**Санкт-Петербургское государственное бюджетное  
профессиональное образовательное учреждение   
«Академия управления городской средой, градостроительства и печати»**

**КУРСОВОЙ ПРОЕКТ**

|  |
| --- |
| по МДК.05.02 Разработка кода информационных систем |
|  |

|  |  |
| --- | --- |
| Тема: | Разработка модуля веб-приложения на стороне клиента |
| «Интернет-магазин зефирных композиций «Зефиролия» | |

|  |  |
| --- | --- |
| Специальность | 09.02.07 «Информационные системы и программирование» |

|  |  |  |  |
| --- | --- | --- | --- |
| Руководитель |  |  | С.А. Марышев |
|  | подпись |  | И.О.Фамилия |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | Бычкова А.М. |  | Группа | 9ИСП-211 |
|  | Фамилия И.О. |  |  |  |

**Работа выполнена с оценкой**

Санкт-Петербург – 2024 год

СОДЕРЖАНИЕ

[ВВЕДЕНИЕ 3](#_Toc163769637)

[1 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ 5](#_Toc163769638)

[1.1 Идентификация предметной области 5](#_Toc163769639)

[1.2 Моделирование бизнес-процессов предметной области 8](#_Toc163769640)

[1.3 Проектирование графического пользовательского интерфейса 11](#_Toc163769641)

[1.4 Проектирование структуры информационной системы 13](#_Toc163769642)

[1.5 Разработка диаграммы использования 14](#_Toc163769643)

[1.6 Разработка ER-диаграммы информационной системы 15](#_Toc163769644)

[2 РЕАЛИЗАЦИЯ ВЕБ-ПРИЛОЖЕНИЯ СО СТОРОНЫ КЛИЕНТА 17](#_Toc163769645)

[2.1 Реализация физической модели базы данных 17](#_Toc163769646)

[2.2 Выбор и обоснование используемого программного обеспечения 20](#_Toc163769647)

[2.3 Разработка дизайна веб-приложения 22](#_Toc163769648)

[2.4 Разработка интерфейса взаимодействия с пользователем 23](#_Toc163769649)

[2.5 Разработка функционала клиентской части приложения 26](#_Toc163769650)

[3 ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ 32](#_Toc163769651)

[3.1 Выбор стратегии тестирования программного продукта 32](#_Toc163769652)

[3.2 Проведение тестирования информационной системы 33](#_Toc163769653)

[ЗАКЛЮЧЕНИЕ 37](#_Toc163769654)

[СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ 38](#_Toc163769655)

ВВЕДЕНИЕ

Социальная значимость создания веб-сайта для онлайн магазина drill одежды обусловлена современными трендами в сфере Интернет-торговли. В современном мире потребители все больше предпочитают удобство онлайн покупок, что делает важным разработку качественного и функционального веб-пространства для предоставления товаров и услуг.

Практическая значимость создания веб-сайта онлайн магазина заключается в возможности изучения обширного объема различных инструментов разработки, а также процессы моделирования системы. Это позволит понять то, как проходит разработка клиентской части веб-приложений. Данные знания окажутся полезными в повышении своей квалификации в программировании.

Цель настоящего курсового проекта заключается в выявлении особенностей разработки веб-сайта для онлайн магазина drill одежды, способного эффективно привлекать клиентов и обеспечивать удобство покупок.

Для достижения поставленной цели определены следующие задачи:

* Определить основные условия и требования к функциональности веб-сайта.
* Разработать интерфейс клиентской части веб-приложения для взаимодействия с пользователями.
* Апробировать разработанный веб-сайт и сформулировать рекомендации по его улучшению.

Объектом курсового проекта является процесс создания веб-сайта для онлайн магазина drill одежды.

Предметом курсового проекта является разработка кода информационной системы.

Практическая часть работы выполнялась при использовании документаций использованных средств.

Курсовой проект состоит из оглавления, введения, трех глав, заключения и списка источников информации.

В первой главе определена предметная область, промоделированы бизнес-процессы, спроектирован графический интерфейс, спроектирована структура информационной системы, разработана диаграмма использования, разработана ER-диаграмма.

Во второй главе описаны реализация физической модели базы данных, выбор программного обеспечения, дизайн веб-приложения, интерфейс взаимодействия с пользователем, разработка функционала клиентской части приложения.

В третьей главе проведено тестирование информационной системы.

# 1 ПРОЕКТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

## 1.1 Идентификация предметной области

Предметной областью настоящей курсовой работы является торговая компания розничной торговли одежды.

Компания-дистрибьютор «DrillDrip» закупает одежду отечественных и зарубежных производителей и реализует их через собственную дистрибьюторскую сеть и сеть магазинов одежды.

Кроме того, компания осуществляет логистическую деятельность, обеспечивая доставку товаров как собственными транспортными средствами, так и используя услуги сторонних перевозчиков. Это означает, что предметная область включает в себя не только торговлю одеждой, но и логистические процессы, управление складами, контроль качества товаров, а также взаимодействие с различными сторонними организациями, такими как поставщики и транспортные компании, что, в свою очередь, подразумевает комплексный подход к управлению цепочкой поставок, включая складирование, транспортировку и доставку продукции от поставщиков до конечных потребителей.

В контексте идентификации предметной области, важно учитывать все аспекты бизнеса компании «DrillDrip», включая не только процессы закупки и реализации товаров, но и управление логистикой и взаимодействие с поставщиками и клиентами.

Взаимодействие с клиентами включает в себя все операции и действия, направленные на привлечение внимания, убеждение совершать покупки товаров компании и успешное завершение транзакции. Он охватывает такие аспекты, как маркетинговые исследования, привлечение внимания клиентов, предоставление информации о продуктах, обслуживание покупателей и заключение сделок. Этот процесс также включает в себя поддержку клиентов после совершения покупки, чтобы обеспечить их удовлетворенность и лояльность к бренду.

Исходя из этого, можно выделить основные бизнес-процессы компании – закупка, складирование запасов, продажа, взаиморасчеты с поставщиками и клиентами.

Основываясь на этом анализе, можно разработать эффективные стратегии управления, оптимизировать процессы и повысить конкурентоспособность компании на рынке одежды.

Оформление поставки одежды на склад магазина сопровождается оформлением следующих документов:

1. заказ к поставщику;
2. приходная накладная;
3. расходная накладная;
4. счет поставщика.

При оформлении продажи используются следующие документы:

1. заказ клиента;
2. счет фирмы;
3. расходная накладная.

Работа магазина запчастей регламентируются следующими нормативными документами:

1. Устав Организации;
2. Закон «О защите персональных данных»;
3. Закон «О защите прав потребителей»;
4. Гражданский кодекс РФ;
5. Налоговый кодекс.

В соответствии с законом «О защите прав потребителей» покупатель может вернуть не подошедшую одежду в течение 14 дней, если товар без брака (и просто разонравился, не подошел в итоге по размеру, оказался не подходящим к туфлям или сумке). Если же товар с браком, то вернуть его можно в течение всего гарантийного срока, который устанавливает производитель или продавец.

Условия возврата одежды в магазин достаточно лояльны по отношению к клиентам. Однако покупателю могут отказать в возврате, если вещь потеряла товарный вид – то есть была ношена (в данном случае имеется в виду не примерка, а полноценное ношение в течение длительного времени, что сказывается на состоянии изделия), на ней отсутствуют ярлыки или есть недостатки, возникшие в результате носки.

В общем случае продавец сначала предложит обменять неподходящую вещь на аналогичную другого размера или фасона. Если покупателю ничего не подойдет, тогда ему обязаны вернуть деньги.

По закону 54-ФЗ «О применении контрольно-кассовой технике» покупатели одежды в обязательном порядке получают чек.

В организационной структуре выделяются ряд подразделений, играющих ключевую роль в успешной реализации автоматизации бизнес-процессов. Можно выделить следующие отделы:

1. Отдел продаж: осуществляет активные продажи продукции компании, включая консультацию клиентов, оформление заказов, проведение торговых операций и поддержание отношений с клиентами.
2. Продавец: непосредственно занимается продажами в розничных магазинах или через онлайн-платформы, оказывает консультации покупателям, помогает с выбором товара и обеспечивает качественное обслуживание.
3. Менеджер склада: отвечает за операционную работу склада, включая прием и отгрузку товаров, их хранение, инвентаризацию, учет и организацию складских процессов для обеспечения эффективной работы логистики и удовлетворения потребностей клиентов.
4. Отдел закупок: занимается поиском и закупкой товаров у поставщиков, ведением переговоров, контролем качества и сроков поставок, а также поддержанием отношений с поставщиками.
5. Группа логистики: организует логистические процессы компании, включая управление поставками, транспортировку, складирование и управление запасами, с целью обеспечения оперативной и эффективной доставки товаров клиентам.
6. Бухгалтерия: осуществляет финансовый учет и отчетность, включая ведение бухгалтерских записей, составление финансовых отчетов, налоговую отчетность, обработку финансовых операций и платежей, а также контроль финансовых потоков в компании.

Полученную организационную структуру можно увидеть на рисунке 1.

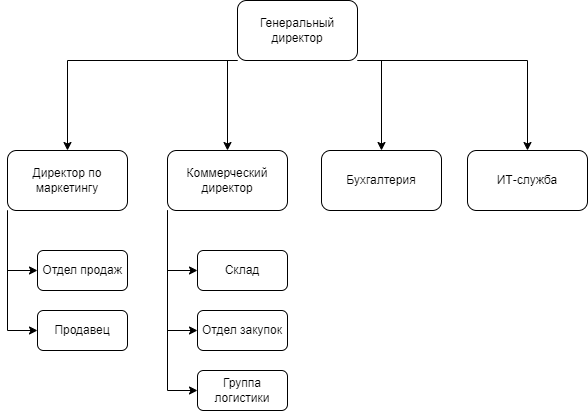


Рисунок – Диаграмма организационной структуры

## 1.2 Моделирование бизнес-процессов предметной области

Контекстная диаграмма IDEF0 представляет собой визуальное описание ключевых элементов и процессов, связанных с реализацией бизнес-процессов в организации. В случае компании «DrillDrip», диаграмма должна описывать процесс осуществления розничной торговли одеждой.

Входные данные включают в себя информацию о заказах клиентов и счетах от поставщиков. Для обработки этих данных требуется контроль со стороны действующего законодательства и внутренних распорядительных документов.

В результате обработки входных данных формируются выходные данные, включающие товар для клиента, счета от фирмы, приходные и расходные накладные, а также заказы к поставщикам.

Этот процесс осуществляется при помощи определенных механизмов, включающих отдел продаж, продавцов, менеджеров склада, отдел закупок, группу логистики и бухгалтерию. Каждый из этих элементов играет свою роль в цепочке процессов, направленных на обеспечение качественной автоматизации бизнес-процессов.

Полученную контекстную диаграмму можно увидеть на рисунке 2.

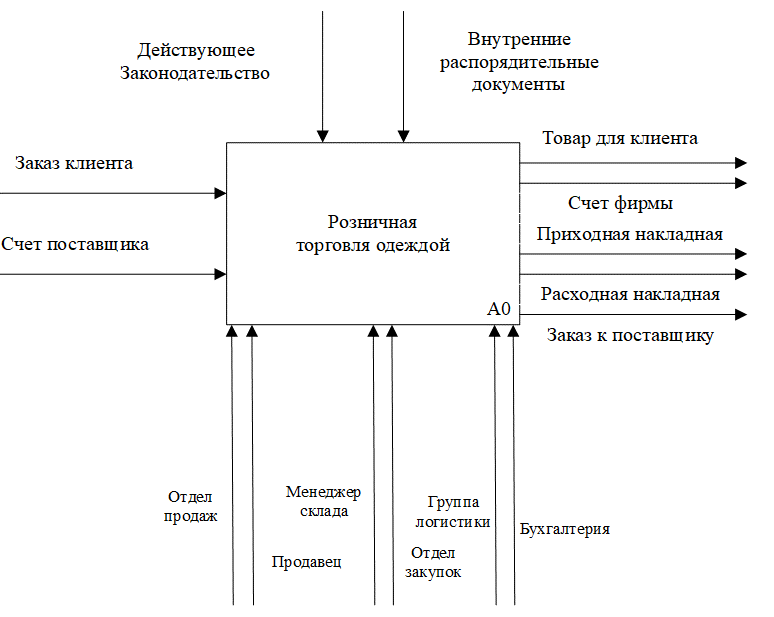


Рисунок – Контекстная диаграмма моделирования бизнес-процесса в нотации IDEF0

Декомпозиция контекстной диаграммы IDEF0 позволила выделить четыре ключевых бизнес-процесса, которые составляют основу операционной деятельности организации.

Первым важным процессом является продажи. В рамках данного процесса необходимо обеспечить клиентов всем необходимым для удобного и безопасного совершения покупок. Это включает в себя обеспечение информацией о продукции, возможность онлайн-консультаций с клиентами, а также удобную систему формирования заказов и выставления счетов. Главная цель этого процесса - обеспечить клиентов качественным и удобным сервисом на всех этапах их онлайн-покупок.

Далее следует запасы, склад. Этот процесс охватывает важные аспекты складского хозяйства, включая прием, размещение и отгрузку товаров. Организация эффективного складского процесса позволяет обеспечить своевременную доставку товаров и поддерживать оптимальный уровень запасов.

Третьим ключевым процессом является планирование и размещение заказов. В рамках этого процесса осуществляется снабжение организации необходимыми ресурсами и материалами, начиная от выявления потребностей и заключения контрактов с поставщиками, и заканчивая контролем качества поставляемых товаров.

И, наконец, четвертым процессом является логистика. Этот процесс описывает управление и координацию всех логистических операций, необходимых для эффективной доставки товаров клиентам, включая планирование маршрутов, управление запасами и организацию транспортировки.

Полученную декомпозицию бизнес-процессов можно увидеть на рисунке 3.

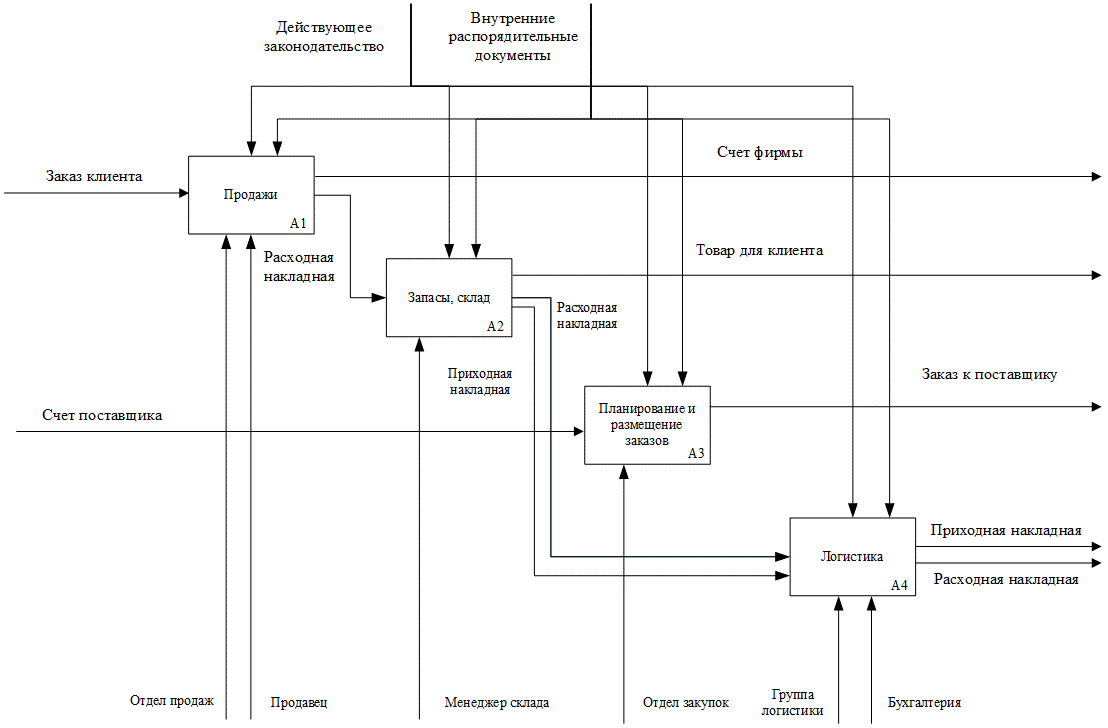


Рисунок – Декомпозиция бизнес-процессов в нотации IDEF0

Данная курсовая работа направлена на реализацию автоматизации бизнес-процесса продажи.

## 1.3 Проектирование графического пользовательского интерфейса

Информационная система для бизнес-процесса продажи обладает широким спектром важного для реализации функционала. В рамках данной курсовой работы были реализованы только ключевые элементы, включающие в себя:

* просмотр товаров в каталогах,
* фильтрация каталогов,
* просмотр информации о товаре,
* просмотр отзывов о товаре,
* создание профиля пользователями,
* возможность входа в существующий профиль,
* изменение данных о профиле,
* возможность добавление товаров в избранное,
* корзина с товарами пользователя,
* оформление заказа.

В данной информационной системе различаются следующие роли: гости и зарегистрированные пользователи.

Зарегистрированным пользователям доступен весь функционал системы. В то время как гости, не имея профиля, лишены возможности оформления заказов и отслеживания их статуса.

Также, для зарегистрированных пользователей система автоматически сохраняет информацию о содержимом их корзины и избранном в базе данных, обеспечивая постоянный доступ к этим данным независимо от устройства или места. В то время как у гостей эта информация хранится локально на их устройствах, что ограничивает возможности доступа и синхронизации данных при использовании различных устройств или при переходе на другие платформы.

Основным аспектом в проектировании графического интерфейса является анализ целевой аудитории. Понимание потребностей потребителей – это ключевой компонент успешной стратегии маркетинга и развития бизнеса. Проведение анализа целевой аудитории позволяет выявить привычки использования аналогичных продуктов и предпочтения в дизайне, что, в свою очередь, поможет в создание максимально удобного и интуитивно понятного интерфейса.

Решение данной задачи принимается различными методами: исследование рынка, выявление мнений и ожиданий, анализ фокус-групп с целевой аудиторией и изучение стратегий конкурентов.

В процессе разработки интерфейса для сайта основное внимание было уделено последнему методу, поскольку он оказался не только эффективным, но и относительно простым в реализации.

В ходе этого исследования многие варианты интерфейсов, реализованные конкурентами, оказались не подходящими для использования в собственных концепциях, так как они не соответствовали личному видению. Тем не менее, некоторые из них непосредственно повлияли на формирование конечной темы своего сайта. Среди таких сайтов: «Стокманн», «Lamoda», «Ozon», «Gerry Weber», «Grailed». Они помогли в разработке дизайна некоторых элементов, а самое главное – в понимании того, где обычно располагаются различные элементы сайта, чтобы обеспечить интуитивное понимание интерфейса.

В ходе тщательного исследования многие варианты интерфейсов, реализованные конкурентами, не соответствовали собственному видению и не подходили для интеграции в собственные концепции. Тем не менее, некоторые из них оказали прямое влияние на формирование общего дизайна сайта. Среди таких сайтов можно назвать «Стокманн», «Lamoda», «Ozon», «Gerry Weber», «Grailed». Они помогли определить существующие элементы сайта, а главное, сыграли ключевую роль в понимании того, как лучше располагать различные компоненты интерфейса для обеспечения интуитивно понятной навигации пользователя.

## 1.4 Проектирование структуры информационной системы

Веб-приложение онлайн-магазина должно предоставлять многофункциональную платформу, способную удовлетворить разнообразные потребности пользователей и обеспечить комфортный и удобный опыт онлайн-шопинга.

На странице каталога представлен обширный ассортимент товаров, а функционал фильтрации позволяет пользователям точно находить нужные им товары по различным критериям, таким как размер, цвет, бренд и другие характеристики.

Страница товара представляет полную информацию о выбранном товаре, включая его галерею изображений, подробное описание, технические характеристики и секцию с отзывами и комментариями от других пользователей, что помогает покупателям принять взвешенное решение о покупке.

На странице с избранными товарами пользователи могут сохранять интересующие их товары для дальнейшего рассмотрения и приобретения.

Страница корзины товаров позволяет пользователям просматривать и управлять товарами, которые они выбрали для покупки, а также оформлять заказ.

Страница авторизации и регистрации предоставляет возможность пользователям войти в свой аккаунт или создать новый, используя как адрес электронной почты, так и номер телефона.

Страница профиля позволяет пользователям управлять своими персональными данными, редактировать личную информацию.

Страницы оформления заказа предоставляют интерфейс для заполнения информации о доставке и оплате заказа.

## 1.5 Разработка диаграммы использования

Диаграмма использования представляет собой визуальное описание функциональности системы, ориентированное на взаимодействие между актерами и системой.

Диаграмма использования строится на основе спроектированной структуры информационной системы. Это позволяет более подробно понять необходимый в реализации функционал. Полученную диаграмму использования можно увидеть на рисунке 4

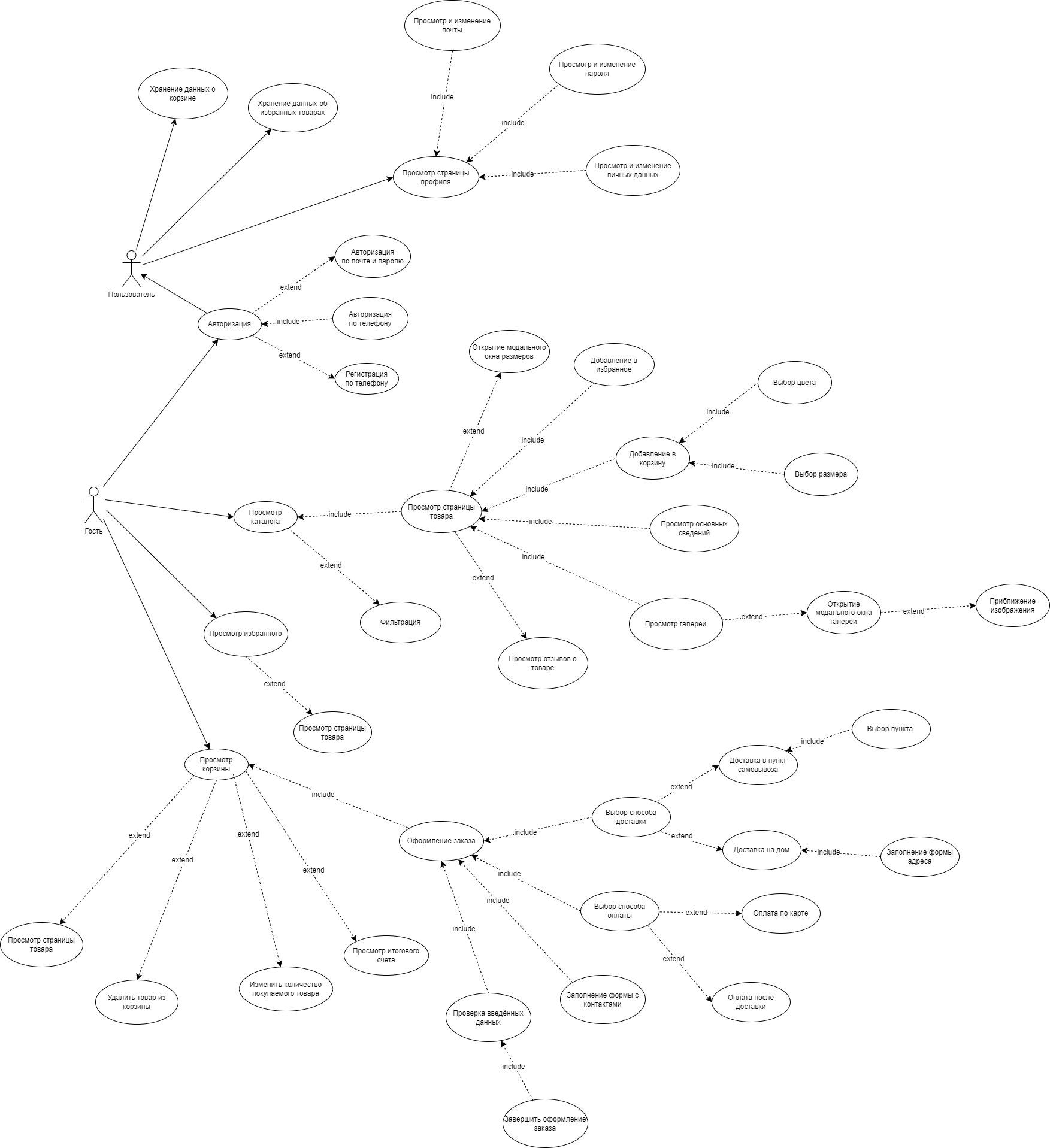


Рисунок – Диаграмма вариантов использования

## 1.6 Разработка ER-диаграммы информационной системы

ER-диаграмма, это визуальное представление структуры базы данных, обычно используемое в проектировании информационных систем.

Начать проектирование базы данных рекомендуется с товаров, поскольку они играют ключевую роль в системе. Каждый товар обладает уникальным названием и описанием, а также привязан к определенному бренду, который в свою очередь имеет собственное название и логотип. Товары также могут быть помечены тегами, каждый из которых требует хранения своего названия. Галерея изображений товара фиксирует положение каждой фотографии в ней. Товары относятся к определенной категории и обладают своими уникальными свойствами. На каждый товар может быть оставлен отзыв, который включает в себя дату написания, рейтинг, текст и информацию о пользователе. Каждый товар имеет хотя бы один цвет и размер, влияющие на его цену и доступные скидки. Размеры могут быть разных типов, например, европейские или российские. Каждый цвет имеет свое название и должен быть представлен на интерфейсе.

Зарегистрированный пользователь является следующим ключевым элементом. Его номер телефона обязателен для регистрации, а электронная почта и пароль - опциональны для альтернативной авторизации. Имя и фамилия пользователя необходимы для оформления заказа. Указание даты рождения позволяет получить праздничные скидки. Также пользователь может загрузить аватар для отображения в своих отзывах. Информацию об избранных товарах и корзине пользователя также необходимо сохранять.

Последним важным аспектом информационной системы является заказ. Важно точно определить, какие товары были заказаны и в каком объеме. Каждый заказ имеет свой статус выполнения, и этих статусов может быть множество, отражая различные этапы обработки. Не менее значимой частью заказа является его место доставки. Важно учитывать разнообразие методов доставки, включая возможность самовывоза и доставку на дом. При выборе самовывоза требуется указать соответствующее местоположение для забора товара, в то время как для доставки на дом необходимо предоставить адрес получателя. Для доставки на дом также важно определить удобную дату и время доставки, учитывая доступные временные слоты в разные дни.

Проведение тщательного анализа данных, необходимых для реализации информационной системы, позволяет разработать эффективную ER-диаграмму.

Полученную ER-диаграмму информационной системы можно увидеть на рисунке 5

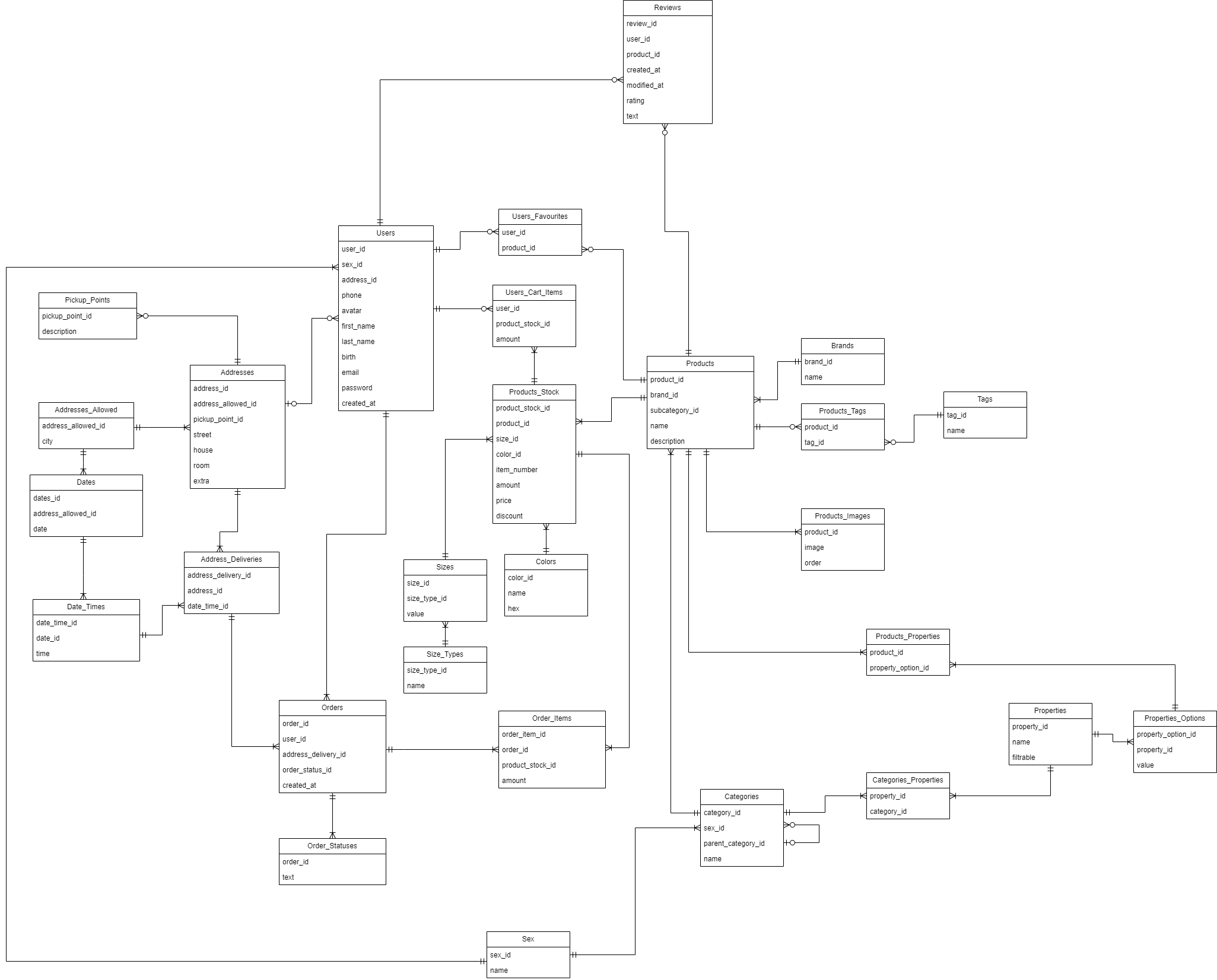


Рисунок – ER-диаграмма информационной системы

# 2 РЕАЛИЗАЦИЯ ВЕБ-ПРИЛОЖЕНИЯ СО СТОРОНЫ КЛИЕНТА

## 2.1 Реализация физической модели базы данных

Перед тем, как приступить к разработке клиентской части веб-приложения, необходимо провести детальное моделирование физической структуры базы данных и детально описать используемые типы данных для каждой из её таблиц. Это важный этап, поскольку позволяет определить, какие данные будут динамическими и как именно они будут взаимодействовать между собой.

Этот анализ имеет ключевое значение для последующего проектирования компонентов веб-сайта, поскольку от него зависит эффективность потока данных между самими компонентами. Тщательное понимание структуры базы данных обеспечивает более гибкую и эффективную работу с данными на клиентской стороне приложения.

Выбор типа данных для основных ключей таблиц и полей, обозначающих количество товаров, основан на нескольких факторах. Использование типа данных INT для ключей таблиц обосновано его способностью хранить целочисленные значения в диапазоне до 2,147,483,647, что обычно более чем достаточно для идентификации уникальных записей в базе данных. Кроме того, INT обеспечивает эффективность при выполнении операций сравнения и сортировки, что важно для производительности запросов.

Применение типа данных DECIMAL для цены товара и скидки обосновано необходимостью точных математических вычислений, связанных с денежными значениями. DECIMAL предоставляет возможность хранить числа с фиксированной точностью, что исключает проблемы с округлением, возникающие при работе с числами, хранящимися в двоичной системе исчисления. Точные операции с деньгами критически важны для финансовой точности и избегания нарушения данных из-за неточных вычислений, поэтому DECIMAL является предпочтительным выбором для хранения цен и скидок.

Большинству текстовых полей был назначен тип данных VARCHAR. Тип данных VARCHAR используется для хранения строковых значений переменной длины. Это означает, что длина строки может изменяться в пределах определенного максимального размера, который указывается при определении поля. VARCHAR эффективно использует память, поскольку он выделяет место только для реально используемых символов, а не резервирует фиксированное количество памяти для каждого поля, как это делает тип данных CHAR.

В контексте текущей информационной системы лимиты количества символов для текстовых полей были установлены с учётом конкретных требований и логики приложения. Например, лимит для описаний и названий был определен на основе предполагаемой длины этих текстовых значений в рамках приложения. Это позволяет оптимизировать использование памяти и обеспечить соответствие размеров данных ожидаемым потребностям приложения. Лимиты для остальных полей были выбраны с учётом разумных пределов, чтобы обеспечить достаточную гибкость для будущего расширения данных, но при этом избежать излишней перегрузки базы данных.

Однако, некоторым полям был назначен тип данных CHAR. Тип данных CHAR используется для хранения строковых значений фиксированной длины. При определении поля CHAR выделяет фиксированное количество памяти для каждого значения, даже если строка, хранящаяся в поле, короче этой длины. Это может привести к избыточному использованию памяти, но может быть полезно в случаях, когда все значения в поле имеют одинаковую длину.

Таким образом, тип данных CHAR отлично подходит для хранения паролей, так как предполагается их шифрование с фиксированным количеством символов. В случае таблицы цветов, HEX-значения всегда имеют фиксированную длину в шестнадцатеричном формате, поэтому CHAR в данном случае также предоставляет эффективное и логичное решение для хранения таких данных.

Тип данных TIMESTAMP был выбран для хранения точных значений даты и времени. TIMESTAMP сохраняет информацию о дате и времени до микросекундной точности. Он идеально подходит для сценариев, где требуется точное отображение временных меток, таких как дата и время создания или обновления записей в базе данных. Это также позволяет точно отслеживать последовательность событий, что может быть критически важно для приложений, работающих с временными данными. Например, для времени создания отзыва или оформления заказа.

Тип данных DATE используется для хранения только даты без времени. Этот тип данных был выбран, когда приложению не требуется сохранять информацию о времени, а только о датах. DATE обеспечивает удобное хранение и манипулирование с календарными датами без учета времени. В случае текущей информационной системы такой тип данных пригодился для хранения доступных дат доставки товара.

Тип данных TIME используется для хранения времени в пределах 24 часов без даты. Он сохраняет информацию о времени с точностью до микросекунд. Этот тип данных подходит, когда приложению необходимо сохранить информацию о времени, но не о дате. В контексте текущей информационной системы TIME используется для хранения возможного времени доставки.

Полученную физическую модель базы данных можно увидеть на рисунке 6.

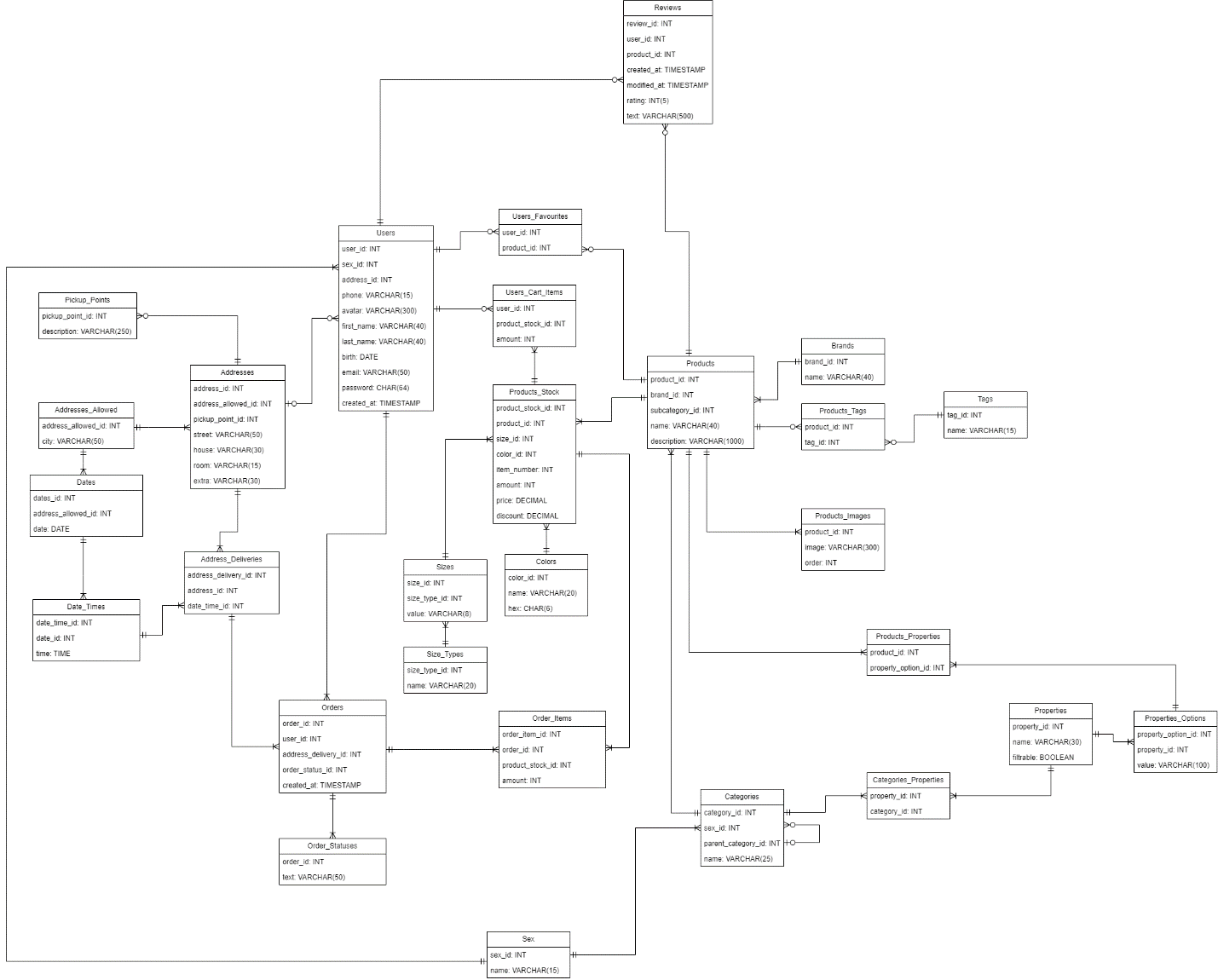


Рисунок 6 – Физическая модель базы данных

## 2.2 Выбор и обоснование используемого программного обеспечения

Перед началом работы над реализацией веб-приложения необходимо определиться с редактором кода. Для программиста наилучшим вариантом будет выбор именно IDE, а не обычных редакторов текста.

Интегрированная среда разработки (Integrated Development Environment, IDE) представляет собой программное обеспечение, которое облегчает разработку программного кода, предоставляя разработчикам удобные инструменты и функциональность для написания, отладки и тестирования кода. IDE обычно включает в себя текстовый редактор, компилятор, отладчик, автоматизированные инструменты сборки и другие инструменты, которые делают процесс разработки более продуктивным и эффективным.

В качестве IDE для реализации информационной системы был выбран Visual Studio Code. Интерфейс приложения можно увидеть на рисунке 7.

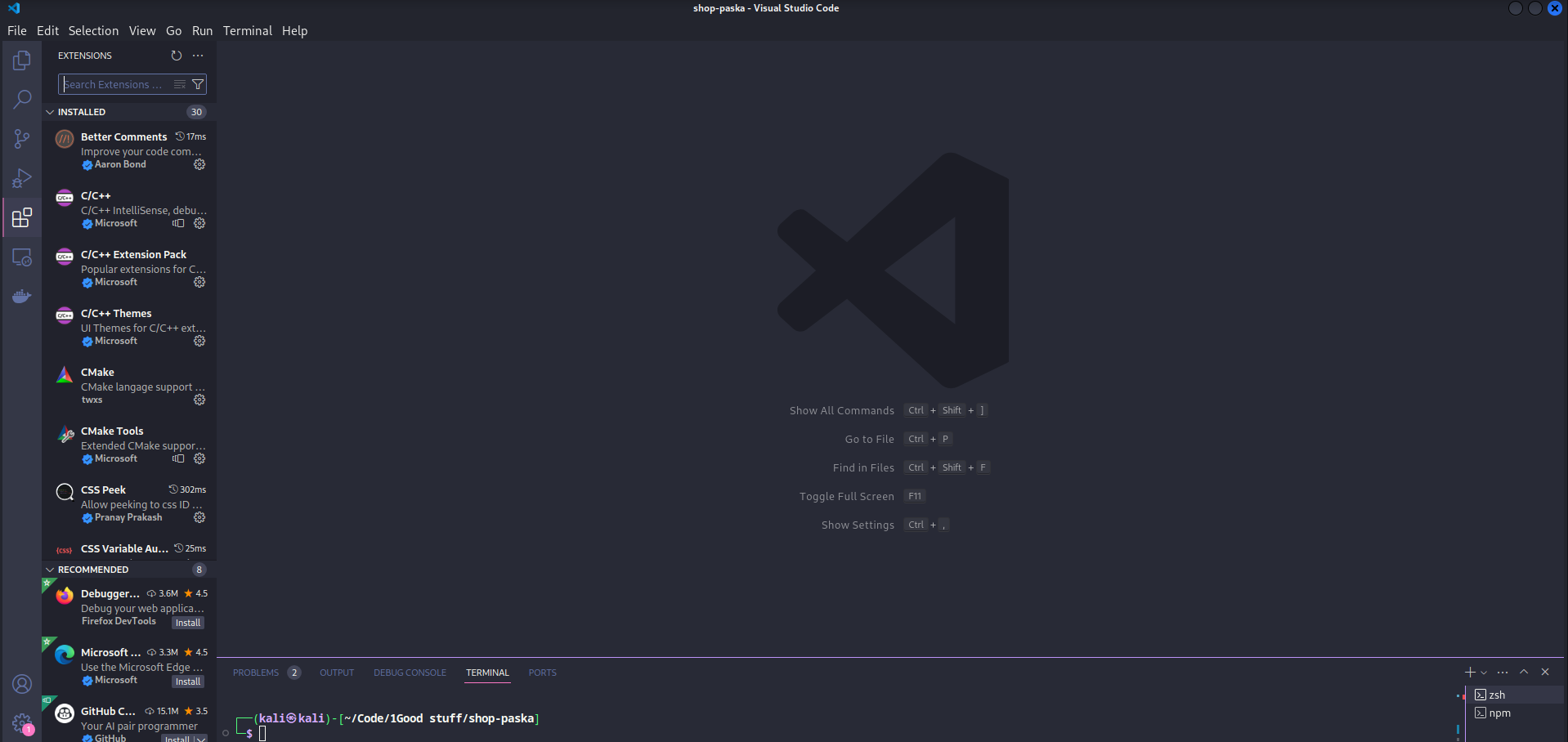


Рисунок 7 – Интерфейс Visual Studio Code

Выбор данного программного обеспечения обоснован следующими преимуществами:

* Бесплатность: Visual Studio Code предоставляется абсолютно бесплатно. Без необходимости оплаты лицензионных сборов или подписок, можно получить мощное инструментальное средство для создания программного обеспечения без дополнительных расходов,
* Простота использования: Visual Studio Code разработан с упором на простоту использования. Его удобный и интуитивно понятный интерфейс делает работу с кодом легкой и приятной,
* Возможность расширения функционала: одним из ключевых преимуществ Visual Studio Code является его огромное сообщество разработчиков и обширный каталог плагинов. В приложении существует возможность находить и устанавливать различные расширения, которые значительно расширяют функциональность Visual Studio Code и делают его подходящим для различных задач и технологий,
* Гибкие настройки приложения: Visual Studio Code позволяет пользователям настраивать и адаптировать среду разработки под свои потребности. С помощью настроек и конфигураций можно изменять внешний вид интерфейса, определять поведение редактора, настраивать расширения и многое другое.
* Функциональность: Visual Studio Code обладает множеством полезных функций, которые помогают улучшить процесс разработки. Это включает в себя быструю навигацию по файлам и проектам, встроенные инструменты для отладки кода, интеграцию с системами контроля версий, автоматическое форматирование кода и многое другое,

Одной из основных проблем в работе над большими проектами является эффективное управление изменениями в коде. При допущении ошибки в написании кода существует опасность нарушения работы всего приложения. Нахождение источника проблемы зачастую становится очень трудоемким и время затратным процессом, снижающим скорость разработки приложения.

Для решения подобных проблем были созданы инструменты контроля версий. В рамках курсовой работы было использовано программное обеспечение Git. Оно предоставляет мощные инструменты для организации рабочего процесса и управления изменениями в коде.

Git решает ряд проблем, возникающих при создании клиентской части веб-приложения, среди которых:

1. Управление версиями кода: Git позволяет разработчикам отслеживать изменения в коде на протяжении всего проекта. Это значит, что можно легко вернуться к предыдущим версиям кода, если что-то идет не так, или просмотреть историю изменений для понимания эволюции проекта.
2. Работа с ветками: Одной из ключевых особенностей Git является возможность работы с ветками. Разработчики могут создавать отдельные ветки для различных задач, что позволяет изолировать изменения и тестировать новый функционал, не затрагивая основной код.
3. Управление конфликтами: В процессе работы нередко возникают конфликты при слиянии изменений. Git предоставляет инструменты для разрешения этих конфликтов, что помогает сохранить целостность кодовой базы.
4. Откат изменений: Иногда в процессе разработки возникают ситуации, когда нужно откатить определенные изменения. Git позволяет легко отменить коммиты или вернуться к предыдущему состоянию проекта.

## 2.3 Разработка дизайна веб-приложения

Выбор темы сайта – это основная задача при разработке дизайна веб-приложения. Учитывая, что целевая аудитория связана с субкультурой уличной жизни, было принято решение оформить цвета сайта в черно-белом стиле. Это решение было также мотивировано тем фактом, что другие сайты, специализирующиеся на продаже одежды в стиле drill выбрали такую же стратегию для построения цветовой схемы.

Средне-темный оттенок сине-голубого был выбран в качестве основного цвета для сайта, отражая стиль и атмосферу, которые характерны для субкультуры уличной жизни. Дополнительные цвета для сайта были подобраны в соответствии с правилами аналогового сочетания цветов, чтобы создать гармоничный и привлекательный общий вид интерфейса.

Выбранные цвета для темы сайта можно увидеть на рисунке 8.



Рисунок 8 – Выбранные цвета для темы сайта

Для сайта были выбраны не стандартные шрифты, чтобы подчеркнуть его уникальность.

Шрифт без засечек, Finlandica, был выбран для использования в большинстве элементов сайта. Его чистые линии и современный вид вписываются в общий стиль и атмосферу сайта. Его можно увидеть на рисунке 9.

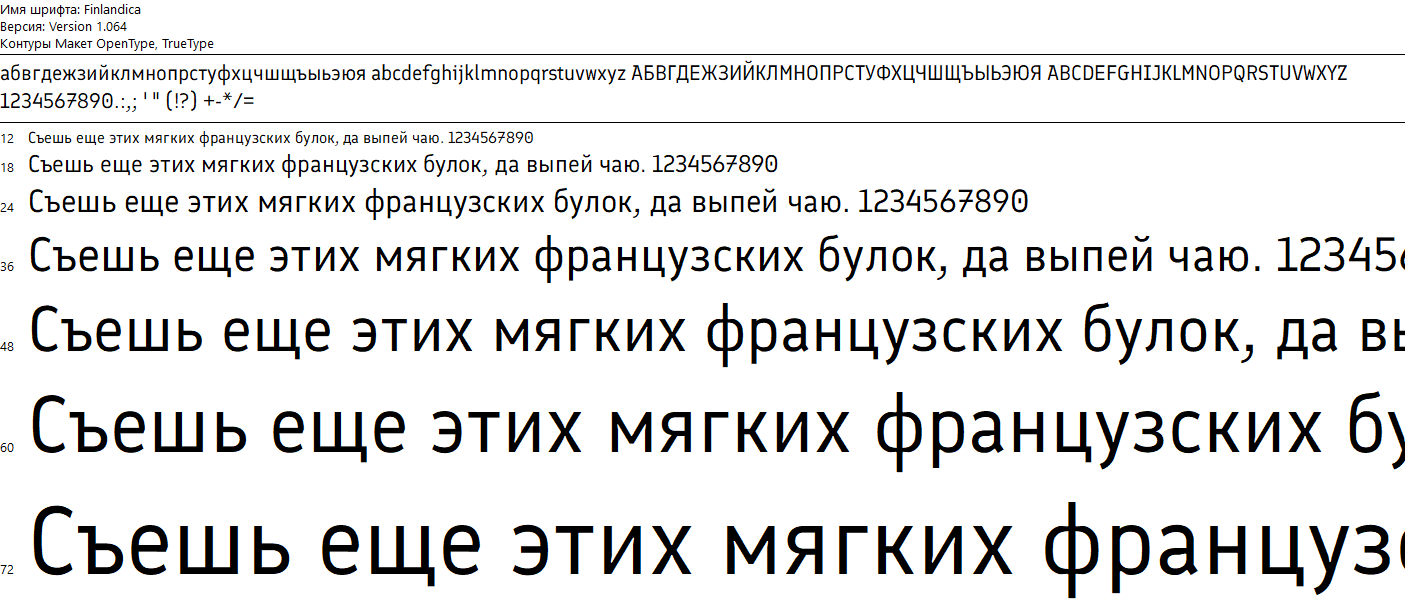


Рисунок 9 – Шрифт Finlandica

Шрифт с засечками, Brygada 1918, был выбран для использования в местах, где предполагается наличие длинных текстов, таких как комментарии пользователей, описания товаров и прочее. Его четкий стиль добавляет читаемости текстовым блокам, сохраняя при этом стиль и атмосферу сайта. Его можно увидеть на рисунке 10.

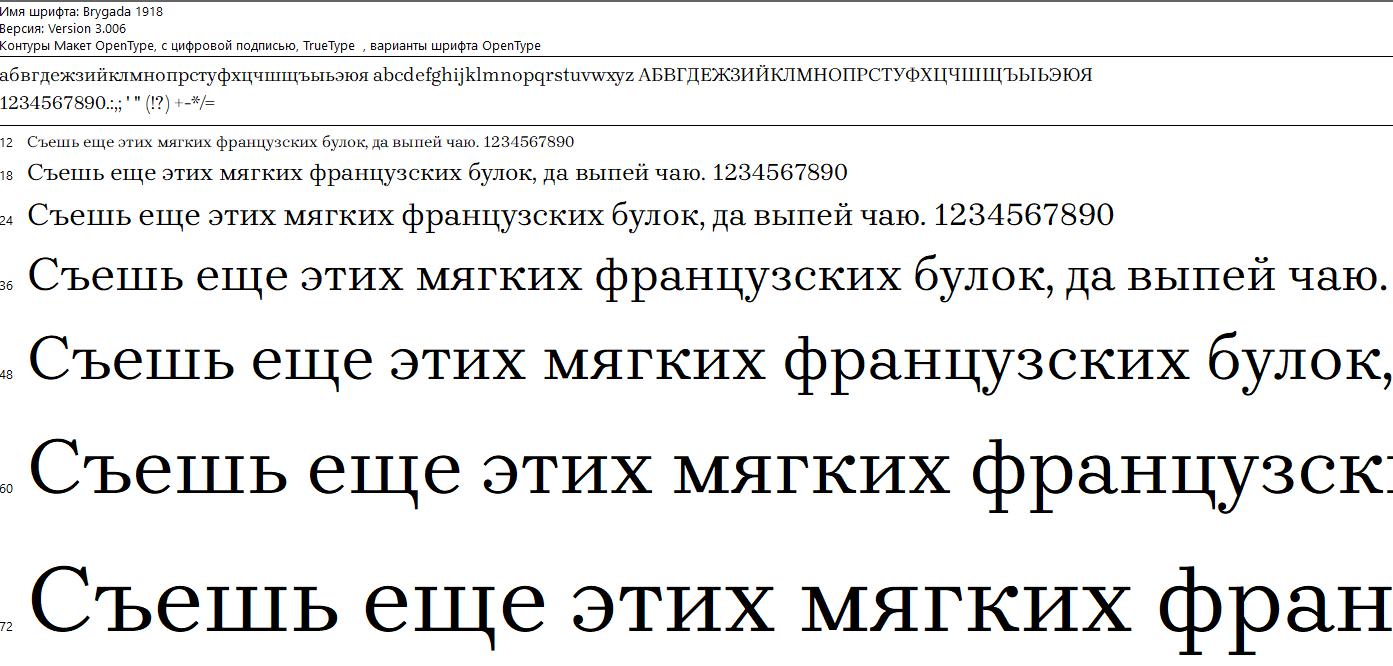


Рисунок 10 – Шрифт Brygada 1918

Такой подход к выбору шрифтов помогает подчеркнуть уникальность и индивидуальность сайта, а также вызвать впечатление у пользователей.

## 2.4 Разработка интерфейса взаимодействия с пользователем

Следующим этапом в разработке интерфейса стало создание макетов компонентов. Внимание было уделено системе дизайна Material Design 3.

Material Design 3 - это последняя версия дизайн-системы, разработанной компанией Google для создания современных и интуитивно понятных пользовательских интерфейсов. Она представляет собой эволюцию предыдущих версий Material Design и включает в себя современные концепции, принципы и компоненты, которые помогают разработчикам и дизайнерам создавать более привлекательные и удобные веб-сайты и приложения.

Все разработанные макеты компонентов для сайта стараются соответствовать рекомендациям Material Design 3. Это позволяет обеспечить пользователям высококачественный и современный опыт пользования, благодаря интуитивно понятным интерфейсам, четким анимациям и консистентному дизайну. Следование рекомендациям Material Design 3 помогает создать гармоничное и привлекательное визуальное впечатление, улучшая удобство использования сайта для каждого пользователя.

Сформированную на основе Material Design 3 цветовую палитру можно увидеть на рисунке 11.



Рисунок 11 – Цвета, используемые компонентами

Компоненты кнопок по системе Material Design 3 существуют трех видов: залитые, обведенные и текстовые. Полученные макеты кнопок в разных состояниях можно увидеть на рисунке 12.

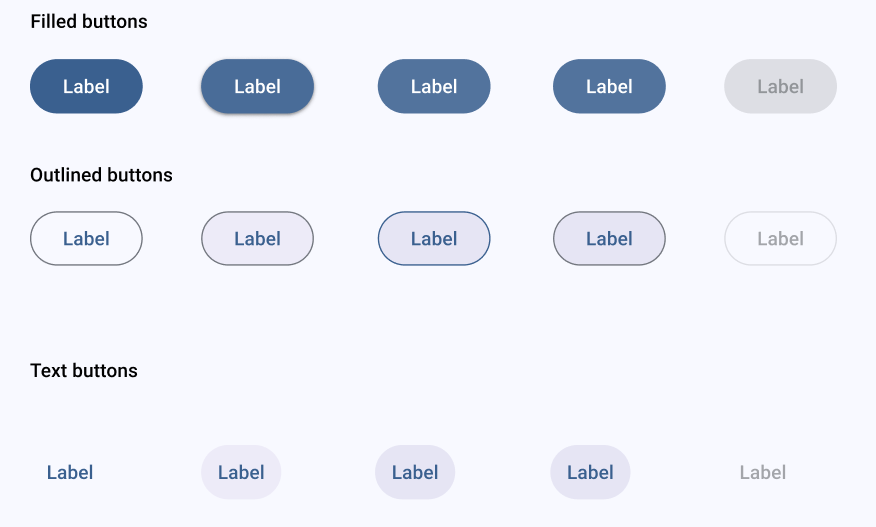


Рисунок 12 – Макеты компонентов кнопки в разных состояниях

Компоненты флажков в Material Design 3 исполнены в виде квадратов с закруглёнными углами. При нажатии на квадрат он заливается основным цветом, и становится виден флажок. Полученный макет можно увидеть на рисунке 13.



Рисунок 13 – Макеты компонентов флажка в разных состояниях

Компоненты кнопок с иконками внутри в Material Design 3 используются в 4 стилях: обычные, залитые, тональные и обведенные. Реализацию таких макетов можно увидеть на рисунке 13.

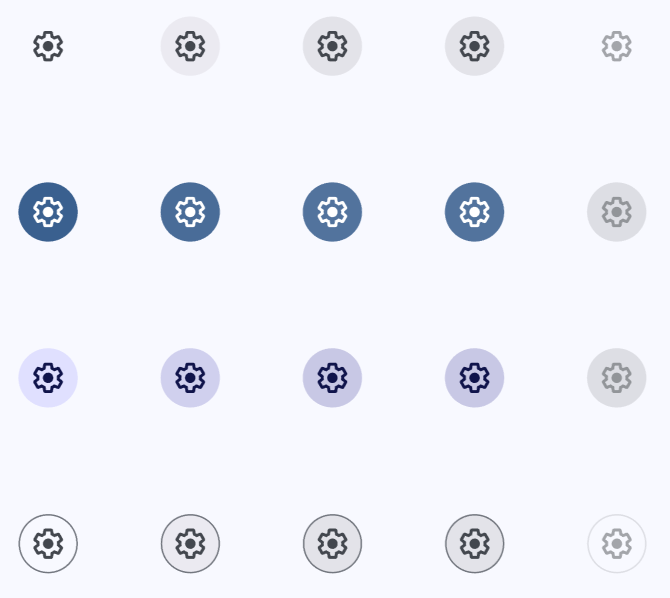


Рисунок 14 – Макеты компонентов кнопок-иконок в разных состояниях

Поля ввода в Material Design 3 созданы в двух разных стилях, причем рекомендуется придерживаться только одного из них. На сайте было решено использовать обведенный стиль полей ввода. Полученный результат можно увидеть на рисунке 15.

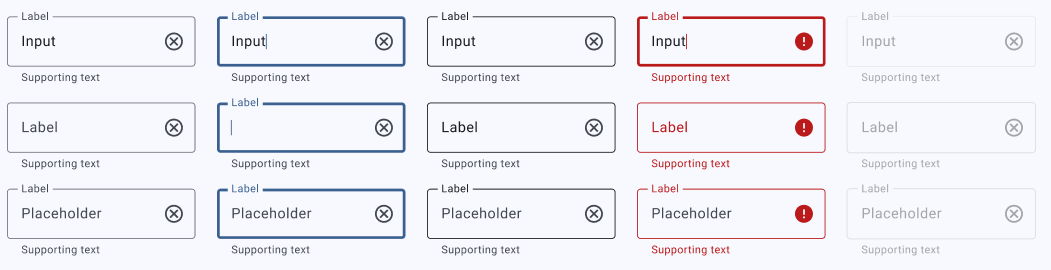


Рисунок 15 – Макеты компонентов ввода

## 2.5 Разработка функционала клиентской части приложения

Перед началом работы над функционалом клиентской части важно осуществить ряд подготовительных мероприятий.

Код сайта должен где-то исполняться. В обычном случае средой исполнения является браузер. Тем не менее, в процессе разработки клиентской части приложения, полагаться исключительно на браузер оказывается непрактичным. Браузерная среда ограничена в своей способности изолировать код, что усложняет процесс отладки и тестирования, особенно при работе над большими проектами. Кроме того, клиентская часть приложения часто взаимодействует с внешними сервисами и приложениями, и важно иметь контроль над этими взаимодействиями, что не возможно в браузерной среде.

Поэтому в качестве среды разработки был выбран Node.js. Node.js представляет собой среду выполнения JavaScript на сервере, что позволяет разработчикам выполнять и отлаживать код сайта вне контекста браузера. Он предоставляет множество преимуществ, таких как:

* язык программирования JavaScript,
* поддержка мощных инструментов,
* возможность создания реального сервера.

Использование Node.js позволяет обеспечить гибкость и эффективность в разработке клиентской части веб-приложений, делая его предпочтительным выбором.

В процессе разработки неизбежно используются сторонние библиотеки, которые обогащают функциональность приложения. Однако, ручная установка таких библиотек может быть чрезвычайно время затратной задачей. Для решения данной проблемы были созданы пакетные менеджеры. В рамках создаваемой информационной системы было решено использовать пакетный менеджер NPM. Сделанный выбор обосновывается следующим рядом преимуществ:

1. Простота установки и обновления: NPM позволяет устанавливать и обновлять сторонние библиотеки с помощью простых команд в терминале. Это делает процесс управления зависимостями более эффективным и удобным.
2. Обширный репозиторий пакетов: NPM имеет огромный репозиторий библиотек и модулей, которые можно легко искать и использовать в своих проектах. Это позволяет разработчикам быстро находить нужные инструменты и решения для своих задач.
3. Возможность автоматического разрешения зависимостей: NPM автоматически разрешает зависимости пакетов, что упрощает процесс установки и обновления библиотек, а также предотвращает конфликты между версиями.

Работа над клиентской частью веб-приложения в рамках стандартных инструментов является крайне неэффективной. Написание кода в тегах HTML, помимо обычного неудобства, сопряжено с рядом опасностей.

Одна из них — глобальная область видимости. Код, размещенный в тегах, хоть и может быть разбит по разным файлам, на самом деле исполняется последовательно. Отдельный файл не создает свою область видимости, что приводит к потенциальной опасности нежелательного доступа к данным между различными частями кода, создавая опасность полного нарушения работы приложения. Были предприняты разные попытки решения проблемы, но каждый из них обладал серьезными недостатками. Например, исполнять код можно в области видимости функции, но из-за этого значительно затрудняется коммуникация между разными модулями системы.

В настоящее время было разработано эффективное решение — сборщик приложений. Один из самых известных — WebPack, который и используется в данном веб-приложении. Его наличие значительно улучшает процесс разработки по следующим причинам:

1. Управление зависимостями и модульность: Webpack позволяет разработчикам организовать свой код в модули и управлять зависимостями между ними. Это делает код более структурированным, облегчает его поддержку и обновление.
2. Сборка и оптимизация ресурсов: Webpack позволяет собирать и оптимизировать ресурсы, такие как JavaScript, CSS, изображения и другие файлы, что позволяет уменьшить размер и улучшить производительность веб-приложения.
3. Поддержка различных типов файлов: Webpack поддерживает различные типы файлов и позволяет использовать различные инструменты и языки, такие как TypeScript, Sass, Less и другие, что обеспечивает гибкость и удобство при разработке.
4. Разделение кода на чанки: Webpack позволяет разбить код на небольшие части, называемые чанками, которые загружаются по мере необходимости. Это позволяет ускорить начальную загрузку приложения и улучшить его производительность.
5. Расширяемость и плагины: Webpack обладает широкой экосистемой плагинов, которые позволяют расширить его функциональность и адаптировать под конкретные требования проекта. Это делает Webpack мощным инструментом для разработки и оптимизации веб-приложений.

JavaScript имеет большой недостаток – это слабо типизированный язык. Типы данных в таком виде языков программирования определяются в процессе выполнения кода. Это позволяет свободно менять содержимое переменных, что облегает процесс разработки. Однако, при работе над крупными проектами возникают сложности, поскольку из контекста кода не всегда понятно, какие данные будут храниться в переменной. Также, самые полезные функции IDE, такие как авто-завершение кода, становятся недоступными. Ситуацию ухудшают неявное приведение типов данных в JavaScript.

Для решения этой проблемы был разработан TypeScript, который представляет собой типизированным транскомпилятором JavaScript. Среди особенностей TypeScript следует отметить следующее:

1. Статическая типизация: TypeScript позволяет определять типы данных переменных и параметров функций на этапе написания кода, что помогает выявить и предотвратить ошибки до момента выполнения программы.
2. Улучшенная поддержка IDE: Благодаря статической типизации, IDE может предоставлять более точные подсказки и автозавершение кода, что улучшает процесс разработки и снижает количество ошибок.
3. Более читаемый код: TypeScript делает код более читаемым и понятным за счет явного указания типов данных, что упрощает поддержку проекта и его дальнейшее развитие.
4. Поддержка новых возможностей ECMAScript: TypeScript активно развивается и поддерживает новые возможности стандарта ECMAScript, позволяя использовать последние инновации в разработке.
5. Простая интеграция: TypeScript легко интегрируется с существующим JavaScript-кодом, что позволяет постепенно переходить на его использование без необходимости полной переписи приложения.

Благодаря данным перечисленным преимуществам TypeScript был использован при разработке клиентской части веб-приложения.

В создании клиентской части веб-приложения возникает непростая задача в необходимости учитывать то, что далеко не все браузеры готовы к встрече с самыми передовыми технологиями. Эта действительность создает серьезные ограничения в процессе разработки, ибо написание современного кода требует особого внимания, так как есть вероятность, что он не будет исполним в широком спектре браузеров. В результате, разработчики вынуждены тщательно взвешивать каждое решение, учитывать совместимость кода с разными платформами и уровнями поддержки.

SWC (Speedy Web Compiler) является незаменимым инструментом для разработчиков веб-приложений, поскольку он обеспечивает транскомпиляцию современного кода, написанного с использованием новых возможностей ECMAScript, в код, который поддерживается большинством браузеров. Среди проблем, решаемых SWC можно перечислить следующие:

1. Трансформация кода для совместимости с браузерами: SWC понимает и обрабатывает современный синтаксис JavaScript, включая последние возможности, предложенные стандартом ECMAScript. После компиляции SWC преобразует современный код на основе новых возможностей ECMAScript в код, который совместим с браузерами. Это включает в себя преобразование синтаксических конструкций, которые могут быть непонятны или не поддерживаться старыми версиями браузеров, в более понятные и совместимые формы.
2. Соответствие стандартам и рекомендациям безопасности: SWC учитывает рекомендации по безопасности и совместимости с браузерами при транскомпиляции кода. Это помогает разработчикам создавать надежные и безопасные приложения, не сталкиваясь с проблемами совместимости.
3. Интеграция с популярными инструментами: SWC интегрируется с различными инструментами разработки, такими как Webpack и Jest, что облегчает использование его в существующих рабочих процессах.
4. Минификация и улучшение производительности: Помимо компиляции, SWC предоставляет возможности для минификации кода и улучшения производительности, что позволяет создавать быстрые и эффективные веб-приложения.

При разработке веб-интерфейсов часто сталкиваются с необходимостью создания эффективных и гибких стилей для визуализации контента. Однако, традиционные CSS-стили могут быть ограничены в своих возможностях и сложны в поддержке. Для решения этой проблемы был использован PostCSS.

PostCSS - это мощный инструмент, позволяющий автоматизировать и улучшить процесс написания и обработки CSS-стилей. Вот несколько его преимуществ:

1. Модульность и гибкость: PostCSS предлагает модульную архитектуру, которая позволяет разработчикам использовать только те функции, которые им нужны. Это делает его гибким инструментом, который можно настроить под конкретные потребности проекта.
2. Плагины и расширяемость: PostCSS имеет огромное количество плагинов, позволяющих расширить его функциональность. Это позволяет разработчикам добавлять новые возможности и автоматизировать различные задачи, такие как автопрефиксирование, минификация и оптимизация стилей.
3. Совместимость с CSS: PostCSS совместим с существующими CSS-файлами, что позволяет использовать его постепенно, без необходимости полной переписи кода. Это облегчает интеграцию PostCSS в существующие проекты и переход на его использование.
4. Поддержка новых возможностей CSS: PostCSS активно развивается и поддерживает новые возможности стандарта CSS. Это позволяет использовать последние инновации в веб-разработке и создавать более современные и производительные стили.

При разработке веб-приложений часто встает вопрос о выборе подходящей технологии для создания пользовательского интерфейса. В этом контексте React выделяется как мощный инструмент, который значительно упрощает процесс разработки и обеспечивает высокую производительность веб-приложений. Его основные преимущества в следующем:

1. Компонентный подход: React основан на концепции компонентов, что позволяет разбить пользовательский интерфейс на множество небольших и переиспользуемых элементов. Это делает код более структурированным, понятным и легко поддерживаемым.
2. Виртуальный DOM: React использует виртуальный DOM для эффективного обновления пользовательского интерфейса. Это позволяет минимизировать количество операций с реальным DOM и улучшить производительность приложения, особенно при работе с большими объемами данных.
3. Однонаправленный поток данных: React пропагандирует однонаправленный поток данных, что облегчает понимание и отслеживание потока информации в приложении. Это помогает избежать неявных ошибок и делает код более надежным и предсказуемым.
4. Широкая экосистема: React имеет огромную экосистему инструментов, библиотек и компонентов, которые помогают ускорить процесс разработки и расширить функциональность приложения. Это включает в себя библиотеки управления состоянием, роутинга, тестирования и многие другие.
5. Поддержка JSX: React использует JSX - расширение языка JavaScript, которое позволяет описывать структуру пользовательского интерфейса в виде кода JavaScript. Это делает разработку интерфейса более удобной и эффективной, а также облегчает внедрение динамических данных и логики в пользовательский интерфейс.

При написании обычного CSS кода разработчики часто сталкиваются с рядом проблем, которые могут затруднить процесс создания и поддержки веб-приложений.

Одной из основных трудностей является сложность управления стилями. Обычный CSS код часто становится сложным и запутанным из-за не структурированности и недостатка организации. Это может привести к трудностям в понимании и поддержке кода, особенно в крупных проектах.

В стандартном CSS приходится повторять одни и те же стили для различных элементов, что приводит к дублированию кода и усложняет его поддержку, причем это также приводит к увеличению размера файлов CSS и ухудшению производительности загрузки страницы.

Поскольку различные браузеры имеют разные интерпретации CSS и его возможностей, разработчикам приходится тратить много времени на исправление ошибок и обеспечение совместимости со всеми популярными браузерами.

Написание правильного CSS – серьезное испытание в разработке веб-приложения, но для решения этой проблемы в рамках курсовой работы был использован Tailwind CSS.

Tailwind CSS – это мощный инструмент для создания пользовательских интерфейсов веб-приложений, который позволяет разработчикам стилизовать свои проекты без необходимости написания большого количества собственных CSS-стилей. Этот инструмент предлагает готовые компоненты и утилиты, которые можно легко применить для быстрой и эффективной разработки.

Одним из ключевых преимуществ Tailwind CSS является его превосходная сочетаемость с компонентами React:

1. Простота использования: Tailwind CSS предоставляет простые классы, которые можно применять к HTML-элементам прямо в JSX-коде React. Это позволяет разработчикам легко стилизовать компоненты, не выходя за рамки JavaScript-файлов.
2. Чистота кода: Вместо того чтобы создавать много CSS-кода для стилизации компонентов, разработчики могут использовать только те классы Tailwind, которые им нужны для каждого компонента. Это делает код более чистым и понятным, а также уменьшает объем CSS-файлов, что положительно сказывается на производительности приложения.
3. Гибкость и настраиваемость: Tailwind CSS предоставляет широкий набор настраиваемых компонентов и утилит. Разработчики могут легко настраивать стили под свои потребности, используя конфигурационные файлы Tailwind.
4. Оптимизация производительности: Tailwind CSS позволяет разработчикам оптимизировать производительность и размер конечного бандла приложения, благодаря модульной архитектуре и возможности настройки используемых стилей.
5. Поддержка адаптивного дизайна: Tailwind CSS предоставляет мощные средства для создания адаптивных интерфейсов, что делает его идеальным выбором для разработки веб-приложений, которые должны быть оптимизированы для разных устройств и разрешений экрана.

# 3 ТЕСТИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

## 3.1 Выбор стратегии тестирования программного продукта

Тестирование веб-приложений является крайне важным этапом в процессе разработки по нескольким причинам. Во-первых, оно позволяет обнаруживать и исправлять ошибки и дефекты в приложении еще на ранних стадиях разработки, что способствует повышению качества конечного продукта. Во-вторых, тестирование помогает уверенно внедрять новые функции и изменения, не боясь нарушить работу уже существующего функционала. Кроме того, тестирование обеспечивает надежность и стабильность работы приложения, что повышает удовлетворенность пользователей и уменьшает количество возможных проблем после выпуска приложения в продакшн.

Unit тесты - это тип тестов, который проверяет отдельные части кода, такие как функции, классы или модули, на корректность работы. Они позволяют изолировать и тестировать отдельные компоненты приложения независимо от других частей. Unit тесты обеспечивают быструю обратную связь о работоспособности отдельных блоков кода и улучшают его надежность и читаемость.

Jest - это популярный инструмент для написания и запуска unit тестов для JavaScript приложений. Он обладает простым и интуитивно понятным синтаксисом, что делает процесс написания тестов более приятным и эффективным. Кроме того, Jest предоставляет широкие возможности для автоматизации тестирования, включая запуск тестов в параллельном режиме и генерацию отчетов о выполнении.

E2E (End-to-End) тесты - это тип тестов, которые моделируют реальные сценарии использования приложения, начиная с отправки запроса от пользователя и заканчивая получением ответа от сервера. Они проверяют работу приложения как единого целого, включая взаимодействие между различными компонентами и слоями приложения. E2E тесты помогают убедиться, что все части приложения работают корректно вместе, что повышает надежность и качество приложения.

Cypress - это инструмент для написания и запуска E2E тестов для веб-приложений. Он обладает виртуальным браузером, позволяя проводить тесты в автоматическом режиме. Кроме того, Cypress предоставляет богатые возможности для взаимодействия с приложением в реальном времени во время выполнения тестов, что упрощает отладку и анализ результатов.

## 3.2 Проведение тестирования информационной системы

Unit тестирование клиентской части веб-приложения было проведено в трех этапах:

1. Тестирование функций-помощников: На этом этапе уделялось особое внимание проверке функций-помощников, которые предоставляли вспомогательные методы для работы с данными и выполнения различных задач. Они были тщательно протестированы с различными входных данных, чтобы убедиться в их корректности и надежности. Например, функции форматирования данных или математических операций проходили через разнообразные сценарии использования, чтобы гарантировать их правильную работу в различных условиях.
2. Тестирование функциональных компонентов: этот этап был сфокусирован на проверке функциональных компонентов, которые отвечали за визуальное отображение данных и пользовательский интерфейс. Использовались различные методы тестирования, включая манипуляции с DOM-деревом, для проверки функциональности компонентов в реальных условиях. Это позволило убедиться, что все компоненты работают корректно и отвечают на ожидаемые пользовательские действия.
3. Тестирование нефункциональных компонентов: На этом этапе мы сосредотачивались на тестировании нефункциональных компонентов, таких как кнопки. Был использован подход снапшотов, который позволяет сохранить текущее состояние компонентов и сравнивать его с ожидаемым результатом. Это помогло в обнаружение любых нежелательных изменений внешнего вида компонентов.

Результаты проведения unit-тестов можно увидеть на рисунке 16.

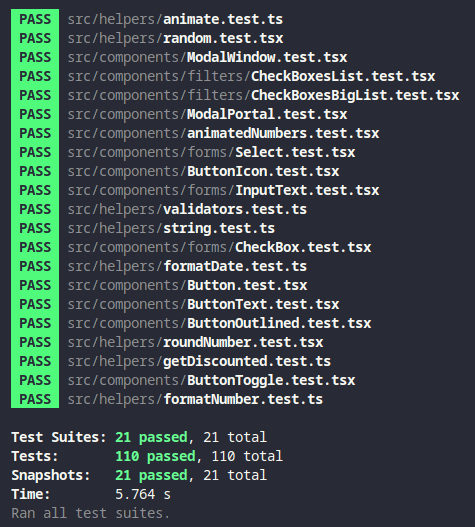


Рисунок – Результаты проведения Unit-тестов

E2E тестирование проводилось по определенным сценариям. Главная задача написанных сценариев – проверка работоспособности осинового функционала сайта. Для этого было проведено тестирование правильной работы функций добавления товаров в корзину, авторизации в систему и управлением профилем пользователя.

Проверка добавления товаров в корзину проводилась по следующим этапам:

1. Открывается основная индексная страница сайта.
2. Осуществляется переход на страницу товара по карточке товара.
3. Выбирается цвет товара.
4. Выбирается размер товара.
5. Нажимается кнопка добавления в корзину.
6. Кнопка добавления в корзину превращается в кнопку перехода в корзину.
7. Осуществляется переход в корзину по новой кнопке.
8. Товар должен оказаться в корзине.
9. Количество товара должно быть 1.
10. Кнопка увеличения количества товара работает. На карточке товара отображается 2.
11. Кнопка уменьшения количества товара работает. На карточке товара отображается 1.
12. Кнопка удаления товара работает. Карточка товара должна пропасть.

Результат проведения данного сценария можно увидеть на рисунке 17.



Рисунок – Результаты проведения тестирования корзины

Проверка авторизации в систему проводилась по следующим этапам:

1. Открывается основная индексная страница сайта.
2. Осуществляется переход на страницу авторизации.
3. Выбирается вид авторизации по почте.
4. Вводятся пароль и почта в нужные поля.
5. Нажимается кнопка авторизации.
6. Должен произойти переход на страницу пользователя.

Результат проведения данного сценария можно увидеть на рисунке 18.

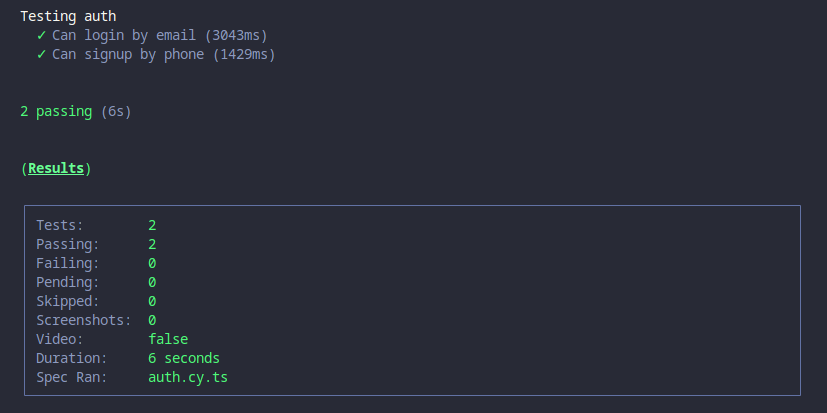


Рисунок – Результаты проведения тестирования авторизации

Проверка управления профилем пользователя проводилась по следующим этапам:

1. Открывается основная индексная страница сайта.
2. Осуществляется переход на страницу профиля.
3. Изменяются все личные данные пользователя.
4. Нажимается кнопка сохранения.
5. Осуществляется переход обратно на индексную страницу сайта.
6. Осуществляется переход на страницу профиля.
7. Введение ранее данные должны остаться.

Результат проведения данного сценария можно увидеть на рисунке 19.

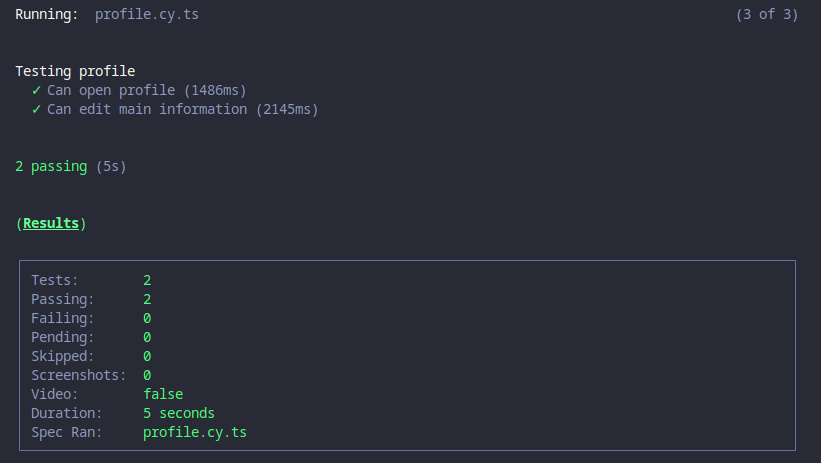


Рисунок – Результаты проведения тестирования профиля

ЗАКЛЮЧЕНИЕ

В ходе выполнения курсового проекта была рассмотрена актуальная тема разработки веб-сайта для онлайн магазина drill одежды. Исходя из социальной и практической значимости данного проекта, была сформулирована его цель – создание функционального и удобного в использовании веб-пространства для продажи товаров.

Для достижения поставленной цели были определены и решены следующие задачи: идентификация предметной области, моделирование бизнес-процессов, проектирование графического пользовательского интерфейса, разработка структуры информационной системы, создание диаграмм использования и ER-диаграммы. Далее была проведена реализация веб-приложения со стороны клиента, включающая в себя разработку физической модели базы данных, выбор и обоснование используемого программного обеспечения, а также создание дизайна, интерфейса и функционала клиентской части приложения. Наконец, была выбрана стратегия тестирования и проведено тестирование информационной системы.

В результате работы были получены сведения о том, как разрабатывается и тестируется веб-приложение для онлайн магазина. Полученные результаты позволили подтвердить актуальность темы исследования и достигнуть поставленной цели.

Таким образом, курсовой проект успешно закончен, и его результаты могут быть использованы для улучшения практического опыта в разработке веб-приложений для онлайн торговли.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Visual Studio Code – официальный сайт [Электронный ресурс]. Режим доступа: https://code.visualstudio.com/
2. Git – официальный сайт [Электронный ресурс]. Режим доступа: https://git-scm.com/
3. Node.js – официальный сайт [Электронный ресурс]. Режим доступа: https://nodejs.org/
4. NPM – официальный сайт [Электронный ресурс]. Режим доступа: https://www.npmjs.com/
5. TypeScript – официальный сайт [Электронный ресурс]. Режим доступа: https://www.typescriptlang.org/
6. WebPack – официальный сайт [Электронный ресурс]. Режим доступа: https://webpack.js.org/
7. PostCSS – официальный сайт [Электронный ресурс]. Режим доступа: https://postcss.org/
8. SWC – официальный сайт [Электронный ресурс]. Режим доступа: https://swc.rs/
9. React – официальный сайт [Электронный ресурс]. Режим доступа: https://react.dev/
10. Jest – официальный сайт [Электронный ресурс]. Режим доступа: https://jestjs.io/
11. Cypress – официальный сайт [Электронный ресурс]. Режим доступа: https://www.cypress.io/