь корзину или нет. Разрешено, что корзина может быть пустой |
| «Пользователь» | «Фотография» | 1 : М (FK – NULL) | Администратор может изменять, добавлять фотографию к товару |
| «Пользователь» | «Заказ» | 1 : М (FK – NULL) | Клиент может оформлять несколько заказов, но может и не оформлять ни одного |
| «Категория» | «Товар» | 1 : М (FK – NOT NULL) | Одна категория содержит много товаров, но каждый товар принадлежит только одной категории |
| «Товар» | «Фотография» | 1 : М (FK – NOT NULL) | Один товар может иметь несколько фотографий, но каждая фотография принадлежит только одному товару |
| «Товар» | «Корзина» | 1 : М (FK – NOT NULL) | Одна корзина содержит несколько позиций товаров |
| «Заказ» | «Заказ\_Товар» | 1 : М (FK – NOT NULL) | Один заказ содержит несколько позиций товаров |
| «Товар» | «Заказ\_Товар» | 1 : М (FK – NULL) | Один товар может находиться в нескольких заказах. Разрешено, что товар может не быть в заказе |

## Разработка физической модели базы данных

Физическая модель базы данных представляет собой реализацию логической модели в конкретной системе управления базами данных (СУБД). Она определяет структуру таблиц, типы данных, ограничения целостности, индексы и связи между таблицами. В качестве СУБД для информационной системы «Каталог канцелярских товаров» была выбрана SQL Server.

Физическая модель базы данных включает следующие таблицы, соответствующие сущностям, определённым на предыдущих этапах проектирования:

– Пользователь – хранит данные о пользователях системы.

– Категория – описывает группы канцелярских товаров.

– Товар – хранит сведения о канцелярских товарах.

– Фотография – хранит данные о фотографиях товаров.

– Корзина – фиксирует товары, добавленные клиентами перед заказом.

– Заказ – хранит информацию о сформированных заказах.

– Заказ\_Товар – связывает заказы и товары (ассоциативная сущность).

Состав и структура таблиц базы данных подробно представлены в таблицах 2.9 – 2.15.

Таблица «Пользователь» предназначена для хранения информации о всех пользователях системы, включая их учетные данные, контактную информацию и роль в системе. Эта таблица является основной для управления доступом и аутентификацией пользователей. Подробное описание физической структуры представлено в таблице 2.9.

Таблица 2.9 - Физическая структура таблицы «Пользователь»

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Имя столбца | Тип данных | IS null | Identity | Primary key | Unique | Check | Default | Foreign key |
| ID\_Пользователя | int | Нет | Да | Да |  |  |  |  |
| Логин | varchar(50) | Нет |  |  | Да |  |  |  |
| Пароль | varchar(255) | Нет |  |  |  |  |  |  |
| Email | varchar(255) | Нет |  |  | Да |  |  |  |
| Роль | varchar(20) | Нет |  |  |  |  |  |  |
| Дата\_регистрации | datetime | Нет |  |  |  |  | GETDATE() |  |

Таблица «Категория» содержит справочные данные о категориях товаров, которые используются для классификации и организации ассортимента в каталоге. Подробное описание физической структуры представлено в таблице 2.10.

Таблица 2.10 - Физическая структура таблицы «Категория»

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Имя столбца | Тип данных | IS null | Identity | Primary key | Unique | Check | Default | Foreign key |
| ID\_Категории | int | Нет | Да | Да |  |  |  |  |
| Название | varchar(100) | Нет |  |  |  |  |  |  |
| Описание | varchar(500) | Да |  |  |  |  |  |  |

Таблица «Товар» является центральной для хранения информации о всех канцелярских товарах, их характеристиках, ценах и наличии на складе. Она связывает товары с категориями и администраторами, управляющими ассортиментом. Подробное описание физической структуры представлено в таблице 2.11.

Таблица 2.11 - Физическая структура таблицы «Товар»

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Имя столбца | Тип данных | IS null | Identity | Primary key | Unique | Check | Default | Foreign key |
| ID\_Товара | int | Нет | Да | Да |  |  |  |  |
| ID\_Пользователя | int | Нет | Да | Да |  |  |  |  |
| Название | varchar(150) | Нет |  |  |  |  |  |  |
| Артикул | varchar(50) | Нет |  |  | Да |  |  |  |
| Цена | decimal(10,2) | Нет |  |  |  | > 0 |  |  |
| Количество\_на\_складе | int | Нет |  |  |  |  | 0 |  |
| Описание | varchar(1000) | Да |  |  |  |  |  |  |
| ID\_Категории | int | Нет |  |  |  |  |  | Категория |
| Дата\_добавления | datetime | Нет |  |  |  |  | GETDATE() |  |

Таблица «Фотография» предназначена для хранения изображений товаров, обеспечивая визуальное представление продукции в каталоге и улучшая пользовательский опыт при просмотре товаров. Подробное описание физической структуры представлено в таблице 2.12.

Таблица 2.12 - Физическая структура таблицы «Фотография»

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Имя столбца | Тип данных | IS null | Identity | Primary key | Unique | Check | Default | Foreign key |
| ID\_Фотографии | int | Нет | Да | Да |  |  |  |  |
| ID\_Товара | int | Нет |  |  |  |  |  | Товар |
| ID\_Пользователя | int | Нет | Да | Да |  |  |  |  |
| Путь\_файла | varchar(500) | Нет |  |  |  |  |  |  |
| Дата\_загрузки | datetime | Нет |  |  |  |  | GETDATE() |  |

Таблица «Корзина» используется для временного хранения товаров, которые клиенты добавили перед оформлением заказа, позволяя им управлять выбранными товарами. Подробное описание физической структуры представлено в таблице 2.13.

Таблица 2.13 - Физическая структура таблицы «Корзина»

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Имя столбца | Тип данных | IS null | Identity | Primary key | Unique | Check | Default | Foreign key |
| ID\_Корзины | int | Нет | Да | Да |  |  |  |  |
| ID\_Пользователя | int | Нет |  |  |  |  |  | Пользователь |
| ID\_Товара | int | Нет |  |  |  |  |  | Товар |
| Дата\_создания | datetime | Нет |  |  |  |  | GETDATE() |  |
| Количество | int | Нет |  |  |  | > 0 |  |  |

Таблица «Заказ» содержит информацию о сформированных клиентами заказах, включая дату оформления, статус выполнения, стоимость и адрес доставки. Подробное описание физической структуры представлено в таблице 2.14.

Таблица 2.14 - Физическая структура таблицы «Заказ»

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Имя столбца | Тип данных | IS null | Identity | Primary key | Unique | Check | Default | Foreign key |
| ID\_Заказа | int | Нет | Да | Да |  |  |  |  |
| ID\_Пользователя | int | Нет |  |  |  |  |  | Пользователь |
| Дата\_заказа | datetime | Нет |  |  |  |  | GETDATE() |  |
| Статус | varchar(20) | Нет |  |  |  |  | Новый |  |
| Сумма | money() | Нет |  |  |  | > 0 |  |  |
| Адрес\_доставки | varchar(500) | Нет |  |  |  |  |  |  |

Таблица «Заказ\_Товар» является ассоциативной сущностью, которая устанавливает связь между заказами и товарами, фиксируя состав каждого заказа с указанием количества и цены товаров на момент оформления. Подробное описание физической структуры представлено в таблице 2.15.

Таблица 2.15 - Физическая структура таблицы «Заказ\_Товар»

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Имя столбца | Тип данных | IS null | Identity | Primary key | Unique | Check | Default | Foreign key |
| ID\_Заказа | int | Нет |  |  |  |  |  | Заказ |
| ID\_Товара | int | Нет |  |  |  |  |  | Товар |
| Количество | int | Нет |  |  |  | > 0 |  |  |
| Цена\_на\_момент\_заказа | money() | Нет |  |  |  | > 0 |  |  |

## Выводы по второму разделу

На основе разработанного структурного подхода к проектированию информационной системы «Каталог канцелярских товаров» были выполнены следующие ключевые этапы:

Была создана концептуальная модель, которая определила основные сущности системы и их взаимосвязи. Модель включает семь основных сущностей: Пользователь, Категория, Товар, Фотография, Корзина, Заказ и Заказ\_Товар, которые полностью отражают логику торгового процесса и структуру хранения информации о канцелярских товарах.

Разработана диаграмма «сущность-связь», которая определила тип и характер взаимосвязей между всеми компонентами системы. Установленные связи обеспечивают целостность данных и позволяют корректно организовать взаимодействие между администраторами системы и клиентами при управлении товарами, формировании заказов и обработке покупок.

Спроектирована физическая модель базы данных на основе СУБД SQL Server. Физическая модель включает семь таблиц с детально определенными типами данных, ограничениями целостности, индексами и внешними ключами. Каждая таблица содержит необходимые поля с установленными правилами валидации, что обеспечивает надежность и безопасность хранения информации.

Разработанная архитектура базы данных обеспечивает эффективное управление ассортиментом товаров, учет наличия продукции на складе, формирование и отслеживание заказов, а также управление пользователями с различными ролями. Таким образом, реализованный структурный подход создал прочную основу для функционирования информационной системы каталога канцелярских товаров, гарантируя логичность организации данных, их целостность и удобство обработки.

# СТРУКТУРНЫЙ ПОДХОД К ПРОЕКТИРОВАНИЮ ПО

## Функциональное моделирование

IDEF0 (Integrated Definition for Function Modeling) — это методология функционального моделирования, предназначенная для описания и анализа бизнес-процессов и функций информационных систем. IDEF0-модели представляют собой графическое отображение системы в виде иерархической структуры функциональных блоков, связанных входами, выходами, управлением и механизмами.

Основной целью функционального моделирования является четкое описание того, что делает система, как она преобразует входные данные в выходные результаты, и какие ресурсы и управляющие воздействия при этом используются. IDEF0-диаграммы позволяют выявить и структурировать все функции системы, определить взаимосвязи между ними и обеспечить понимание системы всеми заинтересованными сторонами.

Каждый функциональный блок в IDEF0 имеет четыре типа связей:

* входы (слева) – данные или объекты, которые преобразуются функцией;
* выходы (справа) – результаты преобразования входных данных;
* управление (сверху) – управляющие воздействия, правила и ограничения, которые определяют, как выполняется функция;
* механизмы (снизу) – ресурсы, инструменты и средства, необходимые для выполнения функции.

Для описания информационной системы каталога канцелярских товаров разработаны три IDEF0-модели: контекстная диаграмма (А0), диаграмма первого уровня декомпозиции (А0) и диаграмма второго уровня декомпозиции (А3).

Контекстная диаграмма представляет систему как единый функциональный блок и определяет границы системы, её взаимодействие с внешней средой.

Входные данные:

* логин и пароль пользователя (для аутентификации);
* данные о товарах;
* данные о заказах от клиентов;
* критерии сортировки товаров;
* критерии поиска товаров.

Выходные данные:

* сообщение об ошибке (при неудачной аутентификации);
* список товаров;
* отчёт о выполненном заказе;
* результаты сортировки товаров.
* результаты поиска товаров.

Управление:

* правила работы каталога канцелярских товаров;

Механизмы:

* клиент (пользователь системы);
* администратор (управляет системой и данными).

Первый уровень декомпозиции разбивает контекстную диаграмму на пять основных подфункций, раскрывая внутреннее устройство системы.

Входные данные:

* логин и пароль;
* данные о товарах;
* данные о заказе от клиента;
* критерии сортировки;
* критерии поиска.

Выходные данные:

* сообщение об ошибке;
* список товаров;
* отчёт о выполненном заказе;
* результаты сортировки товаров;
* результаты поиска товаров.

Управление:

* правила работы каталога канцелярских товаров.

Механизмы:

* клиент;
* администратор.

Функция аутентификации (блок 1) осуществляет проверку учетных данных пользователя и определяет возможность доступа к системе. Функция формирования списка товаров (блок 2) создает полный каталог доступных товаров на основе данных, хранящихся в системе. Функция управления товарами и обработка заказов (блок 3) занимается приемом заказов от клиентов, их обработкой и формированием отчетов. Функции формирования списков после сортировки (блок 4) и поиска (блок 5) позволяют пользователям находить интересующие их товары по различным критериям.

Второй уровень декомпозиции детализирует функцию управления товарами и обработки заказов (блок 3), раскрывая её внутреннюю структуру из четырех подфункций.

Входные данные:

* данные о заказе от клиента;
* список товаров.

Выходные данные:

* отчёт о выполненном заказе.

Управление:

* правила работы каталога канцелярских товаров.

Механизмы:

* администратор.

Функция приема и проверки заказа (блок 1) осуществляет валидацию входящих заказов от клиентов на соответствие требованиям системы и определяет их корректность. Функция формирования заказа (блок 2) создает структурированное представление заказа на основе проверенных данных и готовит его к обработке. Функция обновления данных системы (блок 3) актуализирует информацию о доступности товаров и статусе заказов в хранилище данных. Функция формирования отчётности (блок 4) создает итоговый отчет о выполненном заказе, который является конечным выходом процесса управления товарами и обработки заказов.

Разработанные IDEF0-модели обеспечивают полное описание функциональной архитектуры информационной системы каталога канцелярских товаров, позволяют понять логику её работы и служат основой для дальнейшего проектирования и разработки системы.

Представлены и подробно описаны IDEF0-модели для проектируемого ПО в ПРИЛОЖЕНИИ В.

## Моделирование бизнес-процессов

DFD (Data Flow Diagram) – это методология моделирования потоков данных, предназначенная для описания и анализа бизнес-процессов и информационных систем. DFD-модели представляют собой графическое отображение системы в виде процессов, потоков данных, внешних сущностей и хранилищ данных.

Основной целью моделирования потоков данных является четкое описание того, как данные перемещаются через систему, какие процессы их обрабатывают, где они хранятся и кто их использует. DFD-диаграммы позволяют выявить и структурировать все потоки информации в системе, определить взаимосвязи между процессами и обеспечить понимание системы всеми заинтересованными сторонами.

Основные элементы DFD:

* процесс (круг или овал) – функция или действие, которое преобразует входные данные в выходные;
* внешняя сущность (прямоугольник) – источник или получатель данных, находящийся за пределами системы;
* хранилище данных (две параллельные линии) – место, где данные хранятся для последующего использования;
* поток данных (стрелка) – передача данных между элементами системы.

Для описания информационной системы каталога канцелярских товаров разработаны три DFD-модели: контекстная диаграмма, диаграмма первого уровня и диаграмма второго уровня детализации процесса управления товарами.

Контекстная диаграмма представляет систему как единый процесс «Каталог канцелярских товаров» и определяет её взаимодействие с внешними сущностями:

* клиент – пользователь системы, который просматривает каталог и делает заказы;
* администратор – управляет системой, товарами и обрабатывает заказы.

Потоки данных от клиента к системе:

* логин и пароль;
* критерии поиска товаров;
* критерии сортировки товаров;
* данные о заказе.

От системы к клиенту:

* сообщение об ошибке;
* список товаров;
* результаты поиска товаров;
* результаты сортировки товаров.

От администратора к системе:

* логин и пароль;
* отчет о выполненном заказе;
* данные о товарах.

От системы к администратору:

* сообщение об ошибке.

Диаграмма первого уровня разбивает систему на пять основных процессов:

* «Аутентификация пользователя» принимает логин и пароль от клиента или администратора, взаимодействует с хранилищем «Данные для аутентификации» для чтения учётных данных и выдаёт сообщение об ошибке или данные для аутентификации;
* «Формирование списка товаров» принимает запрос на список товаров, читает данные из хранилища «Список товаров» и выдаёт список товаров клиенту;
* «Управление товарами и обработка заказов» принимает данные о заказе от клиента и список товаров, взаимодействует с хранилищем «Заказы» для чтения и записи данных о заказах и товарах, выдаёт отчёт о выполненном заказе администратору;
* «Формирование списка после сортировки» принимает критерии сортировки от клиента и список товаров, читает данные из хранилища «Список товаров» и выдаёт результаты сортировки клиенту;
* «Формирование списка после поиска» принимает критерии поиска от клиента, читает данные из хранилища «Список товаров» и выдаёт результаты поиска клиенту.

Диаграмма второго уровня детализирует процесс «Управление товарами и обработка заказов» на четыре подпроцесса:

* «Приём и проверка заказа» принимает данные о заказе от клиента, взаимодействует с хранилищем «Заказы» для чтения данных о товарах и выдаёт принятый к обработке заказ;
* «Формирование заказа» принимает принятый к обработке заказ и список товаров, создаёт структурированное представление и выдаёт сформированный заказ взаимодействуя с хранилищем «Сформированные заказы»;
* «Обновление данных системы» принимает сформированный заказ, взаимодействует с хранилищем «Список товаров» для записи обновлённых данных о товарах и заказах, выдаёт актуализированные данные;
* «Формирование        отчётности» принимает актуализированные данные о товарах и сформированный заказ, создаёт отчёт о выполненном заказе для администратора.

Разработанные DFD-модели обеспечивают полное описание потоков данных и процессов обработки информации в системе каталога канцелярских товаров, позволяют понять логику движения данных через систему и служат основой для дальнейшего проектирования базы данных и разработки программного обеспечения.

Представлены и подробно описаны DFD-модели для проектируемого ПО в ПРИЛОЖЕНИИ Г.

# ОБЪЕКТНЫЙ ПОДХОД К ПРОЕКТИРОВАНИЮ И РАЗРАБОТКЕ ПО ЯЗЫКУ UML

## Разработка диаграммы вариантов использования

Диаграмма вариантов использования является одним из ключевых элементов объектно-ориентированного проектирования программного обеспечения и служит для визуализации функциональных требований к системе с точки зрения пользователей.

Диаграмма вариантов использования разработана для каталога канцелярских товаров и отражает взаимодействие двух основных типов пользователей системы с её функциональными возможностями. Основной целью данной диаграммы является наглядное представление функционала системы и определение границ ответственности каждого актора.

В системе каталога канцелярских товаров выделены два основных актера:

Клиент – конечный пользователь системы, который осуществляет поиск и заказ товаров. Этот актор имеет ограниченные права доступа и может взаимодействовать только с клиентской частью системы.

Администратор – привилегированный пользователь, обладающий полными правами на управление системой, включая управление каталогом товаров, категориями и пользователями.

Для актера «Клиент» определены следующие варианты использования:

* просмотр каталога – базовый вариант использования, позволяющий клиенту просматривать доступный ассортимент канцелярских товаров. Данный вариант использования включает в себя функцию поиска товаров (отношение «include»), что обеспечивает клиенту возможность быстрого нахождения нужных позиций;
* поиск товаров – функция, обязательно включаемая в процесс просмотра каталога, позволяющая осуществлять фильтрацию и поиск товаров по различным критериям (название, категория, цена);
* просмотр и детали товара – вариант использования, который может быть расширен (отношение «extend») как из просмотра каталога, так и из процесса поиска товаров. Предоставляет клиенту подробную информацию о выбранном товаре, включая описание, характеристики, цену и наличие;
* добавление товара в корзину – функция, расширяющая просмотр деталей товара, позволяет клиенту формировать корзину для последующего оформления заказа;
* оформление заказов – завершающий вариант использования в цепочке взаимодействия клиента с системой, позволяющий создать заказ на основе товаров в корзине;
* просмотр истории заказов – дополнительная функция, предоставляющая клиенту возможность отслеживать свои предыдущие заказы и их статусы.

Для актера «Администратор» определены следующие варианты использования:

* аутентификация – обязательный вариант использования, обеспечивающий безопасный вход в административную панель системы и проверку прав доступа;
* управление товарами – комплексный вариант использования, включающий в себя три основных операции:
* добавление товара (отношение «extend»);
* редактирование товара (отношение «extend»);
* удаление товара (отношение «extend»).
* управление категориями – функция, позволяющая администратору создавать, изменять и удалять категории товаров для структурирования каталога;
* управление пользователями – вариант использования, обеспечивающий администрирование учетных записей пользователей системы;
* обработка заказа – функция управления заказами клиентов, включающая изменение статусов, подтверждение и обработку заказов. Данный вариант использования включает в себя (отношение «include») функцию просмотра деталей заказа;
* просмотр деталей заказа — обязательная часть процесса обработки заказа, позволяющая администратору видеть полную информацию о составе заказа, клиенте и статусе выполнения.

В разработанной диаграмме использованы два типа отношений между вариантами использования:

Отношение «include» – обозначает обязательное включение одного варианта использования в другой. Например, процесс просмотра каталога обязательно включает функцию поиска товаров, а обработка заказа всегда включает просмотр его деталей.

Отношение «extend» – обозначает необязательное расширение базового варианта использования дополнительной функциональностью при определенных условиях. Например, просмотр деталей товара может расширять процесс просмотра каталога, но не является обязательным действием.

Разработанная диаграмма вариантов использования полностью охватывает функциональные требования к системе каталога канцелярских товаров. Она четко разграничивает права и возможности двух типов пользователей, обеспечивая при этом полноту функционала для эффективной работы интернет-магазина. Диаграмма служит основой для дальнейшей детализации требований и проектирования архитектуры системы.

Представлены и подробно описана диаграмма вариантов использования для проектируемого ПО в ПРИЛОЖЕНИИ Д.

## Разработка диаграммы классов

Диаграмма классов представляет собой статическую структурную модель системы каталога канцелярских товаров, отражающую основные сущности предметной области, их атрибуты, методы и взаимосвязи между ними. Такая диаграмма позволяет визуализировать архитектуру системы и определить, как различные компоненты взаимодействуют между собой. Она служит основой для разработки и помогает разработчикам понять структуру данных и логику приложения.

На основе проведенного анализа предметной области и разработанной диаграммы «сущность-связь» была построена диаграмма классов, включающая семь основных классов: Пользователь, Категория, Товар, Заказ, Корзина, Заказ\_Товар и Фотография.

Класс «Пользователь» является центральным элементом системы и содержит информацию о зарегистрированных пользователях. Данный класс обеспечивает разграничение прав доступа между клиентами и администраторами системы.

Таблица 4.1 – Описание класса «Пользователь»

|  |  |  |  |
| --- | --- | --- | --- |
| **Класс** | **Атрибут** | **Тип данных** | **Описание** |
| Пользователь | Логин | char[50] | Имя пользователя для входа в систему |
| Пароль | char[255] | Хешированный пароль |
| Email | char[255] | Адрес электронной почты |
| Роль | char[20] | Роль пользователя (клиент/администратор) |
| Дата\_регистрации | DateTime | Дата и время регистрации в системе |

Класс «Категория» используется для структурирования и классификации товаров в каталоге. Иерархическая организация категорий позволяет клиентам быстро находить необходимые товары и упрощает навигацию по каталогу.

Таблица 4.2 – Описание класса «Категория»

|  |  |  |  |
| --- | --- | --- | --- |
| **Класс** | **Атрибут** | **Тип данных** | **Описание** |
| **Категория** | Название | String | Наименование категории |
| Описание | String | Описание категории товаров |

Класс «Товар» представляет собой основную сущность каталога и содержит полную информацию о товарных позициях. Класс включает все необходимые атрибуты для отображения товара клиенту и управления складскими остатками.

Таблица 4.3 – Описание класса «Товар»

|  |  |  |  |
| --- | --- | --- | --- |
| **Класс** | **Атрибут** | **Тип данных** | **Описание** |
| **Товар** | Название | сhar [150] | Наименование товара |
| Артикул | сhar [50] | Уникальный артикул товара |
| Цена | decimal (10,2) | Стоимость товара |
| Количество\_на\_складе | Integer | Доступное количество на складе |
| Описание | char [500] | Подробное описание товара |
| Дата\_добавления | DateTime | Дата добавления товара в каталог |

Класс «Заказ» управляет процессом оформления и обработки заказов клиентов. Данный класс отслеживает все этапы жизненного цикла заказа от создания до доставки.

Таблица 4.4 – Описание класса «Заказ»

|  |  |  |  |
| --- | --- | --- | --- |
| **Класс** | **Атрибут** | **Тип данных** | **Описание** |
| Заказ | Дата\_заказа | DateTime | Дата и время создания заказа |
| Статус | char[20] | Статус заказа (новый, в обработке, доставляется и т.д.) |
| Сумма | money | Общая сумма заказа |
| Адрес\_доставки | char[50] | Адрес доставки заказа |

Класс «Корзина» реализует функционал временного хранения товаров перед оформлением заказа. Каждый пользователь имеет свою персональную корзину, в которой может накапливать выбранные товары, изменять их количество и удалять позиции. Корзина не выполняет резервирование на складе – наличие проверяется уже при оформлении заказа.

Таблица 4.5 – Описание класса «Корзина»

|  |  |  |  |
| --- | --- | --- | --- |
| **Класс** | **Атрибут** | **Тип данных** | **Описание** |
| Корзина | Количество | Integer | Количество единиц товара в корзине |

Класс «Заказ\_Товар» является ассоциативным классом, реализующим связь «многие ко многим» между заказами и товарами. Класс сохраняет цену товара на момент оформления заказа, что обеспечивает историческую точность данных.

Таблица 4.6 – Описание класса «Заказ\_Товар»

|  |  |  |  |
| --- | --- | --- | --- |
| **Класс** | **Атрибут** | **Тип данных** | **Описание** |
| Заказ-товар | Количество | Integer | Количество единиц товара в заказе |
| Цена\_на\_момент заказа | money | Цена товара на момент оформления заказа |

Класс «Фотография» обеспечивает хранение и управление изображениями товаров. Возможность загрузки нескольких фотографий для одного товара позволяет продемонстрировать его с разных ракурсов.

Таблица 4.7 – Описание класса «Фотография»

|  |  |  |  |
| --- | --- | --- | --- |
| **Класс** | **Атрибут** | **Тип данных** | **Описание** |
| Фотография | Путь\_файла | char[500] | Путь к файлу изображения в файловой системе |
| Дата\_загрузки | DateTime | Дата и время загрузки изображения |

Методы класса «Пользователь» обеспечивают полный цикл работы пользователя в системе от регистрации до оформления заказов. Данные методы реализуют как клиентский, так и административный функционал.

Таблица 4.8 – Описание методов класса «Пользователь»

|  |  |  |
| --- | --- | --- |
| **Класс** | **Метод** | **Описание** |
| **Пользователь** | регистрация() | Создание новой учетной записи пользователя |
| аутентификация() | Проверка учетных данных при входе в систему | |
| просмотрИстории() | Получение списка всех заказов пользователя | |
| оформитьЗаказ() | Создание нового заказа на основе содержимого корзины | |

Методы класса «Категория» предоставляют администратору полный контроль над структурой каталога. Функционал позволяет создавать, изменять и удалять категории товаров.

Таблица 4.9 – Описание методов класса «Категория»

|  |  |  |
| --- | --- | --- |
| **Класс** | **Метод** | **Описание** |
| **Категория** | добавитьКатегорию() | Создание новой категории товаров |
| редактироватьКатегорию() | Изменение данных существующей категории |
| удалитьКатегорию() | Удаление категории из системы |
| получитьТовары() | Получение списка товаров, принадлежащих категории |

Методы класса «Товар» реализуют все операции по управлению товарными позициями каталога. Функционал включает добавление, редактирование, удаление товаров и получение информации о них.

Таблица 4.10 – Описание методов класса «Товар»

|  |  |  |
| --- | --- | --- |
| **Класс** | **Метод** | **Описание** |
| **Товар** | добавитьТовар() | Внесение нового товара в каталог |
| редактироватьТовар() | Изменение характеристик товара |
| удалитьТовар() | Удаление товара из каталога |
| получитьДетали() | Извлечение полной информации о товаре |
| проверитьНаличие() | Проверка доступности товара на складе |

Методы класса «Заказ» обеспечивают полный жизненный цикл обработки заказов от создания до завершения. Функционал позволяет отслеживать статус заказа и управлять им на всех этапах. Система автоматически отслеживает дату создания и планируемую дату доставки, обновляя статус заказа по мере его прохождения через различные этапы обработки.

Таблица 4.11 – Описание методов класса «Заказ»

|  |  |  |
| --- | --- | --- |
| **Класс** | **Метод** | **Описание** |
| **Заказ** | создатьЗаказ() | Инициализация нового заказа |
| обработатьЗаказ() | Выполнение действий по подтверждению заказа |
| изменитьСтатус() | Обновление статуса заказа |
| рассчитатьСумму() | Вычисление итоговой стоимости заказа |
| отменитьЗаказ() | Отмена заказа |

Методы класса «Корзина» предоставляют клиенту удобный интерфейс для управления выбранными товарами. Функционал позволяет добавлять, удалять товары и изменять их количество перед оформлением заказа.

Таблица 4.12 – Описание методов класса «Корзина»

|  |  |  |
| --- | --- | --- |
| **Класс** | **Метод** | **Описание** |
| **Корзина** | добавитьВКорзину() | Размещение товара в корзину |
| удалитьИзКорзины() | Удаление товара из корзины |
| изменитьКоличество() | Корректировка количества товара |
| очиститьКорзину() | Полное удаление содержимого корзины |
| получитьСодержимое() | Отображение всех товаров в корзине |

Методы класса «Заказ\_Товар» обеспечивают связь между заказами и товарами с сохранением всей необходимой информации. Функционал позволяет формировать состав заказа и рассчитывать стоимость отдельных позиций.

Таблица 4.13 – Описание методов класса «Заказ\_товар»

|  |  |  |
| --- | --- | --- |
| **Класс** | **Метод** | **Описание** |
| **Заказ\_Товар** | добавитьТоварВЗаказ() | Создание связи между заказом и товаром |
| получитьСтоимость() | Расчет стоимости позиции в заказе |

Методы класса «Фотография» обеспечивают работу с изображениями товаров в системе. Функционал включает загрузку новых фотографий, их удаление и получение путей к файлам для отображения.

Таблица 4.14 – Описание методов класса «Фотография»

|  |  |  |
| --- | --- | --- |
| **Класс** | **Метод** | **Описание** |
| **Фотография** | загрузитьФото() | Добавление нового изображения товара |
| удалитьФото() | Удаление изображения из системы |
| получитьПуть() | Получение пути к файлу изображения |

Связи между классами определяют взаимодействие различных компонентов системы и отражают бизнес-логику каталога канцелярских товаров. В таблице представлены все типы связей с указанием множественности и описанием их назначения.

Таблица 4.15 – Связи между классами

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Класс-источник** | **Класс-назначение** | **Тип связи** | **Множественность** | **Название связи** | **Описание** |
| 1 | 2 | 3 | 4 | 5 | 6 |
| Пользователь | Товар | Ассоциация | 1 : 0..\* | создает | Администратор может добавлять множество товаров в каталог |
| Пользователь | Заказ | Ассоциация | 1 : 0..\* | оформляет | Один пользователь может создать несколько заказов |
| Пользователь | Корзина | Ассоциация | 1 : 0..1 | имеет | У каждого пользователя может быть только одна активная корзина |

Продолжение табл. 4.15

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| Пользователь | Фотография | Ассоциация | 1 : 0..\* | загружает | Пользователь может загружать множество изображений |
| Категория | Товар | Ассоциация | 1 : 0..\* | содержит | Категория может содержать множество товаров, каждый товар принадлежит одной категории |
| Товар | Корзина | Ассоциация | 1 : 0..\* | находится в | Товар может находиться в корзинах разных пользователей |
| Товар | Фотография | Ассоциация | 1 : 0..\* | имеет | Товар может иметь несколько изображений |
| Товар | Заказ\_Товар | Ассоциация | 1 : 0..\* | связан с | Товар может входить в состав множества заказов |
| Заказ | Заказ\_Товар | Композиция | 1 : 1..\* | содержит | Заказ обязательно содержит одну или несколько товарных позиций. При удалении заказа удаляются все связанные позиции |

Класс «Заказ\_Товар» является ассоциативным классом, реализующим связь типа «многие ко многим» между классами «Заказ» и «Товар». Данный подход позволяет хранить дополнительную информацию о связи, такую как количество товара в конкретном заказе и цену на момент оформления заказа, что критически важно для исторической точности данных при изменении текущих цен на товары.

Связь между классами «Заказ» и «Заказ\_Товар» реализована как композиция, обозначенная черным ромбом на диаграмме. Это означает сильную зависимость между объектами: экземпляры класса «Заказ\_Товар» не могут существовать без соответствующего экземпляра класса «Заказ». При удалении заказа автоматически удаляются все связанные с ним товарные позиции, что обеспечивает целостность данных.

Все внешние ключи в классах явно указаны через атрибуты с префиксом «ID\_», что обеспечивает прозрачность связей и упрощает последующую реализацию на уровне базы данных.

Разработанная диаграмма классов полностью соответствует принципам объектно-ориентированного проектирования и обеспечивает необходимую функциональность для создания системы каталога канцелярских товаров. Структура классов обладает высокой связностью внутри классов и слабой связанностью между классами, что способствует модульности и масштабируемости системы. Диаграмма служит основой для последующей реализации системы на выбранном языке программирования и может быть использована для автоматической генерации программного кода.

Представлены и подробно описаны диаграмма классов для проектируемого ПО в ПРИЛОЖЕНИИ Е.

## Разработка диаграммы последовательности

Диаграмма последовательности является ключевым инструментом динамического моделирования системы и отражает взаимодействие объектов во времени для выполнения определенных сценариев использования. В рамках проектирования системы каталога канцелярских товаров были разработаны две диаграммы последовательности, отражающие наиболее важные бизнес-процессы: работу клиента с каталогом и административное управление товарами.

Первая диаграмма описывает процесс просмотра каталога и оформления заказа клиентом. Взаимодействие начинается с открытия каталога, когда клиент инициирует запрос к системе. Система обращается к базе данных для получения актуального списка товаров и отображает каталог пользователю. При выборе конкретного товара система запрашивает детальную информацию из базы данных и предоставляет клиенту полное описание, включая цену, характеристики и доступное количество. После принятия решения о покупке клиент добавляет товар в корзину, и система фиксирует этот выбор. Завершающим этапом является оформление заказа, при котором система сохраняет всю информацию в базе данных и отправляет клиенту подтверждение с номером заказа.

Вторая диаграмма последовательности демонстрирует административные функции управления товарами. Процесс начинается с аутентификации администратора, когда система проверяет учетные данные и предоставляет доступ к панели управления. Администратор может выполнять операции добавления нового товара, при этом система валидирует введенные данные перед сохранением в базу данных. Процесс редактирования товара включает передачу идентификатора и новых данных, которые после валидации обновляются в базе данных с последующим подтверждением операции. Удаление товара осуществляется по идентификатору с немедленным удалением соответствующей записи из базы данных. Дополнительно администратор может запрашивать аналитические отчеты, указывая тип и период, система извлекает необходимые данные из базы и формирует структурированный отчет для представления.

Разработанные диаграммы последовательности детально описывают временные аспекты взаимодействия между клиентом, системой и базой данных, демонстрируя последовательность вызовов методов и обмен сообщениями между компонентами. Диаграммы отражают как клиентский функционал просмотра и заказа товаров, так и административные операции по управлению каталогом, что обеспечивает полное понимание динамического поведения системы и служит основой для последующей реализации программного кода.

Представлены и подробно описаны диаграммы последовательности для проектируемого ПО в ПРИЛОЖЕНИИ Ж.

## Разработка диаграммы деятельности

Диаграмма деятельности представляет собой важный инструмент моделирования бизнес-процессов и рабочих потоков системы, демонстрирующий последовательность выполнения действий, точки принятия решений и возможные альтернативные пути выполнения операций. В контексте разработки системы каталога канцелярских товаров были созданы две диаграммы деятельности, отражающие ключевые бизнес-процессы: обработку заказов администратором и процесс оформления заказа клиентом.

Первая диаграмма деятельности описывает процесс обработки заказа администратором системы. Процесс начинается с аутентификации администратора в системе, после чего осуществляется просмотр списка новых заказов. В случае отсутствия новых заказов процесс завершается. При наличии заказов администратор выбирает конкретный заказ для обработки и просматривает его детали. Критическим этапом является проверка наличия товаров на складе, которая определяет дальнейший ход процесса. Если все товары доступны, администратор подтверждает заказ, система автоматически обновляет его статус на "В обработке", формируется накладная, и клиент получает уведомление о готовности заказа к выдаче. В противном случае, при отсутствии необходимых товаров, администратор отменяет заказ, и система автоматически отправляет клиенту уведомление об отмене с указанием причины. После обработки текущего заказа администратор принимает решение о продолжении работы с другими заказами или завершении сеанса работы. Диаграмма использует разделение на дорожки (swimlanes) для четкого распределения действий между администратором и системой: административная дорожка содержит действия, требующие принятия решений человеком, такие как просмотр заказов, проверка наличия товаров и подтверждение или отмена заказа, в то время как системная дорожка включает автоматические операции обновления статусов и отправки уведомлений клиентам.

В диаграмме так же уделено внимание обработке исключительной ситуации, возникающей при ошибке оформления заказа администратором. При оформлении заказа администратор может увидеть ошибку о том что клиент заполнил неполностью заявку

Вторая диаграмма деятельности детализирует процесс оформления заказа клиентом и включает обработку исключительных ситуаций. Процесс начинается с открытия каталога товаров, где клиент просматривает доступный ассортимент, выбирает интересующий товар и изучает его детальное описание. После принятия решения о покупке товар добавляется в корзину, и система обновляет её содержимое. На этом этапе клиент принимает решение о продолжении покупок или переходе к оформлению заказа. Если клиент решает добавить ещё товары, процесс возвращается к просмотру каталога, создавая циклический поток. При переходе к оформлению заказа клиент вводит данные доставки, которые система автоматически валидирует на корректность и полноту. В случае обнаружения ошибок система отображает соответствующие сообщения и предлагает клиенту исправить некорректные данные, после чего валидация повторяется. После успешной валидации клиент выбирает способ оплаты и подтверждает создание заказа. На этом этапе система проверяет актуальное наличие всех товаров из корзины в базе данных. Если все товары доступны, система создаёт запись заказа в базе данных, резервирует товары, отправляет подтверждение на электронную почту клиента и отображает сообщение об успешном оформлении заказа. В случае недоступности некоторых товаров система отображает список недоступных позиций и предоставляет клиенту выбор: продолжить оформление заказа только с доступными товарами, автоматически удалив недоступные из корзины и вернувшись к выбору способа оплаты, либо полностью отменить оформление заказа.

Разработанные диаграммы деятельности обеспечивают детальное понимание логики выполнения бизнес-процессов системы, включая все возможные варианты развития событий, точки принятия решений и обработку исключительных ситуаций. Использование дорожек для разделения ответственности между пользователями и системой повышает читаемость диаграмм и четко определяет границы автоматизации. Диаграммы служат основой для разработки программной логики, тестирования системы и документирования бизнес-правил, обеспечивая согласованное понимание процессов всеми участниками проекта.

Представлены и подробно описаны диаграммы деятельности для проектируемого ПО в ПРИЛОЖЕНИИ И.

## Разработка диаграммы состояний

Диаграмма состояний – это моделирование поведения системы через последовательность состояний и переходов между ними. Она используется для описания всех возможных режимов работы объекта и правил переходов от одного режима к другому.

Диаграмма состояний жизненного цикла заказа используется для моделирования и визуализации всех возможных состояний, в которых может находиться заказ, а также переходов между ними. Она определяет допустимые пути выполнения заказа от момента его создания до архивирования.

Диаграмма состояний необходима для:

* определения правил переходов между состояниями и условий, при которых эти переходы возможны;
* документирования действий, выполняемых при входе в состояние, во время нахождения в нём и при выходе из него;
* обеспечения согласованности бизнес-логики во всей системе;
* обработки исключительных ситуаций (отмена заказа, недостаток товара, возврат);
* упрощения разработки и тестирования системы управления заказами.

Каждое состояние содержит три типа действий: entry (действия при входе в состояние), do (действия во время нахождения в состоянии) и exit (действия при выходе из состояния).

Таблица 4.16 – Описание состояний

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Состояние | Описание | | Entry | Do | Exit |
| 1 | 2 | | 3 | 4 | 5 |
| Создан | Заказ только создан в системе | | Присвоить номер заказа | Ожидать подтверждения оплаты | Зафиксировать время перехода |
| Оплачен | Платёж успешно получен | | Зарезервировать товары на складе | Отправить уведомление клиенту | Обновить статистику оплат |
| В обработке | Заказ готовится к отправке | | Назначить заказ администратору | Проверить наличие товаров | Снять блокировку администратора |
| Подтвержден | | Администратор подтвердил заказ | Сформировать накладную | Передать на склад для комплектации | Закрыть накладную |
| Готов к выдаче | | Товары скомплектованы и упакованы | Уведомить клиента о готовности | Ожидать получения клиентом | Уведомить клиента о готовности Зафиксировать факт выдачи |
| Выполнен | | Заказ доставлен клиенту | Записать дату выполнения | Обновить статистику продаж | Архивировать заказ |

Продолжение табл. 4.17

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 |
| Возврат | Клиент запросил возврат товара | Инициировать процесс возврата | Проверить товар, подготовить к отправке | Обновить статистику возвратов |
| Отменен | Заказ отменён | Освободить зарезервированные товары | Инициировать возврат средств | Уведомить клиента об отмене |

Основной путь заказа (нормальный сценарий):

1. Создан – клиент создал заказ, система зафиксировала время и номер;
2. Оплачен – платёж прошёл успешно, оплата подтверждена;
3. В обработке – товары есть на складе, заказ комплектуется;
4. Подтвержден – товары собраны и готовы к отправке;
5. Готов к выдаче – заказ упакован и ждёт отправки клиенту;
6. Выполнен – заказ отправлен клиенту;
7. Завершен – заказ получен, процесс полностью завершен.

Таблица 4.17 – Переходы состояний

|  |  |  |  |
| --- | --- | --- | --- |
| Из состояния | В состояние | Условие | Описание |
| 1 | 2 | 3 | 4 |
| Создан | Оплачен | ПолученаОплата | Платёж успешно обработан |
| Создан | Отменен | ОтменитьЗаказ | Клиент или администратор отменили заказ |
| Оплачен | В обработке | ЗаказПринятВОбработку | Администратор принял заказ в обработку |
| Оплачен | Отменен | ОплатаОтклонена | Заказ отменён до обработки |
| В обработке | Подтвержден | ЗаказПодтвержден | Все товары в наличии и подтверждены | |
| В обработке | Отменен | НетВНаличии | Товар не в наличии или заказ отменён | |
| Подтвержден | Готов к выдаче | ТоварыСкомплектованы | Товары собраны и упакованы на складе | |

Продолжение табл. 4.17

|  |  |  |  |
| --- | --- | --- | --- |
| 1 | 2 | 3 | 5 |
| Готов к выдаче | Выполнен | ЗаказПолучен | Клиент получил заказ |
| Выполнен | Возврат | ЗапрошенВозврат | Клиент запросил возврат товара |
| Возврат | Завершен | ВозвратЗавершен | Возврат обработан и завершён |
| Отменен | Завершен | - | Архивирование отменённого заказа |

Возврат возможен только из состояния «Выполнен», так как товар должен быть доставлен клиенту. Отмена возможна из состояний «Создан», «Оплачен» и «В обработке». После отправки отмена невозможна. Финальное состояние «Завершен» используется для архивирования всех типов заказов. Переходы в системе однонаправленные без возврата в предыдущие состояния, кроме как через процессы отмены или возврата.

Особые случаи: если товар не в наличии в состоянии «В обработке», заказ переходит в «Отменен»; если заказ долго находится в состоянии «Готов к выдаче», администратор может его отменить; если платёж отклонен, заказ остаётся в «Создан» или переходит в "Отменен».

Представлены и подробно описана диаграмма состояний для проектируемого ПО в ПРИЛОЖЕНИИ К.

# ПРОЕКТИРОВАНИЕ ИНТЕРФЕЙСА

## Проектирование интерфейса ПО

### Спецификация функций ПО на экранных формах

Основной функционал программного обеспечения реализуется посредством набора операций, доступных пользователям в зависимости от их роли в системе. В таблице 5.1 приведена детальная спецификация всех функций, реализуемых на экранных формах, с указанием ролей пользователей, имеющих доступ к каждой функции, и соответствующих форм интерфейса.

Таблица 5.1 – Спецификация функций ПО на экранных формах

|  |  |  |  |
| --- | --- | --- | --- |
| № функции | Название функции | Роль | Перечень форм |
| 1  1.1  1.2  1.3  1.4  1.5  1.6  1.7 | Выбор и вход в систему  Выбор роли  Ввод кода-доступа  Вход - Администратор  Регистрация нового клиента  Вход – Клиент  Смена пароля аккаунта  Восстановление пароля | Администратор, Клиент  Администратор  Администратор  Клиент  Клиент  Клиент  Клиент | N1  N2  N3  N6  N5  N9  N10 |
| 2  2.1  2.2  2.3  2.4  2.5  2.6  2.7 | Управление товарами  Просмотр списка товаров  Просмотр изображения товара  Поиск товара по совпадению  Сортировка товара по критериям  Добавление данных товаров  Редактирование данных товаров  Удаление данных товаров | Администратор, Клиент  Администратор, Клиент  Администратор, Клиент  Администратор, Клиент  Администратор  Администратор  Администратор | N4, N7  N4, N7  N4, N7  N4, N7  N4  N4  N4 |
| 3  3.1  3.2 | Управление корзиной  Добавление товара в корзину  Выбор количества товара для заказа | Клиент | N7  N8 |
| 4  4.1  4.2  4.3 | Управление заказами  Оформление заказа  Выполнение заказов  Формирование отчетов о выполненных заказах | Клиент  Администратор  Администратор | N7  N4  N4 |

Для реализации описанного функционала разработан набор экранных форм, которые обеспечивают удобное взаимодействие пользователя с системой. Перечень всех проектируемых форм, их назначение и ссылки на соответствующие макеты представлены в таблице 5.2.

Таблица 5.2 – Перечень экранных форм

|  |  |  |
| --- | --- | --- |
| № формы | Название | Проект экранной формы |
| 1 | 2 | 3 |
| N1 | Выбор роли | Рисунок Л.1 |
| N2 | Ввод кода-доступа | Рисунок Л.2 |
| N3 | Вход - Администратор | Рисунок Л.3 |
| N4 | Админ-панель | Рисунок Л.4 |
| N5 | Вход - Клиент | Рисунок Л.5 |
| N6 | Регистрация нового клиента | Рисунок Л.6 |
| N7 | Клиент-панель | Рисунок Л.7 |
| N8 | Клиент  Окно «Выбор количества товара для заказа» | Рисунок Л.8 |
| N9 | Клиент  Окно «Смена пароля клиента» | Рисунок Л.9 |
| N10 | Клиент  Окно «Восстановление пароля клиента» | Рисунок Л.10 |

### Граф переходов состояний интерфейса

Логика взаимодействия пользователя, обладающего правами администратора, с интерфейсом системы описывается посредством графа переходов между состояниями. На рисунке 5.1 представлена схема переходов, демонстрирующая все возможные маршруты навигации и операции, доступные администратору при работе с системой.

Граф переходов для роли «Администратор» представлен на рисунке 5.1.

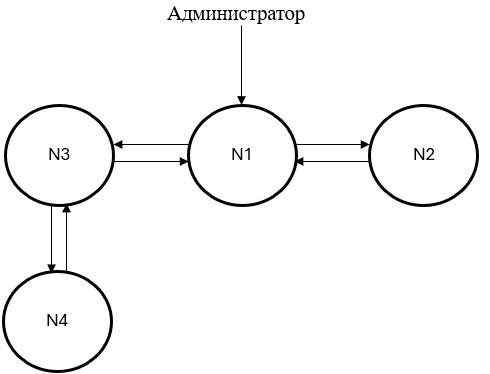
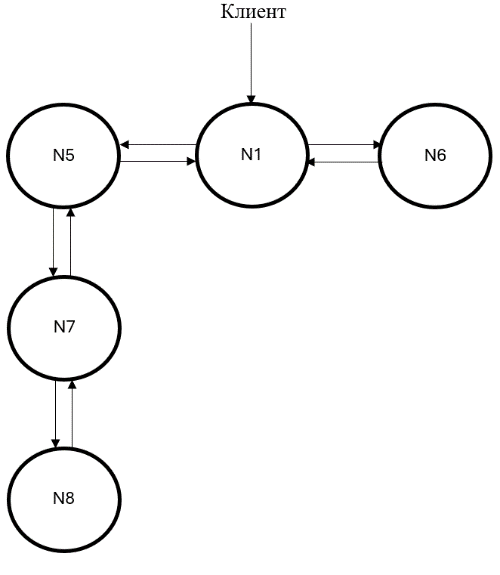


Рисунок 5.1. − Граф переходов для роли «Администратор»

Аналогично, для пользователей с ролью клиента разработана собственная логика навигации по интерфейсу. На рисунке 5.2 показан граф переходов, отражающий последовательность действий и доступные клиенту операции при взаимодействии с системой.

Граф переходов для роли «Клиент» представлен на рисунке 5.2.



N10

N9

Рисунок 5.2. − Граф переходов для роли «Клиент»

### Проект экранных форм

Для информационной системы был разработан пользовательский интерфейс в программе Figma. Проект экранных форм изображен в ПРИЛОЖЕНИИ Л.

Для проекта экранных форм был разработан прототип. На рисунке 5.3 представлен конструктор переходов между формами. Ознакомиться с прототипом можно по ссылке:

https://www.figma.com/design/h2uzc7fvz50awKkMdYmnJ3/Wireframes-1-?node-id=32-1128&t=TVjiwB6fLsYMsKDx-1

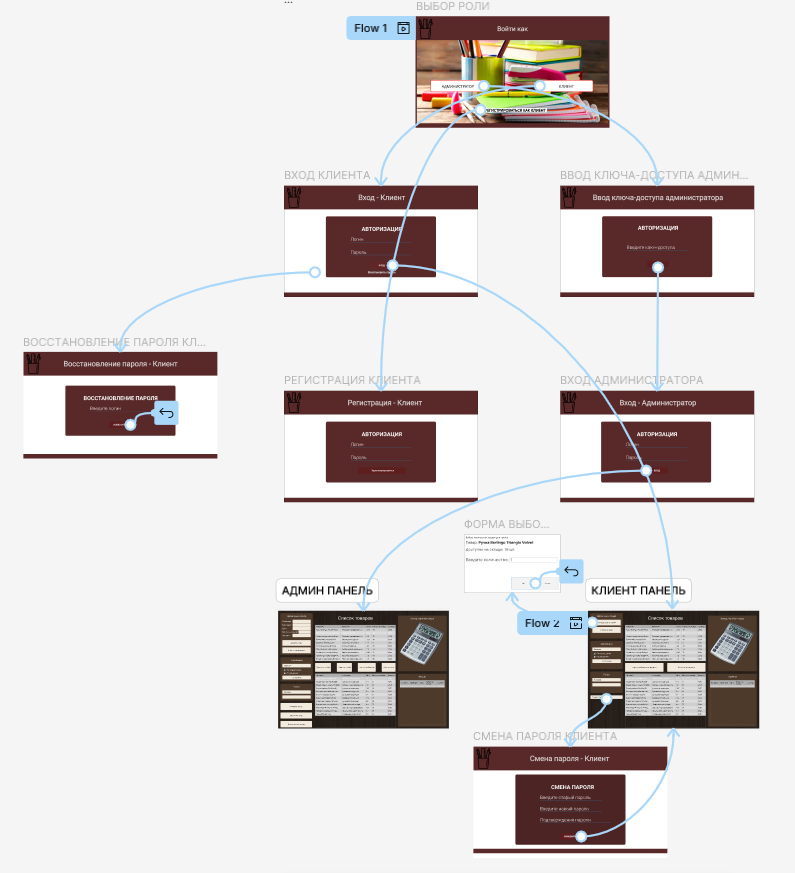


Рисунок 5.3 − Прототип «Каталога канцелярских товаров»

# ТЕСТИРОВАНИЕ ПРОГРАММНОГО СРЕДСТВА

Тестирование программного обеспечения – процесс проверки программного средства, направленный на выявление соответствия между фактическим поведением системы и ожидаемым, определённым техническим заданием и программой испытаний. Основная цель тестирования предполагает проверку функциональности: убедиться, что все функции системы управления каталогом канцелярских товаров реализованы корректно и соответствуют спецификации.

Выбранная основная стратегия – функциональное тестирование, поскольку целью испытаний является проверка корректности реализации ключевых функций, обеспечивающих жизненный цикл заказа: от регистрации пользователя до формирования итогового отчета о заказе. Данная стратегия позволяет выявить отклонения от требований технического задания на уровне пользовательских сценариев, а также убедиться в надёжности и стабильности работы системы в условиях, приближенных к реальной эксплуатации.

План тестирования разработан на основе документа «Программа и методика испытаний» и включает следующие сценарии:

1. Управление каталогом товаров (роль «Администратор»)

1.1 Проверка возможности просмотра списка товаров;

1.2 Добавление нового товара в каталог;

1.3 Удаление товара из каталога;

1.4 Редактирование характеристик товара;

1.5 Поиск товаров по наименованию;

1.6 Поиск товаров по категории канцелярии;

1.7 Поиск товаров по цене;

1.8 Сортировка товаров по наименованию;

1.9 Сортировка товаров по категории;

1.10 Сортировка товаров по цене;

1.11 Проверка сохранения списка товаров в базу данных.

2. Аутентификация пользователей

2.1 Проверка ввода ключа доступа администратора;

2.2 Проверка аутентификации администратора;

2.3 Проверка регистрации клиента;

2.4 Проверка аутентификации клиента;

2.5 Проверка валидации учётных данных;

2.6 Проверка восстановления пароля клиента;

2.7 Проверка смены пароля клиента.

3. Выполнение заказов клиентов (роль «Администратор»)

3.1 Проверка возможности просмотра списка заказов;

3.2 Добавление данных о новом заказе;

3.3 Выполнение заказа;

3.4 Сохранение выполненного заказа.

4. Работа с каталогом товаров (роль «Клиент»)

4.1 Проверка возможности просмотра каталога канцелярских товаров;

4.2 Поиск товаров по наименованию;

4.3 Поиск товаров по категории канцелярии;

4.4 Поиск товаров по цене;

4.5 Поиск товаров по кол-ву на складе;

4.6 Сортировка товаров по наименованию;

4.7 Сортировка товаров по категории;

4.8 Сортировка товаров по цене.

5. Работа с корзиной и заказами (роль «Клиент»)

5.1 Добавление товаров в корзину;

5.2 Просмотр содержимого корзины;

5.3 Удаление товара из корзины;

5.4 Редактирование количества товара в корзине;

5.5 Оформление заказа.

Тестирование программного обеспечения – процесс испытания программного средства, имеющий своей целью проверку соответствия между реальным поведением программы и её ожидаемым поведением на конечном наборе тестов, выбранных определенным образом.

Для каждого тестового сценария определены входные данные, ожидаемые результаты и критерии успешного прохождения теста. Данный подход позволяет систематически проверить все аспекты функционирования системы и обеспечить полноту охвата тестирования.

Тестирование проводится путем последовательного выполнения установленных сценариев и регистрации результатов. При выявлении несоответствий между ожидаемым и фактическим поведением системы фиксируются обнаруженные ошибки и недостатки, которые требуют устранения.

Функциональное тестирование проводилось на основании документа «Программа и методика испытаний». Результаты функционального тестирования представлены в таблице (ПРИЛОЖЕНИЕ Н), которая содержит детальную информацию о каждом выполненном тестовом случае, включая входные данные, ожидаемые результаты, фактические результаты и итоговый статус прохождения теста.

# РЕАЛИЗАЦИЯ ПРОГРАММНОГО СРЕДСТВА

## Структура программного средства

Программное средство предназначено для автоматизации учета ремонта автомобилей в сервисных центрах (СТО). Система позволяет управлять процессом починки, отслеживать этапы выполнения работ и вести статистику по выполненным заказам. С системой работают два пользователя: Администратор и Рабочий.

В таблице 7.1 представлен перечень файлов, входящих в состав проектируемого программного средства, а также их назначение и роль для функционирования системы. Каждый файл выполняет определённую функцию, необходимую для корректной работы проекта, обеспечения интерфейса пользователя и взаимодействия с базой данных.

Таблица 7.1 – Назначение файлов системы

|  |  |
| --- | --- |
| Название файла | Назначение |
| Motor.exe | Исполняемый файл |
| App.config | Подключение к базе данных приложения |
| Motor.accdb | Файл базы данных |
| Guna.UI2.dll | Библиотека для интерфейса |
| MaterialSkin.dll | Библиотека для интерфейса |
| MetroFromework.dll | Библиотека для интерфейса |
| RJCodeAdvance.RJControls.dll | Библиотека для интерфейса |

## Диаграмма развертывания

Логическая структура проектируемого программного средства представлена в виде диаграммы развертывания, которая отображена в ПРИЛОЖЕНИИ М. Диаграмма демонстрирует взаимодействие основных компонентов системы и их размещение на локальном компьютере пользователя.

На диаграмме представлен 1 узел:

* локальный компьютер пользователя.

На локальном ПК пользователя находятся:

* среда выполнения .NET Framework 4.7.2 (Execution Environment);
* исполняемый файл программы (Motor.exe);
* файл подключения к базе данных (App.config);

Библиотеки проекта:

* Guna.UI2.dll (для интерфейса);
* MaterialSkin.dll (для интерфейса);
* MetroFramework.dll (для интерфейса);
* RJCodeAdvance.RJControls.dll (для интерфейса);
* файл базы данных (Motor.accdb);
* СУБД MS Access (Execution Environment).

Взаимодействие компонентов:

* Motor.exe развертывается (deploy) в среде выполнения .NET Framework 4.7.2;
* Motor.exe манифестирует (manifest) файл конфигурации App.config для подключения к базе данных;
* Motor.exe манифестирует (manifest) все DLL библиотеки для отображения интерфейса пользователя;
* Guna.UI2.dll;
* MaterialSkin.dll;
* MetroFramework.dll;
* RJCodeAdvance.RJControls.dll;
* Motor.accdb развертывается (deploy) в СУБД MS Access;
* Motor.exe обращается к базе данных Motor.accdb через MS Access для выполнения операций с данными (создание, чтение, обновление, удаление).

Данная структура позволяет обеспечить логичное распределение компонентов системы, упрощает дальнейшее сопровождение проекта и обеспечивает надёжность функционирования программного средства.

## Установка программного средства

Для корректного развёртывания проектируемого программного средства необходимо ознакомиться с руководством: «Руководство оператора» (ПРИЛОЖЕНИЕ П).

Установка приложения выполняется посредством копирования всех файлов из папки EXE, предоставленной на прилагаемом носителе, на локальный диск компьютера пользователя. Для корректной работы программного средства необходимо наличие установленной среды .NET Framework не ниже версии 4.7.2 и СУБД MS Access, обеспечивающей работу с базой данных.

При развёртывании следует руководствоваться диаграммой развертывания (ПРИЛОЖЕНИЕ М), что позволяет избежать ошибок в установке и обеспечивает правильное взаимодействие всех компонентов проекта. Благодаря такой организации установка программного средства становится интуитивно понятной и позволяет пользователю быстро подготовить систему к использованию.

# СОПРОВОЖДЕНИЕ ПРОГРАММНОГО СРЕДСТВА

Согласно техническому заданию, информационная система должна быть укомплектована следующим программным документом «Руководство оператора».

Программная документация должна быть оформлена в соответствии с ГОСТ 19.505-79 «Единая система программной документации. Руководство оператора. Требования к содержанию и оформлению».

Разработанный программный документ представлен в ПРИЛОЖЕНИИ П «Руководство оператора».

# ЗАКЛЮЧЕНИЕ

В ходе учебной практики были успешно выполнены все поставленные цели и задачи. Проведено проектирование каталога канцелярских товаров, предусматривающего структуру ведения ассортимента, систематизацию товарных позиций и взаимодействие с пользователями.

Спроектированный каталог демонстрирует возможности применения современных информационных технологий для решения задач управления торговым ассортиментом. Использование компьютерных средств для обработки, хранения и передачи информации позволяет представить проектное решение, обеспечивающее эффективное управление каталогом товаров.

Проект каталога предусматривает возможность хранения большого объёма информации о товарах и быстрого доступа к ней, что упрощает поиск необходимых данных и ускоряет их обработку. Кроме того, проект способствует сокращению использования бумажных носителей, что положительно сказывается на рациональном расходовании ресурсов.

Концепция каталога включает удобный интерфейс для администраторов и пользователей, а также логическую структуру данных, обеспечивающую высокую скорость работы и надёжность хранения информации. В результате учебной практики были углублены знания в области проектирования информационных систем, программирования и организации данных, что соответствует целям и задачам практики.

# СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

* 1. Программа и методика испытаний. Требования к содержанию и оформлению: ГОСТ 19.301-2001 ЕСПД – Минск: Изд-во стандартов, 1982.
  2. Руководство оператора. Требования к содержанию и оформлению: ГОСТ 19.505-79 ЕСПД – Минск: Изд-во стандартов, 1982.
  3. Техническое задание. Требования к содержанию и оформлению: ГОСТ 19.201-78 ЕСПД – М.: Издательство стандартов, 1987.
  4. Требования к дипломному (курсовому) проекту: СТП ПГЭК 2.03 – УО «ПГЭК».
  5. Васильев, А.Н. Программирование на C# в примерах и задачах. – М.: Эксмо, 2019. – 320 с.
  6. Керниган, Б. Язык программирования C# / Б. Керниган, Д. Ритчи. – М.: Вильямс, 2016. – 310 с.
  7. Орлов, С.А. Технология разработки программного обеспечения / С.А. Орлов – Спб.: Питер, 2002. – 464 с.
  8. Павловская, Т.А. C#/C#. Программирование на языке высокого уровня / Т.А. Павловская – СПб.: Питер, 2014. – 470 с.
  9. Страуструп, Б. Программирование: принципы и практика с использованием C# / Б. Страуструп – М.: Вильямс; 2012. – 1250 с.
  10. Фленов, М. Проектирование интерфейсов Windows Forms и WPF: C#/CLI. – М.: Диалектика, 2016. – 380 с.
  11. Официальная документация Microsoft Visual Studio [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/visualstudio/
  12. Документация C# [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/csharp/
  13. Учебные материалы по C# [Электронный ресурс] – Режим доступа: https://learn.microsoft.com/ru-ru/dotnet/csharp/tutorials/
  14. Библиотека стандартных классов .NET [Электронный ресурс] – Режим доступа: https://docs.microsoft.com/ru-ru/dotnet/api/
  15. Figma: Официальная документация и учебные материалы [Электронный ресурс] – Режим доступа: https://help.figma.com/hc/ru

# ПРИЛОЖЕНИЕ A (обязательное) Техническое задание

## A Введение

В настоящем техническом задании описаны требования к разработке информационной системы каталога канцелярских товаров. В результате внедрения этого решения ожидается улучшение процесса учёта канцелярских товаров и оперативной обработки заказов.

Целью данного проекта состоит в проектировании автоматизированного инструмента, для формирования удобного и действенного каталога канцелярских товаров. Ожидается, что система поможет свести к минимуму ошибки, увеличить прозрачность информации, а также обеспечить актуальное обновление сведений.

## A1 Основания для разработки

Основанием для разработки является задание на учебную практику «Разработка и сопровождение программного обеспечения» учащегося специальности 5-04-0612-02 «Разработка и сопровождение программного обеспечения информационных систем» Гайделя Алексея Васильевича.

Тема работы «Каталог канцелярских товаров».

## A2 Назначение разработки

## A2.1 Функциональное назначение

Функциональное назначение программы заключается в автоматизации, предоставлении структурированной информации о канцелярских товарах с возможностью поиска, фильтрации и просмотра детальной информации о продукции, обработки заказов.

## А2.2 Эксплуатационное назначение

Программа спроектирована для автоматизации каталога и контроля канцелярских товаров, используемых в торговле. Она предназначена для сотрудников, ответственных за управление товарами и обработку заказов, а также клиентов, занимающихся просмотром каталога и оформлением заказов.

## A3 Требования к программе

### A3.1 Требования к функциональным характеристикам

Приложение должно выполнять следующий функционал: формирование и управление каталогом канцелярских товаров, обеспечение функций поиска и сортировки товаров по различным параметрам, предоставление возможности просмотра детальной информации о товарах включая фотографии, организацию работы с корзиной товаров для клиентов, обработку заказов клиентов администратором, разграничение прав доступа между ролями пользователей.

### A3.2 Требования к надежности

Надежное (устойчивое) функционирование программы должно быть обеспечено выполнением совокупности организационно-технических мероприятий, перечень которых приведен ниже:

* а) организация бесперебойного питания технических средств;
* б) регулярное выполнение рекомендаций Постановления.
* Министерства труда и социальной защиты Республики Беларусь 23 марта 2011 г. № 19 «Об утверждении норм времени на работы по обслуживанию персональных электронно-вычислительных организационной техники и офисного оборудования»;
* в) машин, регулярное выполнение требований ГОСТ 51188-98 «Защита информации»;
* г) испытания программных средств на наличие компьютерных вирусов;
* д) необходимый уровень квалификации пользователей программы.

### A3.3 Условия эксплуатации

Условия эксплуатации программного продукта должны соответствовать климатическим требованиям, установленным для технических средств в части их функционирования. Для работы программного средства требуется не менее 2 сотрудников, при этом конечным пользователем является администратор. Администратор должен владеть базовыми навыками работы с графическим интерфейсом операционной системы. Обслуживание программного обеспечения выполняется согласно требованиям, обеспечивающим стабильное и надёжное функционирование программы.

### A3.4 Требования к составу и параметрам технических средств

В состав технических средств должен входить IBM – совместимый персональный компьютер (ПЭВМ), включающий в себя:

− процессор: 1 гигагерц (ГГц) или более быстрый процессор или SoC;

− оперативная память: 1 гигабайт (ГБ) для 32 – разрядных или 2 ГБ для 64 – разрядных;

− место на жестком диске: 32 ГБ для 64 – битной и 32 – битной ОС;

− видеокарта: с поддержкой DirectX 9;

− разрешение экрана: 800 x 600, минимальный размер диагонали для основного дисплея 7 дюймов и больше.

### A3.5 Требования к информационной и программной совместимости

Приложение должно быть установлено на компьютер, содержащий операционную систему Microsoft Windows 10 или Microsoft Windows 11. Программа будет разработана с использованием языка программирования C#. Требования к информационным структурам и методам решения не предъявляются. Требования к исходным кодам не предъявляются. Требования к программным средствам, используемым программой, не предъявляются. Требования к защите информации и программ не предъявляются.

### A3.6 Требования к маркировке и упаковке

Специальные требования к маркировке и упаковке программы не предъявляются.

### A3.7 Требования к транспортировке и хранению

Требования к транспортировке и хранению: специальные требования к транспортированию и хранению программы не предъявляются.

**А3.8 Специальные требования**

Специальные требования к программе не предъявляются.

## A4 Требования к программной документации

Программная документация должна быть представлена в следующем составе:

* «Техническое задание»;
* «Программа и методика испытаний»;
* «Руководство оператора».

Программная документация должна быть оформлена в соответствии со следующими нормативными документами:

* ГОСТ 19.201-78«ЕСПД. Техническое задание»;
* ГОСТ 19.301-2000 «ЕСПД. Программа и методика испытаний»;
* ГОСТ 19.505-79 «ЕСПД. Руководство оператора».

## A5 Стадии и этапы разработки

Разработка приложения включает в себя следующие стадии, представленные в таблице А.1.

Таблица А.1. – Стадии разработки информационной системы

|  |  |
| --- | --- |
| Этапы курсового проектирования | Срок выполнения |
| Введение. Описание предметной области | 20.10.2025 |
| Постановка задач. Функциональная модель программного средства | 5.11.2025 |
| Разработка технического задания | 10.11.2025 |
| Разработка навигационной схемы приложения | 11.11.2025 |
| Проектирования программного обеспечения | 12.11.2025 |
| Реализация и тестирования программного обеспечения | 13.11.2025 |
| Разработка программной документации | 14.11.2025 |
| Оформление отчета, графической части и приложения | 17.11.2025 |

## A6 Порядок контроля и приёмки

Приемо-сдаточные испытания программы должны проводиться согласно программному документу ГОСТ 19.301-2000 «ЕСПД. Программа и методика испытаний.

# ПРИЛОЖЕНИЕ Б (справочное) SQL-сценарий базы данных

CREATE TABLE Заказ\_6

(

ID\_Заказа integer NOT NULL ,

Дата\_заказа datetime NOT NULL ,

Статус varchar(20) NOT NULL ,

Сумма money NULL ,

Адрес\_доставки varchar(50) NOT NULL ,

ID\_Пользователя integer NOT NULL

)

go

ALTER TABLE Заказ\_6

ADD CONSTRAINT XPKЗаказ\_6 PRIMARY KEY NONCLUSTERED (ID\_Заказа ASC)

go

CREATE TABLE Заказ\_Товар\_7

(

ID\_Товара integer NOT NULL ,

ID\_Заказа integer NOT NULL ,

Количество integer NOT NULL ,

Цена\_на\_момент\_заказа money NOT NULL

)

go

ALTER TABLE Заказ\_Товар\_7

ADD CONSTRAINT XPKЗаказ\_Товар\_7 PRIMARY KEY NONCLUSTERED (ID\_Заказа ASC,ID\_Товара ASC)

go

CREATE TABLE Категория\_1

(

ID\_Категории integer NOT NULL ,

Название varchar(100) NOT NULL ,

Описание varchar(500) NOT NULL

)

go

ALTER TABLE Категория\_1

ADD CONSTRAINT XPKКатегория\_1 PRIMARY KEY NONCLUSTERED (ID\_Категории ASC)

go

CREATE TABLE Корзина\_4

(

ID\_Товара integer NOT NULL ,

ID\_Корзины integer NOT NULL ,

Количество integer NULL ,

ID\_Пользователя integer NOT NULL

)

go

ALTER TABLE Корзина\_4

ADD CONSTRAINT XPKКорзина\_4 PRIMARY KEY NONCLUSTERED (ID\_Корзины ASC,ID\_Пользователя ASC,ID\_Товара ASC)

go

CREATE TABLE Пользователь\_5

(

ID\_Пользователя integer NOT NULL ,

Логин varchar(50) NOT NULL ,

Пароль varchar(255) NOT NULL ,

Email varchar(255) NOT NULL ,

Роль varchar(20) NOT NULL ,

Дата\_регистрации datetime NOT NULL

)

go

ALTER TABLE Пользователь\_5

ADD CONSTRAINT XPKПользователь\_5 PRIMARY KEY NONCLUSTERED (ID\_Пользователя ASC)

go

CREATE TABLE Товар\_2

(

ID\_Категории integer NOT NULL ,

Название varchar(150) NOT NULL ,

Артикул varchar(50) NOT NULL ,

Цена decimal(10,2) NOT NULL ,

Количество\_на\_складе integer NULL ,

Описание varchar(1000) NOT NULL ,

Дата\_добавления datetime NOT NULL ,

ID\_Товара integer NOT NULL ,

ID\_Пользователя integer NULL

)

go

ALTER TABLE Товар\_2

ADD CONSTRAINT XPKТовар\_2 PRIMARY KEY NONCLUSTERED (ID\_Товара ASC)

go

CREATE TABLE Фотография\_3

(

ID\_Товара integer NULL ,

Путь\_файла varchar(500) NULL ,

Дата\_загрузки datetime NULL ,

ID\_Фотографии integer NOT NULL ,

ID\_Пользователя integer NULL

)

go

ALTER TABLE Фотография\_3

ADD CONSTRAINT XPKФотография\_3 PRIMARY KEY NONCLUSTERED (ID\_Фотографии ASC)

go

ALTER TABLE Заказ\_6

ADD CONSTRAINT может\_оформлять FOREIGN KEY (ID\_Пользователя) REFERENCES Пользователь\_5(ID\_Пользователя)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE Заказ\_Товар\_7

ADD CONSTRAINT может\_быть\_в\_заказе FOREIGN KEY (ID\_Товара) REFERENCES Товар\_2(ID\_Товара)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE Заказ\_Товар\_7

ADD CONSTRAINT содержит\_позицию FOREIGN KEY (ID\_Заказа) REFERENCES Заказ\_6(ID\_Заказа)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE Корзина\_4

ADD CONSTRAINT может\_быть\_в\_корзине FOREIGN KEY (ID\_Товара) REFERENCES Товар\_2(ID\_Товара)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE Корзина\_4

ADD CONSTRAINT имеет\_корзину FOREIGN KEY (ID\_Пользователя) REFERENCES Пользователь\_5(ID\_Пользователя)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE Товар\_2

ADD CONSTRAINT содержит FOREIGN KEY (ID\_Категории) REFERENCES Категория\_1(ID\_Категории)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE Товар\_2

ADD CONSTRAINT управляет FOREIGN KEY (ID\_Пользователя) REFERENCES Пользователь\_5(ID\_Пользователя)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE Фотография\_3

ADD CONSTRAINT имеет FOREIGN KEY (ID\_Товара) REFERENCES Товар\_2(ID\_Товара)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

ALTER TABLE Фотография\_3

ADD CONSTRAINT загружает FOREIGN KEY (ID\_Пользователя) REFERENCES Пользователь\_5(ID\_Пользователя)

ON DELETE NO ACTION

ON UPDATE NO ACTION

go

CREATE TRIGGER tD\_Заказ\_6 ON Заказ\_6 FOR DELETE AS

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* DELETE trigger on Заказ\_6 \*/

BEGIN

DECLARE @errno int,

@errmsg varchar(255)

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Заказ\_6 содержит позицию Заказ\_Товар\_7 on parent delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00028f1d", PARENT\_OWNER="", PARENT\_TABLE="Заказ\_6"

CHILD\_OWNER="", CHILD\_TABLE="Заказ\_Товар\_7"

P2C\_VERB\_PHRASE="содержит позицию", C2P\_VERB\_PHRASE="входит в заказ",

FK\_CONSTRAINT="содержит\_позицию", FK\_COLUMNS="ID\_Заказа" \*/

IF EXISTS (

SELECT \* FROM deleted,Заказ\_Товар\_7

WHERE

/\* %JoinFKPK(Заказ\_Товар\_7,deleted," = "," AND") \*/

Заказ\_Товар\_7.ID\_Заказа = deleted.ID\_Заказа

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Заказ\_6 because Заказ\_Товар\_7 exists.'

GOTO ERROR

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Пользователь\_5 может оформлять Заказ\_6 on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Пользователь\_5"

CHILD\_OWNER="", CHILD\_TABLE="Заказ\_6"

P2C\_VERB\_PHRASE="может оформлять", C2P\_VERB\_PHRASE="может быть оформлен клиентом",

FK\_CONSTRAINT="может\_оформлять", FK\_COLUMNS="ID\_Пользователя" \*/

IF EXISTS (SELECT \* FROM deleted,Пользователь\_5

WHERE

/\* %JoinFKPK(deleted,Пользователь\_5," = "," AND") \*/

deleted.ID\_Пользователя = Пользователь\_5.ID\_Пользователя AND

NOT EXISTS (

SELECT \* FROM Заказ\_6

WHERE

/\* %JoinFKPK(Заказ\_6,Пользователь\_5," = "," AND") \*/

Заказ\_6.ID\_Пользователя = Пользователь\_5.ID\_Пользователя

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Заказ\_6 because Пользователь\_5 exists.'

GOTO ERROR

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

RETURN

ERROR:

raiserror @errno @errmsg

rollback transaction

END

go

CREATE TRIGGER tU\_Заказ\_6 ON Заказ\_6 FOR UPDATE AS

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* UPDATE trigger on Заказ\_6 \*/

BEGIN

DECLARE @NUMROWS int,

@nullcnt int,

@validcnt int,

@insID\_Заказа integer,

@errno int,

@errmsg varchar(255)

SELECT @NUMROWS = @@rowcount

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Заказ\_6 содержит позицию Заказ\_Товар\_7 on parent update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="0002d5a7", PARENT\_OWNER="", PARENT\_TABLE="Заказ\_6"

CHILD\_OWNER="", CHILD\_TABLE="Заказ\_Товар\_7"

P2C\_VERB\_PHRASE="содержит позицию", C2P\_VERB\_PHRASE="входит в заказ",

FK\_CONSTRAINT="содержит\_позицию", FK\_COLUMNS="ID\_Заказа" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(ID\_Заказа)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Заказ\_Товар\_7

WHERE

/\* %JoinFKPK(Заказ\_Товар\_7,deleted," = "," AND") \*/

Заказ\_Товар\_7.ID\_Заказа = deleted.ID\_Заказа

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Заказ\_6 because Заказ\_Товар\_7 exists.'

GOTO ERROR

END

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Пользователь\_5 может оформлять Заказ\_6 on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Пользователь\_5"

CHILD\_OWNER="", CHILD\_TABLE="Заказ\_6"

P2C\_VERB\_PHRASE="может оформлять", C2P\_VERB\_PHRASE="может быть оформлен клиентом",

FK\_CONSTRAINT="может\_оформлять", FK\_COLUMNS="ID\_Пользователя" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(ID\_Пользователя)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Пользователь\_5

WHERE

/\* %JoinFKPK(inserted,Пользователь\_5) \*/

inserted.ID\_Пользователя = Пользователь\_5.ID\_Пользователя

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @NUMROWS

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Заказ\_6 because Пользователь\_5 does not exist.'

GOTO ERROR

END

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

RETURN

ERROR:

raiserror @errno @errmsg

rollback transaction

END

go

CREATE TRIGGER tD\_Заказ\_Товар\_7 ON Заказ\_Товар\_7 FOR DELETE AS

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* DELETE trigger on Заказ\_Товар\_7 \*/

BEGIN

DECLARE @errno int,

@errmsg varchar(255)

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Заказ\_6 содержит позицию Заказ\_Товар\_7 on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="0002aac2", PARENT\_OWNER="", PARENT\_TABLE="Заказ\_6"

CHILD\_OWNER="", CHILD\_TABLE="Заказ\_Товар\_7"

P2C\_VERB\_PHRASE="содержит позицию", C2P\_VERB\_PHRASE="входит в заказ",

FK\_CONSTRAINT="содержит\_позицию", FK\_COLUMNS="ID\_Заказа" \*/

IF EXISTS (SELECT \* FROM deleted,Заказ\_6

WHERE

/\* %JoinFKPK(deleted,Заказ\_6," = "," AND") \*/

deleted.ID\_Заказа = Заказ\_6.ID\_Заказа AND

NOT EXISTS (

SELECT \* FROM Заказ\_Товар\_7

WHERE

/\* %JoinFKPK(Заказ\_Товар\_7,Заказ\_6," = "," AND") \*/

Заказ\_Товар\_7.ID\_Заказа = Заказ\_6.ID\_Заказа

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Заказ\_Товар\_7 because Заказ\_6 exists.'

GOTO ERROR

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Товар\_2 может быть в заказе Заказ\_Товар\_7 on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Товар\_2"

CHILD\_OWNER="", CHILD\_TABLE="Заказ\_Товар\_7"

P2C\_VERB\_PHRASE="может быть в заказе", C2P\_VERB\_PHRASE="является товаром",

FK\_CONSTRAINT="может\_быть\_в\_заказе", FK\_COLUMNS="ID\_Товара" \*/

IF EXISTS (SELECT \* FROM deleted,Товар\_2

WHERE

/\* %JoinFKPK(deleted,Товар\_2," = "," AND") \*/

deleted.ID\_Товара = Товар\_2.ID\_Товара AND

NOT EXISTS (

SELECT \* FROM Заказ\_Товар\_7

WHERE

/\* %JoinFKPK(Заказ\_Товар\_7,Товар\_2," = "," AND") \*/

Заказ\_Товар\_7.ID\_Товара = Товар\_2.ID\_Товара

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Заказ\_Товар\_7 because Товар\_2 exists.'

GOTO ERROR

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

RETURN

ERROR:

raiserror @errno @errmsg

rollback transaction

END

go

CREATE TRIGGER tU\_Заказ\_Товар\_7 ON Заказ\_Товар\_7 FOR UPDATE AS

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* UPDATE trigger on Заказ\_Товар\_7 \*/

BEGIN

DECLARE @NUMROWS int,

@nullcnt int,

@validcnt int,

@insID\_Заказа integer,

@insID\_Товара integer,

@errno int,

@errmsg varchar(255)

SELECT @NUMROWS = @@rowcount

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Заказ\_6 содержит позицию Заказ\_Товар\_7 on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="0002e667", PARENT\_OWNER="", PARENT\_TABLE="Заказ\_6"

CHILD\_OWNER="", CHILD\_TABLE="Заказ\_Товар\_7"

P2C\_VERB\_PHRASE="содержит позицию", C2P\_VERB\_PHRASE="входит в заказ",

FK\_CONSTRAINT="содержит\_позицию", FK\_COLUMNS="ID\_Заказа" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(ID\_Заказа)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Заказ\_6

WHERE

/\* %JoinFKPK(inserted,Заказ\_6) \*/

inserted.ID\_Заказа = Заказ\_6.ID\_Заказа

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @NUMROWS

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Заказ\_Товар\_7 because Заказ\_6 does not exist.'

GOTO ERROR

END

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Товар\_2 может быть в заказе Заказ\_Товар\_7 on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Товар\_2"

CHILD\_OWNER="", CHILD\_TABLE="Заказ\_Товар\_7"

P2C\_VERB\_PHRASE="может быть в заказе", C2P\_VERB\_PHRASE="является товаром",

FK\_CONSTRAINT="может\_быть\_в\_заказе", FK\_COLUMNS="ID\_Товара" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(ID\_Товара)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Товар\_2

WHERE

/\* %JoinFKPK(inserted,Товар\_2) \*/

inserted.ID\_Товара = Товар\_2.ID\_Товара

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @NUMROWS

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Заказ\_Товар\_7 because Товар\_2 does not exist.'

GOTO ERROR

END

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

RETURN

ERROR:

raiserror @errno @errmsg

rollback transaction

END

go

CREATE TRIGGER tD\_Категория\_1 ON Категория\_1 FOR DELETE AS

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* DELETE trigger on Категория\_1 \*/

BEGIN

DECLARE @errno int,

@errmsg varchar(255)

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Категория\_1 содержит Товар\_2 on parent delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="000113a3", PARENT\_OWNER="", PARENT\_TABLE="Категория\_1"

CHILD\_OWNER="", CHILD\_TABLE="Товар\_2"

P2C\_VERB\_PHRASE="содержит", C2P\_VERB\_PHRASE="является товаром",

FK\_CONSTRAINT="содержит", FK\_COLUMNS="ID\_Категории" \*/

IF EXISTS (

SELECT \* FROM deleted,Товар\_2

WHERE

/\* %JoinFKPK(Товар\_2,deleted," = "," AND") \*/

Товар\_2.ID\_Категории = deleted.ID\_Категории

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Категория\_1 because Товар\_2 exists.'

GOTO ERROR

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

RETURN

ERROR:

raiserror @errno @errmsg

rollback transaction

END

go

CREATE TRIGGER tU\_Категория\_1 ON Категория\_1 FOR UPDATE AS

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* UPDATE trigger on Категория\_1 \*/

BEGIN

DECLARE @NUMROWS int,

@nullcnt int,

@validcnt int,

@insID\_Категории integer,

@errno int,

@errmsg varchar(255)

SELECT @NUMROWS = @@rowcount

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Категория\_1 содержит Товар\_2 on parent update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="000124d0", PARENT\_OWNER="", PARENT\_TABLE="Категория\_1"

CHILD\_OWNER="", CHILD\_TABLE="Товар\_2"

P2C\_VERB\_PHRASE="содержит", C2P\_VERB\_PHRASE="является товаром",

FK\_CONSTRAINT="содержит", FK\_COLUMNS="ID\_Категории" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(ID\_Категории)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Товар\_2

WHERE

/\* %JoinFKPK(Товар\_2,deleted," = "," AND") \*/

Товар\_2.ID\_Категории = deleted.ID\_Категории

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Категория\_1 because Товар\_2 exists.'

GOTO ERROR

END

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

RETURN

ERROR:

raiserror @errno @errmsg

rollback transaction

END

go

CREATE TRIGGER tD\_Корзина\_4 ON Корзина\_4 FOR DELETE AS

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* DELETE trigger on Корзина\_4 \*/

BEGIN

DECLARE @errno int,

@errmsg varchar(255)

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Товар\_2 может быть в корзине Корзина\_4 on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="0002cf25", PARENT\_OWNER="", PARENT\_TABLE="Товар\_2"

CHILD\_OWNER="", CHILD\_TABLE="Корзина\_4"

P2C\_VERB\_PHRASE="может быть в корзине", C2P\_VERB\_PHRASE="является товаром",

FK\_CONSTRAINT="может\_быть\_в\_корзине", FK\_COLUMNS="ID\_Товара" \*/

IF EXISTS (SELECT \* FROM deleted,Товар\_2

WHERE

/\* %JoinFKPK(deleted,Товар\_2," = "," AND") \*/

deleted.ID\_Товара = Товар\_2.ID\_Товара AND

NOT EXISTS (

SELECT \* FROM Корзина\_4

WHERE

/\* %JoinFKPK(Корзина\_4,Товар\_2," = "," AND") \*/

Корзина\_4.ID\_Товара = Товар\_2.ID\_Товара

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Корзина\_4 because Товар\_2 exists.'

GOTO ERROR

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Пользователь\_5 имеет корзину Корзина\_4 on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Пользователь\_5"

CHILD\_OWNER="", CHILD\_TABLE="Корзина\_4"

P2C\_VERB\_PHRASE="имеет корзину", C2P\_VERB\_PHRASE="принадлежит клиенту",

FK\_CONSTRAINT="имеет\_корзину", FK\_COLUMNS="ID\_Пользователя" \*/

IF EXISTS (SELECT \* FROM deleted,Пользователь\_5

WHERE

/\* %JoinFKPK(deleted,Пользователь\_5," = "," AND") \*/

deleted.ID\_Пользователя = Пользователь\_5.ID\_Пользователя AND

NOT EXISTS (

SELECT \* FROM Корзина\_4

WHERE

/\* %JoinFKPK(Корзина\_4,Пользователь\_5," = "," AND") \*/

Корзина\_4.ID\_Пользователя = Пользователь\_5.ID\_Пользователя

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Корзина\_4 because Пользователь\_5 exists.'

GOTO ERROR

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

RETURN

ERROR:

raiserror @errno @errmsg

rollback transaction

END

go

CREATE TRIGGER tU\_Корзина\_4 ON Корзина\_4 FOR UPDATE AS

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* UPDATE trigger on Корзина\_4 \*/

BEGIN

DECLARE @NUMROWS int,

@nullcnt int,

@validcnt int,

@insID\_Корзины integer,

@insID\_Пользователя integer,

@insID\_Товара integer,

@errno int,

@errmsg varchar(255)

SELECT @NUMROWS = @@rowcount

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Товар\_2 может быть в корзине Корзина\_4 on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="0002f79d", PARENT\_OWNER="", PARENT\_TABLE="Товар\_2"

CHILD\_OWNER="", CHILD\_TABLE="Корзина\_4"

P2C\_VERB\_PHRASE="может быть в корзине", C2P\_VERB\_PHRASE="является товаром",

FK\_CONSTRAINT="может\_быть\_в\_корзине", FK\_COLUMNS="ID\_Товара" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(ID\_Товара)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Товар\_2

WHERE

/\* %JoinFKPK(inserted,Товар\_2) \*/

inserted.ID\_Товара = Товар\_2.ID\_Товара

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @NUMROWS

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Корзина\_4 because Товар\_2 does not exist.'

GOTO ERROR

END

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Пользователь\_5 имеет корзину Корзина\_4 on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Пользователь\_5"

CHILD\_OWNER="", CHILD\_TABLE="Корзина\_4"

P2C\_VERB\_PHRASE="имеет корзину", C2P\_VERB\_PHRASE="принадлежит клиенту",

FK\_CONSTRAINT="имеет\_корзину", FK\_COLUMNS="ID\_Пользователя" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(ID\_Пользователя)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Пользователь\_5

WHERE

/\* %JoinFKPK(inserted,Пользователь\_5) \*/

inserted.ID\_Пользователя = Пользователь\_5.ID\_Пользователя

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @NUMROWS

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Корзина\_4 because Пользователь\_5 does not exist.'

GOTO ERROR

END

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

RETURN

ERROR:

raiserror @errno @errmsg

rollback transaction

END

go

CREATE TRIGGER tD\_Пользователь\_5 ON Пользователь\_5 FOR DELETE AS

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* DELETE trigger on Пользователь\_5 \*/

BEGIN

DECLARE @errno int,

@errmsg varchar(255)

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Пользователь\_5 управляет Товар\_2 on parent delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00048cbe", PARENT\_OWNER="", PARENT\_TABLE="Пользователь\_5"

CHILD\_OWNER="", CHILD\_TABLE="Товар\_2"

P2C\_VERB\_PHRASE="управляет", C2P\_VERB\_PHRASE="управляется администратором",

FK\_CONSTRAINT="управляет", FK\_COLUMNS="ID\_Пользователя" \*/

IF EXISTS (

SELECT \* FROM deleted,Товар\_2

WHERE

/\* %JoinFKPK(Товар\_2,deleted," = "," AND") \*/

Товар\_2.ID\_Пользователя = deleted.ID\_Пользователя

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Пользователь\_5 because Товар\_2 exists.'

GOTO ERROR

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Пользователь\_5 имеет корзину Корзина\_4 on parent delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Пользователь\_5"

CHILD\_OWNER="", CHILD\_TABLE="Корзина\_4"

P2C\_VERB\_PHRASE="имеет корзину", C2P\_VERB\_PHRASE="принадлежит клиенту",

FK\_CONSTRAINT="имеет\_корзину", FK\_COLUMNS="ID\_Пользователя" \*/

IF EXISTS (

SELECT \* FROM deleted,Корзина\_4

WHERE

/\* %JoinFKPK(Корзина\_4,deleted," = "," AND") \*/

Корзина\_4.ID\_Пользователя = deleted.ID\_Пользователя

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Пользователь\_5 because Корзина\_4 exists.'

GOTO ERROR

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Пользователь\_5 может оформлять Заказ\_6 on parent delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Пользователь\_5"

CHILD\_OWNER="", CHILD\_TABLE="Заказ\_6"

P2C\_VERB\_PHRASE="может оформлять", C2P\_VERB\_PHRASE="может быть оформлен клиентом",

FK\_CONSTRAINT="может\_оформлять", FK\_COLUMNS="ID\_Пользователя" \*/

IF EXISTS (

SELECT \* FROM deleted,Заказ\_6

WHERE

/\* %JoinFKPK(Заказ\_6,deleted," = "," AND") \*/

Заказ\_6.ID\_Пользователя = deleted.ID\_Пользователя

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Пользователь\_5 because Заказ\_6 exists.'

GOTO ERROR

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Пользователь\_5 загружает Фотография\_3 on parent delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Пользователь\_5"

CHILD\_OWNER="", CHILD\_TABLE="Фотография\_3"

P2C\_VERB\_PHRASE="загружает", C2P\_VERB\_PHRASE="загружается администратором",

FK\_CONSTRAINT="загружает", FK\_COLUMNS="ID\_Пользователя" \*/

IF EXISTS (

SELECT \* FROM deleted,Фотография\_3

WHERE

/\* %JoinFKPK(Фотография\_3,deleted," = "," AND") \*/

Фотография\_3.ID\_Пользователя = deleted.ID\_Пользователя

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Пользователь\_5 because Фотография\_3 exists.'

GOTO ERROR

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

RETURN

ERROR:

raiserror @errno @errmsg

rollback transaction

END

go

CREATE TRIGGER tU\_Пользователь\_5 ON Пользователь\_5 FOR UPDATE AS

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* UPDATE trigger on Пользователь\_5 \*/

BEGIN

DECLARE @NUMROWS int,

@nullcnt int,

@validcnt int,

@insID\_Пользователя integer,

@errno int,

@errmsg varchar(255)

SELECT @NUMROWS = @@rowcount

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Пользователь\_5 управляет Товар\_2 on parent update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="0005129a", PARENT\_OWNER="", PARENT\_TABLE="Пользователь\_5"

CHILD\_OWNER="", CHILD\_TABLE="Товар\_2"

P2C\_VERB\_PHRASE="управляет", C2P\_VERB\_PHRASE="управляется администратором",

FK\_CONSTRAINT="управляет", FK\_COLUMNS="ID\_Пользователя" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(ID\_Пользователя)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Товар\_2

WHERE

/\* %JoinFKPK(Товар\_2,deleted," = "," AND") \*/

Товар\_2.ID\_Пользователя = deleted.ID\_Пользователя

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Пользователь\_5 because Товар\_2 exists.'

GOTO ERROR

END

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Пользователь\_5 имеет корзину Корзина\_4 on parent update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Пользователь\_5"

CHILD\_OWNER="", CHILD\_TABLE="Корзина\_4"

P2C\_VERB\_PHRASE="имеет корзину", C2P\_VERB\_PHRASE="принадлежит клиенту",

FK\_CONSTRAINT="имеет\_корзину", FK\_COLUMNS="ID\_Пользователя" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(ID\_Пользователя)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Корзина\_4

WHERE

/\* %JoinFKPK(Корзина\_4,deleted," = "," AND") \*/

Корзина\_4.ID\_Пользователя = deleted.ID\_Пользователя

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Пользователь\_5 because Корзина\_4 exists.'

GOTO ERROR

END

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Пользователь\_5 может оформлять Заказ\_6 on parent update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Пользователь\_5"

CHILD\_OWNER="", CHILD\_TABLE="Заказ\_6"

P2C\_VERB\_PHRASE="может оформлять", C2P\_VERB\_PHRASE="может быть оформлен клиентом",

FK\_CONSTRAINT="может\_оформлять", FK\_COLUMNS="ID\_Пользователя" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(ID\_Пользователя)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Заказ\_6

WHERE

/\* %JoinFKPK(Заказ\_6,deleted," = "," AND") \*/

Заказ\_6.ID\_Пользователя = deleted.ID\_Пользователя

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Пользователь\_5 because Заказ\_6 exists.'

GOTO ERROR

END

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Пользователь\_5 загружает Фотография\_3 on parent update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Пользователь\_5"

CHILD\_OWNER="", CHILD\_TABLE="Фотография\_3"

P2C\_VERB\_PHRASE="загружает", C2P\_VERB\_PHRASE="загружается администратором",

FK\_CONSTRAINT="загружает", FK\_COLUMNS="ID\_Пользователя" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(ID\_Пользователя)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Фотография\_3

WHERE

/\* %JoinFKPK(Фотография\_3,deleted," = "," AND") \*/

Фотография\_3.ID\_Пользователя = deleted.ID\_Пользователя

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Пользователь\_5 because Фотография\_3 exists.'

GOTO ERROR

END

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

RETURN

ERROR:

raiserror @errno @errmsg

rollback transaction

END

go

CREATE TRIGGER tD\_Товар\_2 ON Товар\_2 FOR DELETE AS

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* DELETE trigger on Товар\_2 \*/

BEGIN

DECLARE @errno int,

@errmsg varchar(255)

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Товар\_2 имеет Фотография\_3 on parent delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="0005f4c6", PARENT\_OWNER="", PARENT\_TABLE="Товар\_2"

CHILD\_OWNER="", CHILD\_TABLE="Фотография\_3"

P2C\_VERB\_PHRASE="имеет", C2P\_VERB\_PHRASE="принадлежит товару",

FK\_CONSTRAINT="имеет", FK\_COLUMNS="ID\_Товара" \*/

IF EXISTS (

SELECT \* FROM deleted,Фотография\_3

WHERE

/\* %JoinFKPK(Фотография\_3,deleted," = "," AND") \*/

Фотография\_3.ID\_Товара = deleted.ID\_Товара

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Товар\_2 because Фотография\_3 exists.'

GOTO ERROR

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Товар\_2 может быть в корзине Корзина\_4 on parent delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Товар\_2"

CHILD\_OWNER="", CHILD\_TABLE="Корзина\_4"

P2C\_VERB\_PHRASE="может быть в корзине", C2P\_VERB\_PHRASE="является товаром",

FK\_CONSTRAINT="может\_быть\_в\_корзине", FK\_COLUMNS="ID\_Товара" \*/

IF EXISTS (

SELECT \* FROM deleted,Корзина\_4

WHERE

/\* %JoinFKPK(Корзина\_4,deleted," = "," AND") \*/

Корзина\_4.ID\_Товара = deleted.ID\_Товара

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Товар\_2 because Корзина\_4 exists.'

GOTO ERROR

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Товар\_2 может быть в заказе Заказ\_Товар\_7 on parent delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Товар\_2"

CHILD\_OWNER="", CHILD\_TABLE="Заказ\_Товар\_7"

P2C\_VERB\_PHRASE="может быть в заказе", C2P\_VERB\_PHRASE="является товаром",

FK\_CONSTRAINT="может\_быть\_в\_заказе", FK\_COLUMNS="ID\_Товара" \*/

IF EXISTS (

SELECT \* FROM deleted,Заказ\_Товар\_7

WHERE

/\* %JoinFKPK(Заказ\_Товар\_7,deleted," = "," AND") \*/

Заказ\_Товар\_7.ID\_Товара = deleted.ID\_Товара

)

BEGIN

SELECT @errno = 30001,

@errmsg = 'Cannot delete Товар\_2 because Заказ\_Товар\_7 exists.'

GOTO ERROR

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Категория\_1 содержит Товар\_2 on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Категория\_1"

CHILD\_OWNER="", CHILD\_TABLE="Товар\_2"

P2C\_VERB\_PHRASE="содержит", C2P\_VERB\_PHRASE="является товаром",

FK\_CONSTRAINT="содержит", FK\_COLUMNS="ID\_Категории" \*/

IF EXISTS (SELECT \* FROM deleted,Категория\_1

WHERE

/\* %JoinFKPK(deleted,Категория\_1," = "," AND") \*/

deleted.ID\_Категории = Категория\_1.ID\_Категории AND

NOT EXISTS (

SELECT \* FROM Товар\_2

WHERE

/\* %JoinFKPK(Товар\_2,Категория\_1," = "," AND") \*/

Товар\_2.ID\_Категории = Категория\_1.ID\_Категории

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Товар\_2 because Категория\_1 exists.'

GOTO ERROR

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Пользователь\_5 управляет Товар\_2 on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Пользователь\_5"

CHILD\_OWNER="", CHILD\_TABLE="Товар\_2"

P2C\_VERB\_PHRASE="управляет", C2P\_VERB\_PHRASE="управляется администратором",

FK\_CONSTRAINT="управляет", FK\_COLUMNS="ID\_Пользователя" \*/

IF EXISTS (SELECT \* FROM deleted,Пользователь\_5

WHERE

/\* %JoinFKPK(deleted,Пользователь\_5," = "," AND") \*/

deleted.ID\_Пользователя = Пользователь\_5.ID\_Пользователя AND

NOT EXISTS (

SELECT \* FROM Товар\_2

WHERE

/\* %JoinFKPK(Товар\_2,Пользователь\_5," = "," AND") \*/

Товар\_2.ID\_Пользователя = Пользователь\_5.ID\_Пользователя

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Товар\_2 because Пользователь\_5 exists.'

GOTO ERROR

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

RETURN

ERROR:

raiserror @errno @errmsg

rollback transaction

END

go

CREATE TRIGGER tU\_Товар\_2 ON Товар\_2 FOR UPDATE AS

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* UPDATE trigger on Товар\_2 \*/

BEGIN

DECLARE @NUMROWS int,

@nullcnt int,

@validcnt int,

@insID\_Товара integer,

@errno int,

@errmsg varchar(255)

SELECT @NUMROWS = @@rowcount

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Товар\_2 имеет Фотография\_3 on parent update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="0006b4e9", PARENT\_OWNER="", PARENT\_TABLE="Товар\_2"

CHILD\_OWNER="", CHILD\_TABLE="Фотография\_3"

P2C\_VERB\_PHRASE="имеет", C2P\_VERB\_PHRASE="принадлежит товару",

FK\_CONSTRAINT="имеет", FK\_COLUMNS="ID\_Товара" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(ID\_Товара)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Фотография\_3

WHERE

/\* %JoinFKPK(Фотография\_3,deleted," = "," AND") \*/

Фотография\_3.ID\_Товара = deleted.ID\_Товара

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Товар\_2 because Фотография\_3 exists.'

GOTO ERROR

END

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Товар\_2 может быть в корзине Корзина\_4 on parent update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Товар\_2"

CHILD\_OWNER="", CHILD\_TABLE="Корзина\_4"

P2C\_VERB\_PHRASE="может быть в корзине", C2P\_VERB\_PHRASE="является товаром",

FK\_CONSTRAINT="может\_быть\_в\_корзине", FK\_COLUMNS="ID\_Товара" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(ID\_Товара)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Корзина\_4

WHERE

/\* %JoinFKPK(Корзина\_4,deleted," = "," AND") \*/

Корзина\_4.ID\_Товара = deleted.ID\_Товара

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Товар\_2 because Корзина\_4 exists.'

GOTO ERROR

END

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Товар\_2 может быть в заказе Заказ\_Товар\_7 on parent update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Товар\_2"

CHILD\_OWNER="", CHILD\_TABLE="Заказ\_Товар\_7"

P2C\_VERB\_PHRASE="может быть в заказе", C2P\_VERB\_PHRASE="является товаром",

FK\_CONSTRAINT="может\_быть\_в\_заказе", FK\_COLUMNS="ID\_Товара" \*/

IF

/\* %ParentPK(" OR",UPDATE) \*/

UPDATE(ID\_Товара)

BEGIN

IF EXISTS (

SELECT \* FROM deleted,Заказ\_Товар\_7

WHERE

/\* %JoinFKPK(Заказ\_Товар\_7,deleted," = "," AND") \*/

Заказ\_Товар\_7.ID\_Товара = deleted.ID\_Товара

)

BEGIN

SELECT @errno = 30005,

@errmsg = 'Cannot update Товар\_2 because Заказ\_Товар\_7 exists.'

GOTO ERROR

END

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Категория\_1 содержит Товар\_2 on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Категория\_1"

CHILD\_OWNER="", CHILD\_TABLE="Товар\_2"

P2C\_VERB\_PHRASE="содержит", C2P\_VERB\_PHRASE="является товаром",

FK\_CONSTRAINT="содержит", FK\_COLUMNS="ID\_Категории" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(ID\_Категории)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Категория\_1

WHERE

/\* %JoinFKPK(inserted,Категория\_1) \*/

inserted.ID\_Категории = Категория\_1.ID\_Категории

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

IF @validcnt + @nullcnt != @NUMROWS

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Товар\_2 because Категория\_1 does not exist.'

GOTO ERROR

END

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Пользователь\_5 управляет Товар\_2 on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Пользователь\_5"

CHILD\_OWNER="", CHILD\_TABLE="Товар\_2"

P2C\_VERB\_PHRASE="управляет", C2P\_VERB\_PHRASE="управляется администратором",

FK\_CONSTRAINT="управляет", FK\_COLUMNS="ID\_Пользователя" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(ID\_Пользователя)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Пользователь\_5

WHERE

/\* %JoinFKPK(inserted,Пользователь\_5) \*/

inserted.ID\_Пользователя = Пользователь\_5.ID\_Пользователя

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

select @nullcnt = count(\*) from inserted where

inserted.ID\_Пользователя IS NULL

IF @validcnt + @nullcnt != @NUMROWS

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Товар\_2 because Пользователь\_5 does not exist.'

GOTO ERROR

END

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

RETURN

ERROR:

raiserror @errno @errmsg

rollback transaction

END

go

CREATE TRIGGER tD\_Фотография\_3 ON Фотография\_3 FOR DELETE AS

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* DELETE trigger on Фотография\_3 \*/

BEGIN

DECLARE @errno int,

@errmsg varchar(255)

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Товар\_2 имеет Фотография\_3 on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="0002c4c0", PARENT\_OWNER="", PARENT\_TABLE="Товар\_2"

CHILD\_OWNER="", CHILD\_TABLE="Фотография\_3"

P2C\_VERB\_PHRASE="имеет", C2P\_VERB\_PHRASE="принадлежит товару",

FK\_CONSTRAINT="имеет", FK\_COLUMNS="ID\_Товара" \*/

IF EXISTS (SELECT \* FROM deleted,Товар\_2

WHERE

/\* %JoinFKPK(deleted,Товар\_2," = "," AND") \*/

deleted.ID\_Товара = Товар\_2.ID\_Товара AND

NOT EXISTS (

SELECT \* FROM Фотография\_3

WHERE

/\* %JoinFKPK(Фотография\_3,Товар\_2," = "," AND") \*/

Фотография\_3.ID\_Товара = Товар\_2.ID\_Товара

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Фотография\_3 because Товар\_2 exists.'

GOTO ERROR

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Пользователь\_5 загружает Фотография\_3 on child delete no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Пользователь\_5"

CHILD\_OWNER="", CHILD\_TABLE="Фотография\_3"

P2C\_VERB\_PHRASE="загружает", C2P\_VERB\_PHRASE="загружается администратором",

FK\_CONSTRAINT="загружает", FK\_COLUMNS="ID\_Пользователя" \*/

IF EXISTS (SELECT \* FROM deleted,Пользователь\_5

WHERE

/\* %JoinFKPK(deleted,Пользователь\_5," = "," AND") \*/

deleted.ID\_Пользователя = Пользователь\_5.ID\_Пользователя AND

NOT EXISTS (

SELECT \* FROM Фотография\_3

WHERE

/\* %JoinFKPK(Фотография\_3,Пользователь\_5," = "," AND") \*/

Фотография\_3.ID\_Пользователя = Пользователь\_5.ID\_Пользователя

)

)

BEGIN

SELECT @errno = 30010,

@errmsg = 'Cannot delete last Фотография\_3 because Пользователь\_5 exists.'

GOTO ERROR

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

RETURN

ERROR:

raiserror @errno @errmsg

rollback transaction

END

go

CREATE TRIGGER tU\_Фотография\_3 ON Фотография\_3 FOR UPDATE AS

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* UPDATE trigger on Фотография\_3 \*/

BEGIN

DECLARE @NUMROWS int,

@nullcnt int,

@validcnt int,

@insID\_Фотографии integer,

@errno int,

@errmsg varchar(255)

SELECT @NUMROWS = @@rowcount

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Товар\_2 имеет Фотография\_3 on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00032df1", PARENT\_OWNER="", PARENT\_TABLE="Товар\_2"

CHILD\_OWNER="", CHILD\_TABLE="Фотография\_3"

P2C\_VERB\_PHRASE="имеет", C2P\_VERB\_PHRASE="принадлежит товару",

FK\_CONSTRAINT="имеет", FK\_COLUMNS="ID\_Товара" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(ID\_Товара)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Товар\_2

WHERE

/\* %JoinFKPK(inserted,Товар\_2) \*/

inserted.ID\_Товара = Товар\_2.ID\_Товара

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

select @nullcnt = count(\*) from inserted where

inserted.ID\_Товара IS NULL

IF @validcnt + @nullcnt != @NUMROWS

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Фотография\_3 because Товар\_2 does not exist.'

GOTO ERROR

END

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

/\* Пользователь\_5 загружает Фотография\_3 on child update no action \*/

/\* ERWIN\_RELATION:CHECKSUM="00000000", PARENT\_OWNER="", PARENT\_TABLE="Пользователь\_5"

CHILD\_OWNER="", CHILD\_TABLE="Фотография\_3"

P2C\_VERB\_PHRASE="загружает", C2P\_VERB\_PHRASE="загружается администратором",

FK\_CONSTRAINT="загружает", FK\_COLUMNS="ID\_Пользователя" \*/

IF

/\* %ChildFK(" OR",UPDATE) \*/

UPDATE(ID\_Пользователя)

BEGIN

SELECT @nullcnt = 0

SELECT @validcnt = count(\*)

FROM inserted,Пользователь\_5

WHERE

/\* %JoinFKPK(inserted,Пользователь\_5) \*/

inserted.ID\_Пользователя = Пользователь\_5.ID\_Пользователя

/\* %NotnullFK(inserted," IS NULL","select @nullcnt = count(\*) from inserted where"," AND") \*/

select @nullcnt = count(\*) from inserted where

inserted.ID\_Пользователя IS NULL

IF @validcnt + @nullcnt != @NUMROWS

BEGIN

SELECT @errno = 30007,

@errmsg = 'Cannot update Фотография\_3 because Пользователь\_5 does not exist.'

GOTO ERROR

END

END

/\* ERwin Builtin четверг, 30 октября 2025 г. 15:12:37 \*/

RETURN

ERROR:

raiserror @errno @errmsg

rollback transaction

END

go

# ПРИЛОЖЕНИЕ В (справочное) Диаграмма модели IDEF-0

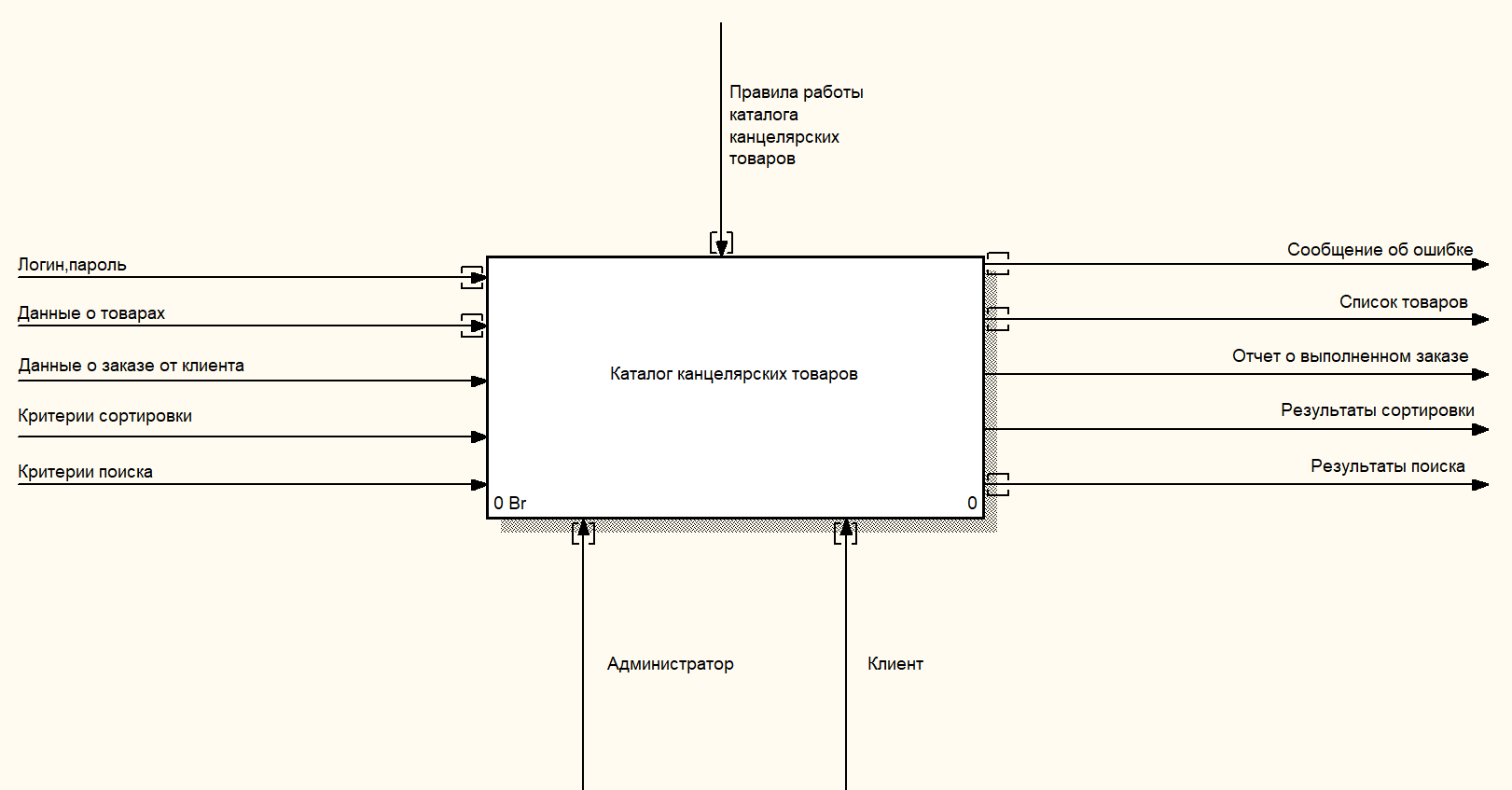


Рисунок В.1 – Диаграмма «IDEF0» Каталог канцелярских товаров

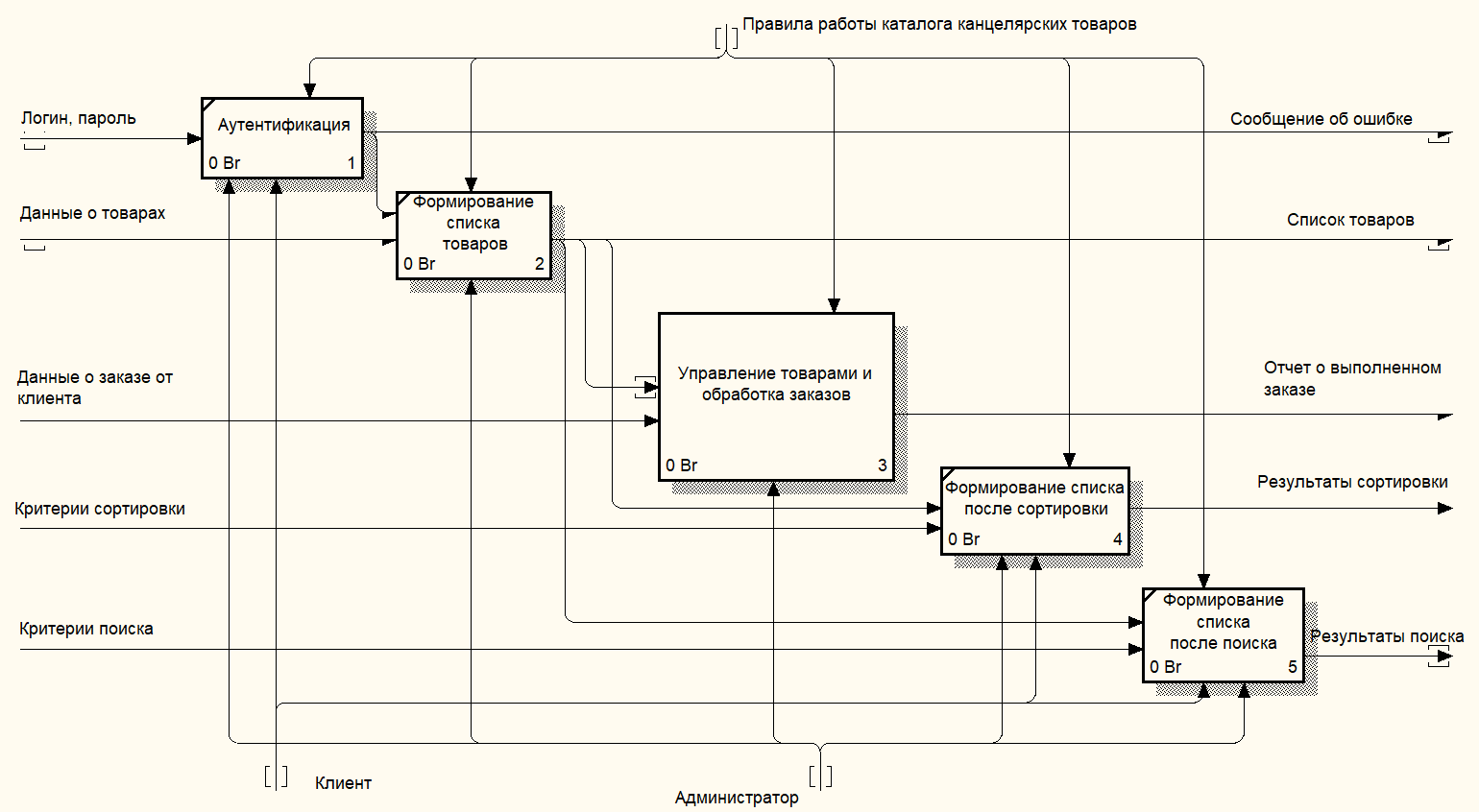


Рисунок В.2 – Диаграмма «IDEF0»   
Декомпозиция блока A0 «Каталог канцелярских товаров»

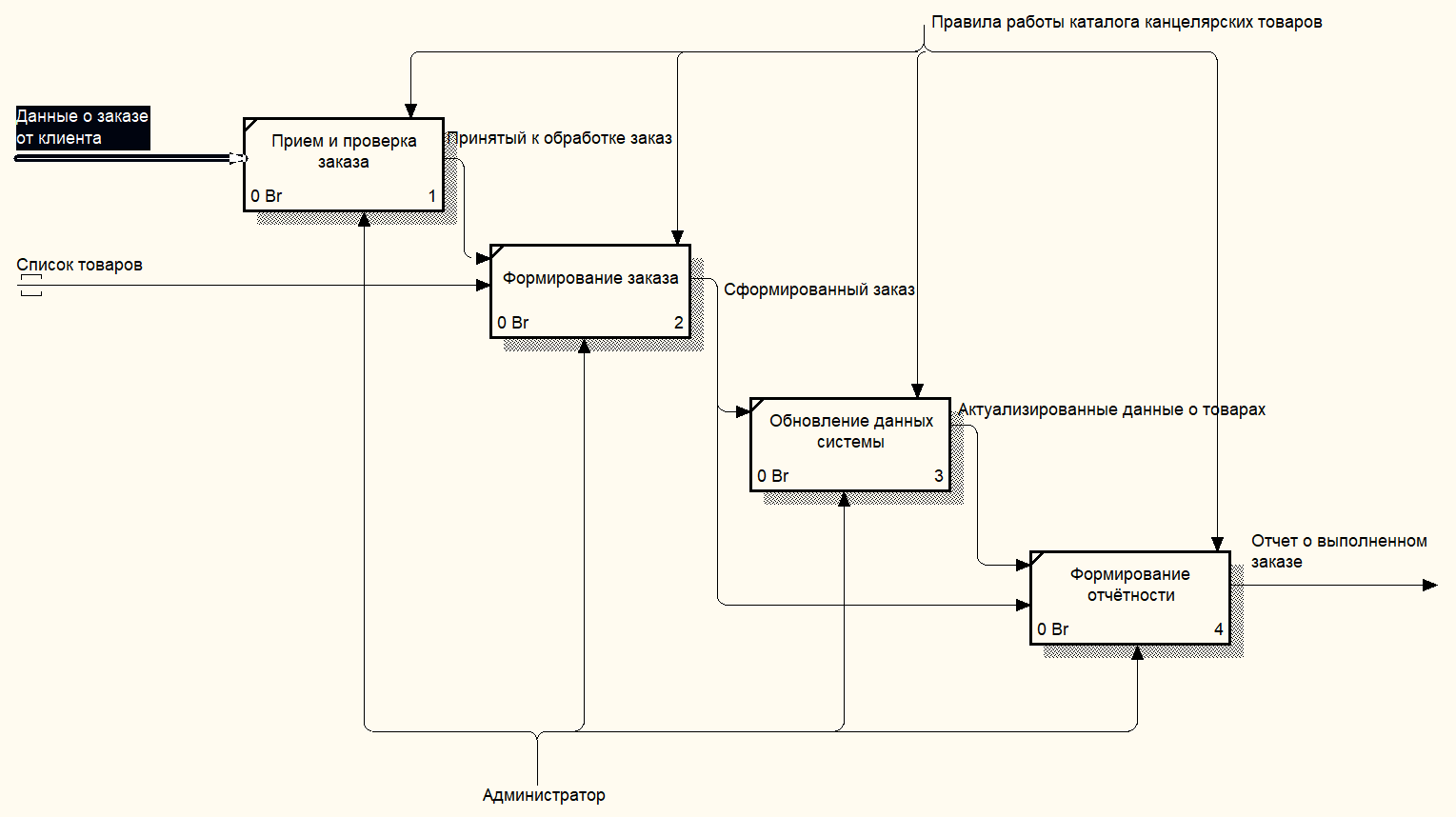


Рисунок В.3 – Диаграмма «IDEF0»   
Декомпозиция блока A3 «Управление товарами и обработка заказов»

# ПРИЛОЖЕНИЕ Г (справочное) Диаграмма модели DFD

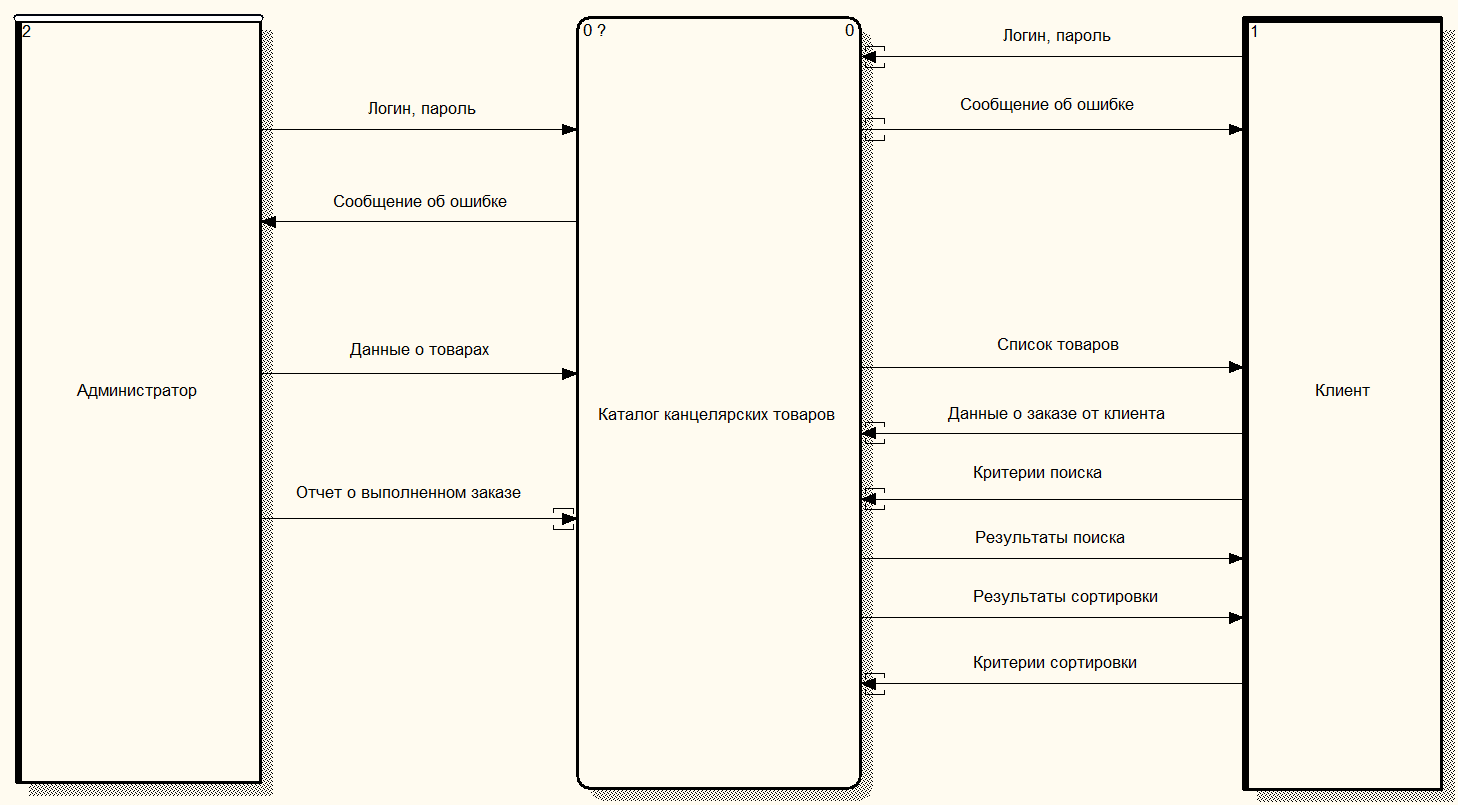


Рисунок Г.1 – Диаграмма «DFD» Каталог канцелярских товаров

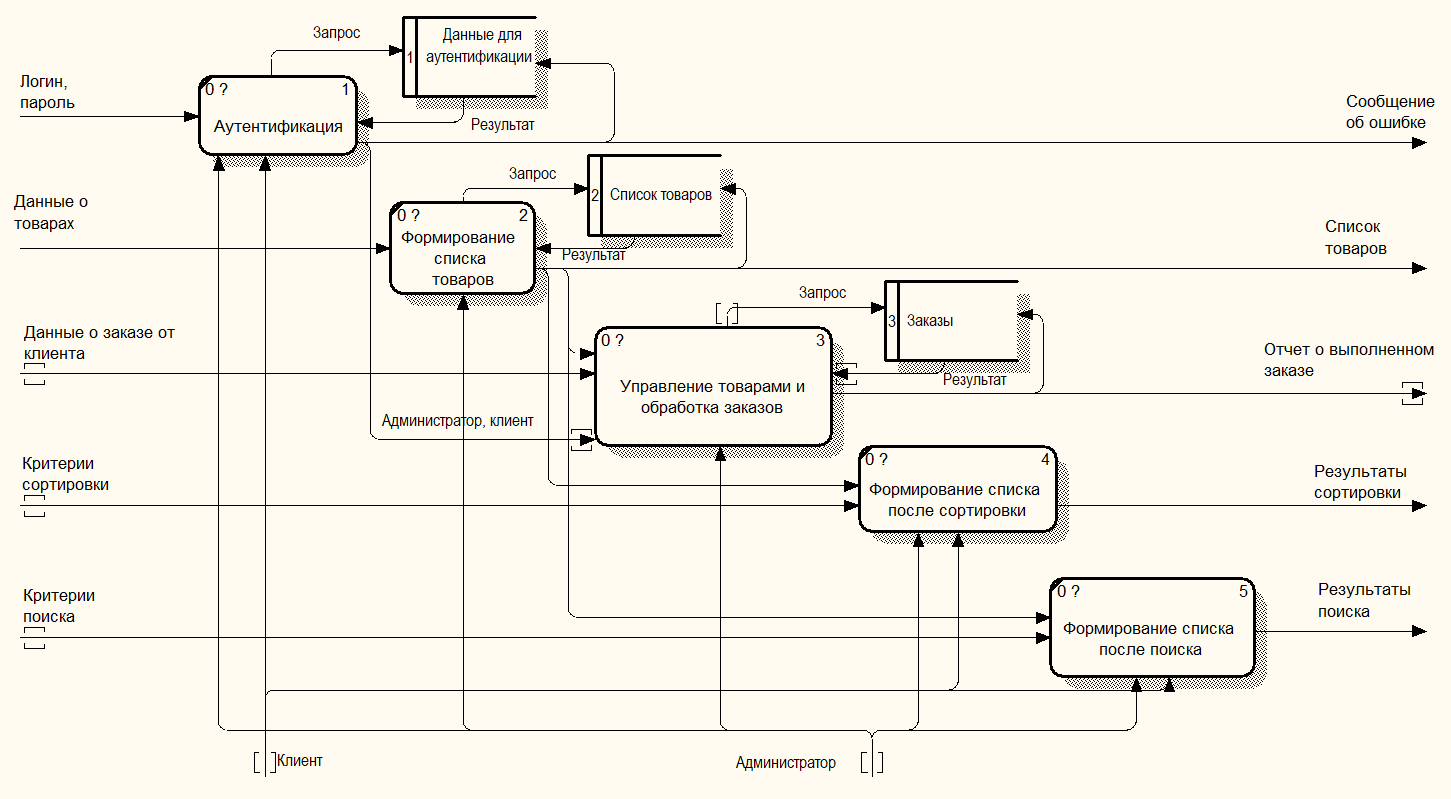


Рисунок Г.2 – Диаграмма «DFD»   
Декомпозиция «Каталог канцелярских товаров»

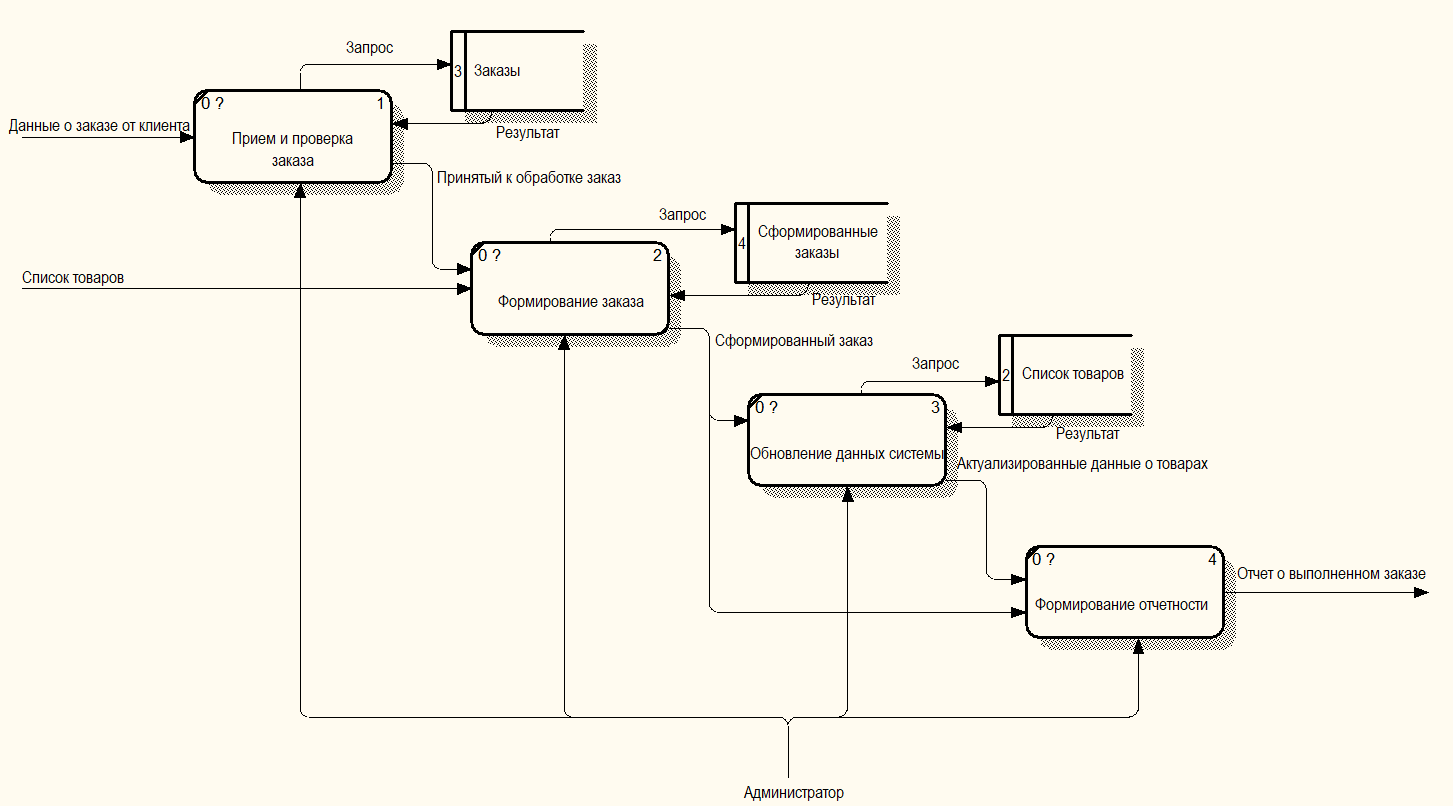
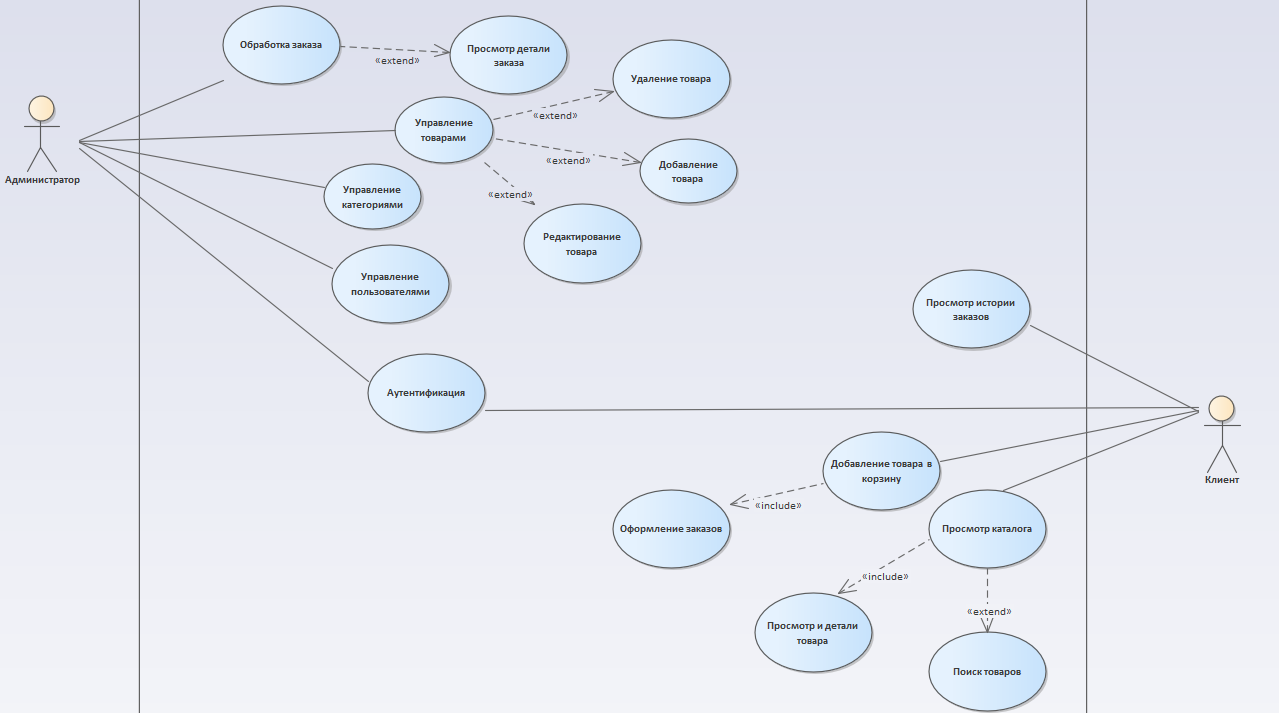
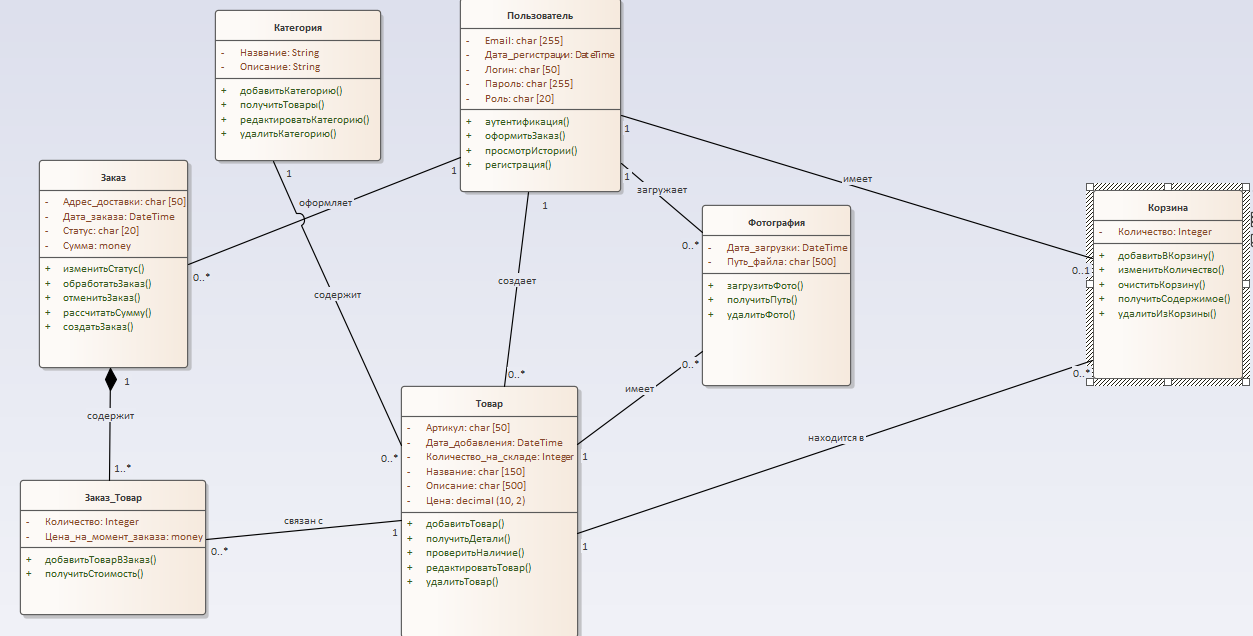


Рисунок Г.3 – Диаграмма «DFD»   
Декомпозиция «Управление товарами и обработка заказов»

# ПРИЛОЖЕНИЕ Д (справочное) Диаграмма вариантов использования



# ПРИЛОЖЕНИЕ Е (справочное) Диаграмма классов



# ПРИЛОЖЕНИЕ Ж (справочное) Диаграмма последовательности

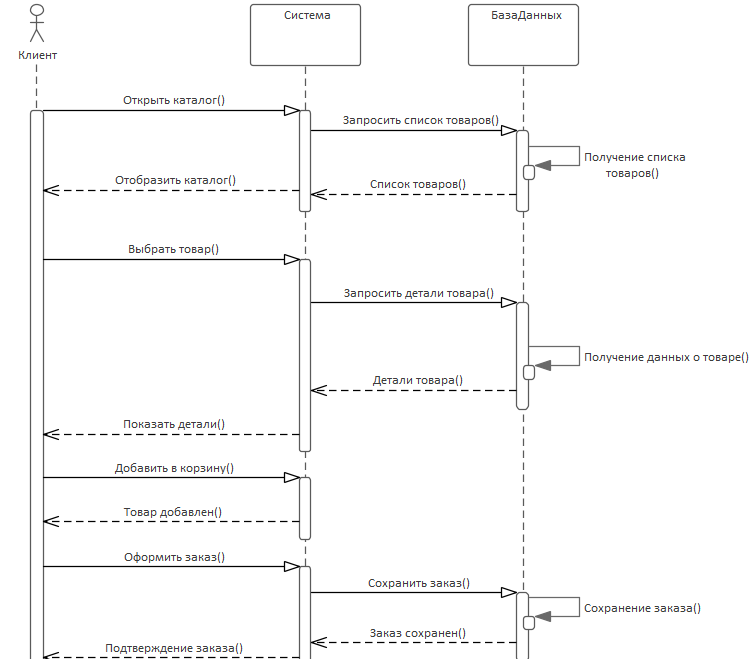
****

Рисунок Ж.1 – Просмотр каталога и оформление заказа

****

Рисунок Ж.2 – Управление товарами администратором

# ПРИЛОЖЕНИЕ И (справочное) Диаграмма деятельности

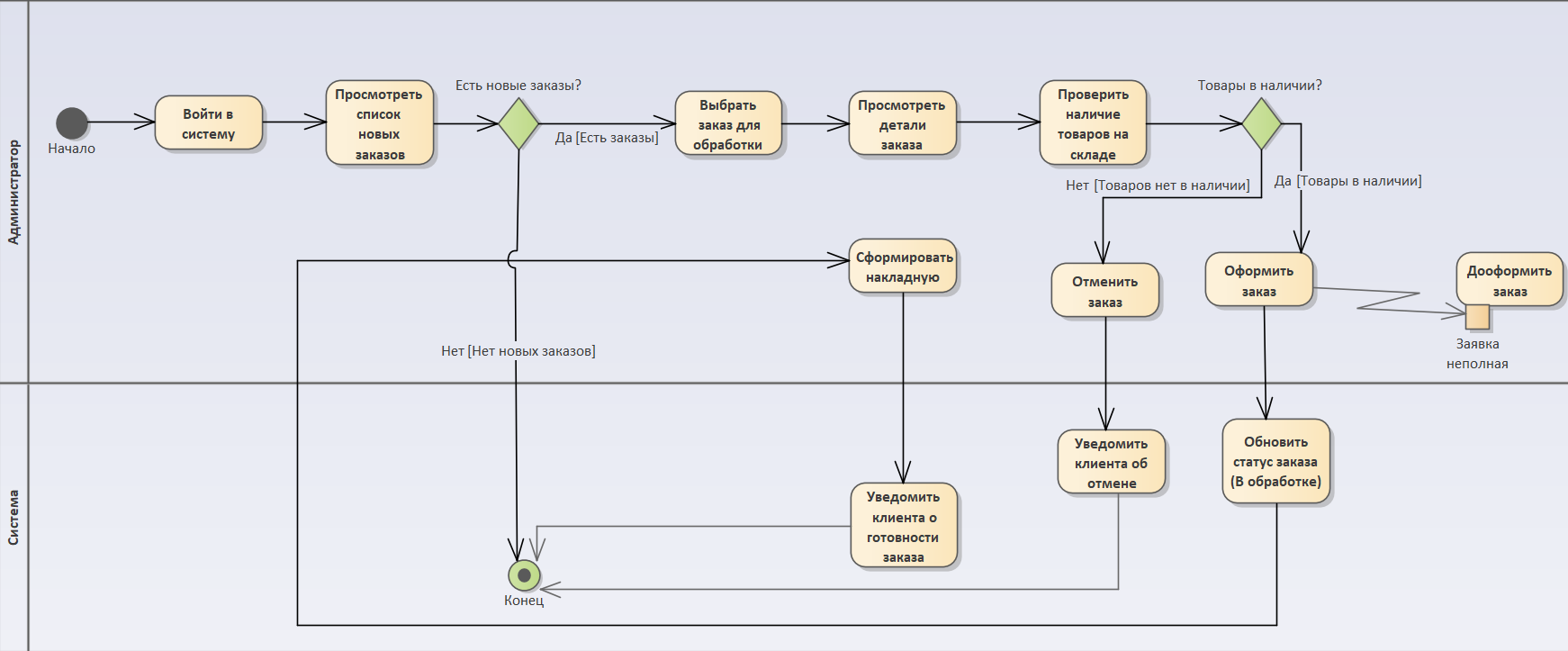


Рисунок И.1 – Обработка заказа администратором

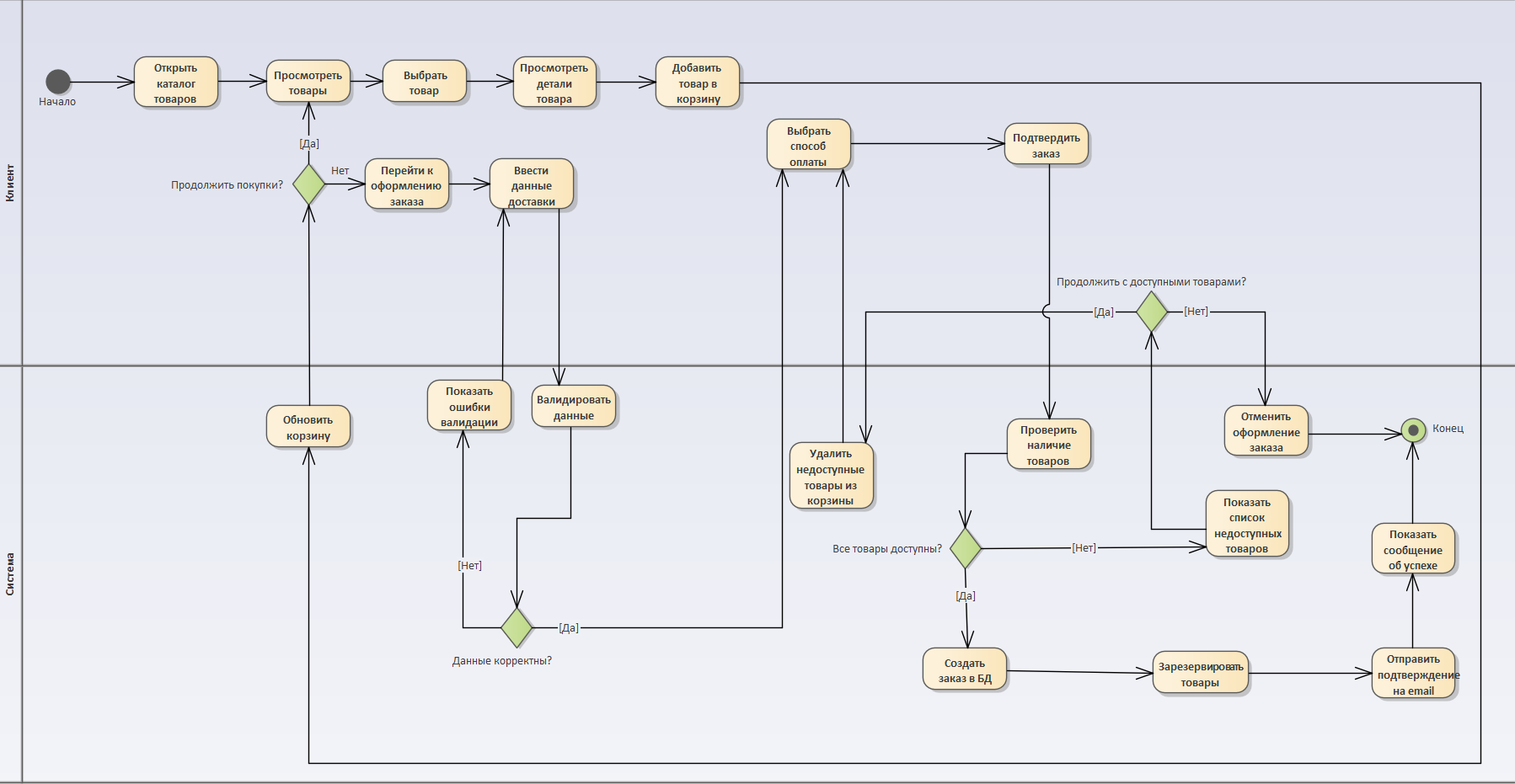
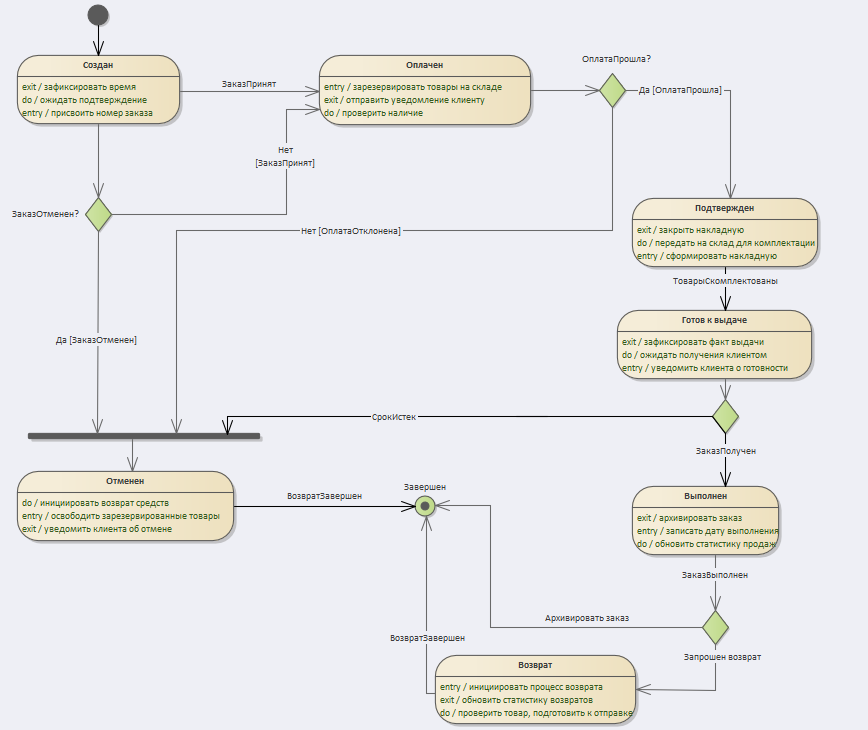


Рисунок И.2 – Оформление заказа клиентом

# ПРИЛОЖЕНИЕ К (справочное) Диаграмма состояний



# ПРИЛОЖЕНИЕ Л (справочное) Проект экранных форм

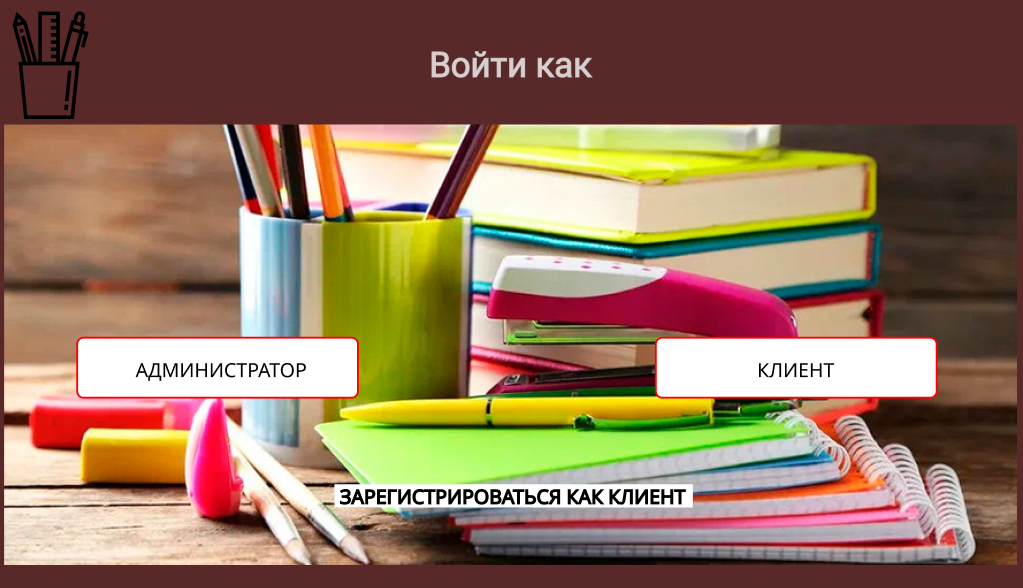


Рисунок Л.1 – Проект экранной формы «Выбор роли» (N1)

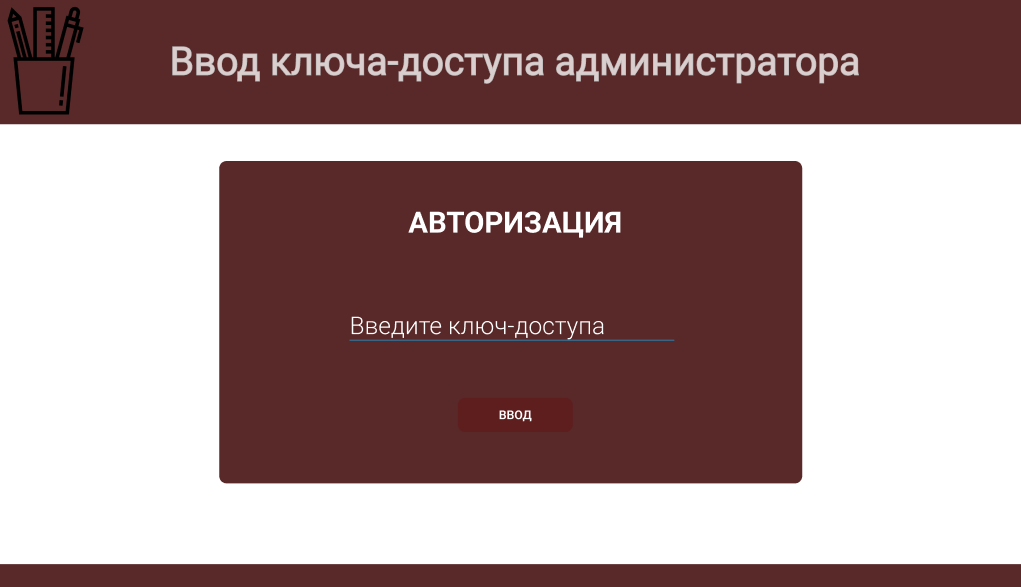


Рисунок Л.2 – Проект экранной формы «Ввод ключа-доступа администратора» (N2)



Рисунок Л.3 – Проект экранной формы «Вход администратора» (N3)

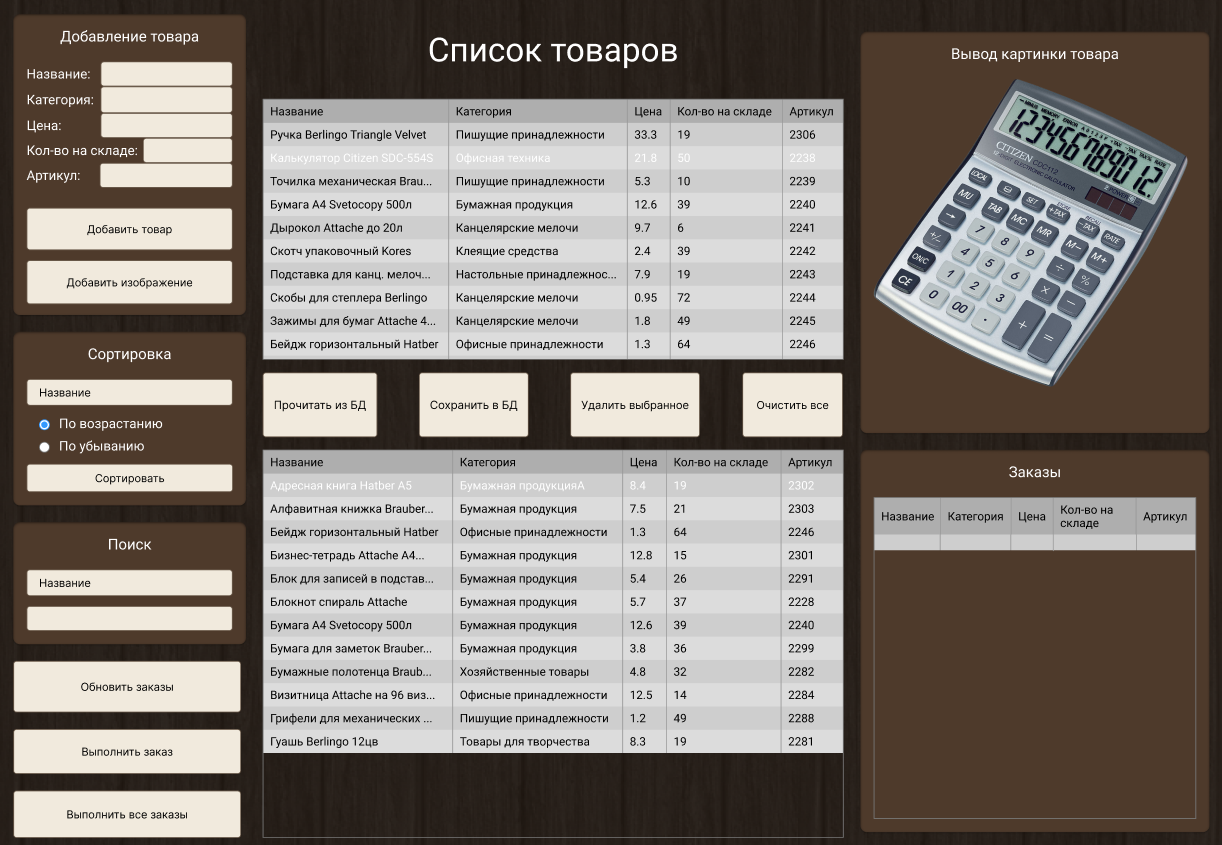


Рисунок Л.4 – Проект экранной формы «Админ-панель» (N4)

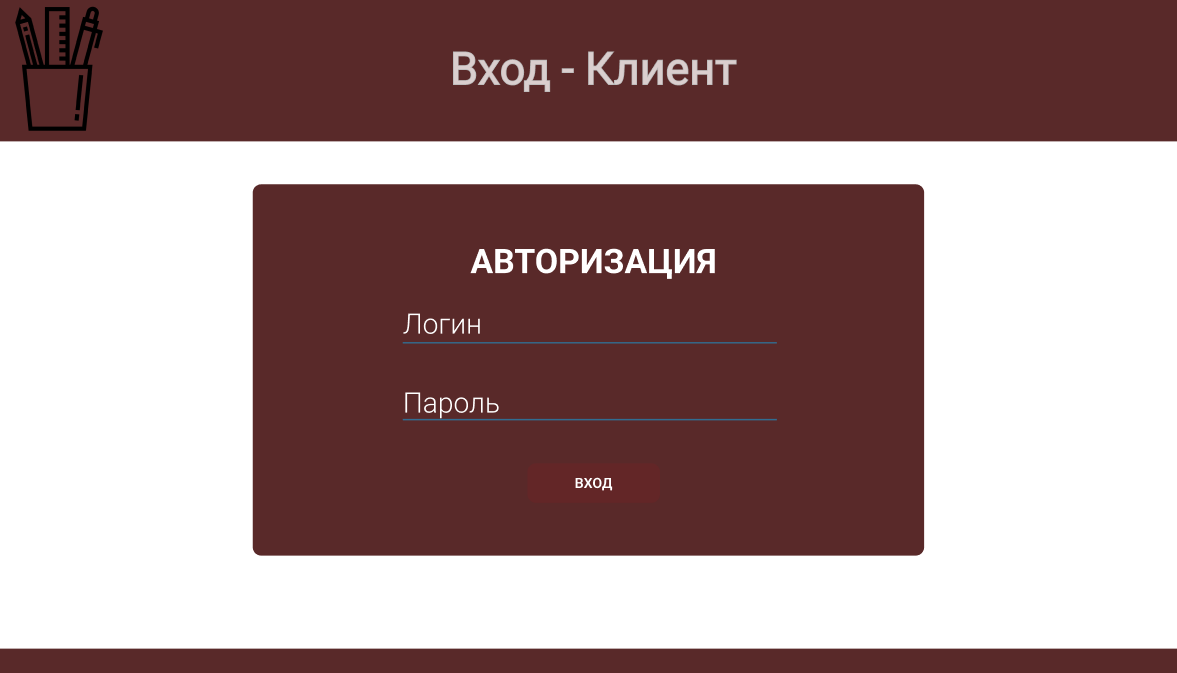


Рисунок Л.5 – Проект экранной формы «Вход клиента» (N5)

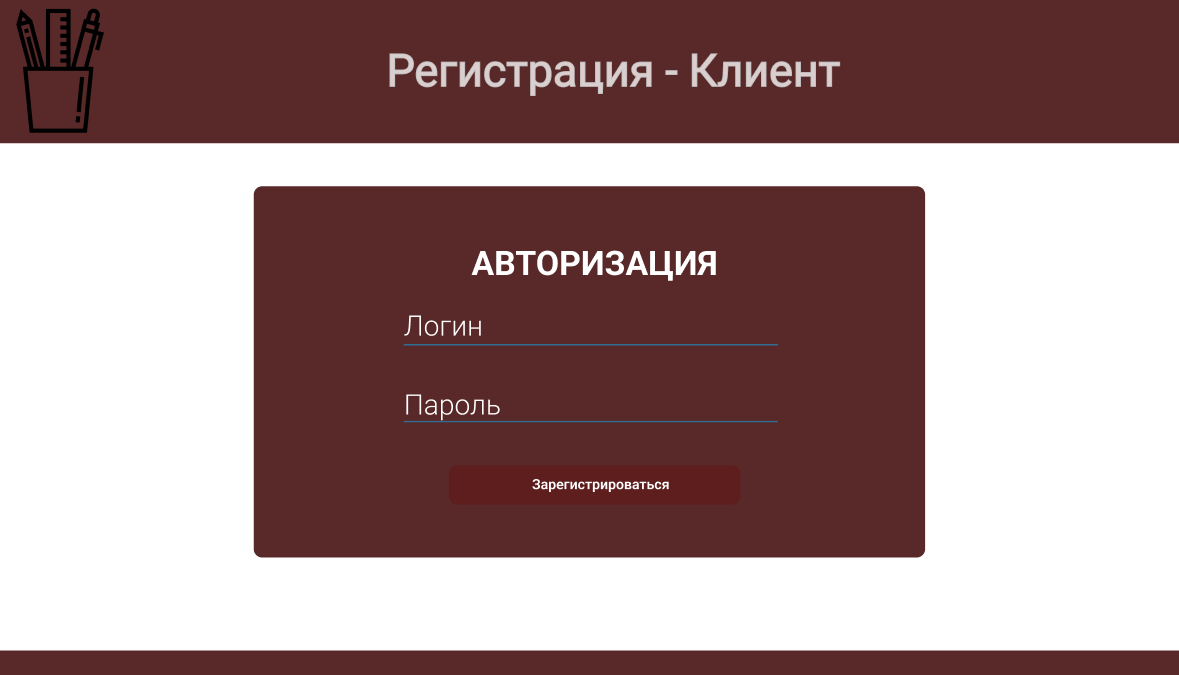


Рисунок Л.6 – Проект экранной формы «Регистрация клиента» (N6)

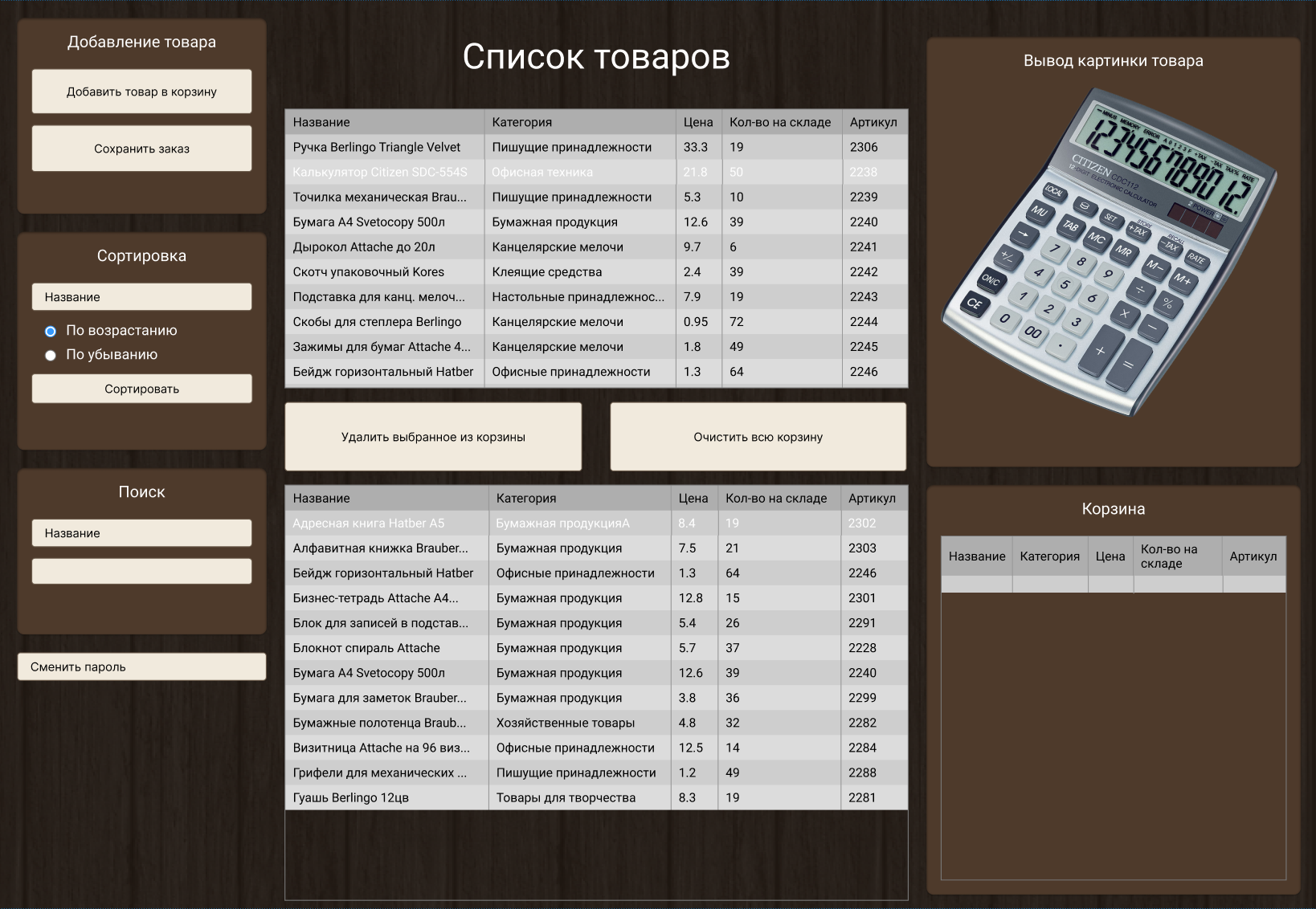


Рисунок Л.7 – Проект экранной формы «Клиент-панель» (N7)

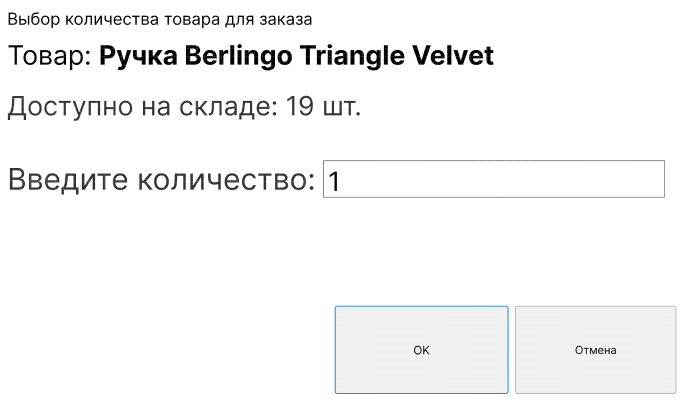


Рисунок Л.8 – Проект экранной формы «Выбор кол-ва товара для заказа» (N8)

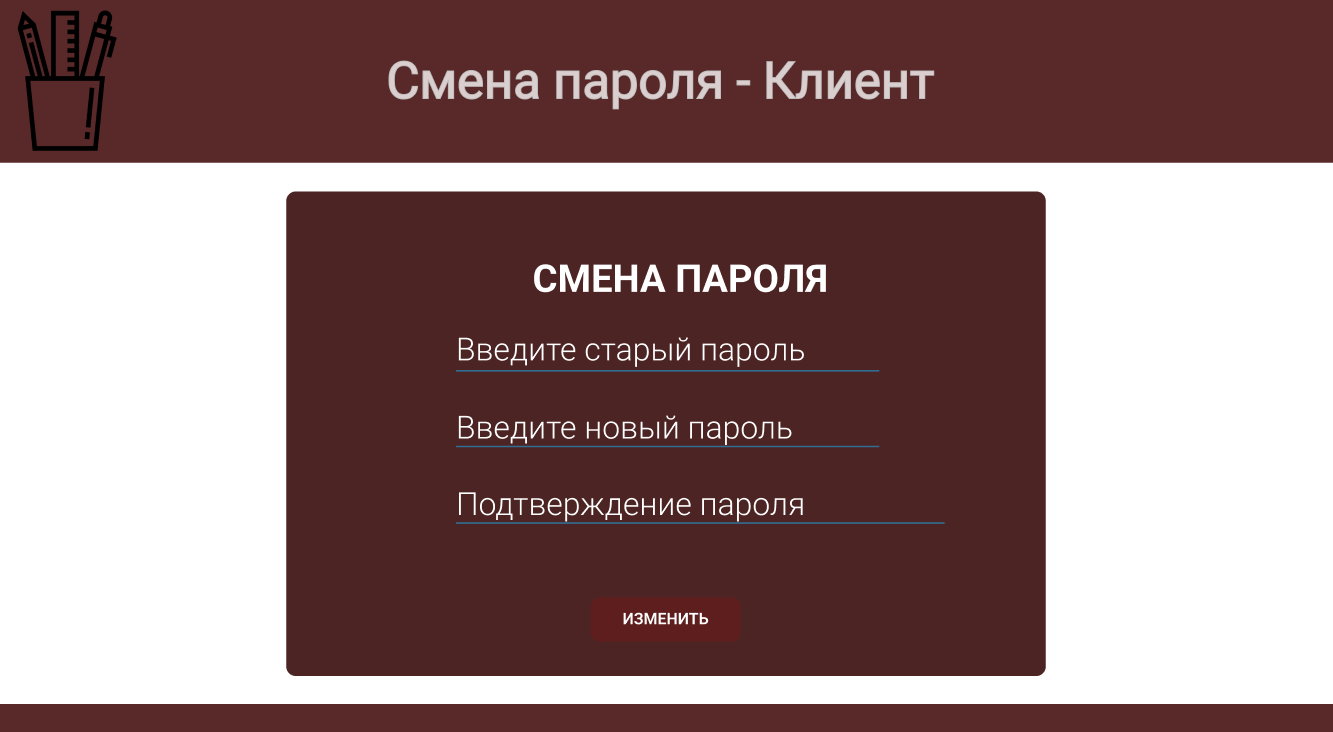


Рисунок Л.9 – Проект экранной формы «Смена пароля клиента» (N9)

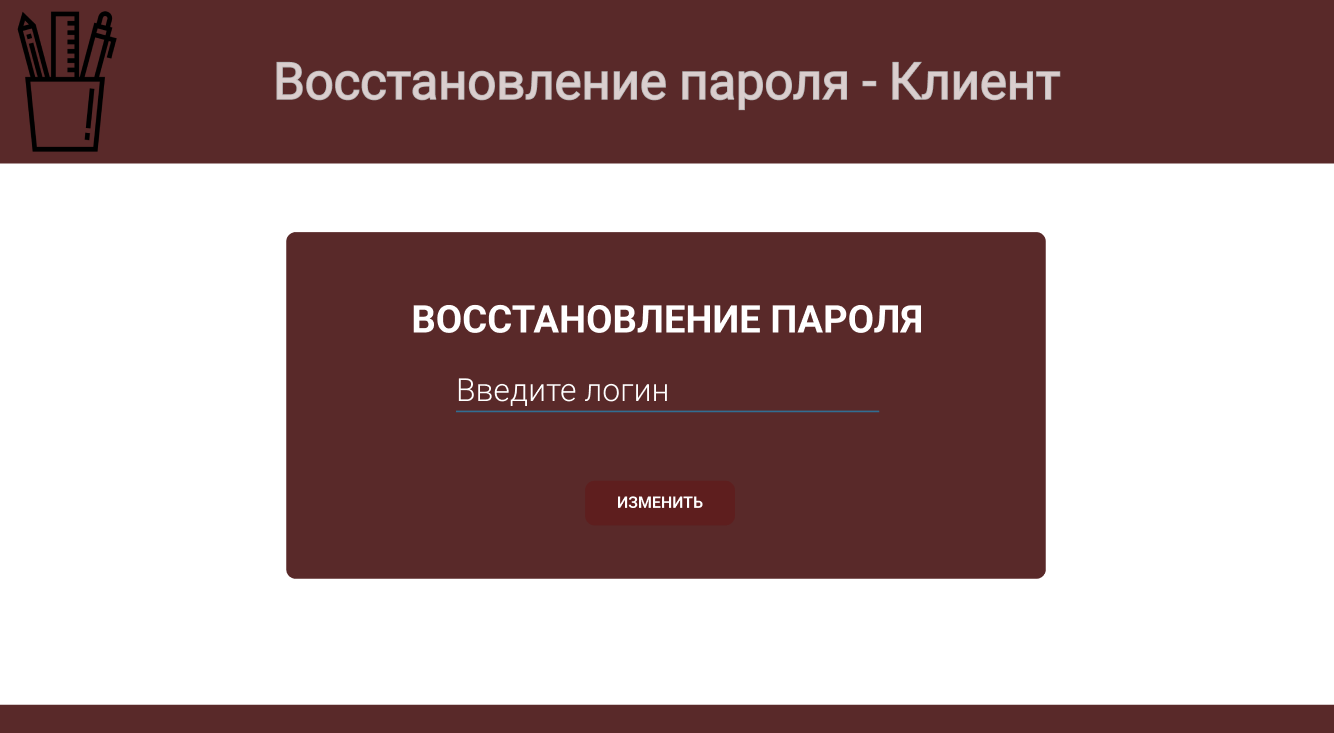
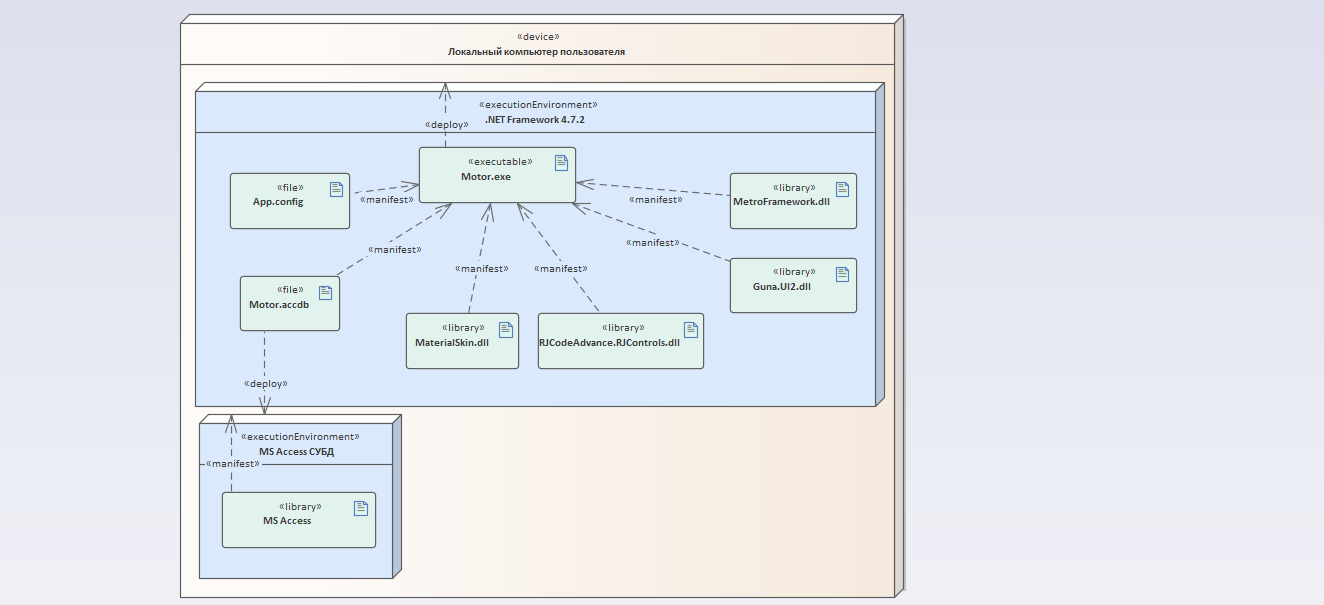


Рисунок Л.10 – Проект экранной формы «Восстановление пароля клиента» (N10)

# ПРИЛОЖЕНИЕ М (справочное) Диаграмма развертывания



# ПРИЛОЖЕНИЕ Н (обязательное) Программа и методика испытаний

**Н1 Объект испытаний**

Объектом испытаний является программное средство «Каталог канцелярских товаров», предназначенное для автоматизации управления товарным ассортиментом и обслуживания клиентов в торговой деятельности.

**Н2 Цель испытаний**

Целью испытания является проверка выполнения функциональных требований изложенных в техническом задании в ПРИЛОЖЕНИИ А и соответствия программы предъявляемым требованиям, а также выявление ошибок и проверка готовности к эксплуатации.

**Н3 Требования к программе**

**Н3.1 Требования к составу обрабатываемых данных**

Программа должна обеспечить хранение и обработку следующих данных:

* данные о товарах (артикул, название, описание, цена, количество на складе, фотография);
* данные о категориях товаров (название, описание);
* данные о пользователях (логин, пароль, роль);
* данные о заказах (дата заказа, статус, сумма);
* данные о товарах в заказе (артикул, количество, цена).

**Н3.2 Требования к функциональным характеристикам**

Программа должна иметь возможность выполнения следующих функций:

Для роли Администратор:

* ввод и проверка ключа доступа администратора;
* аутентификация в системе;
* просмотр полного списка товаров;
* добавление новых товаров в каталог;
* редактирование характеристик товаров;
* удаление товаров из каталога;
* загрузка и сохранение данных в базу данных;
* поиск товаров по различным критериям (название, категория, цена, количество, артикул);
* сортировка товаров (по названию, категории, цене, количеству, артикулу);
* просмотр списка заказов клиентов;
* обработка и выполнение заказов;
* сохранение выполненных заказов.

Для роли Клиент:

* регистрация нового клиента;
* аутентификация в системе;
* восстановление забытого пароля;
* смена пароля;
* просмотр каталога товаров;
* просмотр подробной информации о товаре;
* поиск товаров по различным критериям (название, категория, цена, количество, артикул);
* сортировка товаров (по названию, категории, цене, количеству, артикулу);
* добавление товаров в корзину;
* просмотр содержимого корзины;
* удаление товаров из корзины;
* изменение количества товаров в корзине;
* оформление заказа.

**Н3.3 Требования к надежности**

Программа должна:

* защищать данные пользователей через аутентификацию;
* сохранять целостность данных при операциях;
* обрабатывать ошибки с информативными сообщениями;
* восстанавливаться после сбоев;
* разграничивать права доступа по ролям.

**Н4 Требования к программной документации**

Программная документация должна быть представлена в следующем составе:

− «Техническое задание»;

* «Руководство оператора»;

− «Текст программы».

Программная документация должна быть оформлена в соответствии со следующими нормативными документами:

− ГОСТ 19.201-78 ЕСПД «Техническое задание. Требования к содержанию и оформлению»;

– ГОСТ 19.505-79 ЕСПД. «Руководство оператора». Требования к содержанию и оформлению;

− ГОСТ 19.401-78 ЕСПД «Текст программы. Требования к содержанию и оформлению».

**Н5 Средства и порядок испытаний**

**Н5.1 Средства испытаний**

Минимальные аппаратные требования:

− процессор: 1 гигагерц (ГГц) или более быстрый процессор или SoC;

− оперативная память: 1 гигабайт (ГБ) для 32 – разрядных или 2 ГБ для 64 – разрядных;

− место на жестком диске: 32 ГБ для 64 – битной и 32 – битной ОС;

− видеокарта: с поддержкой DirectX 9;

− разрешение экрана: 800 x 600, минимальный размер диагонали для основного дисплея 7 дюймов и больше.

Компьютер должен быть укомплектованы мышью, клавиатурой, сетевым шнуром, монитором.

**Н5.2 Порядок испытаний**

Испытание программного средства проводятся для следующих тестовых случаев:

1) Проверка функций под ролью «Администратор»;

1.1) проверка работы ввода ключа-доступа;

1.2) проверка работы аутентификации;

1.3) проверка отображения списка товаров;

1.4) добавление новых товаров в каталог;

1.5) добавление изображения к товару;

1.6) редактирование данных товаров;

1.7) удаление и очистка товаров из каталога;

1.8) проверка функций обновления и выполнения заказов;

1.9) проверка функции сохранения выполненных заказов;

1.10) проверка сохранения списка товаров в базу данных;

1.11) поиск товаров по совпадению в названии;

1.12) поиск товаров по совпадению в категории;

1.13) поиск товаров по совпадению в цене;

1.14) поиск товаров по совпадению в количестве на складе;

1.15) поиск товаров по совпадению в артикуле;

1.16) сортировка по названию;

1.17) сортировка по категории;

1.18) сортировка по цене;

1.19) сортировка по количеству на складе;

1.20) сортировка по артикулу.

2) Проверка функций под ролью «Клиент»;

2.1) проверка работы регистрации;

2.2) проверка работы аутентификации;

2.3) проверка работы восстановления пароля;

2.4) проверка работы смены пароля;

2.5) проверка отображения списка товаров;

2.6) добавление товаров в корзину;

2.7) сохранение заказа в базу данных;

2.8) поиск товаров по совпадению в названии;

2.9) поиск товаров по совпадению в категории;

2.10) поиск товаров по совпадению в цене;

2.11) поиск товаров по совпадению в количестве на складе;

2.12) поиск товаров по совпадению в артикуле;

2.13) сортировка по названию;

2.14) сортировка по категории;

2.15) сортировка по цене;

2.16) сортировка по количеству на складе;

2.17) сортировка по артикулу.

**Н6 Методы испытаний**

Работоспособность программного обеспечения проверяется результатом, выдаваемых программной после выполнения действий, заданных по определённому сценарию. Тестирование разработанного программного обеспечения заключается в последовательности действий, направленных на получение результатов.

В таблице Н.1 представлен тест-план тестирования программного обеспечения.

Таблица Н.1 – План тестирования программного средства

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| №  тесто-  вого  приме  ра | №  тест-  требо-  вания | Описание теста | Входные  данные | Ожидаемый  результат | Сценарий  тестирования |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 1 | 1.1 | Проверка работы  ввода ключа-доступа администратора | Ключ-доступа:  HGGX21 | Переход на форму входа администратора | Выбрать роль «Администратор»  В поле «Ключ-доступа» ввести выданный ключ доступа главным администратором и после нажать кнопку «Ввод» |
| 2 | 1.1 | Проверка работы  ввода ключа-доступа  при не  заполненных  значениях | Ключ-доступа:  (пусто) | Вывод предупрежде-ния  «Поле ключа- доступа должно быть заполнено!» | В поле «Ключ-доступа» не вводить выданный главным администратором ключ-доступа и нажать кнопку «Ввод» |

Продолжение табл. Н.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 3 | 1.2 | Проверка работы аутентификации при верном логине и пароле | Логин:  12345  Пароль:  12345 | Вход под ролью «Администратор» | Нажать на кнопку «Администратор» В поля  «Логин» и «Пароль»  ввести  входные  данные и нажать кнопку «Войти» |
| 4 | 1.2 | Проверка работы  аутентификации  при верном  логине, но не  верном пароле | Логин:  12345  Пароль:  54321 | Вывод  ошибки  «Неверный  логин или  пароль» | Нажать на кнопку «Администратор» В поле  «Пароль»  ввести  неправильные  входные  данные и  нажать кнопку  «Войти» |
| 5 | 1.2 | Проверка работы  аутентификации  при не верном  логине, но верном пароле | Логин:  5432  Пароль:  12345 | Вывод  ошибки  «Неверный  логин или  пароль» | Нажать на кнопку «Администратор» В поле  «Логин»  ввести  неправильные  входные  данные и  нажать кнопку «Войти» |
| 6 | 1.2 | Проверка работы  аутентификации при не заполненных  значениях | Логин:  (пусто)  Пароль:  (пусто) | Вывод  предупрежде-ния  «Поля логина и пароля не должны быть пустыми» | Нажать на кнопку «Администратор» В поле  «Логин»  ввести  неправильные входные  данные и нажать |
| 7 | 1.3 | Проверка  отображения  списка  товаров | - | Отображение таблицы товаров с  полями:  «Название»,  «Категория»,  «Цена», «Кол-во на складе» и «Артикул» | Авторизоваться под ролью  «Администратор»  Просмотреть таблицу со списком товаров |

Продолжение табл. Н.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 8 | 1.3 | Проверка  отображения пустого списка  товаров | Нет товаров в списке | Отображение пустой таблицы товаров с  полями | Очистить базу  списка товаров.  Авторизоваться как администра-  тор. Просмотреть |
| 9 | 1.3 | Проверка  корректности  отображаемых  данных | 5 товаров в системе | Таблица содержит ровно 5 товаров с корректными данными | Добавить 5 тестовых товаров. Проверить отображение списка |
| 10 | 1.4 | Проверка добавления нового товара в каталог с валидными данными | Название: Ручка Pentel  Категория: Пишущие принадлежности Цена: 2,3  Кол-во на складе: 3  Артикул: 2222 | Новый товар появляется в таблице | Заполнить все  поля в панели добавления товара.  Нажать кнопку «Добавить товар» |
| 11 | 1.4 | Проверка  добавления  существующего товара | Название: Ручка Pentel  Категория: Пишущие принадлежности  Цена: 2,3  Кол-во на складе: 3  Артикул:222 | Предупреждение  «Товар с таким названием уже существует!» | Попытаться  добавить  товар с существующим названием.  Проверить  предупреждение об  ошибке |
| 12 | 1.4 | Проверка добавления товара с незаполненными  полями | Название:  (пусто)  Категория:  (пусто)  Цена:  (пусто)  Кол-во на складе:  (пусто)  Артикул: (пусто) | Ошибка  «Пожалуйста заполните все поля» | Оставить  обязательные  поля пустыми.  Попытаться  добавить товар |
| 13 | 1.5 | Проверка добавления изображения к товару | Изображение с именем соответствую-щим названию добавляемого товара | Отображение изображения товара | Ввести во все поля данные о товаре. Нажать кнопку «Добавить изображение».  Выбрать |

Продолжение табл. Н.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
|  |  |  |  |  | изображение c именем соответствующему названию товара.  Нажать кнопку «Добавить товар» |
| 14 | 1.5 | Проверка добавления изображения с различным от названия товара именем | Изображение с несоотве-тствующим названию товара  именем | Отсутствие изображения | Ввести во все поля данные о товаре. Нажать кнопку «Добавить изображение». Выбрать изображение с именем несоответствующему названию товара. |
| 15 | 1.6 | Проверка  редактирования  данных  товаров | Изменение  названия товара  «Ручка Pentel» на «Линейка», категории «Пишущие принадлежности» на «Чертежные | Данные о товаре успешно обновляются в таблице | Выбрать товар из таблицы.  Нажать два раза по изменяемому полю и изменить название, категорию, цену, кол-во на складе и артикул. |
|  |  |  | принадлежности», цены с «2,3» на «12,3», кол-ва на складе с «3» до «328», с артикула «222» на «333» |  |  |
| 16 | 1.6 | Проверка отмены  редактирования | Изменение названия товара «Линейка» на «Циркуль» | Данные товара остаются прежними | Начать изменение поля. Внести изменения.  Нажать сочетание клавиш «Ctrl + Z» |
| 17 | 1.6 | Проверка  редактирования с невалидными  данными | Установка пустого значения для поля | Ошибка  «Все поля должны быть заполнены!» | Начать изменение поля.  Внести везде пустые значения. |

Продолжение табл. Н.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
|  |  |  |  |  | Нажать «Enter» либо кнопку «Сохранить в файл» |
| 18 | 1.7 | Проверка  удаления  товара | Удаление выбранного товара | Товар исчезает из таблицы | Выбрать товар.  Нажать кнопку «Удалить выбранное».  Подтвердить удаление в диалоговом окне |
| 19 | 1.7 | Проверка отмены  удаления | Отмена  удаления  товара | Товар остается в таблице | Выбрать товар.  Нажать кнопку «Удалить выбранное». |
|  |  |  |  |  | Отменить удаление в диалоговом окне |
| 20 | 1.7 | Проверка очистки всей таблицы | Очистка всей таблицы | Таблица полностью очищается | Нажать кнопку «Очистить все» |
| 21 | 1.8 | Проверка выполнения заказов | Заказ клиента | Сообщение  «Заказ выполнен».  Списано со склада: «...шт».  Осталось на складе: «...шт»  Отчёт сохранен  Заказ удален из списка ожидающих | Выбрать заказ из таблицы заказов.  Нажать кнопку «Выполнить заказ».  Подтвердить выполнение заказа в диалоговых окнах |
| 22 | 1.8 | Проверка выполнения всех заказов | Заказ клиента с несколькими товарами | Сообщение  «Обработка заказов завершена!».  Успешно выполнено: «... заказов».  Общая сумма: «... руб» Отчет сохранен | Нажать на кнопку «Выполнить все заказы».  Подтвердить выполнение заказов в диалоговых окнах |

Продолжение табл. Н.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 23 | 1.9 | Проверка сохранения выполненных заказов | Заказ клиента с несколькими либо одним товарами | Сохранение выполненных заказов в БД | Нажать на кнопки «Выполнить заказ» или «Выполнить все заказы».  Подтвердить выполнение заказов в диалоговых окнах. |
|  |  |  |  |  | Просмотреть файл «zakazy\_reports.txt» на наличие отчета о выполненных заказах |
| 24 | 1.10 | Проверка сохранения списка товаров | Список товаров в таблице | Сохранение списка товаров базу данных | Нажать на кнопку «Сохранить в файл».  Просмотреть файл «data.txt» на наличие сохраненных товаров |
| 25 | 1.11 | Поиск товара по полному названию | Скотч упаковочный Kores | 1 точный  результат | Ввести в текстовое поле полное  Название товара |
| 26 | 1.11 | Поиск товара по фрагменту названия | Введённый  фрагмент для  поиска | Все товары,  содержащие  введённый  фрагмент | Ввести в текстовое поле часть  названия |
| 27 | 1.12 | Поиск товара по полной категории | Офисная техника | 1 точный  результат | Ввести в текстовое поле полную категорию товара |
| 28 | 1.12 | Поиск товара по фрагменту категории | Введённый  фрагмент для  поиска | Все товары,  содержащие  введённый  фрагмент | Ввести в текстовое поле часть  категории |
| 29 | 1.13 | Поиск товара по полной цене | 2,3 | 1 точный  результат | Ввести в поле точную цену |

Продолжение табл. Н.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 30 | 1.13 | Поиск товара по фрагменту цены | Введённый  фрагмент для  поиска | Все товары,  содержащие  введённый  фрагмент | Ввести в текстовое поле часть  цены |
| 31 | 1.14 | Поиск товара по полному количеству на складе | 14 | 1 точный  результат | Ввести в текстовое поле полное количество на складе товара |
| 32 | 1.14 | Поиск товара по фрагменту количества  товара на складе | Введённый  фрагмент для  поиска | Все товары,  содержащие  введённый  фрагмент | Ввести в текстовое поле часть количества на складе товара |
|  |  |  |  |  |  |
| 33 | 1.15 | Поиск товара по полному артикулу | 2233 | 1 точный  результат | Ввести в текстовое поле полный артикул товара |
| 34 | 1.15 | Поиск товара по фрагменту артикула | Введённый  фрагмент для  поиска | Все товары,  содержащие  введённый  фрагмент | Ввести в текстовое поле часть артикула |
| 35 | 1.16 | Сортировка  товаров по названию  (возрастание А -Я) | Нажатие на  кнопку  «По возрастанию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для  сортировки «Название».  Выбрать вид  сортировки по  возрастанию.  Нажать кнопку  «Сортировать». |
| 36 | 1.16 | Сортировка  товаров по названию  (убывание Я - А) | Нажатие на  кнопку  «По убыванию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для  сортировки «Название».  Выбрать вид  сортировки по  убыванию.  Нажать кнопку  «Сортировать». |
| 37 | 1.17 | Сортировка  товаров по категории  (возрастание А -Я) | Нажатие на  кнопку  «По возрастанию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для  сортировки «Категория».  Выбрать вид |

Продолжение табл. Н.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
|  |  |  |  |  | сортировки по  возрастанию.  Нажать кнопку  «Сортировать». |
| 38 | 1.17 | Сортировка  товаров по категории  (убывание Я - А) | Нажатие на  кнопку  «По убыванию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для  сортировки «Категория».  Выбрать вид  сортировки по  убыванию.  Нажать кнопку  «Сортировать». |
| 39 | 1.18 | Сортировка  товаров по цене  (возрастание) | Нажатие на  кнопку  «По возрастанию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для  сортировки «Цена».  Выбрать вид  сортировки по  возрастанию.  Нажать кнопку  «Сортировать». |
| 40 | 1.18 | Сортировка  товаров по цене  (убывание) | Нажатие на  кнопку  «По убыванию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для  сортировки «Цена».  Выбрать вид  сортировки по  убыванию.  Нажать кнопку «Сортировать». |
| 41 | 1.19 | Сортировка  товаров по количеству на складе  (возрастание) | Нажатие на  кнопку  «По возрастанию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для  сортировки «Кол-во на складе».  Выбрать вид  сортировки по  возрастанию.  Нажать кнопку  «Сортировать». |
| 42 | 1.19 | Сортировка  товаров по количеству на складе  (убывание) | Нажатие на  кнопку  «По убыванию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для сортировки «Кол-во на складе».  Выбрать вид  сортировки по  убыванию. |

Продолжение табл. Н.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
|  |  |  |  |  | Нажать кнопку «Сортировать». |
| 43 | 1.20 | Сортировка  товаров по артикулу  (возрастание) | Нажатие на  кнопку  «По возрастанию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для  сортировки «Артикул».  Выбрать вид  сортировки по  возрастанию.  Нажать кнопку  «Сортировать». |
| 44 | 1.20 | Сортировка  товаров по артикулу  (убывание) | Нажатие на  кнопку  «По убыванию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для  сортировки «Артикул».  Выбрать вид  сортировки по  убыванию.  Нажать кнопку  сортировать. |
| 45 | 2.1 | Проверка работы  регистрации  нового клиента | Логин:  Client  Пароль:  12345 | Регистрация клиента | Нажать на текст «Зарегистрироваться как клиент»  В поля «Логин» и «Пароль» ввести входные данные и нажать кнопку «Регистрация» |
| 46 | 2.1 | Проверка работы  регистрации  при не  заполненных  значениях | Логин:  (пусто)  Пароль:  (пусто) | Вывод предупрежде-ния  «Поля логина и пароля не  должны быть пустыми» | В поля «Логин» и «Пароль» не вводить входные  данные и нажать кнопку «Регистрация» |
| 47 | 2.2 | Проверка работы аутентификации при верном логине и пароле | Логин:  Client  Пароль:  12345 | Вход под ролью «Клиент» | Нажать на кнопку «Клиент»  В поля  «Логин» и «Пароль» ввести  входные данные и нажать кнопку «Войти» |
| 48 | 2.2 | Проверка работы  аутентификации  при верном | Логин:  Client  Пароль:  54321 | Вывод  ошибки  «Неверный  логин или | Нажать на кнопку «Клиент»  В поле  «Пароль» ввести |

Продолжение табл. Н.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
|  |  | логине, но не верном пароле |  | пароль» | неправильные входные данные и нажать кнопку  «Войти» |
| 49 | 2.2 | Проверка работы  аутентификации  при не верном  логине, но верном пароле | Логин:  tneilC  Пароль:  12345 | Вывод  ошибки  «Неверный  логин или  пароль» | Нажать на кнопку «Клиент»  В поле «Логин» ввести неправильные  входные данные и нажать кнопку «Войти» |
| 50 | 2.2 | Проверка работы  аутентификации при не заполненных  значениях | Логин:  (пусто)  Пароль:  (пусто) | Вывод  предупрежде-ния  «Поля логина и пароля не должны быть пустыми» | Нажать на кнопку «Клиент»  В поле  «Логин» ввести  неправильные входные данные и нажать кнопку «Войти» |
| 51 | 2.3 | Проверка  отображения  списка  товаров | - | Отображение таблицы товаров с  полями:  «Название»,  «Категория»,  «Цена», «Кол-во на складе» и «Артикул» | Авторизоваться под ролью  «Клиент»  Просмотреть таблицу со списком товаров |
| 52 | 2.4 | Проверка добавления товаров в корзину | Нажатие на кнопку «Добавить в корзину» | Отображение таблицы с добавленными товарами | Выбрать товар из списка.  Нажать кнопку «Добавить товар».  Выбрать в диалоговом окне количество товара.  Подтвердить.  Нажать кнопку «Сохранить заказ» |
| 53 | 2.5 | Проверка сохранения заказов в файл | Товары в корзине | Сообщение  «Заказ успешно сохранён», | Нажать кнопку  «Сохранить заказ».  Просмотреть файл «zakazy.txt». |

Продолжение табл. Н.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
|  |  |  |  | «Товаров в заказе...»,  «Общая сумма...руб» | Убедиться в том что заказ сохранился |
| 54 | 2.6 | Поиск товара по полному названию | Скотч упаковочный Kores | 1 точный  результат | Ввести в текстовое поле полное название товара |
| 55 | 2.6 | Поиск товара по фрагменту названия | Введённый  фрагмент для  поиска | Все товары,  содержащие  введённый  фрагмент | Ввести в текстовое поле часть названия |
| 56 | 2.7 | Поиск товара по полной категории | Офисная техника | 1 точный  результат | Ввести в текстовое поле полную категорию товара |
| 57 | 2.7 | Поиск товара по фрагменту категории | Введённый  фрагмент для  поиска | Все товары,  содержащие  введённый  фрагмент | Ввести в текстовое поле часть  категории |
| 58 | 2.8 | Поиск товара по полной цене | 2,3 | 1 точный  результат | Ввести в текстовое поле полную цену товара |
| 59 | 2.8 | Поиск товара по фрагменту цены | Введённый  фрагмент для  поиска | Все товары,  содержащие  введённый  фрагмент | Ввести в текстовое поле часть  цены |
| 60 | 2.9 | Поиск товара по полному количеству на складе | 14 | 1 точный  результат | Ввести в текстовое поле полное количество на складе товара |
| 61 | 2.9 | Поиск товара по фрагменту количества товара на складе | Введённый  фрагмент для  поиска | Все товары,  содержащие  введённый  фрагмент | Ввести в текстовое поле часть количества на складе товара |
| 62 | 2.10 | Поиск товара по полному артикулу | 2233 | 1 точный  результат | Ввести в текстовое поле полный артикул товара |
| 63 | 2.10 | Поиск товара по фрагменту артикула | Введённый  фрагмент для  поиска | Все товары,  содержащие  введённый  фрагмент | Ввести в текстовое поле часть артикула |

Продолжение табл. Н.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 64 | 2.11 | Сортировка  товаров по названию  (возрастание А -Я) | Нажатие на  кнопку  «По возрастанию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для  сортировки «Название».  Выбрать вид  сортировки по  возрастанию.  Нажать кнопку  «Сортировать». |
| 65 | 2.11 | Сортировка  товаров по названию  (убывание Я - А) | Нажатие на  кнопку  «По убыванию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для  сортировки «Название».  Выбрать вид  сортировки по  убыванию.  Нажать кнопку  «Сортировать» |
| 66 | 2.12 | Сортировка  товаров по категории  (возрастание А -Я) | Нажатие на  кнопку  «По возрастанию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для  сортировки «Категория».  Выбрать вид  сортировки по  возрастанию.  Нажать кнопку  «Сортировать» |
| 67 | 2.12 | Сортировка  товаров по категории  (убывание Я - А) | Нажатие на  кнопку  «По убыванию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для  сортировки «Категория».  Выбрать вид  сортировки по  убыванию.  Нажать кнопку  «Сортировать» |
| 68 | 2.13 | Сортировка  товаров по цене  (возрастание) | Нажатие на  кнопку  «По возрастанию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для  сортировки «Цена».  Выбрать вид  сортировки по  возрастанию.  Нажать кнопку  «Сортировать». |

Продолжение табл. Н.1

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| 1 | 2 | 3 | 4 | 5 | 6 |
| 69 | 2.13 | Сортировка  товаров по цене  (убывание) | Нажатие на  кнопку  «По убыванию» | Список товаров упорядочен по  заданному критерию | Выбрать критерий для  сортировки «Цена».  Выбрать вид  сортировки по  убыванию.  Нажать кнопку «Сортировать». |
| 70 | 2.14 | Сортировка  товаров по количеству на складе  (возрастание) | Нажатие на  кнопку  «По возрастанию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для  сортировки «Кол-во на складе».  Выбрать вид  сортировки по  возрастанию.  Нажать кнопку  «Сортировать». |
| 71 | 2.14 | Сортировка  товаров по количеству на складе  (убывание) | Нажатие на  кнопку  «По убыванию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для  сортировки «Кол-во на складе».  Выбрать вид  сортировки по  убыванию.  Нажать кнопку «Сортировать». |
| 72 | 2.15 | Сортировка  товаров по артикулу  (возрастание) | Нажатие на  кнопку  «По возрастанию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для  сортировки «Артикул».  Выбрать вид  сортировки по  возрастанию.  Нажать кнопку  «Сортировать». |
| 73 | 2.15 | Сортировка  товаров по артикулу  (убывание) | Нажатие на  кнопку  «По убыванию» | Список товаров упорядочен по заданному критерию | Выбрать критерий для  сортировки «Артикул».  Выбрать вид  сортировки по  убыванию.  Нажать кнопку  сортировать. |

# ПРИЛОЖЕНИЕ П (обязательное) Руководство оператора

**П1 Назначение программы**

Программа предназначена для учета и автоматизации работы с каталогом канцелярских товаров на предприятии. Система позволяет оформлять заказы, просматривать ассортимент, а также вести учет информации о товарах. Администраторы могут добавлять и редактировать позиции каталога, а клиенты – просматривать товары и формировать заказы. Программа ориентирована на сотрудников, отвечающих за обработку заказов и управление товарной базой.

**П2 Условие выполнения программы**

Для функционирования программы, необходимо аппаратное обеспечение со следующими характеристиками:

− процессор: 1 гигагерц (ГГц) или более быстрый процессор или SoC;

− оперативная память: 1 гигабайт (ГБ) для 32 – разрядных или 2 ГБ для 64 – разрядных;

− место на жестком диске: 32 ГБ для 64 – битной и 32 – битной ОС;

− видеокарта: с поддержкой DirectX 9;

− разрешение экрана: 800 x 600, минимальный размер диагонали для основного дисплея 7 дюймов и больше.

Компьютер должен быть укомплектован мышью, клавиатурой, сетевым шнуром, монитором.

**П3 Выполнение программы**

**П3.1 Ввод ключа-доступа под ролью «Администратор»**

Ввод ключа-доступа администратора происходит через выданный главным администратором единый для всех ключ. После успешного заполнения ключа-доступа администратора переносит на окно входа. Окно ввода ключа-доступа представлено на рисунке (см. рис. П.1).

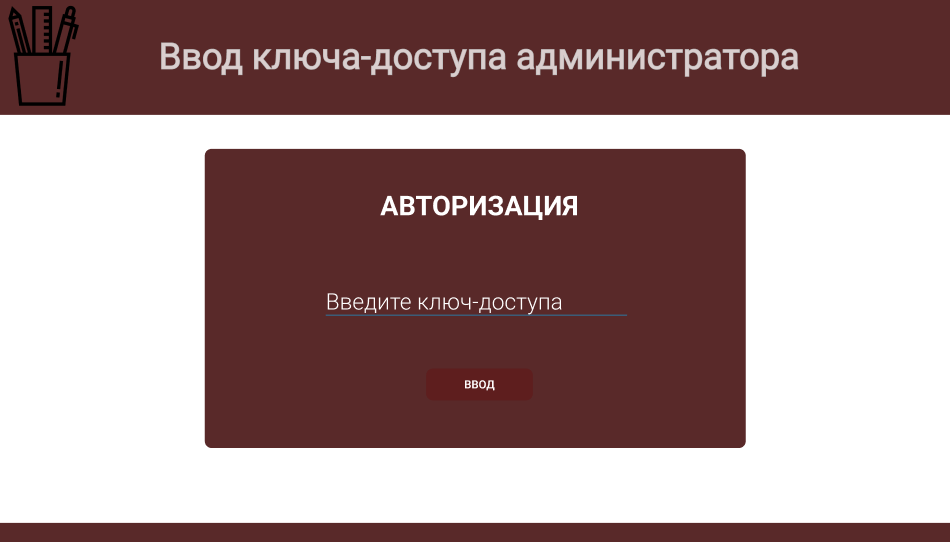


Рисунок П.1 – Окно ввода ключа-доступа администратора

Если ключ-доступа введен неверно система покажет ошибку о том что ключ неправильный (см. рис. П.2).

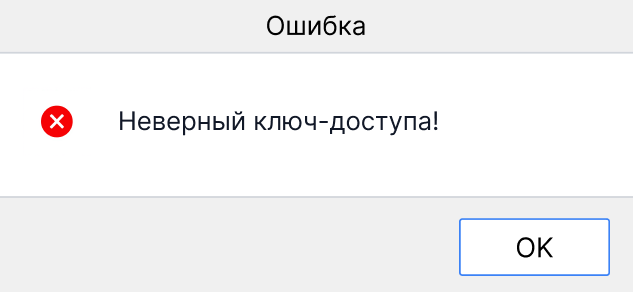


Рисунок П.2 – Ошибка при неверном ключе-доступа

**П3.2 Аутентификация пользователя под ролью «Администратор»**

Аутентификация администратора каталога канцелярских товаров происходит через ввод учетных данных, которые определяют административный уровень доступа к системе управления. После успешного ввода и проверки логина и пароля каталог канцелярских товаров готов к работе под административной ролью, предоставляя полный доступ к функциям управления товарами, категориями и другими возможностями, доступными администратору каталога.

Окно входа и авторизации представлено на рисунке (см. рис. П.3)

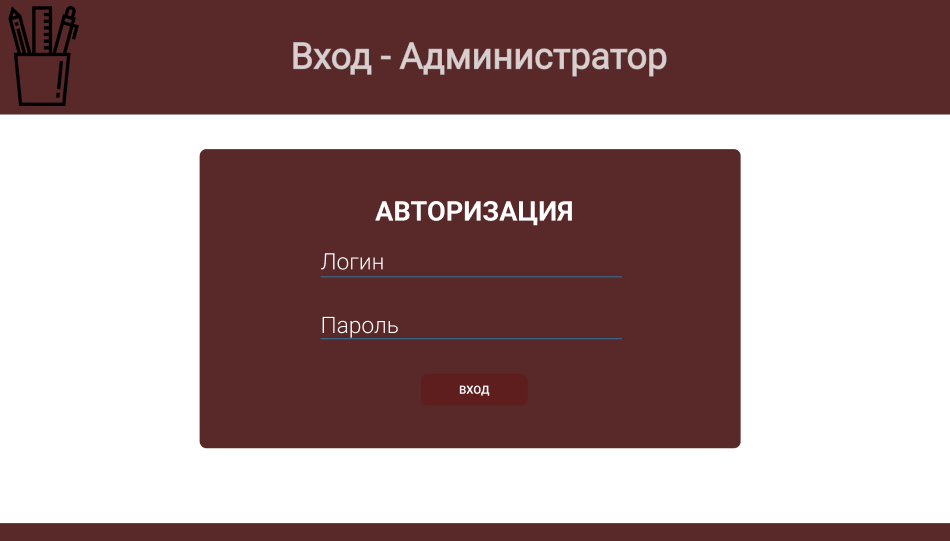


Рисунок П.3 – Окно входа администратора

Если администратор был ранее зарегистрирован в системе управления каталогом канцелярских товаров (имеет свой логин и пароль), то он может войти в систему под своими учетными данными. Если введенные логин или пароль не совпадают с сохраненными в системе, то появится сообщение об ошибке, которое представлено на рисунке (см. рис. П.4).

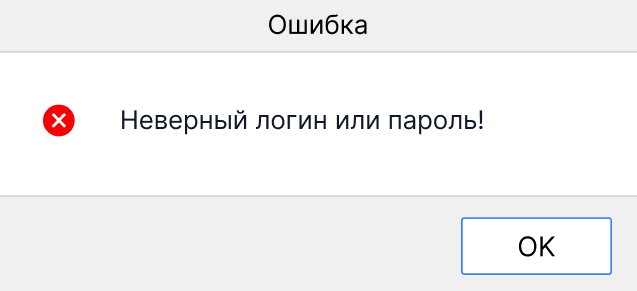


Рисунок П.4 – Ошибка при некорректном вводе данных

**П3.2.1 Просмотр списка товаров под ролью «Администратор»**

Администратор каталога канцелярских товаров может просматривать список товаров при взаимодействии с соответствующей формой управления каталогом, внешний вид которой представлен на рисунке (см. рис. П.5).



Рисунок П.5 – Просмотр списка товаров

**П3.2.2 Редактирование товаров под ролью «Администратор»**

Администратор каталога канцелярских товаров может редактировать информацию о товарах непосредственно в таблице. Для редактирования данных необходимо дважды щелкнуть на нужном поле в списке товаров, после чего поле станет доступным для изменения. После внесения необходимых корректировок изменения сохраняются в каталоге, обновляя информацию о товаре в системе. Процесс редактирования товаров представлен на рисунке (см. рис. П.6).



Рисунок П.6 – Редактирование данных товаров

**П3.2.3 Добавление товаров под ролью «Администратор»**

Если товар ранее не был добавлен в каталог канцелярских товаров, его необходимо внести в систему. Это может выполнить администратор каталога, добавив сначала изображение товара, а после информацию о товаре через панель добавления нового товара (см. рис. П.7).

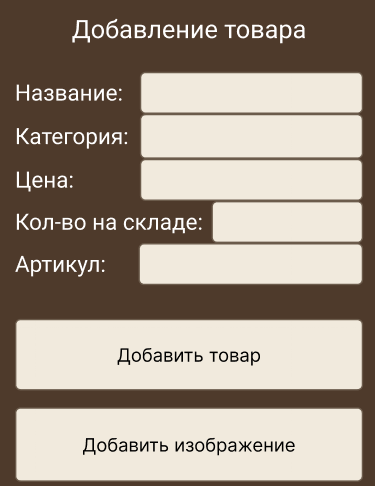


Рисунок П.7 – Панель добавления товара

Если при добавлении нового товара поля останутся незаполненными, отобразится сообщение об ошибке, представленное на рисунке (см. рис. П.8).

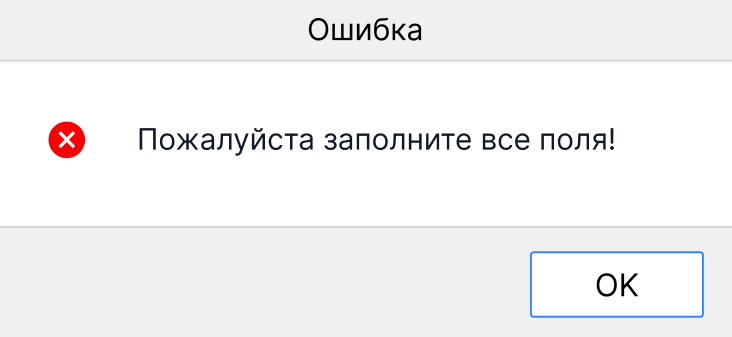


Рисунок П.8 – Ошибка при незаполненных полях при добавлении

**П3.2.4 Сортировка товаров под ролью «Администратор»**

Администратор каталога канцелярских товаров может выполнять сортировку товаров для удобного поиска и систематизации данных. Для сортировки необходимо нажать на выпадающий список и выбрать критерий сортировки (название товара, категория, цена, количество на складе и артикул), затем указать порядок сортировки – по возрастанию или убыванию, после чего нажать кнопку «Сортировать». Панель сортировки товаров представлена на рисунке (см. рис. П.9).

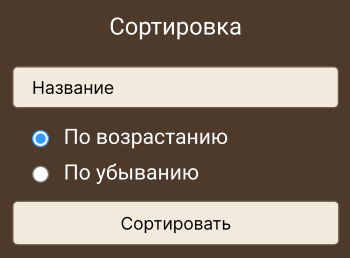


Рисунок П.9 – Панель сортировки

**П3.2.5 Поиск товаров под ролью «Администратор»**

Администратор каталога канцелярских товаров может выполнять поиск товаров для быстрого нахождения необходимой информации. Для поиска используется специальная панель, где из выпадающего списка выбирается критерий поиска (название товара, категория, количество на складе и артикул) после чего в текстовое поле ниже вводится искомое значение. Система автоматически отображает результаты поиска, соответствующие введенным критериям. Панель поиска товаров представлена на рисунке (см. рис. П.10).

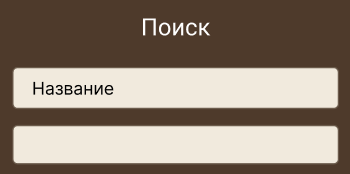


Рисунок П.10 – Панель поиска

**П3.2.6 Просмотр заказов под ролью «Администратор»**

Администратор каталога канцелярских товаров может просматривать список заказов клиентов через соответствующую форму управления. Для обновления информации о заказах необходимо нажать кнопку «Обновить». В таблице заказов отображается информация о всех поступивших заказах от клиентов, включая данные о заказчике, составе заказа, количестве товаров, общей стоимости и статусе выполнения. Если заказы отсутствуют, система выводит соответствующее сообщение (см. рис. П.11). Форма просмотра списка заказов представлена на рисунке (см. рис. П.12).

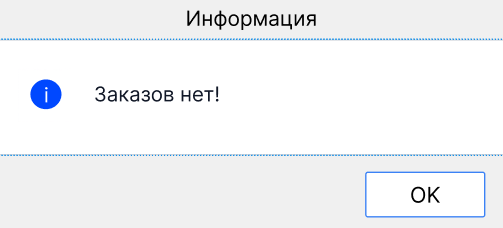


Рисунок П.11 – Сообщение отсутствия заказов



Рисунок П.12 – Список заказов

**П3.2.7 Панель кнопок под ролью «Администратор»**

Между таблицами расположены кнопки для работы с каталогом канцелярских товаров: «Прочитать из БД», «Сохранить в БД», «Выполнить все заказы», «Выполнить заказ», «Обновить заказы», «Очистить все» и «Удалить выбранное». Данные кнопки обеспечивают полный функционал управления товарами, заказами и серверными операциями в системе. Внешний вид кнопок управления представлен на рисунке (см. рис. П.13).

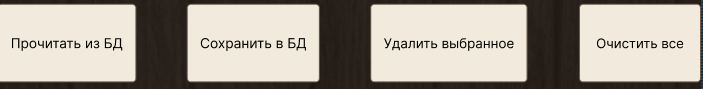




Рисунок П.13 – Кнопки управления

Описание функций кнопок:

– кнопка «Прочитать из БД» отвечает за загрузку данных о канцелярских товарах из базы данных, позволяя импортировать информацию о товарах в каталог, внешний вид представлен на рисунке (см. рис. П.13);

− кнопка «Сохранить в БД» отвечает за экспорт текущих данных каталога канцелярских товаров в базу данных для резервного копирования или передачи, внешний вид представлен на рисунке (см. рис. П.13);

− кнопка «Выполнить все заказы» отвечает за массовую обработку всех заказов канцелярских товаров одновременно, внешний вид представлен на рисунке (см. рис. П.13);

− кнопка «Выполнить заказ» отвечает за обработку и выполнение выбранного конкретного заказа из списка, внешний вид представлен на рисунке (см. рис. П.13);

− кнопка «Обновить заказы» отвечает за актуализацию информации о заказах клиента, внешний вид представлен на рисунке (см. рис. П.13);

− кнопка «Очистить все» отвечает за полное удаление всех данных из каталога товаров, внешний вид представлен на рисунке (см. рис. П.13);

− кнопка «Удалить выбранное» отвечает за удаление выбранного в списке товара, внешний вид представлен на рисунке (см. рис. П.13).

**П3.3 Регистрация пользователя под ролью «Клиент»**

Регистрация нового клиента происходит через создание учетных данных, которые будут определять уровень доступа в системе. После успешного заполнения регистрационной формы и сохранения информации в системе создается новая клиентская учетная запись с соответствующими привилегиями для управления каталогом. Окно регистрации представлено на рисунке (см. рис. П.14).

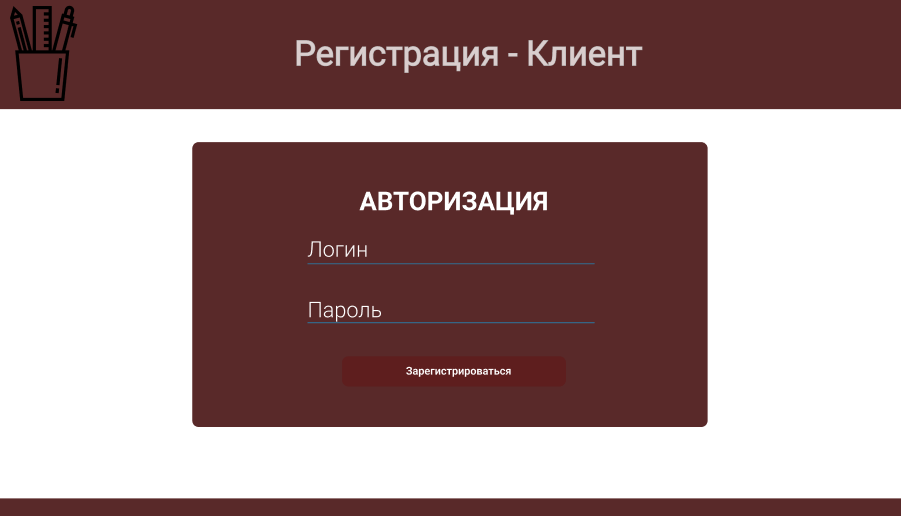


Рисунок П.14 – Окно регистрации клиента

Если клиент с такими учетными данными уже был зарегистрирован в системе ранее (имеет аналогичный логин и пароль), то при попытке повторной регистрации каталог не позволит создать дубликат учетной записи. В случае совпадения логина и пароля с уже существующим в базе данных клиентов появится сообщение об ошибке, уведомляющее о том, что данный логин уже занят и необходимо выбрать другой для успешной регистрации нового клиента в каталоге которое представлено на рисунке (см. рис. П.15).

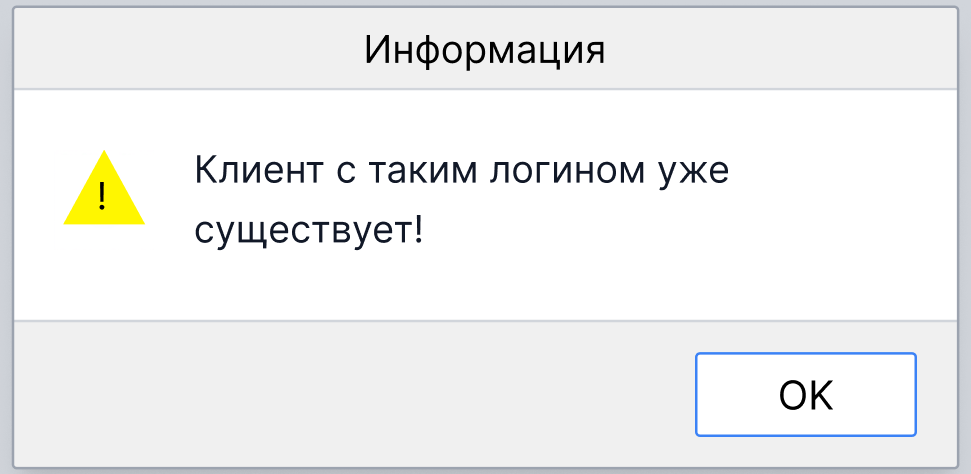


Рисунок П.15 – Предупреждение о существующем клиенте

**П3.4 Аутентификация пользователя под ролью «Клиент»**

Аутентификация клиента каталога канцелярских товаров происходит через ввод учетных данных, которые определяют клиентский уровень доступа к системе управления. После успешного ввода и проверки логина и пароля каталог канцелярских товаров готов к работе под клиентской ролью, предоставляя полный доступ к функциям управления товарами, категориями и другими возможностями, доступными клиенту каталога.

Окно входа и авторизации представлено на рисунке (см. рис. П.16)

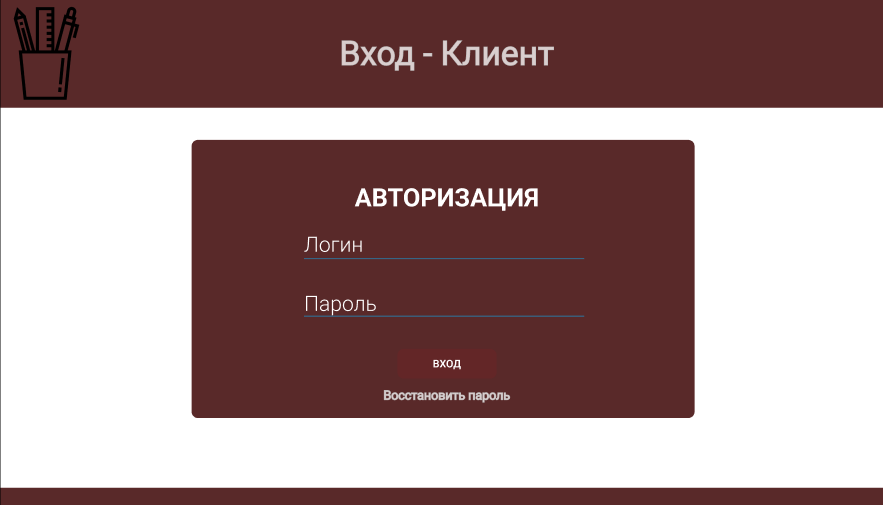


Рисунок П.16 – Окно входа

Если клиент был ранее зарегистрирован в системе управления каталогом канцелярских товаров (имеет свой логин и пароль), то он может войти в систему под своими учетными данными. Если введенные логин или пароль не совпадают с сохраненными в системе, то появится сообщение об ошибке, которое представлено на рисунке (см. рис. П.17).

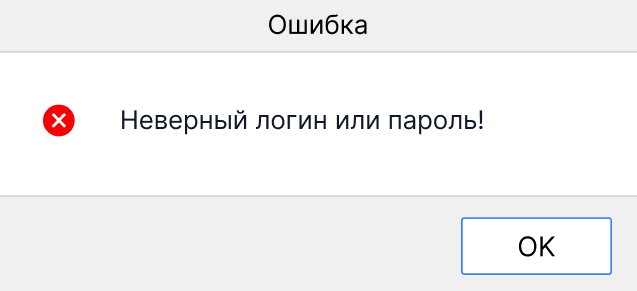


Рисунок П.17 – Ошибка при некорректном вводе данных

**П3.4.1 Просмотр списка товаров под ролью «Клиент»**

Клиент каталога канцелярских товаров может просматривать список товаров при взаимодействии с соответствующей формой управления каталогом, внешний вид которой представлен на рисунке (см. рис. П.6).

**П3.4.2 Добавление товаров в корзину и сохранение заказа под ролью «Клиент»**

Для добавления канцелярского товара в корзину клиент должен выбрать товар из списка и нажать кнопку «Добавить» в панели добавления товара в корзину (см. рис. П.18).

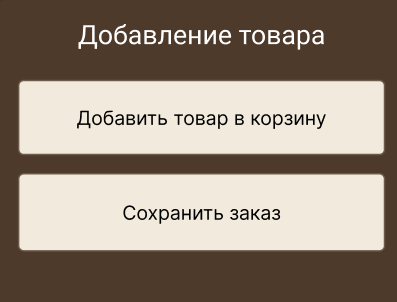


Рисунок П.18 – Добавление товара в корзину

Для сохранения канцелярского товара в корзину клиент должен нажать на кнопку «Сохранить заказ», предварительно добавив желаемые товары в корзину (см. рис. П.18).

Если клиент не добавил товары в корзину и захотел сохранить заказ то выводится соответствующее сообщение (см. рис. П.19).

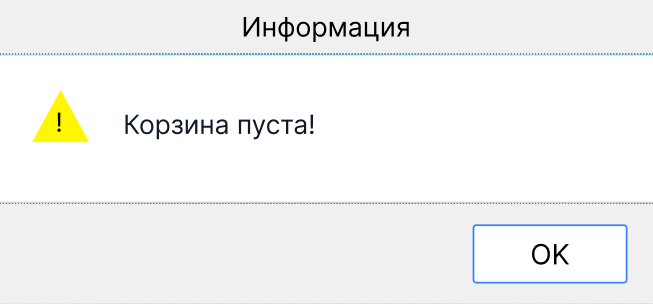


Рисунок П.19 – Сообщение при пустой корзине

**П3.4.3 Сортировка товаров под ролью «Клиент»**

Клиент каталога канцелярских товаров может выполнять сортировку товаров для удобного поиска и систематизации данных. Для сортировки необходимо нажать на выпадающий список и выбрать критерий сортировки (название товара, категория, цена, количество на складе и артикул), затем указать порядок сортировки – по возрастанию или убыванию, после чего нажать кнопку «Сортировать». Панель сортировки товаров представлена на рисунке (см. рис. П.9).

**П3.4.4 Поиск товаров под ролью «Клиент»**

Клиент каталога канцелярских товаров может выполнять поиск товаров для быстрого нахождения необходимой информации. Для поиска используется специальная панель, где из выпадающего списка выбирается критерий поиска (название товара, категория, количество на складе и артикул) после чего в текстовое поле ниже вводится искомое значение. Система автоматически отображает результаты поиска, соответствующие введенным критериям. Панель поиска товаров представлена на рисунке (см. рис. П.10).

**П3.4.5 Панель кнопок под ролью «Клиент»**

Между таблицами расположены кнопки для работы с каталогом канцелярских товаров: «Прочитать из файла», «Удалить из корзины», «Очистить корзину». Данные кнопки обеспечивают полный функционал управления заказами и файловыми операциями в системе. Внешний вид кнопок управления представлен на рисунке (см. рис. П.20).





Рисунок П.20 – Кнопки управления

Описание функций кнопок:

− кнопка «Удалить выбранное из корзины» отвечает за удаление выбранного в корзине товара, внешний вид представлен на рисунке (см. рис. П.20);

− кнопка «Сменить пароль» отвечает за смену пароля клиента (см. рис. П.20);

− кнопка «Очистить всю корзину» отвечает за полное удаление всех данных из корзины, внешний вид представлен на рисунке (см. рис. П.20).

**П3.4.6 Смена пароля под ролью «Клиент»**

Клиент каталога канцелярских товаров может изменить свой пароль для обеспечения безопасности аккаунта. Для изменения пароля используется специальная форма, где последовательно вводятся текущий пароль, новый пароль и подтверждение нового пароля. После заполнения всех полей клиент нажимает кнопку «Изменить» для применения изменений. Система проверяет корректность введенных данных и подтверждает успешное изменение пароля. Форма смены пароля представлена на рисунке (см. рис. П.21).

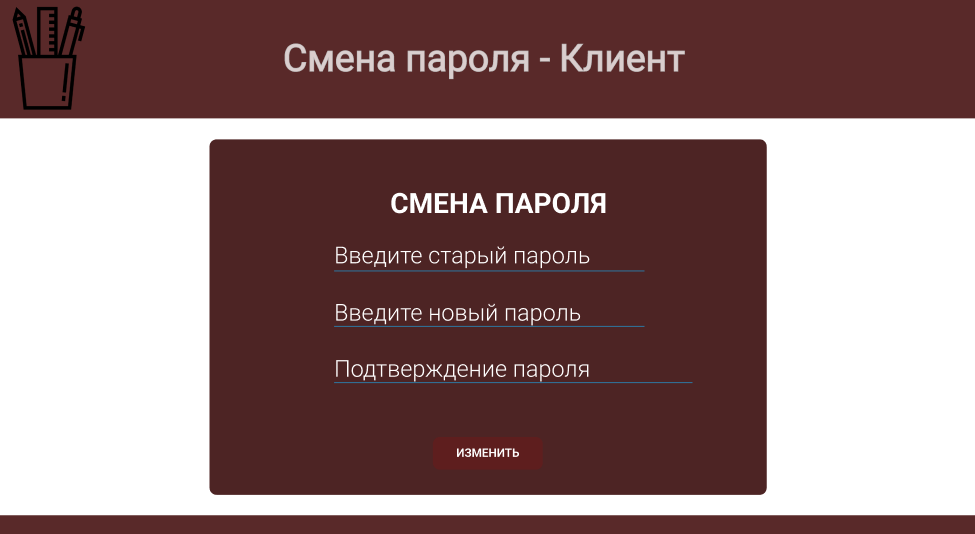
****

Рисунок П.21 **–** Форма сменя пароля клиента

**П3.4.7 Выбор кол-ва товара для заказа под ролью «Клиент»**

Клиент каталога канцелярских товаров может выбрать необходимое количество товара для добавления в заказ. На форме выбора количества отображается название товара, доступное количество на складе и поле для ввода требуемого количества. После введения нужного количества клиент нажимает кнопку «OK» для подтверждения выбора или кнопку «Отмена» для отказа. Система проверяет, что введённое количество не превышает доступное на складе, и добавляет товар в заказ. Форма выбора количества товара представлена на рисунке (см. рис. П.22).

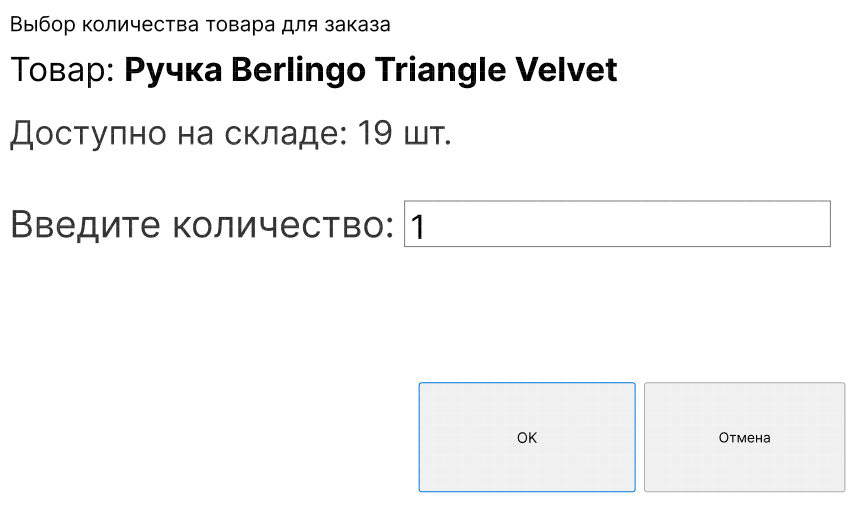
****

Рисунок П.22 **–** Форма выбора количества товара для заказа

Если введенное клиентом количество товара превышает доступное количество на складе то выводится соответствующая ошибка (см. рис. П.23).

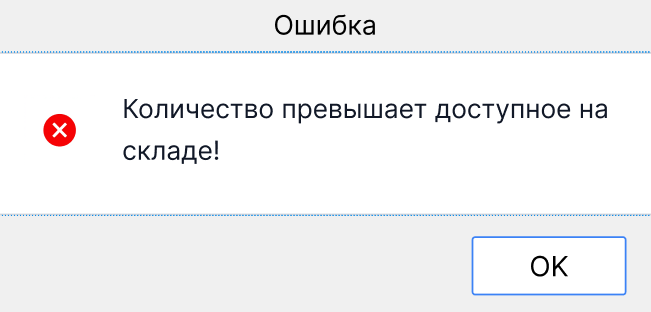


Рисунок П.23 – Ошибка превышения доступного кол-ва на складе товара

**П3.4.8 Восстановление пароля клиента**

Клиент каталога канцелярских товаров может восстановить свой пароль в случае его забывания. Для восстановления пароля клиент вводит свой логин в соответствующее поле на форме восстановления. После введения логина клиент нажимает кнопку «Изменить» для инициирования процесса восстановления пароля. Система проверяет наличие учетной записи с указанным логином и отправляет инструкции по восстановлению пароля. Форма восстановления пароля представлена на рисунке (см. рис. П.24).

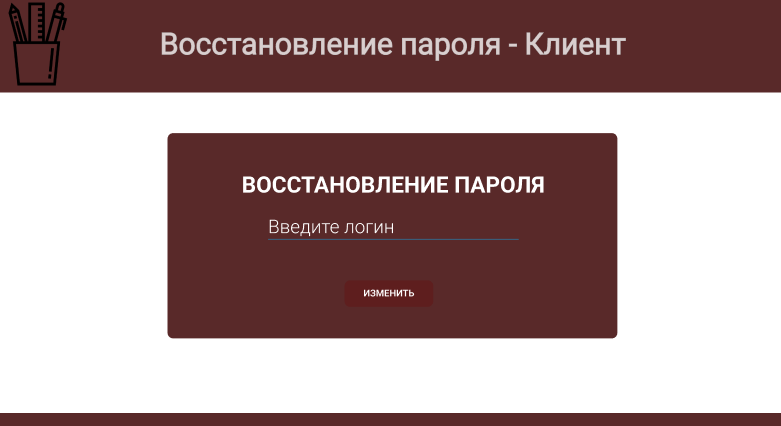
****

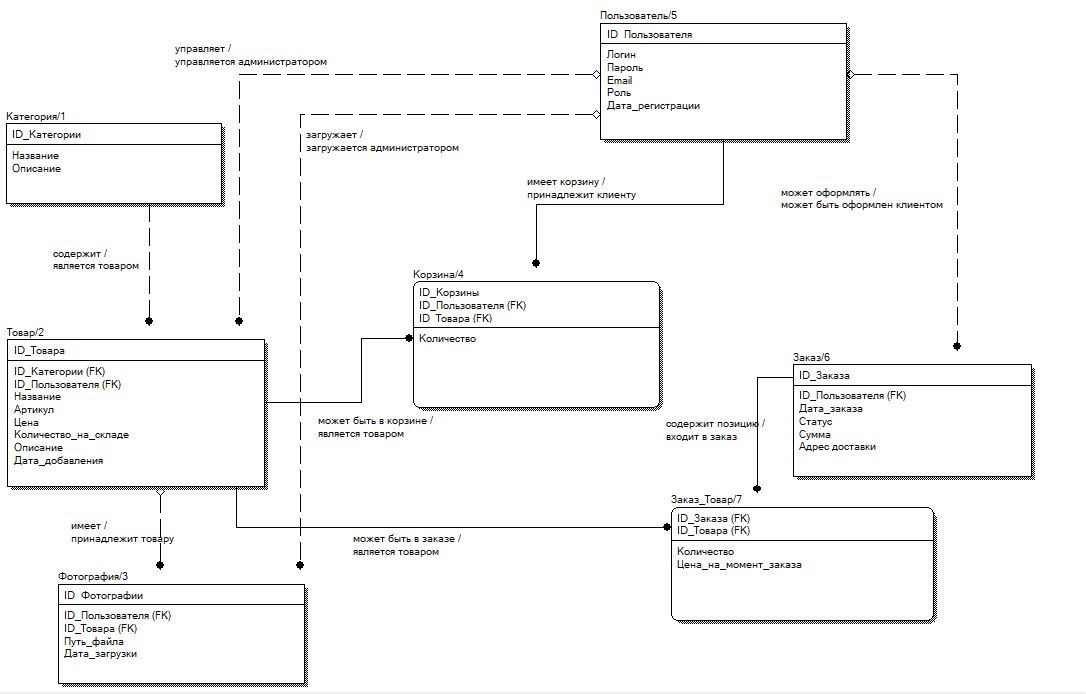
Рисунок П.24 **–** Форма восстановления пароля клиента

В таблице (табл. П.1) приведены основные сообщения оператору при попытке неправильного ввода чего-либо.

Таблица П.1 – Сообщения оператору

|  |  |
| --- | --- |
| Текст сообщения | Причина появления |
| Неверный ключ-доступа | Отображается в случае, если введенный ключ доступа администратора не совпадает с выданным главным администратором |
| Неверный логин или пароль | Отображается в случае, если введенные учетные данные не совпадают с зарегистрированными в системе |
| Пожалуйста заполните все поля | Появляется, когда одно или несколько обязательных полей в форме авторизации, регистрации или добавления товара остаются пустыми |
| Клиент с таким логином уже существует | Отображается в случае, если введенный логин уже зарегистрирован в системе при попытке регистрации нового клиента |
| Корзина пуста | Появляется, когда клиент пытается сохранить заказ без добавленных в корзину товаров |
| Количество превышает доступное на складе | Отображается в случае, если введенное клиентом количество товара превышает доступное количество на складе |
| Заказов нет | Отображается в том случае, если администратор нажимает кнопку «Обновить заказы», а заказов от клиентов нету |

# ПРИЛОЖЕНИЕ Р (справочное) Диаграмма сущность-связь (логическая модель)



# ПРИЛОЖЕНИЕ С (справочное) Диаграмма сущность-связь (физическая модель)

