МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение
высшего образования
«Тольяттинский государственный университет»

Институт математики, физики и информационных технологий
(наименование института полностью)

Кафедра «Прикладная математика и информатика»
(наименование)

09.03.03 Прикладная информатика
(код и наименование направления подготовки, специальности)

Бизнес-информатика
(направленность (профиль) / специализация)

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА (БАКАЛАВРСКАЯ РАБОТА)

на тему «Разработка веб-представительства для предприятия розничной торговли»

Обучающийся А.Е. Бережонов
(И.О. Фамилия) (личная подпись)

Руководитель к.т.н. Д.Г. Токарев
(ученая степень, звание, И.О. Фамилия)

Тольятти 2022

Аннотация

Работа выполнена на 52 страницах, содержит 24 рисунка, 3 таблицы, 29 источников литературы.

Темой выпускной квалификационной работы является разработка веб-представительства для компании розничной торговли, предназначенная для повышения эффективности работы с клиентами.

Цель работы – веб-представительства для компании розничной торговли.

Задачи работы:

- исследовать предметную область и выполнить постановку задачи на разработку веб-приложения для компании розничной торговли с возможностью онлайн заказов;
- спроектировать веб-приложение, провести проектирование схемы базы данных;
- реализовать веб-приложение для компании розничной торговли.

Первая глава работы включает описание деятельности компании по розничной торговли. Представлены основные бизнес-процессы магазина. На основе полученной информации о бизнес-процессах было проведено моделирование в нотации IDEF0.

Вторая глава подробно описывает структуру веб-представительства магазина. Также рассмотрены пункты что является входящей информацией и выходящие потоки информации. Выделены основные объекты веб-приложения и составлена структура базы данных.

Третья глава посвящена практической разработке веб-приложения, что включает разработку интерфейса веб-представительства и схемы взаимодействия клиента и магазина.

Содержание

Введение

Данная работа посвящена разработке веб-представительства предприятия розничной торговли (на примере ООО «Розница К-1»).

Развитие торговли как оптовой, так и розничной в наше время в больших объемах осуществляется с помощью сети интернет, покупатель уже не хочет тратить свое время на походы в магазин, а выбирает способ покупки в интернете, тем более что множество торговых сетей и маленьких магазинов стараются доставлять товары домой клиенту. На данное время актуально будет создать веб-представительство предприятия розничной торговли, чтоыб увеличить рост продаж и позволить клиентам получать всю необходимую информацию, не выходя из дома. [1].

Компании особенно занимающиеся розничной торговлей одежды стараются создать свою визитную карточку в сети интернет, чтобы охватить как можно больше аудиторию своих потребителей и как правило потенциальных покупателей в будущем. На основе этого и создаются веб-сайты компаний, где широко освещаются все товары и акции компании, также дают возможность пользователям, не выходя из дома выбрать товар и купить его, с помощью онлайн-заказа.

Цель работы – разработка веб-представительства предприятия розничной торговли (на примере ООО «Розница К-1»).

Задачи работы:

- исследовать предметную область и выполнить постановку задачи на разработку веб-приложения для компании розничной торговли с возможностью онлайн заказов;
- спроектировать веб-приложение, провести проектирование схемы базы данных;
- реализовать веб-приложение для компании розничной торговли.

Объектом исследования является торговая компания ООО «Розница К-1», которая занимается розничной торговлей.

Предметом исследования является разработка веб-представительства компании.

Выпускная квалификационная работа состоит из введения, трех разделов, заключения, списка используемых источников.

Первая глава работы включает описание деятельности компании по розничной торговли. Представлены основные бизнес-процессы магазина. На основе полученной информации о бизнес-процессах было проведено моделирование в нотации IDEF0 [2].

Вторая глава подробно описывает структуру веб-представительства магазина. Также рассмотрены пункты что является входящей информацией и выходящие потоки информации. Выделены основные объекты веб-приложения и составлена структура базы данных.

Третья глава посвящена практической разработке веб-приложения, что включает разработку интерфейса веб-представительства и схемы взаимодействия клиента и магазина.

Создание веб-приложения торговой фирмы позволит стать более конкурентоспособным на рынке розничной торговли и расширить свою деятельность [3].

1 Исследование процессов компании ООО «Розница К-1»

1.1 Анализ деятельности компании по розничной продаже одежды

На заре развития всемирной паутины никто и подумать не мог, что on-line покупки станут так популярны. Да и когда виртуальная торговля стала реальностью, многие все не решались заказывать одежду в интернет-магазине. Развитее технологий позволило вести продажу в интернет-пространстве, что значительно экономит время клиентов на примерку одежды и поиск нужной вещи путем поиска их в магазине. Виртуальные бутики становятся популярнее с каждым годом.

Онлайн-магазин или как обычно называет, веб-представительство стал оптимальным решением для современных модников и модниц, которые ценят свое время. И это лишь первый из аргументов в его пользу.

Проектирование интернет-магазина значительно экономит средства на зарплатный фонд сотрудников, а также время обслуживания клиентов. Также плюсом является продажа, которая позволяет оплачивать покупки, не выходя из дома [4].

Основная деятельность магазина заключается в розничной продаже одежды. На текущий момент продажа товара осуществляется только в магазинах на территории города, покупателям чтобы купить товар необходимо приходить в магазин и выбирать из представленного ассортимента товара. Таким образом тратиться очень много времени у покупателя, что выбрать и приобрести товар. В настоящее время все больше людей предпочитают пользоваться онлайн-заказами, когда товар могут доставить домой или в пункт выдачи, к тому же предоставляется возможность вернуть товар, если он по каким-то причинам не устроил. Такие покупки сокращают потери времени и позволяют самому выбирать, когда и как выбирать покупки.

В данной предметной области можно обозначить две роли:

– клиент – покупает товар;

– сотрудник – продает товар.

Товары поставляются в магазин только после создания заявки и отправления ее поставщику. На основании заказа поставщик уже формирует и доставляет заказ непосредственно в магазин розничной торговли.

Реализация товара происходит путем продажи товара на кассе. Для осуществления расчетов с покупателями предусмотрен наличный и безналичный расчет, для этого в магазине есть касса и специальное оборудование для магазина (компьютер, монитор, клавиатура и пр.).

После реализации, товар уходит с остатка и не числится на балансе магазина.

Структура магазина представлена на рисунке 1. Магазин занимается розничной продажей одежды.

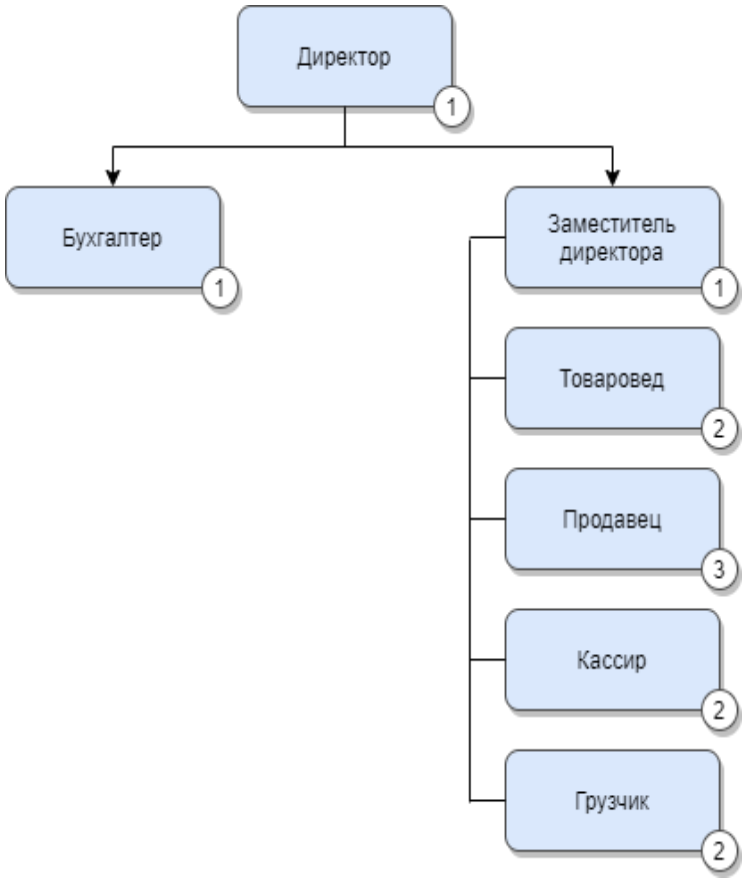

Рисунок 1 – Организационная структура магазина

1.2 Выделение бизнес-процессов магазина

Рассмотри бизнес-процессы магазина.

1. Заказ товара. Заказы формируются директором магазина на основе ассортимента магазина и потребности в товаре. Предварительно выбираются поставщики и постоянно ищут новых производителей, с которыми заключается договор на поставку. Поставщики товаров рассматриваются с разных сторон, важным фактором является прозрачность работы и платежей, также критерий возврата товара. После того, как директором сформирована и отправлена заявка, поставщик формирует заказ и отправляет его в адрес торговой точки.

2. Возврат товара поставщику. Если товар был принят не в надлежащем виде его можно вернуть поставщику. Для этого товаровед или заместитель директора оформляет возвратные документы на товар в положенной форме. При приходе товара поставщику возвращается товар со всеми сопроводительными документами. Директор должен обязательно отслеживать этот процесс, товар, не вовремя возвращенный поставщику, остается на балансе магазина как недостача, так как реализовать его уже нельзя. В программе по реализации товара нет папки или пункта, где хранятся возвратные документы для отслеживания факта возврата.

3. Заключение договора с поставщиком. Перед тем как поставить товар в магазин, необходимо заключить с поставщиком договор. Директор выбирает поставщика по предоставленным коммерческим предложениям, критерием рассмотрения является прозрачность предложенных отношений сотрудничества, наличие возврата товара, а также все возможные бонусы. После выбора поставщика и заключения договора все документы хранятся у бухгалтера, директор теряет все данные о поставщике после отправления документов в бухгалтерию.

На рисунке 2 показан процесс продажи одежды в магазине в нотации IDEF0.

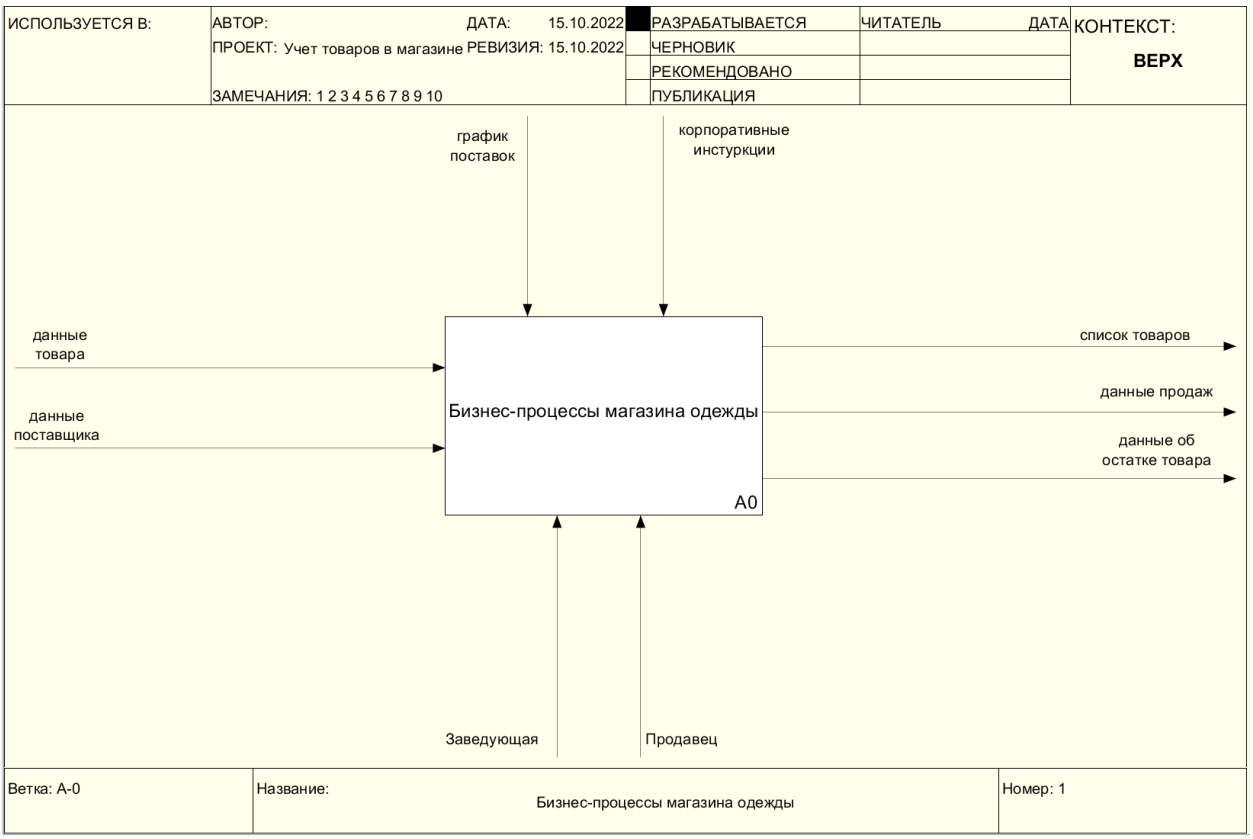

Рисунок 2 – Контекстная диаграмма «как есть»

Бизнес-процессы магазина происходят на основе полученных данных товара и поставщика, после чего осуществляется приемка товара и его реализация. Исходящей информацией будет уже список товаров, информация о продаже и остатки товара в магазине.

Подробный процесс деятельности магазина розничной торговли показан на рисунке 3.

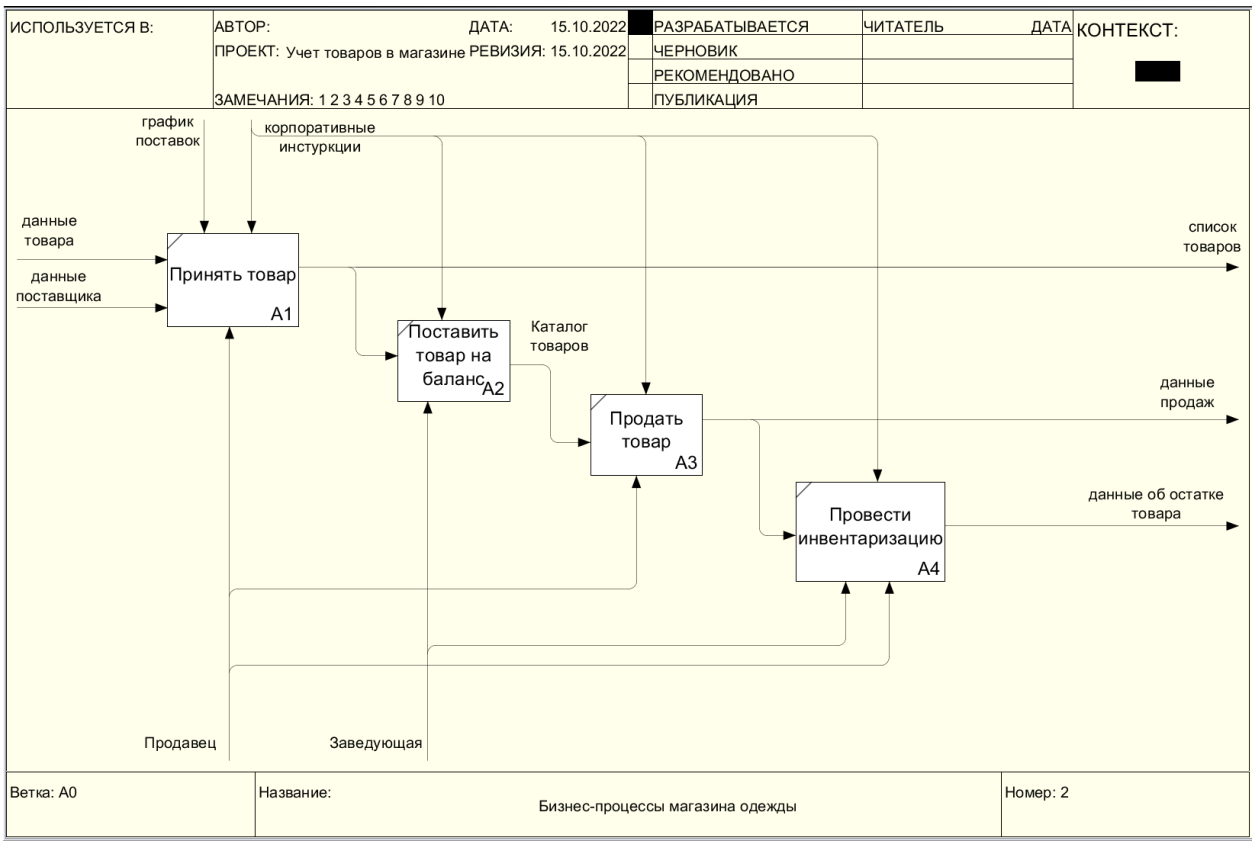

Рисунок 3 – Декомпозиция модели «как есть»

Приемка товара происходят, когда в магазин поставляется товар и появляется данные о товаре и поставщике. После того как товар принимается формируется список товаров и ставится на баланс магазина, для учета товаров.

После реализации товара товар должен автоматически списываться с остатка, таким образом показывается актуальное количество товара в магазине.

Если представить диаграмму потоков то можно наглядно увидеть как происходит обмен данными мужеду объектами, рисунок 4.

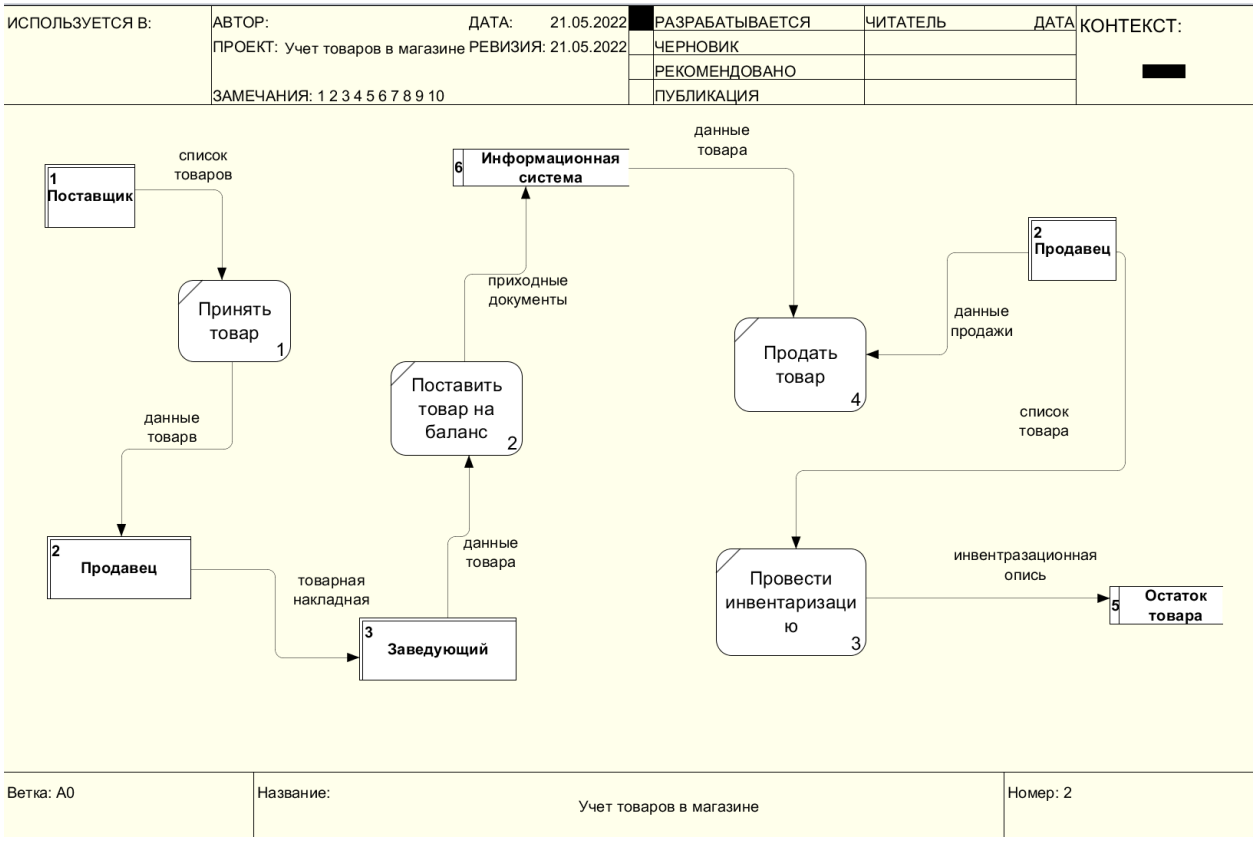

Рисунок 4 – Декомпозиция процесса учета товаров в нотации DFD

На рисунке 4 показан обмен данными между объектами:

– поставщик;
– продавец;
– заведующая;
– информационная система;
– остаток товара.

Основными сущностями, после рассмотрения предметной области целесообразно выбрать следующие [4]:

– список товаров;
– список сотрудников;
– продажи.

Чтобы правильно спроектировать и реализовать веб-приложение для осуществления онлайн продаж, необходимо представлять как работает интернет-магазин.

Опишем процесс интернет-магазина одежды с помощью моделирования. На рисунке 5 показана контекстная диаграмма процесса.

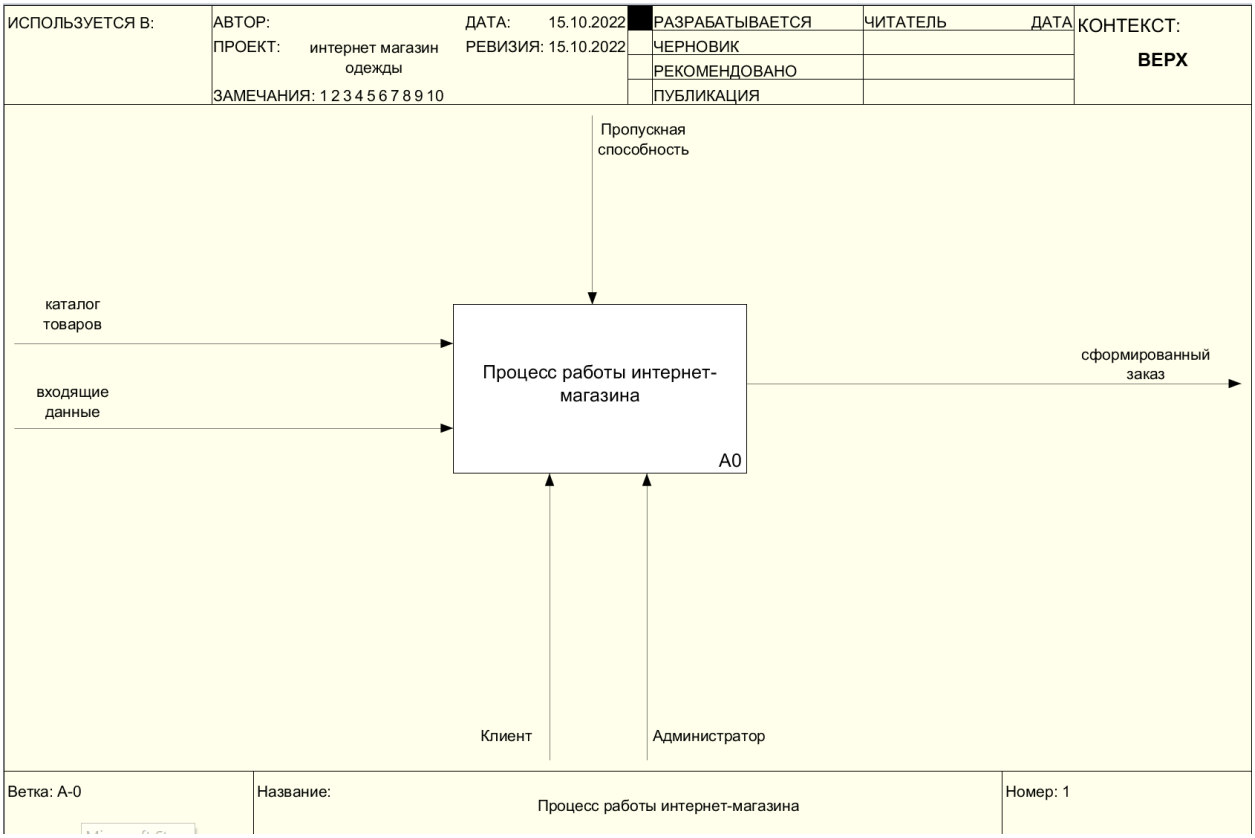

Рисунок 5 – Контекстная диаграмма работы интернет-магазина

Проведем декомпозицию процесса, рисунок 6.

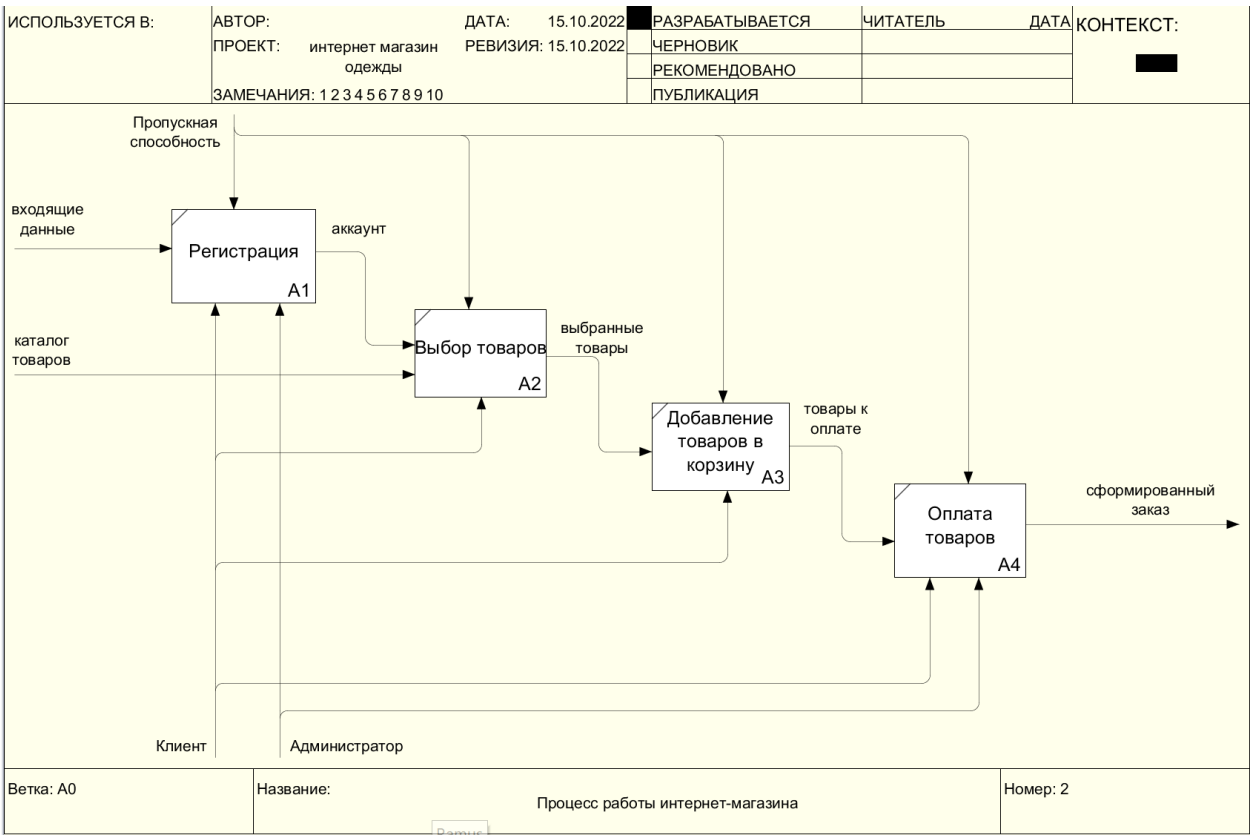

Рисунок 6 – Декомпозиция процесса интернет-магазина

На рисунке показаны четыре процесса:

– регистрация;

– выбор товаров;

– добавление товаров в корзину;

– оплата товаров.

На основе моделирования процесса, необходимо сделать вывод, что для эффективности работы сотрудников интернет-магазина потребуется автоматизировать процессы в интернет-магазине [5].

На рисунке 7 приведена диаграмма потоков данных в нотации DFD подсистемы «Интернет-магазин».

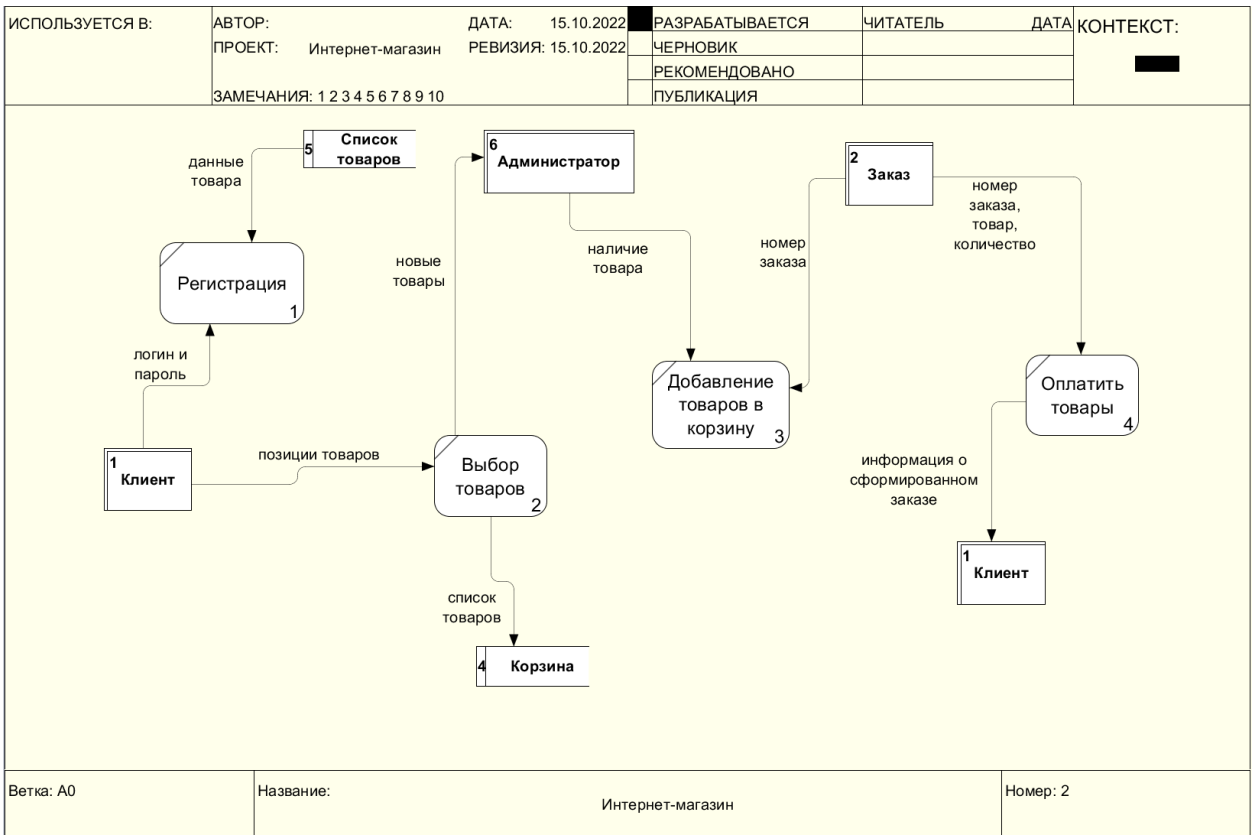

Рисунок 7 – Диаграмма потоков данных в нотации DFD подсистемы «Интернет-магазин»

Словарь данных диаграммы процесса «Интернет-магазин» в нотации DFD приведен в таблице 1.

Таблица 1 – Словарь данных

Процесс		Потоки	
Имя	Определение	Имя	Способ образования
1. Регистрация	Регистрация клиента	Данные заказа	Входной поток процесса из объекта Администратор
2. Выбор товаров	Выбор товаров из каталога	Форма заказа	Входной поток процесса из объекта Администратор

Продолжение таблицы 1

Имя	Определение	Имя	Способ образования
		Данные оплаты за заказ	Выходной поток процесса из объекта
3. Добавление товаров в корзину	Добавление товаров в корзину	Подготовленный заказ	Входной поток процесса из объекта Администратор
		Данные клиента	Входной поток процесса из хранилища БД Клиенты
4. Оплатить товары	Оплатить товары	Оповещение	Входной поток из процесса. Сформировать заказ
		Номер заказа	Входной поток процесса из объекта Администратор
		Документы и заказ	Выходной поток процесса в объект Клиент
		Информация о заказе	Входной поток процесса из хранилища БД Заказы

Входные данные:

- каталог товаров;
- данные клиента;
- наличие товара.

1.3 Постановка задачи

1 Введение

Техническое задание написано на разработку веб-приложение для магазина розничной торговли одежды.

2 Основание для разработки

Веб-приложение принято разрабатывать на основании распоряжения директора магазина розничной торговли ООО «Розница К-1».

3 Назначение

Веб-приложение предназначено для обработки заказов в онлайн-режиме, приеме заказов, продаже и отгрузке товаров по назначению.

4 Требования к программе или программному изделию

4.1 Требования к функциональным характеристикам.

4.1.1 Веб-приложение будет выполнять функции:

– прием онлайн-заказов;
– ведение и обновление информации о товаре на складе;
– производить оплату с помощью предложенных провайдеров;
– вести учет заказов на покупку;
– вести учет закрытых заказов;
– формирование отчетов.

4.1.2 Исходные данные:

– каталог товаров;
– данные о поступившем заказе;

4.1.3 Результаты:

– информация о закрытом заказе;
– информация о проданном товаре.

4.2Требования к надежности.

4.2.1 Приложение должно автоматически проверять критически важные поля для заполнения на отсутствие пустых строк.

4.2.2 Проводить проверку на корректность ввода данных пользователем.

4.3 Требования к составу и параметрам технических средств.

4.3.1 Приложение должно запускаться на ПК с операционной системой Windows.

4.3.2 Минимальная конфигурация:

– тип процессора – Pentium IV и выше;
– объем оперативного запоминающего устройства – от 512 Мб.

4.4 Требования к программной и информационной совместимости.

Приложение должно работать на операционной системе не ниже Windows 10.

5 Требования к программной документации

Документы к программе должны быть выполнены в соответствии с ГОСТ 19.106-78.

К приложению должны быть предоставлены соответствующие документы.

6 Технико-экономические показатели

Показатели выгоды сложно предсказать пока приложение не будет внедрено и протестировано в рабочем режиме.

7 Стадии и этапы разработки

Стадии и этапы разработки согласовываются с заказчиком и оформляются в письменном виде в дополнительном соглашении к техническому заданию.

Таблица 2 – Стадии и этапы разработки

Стадии разработки	Содержание работ	Исполнитель этапа разработки
1. Формирование требований к разработке	1.1 Исследование предметной области	аналитик
	1.2 Обоснование актуальности разработки	аналитик
	1.3 Выбор бизнес-процесса для автоматизации	заказчик
	1.4 Определение требований	заказчик
2. Проектирование	2.1 Выбор модели и методологии проектирования и разработки	проектировщик
	2.2 Выбор инструментов разработки	проектировщик
	2.3 Проектирование программной структуры	проектировщик
	2.4 Инфологическое проектирование	проектировщик
3. Реализация	3.1 Конфигурация инструментов разработки	программист
	3.2 Программная реализация	программист

Продолжение таблицы 2

Стадии разработки	Содержание работ	Исполнитель этапа разработки
	3.3 Реализация базы данных	Программист
4. Тестирование	4.1 Модульное тестирование	программист
	4.2 Интеграционное тестирование	программист
5. Внедрение	5.1 Установка	программист
	5.2 Написание руководства пользователя	программист
	5.3 Написание руководства администратора	программист

8 Порядок контроля и приемки

С помощью тестирования проверяются функции веб-приложения, которые должны отвечать требованиям, указанным в данном техническом задании.

Диаграмма вариантов использования в UML — диаграмма, отражающая отношения между акторами и прецедентами и являющаяся составной частью модели прецедентов, позволяющей описать систему на концептуальном уровне [7].

Также диаграмма вариантов использования показывает ограничения системы, все функции, которые могут осуществлять пользователь в системе.

В данной системе два действующих лица это Клиент и Сотрудник интернет-магазина.

На рисунке 8 показана диаграмма прецендентов.

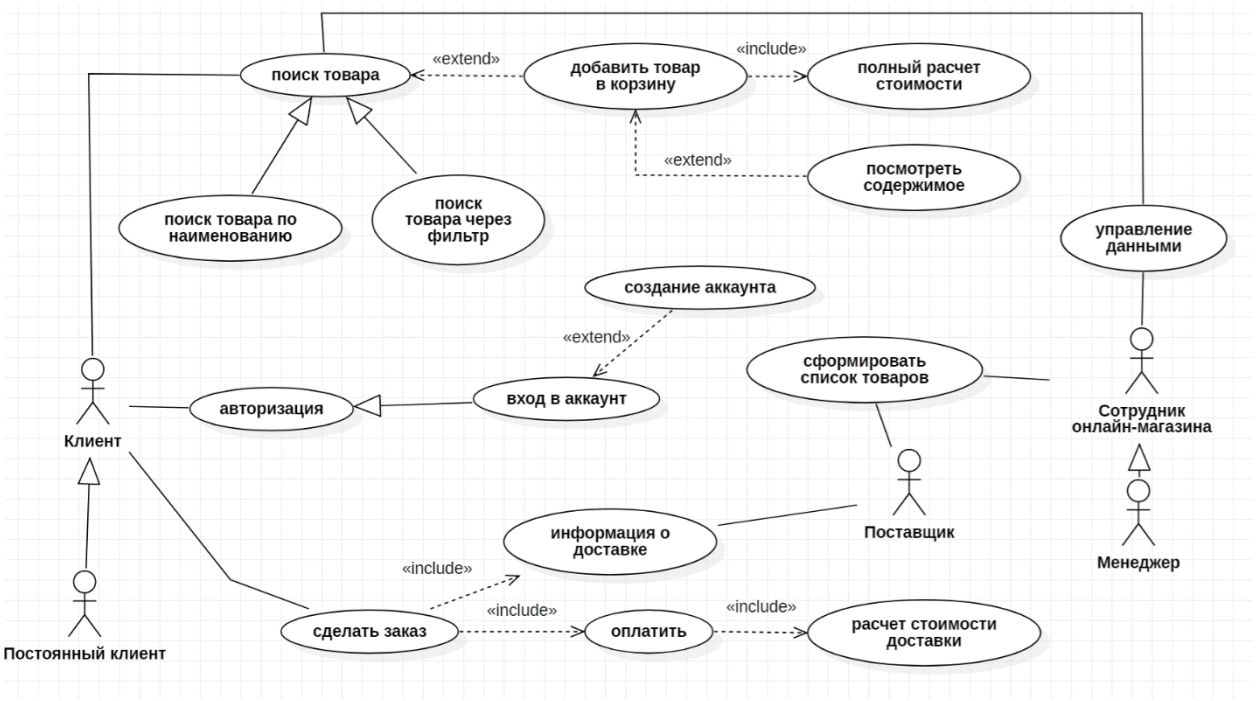

Рисунок 8 – Диаграмма вариантов использования

Функции клиента в веб-приложении:

– авторизация;
– выбор товара;
– оформление заказа;
– оплата заказа.

Функции сотрудника онлайн-магазина:

– управление данными веб-приложения;
– формирование списка товаров.

1.4 Обзор аналогичных программных разработок

Логично будет рассмотреть подобные решения, которые предлагаются на рынке разработчиками, рассмотрим некоторые из них.

RetailCRM [9]. RetailCRM это решение для eCommerce и торговли, которое помогает управлять заказами, клиентами и всеми коммуникациями в едином окне.

Система автоматизирует бизнес-процессы, позволит запустить встроенную Программу лояльности и заняться CRM-маркетингом, чтобы удерживать и возвращать клиентов.

Службы доставки, телефония, платёжные системы, мессенджеры, маркетплейсы и прочее больше 100 функций и модулей.

На рисунке 9 показано окно с заказами.

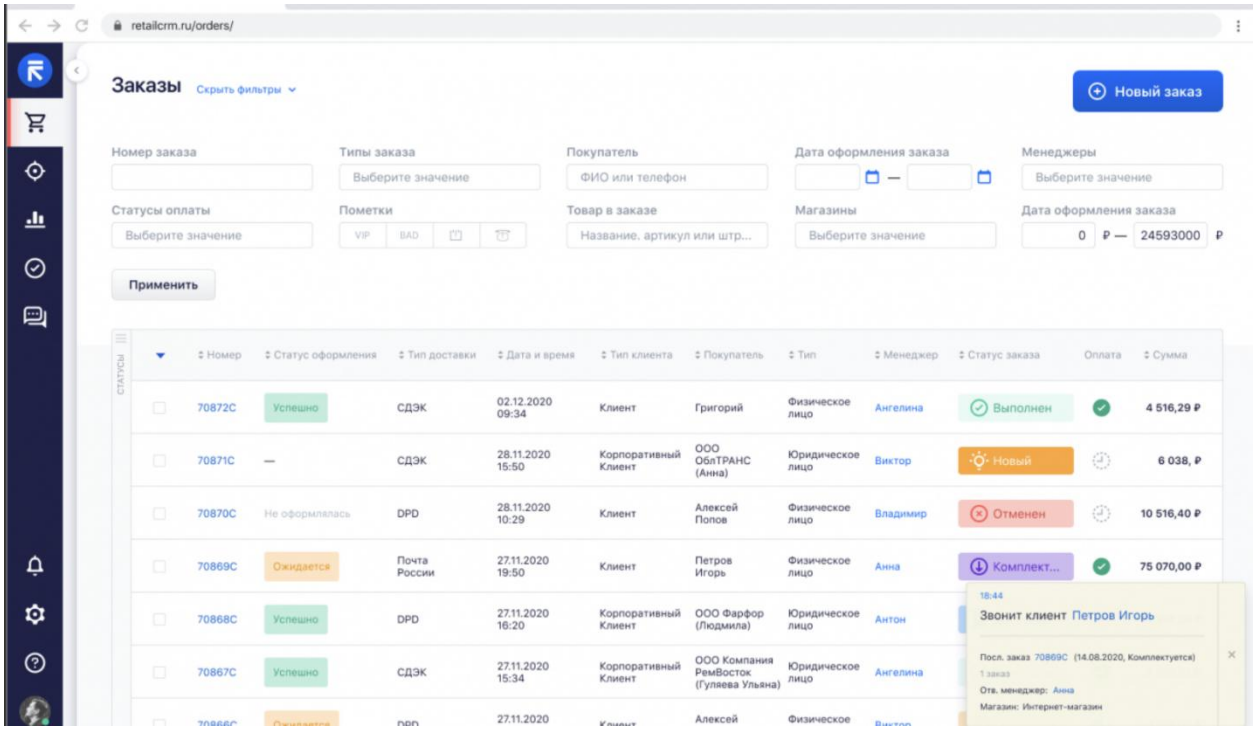

Рисунок 9 – Пример окна с заказами решения RetailCRM

Программное решение «РосБизнесСоф» [10]. Универсальный продукт для управления взаимоотношениями с клиентами, комплексной автоматизации бизнеса, а также успешного развития корпоративной информационной системы. Удобный и понятный интерфейс.

Кроме того, CRM от РосБизнесСофт имеет ряд уникальных разработок в области интеграции программного обеспечения с различными сторонними

приложениями и устройствами (мини-АТС, сканер штрих-кодов, кассовый регистратор и т.д.).

Пример окна со сделками представлен на рисунке 10.

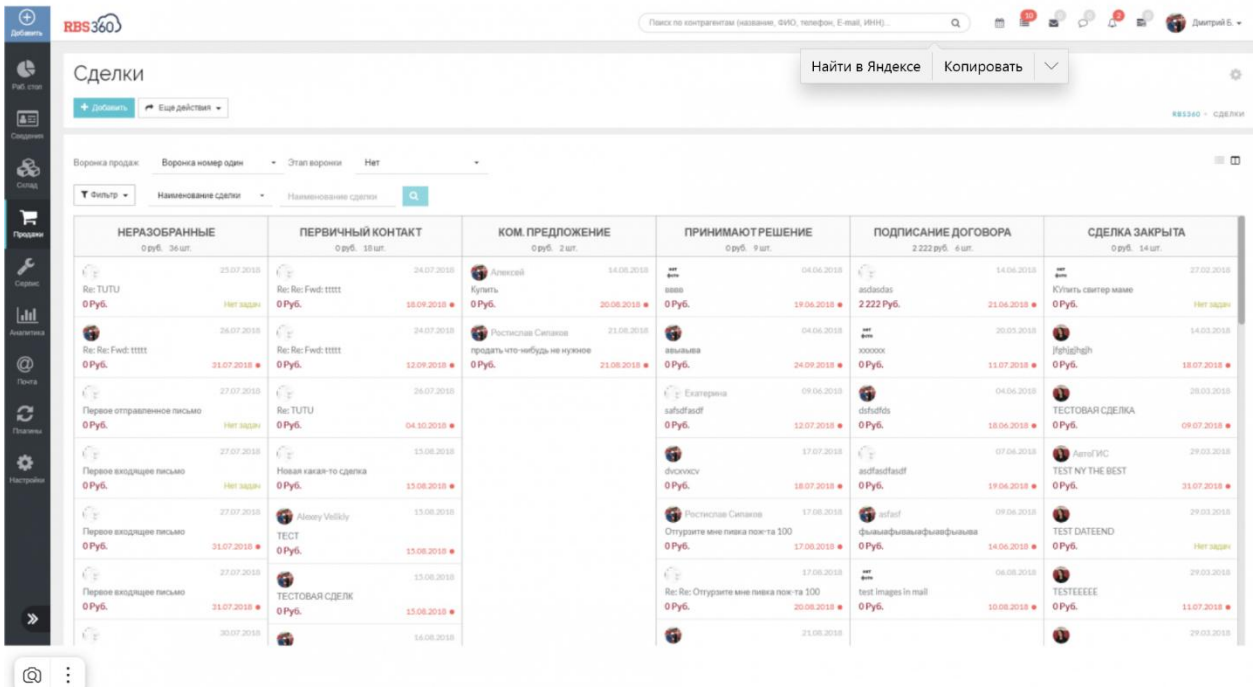

Рисунок 10 – Пример окна со сделками от РосБизнесСофт

Рассмотренные решения больше представляют CRM-систему, которая контактирует с покупателем и предлагает ранее просмотренные товары или отслеживает предпочтения покупателя по критерию просмотров того или иного товара.

Не все рассмотренные приложения соответствуют заявленным требованиям, поэтому для решения поставленной задачи разработка веб-представительства магазина розничной торговли одежды будет разработана собственными силами.

Выводы по первому разделу

В первом разделе работы проведено изучение деятельности магазина розничной торговли, выделены основные бизнес-процессы на основе чего были смоделированы диаграммы в нотации IDEF0, где было показано какие объекты и с какой информацией работают в течении рабочего процесса.

Моделирование позволило провести анализ бизнес-процессов и деятельности всего магазина, выяснилось, что магазин не имеет веб-представительства в сети, что значительно сокращает размер его доходов и аудиторию. Данный анализ подтолкнул руководство магазина к развитию в этом направлении, далее были написаны требования на разработку приложения и рассмотрены аналогичные предложения на рынке.

2 Проектирование веб-приложения магазина

2.1 Выбор технического обеспечения и средств разработки

Рассмотрим языки программирования C++, C# и Java.

C++ – достаточно сильный язык, позволяющий написать практически любую программу, любого направления. С достоинствами язык также имеет и недостатки, которые проявляются как правило в сложности его изучения, не каждый программист может с легкостью изучить язык в короткие сроки и начать писать на нем. Проекты, которые требуют быстрой обработки полученной и информации и получении данных безусловно разрабатываются на языке C++ [11].

Java – объектно-ориентированный язык программирования [12].

Язык Java создавался как язык для написания программного обеспечения для различных бытовых приборов. Самым, пожалуй, большим преимуществом языка является наличие своей виртуальной машины, которая позволяет запускать написанные приложения на любой операционной системе, без корректировок и установки дополнительных компиляторов.

Рассмотрим язык C#. Объектно-ориентированный подход позволяет программистам разрабатывать с помощью C# крупные, но в то же время гибкие, масштабируемые и расширяемые приложения [13].

За последнее время развития it-индустрии самым популярным языков является язык C#. Разработчики Microsoft постоянно совершенствуют его и вносят новинки позволяющие выполнять задачи написанием меньше кода [14].

Разработка веб-приложения будет осуществлена с помощью инструментов СУБД SQL Server, языка программирования C# и среды разработки Visual Studio.

Далее на рисунке показана диаграмма компонентов, которая демонстрирует взаимосвязь модулей приложения между собой как с клиентской частью, так и с частью основной программы [15].

На рисунке 11 показана диаграмма компонентов интернет-магазина.

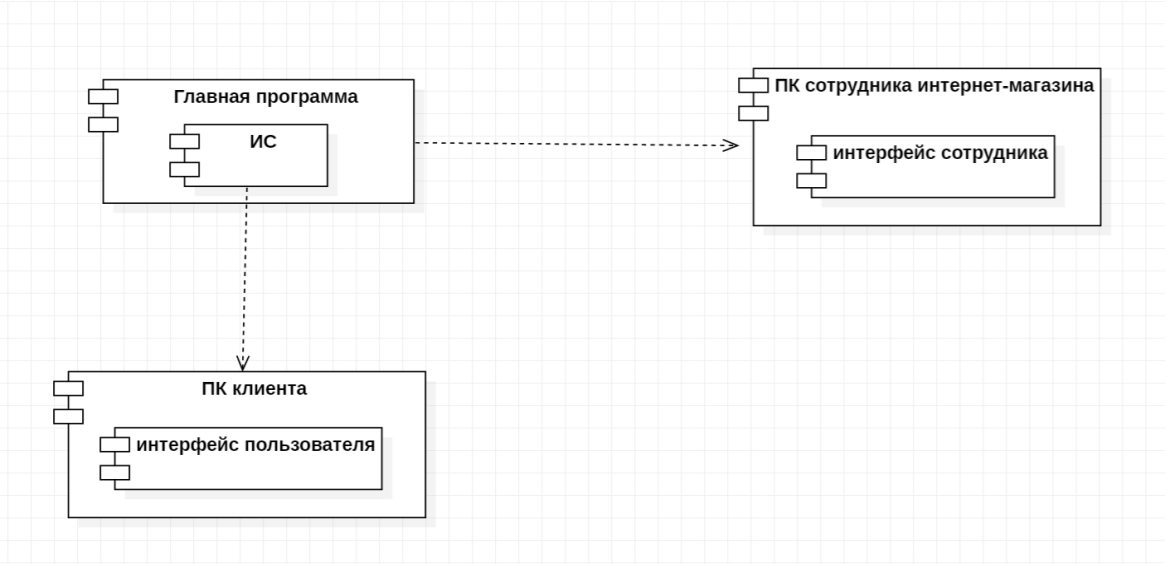

Рисунок 11 – Диаграмма компонентов

Диаграмма развёртывания в UML моделирует физическое развертывание артефактов на узлах. На рисунке 12 показана диаграмма развертывания информационной системы интернет-магазина [16].

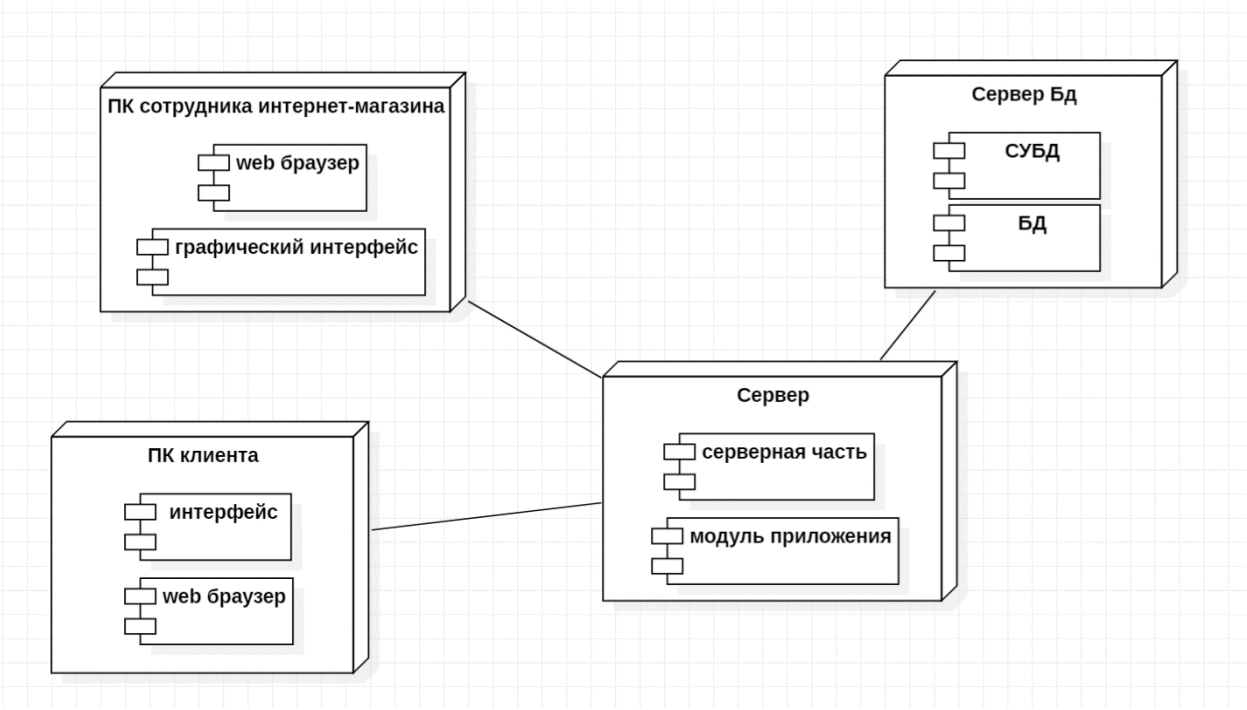

Рисунок 12 – Диаграмма развертывания

2.2 Описание структуры базы данных

Логический уровень. Логическая модель предметной области иллюстрирует сущности, а также их взаимоотношения между собой.

Уникальный идентификатор (ключ) – это минимальный набор атрибутов, предназначенный для уникальной идентификации каждого экземпляра данного типа сущности [20].

Выделим основные сущности, которые потребуются для информационной системы: Клиенты, Товары, Сотрудники, Заказы, Выдача товара.

Сущность Клиенты имеет суррогатный ключ Код клиента, он же и является ключевым атрибутом. Внешних ключей, в сущности, нет.

Сущность Товары имеет суррогатный ключ Код товара, он же и является ключевым атрибутом. Внешних ключей, в сущности, нет.

Сущность Сотрудники имеет первичный ключ Код сотрудника, он же и является ключевым атрибутом. Внешних ключей, в сущности, нет.

Сущность Заказы имеет суррогатный ключ Код заказа, он же и является ключевым атрибутом. Внешними ключами в таблице будут Код клиента, Код товара, что обеспечивает связь с таблицами Клиенты, Товары.

Сущность Выдача товара имеет суррогатный ключ Код выдачи, внешними ключами в таблице будут Код сотрудника, Код заказа, что обеспечивает связь с таблицами Сотрудники, Заказы.

На рисунке 13 показана ER-модель.

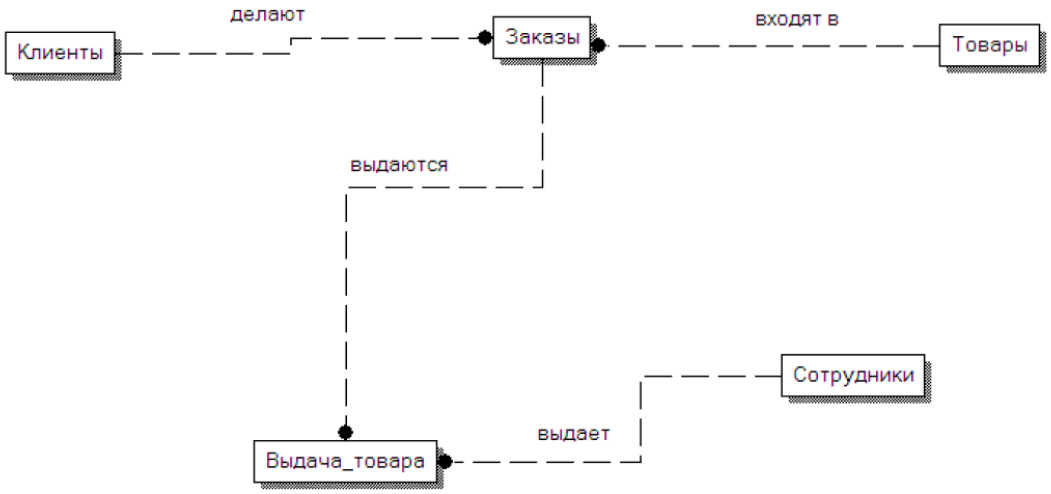

Рисунок 13 – ER-модель данных

После анализа и построения сущностей с атрибутами и первичными ключами определили, что каждая сущность находится в 3НФ.

Третья нормальная форма предполагает, что каждый столбец, не являющийся ключом, должен зависеть только от столбца, который является ключом, то есть должна отсутствовать транзитивная функциональная зависимость.

Каждая сущность имеет первичный ключ и при необходимости внешний ключ, который обеспечивает связь с другой таблицей.

На рисунке 14 показана дополненная модель данных.

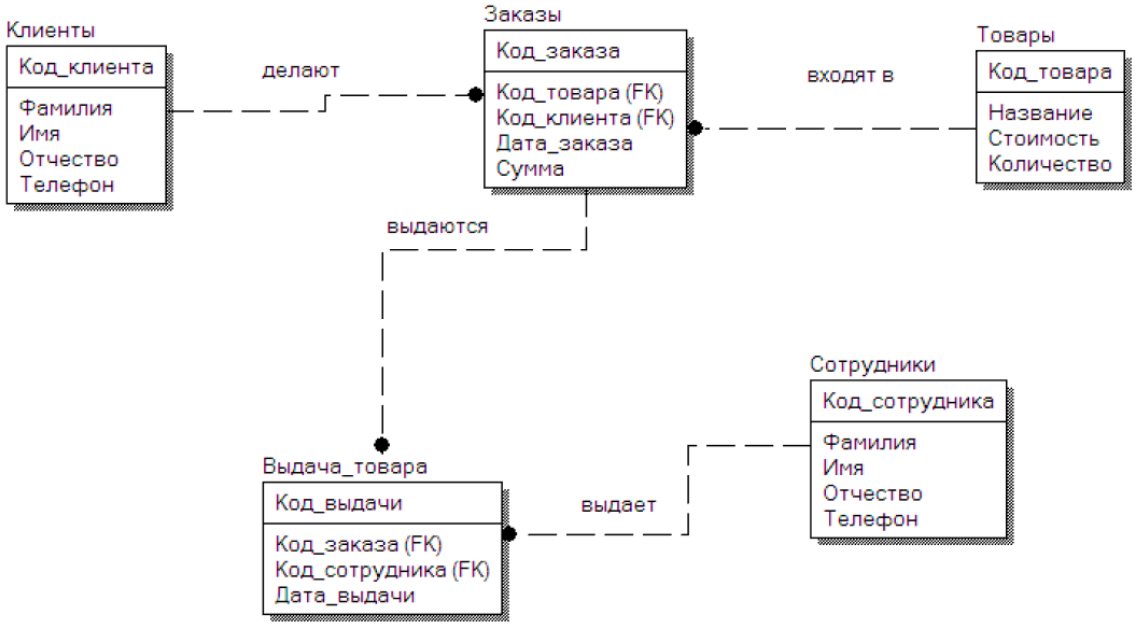

Рисунок 14 - Дополненная модель данных

После определения необходимых сущностей и выбора СУБД, можно перенести нашу диаграмму.

В таблице 3 представим реквизиты сущностей и их ограничения

Таблица 3 – Реквизиты сущностей и их ограничения

Имя сущности	Описание	Тип
Клиенты	Код клиента	Числовой
	Фамилия	25 символов
	Имя	25 символов
	Отчество	25 символов
	Телефон	11 символов
Сотрудники	Код сотрудника	Числовой
	Фамилия	25 символов

Продолжение таблицы 3

Имя сущности	Описание	Тип
	Имя	25 символов
	Отчество	25 символов
	Телефон	11 символов
Товары	Код товара	Числовой
	Название	25 символов
	Стоимость	Денежный
	Количество	Числовой
Заказы	Код заказа	Числовой
	Код товара	Числовой
	Код клиента	Числовой
	Дата заказа	Дата
	Сумма	Денежный
Выдача товара	Код выдачи	Числовой
	Код заказа	Числовой
	Код сотрудника	Числовой
	Дата выдачи	Дата

Выводы по второму разделу

Во втором разделе было проведено проектирование веб-приложение. Выделены основные сущности для хранения информации при осуществлении деятельности магазина. Разработана структура базы данных, построены модели данных.

3 Разработка веб-приложения магазина розничной торговли одеждой

3.1 Описание взаимодействия веб-представительства магазина

Диаграмма взаимодействия или как по-другому ее называют диаграмма последовательности, демонстрирует поэтапное взаимодействие объектов внутри приложения и сообщения между ними [17].

На рисунке 15 показана диаграмма процесса «Создать заказ», в которой участвует клиент, создает заказ на странице заказа, обработчик событий обрабатывает все данные и сохраняет в базу данных.

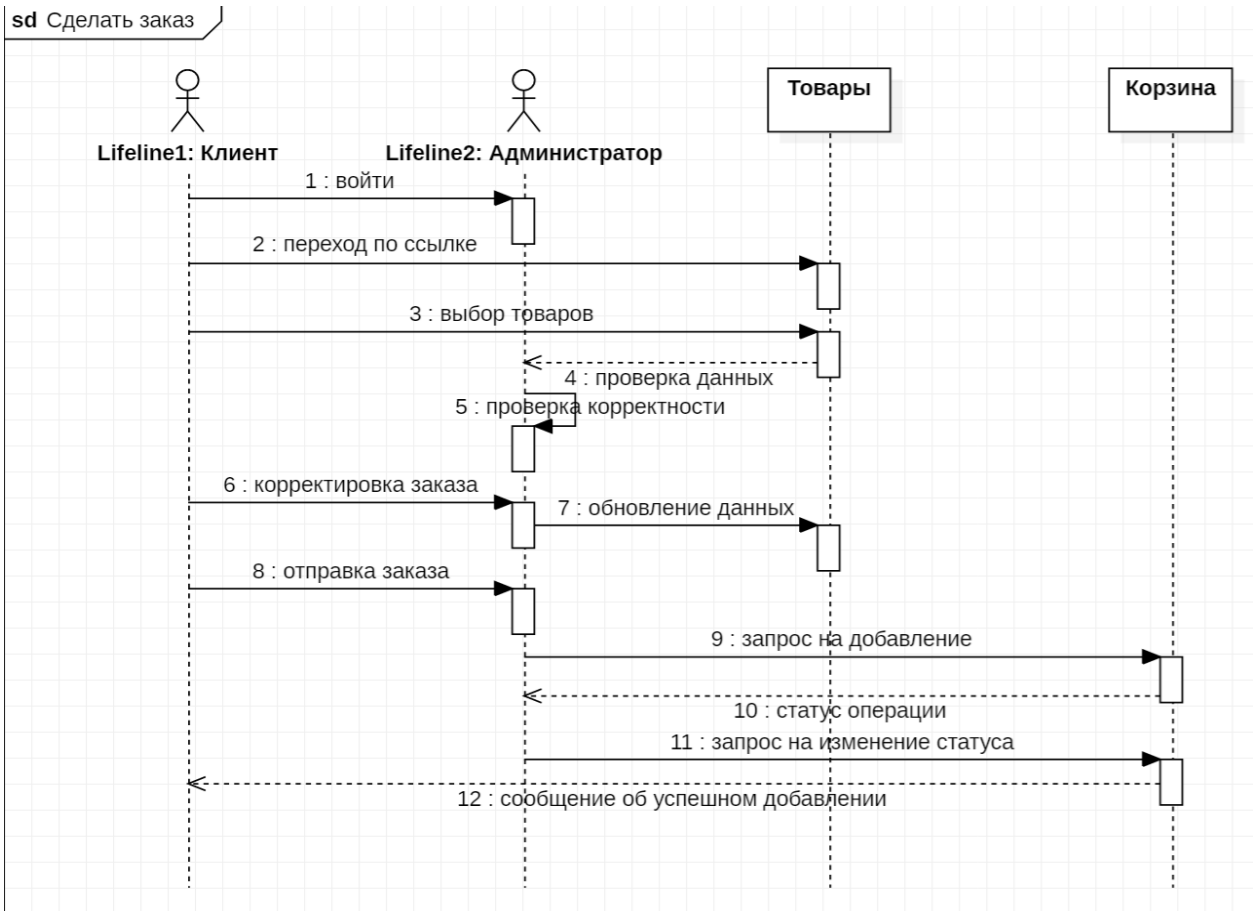

Рисунок 15 – Диаграмма взаимодействия процесса «Создать заказ»

На рисунке 16 показана диаграмма процесса «Регистрация»

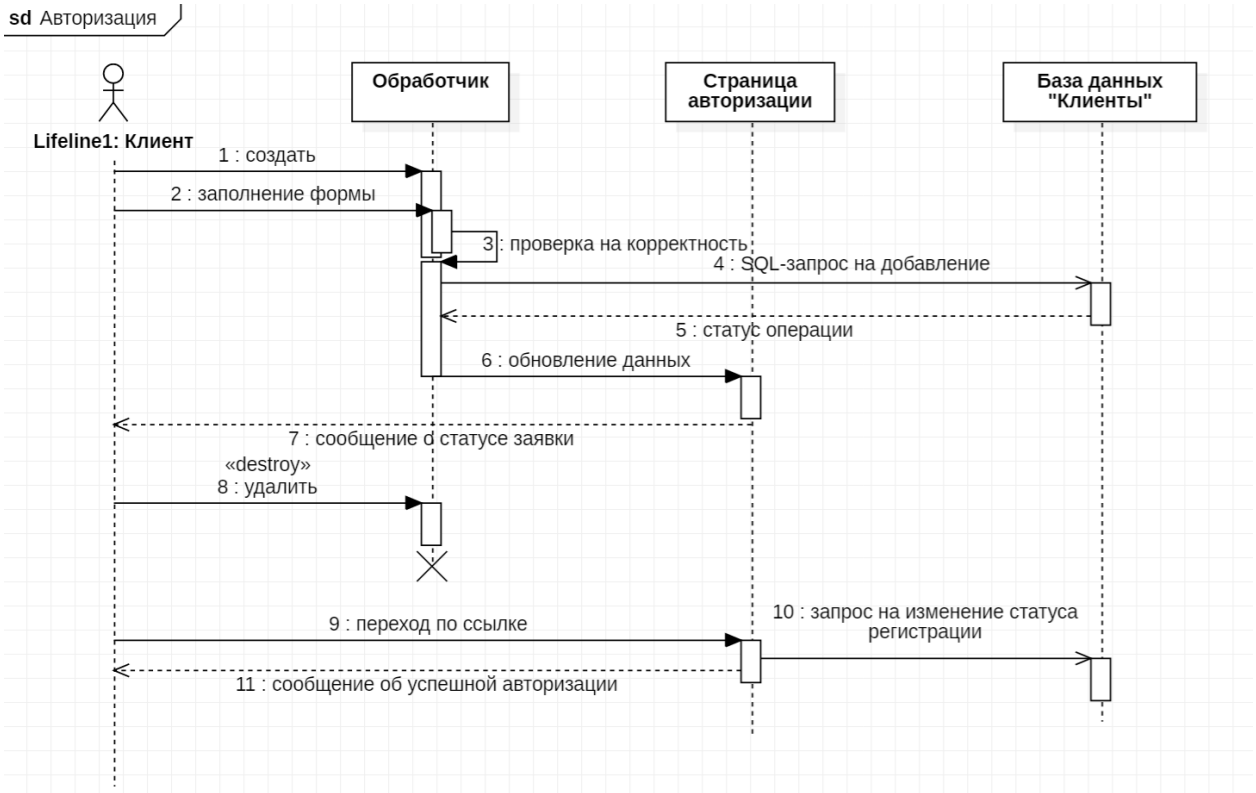

Рисунок 16 – Диаграмма взаимодействия процесса «Авторизация»

Разработка диаграммы классов показывает с точки объектно-ориентированного взаимодействия объектов в приложении [18].

Построим диаграмма классов, на которой будет видно взаимодействие классов, атрибуты класса и методы, рисунок 17.

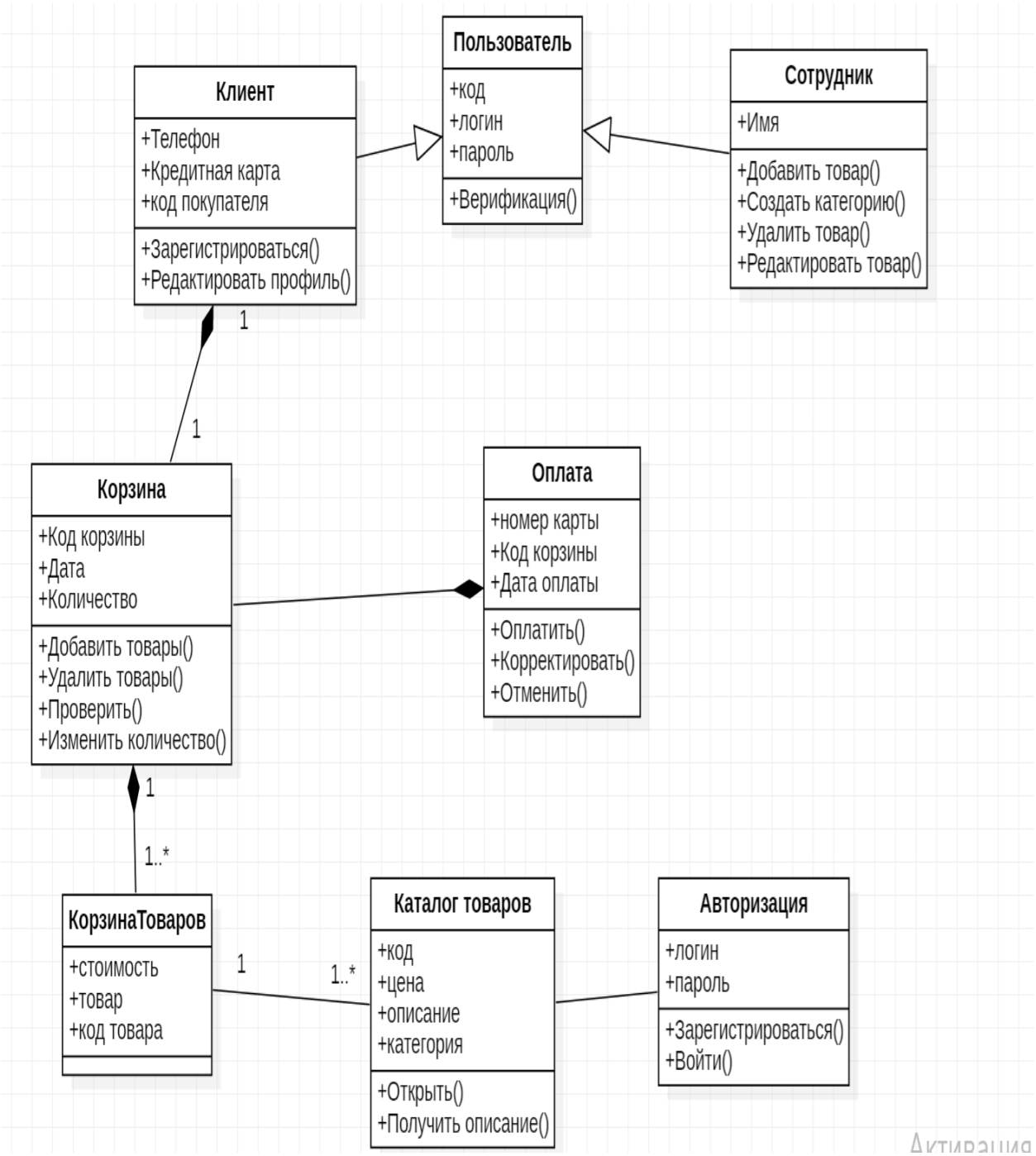

Рисунок 17 – Диаграмма классов

На данном этапе работы были разработаны все необходимые диаграммы, которые помогут команде программистов приступить непосредственно к разработке веб-представительства для предприятия розничной торговли.

3.2 Модели и методологии проектирования и разработки приложения

Методология разработки программного обеспечения – совокупность методов, применяемых на различных стадиях жизненного цикла программного обеспечения и имеющих общий философский подход.

Рассмотрим некоторые из них более подробно.

DevOps – методология активного взаимодействия специалистов по разработке.

Преимущества DEVOPS. Улучшенное сотрудничество и общение Ускоренная доставка программного обеспечения или продуктов Постоянное снижение затрат Улучшенный процесс Ускоренное решение проблем. В мире DevOps нет ни одного волшебного инструмента, который бы соответствовал всем потребностям. Речь идет о выборе правильного инструмента, который соответствует потребностям организации [19].

Недостатки. У подобного подхода много недостатков: изолированность, минимальная связь между отдельными командами разработчиков, тестировщиков, плохая обратная связь между создателями и конечными пользователями. Если на какой-либо стадии создания ПО возникает ошибка, процесс приходится повторять с самого начала. От повторного проектирования с учетом багов до тестирования [20].

Канбан – система, построенная на визуализации процесса выполнения задач команды. Основная идея в этой системе уменьшать количество задач, выполняющихся в данный момент (в колонке «in progress»). В скраме ориентация команды на успешное выполнение спринтов, в Канбане на первом месте задачи. Хорош для проектов, которые в стадии поддержки, где основной функционал уже разработан и остались минимальные доработки и багофиксинг [21].

Основными преимуществами данной методологии являются:

– простота использования;

– наглядность;
– высокая вовлеченность команды в сам процесс;
– высокая гибкость в разработке.

Основными недостатками данной методологии являются:

– нестабильный список задач;
– сложно применять на долгосрочных проектах;
– отсутствие жестких дедлайнов [22].

Для разработки веб-приложения была выбрана методология Канбан, позволяющая на основе уже разработанного ранее ПО доработать функционал.

Работа с веб-приложением начинается после входа на главную страницу. Заказать товар возможно только после прохождения авторизации.

3.3 Конфигурирование инструментов разработки

Для разработки веб-приложения нужно установить среду разработки Visual Studio 2019. Чтобы начать работу необходимо установить программное обеспечение, для этого нужно перейти по адресу visualstudio.microsoft.com [23] и скачать установочный файл, рисунок 18.

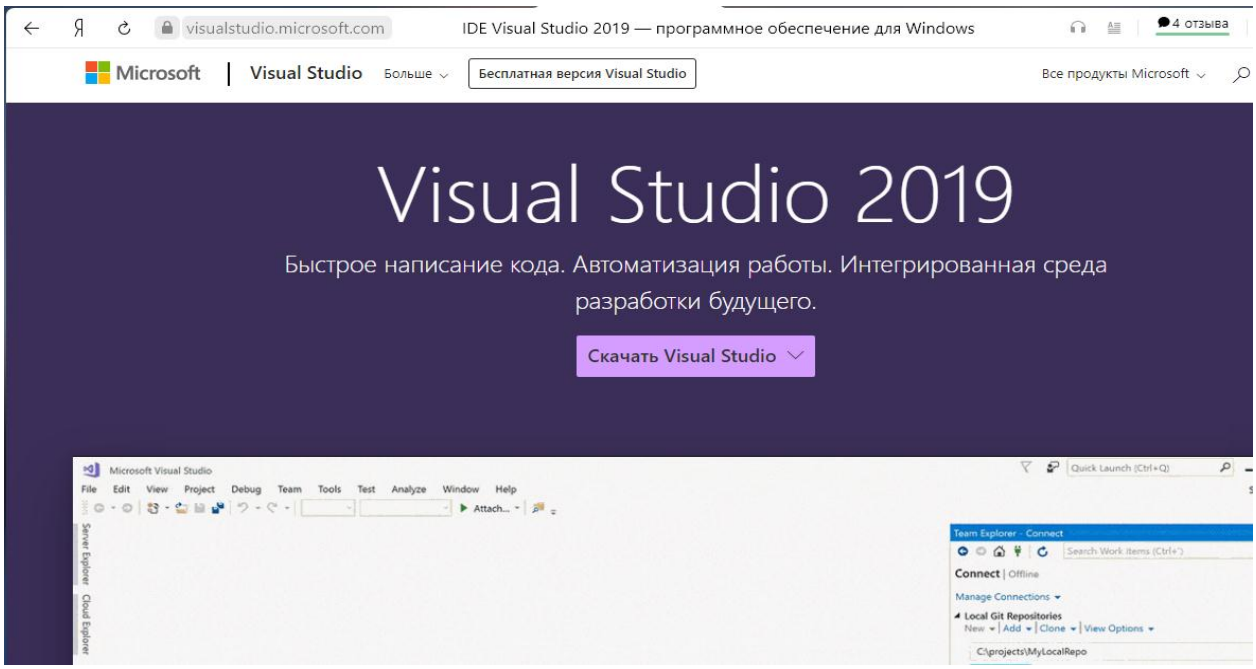

Рисунок 18 – Доступ к программному обеспечению Visual Studio

После загрузки установочного файла, его нужно запустить, после выйдет окно с установкой программы, здесь необходимо нажать кнопку «Install» и программа начнет устанавливаться на диск С в папку Program Files.

После запуска приложения предлагают выбрать пакеты инструментов необходимые для разработки, рисунок 19.

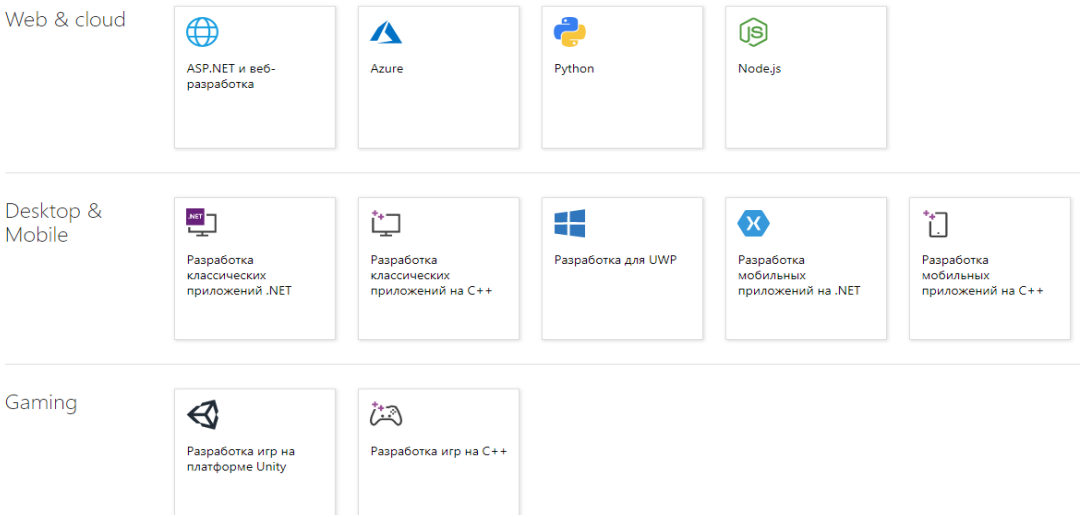

Рисунок 19 – Выбор пакета инструментов

Необходимо выбрать разработку классических приложений .Net и инструмент для веб разработки Node.js, после выбора необходимо установить пакет инструментов [24].

Назвать проект можно как угодно, на свое усмотрение, но обычно называют так, чтобы в дальнейшем было понятно, о чем он.

Также необходимо установить базу данных, для хранения данных была выбрана MS SQL, для установки нужно перейти по ссылке www.microsoft.com и зайти в продукты, там необходимо выбрать SQL Server, разработчики предлагают несколько вариантов приложений, рисунок 20.

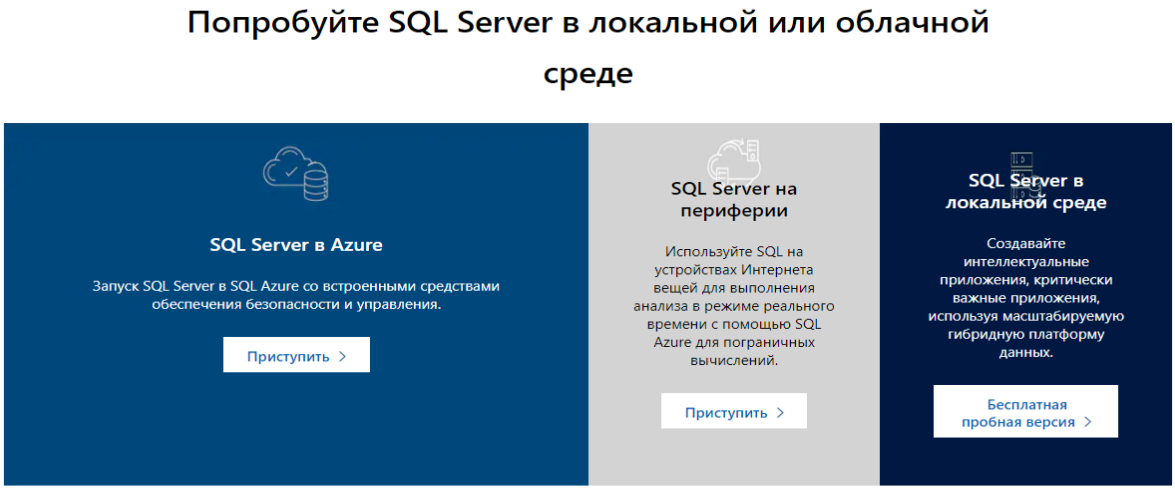

Рисунок 20 – Выбор SQL Server

Выберем SQL Server в локальной среде и установим бесплатную версию базы данных [25].

После установки требуется настроить имя сервера и задать логин и пароль для соединения базы данных.

Так как пока создана локальная база данных развернуть ее можно на локальном сервере localhost, при внедрении приложения на предприятие будет другой сервер для того, чтобы к ней могли обращаться сотрудник интернет-магазина, главный бухгалтер и клиенты.

После ввода пароля загружается меню базы данных, где можно создавать базу, таблицы в базе, писать SQL запросы и т.д., так же можно все это делать напрямую с Visual Studio путем прописывания кода, затем можно зайти в базу и проверить как создалась база или таблицы в ней.

Создадим базу данных [26] и таблицы, необходимые для хранения информации интернет-магазина одежды, пример запроса на создание таблицы Employees показан на рисунке 21.

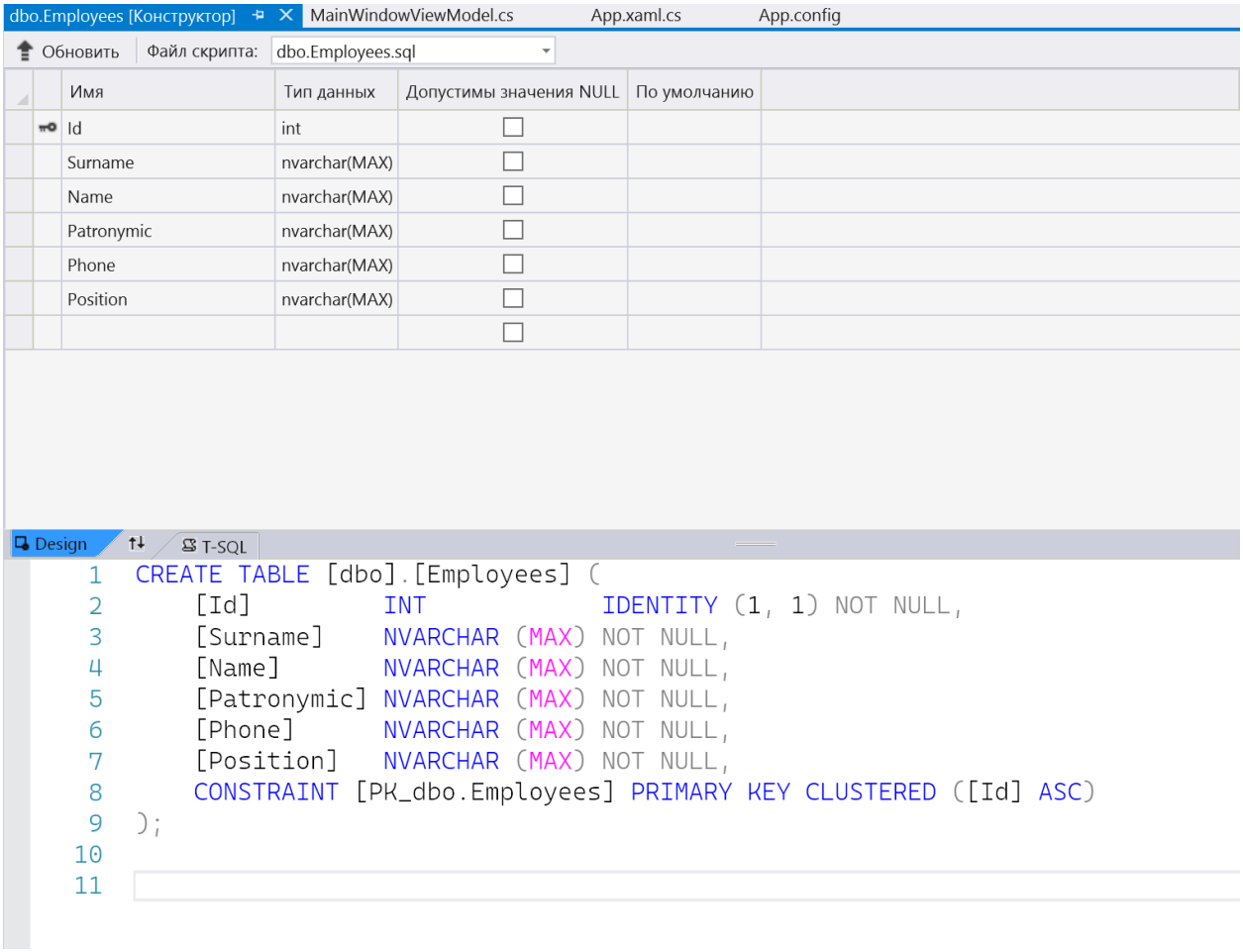

Рисунок 21 – Создание таблиц в СУБД MS SQL Server

3.4 Проектирование интерфейса приложения

Одним из основных требований к веб-интерфейсам является их одинаковый внешний вид и одинаковая функциональность при работе в различных браузерах [27].

Регистрация и авторизация происходит в приложении самостоятельно пользователем.

На рисунке 22 изображен интерфейс интернет-магазина, он позволяет открыть нужный раздел: «Каталог», «Авторизация», «Новинки» или выйти из страницы

Навигация по приложению осуществляется при помощи навигационного меню.

Меню — элемент интерфейса пользователя, позволяющий выбрать одну из нескольких перечисленных опций программы.

Рисунок 22 – Главная страница

В современных программах меню является важнейшим элементом графического интерфейса пользователя [28]. В приложении А приведен программный код представительства.

Меню содержит следующие кнопки:

– каталог,

– новинки,

– категории товаров,

– выход.

Если нажать на кнопку «Каталог», то можно увидеть данные товаров, на рисунке 23 показана форма каталога товаров.

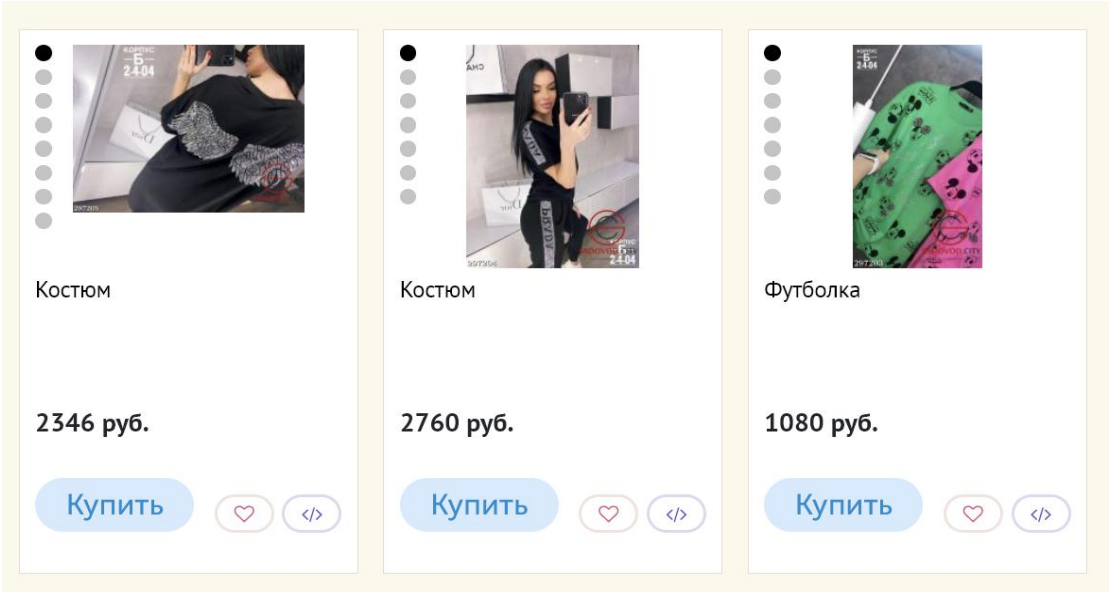

Рисунок 23 – Каталог товаров

Форма создания заказа и добавление товаров показана на рисунке 24.

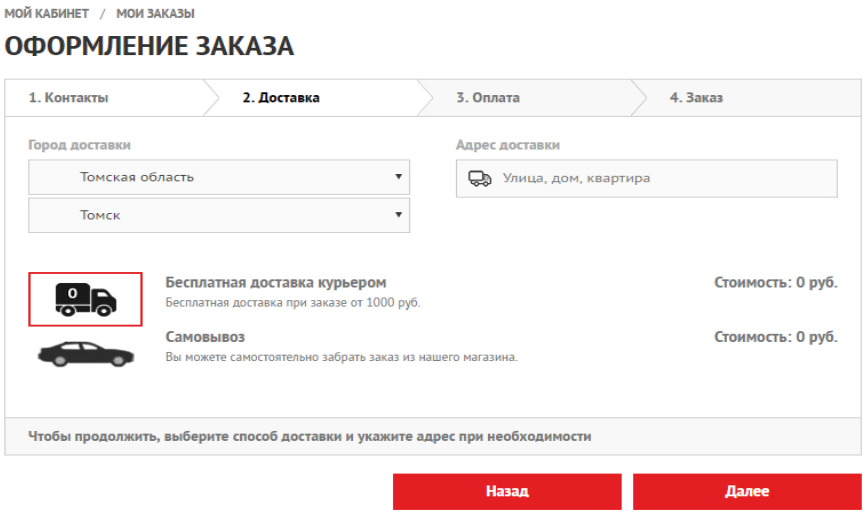

Рисунок 24 – Форма оформления заказа

3.5 Тестирование приложения

Тестирование приложение будет проведено с помощью формулы Миллса, которая даёт возможность оценить первоначальное число ошибок в программе N [29].

$$N = \frac{S \cdot n}{V} \qquad (1)$$

1. Исходные данные

В программу намеренно внесли 7 ошибок. В результате тестирования было обнаружено 8 ошибок, из которых 7 были преднамеренно внесены. Все найденные ошибки исправлены. Перед тестированием предполагалось, что программа содержит не более 2 ошибок. Требуется оценить количество ошибок перед тестированием и степень разработанности программы.

S=7, n=1, V=7

2. Расчёт первоначального количества ошибок N

$$N = \frac{7 \cdot 1}{7} = 1 \qquad (2)$$

По результатам тестирования выходит, что до начала тестирования в программе имелась 1 ошибка.

В данном случае обнаружено меньшее число «собственных» ошибок, чем число предполагаемых ошибок в программе r=2 (n<r).

3. Расчёт меры доверия к модели C.

$$C = \frac{7}{(7 + 2 + 1)} = 0{,}7 \qquad (3)$$

Мера доверия нашему предположению (о том, что в программе не более 2-х ошибок) будет равна 0,7 или 70%, что говорит о надежности прогноза.

Выводы по третьему разделу

В третьем разделе было поведено проектирование взаимодействия приложения, смоделированы диаграммы последовательности, классов. На основе полученных диаграмм была спроектирована и реализована база данных в СУБД MS SQL Server. Затем было разработано веб-приложение магазина одежды, представлена главная страница и форма заказа товара. На завершающем этапе было проведено тестирование с помощью формулы Миллса и выявить первоначальное количество ошибок в приложении.

Заключение

Результатом выпускной квалификационной работы является веб-приложение для магазина розничной торговли одежды, предназначенная для повышения товарооборота компании и представительства своей деятельности в сети Интернет.

В первом разделе работы проведено изучение деятельности магазина розничной торговли, выделены основные бизнес-процессы на основе чего были смоделированы диаграммы в нотации IDEF0, где было показано какие объекты и с какой информацией работают в течении рабочего процесса.

Моделирование позволило провести анализ бизнес-процессов и деятельности всего магазина, выяснилось, что магазин не имеет веб-представительства в сети, что значительно сокращает размер его доходов и аудиторию. Данный анализ подтолкнул руководство магазина к развитию в этом направлении, далее были написаны требования на разработку приложения и рассмотрены аналогичные предложения на рынке.

Во втором разделе было проведено проектирование веб-приложение. Выделены основные сущности для хранения информации при осуществлении деятельности магазина. Разработана структура базы данных, построены модели данных.

В третьем разделе было поведено проектирование взаимодействия приложения, смоделированы диаграммы последовательности, классов. На основе полученных диаграмм была спроектирована и реализована база данных в СУБД MS SQL Server. Затем было разработано веб-приложение магазина одежды, представлена главная страница и форма заказа товара. На завершающем этапе было проведено тестирование с помощью формулы Миллса и выявить первоначальное количество ошибок в приложении.

Цель выпускной квалификационной работы достигнута, все задачи выполнены в полном объеме.

Список используемой литературы

1. Анфилатов В. С. Системный анализ в управлении: Учеб. пособие / В.С. Анфилатов, А.А. Емельянов, А.А. Кукушкин; под ред. А.А. Емельянова. – М.: Финансы и статистика, 2017. – 368 с.

2. База данных. [Электронный ресурс]. – Режим доступа: https://www.bestreferat.ru/referat-226978.html (дата обращения: 14.10.2022).

3. Бесплатная программа для магазина. [Электронный ресурс]. Режим доступа: https://cloudshop.ru/ (дата обращения: 14.10.2022)

4. Буч Г. Введение в UML от создателей языка / Грэди Буч, Джеймс Рамбо, Айвар Якобсон: пер. с англ. — ДМК Пресс, 2015 — 496 с.: ил.

5. Вигерс, Карл. Разработка требований к программному обеспечению = Software Requirements: пер. с англ.; 3-е издание, дополненное / Карл Виггерс, Джой Битти — СПб.: Издательство «BHV», 2020. — 736 с.: ил.

6. Возможности – Большая птица [Электронный ресурс]. Режим доступа: https://bigbird.ru/features (дата обращения: 14.10.2022).

7. Воронин Б. А. Системный анализ: методические указания по выполнению курсового проекта для студентов, обучающихся с применением дистанционных образовательных технологий / Б. А. Воронин. – Томск: ФДО ТУСУР, 2021. – 82 с.

8. Диаграмма классов. [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Диаграмма_классов (дата обращения: 14.10.2022).

9. Информационная система – ИС [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Информационная_система (дата обращения: 14.10.2022).

10. Информатика. Базовый курс: Учебник. / Симонович С.В. - СПб.: Питер, 2015. - 640с.

11. Ковшов А.В. Теория систем и системный анализ: Учебное методическое пособие: Томский межвузовский центр дистанционного образования, 2021. – 45 с.

12. Кон Майк. Пользовательские истории. Гибкая разработка программного обеспечения = User Stories Applied: For Agile Software Development: пер. с англ. / Майк Кон. — М.: Издательство «Вильямс», 2018 — 256 с.: ил.

13. Логическая структура БД. [Электронный ресурс]. – Режим доступа: https://intuit.ru/studies/courses/5/5/lecture/124?page=4/ (дата обращения: 14.10.2022).

14. Магазин – Что такое магазин? [Электронный ресурс]. URL: https://ru.wikipedia.org/wiki/Магазин (дата обращения: 14.10.2022).

15. Методология IDEF0. [Электронный ресурс]. – Режим доступа: https://www.sites.google.com/site/anisimovkhv/learning/pris/lecture/tema6/tema6_2 (дата обращения: 14.10.2022)

16. Объектная декомпозиция. [Электронный ресурс]. URL: https://helpiks.org/6-8507.html (дата обращения: 14.10.2022).

17. Основы построения автоматизированных систем: Учебник/Гвоздева В.А., Лаврентьева И.Ю. – М.: ИД «ФОРУМ»: ИНФРА – М, 2019. – 320 с.

18. Павлов Е. В. Проектирование программных систем: лабораторный практикум: учебное пособие / Е. В. Павлов. — СПб.: ГУАП, 2020.

19. Паттон Джефф. Пользовательские истории. Искусство гибкой разработки программного обеспечения = User Story Mapping: Discover the Whole Story, Build the Right Product: пер. с англ. / Джефф Паттон. — СПб.: Издательство «Питер», 2019 — 288 с.

20. Проектирование баз данных. [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Проектирование_баз_данных (дата обращения: 14.10.2022).

21. Проектирование экономических информационных систем: Учебник/Г.Н. Смирнова, А.А. Сорокин, Ю.Ф. Тельнов. Под ред. Ю.Ф. Тельнова. - М.: Финансы и статистика, 2018. - 512 с.

22. Силич, М. П. Теория систем и системный анализ: Учебное пособие [Электронный ресурс] / М. П. Силич, В. А. Силич. — Томск: ТУСУР, 2021. — 276 с.

23. Товароучетная система для магазина. [Электронный ресурс]. Режим доступа: [Электронный ресурс]. Режим доступа: https://www.ekam.ru/ (дата обращения: 14.10.2022).

24. GeekBrains – Язык программирования C#. [Электронный ресурс]. - Режим доступа: https://geekbrains.ru/posts/yazyk-programmirovaniya-c-sharp-istoriya-specifika-mesto-na-rynke (дата обращения: 14.10.2022).

25. Helpiks.org: Достоинства и недостатки языка [Электронный ресурс]. - Режим доступа: https://helpiks.org/6-21879.html (дата обращения: 14.10.2022).

26. Java-энциклопедия языков программирования: Java. [Электронный ресурс]. – Режим доступа: http://progopedia.ru/language/java/ (дата обращения: 14.10.2022).

27. UML Use Case Diagrams [Электронный ресурс]. — uml-diagrams.org, 2009-2020. — URL: https://www.uml-diagrams.org/use-case-diagrams.html (дата обращения: 14.10.2022).

28. Visual Studio Express, обзор, плюсы и минусы программы [Электронный ресурс]. – Режим доступа: https://8d9.ru/program/visual-studio-express (дата обращения: 14.10.2022).

29. What is Use Case Specification? [Электронный ресурс]. — Visual Paradigm, 2020. — URL: https://www.visual-paradigm.com/guide/use-case/what-is-use-case-specification/ (дата обращения: 14.10.2022).

Приложение А

Фрагменты исходного кода

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows;

namespace Market
{
   internal class DisplayRootRegistry
   {

      public Dictionary<Type, Type> vmToWindowMapping = new Dictionary<Type, Type>();

      public void RegisterWindowType<VM, Win>() where Win : Window, new() where VM : class
      {
         var vmType = typeof(VM);
         if (vmType.IsInterface)
            throw new ArgumentException("Cannot register interfaces");
         if (vmToWindowMapping.ContainsKey(vmType))
            throw new InvalidOperationException(
               $"Type {vmType.FullName} is already registered");
         vmToWindowMapping[vmType] = typeof(Win);
      }

      public void UnregisterWindowType<VM>()
      {
         var vmType = typeof(VM);
         if (vmType.IsInterface)
            throw new ArgumentException("Cannot register interfaces");
         if (!vmToWindowMapping.ContainsKey(vmType))
            throw new InvalidOperationException(
               $"Type {vmType.FullName} is not registered");
         vmToWindowMapping.Remove(vmType);
      }

      public Window CreateWindowInstanceWithVM(object vm)
      {
         if (vm == null)
            throw new ArgumentNullException("vm");
         Type windowType = null;

         var vmType = vm.GetType();
         while (vmType != null && !vmToWindowMapping.TryGetValue(vmType, out windowType))
            vmType = vmType.BaseType;

         if (windowType == null)
            throw new ArgumentException(
               $"No registered window type for argument type {vm.GetType().FullName}");

         var window = (Window)Activator.CreateInstance(windowType);
```

```
window.DataContext = vm;
            return window;
        }


        public Dictionary<object, Window> openWindows = new Dictionary<object, Window>();
        public void ShowPresentation(object vm)


    {
            if (vm == null)
                throw new ArgumentNullException("vm");
            if (openWindows.ContainsKey(vm))
                throw new InvalidOperationException("UI for this VM is already displayed");
            var window = CreateWindowInstanceWithVM(vm);
            window.Show();
            openWindows[vm] = window;
        }

        public void HidePresentation(object vm)
        {
            Window window;
            if (!openWindows.TryGetValue(vm, out window))
                throw new InvalidOperationException("UI for this VM is not displayed");
            window.Close();
            openWindows.Remove(vm);
        }

        public async Task ShowModalPresentation(object vm)
        {
            var window = CreateWindowInstanceWithVM(vm);
            window.WindowStartupLocation = WindowStartupLocation.CenterScreen;
            await window.Dispatcher.InvokeAsync(() => window.ShowDialog());
        }

    }
}



using Market.Model;
using System;
using System.Collections.Generic;
using System.Collections.ObjectModel;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Windows.Input;

namespace Market.ViewModel
{
    internal class MainWindowViewModel: ViewModelBase
    {
```

```
DisplayRootRegistry displayRootRegistry;


        private ObservableCollection<Sale> sales;

        public ObservableCollection<Sale> Sales


{
            get
            {


return sales;
            }
            set
            {
                sales = value;
                OnPropertyChanged("Sales");
            }
        }

        public DateTime Date { get; set; }

        public MainWindowViewModel(DisplayRootRegistry displayRootRegistry)
        {
            this.displayRootRegistry = displayRootRegistry;

            Date = DateTime.Now;

        }

        #region updateCommand

        private RelayCommand updateCommand;
        public ICommand UpdateCommand
        {
            get
            {
                if (updateCommand == null)
                {
                    updateCommand = new RelayCommand(ExecuteUpdateCommand,
CanExecuteUpdateCommand);
                }
                return updateCommand;
            }
        }

        private void ExecuteUpdateCommand(object parameter)
        {
            Sales = new ObservableCollection<Sale>(SaleService.GetSalesByDate(Date));
        }

        private bool CanExecuteUpdateCommand(object parameter)
```

```
        {
            if (Date != new DateTime(1, 1, 1, 0, 0, 0))
            {
                return true;
            }
            else
            {
                return false;
            }
        }


class RelayCommand : ICommand
    {
        private Action<object> execute;
        private Predicate<object> canExecute;


        public RelayCommand(Action<object> execute, Predicate<object> canExecute = null)
        {
            if (execute == null)
            {
                throw new ArgumentNullException("execute");
            }
            this.execute = execute;
            this.canExecute = canExecute;
        }

        public bool CanExecute(object parameter)
        {
            return this.canExecute == null ? true : this.canExecute.Invoke(parameter);
        }

        public event EventHandler CanExecuteChanged
        {
            add
            {
                CommandManager.RequerySuggested += value;
            }
            remove
            {
                CommandManager.RequerySuggested -= value;
            }
        }

        public void Execute(object parameter)
        {
            this.execute.Invoke(parameter);
        }

    }


class ViewModelBase : INotifyPropertyChanged
```

```
{
    public ViewModelBase()
    {

    }

    public event PropertyChangedEventHandler PropertyChanged;
    public virtual void OnPropertyChanged(string propertyName)
    {
        PropertyChangedEventHandler handler = this.PropertyChanged;
        if (handler != null)
        {
            handler.Invoke(this, new PropertyChangedEventArgs(propertyName));
        }
    }
}



namespace Market.Model
{
    class Employee
    {

        public int Id { get; set; }

        public string LastName { get; set; }

        public string FirstName { get; set; }

        public string MiddleName { get; set; }

        public Position Position { get; set; }

        public string EmploymentDate { get; set; }

    }
}


namespace Market.Model
{
    class Product
    {
        public int Id { get; set; }

        public string Name { get; set; }

        public string Quantity { get; set; }

        public Decimal UnitPrice { get; set; }

        public Manufacturer Manufacturer { get; set; }

        public Provider Provider { get; set; }
```

```
        public ProductCategory ProductCategory { get; set; }

    }
}



using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;
using System.Configuration;
using System.Data.SqlClient;
using System.Data;

namespace Market.Model
{
    class SaleService
    {
        static string connectionString =
ConfigurationManager.ConnectionStrings["DefaultConnection"].ConnectionString;

        public static List<Sale> GetSalesByDate(DateTime date)
        {

            List<Sale> sales = new List<Sale>();
            //string sql = "SELECT * FROM Продажи";

            //DateTime date = new DateTime(2020, 11, 12);

            string sql = "SELECT " +
             "Sales.[Код продажи] As SalesId," +
             "Product.[Код товара] As ProductId," +
             "Product.Название As ProductName," +
             "Product.Количество As ProductQuantity," +
             "Product.[Цена за 1 ед] As ProductUnitPrice," +
             "Manufacturer.[Код производителя] As ManufacturerId," +
             "Manufacturer.Название As ManufacturerName," +
             "Manufacturer.Телефон As ManufacturerPhone," +
             "Manufacturer.Город As ManufacturerCity," +
             "ProductProvider.[Код поставщика] As ProviderId," +
             "ProductProvider.Наименование As ProviderName," +
             "ProductProvider.Телефон As ProviderPhone," +
             "ProductCategory.[Код категории] As ProductCategoryId," +
             "ProductCategory.Название As ProductCategoryName," +
             "Employee.[Код сотрудника] As EmployeeId," +
             "Employee.Фамилия As EmployeeLastName," +
             "Employee.Имя As EmployeeFirstName," +
             "Employee.Отчество As EmployeeMiddleName," +
             "Position.[Код должности] As PositionId," +
             "Position.Название As PositionName," +
             "Position.Оклад As PositionSalary," +
             "Employee.[Дата приема на работу] As EmployeeEmploymentDate," +
             "Sales.[Дата продажи] As SaleDate," +
```

```
 "Sales.Сумма As SaleAmount " +
            "FROM [dbo].[Продажи] AS Sales " +
            "JOIN [dbo].[Сотрудники] AS Employee ON Sales.[Сотрудник] = Employee.[Код
сотрудника] " +
            "JOIN [dbo].[Должность] AS Position ON Employee.[Код сотрудника] = Position.[Код
должности] " +
            "JOIN [dbo].[Товары] AS Product ON Sales.[Код продажи] = Product.[Код товара] " +
            "JOIN [dbo].[Производители] AS Manufacturer ON Product.Производитель =
Manufacturer.[Код производителя] " +
            "JOIN [dbo].[Поставщики] AS ProductProvider ON Product.Поставщик =
ProductProvider.[Код поставщика] " +
            "JOIN [dbo].[Категория] AS ProductCategory ON Product.Категория = ProductCategory.[Код
категории] " +
            "WHERE Sales.[Дата продажи] = @date";

        string tableName = "Sale";

        using (SqlConnection connection = new SqlConnection(connectionString))
        {
            connection.Open();

            Sale sale;


            SqlCommand command = new SqlCommand(sql, connection);

            SqlParameter dateParameter = new SqlParameter("@date", date);

            command.Parameters.Add(dateParameter);

            SqlDataAdapter adapter = new SqlDataAdapter(command);


            DataSet ds = new DataSet();

            adapter.Fill(ds, tableName);

            foreach (DataTable dt in ds.Tables)
            {
                foreach (DataRow row in dt.Rows)
                {
                    sale = new Sale()
                    {
                        Id = (int)row["SalesId"],
                        Product = new Product()
                        {
                            Id = (int)row["ProductId"],
                            Name = (string)row["ProductName"],
                            Quantity = (string)row["ProductQuantity"],
                            UnitPrice = (Decimal)row["ProductUnitPrice"],
                            Manufacturer = new Manufacturer()
                            {
                                Id = (string)row["ManufacturerId"],
                                Name = (string)row["ManufacturerName"],
```

```
Phone = (string)row["ManufacturerPhone"],
                        City = (string)row["ManufacturerCity"]
                    },
                    Provider = new Provider()
                    {
                        Id = (int)row["ProviderId"],
                        Name = (string)row["ProviderName"],
                        Phone = (string)row["ProviderPhone"]
                    },
                    ProductCategory = new ProductCategory()
                    {
                        Id = (int)row["ProductCategoryId"],
                        Name = (string)row["ProductCategoryName"]
                    }
                },
                Employee = new Employee()
                {
                    Id = (int)row["EmployeeId"],
                    LastName = (string)row["EmployeeLastName"],
                    FirstName = (string)row["EmployeeFirstName"],
                    MiddleName = (string)row["EmployeeMiddleName"],
                    Position = new Position()
                    {
                        Id = (int)row["PositionId"],
                        Name = (string)row["PositionName"],
                        Salary = (string)row["PositionSalary"]
                    },
                    EmploymentDate = (string)row["EmployeeEmploymentDate"]
                },
                Date = (DateTime)row["SaleDate"],
                Amount = (Decimal)row["SaleAmount"]
            };
            sales.Add(sale);
        }
    }
}

return sales;
}

}
}
```