# Введение

# 1. Анализ предметной области

## 1.1 Краткая характеристика предприятия

Компания "Электрофан" основана в 1999 году и специализируется на продаже электроники и бытовой техники. За время своей деятельности компания зарекомендовала себя как надежный поставщик качественных товаров, предлагающий широкий ассортимент продукции от ведущих мировых производителей.

Основные цели деятельности:

**Увеличение объема продаж**: Достижение лидерских позиций на рынке электроники и бытовой техники, расширение клиентской базы, увеличение доли рынка.

**Максимизация прибыли**: Оптимизация бизнес-процессов, снижение издержек, повышение эффективности продаж, обеспечение стабильного роста прибыли.

**Высокое качество обслуживания клиентов**: Предоставление клиентам качественного сервиса, консультаций, гарантийного и пост-гарантийного обслуживания, формирование лояльности клиентов.

**Развитие и внедрение инноваций**: Постоянное совершенствование бизнес-процессов, внедрение новых технологий, расширение ассортимента товаров с учетом современных тенденций на рынке электроники.

**Формирование позитивного имиджа компании**: Создание узнаваемого бренда, ассоциирующегося с качеством, надежностью и клиентоориентированностью.

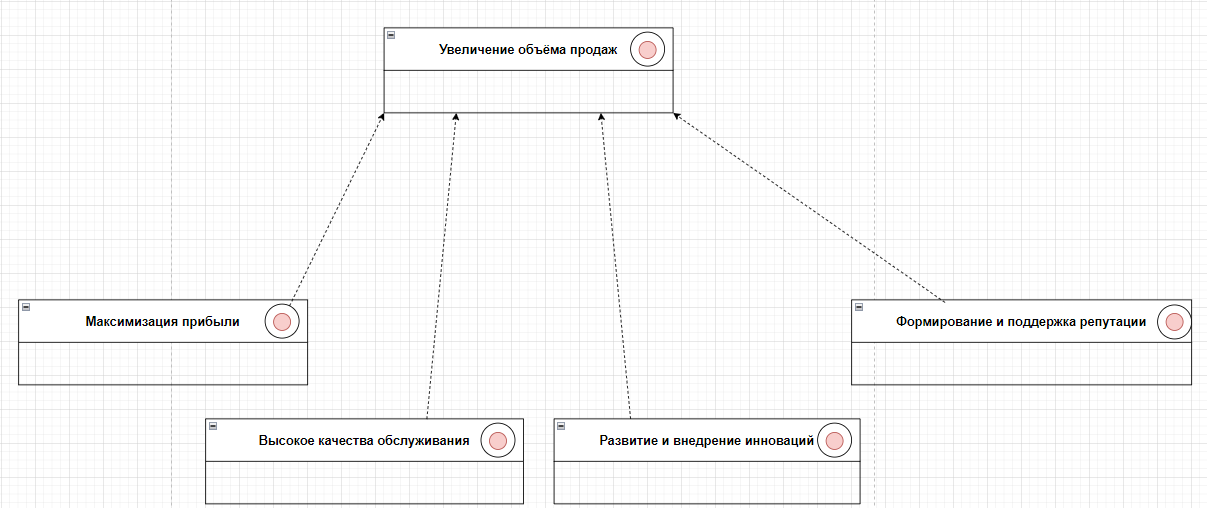


Рисунок 1.1.1 – Цели «Электрофан»

## 1.2 Организационная структура предприятия

Организационная структура ООО "Электрофан"

**Руководство:**

Генеральный директор: осуществляет общее руководство компанией, определяет стратегию развития, принимает ключевые решения.

**Отделы:**

Отдел продаж:

Менеджер по продажам: консультирует клиентов, обрабатывает заказы, оформляет продажи.

Отдел логистики:

Менеджер по складу: управляет складскими запасами, организует прием и отгрузку товаров.

Отдел технической поддержки:

Менеджер по техническим заявкам: принимает и обрабатывает заявки на ремонт, организует взаимодействие с клиентами и сервисными центрами.

Отдел управления товарами:

Администратор товаров: занимается добавлением, редактированием и удалением товаров в базе данных, контролирует актуальность информации о товарах.

Административно-хозяйственный отдел (АХО):

Руководитель АХО: отвечает за административно-хозяйственное обеспечение деятельности компании.

Бухгалтерия: ведет бухгалтерский учет, формирует отчетность.

Контрактная служба: занимается заключением и сопровождением договоров.

Отдел кадров: осуществляет подбор, учет и развитие персонала.

Другие подразделения: гараж, служба охраны, студенческий городок (если применимо).

**Подчиненность отделов:**

Все отделы подчиняются непосредственно генеральному директору.

Внутри отделов существует иерархия подчиненности: менеджеры подчиняются руководителю отдела. В случае отдела управлениями товара, подчиняются супер-вайзеру, в случае отдела логистики, подчиняются заведующему складу.

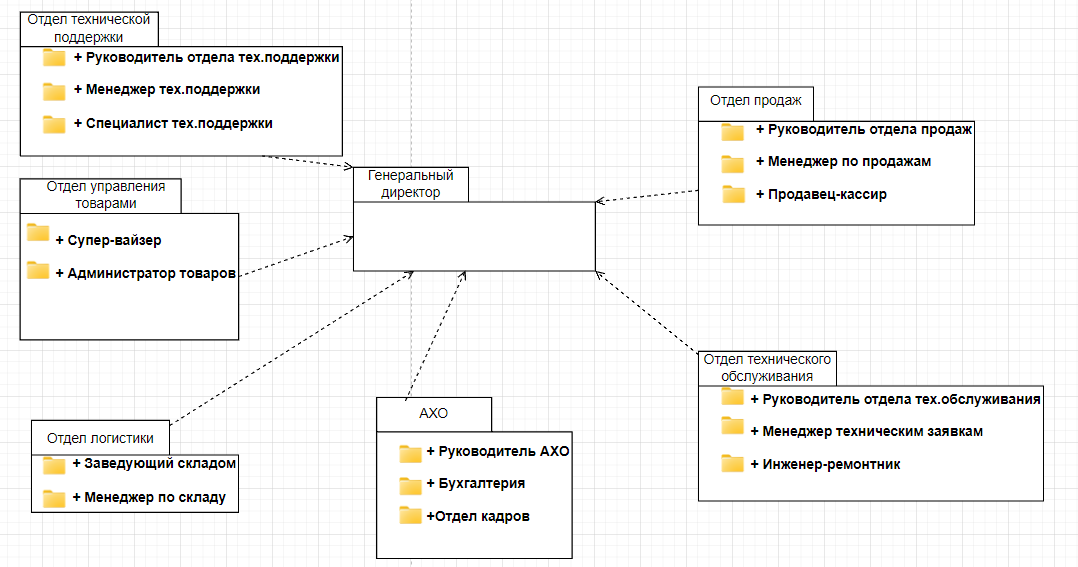


Рисунок 1.2.1 – Организационная структура Электрофан

## 1.3 Состав бизнес-процессов предприятия

Бизнес-процессы предприятия ООО "Электрофан":

1. **Техническая поддержка клиентов**

Обработка запросов: Прием и обработка запросов от клиентов по телефону, email, через сайт.

Консультирование: Предоставление информации о товарах, услугах, ценах, условиях покупки.

Оформление заказов: Прием и обработка заказов от клиентов, проверка наличия товара, оформление документов.

Поддержка клиентов: Решение проблем клиентов, обработка возвратов, предоставление гарантийного обслуживания.

2. **Управление продажами:**

Формирование ассортимента: Анализ рынка, выбор поставщиков, определение ценовой политики.

Продвижение товаров: Рекламные кампании, акции, специальные предложения.

Продажа товаров: Оформление заказов, прием оплаты, выдача товара.

Анализ продаж: Отслеживание динамики продаж, выявление популярных товаров, анализ эффективности маркетинговых мероприятий.

3. **Управление складом:**

Приемка товара: Проверка количества и качества товара, размещение на складе.

Хранение товара: Обеспечение сохранности товара, контроль условий хранения.

Отгрузка товара: Комплектация заказов, упаковка, отправка клиентам или в пункты выдачи.

Инвентаризация: Регулярный пересчет товара, выявление недостач и излишков.

4. **Управление техническими заявками:**

Прием заявок: Регистрация заявок на ремонт от клиентов, оформление документов.

Диагностика: Выявление причины неисправности, определение стоимости ремонта.

Ремонт: Проведение ремонтных работ, замена деталей, тестирование.

Выдача товара: Возврат отремонтированного товара клиенту.

5. **Управление финансами:**

Бухгалтерский учет: Ведение бухгалтерского учета, формирование отчетности.

Планирование бюджета: Составление и контроль исполнения бюджета.

Управление денежными потоками: Контроль за поступлением и расходованием денежных средств.

## 1.4 Описание базового бизнес-процесса

Обработка заказа, является базовым бизнес-процессом компании «Электрофан». Процесс **обработки заказа** в компании «Электрофан» включает в себя следующие этапы:

**Получение заказа**: Клиент оформляет заказ на сайте, по телефону или непосредственно в магазине.

**Проверка заказа**: Менеджер по продажам проверяет наличие товара на складе, уточняет детали заказа с клиентом (адрес доставки, способ оплаты и т.д.).

**Формирование заказа**: Менеджер по продажам создает заказ в системе учета, указывает информацию о клиенте, товарах, цене, способе доставки и оплаты.

**Оплата заказа**: Клиент оплачивает заказ выбранным способом (наличными, банковской картой, электронными деньгами).

**Комплектация заказа**: Менеджер по складу комплектует заказ, проверяет наличие и соответствие товаров, упаковывает заказ.

**Доставка заказа**: Заказ доставляется клиенту курьером, транспортной компанией или выдается в пункте самовывоза.

**Получение товара**: Клиент получает товар, проверяет его комплектность и соответствие заказу.

**Закрытие заказа**: Менеджер по продажам закрывает заказ в системе учета.

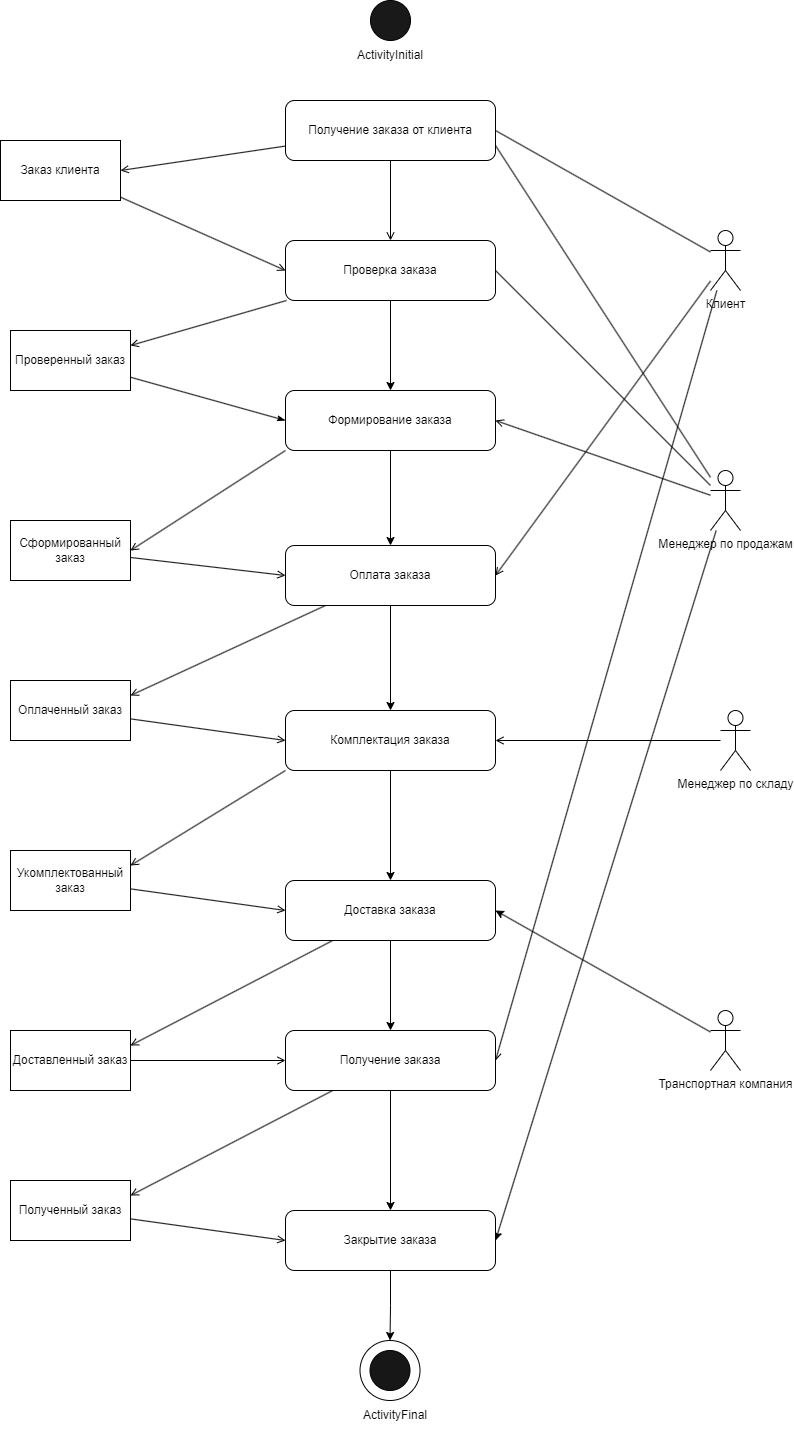


Рисунок 1.4.1 – Обработка заказа

## 1.5 Объекты бизнес-процессов

Описание объектов:

1. **Заказ клиента**:

Описание: Исходный заказ, полученный от клиента через сайт, телефон или в магазине. Может содержать неполную или некорректную информацию. Для этого требуется работа менеджера по продажам, который связывается с клиентом и обсуждает, обговаривает детали заказа. Менеджер по продажам проверяет наличие товаров на складе, возможность и сроки доставки транспортной компании.

**Атрибуты**:

Список товаров:

ID товара (int): Уникальный идентификатор товара.

Наименование (string): Название товара.

Количество (int): Количество заказываемого товара.

Контактная информация:

ФИО клиента (string): Фамилия, имя и отчество клиента.

Телефон (string): Номер телефона клиента.

Email (string): Электронный адрес клиента.

Адрес доставки (string): Адрес доставки заказа.

Способ оплаты (string): Наличный, безналичный, электронный.

Способ доставки (string): Курьер, транспортная компания, самовывоз.

Комментарии (string): Дополнительная информация от клиента.

2. **Проверенный заказ**:

Описание: Заказ, проверенный менеджером по продажам на корректность данных, наличие товаров на складе и соответствие условиям продажи.

Атрибуты:

Все атрибуты Заказа клиента.

Статус (string): "Подтвержден" или "Отклонен".

3. **Сформированный заказ**:

Описание: Заказ, оформленный в системе учета. Содержит все необходимые данные для комплектации и доставки.

Атрибуты:

Все атрибуты Проверенного заказа.

Номер заказа (int): Уникальный номер, присвоенный системой.

Дата создания (DateTime): Дата и время создания заказа.

Статус (string): "Новый", "В обработке", "Оплачен", "Отправлен", "Доставлен", "Отменен".

4. **Оплата заказа**:

Описание: Факт оплаты заказа клиентом.

Атрибуты:

Номер заказа (int): Ссылка на номер заказа.

Сумма оплаты (decimal): Сумма, оплаченная клиентом.

Дата оплаты (DateTime): Дата и время оплаты.

Способ оплаты (string): Наличный, безналичный, электронный.

5. **Укомплектованный заказ**:

Описание: Заказ, собранный на складе и готовый к отправке.

Атрибуты:

Все атрибуты Сформированного заказа.

Статус (string): "Укомплектован".

6. **Доставленный заказ**:

Описание: Заказ, доставленный клиенту.

Атрибуты:

Все атрибуты Сформированного заказа.

Статус (string): "Доставлен".

Дата доставки (DateTime): Дата и время доставки.

7. **Полученный заказ**:

Описание: Заказ, полученный клиентом и подтвержденный им.

Атрибуты:

Все атрибуты Сформированного заказа.

Статус (string): "Завершен".

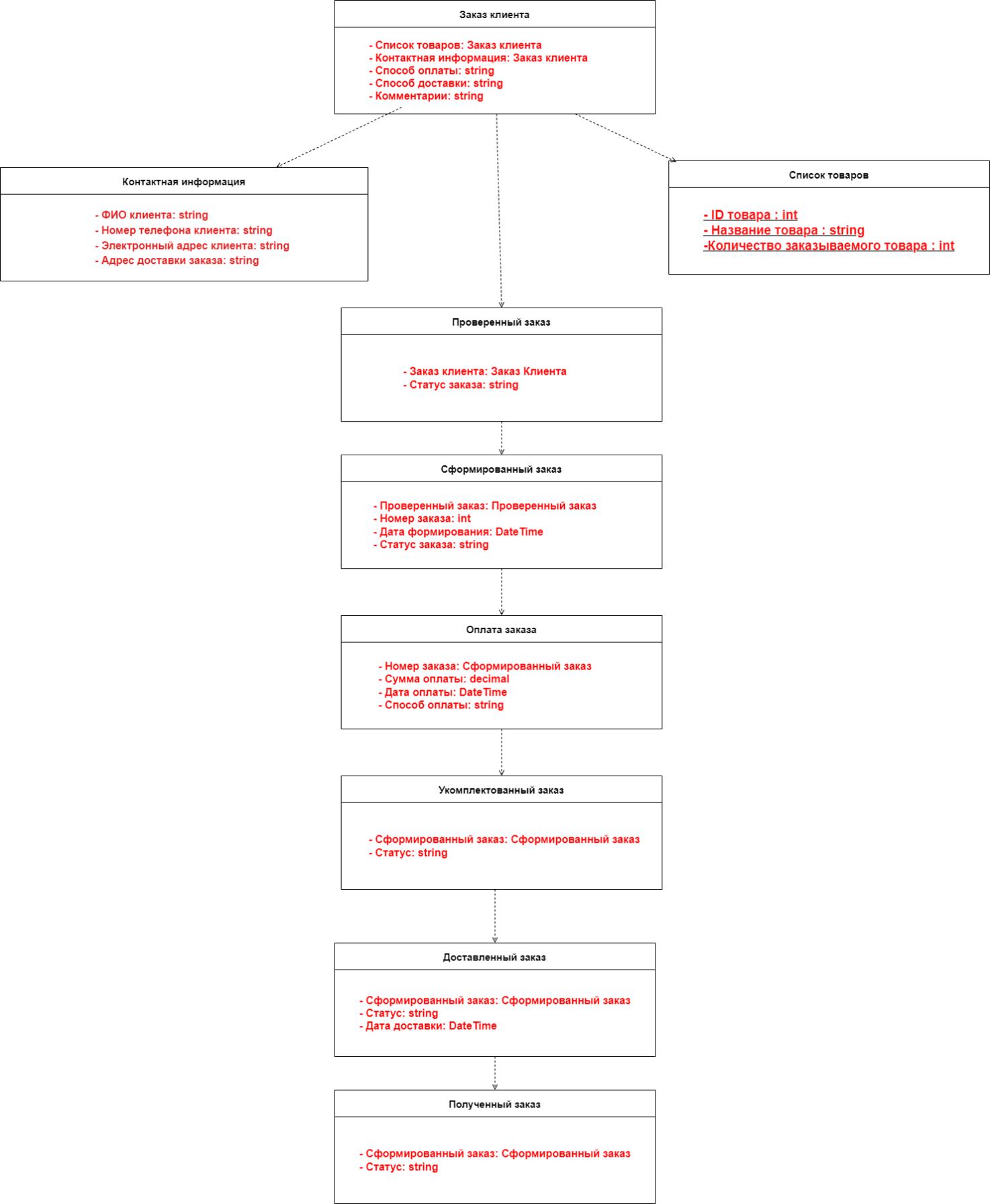


Рисунок 1.5.1 – Объекты бизнес процессов

## 1.6. Типовые бизнес-процессы, требующие автоматизации

1. **Управление заказами**:

Обработка заказов: Автоматизация процесса приема, обработки, оплаты и доставки заказов, включая интеграцию с платежными системами и службами доставки.

Отслеживание заказов: Предоставление клиентам и сотрудникам возможности отслеживать статус заказа в режиме

реального времени.

Формирование документов: Автоматическое создание счетов, накладных, актов выполненных работ и других документов,

связанных с заказами.

2. **Управление складом**:

Учет товаров: Автоматизация учета поступления, хранения и отгрузки товаров, ведение складского учета в режиме реального времени.

Контроль остатков: Автоматическое отслеживание остатков товаров на складе, оповещение о необходимости

пополнения запасов.

3. **Управление техническими заявками**:

Обработка заявок: Автоматизация приема, регистрации и распределения заявок на ремонт между сотрудниками

технической поддержки.

Ремонт: Ведение базы данных по ремонту.

Отслеживание статуса: Предоставление клиентам возможности отслеживать статус своих заявок на ремонт.

4. **Управление товарами**:

Редактирование информации: Автоматизация редактирования информации о товарах

Управление состоянием товаров: Использование информационной системы для изменения статуса товаров, например на «Исключен из продажи»

Контроль за информацией: Мониторинг полноценности и наглядности информации о товарах.



Рисунок 1.6.1 – Бизнес-процессы для автоматизации

# 2 ФОРМИРОВАНИЕ ТРЕБОВАНИЯ

## 2.1 Состав требования

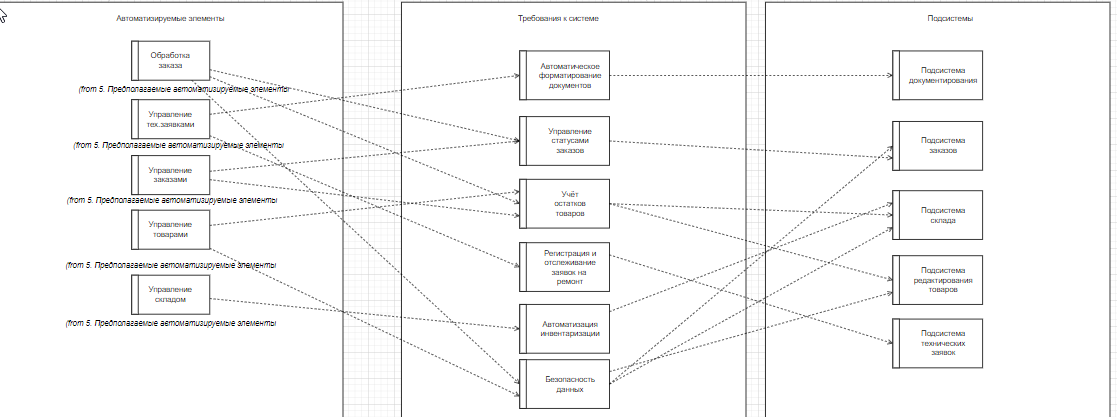


Рисунок 2.1.1 – Состав требований

**Описание требований**:

1. **Автоматическое формирование документов**:

Система должна автоматически формировать счета, накладные, акты выполненных работ и другие необходимые

документы на основе данных из заказов и заявок.

2. **Управление статусами заказов**:

Система должна позволять отслеживать статус заказа на всех этапах его обработки (новый, в обработке, оплачен,

отправлен, доставлен, отменен).

3. **Учет остатков товаров**:

Система должна вести учет остатков товаров на складе в режиме реального времени, автоматически списывая товары при оформлении заказов и пополняя запасы при поступлении новых товаров.

4. **Автоматизация инвентаризации**:

Система должна облегчать проведение инвентаризации, позволяя сопоставлять фактические остатки с данными в системе и выявлять расхождения.

5. **Регистрация и отслеживание заявок**:

Система должна позволять регистрировать заявки на ремонт, назначать ответственных сотрудников,

отслеживать статус заявок и вести историю ремонта.

6. **Безопасность данных**:

Система должна обеспечивать защиту данных от несанкционированного доступа, используя механизмы аутентификации, авторизации и шифрования.

2.2 Определение состава сценариев, реализующих требования

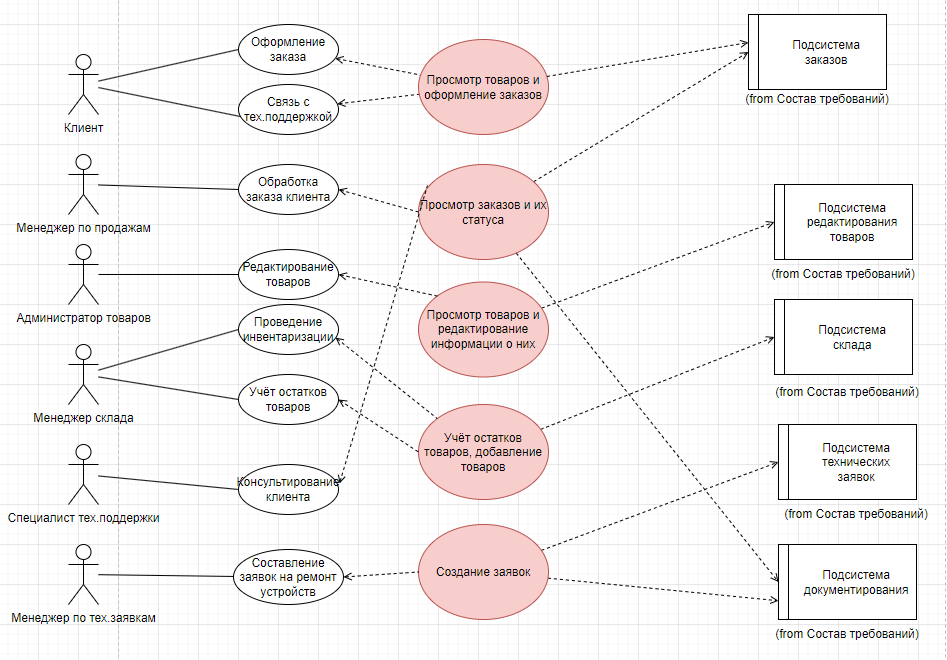


Рисунок 2.1.2 – Диаграмма вариантов использования система

**Подсистема заказов**:

«Оформление заказа»: Клиент и менеджер по продажам взаимодействуют для создания нового заказа, выбора товаров, указания данных для доставки и оплаты.

«Просмотр товаров и оформление заказов»: Клиент просматривает каталог товаров, выбирает нужные позиции и оформляет заказ.

«Просмотр заказов и их статуса»: Клиент и менеджер по продажам могут отслеживать текущий статус заказа (новый, в обработке, доставлен и т.д.).

«Обработка заказа клиента»: Менеджер по продажам проверяет заказ, подтверждает наличие товаров, выбирает способ доставки и оплаты, формирует необходимые документы.

«Связь с тех. Поддержкой»: Клиент и менеджер по продажам могут связаться с технической поддержкой в случае возникновения вопросов или проблем с товаром.

**Подсистема редактирования товаров**:

«Редактирование товаров»: Администратор товаров и менеджер по продажам могут добавлять новые товары в систему, редактировать существующую информацию о товарах (наименование, описание, цена, характеристики и т.д.).

«Просмотр товаров и редактирование информации о них»: Администратор товаров и менеджер по продажам могут просматривать информацию о товарах и вносить необходимые изменения.

**Подсистема склада**:

«Проведение инвентаризации»: Администратор товаров и менеджер склада проводят инвентаризацию клада, сверяя фактическое наличие товаров с данными в системе.

«Учет остатков товаров»: Менеджер склада отвечает за учет остатков товаров на складе, контролирует поступление и отгрузку товаров.

«Учет остатков товаров, добавление товаров»: Менеджер склада вносит информацию о поступлении новых товаров и списывает товары, которые были проданы или использованы в других операциях.

**Подсистема технических заявок**:

«Создание заявок»: Клиент, специалист технической поддержки и менеджер по техническим заявкам могут создавать новые заявки на ремонт устройств.

«Составление заявок на ремонт устройств»: Специалист технической поддержки и менеджер по техническим заявкам заполняют информацию о неисправности устройства и желаемом ремонте.

**Подсистема документирования**:

«Автоматическое формирование документов»: Система автоматически генерирует необходимые документы (счета, накладные, акты выполненных работ) на основе данных из заказов и заявок.

2.3. Разработка содержания сценариев

Диаграмма разработки содержания сценариев изображены на рисунках 2.3.1 – 2.3.3

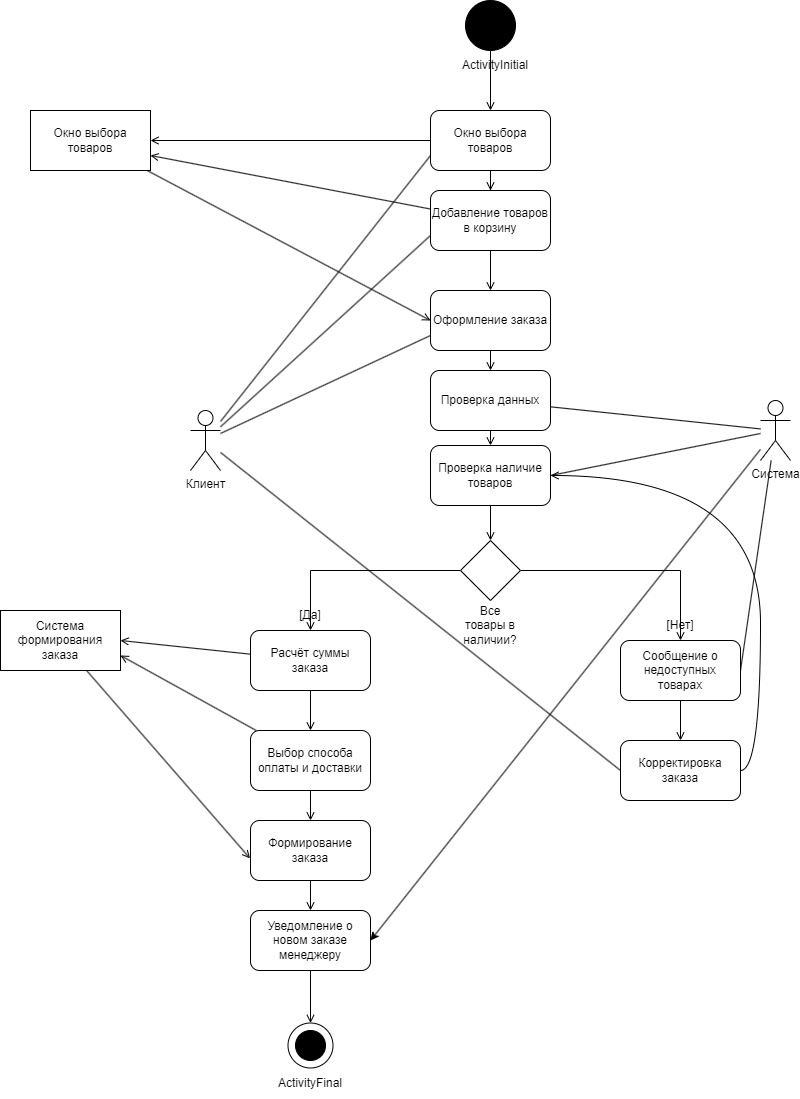


Рисунок 2.3.1 – Диаграмма деятельности «Оформление заказа»

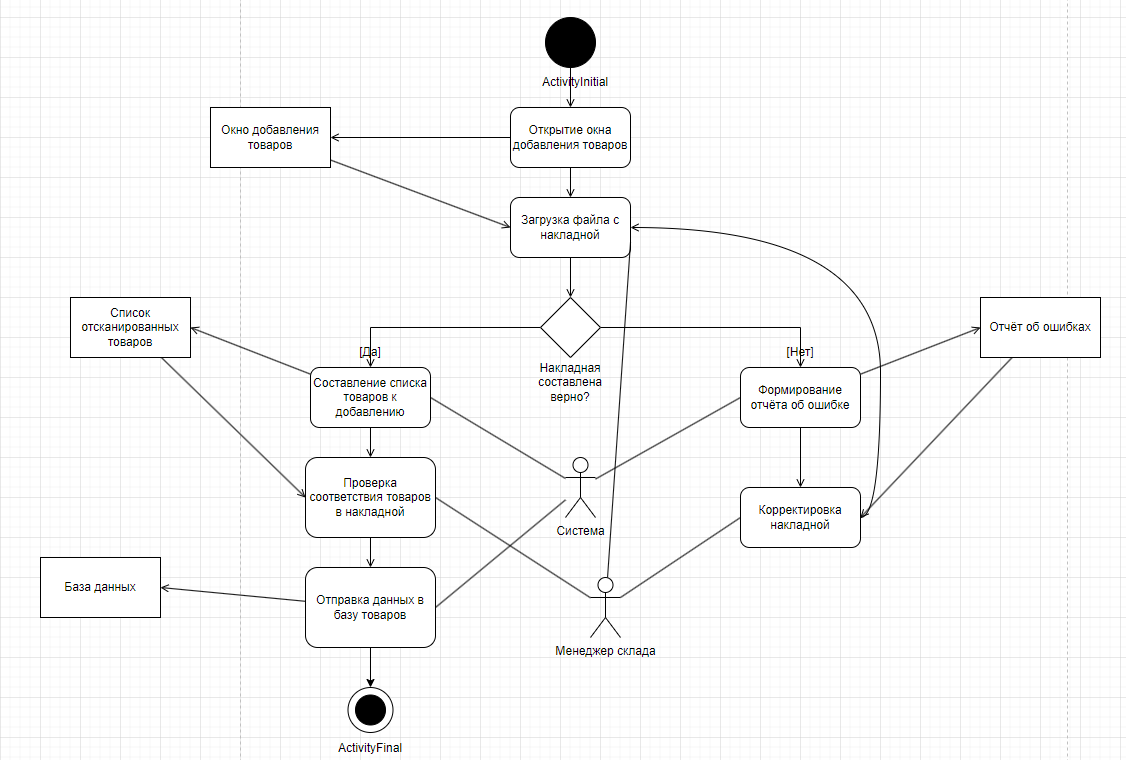


Рисунок 2.3.2 – Диаграмма деятельности «Добавление товаров на склад»

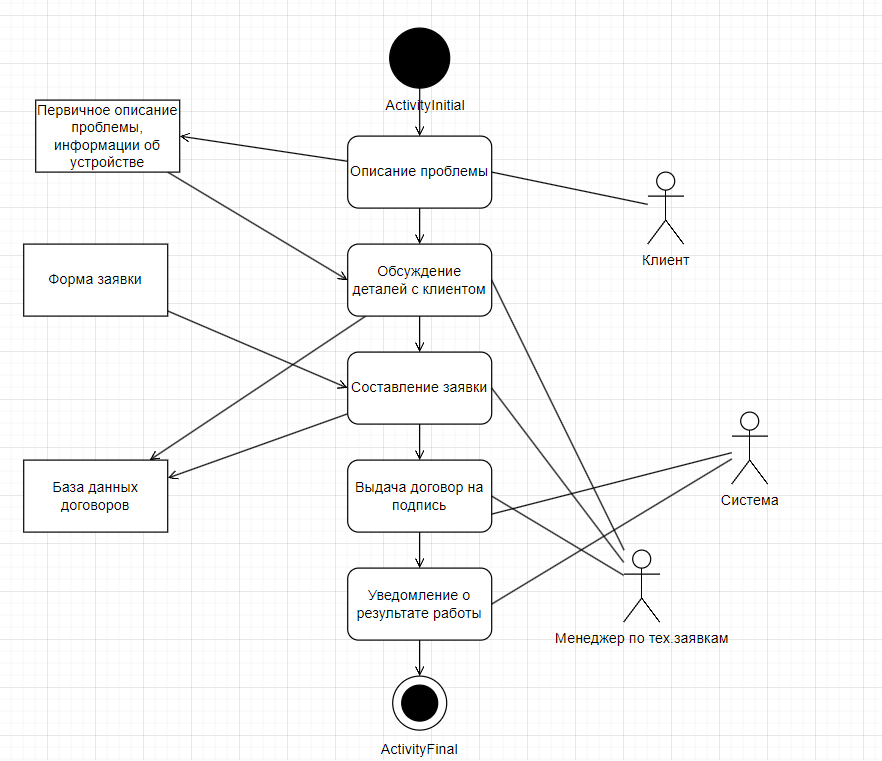


Рисунок 2.3.3 – Диаграмма деятельности «Оформление заявки на ремонт»

**Сценарий «Оформление заказа»**:

Описание: Диаграмма демонстрирует процесс оформления заказа клиентом в системе.

Клиент выбирает товары, добавляет их в корзину и переходит к оформлению заказа.

Система проверяет наличие товаров и, если все товары доступны, рассчитывает общую сумму заказа.

Клиент выбирает способы оплаты и доставки.

Система формирует заказ и отправляет подтверждение клиенту.

Менеджер по продажам получает уведомление о новом заказе.

Если каких-то товаров нет в наличии, система сообщает об этом клиенту, и он может скорректировать заказ.

Объекты:

«Окно выбора товаров»: Интерфейс для выбора товаров клиентом.

«Система формирования заказа»: Система с алгоритмами формирования заказа для клиента

**Сценарий** «**Добавление товаров на склад**»:

Описание:

Диаграмма демонстрирует процесс добавления новых товаров на склад, начиная с загрузки данных из накладной и заканчивая обновлением базы данных товаров.

Процесс:

Открытие окна добавления товаров: Менеджер склада открывает интерфейс для добавления информации о новых товарах.

Загрузка файла с накладной: Менеджер загружает файл с накладной, содержащий данные о поступивших товарах.

Проверка накладной: Система проверяет корректность данных в накладной.

Если накладная составлена верно:

Составление списка товаров к добавлению: Система формирует список товаров для добавления в базу данных, извлекая информацию из накладной.

Проверка соответствия товаров в накладной: Система проверяет, соответствуют ли данные о товарах в накладной информации в базе данных (например, наличие таких товаров, соответствие наименований).

Отправка данных в базу товаров: Если все проверки пройдены успешно, система добавляет информацию о новых товарах в базу данных.

Если накладная составлена неверно:

Формирование отчета об ошибке: Система формирует отчет об ошибках, обнаруженных в накладной.

Корректировка накладной: Менеджер склада корректирует накладную на основе отчета об ошибках и повторно загружает файл.

Объекты:

«Окно добавления товаров»: Интерфейс для ввода данных о новых товарах.

«Файл с накладной»: Электронный документ, содержащий информацию о поступивших товарах.

«Список отсканированных товаров»: Список товаров, отсканированных менеджером склада (не используется в данной схеме, т.к. предполагается загрузка файла).

«Отчет об ошибках»: Документ, содержащий информацию о несоответствиях или ошибках в накладной.

**Сценарий «Оформление заявки на ремонт»:**

Описание: Диаграмма иллюстрирует процесс обработки заявки на ремонт.

Клиент описывает проблему с устройством.

Менеджер по техническим заявкам обсуждает детали с клиентом и составляет заявку.

Система формирует договор на подпись.

После выполнения ремонта, клиент получает уведомление о результате.

Объекты:

«Первичное описание проблемы / Информация об устройстве":Информация, предоставленная клиентом о неисправности.

«Форма заявки»: Документ, содержащий детальное описание проблемы и контактные данные клиента.

«База данных договоров»: Хранилище договоров на ремонт.

# 3 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

## 3.1 Проектирование пользовательского интерфейса

1. **Окно авторизации**:

Описание: Окно предназначено для авторизации пользователей в системе.

Элементы:

Поле "Логин": Поле для ввода логина пользователя.

Поле "Пароль": Поле для ввода пароля пользователя.

Кнопка "Войти": Кнопка для отправки данных авторизации.

Кнопка "Регистрация": Кнопка для перехода к окну регистрации

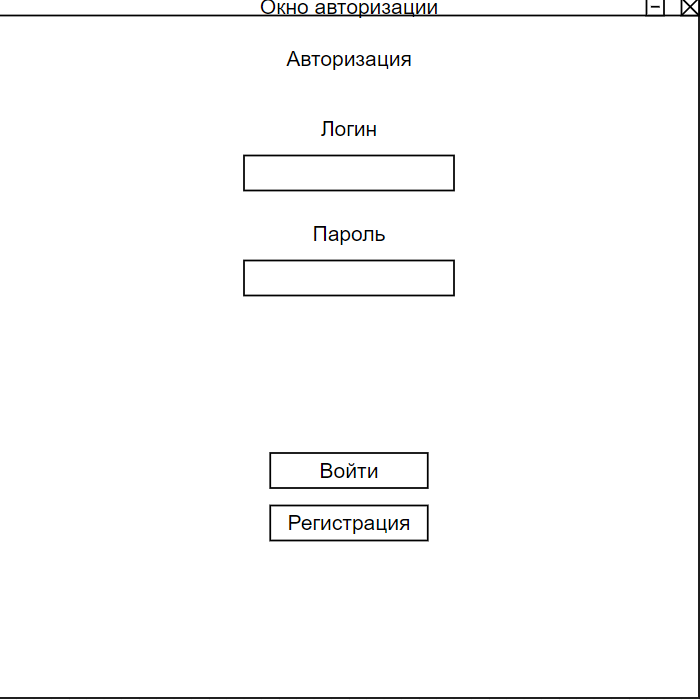


Рисунок 3.1.1 – Интерфейс окна авторизации

2. **Окно регистрации**:

Описание: Окно предназначено для регистрации новых пользователей в системе.

Элементы:

Кнопка "Назад": Кнопка для возврата к окну авторизации.

Поле "Логин": Поле для ввода логина пользователя.

Поле "Пароль": Поле для ввода пароля пользователя.

Поле "Имя": Поле для ввода имени пользователя.

Поле "Фамилия": Поле для ввода фамилии пользователя.

Поле "Отчество": Поле для ввода отчества пользователя.

Кнопка "Зарегистрироваться": Кнопка для отправки данных регистрации.

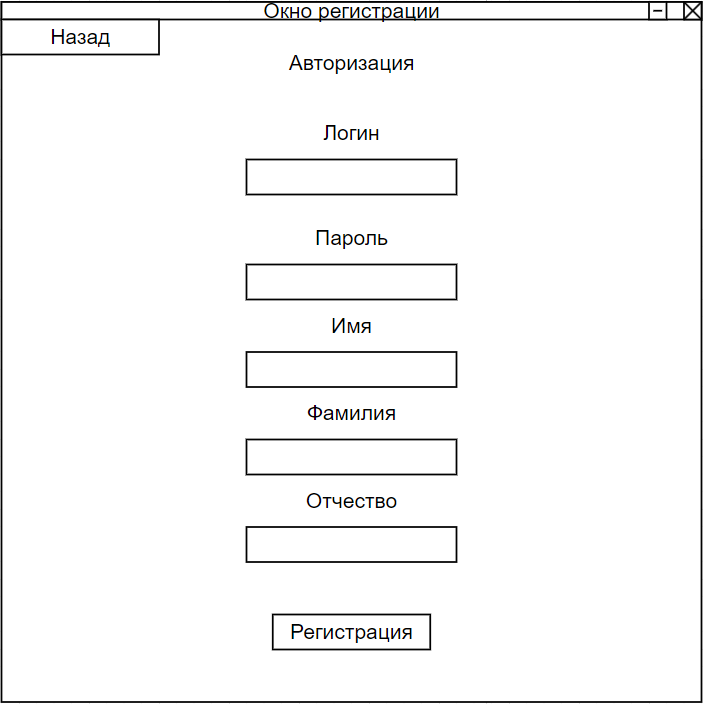


Рисунок 3.1.2 – Интерфейс окна регистрации

3. **Окно "Заявка на тех. обслуживание" ("Добавление заявки"):**

Описание: Окно, предназначенное для создания заявки на техническое обслуживание.

Элементы:

Поле "Фамилия": Поле для ввода фамилии клиента.

Поле "Имя": Поле для ввода имени клиента.

Поле "Отчество": Поле для ввода отчества клиента.

Поле "Описание проблемы": Поле для описания проблемы устройства.

Поле "Наименование устройства": Поле для указания наименования устройства.

Поле "Серийный номер": Поле для ввода серийного номера устройства.

Поле "Сотовый номер клиента": Поле для ввода номера телефона клиента.

Кнопка "Добавить": Кнопка для сохранения заявки в системе.

Кнопка "Составить догоров": Кнопка для генерации договора для подписания

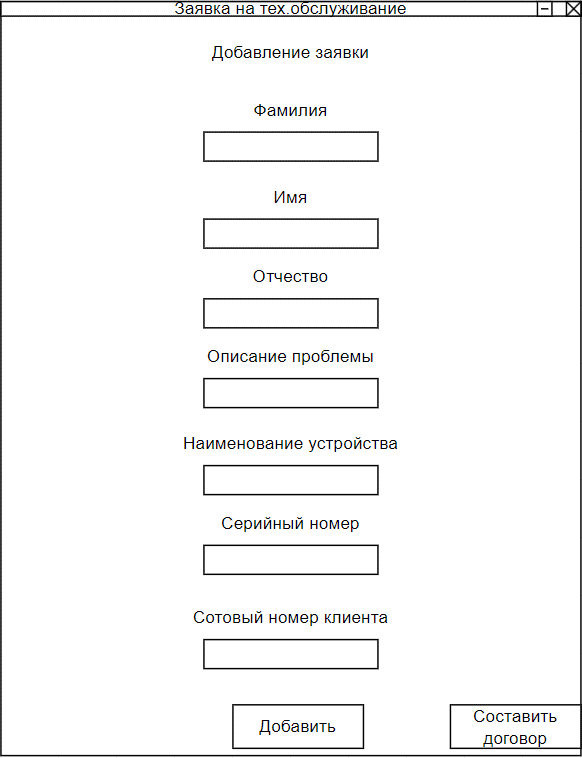


Рисунок 3.1.3 – Интерфейс окна добавление заявки

4. **Окно "Заявки на тех. ремонт":**

Описание: Окно предназначено для просмотра и управления заявками на технический ремонт.

Элементы:

Выпадающий список "Действие": Список доступных действий с заявками (например, "Просмотр",

"Редактирование", "Изменение статуса").

Поле "Фамилия": Поле для фильтрации заявок по фамилии клиента.

Поле "Имя": Поле для фильтрации заявок по имени клиента.

Поле "Отчество": Поле для фильтрации заявок по отчеству клиента.

Поле "Статус заявки": Поле для фильтрации заявок по статусу.

Поле поиска "Поиск клиента": Поле для поиска заявок по указанным критериям.

Таблица заявок: Таблица для отображения списка найденных заявок.

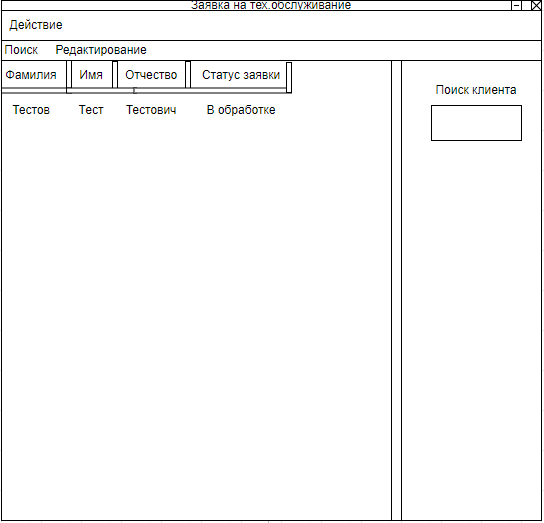


Рисунок 3.1.4 – Интерфейс окна просмотра заявок

5. **Окно менеджера**:

Описание: Окно предназначено для работы менеджера по продажам.

Элементы:

Выпадающий список "Действие": Список доступных действий (например, "Просмотр заказов",

"Создание заказа", "Отчеты").

Поле "Фамилия": Поле для фильтрации заказов/клиентов по фамилии.

Поле "Имя": Поле для фильтрации заказов/клиентов по имени.

Поле "Отчество": Поле для фильтрации заказов/клиентов по отчеству.

Кнопка "Поиск": Кнопка для поиска заказов/клиентов по указанным критериям.

Таблица заказов/клиентов: Таблица для отображения списка найденных заказов/клиентов.

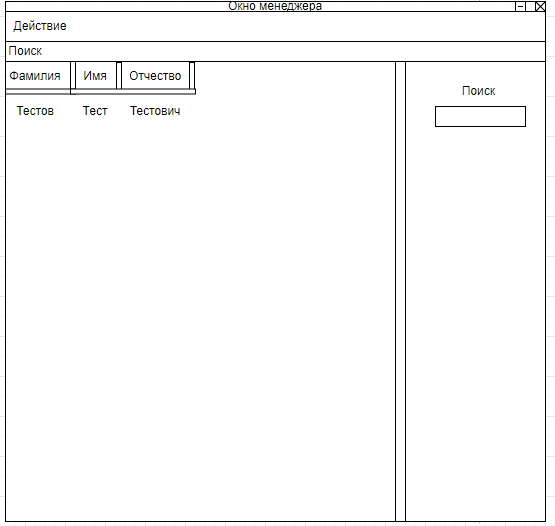


Рисунок 3.1.5 – Интерфейс окна менеджера со списком клиента

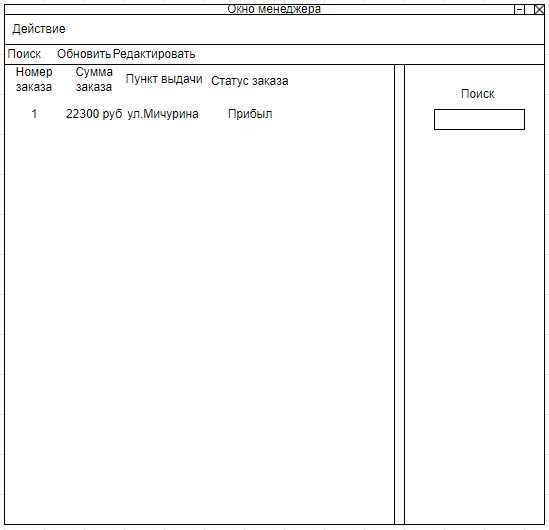


Рисунок 3.1.6 – Интерфейс окна менеджера с заявками

выбранного клиента

6. Меню администратора ("Меню администратора"):

Описание: Окно предназначено для работы администратора системы.

Элементы:

Кнопка "Действие": Выпадающий список доступных действий (например, "Добавить товар", "Редактировать товар",

"Удалить товар", "Добавить пользователя", "Редактировать пользователя", "Удалить пользователя", "Просмотр отчетов").

Кнопка "Назад": Кнопка для возврата к предыдущему окну или выходу из системы.

Таблица товаров/пользователей: Таблица для отображения списка товаров или пользователей.

Поля для редактирования товара/пользователя: Поля для ввода и редактирования данных о товаре

(название, вид товара, цена, количество, изображение) или пользователе (ФИО, логин, пароль, роль).

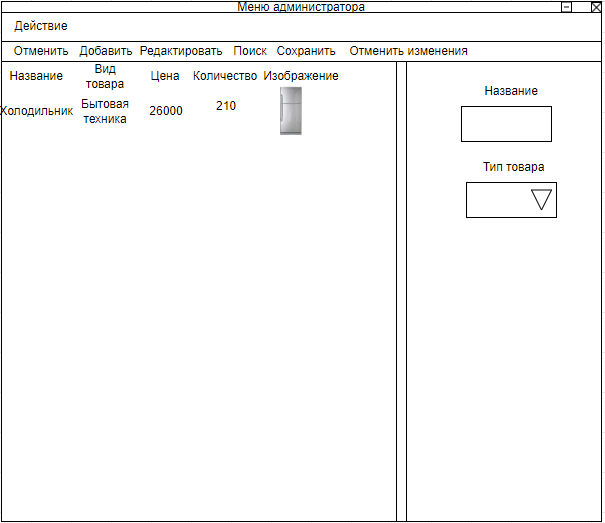


Рисунок 3.1.7 – Интерфейс окна администратора товаров

7. Окно "Добавление товаров":

Описание: Окно предназначено для добавления новых товаров в систему.

Элементы:

Текст "Выберите Excel-файл для сканирования": Инструкция для пользователя.

Кнопка "Выбрать файл": Кнопка для выбора Excel-файла с данными о товарах.

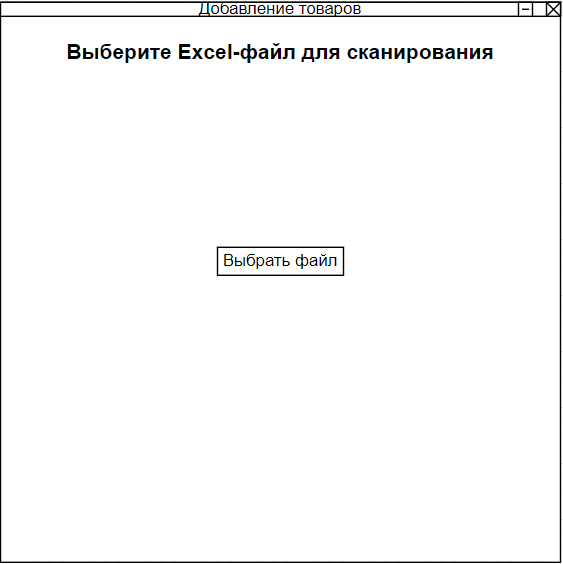


Рисунок 3.1.8 – Интерфейс окна менеджера склада

8. **Окно "Список отсканированных товаров":**

Описание: Окно отображает список товаров, считанных из Excel-файла при добавлении новой поставки. Администратор может проверить данные и подтвердить или отклонить добавление товаров в базу данных.

Элементы:

Заголовок "Список отсканированных товаров": Информирует пользователя о содержимом окна.

Выпадающий список "Действие": Позволяет выбрать действие с выбранным товаром (например, "Редактировать", "Удалить").

Таблица с заголовками:

"Название": Наименование товара.

"Вид товара": Вид товара (категория).

"Цена": Цена товара.

"Количество": Количество товара в поставке.

Кнопка "Подтвердить": Подтверждает добавление всех товаров из списка в базу данных.

Кнопка "Отклонить": Отклоняет добавление товаров из списка, очищает таблицу.

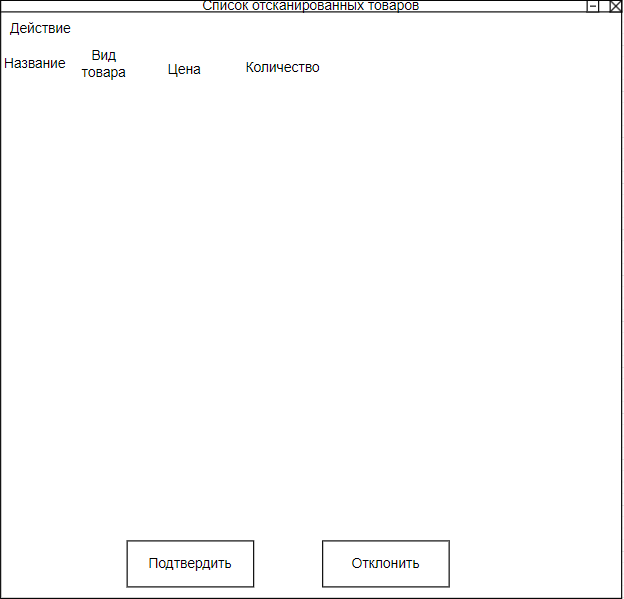


Рисунок 3.1.9 – Интерфейс окна отсканированных товаров

Описание функционала:

Авторизация: Пользователь вводит свои учетные данные (логин и пароль) для доступа к системе.

Регистрация: Новые пользователи могут зарегистрироваться в системе, указав необходимую информацию.

Управление заявками на техническое обслуживание: Сотрудники могут создавать, просматривать, редактировать и изменять статус заявок.

Управление заказами: Менеджеры могут просматривать, создавать и управлять заказами.

Управление товарами: Администратор может добавлять, редактировать и удалять товары в базе данных.

Добавление товаров из Excel: Администратор может загружать данные о товарах из Excel-файла.

Проверка данных: Администратор может проверить данные о товарах, считанные из Excel-файла, и внести необходимые изменения перед добавлением в базу данных.

Редактирование данных: Администратор может редактировать данные о отдельном товаре в списке.

Удаление данных: Администратор может удалить товар из списка.

Подтверждение/отклонение добавления: Администратор может подтвердить добавление всех товаров из списка в базу данных или отклонить добавление и очистить таблицу.

## 3.2 Классы граничных объектов

Главное окно - основное окно приложения, является точкой входа для пользователей.

Окно авторизации - окно для ввода логина и пароля пользователя.

Окно регистрации - окно для регистрации новых пользователей.

Окно менеджера по продажам - окно для работы менеджера по продажам.

Окно менеджера по складу - окно для работы менеджера по складу.

Окно заявки на тех. обслуживание - окно для создания заявки на техническое обслуживание.

Окно Заявок на тех. ремонт : Окно для просмотра и управления заявками на технический ремонт.

Меню администратора - окно для работы администратора системы.

Окно добавления товаров - окно для добавления новых товаров в систему из Excel-файла.

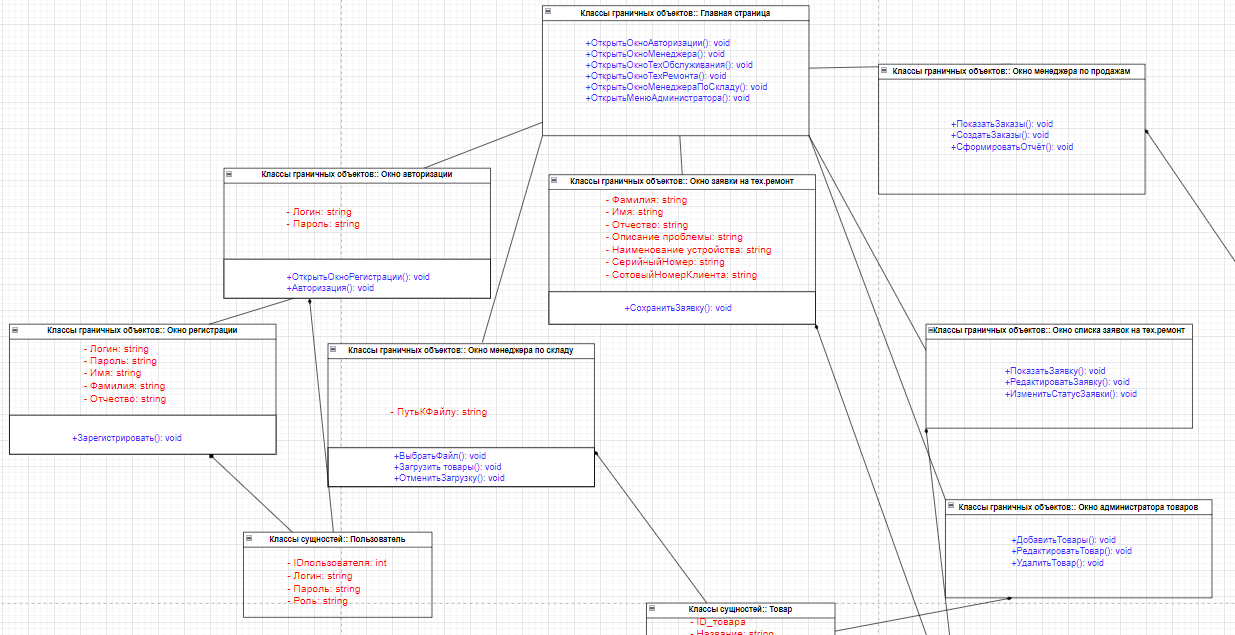


Рисунок 3.2.1 – Диаграмма классов граничных объектов

## 3.3 Класс сущностей

Пользователь - представляет пользователя системы "Электрофан".

Товар - представляет товар, продающийся в магазине "Электрофан".

ЗаявкаНаТехОбслуживание - представляет заявку клиента на техническое обслуживание или ремонт устройства.

Заказ - представляет заказ клиента на покупку товаров.

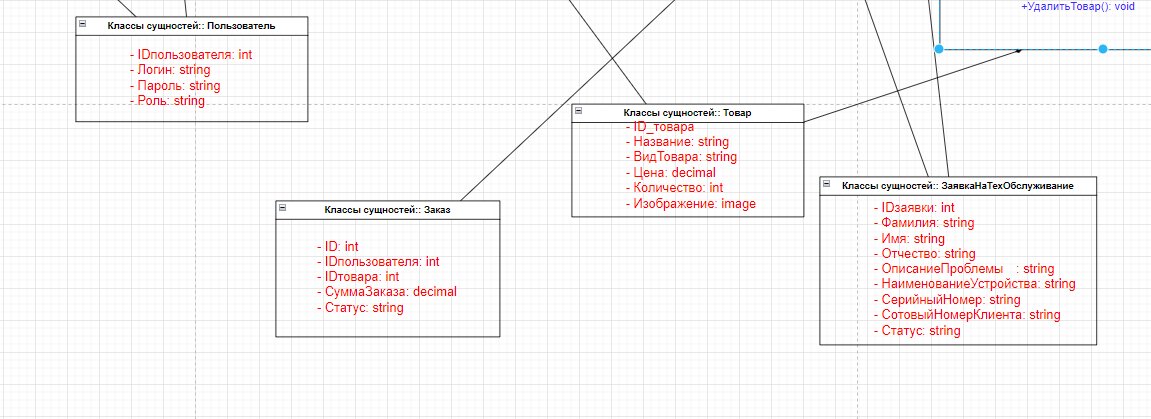


Рисунок 3.3.1 – Диаграмма сущностей

## 3.4 Определение методов объектов

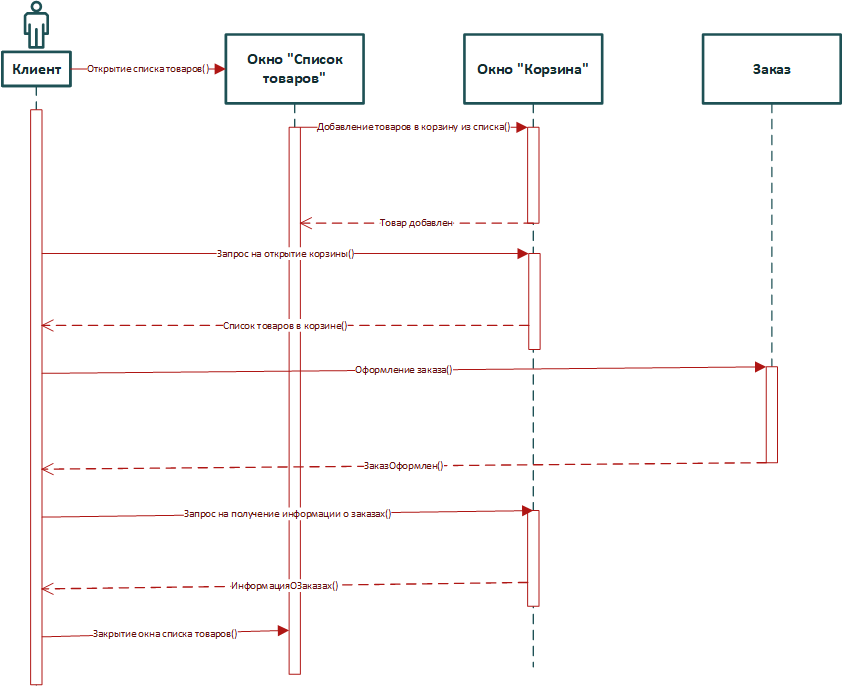


Рисунок 3.4.1 – Диаграмма последовательности «Оформление заказа»

Клиент открывает список товаров. Клиент отправляет сообщение "Открыть список товаров" окну "Список товаров".

Добавление товара в корзину. Клиент выбирает товар и указывает количество, затем отправляет сообщение "Добавление товаров в корзину" окну "Список товаров". Окно "Список товаров" передает информацию о выбранном товаре и его количестве окну "Корзина".

Окно "Корзина" подтверждает добавление товара. Окно "Корзина" возвращает окну "Список товаров" сообщение "Товар добавлен".

Открытие корзины. Окно "Корзина" отправляет окну "Список товаров" сообщение "Запрос на открытие корзины".

Отображение товаров в корзине. Окно "Список товаров" возвращает клиенту список товаров, находящихся в корзине.

Оформление заказа. Клиент нажимает кнопку "Оформить заказ" в окне "Корзина".

Создание заказа. Окно "Корзина" отправляет сообщение "СоздатьЗаказ()" сущности "Заказ", передавая информацию о клиенте и товарах в корзине.

Подтверждение создания заказа. Сущность "Заказ" создает новый заказ в базе данных и возвращает окну "Корзина" сообщение "ЗаказОформлен()", содержащее ID созданного заказа.

Запрос информации о заказе. Окно "Корзина" отправляет сущности "Заказ" сообщение "Запрос на получение информации о заказе", указывая ID заказа.

Получение информации о заказе. Сущность "Заказ" возвращает окну "Корзина" сообщение "ИнформОЗаказе()", содержащее информацию о заказе (номер заказа, дата, сумма, статус и т.д.).

Отображение информации о заказе. Окно "Корзина" отображает клиенту информацию о созданном заказе.

Закрытие окна списка товаров. Клиент закрывает окно "Список товаров".

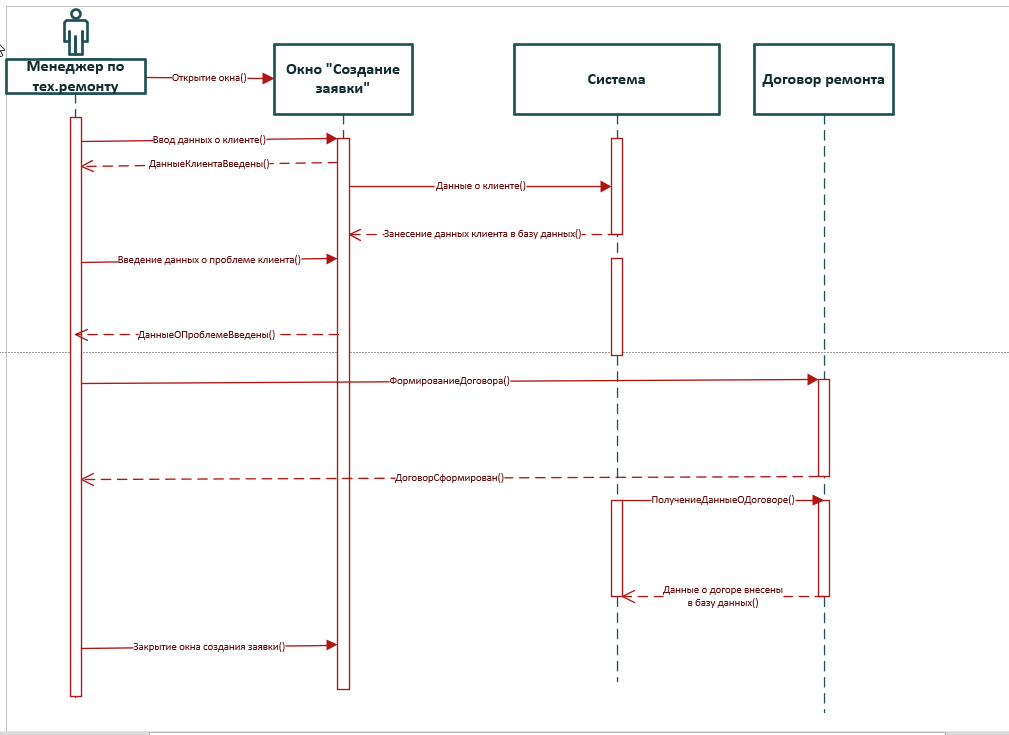


Рисунок 3.4.2 – Диаграмма последовательности «Оформление тех.заявки»

Открытие окна создания договора. Менеджер по тех. обслуживанию открывает окно "Создание договора ремонта".

Выбор заявки. Окно предлагает менеджеру выбрать заявку на ремонт, на основе которой будет создан договор.

Ввод данных о клиенте и заказе. Менеджер вводит данные о клиенте и заказе, связанном с заявкой.

Ввод данных об устройстве. Менеджер вводит данные об устройстве, его серийном номере и описание проблемы.

Запись данных заявки в базу данных. Менеджер сохраняет введенные данные заявки в базу данных. Система возвращает подтверждение о сохранении данных и присваивает заявке уникальный ID.

Формирование договора. Окно "Создание договора ремонта" инициирует формирование договора, передавая данные о заявке объекту "Договор ремонта".

Подтверждение формирования договора. Объект "Договор ремонта" возвращает окну сообщение "ДоговорСформирован()", содержащее ID сформированного договора.

Получение/запись ID договора. Окно "Создание договора ремонта" обращается к системе для получения и сохранения ID сформированного договора.

Предоставление доступа к договору. Объект "Договор ремонта" предоставляет менеджеру доступ к сформированному договору.

Закрытие окна создания договора. Менеджер закрывает окно "Создание договора ремонта".

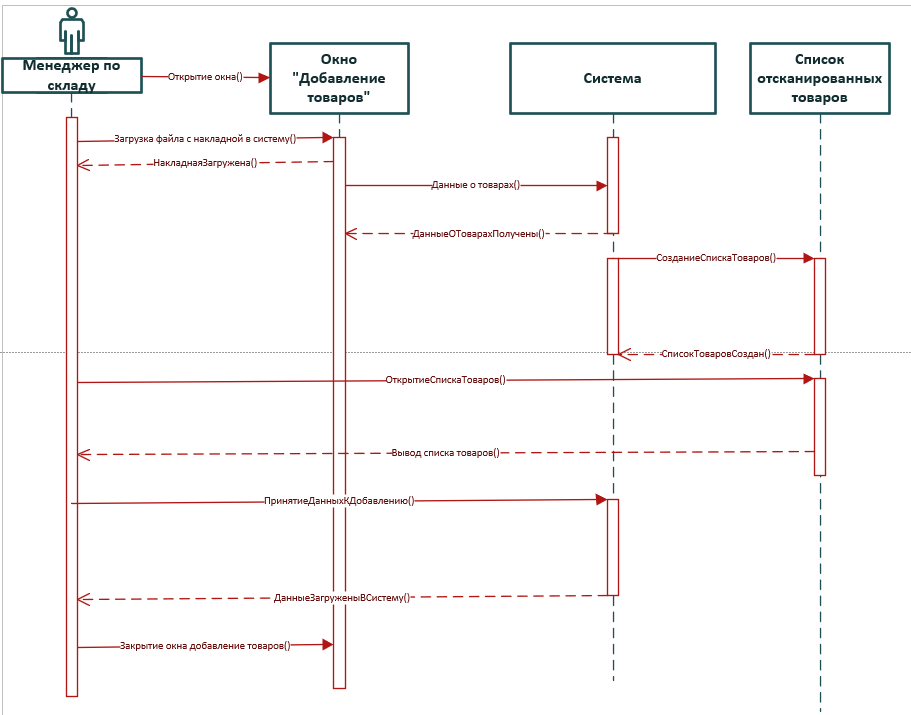


Рисунок 3.4.3 – Диаграмма последовательности «Добавление товаров в систему»

Открытие окна добавления товара. Менеджер по управлению товарами открывает окно "Добавление товара".

Загрузка файла с товарами. Окно "Добавление товара" предоставляет менеджеру возможность загрузить Excel-файл с информацией о товарах.

Запись нового товара. Менеджер отправляет запрос на запись данных о товаре в систему.

Создание описания товаров. Система, получив данные о товарах, инициирует создание списка отсканированных товаров.

Подтверждение создания списка. Список отсканированных товаров сообщает системе о своей готовности.

Передача данных о товаре. Система передает окну "Добавление товара" данные о новом товаре, включая его ID и другие атрибуты.

Добавление товара в список. Окно "Добавление товара" отправляет запрос на добавление товара в список отсканированных товаров.

Вывод списка товаров. Список отсканированных товаров отображается в окне "Добавление товара".

Закрытие окна добавления товара. Менеджер закрывает окно "Добавление товара".

## 3.5 Диаграмма классов

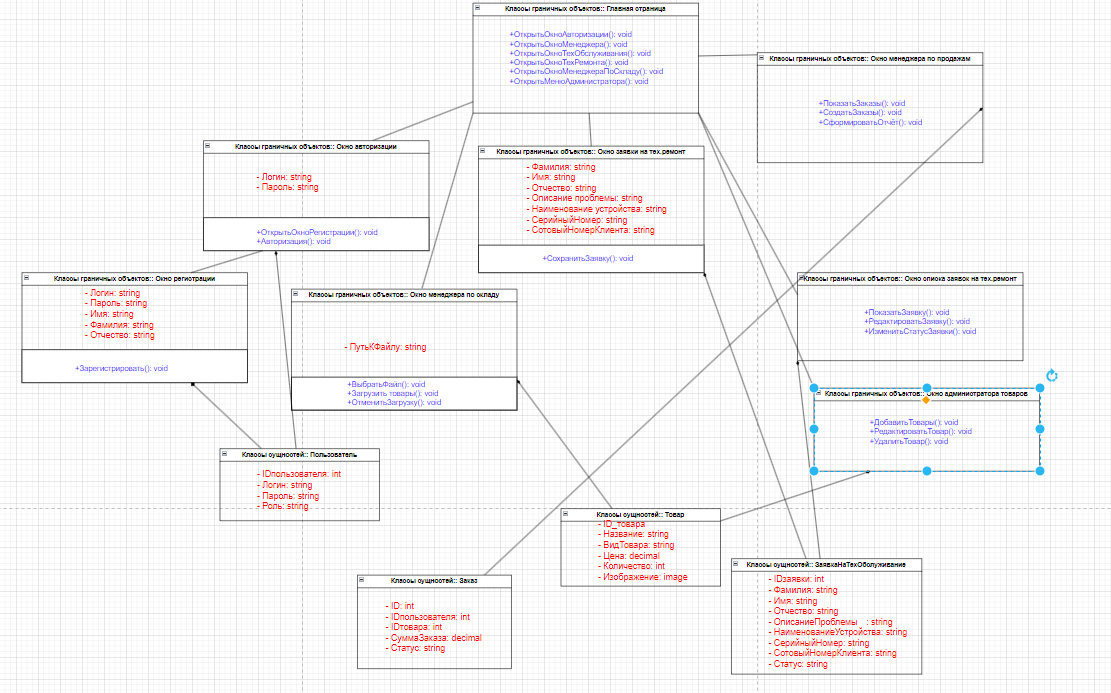


Рисунок 3.5.1 – Диаграмма классов

Главная страница: Основное окно приложения, содержит ссылки на другие окна.

Окно авторизации: Позволяет пользователю ввести логин и пароль для авторизации в системе.

Окно регистрации: Позволяет новому пользователю зарегистрироваться в системе, указав логин, пароль и другие данные.

Окно менеджера: Предоставляет менеджеру по продажам доступ к функциям управления заказами.

Окно менеджера по складу: Предоставляет менеджеру по складу доступ к функциям управления складом.

Окно заявки на тех. ремонт: Позволяет создавать новые заявки на техническое обслуживание или ремонт.

Окно списка заявок на тех. ремонт: Отображает список заявок на ремонт и позволяет управлять ими.

Окно администратора товаров: Предоставляет администратору товаров доступ к функциям управления товарами (добавление, редактирование, удаление).

Пользователь: Хранит информацию о пользователе (логин, пароль, роль).

Товар: Хранит информацию о товаре (название, вид, цена, количество, изображение).

Заказ: Хранит информацию о заказе (номер заказа, дата, клиент, товары, сумма, статус).

ЗаявкаНаТехОбслуживание: Хранит информацию о заявке на тех. обслуживание (номер заявки, дата, клиент, устройство, описание проблемы, статус).

# 4 ВЫБОР ТЕХНОЛОГИЙ РЕАЛИЗАЦИИ

Среда разработки: Visual Studio

Обоснование:

Популярность и поддержка: Visual Studio – широко используемая IDE с обширным сообществом и хорошей документацией.

Интеграция с .NET Framework и WPF: Visual Studio предоставляет мощные инструменты для разработки приложений на WPF, что упрощает создание пользовательского интерфейса "Электрофан".

Отладка и тестирование: Visual Studio включает в себя отладчик и инструменты для тестирования, что помогает обнаруживать и исправлять ошибки в коде.

2. База данных и СУБД: Microsoft SQL Server и SQL Server Management Studio

Обоснование:

Надежность и производительность: SQL Server – надежная и производительная СУБД, способная обрабатывать большие объемы данных, что важно для "Электрофан" с ее растущим ассортиментом товаров и базой клиентов.

Интеграция с .NET Framework: SQL Server хорошо интегрируется с .NET Framework, упрощая разработку приложения и доступ к данным.

Удобный интерфейс администрирования: SQL Server Management Studio предоставляет интуитивно понятный интерфейс для администрирования базы данных, создания таблиц, запросов и хранимых процедур.

3. Фреймворк: Windows Presentation Foundation (WPF)

Обоснование:

Создание насыщенного пользовательского интерфейса: WPF позволяет создавать привлекательные и функциональные пользовательские интерфейсы с использованием XAML и поддержкой анимации, графики и мультимедиа.

Разделение логики и представления: WPF поддерживает паттерн MVVM (Model-View-ViewModel), который позволяет отделить логику приложения от его представления, что упрощает разработку и тестирование.

4. Модель разработки: Каскадная модель (Waterfall Model)

Обоснование:

Четкое определение требований: Исходя из предыдущих запросов и анализа предметной области, можно предположить, что требования к системе "Электрофан" достаточно четко определены и не будут значительно меняться в процессе разработки.

Каскадная модель хорошо подходит для проектов с фиксированными требованиями.

Простая структура и понятность: Каскадная модель проста в понимании и

реализации, что может быть важно для команды разработчиков "Электрофан", если у них небольшой опыт в разработке программного обеспечения.

Детальная документация: Каскадная модель предусматривает создание детальной документации на каждом этапе разработки, что помогает контролировать процесс и обеспечивать качество системы.

5. Операционная система: Windows 10

Обоснование:

Широкая распространенность: Windows 10 – одна из самых популярных операционных систем, что гарантирует совместимость с большинством компьютеров "Электрофан".

Поддержка .NET Framework и WPF: Windows 10 полностью поддерживает

.NET Framework и WPF, что необходимо для работы приложения "Электрофан".

# 5 ПРОЕКТИРОВАНИЕ БАЗЫ ДАННЫХ

## 5.1 Сущности базы данных

Для проектирования базы данных необходимо составить перечень сущностей и атрибутов к ним, с подробным описанием атрибутов и их типов данных, с примерами использования данных.

Таблица 1: Вид\_товара

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название атрибута | Описание | Домен | Тип данных | Диапазон значений | Пример атрибута |
| ID | Уникальный идентификатор вида товара | Целые числа | int | 1, 2, 3, ... | 10 |
| Название | Наименование вида товара | Строка | nvarchar(50) |  | "Смартфоны" |

Таблица 2: Товар

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название атрибута | Описание | Домен | Тип данных | Диапазон значений | Пример атрибута |
| ID | Уникальный идентификатор товара | Целые числа | int | 1, 2, 3, ... | 25 |
| Наименование | Название товара | Строка | nvarchar(50) |  | "Смартфон Samsung Galaxy S23" |
| Вид\_товара | ID вида товара, к которому относится товар | Внешний ключ | int | Ссылка на ID из таблицы Вид\_товара | 10 |
| Количество\_на\_складе | Количество товара на складе | Целые числа | int | 0, 1, 2, ... | 15 |
| Цена | Цена товара | Целые числа | int |  | 79990 |
| Изображение | Изображение товара | Binary data | image |  | (бинарные данные изображения) |

Таблица 3: Роль\_пользователя

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название атрибута | Описание | Домен | Тип данных | Диапазон значений | Пример атрибута |
| ID | Уникальный идентификатор роли пользователя | Целые числа | int | 1, 2, 3, ... | 2 |
| Название | Наименование роли пользователя | Строка | nvarchar(50) |  | "Менеджер" |

Таблица 4: Пользователь

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название атрибута | Описание | Домен | Тип данных | Диапазон значений | Пример атрибута |
| ID | Уникальный идентификатор пользователя | Целые числа | int | 1, 2, 3, ... | 5 |
| Фамилия | Фамилия пользователя | Строка | nvarchar(50) |  | "Иванов" |
| Имя | Имя пользователя | Строка | nvarchar(50) |  | "Иван" |
| Отчество | Отчество пользователя | Строка | nvarchar(50) |  | "Иванович" |
| Роль | ID роли пользователя | Внешний ключ | int | Ссылка на ID из таблицы Роль\_пользователя | 2 |
| Сотовый\_номер | Номер телефона пользователя | Строка | varchar(11) |  | "+79123456789" |

Таблица 5: Данные\_пользователя

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название атрибута | Описание | Домен | Тип данных | Диапазон значений | Пример атрибута |
| ID | Уникальный идентификатор данных пользователя | Целые числа | int | 1, 2, 3, ... | 10 |
| Логин | Логин пользователя | Строка | nvarchar(30) |  | "ivanov.i" |
| Пароль | Пароль пользователя (в зашифрованном виде) | Строка | varchar(MAX) |  | "хэш пароля" |
| ID\_пользователя | ID пользователя, к которому относятся данные | Внешний ключ | int | Ссылка на ID из таблицы Пользователь | 5 |
| Последняя\_дата\_входа | Дата и время последнего входа пользователя | Дата и время | datetime |  | "2024-05-11 12:00:00" |

Таблица 6: Статус\_заказа

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название атрибута | Описание | Домен | Тип данных | Диапазон значений | Пример атрибута |
| ID | Уникальный идентификатор статуса заказа | Целые числа | int | 1, 2, 3, ... | 3 |
| Описание\_статуса | Описание статуса заказа | Строка | nvarchar(50) |  | "Отправлен" |

Таблица 7: Пункт\_Выдачи

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название атрибута | Описание | Домен | Тип данных | Диапазон значений | Пример атрибута |
| ID | Уникальный идентификатор пункта выдачи | Целые числа | int | 1, 2, 3, ... | 8 |
| Адрес\_пункта\_выдачи | Адрес пункта выдачи | Строка | nvarchar(100) |  | "ул. Ленина, 1, г. Москва" |

Таблица 8: Заказ

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название атрибута | Описание | Домен | Тип данных | Диапазон значений | Пример атрибута |
| ID | Уникальный идентификатор заказа | Целые числа | int | 1, 2, 3, ... | 15 |
| ID\_пользователя | ID пользователя, создавшего заказ | Внешний ключ | int | Ссылка на ID из таблицы Пользователь | 5 |
| ID\_пункта\_выдачи | ID пункта выдачи, где будет получен заказ | Внешний ключ | int | Ссылка на ID из таблицы Пункт\_выдачи | 8 |
| Статус\_заказа | ID статуса заказа | Внешний ключ | int | Ссылка на ID из таблицы Статус\_заказа | 3 |
| Сумма\_заказа | Сумма заказа | Целые числа | int |  | 109980 |

Таблица 9: Заказ\_Товар

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название атрибута | Описание | Домен | Тип данных | Диапазон значений | Пример атрибута |
| ID | Уникальный идентификатор записи связи заказа и товара | Целые числа | int | 1, 2, 3, ... | 20 |
| ID\_заказа | ID заказа, к которому относится товар | Внешний ключ | int | Ссылка на ID из таблицы Заказ | 15 |
| ID\_товара | ID товара, входящего в заказ | Внешний ключ | int | Ссылка на ID из таблицы Товар | 25 |
| Количество\_товара | Количество товара в заказе | Целые числа | int | 1, 2, 3, ... | 2 |

Таблица 10: Отсканированные\_документы

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название атрибута | Описание | Домен | Тип данных | Диапазон значений | Пример атрибута |
| ID\_документы | Уникальный идентификатор документа | Целые числа | int | 1, 2, 3, ... | 12345 |
| Дата\_проведения | Дата и время сканирования документа | Дата и время | datetime |  | "2024-05-11 14:30:00" |
| ID\_работника | ID работника, отсканировавшего документ | Внешний ключ | int | Ссылка на ID из таблицы Пользователь | 2 |
| Дата\_документа | Дата документа | Дата | date |  | "2024-05-10" |
| Проведён | Флаг, указывающий, обработан ли документ | Логический | bit | 0 (false) / 1 (true) | 1 |

Таблица 11: Возврат\_товара

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название атрибута | Описание | Домен | Тип данных | Диапазон значений | Пример атрибута |
| ID\_возврата | Уникальный идентификатор возврата товара | Целые числа | int | 1, 2, 3, ... | 5 |
| ID\_товара\_возврата | ID товара, который был возвращен | Внешний ключ | int | Ссылка на ID из таблицы Товар | 25 |
| Сумма\_возврата | Сумма возврата | Целые числа | int |  | 79990 |
| ID\_пользователя | ID пользователя, который вернул товар | Внешний ключ | int | Ссылка на ID из таблицы Пользователь | 5 |

Таблица 12: Технический ремонт

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название атрибута | Описание | Домен | Тип данных | Диапазон значений | Пример атрибута |
| ID\_заявки | Уникальный идентификатор заявки на ремонт | Целые числа | int | 1, 2, 3, ... | 10 |
| ID\_пользователя | ID пользователя, создавшего заявку | Внешний ключ | int | Ссылка на ID из таблицы Пользователь | 5 |
| Описание | Описание проблемы | Строка | nvarchar(200) |  | "Не работает экран" |
| Серийный\_номер\_устройства | Серийный номер устройства | Строка | nvarchar(50) |  | "ABC123456789" |
| Наименование\_устройства | Наименование устройства | Строка | nchar(100) |  | "Смартфон Samsung Galaxy S23" |
| Статус\_заявки | ID статуса заявки | Внешний ключ | int | Ссылка на ID из таблицы Статус\_заявки | 2 |

Таблица 13: Статус\_заявки

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название атрибута | Описание | Домен | Тип данных | Диапазон значений | Пример атрибута |
| ID\_статуса\_заявки | Уникальный идентификатор статуса заявки | Целые числа | int | 1, 2, 3, ... | 2 |
| Статус | Наименование статуса заявки | Строка | nvarchar(50) |  | "В обработке" |

Таблица 14: Договор\_проведения\_ремонта

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Название атрибута | Описание | Домен | Тип данных | Диапазон значений | Пример атрибута |
| ID\_документа | Уникальный идентификатор документа | Строка | varchar(10) |  | "DOG12345" |
| ID\_заявки | ID заявки на ремонт, к которой относится договор | Внешний ключ | int | Ссылка на ID из таблицы Технический ремонт | 10 |
| ID\_пользователя | ID пользователя, подписавшего договор | Внешний ключ | int | Ссылка на ID из таблицы Пользователь | 5 |

На основании спроектированных сущностей и их атрибутов, а также анализа предметной области магазина «Электрофан» была созданная следующая ER-диаграмма(Entity Relationships – диаграмма отношения сущностей)

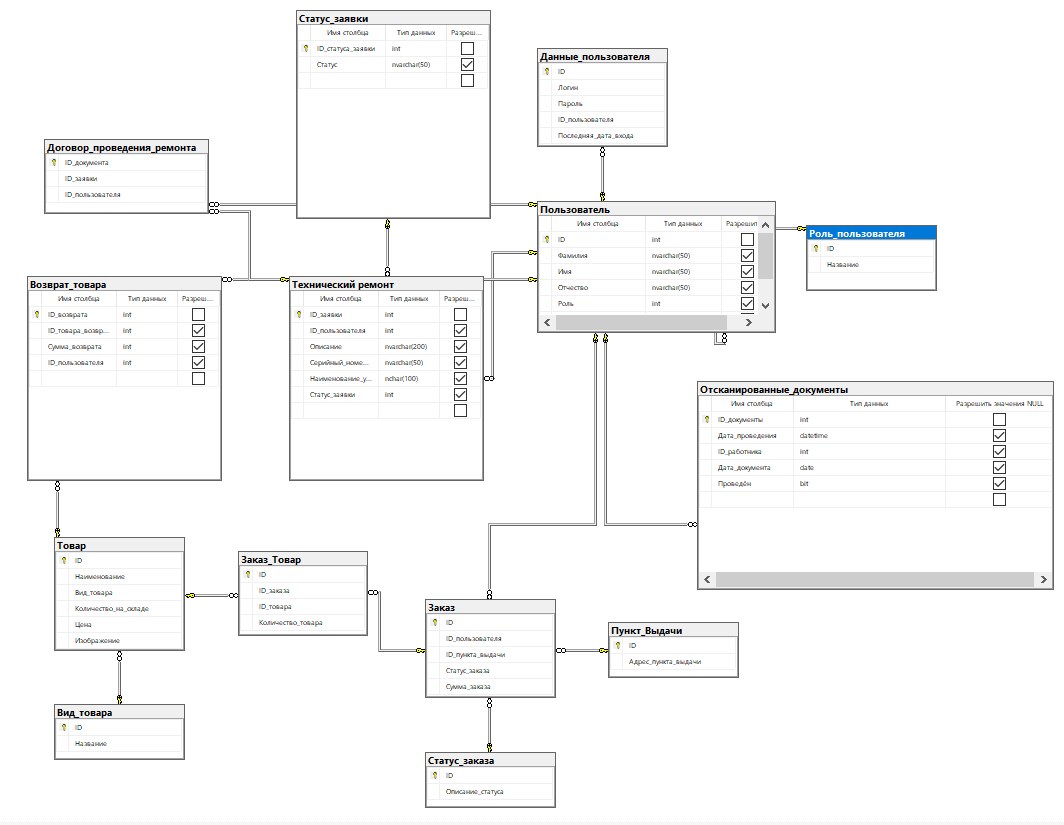


Рисунок 5.1.1 – ER-диаграмма базы данных «Электрофан»

## 5.2 Программирование на стороне SQL-сервера

При разработке приложений с базами данных часто бывает целесообразно переложить часть операций с данными с клиентского приложения на серверную часть СУБД. Это позволяет: упростить программный код приложения; уменьшить объем данных, передаваемых по сети; ускорить работу приложения. Необходимо разработать:

* Триггеры;
* Хранимые процедуры.

Триггер — это хранимая процедура, которая не вызывается непосредственно, а исполняется при наступлении определенного события (вставка, удаление, обновление строки).

Хранимые процедуры — это аналог функций в других языках программирования. Хранимые процедуры могут выполнять действия над данными автоматически: вывод данных, удаление, изменение. Особенностью процедур является то, что есть возможность передавать аргументы, (так же как и функциям в других языка) и выводить различные данные в зависимости от аргумента. Также, процедура является сущностью SQL, которую создают один раз, а затем вызывают, передавая аргументы.

Исходя из анализа предметной области и требований к ИС, были созданы следующие триггеры:

ALTER TRIGGER [dbo].[tr\_GenerateDocumentId]

ON [dbo].[Договор\_проведения\_ремонта]

INSTEAD OF INSERT

AS

BEGIN

INSERT INTO Договор\_проведения\_ремонта (ID\_документа, ID\_заявки, ID\_пользователя)

SELECT 'TR' + RIGHT('000000' + CAST(NEXT VALUE FOR DocumentIdSeq AS varchar(7)), 7),

ID\_заявки,

ID\_пользователя

FROM inserted;

END;

Данные триггер генерирует для каждого договора на тех.ремонт новый уникальный индентификационный номер(первичный ключ).

# 6 РЕАЛИЗАЦИЯ ПРОЕКТА

## 6.1 Разработка архитектуры информационной системы

Описание схемы архитектуры ИС:

Представленная схема(рис 6.6.1) на иллюстрирует трехзвенную архитектуру информационной системы (ИС). Эта архитектура разделяет систему на три основных уровня:

Клиентский уровень (WPF приложение):

Этот уровень отвечает за взаимодействие с пользователем. Он представляет данные в удобной форме и обрабатывает действия пользователя.

Представление данных: Отображение информации для пользователя (графический интерфейс, таблицы, формы).

Функциональная обработка: Обработка действий пользователя (ввод данных, нажатия кнопок, выбор элементов).

Сервер приложений (.NET Framework):

Этот уровень реализует бизнес-логику системы. Он получает запросы от клиентского уровня, обрабатывает их, взаимодействует с базой данных и возвращает результаты обратно клиенту.

Бизнес-логика: Правила и алгоритмы, определяющие работу системы.

Управление данными: Взаимодействие с базой данных для получения, добавления, изменения и удаления данных.

API (Application Programming Interface): Интерфейс для взаимодействия с клиентским уровнем, через который передаются запросы и данные.

Сервер БД (MS SQL Server):

Этот уровень отвечает за хранение данных. Он предоставляет доступ к данным для сервера приложений.

СУБД (MS SQL Server Management Studio - SSMS): Система управления базой данных, которая обеспечивает создание, хранение и управление данными. SSMS является инструментом для администрирования MS SQL Server.

База данных: Хранилище данных, структурированных в соответствии с требованиями системы.

Взаимодействие между уровнями:

Клиент <-> Сервер приложений: Клиент отправляет запросы на сервер приложений через API.

Сервер приложений обрабатывает запросы и возвращает данные клиенту. Взаимодействие может

осуществляться с помощью различных протоколов (например, HTTP, TCP). Сервер приложений <-> Сервер БД: Сервер приложений взаимодействует с базой данных через язык запросов SQL (Structured Query Language) для получения, добавления, изменения иудаления данных.

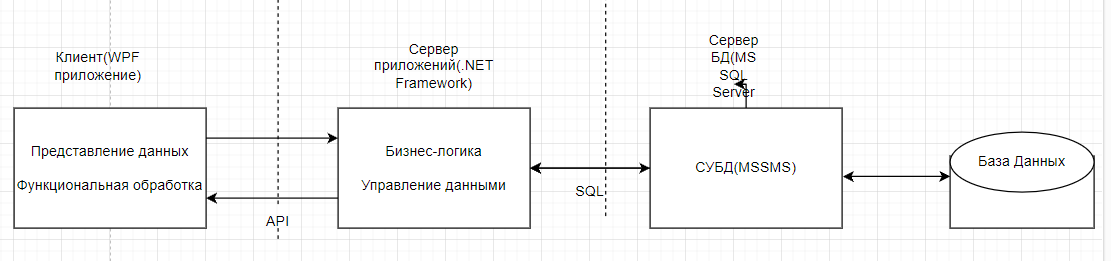


Рисунок 6.1.1 – Архитектура ИС

## 6.2 Пользовательский интерфейс ИС

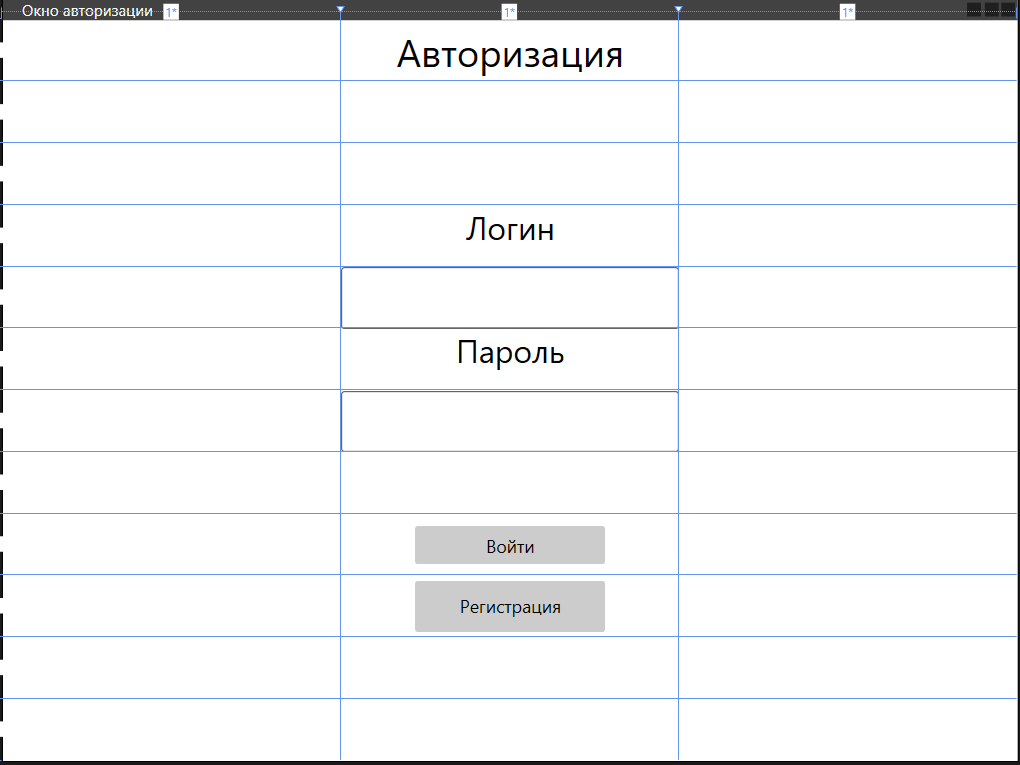


Рисунок 6.2.1 – Окно авторизации

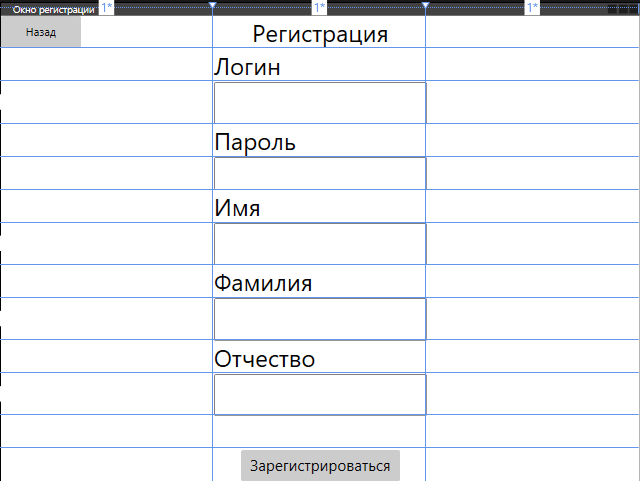


Рисунок 6.2.2 – Окно регистрации

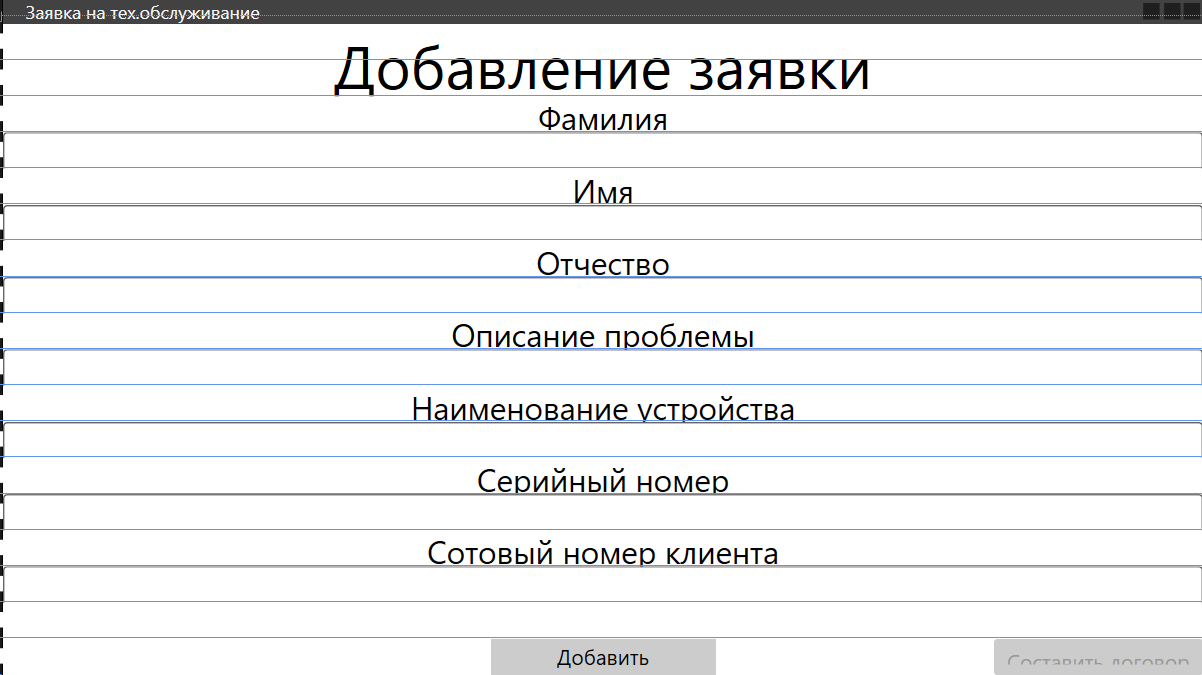


Рисунок 6.2.3 – Окно создания заявки

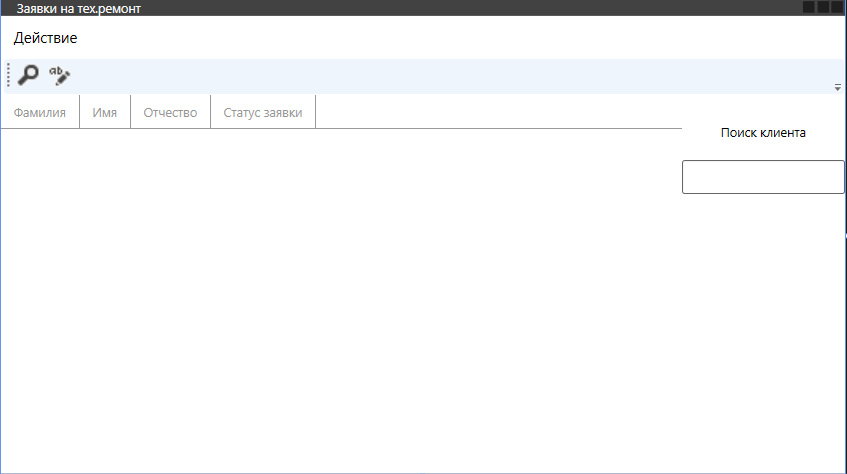


Рисунок 6.2.4 – Окно заявок на тех.ремонт

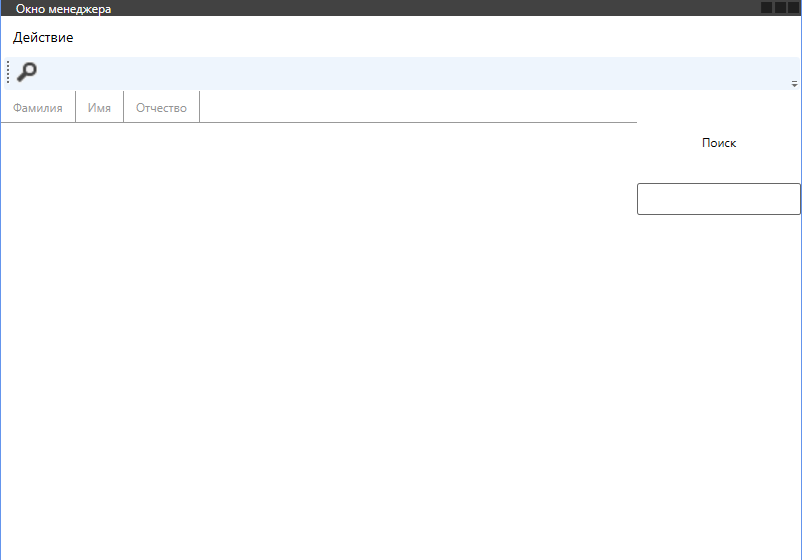


Рисунок 6.2.5 – Окно менеджера по продажам

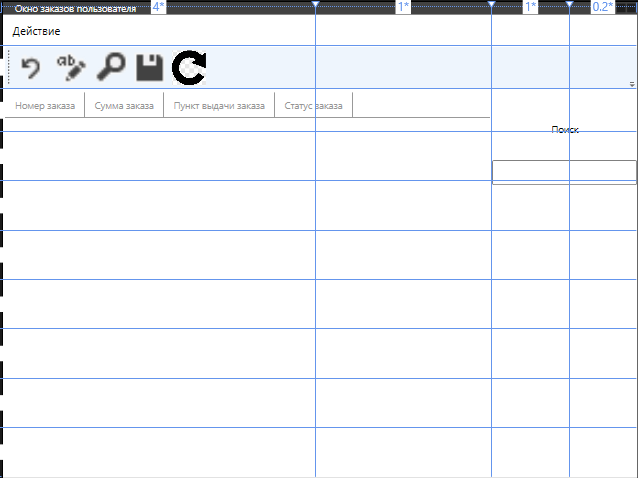


Рисунок 6.2.6 – Окно менеджера по продажам с заявками выбранного клиента

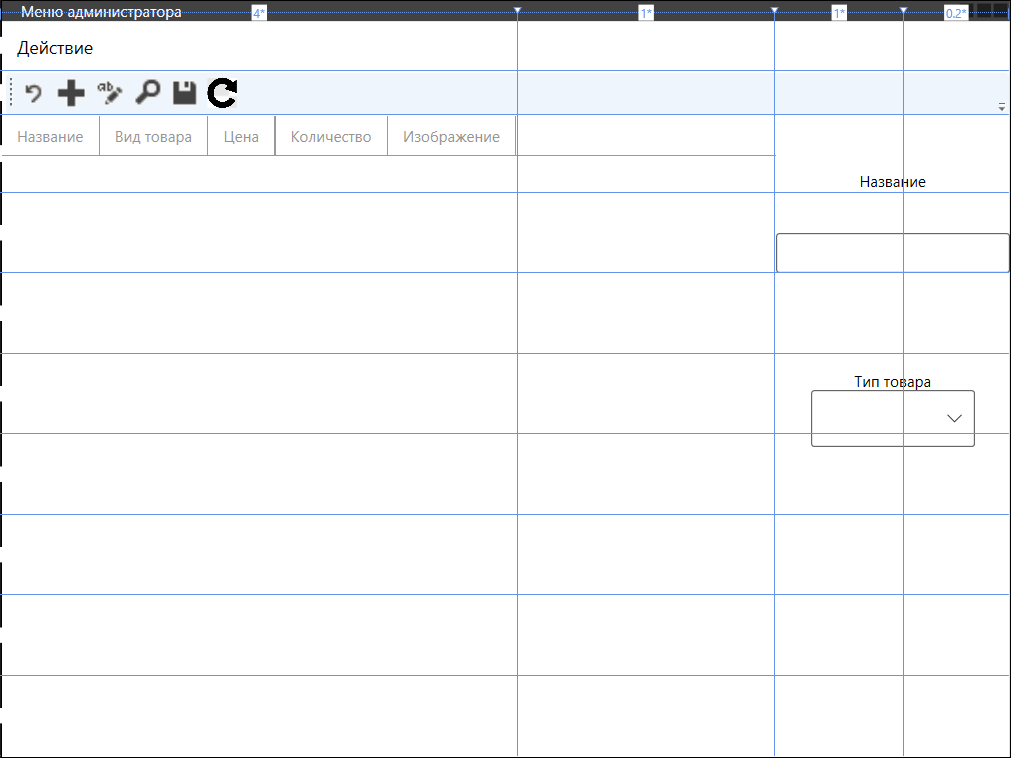


Рисунок 6.2.7 – Окно администратора по товарам

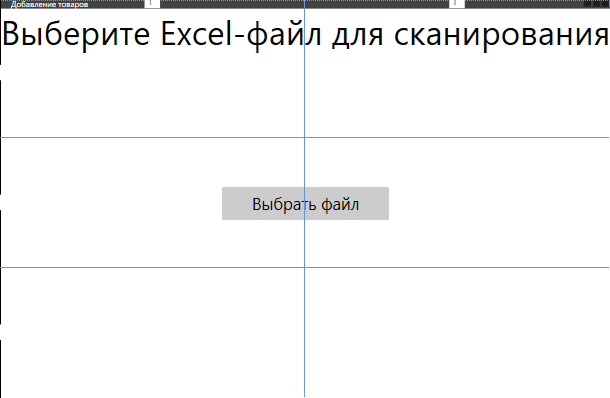


Рисунок 6.2.8 – Окно менеджера склада с добавлением товара

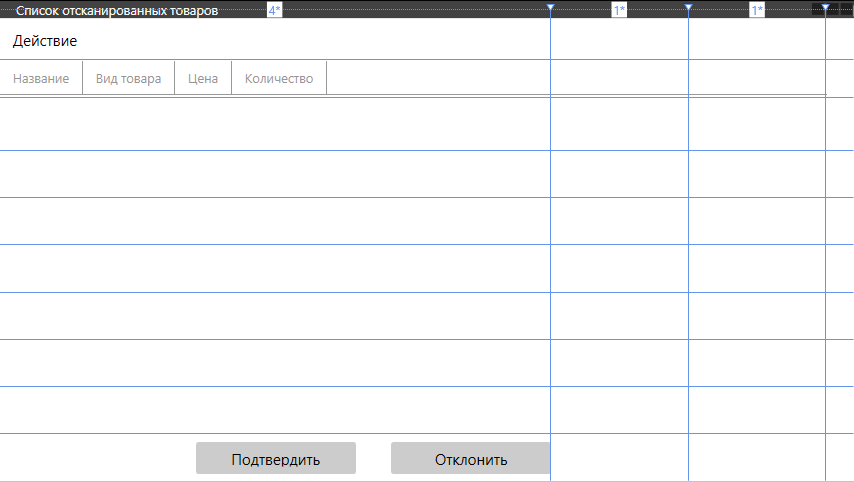


Рисунок 6.2.9 – Окно отсканированных товаров из накладной

## 6.3 Разработка классов граничных объектов

Описание классов граничных объектов и исходный код:

1. Класс MainWindow (Окно авторизации):

Описание: Представляет окно авторизации, где пользователь вводит свои учетные данные (логин и пароль)

для доступа к системе.

Атрибуты:

LoginBox (TextBox): Текстовое поле для ввода логина.

PasswordBox (PasswordBox): Поле для ввода пароля.

EnterButton (Button): Кнопка для авторизации.

RegistrationButton (Button): Кнопка для перехода к окну регистрации.

tries (int): Счетчик неудачных попыток ввода пароля.

Методы:

EnterButton\_Click(object sender, RoutedEventArgs e): Обработчик события нажатия кнопки "Войти".

Проверяет введенные данные на корректность.

Выполняет авторизацию пользователя в системе и открывает соответствующее окно в зависимости

от роли пользователя.

Блокирует систему на 10 секунд, если пользователь трижды ввел неверный пароль.

RegistrationButton\_Click(object sender, RoutedEventArgs e): Обработчик события нажатия кнопки

"Регистрация". Открывает окно регистрации.

GetHash(string input): Статический метод, возвращающий MD5-хэш строки. Используется для хеширования паролей.

Пример:

using System;

using System.Collections.Generic;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Navigation;

using System.Windows.Shapes;

using TestApp.AdminWindows;

namespace TestApp

{

/// <summary>

/// Логика взаимодействия для MainWindow.xaml

/// </summary>

public partial class MainWindow : Window

{

int tries;

public MainWindow()

{

InitializeComponent();

}

public void EnterButton\_Click(object sender, RoutedEventArgs e)

{

if (LoginBox.Text == null)

{

MessageBox.Show("Введите логин");

return;

}

if (PasswordBox.Password == null)

{

MessageBox.Show("Введите пароль");

return;

}

var passwordMD5 = GetHash(PasswordBox.Password);

Данные\_пользователя userData = new Данные\_пользователя();

userData = ElectroShopBDEntities.GetContext().Данные\_пользователя.Where(u => u.Логин == LoginBox.Text && u.Пароль == passwordMD5).FirstOrDefault();

if (userData == null && tries <= 3)

{

MessageBox.Show("Данные введены неверно");

tries++;

return;

}

if (tries >= 3 && userData == null)

{

DateTime date = DateTime.Now;

MessageBox.Show($"Вы ввели неправильные данные больше трёх раз. Система заблокирована на {10} секунд");

while (date.AddSeconds(10) > DateTime.Now)

{

LoginBox.IsEnabled = false;

PasswordBox.IsEnabled = false;

EnterButton.IsEnabled = false;

}

LoginBox.IsEnabled = true;

PasswordBox.IsEnabled = true;

EnterButton.IsEnabled = true;

return;

}

DataTransfer.userID = userData.Пользователь.ID;

DataTransfer.Name = userData.Пользователь.Имя;

DataTransfer.Surname = userData.Пользователь.Фамилия;

DataTransfer.Patronic = userData.Пользователь.Отчество;

if (userData.Пользователь.Роль == 2)

{

if (userData == null)

{

MessageBox.Show("Данные введены неверно");

return;

}

else

{

AdminWindow window = new AdminWindow();

userData.Последняя\_дата\_входа = DateTime.Now;

window.Show();

MessageBox.Show("Добро пожаловать, администратор!");

}

}

else

if (userData.Пользователь.Роль == 3)

{

if (userData == null)

{

MessageBox.Show("Данные введены неверно");

return;

}

userData.Последняя\_дата\_входа = DateTime.Now;

ManagerWindow window = new ManagerWindow();

window.Show();

MessageBox.Show("Добро пожаловать, менеджер!");

}

else

if (userData.Пользователь.Роль == 1)

{

if (userData == null)

{

MessageBox.Show("Данные введены неверно");

return;

}

else

{

userData.Последняя\_дата\_входа = DateTime.Now;

//UserWindow window = new UserWindow();

//window.Show();

MessageBox.Show("Добро пожаловать!");

}

}

else

{

if(userData.Пользователь.Роль == 4)

{

if(userData == null)

{

MessageBox.Show("Данные введены неверно");

return;

}

else

{

userData.Последняя\_дата\_входа = DateTime.Now;

AdminStorageWindow window = new AdminStorageWindow();

window.Show();

MessageBox.Show("Добро пожаловать!");

}

}

}

}

public void RegistrationButton\_Click(object sender, RoutedEventArgs e)

{

RegistrationWindow window = new RegistrationWindow();

window.Show();

this.Close();

}

public static string GetHash(string input)

{

var md5 = MD5.Create();

var hash = md5.ComputeHash(Encoding.UTF8.GetBytes(input));

return Convert.ToBase64String(hash);

}

}

}

2. Класс RegistrationWindow (Окно регистрации):

Описание: Представляет окно регистрации, где новый пользователь может создать учетную запись.

Атрибуты:

LoginBox (TextBox): Текстовое поле для ввода логина.

PasswordBox (PasswordBox): Поле для ввода пароля.

surnameBox (TextBox): Поле для ввода фамилии.

nameBox (TextBox): Поле для ввода имени.

patronicBox (TextBox): Поле для ввода отчества.

RegistrationComplete (Button): Кнопка для завершения регистрации.

BackButton (Button): Кнопка для возврата к окну авторизации.

Методы:

RegistrationComplete\_Click(object sender, RoutedEventArgs e): Обработчик события нажатия кнопки

"Зарегистрироваться".

Проверяет заполненность полей.

Создает новую запись в базе данных для пользователя и его учетных данных.

Хеширует пароль с помощью MD5.

BackButton\_Click(object sender, RoutedEventArgs e): Обработчик события нажатия кнопки "Назад".

Открывает окно авторизации и закрывает окно регистрации.

GetHash(string input): (Такой же, как в классе MainWindow).

Пример:

using System;

using System.Collections.Generic;

using System.Data.Entity.Validation;

using System.Linq;

using System.Security.Cryptography;

using System.Text;

using System.Threading.Tasks;

using System.Windows;

using System.Windows.Controls;

using System.Windows.Data;

using System.Windows.Documents;

using System.Windows.Input;

using System.Windows.Media;

using System.Windows.Media.Imaging;

using System.Windows.Shapes;

namespace TestApp

{

/// <summary>

/// Логика взаимодействия для RegistrationWindow.xaml

/// </summary>

public partial class RegistrationWindow : Window

{

public int roleID;

public RegistrationWindow()

{

InitializeComponent();

}

private void RegistrationComplete\_Click(object sender, RoutedEventArgs e)

{

if (LoginBox.Text == null)

{

MessageBox.Show("Поле логина не должно быть пустым");

return;

}

if (PasswordBox.Password == null)

{

MessageBox.Show("Поле пароль не должно быть пустым");

return;

}

if (surnameBox.Text == null)

{

MessageBox.Show("Поле фамилии не должно быть пустым");

return;

}

if (nameBox.Text == null)

{

MessageBox.Show("Поле имени не должно быть пустым");

return;

}

if (patronicBox.Text == null)

{

MessageBox.Show("Поле отчества не должно быть пустым");

return;

}

int userNumber = ElectroShopBDEntities.GetContext().Данные\_пользователя.Max(d => d.ID\_пользователя).GetValueOrDefault();

if (userNumber == 0)

{

userNumber = 1;

}

else

{

userNumber++;

}

var passwordMD5 = GetHash(PasswordBox.Password);

Данные\_пользователя userdata = new Данные\_пользователя()

{

ID\_пользователя = userNumber,

Логин = LoginBox.Text,

Пароль = passwordMD5,

};

ElectroShopBDEntities.GetContext().Данные\_пользователя.Add(userdata);

Пользователь userInfo = new Пользователь()

{

ID = userNumber,

Фамилия = surnameBox.Text,

Имя = nameBox.Text,

Отчество = patronicBox.Text,

Роль = 1

};

ElectroShopBDEntities.GetContext().Пользователь.Add(userInfo);

try

{

ElectroShopBDEntities.GetContext().SaveChanges();

MessageBox.Show("Успешная регистрация");

}

catch (DbEntityValidationException ex)

{

var sb = new StringBuilder();

foreach (var failure in ex.EntityValidationErrors)

{

sb.AppendFormat("{0} failed validation\n", failure.Entry.Entity.GetType());

foreach (var error in failure.ValidationErrors)

{

sb.AppendFormat("- {0} : {1}", error.PropertyName, error.ErrorMessage);

sb.AppendLine();

}

}

throw new DbEntityValidationException(

"Entity Validation Failed - errors follow:\n" +

sb.ToString(), ex

); // Add the original exception as the innerException

}

}

private void BackButton\_Click(object sender, RoutedEventArgs e)

{

MainWindow window = new MainWindow();

window.Show();

this.Close();

}

public static string GetHash(string input)

{

var md5 = MD5.Create();

var hash = md5.ComputeHash(Encoding.UTF8.GetBytes(input));

return Convert.ToBase64String(hash);

}

}

}

3. Класс TechinicalMaintenanceRequestAddWindow (Окно добавления заявки на тех. обслуживание):

Описание: Представляет окно для создания новой заявки на техническое обслуживание или ремонт.

Атрибуты:

LastNameTextBox (TextBox): Поле для ввода фамилии клиента.

NameTextBox (TextBox): Поле для ввода имени клиента.

PatronicTextBox (TextBox): Поле для ввода отчества клиента.

DescriptionTextBox (TextBox): Поле для описания проблемы.

DeviceNameTextBox (TextBox): Поле для ввода наименования устройства.

SerialNumberTextBox (TextBox): Поле для ввода серийного номера устройства.

TelephoneNumberTextBox (TextBox): Поле для ввода номера телефона клиента.

AddButton (Button): Кнопка для добавления заявки.

CreateDocumentButton (Button): Кнопка для создания договора.

Методы:

AddButton\_Click(object sender, RoutedEventArgs e): Обработчик события нажатия кнопки "Добавить".

Проверяет заполненность полей и корректность номера телефона.

Создает новую заявку в базе данных.

Если клиент не найден в базе, создает новую запись о клиенте.

CreateDocumentButton\_Click(object sender, RoutedEventArgs e): Обработчик события нажатия кнопки "Создать договор".

Открывает шаблон документа Word и заполняет его данными из заявки.

Генерирует уникальный номер договора.

GenerateDocumentId(): Статический метод, генерирующий уникальный номер договора.

OnWordQuit(): Обработчик события закрытия приложения Word.

4. Класс TechinicalMaintenanceRequestWindow (Окно списка заявок на тех. ремонт):

Описание: Представляет окно со списком заявок на технический ремонт.

Атрибуты:

ClientDataGrid (DataGrid): Таблица для отображения заявок.

StatusComboBox (ComboBox): ComboBox для фильтрации заявок по статусу.

NameSearchField (TextBox): Поле для поиска заявок по ФИО клиента.

Методы:

ChangeStatus\_Click(object sender, RoutedEventArgs e): Обработчик события нажатия кнопки

"Изменить статус".

NameSearchField\_TextChanged(object sender, TextChangedEventArgs e): Обработчик события

изменения текста в поле поиска. Фильтрует заявки по ФИО клиента.

ClientDataGrid\_MouseDoubleClick(object sender, MouseButtonEventArgs e): Обработчик

события двойного клика по строке в таблице. Открывает окно с подробной информацией

о заявке.

5. Класс ManagerWindow (Окно менеджера):

Описание: Представляет окно для работы менеджера по продажам.

Атрибуты:

ClientDataGrid (DataGrid): Таблица для отображения клиентов.

NameSearchField (TextBox): Поле для поиска клиентов по ФИО.

ReportButton (Button): Кнопка для генерации отчета.

Методы:

NameSearchField\_TextChanged(object sender, TextChangedEventArgs e): Обработчик события

изменения текста в поле поиска. Фильтрует клиентов по ФИО.

ClientDataGrid\_MouseDoubleClick(object sender, MouseButtonEventArgs e): Обработчик

события двойного клика по строке в таблице. Открывает окно с заказами выбранного

клиента.

Report\_Click(object sender, RoutedEventArgs e): Обработчик события нажатия кнопки

"Отчет". Генерирует отчет по заказам выбранного клиента в формате Word.

GenerateReport(Пользователь user): Метод для генерации отчета по заказам пользователя.

6. Класс AdminWindow (Окно администратора):

Описание: Представляет окно для работы администратора товаров.

Атрибуты:

GoodsData (DataGrid): Таблица для отображения товаров.

TypesOfGoodsComboBox (ComboBox): СomboBox для выбора вида товара при добавлении нового товара.

TypeSearchComboBox (ComboBox): СomboBox для фильтрации товаров по виду.

NameSearchField (TextBox): Поле для поиска товаров по наименованию.

BorderFind (Border): Контейнер, содержащий элементы поиска и фильтрации.

Методы:

Delete\_Executed(object sender, ExecutedRoutedEventArgs e): Удаляет выбранные товары из базы данных.

Delete\_CanExecute(object sender, CanExecuteRoutedEventArgs e): Проверяет, можно ли выполнить

команду удаления.

Edit\_Executed(object sender, ExecutedRoutedEventArgs e): Включает режим редактирования таблицы

товаров.

Edit\_CanExecute(object sender, CanExecuteRoutedEventArgs e): Проверяет, можно ли выполнить

команду редактирования.

Save\_CanExecute(object sender, CanExecuteRoutedEventArgs e): Проверяет, можно ли выполнить

команду сохранения изменений.

Save\_Executed(object sender, ExecutedRoutedEventArgs e): Сохраняет изменения в базе данных.

Undo\_CanExecute(object sender, CanExecuteRoutedEventArgs e): Проверяет, можно ли выполнить

команду отмены изменений.

Undo\_Executed(object sender, ExecutedRoutedEventArgs e): Отменяет изменения в базе данных.

New\_CanExecute(object sender, CanExecuteRoutedEventArgs e): Проверяет, можно ли выполнить

команду добавления нового товара.

New\_Executed(object sender, ExecutedRoutedEventArgs e): Добавляет новую запись о товаре в базу данных.

Find\_CanExecute(object sender, CanExecuteRoutedEventArgs e): Проверяет, можно ли выполнить

команду поиска.

Find\_Executed(object sender, ExecutedRoutedEventArgs e): Показывает контейнер с элементами поиска.

Refresh\_CanExecute(object sender, CanExecuteRoutedEventArgs e): Проверяет, можно ли выполнить

команду обновления данных.

Refresh\_Executed(object sender, ExecutedRoutedEventArgs e): Обновляет данные в таблице товаров.

NameSearchField\_TextChanged(object sender, TextChangedEventArgs e): Фильтрует товары по

наименованию.

TypeSearchComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e): Фильтрует

товары по виду.

SelectImageButton\_Click(object sender, RoutedEventArgs e): Открывает диалоговое окно для выбора

изображения товара.

7. Класс AdminStorageWindow (Окно управления складом администратора):

Описание: Представляет окно для управления складом и импорта данных из Excel.

Методы:

ScanningExcelButton\_Click(object sender, RoutedEventArgs e): Обработчик события нажатия кнопки "Сканировать Excel".

Открывает диалоговое окно для выбора Excel-файла.

Считывает данные из файла и выполняет валидацию.

Добавляет данные в базу данных.

Формирует отчет об ошибках, если они обнаружены.

DetermineProductType(string productName): Определяет вид товара по его названию.

## 6.4 Разработка классов сущностей

Описание классов сущностей и исходный код:

1. Класс Вид\_товара:

Описание: Представляет категорию товара.

Атрибуты:

* ID (int): Уникальный идентификатор вида товара.
* Название (string): Название вида товара (например, "Смартфоны", "Телевизоры", "Ноутбуки").

Связи:

* Товар (ICollection<Товар>): Коллекция товаров, относящихся к этому виду товара.

Исходный код:

namespace TestApp

{

using System;

using System.Collections.Generic;

public partial class Вид\_товара

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Вид\_товара()

{

this.Товар = new HashSet<Товар>();

}

public int ID { get; set; }

public string Название { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Товар> Товар { get; set; }

}

}

2. Класс Товар:

Описание: Представляет конкретный товар, продающийся в магазине.

Атрибуты:

* ID (int): Уникальный идентификатор товара.
* Наименование (string): Название товара.
* Вид\_товара (int): ID вида товара, к которому относится товар (внешний ключ к таблице Вид\_товара).
* Количество\_на\_складе (int): Количество товара на складе.
* Цена (int): Цена товара.
* Изображение (byte[]): Изображение товара в виде массива байтов.

Связи:

* Вид\_товара1 (Вид\_товара): Связь с классом Вид\_товара, определяющая категорию товара.
* Возврат\_товара (ICollection<Возврат\_товара>): Коллекция записей о возвратах товара.
* Заказ\_Товар (ICollection<Заказ\_Товар>): Коллекция записей о товарах, входящих в заказы.

Исходный код:

namespace TestApp

{

using System;

using System.Collections.Generic;

public partial class Товар

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Товар()

{

this.Возврат\_товара = new HashSet<Возврат\_товара>();

this.Заказ\_Товар = new HashSet<Заказ\_Товар>();

}

public int ID { get; set; }

public string Наименование { get; set; }

public Nullable<int> Вид\_товара { get; set; }

public Nullable<int> Количество\_на\_складе { get; set; }

public Nullable<int> Цена { get; set; }

public byte[] Изображение { get; set; }

public virtual Вид\_товара Вид\_товара1 { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Возврат\_товара> Возврат\_товара { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Заказ\_Товар> Заказ\_Товар { get; set; }

}

}

3. Класс Пользователь:

Описание: Представляет пользователя системы.

Атрибуты:

* ID (int): Уникальный идентификатор пользователя.
* Фамилия (string): Фамилия пользователя.
* Имя (string): Имя пользователя.
* Отчество (string): Отчество пользователя.
* Роль (int): ID роли пользователя (внешний ключ к таблице Роль\_пользователя).
* Сотовый\_номер (string): Номер телефона пользователя.

Связи:

* Возврат\_товара (ICollection<Возврат\_товара>): Коллекция записей о возвратах товаров, оформленных пользователем.
* Данные\_пользователя (ICollection<Данные\_пользователя>): Коллекция записей о учетных данных пользователя.
* Договор\_проведения\_ремонта (ICollection<Договор\_проведения\_ремонта>): Коллекция записей о договорах на ремонт, подписанных пользователем.
* Заказ (ICollection<Заказ>): Коллекция записей о заказах, оформленных пользователем.
* Отсканированные\_документы (ICollection<Отсканированные\_документы>): Коллекция записей о сканированных документах, отсканированных пользователем.
* Роль\_пользователя (Роль\_пользователя): Связь с классом Роль\_пользователя, определяющая роль пользователя в системе.
* Технический\_ремонт (ICollection<Технический\_ремонт>): Коллекция заявок на ремонт, оформленных пользователем.

Исходный код:

namespace TestApp

{

using System;

using System.Collections.Generic;

public partial class Пользователь

{

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2214:DoNotCallOverridableMethodsInConstructors")]

public Пользователь()

{

this.Возврат\_товара = new HashSet<Возврат\_товара>();

this.Данные\_пользователя = new HashSet<Данные\_пользователя>();

this.Договор\_проведения\_ремонта = new HashSet<Договор\_проведения\_ремонта>();

this.Заказ = new HashSet<Заказ>();

this.Отсканированные\_документы = new HashSet<Отсканированные\_документы>();

this.Технический\_ремонт = new HashSet<Технический\_ремонт>();

}

public int ID { get; set; }

public string Фамилия { get; set; }

public string Имя { get; set; }

public string Отчество { get; set; }

public Nullable<int> Роль { get; set; }

public string Сотовый\_номер { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Возврат\_товара> Возврат\_товара { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Данные\_пользователя> Данные\_пользователя { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Договор\_проведения\_ремонта> Договор\_проведения\_ремонта { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Заказ> Заказ { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Отсканированные\_документы> Отсканированные\_документы { get; set; }

public virtual Роль\_пользователя Роль\_пользователя { get; set; }

[System.Diagnostics.CodeAnalysis.SuppressMessage("Microsoft.Usage", "CA2227:CollectionPropertiesShouldBeReadOnly")]

public virtual ICollection<Технический\_ремонт> Технический\_ремонт { get; set; }

}

}

4. Класс Заказ:

Описание: Представляет заказ на покупку товара.

Атрибуты:

* ID (int): Уникальный идентификатор заказа.
* ID\_пользователя (int): ID пользователя, оформившего заказ.
* ID\_пункта\_выдачи (int): ID пункта выдачи, где будет получен заказ.
* Статус\_заказа (int): ID статуса заказа (внешний ключ к таблице Статус\_заказа).
* Сумма\_заказа (int): Общая сумма заказа.

Связи:

* Пользователь (Пользователь): Связь с классом Пользователь, определяющая пользователя, оформившего заказ.
* Пункт\_Выдачи (Пункт\_Выдачи): Связь с классом Пункт\_Выдачи, определяющая место получения заказа.
* Статус\_заказа1 (Статус\_заказа): Связь с классом Статус\_заказа, определяющая статус заказа.
* Заказ\_Товар (ICollection<Заказ\_Товар>): Коллекция товаров, входящих в заказ.

5. Класс Заказ\_Товар:

Описание: Представляет связь между заказом и товаром.

Атрибуты:

* ID (int): Уникальный идентификатор записи связи.
* ID\_заказа (int): ID заказа (внешний ключ к таблице Заказ).
* ID\_товара (int): ID товара (внешний ключ к таблице Товар).
* Количество\_товара (int): Количество товара в заказе.

6. Класс Отсканированные\_документы:

Описание: Представляет информацию о сканированных документах, например, о поступлении товаров на склад.

Атрибуты:

* ID\_документы (int): Уникальный идентификатор документа.
* Дата\_проведения (DateTime): Дата и время сканирования документа.
* ID\_работника (int): ID работника, отсканировавшего документ (внешний ключ к таблице Пользователь).
* Дата\_документа (DateTime): Дата документа.
* Проведён (bool): Флаг, указывающий, был ли документ обработан (true - да, false - нет).

7. Класс Возврат\_товара:

Описание: Представляет информацию о возврате товара клиентом.

Атрибуты:

* ID\_возврата (int): Уникальный идентификатор возврата.
* ID\_товара\_возврата (int): ID товара, который был возвращен (внешний ключ к таблице Товар).
* Сумма\_возврата (int): Сумма возврата.
* ID\_пользователя (int): ID пользователя, который вернул товар (внешний ключ к таблице Пользователь).

Описание: Хранит информацию о учетных данных пользователя.

Атрибуты:

* ID (int): Уникальный идентификатор записи учетных данных.
* Логин (string): Логин пользователя.
* Пароль (string): Хешированный пароль пользователя.
* ID\_пользователя (int): ID пользователя (внешний ключ к таблице Пользователь).
* Последняя\_дата\_входа (DateTime): Дата и время последнего входа пользователя в систему.

9. Класс Пункт\_Выдачи:

Описание: Представляет пункт выдачи товаров.

Атрибуты:

* ID (int): Уникальный идентификатор пункта выдачи.
* Адрес\_пункта\_выдачи (string): Адрес пункта выдачи.

Связи:

* Заказ (ICollection<Заказ>): Коллекция заказов, которые могут быть получены в этом пункте выдачи.

10. Класс Роль\_пользователя:

Описание: Представляет роль пользователя в системе.

Атрибуты:

* ID (int): Уникальный идентификатор роли.
* Название (string): Название роли (например, "Администратор", "Менеджер", "Сотрудник тех. поддержки").

Связи:

* Пользователь (ICollection<Пользователь>): Коллекция пользователей, имеющих эту роль.

11. Класс Статус\_заказа:

Описание: Представляет статус заказа.

Атрибуты:

* ID (int): Уникальный идентификатор статуса.
* Описание\_статуса (string): Описание статуса заказа (например, "Новый", "Оплачен", "Отправлен", "Доставлен").

12. Класс Статус\_заявки:

Описание: Представляет статус заявки на ремонт.

Атрибуты:

* ID\_статуса\_заявки (int): Уникальный идентификатор статуса.
* Статус (string): Название статуса (например, "Новая", "В обработке", "Завершена").

13. Класс Технический\_ремонт:

Описание: Представляет заявку на технический ремонт.

Атрибуты:

* ID\_заявки (int): Уникальный идентификатор заявки.
* ID\_пользователя (int): ID пользователя, оформившего заявку.
* Описание (string): Описание проблемы с устройством.
* Серийный\_номер\_устройства (string): Серийный номер устройства.
* Наименование\_устройства (string): Название устройства.
* Статус\_заявки (int): ID статуса заявки (внешний ключ к таблице Статус\_заявки).

14. Класс Договор\_проведения\_ремонта:

Описание: Представляет договор на проведение ремонта.

Атрибуты:

* ID\_документа (string): Уникальный идентификатор договора.
* ID\_заявки (int): ID заявки на ремонт (внешний ключ к таблице Технический\_ремонт).
* ID\_пользователя (int): ID пользователя, подписавшего договор (внешний ключ к таблице Пользователь).

## 6.5 Разработка программного кода

Пояснение работы отдельных функций приложения:

1. MainWindow.EnterButton\_Click(object sender, RoutedEventArgs e):

Цель: Обрабатывает нажатие кнопки "Войти".

Функциональность:

* Проверяет ввод логина и пароля: Убеждается, что пользователь ввел логин и пароль (не пустые строки). Если хотя бы одно поле пустое, выводит сообщение об ошибке и прекращает обработку.
* Хеширует пароль: Преобразует введенный пароль в MD5-хэш с помощью статического метода GetHash().
* Ищет пользователя в базе данных: Использует LINQ запрос для поиска записи в таблице Данные\_пользователя с введенным логином и хешированным паролем.
* Обработка неудачных попыток авторизации: Если пользователь не найден в базе данных или ввел неверный пароль, выводит сообщение об ошибке. Ведет счетчик неудачных попыток tries. Если количество неудачных попыток превышает 3, блокирует систему на 10 секунд.
* Проверка роли пользователя: Если авторизация успешна, определяет роль пользователя (администратор, менеджер или обычный пользователь).
* Открытие соответствующего окна: Открывает окно в зависимости от роли пользователя (окно администратора, окно менеджера или окно обычного пользователя).
* Обновление даты последнего входа: Записывает текущую дату и время в поле Последняя\_дата\_входа в таблице Данные\_пользователя.

2. MainWindow.RegistrationButton\_Click(object sender, RoutedEventArgs e):

Цель: Обрабатывает нажатие кнопки "Регистрация".

Функциональность: Открывает окно регистрации RegistrationWindow.

3. RegistrationWindow.RegistrationComplete\_Click(object sender, RoutedEventArgs e):

Цель: Обрабатывает нажатие кнопки "Зарегистрироваться".

Функциональность:

* Проверяет заполненность полей: Убеждается, что пользователь заполнил все необходимые поля. Если хотя бы одно поле пустое, выводит сообщение об ошибке и прекращает обработку.
* Генерирует уникальный ID пользователя: Выбирает следующий свободный ID в таблице Пользователь.
* Хеширует пароль: Преобразует введенный пароль в MD5-хэш.
* Создает новую запись о пользователе: Добавляет новую запись в таблицы
* Данные\_пользователя и Пользователь с введенными данными.
* Обработка ошибок: Использует try-catch блок для обработки исключений при сохранении данных в базу данных.

4. RegistrationWindow.BackButton\_Click(object sender, RoutedEventArgs e):

Цель: Обрабатывает нажатие кнопки "Назад".

Функциональность: Закрывает окно регистрации и открывает окно авторизации.

5. TechinicalMaintenanceRequestAddWindow.AddButton\_Click(object sender, RoutedEventArgs e):

Цель: Обрабатывает нажатие кнопки "Добавить".

Функциональность:

* Проверяет заполненность полей: Убеждается, что пользователь заполнил все необходимые поля. Если хотя бы одно поле пустое, выводит сообщение об ошибке и прекращает обработку.
* Проверяет корректность номера телефона: Убеждается, что введенный номер телефона состоит только из цифр.
* Ищет пользователя в базе данных: Использует LINQ запрос для поиска записи в таблице
* Пользователь с введенным ФИО.
* Создает нового пользователя: Если пользователь не найден в базе данных, создает новую запись о пользователе с введенным ФИО и номером телефона.
* Создает новую заявку: Добавляет новую запись в таблицу Технический\_ремонт с введенными данными.
* Обработка ошибок: Использует try-catch блок для обработки исключений при сохранении данных в базу данных.
* Активация кнопки "Создать договор": Включает кнопку "Создать договор", чтобы создать договор на проведение ремонта на основе созданной заявки.

6. TechinicalMaintenanceRequestAddWindow.CreateDocumentButton\_Click(object sender, RoutedEventArgs e):

Цель: Обрабатывает нажатие кнопки "Создать договор".

Функциональность:

* Проверяет наличие заявки и пользователя: Убеждается, что заявка и пользователь были успешно сохранены в базе данных.
* Генерирует уникальный ID договора: Создает уникальный номер договора с помощью метода GenerateDocumentId().
* Создает новую запись о договоре: Добавляет новую запись в таблицу
* Договор\_проведения\_ремонта с информацией о заявке, пользователе и ID договора.
* Открывает шаблон договора: Использует библиотеку Microsoft Word для открытия шаблона договора.
* Заполняет договор: Вставляет данные из заявки в соответствующие закладки шаблона.
* Сохраняет договор: Сохраняет заполненный договор в формате Word.

7. TechinicalMaintenanceRequestWindow.ChangeStatus\_Click(object sender, RoutedEventArgs e):

Цель: Обрабатывает нажатие кнопки "Изменить статус".

Функциональность: Изменяет статус выбранной заявки на ремонт.

(В предоставленном коде этот метод не полностью реализован).

8. TechinicalMaintenanceRequestWindow.NameSearchField\_TextChanged(object sender, TextChangedEventArgs e):

Цель: Обрабатывает изменение текста в поле поиска.

Функциональность: Фильтрует список заявок по ФИО клиента.

9. TechinicalMaintenanceRequestWindow.ClientDataGrid\_MouseDoubleClick(object sender, MouseButtonEventArgs e):

Цель: Обрабатывает двойной щелчок мышью по строке в таблице заявок.

Функциональность: Открывает окно с подробной информацией о выбранной заявке.

10. ManagerWindow.NameSearchField\_TextChanged(object sender, TextChangedEventArgs e):

Цель: Обрабатывает изменение текста в поле поиска.

Функциональность: Фильтрует список клиентов по ФИО.

11. ManagerWindow.ClientDataGrid\_MouseDoubleClick(object sender, MouseButtonEventArgs e):

Цель: Обрабатывает двойной щелчок мышью по строке в таблице клиентов.

Функциональность: Открывает окно с заказами выбранного клиента.

12. ManagerWindow.Report\_Click(object sender, RoutedEventArgs e):

Цель: Обрабатывает нажатие кнопки "Отчет".

Функциональность: Создает и сохраняет отчет по заказам выбранного клиента в формате Word.

13. AdminWindow.Delete\_Executed(object sender, ExecutedRoutedEventArgs e):

Цель: Удаляет выбранные товары из базы данных.

Функциональность:

* Проверяет выбор товара в таблице.
* Удаляет записи о товаре из таблицы Товар и связанные записи из других таблиц (например, Заказ\_Товар).

14. AdminWindow.Edit\_Executed(object sender, ExecutedRoutedEventArgs e):

Цель: Включает режим редактирования таблицы товаров.

Функциональность:

* Делает таблицу GoodsData доступной для редактирования.

15. AdminWindow.Save\_Executed(object sender, ExecutedRoutedEventArgs e):

Цель: Сохраняет изменения в базе данных.

Функциональность:

* Использует контекст данных ElectroShopBDEntities для сохранения изменений в таблице Товар

16. AdminWindow.Undo\_Executed(object sender, ExecutedRoutedEventArgs e):

Цель: Отменяет изменения в базе данных.

Функциональность:

* Сбрасывает все изменения, которые были сделаны в таблице Товар, возвращая их к исходному состоянию.

17. AdminWindow.New\_Executed(object sender, ExecutedRoutedEventArgs e):

Цель: Добавляет новую запись о товаре в базу данных.

Функциональность:

* Создает новый объект Товар с начальными значениями (название, вид товара, цена, количество, изображение).
* Добавляет новый объект в контекст данных ElectroShopBDEntities.
* Сохраняет изменения в базе данных.

18. AdminWindow.NameSearchField\_TextChanged(object sender, TextChangedEventArgs e):

Цель: Фильтрует товары по наименованию.

Функциональность:

* Использует LINQ запрос для получения списка товаров, соответствующих введенному наименованию.
* Обновляет источник данных для таблицы GoodsData.

19. AdminWindow.TypeSearchComboBox\_SelectionChanged(object sender, SelectionChangedEventArgs e):

Цель: Фильтрует товары по виду.

Функциональность:

* Получает выбранный вид товара из ComboBox.
* Использует LINQ запрос для получения списка товаров, относящихся к выбранному виду товара.
* Обновляет источник данных для таблицы GoodsData.

20. AdminWindow.SelectImageButton\_Click(object sender, RoutedEventArgs e):

Цель: Открывает диалоговое окно для выбора изображения товара.

Функциональность:

* Использует OpenFileDialog для выбора изображения.
* Загружает выбранное изображение в байтовый массив и сохраняет его в поле Изображение текущего товара.

21. AdminStorageWindow.ScanningExcelButton\_Click(object sender, RoutedEventArgs e):

Цель: Считывает данные о товарах из Excel-файла и добавляет их в базу данных.

Функциональность:

* Открывает диалоговое окно для выбора Excel-файла: Использует OpenFileDialog.
* Считывает данные из Excel-файла: Использует библиотеку EPPlus для чтения данных из Excel-файла.
* Выполняет валидацию данных: Проверяет, что в Excel-файле корректно заполнены все необходимые поля (название, количество, цена) и что они имеют правильный формат.
* Определяет вид товара: Использует метод DetermineProductType() для определения вида товара на основе названия товара.
* Добавляет новые товары в базу данных: Создает новые объекты Товар и сохраняет их в базе данных.
* Обновляет количество существующих товаров: Если товар уже существует в базе данных, увеличивает количество товара на складе на значение из Excel-файла.
* Создает отчет об ошибках: Если при обработке файла обнаружены ошибки, создает текстовый файл с описанием ошибок и сохраняет его на диск.

22. AdminStorageWindow.DetermineProductType(string productName):

Цель: Определяет вид товара по его названию.

Функциональность:

* Использует словарь productTypes для сопоставления слов из названия товара с видом товара.
* Возвращает вид товара, соответствующий названию, или значение по умолчанию, если вид товара не найден.

23. AdminStorageListWindow.AdminStorageListWindow(List<object[]> productList, Отсканированные\_документы document):

Цель: Создает окно с списком отсканированных товаров.

Функциональность:

* Создает список товаров ProductList из данных, считанных из Excel-файла.
* Проверяет, существует ли товар в базе данных.
* Если товар существует, добавляет его в список ProductList.
* Если товар не существует, создает новый товар и добавляет его в список ProductList.
* Инициализирует ComboBox ProductTypeComboBox списком видов товаров.

24. AdminStorageListWindow.FindExistingProduct(string productName):

Цель: Ищет существующий товар в базе данных по названию.

Функциональность:

Использует LINQ запрос для поиска товара по названию.

Возвращает найденный товар или null, если товар не найден.

25. AdminStorageListWindow.GetNextProductId():

Цель: Получает следующий свободный ID для нового товара.

Функциональность:

* Получает максимальный ID из таблицы Товар.
* Увеличивает максимальный ID на 1 и возвращает его.

26. AdminStorageListWindow.GetProductTypeId(object type):

Цель: Получает ID вида товара по его названию.

Функциональность:

* Использует LINQ запрос для поиска вида товара по названию.
* Возвращает найденный ID вида товара или 0, если вид товара не найден.

27. AdminStorageListWindow.AcceptChangesButton\_Click(object sender, RoutedEventArgs e):

Цель: Подтверждает изменения в списке отсканированных товаров и сохраняет их в базу данных.

Функциональность:

* Проверяет, существует ли товар в базе данных.
* Если товар существует, обновляет его количество на складе.
* Если товар не существует, создает новую запись о товаре в базе
* данных.
* Изменяет статус отсканированного документа на "Проведён".
* Сохраняет изменения в базе данных.

28. AdminStorageListWindow.DeclineChangesButton\_Click(object sender, RoutedEventArgs e):

Цель: Отклоняет изменения в списке отсканированных товаров и закрывает окно.

Функциональность:

* Закрывает окно с списком отсканированных товаров.

29.ManagerOrdersWindow.ManagerOrdersWindow(Пользователь selectedUser):

Цель: Создает окно с заказами выбранного пользователя.

Функциональность:

* Загружает список заказов выбранного пользователя из базы данных и отображает их в таблице.

30. ManagerOrdersWindow.NameSearchField\_TextChanged(object sender, TextChangedEventArgs e):

Цель: Фильтрует список заказов по наименованию товара.

31. ManagerOrdersWindow.ClientDataGrid\_MouseDoubleClick(object sender, MouseButtonEventArgs e):

Цель: Открывает окно с информацией о выбранном заказе.

32. ManagerOrdersWindow.Report\_Click(object sender, RoutedEventArgs e):

Цель: Генерирует отчет о выбранном заказе.

33. ManagerOrdersWindow.GenerateReport(Заказ selectedOrder):

Цель: Создает отчет по заказу в формате Word.

Функциональность:

* Открывает шаблон Word.
* Заполняет шаблон данными из заказа.
* Сохраняет отчет на диск.

35. ElectroShopBDEntities:

Описание: Класс контекста данных Entity Framework, обеспечивающий взаимодействие с базой данных.

Функциональность:

* Предоставляет методы для создания, чтения, обновления и удаления данных в базе данных.