МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт информационных и телекоммуникационных технологий

Кафедра "Программная инженерия"

Туремуратов Ж.М.

Создание и разработка интернет системы покупок.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту

Специальность 5В060200 – Информатика

Алматы 2019

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт информационных и телекоммуникационных технологий

Кафедра "Программная инженерия"

ДОПУЩЕН К ЗАЩИТЕ
Заведующий кафедрой ПИ
канд. техн. наук, доцент,
ассистент-профессор
_____ Р. Юнусов
"__" _____ 2019 г.

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
к дипломному проекту

На тему: "Создание и разработка интернет системы покупок"

по специальности 5В060200 – Информатика

Выполнил Туремуратов Ж.М.

Научный руководитель
лектор
_____ Г.А.Омарова
"__" _____ 2019 г.

Алматы 2019

ГРАФИК

подготовки дипломного проекта

Наименование разделов, перечень разрабатываемых вопросов	Сроки представления научному руководителю и консультантам	Примечание
1. Анализ предметной области, разработка технического задания на проектирование программного комплекса	20.05.19	нет
2. Реализация сервеной части приложения	20.05.19	нет
3. Разработка пользовательского интерфейса, реализация функционала корзины	20.05.19	нет
4. Реализация панели администратора	20.05.19	нет
5. Реализация взаимодействия с API и авторизации для администратора. Тестирование приложения	20.05.19	нет
6. Написание пояснительной записки к дипломному проекту	20.05.19	нет

Подписи

консультантов и нормоконтролера на законченный дипломный проект
с указанием относящихся к ним разделов проекта

Наименования разделов	Консультанты, И.О.Ф. (уч. степень, звание)	Дата подписания	Подпись
Программное обеспечение	З.М. Өмірбекова Ассистент	20.05.19	
Нормоконтролер	А.А. Иржанова Тьютор	20.05.19	

Научный руководитель _____ Г.А.Омарова

Задание принял к исполнению обучающийся _____ Ж.М.Туремуратов

Дата "20" января 2019 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

СӘТБАЕВ УНИВЕРСИТЕТІ

Институт информационных и телекоммуникационных технологий

Кафедра "Программная инженерия"

5В060200 – Информатика

УТВЕРЖДАЮ
Заведующий кафедрой ПИ
канд. техн. наук, доцент,
ассистент-профессор
_____ Р. Юнусов
"22" мая 2019 г.

ЗАДАНИЕ
на выполнение дипломного проекта

Обучающемуся *Туремуратову Жасарал Маратовичу*
Тема: *Создание и разработка интернет системы покупок.*
Утверждена *приказом проректора по академической работе № 1162 - б от "16" октября 2018 г.*
Срок сдачи законченного проекта *" 22 " мая 2019 г.*
Исходные данные к дипломному проекту: *Техническое задание, описание необходимых функций проекта.*
Перечень подлежащих разработке в дипломном проекте вопросов:
а) реализация функционала web-приложения для покупки одежды;
б) проектирование и разработка пользовательского интерфейса модуля;
в) разработка, отладка, тестирование программного комплекса.
Перечень графического материала (с точным указанием обязательных чертежей): *представлены 19 рисунков, 1 таблица.*
Рекомендуемая основная литература: *из 6 наименований.*

АННОТАЦИЯ

Целью данного дипломного проекта была разработка системы для совершения онлайн покупок, которая автоматизирует работу магазина одежды.

Чтобы достичь этой цели были проделаны следующие работы: изучение предметной области, и в свою очередь на основе результатов изучения формирование технического задания для разработки программного обеспечения.

В результате проделанной работы были разработаны базы данных на основе документоориентированной системы управления базами данных MongoDB, сервер на основе Node.js + Express, веб-приложение на основе React.js.

Итогом данного дипломного проекта является интернет сервис для покупки и заказа одежды

Данный дипломный проект содержит пояснительную записку объемом 40 страниц, включая 19 иллюстраций, 1 таблица, список литературы из 6 наименований, 3 приложения.

ANNOTATION

The aim of this graduation project was to develop a system for making online purchases that automates the work of a clothing store.

In the introduction of the thesis describes the purpose of the project and the tasks performed.

To achieve this goal, the following work was carried out: the study of the subject area, and in turn, based on the results of the study, the formation of a technical task for development software.

As a result of this work, databases were developed based on the MongoDB document-oriented database management system, a server based on Node.js + Express, a web application based on React.js.

The result of this graduation project is an Internet service for the purchase and order of clothing.

This thesis project contains an explanatory note volume 40 pages, including 19 illustrations, 1 tables, bibliography of 6 titles, 3 applications.

АҢДАТПА

Дипломдық жобаның мақсаты киім дүкенінің жұмысын автоматтандыратын онлайн-сатып алу жүйесін жасау болды.

Диссертацияда жобаның мақсаты мен орындалатын тапсырмалар сипатталады.

Осы мақсатқа жету үшін келесі жұмыс жүргізілді: пәндік саланы зерттеу және өз кезегінде зерттеу нәтижелеріне негізделген, дамудың техникалық тапсырмасын қалыптастыру бағдарламалық қамтамасыз ету.

Осы жұмыстың нәтижесінде деректер базасы Node.js + Express негізіндегі серверге, React.js негізіндегі веб-қосымшаға негізделген, MongoDB құжатқа бағытталған дерекқорды басқару жүйесі негізінде әзірленді.

Бұл жобаның нәтижесі - киім сатып алу және тапсырыс беру үшін интернет қызметі.

Бұл тезис жобасы түсіндірме жазба көлемін қамтиды 40 бет, оның ішінде 15 суреттер, 1таблица, 6 библиография тақырыптары, 3 өтінім.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В данной работе рассматривается создание интернет сервиса для покупки товаров. Из-за многочисленных преимуществ и выгод все больше и больше людей говорят, что они предпочитают покупки в Интернете, а не обычные покупки. Процесс принятия решения покупателем в последние годы кардинально изменился. Покупатели проводят обширные исследования в Интернете, прежде чем говорить с продавцом. Покупатели также совершают больше прямых покупок в Интернете и с помощью своего смартфона, никогда не заходя в традиционные торговые точки. Интернет делает бизнес намного проще и быстрее. Это привело к изменениям в способе ведения бизнеса людьми с быстро растущей всемирной тенденцией к онлайн-покупкам или электронной коммерции.

Покупки в интернете могут сэкономить время, как для покупателя, так и для продавца, уменьшая количество телефонных звонков о наличии, технических характеристиках, часах работы или другой информации, которую можно легко найти на страницах компании и продукта.

Целью данного проекта является построение системы для покупки товаров в режиме реального времени.

Актуальность интернет – магазинов возрастает с каждым днем. Так как интернет стал неотъемлемой частью человеческой жизни.

И для достижения данной цели были поставлены такие задачи:

– Изучить психологию интернет покупок.

– Выбрать наилучшее решение для реализации проекта.

– Реализовать основной функционал веб-сайта: заказ, доставка.

– Построить максимально удобный и легкий в использовании интерфейс.

1. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

1.1 Постановка задачи

Необходимо разработать интернет сервис, который позволит клиенту магазина покупать товары из магазина в режиме реального времени.

Основной функцией интернет – сервиса является удобная организация покупок для покупателей данного интернет магазина. Должна быть корзина покупок, куда пользователь может добавлять товары, которые он собирается купить. После заказа товара корзина должна быть пустой.

Система должна быть оснащена удобным поиском товаров, фильтрацией товаров по цене.

Также необходимая функция сервиса – предоставление администратору системы возможности обновлять информацию о товарах, а также добавлять и удалять товары, также видеть список заказов.

Сервис должен работать достаточно быстро и иметь интуитивно-понятный графический интерфейс.

Процесс разработки интернет сервиса для организации покупок в режиме реального времени включает в себя:

– Изучение психологии покупателей интернет-магазинов.

– Написание технического задания.

– Проектирование базы данных, в которой будет храниться информация о товарах, администраторах системы, заказах.

– Разработку серверной части проекта.

– Разработку веб-приложения на платформе React.

Конечными пользователями данного интернет – сервиса онлайн покупок являются покупатели и администратор системы.

1.2 Термины и сокращения

Термины использованные при разработке, а также связанные с реализацией и использованными технологиями отображены в таблице 1.

Таблица 1 – Термины , сокращения, и их определения

Сокращение или термин	Определение
БД	База данных
NoSQL	Not only Structured Query Language
UML	Unified Modeling Language
HTML	HyperText Markup Language
CSS	Cascading Style Sheets

СУБД	Система управления базами данных
MERN	Mongo-Express-React-Node

1.3 Психология покупателей интернет-магазинов

Статистика показывает что люди с каждым днем предпочитают совершать покупки в интернете, поэтому понимание покупателя интернет-магазина приобретает все большую важность.

Те, кто проводит больше времени в социальных сетях, чаще совершают импульсивные покупки в Интернете: эти клиенты, скорее всего, станут более потребительскими и будут иметь более высокий уровень путаницы с идентичностью.

Неудивительно, что импульсивные покупки также стимулируются акциями и распродажами . По мере того как общая сумма, потраченная на другие товары, увеличивается, покупатели с большей вероятностью добавляют импульсные покупки в свою корзину.

Что еще интереснее, дизайн сайта также играет свою роль; более эстетичный веб-сайт, скорее всего, будет стимулировать импульсивное поведение при покупке.

Более того, клиенты принимают решение о покупке продукта всего за 90 секунд. Также покупателей волнует и скорость работы ресурса в целом. Если товарная страница открывается дольше 3 секунд, 57% пользователей просто уходят с нее, а 80% из тех, кто покинул страницу, не вернутся уже никогда.

Чтобы стимулировать импульсивные покупки:

- убедиться, что сайт прост в использовании;
- имеет четкое путешествие для пользователя;
- страницы загружается быстро;

Эти факторы влияют на уровень продаж, что означает, что они помогают клиентам найти то, что они ищут, более эффективно и повышают вероятность покупки.

Цвет сайта может оказывать незначительное влияние на восприятие ваших пользователей, особенно благодаря использованию синего цвета.

Синие оттенки, как правило, увеличивают «ощущение потока» - сочетание удовольствия от веб-сайта и концентрации - по сравнению с желтыми оттенками.

Исследования также показали, что синие оттенки могут увеличить патронаж клиента независимо от цены продукта, в то время как красные

оттенки делают покупателей более чувствительными к цене из-за коннотации с «продажами» и снижением цен; этот вывод сильнее для ярко-красного фона, чем для темно-красного фона.

Организация продуктов в сегментированные подкатегории создает впечатление большего разнообразия продуктов, а также облегчает клиентам быстрый поиск того, что они ищут. Исследования показали, что это большее разнообразие может оставить у покупателей положительное впечатление о магазине, а значит, они с большей вероятностью вернутся.

Недавнее исследование показало, что навигация по сайту оказывает огромное влияние на посетителей и может стать частью их намерения приобрести.

Надо тщательно продумать дизайн своих ссылок – как правило, они должны помочь клиентам найти нужную информацию в 3 клика или меньше.

Показатель отказа от покупки на последнем этапе конверсионного пути составляет в среднем 67,4%.

В ходе исследования было установлено, что:

– 41% покупателей отказываются от продукта из-за дополнительных расходов;

– 29% респондентов пугает обязательная регистрация;

– 11% покупателей смущают непонятные условия доставки;

– для 10% покупателей процесс оформления заказов и оплаты чересчур долгий;

– 1% покупателей заявили о других причинах, побудивших их прервать заказ.

Как правило, 24% интернет-магазинов требуют регистрации. Например, ASOS удалось вдвое снизить коэффициент отказов за счет возможности оформить заказа в качестве гостя.

Средняя продолжительность процесса оформления заказов и оплаты среди 100 крупнейших сайтов eCommerce составляет 5.08 шагов. В онлайн-магазинах оформление заказа должно укладываться максимум в 5 шагов. Более того, 50% онлайн-магазинов дважды запрашивают одну и ту же информацию.

Понятные кнопки призыва к действию помогают покупателям лучше ориентироваться при заказе. Кнопки «далее», «продолжить», «следующий» стоит сделать заметными и кликабельными.

Индикатор выполнения заказа помогает покупателям сориентироваться, на каком этапе они находятся сейчас.

Дизайн страницы заказа должен быть простым, четким и понятным, чтобы покупатель не запутался в процессе. Должна быть кнопка «Назад», чтобы покупатели могли внести поправки в заказ.

1.4 Функциональные возможности

Система должна иметь следующие функциональные возможности:

– Поиск и просмотр товаров клиентом;
– просмотр широкой информации о товаре, его описания и иллюстрации;
– добавление и удаление интересующегося товара в корзину покупок;
– просмотр клиентом списка товаров в корзине, его общей цены;
– возможность выбора цвета и размера товара для заказа;
– возможность выбора категории интересующего товара;
– фильтрация товаров по размеру, цвету и цене;
– сортировка товаров по имени и цене;
– оформление заказа на покупку товара;
– форма входа для администратора;
– отдельная форма для администратора при его входе;
– список всех товаров в удобном виде для администратора;
– возможность удалять и изменять товары администратором;
– просмотр администратором списка добавленных, измененных и удаленных товаров;
– просмотр администратором списка заказов клиентов и информации о клиентах;

1.5 Технологии для разработки проектной части

Для разработки проектной части были использованы следующие технологии:

1. MongoDB - это база данных документов с открытым исходным кодом, которая обеспечивает постоянство данных вашего приложения и разработана с учетом масштабируемости и гибкости разработчика. MongoDB устраняет разрыв между хранилищами значений ключей, которые являются быстрыми и масштабируемыми, и реляционными базами данных, которые имеют богатую функциональность. Вместо хранения данных в строках и столбцах, как это делается в реляционной

базе данных, MongoDB сохраняет документы JSON в коллекциях с динамическими схемами.

Модель данных документа MongoDB позволяет легко хранить и комбинировать данные любой структуры, не отказываясь от сложных правил проверки, гибкого доступа к данным и богатой функциональности индексирования. Вы можете динамически изменять схему без простоев, что очень важно для быстро развивающихся приложений.

Его можно масштабировать внутри и между географически распределенными центрами обработки данных, обеспечивая высокий уровень доступности и масштабируемости. По мере роста ваших развертываний база данных легко масштабируется без простоев и без изменения вашего приложения.

2. React - (иногда называемый ReactJS), библиотека JavaScript, разработанная Facebook для создания интерактивных / реактивных пользовательских интерфейсов. Как и Angular, React разбивает внешнее приложение на компоненты. Каждый компонент может содержать свое собственное состояние, и родительский элемент может передавать свое состояние своим дочерним компонентам, и эти компоненты могут передавать изменения родительскому элементу с помощью функций обратного вызова. Компоненты React обычно реализуются с использованием JSX - расширения JavaScript, которое позволяет встроить синтаксис HTML в код.
3. Библиотека Redux — шаблон для JavaScript предназначенный для управления состоянием приложения. Чаще всего используется в связке с React или Angular для разработки клиентской части. Содержит ряд инструментов, позволяющих значительно упростить передачу данных хранилища через контекст. Создатели: Дэн Абрамов и Эндрю Кларк.
4. Node.js — это среда выполнения JavaScript, которая запускает ваше внутреннее приложение (через Express).

   Node.js основан на движке Google V8 JavaScript, который используется в браузерах Chrome. Он также включает в себя ряд модулей, которые предоставляют функции, необходимые для реализации веб-приложений, включая сетевые протоколы, такие как HTTP. Сторонние модули, включая драйвер MongoDB, могут быть установлены с помощью npmинструмента. Node.js - это асинхронный механизм, управляемый событиями, в котором приложение выполняет запрос, а затем продолжает работать над другими полезными задачами, а не останавливается, ожидая ответа. По завершении запрошенного задания приложение информируется о результатах посредством обратного вызова. Это позволяет параллельно выполнять большое количество операций, что важно при масштабировании приложений. MongoDB также был

разработан для асинхронного использования, поэтому он хорошо работает с приложениями Node.js.

5. Express.js - это фреймворк для веб-приложений, в котором выполняется код вашего JavaScript-приложения. Express работает как модуль в среде Node.js.

   Express может обрабатывать маршрутизацию запросов к нужным частям вашего приложения (или к различным приложениям, работающим в одной среде).

   Вы можете запустить полную бизнес-логику приложения в Express и даже сгенерировать окончательный HTML-код, который будет отображаться браузером пользователя. С другой стороны, Express можно использовать для простого предоставления REST API - предоставления внешнему приложению доступа к ресурсам, которые ему необходимы, например, к базе данных.
6. Mongoose - предоставляет прямое решение на основе схемы для моделирования данных приложения. Он включает встроенное литье типа, проверку, построение запросов, крючки бизнес-логики и многое другое из коробки.
7. Nodemon - это инструмент, который помогает разрабатывать node.приложения на основе js путем автоматического перезапуска приложения узла при обнаружении изменений файлов в каталоге.
8. JWT - JSON Web Token (JWT), это открытый стандарт (RFC 7519), который определяет компактный и автономный способ безопасной передачи информации между сторонами в качестве объекта JSON. Эта информация может быть проверена и доверена, поскольку она имеет цифровую подпись.
9. Morgan - промежуточное ПО регистратора HTTP-запросов для Node.js.
10. React-router - это набор навигационных компонентов, декларативно составляющих приложение. Если вы хотите иметь заметные URL-адреса для своего веб-приложения или составной способ навигации в React Native, React Router работает везде, где React рендеринг.

1.6 Архитектура веб-приложения

Архитектура веб-приложений - это структура, состоящая из связей и взаимодействий между компонентами приложений, такими как пользовательский интерфейс, сервер и базы данных. Общая концепция архитектуры веб-приложений соответствует концепции пользователя браузера, который запускает приложение, которое может работать на нескольких веб-сайтах.

По сути, архитектуры веб-приложений могут быть определены с изображением этого процесса:

– Пользователь просматривает определенный URL, который находит и запрашивает браузер.

– По сети данные отправляются с сервера в браузер, а затем выполняются браузером, чтобы он мог отобразить запрошенную страницу.

– Пользователь просматривает и взаимодействует со страницей.

Архитектура веб-приложений включает в себя все подкомпоненты и взаимозаменяемость внешних приложений для всего приложения.

В то время как мир технологий продолжает развиваться, приложения считаются главными в этом трансформационном процессе. Современная архитектура приложений и ее развитие постоянно совершенствуются как в своих интерфейсных, так и в серверных возможностях.

В частности, на бэкэнде или на стороне сервера существует множество подходов к архитектуре разработки приложений, которые появляются для удовлетворения и решения текущих потребностей разработки, таких как микросервисы, серверные архитектуры и одностраничные приложения.

Архитектуры веб-приложений состоят из нескольких компонентов, которые помогают создавать его цифровую структуру.

Эти компоненты можно разделить на две области: компоненты приложения пользовательского интерфейса и структурные компоненты.

Компоненты приложения пользовательского интерфейса относятся к веб-страницам.

Структурные компоненты, являющиеся настоящим ядром процесса разработки приложений:

– Веб-браузер или клиент.

– Сервер веб-приложений.

– Сервер базы данных.

Веб-браузер или клиент - это представление интерфейса функциональности веб-приложения, с которой взаимодействует пользователь. Этот контент, предоставляемый клиенту, может быть разработан с использованием HTML, JavaScript и CSS и не требует адаптации, связанной с операционной системой. По сути, веб-браузер или клиент управляет взаимодействием конечных пользователей с приложением.

Сервер веб-приложений управляет бизнес-логикой и постоянством данных и может быть построен с использованием PHP, Python, Java, Ruby, .NET, Node.js и других языков. Он состоит как минимум из централизованного

концентратора или центра управления для поддержки многоуровневых приложений.

Сервер базы данных предоставляет и хранит соответствующие данные для приложения. Кроме того, он также может предоставлять бизнес-логику и другую информацию, которой управляет сервер веб-приложений.

Основной целью архитектуры веб-сервера является выполнение запросов клиентов на веб-сайт. Клиентами, как правило, являются браузеры и мобильные приложения, которые отправляют запросы с использованием безопасного протокола HTTP, либо для ресурсов страницы, либо для REST API.

Архитектура веб-приложений Node.js стала сильным кандидатом для разработки веб-приложений. Все больше и больше разработчиков чувствуют себя комфортно с инфраструктурой Node.js для решения проектных и структурных требований в архитектуре разработки приложений. Node.js написан с использованием JavaScript и представляет собой ту же технологию, что и компоненты веб-интерфейса, что позволяет тем же разработчикам программировать серверные службы и пользовательские интерфейсы веб-интерфейса.

Архитектура веб-приложений Node.js обеспечивает скорость и эффективность среды разработки, что делает ее ключевым игроком в отрасли. По своей сути Node.js был разработан с учетом интеграции, что неудивительно, поскольку все больше и больше предприятий используют эту среду для интеграции многочисленных служб и систем через единый пользовательский интерфейс.

Архитектура веб-приложения Node.js обеспечивает согласованность, совместное использование и повторное использование кода, простую передачу знаний и большое количество бесплатных инструментов.

Эти преимущества обеспечивают гибкость и эффективность при создании надежного веб-приложения.

2. ПРОЕКТНАЯ ЧАСТЬ

2.1 Логическая и физическая модели базы данных

2.1.1 Описание логической и физической моделей базы данных

Логическая модель базы данных представляет собой наименование сущностей и их атрибутов на естественном языке, а также отображает связи сущностей между собой. Физическая модель базы данных представляет сущности в таком виде, в котором они будут храниться в базе данных.

2.1.2 Логическая модель базы данных

Логическая модель базы данных (см. Рисунок 1) имеет четыре сущностей: «Администратор», «Продукты», «Заказы», «Журнал».

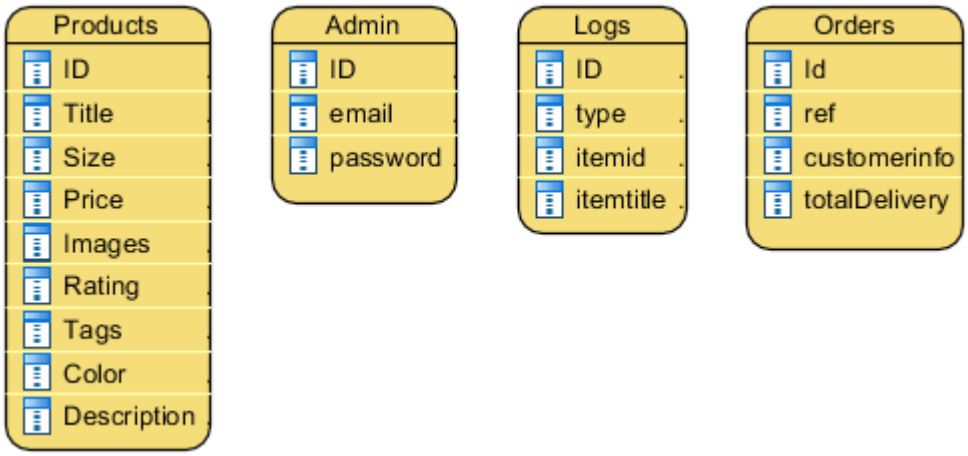


Рисунок 1 – Логическая модель базы данных

1. Сущность «Администратор» («Admin») хранит информацию о администраторе, зарегистрированного в базе данных системы. Она имеет такие атрибуты как id администратора; email – электронная почта для входа в систему администратора; password – пароль администратора для входа в систему.
2. Сущность «Продукты» («Products») – сущность для хранения списка товаров. Имеет такие атрибуты как id товара; title – имя товара; size – размеры товара; price – цена товара; images – картинки товара; rating – оценки товара; tags – категорий товара; color – цвета товара; description – описание товара.

3. Сущность «Журнал» («Logs») – хранит историю изменения, добавления, удаления товаров. Имеет атрибуты id; type – тип действия который сделал администратор; time – время в которое сделано действие администратором; itemid – id товара; itemtitle – название товара.
4. Сущность «Заказ» («Orders») хранит информацию о заказах сделанных клиентами системы. Имеет такие атрибуты как id – идентификатор заказа; ref – информация о товаре; customerinfo – информация о клиенте; totalamount – информация о цене заказа.

2.1.3 Физическая модель базы данных

Физическая модель содержит информацию о всех объектах базы данных. В физической модели показываются промежуточные таблицы и типы данных атрибутов. Физическая модель включает в себя четыре основных таблиц: «Администратор», «Продукты», «Журнал», «Заказы». Она представлена на рисунке (см. Рисунок 2).

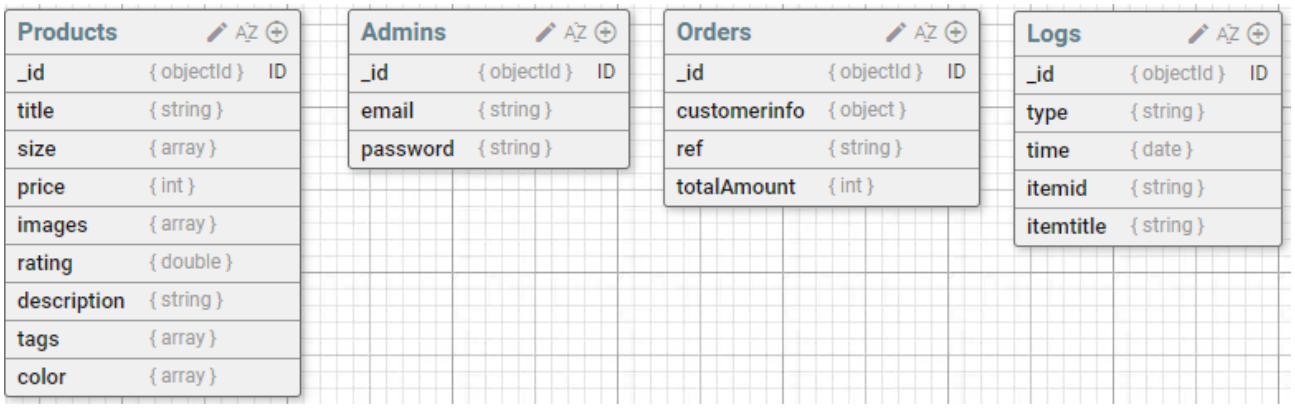


Рисунок 2 – Физическая модель базы данных

1. Коллекция «Администратор» («Admin») имеет такие атрибуты как:
– id (int) – id администратора, первичный ключ;
– email (String) – электронная почта администратора для входа в систему;
– password (String) – пароль администратора.
2. Коллекция «Продукты» («Products») имеет такие атрибуты как:
– id (ObjectId) – id товара, первичный ключ;
– title (String) – название товара;
– price (Int32) – цена товара;
– size (Array) – массив размеров товара;
– images (Array) – массив ссылок на изображения;
– rating (Double) – оценка товара;
– description (String) – описание товара;
– tags (Array) – массив категорий товара;

– color (Array) – массив цветов товара.
3. Коллекция «Заказ» («Orders») имеет такие атрибуты как:
– id (ObjectId) – id заказа, первичный ключ;
– customerinfo (Object) – информация о клиенте который сделал заказ;
– ref (String) – информация о заказанном товаре;
– totalAmount (Int32) – сумма заказа.
4. Коллекция «Журнал» («Logs») имеет такие атрибуты как:
– id (ObjectId) – id просьбы, первичный ключ;
– type (String) – тип действия совершенного администратором;
– time (Date) – время в которое сделано действие;
– itemid (String) – id товара;
– itemtitle (String) – название товара.

2.2 Диаграмма вариантов использования

Диаграмма вариантов использования показывает как покупатель взаимодействует с ситемой (см. Рисунок 3).

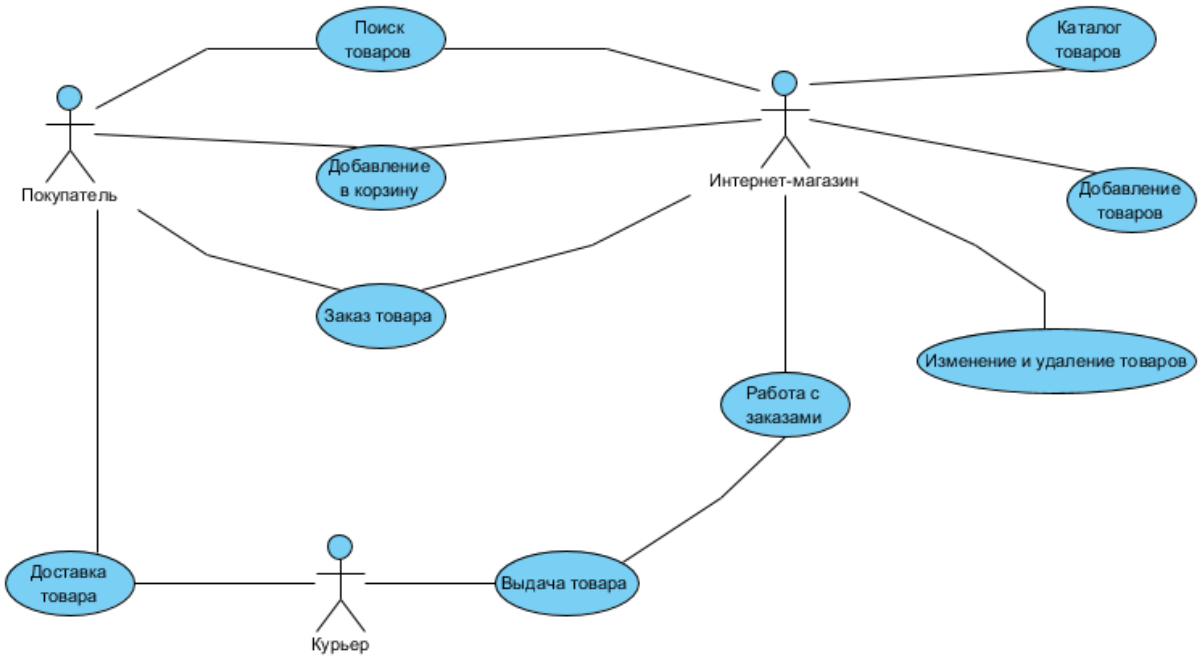


Рисунок 3 – Диаграмма вариантов использования системы

2.3 Диаграмма последовательностей

Диаграмма последовательностей отображает взаимодействие объектов в динамике (см. Рисунок 4).

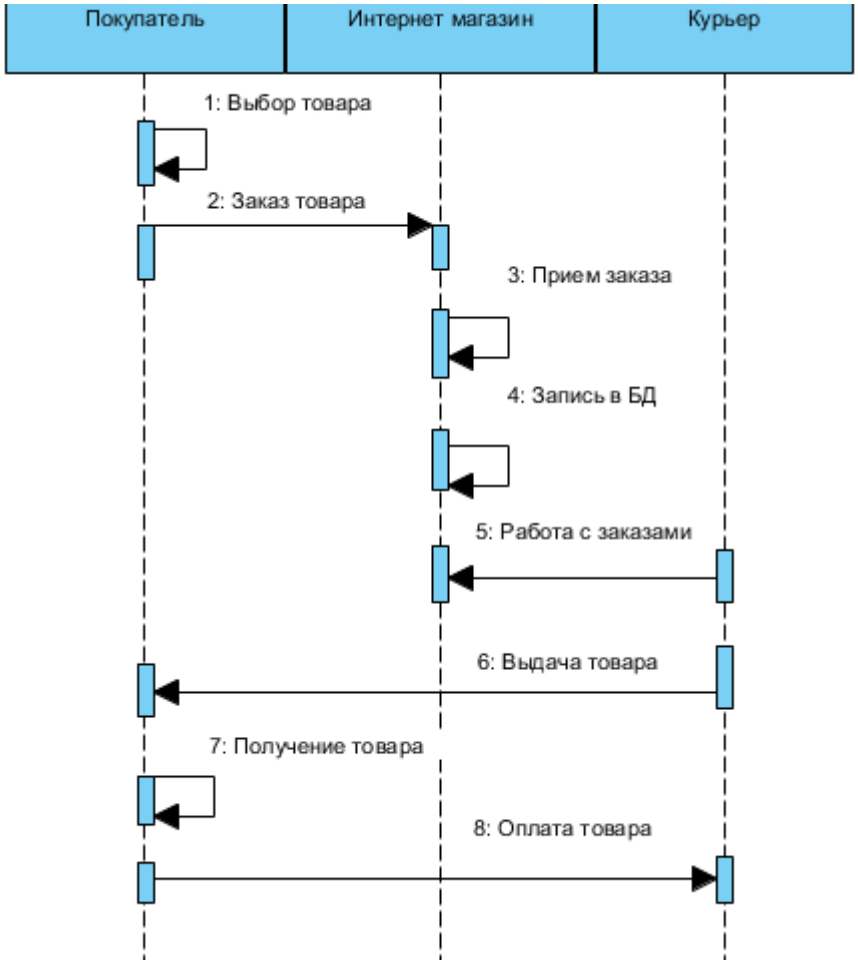


Рисунок 4 – Диаграмма последовательностей

2.4 Функциональность работы веб-приложения

При открытии сайта в браузере отображается главная страница.

1. Главная страница (см. Рисунок 4) – страница которую видит клиент попадая на сайт. Здесь он может выбрать какую одежду он хочет заказать.

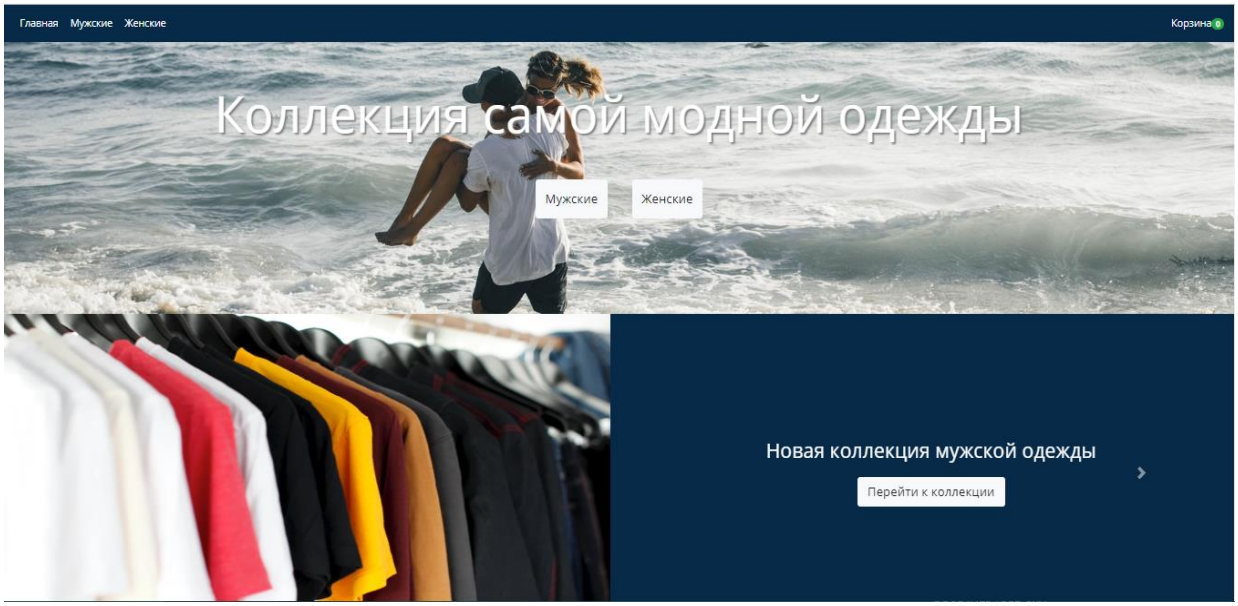


Рисунок 4 – Главная страница сервиса

2. Страница выбора категории мужской одежды (см. Рисунок 5) – страница которую видит пользователь при переходе к выбору мужской одежды. Здесь он может выбрать 4 вида одежды.

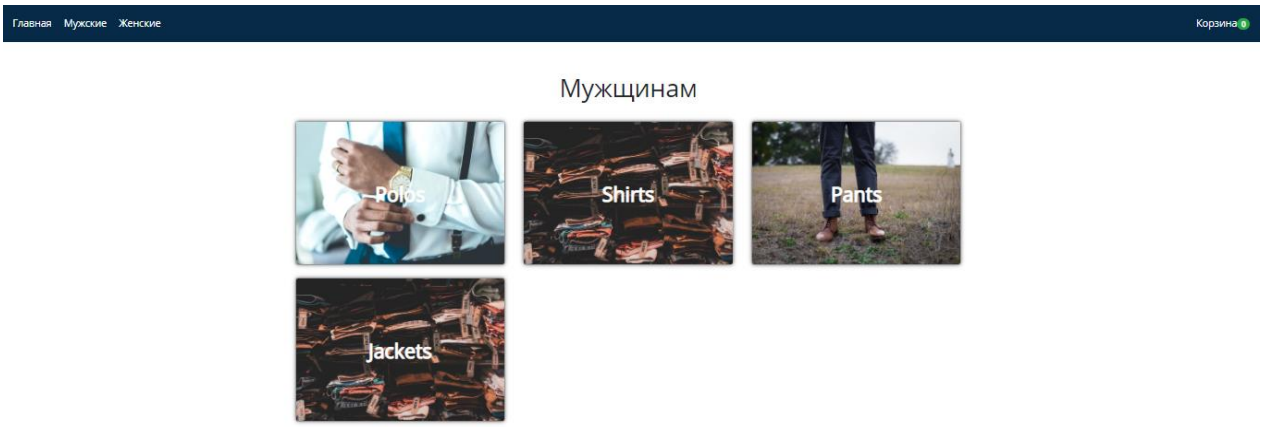


Рисунок 5 – Страница выбора категории мужской одежды

3. Страница выбора категории женской одежды (см. Рисунок 6) – страница которую видит клиент при переходе к выбору женской одежды. Здесь он может выбрать 4 вида одежды.

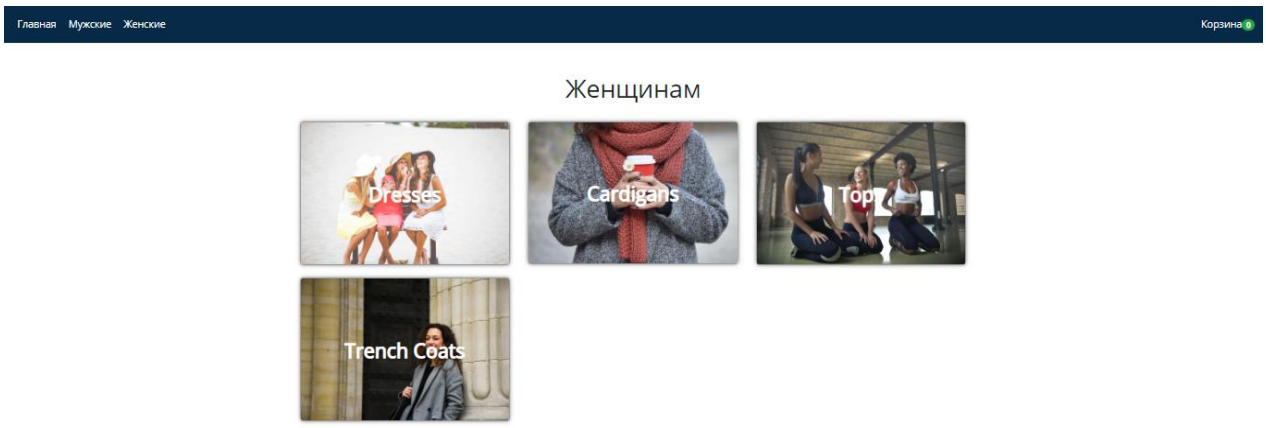


Рисунок 6 – Страница выбора категории женской одежды

4. Страница просмотра и выбора мужской одежды (см. Рисунок 7) – здесь клиент может отсортировать товары по цене, название. Клиент может выбрать категории одежды, размеры, цену по которой он хочет отфильтровать.

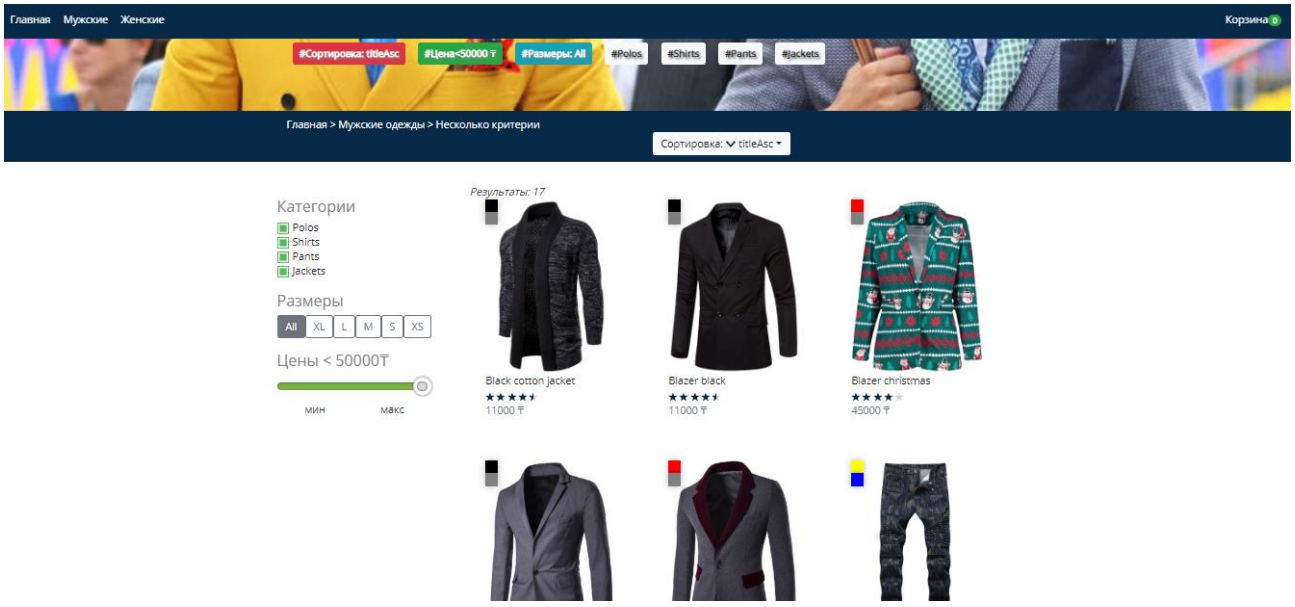


Рисунок 7 – Страница просмотра и выбора мужской одежды

5. Страница просмотра и выбора женской одежды (см. Рисунок 8) – здесь клиент может отсортировать товары по цене, название. Клиент может выбрать категории одежды, размеры, цену по которой он хочет отфильтровать.

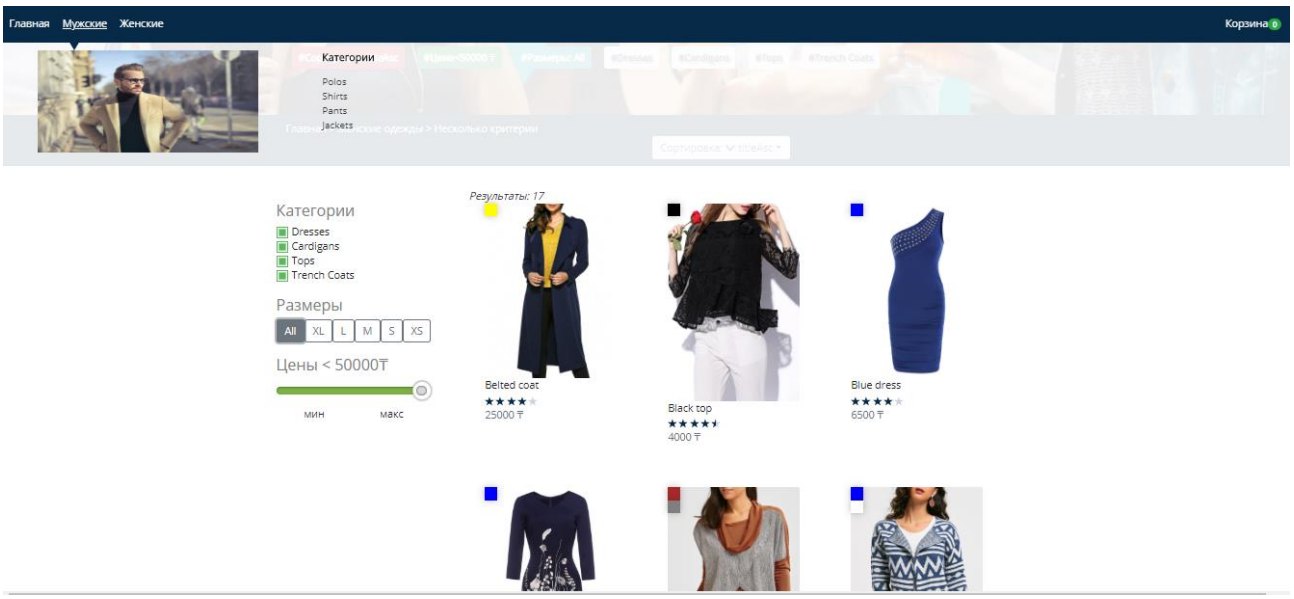


Рисунок 8 – Страница просмотра и выбора женской одежды

6. Страница просмотра товара (см. Рисунок 9) – просмотр описания, иллюстрации к конкретному товару. На этой странице клиент также видит цену товара, рейтинг. Для дальнейшего заказа товара клиент должен выбрать цвет и размер товара.

Рисунок 9 – Страница просмотра товара.

7. Окно при добавлений товара в корзину (см. Рисунок 10) – окно которое выходит при добавлений товара в корзину, здесь клиент может перейти к станице заказа, перейти в корзину или же закрыть это окно.

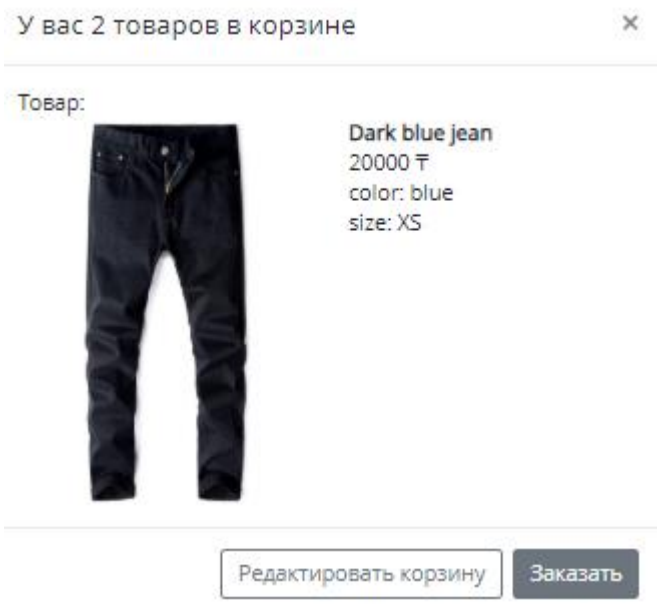


Рисунок 10 – Окно при добавлений товара в корзину

8. Страница заказа товара (см. Рисунок 11) – страница для заказа товара. Здесь клиент видит детали этого заказ, далее он должен сначала написать свой e-mail, ввести свое имя и адрес для доставки товара. Далее он должен подтвердить свой заказ.

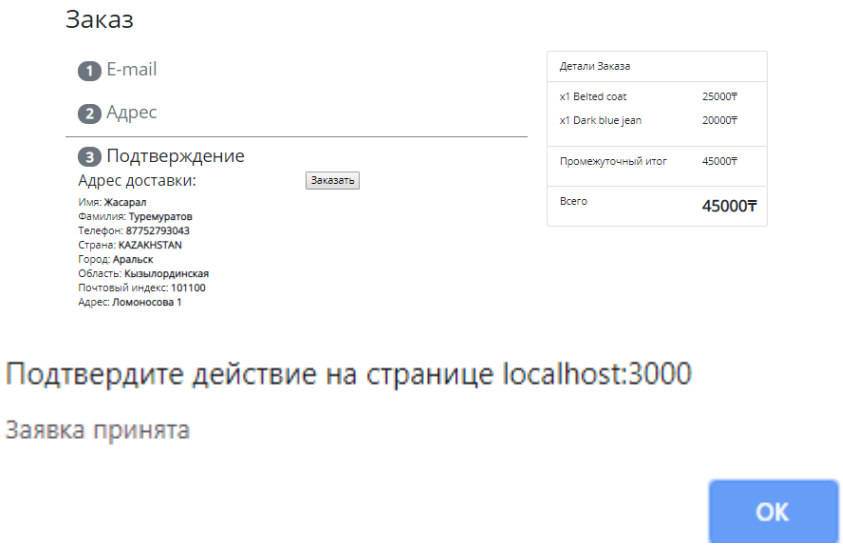


Рисунок 11 – Страница заказа товара и окно, которое выходит при подтверждении заказа товара

9. Страница корзины товаров (см. Рисунок 12) – список товаров которые клиент добавил в корзину. Здесь клиент может увеличить или уменьшить

количество определенного товара. Также клиент может удалить определенный товар.

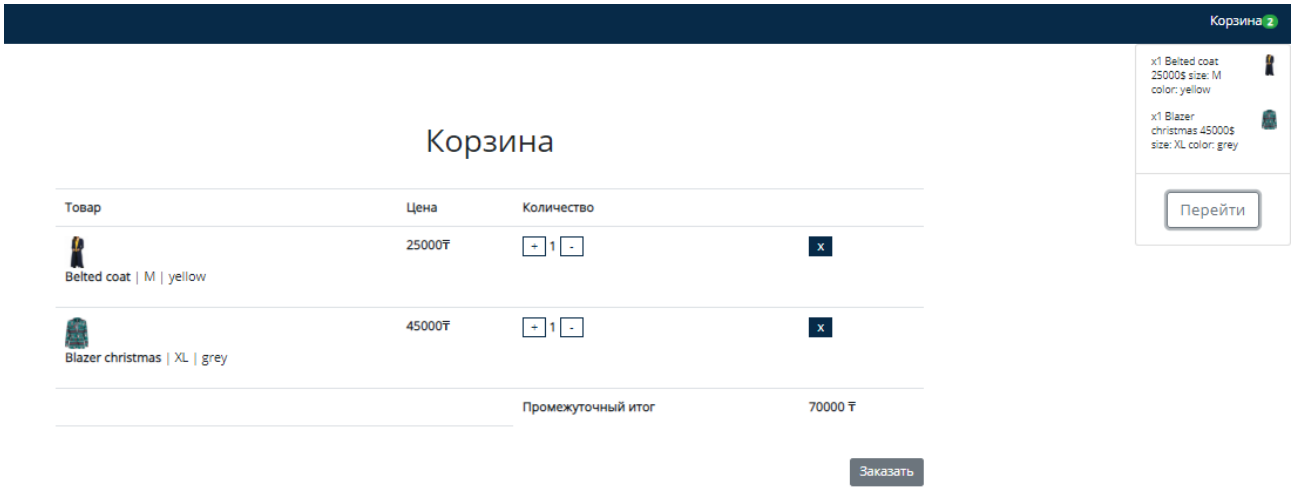


Рисунок 12 – Страница корзины товаров

10. Страница для перехода в панель администартора (см. Рисунок 13) – для перехода в панель администратора надо перейти по адресу http://localhost:3000/admin. После которого вам будет предложено авторизоваться от имени администратора.

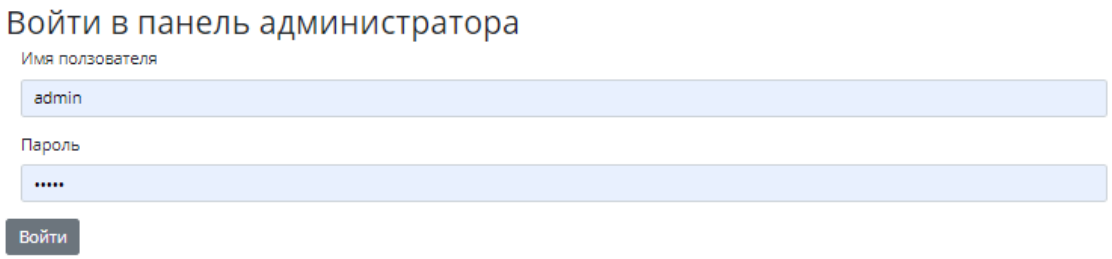
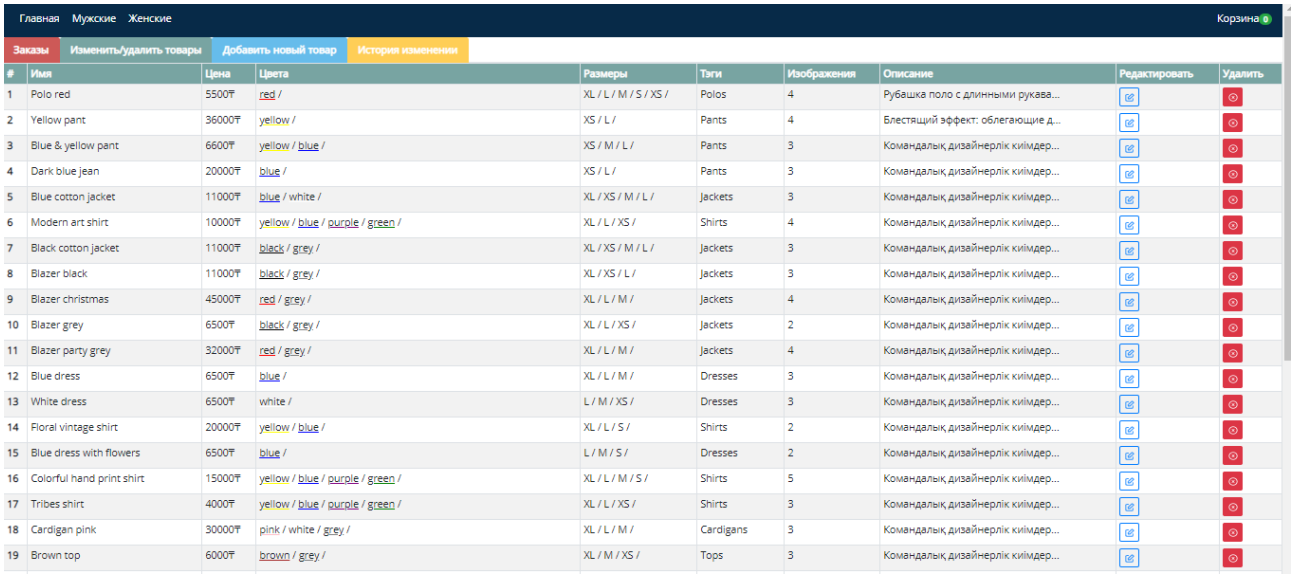


Рисунок 13 – Страница для перехода в панель администартора

11. Страница администратора (см. Рисунок 14) – просмотр списка товаров, здесь администратов может удалить товар из списка или изменить.



#	Имя	Цена	Цвета	Размеры	Тэги	Изображения	Описание	Редактировать	Удалить
1	Polo red	5500₸	red /	XL / L / M / S / XS /	Polos	4	Рубашка поло с длинными рукава...		
2	Yellow pant	36000₸	yellow /	XS / L /	Pants	4	Блестящий эффект: облегающие д...		
3	Blue & yellow pant	6600₸	yellow / blue /	XS / M / L /	Pants	3	Командалық дизайнерлік киімдер...		
4	Dark blue jean	20000₸	blue /	XS / L /	Pants	3	Командалық дизайнерлік киімдер...		
5	Blue cotton jacket	11000₸	blue / white /	XL / XS / M / L /	Jackets	3	Командалық дизайнерлік киімдер...		
6	Modern art shirt	10000₸	yellow / blue / purple / green /	XL / L / XS /	Shirts	4	Командалық дизайнерлік киімдер...		
7	Black cotton jacket	11000₸	black / grey /	XL / XS / M / L /	Jackets	3	Командалық дизайнерлік киімдер...		
8	Blazer black	11000₸	black / grey /	XL / XS / L /	Jackets	3	Командалық дизайнерлік киімдер...		
9	Blazer christmas	45000₸	red / grey /	XL / L / M /	Jackets	4	Командалық дизайнерлік киімдер...		
10	Blazer grey	6500₸	black / grey /	XL / L / XS /	Jackets	2	Командалық дизайнерлік киімдер...		
11	Blazer party grey	32000₸	red / grey /	XL / L / M /	Jackets	4	Командалық дизайнерлік киімдер...		
12	Blue dress	6500₸	blue /	XL / L / M /	Dresses	3	Командалық дизайнерлік киімдер...		
13	White dress	6500₸	white /	L / M / XS /	Dresses	3	Командалық дизайнерлік киімдер...		
14	Floral vintage shirt	20000₸	yellow / blue /	XL / L / S /	Shirts	2	Командалық дизайнерлік киімдер...		
15	Blue dress with flowers	6500₸	blue /	L / M / S /	Dresses	2	Командалық дизайнерлік киімдер...		
16	Colorful hand print shirt	15000₸	yellow / blue / purple / green /	XL / L / M / S /	Shirts	5	Командалық дизайнерлік киімдер...		
17	Tribes shirt	4000₸	yellow / blue / purple / green /	XL / L / XS /	Shirts	3	Командалық дизайнерлік киімдер...		
18	Cardigan pink	30000₸	pink / white / grey /	XL / L / M /	Cardigans	3	Командалық дизайнерлік киімдер...		
19	Brown top	6000₸	brown / grey /	XL / M / XS /	Tops	3	Командалық дизайнерлік киімдер...		

Рисунок 14 – Страница администратора

12. Страница добавления нового товара (см. Рисунок 15) – здесь администратор добавляет новый новар, вводит имя товара, цену и тд.

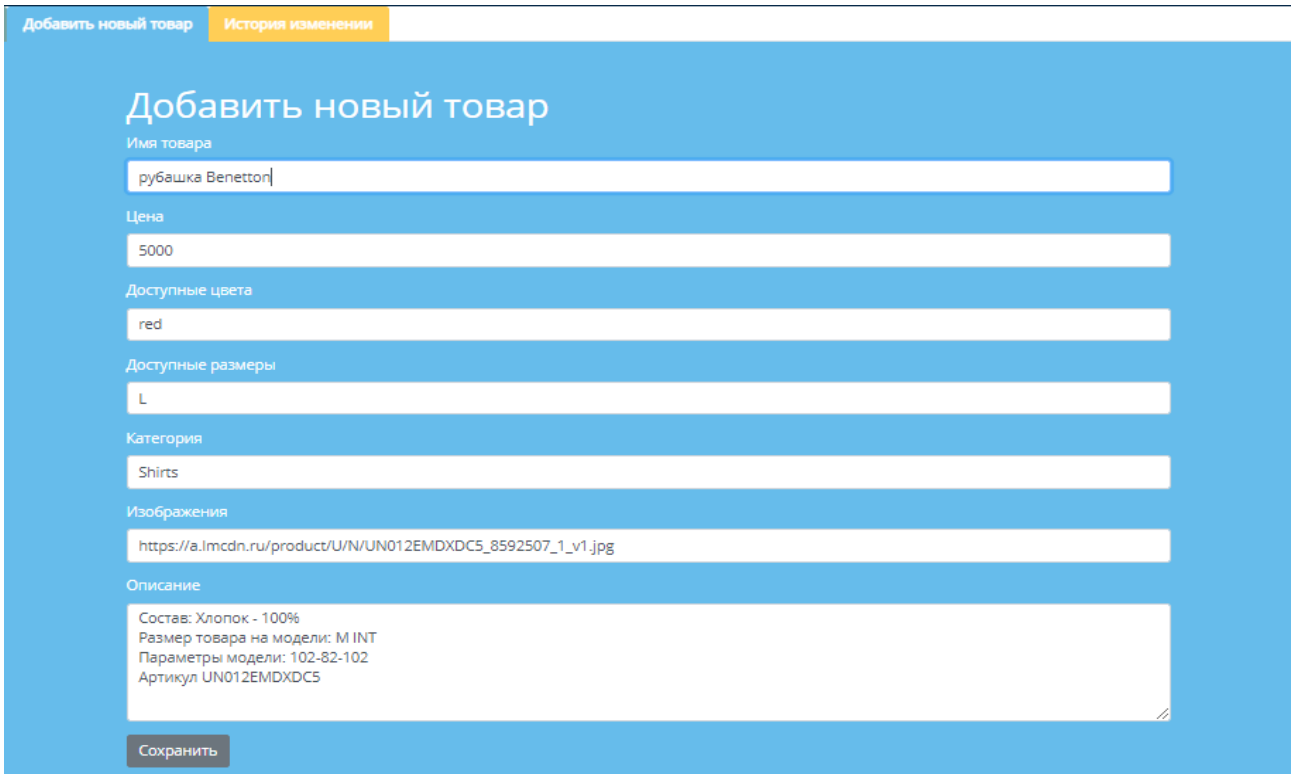


Рисунок 15 – Страница добавления нового товар

В списке товаров можно увидеть что товар был добавлен (см. Рисунок 16).

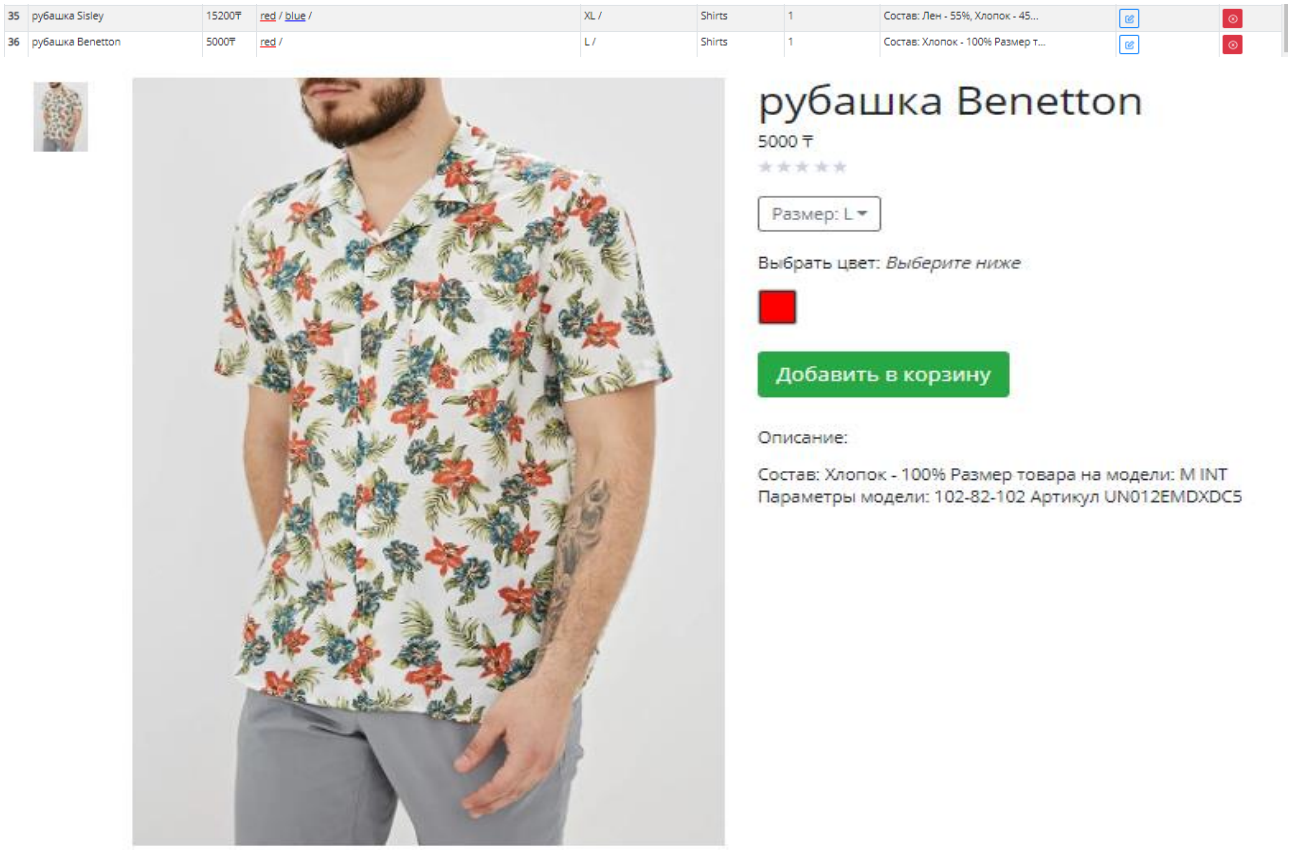


Рисунок 16 – Список добавленных товаров

13. Страница «История изменений» (см. Рисунок 17) – здесь администратор может получить информацию об измененных, удаленных и добавленных товарах.

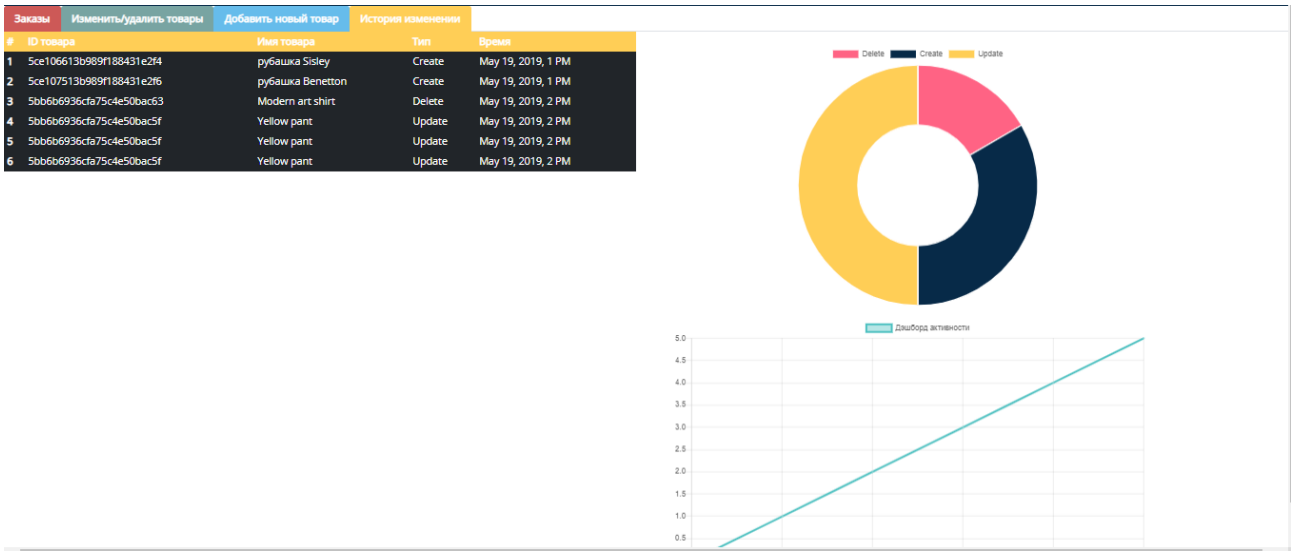


Рисунок 17 – Страница «История изменений»

14. Окна при редактирований товара и удалений (см. Рисунок 18) – здесь администратор может изменить товар или подтвердить удаление.

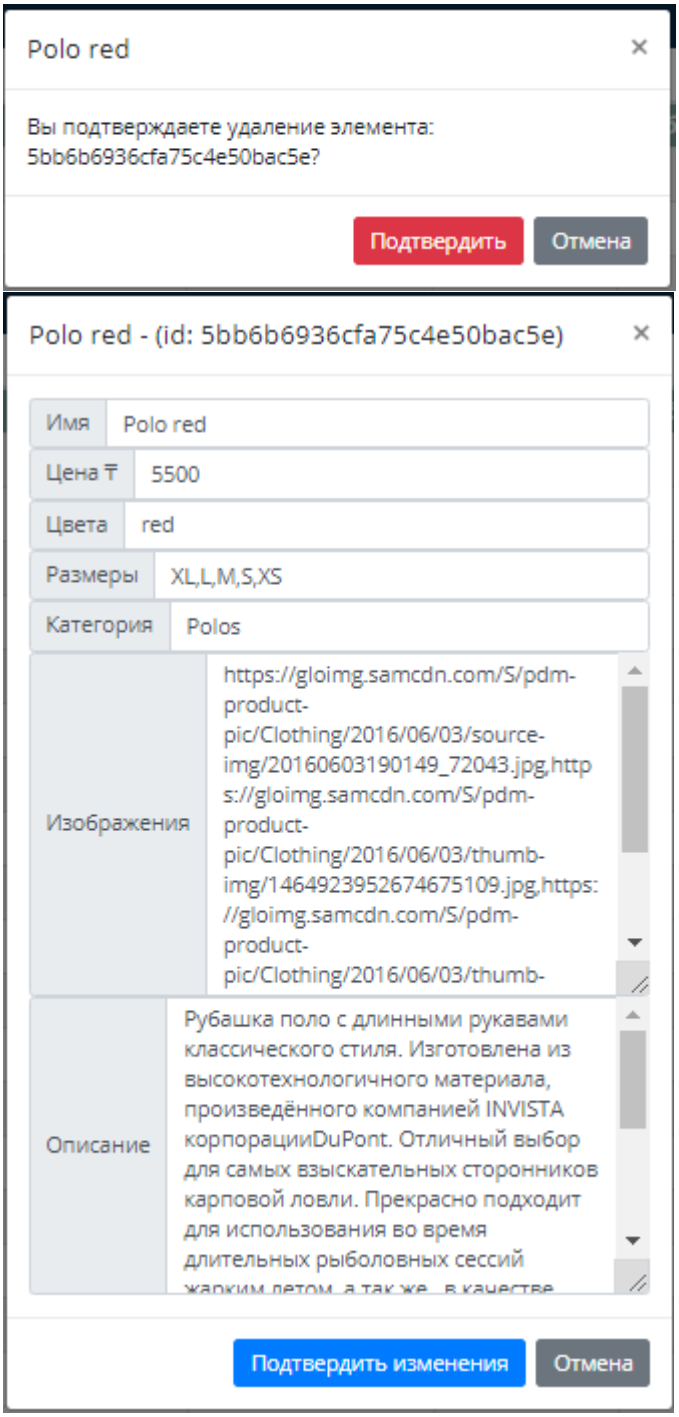


Рисунок 18 – Окна при редактирований товара и удалений

15.Окно списка заказов (см. Рисунок 19) – здесь администратор видит список заказов и информацию о клиенте который сделал заказ.

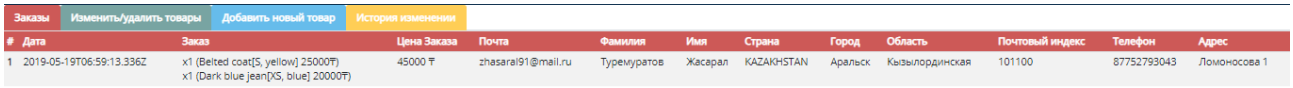


Рисунок 19 – Окно списка заказов

ЗАКЛЮЧЕНИЕ

Разрботка дипломного проекта под названием «Интернет - система покупок» была выполнена успешно. Проект был разработан с большой ответственностью и без ошибок и в то же время она является эффективным и менее трудоемким.

Цель этого проекта была направлена на разработку веб-приложения для приобретения одежды из интернет - магазина.

Этот проект помог мне в получении ценной информации и практического знания по нескольким такие темы, как разработка веб-страниц с помощью React, управление базой данных с использованием MongoDB. Кроме того, проект помог нам понимание о фазах развития проекта и программного обеспечения жизненного цикла разработки.

Этот проект дал мне большое удовлетворение, разработав приложение, которое может быть реализовано в любых близлежащих магазинах или фирменных магазинах, торгующие различными видами одежды. Существует ряд нововведении для дальнейшего развития этого проекта. Ряд функций может быть добавлен к этой системе в будущем, для полного обеспечения интернет – магазина.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Официальная документация React // Электронная версия на сайте https://reactjs.org/
2. Онлайн-учебник по JavaScript // Электронная версия на сайте https://learn.javascript.ru/
3. Документация по Node.js // Электронная версия на сайте http://www.nodebeginner.ru/
4. Документация по MongoDB // Электронная версия на сайте https://www.mongodb.com/
5. Документация по React // Электронная версия на сайте http://krambertech.github.io/spa-webinar/
6. Документация по React // Электронная версия на сайте https://habr.com/ru/company/ruvds/blog/432636/

Приложение А

Техническое задание

А.1.1 Техническое задание на разработку системы интернет-покупок

Настоящее техническое задание распространяется на разработку системы интернет-покупок. Предполагается, что использовать данную систему будут покупатели, администраторы магазина.

А.1.2 Назначение

Система предназначена для быстрого и удобного поиска и заказа одежды в режиме онлайн.

А.1.3 Требования к функциональным характеристикам

Система должна обеспечивать возможность выполнения следующих функций:

– Авторизация администратора.

– Администрирование товаров (добавление, удаление, изменение).

– Поиск товара по заданным критериям:

- размер;
- цена;
- категория товара.

– Удобная и функциональная корзина покупок.

– Полное оформление заказа.

А.1.4 Требования к надежности

Предусмотреть контроль вводимой информации. Предусмотреть блокировку некорректных действий пользователя при работе с системой. Обеспечить целостность хранимой информации.

А.1.5 Требования к составу и параметрам технических средств

Система должна работать на IBM-совместимых персональных компьютерах. Минимальная конфигурация: тип процессора – Pentium и выше; объем ОЗУ – 32 Мб и выше.

А.1.6 Требования к информационной и программной совместимости

Система должна работать под управлением операционных систем семейства Windows, MacOS.

А.1.7 Требования к программной документации

Разрабатываемые программные модули должны быть самодокументированы, т.е. тексты программ должны содержать все необходимые комментарии.

Программная система должна включать справочную информацию о работе и подсказки пользователю.

Приложение Б

Код приложения

Серверная и клиентская части проекта находятся в GitHub и доступны по следуйщей ссылке:

Проект: https://github.com/ZHasaral/Diploma Для запуска необходимо скачать репозиторий с Github и запустить с помощью команды npm start .

Приложение В

Код маршрутизации серверной и клиентской части

```
router.get('/productsdata', (req, res)=> {
  ModelProducts.find((err, x) => {
      err ? res.status(500).send(err) :
      res.json(x)
  })
})

router.get('/productsdata/:id', (req, res)=> {
  ModelProducts.findById(req.params.id)
    .then(x => {
      if (!x) return res.status(404).end()
      return res.json(x)
    })
    .catch(err => next(err))
})

router.post('/update/item', (req, res, next)=> {
  const { id, title, price, color, size, tags, images, description } = req.body
  ModelProducts.findById(id, (err, model) => {
    if (err) return handleError(err);

    model.set({ title, price, color, size, tags, images, description });
    model.save(function (err, updatedItem) {
      if (err) return console.log(err);
      next();
    });
  }).then(()=>{
    const logUpdate = new ModelLog({
      type: 'Update',
      time: new Date(),
      itemid: id,
      itemtitle: title
    });
    logUpdate.save(function(err, logSaved){
      if(err){return next(err);}
      res.end();
```

```
});
  }).catch(err => next(err));
})

router.post('/add/item', (req, res, next)=> {
  const { title, price, description, size, tags, images, color  } = req.body
  const newItem = new ModelProducts({
    title,
    price,
    color,
    size,
    tags,
    images,
    description
  });
  newItem.save((err, saveditem) => {
    if (err) return console.log(err);
    next();
  })
  const logAdd = new ModelLog({
    type: 'Create',
    time: new Date(),
    itemid: newItem._id,
    itemtitle: title
  });
  logAdd.save(function(err, logSaved){
    if(err){return next(err);}
    res.end();
  });
})

router.post('/add/orders', (req, res, next)=> {
  const { ref, customerinfo, order, totalDelivery, totalAmount } = req.body
  const newOrder = new ModelOrders({
    ref,
    customerinfo,
    order,
    totalDelivery,
    totalAmount
  });
  newOrder.save((err, saveditem) => {
```

```
    if (err) return console.log(err);
    res.end();
  })
})

router.get('/orders', (req, res)=> {
  ModelOrders.find((err, x) => {
      err ? res.status(500).send(err) :
      res.json(x)
  })
})

router.post('/delete/item', (req, res, next)=> {
  const { id, title } = req.body
  ModelProducts.findByIdAndRemove(id, (err, model) => {
    if (err) return handleError(err);
    res.send('item: '+id+' deleted');
  })
  .then(()=>{
    const logDelete = new ModelLog({
      type: 'Delete',
      time: new Date(),
      itemid: id,
      itemtitle: title
    });
    logDelete.save(function(err, logSaved){
      if(err){return next(err);}
      res.end();
    });
  }).catch(err => next(err));
});

router.get('/log', (req, res)=> {
  ModelLog.find((err, x) => {
      err ? res.status(500).send(err) :
      res.json(x)
  })
})

router.get('/secret', passport.authenticate('jwt', {session: false}), (req, res)=> {
```

Продолжение приложения В

```
res.json({authorization: true});
});
router.post('/signin', passport.authenticate('local', {session:false}),
Authentication.signin);

router.post('/signup', Authentication.signup);

module.exports = router;
```

2. Клиентская часть:

```
const Router = () => (
  <div>
    <NavbarContainer />
      <Switch>
        <Route exact path='/' component={HomepageContainer} />
        <Route exact path='/productslist' component={ItemsListContainer} />
        <Route exact path='/item/:id/:item' component={ItemContainer} />
        <Route exact path='/checkout' component={CheckoutContainer} />
        <Route exact path='/cart' component={CartContainer} />
        <Route exact path='/productslist/:gender' component={ItemsListContainer} />
        <Route exact path='/category/:gender'
component={ItemsListGenderHomepage} />
        <Route exact path='/admin' component={AdminContainer} />
        <Route exact path='/dashboard' component={Secret} />
        <Route component={Empty}/>
      </Switch>
    <Footer />
  </div>
);
```

СПЕЦИФИКАЦИЯ

Формат	Зона	Поз.	Обозначение	Наименование	Кол.	Приме-чание

					Дипломный проект	Лист
Изм.	Лист	№ докум.	Подпись	Дата		60