**СОДЕРЖАНИЕ**

[**ВВЕДЕНИЕ** 2](#_Toc197004675)

[**2.1 Описание и анализ предметной области** 3](#_Toc197004676)

[**2.2 Обзор функциональности аналогов программного средства.** 5](#_Toc197004677)

[**3** **МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ И РАЗРАБОТКА ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ** 10](#_Toc197004678)

[**3.1 Анализ и формализация бизнес-процессов предметной области** 10](#_Toc197004679)

[**3.2 Разработка спецификации требований к программному средству** 14](#_Toc197004680)

[**3.3 Образ предлагаемого решения** 30](#_Toc197004681)

[**4 ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ПРОГРАММНОГО СРЕДСТВА, РАЗРАБОТАННОГО ПО ИНДИВИДУАЛЬНОМУ ЗАКАЗУ** 32](#_Toc197004682)

[**4.1 Характеристика программного средства** 32](#_Toc197004683)

[**4.2 Расчет затрат на разработку и определение цены программного средства** 33](#_Toc197004684)

[**4.3 Расчет результата от разработки и реализации программного средства** 36](#_Toc197004685)

[**4.4 Расчет показателей экономической эффективности разработки и реализации программного средства** 37](#_Toc197004686)

[**5 ПЛАН-ПРОСПЕКТ ДИПЛОМНОГО ПРОЕКТА** 38](#_Toc197004687)

[**ЗАКЛЮЧЕНИЕ** 40](#_Toc197004688)

# **ВВЕДЕНИЕ**

Преддипломная практика проходила на базе ООО «АйконГеймс» в отделе маркетинга. Организация является коммерческой структурой, основным направлением деятельности которой является разработка и издание видеоигр, а также продажа сопутствующей атрибутики.

Перед началом практики был проведен инструктаж по технике безопасности и охране труда.

Для дипломного проекта была выбрана тема «Программное средство анализа продаж и заказов в розничном магазине». Актуальность данной темы обусловлена необходимостью совершенствования процессов управления продажами и заказами, что позволяет повысить эффективность работы отдела реализации, улучшать взаимодействие с клиентами и контролировать товарные запасы. В связи с этим в ходе преддипломной практики особое внимание было уделено вопросам сбора, обработки и анализа данных о продажах и заказах, а также разработке инструментов для визуализации и прогнозирования спроса. Кроме того, были изучены следующие аспекты:

* методы автоматизированного анализа продаж и заказов;
* алгоритмы прогнозирования потребительского спроса;
* технологии обработки и визуализации данных;
* подходы к интеграции аналитических систем в процессы компании.

Анализ текущих бизнес-процессов и требований компании показал необходимость разработки программного средства, позволяющего автоматизировать анализ данных о продажах и заказах, выявлять тенденции и предоставлять рекомендации для повышения эффективности работы отдела.

Для достижения поставленной цели требовалось решить следующие задачи:

* проанализировать существующие методы учета и анализа продаж в компании;
* определить ключевые показатели, на основании которых будет осуществляться оценка эффективности продаж;
* разработать алгоритмы сбора, обработки и визуализации данных;
* реализовать программное средство на выбранной технологической платформе;
* подготовить отчет по преддипломной практике и сопроводительный графический материал.

Во время прохождения преддипломной практики использовались принципы системности и детерминизма, что обеспечило комплексный подход к выполнению задач и способствовало достижению поставленной цели. Данный подход позволил повысить эффективность анализа данных и обеспечить разработку решения, соответствующего потребностям компании.

1. **АНАЛИЗ ЛИТЕРАТУРНЫХ ИСТОЧНИКОВ   
   И ПРОГРАММНЫХ РЕШЕНИЙ ПО ТЕМЕ ДИПЛОМНОГО ПРОЕКТА**

**1.1 Описание и анализ предметной области**

Розничный магазин – это коммерческое предприятие, занимающееся продажей товаров конечным потребителям для личного или домашнего использования. В розничном магазине осуществляется ряд ключевых бизнес-процессов, направленных на обеспечение бесперебойной торговли. К основным процессам относятся: заказ товаров у поставщиков, прием и хранение товаров, реализация продукции покупателям, контроль товарных остатков, анализ продаж и заказов, а также обслуживание клиентов. В этих процессах участвуют различные роли: поставщик, менеджер по закупкам, продавец, кассир, клиент, логист, аналитик. Каждый из них выполняет свою функцию, способствующую успешному функционированию розничной торговли.

Заказ – это процесс, при котором покупатель выбирает товары, оформляет их приобретение и ожидает исполнения сделки. Он может быть оформлен в магазине, через интернет или по телефону. Процесс заказа включает несколько стадий: формирование запроса, подтверждение наличия товара, оформление сделки (договор купли-продажи), оплату и доставку товара покупателю. Заказы могут быть предварительными (когда товар заказывается у поставщика) или розничными (когда товар уже имеется в наличии и резервируется за клиентом).

Продажа – это завершение процесса заказа, при котором товар передается клиенту, а продавец получает оплату. Продажа может включать дополнительные этапы, такие как обработка возвратов, выдача чека и гарантийных документов. Продажа считается состоявшейся, если выполнены все условия договора купли-продажи, включая оплату и передачу товара. Розничная торговля является одной из наиболее значимых отраслей экономики, обеспечивающей широкий ассортимент товаров для населения и бизнеса.

Ключевые бизнес процессы розничного магазина:

1. Заказ товара у поставщика. Данный процесс осуществляется с целью обеспечения магазина товарами для их последующей продажи.
2. Реализация товара. Приобретение товара клиентом магазина осуществляется с помощью формирования заказа. Во время ожидания поставки клиент магазина может отслеживать статус заказа (этапы транспортировки, хранение товара на другом складе).
3. Анализ заказов и продаж. Процесс анализа заказов и продаж в розничном магазине включает сбор и обработку данных о продажах, оценку эффективности товаров, выявление тенденций покупательского поведения и оптимизацию ассортимента. Это помогает принимать обоснованные решения для улучшения продаж и управления запасами.

В условиях высокой конкуренции и быстрого развития технологий эффективное управление заказами и продажами становится ключевым фактором успешного ведения бизнеса. Владельцы магазинов сталкиваются с необходимостью оперативного контроля товарных остатков, своевременного пополнения ассортимента, анализа покупательского спроса и оптимизации логистических процессов. В связи с этим возрастает потребность в специализированных программных средствах, способных автоматизировать учетно-аналитическую деятельность, минимизировать влияние человеческого фактора и повысить точность управления бизнес-процессами.

Анализ заказов и анализ продаж являются важными инструментами для управления бизнесом. Анализ заказов направлен на изучение структуры спроса, временных характеристик выполнения заказов, частоты повторных заказов и эффективности поставщиков. Ключевыми показателями анализа заказов являются:

1. Конверсия продаж: это процентное соотношение посетителей, совершивших покупку, к общему числу посетителей за определенный период. Этот показатель оценивает эффективность процесса продаж. Конверсия продаж считается по следующей формуле

|  |  |
| --- | --- |
|  | (1.1) |

1. Среднее время выполнения заказа: средний показатель разницы времени с момента размещения заказа до его завершения.

Анализ продаж ориентирован на изучение тенденций в покупательском спросе, прибыльности товаров, сезонных колебаний и эффективности маркетинговых стратегий. Основные аналитические показатели:

1. Выручка от продаж: сумма средств, полученных торговой организацией, продавцом от продажи партии товара за определенный период времени. Выручка рассчитывается по следующей формуле

|  |  |
| --- | --- |
|  | (1.2) |

где – цена *i*-го товара, – количество *i*-го проданного товара

1. Объем продаж: количество проданных товаров за определенный период.
2. Маржинальная прибыль: разница между выручкой от реализации и переменными издержками за конкретный объем продукции, проданной компанией. Маржинальная прибыль рассчитывается по следующей формуле

|  |  |
| --- | --- |
|  | (1.3) |

где – траты, которые напрямую зависят от объемов продаж.

1. Средний чек: это средняя сумма денег, которую клиенты компании тратят за одну покупку. Средний чек рассчитывается по следующей формуле

|  |  |
| --- | --- |
|  | (1.4) |

где – общее количество успешно выполненных заказов

Анализ данных о продажах и заказах может проводиться с использованием различных методов: статистические модели, машинное обучение, прогнозирование временных рядов. Например, метод скользящей средней позволяет выявлять тренды продаж, а регрессионный анализ помогает определять факторы, влияющие на спрос. Современные системы бизнес-аналитики (BI-системы) позволяют строить динамические отчеты, визуализировать данные и выявлять закономерности в продажах.

Комплексный анализ заказов и продаж в розничном магазине позволяет оптимизировать товарные запасы, выявлять популярные товары, повышать удовлетворенность клиентов и увеличивать прибыльность бизнеса. Разработка специализированного программного средства анализа заказов и продаж позволит автоматизировать учетно-аналитическую деятельность, повысить точность прогнозирования спроса, минимизировать влияние человеческого фактора и создать основу для стратегического развития магазина. Благодаря такому подходу предприниматели смогут оперативно реагировать на изменения рынка, принимать обоснованные управленческие решения и добиваться устойчивого роста финансовых показателей.

## **Обзор функциональности аналогов программного средства.**

Современный рынок программного обеспечения предлагает широкий выбор решений для автоматизации анализа продаж и заказов в розничной торговле. Эти системы позволяют значительно повысить эффективность бизнеса за счет автоматизации учета, управления товарными запасами, финансового анализа и взаимодействия с клиентами. В данном разделе представлен обзор наиболее популярных программных продуктов, используемых в сфере розничной торговли. Рассмотрим их основные функциональные возможности, преимущества и недостатки.

1С:Розница – это одно из наиболее распространенных решений для автоматизации учета продаж и заказов в розничных магазинах. Данный программный продукт широко используется как небольшими торговыми точками, так и крупными торговыми сетями.

Функциональные возможности:

* Автоматизация учета продаж и заказов, в том числе в многоточечной торговой сети.
* Управление складскими запасами с возможностью инвентаризации и учета товарных остатков.
* Финансовый учет, включая расчет себестоимости товаров и управление кассовыми операциями.
* Инструменты для работы с клиентами, включая поддержку программ лояльности, дисконтных карт и акций.
* Генерация детализированных аналитических отчетов по продажам, товарообороту и прибыльности товаров.
* Интеграция с кассовым оборудованием, эквайринговыми терминалами, сканерами штрих-кодов.

Преимущества:

* Гибкость настройки под различные форматы розничной торговли.
* Интеграция с другими продуктами 1С (например, 1С:Бухгалтерия).
* Развитая аналитическая система и возможность детального учета всех операций.

Недостатки:

* Высокая сложность внедрения и необходимость обучения персонала.
* Высокая стоимость лицензий и технического обслуживания.
* Повышенные требования к серверному оборудованию при использовании в крупных компаниях.

МойСклад – облачное программное решение для управления заказами, продажами и складскими запасами, ориентированное на малый и средний бизнес.

Функциональные возможности:

* Автоматизация учета продаж и заказов с возможностью интеграции с интернет-магазинами.
* Управление складом, включая учет товарных остатков, контроль поставок и инвентаризацию.
* Финансовый учет, ведение платежных операций, расчет рентабельности продаж.
* CRM-модуль для управления клиентской базой, анализа покупательской активности и сегментации клиентов.
* Гибкая система отчетности с возможностью экспорта данных в различные форматы.
* Доступ к системе через веб-интерфейс без необходимости установки на локальные компьютеры.

Преимущества:

* Простота внедрения и использования, интуитивно понятный интерфейс.
* Доступность для малого бизнеса благодаря гибкой ценовой политике.
* Возможность работы в облаке без привязки к конкретному устройству.

Недостатки:

* Ограниченный функционал для крупных розничных сетей.
* Ограниченные возможности по настройке интеграций с оборудованием.

RetailCRM – специализированная система для автоматизации процессов управления продажами, заказами и клиентской базой в розничной торговле.

Функциональные возможности:

* Автоматизация многоканальных продаж, синхронизация заказов из онлайн-магазинов, маркетплейсов и торговых точек.
* Управление клиентской базой с возможностью анализа покупательского поведения и персонализации предложений.
* Контроль товарных запасов, прогнозирование потребности в товарах, автоматическое формирование заказов поставщикам.
* Финансовая аналитика, расчет прибыли, оценка эффективности продаж.
* Поддержка интеграции с кассовым оборудованием, ERP-системами, платежными системами.
* Инструменты для автоматизированного маркетинга, включая e-mail и SMS-рассылки.

Преимущества:

* Развитые возможности по работе с клиентами и аналитике продаж.
* Гибкость интеграции с различными системами и сервисами.
* Поддержка многоканальных продаж и e-commerce.

Недостатки:

* Высокая стоимость лицензии и поддержки.
* Сложность освоения для пользователей без технической подготовки.
* Избыточность для небольших магазинов с низким объемом заказов.

Атол Sigma – облачное программное решение для автоматизации розничной торговли, предназначенное для малого и среднего бизнеса.

Функциональные возможности:

* Учет продаж и заказов с возможностью интеграции с кассовыми аппаратами и платежными системами.
* Автоматизированный контроль товарных остатков и складской учет.
* Работа с клиентами, поддержка программ лояльности и бонусных карт.
* Инструменты отчетности, позволяющие анализировать продажи, товарооборот и эффективность персонала.
* Онлайн-доступ к данным с возможностью удаленного управления торговыми точками.

Преимущества:

* Простота внедрения и использования.
* Доступность для малого бизнеса.
* Интеграция с фискальными регистраторами и эквайринговыми системами.

Недостатки:

* Ограниченные аналитические возможности.
* Недостаточный функционал для работы с большими объемами данных.
* Ограниченные возможности масштабирования для крупных торговых сетей.

Сравнительный обзор функциональности аналогов разрабатываемого программного средства представлен в таблице 1.1.

Таблица 1.1 – Сравнительный обзор функциональности аналогов

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Параметр | 1С:Розница | МойСклад | RetailCRM | Атол Sigma |
| Основное назначение | Управление розничной торговлей | Учет продаж и складских запасов | CRM и многоканальные продажи | Управление кассой и складом |
| Учет заказов и продаж | Поддерживается | Поддерживается | Поддерживается | Поддерживается |
| Складской учет | Полный функционал | Ограниченный | Полный функционал | Ограниченный |
| Финансовый учет | Полный функционал | Базовый учет | Полный функционал | Базовый учет |
| CRM и работа с клиентами | Базовая CRM | Развитая CRM | Расширенная CRM | Поддержка бонусных карт |
| Аналитика и отчетность | Детализированные отчеты | Гибкие отчеты | Глубокая аналитика | Базовые отчеты |

Продолжение таблицы 1.1

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Интеграция с оборудованием | Широкая поддержка | Ограниченная поддержка | Поддерживается | Полная поддержка |
| Поддержка онлайн-торговли | Ограниченная | Полная | Полная | Отсутствует |
| Простота использования | Сложное внедрение | Простая настройка | Сложное внедрение | Простая настройка |
| Стоимость лицензии | Высокая | Средняя | Высокая | Средняя |

Существующие программные решения для управления продажами и заказами в розничной торговле обладают как сильными, так и слабыми сторонами. Среди основных недостатков можно выделить сложность внедрения и настройки в таких системах, как 1С:Розница и RetailCRM, высокую стоимость лицензий и технического обслуживания, а также ограниченные возможности масштабирования в МойСклад и Атол Sigma. Эти проблемы можно устранить, разработав более доступное и интуитивное программное средство с гибкими настройками, упрощающими адаптацию под бизнес-процессы заказчика.

При этом важные преимущества существующих решений, такие как развитая аналитическая система, поддержка многоканальных продаж, интеграция с онлайн-торговлей и CRM-функционал для управления клиентской базой, могут быть использованы в разрабатываемом программном продукте. Оптимальное сочетание простоты использования, широких аналитических возможностей и гибкости интеграции позволит создать эффективное программное средство, отвечающее потребностям как малого, так и крупной розницы.

1. **МОДЕЛИРОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ   
   И РАЗРАБОТКА ТРЕБОВАНИЙ К ПРОГРАММНОМУ СРЕДСТВУ**

**2.1 Анализ и формализация бизнес-процессов предметной области**

Модель «как есть» (*AS-IS*) в *BPMN* отображает текущее состояние бизнес-процесса до его улучшения. Она помогает понять, как процесс функционирует сейчас, выявить слабые места и возможности для улучшения. Модель «как есть» (*AS-IS*) в нотации *BPMN* для бизнес-процесса анализа заказов и продажпредставлена на рисунке 2.2.

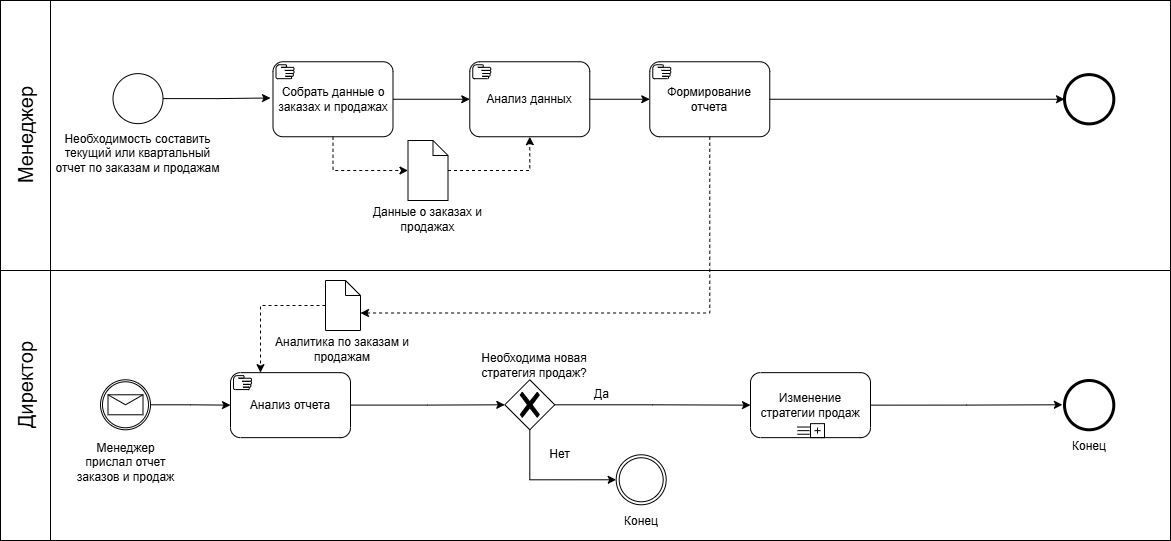


Рисунок 2.2 – Модель «как есть» бизнес-процесса анализа заказов и продаж в нотации *BPMN*

Диаграмма бизнес-процесса «как есть» (*AS-IS*) описывает текущий, неоптимизированный процесс анализа заказов и продаж в розничном магазине без использования автоматизированной системы. Со стороны менеджера процесс основан на ручном сборе данных о заказах и продаж из нецентрализованных источников и ручном подсчете показателей продаж и заказов с целью их анализа, и формировании отчета. Со стороны директора – это анализ отчета и принятия решения касательно стратегии продаж. Для декомпозиции бизнес процесса, выделим его основные функции. Функции бизнес процесса представлены в таблице 2.1.

Таблица 2.1 – Функции процесса

| № | Наименование функции | Роль | ПО |
| --- | --- | --- | --- |
| 1 | Сбор данных о заказах и продажах | Менеджер | Excel, ручной способ |
| 2 | Анализ данных | Менеджер | Excel, ручной способ |
| 3 | Формирование отчета | Менеджер | Excel, ручной способ |
| 4 | Анализ отчета | Директор | Ручной способ |

Для анализа функций процесса аналитики заказов и продаж розничного магазина важно рассмотреть каждую функцию с точки зрения их вклада в создание ценности, дублирования, чрезмерного контроля, а также возможных узких мест, которые могут замедлять процесс. Также следует определить места возвратов в схеме и передачу результата процесса его потребителю. Результаты анализа функций представлен в таблице 3.2.

Таблица 2.2 – Результаты анализа функций

| Наименование функции | По отношению к созданию ценности | Имеет дублирование | Излишний контроль | Узкое место |
| --- | --- | --- | --- | --- |
| Сбор данных о заказах и продажах | - | Нет | Нет | Нет |
| Анализ данных | ? | Нет | Нет | Нет |
| Формирование отчета | + | Нет | Нет | Нет |
| Анализ отчета | + | Нет | Нет | Нет |
| Количество возвратов | 1 возврат. Анализ отчета. | | | |
| Передача результата процесса его потребителю | Результат передается потребителю (директор получает отчет) | | | |

Сбор данных о заказах и продажах: Не создает ценности, поскольку каждый раз вручную заниматься сбором данных о заказах и продажах занимает много времени, а также может привести к недочетам в данных.

Анализ данных: Ценность под вопросом, поскольку исходя из предыдущей функции могут присутствовать ошибки в анализируемых данных.

Возвраты могут происходить на стадии анализа отчета. Директор может попросить переделать отчет с другими показателями либо с другим периодам.

Передача результата осуществляется в виде отчета аналитики. Менеджер предоставляет директору отчет и впоследствии директор его анализирует что является конечной целью процесса.

Для наглядности составим гистограмму, представленную на рисунке 3.3.

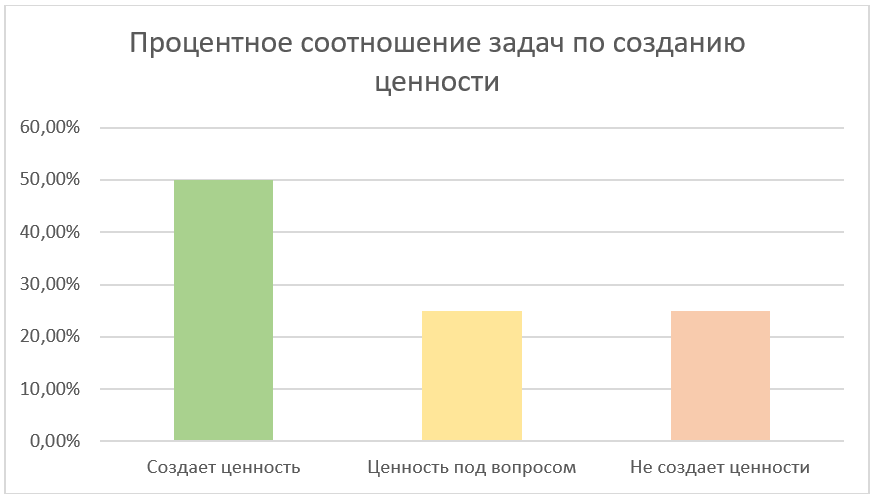


Рисунок 2.3 - Гистограмма распределения ценности задач

На основании анализа графической схемы процесса анализа заказов и продаж розничного магазина, можно выявить несколько проблем, которые влияют на эффективность процесса. Перечень бизнес-проблем процесса представлен в таблице 2.3.

Таблица 2.3 – Описание бизнес проблем процесса

| № | Описание бизнес-проблем процесса |
| --- | --- |
| 1 | **Отсутствие централизованной системы:** Отсутствие централизованной системы заказов ведет к возможной потере данных о заказах и продажах, а также отнимает много времени на сбор данных. |
| 2 | **Нет автоматизации расчётов:** Отсутствие системы расчётов показателей продаж ведет к увеличению времени анализа расчётов показателей аналитики и формированию отчета, а так же в возможным ошибочным показателем ввиду неполных либо ошибочных данных. |

На основании результатов анализа, полученных при выполнении указанных выше техник, разработаем мероприятия по автоматизации усовершенствованного бизнес-процесса для каждой функции в таблице 3.7.

Таблица 2.4 – Мероприятия по автоматизации бизнес-процесса

| № | Наименование функции | Мероприятие |
| --- | --- | --- |
| 1 | Сбор данных о заказах и продажах | Автоматизировать процесс сбора данных, путем создания системы учета заказов и продаж |
| 2 | Анализ данных | Автоматизировать процесс анализа данных, путем создания системы анализа данных заказов и продаж |
| 3 | Формирование отчета | Автоматизировать процесс анализа данных, путем создания системы анализа данных заказов и продаж |
| 4 | Анализ отчета | Автоматизировать процесс анализа отчета, путем создания системы анализа показателей продаж, которая будет решать какую стратегию применить (или оставить все как есть) |

Автоматизация анализа заказов и продаж в розничном магазине позволит сократить большинство ошибок ручного сбора данных о заказах и продажах, ручного анализа показателей и формирования отчета, так же система позволит сэкономить большое количество времени на формирование итогового отчета.

В результате корректировки бизнес-процесса была разработана модель «как будет» бизнес-процесса анализа заказов и продаж в нотации BPMN, которая представлена на рисунке 2.6.

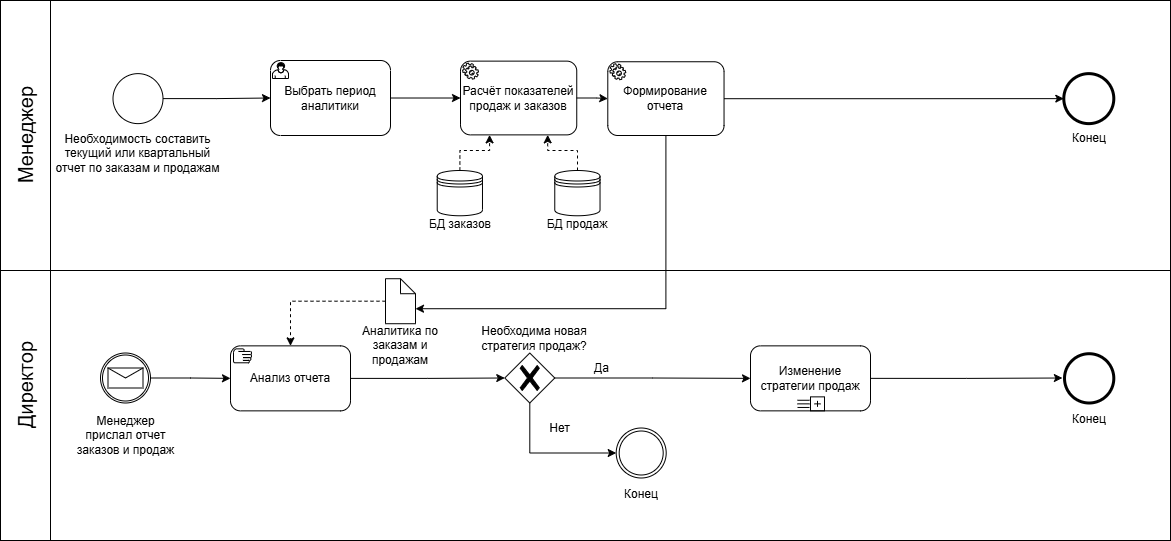


Рисунок 2.6 – Модель «как будет» бизнес-процесса анализа заказов и продаж в нотации *BPMN*

Модель «как будет» бизнес-процесса, представленная в нотации *BPMN*, описывает усовершенствованный процесс анализа заказов и продаж с учетом автоматизации ключевых этапов.

**2.2 Разработка спецификации требований к программному средству**

Разрабатываемое программное средство должно обеспечивать процесс анализа заказов и продаж, учета заказов и продаж и управления товарами. В процесс аналитики заказов и продаж входит просмотр истории продаж, просмотр динамики заказов и продаж, а также основные экономические показатели. В процесс учета заказов и продаж входит входит выбор товаров из каталога, оформления заявки на покупку товаров, отслеживания состояния заказа. В процесс управления товарами и складами входит создание, редактирование товаров и складов, отслеживание количество товаров на складе. Областью применения программного средства является сфера розничных продаж.

Функции программного средства подразделяются на две категории в зависимости от роли пользователя: клиент, менеджер. Для клиента программное средство должно предоставлять доступ к каталогу товаров и возможность оставит заявку на покупку товаров и отслеживать состояние заказа. Для менеджера программное средство должно предоставлять доступ к управления заявками клиентов, складами и товарами, а также к системе аналитики заказов и продаж. В таблице 3.8 представлены основные функции программного средства, распределенные по группам пользователей.

Таблица 2.8 – Основные функции программного средства

|  |  |
| --- | --- |
| Функция | Тип пользователя |
| Просмотр аналитики заказов и продаж | Менеджер |
| Генерация аналитического отчета | Менеджер |
| Управление заявками пользователей | Менеджер |
| Управление складами и товарами | Менеджер |
| Управление заявками пользователей | Менеджер |
| Просмотр каталога товаров | Клиент |
| Оформление заявки | Клиент |

Далее перейдем к описанию бизнес-требований к разрабатываемому программному продукту. Для этого необходимо описать основные преимущества, которые новое программное средство дает ее заказчикам, покупателям и пользователям.

Бизнес-требования – это высокоуровневая бизнес-цель организации или заказчиков системы. Бизнес-требования выражают цель, ради которой разрабатывается продукт. Бизнес-требования описывают основные преимущества, которые онлайн-сервиса выявления плагиата в научных статьях дает пользователям. Бизнес-требования непосредственно влияют на то, какие пользовательские требования будут реализованы и в какой последовательности.

Опишем основные потребности бизнеса:

* Управление запасами – поддержание оптимального количества товаров, предотвращение дефицита и излишков.
* Анализ спроса и продаж – выявление трендов, прогнозирование потребностей клиентов.
* Автоматизация процессов – ускорение операций, снижение ошибок и затрат.
* Управление клиентами – повышение лояльности, персонализированные предложения.
* Финансовый контроль – анализ прибыли, расходов и рентабельности.

Следующий этап разработки программного средства – разработка бизнес-требований к разрабатываемому программному средству. Для этого необ ходимо определить основные преимущества, которые мы реализуем в разрабатываемом средстве.

Бизнес-цели:

1. *BO-1* – Повышение эффективности розничного магазина за счет автоматизированного анализа продаж и заказов, что позволит оптимизировать запасы товаров и улучшить прогнозирование спроса.
2. *BO-2* – Автоматизация учета продаж и заказов для сокращения временных затрат сотрудников и снижения вероятности ошибок, связанных с ручным вводом данных.
3. *BO-3* – Увеличение дохода магазина на 15% за счет детального анализа покупательского поведения и выявления популярных товаров.
4. *BO-4* – Снижение уровня нереализованных остатков товаров на 20% благодаря оперативному контролю запасов и прогнозированию потребностей покупателей.

Критерии успеха:

1. *SM-1* – Увеличение точности прогноза спроса и оптимизации запасов минимум на 25% в течение первого года эксплуатации системы.
2. *SM-2* – Сокращение времени на обработку заказов и продаж минимум на 30% за счет автоматизации учета и отчетности.
3. *SM-3* – Повышение коэффициента удовлетворенности клиентов не менее чем на 20% благодаря улучшенной аналитике заказов и персонализированным предложениям.
4. *SM-4* – Достижение уровня автоматической фиксации и обработки заказов не менее 90%, исключая человеческий фактор в учете продаж.
5. *SM-4*– Увеличение скорости формирования аналитических отчетов до 1 минуты на крупные объемы данных, что ускорит принятие управленческих решений.

Бизнес-риски:

1 *RI-1* – Риск некорректного анализа данных. Ошибки в алгоритмах обработки продаж и заказов могут привести к неверным прогнозам спроса, что может негативно повлиять на запасы и объемы закупок. Требуется тщательная настройка системы и тестирование алгоритмов.

2 *RI-2* – Риск отказа пользователей от внедрения. Сотрудники магазина могут сопротивляться использованию новой системы из-за необходимости обучения и изменения привычных процессов. Для снижения риска требуется интуитивный интерфейс и обучение персонала.

3 *RI-3* – Риск технических сбоев. Возможны сбои в работе ПО, потеря данных или ошибки при интеграции с другими учетными системами. Для минимизации риска необходимо внедрение системы резервного копирования и регулярное тестирование.

4 *RI-4* – Риск утечки данных. Несанкционированный доступ к аналитике продаж и клиентской базе может привести к потере конкурентных преимуществ и юридическим последствиям. Требуется защита данных путем шифрования и разграничения уровней доступа.

5 *RI-5* – Риск неконкурентоспособности. Конкуренты могут предложить более функциональные или доступные решения, что снизит привлекательность программного средства. Для минимизации риска требуется постоянное развитие продукта и добавление новых функций.

Функциональные требования – это заявления о том, как должна вести себя система. Они определяют, что система должна делать, чтобы удовлетворить потребности или ожидания пользователя. Функциональные требования можно рассматривать как функции, которые обнаруживает пользователь.

Диаграмма вариантов использования (*Use Case Diagram*) является одним из наиболее популярных инструментов в разработке программного обеспечения для моделирования функциональных требований и поведения системы из точки зрения конечных пользователей.

Спецификация функциональных требований представлена диаграммой «*Use Case*», представленной на рисунке 2.7.

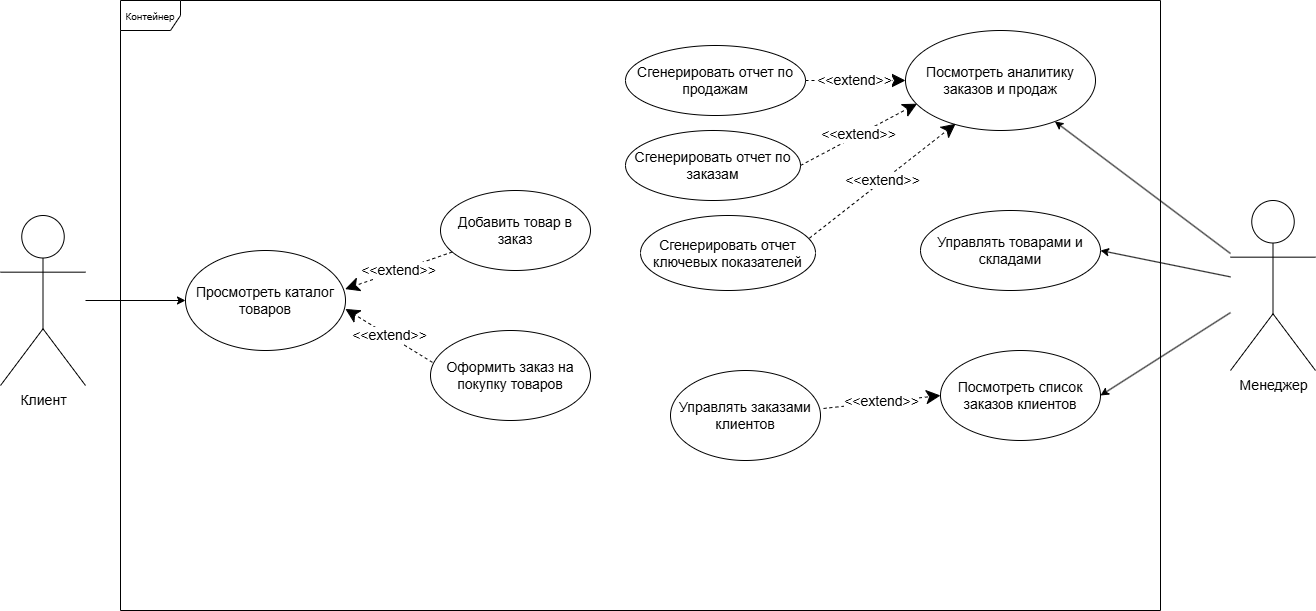


Рисунок 2.7 – Диаграмма «*Use Case*»

Описание действующих лиц представлено в таблице 2.9.

Таблица 2.9 – Описание действующих лиц

|  |  |
| --- | --- |
| Действующее лицо | Вариант использования |
| Менеджер | *UC*-1 Посмотреть аналитику заказов и продаж |
| Менеджер | *UC*-2 Сгенерировать отчет по продажам |
| Менеджер | *UC*-3 Сгенерировать отчет по заказам |
| Менеджер | *UC*-4 Сгенерировать отчет ключевых показателей |
| Менеджер | *UC*-5 Посмотреть список заказов клиентов |
| Менеджер | *UC*-6 Управлять заказами клиентов |
| Менеджер | *UC*-7 Управлять товарами и складами |
| Клиент | *UC*-8 Просмотреть каталог товаров |
| Клиент | *UC*-9 Добавить товар в заказ |
| Клиент | *UC*-10 Оформить заявку на покупку товаров |

В таблице 2.10 приведено подробное описание вариантов использования. В данной таблице описываются ход действий, пред- и постусловия, альтернативные потоки действий и исключения.

Таблица 2.10 – Описание вариантов использования

|  |  |
| --- | --- |
| **Имя варианта использования** | ***UC*-1 Посмотреть аналитику заказов и продаж** |
| Краткое описание варианта  Использования | Данный вариант использования описывает процесс просмотра аналитики заказов и продаж |
| Действующие лица, взаимодействующие с вариантом использования | Менеджер |
| Предусловия для выполнения  варианта использования | В за выбранный временной период в системе должна быть зарегистрирована по крайнем мере одна продажа |
| Основной поток действий при  исполнении варианта использования | Данный вариант использования начинает выполняться, когда пользователь решает посмотреть аналитику и историю продаж. Пользователь может выбрать тип аналитики: по продажам или по заказам. Так же пользователь может выбрать временные рамки аналитики. |
| Альтернативный поток действий при исполнении варианта  Использования | Отсутствует |
| Исключения при выполнении  варианта использования | Вывод сообщения о отсутствие продаж за данные период |
| Постусловия после выполнения  варианта использования. | Отсутствуют |
| Связь с другими вариантами использования | *UC-2, UC-3, UC-4* |
| **Имя варианта использования** | ***UC*-2 Сгенерировать отчет по продажам** |
| Краткое описание варианта  Использования | Данный вариант использования описывает процесс генерации отчета по продажам |
| Действующие лица, взаимодействующие с вариантом использования | Менеджер |

Продолжение таблицы 2.10

|  |  |
| --- | --- |
| Предусловия для выполнения  варианта использования | В за выбранный временной период в системе должна быть зарегистрирована по крайнем мере одна продажа |
| Основной поток действий при  исполнении варианта использования | Данный вариант использования начинает выполняться, когда пользователь решает сгенерировать отчет по продажам. Пользователь может выбрать временные рамки отчета. |
| Альтернативный поток действий при исполнении варианта  Использования | Отсутствует |
| Исключения при выполнении  варианта использования | Вывод сообщения о отсутствие продаж за данные период |
| Постусловия после выполнения  варианта использования. | Отсутствуют |
| Связь с другими вариантами использования | *UC-1* |

|  |  |
| --- | --- |
| **Имя варианта использования** | ***UC*-3 Сгенерировать отчет по заказам** |
| Краткое описание варианта  Использования | Данный вариант использования описывает процесс генерации отчета по заказам |
| Действующие лица, взаимодействующие с вариантом использования | Менеджер |
| Предусловия для выполнения  варианта использования | В за выбранный временной период в системе должна быть зарегистрирован по крайнем мере один заказ |
| Основной поток действий при  исполнении варианта использования | Данный вариант использования начинает выполняться, когда пользователь решает сгенерировать отчет по заказам. Пользователь может выбрать временные рамки отчета. |

Продолжение таблицы 2.10

|  |  |
| --- | --- |
| **Имя варианта использования** | ***UC*-3 Сгенерировать отчет по заказам** |
| Альтернативный поток действий при исполнении варианта  Использования | Отсутствуют |
| Исключения при выполнении  варианта использования | Вывод сообщения о отсутствие заказов за данные период |
| Постусловия после выполнения  варианта использования. | Отсутствуют |
| Связь с другими вариантами использования | *UC*-1 |

|  |  |
| --- | --- |
| **Имя варианта использования** | ***UC*-4 Сгенерировать отчет ключевых показателей** |
| Краткое описание варианта  Использования | Данный вариант использования описывает процесс просмотра аналитики заказов и продаж |
| Действующие лица, взаимодействующие с вариантом использования | Менеджер |
| Предусловия для выполнения  варианта использования | В за выбранный временной период в системе должна быть зарегистрирована по крайнем мере одна продажа |
| Основной поток действий при  исполнении варианта использования | Данный вариант использования начинает выполняться, когда пользователь решает сгенерировать отчет по ключевым показателям. Пользователь может выбрать временные рамки отчета. |
| Альтернативный поток действий при исполнении варианта  Использования | Отсутствует |
| Исключения при выполнении  варианта использования | Вывод сообщения о отсутствие продаж за данные период |
| Постусловия после выполнения  варианта использования. | Отсутствуют |
| Связь с другими вариантами использования | *UC-1* |

Продолжение таблицы 2.10

|  |  |
| --- | --- |
| **Имя варианта использования** | ***UC*-5 Посмотреть список заказов клиентов** |
| Краткое описание варианта  Использования | Данный вариант использования описывает просмотр пользовательских заказов |
| Действующие лица, взаимодействующие с вариантом использования | Менеджер по продажам |
| Предусловия для выполнения  варианта использования | Отсутствуют |
| Основной поток действий при  исполнении варианта использования | Данный вариант использования начинает выполняться, когда менеджер решает посмотреть список пользовательских заказов |
| Альтернативный поток действий при исполнении варианта  Использования | Отсутствуют |
| Исключения при выполнении  варианта использования | Отсутствуют |
| Постусловия после выполнения  варианта использования. | Отсутствуют |
| Связь с другими вариантами использования | *UC-6* |

|  |  |
| --- | --- |
| **Имя варианта использования** | ***UC*-6 Управлять заказами клиентов** |
| Краткое описание варианта  Использования | Данный вариант использования описывает процесс управление заявкой пользователя |
| Действующие лица, взаимодействующие с вариантом использования | Менеджер |
| Предусловия для выполнения  варианта использования | Отсутствует |

|  |  |
| --- | --- |
| Альтернативный поток действий при исполнении варианта  Использования | Отсутствуют |

|  |  |
| --- | --- |
| Исключения при выполнении  варианта использования | Отсутствуют |
| Постусловия после выполнения  варианта использования. | Отсутствуют |
| Связь с другими вариантами использования | *UC*-4 |

Продолжение таблицы 2.10

|  |  |
| --- | --- |
| **Имя варианта использования** | ***UC*-7 Управлять товарами и складами** |
| Краткое описание варианта  Использования | Данный вариант использования описывает процесс управления товарами |
| Действующие лица, взаимодействующие с вариантом использования | Менеджер |
| Предусловия для выполнения  варианта использования | Отсутствует |
| Основной поток действий при  исполнении варианта использования | Данный вариант использования начинает выполняться, когда пользователь решает управлять товарами. Пользователь имеет возможность: добавлять товары, редактировать товары, удалять товары, управлять ценами товаров. Товары на склад добавляются из списка существующих товаров. |
| Альтернативный поток действий при исполнении варианта  Использования | Отсутствует |
| Исключения при выполнении  варианта использования | Отсутствуют |
| Постусловия после выполнения  варианта использования. | Отсутствуют |
| Связь с другими вариантами использования | Отсутствует |
| **Имя варианта использования** | ***UC*-8 Просмотреть каталог товаров** |
| Краткое описание варианта  использования | Данный вариант использования описывает просмотр каталога товаров |
| Действующие лица, взаимодействующие с вариантом использования | Клиент, |
| Предусловия для выполнения  варианта использования | Пользователь должен быть авторизован как клиент или компания |

Продолжение таблицы 2.10

|  |  |
| --- | --- |
| **Имя варианта использования** | ***UC*-8 Просмотреть каталог товаров** |
| Основной поток действий при исполнении варианта использования | Данный вариант использования начинает выполняться, когда пользователь решает посмотреть каталог товаров. |
| Альтернативный поток действий при исполнении варианта  использования | Отсутствуют |

|  |  |
| --- | --- |
| Исключения при выполнении  варианта использования | Отсутствуют |
| Постусловия после выполнения  варианта использования. | Отсутствуют |
| Связь с другими вариантами использования | *UC-9, UC-10* |
| **Имя варианта использования** | ***UC*-9 Добавить товар в заказ** |
| Краткое описание варианта  использования | Данный вариант использования описывает добавление товара в заявку |
| Действующие лица, взаимодействующие с вариантом использования | Клиент |
| Предусловия для выполнения  варианта использования | Отсутствуют |
| Основной поток действий при  исполнении варианта использования | Данный вариант использования начинает выполняться, когда пользователь решает добавить товар в заявку. |
| Альтернативный поток действий при исполнении варианта  использования | Отсутствуют |
| Исключения при выполнении  варианта использования | Отсутствуют |
| Постусловия после выполнения  варианта использования. | Товар добавляется в заказ |
| Связь с другими вариантами использования | *UC*-8 |

Продолжение таблицы 2.10

|  |  |
| --- | --- |
| **Имя варианта использования** | ***UC*-9 Оформить заявку на покупку товаров** |
| Краткое описание варианта  Использования | Данный вариант использования описывает процесс оформление заявки на покупку товаров |
| Действующие лица, взаимодействующие с вариантом использования | Клиент |
| Предусловия для выполнения  варианта использования | Пользователь должен добавить хотя бы 1 товар в заявку. |
| Основной поток действий при  исполнении варианта использования | Данный вариант использования начинает выполняться, когда пользователь решает оформить заявку на покупку товаров.  Для оформления заявки необходимо заполнить контактную информацию,  информацию о доставке, типе оплаты. |
| Альтернативный поток действий при исполнении варианта  Использования | Отсутствуют |
| Исключения при выполнении  варианта использования | Вывод сообщения о том, что пользователь должен добавить хотя бы 1 товар в заявку. |
| Постусловия после выполнения  варианта использования. | При успешном выполнении варианта использования заявка на покупку товаров добавляется в базу данных пользовательских заявок |
| Связь с другими вариантами использования | *UC*-8 |

Рассмотрим функциональные требования к каждой роли пользователя. Пользователь, авторизированный как клиент, может оформить заказ любых товаров, представленных в каталоге. После оформления заказа, подтверждения ее менеджером, сборки заказа и доставки товаров, пользователь получает свой заказ.

В таблице 2.11 описаны функциональные требования оформления заказов.

Таблица 2.11 – Функциональные требования оформления заказов

|  |  |
| --- | --- |
| Функция | Описание |
| Оформление заказа.  Выбор товаров  .Товары  .Количество | Выбор товаров в каталоге  Пользователь должен выбрать необходимые ему товары, представленный в каталоге.  Пользователь должен выбрать количество данного товара, которое он хочет приобрести. |
| Оформление заказа.  Оформление  .Заполнение данных о оформлении | Оформление заказа клиентом  Пользователь должен заполнить все предложенные поля, для оформления заказа:   * Клиент: заполняет ФИО, номер телефона, адрес и время доставки, тип оплаты; * Компания: заполняет ФИО и номер телефона сотрудника, который делает заказ, информацию о компании, тип оплаты – банковский перевод, реквизиты доступны на странице оформления. |
| .Отправка | По завершению заполнения всех полей, пользователь нажимает на кнопку «Оформить заказ» и данные о заказе отправляются менеджеру. |

Продолжение таблицы 2.11

|  |  |
| --- | --- |
| Функция | Описание |
| Оформление заказа.  Завершение  .Управление оформлением  .Статус заказа | Получение товара  Пользователю предоставляется возможность просматривать свои заказы, отслеживать их состояние  Менеджер обрабатывает данные о заказе, подтверждает, отклоняет или откладывает его. При подтверждении заказа, он собирается и готовится к доставке. При доставке клиента и оплате клиента, менеджер подтверждает продажу. |

Пользователь, авторизированный как менеджер, может управлять заказами клиентов. Менеджер выбирает склад для реализации заказа, если товаров хватает, переводит заказ в работу, иначе откладывает или отменяет его. В таблице 2.12 описаны функциональные требования управления заказами.

Таблица 2.12 – Функциональные требования управления заказами

|  |  |
| --- | --- |
| Функция | Описание |
| Просмотр заказов.  Выбор заказа  .Заказ | Просмотр и выбор заказа  Менеджер должен выбрать необходимый заказ для работы с ним. |
| Управление заказом.  Управление  .Выбор склада  .Подтверждение | Оформление заказа клиентом  Менеджеры просматривает товары заказа пользователя, выбирает необходимый склад, для реализации заказа. Если товаров не хватает, менеджер может отложить или отменить заказ. Если товаров хватает, менеджер подтверждает заказ, товары списываются со склада, заказа готовится к отправке  Если товар доставлен и клиент оплатил заказ, менеджер подтверждает продажу, продажа сохраняется в базу данных. |

Пользователь, авторизированный как менеджер, имеет возможность управления складами и товарами. Пользователь может добавлять, удалять, создавать новые и редактировать склады и товары. Так же пользователь может управлять товарами на складе, добавлять новые товары из системы, менять количество уже существующих. В таблице 2.13 описаны функциональные требования управления складами и товарами.

Таблица 2.13 – Функциональные требования управления складами и товарами

|  |  |
| --- | --- |
| Функция | Описание |
| Управление товарами.  Просмотр  .Создание  .Редактирование  .Удаление | Просмотр существующих товаров в системе.  Пользователь может создать и добавить новый товар в систему.  Пользователь может редактировать товар.  Пользователь может удалить товар, если он не находится ни на одном складу. |
| Управление складами.  Просмотр  .Склад | Просмотр информации о складах в таблица информации о складах  Пользователь должен выбрать необходимый склад для работы с ним |
| Управление складом.  Управление  .Управление товарами  .Редактирование склада  .Удаление склада | Управление складом  Пользователь должен управлять товарами склада: добавлять товары, удалять товары, редактировать их количество.  Пользователь может редактировать информацию и складе: название, адрес, тип склада.  Если склад пустой, пользователь может удалить его из системы. |

Пользователь, авторизированный как менеджер, имеет возможность просматривать аналитику заказов и продаж и генерировать аналитические отчеты. В таблице 2.13 описаны функциональные требования аналитики продаж.

Таблица 2.13 – Функциональные требования аналитики заказов и продаж

|  |  |
| --- | --- |
| Функция | Описание |
| Аналитика продаж  .Просмотр  .Выбор периода продаж  .Выбор типа аналитики  .Просмотр истории продаж  .Генерация отчета по продажам  .Генерация отчета по заказам  .Генерация отчета ключевых показателей | Просмотр ключевых показателей розничного магазина, аналитики продаж, аналитики заказов  Менеджер может выбрать период для анализа  Менеджер может выбрать тип отображаемой аналитики: по продажам, по заказам, по ключевым показателям.  Менеджер может посмотреть подробную информацию о каждой продаже  Менеджер может сгенерировать отчет по продажам  Менеджер может сгенерировать отчет по заказам  Менеджер может сгенерировать отчет ключевых показателей |

Таким образом были разработаны требования для программного средства.

Далее перейдем к описанию бизнес-контекста. В этом разделе необходимо подробнее представить профили основных категорий заинтересованных в проекте лиц.

В таблице 2.14 ниже рассмотрим лица, которые активно вовлечены в проект, на которых активно влияет результат проекта и которые сами могут влиять на проект.

Таблица 2.14 – Профили заинтересованных лиц

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Заинтересованное лицо | Основная ценность | Отношение | Основные интересы | Ограничения |
| Клиент | Удобство использование для поиска товаров и оформления заказов | Позитивные ожидания от использования интернет-магазина | Простота использования, быстрота процесса оформления заказа, удобный поиск | Необходима регистрация в системе |

Продолжение таблицы 2.14

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Заинтересованное лицо | Основная ценность | Отношение | Основные интересы | Ограничения |
| Менеджер | Более эффективное использование рабочего времени для аналитики заказов и продаж, ответа на заявки клиентов, управлением товарами и складами магазина | Позитивные ожидания от использования системы | Простота использования, простая и понятная система аналитики и отчетов, быстрый поиск и ответ на заявки клиентов | Необходима регистрация в системе |

Нефункциональные требования – это ограничения или требования, предъявляемые к системе. Они определяют атрибут качества программного обеспечения. Нефункциональные требования касаются таких вопросов, как масштабируемость, производительность, переносимость, безопасность, надежность и многие другие.

Были выявлены следующие нефункциональные требования:

* программное средство должно быть разработано для операционной системы *Windows* 8 и выше с возможной предустановкой библиотек или пакетов выбранной среды программирования;
* программное средство должно состоять из клиентского и серверного компонентов, расположенных на разных физических узлах;
* к программному средству должна быть подключена база данных;
* программное средство должно поддерживаться браузерами *Yandex, Firefox, Chrome* либо *Opera* актуальной версии;
* система должна быть способна сохранять работоспособность и восстанавливаться после сбоев;
* время загрузки сайта – 3 секунды;
* при потере соединения все введенные данные должны быть сохранены;
* пользователи всегда должны иметь доступ к системе, даже в ночное время;
* риски от внешних атак должны быть максимально снижены;
* данные о клиентах не должны распространяться публично, то есть должна быть организована работа с приватными данными.

Таким образом, разработка спецификации требований к программному средству является важным этапом проектирования, определяющим функциональные, нефункциональные и бизнес-требования к онлайн-сервису. В рамках данного процесса были сформулированы ключевые бизнес-цели, критерии успеха и возможные риски, которые необходимо учитывать при создании продукта. Спецификация требований обеспечивает четкое понимание задач, которые должен решать сервис, а также ожидаемые результаты его внедрения. Определенные требования позволят разработчикам создать удобный, надежный и востребованный инструмент для проверки научных статей на заимствования. Четкое соответствие спецификации обеспечит высокий уровень точности анализа, удобную интеграцию с образовательными платформами и защиту пользовательских данных.

**2.3 Образ предлагаемого решения**

Предлагаемое программное средство анализа заказов и продаж в розничном магазине представляет собой мощный инструмент для автоматизации процессов учета и анализа данных о продажах и заказах, а также управления ими. Система предназначена для повышения эффективности работы розничных магазинов, оптимизации бизнес-процессов и улучшения взаимодействия с клиентами, что в свою очередь способствует росту прибыли и повышению конкурентоспособности предприятия.

Основной функционал системы заключается в детализированном анализе данных о продажах и заказах. Вся информация о товарах, продажах и заказах собирается в реальном времени, что позволяет оперативно реагировать на изменения и исключает возможность ошибок, связанных с ручным вводом данных.

Кроме того, программное средство автоматизирует процесс учета товаров, остатков на складе, а также генерирует отчеты по продажам и заказам. Это значительно ускоряет процессы учета и анализа данных, повышает точность отчетности и исключает человеческий фактор. Программа также позволяет управлять заявками клиентов, контролировать их статус и оперативно информировать покупателей о ходе выполнения их заказов.

Система имеет функции прогнозирования спроса на основе исторических данных, что позволяет более точно планировать закупки и минимизировать риски излишков или дефицита товаров. Это особенно важно для розничных магазинов, где постоянное поддержание баланса между спросом и предложением является залогом успешной работы. Функциональная архитектуру решения представлена на рисунке 2.8.



Рисунок 2.8 – Функциональная архитектуру решения

Архитектура системы построена по принципу клиент-сервер, где серверная часть обрабатывает и хранит данные, а клиентская часть предоставляет пользователю удобный интерфейс для работы с программным обеспечением. Обмен данными между клиентом и сервером осуществляется через *RESTful API*, что гарантирует высокую скорость работы и масштабируемость системы. Серверная часть будет реализована с использованием современных технологий, с применением фреймворков для эффективной обработки запросов и работы с базами данных. В свою очередь, клиентская часть будет представлена в виде веб-приложения, что обеспечит доступ к системе с любого устройства, подключенного к интернету.

Преимущества предлагаемого решения заключаются в значительном сокращении времени на обработку заказов и данных о продажах, повышении точности прогнозирования и анализа, а также в улучшении взаимодействия с клиентами. Благодаря автоматизации, программа будет способствовать более эффективной организации работы розничного магазина, улучшению качества обслуживания и увеличению прибыли.

1. **ПРОЕКТИРОВАНИЕ И РАЗРАБОТКА ПРОГРАММНОГО СРЕДСТВА**

**3.1 Архитектурные решения и технологии реализации программного средства**

Рассмотрим диаграмму C4, которая визуализирует архитектуру системы на разных уровнях детализации – от общего контекста до внутренней реализации компонентов. Эта диаграмма поможет:

* понять масштаб системы и ее место в экосистеме;
* определить ответственность каждого компонента;
* выявить критические зависимости между частями системы.

Рассмотрим контейнерный уровень диаграммы, представленный на рисунке 3.1.

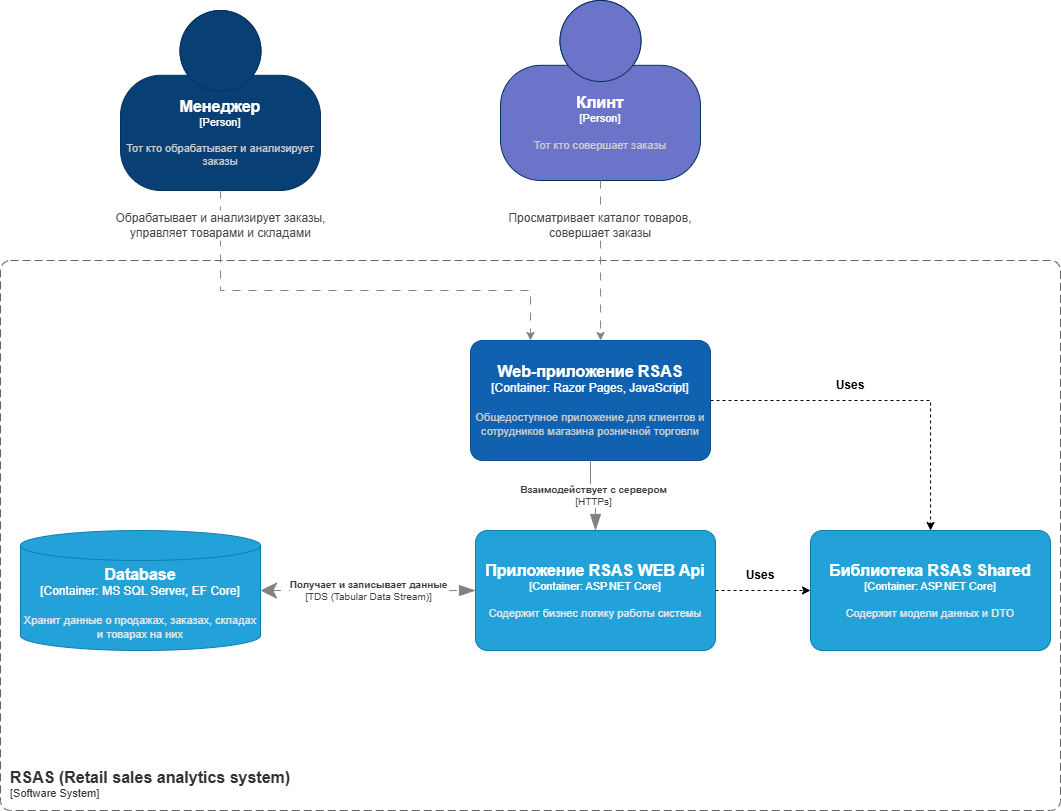


Рисунок 3.1 – Диаграмма С4 на контейнерном уровне

Он отражает ключевые компоненты системы и их взаимодействие.

Основные контейнеры:

1. Web-приложение: технологии: *Asp.Net Core, Razor Pages, JavaScript, Html, Css*; роль: интерфейс для взаимодействия с системой для менеджера и клиента; взаимодействие: обмен данными с *WEB Api* посредству *HTTPs* запросов.
2. Приложение WEB Api: технологии: *Asp.Net Core, Entity Framework Core, Identity*; роль: ядро системы, реализует бизнес логику, взаимодействует с ВД, представляет *Api* для взаимодействия; взаимодействие: взаимодействует с базой данных.
3. Библиотека *Shared*: технологии: *Asp.Net Core;* роль: содержит модели данных и *DTOs,* на которые ссылается *WEB Api и WEB* приложение*.*
4. База данных: технологии: *MS SQL Server;* роль: хранение данных; взаимодействие: используется серверной частью для записи/чтения данных

Теперь рассмотрим диаграмму С4 на компонентом уровне (рисунок 3.2).

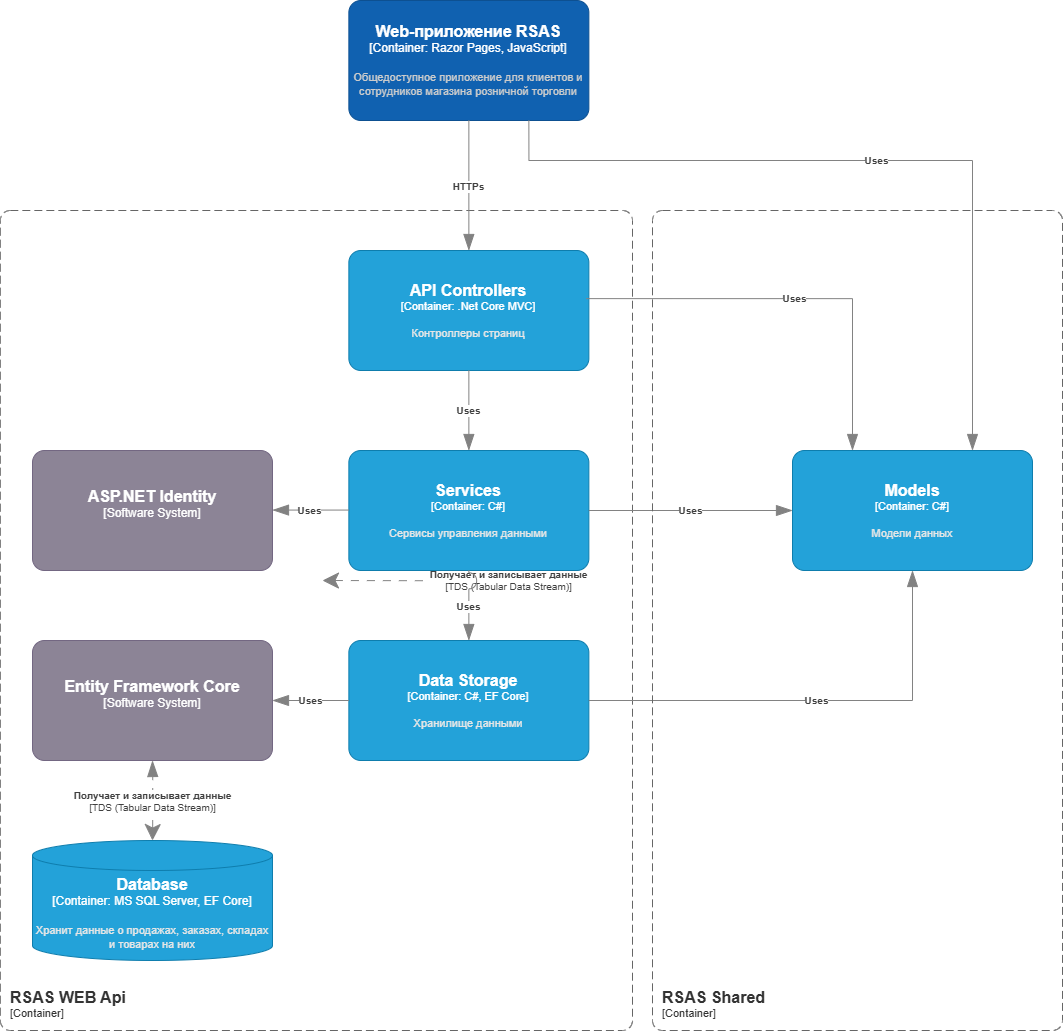


Рисунок 3.2 – Диаграмма С4 на компонентном уровне

На этом уровне детализируется внутренняя структура контейнеров, выделяя ключевые компоненты системы и их взаимодействие. Диаграмма показывает, как организована логика внутри каждого контейнера.

Основные компоненты системы:

1. *API Controller*: Контроллеры предоставляющие *API* для взаимодействия с системой.
2. *Services*: Сервисы которые предоставляют простой интерфейс работы с данными для контроллеров.
3. *Models*: Модели данных системы.
4. *Data Storage*: Прослойка для управления базой данных.

Кроме того, *RSAS* Backend использует следующие внешние системы:

1. Entity Framework Core: ORM-инструмент доступа к данным БД
2. ASP.Net Identity: Система аутентификации и авторизации

Основным архитектурным решением является клиент-серверная архитектура. Она состоит из 3 основных компонентов: сервер, клиент и база данных. В свою очередь сервер состоит из 3 слоев: контроллеры, сервисы, хранилища данных (рисунок 3.3).

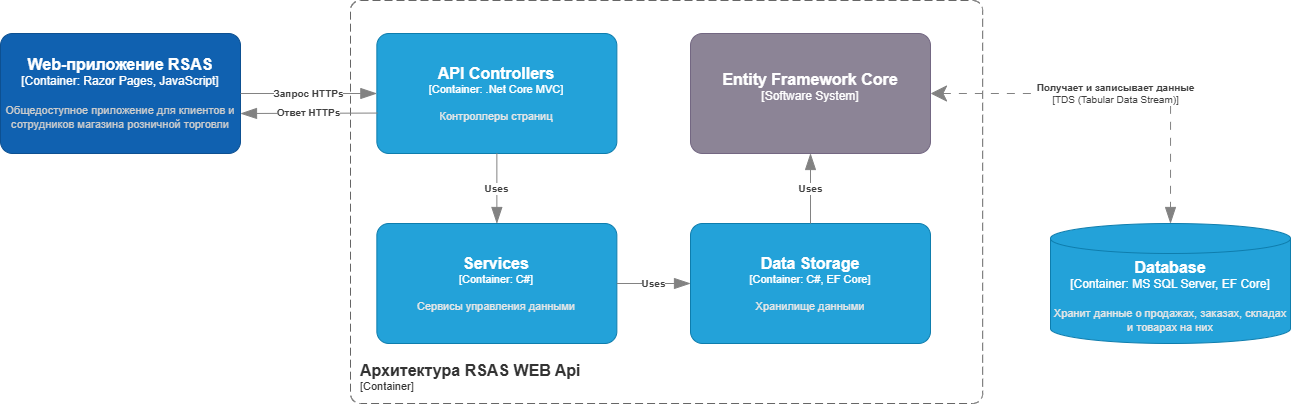


Рисунок 3.3 – Клиент-серверная архитектур

Таким образом, *C4*-модель предоставляет структурированный подход к описанию архитектуры программного средства. В её рамках можно последовательно исследовать систему, начиная с общего контекста и заканчивая компонентами, что помогает разработчикам легко понять ключевые аспекты архитектуры. Рассматриваемая архитектура системы включает четкое разделение логики на веб-часть, серверную часть и взаимодействие с базой данных, что позволяет системе быть гибкой и масштабируемой.

Для разрабатываемого программного средства была выбрана клиент-серверная архитектура. Исходя из бизнес-требования программное средство должно быть веб-приложение. Это позволит клиентам и сотрудникам пользоваться системой в любом месте и в любое время. Необходимо провести сравнительный анализ языков программирования для написания программы.

Самые распространенные языки для разработки веб-приложений – это *JavaScript*, *C#* и *Python*. Соответственно, необходимо произвести сравнительный анализ языков и выбрать тот, который лучше всего подойдет для данной разработки. Сравнительный анализ языков для разработки серверной части программного средства представлен в таблице 3.1.

Таблица 3.1 – Сравнительный анализ языков для разработки веб-приложений

|  |  |  |  |
| --- | --- | --- | --- |
| Характеристика | Язык | | |
| *C#* | *JavaScript* | *Python* |
| Типизация | Статическая | Слабая | Динамическая |
| Использования на сервере | Да | Да | Да |
| Использования на клиенте | Да | Да | Да |
| Фреймворки/  Библиотеки | *.NET, ASP.NET, Entity Framework, LINQ* | *React, Angular, Vue.js, jQuery* | *Django, Flask, Pyramid, Tornado* |
| Кроссплатформенность | Да (с помощью *.NET Core*) | Да | Да |
| Поддержка ООП | Сильная | Слабая | Слабая |
| Уровень сложности | Средний | Низкий | Низкий |
| Скорость выполнения | Высокая | Низкая | Низкая |
| Безопасность | Высокая | Средняя | Средняя |

Как видно из таблицы, *C#* имеет ряд преимуществ перед *JavaScript* и *Python* в области веб-разработки. Он имеет статическую типизацию, поддержку ООП, высокую скорость выполнения и высокий уровень безопасности. Кроме того, *C*# имеет широкий набор фреймворков и библиотек, таких как *.NET, ASP.NET*, *Entity Framework* и *LINQ*, которые позволяют разрабатывать масштабируемые веб-приложения. Также *C#* является кроссплатформенным языком программирования благодаря *.NET Core*. [8]

В качестве платформы для разработки серверной части на языке *C#* была выбран фреймворк *ASP.NET*, который является одним из самых популярных фреймворков для разработки веб-приложений на языке *C#*. Он предоставляет множество инструментов и возможностей для создания качественных и масштабируемых приложений. [9]

Ниже перечислены некоторые преимущества использования *ASP.NET* для разработки веб-приложений на языке *C#*:

1. Безопасность: *ASP.NET Core* обладает высоким уровнем безопасности благодаря встроенным механизмам аутентификации и авторизации, а также возможностям для защиты от атак вроде *SQL*-инъекций.
2. Быстродействие: *ASP.NET Core* использует *JIT*-компиляцию, что позволяет повысить скорость выполнения кода и снизить время отклика сервера на запросы клиентов.
3. Удобство разработки: *ASP.NET Core* предоставляет инструменты для создания многопоточных приложений и возможности для легкой интеграции с другими технологиями, такими как *JavaScript* и *HTML*.
4. Поддержка для множества платформ: *ASP.NET Core* позволяет разрабатывать кросс-платформенные приложения, что облегчает их развертывание на различных операционных системах.
5. Обширная документация и сообщество: *ASP.NET Core* имеет обширную документацию и активное сообщество, что облегчает изучение и решение проблем в процессе разработки.
6. Масштабируемость: *ASP.NET Core* обладает возможностью масштабирования на уровне кода и на уровне инфраструктуры, что позволяет разрабатывать приложения, которые могут масштабироваться в зависимости от потребностей бизнеса.

В целом, использование *ASP.NET Core* для разработки веб-приложений на языке *C#* позволяет создавать безопасные, масштабируемые и быстродействующие приложения с помощью множества инструментов и возможностей, что делает его отличным выбором для создания веб-приложений на языке *C#*.

Из-за тесной совместимости с *ASP.NET Core* в качестве технологии для разработки клиентской части программного средства была выбрана технология *Razor Pages*. Это современный и удобный инструмент для разработки клиентской части веб-приложений на языке *C#*. Вот несколько преимуществ, которые делают *Razor Pages* хорошим выбором для разработки клиентской части:

1. Удобство и простота: *Razor Pages* предоставляет удобный и простой подход к разработке клиентской части, что делает его доступным даже для начинающих разработчиков.
2. Чистый и структурированный код: *Razor Pages* позволяет создавать чистый и структурированный код благодаря ясному разделению логики и представлений.
3. Возможности рендеринга: *Razor Pages* предоставляет возможности для рендеринга *HTML*, что делает его удобным инструментом для создания клиентской части веб-приложений.
4. Поддержка модели: *Razor Pages* позволяет создавать модели, которые представляют данные, и использовать их для связывания данных между клиентской и серверной частями приложения.
5. Поддержка библиотек *JavaScript*: *Razor Pages* позволяет использовать библиотеки *JavaScript*, такие как *jQuery* и *Angular*, что делает его более гибким и мощным инструментом для разработки клиентской части.
6. Поддержка маршрутизации: *Razor Pages* поддерживает маршрутизацию, что позволяет легко управлять навигацией в приложении и обрабатывать запросы клиента.
7. Поддержка валидации данных: *Razor Pages* позволяет легко реализовывать валидацию данных на клиентской стороне, что улучшает пользовательский интерфейс и повышает надежность приложения.
8. Поддержка тестирования: *Razor Pages* позволяет легко тестировать клиентскую часть приложения, что обеспечивает высокое качество кода и надежность приложения.

В целом, *Razor Pages* предоставляет множество преимуществ для разработки клиентской части веб-приложений, что делает его привлекательным выбором для разработчиков, которые хотят создать мощные и гибкие приложения с чистым и структурированным кодом.

Для разработки программного средства необходимо выбрать систему управления базами данных. Реляционная база данных идеально подходит для хранения структурированных данных. Также поддерживают разрешения на доступ, определяющие, кто может читать и редактировать данные.

В *ASP.NET* имеется множество вариантов для работы с базами данных, однако из-за тесной интеграции с *ASP.NET Core* использование *MS SQL Server* и *Entity Framework Core (EF Core)* имеет ряд преимуществ.

Преимущества *MS SQL Server* и *Entity Framework Core(EF Core):*

1. Удобство использования: Оба инструмента имеют понятный и удобный интерфейс, что облегчает их использование даже для начинающих разработчиков.
2. Безопасность: *MS SQL* обладает мощными средствами защиты, что делает его безопасным выбором для хранения конфиденциальной информации. *EF Core* предоставляет возможность использования параметризованных запросов, что предотвращает атаки типа *SQL*-инъекций.
3. Масштабируемость: *MS SQL и EF Core* обладают возможностями для масштабирования приложений, что позволяет создавать приложения, способные работать с большими объемами данных и многопользовательскими сценариями.
4. Эффективность: *MS SQL и EF Core* обеспечивают эффективное выполнение запросов, что обеспечивает высокую скорость работы приложения.
5. Обширная документация и сообщество: *MS SQL и EF Core* имеют обширную документацию и активное сообщество, что облегчает изучение и решение проблем в процессе разработки.
6. Возможности *ORM*: *EF Core* предоставляет возможности *ORM* (*Object-Relational Mapping*), что облегчает работу с базами данных и ускоряет процесс разработки приложения.
7. Поддержка транзакций: *MS SQL* и *EF Core* обладают возможностью работы с транзакциями, что позволяет обеспечить целостность данных и предотвратить их повреждение в случае сбоя.
8. В целом, использование *MS SQL* и *EF Core в ASP.NET* позволяет создавать безопасные, масштабируемые и эффективные приложения с возможностями *ORM* и поддержкой транзакций, что делает их отличным выбором для работы с базами данных в контексте *ASP.NET*.

На основе анализа предметной области и данных сравнения следует вывод, что оптимальным языком программирования является *C#*. Оптимальной платформой для разработки серверной части программного средства является *ASP.NET Core*. Оптимальной технологией для разработки клиентской части является *Razor Pages*. Оптимальной СУБД является *MS SQL Sever* в паре с фреймворком *Entity Framework Core*.

Диаграмма классов, разработанная для программного средства, представленная отображает типы классов системы и различного рода свя-зи, которые существуют между ними. Она занимает центральное место в проектировании объектно-ориентированной системы. Диаграмма классов представлена на рисунке 3.4.

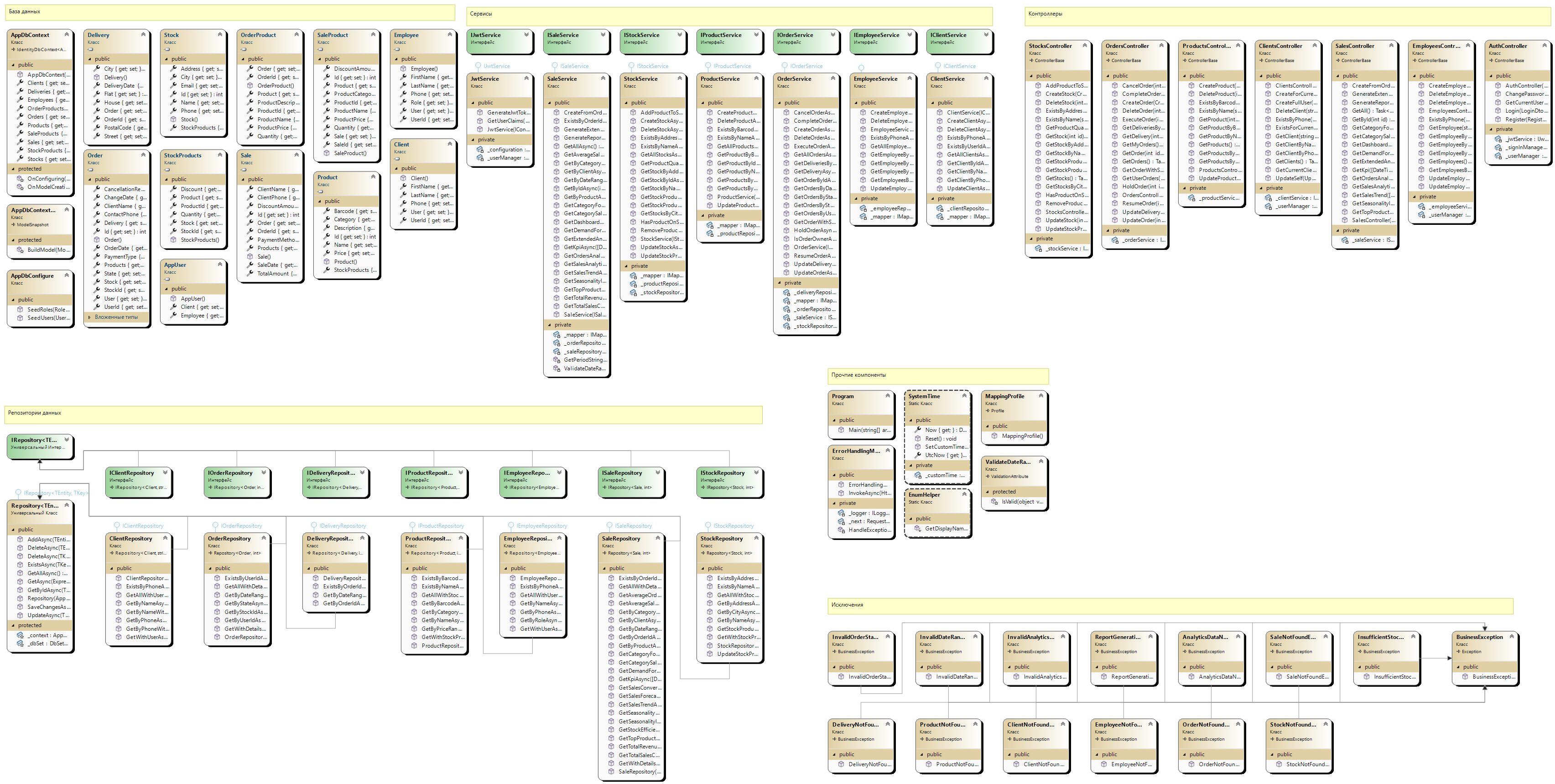


Рисунок 3.4 – Диаграмма классов

Разработанное серверное приложение построено на основе современного фреймворка ASP.NET Core с использованием принципов многослойной архитектуры. Такой подход обеспечивает четкое разделение ответственности между компонентами системы, что значительно упрощает поддержку и дальнейшее развитие проекта.

Архитектура приложения реализована в соответствии с принципами SOLID и состоит из нескольких ключевых слоев. На нижнем уровне располагается слой доступа к данным, который обеспечивает взаимодействие с базой данных через *Entity Framework Core*. Следующим идет слой бизнес-логики, содержащий основные сервисы приложения и обработку бизнес-правил. Верхний слой представлен *REST API* контроллерами, которые обрабатывают *HTTP*-запросы и предоставляют интерфейс для взаимодействия с клиентскими приложениями.

В разработанной системе контроллеры являются точкой входа для всех клиентских запросов. Они реализуют *REST API* интерфейс и отвечают за маршрутизацию запросов к соответствующим сервисам. Каждый контроллер специализируется на определенной функциональной области системы и следует принципу единой ответственности.

Контроллеры обеспечивают валидацию входных данных, управление авторизацией и аутентификацией пользователей, а также форматирование ответов сервера. Для обработки ошибок реализован механизм глобальной обработки исключений, который гарантирует единообразное представление ошибок клиенту.

Особое внимание уделено безопасности *API* - все конечные точки защищены с использованием JWT-токенов и системы ролей, что обеспечивает надежную защиту от несанкционированного доступа.

Сервисный слой является ключевым компонентом приложения, реализующим всю бизнес-логику системы. Каждый сервис представляет собой класс, инкапсулирующий определенную функциональность и взаимодействующий с соответствующими репозиториями для работы с данными.

В основе сервисного слоя лежит принцип инверсии зависимостей, что позволяет легко заменять реализации сервисов и упрощает процесс тестирования. Сервисы реализуют интерфейсы, определяющие их контракты, что обеспечивает слабую связанность компонентов системы.

Слой репозиториев обеспечивает абстракцию над механизмом доступа к данным, скрывая детали работы с базой данных от вышележащих слоев приложения. Каждый репозиторий специализируется на работе с определенной сущностью и предоставляет набор методов для выполнения CRUD-операций.

Репозитории реализуют паттерн Unit of Work, что позволяет управлять транзакциями и обеспечивать целостность данных при выполнении сложных операций. Использование асинхронных методов в репозиториях повышает производительность системы при работе с базой данных.

Для работы с базой данных используется Entity Framework Core - современный ORM-фреймворк, который обеспечивает удобный способ взаимодействия с базой данных через объектную модель. В проекте реализован подход Code First, при котором структура базы данных определяется через классы моделей и их конфигурацию.

Система миграций позволяет отслеживать изменения в схеме базы данных и применять их автоматически. Для повышения производительности реализованы механизмы кэширования и ленивой загрузки данных. Особое внимание уделено оптимизации запросов и индексированию таблиц.

В проекте реализован ряд вспомогательных сервисов, обеспечивающих дополнительную функциональность системы. Среди них особо стоит отметить сервис аутентификации и авторизации, построенный на основе JWT-токенов, сервис логирования для отслеживания работы системы и сервис кэширования для оптимизации производительности.

Отдельного внимания заслуживает система обработки ошибок, которая обеспечивает централизованную обработку исключений и формирование информативных сообщений об ошибках. Реализованный механизм middleware позволяет гибко настраивать обработку HTTP-запросов и внедрять дополнительную функциональность в конвейер обработки запросов.

Все компоненты системы разработаны с учетом возможности горизонтального масштабирования и легкой интеграции новых функциональных возможностей. Модульная архитектура и использование современных паттернов проектирования обеспечивают высокую гибкость и расширяемость системы.

**3.2 Проектирование и разработка пользовательского интерфейса**

При разработке пользовательского интерфейса была использована система дизайна, основанная на фреймворке Bootstrap 5 с темой Quartz. Данный подход обеспечивает единообразие компонентов интерфейса и соответствие современным стандартам веб-дизайна. Основные принципы системы дизайна включают:

* Использование модульной сетки для организации контента.
* Единую цветовую схему с акцентными цветами для важных элементов.
* Типографическую систему с четкой иерархией заголовков.
* Согласованные отступы и интервалы между элементами.
* Унифицированные компоненты интерфейса.

Дизайн схема системы представлена на рисунке 3.5, где показаны основные элементы и цвета, используемые в интерфейсе приложения.

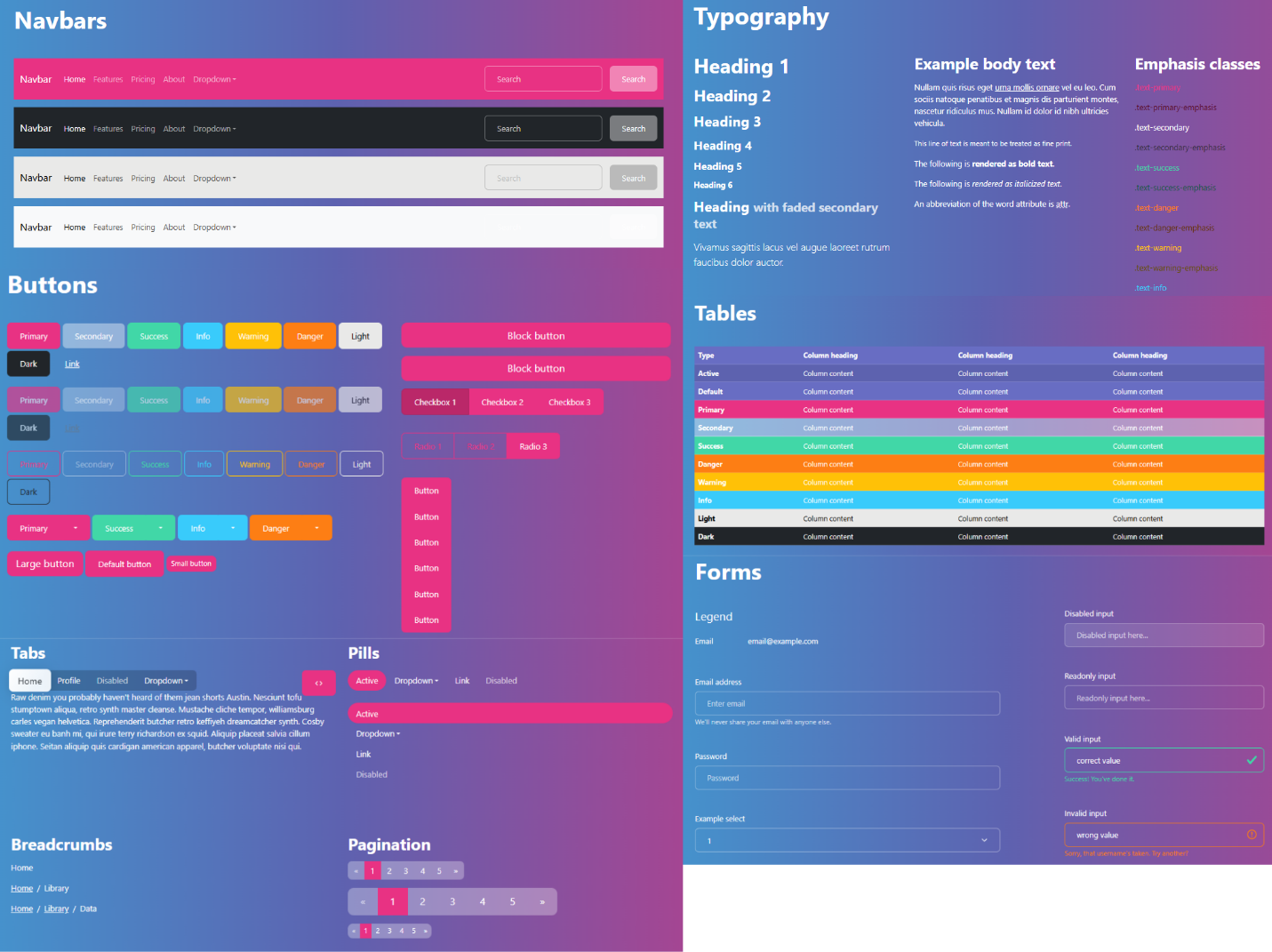


Рисунок 3.5 – Дизайн схема системы

При разработке пользовательского интерфейса использован современный стек технологий:

Базовые технологии:

* ASP.NET Core Razor Pages для серверного рендеринга
* HTML5 для структуры страниц
* CSS3 для стилизации
* JavaScript для клиентской логики

Фреймворки и библиотеки:

* Bootstrap 5 для адаптивной верстки
* jQuery для манипуляций с DOM
* Chart.js для визуализации данных
* Toastr для системы уведомлений

Структура пользовательского интерфейса построена по модульному принципу, где каждый компонент является независимым и переиспользуемым. В системе реализованы следующие ключевые компоненты:

1. Навигационная система:

* Главное меню с разделами
* Подменю для каждого раздела
* Хлебные крошки для навигации
* Быстрые действия

1. Информационные панели:

* Карточки с ключевыми показателями
* Графики и диаграммы
* Таблицы с данными
* Статистические виджеты

1. Формы и элементы ввода**:**

* Текстовые поля с валидацией
* Выпадающие списки
* Календари для выбора дат
* Чекбоксы и радиокнопки

Особое внимание уделено обеспечению качественного пользовательского опыта:

1. Система обратной связи:

* Уведомления о результатах действий
* Индикаторы загрузки
* Подтверждение важных операций

Подсказки и справочная информация

1. Оптимизация производительности:

* Асинхронная загрузка данных
* Кэширование часто используемой информации
* Ленивая загрузка компонентов
* Оптимизация изображений

1. Доступность:

* Семантическая разметка
* Поддержка клавиатурной навигации
* Контрастные цвета

Все компоненты интерфейса разработаны с учетом современных требований к безопасности веб-приложений и регулярно обновляются для поддержания актуального уровня защиты.

В ходе разработки был использован сервис *draw.io* для создания прототипов экранов пользовательского интерфейса. Рассмотрим прототипы основных экранов пользовательского интерфейса программного средства. На рисунке 3.6 представлен прототип экрана аналитики.

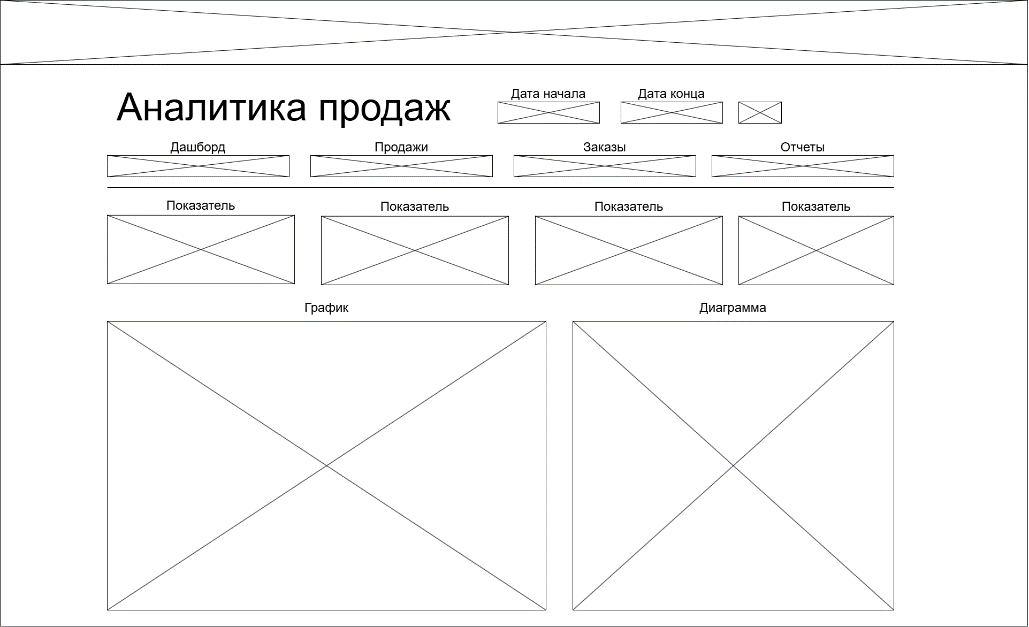


Рисунок 3.6 – Прототип экрана аналитики

На рисунке 3.7 представлен прототип экрана управления товарами.

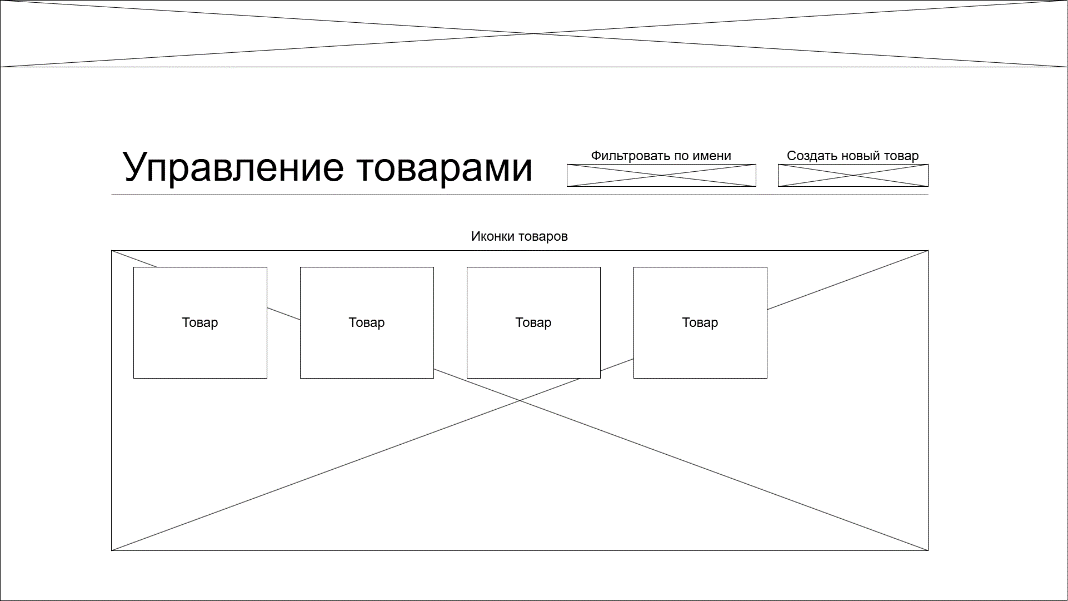


Рисунок 3.7 – Прототип экрана управления товарами

На рисунке 3.8 представлен прототип экрана управления складом.

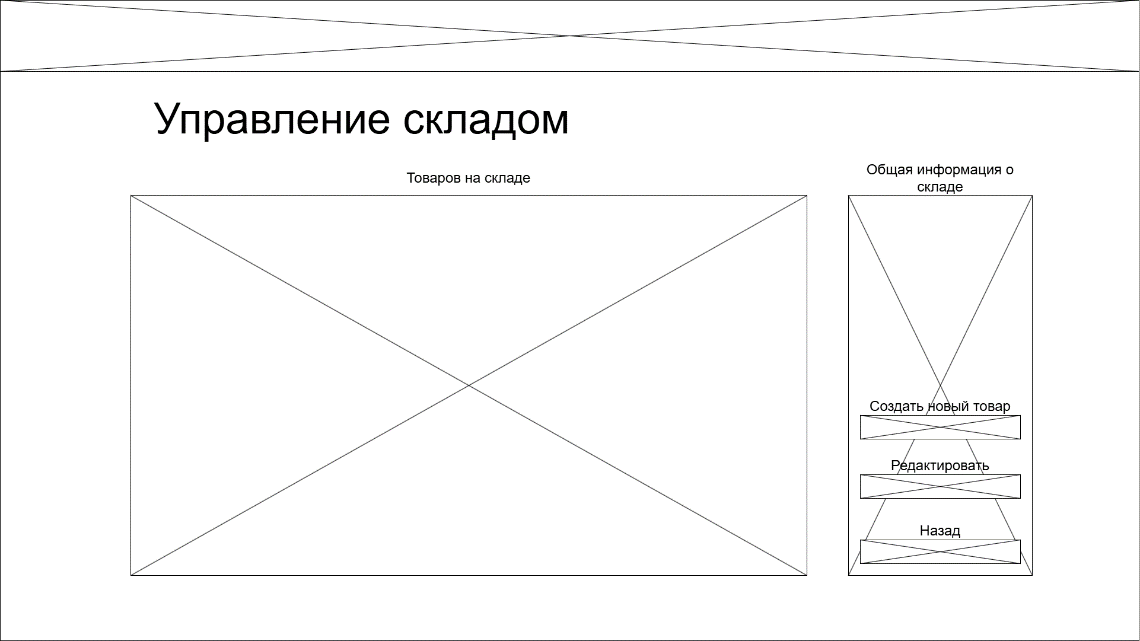


Рисунок 3.8 – Прототип экрана управления складом

На рисунке 3.9 представлен прототип экрана каталога товаров.

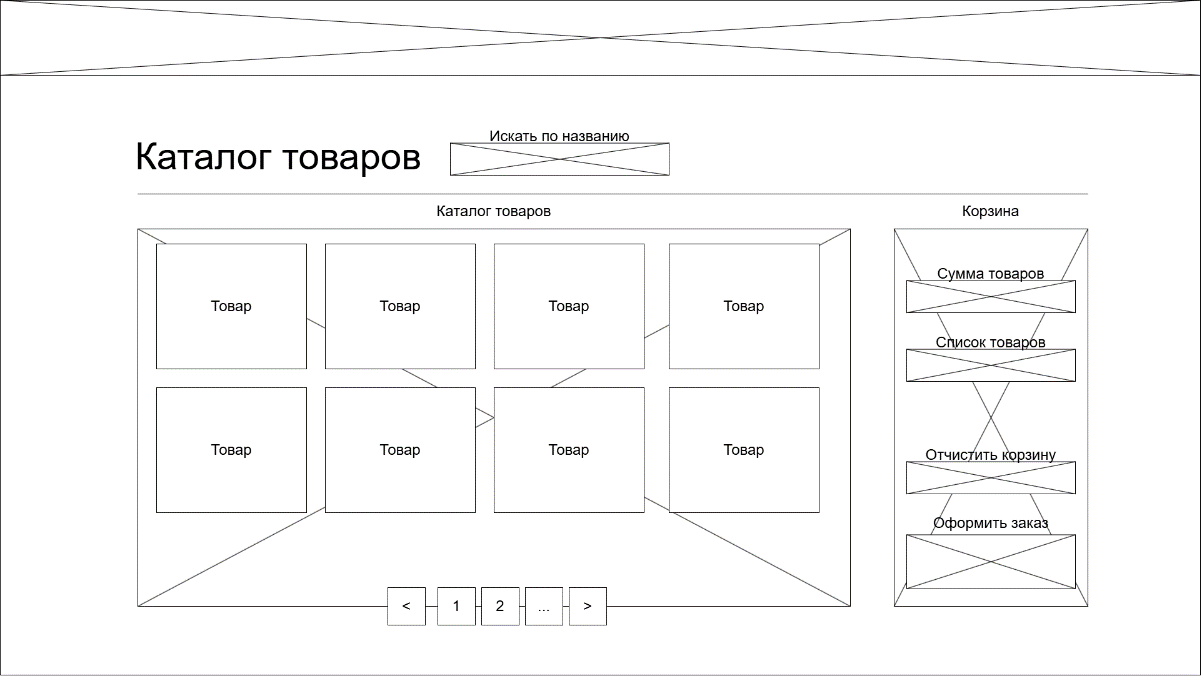


Рисунок 3.9 – Прототип экрана каталога товаров

На этапе проектирования интерфейса программного средства спроектирован весь путь пользователя от начальных экранов и до выполнения целевых действий, которых в программном средстве может быть несколько. Для каждой роли была разработана *user-flow* диаграмма.

Рассмотрим *user-flow* диаграмму для роли клиент на рисунке 3.10.

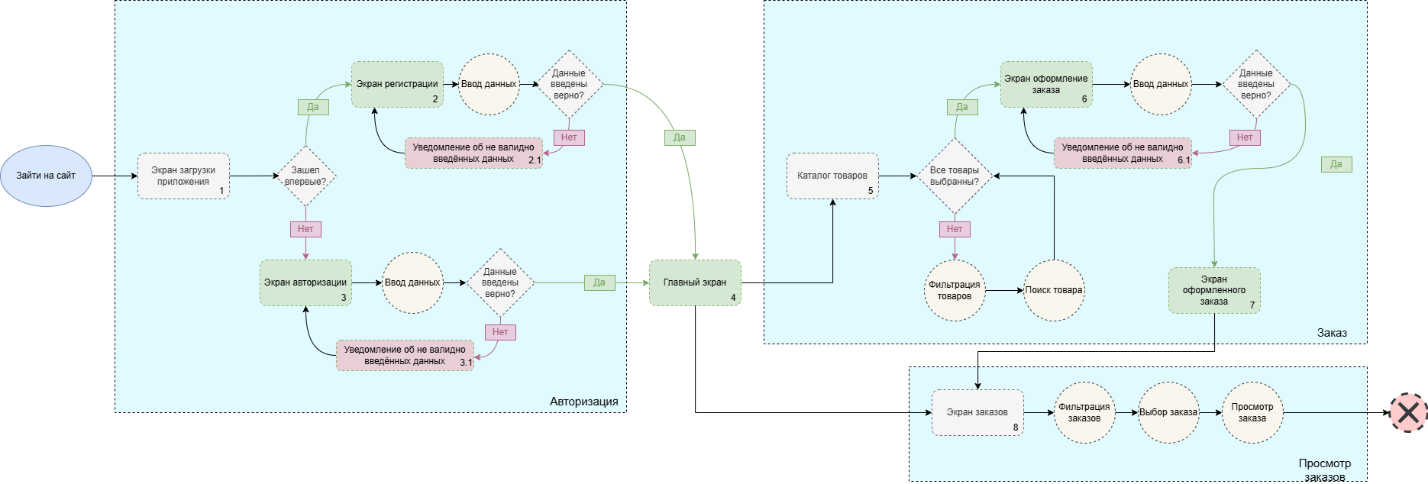


Рисунок 3.10 – *User-flow* диаграмма для роли клиент

Рассмотрим *user-flow* диаграмму для роли менеджер на рисунке 3.11.

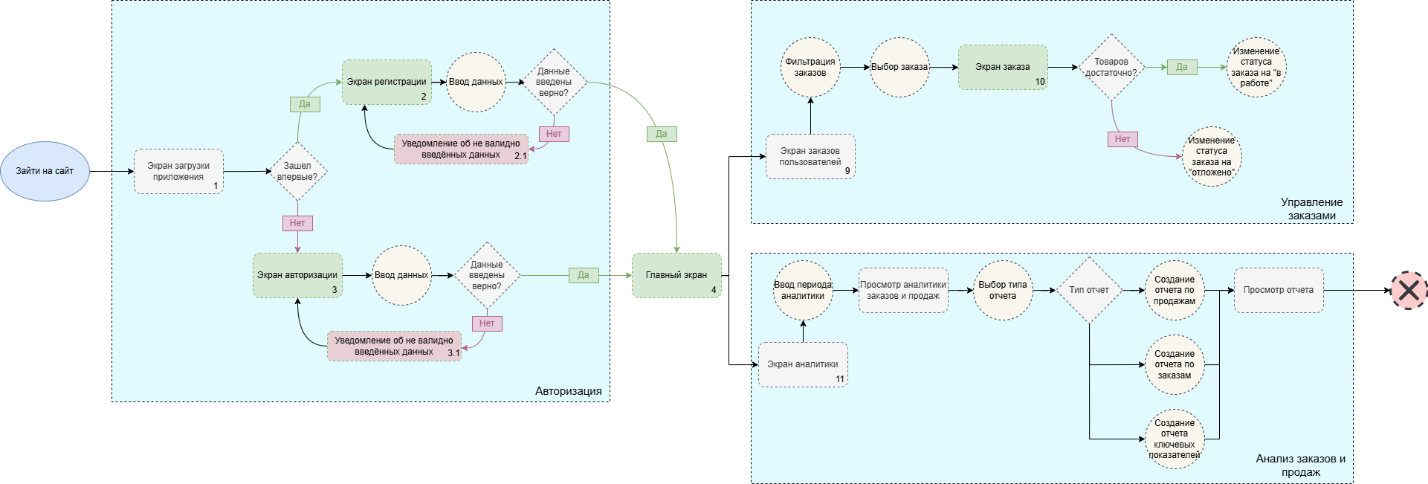


Рисунок 3.11 – *User-flow* диаграмма для роли менеджер

Таким образом, были спроектированы прототипы пользовательского интерфейса разрабатываемого программного средства, а также составлена диаграмма *User Flow* программного средства.

**3.3 Разработка модели данных**

Проектирование информационных систем начинается с создания модели предметной области, которая состоит из связанных между собой объектов с определенными свойствами. Для упрощения модели информации проектируемой системы, разработчики стремятся исключить ошибки при добавлении, изменении или удалении данных, а также избежать дублирования информации.

Целью проектирования является создание читаемой и удобной модели, что облегчает понимание информации и управление ею. Для достижения этой цели применяются процессы нормализации, которые улучшают структуру модели. Каждая следующая нормальная форма содержит свойства предыдущей и дополнительные ограничения, что приводит к постоянному улучшению структуры. В теории реляционных баз данных наиболее часто используются первые три нормальные формы, каждая из которых имеет свой набор ограничений.

Информационная модель системы представляет собой базу данных на основе *MS SQL Server*, управляемую при помощи *Entity Framework Core*. В процессе информационного моделирования были выделены следующие сущности:

* пользователь;
* продукт;
* склад;
* заказ;
* продажа.

База данных должна быть приведена к третей нормальной форме, у каждой таблицы должен иметься всего один первичный ключ, а каждое не ключевое поле не транзитивно должно зависеть от первичного ключа, т.е. изменив значение в одном столбце не потребуется изменение в другом столбце.

Физическая модель базы данных определяет способ размещения данных в среде хранения и способы доступа к этим данным, которые поддерживаются на физическом уровне.

Физическая модель базы данных представлена на рисунке 3.12.

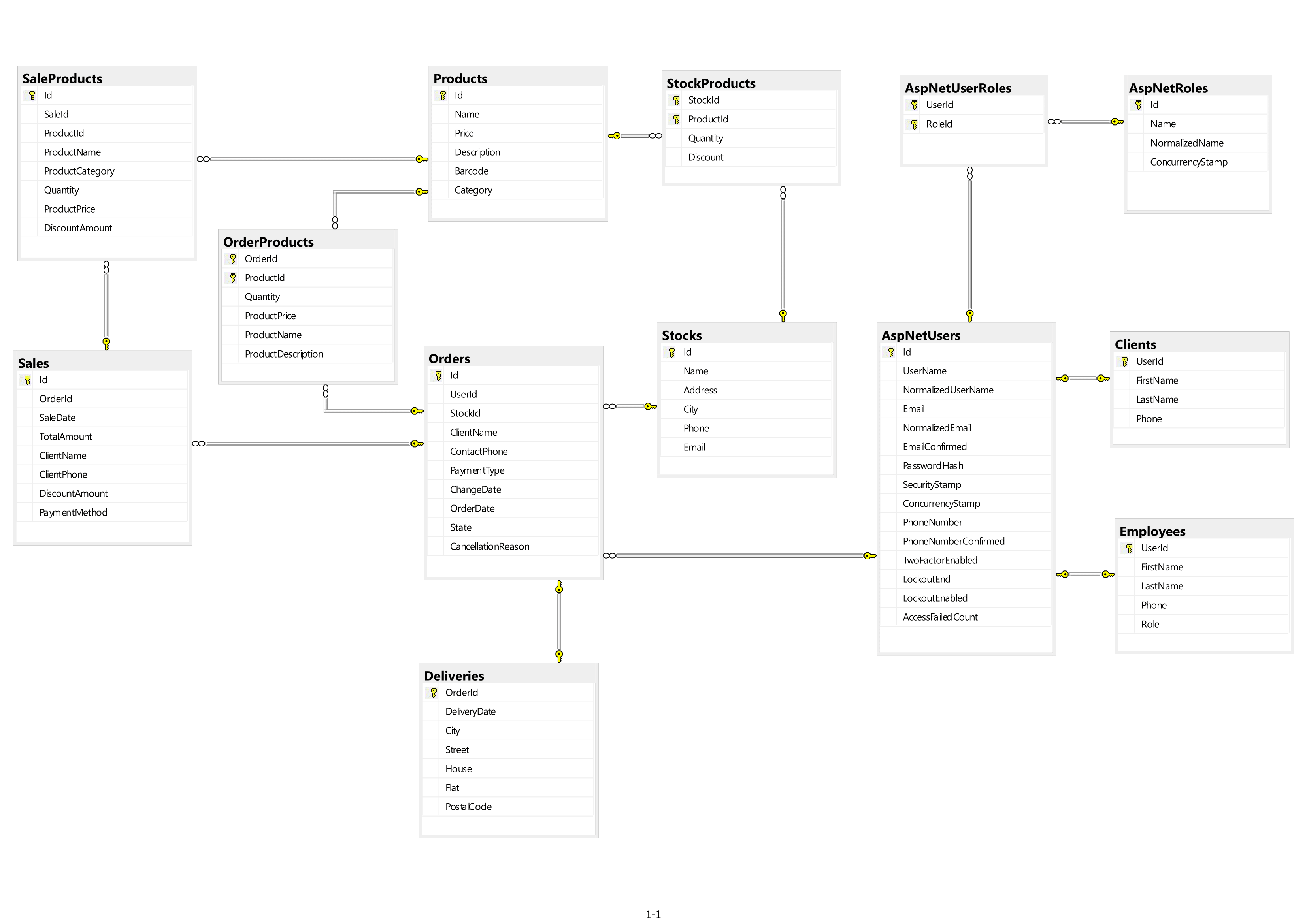


Рисунок 3.12 – Физическая модель базы данных

В таблице 3.2 подробно описаны атрибуты каждой сущности.

Таблица 3.2 – Описание сущностей базы данных

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Назначение  атрибута | Тип данных | Примечание |
| *AspNetUser* (таблица пользователь): | | | |
| *Id* | Идентификатор пользователя | *nvarchar(450)* | Первичный ключ  (GUID) |
| *UserName* | Логин пользователя | *nvarchar(450)* |  |
| *PasswordHash* | Хешированый  пароль пользователя | *nvarchar(MAX)* |  |

Продолжение таблицы 3.2

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Наименование поля | Назначение  атрибута | Тип данных | | Примечание |
| *Email* | Почта пользователя | *nvarchar(256)* | |  |
| *AspNetUserRoles* (промежуточная таблица связи пользователя и роли): | | | | |
| *UserId* | Идентификатор пользователя | *nvarchar(450)* | | Первичный ключ,  Внешний ключ |
| *RoleId* | Идентификатор роли | *nvarchar(450)* | | Первичный ключ,  Внешний ключ |
| *AspNetRoles* (таблица ролей): | | | | |
| *Id* | Идентификатор роли | *nvarchar(450)* | | Первичный ключ  (GUID) |
| *Name* | Название роли | *nvarchar(MAX)* | |  |
| *Clients* (таблица информации о пользователе): | | | | |
| *UserId* | Идентификатор пользователя | *nvarchar(450)* | | Первичный ключ Внешний ключ |
| *FirstName* | Имя | *nvarchar(MAX)* | |  |
| *LastName* | Фамилия | *nvarchar(MAX)* |  | |
| *Phone* | Телефон | *nvarchar(MAX)* |  | |

Продолжение таблицы 3.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Наименование поля | Назначение  атрибута | | | Тип данных | | Примечание |
| *Employees* (таблица информации о сотрудниках): | | | | | | |
| *UserId* | | Идентификатор пользователя | *nvarchar(450)* | | Первичный ключ Внешний ключ | |
| *FirstName* | | Имя | *nvarchar(MAX)* | |  | |
| *LastName* | | Фамилия | *nvarchar(MAX)* | |  | |
| *Phone* | | Телефон | *nvarchar(MAX)* | |  | |
| *Role* | | Должность | *nvarchar(MAX)* | |  | |
| *Products* (таблица товаров): | | | | | | |
| *Id* | | Идентификатор товара | *int32* | | Первичный ключ | |
| *Name* | | Название товара | *nvarchar(MAX)* | |  | |
| *Description* | | Описание товара | *nvarchar(MAX)* | |  | |
| *Barcode* | | Штрих-код товара | *nvarchar(MAX)* | |  | |
| *Category* | | Категория товара | *nvarchar(MAX)* | |  | |
| *RetailPrice* | | Цена | *Decimal* | |  | |

Продолжение таблицы 3.2

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| Наименование поля | Назначение  атрибута | | | Тип данных | | Примечание |
| *Stock* (таблица складов): | | | | | | |
| *Id* | | Идентификатор склада | *int32* | | Первичный ключ | |
| *Name* | | Название склада | *nvarchar(MAX)* | |  | |
| *Address* | | Адрес склада | *nvarchar(MAX)* | |  | |
| *City* | | Город склада | *nvarchar(MAX)* | |  | |
| *Phone* | | Телефон склада | *nvarchar(MAX)* | |  | |
| *StockProducts* (промежуточная таблица связи складов и товаров): | | | | | | |
| *StockId* | | Идентификатор склада | *int32* | | Первичный ключ  Внешний ключ | |
| *ProductId* | | Идентификатор товара | *int32* | | Первичный ключ  Внешний ключ | |
| *Quantity* | | Количество товаров | *int32* | |  | |
| *Discount* | | Скидка на товар | *Decimal* | |  | |
| *Order* (таблица заказов): | | | | | | |
| *Id* | | Идентификатор заказа | *int32* | | Первичный ключ | |
| *UserId* | | Идентификатор пользователя | *nvarchar(MAX)* | | Внешний ключ | |

Продолжение таблицы 3.2

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Назначение  атрибута | *Тип данных* | Примечание |
| *StockId* | Идентификатор склада | *int32* | Внешний ключ |
| *ClientName* | ФИО клиента | *nvarchar(MAX)* |  |
| *ContactPhone* | Контактный телефон | *nvarchar(MAX)* |  |
| *PaymentType* | Тип оплаты | *int32* |  |
| *ChangeDate* | Дата изменения | *datetime2(7)* |  |
| *OrderDate* | Дата заказа | *datetime2(7)* |  |
| *State* | Состояние заказа | *int32* |  |
| *Cancellation*  *Reason* | Причина отмены | *nvarchar(MAX)* |  |
| *OrderProducts* (промежуточная таблица связи заказа и товаров): | | | |
| *OrderId* | Идентификатор заказа | *int32* | Первичный ключ  Внешний ключ |
| *ProductId* | Идентификатор товара | *int32* | Внешний ключ |

Продолжение таблицы 3.2

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Назначение  атрибута | Тип данных | Примечание |
| *Quantity* | Количество товаров | *int32* |  |
| *ProductName* | Название товара | *nvarchar(MAX)* |  |
| *Product*  *Description* | Описание товара | *nvarchar(MAX)* |  |
| *ProductPrice* | Цена товара | *Decimal* |  |
| *Deliveries*(таблица информации о доставке): | | | |
| *OrderId* | Идентификатор  заказа | *int32* | Первичный ключ  Внешний ключ |
| *City* | Город | *nvarchar(MAX)* |  |
| *Street* | Улица | *nvarchar(MAX)* |  |
| *House* | Дом | *nvarchar(MAX)* |  |
| *Flat* | Квартира | *nvarchar(MAX)* |  |
| *PostalCode* | Почтовый индекс | *int32* |  |
| *DeliveryDate* | Дата доставки | *datetime2(7)* |  |
| *Sales*(таблица продаж): | | | |
| *Id* | Идентификатор  продажи | *int32* | Первичный ключ |
| *OrderId* | Идентификатор  заказа | *nvarchar(450)* | Внешний ключ |

Окончание таблицы 3.2

|  |  |  |  |
| --- | --- | --- | --- |
| Наименование поля | Назначение  атрибута | Тип данных | Примечание |
| *SaleDate* | Дата продажи | *datetime2(7)* |  |
| *ClientName* | ФИО клиента | *nvarchar(450)* |  |
| *ClientPhone* | Телефон клиента | *nvarchar(450)* |  |
| *PaymentMethod* | Тип оплаты | *int32* |  |

Рассмотрим скрипт генерации базы данных:

USE [RsasDB]

GO

/\*\*\*\*\*\* Object: Table [dbo].[AspNetRoles] Script Date: 01.05.2025 21:48:46 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AspNetRoles](

[Id] [nvarchar](450) NOT NULL,

[Name] [nvarchar](256) NULL,

[NormalizedName] [nvarchar](256) NULL,

[ConcurrencyStamp] [nvarchar](max) NULL,

CONSTRAINT [PK\_AspNetRoles] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[AspNetUsers] Script Date: 01.05.2025 21:48:46 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[AspNetUsers](

[Id] [nvarchar](450) NOT NULL,

[UserName] [nvarchar](256) NULL,

[NormalizedUserName] [nvarchar](256) NULL,

[Email] [nvarchar](256) NULL,

[NormalizedEmail] [nvarchar](256) NULL,

[EmailConfirmed] [bit] NOT NULL,

[PasswordHash] [nvarchar](max) NULL,

[SecurityStamp] [nvarchar](max) NULL,

[ConcurrencyStamp] [nvarchar](max) NULL,

[PhoneNumber] [nvarchar](max) NULL,

[PhoneNumberConfirmed] [bit] NOT NULL,

[TwoFactorEnabled] [bit] NOT NULL,

[LockoutEnd] [datetimeoffset](7) NULL,

[LockoutEnabled] [bit] NOT NULL,

[AccessFailedCount] [int] NOT NULL,

CONSTRAINT [PK\_AspNetUsers] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Clients] Script Date: 01.05.2025 21:48:46 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Clients](

[UserId] [nvarchar](450) NOT NULL,

[FirstName] [nvarchar](max) NOT NULL,

[LastName] [nvarchar](max) NOT NULL,

[Phone] [nvarchar](max) NOT NULL,

CONSTRAINT [PK\_Clients] PRIMARY KEY CLUSTERED

(

[UserId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Deliveries] Script Date: 01.05.2025 21:48:46 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Deliveries](

[OrderId] [int] NOT NULL,

[DeliveryDate] [datetime2](7) NOT NULL,

[City] [nvarchar](max) NOT NULL,

[Street] [nvarchar](max) NOT NULL,

[House] [nvarchar](max) NOT NULL,

[Flat] [nvarchar](max) NOT NULL,

[PostalCode] [nvarchar](max) NOT NULL,

CONSTRAINT [PK\_Deliveries] PRIMARY KEY CLUSTERED

(

[OrderId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Employees] Script Date: 01.05.2025 21:48:46 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Employees](

[UserId] [nvarchar](450) NOT NULL,

[FirstName] [nvarchar](max) NOT NULL,

[LastName] [nvarchar](max) NOT NULL,

[Phone] [nvarchar](max) NOT NULL,

[Role] [nvarchar](max) NOT NULL,

CONSTRAINT [PK\_Employees] PRIMARY KEY CLUSTERED

(

[UserId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[OrderProducts] Script Date: 01.05.2025 21:48:46 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[OrderProducts](

[OrderId] [int] NOT NULL,

[ProductId] [int] NOT NULL,

[Quantity] [int] NOT NULL,

[ProductPrice] [decimal](18, 2) NOT NULL,

[ProductName] [nvarchar](max) NOT NULL,

[ProductDescription] [nvarchar](max) NOT NULL,

CONSTRAINT [PK\_OrderProducts] PRIMARY KEY CLUSTERED

(

[OrderId] ASC,

[ProductId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Orders] Script Date: 01.05.2025 21:48:46 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Orders](

[Id] [int] IDENTITY(1,1) NOT NULL,

[UserId] [nvarchar](450) NOT NULL,

[StockId] [int] NULL,

[ClientName] [nvarchar](max) NOT NULL,

[ContactPhone] [nvarchar](max) NOT NULL,

[PaymentType] [int] NOT NULL,

[ChangeDate] [datetime2](7) NOT NULL,

[OrderDate] [datetime2](7) NOT NULL,

[State] [int] NOT NULL,

[CancellationReason] [nvarchar](max) NOT NULL,

CONSTRAINT [PK\_Orders] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Products] Script Date: 01.05.2025 21:48:46 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Products](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](max) NOT NULL,

[Price] [decimal](18, 2) NOT NULL,

[Description] [nvarchar](max) NOT NULL,

[Barcode] [nvarchar](max) NOT NULL,

[Category] [nvarchar](max) NOT NULL,

CONSTRAINT [PK\_Products] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[SaleProducts] Script Date: 01.05.2025 21:48:46 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[SaleProducts](

[Id] [int] IDENTITY(1,1) NOT NULL,

[SaleId] [int] NOT NULL,

[ProductId] [int] NOT NULL,

[ProductName] [nvarchar](max) NOT NULL,

[ProductCategory] [nvarchar](max) NOT NULL,

[Quantity] [int] NOT NULL,

[ProductPrice] [decimal](18, 2) NOT NULL,

[DiscountAmount] [decimal](18, 2) NOT NULL,

CONSTRAINT [PK\_SaleProducts] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Sales] Script Date: 01.05.2025 21:48:46 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Sales](

[Id] [int] IDENTITY(1,1) NOT NULL,

[OrderId] [int] NOT NULL,

[SaleDate] [datetime2](7) NOT NULL,

[TotalAmount] [decimal](18, 2) NOT NULL,

[ClientName] [nvarchar](max) NOT NULL,

[ClientPhone] [nvarchar](max) NOT NULL,

[DiscountAmount] [decimal](18, 2) NOT NULL,

[PaymentMethod] [nvarchar](max) NOT NULL,

CONSTRAINT [PK\_Sales] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[StockProducts] Script Date: 01.05.2025 21:48:46 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[StockProducts](

[StockId] [int] NOT NULL,

[ProductId] [int] NOT NULL,

[Quantity] [int] NOT NULL,

[Discount] [decimal](18, 2) NOT NULL,

CONSTRAINT [PK\_StockProducts] PRIMARY KEY CLUSTERED

(

[StockId] ASC,

[ProductId] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY]

GO

/\*\*\*\*\*\* Object: Table [dbo].[Stocks] Script Date: 01.05.2025 21:48:46 \*\*\*\*\*\*/

SET ANSI\_NULLS ON

GO

SET QUOTED\_IDENTIFIER ON

GO

CREATE TABLE [dbo].[Stocks](

[Id] [int] IDENTITY(1,1) NOT NULL,

[Name] [nvarchar](max) NOT NULL,

[Address] [nvarchar](max) NOT NULL,

[City] [nvarchar](max) NOT NULL,

[Phone] [nvarchar](max) NOT NULL,

[Email] [nvarchar](max) NOT NULL,

CONSTRAINT [PK\_Stocks] PRIMARY KEY CLUSTERED

(

[Id] ASC

)WITH (PAD\_INDEX = OFF, STATISTICS\_NORECOMPUTE = OFF, IGNORE\_DUP\_KEY = OFF, ALLOW\_ROW\_LOCKS = ON, ALLOW\_PAGE\_LOCKS = ON, OPTIMIZE\_FOR\_SEQUENTIAL\_KEY = OFF) ON [PRIMARY]

) ON [PRIMARY] TEXTIMAGE\_ON [PRIMARY]

GO

ALTER TABLE [dbo].[Orders] ADD DEFAULT (N'') FOR [CancellationReason]

GO

ALTER TABLE [dbo].[Sales] ADD DEFAULT (N'') FOR [ClientName]

GO

ALTER TABLE [dbo].[Sales] ADD DEFAULT (N'') FOR [ClientPhone]

GO

ALTER TABLE [dbo].[Sales] ADD DEFAULT ((0.0)) FOR [DiscountAmount]

GO

ALTER TABLE [dbo].[Sales] ADD DEFAULT (N'') FOR [PaymentMethod]

GO

ALTER TABLE [dbo].[Clients] WITH CHECK ADD CONSTRAINT [FK\_Clients\_AspNetUsers\_UserId] FOREIGN KEY([UserId])

REFERENCES [dbo].[AspNetUsers] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[Clients] CHECK CONSTRAINT [FK\_Clients\_AspNetUsers\_UserId]

GO

ALTER TABLE [dbo].[Deliveries] WITH CHECK ADD CONSTRAINT [FK\_Deliveries\_Orders\_OrderId] FOREIGN KEY([OrderId])

REFERENCES [dbo].[Orders] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[Deliveries] CHECK CONSTRAINT [FK\_Deliveries\_Orders\_OrderId]

GO

ALTER TABLE [dbo].[Employees] WITH CHECK ADD CONSTRAINT [FK\_Employees\_AspNetUsers\_UserId] FOREIGN KEY([UserId])

REFERENCES [dbo].[AspNetUsers] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[Employees] CHECK CONSTRAINT [FK\_Employees\_AspNetUsers\_UserId]

GO

ALTER TABLE [dbo].[OrderProducts] WITH CHECK ADD CONSTRAINT [FK\_OrderProducts\_Orders\_OrderId] FOREIGN KEY([OrderId])

REFERENCES [dbo].[Orders] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[OrderProducts] CHECK CONSTRAINT [FK\_OrderProducts\_Orders\_OrderId]

GO

ALTER TABLE [dbo].[OrderProducts] WITH CHECK ADD CONSTRAINT [FK\_OrderProducts\_Products\_ProductId] FOREIGN KEY([ProductId])

REFERENCES [dbo].[Products] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[OrderProducts] CHECK CONSTRAINT [FK\_OrderProducts\_Products\_ProductId]

GO

ALTER TABLE [dbo].[Orders] WITH CHECK ADD CONSTRAINT [FK\_Orders\_AspNetUsers\_UserId] FOREIGN KEY([UserId])

REFERENCES [dbo].[AspNetUsers] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[Orders] CHECK CONSTRAINT [FK\_Orders\_AspNetUsers\_UserId]

GO

ALTER TABLE [dbo].[Orders] WITH CHECK ADD CONSTRAINT [FK\_Orders\_Stocks\_StockId] FOREIGN KEY([StockId])

REFERENCES [dbo].[Stocks] ([Id])

GO

ALTER TABLE [dbo].[Orders] CHECK CONSTRAINT [FK\_Orders\_Stocks\_StockId]

GO

ALTER TABLE [dbo].[SaleProducts] WITH CHECK ADD CONSTRAINT [FK\_SaleProducts\_Products\_ProductId] FOREIGN KEY([ProductId])

REFERENCES [dbo].[Products] ([Id])

GO

ALTER TABLE [dbo].[SaleProducts] CHECK CONSTRAINT [FK\_SaleProducts\_Products\_ProductId]

GO

ALTER TABLE [dbo].[SaleProducts] WITH CHECK ADD CONSTRAINT [FK\_SaleProducts\_Sales\_SaleId] FOREIGN KEY([SaleId])

REFERENCES [dbo].[Sales] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[SaleProducts] CHECK CONSTRAINT [FK\_SaleProducts\_Sales\_SaleId]

GO

ALTER TABLE [dbo].[Sales] WITH CHECK ADD CONSTRAINT [FK\_Sales\_Orders\_OrderId] FOREIGN KEY([OrderId])

REFERENCES [dbo].[Orders] ([Id])

GO

ALTER TABLE [dbo].[Sales] CHECK CONSTRAINT [FK\_Sales\_Orders\_OrderId]

GO

ALTER TABLE [dbo].[StockProducts] WITH CHECK ADD CONSTRAINT [FK\_StockProducts\_Products\_ProductId] FOREIGN KEY([ProductId])

REFERENCES [dbo].[Products] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[StockProducts] CHECK CONSTRAINT [FK\_StockProducts\_Products\_ProductId]

GO

ALTER TABLE [dbo].[StockProducts] WITH CHECK ADD CONSTRAINT [FK\_StockProducts\_Stocks\_StockId] FOREIGN KEY([StockId])

REFERENCES [dbo].[Stocks] ([Id])

ON DELETE CASCADE

GO

ALTER TABLE [dbo].[StockProducts] CHECK CONSTRAINT [FK\_StockProducts\_Stocks\_StockId]

GO

Данная схема базы данных находится в третьей нормальной форме, поскольку каждая таблица имеет первичный ключ, позволяющий уникально идентифицировать данные. Это гарантирует, что любые данные в таблице можно найти однозначно, используя заданный ключ, что исключает проблемы дублирования информации и упрощает поиск.

Все атрибуты в таблицах атомарные, что означает отсутствие необходимости в преобразованиях при вставке или извлечении данных. Это также устраняет вероятность дублирования данных, поскольку каждый атрибут содержит минимально неделимое значение. Благодаря этому данные хранятся в ясной и удобной для обработки форме, что способствует поддержанию целостности информации в системе.

Также все неключевые атрибуты таблиц непосредственно зависят от первичного ключа и не имеют «неявных» зависимостей друг от друга. Благодаря этому данные в базе не подвержены аномалиям при вставке, удалении или обновлении, что означает, что изменение одних данных не повлияет на целостность других.

**3.4 Описание организации бизнес-логики программного средств**

Система построена на основе многоуровневой архитектуры (*N-Tier Architecture*), где каждый уровень имеет четко определенные обязанности и взаимодействует только с соседними уровнями. Основные уровни включают:

Уровень представления (Presentation Layer): Frontend часть реализована с использованием ASP.NET Core Razor Pages, что обеспечивает эффективный серверный рендеринг и высокую производительность. Особенностью архитектуры является использование компонентного подхода, где каждый компонент (например, *\_ProductPartial.cshtml*) инкапсулирует свою логику отображения и взаимодействия с пользователем.

Уровень приложения (Application Layer): Этот уровень содержит контроллеры и сервисы, которые обрабатывают бизнес-логику. Контроллеры (*AuthController, OrderController, SaleController* и др.) отвечают за обработку HTTP-запросов и маршрутизацию, в то время как сервисы (*OrderService, SaleService, ProductService*) реализуют бизнес-логику. Особенностью является использование паттерна CQRS (Command Query Responsibility Segregation) для разделения операций чтения и записи.

Уровень домена (Domain Layer): Содержит бизнес-модели и правила предметной области. Модели данных (*Order, Sale, Product, Stock*) представляют сущности бизнес-логики и содержат валидацию и бизнес-правила. Использование *Entity Framework Core* позволяет эффективно работать с данными и поддерживать целостность бизнес-правил.

Уровень доступа к данным (Data Access Layer): Реализован с использованием паттерна Repository, который абстрагирует доступ к данным. Каждый репозиторий (OrderRepository, SaleRepository, ProductRepository) отвечает за работу с определенным типом данных и обеспечивает единый интерфейс для доступа к данным.

Бизнес-логика системы организована вокруг основных сущностей: заказы, продажи, товары, склады и клиенты. Каждая сущность имеет свой набор бизнес-правил и валидаций, которые обеспечивают корректность данных и соблюдение бизнес-процессов.В системе реализована сложная логика обработки заказов, которая включает проверку наличия товаров на складах, управление статусами заказов и автоматическое создание продаж при выполнении заказов. Особое внимание уделено обработке исключительных ситуаций, таких как недостаточное количество товара на складе или попытка изменить завершенный заказ.Аналитический модуль построен с использованием паттерна Strategy, что позволяет легко добавлять новые типы аналитики и изменять существующие. Реализованы различные виды анализа: по продажам, по категориям товаров, по временным периодам. Это обеспечивает гибкость в получении и обработке аналитических данных.Система безопасности построена на основе JWT аутентификации и ролевой модели доступа. Реализована двухуровневая система прав: менеджер имеет полный доступ к системе, а пользователь - ограниченный набор функций. Это обеспечивает безопасность данных и контроль доступа к функционалу системы. Кэширование часто используемых данных реализовано с использованием паттерна *Cache-Aside*, что значительно повышает производительность системы. Особое внимание уделено кэшированию аналитических данных, которые требуют сложных вычислений. В целом, архитектура и бизнес-логика системы построены с учетом принципов SOLID и обеспечивают высокую степень гибкости, расширяемости и поддерживаемости кода. Это позволяет легко адаптировать систему под изменяющиеся требования и масштабировать её по мере роста бизнеса.

При разработке системы активно применялись паттерны проектирования.

Builder: В Program.cs активно используется паттерн Builder для конфигурации приложения. WebApplicationBuilder позволяет пошагово настраивать различные компоненты приложения, такие как аутентификация, авторизация, CORS и другие сервисы. Это обеспечивает гибкость и читаемость конфигурации.

Dependency Injection**:** Система активно использует внедрение зависимостей (DI) для управления зависимостями между компонентами. Все сервисы регистрируются в контейнере DI в Program.cs, что позволяет легко заменять реализации и тестировать компоненты, например:

builder.Services.AddScoped<IOrderService, OrderService>();

builder.Services.AddScoped<ISaleService, SaleService>();

Repository: Паттерн Repository используется для абстрагирования доступа к данным. Каждый репозиторий предоставляет единый интерфейс для работы с данными, скрывая детали реализации, например:

public interface IRepository<TEntity, TKey>

{

    Task<IEnumerable<TEntity>> GetAllAsync();

    Task<TEntity?> GetByIdAsync(TKey id);

    Task<TEntity> AddAsync(TEntity entity);

    Task UpdateAsync(TEntity entity);

    Task DeleteAsync(TKey id);

}

Facade: Сервисный слой (OrderService, SaleService) реализует паттерн Facade, предоставляя упрощенный интерфейс для работы с комплексной подсистемой. Это позволяет клиентскому коду работать с высокоуровневыми операциями, не вдаваясь в детали реализации.

Strategy:В аналитическом модуле используется паттерн Strategy для реализации различных алгоритмов анализа данных. Это позволяет легко добавлять новые типы анализа и изменять существующие без изменения клиентского кода.

Chain of Responsibility: Middleware в ASP.NET Core реализует паттерн Chain of Responsibility для обработки HTTP-запросов. Каждый middleware в цепочке может обработать запрос или передать его следующему middleware:

app.UseMiddleware<ErrorHandlingMiddleware>();

app.UseAuthentication();

app.UseAuthorization();

Data Transfer Object (DTO):Система активно использует DTO для передачи данных между слоями. Это позволяет контролировать, какие данные передаются, и обеспечивает дополнительный уровень безопасности:

public class OrderDto

{

    public int Id { get; set; }

    public string Status { get; set; }

    public decimal TotalAmount { get; set; }

*// ...*

}

Unit of Work:Entity Framework Core реализует паттерн Unit of Work, который обеспечивает атомарность операций с базой данных. Все изменения в рамках одной транзакции либо применяются все вместе, либо откатываются.

Далее спроектируем и опишем диаграмму деятельности   
для процесса просмотра аналитики. Диаграмма деятельности   
процесса просмотра аналитики представлена на рисунке 3.13.

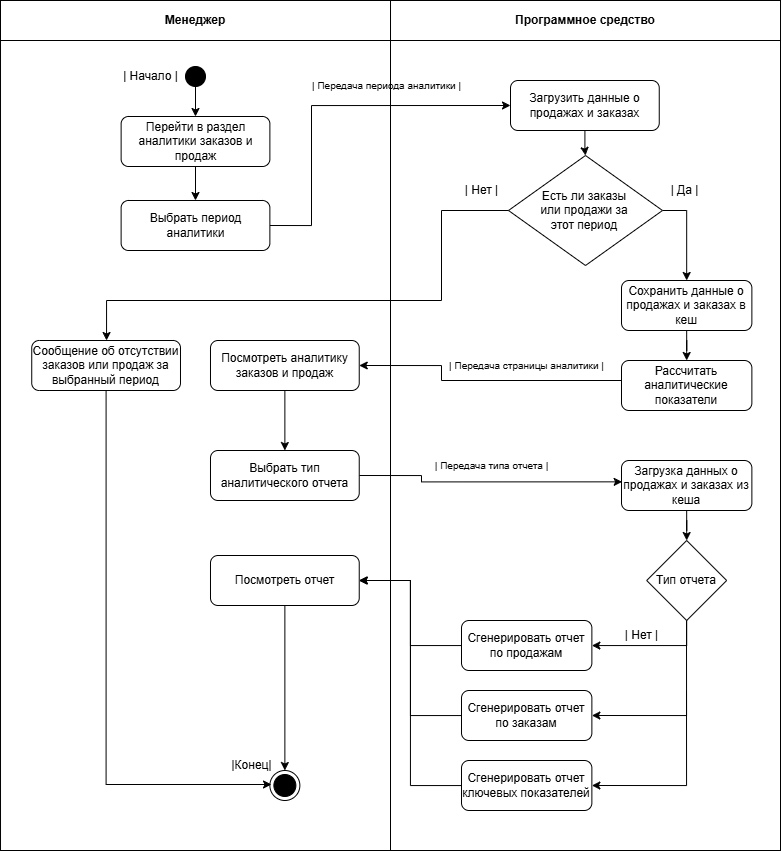


Рисунок 3.13 – Диаграмма деятельности процесса просмотра аналитики

Далее спроектируем и опишем диаграмму последовательности   
для процесса просмотра аналитики, представленную на рисунке 3.14.

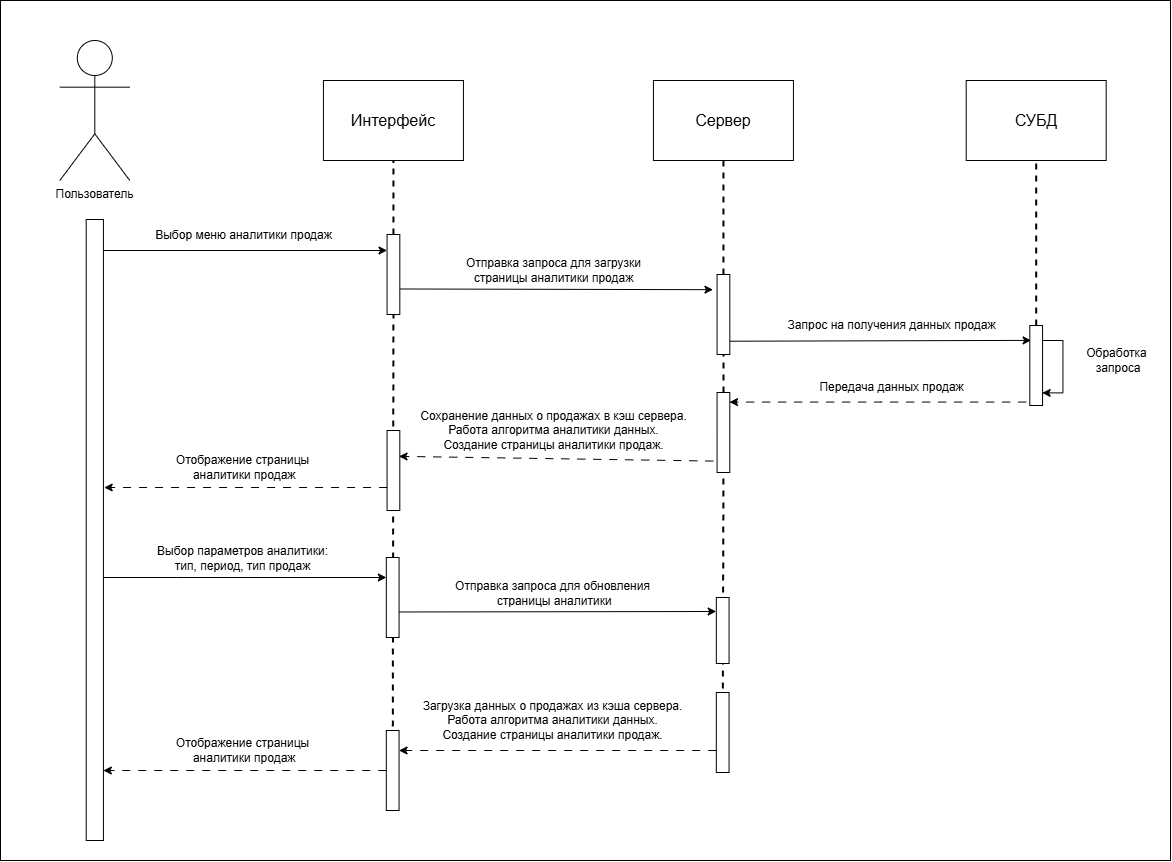


Рисунок 3.14 – Диаграмма деятельности процесса просмотра аналитики

Таким образом была описана архитектура системы, описаны основные бизнес-процессы, были построены диаграммы деятельности и последовательности, а также представлены основные паттерны проектирования, которые используются при разработке программного средства.

**3.5 Спецификация к разработанному API и механизмы обеспечения информационной безопасности**

*Swagger* – это профессиональный набор инструментов для разработчиков *API*. Данный набор инструментов активно разрабатывается *SmartBear Software* и поддерживается сообществом открытого исходного кода.

*OpenAPI* – это спецификация для описания *API*.

*Swagger* использует спецификацию *OpenAPI* для описания и документирования *API*, а инструменты *Swagger* позволяют использовать эту спецификацию для создания и тестирования *API*, а также для генерации клиентского кода.

Разработанная спецификация *OpenAPI* (для системы анализа заказов и продаж) представлена ниже:

{

"openapi": "3.0.1",

"info": {

"title": "RSAS WEB API",

"description": "API для системы анализа заказов и продаж",

"version": "v1"

},

"paths": {

"/api/Sales": {

"get": {

"tags": [

"Sales"

],

"responses": {

"200": {

"description": "Success",

"content": {

"text/plain": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/SaleDto"

}

}

},

"application/json": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/SaleDto"

}

}

},

"text/json": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/SaleDto"

}

}

}

}

},

"401": {

"description": "Unauthorized - Требуется аутентификация"

},

"403": {

"description": "Forbidden - Недостаточно прав"

}

},

"security": [

{

"Bearer": [ ]

}

]

}

},

"/api/Sales/from-order/{orderId}": {

"post": {

"tags": [

"Sales"

],

"parameters": [

{

"name": "orderId",

"in": "path",

"required": true,

"schema": {

"type": "integer",

"format": "int32"

}

}

],

"responses": {

"200": {

"description": "Success",

"content": {

"text/plain": {

"schema": {

"$ref": "#/components/schemas/SaleDto"

}

},

"application/json": {

"schema": {

"$ref": "#/components/schemas/SaleDto"

}

},

"text/json": {

"schema": {

"$ref": "#/components/schemas/SaleDto"

}

}

}

},

"401": {

"description": "Unauthorized - Требуется аутентификация"

},

"403": {

"description": "Forbidden - Недостаточно прав"

}

},

"security": [

{

"Bearer": [ ]

}

]

}

},

"/api/Sales/analytics": {

"get": {

"tags": [

"Sales"

],

"parameters": [

{

"name": "startDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

},

{

"name": "endDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

}

],

"responses": {

"200": {

"description": "Success",

"content": {

"text/plain": {

"schema": {

"$ref": "#/components/schemas/SalesAnalyticsDto"

}

},

"application/json": {

"schema": {

"$ref": "#/components/schemas/SalesAnalyticsDto"

}

},

"text/json": {

"schema": {

"$ref": "#/components/schemas/SalesAnalyticsDto"

}

}

}

},

"401": {

"description": "Unauthorized - Требуется аутентификация"

},

"403": {

"description": "Forbidden - Недостаточно прав"

}

},

"security": [

{

"Bearer": [ ]

}

]

}

},

"/api/Sales/orders-analytics": {

"get": {

"tags": [

"Sales"

],

"parameters": [

{

"name": "startDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

},

{

"name": "endDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

}

],

"responses": {

"200": {

"description": "Success",

"content": {

"text/plain": {

"schema": {

"$ref": "#/components/schemas/OrdersAnalyticsDto"

}

},

"application/json": {

"schema": {

"$ref": "#/components/schemas/OrdersAnalyticsDto"

}

},

"text/json": {

"schema": {

"$ref": "#/components/schemas/OrdersAnalyticsDto"

}

}

}

},

"401": {

"description": "Unauthorized - Требуется аутентификация"

},

"403": {

"description": "Forbidden - Недостаточно прав"

}

},

"security": [

{

"Bearer": [ ]

}

]

}

},

"/api/Sales/dashboard": {

"get": {

"tags": [

"Sales"

],

"parameters": [

{

"name": "startDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

},

{

"name": "endDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

}

],

"responses": {

"200": {

"description": "Success",

"content": {

"text/plain": {

"schema": {

"$ref": "#/components/schemas/DashboardAnalyticsDto"

}

},

"application/json": {

"schema": {

"$ref": "#/components/schemas/DashboardAnalyticsDto"

}

},

"text/json": {

"schema": {

"$ref": "#/components/schemas/DashboardAnalyticsDto"

}

}

}

},

"401": {

"description": "Unauthorized - Требуется аутентификация"

},

"403": {

"description": "Forbidden - Недостаточно прав"

}

},

"security": [

{

"Bearer": [ ]

}

]

}

},

"/api/Sales/forecast/categories": {

"get": {

"tags": [

"Sales"

],

"parameters": [

{

"name": "days",

"in": "query",

"schema": {

"type": "integer",

"format": "int32",

"default": 30

}

},

{

"name": "startDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

},

{

"name": "endDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

}

],

"responses": {

"200": {

"description": "Success",

"content": {

"text/plain": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/CategoryForecastDto"

}

}

},

"application/json": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/CategoryForecastDto"

}

}

},

"text/json": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/CategoryForecastDto"

}

}

}

}

},

"401": {

"description": "Unauthorized - Требуется аутентификация"

},

"403": {

"description": "Forbidden - Недостаточно прав"

}

},

"security": [

{

"Bearer": [ ]

}

]

}

},

"/api/Sales/forecast/demand": {

"get": {

"tags": [

"Sales"

],

"parameters": [

{

"name": "days",

"in": "query",

"schema": {

"type": "integer",

"format": "int32",

"default": 30

}

},

{

"name": "startDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

},

{

"name": "endDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

}

],

"responses": {

"200": {

"description": "Success",

"content": {

"text/plain": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/DemandForecastDto"

}

}

},

"application/json": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/DemandForecastDto"

}

}

},

"text/json": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/DemandForecastDto"

}

}

}

}

},

"401": {

"description": "Unauthorized - Требуется аутентификация"

},

"403": {

"description": "Forbidden - Недостаточно прав"

}

},

"security": [

{

"Bearer": [ ]

}

]

}

},

"/api/Sales/seasonality": {

"get": {

"tags": [

"Sales"

],

"parameters": [

{

"name": "years",

"in": "query",

"schema": {

"type": "integer",

"format": "int32",

"default": 3

}

},

{

"name": "startDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

},

{

"name": "endDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

}

],

"responses": {

"200": {

"description": "Success",

"content": {

"text/plain": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/SeasonalityImpactDto"

}

}

},

"application/json": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/SeasonalityImpactDto"

}

}

},

"text/json": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/SeasonalityImpactDto"

}

}

}

}

},

"401": {

"description": "Unauthorized - Требуется аутентификация"

},

"403": {

"description": "Forbidden - Недостаточно прав"

}

},

"security": [

{

"Bearer": [ ]

}

]

}

},

"/api/Sales/report": {

"post": {

"tags": [

"Sales"

],

"parameters": [

{

"name": "type",

"in": "query",

"schema": {

"$ref": "#/components/schemas/ReportType"

}

},

{

"name": "format",

"in": "query",

"schema": {

"$ref": "#/components/schemas/ReportFormat"

}

},

{

"name": "startDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

},

{

"name": "endDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

}

],

"requestBody": {

"content": {

"application/json": {

"schema": {

"$ref": "#/components/schemas/ReportFormattingSettings"

}

},

"text/json": {

"schema": {

"$ref": "#/components/schemas/ReportFormattingSettings"

}

},

"application/\*+json": {

"schema": {

"$ref": "#/components/schemas/ReportFormattingSettings"

}

}

}

},

"responses": {

"200": {

"description": "Success",

"content": {

"text/plain": {

"schema": {

"$ref": "#/components/schemas/ReportDto"

}

},

"application/json": {

"schema": {

"$ref": "#/components/schemas/ReportDto"

}

},

"text/json": {

"schema": {

"$ref": "#/components/schemas/ReportDto"

}

}

}

},

"401": {

"description": "Unauthorized - Требуется аутентификация"

},

"403": {

"description": "Forbidden - Недостаточно прав"

}

},

"security": [

{

"Bearer": [ ]

}

]

}

},

"/api/Sales/report/extended": {

"post": {

"tags": [

"Sales"

],

"parameters": [

{

"name": "type",

"in": "query",

"schema": {

"$ref": "#/components/schemas/ReportType"

}

},

{

"name": "format",

"in": "query",

"schema": {

"$ref": "#/components/schemas/ReportFormat"

}

},

{

"name": "startDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

},

{

"name": "endDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

}

],

"requestBody": {

"content": {

"application/json": {

"schema": {

"$ref": "#/components/schemas/ReportFormattingSettings"

}

},

"text/json": {

"schema": {

"$ref": "#/components/schemas/ReportFormattingSettings"

}

},

"application/\*+json": {

"schema": {

"$ref": "#/components/schemas/ReportFormattingSettings"

}

}

}

},

"responses": {

"200": {

"description": "Success",

"content": {

"text/plain": {

"schema": {

"$ref": "#/components/schemas/ReportDto"

}

},

"application/json": {

"schema": {

"$ref": "#/components/schemas/ReportDto"

}

},

"text/json": {

"schema": {

"$ref": "#/components/schemas/ReportDto"

}

}

}

},

"401": {

"description": "Unauthorized - Требуется аутентификация"

},

"403": {

"description": "Forbidden - Недостаточно прав"

}

},

"security": [

{

"Bearer": [ ]

}

]

}

},

"/api/Sales/extended": {

"get": {

"tags": [

"Sales"

],

"parameters": [

{

"name": "startDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

},

{

"name": "endDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

}

],

"responses": {

"200": {

"description": "Success",

"content": {

"text/plain": {

"schema": {

"$ref": "#/components/schemas/ExtendedSalesAnalyticsDto"

}

},

"application/json": {

"schema": {

"$ref": "#/components/schemas/ExtendedSalesAnalyticsDto"

}

},

"text/json": {

"schema": {

"$ref": "#/components/schemas/ExtendedSalesAnalyticsDto"

}

}

}

},

"401": {

"description": "Unauthorized - Требуется аутентификация"

},

"403": {

"description": "Forbidden - Недостаточно прав"

}

},

"security": [

{

"Bearer": [ ]

}

]

}

},

"/api/Sales/kpi": {

"get": {

"tags": [

"Sales"

],

"parameters": [

{

"name": "startDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

},

{

"name": "endDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

}

],

"responses": {

"200": {

"description": "Success",

"content": {

"text/plain": {

"schema": {

"$ref": "#/components/schemas/KpiDto"

}

},

"application/json": {

"schema": {

"$ref": "#/components/schemas/KpiDto"

}

},

"text/json": {

"schema": {

"$ref": "#/components/schemas/KpiDto"

}

}

}

},

"401": {

"description": "Unauthorized - Требуется аутентификация"

},

"403": {

"description": "Forbidden - Недостаточно прав"

}

},

"security": [

{

"Bearer": [ ]

}

]

}

},

"/api/Sales/top-products": {

"get": {

"tags": [

"Sales"

],

"parameters": [

{

"name": "count",

"in": "query",

"schema": {

"type": "integer",

"format": "int32",

"default": 10

}

},

{

"name": "startDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

},

{

"name": "endDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

}

],

"responses": {

"200": {

"description": "Success",

"content": {

"text/plain": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/TopProductResultDto"

}

}

},

"application/json": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/TopProductResultDto"

}

}

},

"text/json": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/TopProductResultDto"

}

}

}

}

},

"401": {

"description": "Unauthorized - Требуется аутентификация"

},

"403": {

"description": "Forbidden - Недостаточно прав"

}

},

"security": [

{

"Bearer": [ ]

}

]

}

},

"/api/Sales/category-sales": {

"get": {

"tags": [

"Sales"

],

"parameters": [

{

"name": "startDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

},

{

"name": "endDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

}

],

"responses": {

"200": {

"description": "Success",

"content": {

"text/plain": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/CategorySalesResultDto"

}

}

},

"application/json": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/CategorySalesResultDto"

}

}

},

"text/json": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/CategorySalesResultDto"

}

}

}

}

},

"401": {

"description": "Unauthorized - Требуется аутентификация"

},

"403": {

"description": "Forbidden - Недостаточно прав"

}

},

"security": [

{

"Bearer": [ ]

}

]

}

},

"/api/Sales/trend": {

"get": {

"tags": [

"Sales"

],

"parameters": [

{

"name": "startDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

},

{

"name": "endDate",

"in": "query",

"schema": {

"type": "string",

"format": "date-time"

}

},

{

"name": "interval",

"in": "query",

"schema": {

"type": "string",

"default": "1d"

}

}

],

"responses": {

"200": {

"description": "Success",

"content": {

"text/plain": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/SalesTrendResultDto"

}

}

},

"application/json": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/SalesTrendResultDto"

}

}

},

"text/json": {

"schema": {

"type": "array",

"items": {

"$ref": "#/components/schemas/SalesTrendResultDto"

}

}

}

}

},

"401": {

"description": "Unauthorized - Требуется аутентификация"

},

"403": {

"description": "Forbidden - Недостаточно прав"

}

},

"security": [

{

"Bearer": [ ]

}

]

}

},

}

Разработанная документация для API авторизации представлена на рисунке 3.15.

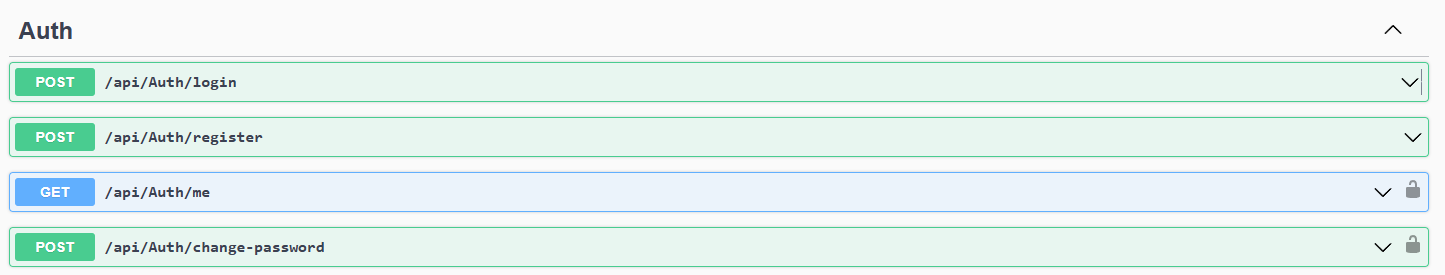


Рисунок 3.15 – Документация API авторизации

Разработанная документация для API заказов представлена на рисунке 3.16.



Рисунок 3.16 – Документация API заказов

Разработанная документация для API товаров представлена на рисунке 3.17.



Рисунок 3.17 – Документация API товаров

Разработанная документация для API аналитики представлена на рисунке 3.18.



Рисунок 3.18 – Документация API аналитики

Разработанная документация для API складов представлена на рисунке 3.19.

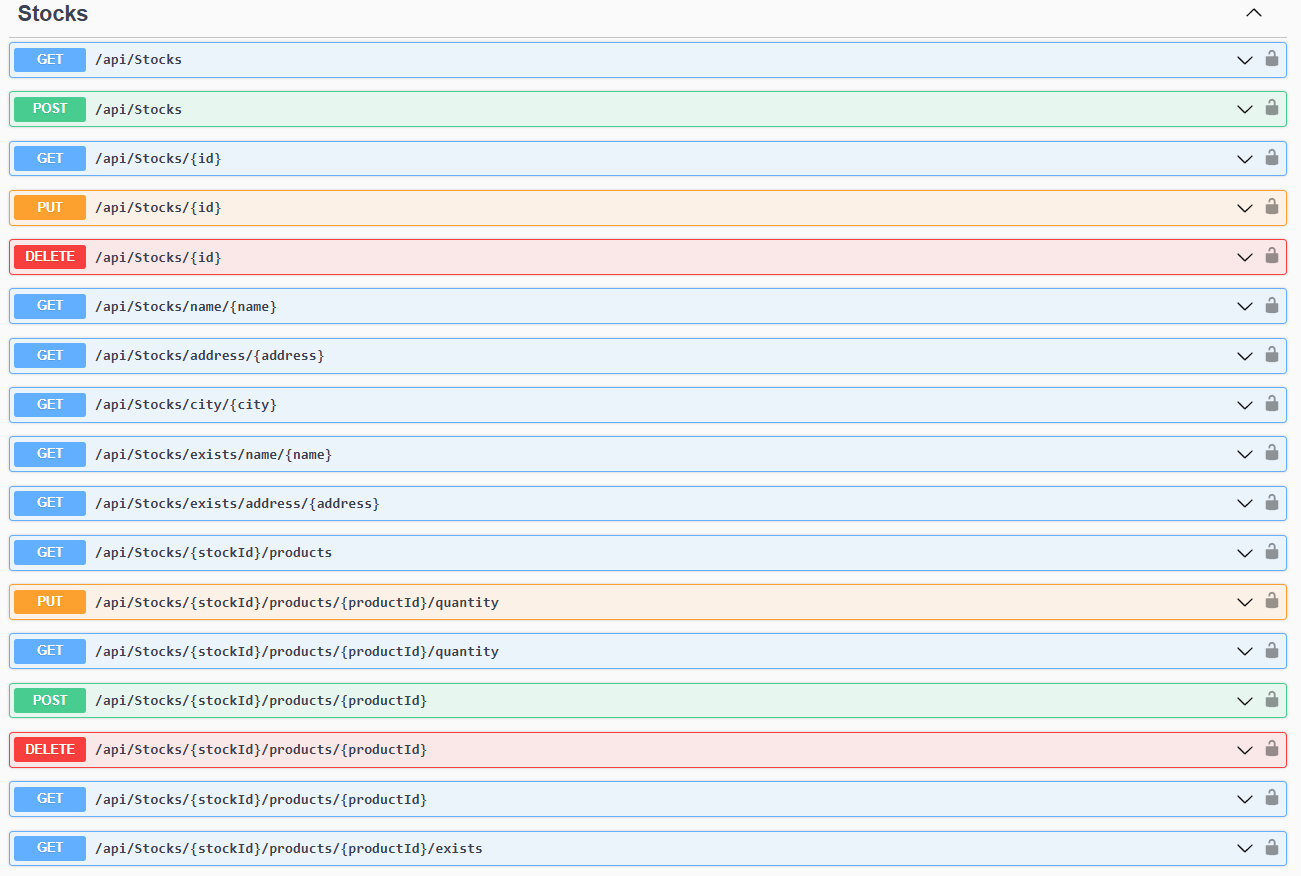


Рисунок 3.19 – Документация API складов

Разработанная документация для API клиентов представлена на рисунке 3.20.

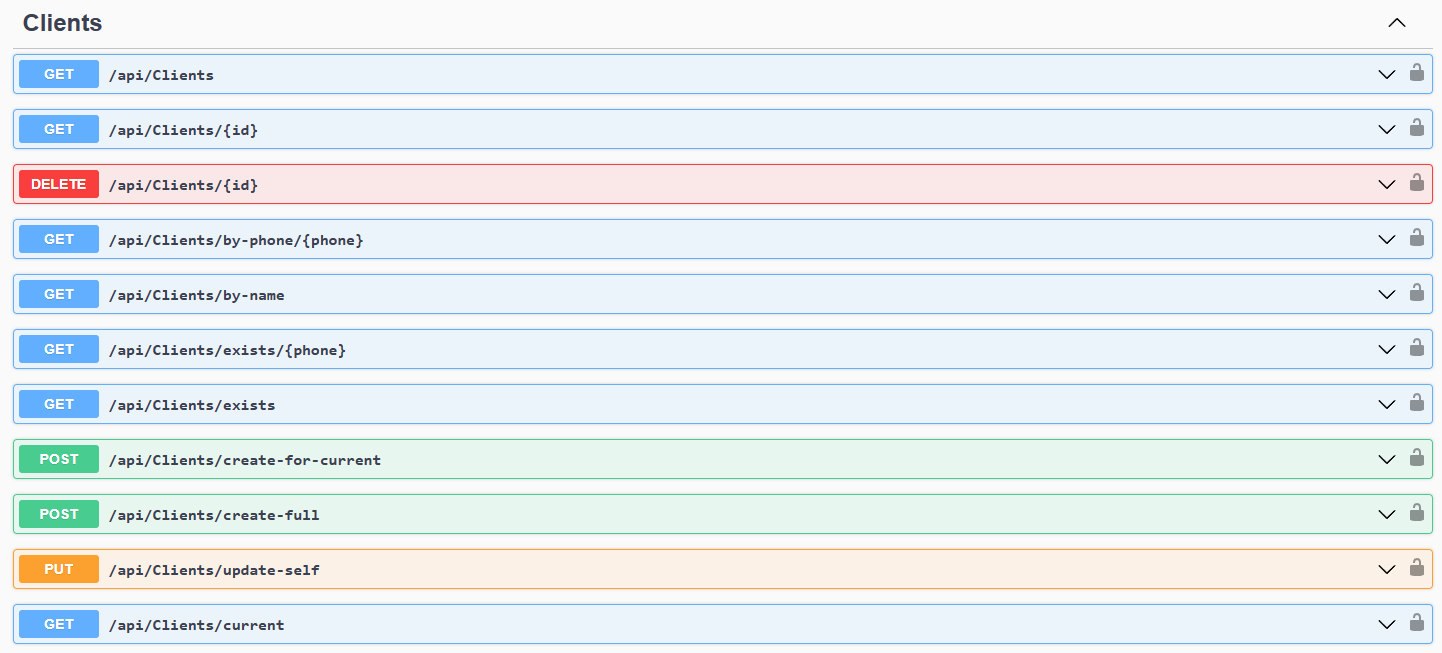


Рисунок 3.20 – Документация API клиентов

В разработанной системе анализа заказов и продаж реализован комплексный подход к обеспечению информационной безопасности, включающий несколько уровней защиты.

Аутентификация и авторизация. Система использует современный механизм аутентификации на основе JWT (JSON Web Tokens). При успешной аутентификации пользователь получает токен, который содержит информацию о его идентификаторе, роли и времени действия. Токен подписывается с использованием алгоритма HMAC-SHA256 и имеет ограниченный срок действия, что обеспечивает защиту от перехвата и повторного использования. На стороне клиента токен хранится в защищенном HTTP-only cookie с флагом Secure и SameSite=Strict, что предотвращает атаки XSS и CSRF.Реализована двухуровневая система ролей:

* Менеджер (Manager) - имеет полный доступ к системе
* Клиент (Client) - имеет ограниченный доступ к функционалу

Для каждой роли определены соответствующие политики авторизации, которые проверяются при доступе к защищенным ресурсам.

Защита данных. Все пароли пользователей хранятся в хешированном виде с использованием стандартных механизмов ASP.NET Core Identity. При регистрации и смене пароля применяются строгие требования к сложности:

* Минимальная длина 8 символов
* Обязательное наличие заглавных и строчных букв
* Обязательное наличие цифр

Для защиты от SQL-инъекций используется Entity Framework Core, который автоматически экранирует все параметры запросов. Дополнительно реализована валидация всех входных данных на стороне сервера с использованием атрибутов валидации и FluentValidation.

Безопасность API. REST API защищен несколькими механизмами:

1. Обязательная аутентификация для всех эндпоинтов (кроме публичных)
2. Валидация входных данных на уровне DTO
3. Централизованная обработка ошибок через middleware
4. Защита от переполнения запросами (rate limiting)
5. CORS политики, ограничивающие доступ только с доверенных источников

Безопасность клиентской части. На стороне клиента реализованы следующие меры безопасности:

1. Валидация форм на стороне клиента
2. Защита от XSS через автоматическое экранирование данных
3. Безопасное хранение JWT токена
4. Обработка истечения срока действия токена
5. Защита от CSRF через SameSite cookies

Реализованные механизмы безопасности обеспечивают надежную защиту системы от различных типов атак и несанкционированного доступа, соответствуя современным стандартам информационной безопасности.

# **4** **ТЕСТИРОВАНИЕ И ПРОГРАММНАЯ ДОКУМЕНТАЦИЯ К ПРОГРАММНОМУ СРЕДСТВУ**

**4.1 Функциональное тестирование программного средства**

**4.2 Модульное и интеграционное тестирование бизнес-логики программного средства**

**4.3 Руководство по развертыванию программного средства**

**4.4 Руководство пользователя**

1. **ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ ПРОГРАММНОГО СРЕДСТВА, РАЗРАБОТАННОГО ПО ИНДИВИДУАЛЬНОМУ ЗАКАЗУ**
   1. **Характеристика программного средства**

Целью дипломного проекта является разработка программного средства для автоматизации процесса анализа продаж и заказов в розничном магазине.

Разработанное программное средство представляет собой веб-приложение, предназначенное для автоматизации учета и анализа продаж, заказов и товарных запасов. Основное назначение программы заключается в систематизации данных о продажах и заказах, контроле товарных остатков, мониторинге динамики продаж, выявлении ключевых тенденций и формировании отчетности. Программа позволяет оперативно анализировать продажи, учитывать заказы и управлять ассортиментом, что способствует повышению эффективности работы магазина.

Программное средство обеспечивает точный учет всех операций, связанных с заказами и продажами, формирует детализированные аналитические отчеты и визуализирует динамику товарооборота. Это упрощает процесс управления магазином, минимизирует риски дефицита или избыточных остатков и позволяет принимать обоснованные управленческие решения. Дополнительно программа предоставляет средства для мониторинга эффективности продаж, анализа спроса и автоматического формирования отчетов, что способствует улучшению стратегического планирования.

Экономическая целесообразность инвестиций в разработку и реализацию программного средства осуществляется на основе расчета и оценки следующих показателей:

* затрат на разработку и цены программного средства;
* прибыли;
* рентабельности инвестиций.

Данное программное средство разработано по индивидуальному заказу и реализуется по отпускной цене.

* 1. **Расчет затрат на разработку и определение цены программного средства**

Основная заработная плата исполнителей проекта определяется по формуле

|  |  |
| --- | --- |
|  | (5.1) |

где *n* – категории исполнителей, занятых разработкой программного средства; Зч*i*– часовой оклад плата *i*-й категории, руб.; Кпр – коэффициент премий (1,5); *ti* – трудоемкость работ, выполняемых исполнителем *i*-й категории, ч.

На 2025 год расчетная норма рабочего времени для пятидневной рабочей недели составляет 168 часов, 8 часов работы в день, среднемесячная расчетная норма рабочего времени – 21 день (согласно mintrud.gov.by). По состоянию на 27.02.2025, среднемесячная заработная плата *IT*-специалистов в Республике Беларусь составляет 7125 рублей (согласно salaries.devby.io).

В таблице 5.1 представлен расчет затрат на основную заработную плату разработчиков.

Таблица 5.1 ­– Расчет затрат на основную заработную плату разработчиков

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Категория исполнителя | Месячный оклад, р. | Часовой оклад, р. | Трудоемкость работ, ч | Итого, р. |
| Руководитель проекта | 3200,00 | 19,04 | 40,00 | 761,90 |
| Бизнес-аналитик | 4700,00 | 27,98 | 40,00 | 1119,20 |
| Программист *C#* | 7500,00 | 44,64 | 80,00 | 3571,24 |
| Программист *JS, CSS, HTML* | 6800,00 | 40,48 | 80,00 | 3238,09 |
| Тестировщик | 2500,00 | 14,88 | 40,00 | 595,24 |
| Дизайнер | 3000,00 | 17,86 | 20,00 | 357,14 |
| Итого | | | | 9642,81 |
| Премия и иные стимулирующие выплаты (50 %) | | | | 4821,40 |
| Всего затрат на основную заработную плату (Зо) | | | | 14464,21 |

Таким образом основная заработная плата будет равна

Дополнительная заработная плата разработчиков вычисляется по формуле

|  |  |
| --- | --- |
|  | (5.2) |

где Нд – норматив дополнительной заработной платы (10%); Зо – основная заработная плата исполнителей проекта.

Таким образом, дополнительная заработная плата составит

Отчисления в фонд социальной защиты населения и на обязательное страхование определяются по формуле

|  |  |
| --- | --- |
|  | (5.3) |

где Нсоц – норматив отчислений в ФСЗН и Белгосстрах (в соответствии с действующим законодательством – 34,6%).

Таким образом, отчисления на социальные нужды составляют

Рассчитаем прочие расходы по формуле

|  |  |
| --- | --- |
|  | (5.4) |

где Нпр – норматив прочих расходов (30%).

Таким образом, прочие расходы будут равны

Общая сумма затрат на разработку вычисляется по формуле

|  |  |
| --- | --- |
|  | (5.5) |

Таким образом общая сумма затрат составит

Плановая прибыль, включаемая в цену программного средства, вычисляется по формуле

|  |  |
| --- | --- |
|  | (5.6) |

где Рп.с – рентабельность затрат на разработку программного средства (30%).

Таким образом, плановая прибыль будет равна

Отпускная цена программного средства вычисляется по формуле

|  |  |
| --- | --- |
|  | (5.7) |

Таким образом отпускная цена программного средства будет равна

Налог на добавленную стоимость определяется по формуле

|  |  |
| --- | --- |
|  | (5.8) |

где Нд.с – ставка налога на добавленную стоимость согласно действующему законодательству, % (20%).

Тогда НДС будет равен

Таким образом, цена с учетом НДС составит

Все расчеты сметы затрат и отпускной цены сведены в таблицу 5.2.

Таблица 4.2 ­– Смета затрат на разработку программного средства

|  |  |
| --- | --- |
| Наименование статей | Значение, руб. |
| Основная заработная плата | 14464,21 |
| Дополнительная заработная плата | 1446,42 |
| Отчисления на социальные нужды | 5727,83 |
| Прочие расходы | 4339,26 |
| Общая сумма затрат на разработку | 25977,72 |
| Плановая прибыль, включаемая в цену программного средства | 7793,32 |
| Отпускная цена программного средства | 33771,04 |
| Сумма налога на добавленную стоимость | 5628,50 |
| Цена с учетом НДС | 39399,54 |

По итогу были рассчитаны статьи расходов, составлена смета затрат на разработку программного средства, а также рассчитана отпускная цена программного средства.

* 1. **Расчет результата от разработки и реализации программного средства**

Экономический эффект организации-разработчика программного средства представляет собой прирост чистой прибыли, полученной от разработки и реализации программного средства заказчику. Прирост чистой прибыли, полученной от продажи, можно посчитать по формуле

|  |  |
| --- | --- |
|  | (5.9) |

где Пп.с – прибыль, включаемая в цену программного средства, руб.; Нп – ставка налога на прибыль (20%).

Таким образом, прирост чистой прибыли равен

* 1. **Расчет показателей экономической эффективности разработки и реализации программного средства**

Оценка экономической эффективности разработки осуществляется с помощью расчета простой нормы прибыли (рентабельности инвестиций на разработку программного средства) по формуле

|  |  |
| --- | --- |
|  | (5.10) |

где ΔПч – прирост чистой прибыли, полученной от разработки программного средства, руб.; Зр – затраты на разработку программного средства, руб.

Рассчитаем рентабельность инвестиций по формуле

В результате экономического обоснования разработки и реализации программного средства были получены следующие результаты:

* полная общая сумма затрат на разработку и цена с учетом НДС составляют 25977,72 руб. и 39399,54 руб. соответственно;
* прирост чистой прибыли равен 6234,66 руб;
* уровень рентабельности инвестиций составляет 24 %.

Учитывая все показатели, можно сделать вывод о том, что разработка данного программного средства является эффективной и вложение инвестиций в разработку целесообразны.

# **ЗАКЛЮЧЕНИЕ**

Отправной точкой работы над дипломным проектом стал анализ предметной области и существующих решений в сфере автоматизации анализа заказов и продаж в розничных магазинах. В ходе исследования были изучены методы сбора, обработки и визуализации данных, а также подходы к аналитике продаж и прогнозированию спроса. Проведенный обзор аналогичных программных решений позволил выявить их ключевые преимущества и недостатки, что послужило основой для формирования требований к разрабатываемому программному средству.

В рамках преддипломной практики основное внимание было уделено анализу предметной области и моделированию бизнес-процессов. В результате были разработаны модели, отражающие ключевые аспекты автоматизированного анализа заказов и продаж, а также выявлены основные проблемы, возникающие в процессе учета и обработки данных в розничной торговле. Моделирование бизнес-процессов позволило определить оптимальные подходы к управлению товарными запасами, обработке клиентских заказов и формированию аналитических отчетов.

На основе проведенного анализа была спроектирована концепция программного средства, предназначенного для автоматизации анализа заказов и продаж. Основное внимание уделялось разработке архитектурных решений, модели данных и пользовательских сценариев. В ходе работы были определены ключевые функции системы, включая сбор и обработку данных о продажах и заказах, выявление тенденций и закономерностей, формирование аналитических отчетов и визуализация ключевых показателей деятельности магазина. Дополнительно были предложены механизмы управления товарными запасами и учета клиентских заказов, обеспечивающие комплексный подход к управлению продажами.

Результатом работы стало формирование четкого представления о требованиях к разрабатываемому программному средству, его функциональных возможностях и архитектуре. Проведенный анализ позволил заложить прочную основу для дальнейшей реализации системы, а также определить возможные пути ее расширения и интеграции с внешними учетными системами.

По итогам выполнения преддипломной практики был составлен отчет, включающий следующие разделы:

* описание организационной структуры предприятия;
* анализ литературных источников и программных решений по теме дипломного проекта;
* моделирование предметной области и разработка требований к программному средству
* экономическое обоснование программного средства, разработанного по индивидуальному заказу;
* план-проспект дипломного проекта.

Кроме того, были подготовлены графические материалы, включающие:

* BPMN-модель процессов анализа заказов и продаж;
* архитектурную схему программного средства;
* модель данных и связи между сущностями;
* диаграммы пользовательских сценариев работы с системой.

Таким образом, в ходе преддипломной практики были успешно решены все поставленные задачи, а полученные результаты обладают высокой практической значимостью. В дальнейшем возможны расширение функциональности системы, интеграция с внешними учетными системами и совершенствование алгоритмов анализа данных для повышения точности прогнозирования продаж.