МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РФ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет вычислительной техники

Кафедра МОиПЭВМ

Направление подготовки 09.03.02 «Информационные системы и технологии»

Направленность Информационные системы и технологии

БАКАЛАВРСКАЯ РАБОТА

на тему

Разработка программного комплекса для онлайн-торговли автозапчастями

Студент _____ Петриш Анна Константиновна
(подпись, дата) (ФИО полностью)

Руководитель _____ Попова Н.А.
(подпись, дата) (фамилия, инициалы)

Нормоконтролер _____ Такташкин Д.В.
(подпись, дата) (фамилия, инициалы)

Работа допущена к защите (протокол заседания кафедры от 03.06.2026 г. № 12)

Заведующий кафедрой _____ Козлов А.Ю.
(подпись) (фамилия, инициалы)

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

Секретарь ГЭК _____ Попова Н.А.
(подпись) (фамилия, инициалы)

Пенза, 2026

Федеральное государственное бюджетное образовательное учреждение высшего образования
ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ПОЛИТЕХНИЧЕСКИЙ ИНСТИТУТ

Факультет	*Кафедра*
Вычислительной техники	МО и ПЭВМ

«Утверждаю»
Заведующий кафедрой
_____А.Ю. Козлов
«26» декабря 2025 г

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
БАКАЛАВРА

1. Студент Петриш Анна Константиновна гр. 22ВИ1 факультета ВТ
<Фамилия, имя, отчество полностью>

2.Тема работы «Разработка программного комплекса для онлайн-торговли автозапчастями».

Тема утверждена приказом ПГУ №1583/о от "30" декабря 2025 г.

3. Руководитель работы Попова Н.А.

4. Задание на работу (назначение разработки, исходные данные и т.п.).
Назначение: автоматизация деятельности магазина автозапчастей.

Основные функции системы:
1. возможность работы с базой автозапчастей (просмотр каталога товаров, их характеристик, цены, описания, привязка к маркам и моделям автомобилей);
2. возможность работы с базой пользователей (регистрация новых пользователей, авторизация, просмотр личных данных, разграничение ролей «покупатель» и «администратор»).
3. формирование заказа покупателем (просмотр каталога, фильтрация товаров по категориям, производителям маркам и моделям автомобилей, добавление в корзину, оформление заказа).
4. возможность отслеживания статуса заказа (не оформлен, оформлен доставлен).
5. формирование аналитических отчетов.
6. ведение складского учета в приложении 1С (резервирование и списание товаров при оформлении заказов, контроль актуальных остатков, формирование приходных и расходных документов).

Техгология разработки: многоуровневая клиент-серверная архитектура с интеграцией внешнего приложения и системы складского учета.
Язык программирования: C# (для разработки клиентского приложения и серверного API), встроенный язык программирования 1С:Предприятие (для подсистемы складского учета).
Среда программирования: Microsoft Visual Studio 2022 (для WinForms-приложения и API) + технологическая платформа 1С:Предприятие 8.3 (для конфигурации складского учета).

5. Перечень подлежащих разработке вопросов.
1. анализ предметной области и постановка задачи;
2. проектирование программного комплекса;
3. разработка программного комплекса;
4. тестирование программного комплекса;
5. оформление ВКР.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1	Анализ предметной области и требований к системе	10%	07.04.2026 – 13.04.2026	
2	Проектирование программных средств	20%	14.04.2026 – 25.04.2026	
3	Реализация программных средств	30%	27.04.2026 – 11.05.2026	
4	Тестирование программных средств	20%	12.05.2026 – 21.05.2026	
5	Оформление пояснительной записки	20%	22.05.2026 – 02.06.2026	

Дата выдачи задания «26» декабря 2025г.

Руководитель бакалаврской работы _____ 26.12.2025 г.
(подпись, дата)

Задание к исполнению принял студент _____ 26.12.2025 г.
(подпись, дата)

Работу к защите допустить
Декан ФВТ _____ Л.Р. Фионова
(подпись, дата)

Реферат

Пояснительная записка содержит 96 листов, 43 рисунка, 23 таблицы, 20 использованных источников и 3 приложения.

КОНСОЛЬНОЕ ПРИЛОЖЕНИЕ, WINDOWS FORMS, 1С ПРЕДПРИЯТИЕ 8.3, C#, OPENAPI, MS SQL SERVER

Объектом исследования является процесс онлайн-торговли автозапчастями и его автоматизация.

Предметом исследования являются программные средства и методы разработки программного комплекса, обеспечивающего регистрацию и авторизацию пользователей, просмотр и фильтрацию товаров, формирование корзины и оформление заказов, ведение складского учета товаров.

Создание программного комплекса выполнялось с использованием двух языков: C# и встроенного языка платформы 1С. Инструментарий разработки включал Project Libre, Visual Paradigm версии 17.2, среду Visual Studio, а также конфигуратор «1С:Предприятие 8.3».

Итогом выпускной квалификационной работы стал готовый программный продукт. Он объединяет фронтальную и серверную составляющие онлайн-магазина по продаже автозапчастей, а также модуль управления складскими остатками.

Содержание

Введение 7
1 Анализ предметной области и постановка задачи 10
1.1 Постановка задачи 10
1.2 Анализ информационных систем аналогичного назначения 13
1.3 Выбор технологий проектирования системы 16
1.4 Выбор инструментальных средств 18
1.5 Анализ требований к разрабатываемому программному комплексу 20
2 Проектирование информационной системы 22
2.1 Анализ бизнес-процессов программного комплекса 22
2.2 Проектирование модели базы данных 25
2.3 Архитектурное проектирование программного комплекса 27
2.3.1 Высокоуровневое проектирование программных средств 29
2.3.2 Проектирование серверной части интернет-магазина 30
2.3.3 Проектирование клиентской части интернет-магазина 33
2.3.4 Проектирование системы складского учета 34
2.4 Планирование разработки и оценка бюджета 34
3 Проектирование пользовательского интерфейса 37
3.1 Проектирование интерфейса интернет-магазина 37
3.2 Проектирование интерфейса системы складского учета 41
4 Реализация 46
4.1 Реализация серверной части интернет-магазина 46
4.2 Реализация клиентской части интернет-магазина 49
4.3 Реализация системы складского учета 57
5 Контроль качества автоматизированной информационной системы 67
5.1 Функциональное тестирование 67
5.1.1 Классы эквивалентности 67
5.1.2 Граничные условия 68
5.2 Нефункциональное тестирование 69
5.3 Анализ качества кода 73
Заключение 77
Список использованных источников 78

Приложение А (справочное) Глоссарий 80
Приложение Б (обязательное) Листинг программы 84
Приложение В (рекомендуемое) Описание атрибутов модели базы данных..... 93

Введение

Продажа автозапчастей через интернет – быстрорастущий бизнес. Покупателям и сотрудникам автосервисов нужно иметь возможность быстро находить детали, видеть их цену и наличие, а также удобно оформлять заказы. Сейчас все больше людей покупают товары онлайн. Количество автомобилей растет, и пользователи хотят, чтобы учет товаров был прозрачным, а заказы обрабатывались быстро.

На сегодняшний день на рынке представлено большое количество решений, предназначенных для автоматизации интернет-торговли и складского учета. Тем не менее, если внимательно присмотреться к этому многообразию, становится очевидно, что крайне редко встречается такой программный продукт, который одновременно удовлетворял бы двум требованиям.

С одной стороны, он должен обладать интуитивно понятным интерфейсом для поиска автомобильных запчастей по разным параметрам (включая категории товаров и марки машин).

С другой стороны, в него обязана быть встроена полноценная профессиональная система складского учета на базе платформы 1С. Благодаря такому симбиозу обеспечивается достоверность информации об остатках на складе и безошибочное оформление заказов.

Современная торговля автозапчастями через интернет сегодня полностью зависит от автоматизации на базе актуальных цифровых технологий. Любое ПО, которое используется для подобной онлайн-торговли, обязано включать ряд ключевых функций. А именно: ведение товарной базы (атрибуты, категории деталей), плюс полноценное управление корзиной покупателя и всеми стадиями обработки заказов. При этом критически важно обеспечить эффективное взаимодействие с системой складского учета (1С), чтобы предоставлять пользователям только достоверную информацию о наличии товаров и избегать ситуаций с невыполненными заказами.

Предмет работы: разработка программного комплекса для автоматизации онлайн-торговли автозапчастями, объединяющего десктопное приложение для покупателей и администраторов с системой складского учета.

Объектом работы является: автоматизация онлайн-торговли автозапчастями.

Основная цель работы – спроектировать и реализовать программный комплекс, предназначенный для продажи автозапчастей через интернет. Ключевые функции системы: разграничение доступа между администраторами и обычными пользователями, удобная фильтрация номенклатуры, механизм формирования заказов и корзины, а также синхронизация со складской учетной системой на базе 1С.

Для достижения поставленной цели требуется решить ряд промежуточных задач:

– изучить и систематизировать требования к будущему решению;

– проанализировать несколько существующих аналогов, работающих в той же предметной области;

– обосновать выбор технологического стека и инструментов разработки;

– детально разобрать бизнес-процессы, связанные с интернет-продажами и складским учетом;

– построить набор UML-диаграмм, отражающих логику работы комплекса;

– составить план разработки и рассчитать ее примерный бюджет;

– разработать создать программный комплекс в соответствии со сформулированными требованиями;

– провести тестирование готового продукта.

Итогом выполненной работы стал программный комплекс для онлайн-торговли автозапчастями, который включает в себя:

– базу данных автозапчастей с возможностью добавления и изменения информации о товарах, их категориях, а также совместимости с конкретными моделями автомобилей;

– ролевую модель с двумя типами учетных записей (администратор и обычный пользователь), где для каждой роли строго заданы допустимые операции и объем доступа к данным;

– возможность просмотра товаров, фильтрации по категориям и маркам автомобилей, добавления в корзину и оформления заказов через API (Swagger) в приложении на C#;

– интеграцию с приложением 1С, обеспечивающую складской учет при оформлении заказов;

– формирование отчетов по заказам, остаткам и движению товаров.

1 Анализ предметной области и постановка задачи

1.1 Постановка задачи

Сбор и последующая обработка данных – главное назначение любой информационной системы (ИС). При этом система объединяет сразу несколько типов ресурсов – организационные, технические и финансовые, чтобы обеспечить надежное хранение информации и эффективный обмен ею.

Если говорить конкретно об онлайн-торговле автозапчастями, то внедрение ИС и сопутствующих технологий дает возможность заметно ускорить и упростить целый ряд рутинных операций. В их число входят:

- поиск нужной запчасти;
- проверка совместимости автозапчасти с автомобилем;
- возможность поиска запчастей по производителю и категории;
- добавление позиции в виртуальную корзину;
- оформление заявки на покупку;
- оперативная проверка складских остатков.

Сейчас в интернете продают автозапчасти, и для этого используют много разных компьютерных программ. Можно выделить три главные вещи, помогающие в работе бизнеса.

Существуют программы, которые хранят всё про товары: название детали, цену, к каким машинам подходит. Они же помогают добавлять новые запчасти, исправлять ошибки в списках, сортировать, к тому же ещё эти программы запоминают покупателей и следят за заказами.

Также есть специальные системы для работы с клиентами (их называют CRM). Они нужны, чтобы помнить, кто что покупал раньше, что позволяет анализировать предпочтения и историю каждого покупателя.

А есть программы, которые делают только одно конкретное дело, например, сами подбирают нужную деталь, если ввести марку машины и её модель. Это очень удобно: не надо самому искать, программа всё сделает автоматически.

Для предприятий, реализующих автозапчасти через интернет, в настоящее время существует довольно много готовых программных продуктов. Все их допустимо условно разделить на три основные группы.

К первой группе относят корпоративные информационные системы (КИС), а также решения для планирования ресурсов предприятия (так называемые ERP-системы) и программное обеспечение, отвечающее за складской учет.

Вторая группа объединяет информационно-аналитические системы (ИАС). Их главная задача – проводить анализ объемов продаж и на его основе прогнозировать будущий спрос.

Третью группу составляют простые учетные системы, которые не обладают избыточной сложностью и ориентированы на базовые задачи.

Крупные решения класса ERP и полноценные корпоративные информационные системы (КИС) редко оказываются удачным выбором для сегмента малого и среднего предпринимательства. Причины связаны с высокими финансовыми затратами на внедрение, необходимостью выделять большие временные и кадровые ресурсы, а также со сложностью освоения такого ПО рядовыми сотрудниками. Плюс ко всему, универсальные ERP-системы демонстрируют слабую гибкость при попытке подстроить их под специфику продаж автозапчастей.

Крупные универсальные системы для управления предприятиями не могут быть использованы в рассматриваемой сфере по нескольким причинам. Главной из них является то, что они не способны работать со специфическими задачами автозапчастей. Кроме этого, такие системы рассчитаны на большие объёмы, а это не соответствует масштабам малого бизнеса.

Если говорить об информационно-аналитических системах, то они дешевле и проще внедряются по сравнению с теми же ERP. Но тут есть свой минус. Чаще всего такие решения продаются не отдельно, а как часть дорогих управленческих комплексов, поэтому небольшие компании почти никогда их не используют.

У маленьких предприятий, которые продают автозапчасти через интернет, есть свои характерные черты:

– они работают в очень конкретной, узкой нише;

– у них мало свободных денег и сотрудников;

– набор функций, который им нужен, строго ограничен;

– персонал необходимо обучить работе с системой в максимально короткие сроки.

С учётом всего перечисленного самым удобным вариантом для малого и среднего бизнеса становится подход, при котором заказывают полностью готовое автоматизированное решение. Его делают сразу под все задачи, без необходимости что-то дорабатывать или покупать отдельно. Под этот формат подходят несложные учетные системы. Они стоят недорого, закрывают все ключевые потребности бизнеса и при этом не содержат избыточных, неиспользуемых на практике модулей.

За счет такого подхода удается улучшить сервис для покупателей: система становится понятной и удобной в ежедневной работе, поиск товара и оформление заказа происходят быстрее, а отсутствие лишних опций не отвлекает сотрудников.

Ключевая задача при создании информационной системы – наладить комфортную работу с данными в режиме реального времени. Когда информация накапливается и структурируется, появляется возможность анализировать продажи в различных временных срезах, определять наиболее ходовые позиции автозапчастей, строить прогнозы спроса. Но главное – только такая система дает оперативный доступ к сведениям о складских остатках, текущих ценах и оформленных заказах. Эффективность здесь напрямую зависит от того, насколько грамотно организовано хранение данных и их обработка. В итоге пользователи затрачивают намного меньше времени для поиска необходимой информации. Внедрение информационных систем в процесс интернет-продаж автозапчастей на любом этапе неизбежно ведет к росту общей эффективности бизнеса.

1.2 Анализ информационных систем аналогичного назначения

В настоящее время доступно довольно много решений для автоматизации интернет-продаж и учета складских запасов. Немало онлайн-витрин добились коммерческого успеха как раз за счет внедрения подобных автоматизированных продуктов.

Среди владельцев российских интернет-магазинов особенно востребована конфигурация «1С:Предприятие. Управление торговлей». Это решение стабильно входит в число лидеров по частоте использования.

Высокий спрос на эту платформу объясняется тем, что она закрывает широкий круг задач: наполнение и ведение товарной номенклатуры, контроль складских остатков, прием и сопровождение клиентских заказов, взаимодействие с покупателями, а также сбор и анализ данных о продажах.

Рассматриваемое программное решение обладает несколькими ключевыми характеристиками, которые выделяют его среди аналогов. Во-первых, оно допускает доработку и дополнение функционала за счёт установки новых модулей и гибкой настройки параметров под нужды конкретного торгового предприятия. Во-вторых, система поддерживает интеграцию с различными электронными площадками, что расширяет каналы сбыта. В-третьих, пользовательский интерфейс поддаётся индивидуализации. И наконец, в системе предусмотрено чёткое разделение доступа между работниками, что позволяет ограничить полномочия в зависимости от должностных обязанностей.

Важно отметить, что архитектура продукта построена по модульному принципу. В любой момент можно отключить те блоки, которые в данный момент не задействованы, и активировать другие, если возникнет такая необходимость. Такой подход делает систему гибкой и адаптируемой к меняющимся условиям ведения бизнеса.

Работоспособность и взаимодействие всех перечисленных компонентов обеспечиваются за счёт следующего:

– ведение каталога товаров с возможностью указания характеристик, совместимости с автомобилями, артикулов и цен;

– автоматизация обработки заказов;

– ведение складского учета, что обозначает резервирование товаров под заказы и контроль остатков;

– формирование отчетности с информацией о продажах, остатках, прибыли;

– интеграция с сайтом интернет-магазина через API (обмен данными о товарах, ценах, остатках, заказах);

– работа с разными типами цен (розничная, оптовая, специальные предложения);

– управление клиентской базой и историей покупок.

Для разных категорий сотрудников есть разные роли, которые помогают избежать ошибок и ограничивают доступ к определенным функциям.

Система «1С:Управление торговлей» представляет собой эффективный инструмент для ведения управленческого и бухгалтерского учёта в торговых предприятиях. Её основное назначение – сбор, систематизация и структурирование всей входящей информации с целью последующего анализа. Все данные аккумулируются в единой базе и выводятся в виде наглядных отчётных форм, что значительно упрощает процесс принятия управленческих решений.

Благодаря отчетам есть возможность анализировать данные:

– продаж за период, по какому-либо товару, какой-либо категории или клиенту;

– движений товаров на складах;

– популярных запчастей и категорий, к которым относятся запчасти;

– дебиторских задолженностей и задолженностей клиентов.

К достоинствам системы можно отнести:

– широкое распространение (большое сообщество, много готовых решений) и надежность;

– возможность полной автоматизации учета – от поступления товара до финальной отчетности;

– система помогает контролировать работу сотрудников и избегать ошибок в учете.

Следующий анализируемый программный продукт – «МойСклад». Это облачная система для учета товаров, заказов и продаж, которая подходит для малого и среднего бизнеса.

Данное приложение работает через браузер, не требует установки на компьютер и доступно с любого устройства через интернет. Оно также может интегрироваться с сайтами интернет-магазинов, маркетплейсами и службами доставки.

Стоимость продукта начинается от 2000 – 3000 рублей в месяц в зависимости от количества заказов и пользователей.

Функциональные возможности:

– учет товаров, остатков и движений на складе;
– работа с заказами клиентов;
– печать товарных чеков и накладных;
– интеграция с сайтом для выгрузки товаров и остатков;
– работа с маркированными товарами и ЕГАИС (при необходимости);
– формирование отчетов по продажам, прибыли, остаткам.

Преимущества программы:

– невысокая стоимость и отсутствие затрат на сервер;
– простота использования и быстрый старт;
– работа из любого места, где есть интернет;
– наличие бесплатного тарифа для малых объёмов продаж.
– регулярные обновления и техническая поддержка.

Недостатки программы:

– ограниченный функционал для крупных компаний;
– зависимость от интернета;
– сложность настройки под специфические задачи;
– ежемесячная плата (не разовая покупка).

Сравнительный анализ «1С:Управление торговлей», «МойСклад» и разрабатываемого программного комплекса представлен в таблице 1.

Таблица 1 – Сравнительный анализ систем аналогичного назначения

Критерий	**«1С:Управление торговлей»**	**«МойСклад»**	**Разрабатываемый комплекс**
1	2	3	4
Качество интерфейса	среднее	высокое	высокое
Возможности функционала	высокие	средние	средние
Доступность (цена)	низкая	высокая	высокая
Трудоемкость внедрения	высокая	низкая	низкая
Интеграция с сайтом	сложная (через сторонние решения)	простая (API)	простая (API Swagger)
Работа со складом в 1С	есть	ограниченно	есть (через отдельное приложение 1С)

Сравнительный обзор существующих систем для интернет-торговли автокомпонентами показал их сильные и слабые стороны. Создаваемое приложение предлагает набор функций, который сопоставим с лидерами рынка в сфере автоматизации продаж автотоваров, но при этом имеет более низкую стоимость внедрения, простой интерфейс и удобную интеграцию сайта со складским учетом в 1С.

1.3 Выбор технологий проектирования системы

В процессе создания программного комплекса использовался ряд инструментов, предназначенных для визуального проектирования и моделирования. В частности, применялись такие решения, как ArgoUML и Visual Paradigm Community. Первый инструмент даёт возможность строить диаграммы высокой сложности, а второй – представляет собой полноценную среду, поддерживающую язык UML, и используется для проектирования различных приложений и программных модулей [1].

Для начала стоит пояснить, что UML (Unified Modeling Language) – это унифицированный язык моделирования, относящийся к графическим языкам, он нужен для того, чтобы наглядно изображать в виде диаграмм различные процессы, связи между объектами и логику работы будущей системы.

Visual Paradigm for UML – это платформа, ориентированная на объектно-ориентированное проектирование с использованием UML. Для некоммерческих проектов она предоставляется бесплатно. Среди её возможностей имеются такие, как полная поддержка версии спецификации UML 2.1, подключение дополнительных плагинов, вставка графических изображений, а также базовые функции вывода на печать и создания документации. Еще в этом продукте предусмотрен специальный механизм расширения, позволяющий адаптировать стандартный UML под конкретные задачи заказчика, эта особенность выгодно отличает Visual Paradigm for UML от подходов, основанных на структурном моделировании.

ArgoUML является инструментом, распространяемым как свободное программное обеспечение с открытым исходным кодом. Главная цель его использования – создание диаграмм высокого качества, способных отражать бизнес-логику и соответствовать требованиям статистического анализа. Благодаря тому, что ArgoUML написан на Java SE, он работает на любой операционной системе, где установлена поддержка Java. Его самая сильная сторона в том, что он имеет широкий выбор типов диаграмм, с помощью которых можно подробно описывать разные стороны проектируемой системы. Что в итоге позволяет разработчику получить полное и целостное представление о том, как устроена система и как она должна функционировать.

В ArgoUML интерфейс выполнен интуитивно понятно и просто. Рабочая область поделена на зоны: справа располагается область редактирования, а слева – иерархическое дерево проекта, нижняя же панель служит для обмена информацией. Такая компоновка заметно упрощает освоение и повседневное использование графической среды.

1.4 Выбор инструментальных средств

При проектировании современных информационных систем сегодня задействуется множество разнообразных технологических средств. Конкретно в данном проекте при создании программного решения для онлайн-продажи автозапчастей встала задача обоснованного выбора технологий. Для ее решения были сформулированы и проанализированы несколько определяющих критериев.

В итоговый перечень требований вошли:

– экономическая доступность, затраты на создание автоматизированной системы не должны превышать разумных пределов;

– скорость и простора реализации, технологический стек обязан обеспечивать быструю и нетрудоемкую разработку;

– гибкость при сопровождении, предполагается легкая корректировка системы уже после ее запуска, на этапе реальной эксплуатации;

– интеграционная способность, необходимо объединить базу данных, клиентское приложение и учетную складскую подсистему;

– интерфейс и эргономика, широкий набор UI-возможностей.

С учетом вышеперечисленных пунктом, решилось, что создаваться программный комплекс будет на основе трех ключевых технологий: база данных Microsoft SQL Server, приложение на языке C# в среде разработки Microsoft Visual Studio с API (Swagger) и система для ведения складского учета в «1С:Предприятие 8.3».

База данных Microsoft SQL Server выбрана в качестве основного хранилища информации об автозапчастях и их категориях, о пользователях и их заказах, производителях, складах и других объектах. Microsoft SQL Server является надежной и производительной системой управления базами данных, обеспечивающей быстрый доступ к информации, целостность данных и безопасность [2]. Она хорошо масштабируется и подходит для работы с большими объемами номенклатуры автозапчастей.

Приложение на языке C# с API (Swagger) разрабатывается в среде Microsoft Visual Studio [3]. C# – это современный, мощный и удобный язык программирования, который позволяет быстро создавать приложения с удобным графическим интерфейсом. API (программный интерфейс) на базе Swagger обеспечивает удобное взаимодействие между клиентской частью (сайтом или мобильным приложением) и сервером. Через API пользователи и администраторы получают доступ к каталогу товаров, фильтрации, корзине и оформлению заказов [4].

Для ведения складского учета в рамках данного проекта была выбрана платформа «1С:Предприятие 8.3». Данное решение представляет собой программную экосистему, включающую базовое ядро и множество отраслевых приложений, созданных на его основе [5]. Благодаря такой архитектуре появляется возможность выстроить эффективный складской учет на единой технологической базе.

Одно из главных достоинств платформы «1С:Предприятие 8.3» – гибкость подстройки под потребности компаний разного масштаба. С ее помощью можно оптимизировать рабочие процессы в организациях с любым количеством автоматизированных рабочих мест, что одинаково хорошо подходит и небольшим фирмам, и крупным предприятиям.

Немаловажно и то, что платформа способна функционировать в разных операционных средах. Дополнительный плюс – активное развитие продукта: пользователи регулярно получают обновления, которые расширяют имеющиеся возможности и повышают производительность в соответствии с актуальными рыночными запросами. Совокупность перечисленных характеристик позволяет считать «1С:Предприятие 8.3» оптимальным выбором для организации складского учета в разрабатываемом программном комплексе.

Таким образом, выбранный набор технологий позволяет создать полноценную автоматизированную систему для онлайн-торговли автозапчастями, где клиенты удобно оформляют заказы через приложение, а складской учет ведется в системе 1С. Все компоненты взаимодействуют между

собой через API и общую базу данных, что обеспечивает актуальность данных и надежность работы.

1.5 Анализ требований к разрабатываемому программному комплексу

В функционал программного комплекса для онлайн-торговли автозапчастями должны входить как стандартные модули (каталог, администрирование пользователей, прием заказов), так и дополнительные – например, накопление статистики продаж с возможностью последующего анализа. При этом само решение предъявляются требования простоты и понятности для обеих категорий пользователей [6]. Эффект повышения эффективности труда сотрудников обеспечивается благодаря автоматизации процедур оформления заказов, оперативного поиска товарных позиций и удобного управления складом.

Основные функции программного комплекса:

- просмотр каталога автозапчастей с подробной информацией о каждом товаре (название, описание, цена, совместимость с автомобилями);
- фильтрация товаров по категориям и маркам автомобилей;
- добавление товаров в корзину и оформление заказов;
- разделение прав доступа: обычные пользователи (только покупка) и администраторы (управление товарами, заказами и пользователями);
- складской учет в 1С, контроль остатков, резервирование товаров при оформлении заказа, списание со склада;
- создание отчетных форм, позволяющих анализировать деятельность магазина.

Дополнительные функции программного комплекса:

- удобный поиск и фильтрация запчастей по марке, модели, году выпуска, категории и артикулу;
- отображение актуальных остатков на складе при просмотре товара (данные поступают из 1С);

- возможность отслеживать статус заказа (принят, собран, доставлен);
- формирование итоговой стоимости заказа с учетом возможных скидок;
- автоматический подбор совместимых запчастей по заданным параметрам автомобиля.

Качество интерфейса напрямую влияет на то, насколько комфортно люди будут работать с многофункциональной системой. Экран должен быть понятным, удобным и не раздражать при ежедневном использовании. Надписи на элементах управления, названия разделов и их внешний вид обязаны точно соответствовать выполняемым действиям. Покупатель или сотрудник не должен ломать голову над тем, как отыскать товар или оформить сделку, каждое действие следует сделать интуитивно очевидным.

Исходя из этого, создаваемый комплекс следует реализовать с учетом каждого из перечисленных требований: удобный каталог и поиск для покупателей, мощные инструменты управления для администраторов, надежный складской учет в 1С, а также понятный и приятный интерфейс для всех категорий пользователей.

2 Проектирование информационной системы

2.1 Анализ бизнес-процессов программного комплекса

На этапе проектирования в качестве одной из ключевых задач всегда считается построение модели бизнес-процессов организации. Говоря иными словами, требуется разработать такую модель, которая максимально подробно и адекватно отражала бы предметную область, включая сведения обо всех вовлеченных в процессы участниках. Оптимальным инструментом для решения данной задачи выступает диаграмма прецедентов (ее еще называют диаграммой вариантов использования).

Данная диаграмма наглядно демонстрирует принцип будущей работы информационной системы. Она строится на основе вариантов использования, то есть конкретных действий и функций, которые разрабатываемая система предоставляет внешним пользователям или другим системам. Внешние по отношению к системе сущности, которые с ней взаимодействуют называются актерами. Актером может быть другая система, любой объект или субъект, обращающаяся к разрабатываемой системе извне. Каждый вариант использования отражает и фиксирует последовательность действий, выполняемых системой в качестве ответа на обращение пользователя [7].

В самом начале проектного цикла, на стадии формирования требований и общей концепции, именно диаграмма вариантов использования играет центральную роль. Для построения этой диаграммы существуют основные цели, заключающиеся в следующем:

– определение контекста и границ моделируемой области на ранних этапах разработки;

– описание функционального поведения создаваемого решения;

– подготовка первичной документации, которая является необходимой для грамотного и эффективного взаимодействия между разработчиками, заказчиками и конечными пользователями.

Также, чтобы спроектировать диаграмму вариантов использования необходимо определить функции разрабатываемого программного камплекса. В результате анализа предметной области и продуктов-аналогов были выделены следующие основные функциональные требования к системе:

– возможность входа в систему под существующими учетными записями;

– возможность регистрации с разграничением ролей (администратор и покупатель), для покупателей регистрация должна открывать доступ к просмотру каталога товаров и заказов, для администраторов – к управлению каталогом и остатками товаров, видами автомобилей, к просмотру пользователей и отчетности;

– администраторы и покупатели должны иметь возможность просматривать каталог товаров, их цену, описание, совместимость с автомобилями и производителя;

– возможность фильтровать товары по марке и модели автомобиля, что позволит покупателям быстрее находить запчасти для конкретного транспортного средства, а фильтрация по производителю – приобретать товары предпочтительных производителей;

– система должна предоставлять возможность просматривать список пользователей, что позволит администратору отслеживать историю продаж;

– каждый пользователь (покупатель или администратор) должен иметь возможность просмотреть свои заказы, но администратор также будет иметь доступ к списку всех заказов;

– администратор должен иметь доступ к системе складского учета, для отслеживания движений товаров по складам.

С целью отображения всех ключевых функций разрабатываемого программного комплекса, предназначенного для онлайн-продажи автозапчастей, была построена диаграмма вариантов использования, представление которой продемонстрированно на рисунке 1.

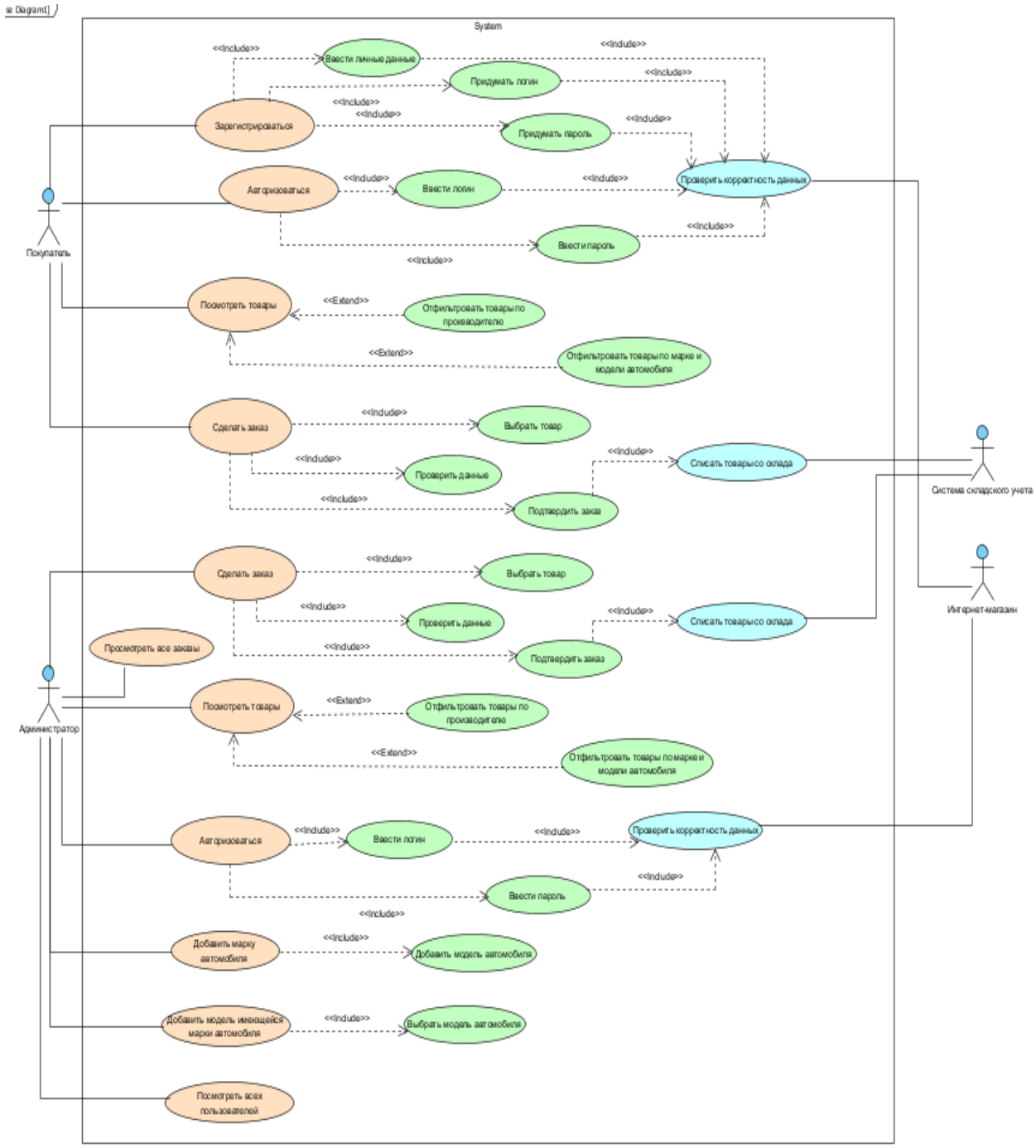


Рисунок 1 – Диаграмма вариантов использования программного комплекса онлайн-торговли автозапчастями

На диаграмме отображены функциональные требования к системе. Видно, какие функции имеет возможность выполнять каждый обычный пользователь, именуемый на диаграмме «Покупатель», а какие администратор.

Для удобства на диаграмме бизнес-процессы выделены оранжевым цветом; бизнес-операции – зеленым, а операции над данными – голубым.

2.2 Проектирование модели базы данных

На этапе разработки системы управления данными в качестве основного средства моделирования был выбран программный продукт Visual Paradigm Community Edition. Он относится к классу CASE-средств. С его помощью удаётся создавать различные типы моделей.

Стоит отметить, Visual Paradigm при построении моделей баз данных не только помогает с выбором СУБД, но и позволяет в автоматическом режиме генерировать физическую модель на основе ранее созданной логической. В свою очередь, полученная физическая модель становится базой для формирования SQL-скрипта, который затем используется при непосредственном развёртывании базы данных на сервере [8].

Для того, чтобы структурировать информацию о проектируемом программном комплексе и в дальнейшем построить базу данных, была разработана логическая модель базы данных. Она выполнена в распространённой нотации ER (сущность-связь). Такая модель представляет собой абстрактное описание данных, где каждый элемент соотносится с каким-либо объектом из предметной области. Подобный подход считается универсальным, поскольку он существенно облегчает последующий выбор конкретной СУБД. Опираясь на логическую схему, разработчики могут объективно оценить, какая из существующих систем управления базами данных оптимально подойдёт для реализации поставленных задач.

Что касается практической части работы, то в рамках разрабатываемого программного комплекса для онлайн-торговли автозапчастями была построена логическая модель данных. Она состоит из 15 таблиц, каждая из которых отражает определённый аспект функционирования торговой площадки. В модели детально прописаны все сущности, четко определены связи между ними и для каждой из них указаны первичные и внешние ключи.

Полное графическое представление логической модели данной системы можно увидеть на рисунке 2.

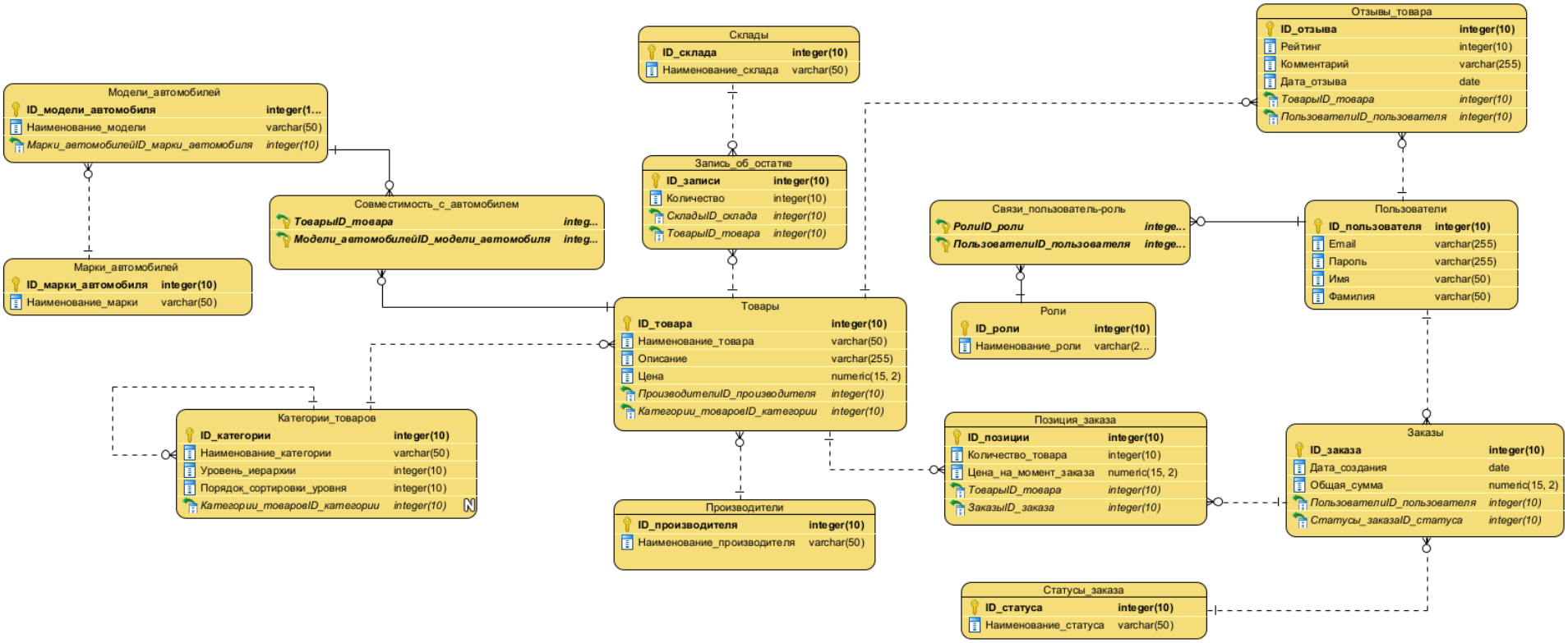


Рисунок 2 – Логическая модель базы данных программного комплекса для онлайн-торговли автозапчастями

Описание атрибутов логической модели базы данных программного комплекса для онлайн-торговли автозапчастями приведено в приложении В.

После того как были определены основные элементы базы данных и установлены связи между ними, возникает необходимость в корректировке ее архитектуры. Главная цель – исключить возможные дублирования информации и прочие недостатки в организации данных. Такая доработка продиктована требованиями к соблюдению целостности хранимой информации.

Спроектированная в ходе работы база данных соответствует трем основным условиям нормализации:

– каждый атрибут в таблицах является неделимым (простым);

– любой атрибут, который не входит в состав первичного ключа, должен находиться в полной зависимости от уникального идентификатора записи;

– между не ключевыми атрибутами и первичным ключом не должно существовать транзитивных зависимостей – они полностью отсутствуют.

Поскольку все перечисленные требования выполнены, можно с уверенностью утверждать, что разработанная модель базы данных соответствует третьей нормальной форме. Иными словами, она является полностью нормализованной. Таким образом, проектирование структуры данных осуществлялось с соблюдением всех стандартов, касающихся целостности информации и ее логической организации.

2.3 Архитектурное проектирование программного комплекса

На начальном этапе создания любой информационной системы самое важное четко понимать, зачем она вообще нужна. Как только главное назначение определено, из этого вытекает несколько взаимосвязанных подзадач. Вот ключевые из них:

– система должна без каких-либо проблем переживать любые изменения, которые могут повлиять работоспособность;

– система должна обладать всеми необходимыми функциями;

– система обязана соответствовать ожидаемым показателям скорости реакции на действия пользователей и производительности.

Обычно создание любой информационной системы разбивается на несколько этапов. На каждом этапе создаются и изменяются разные модели, которые привязаны к конкретной стадии жизненного цикла. А параллельно решаются вопросы безопасности, надежности, обслуживания и удобства использования. Над моделями (организационная структура, техническое задание, проектные решения, функциональные требования) работают разные члены проектной группы. Готовые модели отправляются в специальное хранилище, где остаются доступными для последующего применения и анализа. Весь проект ведется с помощью CASE-средств и UML-диаграмм – они сильно упрощают разработку и анализ.

Создаваемый в рамках данной работы программный продукт для интернет-продаж автозапчастей состоит из двух главных частей: самого онлайн-магазина (реализован на C# с использованием Windows Forms) и модуля складского учета (выполнен на платформе «1С:Предприятие 8.3»). Обе части работают с единой базой данных под управлением Microsoft SQL Server, выбор именно этой СУБД обусловлен её высокой производительностью, надёжностью и совместимостью с обоими используемыми средствами разработки. Через нее они обмениваются сведениями о товарной номенклатуре, заказах покупателей и текущих складских остатках.

Функционал интернет-магазина дает возможность просматривать ассортимент, отфильтровывать детали по категориям, маркам машин и фирмам-производителям, собирать виртуальную корзину и оформлять покупки. Со своей стороны, конфигурация 1С решает задачи учета на складе: резервирование товаров, списание при поступлении заказа, контроль остатков и формирование отчетных форм.

Благодаря тому, что оба приложения работают через общую базу данных, информация о наличии товаров и заказах остается актуальной в режиме реального времени.

2.3.1 Высокоуровневое проектирование программных средств

При высокоуровневом проектировании комплекса под онлайн-торговлю автозапчастями нужно заложить общую архитектуру, четко разделить обязанности между составными частями и выбрать технологии для каждого уровня. Реализация будет выполнена в виде многослойного клиент-серверного приложения, где сервер занимается приемом запросов и работой с хранилищем данных, а клиент дает пользователю возможность управлять системой [9].

Для наглядного представления архитектуры разрабатываемого программного комплекса и взаимосвязей между его основными частями была построена диаграмма компонентов.

Диаграмма компонентов – одна из диаграмм языка UML, предназначенная для визуализации статической структуры программной системы с точки зрения ее разбиения на крупные составные части – компоненты [10]. Компонент в данном контексте понимается как модуль системы, имеющий четко определенные границы, предоставляющий и потребляющий интерфейсы и способный существовать и развиваться относительно независимо от других компонентов.

Данная диаграмма позволяет абстрагироваться от внутренней реализации каждого компонента и сфокусироваться на их взаимодействии на высоком уровне. Результат моделирования представлен на рисунке 3.

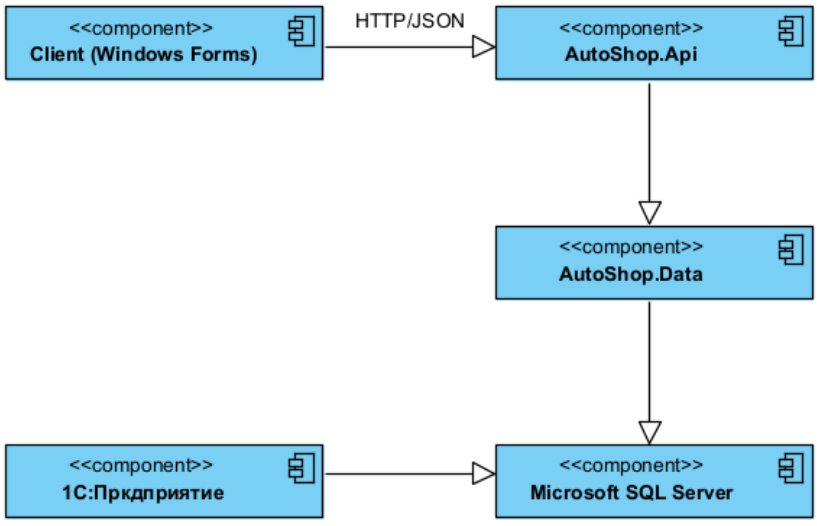


Рисунок 3 – Диаграмма компонентов программного комплекса

Архитектура решения организована по многослойному принципу. Диаграмма демонстрирует основные программные компоненты системы и связи между ними. Разрабатываемый комплекс включает следующие компоненты:

– компонент Client (WinForms) – интернет-магазин, клиентское приложение, реализованное на платформе Windows Forms с использованием языка C#, данный компонент обеспечивает пользовательский интерфейс для покупателей и администраторов, предоставляя функционал просмотра каталога автозапчастей, фильтрации по категориям, производителям и маркам автомобилей, формирования корзины и оформления заказов;

– компонент AutoShop.Api – серверный компонент, реализующий REST API со спецификацией Swagger, который принимает HTTP-запросы от клиентского приложения, выполняет бизнес-логику и обеспечивает взаимодействие с уровнем данных;

– компонент AutoShop.Data обеспечивает доступ к данным, выполняет операции чтения и записи в реляционную базу данных, абстрагируя вышележащие уровни от конкретной реализации СУБД;

– компонент базы данных на платформе Microsoft SQL Server содержит все таблицы, необходимые для хранения информации о пользователях, товарах, заказах, автомобилях, совместимости запчастей, а также остатках на складах;

– компонент 1С:Предприятие (система складского учета), реализованный на платформе 1С:Предприятие 8.3 обеспечивает ведение складского учета, резервирование и списание товаров при оформлении заказов, контроль актуальных остатков, формирование складской отчетности.

2.3.2 Проектирование серверной части интернет-магазина

Проектирование серверной части интернет-магазина автозапчастей включает в себя определение архитектуры серверного приложения, выбор технологий, а также описание основных компонентов, обеспечивающих бизнес-логику и взаимодействие с базой данных.

Серверная часть будет разработана с использованием фреймворка ASP.NET Core, который обеспечивает модульную, кроссплатформенную и производительную среду для разработки веб-приложений [11]. Серверное приложение будет следовать многослойной архитектуре, разделяя логику на несколько уровней:

– уровень представления реализован как отдельное WinForms-приложение, которое предоставляет пользовательский интерфейс для работы с системой, отправляет запросы к API и отображает полученные данные;

– уровень контроллеров принимает HTTP-запросы от клиента, выполняет базовую валидацию и передает управление на уровень бизнес-логики, возвращая результаты в формате JSON [12];

– уровень бизнес-логики реализован содержит основную логику работы магазина автозапчастей, включая обработку заказов, управление остатками, расчеты и бизнес-правила, взаимодействуя с уровнем доступа к данным;

– уровень доступа к данным работает с сущностями и подключением к БД, где выполняется вся работа с базой данных Microsoft SQL Server, включая определение структуры сущностей и настройку контекста подключения;

– уровень данных представляет собой базу данных SQL Server, которая обеспечивает надежное хранение всей информации о запчастях, клиентах, заказах и других сущностях системы.

Серверная часть будет взаимодействовать с клиентской частью через REST API, предоставляя JSON-ответы на запросы. Многослойную архитектуру серверной части лучше всего позволит отобразить диаграмма классов.

Диаграмма классов – одна из ключевых диаграмм языка UML, предназначенная для документирования статической структуры объектно-ориентированной системы [13]. Она отображает, какие классы входят в состав системы, какие атрибуты и методы эти классы имеют, а также какие отношения существуют между ними. Разработанная диаграмма классов представлена на рисунке 4.

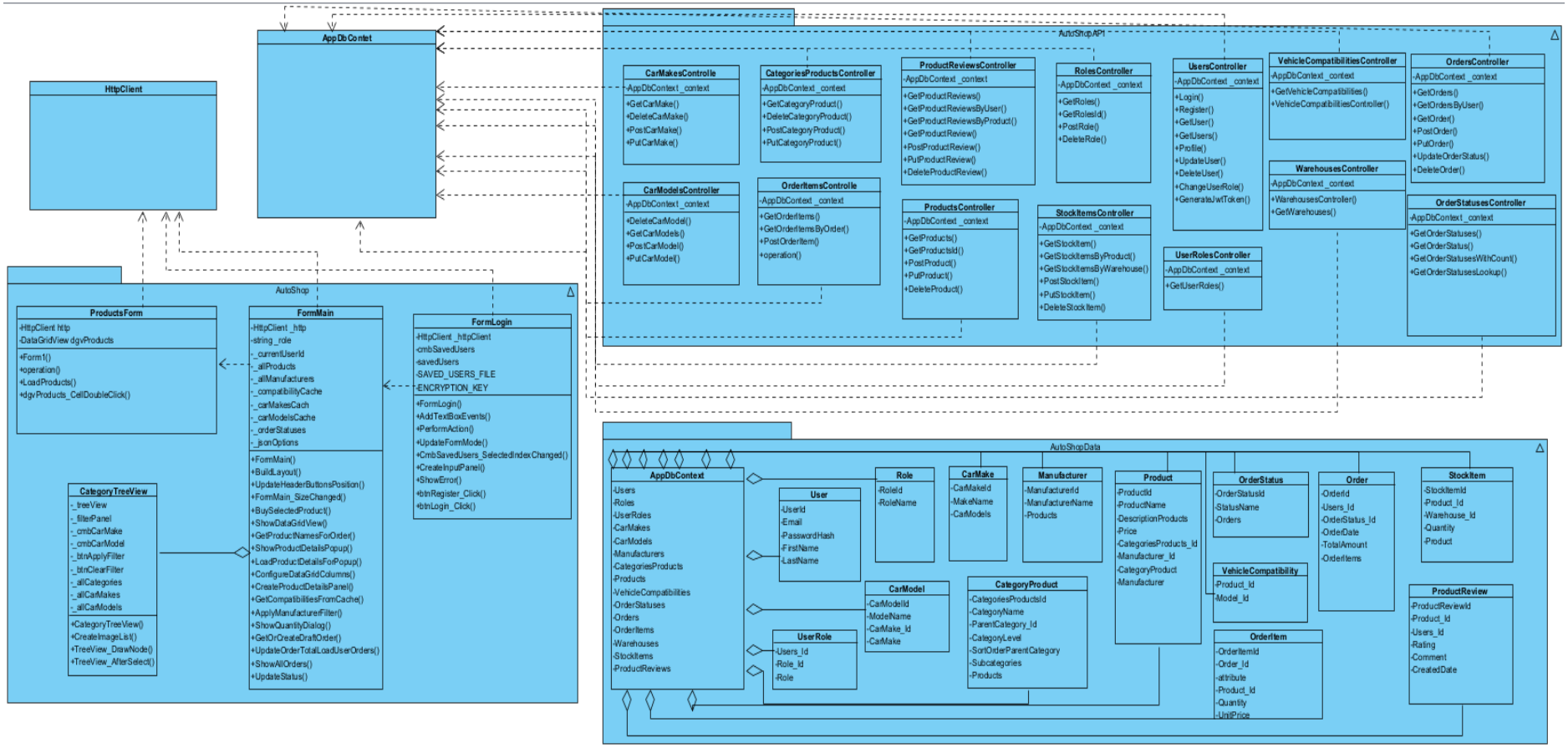


Рисунок 4 – Диаграмма классов интернет-магазина автозапчастей

Диаграмма классов на данном рисунке отображает основные компоненты приложения для интернет-магазина «Автозапчасти». Главные классы включают сервисы и репозитории для взаимодействия с API. Класс «Form Main» является основной точкой взаимодействия пользователя с приложением и содержит методы для загрузки каталога товаров, пользователей и марок/моделей автомобилей. Класс «CategoryTreeView» отображает информацию о каталоге товаров, имеющем иерархическую структуру.

2.3.3 Проектирование клиентской части интернет-магазина

Клиентская часть комплекса для онлайн-торговли автозапчастями будет разрабатываться на основе Windows Forms с использованием языка C#. Она будет предоставлять пользователям удобный интерфейс для взаимодействия с серверной частью и выполнения всех основных функций системы.

Клиентское приложение будет построено по модульному принципу, где каждый модуль отвечает за определенную функциональность.

Модуль авторизации: обеспечивает вход в систему и регистрацию новых пользователей (покупатели и администраторы).

Модуль управления товарами и заказами: позволяет пользователям просматривать товары и управлять своими заказами.

Модуль управления учетной записью: предоставляет возможности редактирования личной информации.

Основные формы клиентской части:

– форма FormLogin для авторизации пользователей;

– главная форма FormMain для просмотра каталога товаров и заказов.

Каждая форма взаимодействует с сервером через HTTP-клиент, отправляя асинхронные запросы к REST API (Swagger) и обрабатывая полученные ответы. Для этого используется HttpClient из библиотеки System.Net.Http.

В клиентской части будет реализована обработка ошибок, таких как некорректный ввод данных или сбои при соединении с сервером, с выводом соответствующих сообщений пользователю.

2.3.4 Проектирование системы складского учета

Подсистема складского учета будет разработана в виде отдельной конфигурации на платформе «1С:Предприятие 8.3». Она предназначена для:

- чтения оформленных заказов из общей базы данных SQL Server;
- резервирования и списания автозапчастей на складе;
- ведения справочников товаров, складов и поставщиков;
- формирования приходных и расходных документов;
- учета движения товаров с использованием регистров накопления;
- генерации отчетов по движению товаров, их остаткам и продажам.

Для ведения складского учета планируется разработка системы, связанной с той же базой данных Microsoft SQL Server, что и интернет-магазин автозапчастей, только через собственные механизмы доступа, имеющиеся в «1С:Предприятие 8.3» (COM-соединение или ODBC), такой подход обеспечит единое информационное пространство без дублирования данных.

Для получения заказов из интернет-магазина в 1С будет реализован механизм внешнего подключения к Microsoft SQL Server. Алгоритм работы:

Система через ODBC будет подключаться к базе данных Auto_shop и выполнять выборку из таблиц необходимых для складского учета, а после формировать справочники с данными из базы данных.

Создание документов «Заказ клиента» и «Поступление товаров» будет осуществляться на основании данных, загруженных в справочники и связанных между собой через внешние ключи в базе данных. При добавлении нового заказа в базу данных система складского учета будет подгружать эти данные и формировать документ «Заказ клиента».

На основании двух этих документов будут формироваться записи в регистр накопления «Остатки товаров», для учета товаров на складах.

2.4 Планирование разработки и оценка бюджета

План разработки и смета проекта составлялись с помощью приложения Project Libre. Отсчет начинается с 8 апреля 2026 года. По результатам

планирования получилось 36 задач, охватывающих весь путь создания системы: от изучения предметной сферы до финального оформления пояснительной записки.

Вся информация о задачах проекта их длительности и данных о предшествующих задачах представла на рисунке 5.

	Название	Продолжительность	Начало	Окончание	Предшествующие	Название ресурса
1	**Начало проекта**	38 дней	**08.04.26 8:00**	**03.06.26 17:00**		
2	**Постановка задачи и анализ требований**	7 дней	**08.04.26 8:00**	**16.04.26 17:00**		
3	Анализ предметной области	2 дней	08.04.26 8:00	09.04.26 17:00		Аналитик;Компьютер
4	Обзор аналогов	2 дней	11.04.26 8:00	14.04.26 17:00	3	Аналитик;Компьютер
5	Анализ функциональных требований	2 дней	15.04.26 8:00	16.04.26 17:00	4	Аналитик;Компьютер
6	Анализ нефункциональных требований	2 дней	15.04.26 8:00	16.04.26 17:00	4	Аналитик;Компьютер
7	Завершение постановки задачи и анализа требований	0 дней	16.04.26 17:00	16.04.26 17:00	6	
8	**Проектирование программных средств**	12 дней	**17.04.26 8:00**	**06.05.26 17:00**		
9	Проектирование архитектуры приложения	3 дней	17.04.26 8:00	21.04.26 17:00		Аналитик;Компьютер;Программист;Проектировщик
10	Проектирование структуры хранения	2 дней	22.04.26 8:00	23.04.26 17:00	9	Компьютер;Проектировщик
11	Проектирование приложения	4 дней	24.04.26 8:00	29.04.26 17:00	10	Компьютер;Программист;Проектировщик
12	Проектирование пользовательского интерфейса	3 дней	30.04.26 8:00	06.05.26 17:00	11	Дизайнер;Компьютер;Программист;Проектировщик
13	Завершение проектирования программных средств	0 дней	06.05.26 17:00	06.05.26 17:00	12	
14	**Реализация программных средств**	12 дней	**05.05.26 8:00**	**21.05.26 17:00**		
15	Реализация слоя данных приложения	3 дней	05.05.26 8:00	07.05.26 17:00		Программист;Компьютер
16	Реализация слоя бизнес-логики приложения	5 дней	08.05.26 8:00	15.05.26 17:00	15	Программист;Компьютер
17	Реализация слоя представления приложения	4 дней	18.05.26 8:00	21.05.26 17:00	16	Дизайнер;Компьютер;Программист;Проектировщик
18	Завершение реализации программных средств	0 дней	21.05.26 17:00	21.05.26 17:00	17	
19	**Тестирование**	8 дней	**21.05.26 8:00**	**01.06.26 17:00**		
20	Тестирование методом черного ящика	3 дней	21.05.26 8:00	25.05.26 17:00		Компьютер;Тестировщик
21	Модульное тестирование добавления книги	3 дней	26.05.26 8:00	28.05.26 17:00	20	Тестировщик;Компьютер
22	Анализ сложности кода модуля	2 дней	29.05.26 8:00	01.06.26 17:00	21	Программист;Тестировщик;Компьютер
23	Подготовка документации	0 дней	01.06.26 17:00	01.06.26 17:00	22	Бумага;Принтер;Компьютер;Аналитик
24	Оформление и печать документации	2 дней	02.06.26 8:00	03.06.26 17:00	23	Бумага;Принтер;Компьютер;Аналитик
25	Завершение проекта	0 дней	03.06.26 17:00	03.06.26 17:00	24	

Рисунок 5 – Список задач проекта

Для расчета бюджета проекта были определены трудовые ресурсы, участвующие в разработке. Для каждого ресурса установлена почасовая ставка, что позволило рассчитать итоговую стоимость проекта. Список ресурсов, используемых при выполнении задач представлен на рисунке 6.

№	Название ресурса	Тип	Стандартная ставка, руб/ч	Начисление	Базовый календарь	Ставка сверхурочный, руб/ч	Макс. Единиц	Затраты на использование
1	Аналитик	Трудовой	252,00	Пропорционально	Стандартный	348,00	100%	0,00 ₽
2	Программист	Трудовой	300,00	Пропорционально	Стандартный	450,00	100%	0,00 ₽
3	Дизайнер	Трудовой	265,00	Пропорционально	Стандартный	415,00	100%	0,00 ₽
4	Проектировщик	Трудовой	290,00	Пропорционально	Стандартный	435,00	100%	0,00 ₽
5	Компьютер	Материальный	60 000,00 ₽	В начале				0,00 ₽
6	Бумага	Материальный	500,00 ₽	В начале				0,00 ₽
7	Принтер	Материальный	7 000,00 ₽	В начале				0,00 ₽

Рисунок 6 – Список ресурсов

На основе выделенных задач и закрепленных за ними ресурсов автоматически была построена диаграмма Ганта, представленная на рисунке 7. Она показывает не только сроки, но взаимосвязь рабочих этапов. Помогает

узнать, когда и кому нужно заняться очередной задачей, что упрощает координацию действий персонала. [14].

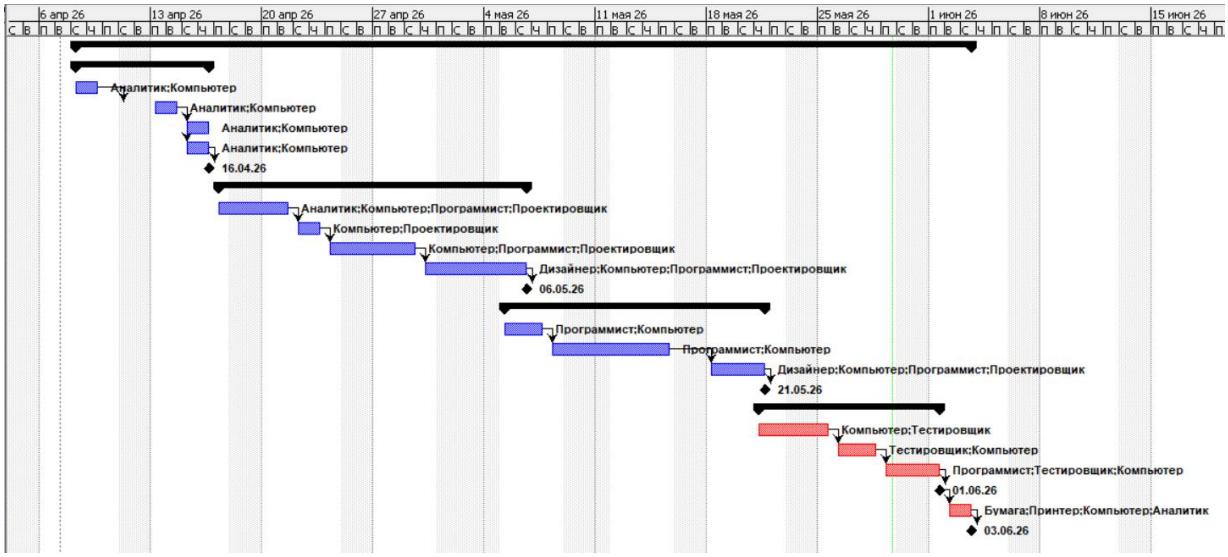


Рисунок 7 – Диаграмма Ганта для задач проекта

Таким образом, разработка проекта составит 38 полных рабочих дней с 08.04.26 по 03.06.26. Оценка бюджета проекта составляет 187 709,00 рублей.

3 Проектирование пользовательского интерфейса

3.1 Проектирование интерфейса интернет-магазина

То, через что человек управляет программой или устройством, принято называть пользовательским интерфейсом. Если элементы управления графические – кнопки, окна, поля ввода, то говорят о графическом интерфейсе. Любой интерфейс обязан быть удобным, чтобы пользователь мог работать с ним интуитивно, без лишних раздумий.

WinForms-приложение, создаваемое для интернет-продажи автозапчастей, будет иметь незамысловатый и понятный интерфейс. Он рассчитан обычных покупателей и на администраторов. Программа включает в себя форму для входа (авторизации и регистрации) и главное окно. В последнем есть область с меню навигации и рабочее поле, где отображаются нужные данные.

При запуске приложения отображается окно авторизации, пользователю предлагается зарегистрироваться или войти в уже существующий аккаунт.

Интерфейс окна авторизации показан рисунке 8.

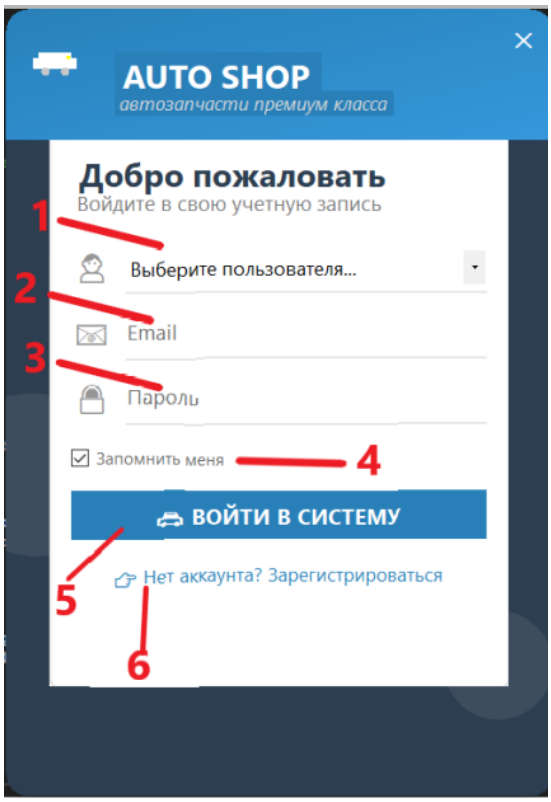


Рисунок 8 – Окно авторизации пользователя

На данном рисунке представлена структура интерфейса окна авторизации, которая содержит следующие элементы:

– выпадающий список для выбора аккаунта из списка пользователей, недавно заходивших в систему (1);

– обязательное поле для ввода логина (2);

– обязательное поле для ввода пароля (3);

– кнопка «Запомнить меня» для запоминания данных для входа в аккаунт (4);

– кнопка «Войти в систему», для входа в аккаунт после заполнения всех обязательных полей (5);

– кнопка регистрации, которая открывает окно для создания нового аккаунта пользователя (6).

Интерфейс окна регистрации пользователей в интернет-магазине автозапчастей продемонстрирован на рисунке 9.

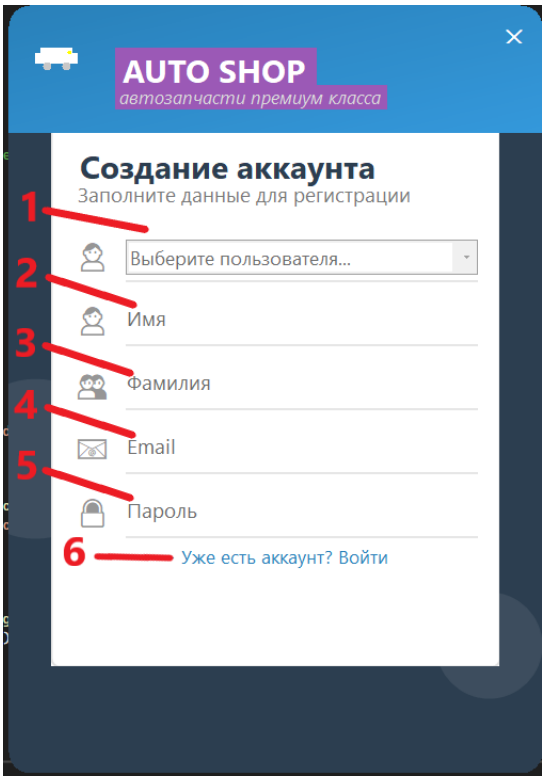


Рисунок 9 – Окно регистрации пользователя в интернет-магазине автозапчастей

Иллюстрация, отображающая структуру интерфейса окна регистрации пользователя в интернет-магазине, содержит следующие элементы:

– недоступный для выбора выпадающий список, выбора пользователей (1);

– обязательное поле для ввода имени (2);

– обязательное поле для ввода фамилии (3);

– обязательное поле для ввода логина (4);

– обязательное поле для ввода пароля (5);

– кнопка для возврата на окно авторизации (6).

После авторизации открывается главная форма приложения. Для быстрого введения пользователя в работу с системой на главной форме расположены самые востребованные элементы: каталог товаров с фильтрами по категориям и маркам автомобилей, корзина и заказы пользователя. В случае, если вход в систему произошел от имени администратора, вдобавок к предыдущим элементам на главной форме будут доступны еще такие, как:

– список всех пользователей, зарегистрированных в системе;

– список марок и моделей автомобилей, с которыми работает магазин;

– список всех заказов с полной информацией;

– отчеты по складам и продажам.

Интерфейс главной формы интернет-магазина показан на рисунке 10.

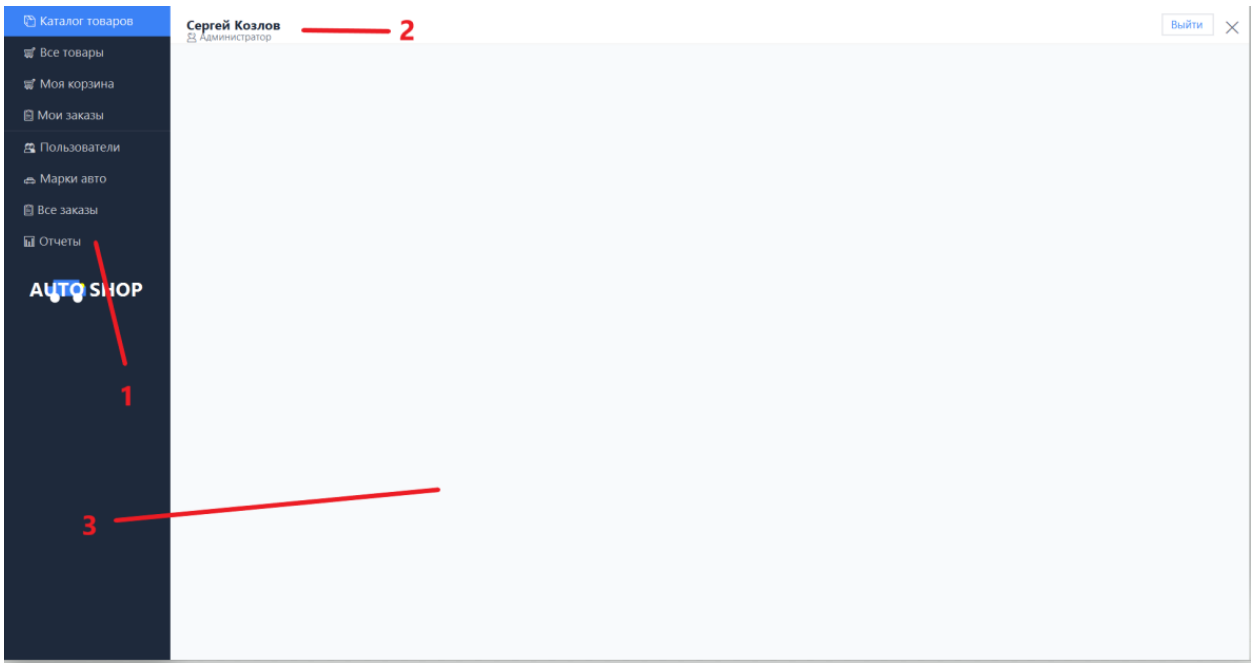


Рисунок 10 – Главная форма интернет-магазина

На данном рисунке представлена структура интерфейса главной формы приложения, которая включает в себя панель навигации, панель с информацией о пользователе и рабочую область.

Панель с информацией о пользователе расположена в верху главной формы и отображает данные об аккаунте, под которым был совершен вход в систему, а именно имя, фамилию и роль.

Основной инструмент для навигации между разделами приложения – панель навигации или панель кнопок, она разделяет весь функционал приложения на логические блоки: «Каталог товаров», «Все товары», «Моя корзина», «Мои заказы» (для обычных пользователей и администраторов), а также «Пользователи», «Марки авто», «Все заказы» и «Отчеты» (для администраторов). Панель навигации является верхним уровнем иерархии интерфейса и располагается в левой части главного окна.

Основная рабочая область, расположенная справа от панели навигации, является связующим звеном между всеми разделами приложения. В зависимости от выбранного пункта навигации в ней отображается список товаров, содержимое корзины, история заказов или форма редактирования учетной записи. Например, при выборе раздела «Все товары» пользователь увидит список всех имеющихся в интернет-магазине товаров, сможет просмотреть информацию о каждой запчасти, а именно: описание, цену, совместимость с автомобилями. Если пользователь откроет раздел «Каталог товаров», то в рабочей области отобразится дерево категорий автозапчастей, с возможностью отфильтровать категории и товары по марке и модели автомобиля.

Просмотреть свои предыдущие действия пользователь может с помощью списка своих заказов, который доступен в разделе «Мои заказы». Это позволяет быстро вернуться к оформленным ранее заказам или повторить покупку.

Для выполнения различных операций с объектами системы используются элементы управления: кнопки, текстовые поля, выпадающие списки, числовые счетчики и таблицы. Например, кнопка «Добавить в корзину» позволяет

поместить выбранную автозапчасть в заказ, кнопка «Оформить заказ» инициирует процесс создания заказа и передачи его в 1С.

3.2 Проектирование интерфейса системы складского учета

Платформа «1С:Предприятие 8.3» относится к числу продуктов, отличающихся продуманной эргономикой и комфортным взаимодействием с конечным пользователем. Её интерфейсное решение базируется на структуре основного окна, в котором выделяются две ключевые зоны, навигационная панель разделов и поле для работы с данными [15].

Процесс освоения системы для нового сотрудника значительно ускоряется благодаря стартовому экрану. На нём размещаются те элементы и команды, которые наиболее часто задействуются в рамках текущей конфигурации.

Навигация по системе осуществляется через панель разделов. Данный элемент интерфейса закреплён в верхней части окна и выполняет две взаимосвязанные задачи. С одной стороны, он систематизирует все доступные функции и подсистемы, делая их обозримыми и логически упорядоченными. С другой – он представляет собой верхний уровень иерархии, объединяющий все компоненты, входящие в состав конкретной конфигурации.

Кроме того, разработчиками предусмотрены специализированные команды, которые расширяют функциональность отдельных объектов учётной системы. Такие команды программируются на этапе создания конфигурации, однако их ценность для конечного пользователя сложно переоценить. Благодаря им можно быстро выполнять различные операции, к примеру, формировать и отправлять печатные формы документов.

Главная панель в интерфейсе играет роль элемента, обеспечивающего связь между всеми остальными панелями, гарантируя быстрый переход с одной на другую.

Если пользователю необходимо вернуться к недавно открытым объектам, он может обратиться к истории – там перечислены все его предыдущие действия, и есть возможность быстро перейти к любому из них.

В нижней части основного окна, которая называется «подвалом», при желании можно разместить информационную панель. Обычному пользователю она дает немного полезной информации, эти данные нужны прежде всего разработчику для проверки корректности работы конфигурации. Ее главное назначение – отображение технических параметров: показателей производительности, количества обращений к серверу и искусственно создаваемых задержек [16].

Разработанная в рамках данного проекта конфигурация включает в себя пять разделов:

– «Главное»;
– «Автомобили»;
– «Контрагенты и заказы»;
– «Товары и производители»;
– «Учет товаров».

Рисунок 11 демонстрирует, как выглядит интерфейс начального окна разрабатываемой системы для учета на складе.

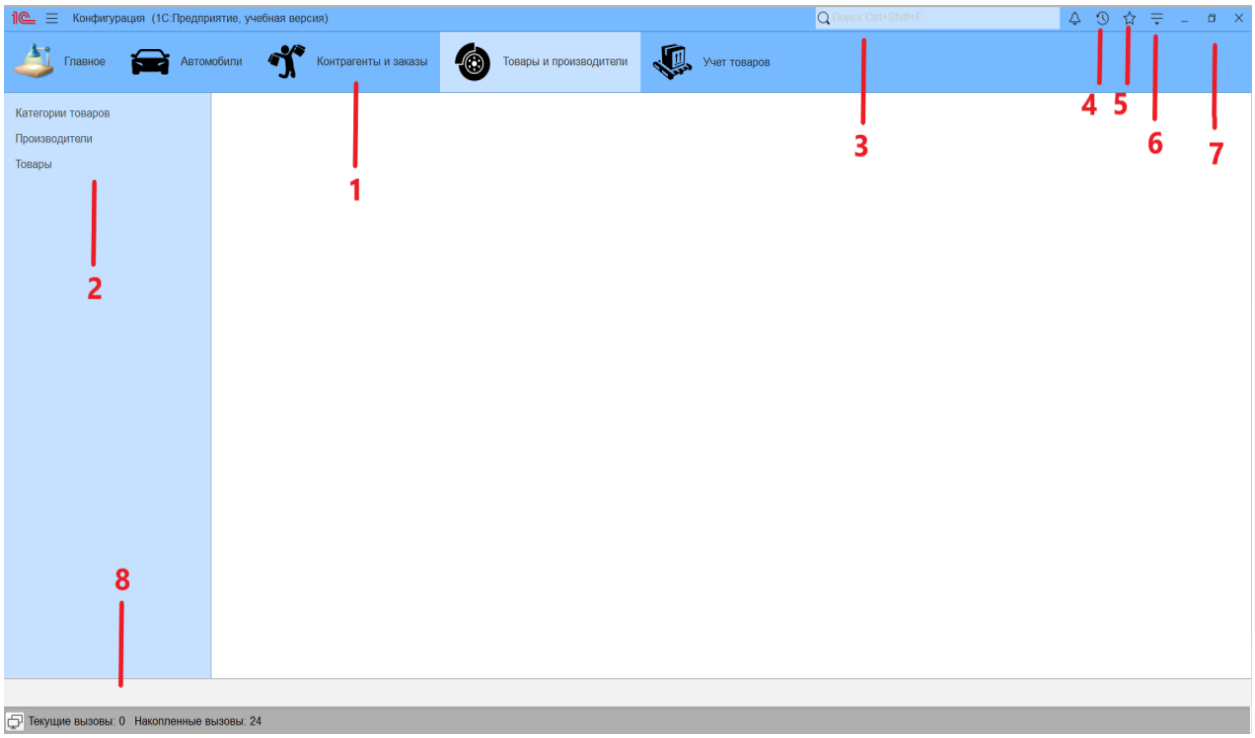


Рисунок 11 – Интерфейс начального окна системы для ведения складского учета автозапчастей

На данном рисунке представлены такие элементы интерфейса как:

– подсистемы разрабатываемой конфигурации (1);

– объекты выбранной подсистемы разрабатываемой конфигурации (2);

– поле поиска объектов (3);

– кнопка «История», при нажатии на которую будут отображены последние использованные объекты (4);

– кнопка «Избранное», нажав на которую можно посмотреть объекты, избранные пользователем, и перейти к ним (5);

– кнопка «Сервис и настройки» (6);

– кнопки «Свернуть», «Восстановить», «Закрыть» (7);

– строка отображения открытых объектов (8).

Объектами подсистем разрабатываемой конфигурации могут выступать самые разные компоненты, например, нормативно-справочная информация (справочники), регистры для хранения остатков автозапчастей или приходные и расходные документы.

На рисунке 12 продемонстрирован интерфейс, который увидит пользователь, если перейдет на вкладку регистра «Остатки товаров».

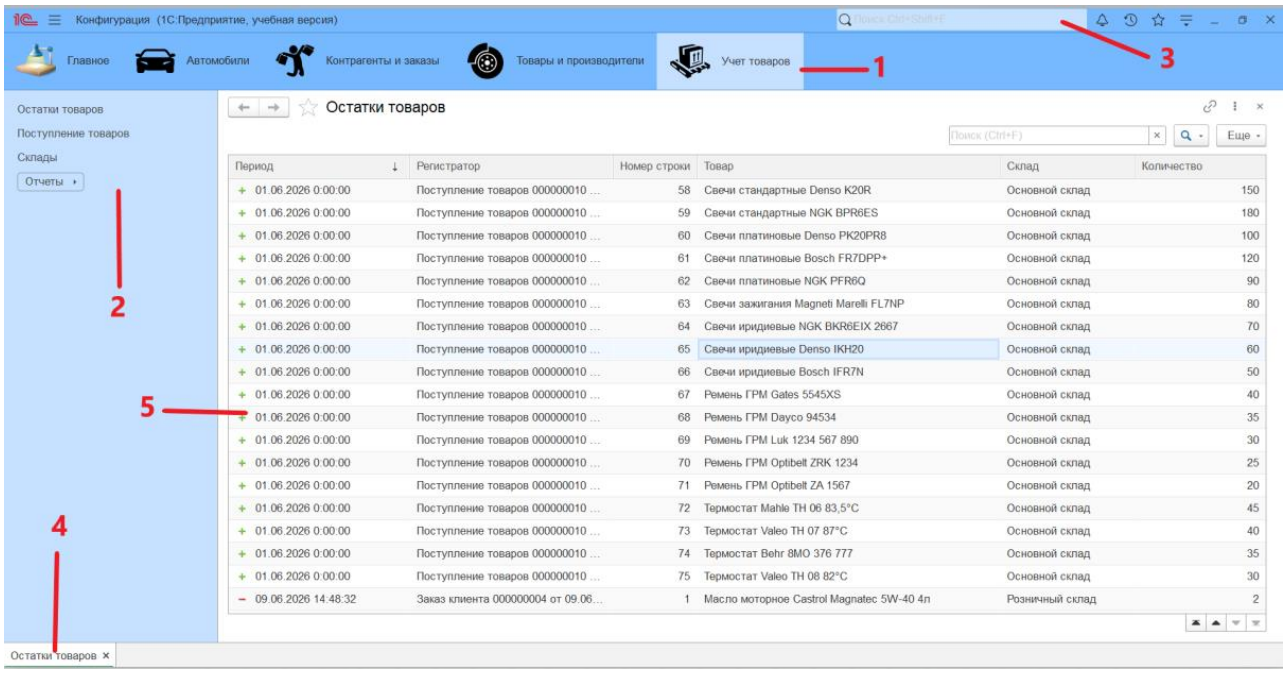


Рисунок 12 – Окно интерфейса объекта «Остатки товаров» системы для ведения складского учета автозапчастей

Устройство всех объектов подсистем выполнено по схожему принципу. Как выглядит интерфейс регистра остатков «Остатки товаров», показано на рисунке. Он состоит из следующего набора элементов:

– подсистемы разрабатываемой конфигурации (1);
– объекты выбранной подсистемы разрабатываемой конфигурации (2);
– поле поиска объектов (3);
– строка отображения открытых объектов (4);
– список движений по документам (5).

Если пользователь выберет объект «Отчеты», ему станет доступно формирование отчета об остатках товаров за нужный промежуток времени. Интерфейс такого отчета продемонстрирован на рисунке 13.

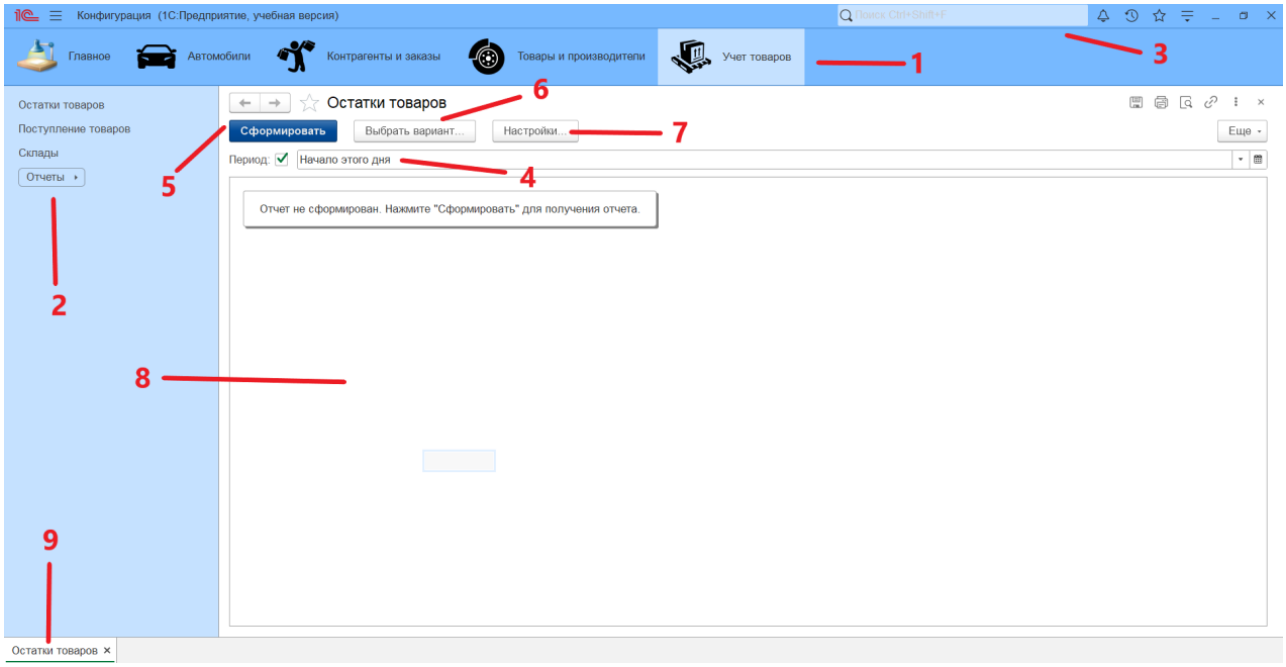


Рисунок 13 – Окно интерфейса объекта «Отчеты» системы для ведения складского учета автозапчастей

Представленный объект интерфейса содержит следующие элементы:

– подсистемы разрабатываемой конфигурации (1);
– объекты выбранной подсистемы разрабатываемой конфигурации (2);
– поле поиска объектов (3);
– поле определения периода для формирования отчета (4);

– кнопка «Сформировать» для формирования отчет (5);

– кнопка «Выбрать вариант», которая позволяет выбрать вариант формируемого отчета (6);

– кнопка «Настройки» для изменения периода уже после формирования отчета (7);

– поле для отображения отчета (8);

– строка отображения открытых объектов (9).

Представленный набор элементов интерфейса обеспечивает пользователю удобную навигацию по подсистемам и быстрый доступ к ключевым функциям формирования отчётности, что делает работу с учётными данными наглядной и эффективной.

4 Реализация

4.1 Реализация серверной части интернет-магазина

Серверная часть создавалась на основе фреймворка ASP.NET Core с использованием архитектурного шаблона MVC (Model, View, Controller). Если говорить простыми словами, этот подход делит программу на три самостоятельные части: модель, представление и контроллер. Такая схема позволяет менять каждую из них по отдельности, не трогая остальные, что сильно упрощает разработку и дальнейшее сопровождение приложений. Широкое распространение данной архитектуры в современной разработке также обеспечивает наличие большого количества готовых решений и документации, что сокращает время на реализацию типовых задач.

Модель – это сердце логики приложения. Она занимается данными, определяет способы работы с ними и общую структуру программы. Модель получает сигналы от контроллера, на их основе либо выдает информацию, либо меняет собственное состояние, а затем передает данные в представление.

Представление отвечает за то, как информация выглядит для пользователя. Оно берет данные от модели и превращает их в понятный интерфейс, например, в виде таблицы или списка. Проще говоря, представление задает внешний облик приложения и то, как пользователь будет с ним взаимодействовать.

Контроллер – это связующее звено между пользователем и системой. Он обрабатывает действия человека (клики, ввод данных), проверяет полученную информацию и отправляет ее модели. Контроллер решает, как приложение должно отреагировать на то или иное действие пользователя, а также может заниматься фильтрацией данных и проверкой прав доступа (авторизацией и регистрацией).

В приложении Б приведены листинги основных модулей разрабатываемого приложения.

В качестве СУБД для развертывания базы данных выбран MS SQL [17]. Как устроена эта база данных, можно увидеть на диаграмме (рисунок 14).

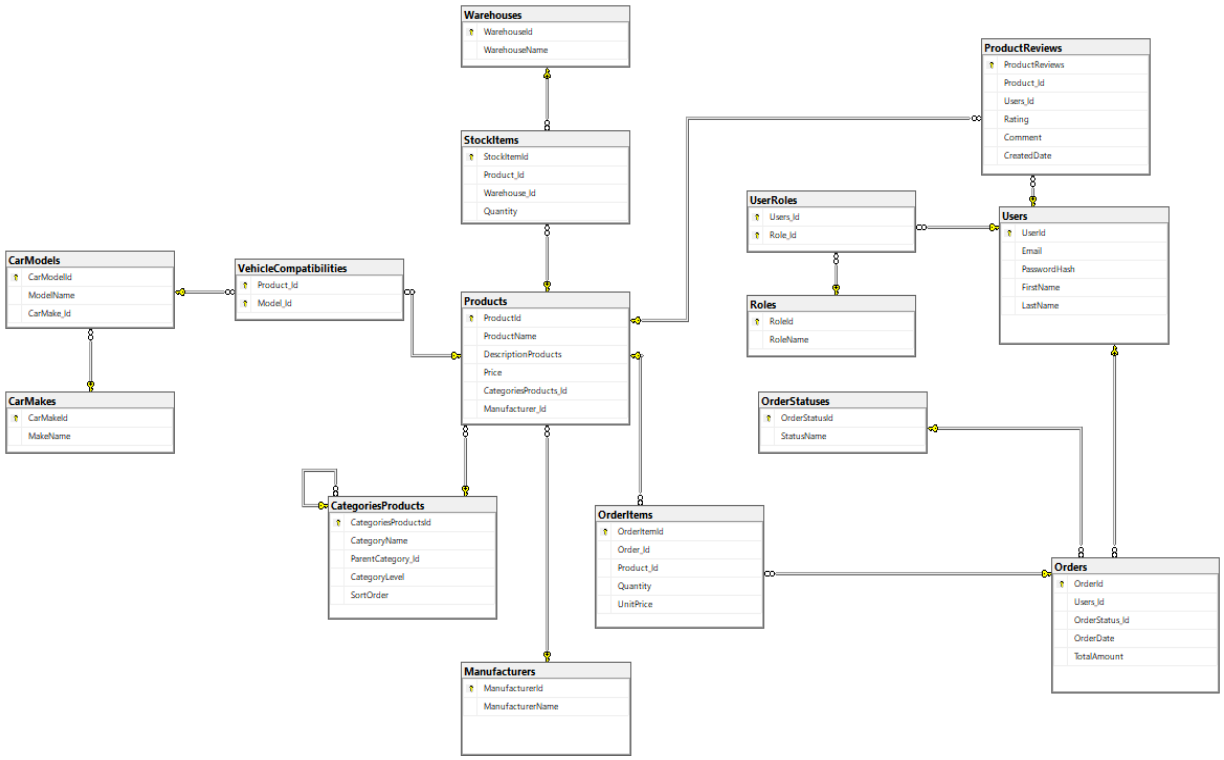


Рисунок 14 – Диаграмма базы данных

База данных интернет-магазина автозапчастей состоит из 11 основных таблиц:

– таблица Roles (Роли) содержит информацию о ролях, предназначенных каждому из пользователей;

– таблица Users (Пользователи) хранит данные о пользователях, включая контактные данные и данные для входа в личный кабинет (логин и пароль);

– таблица CarMakes (Марки автомобилей) содержит записи о марках автомобилей;

– таблица CarModels (Модели автомобилей) содержит записи о моделях автомобилей и принадлежности каждой модели к конкретной марке;

– таблица Manufacturers (Производители) хранит информацию о производителях автозапчастей;

– таблица CategoriesProducts (Категории товаров) содержит данные о категориях товаров, имеет внешний ключ на саму себя для создания иерархии в категориях (подкатегорий);’

– таблица Products (Товары) хранит информацию о товарах, а именно их наименование, описание, цену, наименование категории, к которой принадлежит товар и производителя;

– таблица OrderStatuses (Статусы заказа) содержит данные о статусах заказов;

– таблица Orders (Заказы) хранит информацию о заказе, а именно о пользователе, сделавшем заказ, дате заказа, статусе заказа и общей суммы;

– таблица Warehouses (Склады) содержит данные складов;

– таблица ProductReviews (Отзывы товара) хранит информацию об отзывах на конкретный товар и пользователе, который этот отзыв оставил.

Также в базе есть 4 вспомогательные таблицы, которые обеспечивают грамотное взаимодействие данных между основными таблицами:

– таблица UserRoles (Пользователь-роль) позволяет присвоить конкретную роль каждому пользователю;

– таблица VehicleCompatibilities (Совместимость с автомобилем) содержит внешние ключи на товар и автомобиль, что позволяет присвоить каждому товару совместимость с моделью автомобиля;

– таблица OrderItems (Позиция заказа) содержит информацию о каждом товаре в заказе, его количество и цену, а также внешний ключ на заказ, к которому эта позиция относится;

– таблица StockItems (Запись об остатке) содержит информацию о каждом товаре и его количестве на конкретном складе.

Эта структура базы данных обеспечивает надежное хранение и управление всеми необходимыми данными для эффективного функционирования информационной системы автомагазина.

В приложении используется класс AppDbContext, который инкапсулирует всю логику взаимодействия с базой данных, что позволяет абстрагироваться от конкретных подключений к источникам данных и является промежуточным звеном между классами, непосредственно взаимодействующими с данными, и остальной программой. Этот подход позволяет отделить бизнес-логику от

деталей хранения данных, что облегчает тестирование и замену источника данных при необходимости.

4.2 Реализация клиентской части интернет-магазина

Клиентская часть реализована в виде настольного приложения. Приложение предназначено для двух категорий пользователей: покупателей (просмотр каталога, фильтрация, корзина, оформление заказов) и администраторов (управление товарами, просмотр и редактирование заказов).

Окно авторизации является первым окном, которое видит пользователь, ему необходимо ввести для входа свой логин и пароль после чего нажать кнопку «Войти в систему», окно авторизации представлено на рисунке 15.

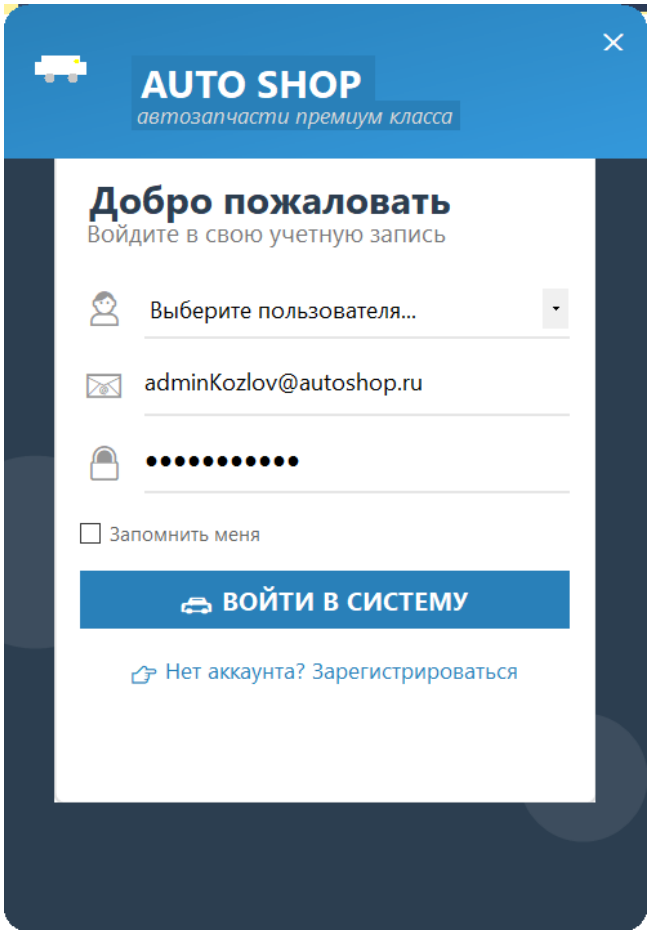


Рисунок 15 – Окно авторизации

На рисунке представлено окно авторизации, с введенными данными для входа от имени администратора. В приложении предусмотрена обработка ошибок, в случае, если пользователь введет некорректные данные, появится

соответствующее сообщение. После входа в систему откроется главная форма интернет-магазина, представленное на рисунке 16.

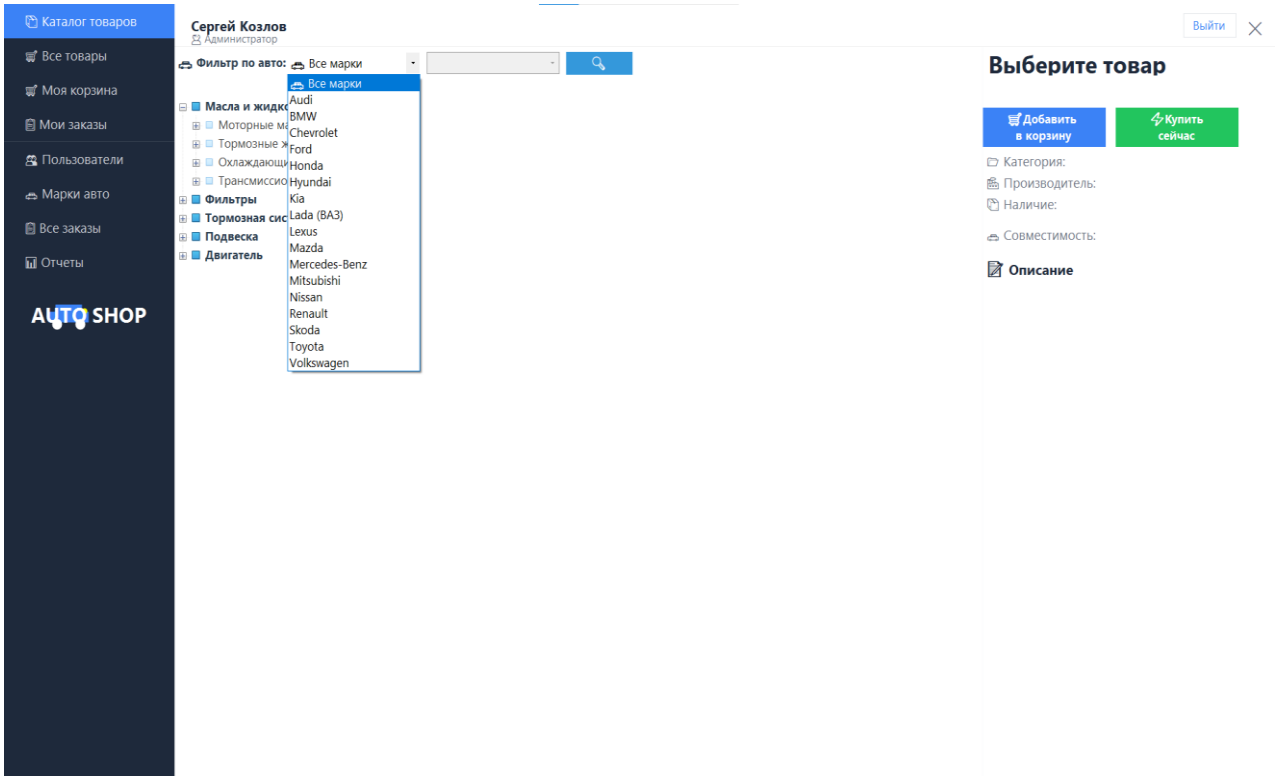


Рисунок 16 – Главная форма интернет-магазина

Главная форма интернет-магазина содержит перечень разделов, которые может открыть пользователь. Первые четыре раздела доступны как администратору, так и обычным пользователям, а последующие четыре только администратору.

На рисунке представлена форма с открытым разделом – «Категории товаров», он содержит дерево категорий автомобильных товаров. Пользователь имеет возможность не только посмотреть товары каждой категории, но и отфильтровать их по марке и модели автомобиля. При выборе товара справа на форме будет отображена информация о нем:

– полное наименование товара;
– стоимость;
– категория, к которой относится товар;
– наименования производителя;
– наличие, а именно имеющиеся количество товара;

– совместимость с автомобилями;

– краткое описание.

Пользователь вправе либо купить выбранный товар немедленно, либо отложить его в корзину. Когда одно из этих действий совершено, на экране появляется окно, изображенное на рисунке 17.

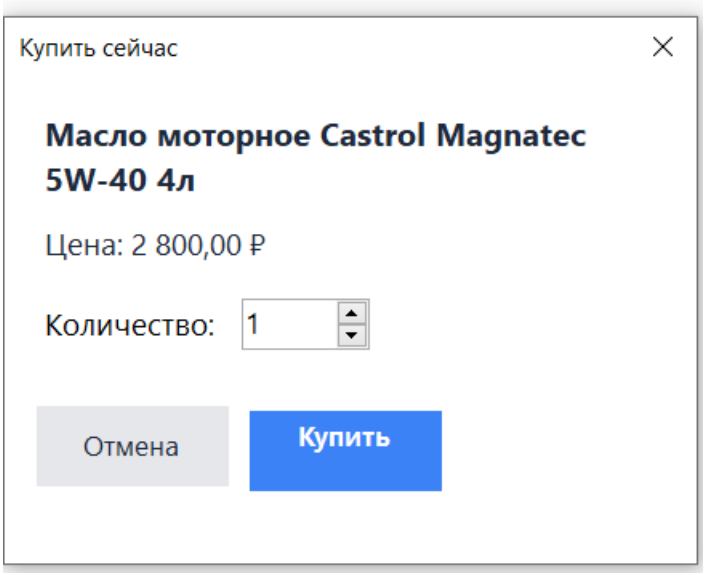


Рисунок 17 – Оформление покупки

В представленном на рисунке окне пользователю перед покупкой или добавлением товара в корзину необходимо указать количество и проверить параметры товара, цену и наименование. После успешной покупки или добавления товара в корзину, откроется окно с соответствующим сообщением, представленном на рисунке 18.

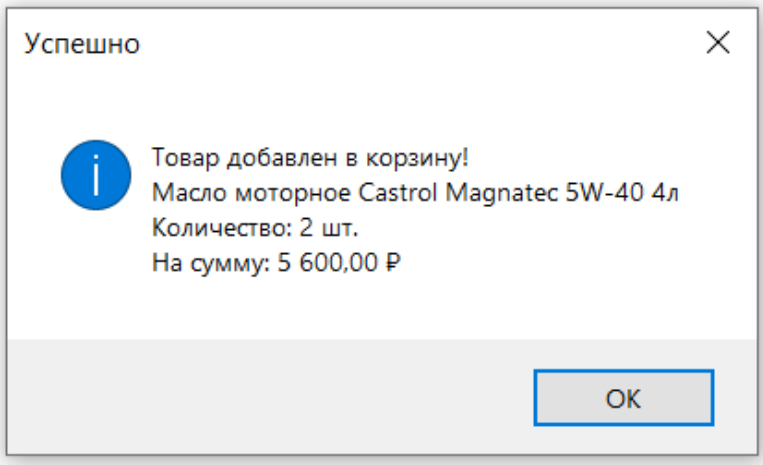


Рисунок 18 – Окно с сообщением пользователю

Теперь посмотреть все отложенные позиции можно в разделе под названием «Моя корзина» – его внешний вид показан на рисунке 19.

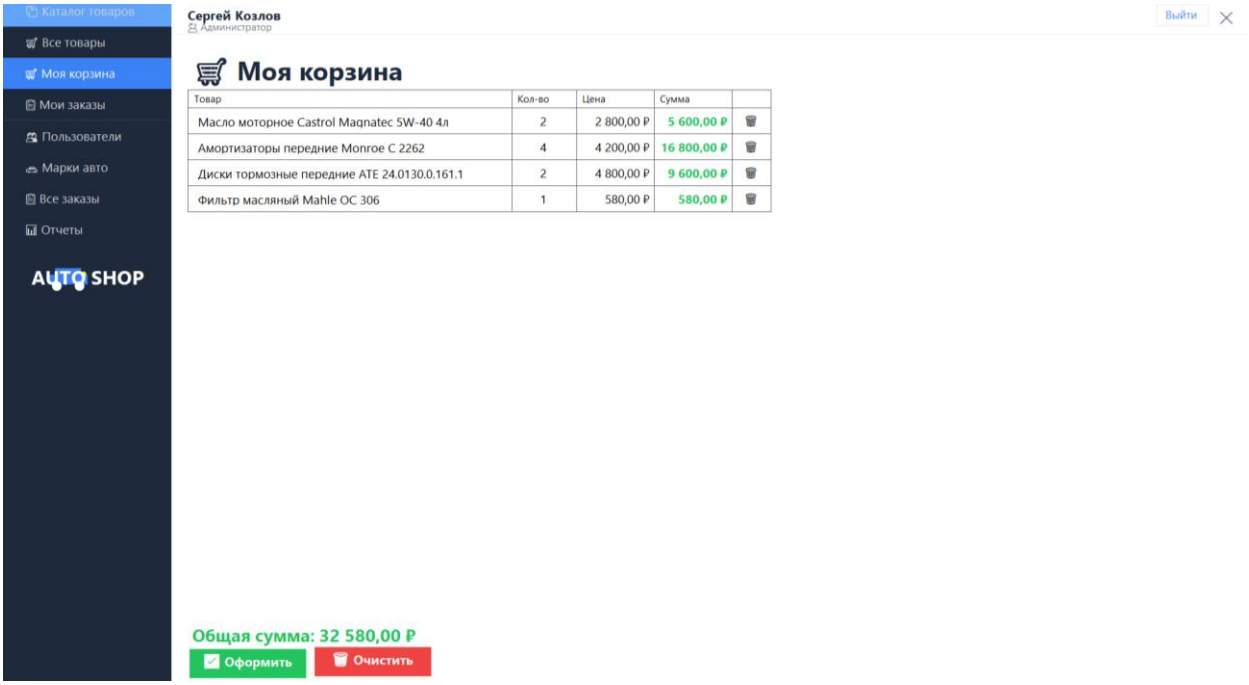


Рисунок 19 – Раздел «Моя корзина»

На рисунке показаны все товары, которые ранее были добавлены в корзину. Пользователь может оформить заказ, очистить корзину целиком или удалить отдельную позицию. Информация об оформленном заказе затем становится доступной в разделе «Мои заказы» (Рисунок 20).

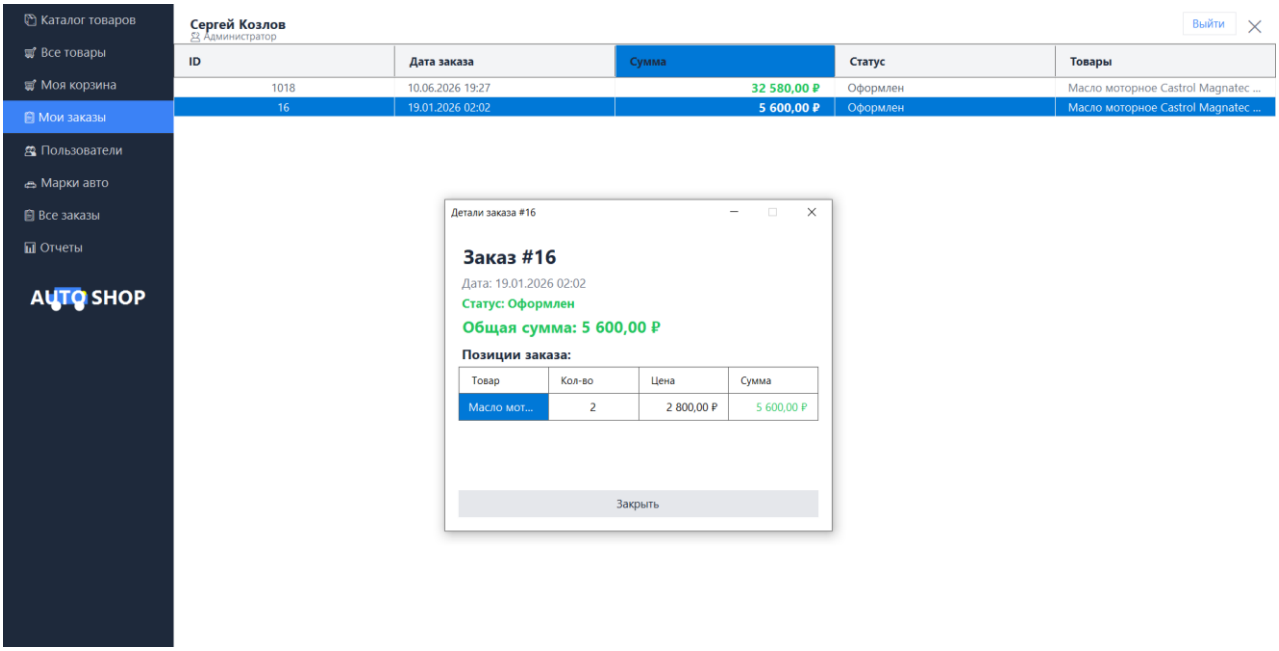


Рисунок 20 – Раздел «Мои заказы»

Представленный на рисунке раздел с заказами пользователя отображает, как список всех заказов, так и информацию о каком-то конкретном, что тоже представлено на рисунке.

Кроме раздела «Категории товаров» есть раздел «Все товары», он представлен на рисунке 21.

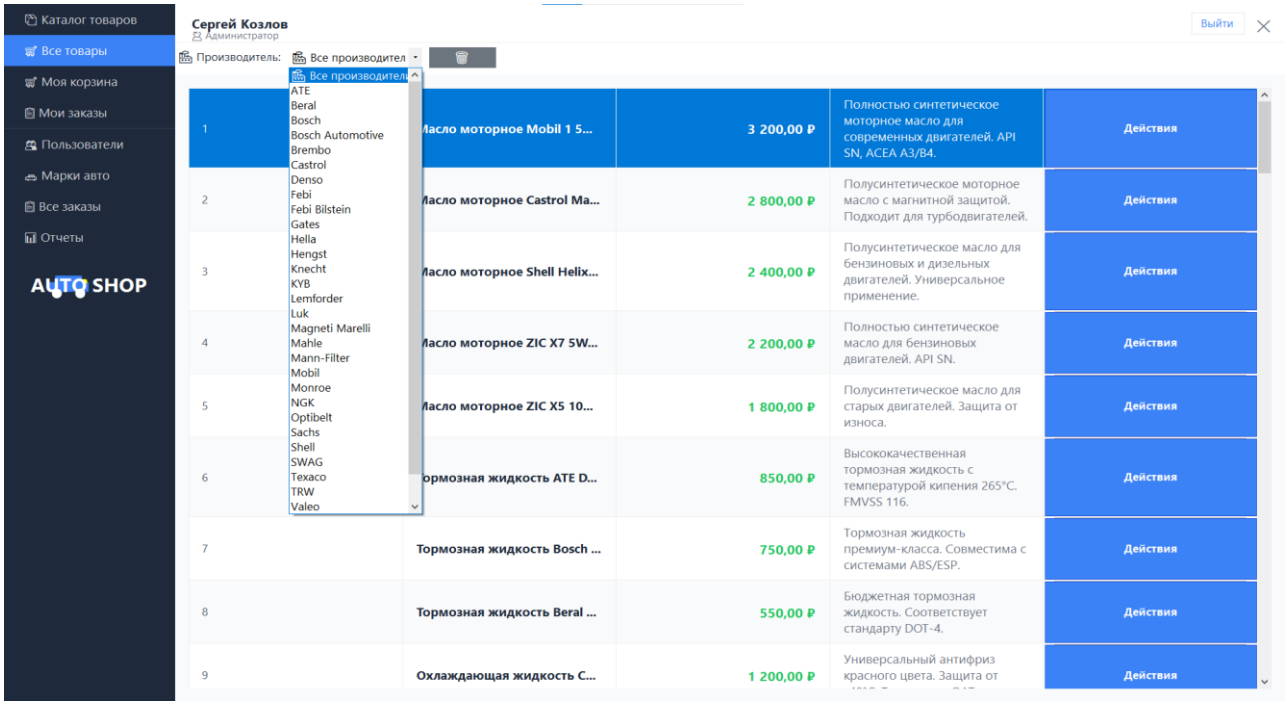


Рисунок 21 – Раздел «Все товары»

Данный раздел также, как и «Категории товаров», содержит перечень товаров, которыми торгует интернет-магазин. Ключевым отличием является то, что в разделе «Все товары» отображен список всех товаров, а не по категориям, с возможностью фильтрации по нужному производителю. Такая структура удобна для покупателей, которые уже знают, что именно им нужно, и хотят быстро найти товар без последовательного перебора категорий. Благодаря гибкой системе фильтров пользователь может сузить выборку, что значительно экономит время при поиске.

В строке каждого товара указано его полное наименование, стоимость и описание. Кнопка «Действия» дает возможность добавить товар в корзину, купить его или открыть окно с более подробной информацией, которое представлено на рисунке 22.

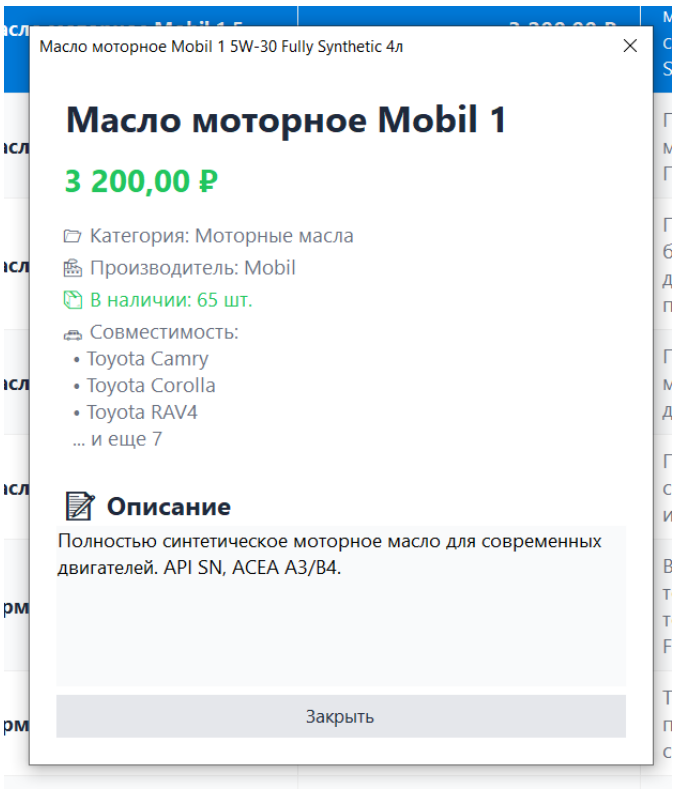


Рисунок 22 – Окно с полной информацией о товаре

Четыре рассмотренных раздела доступны всем пользователям. Только администратор имеет возможность просматривать остальные, один из таких представлен на рисунке 23.

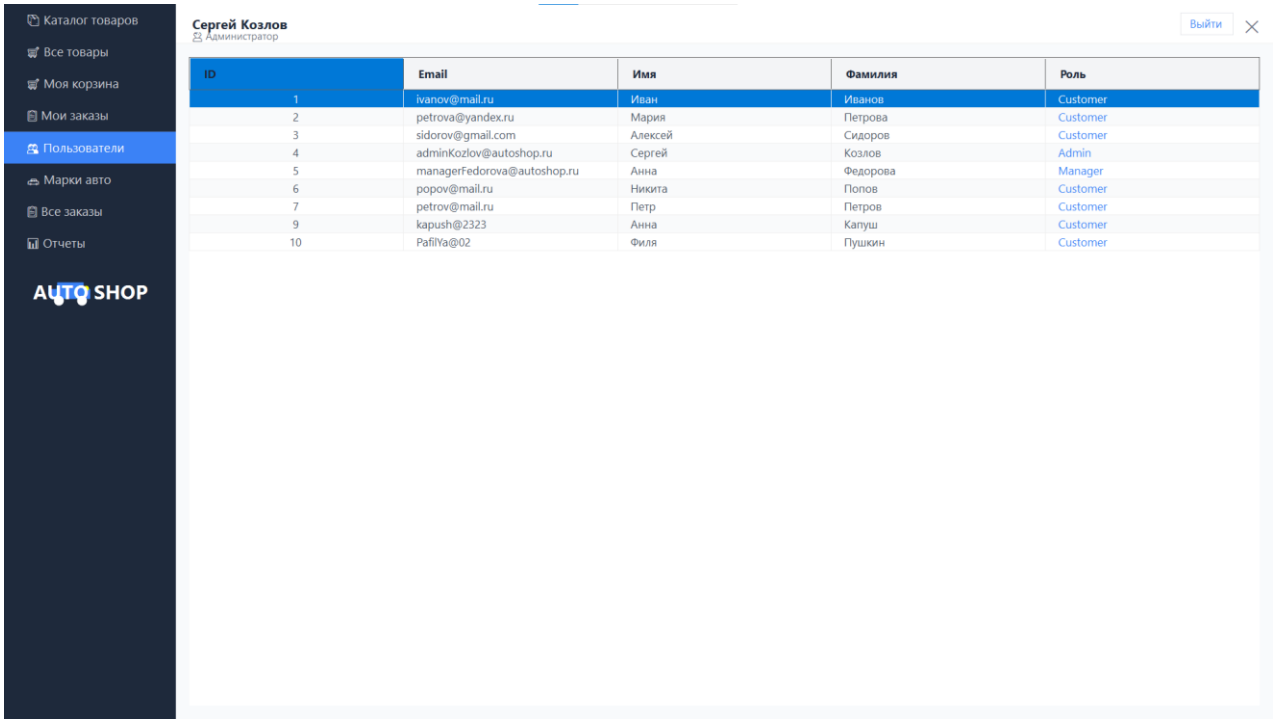


Рисунок 23 – Раздел «Пользователи»

Данный раздел дает возможность администратору просматривать информацию о пользователях, зарегистрированных в системе. Также он может просматривать и информацию о заказах этих пользователей в разделе «Все заказы», представленном на рисунке 24.

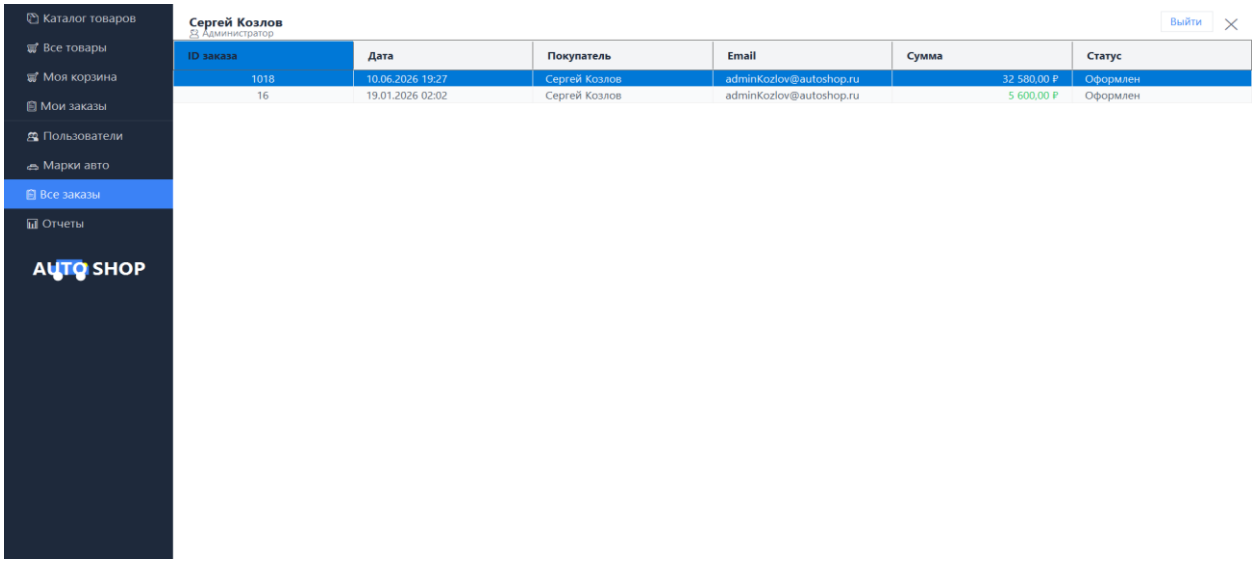


Рисунок 24 – Раздел «Все заказы»

Для поддержания актуальной информации о том, с какими автомобилями работает интернет-магазин, есть раздел «Марки авто», который показан на рисунке 25.

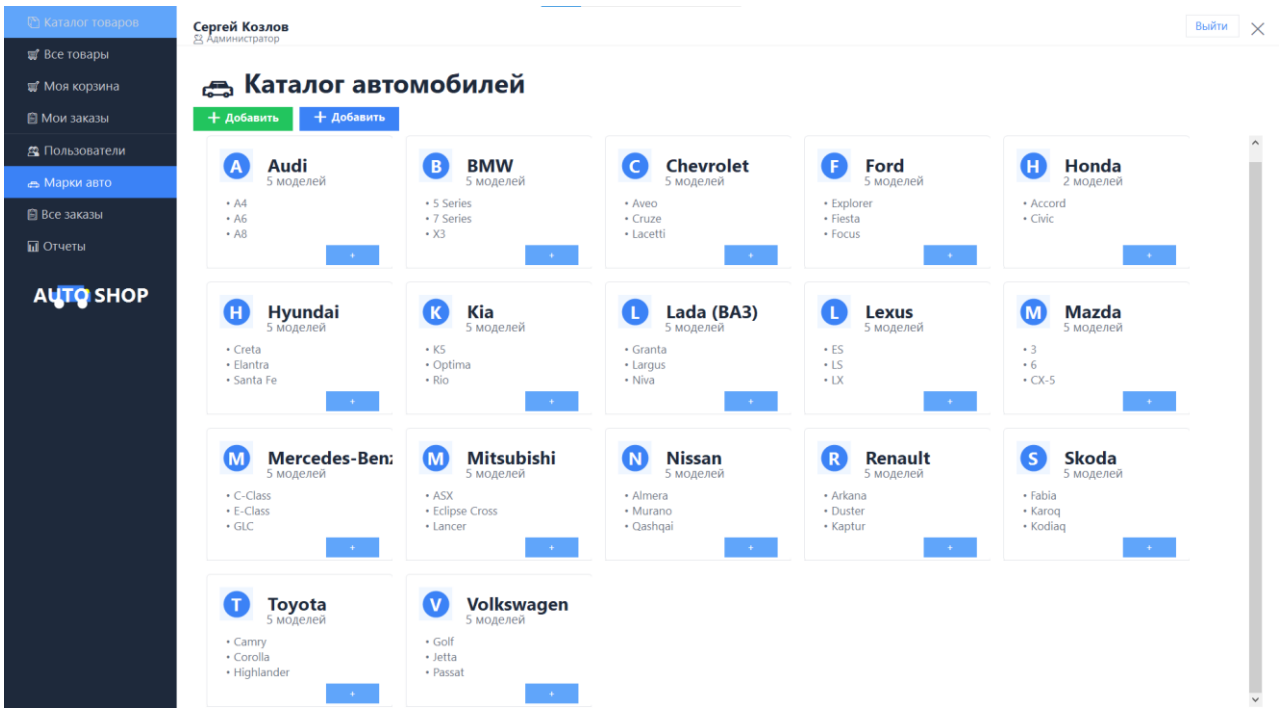


Рисунок 25 – Раздел «Марки авто»

В данном разделе администратор имеет возможность добавлять наименование марок автомобилей и их моделей, или модели автомобилей уже для имеющихся марок.

Для просмотра отчетности был предусмотрен раздел «Отчеты», представленном на рисунке 26.

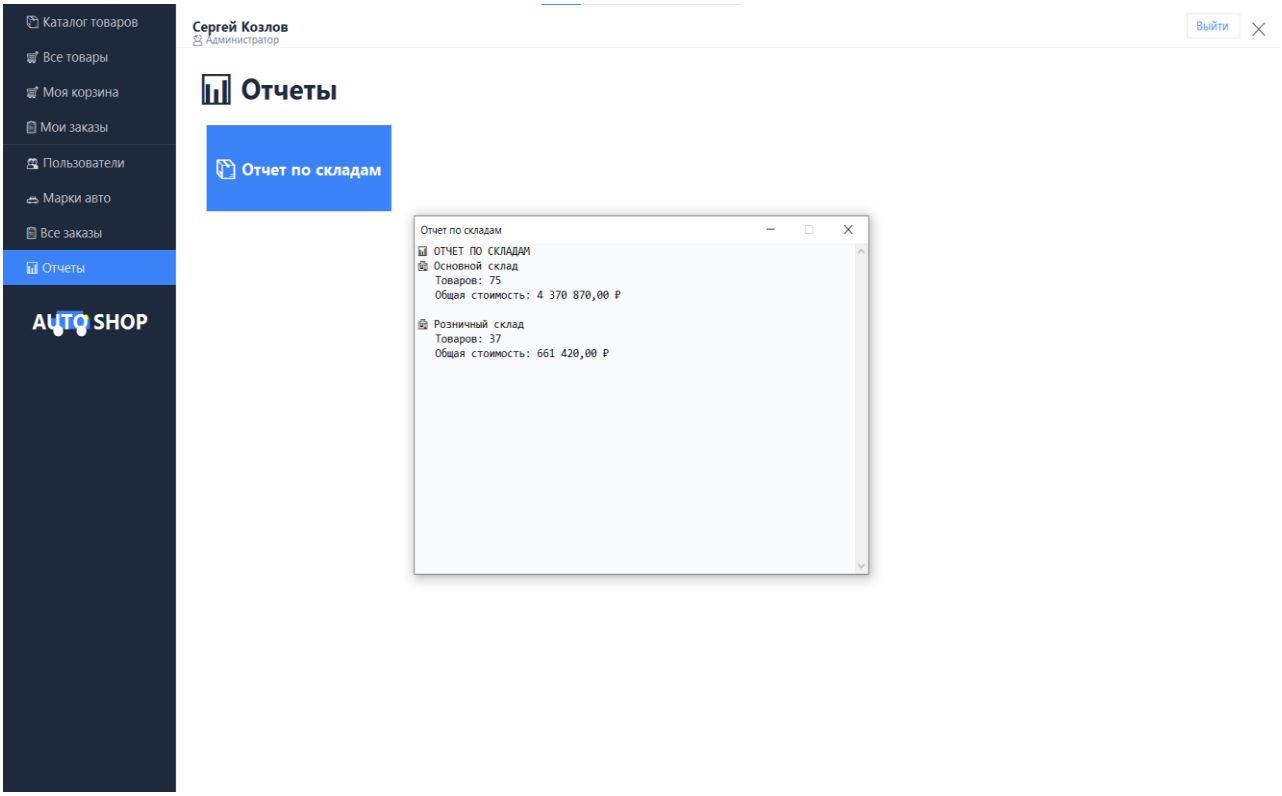


Рисунок 26 – Раздел «Отчеты»

Представленный раздел позволяет сформировать и посмотреть отчет о товарах, содержащихся на складах.

Все действия в интернет-магазине:

– добавление товаров в корзину;
– создание заказов;
– добавление новых марок и моделей автомобилей;
– списание товаров со складов;
– добавление пользователей.

Несут за собой последующие изменения в базе данных и в системе складского учета «1С:Педприятие 8.3».

4.3 Реализация системы складского учета

Формирование объектов конфигурации внутри платформы «1С:Предприятие 8.3» происходит на основе ряда базовых элементов. Эти элементы отражают те сущности и процессы, которые реально существуют в выбранной предметной области.

При создании складского учетного приложения применялись объекты конфигурации разных типов:

– справочники и формы их списков;

– документы и формы документов;

– отчеты;

– регистры накопления с типом «остатки».

Эти компоненты помогают грамотно построить и организовать работу с данными.

Для обеспечения возможности наглядно представить автоматизированную информационную систему для ведения складского учета автозапчастей, был выбран подход, помогающий разработчикам лучше понять внутреннее устройство системы и ее возможности – построение диаграммы компонентов.

Диаграмма компонентов является одним из главных инструментов, она не просто показывает статику, она еще и дает понять, от чего зависят компоненты и как именно они связаны между собой.

На разработанной диаграмме такого типа отображаются:

– файлы программы, к которым относятся текстовые файлы, файлы с данными и любые другие, касающиеся программы;

– библиотеки, включающие в себя наборы функций, собранные под одну обертку для многократного применения;

– модули, отдельные части программы, каждая из которых отвечает за свою конкретную задачу;

– исполняемые файлы, те, что содержат машинный код и запускаются на выполнение;

– пакеты, группы компонентов, в которые входят файлы, библиотеки, модули.

Результат в виде диаграммы компонентов системы для ведения складского учета автозапчастей проиллюстрирован на рисунке 27.

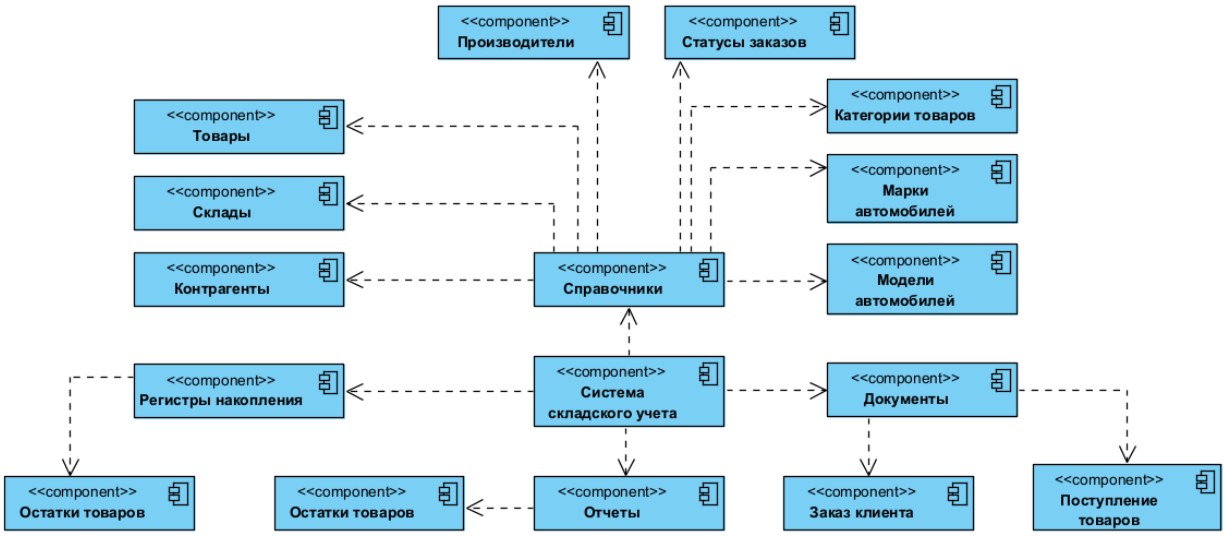


Рисунок 27 – Диаграмма компонентов системы для ведения складского учета автозапчастей

С помощью построенной диаграммы можно хорошо представить, как выглядит конфигурация изнутри. Ниже приведён детальный разбор структуры и характерных черт каждого из перечисленных объектов.

Справочники – тип конфигурационных объектов, который по праву считается одним из самых простых среди всех элементов ссылочного типа. Если говорить более точно, то справочник – это объект, который идентифицируется не своим содержимым, а уникальной ссылкой на него. Такое устройство делает справочники очень удобными для организации всевозможных связей между частями системы. Главным отличием справочника от документа является то, что он не формирует движений по регистрам.

В рамках создаваемой автоматизированной информационной системы для ведения складского учета автозапчастей были разработаны такие справочники, как:

– «Контрагенты»;

– «Производители»;
– «Категории товаров»;
– «Товары»;
– «Марки автомобилей»;
– «Модели автомобилей»;
– «Статусы заказов»;
– «Склады».

Справочник под названием «Контрагенты» хранит сведения обо всех контрагентах, у которых есть учетная запись в интернет-магазине. Доступ к нему открыт внутри подсистемы «Контрагенты и заказы». Внешний вид формы этого справочника показан на рисунке 28.

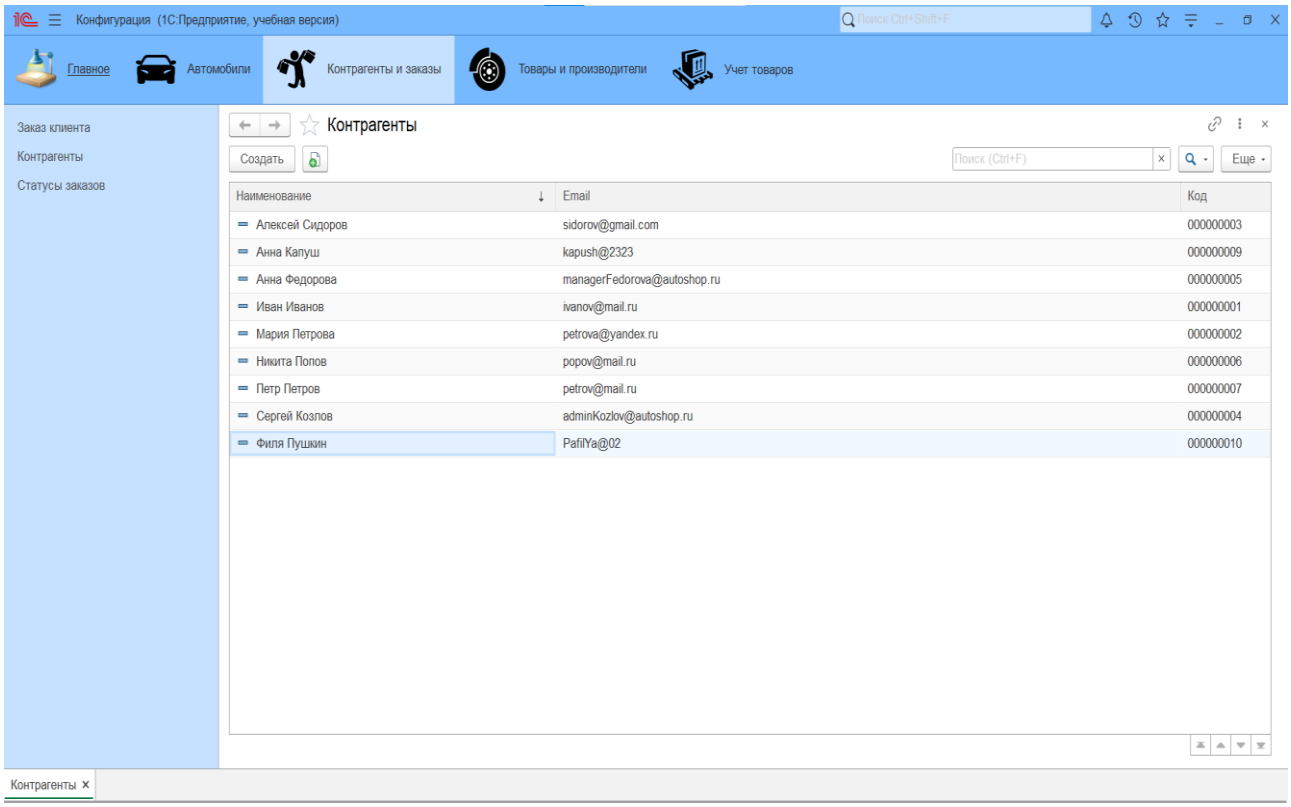


Рисунок 28 – Форма списка справочника «Контрагенты»

В справочнике «Производители» собраны данные обо всех изготовителях автозапчастей и сопутствующих товаров. Доступ к нему открыт через подсистему «Товары и производители». Внешний вид формы этого справочника показан на рисунке 29.

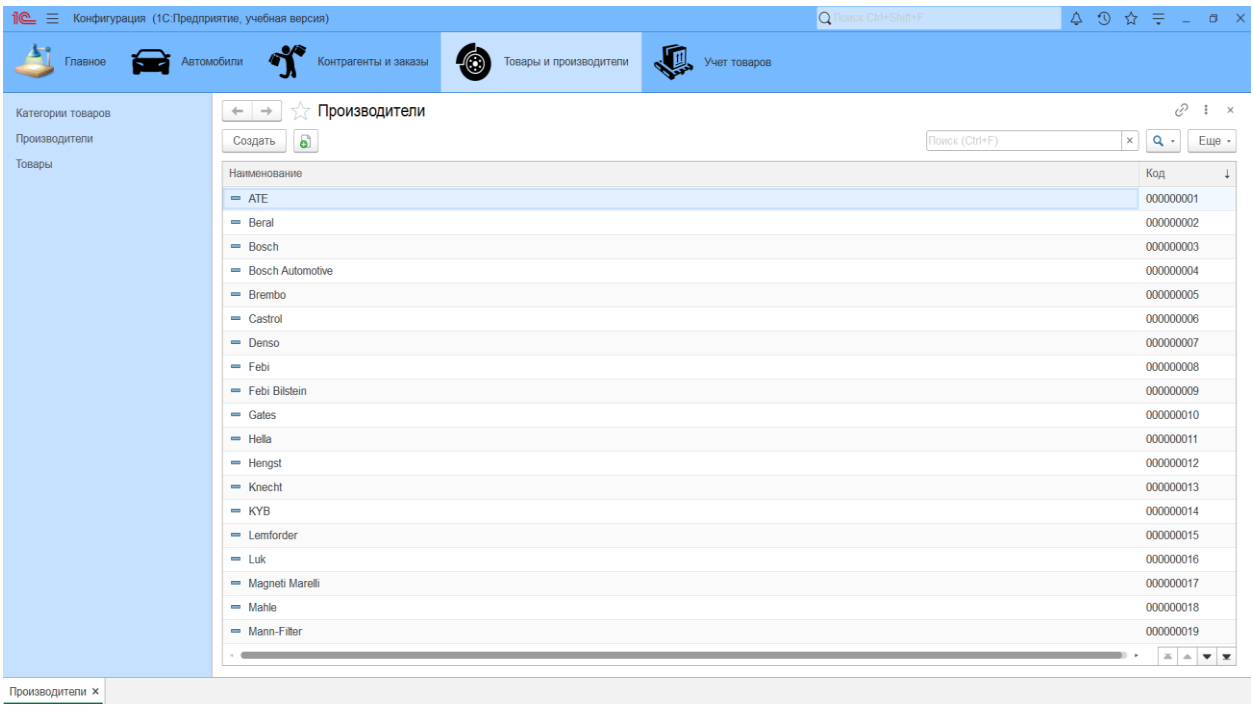


Рисунок 29 – Форма списка справочника «Производители»

Все возможные группировки товаров (категории), представленные в магазине, собраны в справочнике «Категории товаров». Этот справочник входит в состав подсистемы «Товары и производители». Иллюстрация 30 показывает внешний вид формы данного справочника.

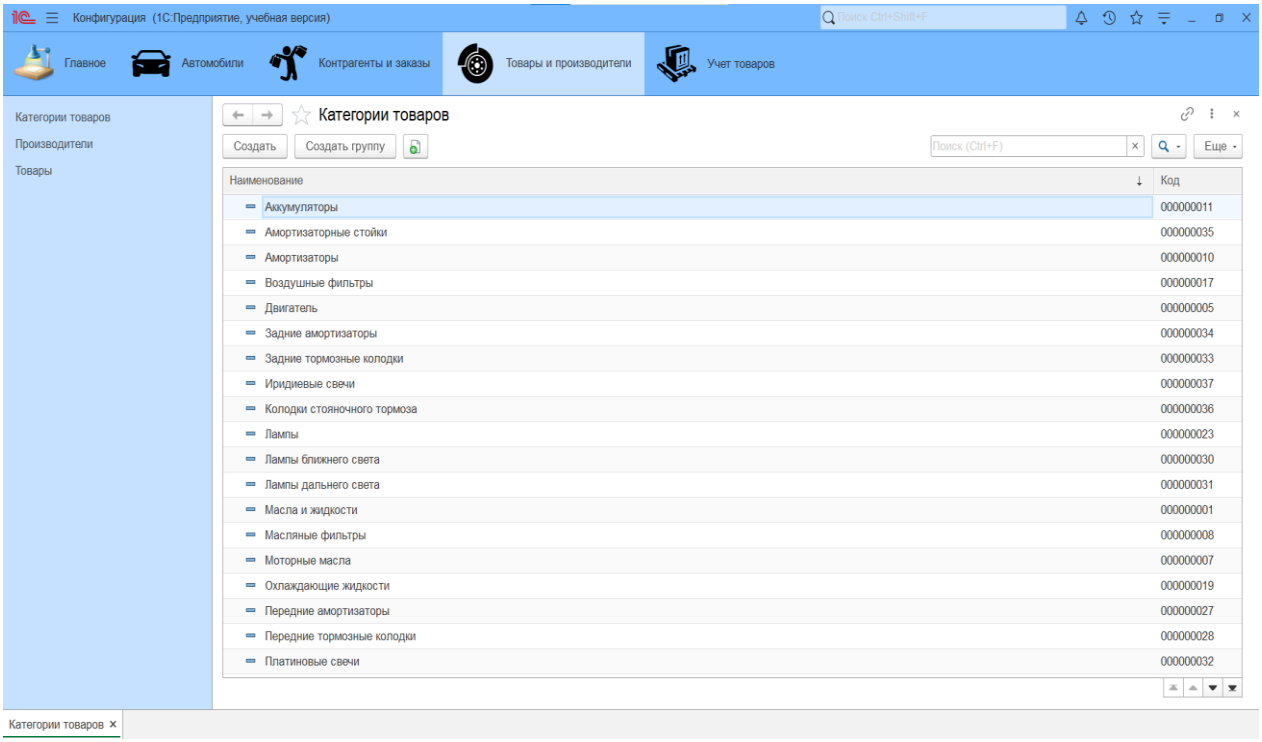


Рисунок 30 – Форма списка справочника «Категории товаров»

В справочнике «Товары» хранятся сведения обо всей номенклатуре, представленной в онлайн-магазине. Данный справочник расположен внутри подсистемы «Товары и производители». Как устроена его форма, демонстрирует рисунок 31.

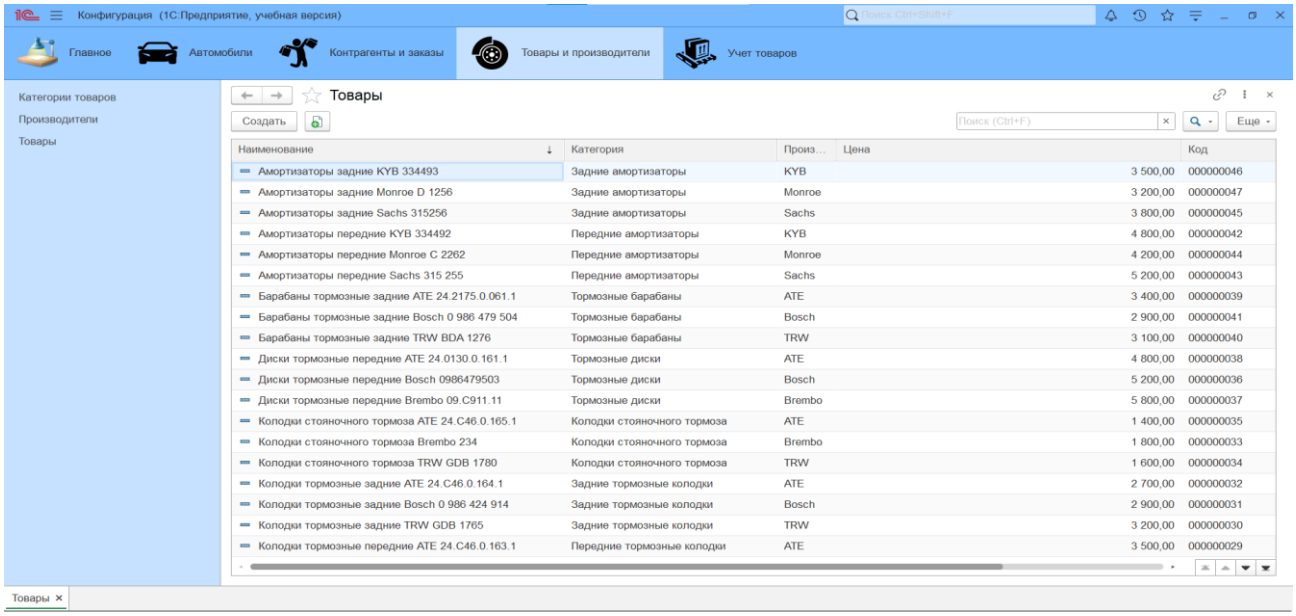


Рисунок 31 – Форма списка справочника «Товары»

Справочник под названием «Марки автомобилей» фиксирует данные обо всех марках транспортных средств, с которыми имеет дело интернет-магазин. Относится он к подсистеме «Автомобили». Рисунок 32 показывает форму данного справочника.

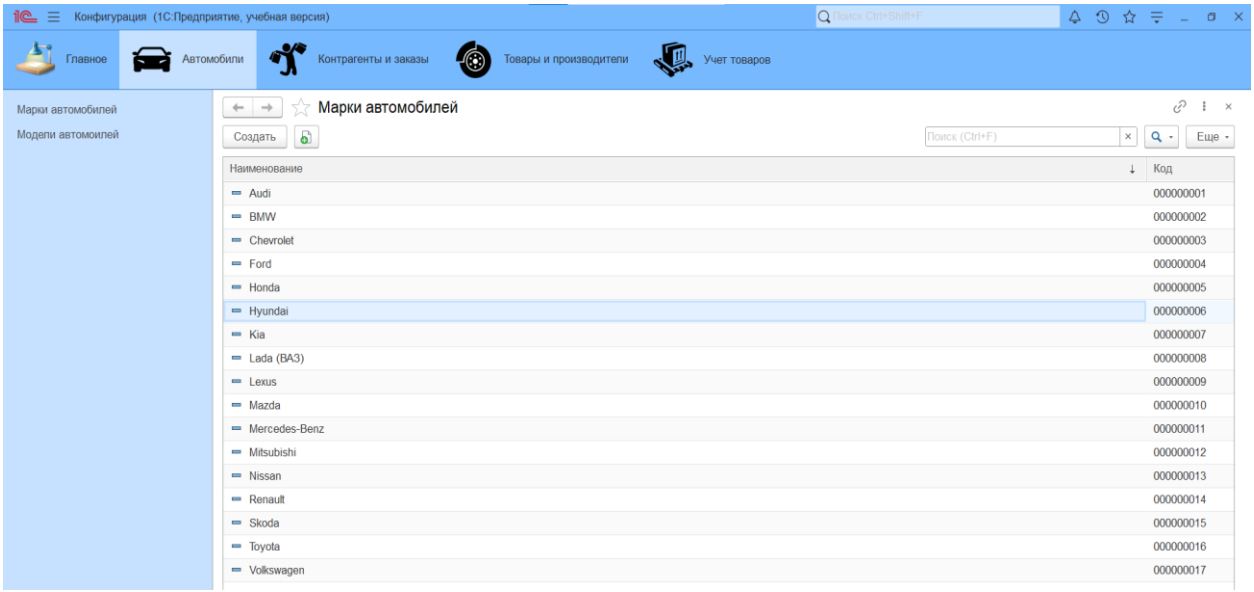


Рисунок 32 – Форма списка справочника «Марки автомобилей»

Данные о моделях машин, доступных в интернет-магазине, содержит справочник «Модели автомобилей». Он относится к подсистеме «Автомобили». Рисунок 33 иллюстрирует форму этого справочника.

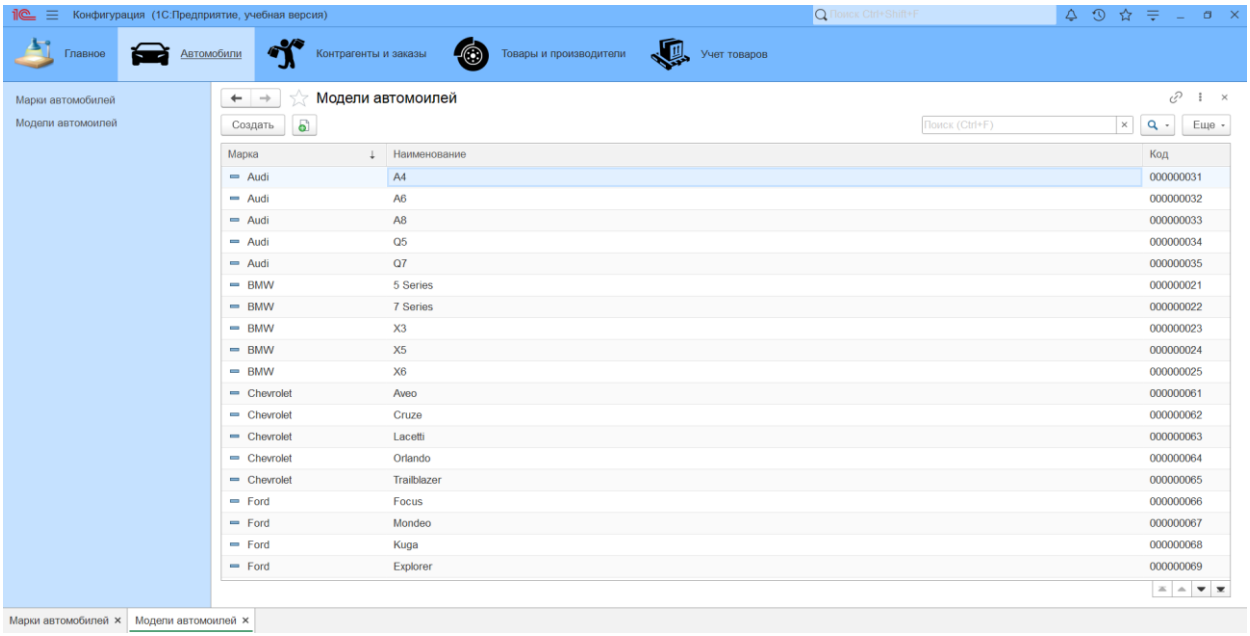


Рисунок 33 – Форма списка справочника «Модели автомобилей»

Данные о статусах, которыми могут обладать заказы пользователей содержит справочник «Статусы заказов», справочник принадлежит и является доступным в подсистеме «Контрагенты и заказы». На рисунке 34 продемонстрирована форма списка справочника «Статусы заказов».

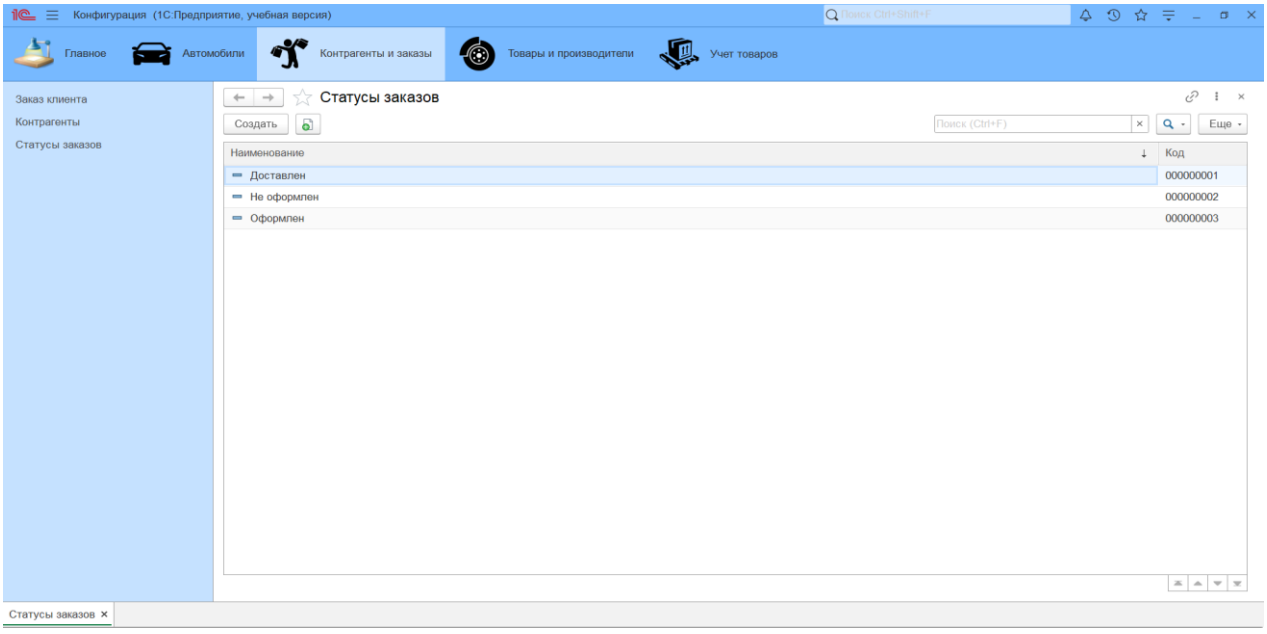


Рисунок 34 – Форма списка справочника «Статусы заказов»

Данные о складах, где лежат товары, содержит справочник «Склады». Относится он к подсистеме «Товары и производители». Рисунок 35 показывает, как выглядит форма этого справочника.

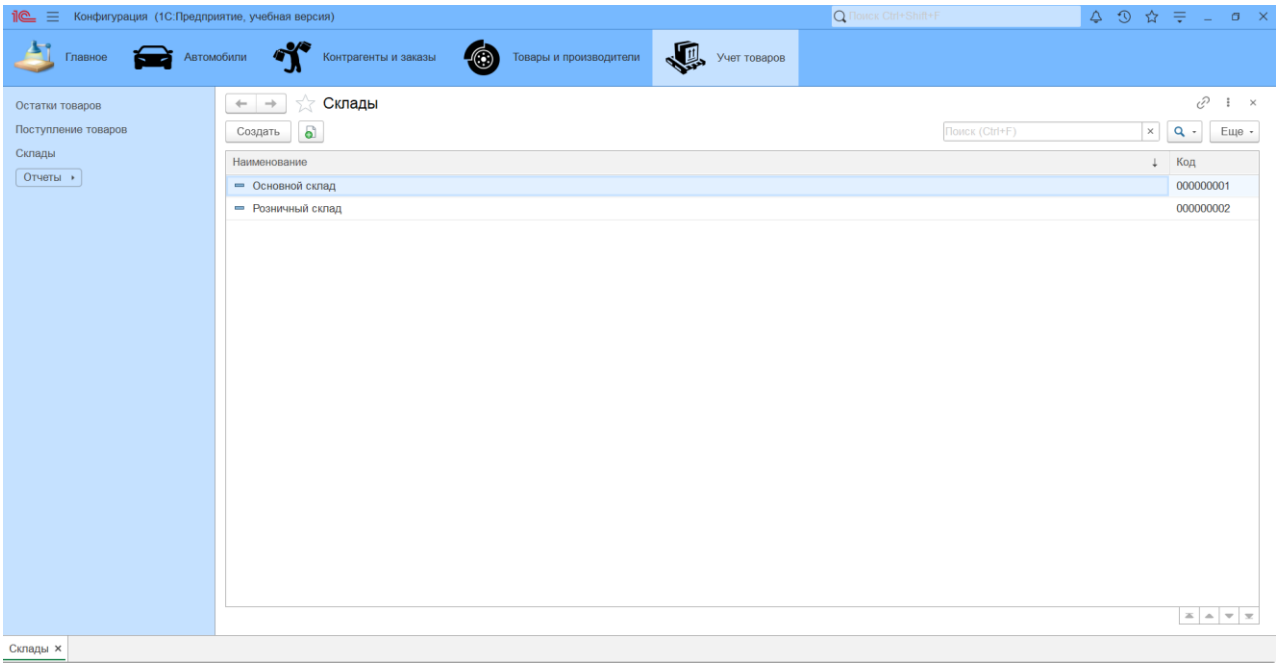


Рисунок 35 – Форма списка справочника «Склады»

Все перечисленные справочники образуют фундамент системы и выступают в роли связующих элементов, которые обеспечивают взаимодействие между всеми объектами конфигурации.

Документы фиксируют т отображают в системе полную информацию обо всех операциях и процессах, которые имеют место в деятельности компании. Иными словами, любой факт хозяйственной жизни организации, будь то поступление товаров на склад, их продажа, списание или перемещение между складами, должен быть зафиксирован с помощью соответствующего документа.

Для создаваемой автоматизированной информационной системы разработаны следующие справочники:

– «Заказ клиента»;

– «Поступление товаров».

Один из созданных документов имеет название «Заказ клиента». Его основное предназначение – фиксировать в системе тот момент, когда покупатель оформляет заявку, а также отслеживать все этапы её последующей обработки.

Получить доступ к данному документу можно через специальную подсистему, которая именуется «Контрагенты и заказы». Что касается внешнего оформления формы этого документа, то его можно посмотреть на рисунке 36.

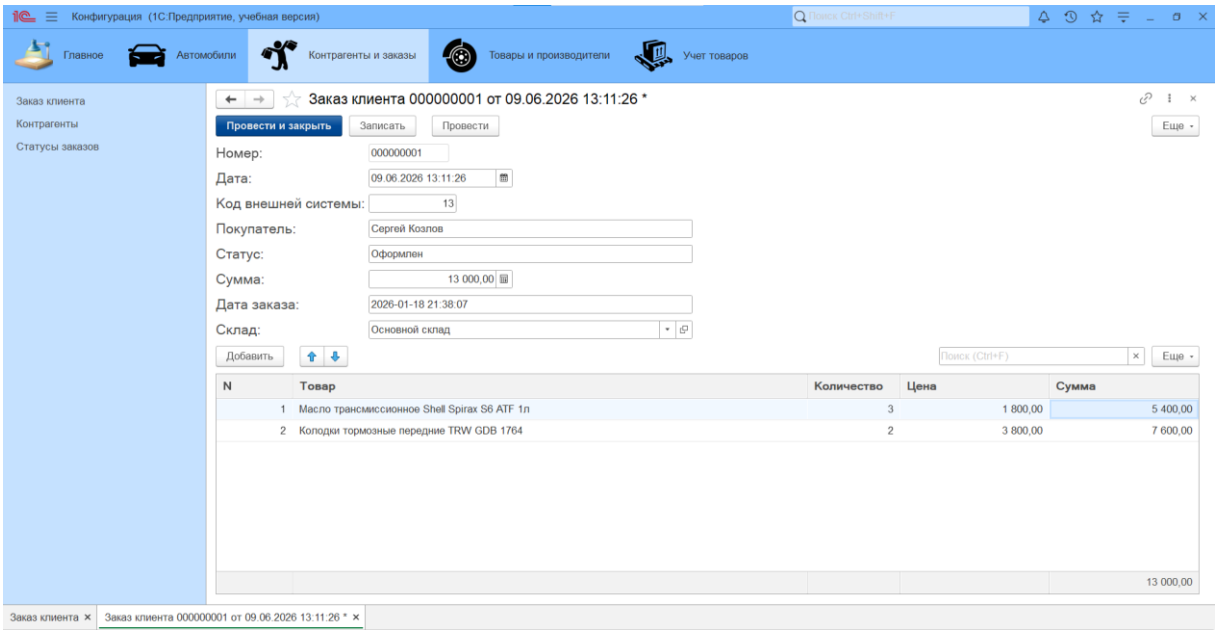


Рисунок 36 – Форма документа «Заказ клиента»

Документ называется «Поступление товаров» предназначен для фиксирования в системе факта прихода товарных позиций на склады, документ содержится в подсистеме «Учет товаров». Внешний вид формы документа «Поступление товаров» можно увидеть на рисунке 37.

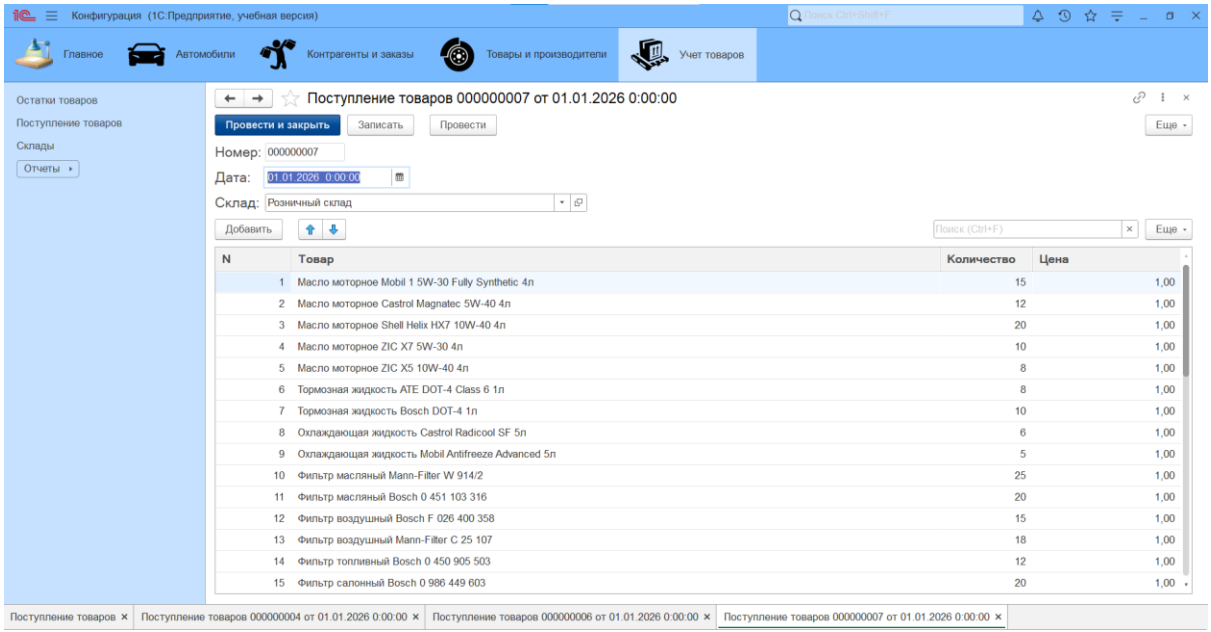


Рисунок 37 – Форма документа «Поступление товаров»

Регистры накопления в конфигурации отвечают за учетную информацию, которая возникает при проведении документов. По своей сути это числовое хранилище, через которое легко получать нужные данные. Для создаваемой системы складского учета был спроектирован один такой регистр – «Остатки товаров».

Регистр «Остатки товаров» принадлежит к виду «Остатки». С его помощью ведется подсчет того, сколько товаров осталось на складах, его регистраторами служат документы «Заказ клиента» и «Поступление товаров». Внешний данного регистра продемонстрирован на рисунке 38.

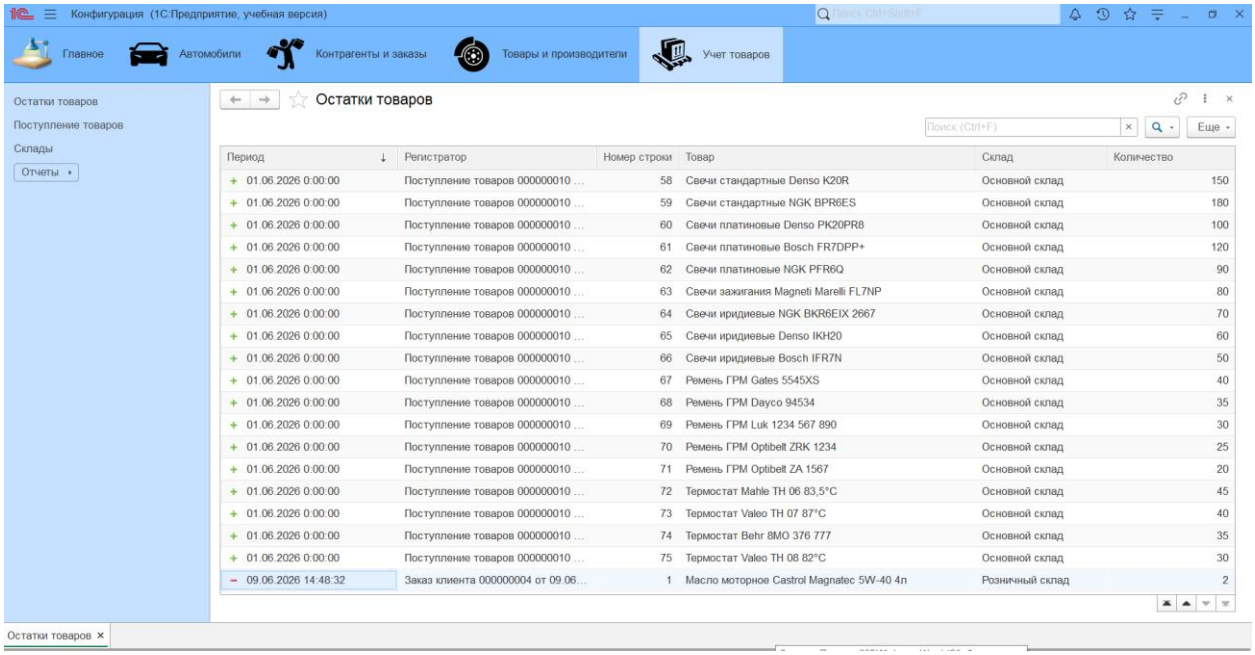


Рисунок 38 – Регистр накопления «Остатки товаров»

Отчёт – это не что иное, как наглядное визуальное представление тех хозяйственных операций, которые были предварительно зафиксированы в документах. Простыми словами, если документы служат для ввода и регистрации фактов (поступление товаров, заказ клиента), то отчёты нужны для того, чтобы увидеть эти факты в агрегированном, обработанном и удобном для восприятия виде.

Для разрабатываемой автоматизированной информационной системы был создан отчёт под названием «Остатки товаров». Данный отчёт выдаёт информацию об остатках по всему списку номенклатуры. При этом сведения

группируются по двум параметрам: по наименованиям товаров и по складам. Отчет «Остатки товаров» реализован при помощи схемы компоновки данных. Его интерфейс демонстрируется на рисунке 39. А запрос, выступающий в роли источника данных для указанной схемы компоновки этого отчета, представлен на рисунке 40.

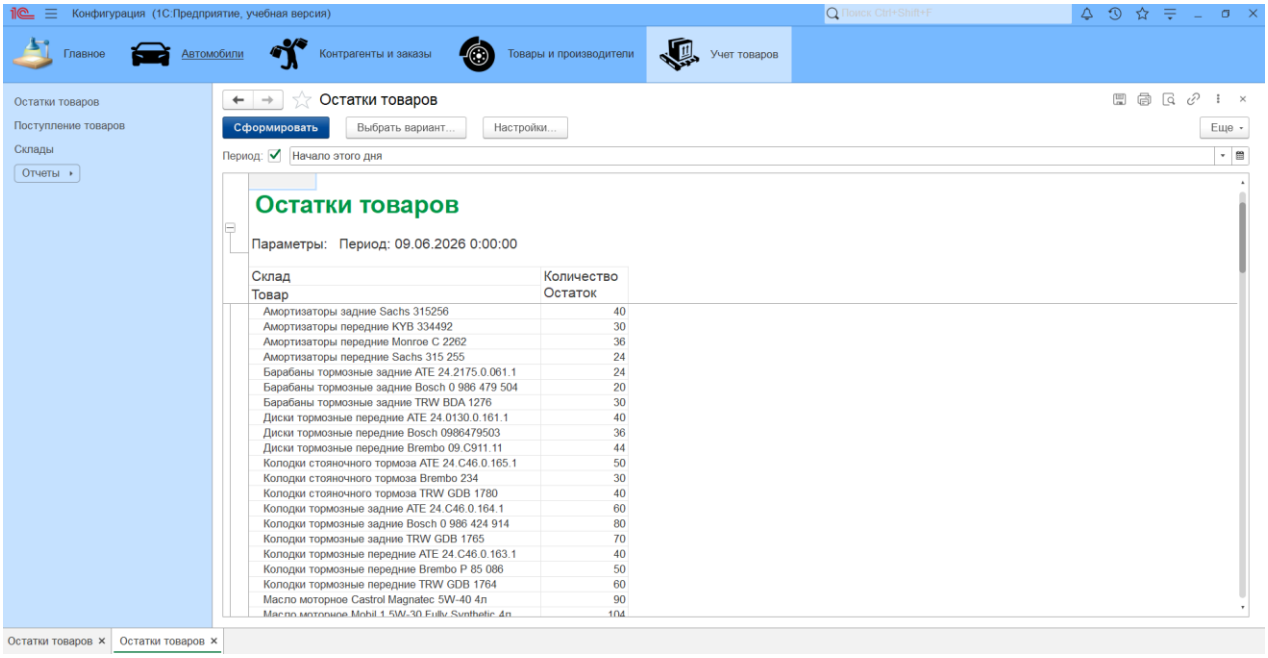


Рисунок 39 – Отчет «Остатки товаров»

Запрос:

```
ВЫБРАТЬ
    ОстаткиТоваровОстатки.Товар КАК Товар,
    ОстаткиТоваровОстатки.Склад КАК Склад,
    ОстаткиТоваровОстатки.КоличествоОстаток КАК КоличествоОстаток
ИЗ
    РегистрНакопления.ОстаткиТоваров.Остатки КАК ОстаткиТоваровОстатки
```

Рисунок 40 – Запрос для компоновки данных отчета «Остатки товаров»

Таким образом, общая архитектура системы насчитывает восемь справочников, два документа, один регистр накопления и один отчёт. Данный набор является достаточным для реализации заявленного функционала системы для ведения складского учета автозапчастей.

5 Контроль качества автоматизированной информационной системы

5.1 Функциональное тестирование

5.1.1 Классы эквивалентности

При тестировании применялся подход, основанный на классах эквивалентности. Суть метода в следующем: если на каком-либо одном представителе класса не обнаружено дефекта, то с большой вероятностью он не проявится и на всех остальных элементах той же группы.

К полю, предназначенному для ввода наименования новой автозапчасти, предъявляются такие требования:

– длина от 2 до 50 символов;

– содержит русские буквы (а-я, А-Я), латинские буквы (a-z, A-Z), цифры (0-9), дефис (-), пробел;

– должно начинаться с буквы (например, не может начинаться с дефиса (-)).

Классы эквивалентности для поля ввода наименования автозапчасти представлены в таблице 2.

Таблица 2 – Классы эквивалентности для поля ввода наименования

Параметр	**Правильные КЭ**	**Неправильные КЭ**
Длина наименования	2-50(1)	0-1(2) 51- ∞ (3)
Структура	Начинается с буквы(4)	Начинается не с буквы (5)
Содержание	Буквы Латинские (6) Буквы Русские (7) Цифры (8) Пробел (9) Дефис (10)	Остальное (11)

Тест-кейсы описанных классов эквивалентности для поля ввода наименования автозапчасти представлены в таблице 3.

Таблица 3 – Тест-кейсы для описанных классов эквивалентности

№	КЭ	Входные данные	Выходные данные
1.	1,4,6,7,8,9,10	Тормозная жидкость Beral DOT-4 1л	Успешно
2.	2,4,6	E	Ошибка: Меньше 2-х символов
3.	3,4,6,7,8,9,10	Масло моторное Mobil 1 5W-30 Fully Synthetic 4л Полностью синтетическое моторное масло для современных двигателей	Ошибка: Больше 50-ти символов
4.	1,5,7,10	-Мотор	Ошибка: Начинается не с буквы
5.	1,4,6,7,8,9,11	Охлаждающая жидкость Beral G12 5л!	Ошибка: Содержит запрещенные символы
6.	1,4,8,9,10	-2 Mobil	Ошибка: Не может не содержать русские буквы
7.	1,4,7,9	Тормозные колодки	Успешно
8.	1,4,7,10	Привод-редуктор	Успешно
9.	1,4,6,7,9	Масло моторное Mobil	Успешно

Представленные тест-кейсы покрывают все выделенные классы эквивалентности для поля ввода наименования автозапчасти.

5.1.2 Граничные условия

Граничными условиями называется набор математических ограничений, которые накладываются на дифференциальное уравнение на границах расчетной области [18]. В рамках этого тестирования нужно было записать границы поля в таблицу и составить недостающие тесты на границах. Границы поля наименования товара указаны в таблице 4.

Таблица 4 – Границы тестирования

Границы	Содержание
Физические	Строка отрицательной длины (null или отсутствие строки)
Логические	-
Технологические	Количество символов больше максимально возможного по типу данных
Произвольные	Наименование не больше 50 символов

Тест-кейсы по граничным условиям отображены в таблице 5.

Таблица 5 – Тест-кейсы для проверки граничных условий

№	Границы	Входные данные	Выходные данные
1.	Ф1 (отсутствие названия)	null	Ошибка: Наименование товара не может быть null
2.	Ф2 (Пустая строка «»), П1 (0 символов)	«» (пустая строка)	Ошибка: Наименование товара не может быть пустым
3.	П1 (51 символ)	«Масло моторное Mobil 1 5W-30 Fully Synthetic 4л Полностью синтетическое моторное масло для современных двигателей»	Ошибка: Превышена максимальная длина (макс. 50)
4.	П2 (1 символ)	«А» (1 символ)	Ошибка: Длинна меньше минимальной (мин. 2)
5.	П3 (2 символа)	«ГУ» (2 символа)	Успешно (минимальная допустимая длина)
6.	П4 (50 символов)	« ГУ » + «а»*48 (ровно 50 символов)	Успешно (максимальная допустимая длина)

Стоит подчеркнуть, что описанные методы охватывают всего лишь некоторую небольшую долю необходимых проверок, если подробней, они позволяют убедиться, что отдельные элементы программы работают правильно и их результат взаимодействия в процессе функционирования системы верен.

5.2 Нефункциональное тестирование

Чтобы определить корректно ли работает разработанное приложение интернет-магазина автозапчастей были созданы unit-тесты [19]. Во время выполнения тестирования было разработано два unit-теста для проверки функциональности иерархических запросов к таблицам CategoriesProducts (категории товаров) и Products (товары). Тестирование выполнялось с использованием фреймворка tSQLt, который обеспечивает изоляцию тестов и автоматический откат изменений после каждого теста.

Целью тестирования является проверка корректности работы иерархических запросов, строящих дерево категорий товаров. Тестирование направлено на оценку правильности построения многоуровневых иерархических структур и вычисления статистических показателей.

Результаты первого теста представлены на рисунке 41, код теста приведен в листинге 1.

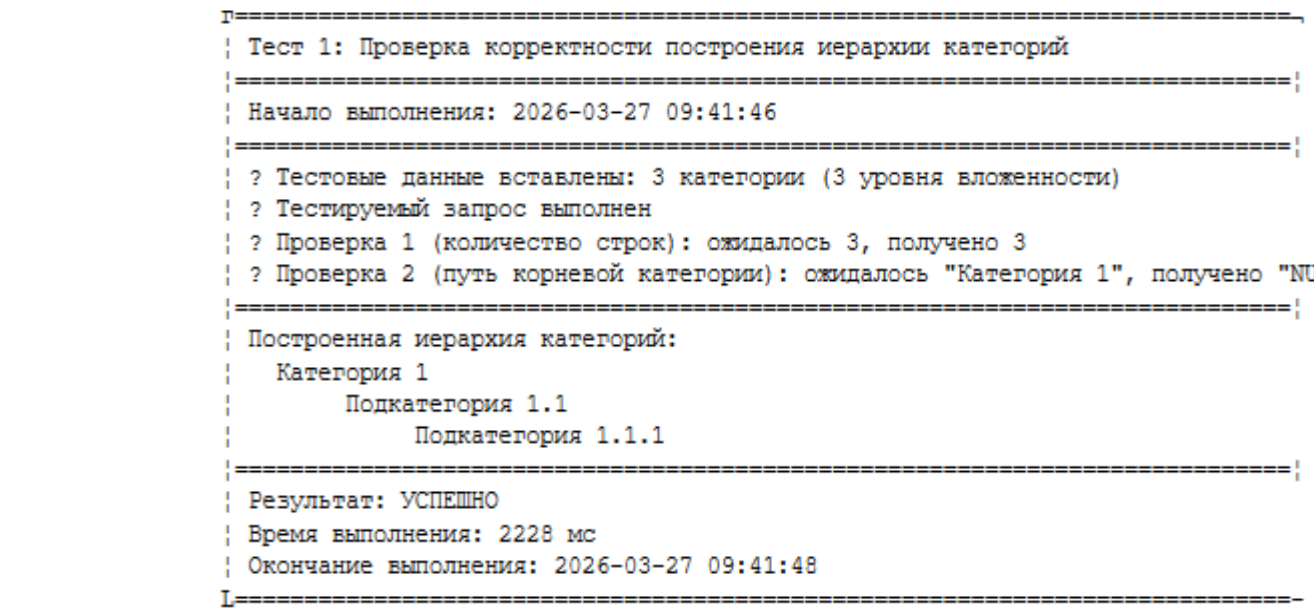


Рисунок 41 – Результаты теста 1

Листинг 1 – Код теста 1

```
WITH CategoryHierarchy AS (
    SELECT
        cp.CategoriesProductsId,
        cp.CategoryName,
        cp.ParentCategory_Id,
        cp.CategoryLevel,
        CAST(cp.CategoryName AS NVARCHAR(500)) AS FullPath,
        CAST(cp.CategoriesProductsId AS NVARCHAR(500)) AS IdPath
    FROM CategoriesProducts cp
    WHERE cp.ParentCategory_Id IS NULL
    UNION ALL

    SELECT
        c.CategoriesProductsId,
        c.CategoryName,
        c.ParentCategory_Id,
        c.CategoryLevel,
        CAST(ch.FullPath + ' > ' + c.CategoryName AS
NVARCHAR(500)),
        CAST(ch.IdPath + ' > ' + CAST(c.CategoriesProductsId AS
NVARCHAR(10)) AS NVARCHAR(500))
    FROM CategoriesProducts c
    INNER JOIN CategoryHierarchy ch ON c.ParentCategory_Id =
ch.CategoriesProductsId
)
SELECT * FROM CategoryHierarchy
ORDER BY FullPath;
```

В ходе выполнения теста 1 проверялась корректность построения многоуровневой иерархии категорий товаров. Тест подтвердил, что исходные данные корректно формируют древовидную структуру, начиная с корневых элементов, механизм рекурсии правильно обходит все уровни вложенности и вычисляет полный путь и путь идентификаторов для каждой категории, а все ожидаемые связи между родительскими и дочерними элементами установлены верно и ошибок нарушения ссылочной целостности не обнаружено.

Результаты второго теста представлены на рисунке 42, код теста приведен в листинге 2.

```
¦ Тест 2: Проверка корректности подсчёта количества подкатегорий
¦================================================================================¦
¦ Начало выполнения: 2026-03-27 09:38:01
¦================================================================================¦
¦ ? Тестовые данные вставлены: 4 категории
¦   - Родитель (уровень 1)
¦   - Дочерняя 1 (уровень 2, Parent=Родитель)
¦   - Дочерняя 2 (уровень 2, Parent=Родитель)
¦   - Внучка 1 (уровень 3, Parent=Дочерняя 1)
¦ ? Тестируемый запрос выполнен
¦ ? Проверка (количество подкатегорий): ожидалось 2, получено 2
¦================================================================================¦
¦ Структура иерархии:
¦   Родитель (должно быть 2 подкатегории)
¦   +- Дочерняя 1
¦   ¦  L- Внучка 1
¦   L- Дочерняя 2
¦================================================================================¦
¦ Результат: УСПЕШНО
¦ Время выполнения: 97 мс
¦ Окончание выполнения: 2026-03-27 09:38:01
```

Рисунок 42 – Результаты теста 2

Листинг 2 – Код теста 2

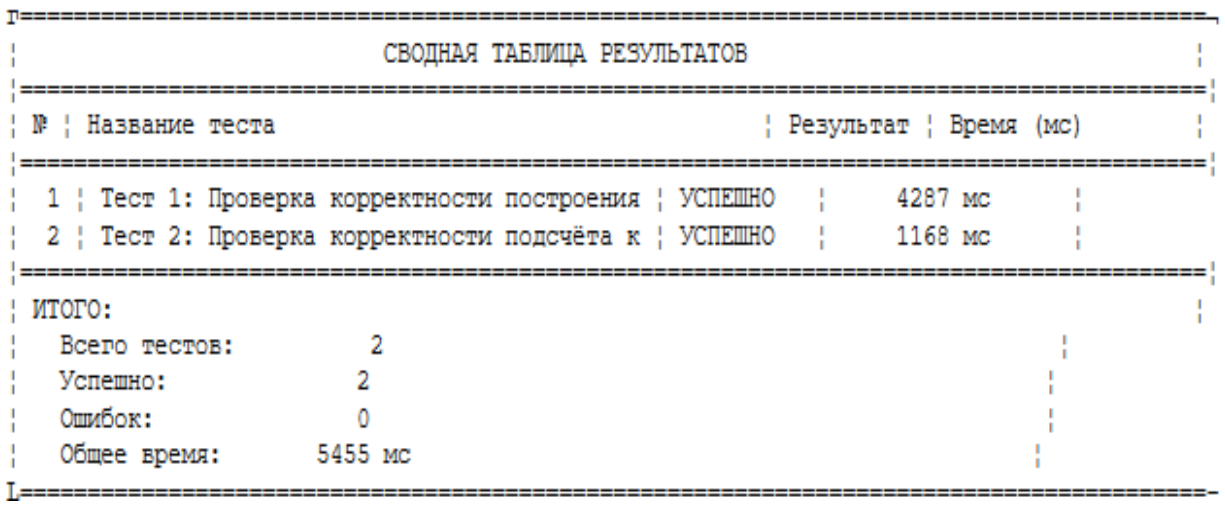
```
WITH CategoryHierarchy AS (
    SELECT
        cp.CategoriesProductsId,
        cp.CategoryName,
        cp.ParentCategory_Id,
        cp.CategoryLevel,
        CAST(cp.CategoryName AS NVARCHAR(500)) AS FullPath
    FROM CategoriesProducts cp
    WHERE cp.ParentCategory_Id IS NULL

    UNION ALL
```

```
    SELECT
        c.CategoriesProductsId,
        c.CategoryName,
        c.ParentCategory_Id,
        c.CategoryLevel,
        CAST(ch.FullPath + ' > ' + c.CategoryName AS
NVARCHAR(500))
    FROM CategoriesProducts c
    INNER JOIN CategoryHierarchy ch ON c.ParentCategory_Id =
ch.CategoriesProductsId
)
SELECT
    CategoryName,
    (SELECT COUNT(*) FROM CategoriesProducts WHERE
ParentCategory_Id = ch.CategoriesProductsId) AS SubcategoriesCount
FROM CategoryHierarchy ch
WHERE CategoryName = 'Родитель';
```

Тест оценивал корректность расчета количества дочерних подкатегорий для заданной родительской категории. Результаты теста показали, что рекурсивный запрос правильно идентифицирует все дочерние элементы на всех уровнях вложенности, функция агрегации с условием возвращает точное количество прямых подкатегорий для каждого узла, для корневых и промежуточных категорий значения соответствуют реальной структуре данных, ошибок в подсчете не зафиксировано.

Сводная таблица результатов, построенная в Microsoft SQL Server представлена на рисунке 43.



СВОДНАЯ ТАБЛИЦА РЕЗУЛЬТАТОВ

№	Название теста	Результат	Время (мс)
1	Тест 1: Проверка корректности построения	УСПЕШНО	4287 мс
2	Тест 2: Проверка корректности подсчёта к	УСПЕШНО	1168 мс

ИТОГО:

Всего тестов:	2
Успешно:	2
Ошибок:	0
Общее время:	5455 мс

Рисунок 43 – Анализ выполнения тестов

В таблице 6 представлены результаты прохождения каждого теста.

Таблица 6 – Функциональная спецификация unit-тестов

№ Теста	**Время выполнения**	**Количество проверок**	**Результат**	**Время выполнения**
Тест 1	Проверка корректности иерархии	2 проверки	Успешно	4287 мс
Тест 2	Проверка количества подкатегорий	2 проверки	Успешно	1168 мс

Разработанные unit-тесты на базе фреймворка tSQLt подтвердили корректность работы иерархических запросов к таблицам CategoriesProducts и Products. Оба теста завершились успешно, каждая проверка подтвердила ожидаемое поведение системы. Реализованные в MS SQL Server иерархические запросы полностью соответствуют функциональным требованиям и могут использоваться для построения каталога товаров интернет-магазина автозапчастей без дополнительной доработки.

5.3 Анализ качества кода

Для определения характеристик и критериев качества программного обеспечения применяются метрики. Метрика представляет собой измерительную систему для оценки качества программного обеспечения. Такие измерения могут проводиться на двух уровнях. Первый – уровень критериев, который позволяет напрямую сопоставлять программы по качеству, но требует субъективных оценок свойств, второй – уровень отдельных характеристик, дающий возможность получать объективные и точные данные.

Анализируемым кодом является метод AddToCart на языке программирования C#, который координирует добавление заказа, обновление данных в БД с обработкой ошибок и откатом транзакций. Код данного метода приведен в листинге 3.

Листинг 3 – Метод AddToCart

```
private async Task AddToCart(Product product, int quantity)
{
    try
    {
```

```
        var draftOrder = await GetOrCreateDraftOrder();
        if (draftOrder == null)
        {
            ShowErrorMessage("Не удалось создать заказ");
            return;
        }
        var orderItem = new OrderItem
        {
            Order_Id = draftOrder.OrderId,
            Product_Id = product.ProductId,
            Quantity = quantity,
            UnitPrice = product.Price
        };
        var response = await
_http.PostAsJsonAsync("api/orderitems", orderItem, _jsonOptions);
        if (response.IsSuccessStatusCode)
        {
            await UpdateOrderTotal(draftOrder.OrderId);
            ShowSuccessMessage($"Товар добавлен в
корзину!\n{product.ProductName}\nКоличество: {quantity} шт.\nНа
сумму: {product.Price * quantity:C2}");
        }
        else
        {
            ShowErrorMessage("Ошибка при добавлении товара в
корзину");
        }
    }
    catch (Exception ex)
    {
        ShowErrorMessage($"Ошибка: {ex.Message}");
    }
}
```

Расчет метрики SLOC на примере кода из листинга 3 представлен в таблице 7.

Таблица 7 – Расчет метрики SLOC для кода из листинга 3

Метрика	**Результат**
Физические строки кода	33
Логические строки кода	23
Исходные строки кода	33
Пустые строки	1

Полученное значение метрики подтверждает, что код функции компактен и обладает невысокой сложностью, что делает его удобным для анализа и сопровождения.

Традиционно размер программы оценивается количеством строк (операторов) исходного кода. Такая оценка носит номинальный характер, она позволяет классифицировать программы по объему, но недостаточна для точного определения сложности. При близких размерах программ решающее значение приобретают другие факторы, влияющие на сложность.

К группе оценок размера программ можно отнести метрику Холстеда [20].

В основе метрики Холстеда лежат четыре измеряемых характеристики программного комплекса:

– n1 – количество уникальных операторов программы, включающее в себя символы разделители, имена процедур и знаки операций (словарь операторов);

– n2 – количество уникальных операндов программы, по-другому словарь операндов;

– N1 – общее число операторов в программе;

– N2 – общее число операндов в программе.

Таблица 8 содержит результаты расчета количества операторов и операндов для тестируемого кода метод AddToCart.

Таблица 8 – Определение значений переменных

Обозначение	**Количество**	**Описание**
n1	39	Число уникальных операторов
n2	21	Число уникальных операндов
N1	85	Общее число операторов
N2	28	Общее число операндов

Ниже произведен расчет метрики Холстеда на примере кода из листинга 3.

Длина кода программы рассчитывается по формуле:

$$N = N1 + N2 = 85 + 28 = 113,$$

где N – длина программы;

N_1 – общее число операторов в программе;

N_2 – общее число операндов в программе.

Словарь программы рассчитывается по формуле:

$$\eta = \eta_1 + \eta_2 = 39 + 21 = 60,$$

По метрике Холстеда вычислительная сложность программы рассчитывается по следующей формуле:

$$V = N \log_2 \eta = 113 \times 5.91 = 667.8,$$

где V – вычислительная сложность программы по метрике Холстеда;

N – длина программы;

η – словарь программы.

Сложность кода программы вычисляется по формуле:

$$D = \frac{\eta_1}{2} \times \frac{N_2}{\eta_2} = 26,$$

где D – сложность кода программы;

N_2 – общее число операндов в программе;

η_1 – число уникальных операторов программы;

η_2 – число уникальных операндов программы.

Усилие для написания и понимания кода рассчитывается по формуле:

$$E = D \times V = 26 \times 667.8 = 17362.8,$$

где E – усилие для написания и понимания кода;

D – сложность кода программы;

V – вычислительная сложность программы по метрике Холстеда.

При оценке кода с помощью метрики Холстеда было определено, что сложность метода является умеренной, в коде сбалансировано количество операторов и переменных, а объем программы указывает на то, что метод легко прочитать и понять с первого раза, он не перегружен лишней логикой и все переменные используются по назначению. Удовлетворение требуемым ограничениям на значения метрик качества определяет код рассматриваемой программы как успешный вариант реализации.

Заключение

В ходе выполнения дипломного проекта была проделана следующая работа. На первом этапе изучена специфика предметной области, проанализированы существующие аналоги программных продуктов для онлайн-торговли автозапчастями, а также сформулированы функциональные и нефункциональные требования к создаваемой системе. Совокупность проведённых исследований и поставленных задач позволила определить стратегию дальнейшей разработки и избежать ошибок на ранних стадиях проектирования.

Следующим шагом стало проектирование архитектуры будущего приложения. Для этого задействовались специализированные CASE-инструменты и различные типы UML-диаграмм. В их число вошли: диаграмма вариантов использования, диаграмма классов, диаграммы компонентов, а также логическая модель данных схема базы данных. Каждая из перечисленных нотаций позволила рассмотреть систему с определённого ракурса и детально проработать её элементы

Отдельное внимание в работе было уделено обоснованию выбора технологического стека и средств разработки. Принимая во внимание специфику будущей эксплуатации и бюджетные ограничения, предпочтение отдавалось решениям, обеспечивающим достаточную производительность, масштабируемость и простоту внедрения.

Завершающим этапом работы стало тестирование созданного продукта. Проверка проводилась в двух направлениях: функциональном и автоматизированном. При функциональном тестировании применялись методы классов эквивалентности и анализа граничных значений. Для автоматизированной проверки использовались модульные тесты (unit-тесты). Кроме того, были проанализированы метрики кода, что дало представление о качестве написанного программного кода.

Список использованных источников

1. Гради, Б. Язык UML. Руководство пользователя / Б. Гради, Р. Джеймс, Я. Ивар. – М., 2006. – 487 с.

2. Microsoft. Документация по SQL Server [Электронный ресурс]. – Режим доступа: https://learn.microsoft.com/ru-ru/sql/sql-server/ (дата обращения: 14.05.2026).

3. Троелсен, Э. Язык программирования C# и платформа .NET 6 / Э. Троелсен, Ф. Джепикс. – М. : Вильямс, 2022. – 1440 с.

4. Swagger. Документация по OpenAPI (Swagger) [Электронный ресурс]. – Режим доступа: https://swagger.io/docs/ (дата обращения: 15.05.2026).

5. Лаптев, В. В. 1С:Предприятие. Практическая разработка / В. В. Лаптев, А. В. Морозов. – М. : 1С-Паблишинг, 2020. – 512 с.

6. Обзор систем автоматизации торговли [Электронный ресурс] // TAdviser. – Режим доступа: https://www.tadviser.ru/ (дата обращения: 16.05.2026)

7. Варианты использования [Электронный ресурс] // Диаграмма вариантов использования. – Режим доступа: http://www.telenir.net/uchebniki/samouchitel_uml/p4.php (дата обращения: 28.05.2026).

8. Рамбо, Дж. UML 2.0. Объектно-ориентированное моделирование и разработка / Дж. Рамбо, М. Блаха. – Санкт-Петербург : Питер, 2007. – 544 с.

9. Многоуровневая архитектура в ASP.NET MVC 5 // Metanit [Электронный ресурс]. URL: https://metanit.com/sharp/mvc5/23.5.php (дата обращения: 28.05.2026).

10. Простое руководство по диаграмме компонентов // Creately [Электронный ресурс]. URL: https://creately.com/blog/ru/uncategorized–ru/учебное–пособие–по–компонентной–диаг/ (дата обращения: 28.05.2026).

11. Либерти, Дж. ASP.NET Core. Разработка приложений / Дж. Либерти, Д. Хорвиц, Б. Хейтман ; перевод с английского. – Санкт-Петербург : Питер, 2020. – 928 с.

12. F5 Networks. Что такое HTTP? [Электронный ресурс] // F5 Cloud Docs. – Режим доступа: https://f5clouddocs.vercel.app/ru/terminology/http.html (дата обращения: 31.05.2026).

13. Диаграмма классов UML: руководство с примерами [Электронный ресурс] // Lucidchart. – Режим доступа: https://www.lucidchart.com/pages/ru/диаграмма-классов-uml (дата обращения: 31.05.2026).

14. Диаграмма Ганта: полное руководство по созданию и использованию // Мегаплан [Электронный ресурс]. URL: https://megaplan.ru/blog/crm/diagramma-ganta-polnoe-rukovodstvo/ (дата обращения: 2.06.2026).

15. Архитектура платформы 1С: Предприятие // Концепция пользовательского интерфейса [Электронный ресурс]. https://v8.1c.ru/platforma/kontseptsiya-polzovatelskogo-interfeysa/ обращения: 01.06.2024).

16. Радченко, М. Г. 1С:Предприятие 8.3. Практическое пособие разработчика. Примеры и типовые приемы / М. Г. Радченко, Е. Ю. Хрусталева. – М. : 1С-Паблишинг, 2013. – 964 с.

17. Петров, В. Н. Информационные системы и технологии управления базами данных : учебное пособие / В. Н. Петров. – Санкт-Петербург : Питер, 2019. – 288 с.

18. Петров, В. Н. Информационные системы и технологии управления базами данных : учебное пособие / В. Н. Петров. – Санкт-Петербург : Питер, 2019. – 288 с.

19. Unit-тестирование // SkillFactory [Электронный ресурс]. URL: https://blog.skillfactory.ru/glossary/unit-testirovanie/ (дата обращения: 02.05.2026).

20. Применение метрик Холстеда для количественного оценивания характеристик программ ЭВМ: [Электронный ресурс] // cyberleninka.ru. Теория.URL: https://cyberleninka.ru/article/n/primenenie-metrik-holsteda-dlya kolichestvennogo-otsenivaniya-harakteristik-programm-evm. (Дата обращения: 02.06.2026).

Приложение А

(справочное)

Глоссарий

Автоматизированная информационная система (АИС) – совокупность программных и аппаратных средств, предназначенных для автоматизации процессов сбора, хранения, обработки и передачи информации в рамках предметной области.

Актер (Actor) – внешняя сущность (пользователь, другая система, устройство), взаимодействующая с информационной системой и инициирующая выполнение определенных функций.

API (Application Programming Interface) – набор правил, протоколов и инструментов, определяющих способы взаимодействия между программными компонентами или приложениями.

Бизнес-процесс – последовательность взаимосвязанных действий или операций, выполняемых для достижения конкретной цели в рамках деятельности организации.

Вариант использования (Use Case) – описание набора действий, выполняемых системой в ответ на запрос актера, приводящее к измеримому результату, значимому для этого актера.

Диаграмма Ганта – тип диаграммы, используемый для визуализации графика выполнения задач проекта, отображающий длительность, последовательность и взаимосвязи работ.

Диаграмма классов – тип диаграммы UML, отображающий статическую структуру объектно-ориентированной системы, включая классы, их атрибуты, методы и связи между ними.

Диаграмма компонентов – это UML-схема, которая демонстрирует деление программы на крупные составные части и существующие между ними взаимосвязи.

Диаграмма прецедентов (диаграмма вариантов использования) – тип диаграммы UML, представляющий функциональные требования к системе через описание взаимодействия между актерами и вариантами использования.

Документ (в 1С) – объект конфигурации, предназначенный для отражения хозяйственной операции во времени (поступление товаров, заказ клиента) и формирования движений по регистрам.

Единый язык моделирования (UML, Unified Modeling Language) – графический язык визуализации, описания, конструирования и документирования объектно-ориентированных систем.

Информационная система – совокупность методов и средств, используемая для хранения, обработки и выдачи информации для достижения поставленной цели.

Иерархический запрос – рекурсивный SQL-запрос, использующий обобщенное табличное выражение (CTE) для обхода древовидных структур данных, хранящихся в реляционной таблице.

Класс эквивалентности – множество входных данных, которые предположительно обрабатываются программой одинаковым образом, что позволяет сократить количество тестовых случаев.

Логическая модель данных – абстрактное представление структуры данных, описывающее сущности, их атрибуты и связи между ними без привязки к конкретной системе управления базами данных.

Метрика качества программного обеспечения – измеримая характеристика программного продукта, используемая для количественной оценки его свойств, в которые входят сложность и надежность.

Метрика SLOC (Source Lines of Code) – метрика размера программы, измеряемая количеством строк исходного кода (физических, логических, пустых, исходных).

Метрика Холстеда – метрика сложности программного обеспечения, основанная на подсчете количества уникальных операторов, операндов и их вхождений в коде программы.

Многоуровневая архитектура – способ организации программного обеспечения, при котором система разделяется на независимые уровни

(представления, бизнес-логики, доступа к данным), каждый из которых решает определенный круг задач.

Обобщенное табличное выражение (CTE, Common Table Expression) – временный результирующий набор данных, который можно ссылаться в рамках выполнения одного SQL-запроса, особенно полезный для построения рекурсивных иерархических запросов.

Платформа «1С:Предприятие 8.3» – технологическая платформа для разработки и использования прикладных решений автоматизации деятельности организаций.

Регистр накопления (в 1С) – объект конфигурации, предназначенный для хранения информации о движении ресурсов (остатках, оборотах) в разрезе различных измерений.

Регистратор (в 1С) – документ или другой объект, который является источником записей (движений) в регистре накопления.

Справочник (в 1С) – объект конфигурации ссылочного типа, предназначенный для хранения условно-постоянной информации, используемой в других объектах системы (товары, контрагенты, склады).

Тест-кейс – артефакт тестирования, описывающий набор входных данных, ожидаемый результат и условия выполнения для проверки конкретной функциональности.

Фреймворк – программная платформа, определяющая структуру и архитектуру приложения, предоставляющая готовые компоненты и механизмы для упрощения разработки.

Функциональное тестирование – тестирование, направленное на проверку корректности реализации функциональных требований к системе.

Юнит-тест (Unit-тест) – автоматизированный тест, проверяющий корректность работы отдельного, изолированного модуля (функции, метода) программы.

tSQLt – специализированный фреймворк для модульного тестирования объектов баз данных Microsoft SQL Server.

Приложение Б

(обязательное)

Листинг программы

Листинг Б.1 – Код модуля документа «Заказ клиента»

```
Процедура ОбработкаПроведения(Отказ, РежимПроведения)
    Движения.ОстаткиТоваров.Записывать = Истина;
    Для каждого СтрокаТЧ Из Товары Цикл
        ТекущийОстаток = ПолучитьОстатокТовара(СтрокаТЧ.Товар);
        Если ТекущийОстаток < СтрокаТЧ.Количество Тогда
            Сообщить("Ошибка: Недостаточно товара '" +
СтрокаТЧ.Товар.Наименование + "' на складе!");
            Сообщить("  Доступно: " + ТекущийОстаток + ",
требуется: " + СтрокаТЧ.Количество);
            Отказ = Истина;
            Возврат;
        КонецЕсли;
        Движение = Движения.ОстаткиТоваров.Добавить();
        Движение.ВидДвижения = ВидДвиженияНакопления.Расход;
        Движение.Период = Дата;
        Движение.Товар = СтрокаТЧ.Товар;
        Движение.Склад = Склад;
        Движение.Количество = СтрокаТЧ.Количество;
    КонецЦикла;
КонецПроцедуры

// Функция для получения текущего остатка товара
Функция ПолучитьОстатокТовара(Товар)
    Запрос = Новый Запрос;
    Запрос.Текст =
     "ВЫБРАТЬ
     |    ОстаткиТоваровОстатки.КоличествоОстаток КАК Остаток
     |ИЗ
     |    РегистрНакопления.ОстаткиТоваров.Остатки КАК
ОстаткиТоваровОстатки
     |ГДЕ
    |    ОстаткиТоваровОстатки.Товар = &Товар
    |";
     Запрос.УстановитьПараметр("Товар", Товар);
    Результат = Запрос.Выполнить();
    Выборка = Результат.Выбрать();
    Если Выборка.Следующий() Тогда
        Возврат Выборка.Остаток;
    Иначе
        Возврат 0;
    КонецЕсли;
КонецФункции
```

Листинг Б.2 – Код модуля менеджера справочника «Товары»

```
Процедура ЗагрузитьВсеТовары() Экспорт
    Сообщить("Начинаем загрузку товаров...");
    Справочники.Производители.ЗагрузитьВсехПроизводителей();
    Справочники.Категории.ЗагрузитьВсеКатегории();
    Запрос = Новый Запрос;
    Запрос.Текст = "
```

```
    |ВЫБРАТЬ
    |    dbo_Products.ProductId КАК ProductId,
    |    dbo_Products.ProductName КАК ProductName,
    |    dbo_Products.Price КАК Price,
    |    dbo_Products.Manufacturer_Id КАК Manufacturer_Id,
    |    dbo_Products.CategoriesProducts_Id КАК Category_Id
    |ИЗ
    |    ВнешнийИсточникДанных.БД.Таблица.dbo_Products КАК
dbo_Products
    |";
    Результат = Запрос.Выполнить();
    Выборка = Результат.Выбрать();
    КоличествоЗагружено = 0;
    Пока Выборка.Следующий() Цикл
        // Ищем производителя
        ПроизводительСсылка =
Справочники.Производители.ПустаяСсылка();
        Если Выборка.Manufacturer_Id <> 0 Тогда
            ПроизводительСсылка =
Справочники.Производители.НайтиПоРеквизиту("КодВнешнейСистемы",
Выборка.Manufacturer_Id);
        КонецЕсли;
        КатегорияСсылка = Справочники.Категории.ПустаяСсылка();
        Если Выборка.Category_Id <> 0 Тогда
            КатегорияСсылка =
Справочники.Категории.НайтиПоРеквизиту("КодВнешнейСистемы",
Выборка.Category_Id);
        КонецЕсли;
        ТоварСсылка =
Справочники.Товары.НайтиПоРеквизиту("КодВнешнейСистемы",
Выборка.ProductId);

        Если ТоварСсылка = Справочники.Товары.ПустаяСсылка() Тогда
            НовыйТовар = Справочники.Товары.СоздатьЭлемент();
            НовыйТовар.КодВнешнейСистемы = Выборка.ProductId;
            НовыйТовар.Наименование = СокрЛП(Выборка.ProductName);
            Если ПроизводительСсылка <>
Справочники.Производители.ПустаяСсылка() Тогда
                НовыйТовар.Производитель = ПроизводительСсылка;
            КонецЕсли;
            Если КатегорияСсылка <>
Справочники.Категории.ПустаяСсылка() Тогда
                НовыйТовар.Категория = КатегорияСсылка;
            КонецЕсли;
            Попытка
                НовыйТовар.Записать();
                КоличествоЗагружено = КоличествоЗагружено + 1;
            Исключение
                Сообщить("Ошибка при записи товара " +
Выборка.ProductName + ": " + ОписаниеОшибки());
            КонецПопытки;
        Иначе
            ТоварОбъект = ТоварСсылка.ПолучитьОбъект();
```

```
            Если ТоварОбъект <> Неопределено Тогда
                ТоварОбъект.Наименование =
СокрЛП(Выборка.ProductName);
                Если ПроизводительСсылка <>
Справочники.Производители.ПустаяСсылка() Тогда
                    ТоварОбъект.Производитель =
ПроизводительСсылка;
                КонецЕсли;
                Если КатегорияСсылка <>
Справочники.Категории.ПустаяСсылка() Тогда
                    ТоварОбъект.Категория = КатегорияСсылка;
                КонецЕсли;
                ТоварОбъект.Записать();
                КоличествоЗагружено = КоличествоЗагружено + 1;
            КонецЕсли;
        КонецЕсли;
    КонецЦикла;
    Сообщить("Загружено товаров: " + КоличествоЗагружено);
КонецПроцедуры
```

Листинг Б.3 – Файл «AddDbContext.cs»

```
using Microsoft.EntityFrameworkCore;
using System.ComponentModel.DataAnnotations;
using System.ComponentModel.DataAnnotations.Schema;
namespace AutoShop.Data
{
    public class AppDbContext : DbContext
    {
        public AppDbContext(DbContextOptions<AppDbContext>
options) : base(options) { }
        // Таблицы аутентификации
        public DbSet<User> Users { get; set; }
        public DbSet<Role> Roles { get; set; }
        public DbSet<UserRole> UserRoles { get; set; }
        // Авто
        public DbSet<CarMake> CarMakes { get; set; }
        public DbSet<CarModel> CarModels { get; set; }
        // Каталог
        public DbSet<Manufacturer> Manufacturers { get; set; }
        public DbSet<CategoryProduct> CategoriesProducts { get;
set; }
        public DbSet<Product> Products { get; set; }
        // Совместимость товара и авто
        public DbSet<VehicleCompatibility> VehicleCompatibilities
{ get; set; }
        // Заказы
        public DbSet<OrderStatus> OrderStatuses { get; set; }
        public DbSet<Order> Orders { get; set; }
        public DbSet<OrderItem> OrderItems { get; set; }
        // Склад
        public DbSet<Warehouse> Warehouses { get; set; }
        public DbSet<StockItem> StockItems { get; set; }
        // Отзывы
```

```
        public DbSet<ProductReview> ProductReviews { get; set; }
        protected override void OnModelCreating(ModelBuilder
modelBuilder)
        {
            base.OnModelCreating(modelBuilder);
            modelBuilder.Entity<UserRole>()
                .HasKey(ur => new { ur.Users_Id, ur.Role_Id });
            modelBuilder.Entity<UserRole>()
                .HasOne(ur => ur.User)
                .WithMany(u => u.UserRoles
                .HasForeignKey(ur => ur.Users_Id);
            // .HasPrincipalKey(u => u.UserId);
            modelBuilder.Entity<UserRole>()
                .HasOne(ur => ur.Role)
                .WithMany(r => r.UserRoles
                .HasForeignKey(ur => ur.Role_Id);
            // .HasPrincipalKey(r => r.RoleId
            // VehicleCompatibilities: составной ключ Product_Id +
Model_Id
            modelBuilder.Entity<VehicleCompatibility>()
                .HasKey(vc => new { vc.Product_Id, vc.Model_Id });
            base.OnModelCreating(modelBuilder);
        }

        // Users
        public class User
        {
            [Key]
            [Column("UserId")] // Явно указываем имя столбца в БД
            public int UserId { get; set; }
            [Column("Email")]
            public string Email { get; set; } = "";

            [Column("PasswordHash")]
            public string PasswordHash { get; set; } = "";
            [Column("FirstName")]
            public string FirstName { get; set; } = "";
            [Column("LastName")]
            public string LastName { get; set; } = "";
            public ICollection<UserRole> UserRoles { get; set; } =
new List<UserRole>();
            public ICollection<Order> Orders { get; set; } = new
List<Order>();
            public ICollection<ProductReview> ProductReviews {
get; set; } = new List<ProductReview>();
        }

        // Roles
        public class Role
        {
            [Key] public int RoleId { get; set; }
            public string RoleName { get; set; } = "";
```

```
            // Навигационные свойства
            public ICollection<UserRole> UserRoles { get; set; } =
new List<UserRole>();
        }

        // UserRoles
        public class UserRole
        {
            [Column("Users_Id")] public int Users_Id { get; set; }
            public User User { get; set; } = null!;
            [Column("Role_Id")] public int Role_Id { get; set; }
            public Role Role { get; set; } = null!;
        }

        // CarMakes
        public class CarMake
        {
            [Key] public int CarMakeId { get; set; }
            public string MakeName { get; set; } = "";
            public ICollection<CarModel> CarModels { get; set; } =
new List<CarModel>();
        }

        // CarModels
        public class CarModel
        {
            [Key] public int CarModelId { get; set; }
            public string ModelName { get; set; } = "";
            [Column("CarMake_Id")]
            public int CarMake_Id { get; set; }
            [ForeignKey(nameof(CarMake_Id))]
            public CarMake CarMake { get; set; } = null!;
            public ICollection<VehicleCompatibility>
VehicleCompatibilities { get; set; } = new
List<VehicleCompatibility>();
        }

        // Manufacturers
        public class Manufacturer
        {
            [Key] public int ManufacturerId { get; set; }
            public string ManufacturerName { get; set; } = "";
            // Навигационные свойства
            public ICollection<Product> Products { get; set; } =
new List<Product>();
        }

        // CategoriesProducts
        public class CategoryProduct
        {
            [Key] public int CategoriesProductsId { get; set; }
            public string CategoryName { get; set; } = "";
            [Column("ParentCategory_Id")]
```

```
            public int? ParentCategory_Id { get; set; }
            public int CategoryLevel { get; set; }
            public int SortOrder { get; set; }
            // Навигационные свойства
            [ForeignKey(nameof(ParentCategory_Id))]
            public CategoryProduct? ParentCategory { get; set; }
            public ICollection<CategoryProduct> Subcategories {
get; set; } = new List<CategoryProduct>();
            public ICollection<Product> Products { get; set; } =
new List<Product>();
        }

        // Products
        public class Product
        {
            [Key] public int ProductId { get; set; }
            public string ProductName { get; set; } = "";
            public string? DescriptionProducts { get; set; }
            public decimal Price { get; set; }
            [Column("CategoriesProducts_Id")]
            public int CategoriesProducts_Id { get; set; }
            [Column("Manufacturer_Id")]
            public int Manufacturer_Id { get; set; }
            // Навигационные свойства
            [ForeignKey(nameof(CategoriesProducts_Id))]
            public CategoryProduct CategoryProduct { get; set; } =
null!;
            [ForeignKey(nameof(Manufacturer_Id))]
            public Manufacturer Manufacturer { get; set; } =
null!;
            public ICollection<VehicleCompatibility>
VehicleCompatibilities { get; set; } = new
List<VehicleCompatibility>();
            public ICollection<OrderItem> OrderItems { get; set; }
= new List<OrderItem>();
            public ICollection<StockItem> StockItems { get; set; }
= new List<StockItem>();
            public ICollection<ProductReview> ProductReviews {
get; set; } = new List<ProductReview>();
        }

        // VehicleCompatibilities
        public class VehicleCompatibility
        {
            [Column("Product_Id")] public int Product_Id { get;
set; }
            [ForeignKey("Product_Id")]
            public Product Product { get; set; } = null!;
            [Column("Model_Id")] public int Model_Id { get; set; }
            [ForeignKey("Model_Id")]
            public CarModel CarModel { get; set; } = null!;
        }
```

```
        // OrderStatuses
        public class OrderStatus
        {
            [Key] public int OrderStatusId { get; set; }
            public string StatusName { get; set; } = "";
            // Навигационные свойства
            public ICollection<Order> Orders { get; set; } = new
List<Order>();
        }

        // Orders
        public class Order
        {
            [Key] public int OrderId { get; set; }
            [Column("Users_Id")]
            public int Users_Id { get; set; }
            [Column("OrderStatus_Id")]
            public int OrderStatus_Id { get; set; }
            public DateTime OrderDate { get; set; }
            public decimal TotalAmount { get; set; }
            // Навигационные свойства
            [ForeignKey(nameof(Users_Id))]
            public User User { get; set; } = null!;
            [ForeignKey(nameof(OrderStatus_Id))]
            public OrderStatus OrderStatus { get; set; } = null!;

            public ICollection<OrderItem> OrderItems { get; set; }
= new List<OrderItem>();
        }

        // OrderItems
        public class OrderItem
        {
            [Key] public int OrderItemId { get; set; }
            [Column("Order_Id")]
            public int Order_Id { get; set; }
            [Column("Product_Id")]
            public int Product_Id { get; set; }
            public int Quantity { get; set; }
            public decimal UnitPrice { get; set; }
            [ForeignKey(nameof(Order_Id))]
            public Order Order { get; set; } = null!;
            [ForeignKey(nameof(Product_Id))]
            public Product Product { get; set; } = null!;
        }

        // Warehouses
        public class Warehouse
        {
            [Key] public int WarehouseId { get; set; }
            public string WarehouseName { get; set; } = "";
            // Навигационные свойства
```

```
            public ICollection<StockItem> StockItems { get; set; }
= new List<StockItem>();
        }

        // StockItems
        public class StockItem
        {
            [Key] public int StockItemId { get; set; }
            [Column("Product_Id")]
            public int Product_Id { get; set; }
            [Column("Warehouse_Id")]
            public int Warehouse_Id { get; set; }
            public int Quantity { get; set; }
            // Навигационные свойства
            [ForeignKey(nameof(Product_Id))]
            public Product Product { get; set; } = null!;
            [ForeignKey(nameof(Warehouse_Id))]
            public Warehouse Warehouse { get; set; } = null!;
         ProductReviews
        public class ProductReview
        {
            [Key]
            public int ProductReviewId { get; set; }
            [Column("Product_Id")]
            public int Product_Id { get; set; }

            [Column("Users_Id")]
            public int Users_Id { get; set; }

            public int Rating { get; set; }
            public string? Comment { get; set; }
            public DateTime CreatedDate { get; set; }

            // Навигационные свойства
            [ForeignKey(nameof(Product_Id))]
            public Product Product { get; set; } = null!;

            [ForeignKey(nameof(Users_Id))]
            public User User { get; set; } = null!;
        }
    }
}
```

Приложение В

(рекомендуемое)

Описание атрибутов модели базы данных

Таблица В.1 – Описание атрибутов сущности «Товары»

Атрибуты	**Нулевое значение**	**Ключевое значение**
ID_товара	Не допустимо	PK
Наименование_товара	Не допустимо	
Описание	Не допустимо	
Цена	Не допустимо	
ПроизводителиID_производителя	Не допустимо	FK
Категории_товаровID_категории	Не допустимо	FK

Таблица В.2 – Описание атрибутов сущности «Запись об остатке»

Атрибуты	**Нулевое значение**	**Ключевое значение**
ID_записи	Не допустимо	PK
Количество	Не допустимо	
СкладыID_склада	Не допустимо	FK
ТоварыID_товара	Не допустимо	FK

Таблица В.3 – Описание атрибутов сущности «Категории товаров»

Атрибуты	**Нулевое значение**	**Ключевое значение**
ID_категории	Не допустимо	PK
Наименование_категории	Не допустимо	
Уровень_иерархии	Не допустимо	
Порядок_сортировки_уровня	Не допустимо	
Категории_товаровID_категории	Допустимо	FK

Таблица В.4 – Описание атрибутов сущности «Совместимость с автомобилем»

Атрибуты	**Нулевое значение**	**Ключевое значение**
Марки_автомобилейID_марки_автомобиля	Не допустимо	PK
Модели_автомобилейID_модели_автомобиля	Не допустимо	PK

Таблица В.5 – Описание атрибутов сущности «Марки автомобилей»

Атрибуты	**Нулевое значение**	**Ключевое значение**
ID_марки_автомобиля	Не допустимо	PK
Наименование_марки_автомобиля	Не допустимо	

Таблица В.6 – Описание атрибутов сущности «Модели автомобилей»

Атрибуты	**Нулевое значение**	**Ключевое значение**
ID_модели_автомобиля	Не допустимо	PK
Наименование_модели	Не допустимо	
Марки_автомобилейID_марки_автомобиля	Не допустимо	FK

Таблица В.7 – Описание атрибутов сущности «Склады»

Атрибуты	**Нулевое значение**	**Ключевое значение**
ID_склада	Не допустимо	PK
Наименование_склада	Не допустимо	

Таблица В.8 – Описание атрибутов сущности «Пользователи»

Атрибуты	**Нулевое значение**	**Ключевое значение**
ID_пользователя	Не допустимо	PK
Email	Не допустимо	
Пароль	Не допустимо	
Имя	Не допустимо	
Фамилия	Не допустимо	

Таблица В.9 – Описание атрибутов сущности «Роли»

Атрибуты	**Нулевое значение**	**Ключевое значение**
ID_роли	Не допустимо	PK
Наименование_роли	Не допустимо	

Таблица В.10 – Описание атрибутов сущности «Связи пользователь-роль»

Атрибуты	**Нулевое значение**	**Ключевое значение**
РолиID_роли	Не допустимо	PK
ПользователиID_пользователя	Не допустимо	PK

Таблица В.11 – Описание атрибутов сущности «Заказы»

Атрибуты	**Нулевое значение**	**Ключевое значение**
ID_заказа	Не допустимо	PK
Дата_создания	Не допустимо	
Общая_сумма	Не допустимо	
ПользователиID_пользователя	Не допустимо	FK
Статусы_заказаID_статуса	Не допустимо	FK

Таблица В.12 – Описание атрибутов сущности «Позиция заказа»

Атрибуты	**Нулевое значение**	**Ключевое значение**
ID_позиции	Не допустимо	PK
Количество_товара	Не допустимо	
Цена_на_момент_заказа	Не допустимо	
ТоварыID_товара	Не допустимо	FK
ЗаказыID_заказа	Не допустимо	FK

Таблица В.13 – Описание атрибутов сущности «Отзывы о товарах»

Атрибуты	**Нулевое значение**	**Ключевое значение**
ID_отзыва	Не допустимо	PK
Рейтинг	Не допустимо	
Комментарий	Не допустимо	
Дата_отзыва	Не допустимо	
ТоварыID_товара	Не допустимо	FK
ПользователиID_пользователя	Не допустимо	FK

Таблица В.14 – Описание атрибутов сущности «Статусы заказа»

Атрибуты	**Нулевое значение**	**Ключевое значение**
ID_статуса	Не допустимо	PK
Наименование_статуса	Не допустимо	

Таблица В.15 – Описание атрибутов сущности «Производители»

Атрибуты	**Нулевое значение**	**Ключевое значение**
ID_производителя	Не допустимо	PK
Наименование_производителя	Не допустимо	