

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	6
1 ОБЗОР ЛИТЕРАТУРЫ.....	7
1.1 Обзор существующих аналогов.....	7
1.2 Spring Framework.....	12
1.3 Язык программирования Java.....	13
1.4 Архитектура веб-приложений	13
1.5 Язык разметки HTML.....	16
1.6 База данных MySQL	16
2 Системное проектирование.....	17
2.1 Блок новостей.....	17
2.2 Блок отзывов.....	18
2.3 Блок базы данных.....	18
2.4 Блок администрирования	19
2.5 Блок контактов	19
2.7 Блок каталога товаров	20
2.8 Блок обратной связи	20
2.9 Блок работы с базой данных	20
3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ	22
3.1 Описание структуры приложения.....	22
3.2 Анализ диаграммы классов.....	23
4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ	40
4.1 Создание подкаталога внутри каталога.....	40
4.2 Создание и отправка отзыва о магазине.....	41
4.3 Создание продукта в подкаталоге	42
4.4 Создание заказа и работа с корзиной.....	45
4.5 Создание пользователя в системе	47
4.6 Добавление и просмотр контактов магазина:	48
5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ.....	49
6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ.....	53
6.1 Развертывание приложения и настройка сервера.....	53
6.2 Руководство по использованию программного средства.....	54
7 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЯ ИНТЕРНЕТ-МАГАЗИНА КОМПЬЮТЕРНОЙ ТЕХНИКИ	66
7.1 Определение трудоемкости разработки программного продукта	66
7.2 Определение стоимости машино-часа работы персонального компьютера.....	68
7.3 Определение себестоимости создания программного продукта	72
7.4 Определение оптовой и отпускной цены программного продукта	73
7.5 Определение годовых текущих затрат, связанных с эксплуатацией задачи.....	74
7.6 Определение годовых текущих затрат, связанных с эксплуатацией задачи, при использовании аналога	76
7.7 Определение ожидаемого прироста прибыли в результате внедрения программного продукта.....	77

ВВЕДЕНИЕ

Веб-приложения широко востребованы специализированными компаниями на территории РБ и за рубежом. В настоящее время веб-технологии активно развиваются в торговле, промышленности, учреждениях образования.

Для успешного продвижения товара на рынке большинство торговых компаний используют веб-технологии, так как в настоящее время широко распространена тенденция купли и продажи товара с помощью информационных технологий и с каждым годом число посетителей веб-приложений постоянно растет.

Веб-приложение позволяет размещать различную информацию о компании и её услугах не только в текстовом, но в графическом и мультимедийном формате, что позволяет клиентам получить полный объем информации о продукте или компании.

Веб-приложение является виртуальным офисом компании работающим круглосуточно. Актуальность создания веб-приложения заключается в том, что это самый удобный способ донести до огромной аудитории информацию о компании, её услугах и товарах сжато и в то же время полноценно.

Целью дипломного проекта является разработка веб-приложения интернет-магазина компьютерной техники, которое предоставит возможность выбирать товар по техническим характеристикам и категориям, просматривать отзывы клиентов о данном товаре, что бы иметь представление о достоинствах и недостатках конкретной модели со стороны обычного пользователя, так же для клиентов или потенциальных сотрудников будет реализована форма замечаний и предложений с обратной связью на почту компании. Кроме того, следует учитывать то, что ввиду развития информационных технологий с каждым годом посетители интернет ресурсов становятся всё более нетерпеливыми и требовательными. Конечному пользователю важен быстрый результат, поэтому не следует нагружать сайт бесполезным контентом и сложной для понимания архитектурой.

В соответствии с поставленной целью были определены следующие задачи:

- выбор технологий для создания системы;
- разработка административной и пользовательской частей системы;
- регистрация пользователей в системе;
- размещение товара на сайте администратором;
- поиск товара по названию;
- сортировка товара по параметрам и критериям;
- форма обратной связи.

1 ОБЗОР ЛИТЕРАТУРЫ

1.1 Обзор существующих аналогов

На этапе проектирования системы были тщательно изучены существующие аналоги.

1. *kst.by* [1] – веб-приложение интернет магазина цифровой и бытовой техники с постоянно расширяющимся ассортиментом каталога товаров на территории РБ. На сайте реализована простейшая система поиска цифровой и бытовой техники. Разработано окно замечаний и предложений, где пользователь сможет внести свою жалобу или предложение о деятельности организации. Главная страница приложения изображена на рисунке 1.1.

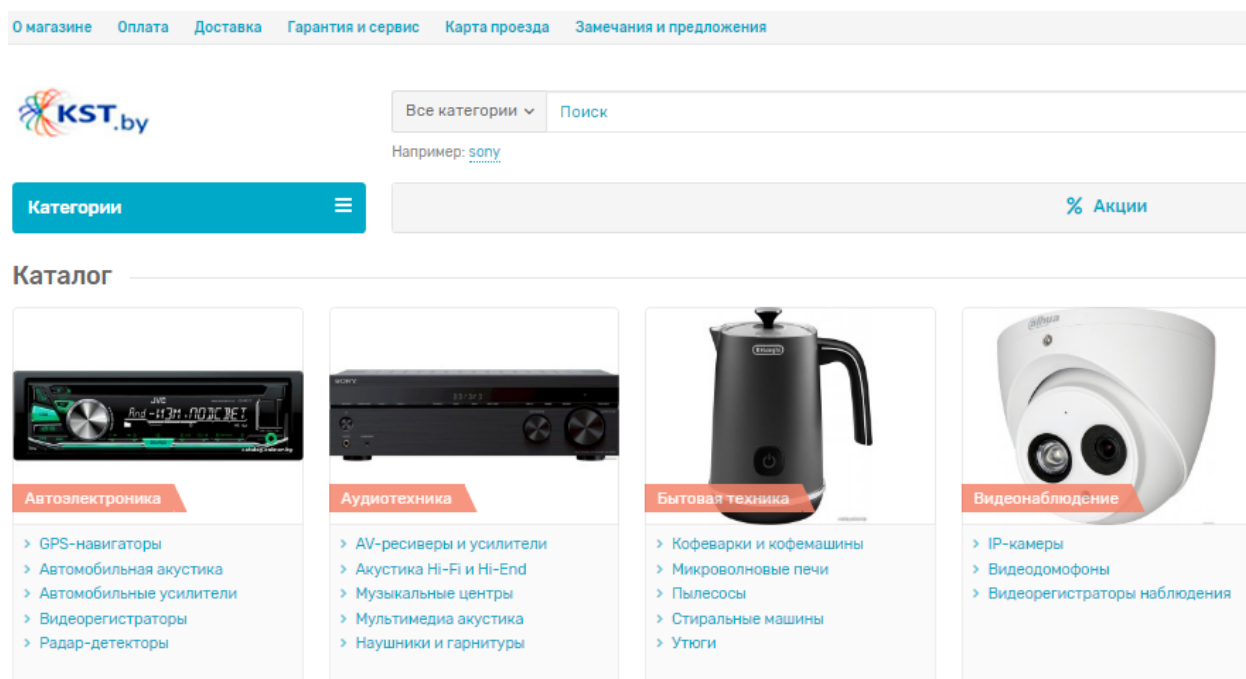


Рисунок 1.1 – Веб-приложение[1]

Программное средство не требует установки на устройство пользователя. Используемый язык программирования PHP, JavaScript. Приложение используют веб-сервис для получения данных.

Системные требования к программному обеспечению пользователей: браузер Microsoft Internet Explorer 9 и выше, Mozilla Firefox 16 и выше, Opera 12.0 и выше, Google Chrome, Safari.

Основные достоинства:

- поиск товара по категориям;
- поиск товара по акции;
- присутствует возможность сравнения товара;
- Форма обратной связи.

К основным недостаткам относятся:

– отсутствует возможность поиска товара по параметрам производительности.

2. *ttn.by* [2] – веб-приложение одного из крупнейших интернет магазинов. Более девяти лет на рынке, огромный каталог электронный, бытовых, офисных, спортивных товаров. Главная страница приложения изображена на рисунке 1.2.

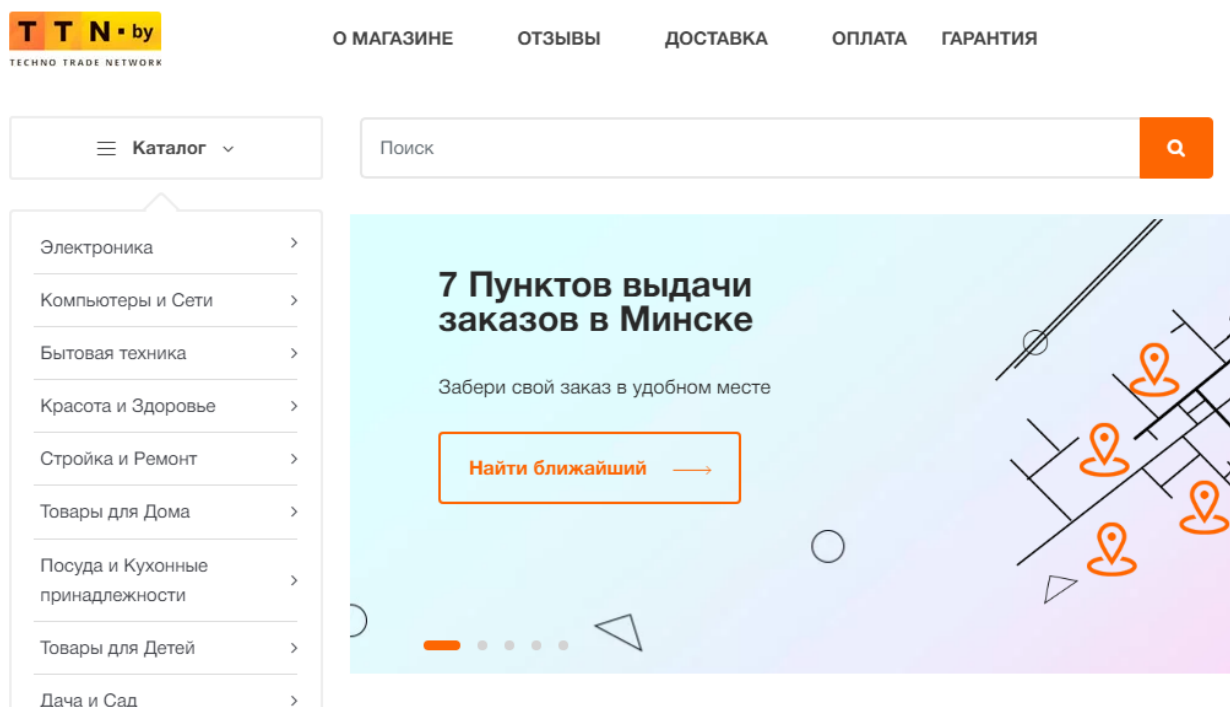


Рисунок 1.2 – Веб-приложение[2]

Используемые фреймворки Ruby on Rails, PHP. Приложение использует веб-сервис для получения данных.

Системные требования к программному обеспечению пользователей: браузер Microsoft Internet Explorer 9 и выше, Mozilla Firefox 16 и выше, Opera 12.0 и выше, Google Chrome, Safari.

Основные достоинства:

- поиск ближайших пунктов выдачи товара;
- поиск товара по категориям;
- администрирование системы;
- раздел отзывов под каждым товаром.

Основные недостатки приложения:

- отсутствует возможность фильтра товара по параметрам производительности;
- отсутствие возможности обратной связи;
- нет возможности сравнивать товар.

3. *bittrade.by* [3] – небольшое веб-приложение для интернет-магазина исключительно цифровой техники. Специализируется на продаже компьютеров и комплектующих для ПК (процессоры, видеокарты, материнские платы, жесткие диски), ноутбуки, мониторы, принтеры и другие периферийные устройства. В данном интернет-магазине можно выбрать и купить компьютер и комплектующие, по весьма доступным ценам. Главная страница приложения изображена на рисунке 1.3.

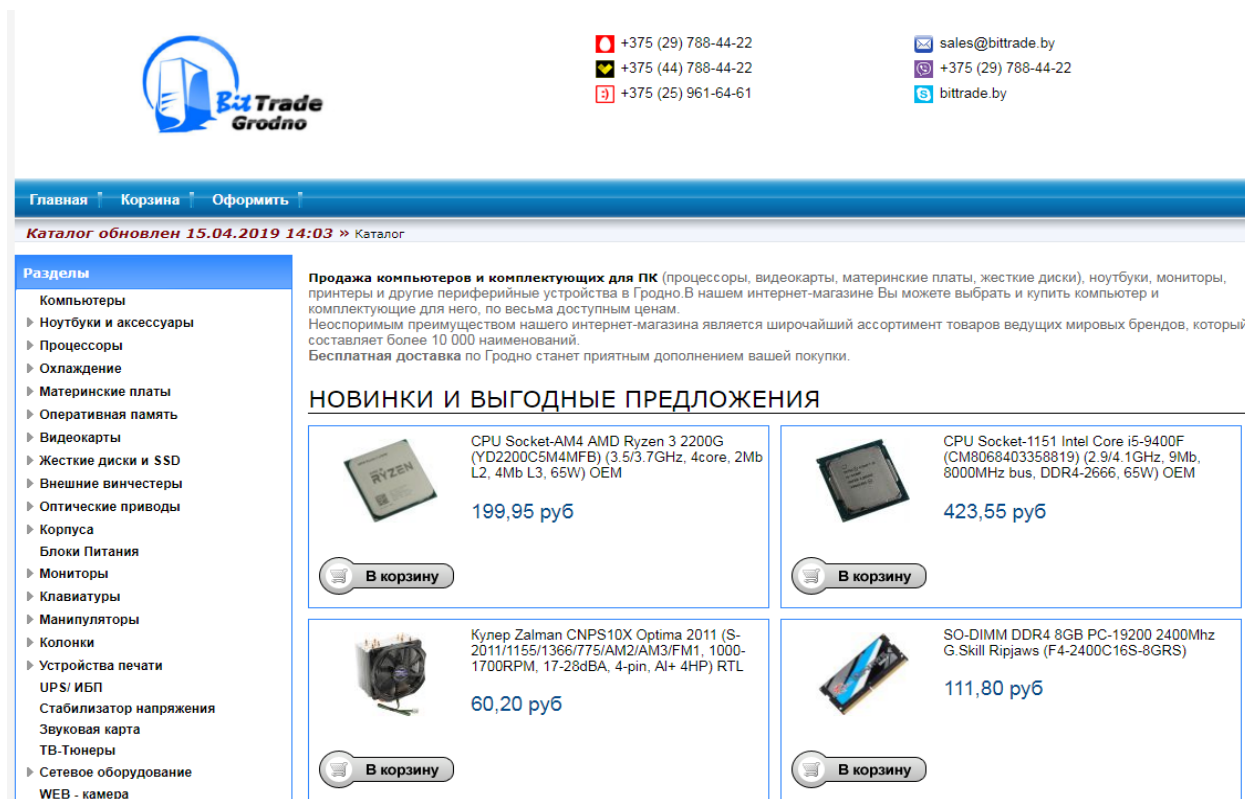


Рисунок 1.3 – Веб-приложение[3]

Используемый язык программирования PHP, JavaScript. Приложение использует веб-сервис для получения данных.

Системные требования к программному обеспечению пользователей: браузер Microsoft Internet Explorer 9 и выше, Mozilla Firefox 16 и выше, Opera 12.0 и выше, Google Chrome, Safari.

Основные достоинства:

- есть возможность поиска товара по категориям;
- поисковая строка для поиска по названию.

Основные недостатки приложения:

- устаревший графический дизайн;
- отсутствует административная часть приложения;
- отсутствует форма обратной связи с владельцами сайта;
- отсутствует возможность поиска товара по параметрам производительности.

4. *ram.by* [4] – Интернет-магазин фирмы, специализирующейся на продаже компьютеров и компьютерных комплектующих, основанный в 2009 году. Магазин удобен тем, что в режиме онлайн можно собрать компьютер по комплектующим, в нём есть онлайн-чат, с помощью которого можно в любое время задать сотрудникам вопрос и получить на него ответ. Так же для неподвинутых пользователей организован видеогид, который на практике показывает как быстро заказать нужный товар, а так же показывает схему проезда к офису компании. Главная страница приложения изображена на рисунке 1.4.

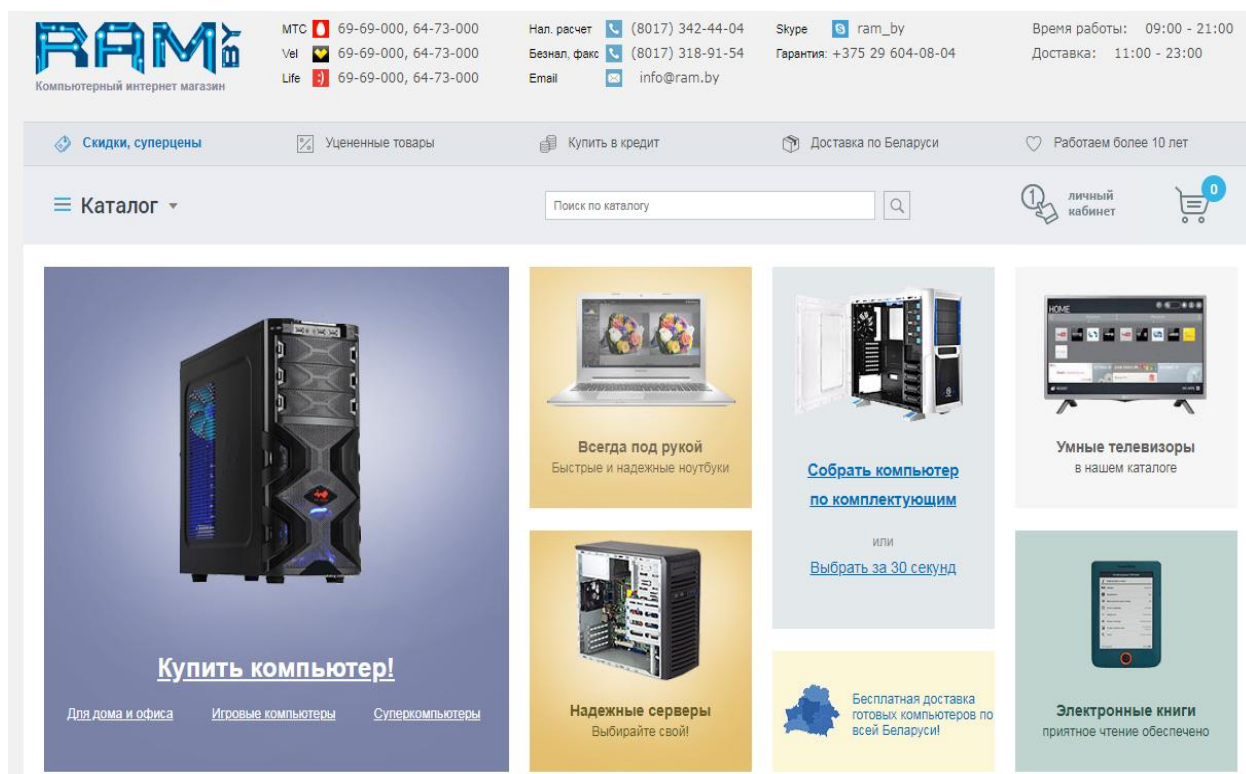


Рисунок 1.4 – Веб-приложение[4]

Используемый язык программирования PHP, JavaScript. Приложение использует веб-сервис для получения данных.

Системные требования к программному обеспечению пользователей: браузер Microsoft Internet Explorer 9 и выше, Mozilla Firefox 16 и выше, Opera 12.0 и выше, Google Chrome, Safari.

Недостатки данного приложения:

- отсутствует форма для регистрации в системе;
- отсутствует возможность фильтра товара по параметрам производительности.

5. *cooler.by* [5] – Интернет-магазин компьютерной техники от лидирующих мировых производителей. Компьютеры поделены на разделы: Офисный, Школьный, Домашний, Игровой и Мультимедийный компьютер. В каждом разделе компьютерная техника поделена на готовые решения в

комплекте с монитором, клавиатурой, мышью и колонками. На страницах интернет-магазина также представлены таблицы вариантов сборки системных блоков. Выбранный персональный компьютер можно укомплектовать периферийной компьютерной техникой представленной в разделе "Периферия". Если вы не нашли нужную конфигурацию - можно отправить сборку онлайн через конфигуратор. Так же на сайте есть форма обратной связи, где можно задать вопрос специалистам, а так же заказать звонок. Главная страница приложения изображена на рисунке 1.5.



Рисунок 1.5 – Веб-приложение[5]

Основные недостатки приложения:

- отсутствует строка для поиска товара;
- отсутствует возможность авторизации.

6. *cursor.by* [6] – Интернет-магазин специализируется на продаже компьютеров, ноутбуков и ЖК-телевизоров.

Компания KURSОР.by реализует только официальную сертифицированную технику. На все товары распространяется гарантия авторизованных сервис-центров, фирм-производителей и официальных партнеров. Заказывая и покупая технику в данном магазине, покупатель может быть уверен, что в случае возникновения вопросов или каких-либо неполадок с оборудованием, он быстро и своевременно получит квалифицированную помощь. Главная страница приложения изображена на рисунке 1.6.

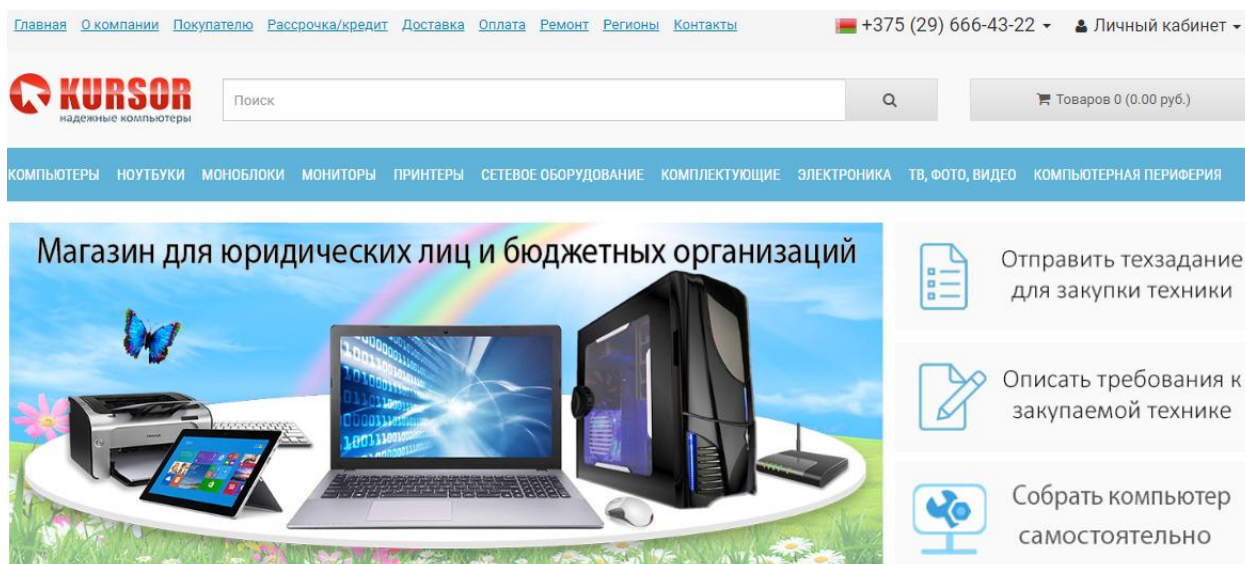


Рисунок 1.6 – Веб-приложение[6]

Используемый язык программирования PHP, JavaScript. Приложение использует веб-сервис для получения данных.

Системные требования к программному обеспечению пользователей: браузер Microsoft Internet Explorer 9 и выше, Mozilla Firefox 16 и выше, Opera 12.0 и выше, Google Chrome, Safari.

К основным недостаткам приложения относятся:

- отсутствие формы обратной связи.

1.2 Spring Framework

Spring Framework [6] обеспечивает комплексную модель разработки и конфигурации для современных бизнес-приложений на Java - на любых платформах. Ключевой элемент Spring - поддержка инфраструктуры на уровне приложения: основное внимание уделяется "водопроводу" бизнес-приложений, поэтому разработчики могут сосредоточиться на бизнес-логике без лишних настроек в зависимости от среды исполнения.

Несмотря на то, что Spring не обеспечивал какую-либо конкретную модель программирования, он стал широко распространённым в Java-сообществе главным образом как замена модели Enterprise JavaBeans. Spring предоставляет большую свободу Java-разработчикам в проектировании. Кроме того, он предоставляет хорошо документированные и лёгкие в использовании средства решения проблем, возникающих при создании приложений больших масштабов.

Особенности ядра Spring применимы в любом Java-приложении, и существует множество расширений и усовершенствований для построения веб-приложений на Java Enterprise платформе. По этим причинам Spring приобрёл большую популярность и признаётся разработчиками как стратегически важный фреймворк.

Spring имеет собственную MVC-платформу веб-приложений, которая не была первоначально запланирована. Разработчики Spring решили написать её как реакцию на то, что они восприняли как неудачность конструкции (тогда) популярного Apache Struts, а также других доступных веб-фреймворков. В частности, по их мнению, было недостаточным разделение между слоями представления и обработки запросов, а также между слоем обработки запросов и моделью.

1.3 Язык программирования Java

Java [7] – это технология, которую используют для разработки онлайн-приложений, то есть программ, запускающихся и работающих прямо в вашем браузере.

Подобные приложения существенно расширяют возможности интернета и с помощью технологии Java мы можем играть в онлайн-игры, общаться в различных онлайн-чатах, загружать фото и видео на различные сайты и многое-многое другое.

Для того, чтобы воспользоваться всеми этими преимуществами необходимо, чтобы на компьютере была установлена среда Java. То есть это специальная программа, позволяющая запускать Java-контент прямо в вашем браузере. Если среда не установлена на вашем компьютере, то многие онлайн-приложения и веб-сайты просто не будут работать. Обычно в этом случае когда вы заходите на веб-страницу, содержащую Java-контент, появляется сообщение о необходимости скачать и установить на ваш компьютер Java.

Программы на Java транслируются в байт-код Java, выполняемый виртуальной Java машиной – программой, обрабатывающей байтовый код и передающей инструкции оборудованию как интерпретатор.

Достоинством подобного способа выполнения программ является полная независимость байт-кода от операционной системы и оборудования, что позволяет выполнять Java-приложения на любом устройстве, для которого существует соответствующая виртуальная машина. Другой важной особенностью технологии Java является гибкая система безопасности, в рамках которой исполнение программы полностью контролируется виртуальной машиной.

Любые операции, которые превышают установленные полномочия программы (например, попытка несанкционированного доступа к данным или соединения с другим компьютером), вызывают немедленное прерывание.

1.4 Архитектура веб-приложений

Веб-приложение - это прикладная программа, которая хранится на удаленном сервере и взаимодействует с клиентом через интерфейс браузера. Веб-приложение освобождает разработчика от ответственности за создание

клиентской программы для определенной модели компьютера или конкретной операционной системы, поскольку веб-приложение может использовать любой клиент с наличием доступа к сети Интернет. Веб-приложения обычно используют комбинацию серверных (Java, ASP, PHP, Ruby и т.д.) и клиентских сценариев (HTML, Javascript и т.д.) при разработке приложения. Клиентская часть отвечает за представление обработанной информации в окне браузера, в то время как сервер отвечает за хранение, получение и изменение информации в системе хранения данных.

К функции, которые выполняет веб-приложение относятся следующие:

- получение данных от клиента;
- передача данных;
- авторизация пользователей;
- отображение быстро изменяющейся информации;
- обеспечение одновременного доступа нескольким клиентам;
- обеспечение высокой мобильности.

Создание веб-приложений позволяет решать самые разные задачи коммерческих компаний. Веб-приложение схоже с веб-сайтом, например, доступ к веб-приложению и веб-сайту можно получить по ссылке. Отличие заключается в том, что веб-сайт представляет собой набор связанных HTML-страниц, к которым пользователь имеет доступ, а веб-приложение формирует ответ динамически, в зависимости от запроса пользователя. Это значит, что веб-приложение схоже с настольным приложением, только код выполняется на сервере и результат передается клиенту по сети, в качестве интерфейса используется браузер.

У веб-приложений нет проблем с поддержкой старых версий программ и обратной совместимостью. Существует только одна версия, в которой работают все пользователи, и в случае выхода новой все, без исключения, автоматически переходят на нее, порой даже не замечая этого.

Архитектуру современных веб-приложения можно разделить на монолитную и микросервисную.

Монолитная архитектура является классической архитектурой веб-приложений. Чаще всего такое приложение состоит из нескольких слоев: слой представления, слой бизнес-логики, слой данных. Поведение таких приложений полностью реализуется в собственном процессе, а все приложение обычно развертывается как один элемент. При горизонтальном масштабировании таких приложений приходится дублировать его на нескольких серверах или виртуальных машинах.

По мере роста и развития приложений монолитная архитектура столкнулась со множеством проблем:

- постоянный рост и увеличение сложности системы;
- сложность добавления нового функционала;
- увеличение количества ошибок;
- рост количества времени на тестирование и отладку;
- устаревание используемых технологий;

- увеличение стоимости поддержки;
- сложность передачи знаний при смене участников проекта.

Микросервисы – это разновидность сервис-ориентированной архитектуры, которая применяется при разработке распределенных программных систем. Модули в микросервисной архитектуре взаимодействуют через сеть, при этом выполняя единую задачу. Из-за перечисленных выше минусов монолитной архитектуры в настоящее время микросервисные приложения постепенно вытесняют монолитные приложения и превращаются в стандарт развития программных систем.

Основное отличие микросервисной архитектуры от монолитной – использование более простых и маленьких программ, называемых модулями, в то время как в монолитной архитектуре используются взаимодействия внутри одного процесса. Микросервисные приложения легко поддаются горизонтальному масштабированию, так как модули могут находиться на разных серверах и взаимодействовать через сеть.

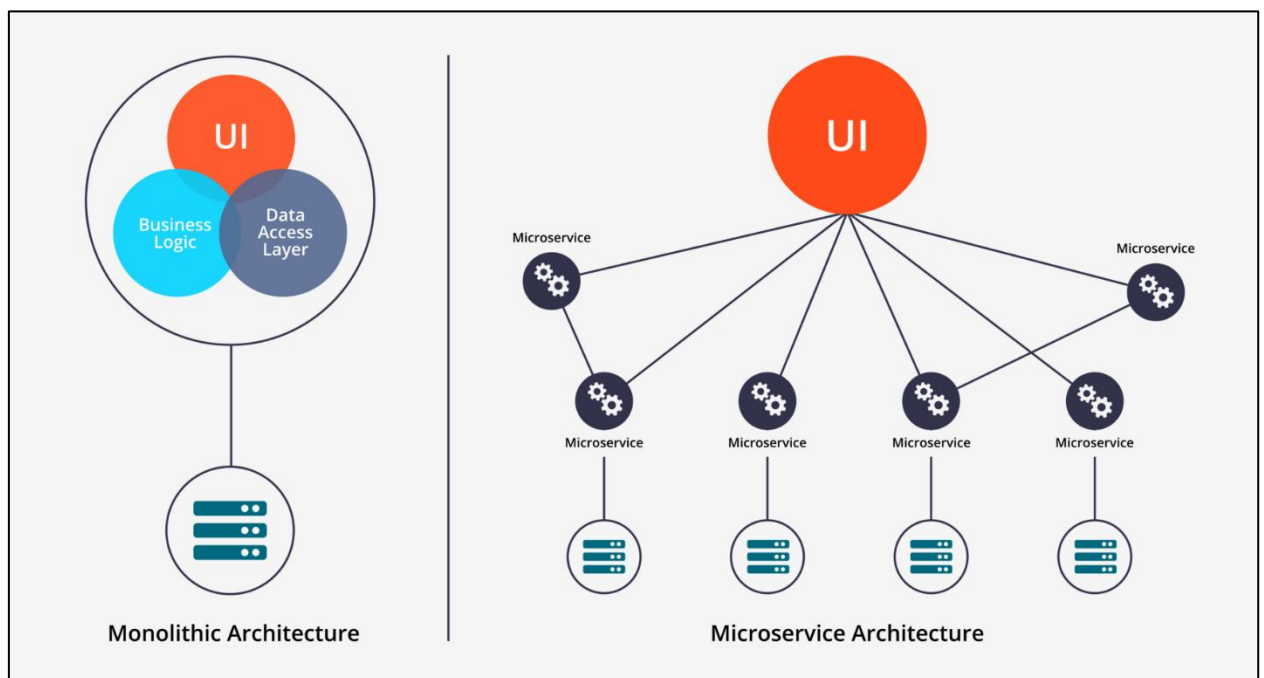


Рисунок 1.7 – Сравнение монолитной и микросервисной архитектур [7]

Таким образом, монолитная архитектура имеет несколько преимуществ перед микросервисной:

- симметричная архитектура;
- независимость микросервисов;
- взаимозаменяемость микросервисов;
- разделение логики приложения на более простые программы;
- написание микросервисов с помощью любых программных средств и на любых языках программирования.

Каждый модуль в микросервисной архитектуре – это программа, которая выполняет определенную часть бизнес-логики. В продукт с использованием такого подхода к проектированию лего добавляются новые функции путем добавления новых модулей без крупного изменения основного приложения, как в случае монолитной архитектуры.

1.5 Язык разметки HTML

HTML [8] – стандартный язык разметки в веб-приложениях. Язык HTML интерпретируется браузерами, полученный в результате интерпретации форматированный текст отображается на экране монитора компьютера или мобильного устройства.

Во всемирной паутине HTML-страницы в веб-приложениях, как правило, передаются браузерам от сервера по протоколам HTTP или HTTPS, в виде простого текста или с использованием шифрования.

HTML создавался как язык для обмена научной и технической документацией, пригодный для использования людьми, не являющимися специалистами в области вёрстки. HTML успешно справлялся с проблемой сложности SGML путём определения небольшого набора структурных и семантических элементов – дескрипторов. Дескрипторы также часто называют «тегами». С помощью HTML можно легко создать относительно простое, но красиво оформленное веб-приложение. Помимо упрощения структуры документа, в HTML внесена поддержка гипертекста. Мультимедийные возможности были добавлены позже.

1.6 База данных MySQL

MySQL [9] – это система управления реляционными базами данных с открытым исходным кодом, используемая в разных приложениях. Сегодня MySQL является СУБД, стоящей за многими ведущими веб-сайтами в мире и бесчисленным количеством корпоративных и ориентированных на потребителя веб-приложений.

SQL означает язык структурированных запросов, который используется для коммуникации с другими программами. Сверх того, MySQL имеет свои собственные расширенные функции SQL для того чтобы обеспечить пользователям дополнительный функционал. Большим достоинством MySQL является возможность работы с интерфейсом программного приложения, который может обеспечить простой доступ из программы пользователя к СУБД.

MySQL также позволяет пользователям выбирать наиболее эффективный механизм хранения для любого вида таблицы, поскольку программа может использовать несколько механизмов хранения для отдельных таблиц.

2 СИСТЕМНОЕ ПРОЕКТИРОВАНИЕ

Для обеспечения гибкой архитектуры приложения было принято решение разбить его на функциональные блоки (модули). Благодаря данному подходу появляется возможность изменять модули по отдельности, не затрагивая систему целиком.

В разрабатываемом веб-приложении выделены следующие блоки:

- Блок новостей
- Блок отзывов
- Блок замечаний и предложений
- Блок контактов
- Блок каталога товаров
- Блок вакансий
- Блок администрирования
- Блок работы с базой данных
- Блок базы данных

Перечисленные блоки и связи между ними показаны на чертеже ГУИР.400201.019 С1.

Для каждого модуля определен свой список выполняемых задач. Работа системы обеспечивается взаимодействием модулей между собой. Взаимодействие модулей сводится к обмену данными при помощи различных форматов и протоколов.

2.1 Блок управления приложением

Модуль управления приложением является центральным модулем приложения. Он отвечает за все функции приложения, пересылку данных, корректное взаимодействие между всеми остальными компонентами системы.

Основной частью модуля управления являются сервисы и ресурсы программной платформы Java Spring.

Сервисы реализуют бизнес-логику приложения, отвечают за взаимодействие с базой данных.

Ресурсы отвечают за предоставление функционала, реализованного в сервисах, клиентской стороне. Ресурсы могут ограничивать права доступа к тем или иным функциям на серверной стороне в зависимости от роли пользователя, организовать различные пути доступа к необходимым функциям. По умолчанию ресурсы имеют методы для создания, обновления, удаления и отображения данных.

Чаще всего ресурс принимает некоторые параметры, передает их в сервис, получает от него данные и возвращает их клиентской стороне. В случае возникновения ошибок ресурс может отправить клиенту информацию об ошибке сервера.

2.2 Блок отзывов

Блок отзывов представляет собой клиентскую часть веб-приложения. Этот блок включает в себя инструменты, предоставляющие пользователю возможность взаимодействовать с приложением через браузер.

Пользователь отправляет отзыв с главной страницы. Для отправки отзыва нужно ввести отзыв и имя клиента. После ввода данные отправляются на сервер. Передача на сервер происходит с помощью метода GET, для задания метода в теге <form> используется атрибут method, а его значениями выступают ключевые слова post.

Работа с формой по умолчанию происходит в текущей вкладке браузера, при этом допустимо при отправке формы изменить этот параметр и открывать обработчик формы в новой вкладке или во фрейме.

Связь между формой и ее элементами происходит через идентификатор формы, а к элементам следует добавить атрибут form со значением, равным этому идентификатору.

Для того, чтобы вывести отзывы на страницу, клиентская часть веб-приложения отправляет запрос на сервер, откуда происходит чтение данных на веб-страницу.

2.3 Блок базы данных

Для реализации хранилища данных в системе используется реляционная база данных, представленная продуктом MySQL.

MySQL - свободная реляционная система управления базами данных. Она обеспечивает достаточный уровень производительности для маленьких и средних приложений, имеет широкий набор инструментов для разработки на различных языках программирования, управления правами доступа, мониторинга ресурсов и состояния базы данных. Существует как консольный, так и графический интерфейс для взаимодействия с базами данных. Продукт достаточно прост в установке и использовании, не требует большого количества вычислительных ресурсов.

Реляционная база данных обеспечивает простоту разработки новых объектов и сущностей приложения. Каждая таблица в базе данных соответствует определенной сущности на стороне приложения. Реляционные базы данных обеспечивают простоту понимания структуры для конечных пользователей, при их проектировании применяются строгие правила, которые позволяют избежать денормализации данных.

К недостаткам реляционных моделей можно отнести сравнительно низкую скорость доступа к данным, быстрое увеличение количества таблиц при ошибках в изначальной архитектуре.

Тем не менее, предметная область данного приложения успешно проецируется на реляционную базу данных, ее можно легко представить в виде совокупности таблиц.

2.4 Блок администрирования

Блок администрирования представлен админ-панелью, которая предоставляет доступ ко всем возможностям системы управления контентом. Административный блок предполагает адаптивный дизайн и наличие большого количества готовых элементов.

С помощью административного блока можно добавлять, изменять, редактировать всю информацию, которая находится на сайте.

В интерфейсе блока администрирования используются основные инструменты:

- сетки;
- шаблоны;
- типографика;
- медиа;
- таблицы;
- формы;
- алерты;
- навигация.

Для отображения списков элементов используются Ajax-таблицы с данными. Таблицы реализованы при помощи библиотеки `ajax`, позволяющей один раз описать внешний вид и поведение всех таблиц и переиспользовать данный шаблон для всех типов сущностей.

Благодаря использованию Ajax, навигация по страницам таблицы осуществляется без перезагрузки страницы. При этом информация загружается из базы ограниченными текущей страницей участками, что снижает нагрузку на хранилище данных и обеспечивает высокую производительность.

Действия с интерфейсом преобразуются в операции с элементами DOM с помощью которых обрабатываются данные, доступные пользователю, в результате чего изменяется их представление. Здесь же производится обработка щелчков мышью и перемещений, а также нажатий клавиш.

Каскадные таблицы стилей, или CSS обеспечивают согласованный внешний вид элементов приложения и упрощают обращение к DOM-объектам.

Объект `XMLHttpRequest` используется для асинхронного взаимодействия с сервером, обработки запросов пользователя и загрузки в процессе работы необходимых данных.

2.5 Блок контактов

Блок контактов представлен отдельной веб-страницей, на которой расположена карта проезда к офису компании.

Данный блок имеет описание всех офисов владельцев интернет-магазина. Здесь есть возможность просмотреть и перейти на социальные сети компании.

2.6 Блок обратной связи

Блок обратной связи приложения представлен системой отправки писем на указанные в конфигурации приложения почтовые ящики. На стороне клиента обратная связь доступна через специальные формы. После заполнения формы и запроса на отправку данных, информация, представленная пользователем, отражается в базе данных и отправляется на почтовый ящик получателя.

Для того, чтобы отправка формы осуществлялась без перезагрузки страницы, нужно воспользоваться технологией ајах. Предусмотрена возможность просмотра отправленных писем из панели администрирования.

2.7 Блок каталога товаров

Блок каталога товара имеет отдельную веб-страницу, состоящую из отдельных категорий товаров. Каждая категория состоит из четырех главных составляющих: наименование продукции, описание, фотография и цена. Каталог имеет возможность дополнительных сервисов, например, таких как сравнение товара или выбор продукции по определенным характеристикам.

Каталог товаров имеет навигацию с полным отображением пройденного пути. Это необходимо для быстрого ориентирования. Каталог товаров имеет функцию сортировки продукции, которая проводится по типу товара. Также присутствует фильтр продукции по определенным параметрам.

В каталоге предусмотрена возможность поиска продукции по определенным характеристикам. Данный процесс поиска не должен быть слишком долгим, иначе посетитель просто покинет сайт, так и не дождавшись результатов работы сервиса.

Технология JavaScript используется для отображения более подробного описания продукции в новом окне.

2.8 Блок работы с базой данных

Блок работы с базой данных реализует всю работу с базой данных в приложении. Каждой таблице в базе данных будет соответствовать определенная сущность на языке Java.

При взаимодействии с базой данных используется библиотека Hibernate, позволяющая освободить разработчика от значительного объема сравнительно низкоуровневого программирования при работе в объектно-ориентированных средствах в реляционной базе данных.

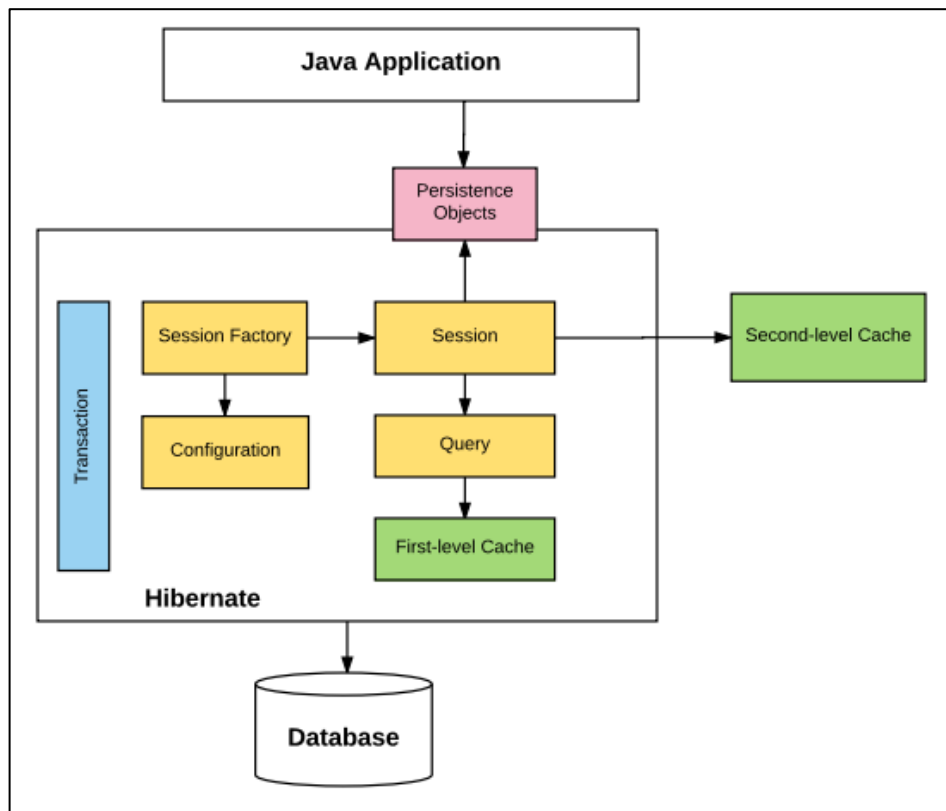


Рисунок 2.2 – Архитектура Hibernate [10]

Библиотека Hibernate не только решает задачу связи классов Java с таблицами базы данных (и типов данных Java с типами данных SQL), но также предоставляет средства для автоматической генерации и обновления набора таблиц, построения запросов и обработки полученных данных и может значительно уменьшить время разработки, которое обычно тратится на ручное написание SQL- и JDBC-кода.

Hibernate автоматизирует генерацию SQL-запросов и освобождает разработчика от ручной обработки результирующего набора данных и преобразования объектов, максимально облегчая перенос (портирование) приложения на любые базы данных SQL.

3 ФУНКЦИОНАЛЬНОЕ ПРОЕКТИРОВАНИЕ

В данном разделе описывается функционирование и структура разрабатываемого веб-приложения.

3.1 Описание структуры приложения

Программный продукт строится на следующих технологиях:

- Spring Boot – для упрощения управления приложением, дает возможность быстрого конфигурирования параметров приложения;
- Maven – для конфигурации сборки, тестирования и запуска приложения, обеспечивает просто управление зависимостями;
- Spring Data JPA – это набор более широкого семейства Spring Data, упрощающий взаимодействие репозитория на основе JPA. Данный модуль содействует с расширенной поддержкой слоев доступа к данным на основе JPA. Это упрощает создание Spring приложений, использующих технологии доступа к данным;
- MySQL - это свободно доступная система управления реляционными базами данных с открытым исходным кодом, использующая язык структурированных запросов (SQL).

Так как использование таких технологий само по себе предполагает некоторые архитектурные шаблоны, приложение спроектировано с использованием предметно-ориентированного подхода, в котором существуют понятия предметной области и модели. Каждый объект предметной области представлен таблицей в базе данных. Вокруг каждого из этих объектов строится архитектура сервисов и служб, которые используются для управления предметами области, включая их создание, изменение и удаление.

Приложение разделено на следующие пакеты, которые сгруппированы по функционалу:

1. Пакет *config* – включает в себя классы конфигурации приложения такие как настройки почтового отправителя, глобальные настройки пользователей, конфигурации веб-безопасности.
2. Пакет *domain* – включает в себя основные сущности предметной области приложения, такие как подсистема, параметры подсистемы, объект подсистемы, пользователь и другие.
3. Пакет *service* – включает в себя все интерфейсы и сервисы, которые управляют предметной областью. Пакет включает в себя пакеты *dto*, *impl*, *util*, которые содержат модели объектов для передачи между компонентами приложения, реализации интерфейсов сервисов, классы взаимосвязи между сущностями и моделями и утилиты сервисов соответственно.
4. Пакет *repository* содержит репозитории реализованные в виде интерфейсов для управления базой данных.

5. Пакет *Controller* содержит контроллеры, которые предназначены для того, что бы обрабатывать приходящие HTTP запросы. Контроллеры являются конечными точками приложения, к которым обращается пользователь.

3.2 Анализ диаграммы классов

3.2.1 Класс *MainController*

Класс *MainController* предназначен для обработки запросов, поступающих с главной страницы сайта.

Класс реализует следующие методы:

- Метод *showProductsBySubcatalog* является обработчиком *get* запроса, который отправляется на сервер, когда пользователь выбирает подкаталог из списка подкаталогов внутри каталога. Метод получает имя каталога и имя подкаталога с помощью которых в базе данных происходит поиск необходимых данных, создаются соответствующие объекты и посылаются на страницу в виде модели. Результатом отработки данного метода на главной странице появляется список продуктов соответствующий выбору пользователя;

- *showProductsByCatalog* – обработчик *get* запроса, который принимает имя выбранного пользователем каталога и возвращает объекты в виде модели, представляющие из себя список всех продуктов данного каталога;

- *mainPage* – обработчик *get* запроса, который обрабатывает по переходу пользователя на главную страницу сайта. Возвращает на главную страницу объекты каталога всех каталогов из базы данных;

- *getCatalogs* – метод, который возвращает контейнер вида ключ-значение, где ключом является элемент перечисления, а значение русифицированная версия этого элемента;

- *getSubcatalogs* – метод, возвращающий контейнер вида ключ-значение, где ключом является название каталога, а контейнером итерация названий подкаталогов, соответствующая данному каталогу, взятая из базы данных .

3.2.2 Класс *CatalogController*

Класс *CatalogController* является административным классом имеющим аннотацию *Preauthorize*, которая ограничивает метод пользователям не имеющим статус администратора и предназначен для добавления подкаталогов.

Класс реализует следующие методы:

- *showCatalogs* – обработчик *get* запроса, возвращающий список каталогов;

- *editCatalog* – обработчик *get* запроса, возвращающий список подкаталогов, соответствующий данному каталогу;

– *addSubcatalogToCatalog* – обработчик *post* запроса отработывающий при добавлении администратором нового каталога. Данный метод принимает имя каталога, введенного администратором в текстовое поле, создает объект каталога и записывает его в базу данных.

3.2.3 Класс *ContactsController*.

Класс *ContactsController* предназначен для редактирования контактов владельцев сайта.

Класс реализует следующие методы:

– *showContactsPage* – обработчик *get* запроса получает из базы данных список контактов, таких как номера телефонов, график работы, адреса, ссылки и посылает их в виде модели на страницу контактов;

– *saveContacts* – обработчик *post* запроса, является административным методом, принимающим заполненную администратором форму контактов. Создает в базе данных таблицу, если она еще не создана и отправляет в нее контакты владельца;

– *getIterable* – данный метод преобразует строку в массив строк по заданному разделителю. Он необходим так как администратор вводит список контактов в одну строку, что бы хранить их в одной таблице.

3.2.4 Класс *EmailController*

Класс *EmailController* предназначен для отображения формы обратной связи и отправки сообщений на почту владельца сайта.

Класс реализует следующие методы:

– *showContactUsPage* – обработчик *get* запроса, возвращающий страницу обратной связи с формами для заполнения;

– *sendMessageToEmail* – обработчик *post* запроса, отправляющий заполненную пользователем форму на почту владельца сайта.

3.2.5 Класс *RegistrationController*

Класс предназначен для регистрации пользователей в системе, методы данного класса работают при манипуляции пользователя со страницей регистрации.

Класс реализует следующие методы:

– *showRegistrationPage* – обработчик *get* запроса, возвращающий страницу регистрации;

– *addUser* – обработчик *post* запроса позволяющий зарегистрировать нового пользователя в системе. Метод принимает объект пользователя, сначала выполняется поиск в базе данных по имени пользователя, если пользователь с данным именем уже существует в системе, то на страницу регистрации возвращается сообщение, указывающее на то, что пользователь уже существует. Иначе пользователю присваивается роль *USER* и его объект

сохраняется в базе данных, после чего происходит перенаправление на главную страницу сайта.

3.2.6 Класс *ReviewsController*

Класс *ReviewsController* отвечает на запросы пользователя, которые поступают от него на странице отзывов.

Класс реализует следующие методы:

- *showReviews* метод *get* запроса, сначала производит поиск всех сообщений в базе данных, отправляет их в виде модели на страницу отзывов, после чего возвращает саму страницу;

- *deleteMessage* – административный метод *post* запроса обрабатывающий при нажатии администратором на кнопку удаления сообщения, находящейся напротив сообщения. Метод принимает объект сообщения по его идентификатору, производит его удаление из базы данных, возвращает оставшиеся сообщения на страницу отзывов в виде модели, после чего возвращает саму страницу;

- *addMessage* – метод *post* запроса принимает объект авторизованного в системе пользователя и строку с отзывом, после чего создает объект сообщения и связывает его с пользователем, если он авторизован. Далее метод сохраняет сообщение в базе данных, обновляет список сообщений и отправляет их в виде модели на страницу отзывов.

3.2.7 Класс *UserController*

Класс *UserController* является административным и отвечает на запросы администратора, поступающие от него на странице списка пользователей.

Класс реализует следующие методы:

- *userList* – метод *get* запроса обрабатывающий при входе администратора на страницу возвращает список всех пользователей из базы данных в виде модели;

- *userEditForm* – метод *get* запроса обрабатывающий при попытке администратора изменить параметры пользователя, возвращает на страницу объект выбранного пользователя и список ролей;

- *userSave* – метод *post* запроса обрабатывающий в случае сохранения изменений параметров пользователя, принимает отправленную со страницы форму в виде ключ-значение и присваивает пользователю роли содержащиеся в переданной форме, напротив которых администратор выставил флажки.

3.2.8 класс *ProductController*

Класс *ProductController* является административным классом и содержит в себе методы добавления, изменения, удаления всех возможных типов продуктов, таких как компьютер, телефон, монитор, моноблок, ноутбук, принтер, планшет. Методы для каждого типа продукта аналогичны

и отличаются только параметрами самих устройств, поэтому имеет смысл рассмотреть работу класса на примере одного продукта. Для примера рассмотрим манипуляции над продуктом типа компьютер. Для данного продукта реализованы следующие методы:

- *addComputer* – метод *post* запроса который начинает свою работу после отправки на сервер администратором заполненной формы товара. Создается объект компьютер, после чего из формы извлекаются параметры для заполнения полей данного объекта, такие как количество процессоров ПК, название процессора, количество ядер, тактовая частота и т.д., после чего данному объекту присваивается каталог, в котором администратор добавлял данный товар и происходит сохранение продукта в базе данных. Возвращает метод страницу подкаталога, в который был добавлен данный товар;

- *editComputer* – метод *get* запроса отработывающий в случае попытки администратора изменить характеристики продукта, принимает *id* продукта для которого производятся изменения в качестве параметра адресной строки и с помощью аннотации *PathVariable* по полученному идентификатору из базы данных извлекается объект продукта который отправляется на страницу редактирования после чего администратора перенаправляет на страницу редактирования продукта с заранее заполненными полями конфигурации данного продукта;

- *saveComputer* – метод *post* запроса отработывающий при попытке администратора применить изменения редактируемого продукта. С помощью аннотации *RequestParam* из базы данных извлекается продукт по идентификатору переданному на сервер через форму. Поля данного продукта заполняются параметрами формы после чего происходит сохранение в базу данных;

- *saveFile* – метод принимает файл загруженный на сервер через форму, происходит проверка имени файла, если оно существует, значит файл был загружен и его нужно сохранить. Для этого создается уникальный идентификатор *UUID* для файла, т.к. имена файлов могут совпадать, далее данный файл сохраняется на сервере, а так же его имя сохраняется в базе данных продукта, для которого этот метод был вызван;

- *setComp* – метод установки параметров компьютера реализован потому, что его необходимо вызывать в случае добавления нового продукта в каталог и сохранения измененного продукта в каталоге. Метод принимает форму в виде ключ-значение и объект, для которого необходимо установить параметры из формы. Возвращает объект, поля которого в дальнейшем будут сохранены в базе данных;

- *showSubcatalogProducts* – метод *get* запроса, принимающий имя каталога и объект подкаталога, полученный по идентификатору в качестве которого выступает путь в адресной строке. Происходит поиск всех продуктов подкаталога внутри данного каталога по идентификатору

подкаталога, после чего на страницу возвращается список продуктов, соответствующий данному подкаталогу.

3.2.9 Класс *Contacts*

Класс *Contacts* является сущностью связанной с таблицей *contacts* в базе данных.

Данный класс является моделью сущности с которой работают пользователи и администратор обращаясь с запросами к *ContactsController*.

ContactsController обращается к репозиторию *ContactsRepo* с помощью которого он может получить список контактов компании, который пользователь может увидеть в виде модели *contacts* или создать новый экземпляр данной модели.

Сущность *Contacts* является списком контактов компании или организации владеющей интернет-магазином. После заполнения формы контактов администратором данная сущность сохраняется в базе данных. На странице контактов перед пользователем видны контакты компании, администратор же видит только формы для заполнения. Если список контактов уже был заполнен ранее, он видит эти данные и может их изменять.

Класс связывается с таблицей *contacts* базы данных, название которой указывается в аннотации *Table* перед классом.

Класс *Contacts* обладает следующими полями:

- *id*;
- *timeWorking*;
- *phoneNumbers*;
- *address*.

Поле *id* является идентификатором сущности, который генерируется автоматически с помощью аннотации *GeneratedValue*.

Поле *timeWorking* является строкой в базе данных, в которой указывается время работы компании через.

Поле *phoneNumbers* является строкой в базе данных, в которой указываются номера телефонов компании.

Поле *links* является строкой в базе данных, в которой указываются ссылки на социальные сети компании.

Поле *address* является строкой в базе данных, в которой указываются адреса офисов компании владельца.

Методы класса *Contacts*:

– Метод *getId* является методом, который возвращает уникальный идентификатор объекта с которым работает администратор. Данный метод позволяет при переходе на определенную страницу достать объект по идентификатору;

– Метод *setId* позволяет устанавливать уникальный идентификатор объекта;

- Метод *getTimeWorking* является методом, который позволяет извлекать из базы данных строку с графиком работы офисов. Данная строка представляет из себя множество строк, разделенных специальным символом разделения. Это позволяет сократить количество записей в базе данных
- Метод *setTimeWorking* позволяет установить в объект строку содержащую информацию о графике работы офисов для последующего его сохранения в базе данных;
- Метод *getPhoneNumbers* позволяет извлечь из объекта контакты офисов компании;
- Метод *setPhoneNumbers* позволяет установить номера офисов компании;
- Метод *getAddress* извлекает из объекта строку включающую в себя адреса офисов владельца;
- Метод *setAddress* присваивает объекту строку включающую в себя адреса офисов владельца.

3.2.10 Класс *Message*

Класс *Message* является сущностью в базе данных связанной с таблицей *message*.

Данный класс является моделью сущности с которой работают пользователи и администратор обращаясь с запросами к *ReviewsController*.

ReviewsController обращается к репозиторию *MessageRepo* с помощью которого он может получать сообщения пользователей, как зарегистрированных так и нет, оставленных на странице отзывов.

Сущность *Message* является списком отзывов о сайте в базе данных. Данная сущность может быть изменена любым посетителем интернет-магазина. Администратор магазина, в отличии от обычного пользователя, зарегистрированного или нет, напротив каждого сообщения видит кнопку удаления, нажав на которую вызывается соответствующий метод контроллера, в котором по уникальному идентификатору производится удаление сообщения.

Класс связывается с таблицей *messages* базы данных, название которой указывается в аннотации *Table* перед классом.

Класс *Message* обладает следующими полями:

- *Id*;
- *Msg*;
- *Author*.

Поле *id* является идентификатором сущности, который генерируется автоматически с помощью аннотации *GeneratedValue*.

Поле *msg* является строкой в базе данных, которая содержит в себе отзыв о работе сайта.

Поле *author* является объектом типа *User* и представляет собой связь *ManyToOne* в базе данных, что говорит о том, что один пользователь может быть автором нескольких сообщений. Это поле представлено в виде колонки

с названием *user_id* в таблице *message* и содержит в себе идентификатор автора сообщения.

Методы класса *Message*:

- Метод *getId* является методом, который возвращает уникальный идентификатор объекта с которым работают пользователи. Данный метод позволяет при переходе на определенную страницу достать объект по идентификатору;
- Метод *setId* позволяет устанавливать уникальный идентификатор объекта;
- Метод *getAuthorName* позволяет получить имя пользователя оставившего отзыв, если пользователь был зарегистрирован в системе, в противном случае возвращает строку *UNKNOWNUSER*, которая указывает на то, что пользователь оставивший отзыв не был зарегистрирован;
- Метод *getMsg* позволяет получить из объекта строку включающую в себя текст с отзывом;
- Метод *setMsg* позволяет установить в объект строку включающую в себя текст с отзывом.

3.2.11 Класс *Subcatalog*

Класс *Subcatalog* является сущностью в базе данных связанной с таблицей *subcatalogs*.

Данный класс является моделью сущности с которой работают пользователи и администратор обращаясь с запросами к *CatalogController*.

CatalogController обращается к репозиторию *SubcatalogRepo* с помощью которого он может получать названия подкаталогов из базы данных.

Сущность *Subcatalog* является списком подкаталогов в базе данных. Данная сущность может быть изменена только администратором веб-приложения. Администратор может добавлять товар в подкаталоги, изменять его, удалять. Так же он может удалять подкаталог целиком, тогда все продукты, соответствующий данному подкаталогу будут найдены в базе данных с помощью метода *findBySubcatalogId* репозитория *SubcatalogRepo* и удалены соответственно.

Класс связывается с таблицей *subcatalogs* базы данных, название которой указывается в аннотации *Table* перед классом.

Класс *Subcatalog* обладает следующими полями:

- *id*;
- *name*;
- *catalogs*.

Поле *id* является идентификатором сущности, который генерируется автоматически с помощью аннотации *GeneratedValue*.

Поле *name* является строкой в базе данных, которая содержит в себе название подкаталога, созданного администратором.

Поле *catalogs* является объектом перечисления каталогов, которое связывается в базе данных с таблицей подкаталогов с помощью аннотации *CollectionTable*, образующей новую таблицу с именем *category_type*, в которой идентификатор каждого подкаталога соответствует имени одного из каталогов.

Методы класса *Subcatalog*:

- Метод *getId* является методом, который возвращает уникальный идентификатор объекта с которым работает администратор. Данный метод позволяет при переходе на определенную страницу достать объект по идентификатору;
- Метод *setId* позволяет устанавливать уникальный идентификатор объекта;
- Метод *getName* позволяет достать из объекта строку, содержащую в себе название каталога;
- Метод *setName* позволяет установить в объект строку с названием каталога;
- Метод *getCatalog* позволяет получить имя каталога соответствующее данному подкаталогу;
- Метод *setCatalog* присваивает имя каталога объекту подкаталога, который был создан в данном каталоге.

3.2.12 Класс User

Класс *User* является сущностью в базе данных связанной с таблицей *users*.

Данный класс является моделью сущности с которой работают пользователи и администратор обращаясь с запросами к *UserController* и *RegistrationController*.

Контроллеры обращаются к репозиторию *UserRepo* с помощью которого они могут получить информацию о пользователе в базе данных, а так же их список ролей.

Сущность *User* является списком пользователей в базе данных, которая может быть изменена незарегистрированными пользователями и администратором. Администратор может редактировать роли пользователя.

Класс связывается с таблицей *users* базы данных, название которой указывается в аннотации *Table* перед классом.

Класс *User* обладает следующими полями:

- *id*;
- *username*;
- *password*;
- *roles*.

Поле *id* является идентификатором сущности, который генерируется автоматически с помощью аннотации *GeneratedValue*.

Поле *username* является именем пользователя, которое он будет вводить при входе в систему.

Поле *password* является паролем пользователя, который он будет вводить при входе в систему.

Поле *roles* является списком ролей пользователя, вошедшего в систему. По умолчанию зарегистрированному пользователю присвоена роль *USER*, однако администратор может повысить эту роль до роли *ADMIN*. Роли пользователя отображены в таблице с названием *user_roles*, где первая колонка это идентификатор пользователя, а вторая, соответственно роль, присвоенная ему.

Методы класса *User*:

- Метод *getId* является методом, который возвращает уникальный идентификатор объекта с которым работает администратор. С помощью данного метода можно достать объект типа пользователь из базы данных при переходе на определенную страницу;

- Метод *setId* позволяет устанавливать уникальный идентификатор объекта;

- Метод *getUsername* возвращает из объекта строку с именем пользователя;

- Метод *setUsername* устанавливает в объект строку с именем пользователя;

- Метод *isAccountNonLocked* возвращает из объекта пользователя булево значение, указывающее на то, заблокирован пользователь или нет;

- Метод *getPassword* возвращает из объекта пользователя его пароль;

- Метод *setPassword* присваивает пароль объекту пользователя;

- Метод *getRoles* позволяет получить из объекта список ролей пользователя в виде перечисления;

- Метод *setRoles* позволяет установить в объект список ролей пользователя в виде перечисления;

- Метод *isEnabled* возвращает булево значение, указывающее на то, включен ли пользователь в систему.

3.2.13 Класс *MailSender*

Класс *MailSender* является классом, предназначенным для отправки письма на почту. Адрес отправителя указывается в *application.property* в виде строки и с помощью аннотации *Value* получаем поля из контекста.

Для отправки сообщения на почту используется метод *send*, который принимает в себя заголовок письма, текст письма и адрес получателя. Для отправки сообщения внутри метода создается объект типа *SimpleMailMessage*, которому присваиваются все полученные поля, после чего происходит обращение к *JavaMailSender* для отправки сообщения.

3.2.14 Класс *MailConfig*

Так как *JavaMailSender* в текущей реализации Spring без дополнительных конфигураций не работает, понадобится конфигурационный

класс *MailConfig* в котором необходимо создать метод конфигурации бина для отправки сообщения на почтовый сервер.

Данный класс имеет один метод *getMailSender*, в котором для определения локальных параметров в виде свойств бина используется *JavaMailSenderImpl*, настройки которого заполняются значениями файла конфигурации *application.property*. В данном файле указывается почта отправителя, пароль от нее, тип соединения, порт. Для того, что бы узнать какие данные туда вводить, на своем почтовом сервисе необходимо зайти в раздел настройки почтовых программ на компьютере, и в разделе настроек исходящей почты можно будет найти необходимые значения для файла конфигурации.

3.2.15 Класс Product

Класс *Product* является абстрактным классом от которого наследуются все типы продуктов интернет-магазина и содержит общие для каждого продукта поля и методы.

Данный класс не является сущностью в базе данных, однако его поля будут присутствовать в таблицах всех классов сущностей, наследующихся от него. Такую функциональность обеспечивает аннотация *MappedSuperClass*.

Класс *Product* обладает следующими полями:

- Поле *name* содержит в себе название продукта, которое будет задано администратором, при добавлении продукта в каталог;
- Поле *price* содержит в себе цену продукта;
- Поле *description* содержит в себе описание продукта;
- Поле *filename* содержит в себе имя файла сгенерированное его при сохранении, хранящее в себе адрес сохраненного на сервере файла.

Методы класса *Product*:

- Метод *getName* позволяет извлечь из объекта строку с именем продукта;
- Метод *setName* позволяет установить в объект строку с именем продукта;
- Метод *getPrice* извлекает из объекта продукта число с дробной частью, которое является его ценником;
- Метод *setPrice* позволяет установить в объект вещественное число ценник продукта;
- Метод *getDescription* позволяет извлечь из объекта строку с описанием продукта;
- Метод *setDescription* позволяет установить в объект продукта строку с его описанием;
- Метод *getFilename* позволяет извлечь из объекта строку с именем файла хранящегося на сервере;
- Метод *setFilename* позволяет установить в объект строку с именем файла, хранящегося на сервере.

Для каждого поля класса *Product* существует методы *get* и *set*, которые позволяют получить или установить содержимое полей.

3.2.16 Класс *Computer*

Класс *Computer* является сущностью в базе данных связанной с таблицей *computers*.

Данный класс является моделью сущности с которой работают все пользователи обращаясь с запросами к *ProductController*.

Класс *Computer* обладает следующими полями:

- *id* является идентификатором сущности, который генерируется автоматически с помощью аннотации *GeneratedValue*;
- Поле *numCPU* является целочисленным полем описания процессора и хранит в себе число равное количеству процессоров;
- Поле *nameCPU* является строковым значением и содержит в себе название процессора;
- Поле *numOfCores* является целочисленным полем описания процессора и содержит в себе информацию о количестве ядер центрального процессора;
- Поле *frequency* является целочисленным полем и хранит в себе информацию о тактовой частоте процессора;
- Поле *sharedCache* является целочисленным полем и хранит в себе информацию о кэш-памяти (общий, L2 или L3);
- Поле *chipset* является строковым значением и хранит в себе информацию о чипсете;
- Поле *numOfSlots* является строковым параметром и хранит в себе информацию о количестве слотов памяти;
- Поле *maxMemorySize* является целочисленным полем и хранит в себе информацию о максимальном объеме памяти;
- Поле *typeRAM* является строковым полем и хранит в себе информацию о типе оперативной памяти;
- Поле *frequencyRAM* является целочисленным полем и хранит в себе информацию о частоте оперативной памяти;
- Поле *graphicAdapter* является строковым полем и содержит информацию о графическом адаптере видеокарты;
- Поле *typeOfGraphicAdapter* является строковым параметром содержит в себе информацию о типе графического адаптера видеокарты;
- Поле *storageCapacity* является целочисленным полем и хранит в себе информацию о емкости накопителя;
- Поле *memoryCards* является строковым параметром и содержит в себе информацию о карте памяти;
- Поле *configCapacity* является строковым параметром и содержит в себе информацию о конфигурации накопителя;
- Поле *rotationalSpeed* является целочисленным полем и хранит в себе информацию о скорости вращения;

- Поле *storageType* является строковым полем и содержит в себе информацию о типе хранилища данных;

- Поле *subcatalog* это объект типа *Subcatalog* который связан с классом *Computer* отношением *ManyToOne* и выступает в роли колонки в таблице *computers* с названием *category_id*, которое говорит о том, в каком подкаталоге находится данный компьютер.

Для каждого поля класса *Computer* существует методы *get* и *set*, которые позволяют получить или установить содержимое полей.

3.2.17 Класс *Monitor*

Класс *Monitor* является сущностью в базе данных связанной с таблицей *monitors*.

Данный класс является моделью сущности с которой работают все пользователи обращаясь с запросами к *ProductController*.

Класс *Monitor* обладает следующими полями:

- Поле *id* является идентификатором сущности, который генерируется автоматически с помощью аннотации *GeneratedValue*;

- Поле *diagonal* является целочисленным полем, содержащим в себе информацию о диагонали экрана;

- Поле *matrix* является строковым полем, содержащим в себе информацию о типе матрицы;

- Поле *aspectRatio* является строковым полем, содержащим в себе информацию о соотношении сторон;

- Поле *resolution* является строковым значением, содержащим в себе информацию о разрешении экрана;

- Поле *screenRefreshRate* является целочисленным полем, хранящим в себе информацию о частоте обновления экрана;

- Поле *support3d* является строковым полем, хранящим в себе информацию о поддержке 3D;

- Поле *subcatalog* это объект типа *Subcatalog* который связан с классом *Monitor* отношением *ManyToOne* и выступает в роли колонки в таблице *monitors* с названием *category_id*, которое говорит о том, в каком подкаталоге находится данный монитор.

Для каждого поля класса *Monitor* существует методы *get* и *set*, которые позволяют получить или установить содержимое полей.

3.2.18 Класс *Monoblock*

Класс *Monoblock* является сущностью в базе данных связанной с таблицей *monoblocks*.

Данный класс является моделью сущности с которой работают все пользователи обращаясь с запросами к *ProductController*;

Класс *Monoblock* обладает следующими полями:

- Поле *id* является идентификатором сущности, который генерируется автоматически с помощью аннотации *GeneratedValue*;
- Поле *screen3d* является строковым полем и содержит информацию о поддержке 3D;
- Поле *lightSensor* является строковым полем и содержит информацию о датчике освещенности;
- Поле *diagonal* является целочисленным полем и хранит в себе информацию о диагонали экрана;
- Поле *screenSurface* является строковым полем и хранит в себе информацию о поверхности экрана;
- Поле *resolutionH* является целочисленным полем и хранит в себе информацию о разрешении экрана по высоте;
- Поле *resolutionW* является целочисленным полем и хранит в себе информацию о разрешении экрана по ширине;
- Поле *turboFrequency* является целочисленным полем и хранит в себе информацию о турбо-частоте;
- Поле *frequency* является целочисленным полем и хранит в себе информацию о тактовой частоте;
- Поле *nameCPU* является строковым полем и хранит в себе информацию о названии процессора;
- Поле *numOfCores* является целочисленным полем и хранит в себе информацию о количестве ядер;
- Поле *cache* является строковым полем и хранит в себе информацию о кэш-памяти;
- Поле *numOfSlots* является целочисленным полем и хранит в себе информацию о количестве слотов памяти;
- Поле *sizeRAM* является целочисленным полем и хранит в себе информацию о объеме оперативной памяти;
- Поле *typeRAM* является строковым полем и хранит в себе информацию о типе оперативной памяти;
- Поле *capacitySSD* является целочисленным полем и хранит в себе информацию о емкости SSD;
- Поле *capacityHDD* является целочисленным полем и хранит в себе информацию о емкости HDD;
- Поле *typeCapacity* является целочисленным полем и хранит в себе информацию о типе накопителя;
- Поле *subcatalog* это объект типа *Subcatalog* который связан с классом *Monoblock* отношением *ManyToOne* и выступает в роли колонки в таблице *monoblocks* с названием *category_id*, которое говорит о том, в каком подкаталоге находится данный моноблок.

Для каждого поля класса *Computer* существует методы *get* и *set*, которые позволяют получить или установить содержимое полей.

3.2.18 Класс Notebook

Класс *Notebook* является сущностью в базе данных связанной с таблицей *notebooks*.

Данный класс является моделью сущности с которой работают все пользователи обращаясь с запросами к *ProductController*.

Класс *Notebook* обладает следующими полями:

- Поле *id* является идентификатором сущности, который генерируется автоматически с помощью аннотации *GeneratedValue*;
- Поле *diagonal* является строковым полем и содержит в себе информацию о диагонали экрана;
- Поле *touchscreen* является строковым полем и содержит в себе информацию о сенсорном экране;
- Поле *matrixFrequency* является строковым полем и содержит в себе информацию о частоте матрицы;
- Поле *resolution* является строковым полем и содержит в себе информацию о разрешении экрана;
- Поле *turboFrequency* является строковым полем и содержит в себе информацию о турбо-частоте;
- Поле *frequency* является строковым полем и содержит в себе информацию о тактовой частоте;
- Поле *nameCPU* является строковым полем и содержит в себе информацию о модели центрального процессора;
- Поле *numOfCorest* является строковым полем и содержит в себе информацию о количестве ядер;
- Поле *subcatalog* это объект типа *Subcatalog* который связан с классом *Notebook* отношением *ManyToOne* и выступает в роли колонки в таблице *notebooks* с названием *category_id*, которое говорит о том, в каком подкаталоге находится данный ноутбук.

Для каждого поля класса *Notebook* существует методы *get* и *set*, которые позволяют получить или установить содержимое полей.

3.2.19 Класс Phone

Класс *Phone* является сущностью в базе данных связанной с таблицей *phones*.

Данный класс является моделью сущности с которой работают все пользователи обращаясь с запросами к *ProductController*.

Класс *Phone* обладает следующими полями:

- Поле *id* является идентификатором сущности, который генерируется автоматически с помощью аннотации *GeneratedValue*;
- Поле *camera* является строковым полем и включает в себя информацию о;
- Поле *numOfSim* является целочисленным полем и включает в себя информацию о количестве сим карт телефона;

- Поле *numOfDots* является целочисленным полем и хранит в себе информацию о количестве точек матрицы;
- Поле *OS* является строковым полем и включает в себя информацию о операционной системе;
- Поле *screenSize* является строковым полем и включает в себя информацию о размере экрана;
- Поле *screenResolution* является строковым полем и включает в себя информацию о разрешении экрана;
- Поле *RAM* является целочисленным полем и включает в себя информацию о объеме оперативной памяти;
- Поле *flashMemory* является целочисленным полем и включает в себя информацию о флэш-памяти;
- Поле *numOfCores* является целочисленным полем и включает в себя информацию о количестве ядер;
- Поле *platform* является строковым полем и включает в себя информацию о платформе;
- Поле *capacityCPU* является строковым полем и включает в себя информацию о разрядности процессора;
- Поле *frequencyCPU* является строковым полем и включает в себя информацию о тактовой частоте процессора;
- Поле *nameCPU* является строковым полем и включает в себя информацию о названии центрального процессора;
- Поле *subcatalog* это объект типа *Subcatalog* который связан с классом *Phone* отношением *ManyToOne* и выступает в роли колонки в таблице *phones* с названием *category_id*, которое говорит о том, в каком подкаталоге находится данный телефон.

Для каждого поля класса *Phone* существует методы *get* и *set*, которые позволяют получить или установить содержимое полей.

3.2.20 Класс *Printer*

Класс *Printer* является сущностью в базе данных связанной с таблицей *printers*.

Данный класс является моделью сущности с которой работают все пользователи обращаясь с запросами к *ProductController*.

Класс *Printer* обладает следующими полями:

- Поле *id* является идентификатором сущности, который генерируется автоматически с помощью аннотации *GeneratedValue*;
- Поле *numOfColors* является целочисленным полем и содержит в себе информацию о;
- Поле *OS* является строковым полем и содержит в себе информацию о операционной системе;
- Поле *supportedCartridges* является строковым полем и содержит в себе информацию о поддерживаемых картриджах;

- Поле *resourceCartridge* является строковым полем и содержит в себе информацию о ресурсах картриджа;
- Поле *printSpeed* является целочисленным полем и содержит в себе информацию о скорости печати;
- Поле *printTechnology* является строковым полем и содержит в себе информацию о технологии печати;
- Поле *format* является строковым полем и содержит в себе информацию о формате печати;
- Поле *timeForFirstPage* является целочисленным полем и содержит в себе информацию о времени для выхода первой страницы;
- Поле *maxMonthLoad* является строковым полем и содержит в себе информацию о максимальной месячной загрузке принтера;
- Поле *maxResDpi* является целочисленным полем и содержит в себе информацию о максимальном разрешении;
- Поле *workingNoise* является строковым полем и содержит в себе информацию о уровне шума принтера;
- Поле *bultInStapler* является строковым полем и содержит в себе информацию о наличии встроенного вшивателя;
- Поле *inputTrayCapacity* является строковым полем и содержит в себе информацию о вместимости входных лотков;
- Поле *outputTrayCapacity* является строковым полем и содержит в себе информацию о вместимости выходных лотков;
- Поле *subcatalog* это объект типа *Subcatalog* который связан с классом *Printer* отношением *ManyToOne* и выступает в роли колонки в таблице *printers* с названием *category_id*, которое говорит о том, в каком подкаталоге находится данный принтер.

Для каждого поля класса *Printer* существует методы *get* и *set*, которые позволяют получить или установить содержимое полей.

3.2.21 Класс TheTablet

Класс *TheTablet* является сущностью в базе данных связанной с таблицей *printers*.

Данный класс является моделью сущности с которой работают все пользователи обращаясь с запросами к *ProductController*.

Класс *TheTablet* обладает следующими полями:

- Поле *id* является идентификатором сущности, который генерируется автоматически с помощью аннотации *GeneratedValue*;
- Поле *voiceCalls* является строковым полем и включает в себя информацию о голосовых вызовах;
- Поле *graphicAccelerator* является строковым полем и включает в себя информацию о графическом ускорителе;
- Поле *diagonal* является строковым полем и включает в себя информацию о диагонали экрана;

- Поле *matrix* является строковым полем и включает в себя информацию о матрице экрана;
- Поле *OS* является строковым полем и включает в себя информацию о операционной системе планшета;
- Поле *nameCPU* является строковым полем и включает в себя информацию о центральном процессоре;
- Поле *screenResolution* является строковым полем и включает в себя информацию о разрешении экрана;
- Поле *RAM* является строковым полем и включает в себя информацию о оперативной памяти;
- Поле *memory* является строковым полем и включает в себя информацию о внутренней памяти;
- Поле *numOfCores* является строковым полем и включает в себя информацию о количестве ядер процессора;
- Поле *frequency* является строковым полем и включает в себя информацию о тактовой частоте процессора;
- Поле *subcatalog* это объект типа *Subcatalog* который связан с классом *TheTablet* отношением *ManyToOne* и выступает в роли колонки в таблице *thetablets* с названием *category_id*, которое говорит о том, в каком подкаталоге находится данный планшет.

Для каждого поля класса *TheTablet* существует методы *get* и *set*, которые позволяют получить или установить содержимое полей.

4 РАЗРАБОТКА ПРОГРАММНЫХ МОДУЛЕЙ

4.1 Создание подкаталога внутри каталога

Для того, чтобы создать подкаталог и присвоить ему каталог, сначала необходимо заполнить форму.

```
<form method="post" action="/catalogList/addSubcatalog">
  <input class="form-control" type="text" name="name"
    id="formpC1">
  <input type="hidden" name="catalogName" value=
    "${catalogName}">
  <input type="hidden" value="${_csrf.token}" name=
    "_csrf">
  <button type="submit" class="btn btn-success">
    Сохранить</button>
</form>
```

После нажатия на кнопку «Сохранить», форма отправляется на соответствующий контроллер, который в качестве параметра принимает название создаваемого подкаталога и каталога, в котором этот подкаталог создается, после чего происходит поиск в базе данных на наличие данного подкаталога, результатом поиска является объект подкаталога `subcatalog`;

```
subcatalog = subcatalogRepo.findByName(subcatalog.getName())
```

Если в базе данных присутствует запись с именем подкаталога совпадающим с именем, по которому происходил поиск, осуществляется сравнение с именем каталога. Если имена совпадают, это значит что в данном каталоге уже существует такой подкаталог и на страницу возвращается сообщение об ошибке.

```
if(subcatalog.getCatalogName() == catalogName){
    subcatalogs.setCatalogs(Collections.singleton
        (Catalog.valueOf(catalogName)));
    model.addAttribute("message", "subcatalog exists");
    model.addAttribute("subcatalogs",subcatalogRepo.
        findByCatalogs(Catalog.valueOf(catalogName)));
    model.addAttribute("catalogName",catalogName);
    return "catalogEdit";
}
```

Если записей с именем создаваемого подкаталога не было найдено, значит его можно сохранить, для этого в объект подкаталога `subcatalog` устанавливается имя нового подкаталога и происходит сохранение в базе данных с помощью объекта репозитория работающего с базой данных `subcatalogRepo`.

```
subcatalogs.setCatalogs(Collections.singleton(Catalog.valueOf(
    f(catalogName)));
subcatalogRepo.save(subcatalogs);
```

4.2 Создание и отправка отзыва о магазине

Для того, что бы отправить отзыв пользователь заполняет форму отзыва на обработку соответствующему контроллеру.

Для отправки сообщения на почту необходимо создать соответствующий метод в классе `MailSender` со следующими параметрами:

- `emailTo` – почта адресата;
- `subject` – тема письма;
- `message` – сообщение письма.

```
public void send(String emailTo, String subject, String
    message) {
    SimpleMailMessage mailMessage = new
        SimpleMailMessage();

    mailMessage.setFrom(username);
    mailMessage.setTo(emailTo);
    mailMessage.setSubject(subject);
    mailMessage.setText(message);

    mailSender.send(mailMessage);
}
```

`MailSender` это объект класса `JavaMailSender`, который в текущей реализации фреймворка `Spring Boot` не работает. Так как бин автоматически не генерируется, это придется сделать самостоятельно.

В конфигурации проекта создается конфигурационный класс `MailConfig` в котором и будет описана конфигурация для отправки сообщений.

Класс содержит следующие поля:

- `host` – почтовый сервер;
- `username` – почта отправителя;
- `password` – пароль отправителя;
- `port` – порт конфигураций исходящей почты;
- `protocol` – протокол защиты соединения;
- `debug` – поле лога.

Для отправки почты по протоколу `SMTP` необходимо использовать `JavaMailSenderImpl`, который реализует интерфейс `JavaMailSender` и служит для отправки почты.

```

public JavaMailSender getMailSender() {
    JavaMailSenderImpl mailSender = new
        JavaMailSenderImpl();

    mailSender.setHost(host);
    mailSender.setPort(port);
    mailSender.setUsername(username);
    mailSender.setPassword(password);

    Properties properties =
        mailSender.getJavaMailProperties();

    properties.setProperty("mail.transport.protocol",
        protocol);
    properties.setProperty("mail.debug", debug);

    return mailSender;
}

```

Так же в данном классе устанавливаются неявные properties протокола передачи почты и отладки, для того, чтобы в логе при возникновении ошибки можно было идентифицировать ее причину.

4.3 Создание продукта в подкаталоге

Для того, чтобы создать продукт в подкаталоге администратор заполняет форму с техническими характеристиками соответствующего продукта, основными полями которой является поле название продукта и его цены. Без заполнения этих полей форма не будет отправлена на сервер.

После отправки формы управление передается контроллеру, в котором создается объект соответствующего продукта и заполняются его поля.

Рассмотрим метод добавления продукта класса Phone. Добавление продукта данного класса осуществляется вызовом метода `addPhone()`:

```

public String addPhone
    (@PathVariable("subcatalog") Subcatalog
        subcatalog,
        @RequestParam("file") MultipartFile file,
        @RequestParam Map<String, String> form) throws
        IOException {
    Phone phone = new Phone();
    phone = setPhone(phone, form);
    phone.setFilename(saveFile(file));
    phone.setSubcatalog(subcatalog);
    phoneRepo.save(phone);
    return "redirect:/catalogList/{catalogName}/
        edit/"+subcatalog.getId();
}

```

Данный метод получает следующие параметры:

– Subcatalog subcatalog – объект класса Subcatalog. С помощью аннотации `@PathVariable("subcatalog")` из адресной строки берется название подкаталога, с помощью которого осуществляется поиск в базе данных соответствующей записи, которой будет проинициализирован данный объект;

– `MultipartFile file` – получает файл загруженный в форму;

– `Map<String, String> form` – получает все поля формы с описанием продукта.

В данном методе сначала создается объект класса Phone, после чего с помощью метода `setPhone()` инициализируется все поля данного объекта. Рассмотрим метод инициализации объекта:

```
public Phone setPhone(Phone phone, Map<String, String>
    form) {
    phone.setName(form.get("name"));
    if(!form.get("price").isEmpty())
        phone.setPrice(Double.parseDouble(form.get(
            "price")));
    phone.setDescription(form.get("description"));
    phone.setCamera(form.get("camera"));
    if(!form.get("numOfSim").isEmpty())
        phone.setNumOfSim(Integer.parseInt(form.get(
            "numOfSim")));
    if(!form.get("numOfDots").isEmpty())
        phone.setNumOfDots(Integer.parseInt(form.get(
            "numOfDots")));
    phone.setOS(form.get("OS"));
    phone.setScreenSize(form.get("screenSize"));
    phone.setScreenResolution(form.get(
        "screenResolution"));
    if(!form.get("RAM").isEmpty())
        phone.setRAM(Integer.parseInt(
            form.get("RAM")));
    if(!form.get("flashMemory").isEmpty())
        phone.setFlashMemory(Integer.parseInt(
            form.get("flashMemory")));
    if(!form.get("numOfCores").isEmpty())
        phone.setNumOfCores(Integer.parseInt(
            form.get("numOfCores")));
    phone.setPlatform(form.get("platform"));
    phone.setCapacityCPU(form.get("capacityCPU"));
    phone.setFrequencyCPU(form.get("frequencyCPU"));
    phone.setNameCPU(form.get("nameCPU"));
    return phone;
}
```

В данном методе целочисленные поля формы перед инициализацией проходят проверку на заполняемость, так как их необходимо привести к соответствующим типам.

После инициализации обычных полей объекта выполняется работа по сохранению файла на сервере. Сохранение файла инициируется функцией `SaveFile()`. Рассмотрим выполнение функции сохранения файла:

```
public String saveFile(MultipartFile file) throws
    IOException {
    if (file != null &&
        !file.getOriginalFilename().isEmpty()) {
        File uploadDir = new File(uploadPath);
        if (!uploadDir.exists()) {
            uploadDir.mkdir();
        }
        String uuidFile = UUID.randomUUID().toString();
        String resultFilename = uuidFile + "." +
            file.getOriginalFilename();
        file.transferTo(new File(uploadPath + "/" +
            resultFilename));
        return resultFilename;
    }
    return null;
}
```

В данном методе осуществляется проверка на то, что файл есть и что его имя содержит символы. Далее создается уникальный идентификатор `UUID`, с помощью которого генерируется уникальное имя файла, под которым он будет сохранен на сервере. Данное имя так же присваивается полю имени файла и сохраняется в базе данных с помощью объекта `phoneRepo` класса `PhoneRepo`.

4.4 Создание и просмотр отзывов

На странице отзывов заполняется форма, которая будет передана методу контроллера `addReview()`. Рассмотрим данный метод:

```
public String addMessage(
    @AuthenticationPrincipal User user,
    @RequestParam String msg, Map<String, Object>
    model) {
    Message message = new Message(msg, user);
    messageRepo.save(message);
    Iterable<Message> messages = messageRepo.findAll();
    model.put("messages", messages);
    return "reviews";
}
```

Для получения имени пользователя, оставившего сообщение, с помощью аннотации `@AuthenticationPrincipal` из контекста берется текущий пользователь страницы и присваивается объекту класса `User`.

Создается объект сообщения `message`, конструктор которого принимает текст сообщения и объект пользователя, оставившего его, после чего данное сообщение сохраняется в базе данных.

Для того, что бы отобразить на странице обновленный список отзывов, создается `Iterable<Message>` в который с помощью интерфейса `MessageRepo` из базы данных достаётся весь текущий список отзывов, который в дальнейшем будет отправлен на страницу в виде модели.

4.4 Создание заказа и работа с корзиной

Создание заказа товара начинается на странице с описанием товара, где пользователь помещает его в корзину нажимая соответствующую кнопку. Функция, вызванная нажатием кнопки обрабатывается скриптом и принимает в себя следующие поля:

- `productType` – тип продукта;
- `id` – идентификатор продукта;
- `productName` – название продукта;
- `price` – цена продукта.

Данные поля хранятся в формате JSON в локальном хранилище браузера.

Для того, чтобы добавить продукт в хранилище, необходимо проверить есть ли там данный продукт:

```
if (localStorage.getItem(String(id)) != null) {  
    amount = JSON.parse(localStorage.getItem(String(id))  
        ).amount;  
    amount++;  
    comp = {  
        productType: pt,  
        amount: amount,  
        id: id,  
        compName: compName,  
        price: price  
    };  
    localStorage.setItem(id, JSON.stringify(comp));  
}
```

Если данный продукт уже существует, увеличивается счетчик `amount`, который говорит о количестве данного вида продукта в корзине, создается объект типа JSON и записывается в local storage браузера.

Если по идентификатору не было найдено продуктов, значит в корзине его нет. Тогда создается новый объект с количеством равным единице и записывается в хранилище.

Когда пользователь нажимает на корзину, в модальном окне корзины создается тело таблицы, в которое из локального хранилища записывается информация о продукте, его название, цена и количество.

Тело таблицы генерируется JavaScript кодом, где в переменную тела таблицы помещается элемент тела таблицы со страницы HTML по его идентификатору.

```
var tbody = document.getElementById('#mcc');
```

Далее из локального хранилища извлекаются JSON объекты и создаются строки и ячейки таблицы:

```
id = JSON.parse(localStorage.key(i));
obj = JSON.parse(localStorage.getItem(String(id)));
var tr = document.createElement('tr');
var th = document.createElement('th');
var td1 = document.createElement('td');
var td2 = document.createElement('td');
```

После создания всех необходимых элементов таблицы им присваиваются значения, которые и будут являться списком продуктов:

```
th.innerHTML = obj.amount;
td1.innerHTML = obj.compName;
td2.innerHTML = (obj.price.replace(",", ".") + " руб");
```

Для того, чтобы передать информацию о заказе в форму была разработана функция `checkout()`, которая берет все объекты продукта из локального хранилища и передает их на страницу в виде JSON строки.

```
function checkout() {
    var jsonObj;
    jsonObj = {
        "products": []
    };
    for (var i = 0; i < localStorage.length; i++) {
        id = JSON.parse(localStorage.key(i));
        obj = JSON.parse(localStorage.getItem(String(id)));
        jsonObj.products[i] = obj;
    }
    document.querySelector('#JSONfield').value
        = JSON.stringify(jsonObj);
}
```

Полученная строка отправляется в форме в обработчик корзины и вызывается метод `order()`. Рассмотрим реализацию данного метода:

```
public String checkout(@RequestParam String jsonfieldd,
Model model) {
    Gson gson = new GsonBuilder().create();
    Map jsonMap = gson.fromJson(jsonfieldd, Map.class);
    ArrayList<LinkedTreeMap<String, String>> arrayList =
```

```

        new ArrayList<LinkedTreeMap<String, String>>());
    arrayList = (ArrayList<LinkedTreeMap<String,
        String>>) jsonMap.get("products");
    for (int i = 0; i < arrayList.size(); i++) {
        arrayList.get(i).put("price", arrayList.
            get(i).get("price").replaceAll("\\p{Z}", ""));
    }
    model.addAttribute("products", arrayList);
    model.addAttribute("catalogs", getCatalogs());
    model.addAttribute("subb", getSubcatalogs());
    return "checkout";
}

```

Для того, чтобы извлечь данные из JSON создается объект класса Gson с помощью которого строка преобразуется в хешированный список. Так как в качестве ключа выступает тип продукта, а значением является весь список продуктов карта преобразуется в массив `LinkedTreeMap<>`, из которого извлекается информация о заказе и в виде модели передается на страницу с оформлением заказа.

4.5 Создание пользователя в системе

Создание пользователя начинается после заполнения формы регистрации пользователем. После отправки формы управление передается контроллеру регистрации, который выполняет метод `addUser()`.

В качестве параметра метод принимает объект пользователя, которому присваивается имя пользователя переданное из формы.

Происходит проверка на наличие пользователя с данным именем в системе. В базе данных происходит поиск по имени пользователя:

```

User userFromDb = userRepo.findByUsername(
    users.getUsername());

```

Если пользователь существует, тогда на страницу вернется сообщение об ошибке.

```

if(userFromDb != null){
    model.put("message", "User exists");
    return "registration";
}

```

Если пользователя с введенным в форму именем не существует, данный объект сохранится в базе данных и ему будет присвоена роль обычного пользователя:

```

users.setRoles(Collections.singleton(Role.USER));
userRepo.save(users);

```

```
return "redirect:/";
```

4.6 Добавление и просмотр контактов магазина:

Добавление контактов начинается после перехода пользователя с правами администратора на страницу контактов. Список контактов состоит из следующих полей:

- `timeWorking` – время работы;
- `phoneNumbers` – номера телефонов;
- `address` – адреса;
- `links` – ссылки.

Так как данные хранятся в одной строке в базе данных, в поля формы ввода, данные, которые имеют множественное число записываются через знак-разделитель «/».

После отправки формы вызывается создается объект контактов, инициализируется данными из нее и сохраняется в базе данных:

```
Contacts con = new Contacts();
con.setLinks(form.get("links"));
con.setTimeWorking(form.get("timeWork"));
con.setPhoneNumbers(form.get("phoneNumbers"));
con.setAddress(form.get("address"));
contactsRepo.save(con);
```

Далее на страницу в виде модели возвращается список контактов, что бы иметь возможность отредактировать текущую версию списка.

```
model.put("phoneNumbers",getIterable(con.getPhoneNumber()));
model.put("links", getIterable(con.getLinks()));
model.put("timeWork",getIterable(con.getTimeWorking()));
model.put("address",con.getAddress());
return "contacts";
```

5 ПРОГРАММА И МЕТОДИКА ИСПЫТАНИЙ

Тестирование программного обеспечения является необходимым этапом разработки программного продукта.

Этот процесс исследования программы позволяет получить информацию о ее качестве и своевременно исправить все недостатки, если они будут обнаружены.

Тестирование программного продукта – важный этап разработки, так как при реализации программного продукта разработчик не может учесть все возможные варианты работы программы, в результате чего могут возникать исключительные ситуации.

Ошибки могут также возникать в протестированном и готовом приложении.

Проведение тестирования программного продукта имеет две различные цели: первое – это демонстрация заказчику, что программа соответствует требованиям, второе разработчику необходимо выявить ситуации, в которых поведение программы является неправильным или нежелательным, для последующего их устранения.

Тестирование программного продукта проводилось в два этапа:

- тестирование и отладка отдельных частей программного средства в процессе их реализации;
- полное тестирование программного средства после завершения работы с кодом программы.

Эти этапы тестирования являются важными и необходимыми для выполнения.

Первый этап тестирования – позволяет обнаруживать и исправлять локальные ошибки для конкретного функционального блока программы.

Выявление локальных ошибок в веб-приложении не должно составлять каких-либо сложностей, гораздо сложнее выявить ошибку в некотором модуле программы, производя тестирование всего программного продукта.

После выявления и исправления ошибок в каждом модуле программы, вполне вероятным остается факт, что ошибки могут возникнуть при взаимодействии различных модулей программы.

Второй этап тестирования так же важен, так как в процессе его выполнения проверяется корректность поведения всей программы в целом.

Тестирование клиентской и административной части проводилось на персональных компьютерах со следующими конфигурациями:

- процессор: Intel Pentium 4 3.0 GHz (2 ядра), оперативная память: 4 ГБ DDR3, видеокарта: Intel HD Graphics, операционная система: любая;
- процессор: Intel Core i32100 CPU 3.00 GHz (4 ядра), оперативная память: 8 ГБ DDR3, видеокарта: Intel HD Graphics, операционная система: любая.

Тесты интерфейса серверной части представлены в таблице 5.1.

Таблица 5.1 – Тестирование административной части

Тестируемая веб-страница	Содержание теста	Ожидаемый результат	Результат теста
1	2	3	4
Интерфейс администратора	В разделе новостей добавить новый продукт.	Сообщение об ошибке с предложением повторного ввода данных.	Не пройден, создается некорректная учетная запись. Добавлена проверка данных, выполняется ожидаемый результат.
Интерфейс администратора	В разделе обратной связи просмотреть обращение.	Запуск веб-страницы просмотра обратной связи.	Пройден. Запущено главное окно приложения.
Интерфейс администратора	Просмотреть раздел отзывов.	Открытие веб-страницы с отзывами.	Пройден. Открытие веб-страницы приложения.
Интерфейс администратора	В меню подкаталогов добавить новый подкаталог.	Сообщение об ошибке с предложением повторного ввода данных.	Пройден. Добавление нового подкаталога.
Интерфейс администратора	В меню редактирования ролей удалить все роли пользователя.	Сообщение об ошибке с предложением присвоения роли.	Пройден, появляется сообщение о необходимости присвоить роль пользователю.
Интерфейс администратора	В меню редактирования вакансий не ввести данные.	Сообщение об ошибке с предложением повторного ввода данных.	Не пройден, аварийное завершение программы. Добавлена проверка данных, выполняется ожидаемый результат.
Интерфейс администратора	В меню редактирования каталога товаров не ввести данные.	Сообщение об ошибке с предложением повторного ввода данных.	Не пройден, редактирование не происходит. Добавлена проверка данных, выполняется ожидаемый результат.
Интерфейс администратора	В разделе вакансий удалить одну вакансию.	Сообщение об успешном удалении вакансии.	Пройден, появляется сообщение о удалении вакансии.

Тесты интерфейса клиентской части представлены в таблице 5.2.

Таблица 5.2 – Тестирование клиентской части

Тестируемое окно	Содержание теста	Ожидаемый результат	Результат теста
1	2	3	4
Окно авторизации.	Ввести некорректный логин и пароль	Сообщение об ошибке с шаблоном ввода корректных данных.	Пройден, успешная авторизация администратора.
Страница отзывов.	Нажать кнопку «Отправить отзыв» при отключении БД.	Сообщение о недоступности БД.	Не пройден, ничего не происходит. Добавлена проверка на доступность БД перед отправкой запроса.
Страница отзывов.	Ввести некорректные данные в отзыве.	Сообщение об ошибке с предложением повторного ввода данных.	Не пройден, аварийное завершение программы. Добавлена проверка данных, выполняется ожидаемый результат.
Страница обратной связи.	Нажать кнопку «Отправить» для обратной связи при выключенной БД.	Сообщение о недоступности БД.	Не пройден, ничего не происходит. Добавлена проверка на доступность БД перед отправкой запроса, выполняется ожидаемый результат.
Главная страница.	Навести курсор на каталог и выбрать соответствующий подкаталог.	Страница с продуктами выбранного подкаталога.	Пройден, отображается страница с продуктами подкаталога.
Страница с описанием продукта.	Нажать кнопку «Добавить в корзину», находясь на странице с описанием продукта.	Сообщение о том, что продукт успешно добавлен в корзину.	Пройден, появляется сообщение об успешном добавлении продукта в корзину.

Продолжение таблицы 5.2

Модальное окно корзины.	Нажать кнопку «Оформить».	Пользователь попадает на страницу оформления заказа с информацией о своем заказе.	Пройден, пользователь попадает на страницу с оформлением заказа.
Страница оформления заказа.	Нажать кнопку «Отправить заказ».	Сообщение о необходимости заполнения полей с контактными данными.	Пройден, появляется сообщение о необходимости заполнения полей контактных данных.

Из таблиц 5.1 и 5.2 следует, что этап тестирования интерфейсов оказался достаточно продуктивным, т.к. выявил множество серьезных ошибок, приводивших к некорректной работе приложения, в том числе, к аварийным завершениям. Обнаруженные проблемы были оперативно устранены.

6 РУКОВОДСТВО ПОЛЬЗОВАТЕЛЯ

6.1 Развертывание приложения и настройка сервера

Для стабильной работы веб-приложения, обрабатывающего достаточно большое количество пользовательских запросов, а также хранящего относительно большие объемы данных, необходим сервер, обладающий достаточными аппаратными ресурсами, а также инструменты, позволяющие обновлять приложение, осуществлять поддержку и исправление проблем в приемлемые сроки.

В разработке данного программного продукта использовался сервер для приложений Tomcat, но при желании приложение можно развернуть на любом сервере, поддерживающем выполнения Java-приложений, таком как Weblogic или WebSphere.

Для автоматизации сборки проекта использован фреймворк Maven – инструмент для компиляции проекта, сборки его в WAR или JAR-файлы, генерации документации и разрешения зависимостей внешних библиотек приложения.

Так как проект разработан на языке Java, такие серверы приложений, как Tomcat, существуют на всех основных операционных системах, фреймворк Maven не зависит от операционной системы в принципе, нет разницы, на какой операционной системе работает сервер, на котором будет развернуто приложение.

Перед сборкой реального пакета для развертывания на сервере, имеет смысл протестировать его сборку. Чтобы осуществить сборку приложения в режиме, в котором оно будет собрано на реально сервере, необходимо выполнить команду:

```
./mvnw -Pprod
```

Эта команда скомпилирует, протестирует и соберет в пакет приложение с теми настройками, с которыми оно будет собрано на реальной машине.

Чтобы собрать приложение в исполняемый WAR-файл с помощью Maven, необходимо выполнить команду:

```
./mvnw -Pprod package
```

Команда сгенерирует два файла:

```
target/appmonitor-0.0.1-SNAPSHOT.war  
target/appmonitor -0.0.1-SNAPSHOT.war.original
```

Первый файл – это исполняемый WAR-файл (далее будет описан процесс его запуска). Он также может быть развернут на сервере

приложений, но в его состав включены все runtime-библиотеки, что увеличивает его объем, поэтому рекомендуется использовать второй файл с расширением «.original», если необходимо развернуть приложение на таких серверах, как Tomcat, Weblogic или Websphere.

Вместо развертывания приложения на сервере, многие пользователи предпочитают запускать приложение непосредственно из WAR-файла.

Чтобы запустить WAR-файл, созданный на предыдущем этапе, на операционных системах MacOS и Linux необходимо выполнить команду:

```
./appmonitor-0.0.1-SNAPSHOT.war
```

На операционной системе Windows необходимо выполнить команду:

```
java -jar appmonitor-0.0.1-SNAPSHOT.war
```

Минимальные требования для работы административной и клиентской части системы:

- процессор Intel Pentium 4 или старше;
- 512 мегабайт оперативной памяти;
- операционная система семейства Windows.

Приложение использует такие технологии, как Spring Boot и JavaScript, поэтому для запуска на клиентской стороне существуют следующие минимальные системные требования для версии браузеров:

- Google Chrome 60+;
- Mozilla Firefox 54+;
- Opera 47+;
- Internet Explorer 10+;
- Microsoft Edge 14+;

6.2 Руководство по использованию программного средства

6.2.1 Главная страница веб-приложения

Перейдя по ссылке на главную страницу приложения пользователь видит основную страницу со списком продукции интернет-магазина. Для того, что бы перейти в другие разделы веб-приложения необходимо перейти по пунктам меню:

Меню состоит из разделов:

- Авторизация;
- Контакты;
- Отзывы;
- Замечания и предложения.

Главная страница веб-приложения изображена на рисунке 6.1.

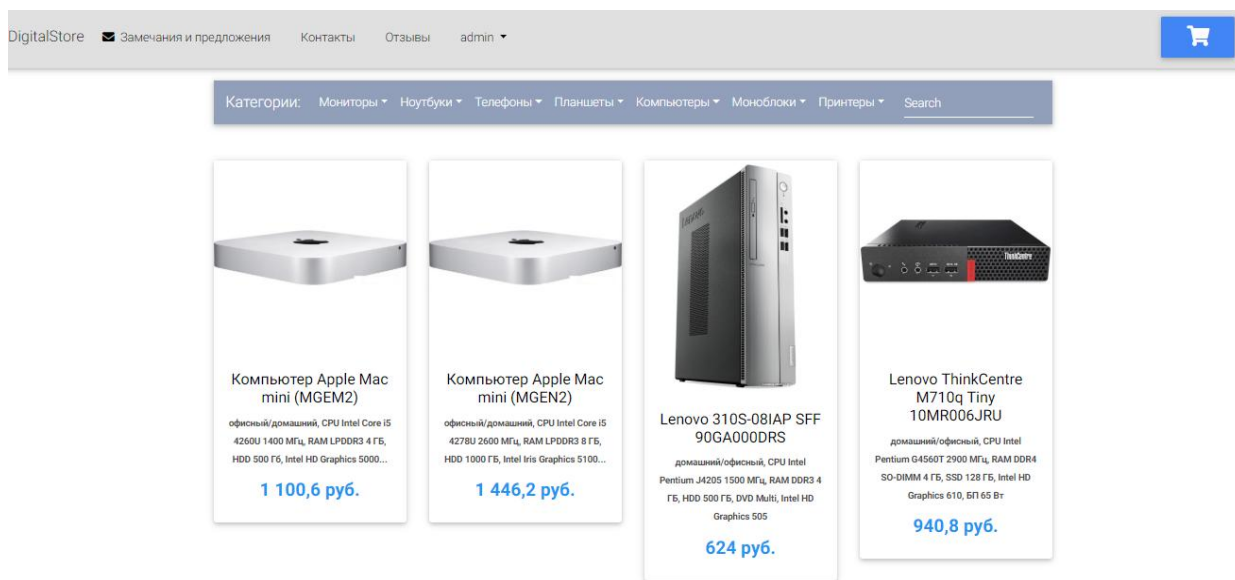


Рисунок 6.1 – Главная страница веб-приложения

Карточка каждого товара содержит его имя, краткое описание и цену. При нажатии на карточку с товаром осуществляется переход на страницу товара, на которой пользователь сможет увидеть увеличенную фотографию товара, подробное его описание в виде таблицы.

6.2.2 Регистрация и авторизация

Для авторизации в приложении, пользователю необходимо перейти на окно авторизации, выбрав необходимый пункт на панели навигации (см. рисунок 6.2).

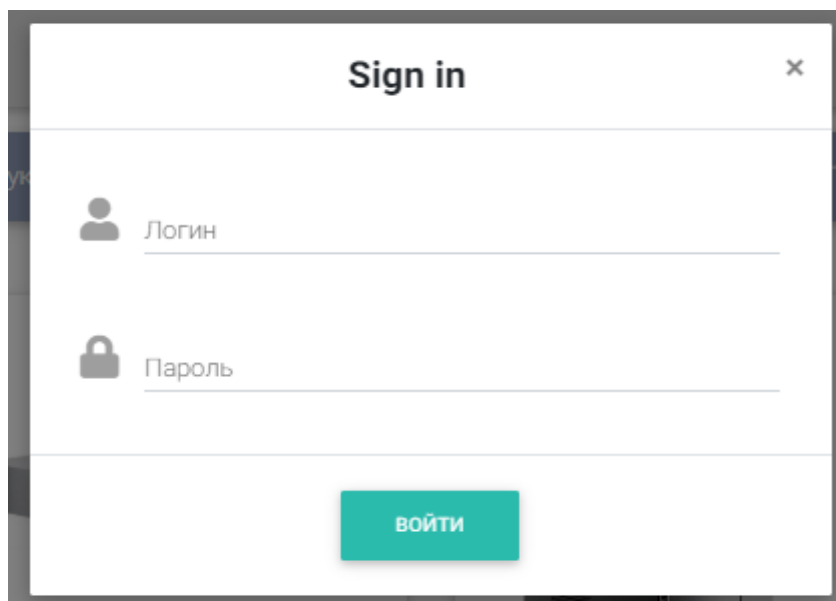


Рисунок 6.2 – Модальное окно авторизации

Если у пользователя в данный момент не существует логина и пароля, ему необходимо зарегистрироваться. Переход на страницу регистрации (см. рисунок 6.3) осуществляется при нажатии на выпадающую при наведении на пункт меню авторизации кнопку регистрация.

Для регистрации пользователю необходимо ввести следующие данные:

- имя пользователя (логин);
- пароль;
- подтверждение пароля.

После нажатия в модальном окне на кнопку зарегистрироваться пользователь с правами администратора вместо кнопки авторизации будет видеть свое имя, а при наведении на него увидит список страниц, которые можно редактировать и кнопку выхода из системы.

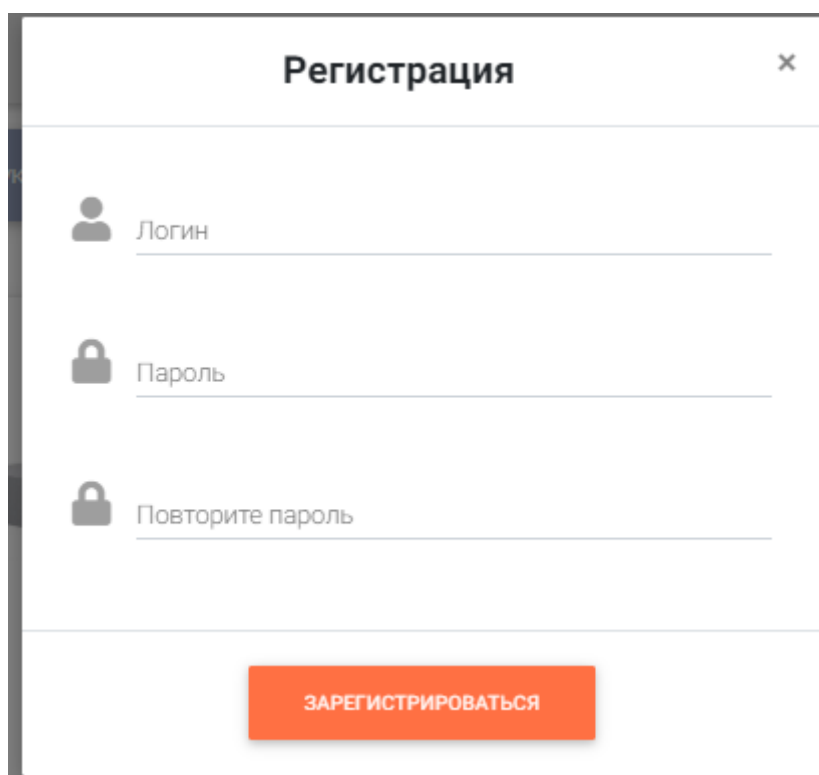


Рисунок 6.3 – Модальное окно Регистрации

6.2.3 Создание подкаталогов

Пользователь с правами администратора может менять всю основную информацию сайта. Изначально на сайте существует список каталогов (см. рисунок 6.4).

Для того, что бы создать подкаталог в одном из каталогов, пользователю приложения с правами администратора необходимо выбрать пункт меню «Редактировать каталог», после чего он попадет на административную страницу каталогов. Страница с административным списком каталогов представлена на рисунке 6.4.

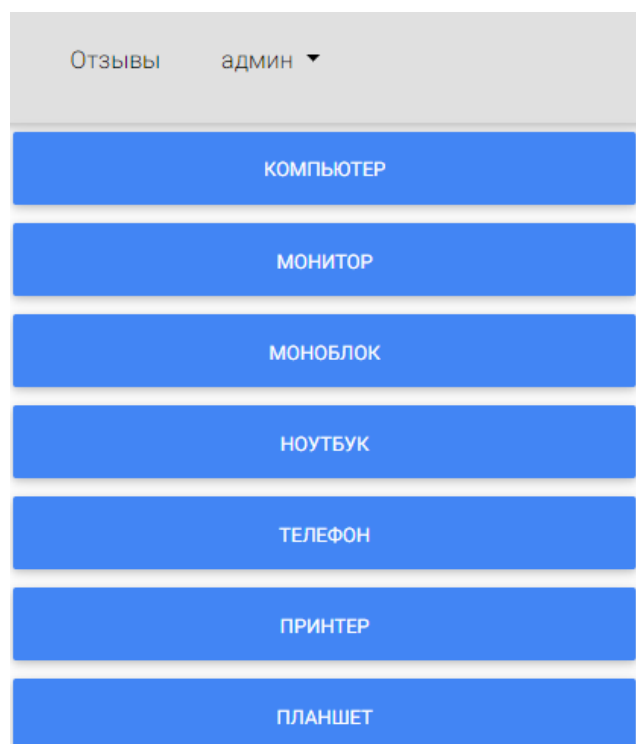


Рисунок 6.4 – Административная страница каталогов

После нажатия на одну из кнопок каталога осуществится переход на страницу подкаталогов, которая так же является административной. Страница списка подкаталогов представлена на рисунке 6.5.

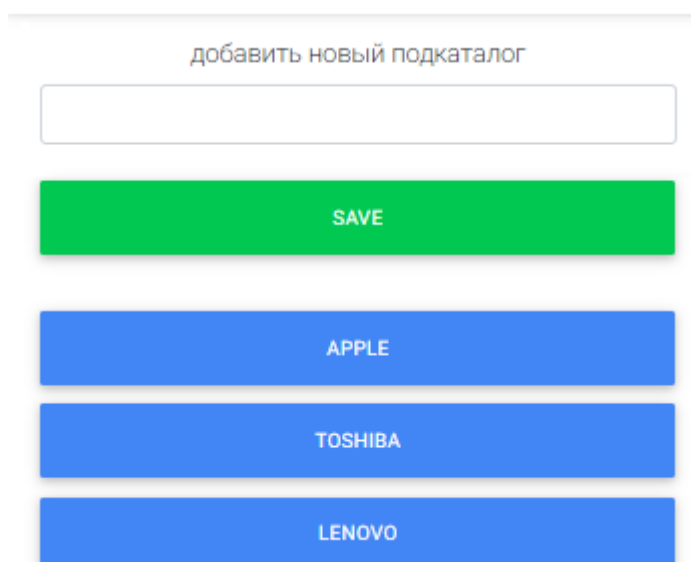


Рисунок 6.5 – Административная страница подкаталогов

Для создания подкаталога в текстовом поле необходимо ввести имя нового подкаталога и нажать кнопку «SAVE». Если каталог с введенным

именем существует в базе данных, на страницу будет выведено сообщение в модальном окне.

6.2.4 Управление подкаталогом

Для того, что бы создать продукт, в списке подкаталогов (см. рисунок 6.5) необходимо выбрать 1 из подкаталогов и перейти на его страницу. Вид административной страницы подкаталога представлен на рисунке 6.6.

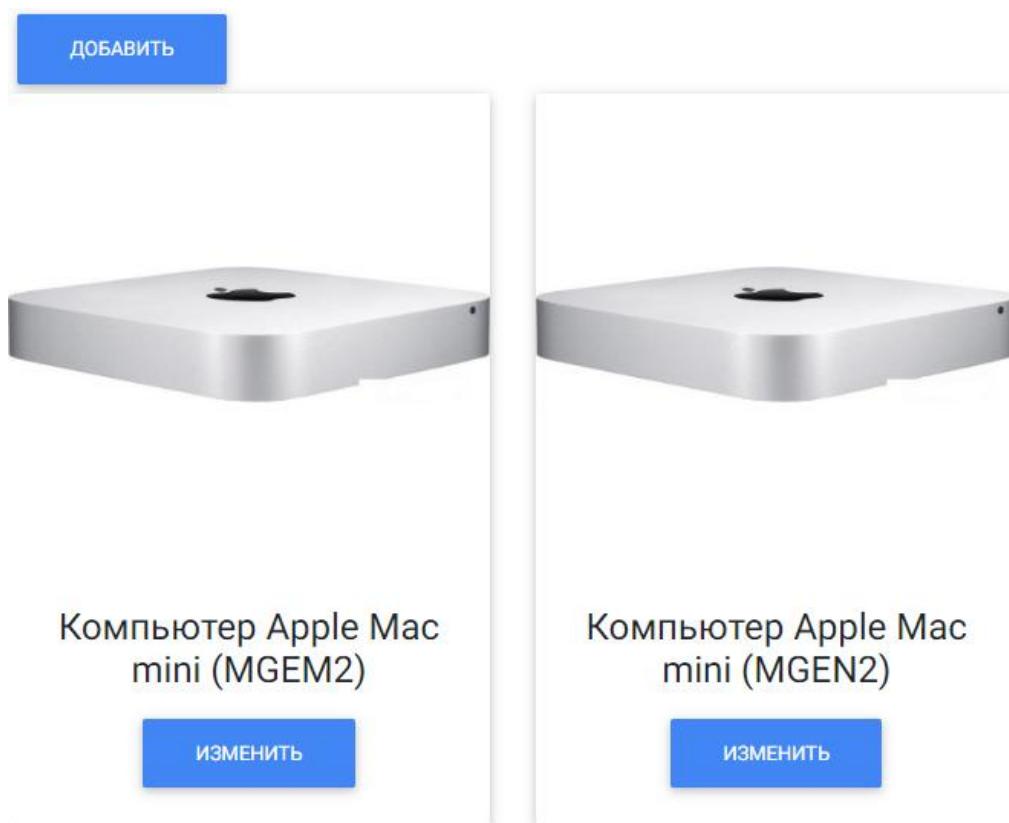


Рисунок 6.6 – Административная страница подкаталога

Что бы добавить новый продукт, необходимо нажать на кнопку «Добавить» вверху страницы, после чего в выпадающем меню заполнить поля формы (см. рисунок 6.7) с соответствующими типу продукта характеристиками.

Обязательные для заполнения поля:

- название продукта;
- цена;
- описание;
- в наличии.

Форма заполнения продукта типа «Компьютер» представлена на рисунке 6.7.

Название продукта:	Чипсет:	Емкость накопителя:
<input type="text"/>	<input type="text"/>	<input type="text"/>
Цена:	Кол-во слотов памяти:	Карты памяти:
<input type="text"/>	<input type="text"/>	<input type="text"/>
Описание:	Максимальный объем памяти:	Конфигурация накопителя:
<input type="text"/>	<input type="text"/>	<input type="text"/>
Количество процессоров:	Объем памяти:	Скорость вращения:
<input type="text"/>	<input type="text"/>	<input type="text"/>
Название процессора:	Тип оперативной памяти:	Тип хранения:
<input type="text"/>	<input type="text"/>	<input type="text"/>
Кол-во ядер:	Частота оперативной памяти:	Выберите Файл <input type="button" value="Browse"/>
<input type="text"/>	<input type="text"/>	
Тактовая частота:	Графический адаптер(видеокарта):	
<input type="text"/>	<input type="text"/>	
Кэш(общий, L2 или L3):	Тип графического адаптера:	
<input type="text"/>	<input type="text"/>	

Рисунок 6.7 – Форма для добавления продукта «Компьютер»

Для того, что бы отредактировать описание продукта, на странице подкаталога в карточке соответствующего продукта необходимо нажать кнопку «Изменить», после чего будет выполнен переход на страницу редактирования продукта, на которой будет выведена форма с заполненными ранее полями. Далее администратор вносит правки в форму и сохраняет ее, так же как и в случае добавления нового продукта. Продукт так же можно удалить, нажав соответствующую кнопку внизу формы существующего продукта.

6.2.4 Просмотр информации о продукте

При нажатии на главной странице сайта на карточку продукта, осуществляется переход на страницу с описанием соответствующего продукта (см. рисунок 6.8). В описании приведено увеличенное изображение продукта, его технические характеристики, а так же информация о наличии продукта в магазине.

Контакты

Отзывы

Авторизация

Категории:

Мониторы

Ноутбуки

Телефоны


Планшеты

Компьютеры

Моноблоки

Принтеры

Search



Компьютер Apple Mac mini (MGEN2)

Наличие: Есть в наличии

Описание: официальный/домашний, CPU Intel Core i5 4278U 2600 МГц, RAM LPDDR3 8 Гб, HDD 1000 Гб, Intel Iris Graphics 5100...

723,2 руб (с НДС)

В корзину

Общая информация

Количество ядер	1
Процессор	Intel Core i5
Количество ядер	2
Частота тактирования	2 600

Рисунок 6.8 – Страница с описанием продукта

6.2.5 Просмотр страницы отзывов

При нажатии в главном меню шапки на кнопку «Отзывы» осуществляется переход на страницу, на которой отображены сообщения, оставленные пользователями магазина (см. рисунок 6.9).

Страница содержит таблицу, в которой пользователи без прав администратора видят колонки и текстовое поле для ввода текста отзыва. В колонке «Пользователь» отображено имя пользователя, если он авторизован на сайте, иначе отображается имя «UNKNOWNUSER».

Пользователи с правами администратора видят колонку «Действие», в ней напротив каждого сообщения находится кнопка удаления сообщения. Обычные пользователи эту колонку не видят и не имеют доступа к обработчикам этих кнопок.

#	Пользователь	Отзыв	Действие
1	UNKNOWNUSER	Доставка быстрая , курьер очень вежлив . Даже перезвонили узнать доволен ли я покупкой.	DELETE
2	UNKNOWNUSER	Хороший магазин.	DELETE
3	UNKNOWNUSER	Все понравилось.	DELETE
4	UNKNOWNUSER	Спасибо огромное Вам за работу. Отличный интернет-магазин!!!	DELETE
5	VasyaPupkin	Приятно покупать	DELETE

Текст

[SEND](#)

Рисунок 6.9 – Страница отзывов

6.2.6 Просмотр страницы контактов

При нажатии в главном меню шапки на кнопку «Контакты» осуществляется переход на страницу, на которой отображены контакты владельцев интернет-магазина.

Вид страницы глазами пользователя с правами администратора(см. рисунок 6.10) представляет из себя форму, поля которой заполняются с разделителем.

Время работы

Пн-Пт: с 09:00 до 18:00 / Сб-Вс: выходной

Phone numbers

+375 (17) 253-28-73 / +375 (29) 666-43-22 / +375 (29) 277-02-01 / +375 (25) 667-57-63

Ссылки

info@digitalstore.by / Skype digitalstore.by / Viber 375296664322

Адрес

г. Минск, ул. Калинина, д. 7, комн. 6, 1 этаж

[SAVE](#)

Рисунок 6.10 – Форма редактирования страницы контактов

Отображение страницы для пользователей не обладающих правами администратора представлен на рисунке 6.11.

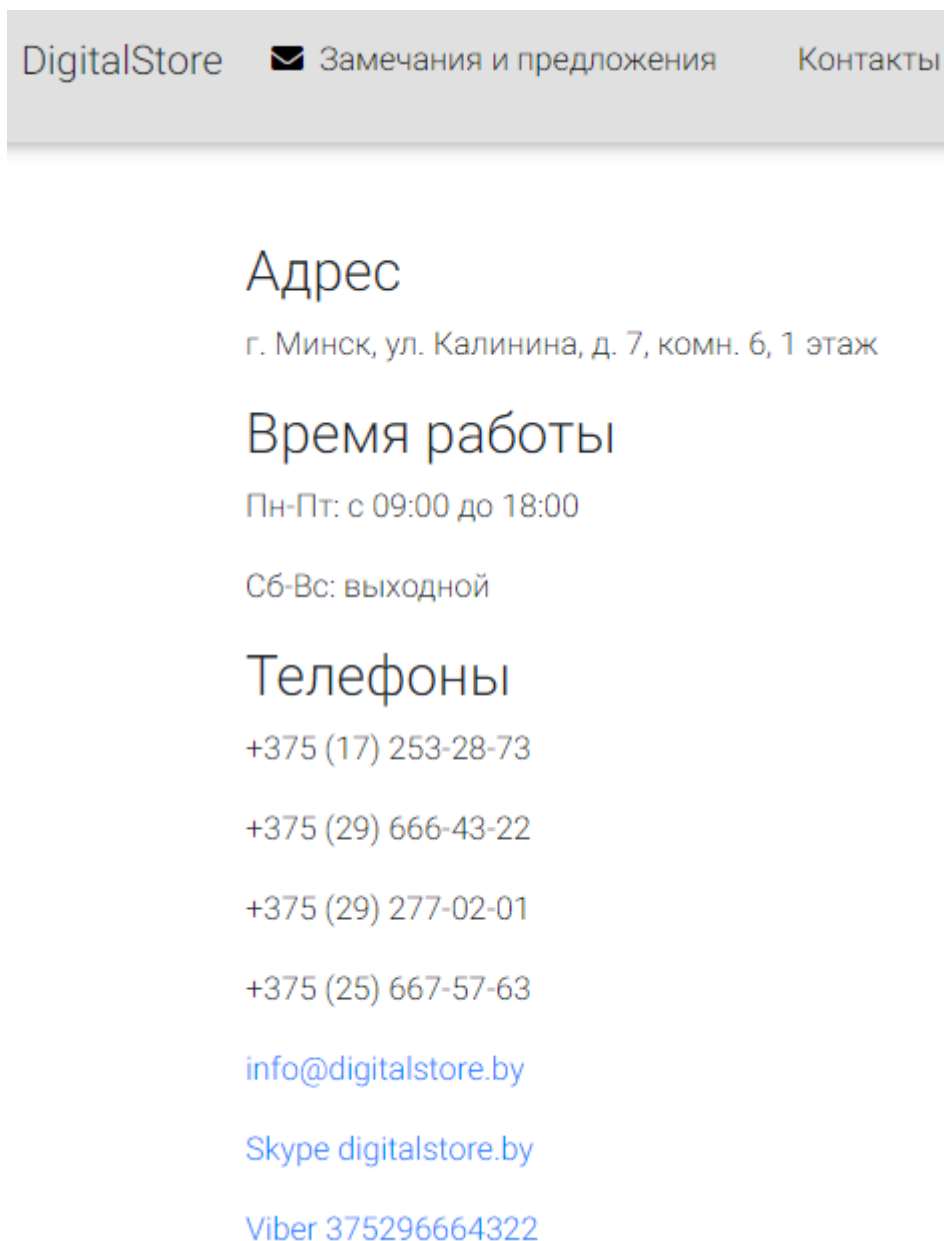


Рисунок 6.11 – Содержимое страницы контактов

6.2.7 Просмотр страницы замечаний и предложений

При нажатии в главном меню шапки на кнопку «Замечания и предложения» осуществляется переход на страницу, содержащей форму замечаний и предложений (см. рисунок 6.12).

Пользователю предлагается заполнить форму, в которой он указывает свою почту, тему письма и текст письма, который при нажатии на кнопку «Отправить» отправляется на почту владельца магазина.

Оставьте ваш отзыв

Если у вас есть какие-либо замечания или предложения заполните следующую форму

Ваше имя	Ваш email
Тема письма	
Ваше сообщение	
<input type="button" value="ОТПРАВИТЬ"/>	

Рисунок 6.12 – Страница замечаний и предложений

6.2.8 Управление ролями пользователей

Для того, что бы отредактировать роли пользователей либо изменить их имя необходимо в выпадающем меню администратора перейти на страницу со списком пользователей (см. рисунок 6.13) и выбрать соответствующего пользователя для редактирования.

Name	Role	
admin	ADMIN, USER	<input type="button" value="ИЗМЕНИТЬ"/>
user1	USER	<input type="button" value="ИЗМЕНИТЬ"/>
admin2	ADMIN, USER	<input type="button" value="ИЗМЕНИТЬ"/>
VasyaPupkin	USER	<input type="button" value="ИЗМЕНИТЬ"/>

Рисунок 6.13 – Страница списка пользователей

После выбора пользователя осуществится переход на страницу с формой редактирования ролей (см. рисунок 6.14).

Имя пользователя

VasyaPupkin

☒ USER

☐ ADMIN

Сохранить

Рисунок 6.14 – Форма редактирования ролей

6.2.9 Работа с корзиной

Находясь на странице с описанием продукта (см. рисунок 6.8) пользователь может заказать доставку данного продукта нажав на кнопку «В корзину». Содержимое объекта продукт попадает в локальное хранилище браузера, а из него отображается в корзине, просмотреть которую можно нажав на значок корзины на панели главного меню сайта.

Содержимое корзины с заказами представлено на рисунке 6.15. После нажатия кнопки «Оформить» пользователь попадет на страницу оформления заказа, где необходимо ввести контактные данные и отправить их на почту владельца сайта.

Корзина ×			
Кол-во	Название продукта	Цена за единицу	Действие
1	Lenovo 310S-08IAP SFF 90GA000DRS	624 руб	×
1	Компьютер Apple Mac mini (MGEM2)	1 100.6 руб	×
2	Компьютер Apple Mac mini (MGEM2)	1 446.2 руб	×
1	Lenovo ThinkCentre M710q Tiny 10MR006JRU	940.8 руб	×
ЗАКРЫТЬ		ОФОРМИТЬ	

Рисунок 6.15 – Корзина с товарами

6.2.10 Просмотр страницы с оформлением заказа

При нажатии на кнопку корзины «Оформить», пользователь попадет на страницу оформления заказа (см. рисунок 6.16), где ему необходимо будет ввести свои контактные данные для обратной связи.

При нажатии на кнопку оформления заказа, на почту владельца сайта придет сообще о том, что один из пользователей хочет заказать доставку продукта. В письме указаны все продукты, их количество, цена и все данные из полей формы личных данных и адреса доставки.

Название	Кол-во	Цена
Lenovo 310S-08IAP SFF 90GA000DRS	1	624 руб.
Компьютер Apple Mac mini (MGEM2)	1	1 100,6 руб.
Компьютер Apple Mac mini (MGEM2)	2	2 892,4 руб.
Lenovo ThinkCentre M710q Tiny 10MR006JRU	1	940,8 руб.

Сумма заказа: 5 557,8 руб.

Личные данные

Адрес доставки

Ваше имя

Ваша фамилия

Ваш e-mail

Ваш контактный телефон

Минская область

Беларусь

Ваш адрес

Индекс

ВСЕ ДАННЫЕ ВЕРНЫ, ОФОРМИТЬ ЗАКАЗ

Рисунок 6.16 – Страница оформления товара

7 ТЕХНИКО-ЭКОНОМИЧЕСКОЕ ОБОСНОВАНИЕ РАЗРАБОТКИ ВЕБ-ПРИЛОЖЕНИЯ ИНТЕРНЕТ-МАГАЗИНА КОМПЬЮТЕРНОЙ ТЕХНИКИ

В данном дипломном проекте разработан программный продукт, предназначенный для продажи различной компьютерной техники.

Приложение даст возможность просматривать каталоги продукции, так же каждый продукт в отдельности с описанием всех его характеристик, находить товар по его основным параметрам производительности. Для потенциальных сотрудников реализована форма обратной связи, в которой пользователь может написать письмо на почту владельца сайта и оставить свои контакты для дальнейшего сотрудничества. Репутацию сайта можно оценить на странице отзывов.

Так же у приложения есть административная часть, которая позволяет пользователю с правами администратора производить следующие действия:

- Создание подкаталогов для уже существующих каталогов;
- Редактирование контактных данных;
- Удаление комментариев на странице отзывов;
- Добавление продуктов в подкаталоги;
- Управление страницей вакансий;
- Управление ролями пользователей.

7.1 Определение трудоемкости разработки программного продукта

Трудоемкость создания программного продукта может быть определена укрупненным методом. При этом необходимо воспользоваться формулой:

$$T_{p3} = T_{oa} + T_{bc} + T_{п} + T_{отл} + T_{др} + T_{до}, \quad (7.1)$$

где T_{oa} – трудоемкость подготовки описания задачи и исследования алгоритма решения;

T_{bc} – трудоемкость разработки блок-схемы алгоритма;

T_n – трудоемкость программирования по готовой блок-схеме;

$T_{отд}$ – трудоемкость отладки программы на персональном компьютере;

$T_{\text{др}}$ – трудоемкость подготовки документации по задаче в рукописи;

$T_{до}$ – трудоемкость редактирования, печати и оформления документации по задаче.

Составляющие приведенной формулы (7.1) определяются, в свою очередь, через условное число операторов Q в разрабатываемом программном продукте по формуле:

$$Q = q \cdot C \cdot (1 + p), \quad (7.2)$$

где q – число операторов в программе;

C – коэффициент сложности программы;

p – коэффициент коррекции программы в ходе её разработки.

Составляющие трудоёмкости разработки программы определяются по формулам:

$$T_{oa} = \frac{Q \cdot W \cdot K}{80}, \quad (7.3)$$

$$T_{6c} = \frac{Q \cdot K}{24}, \quad (7.4)$$

$$T_{п} = \frac{Q \cdot K}{24}, \quad (7.5)$$

$$T_{отл} = \frac{Q \cdot K}{4,6}, \quad (7.6)$$

$$T_{др} = \frac{Q \cdot K}{18}, \quad (7.7)$$

$$T_{до} = 0,75 \cdot T_{др}, \quad (7.8)$$

где Q – условное число операторов в разрабатываемом программном продукте;

W – коэффициент увеличения затрат труда вследствие недостаточного или некачественного описания задачи;

K – коэффициент квалификации разработчика алгоритмов и программ.

При подсчете всех операторов в программе получим $q = 1275$ и приняв $C = 1,4$, а $p = 0,2$, то по формуле (7.2), получим условное число операторов:

$$Q = 1275 \cdot 1,4 \cdot (1 + 0,2) = 2142. \quad (7.9)$$

При стаже работы разработчика до двух лет получаем $K = 0,8$ и $W = 1,2$. Используя эти данные и результат выражения (7.9), вычислим значения формул (7.3) – (7.7):

$$T_{oa} = \frac{2142 \cdot 1,2 \cdot 0,8}{80} \approx 25,71, \quad (7.10)$$

$$T_{\text{бс}} = \frac{2142 \cdot 0,8}{24} \approx 71,4, \quad (7.11)$$

$$T_{\text{п}} = \frac{2142 \cdot 0,8}{24} \approx 71,4, \quad (7.12)$$

$$T_{\text{отл}} = \frac{2142 \cdot 0,8}{4,6} \approx 372,52, \quad (7.13)$$

$$T_{\text{др}} = \frac{2142 \cdot 0,8}{18} \approx 95,2. \quad (7.14)$$

Используя результат (7.14) по формуле (7.8) получим значение трудоемкости редактирования, печати и оформления документации по задаче:

$$T_{\text{до}} = 0,75 \cdot 95,2 \approx 71,4. \quad (7.15)$$

Зная значения формул (7.10) – (7.15) по формуле (7.1) вычислим значение трудоемкости разработки:

$$T_{\text{рз}} = 25,71 + 71,4 + 71,4 + 372,52 + 95,2 + 71,4 = 708. \quad (7.16)$$

7.2 Определение стоимости машино-часа работы персонального компьютера

Стоимость машино-часа работы персонального компьютера определяется по формуле:

$$S_{\text{мч}} = C_{\text{э}} + \frac{A_{\text{пк}} + P_{\text{пк}} + A_{\text{пл}} + P_{\text{пл}} + P_{\text{ар}}}{\Phi_{\text{пк}}}, \quad (7.17)$$

где $C_{\text{э}}$ – расходы на электроэнергию за час работы персонального компьютера, руб.;

$A_{\text{пк}}$ – годовая величина амортизационных отчислений на реновацию персонального компьютера;

$P_{\text{пк}}$ – годовые затраты на ремонт и техническое обслуживание персонального компьютера, руб.;

$A_{\text{пл}}$ – годовая величина амортизационных отчислений на реновацию производственных площадей, занимаемых персональным компьютером, руб.;

$P_{\text{пл}}$ – годовые затраты на ремонт и содержание производственных площадей, руб.;

$P_{ар}$ – годовая величина арендных платежей за помещение, занимаемое персональным компьютером, руб.;

$\Phi_{пк}$ – годовой фонд времени работы персонального компьютера, ч.

Расходы на электроэнергию за час работы персонального компьютера определяются по формуле:

$$C_э = N_э \cdot k_{ис} \cdot Ц_э, \quad (7.18)$$

где $N_э$ – установленная мощность блока питания персонального компьютера, кВт;

$k_{ис}$ – коэффициент использования энергоустановок по мощности;

$Ц_э$ – стоимость киловатт-часа электроэнергии, р.

По паспортным данным блока питания персонального компьютера, на котором ведется разработка программного продукта, равна 0,065 кВт. Коэффициент использования энергоустановок по мощности примем равным 0,9, а стоимость киловатт-часа электроэнергии – 0,25 руб. Используя эти данные можно вычислить расходы на электроэнергию за час работы персонального компьютера:

$$C_э = 0,065 \cdot 0,9 \cdot 0,25 \approx 0,015 \text{ руб.} \quad (7.19)$$

Годовая величина амортизационных отчислений на реновацию персонального компьютера определяется по формуле:

$$A_{пк} = Ц_{пк}^б \cdot \frac{H_{пк}^a}{100}, \quad (7.20)$$

где $Ц_{пк}^б$ – балансовая стоимость персонального компьютера, руб.;

$H_{пк}^a$ – норма амортизационных отчислений на персональный компьютер, %.

Балансовая стоимость персонального компьютера определяется по формуле:

$$Ц_{пк}^б = Ц_{пк} \cdot k_y \cdot k_m, \quad (7.21)$$

где $Ц_{пк}$ – цена персонального компьютера на момент её выпуска, р.;

k_y – коэффициент удорожания персонального компьютера;

k_m – коэффициент, учитывающий затраты на монтаж и транспортировку персонального компьютера.

Приняв стоимость нового персонального компьютера равной 1140 руб., коэффициент удорожания персонального компьютера равным 1, а коэффициент, учитывающий затраты на монтаж и транспортировку персонального компьютера, равным 1,05, по формуле (7.21) вычислим

балансовую стоимость персонального компьютера:

$$\text{Ц}_{\text{ПК}}^{\text{б}} = 1140 \cdot 1 \cdot 1,05 \approx 1197,00 \text{ руб.} \quad (7.22)$$

При норме амортизационных отчислений на персональный компьютер, равной 10%, годовая величина амортизационных отчислений на реновацию персонального компьютера составит:

$$\text{А}_{\text{ПК}} = 1197,00 \cdot \frac{10}{100} = 119,70 \text{ руб.} \quad (7.23)$$

Годовые затраты на ремонт и техническое обслуживание персонального компьютера укрупнённо могут быть определены по формуле:

$$\text{Р}_{\text{ПК}} = \text{Ц}_{\text{ПК}}^{\text{б}} \cdot k_{\text{ро}}, \quad (7.24)$$

где $k_{\text{ро}}$ – коэффициент, учитывающий затраты на ремонт и техническое обслуживание персонального компьютера, в том числе затраты на запчасти, зарплату ремонтного персонала и др.

Приняв $k_{\text{ро}} = 0,13$, годовые затраты на ремонт и техническое обслуживание персонального компьютера составят:

$$\text{Р}_{\text{ПК}} = 1197,00 \cdot 0,13 \approx 155,61 \text{ руб.} \quad (7.25)$$

Годовая величина амортизационных отчислений на реновацию производственных площадей, занятых персональным компьютером, определяется по формуле:

$$\text{А}_{\text{пл}} = \text{Ц}_{\text{пл}}^{\text{б}} \cdot \frac{\text{Н}_{\text{пл}}^{\text{а}}}{100}, \quad (7.26)$$

где $\text{Ц}_{\text{пл}}^{\text{б}}$ – балансовая стоимость площадей, руб.;

$\text{Н}_{\text{пл}}^{\text{а}}$ – норма амортизационных отчислений на производственные площади, %.

Балансовая стоимость площадей определяется по формуле:

$$\text{Ц}_{\text{пл}}^{\text{б}} = S_{\text{ПК}} \cdot k_{\text{д}} \cdot \text{Ц}_{\text{пл}}, \quad (7.27)$$

где $S_{\text{ПК}}$ – площадь, занимаемая персональным компьютером, м^2 ;

$k_{\text{д}}$ – коэффициент, учитывающий дополнительную площадь;

$\text{Ц}_{\text{пл}}$ – цена квадратного метра производственной площади, р.

При условии того, что персональный компьютер занимает площадь

равную $1,5 \text{ м}^2$, а $k_d = 3$ и стоимость одного квадратного метра производственной площади равна 300 руб., тогда балансовая стоимость площадей по формуле (7.27) составит:

$$\text{Ц}_{\text{пл}}^6 = 1,5 \cdot 3 \cdot 300 = 1350,00 \text{ руб.} \quad (7.28)$$

Приняв $\text{Н}_{\text{пл}}^a = 1,2\%$, по формуле (7.26) получим:

$$\text{А}_{\text{пл}} = 1350,00 \cdot \frac{1,2}{100} \approx 16,20 \text{ руб.} \quad (7.29)$$

Годовые затраты на ремонт и содержание производственных площадей укрупнённо могут быть определены по формуле:

$$\text{Р}_{\text{пл}} = \text{Ц}_{\text{пл}}^6 \cdot k_{\text{рэ}}, \quad (7.30)$$

где $k_{\text{рэ}}$ – коэффициент, учитывающий затраты на ремонт и эксплуатацию производственных площадей.

Приняв $k_{\text{рэ}} = 0,05$, по формуле (7.30) получим:

$$\text{Р}_{\text{пл}} = 1350,00 \cdot 0,05 = 67,50 \text{ руб.} \quad (7.31)$$

Годовая величина арендных платежей за помещение, занимаемое персональным компьютером, рассчитывается по формуле:

$$\text{Р}_{\text{ар}} = S_{\text{пк}} \cdot k_d \cdot k_{\text{ар}} \cdot k_{\text{комф}} \cdot k_{\text{пов}} \cdot 12, \quad (7.32)$$

где $k_{\text{ар}}$ – ставка арендных платежей за помещение;

$k_{\text{комф}}$ – коэффициент комфортности помещения;

$k_{\text{пов}}$ – повышающий коэффициент, учитывающий географическое размещение площади.

Для офиса расположенного в Минске в помещении цокольного этажа соответствуют значения $k_{\text{комф}} = 0,9$ и $k_{\text{пов}} = 0,9$, и приняв ставку арендных платежей за помещение равной $13,60 \text{ руб./м}^2$, получим:

$$\text{Р}_{\text{ар}} = 1,5 \cdot 3 \cdot 13,60 \cdot 0,9 \cdot 0,9 \cdot 12 = 594,86 \text{ руб.} \quad (7.33)$$

Годовой фонд времени работы персонального компьютера определяется исходя из режима её работы, и может быть рассчитан по формуле:

$$\Phi_{\text{пк}} = t_{\text{сч}} \cdot T_{\text{сг}}, \quad (7.34)$$

где t_{cc} – среднесуточная фактическая загрузка персонального компьютера, ч;

$T_{сг}$ – среднее количество дней работы персонального компьютера в год.

Приняв, что среднесуточная фактическая нагрузка равна 8 часам, а в году 305 рабочих дней, получим по формуле (7.34):

$$\Phi_{пк} = 8 \cdot 305 = 2440 \text{ ч.} \quad (7.35)$$

Теперь по формуле (7.17) получим значение стоимости работы машино-часа персонального компьютера:

$$S_{мч} = 0,015 + \frac{119,70 + 155,61 + 16,20 + 67,50 + 594,86}{2440} \approx 0,40 \text{ руб.} \quad (7.36)$$

7.3 Определение себестоимости создания программного продукта

Для определения себестоимости создания программного продукта необходимо определить затраты на заработную плату разработчика по формуле:

$$З_{pz} = T_{pz} \cdot t_{чр} \cdot K_{пр} \cdot \left(1 + \frac{H_d}{100}\right) \cdot \left(1 + \frac{H_{соц}}{100}\right), \quad (7.37)$$

где T_{pz} – трудоёмкость разработки программного продукта, чел-ч;

$t_{чр}$ – среднечасовая ставка работника, осуществлявшего разработку программного продукта, руб.;

$K_{пр}$ – коэффициент, учитывающий процент премий в организации-разработчике;

H_d – норматив дополнительной заработной платы;

$H_{соц}$ – норматив отчислений от фонда оплаты труда.

Среднечасовая ставка работника определяется исходя из единой тарифной системы оплаты труда в Республике Беларусь по следующей формуле:

$$t_{чр} = \frac{ЗП_{1р} \cdot k_T}{170}, \quad (7.38)$$

где $ЗП_{1р}$ – среднемесячная заработная плата работника первого разряда;

k_T – тарифный коэффициент работника соответствующего разряда;

170 – среднее нормативное количество рабочих часов в месяце.

При условии того, что у сотрудника, разрабатываемого проект, одиннадцатый разряд первой категории, тарифный коэффициент работника

составит 3,54, а размер заработной платы – 450 руб. Тогда по формуле (7.38) среднечасовая ставка работника составит:

$$t_{\text{чр}} = \frac{450 \cdot 3,54}{170} \approx 9,37 \text{ руб.} \quad (7.39)$$

Примем $H_{\text{соц}} = 34,6\%$, $K_{\text{пр}} = 1,7$, а $H_{\text{д}} = 15$, тогда:

$$\begin{aligned} Z_{\text{рз}} &= 708 \cdot 9,37 \cdot 1,7 \cdot \left(1 + \frac{15}{100}\right) \cdot \left(1 + \frac{34,6}{100}\right) \approx \\ &\approx 17456,80 \text{ руб.} \end{aligned} \quad (7.40)$$

В себестоимость разработки программного продукта включаются также затраты на отладку программного продукта в процессе его создания. Затраты на отладку программы определяются по формуле:

$$Z_{\text{от}} = T_{\text{отл}} \cdot S_{\text{мч}}. \quad (7.41)$$

Используя расчеты, приведенные выше, получим:

$$Z_{\text{от}} = 372,52 \cdot 0,40 \approx 149,008 \text{ руб.} \quad (7.42)$$

Себестоимость разработки программного продукта определяется по формуле:

$$C_{\text{пр}} = Z_{\text{рз}} \cdot F + Z_{\text{от}}, \quad (7.43)$$

где F – коэффициент накладных расходов проектной организации без учёта эксплуатации персонального компьютера.

При условии того, что $F = 1,15$, по формуле (7.43) получим:

$$C_{\text{пр}} = 17456,80 \cdot 1,15 + 149,008 \approx 20224,328 \text{ руб.} \quad (7.44)$$

7.4 Определение оптовой и отпускной цены программного продукта

Оптовая цена складывается из себестоимости создания программного продукта и плановой прибыли на программу.

Оптовая цена программного продукта определяется по формуле:

$$C_{\text{о}} = C_{\text{пр}} + P_{\text{р}}, \quad (7.45)$$

где P_p – плановая прибыль на программу, руб.

Плановая прибыль на программу определяется по формуле

$$P_p = C_{пр} \cdot H_{п}, \quad (7.46)$$

где $H_{п}$ – норма прибыли проектной организации.

Приняв норму прибыли проектной организации равной 0,25, по формуле (7.46) плановая прибыль составит:

$$P_p = 20224,328 \cdot 0,25 \approx 5056,08 \text{ руб.} \quad (7.47)$$

Оптовая цена программного продукта составит:

$$C_o = 20224,328 + 7552,19 = 27776,51 \text{ руб.} \quad (7.48)$$

Отпускная цена программы определяется по формуле

$$C_{пр} = C_o + (З_{pz} + P_p) \cdot \text{НДС}, \quad (7.49)$$

где НДС – ставка налога на добавленную стоимость.

На текущий момент ставка налога на добавленную стоимость составляет 20%. Тогда по формуле (7.49) отпускная цена программы составит:

$$C_{пр} = 27776,51 + (20224,328 + 5056,08) \cdot 0,2 \approx 32832,526 \text{ руб.} \quad (7.50)$$

7.5 Определение годовых текущих затрат, связанных с эксплуатацией задачи

Для расчёта годовых текущих затрат, связанных с эксплуатацией программного продукта, необходимо определить время решения данной задачи на персональном компьютере.

Время решения задачи на персональном компьютере определяется по формуле:

$$T_3 = (T_{вв} + T_p + T_{выв}) \cdot \frac{1 + d_{пз}}{60}, \quad (7.51)$$

где $T_{вв}$ – время ввода в персональный компьютер исходных данных, необходимых для решения задачи, мин;

T_p – время вычислений, мин;

$T_{\text{выв}}$ – время вывода результатов решения задачи, мин;
 $d_{\text{пз}}$ – коэффициент, учитывающий подготовительно-заключительное время.

Время ввода в персональный компьютер исходных данных может быть определено по формуле

$$T_{\text{вв}} = \frac{K_z \cdot H_z}{100}, \quad (7.52)$$

где K_z – среднее количество знаков, набираемых с клавиатуры при вводе исходных данных;

H_z – норматив набора 100 знаков, мин.

Среднее число символов, набираемых с клавиатуры при вводе исходных данных, составляет 354,52. Приняв $H_z = 6$, тогда по формуле (7.52):

$$T_{\text{вв}} = \frac{298,96 \cdot 6}{100} \approx 21,27 \text{ мин.} \quad (7.53)$$

T_p и $T_{\text{выв}}$ по сравнению с $T_{\text{вв}}$ занимают ничтожно малое время, и стремятся к нулю. Приняв $d_{\text{пз}} = 0,15$, получим:

$$T_z = (21,27 + 0 + 0) \cdot \frac{1 + 0,15}{60} \approx 0,53 \text{ ч.} \quad (7.54)$$

На основе рассчитанного времени решения задачи может быть определена заработная плата пользователя данного программного продукта. Затраты на заработную плату пользователю программного продукта определяются по формуле:

$$Z_{\text{п}} = T_z \cdot k \cdot t_{\text{чп}} \cdot K_{\text{пр}} \cdot \left(1 + \frac{H_{\text{д}}}{100}\right) \cdot \left(1 + \frac{H_{\text{соц}}}{100}\right), \quad (7.55)$$

где k – периодичность решения задачи в течение года, раз/год;

$t_{\text{чп}}$ – среднечасовая ставка пользователя программы, руб.

Пользоваться данным программным продуктом будет сотрудник восьмого разряда без категории, для которого соответствует тарифный коэффициент работника в размере 2,97, а заработная плата – 300 руб. Тогда по формуле (7.38), среднечасовая ставка пользователя программы составит:

$$t_{\text{чп}} = \frac{300 \cdot 2,97}{170} \approx 5,24 \text{ руб.} \quad (7.56)$$

В среднем данным продуктом будут пользоваться 6700 раз в год. При условии того, что в организации, где работает пользователь, приняты $H_{\text{соц}} = 34,6\%$, $K_{\text{пр}} = 1,5$, а $H_{\text{д}} = 10$, затраты на заработную плату пользователю программного продукта составят:

$$Z_{\text{п}} = 0,53 \cdot 6700 \cdot 5,24 \cdot 1,5 \cdot \left(1 + \frac{10}{100}\right) \cdot \left(1 + \frac{34,6}{100}\right) \approx 31324,67 \text{ руб.} \quad (7.57)$$

В состав затрат, связанных с решением задачи, включаются также затраты, связанные с эксплуатацией персонального компьютера.

Затраты на оплату аренды персонального компьютера для решения задачи определяются по следующей формуле

$$Z_{\text{а}} = T_{\text{з}} \cdot k \cdot S_{\text{мч}}. \quad (7.58)$$

В связи с тем, что компьютеры разработчика и пользователя данным программным продуктом совпадают, то используя результат вычисления (7.36), получим затраты на оплату аренды персонального компьютера для решения задачи в размере:

$$Z_{\text{а}} = 0,53 \cdot 6700 \cdot 0,40 \approx 1420 \text{ руб.} \quad (7.59)$$

Годовые текущие затраты, связанные с эксплуатацией задачи, определяются по формуле:

$$Z_{\text{т}} = Z_{\text{п}} + Z_{\text{а}} = 31324,67 + 1420 = 32744,67 \text{ руб.} \quad (7.60)$$

7.6 Определение годовых текущих затрат, связанных с эксплуатацией задачи, при использовании аналога

Для расчёта годовых текущих затрат, связанных с эксплуатацией аналога программного продукта, необходимо определить время решения данной задачи на персональном компьютере.

Время решения задачи на персональном компьютере определяется по формуле (7.51). В то время как время ввода в персональный компьютер исходных данных может быть определено по формуле (7.52).

Среднее число символов, набираемых с клавиатуры при вводе исходных данных, при использовании аналога, составляет 438,52. Приняв $H_{\text{з}} = 6$, тогда:

$$T_{\text{вва}} = \frac{438,52 \cdot 6}{100} \approx 26,31 \text{ мин.} \quad (7.61)$$

$T_{\text{ра}}$ и $T_{\text{выва}}$ по сравнению с $T_{\text{вв}}$ занимают ничтожно малое время и в случае использования аналога, и стремятся к нулю. Приняв $d_{\text{пза}} = 0,25$, получим:

$$T_{\text{за}} = (26,31 + 0 + 0) \cdot \frac{1 + 0,25}{60} \approx 0,54 \text{ ч.} \quad (7.62)$$

В среднем данным продуктом будут пользоваться 7500 раз в год. При условии того, что в организации, где работает пользователь, приняты $H_{\text{соц}} = 34,6\%$, $K_{\text{пр}} = 1,5$, $H_{\text{д}} = 10$, а среднечасовая ставка пользователя программы вычислена по формуле (7.56), то затраты на заработную плату пользователя аналога программного продукта составят:

$$\begin{aligned} Z_{\text{па}} &= 0,54 \cdot 6700 \cdot 5,24 \cdot 1,5 \cdot \left(1 + \frac{10}{100}\right) \cdot \left(1 + \frac{34,6}{100}\right) \approx \\ &\approx 42104,53 \text{ руб.} \end{aligned} \quad (7.63)$$

Затраты на оплату аренды персонального компьютера для решения задачи определяются по формуле (7.58).

В связи с тем, что компьютеры разработчика и пользователя данным программным продуктом совпадают, то используя результат вычисления (7.36), получим затраты на оплату аренды персонального компьютера для решения задачи, при использовании аналога, в размере:

$$Z_{\text{аа}} = 0,54 \cdot 6700 \cdot 0,40 \approx 1448 \text{ руб.} \quad (7.64)$$

Годовые текущие затраты, связанные с эксплуатацией задачи, при использовании аналога, определяются по формуле:

$$Z_{\text{та}} = Z_{\text{па}} + Z_{\text{аа}} = 42104,53 + 1448 = 43552,53 \text{ руб.} \quad (7.65)$$

7.7 Определение ожидаемого прироста прибыли в результате внедрения программного продукта

Ожидаемый прирост прибыли в результате внедрения задачи взамен использования аналога укрупнённо может быть определен по формуле:

$$P_y = (Z_{\text{та}} - Z_{\text{т}}) \cdot \left(1 - \frac{C_{\text{нп}}}{100}\right), \quad (7.66)$$

где $C_{\text{нп}}$ – ставка налога на прибыль.

На данный момент ставка налога на прибыль составляет 18%. Тогда:

$$P_y = (43552,53 - 32744,67) \cdot \left(1 - \frac{18}{100}\right) \approx 8862,44 \text{ руб.} \quad (7.67)$$

7.8 Расчет показателей эффективности использования программного продукта

Для определения годового экономического эффекта от разработанной программы необходимо определить суммарные капитальные затраты на разработку и внедрения программы по формуле:

$$K_o = K_z + C_{\text{пр}}, \quad (7.68)$$

где K_z – капитальные и приравненные к ним затраты;

$C_{\text{пр}}$ – отпускная цена программы.

Капитальные и приравненные к ним затраты определяются по формуле:

$$K_z = C_{\text{пк}}^b \cdot \left(1 - x \cdot \frac{H_{\text{пк}}^a}{100}\right) \cdot T_z \cdot \frac{k}{\Phi_{\text{пк}}}, \quad (7.69)$$

где $C_{\text{пк}}^b$ – балансовая стоимость комплекта вычислительной техники, необходимой для решения задачи, руб.;

x – возраст используемого персонального компьютера, лет;

При условии того, что $x = 2$:

$$K_z = 1197,00 \cdot \left(1 - 2 \cdot \frac{10}{100}\right) \cdot 0,53 \cdot \frac{6700}{6700} \approx 507,53 \text{ руб.} \quad (7.70)$$

Суммарные капитальные затраты составят:

$$K_o = 507,53 + 32832,526 = 33340,1 \text{ руб.} \quad (7.71)$$

Годовой экономический эффект от замены используемого программного обеспечения определяется по формуле:

$$\text{ЭФ} = P_y - E \cdot K_o, \quad (7.72)$$

где E – коэффициент эффективности, равный ставке по кредитам на рынке долгосрочных кредитов.

Примем $E = 0,1$, получим:

$$\text{ЭФ} = 8862,44 - 0,1 \cdot 33340,1 \approx 5528,34 \text{ руб.} \quad (7.73)$$

Срок возврата инвестиций определяется по формуле:

$$T_v = \frac{K_o}{P_y} = \frac{33340,1}{8862,44} \approx 3,7 \text{ лет.} \quad (7.74)$$

Результаты технико-экономических показателей проекта приведены в таблице 7.1.

В результате технико-экономического обоснования инвестиций в разработку программного продукта были получены следующие значения показателей их эффективности:

1. Прирост условной прибыли составит 8862,44 руб.
2. Годовой экономический эффект составит 5528,34 руб.
3. Все инвестиции окупаются за 3,7 лет.

Таким образом, разработка данного программного продукта является эффективной и инвестиции в её реализацию целесообразны.

Таблица 7.1 – Техничко-экономические показатели проекта

Наименование показателя	Величина показателя
1. Трудоёмкость решения задачи, ч	708
2. Периодичность решения задачи, раз/год	6700
3. Годовые текущие затраты, связанные с решением задачи, руб.	32744,67
4. Отпускная цена программы, руб.	33340,1
5. Степень новизны программы	В
6. Группа сложности алгоритма	3
7. Прирост условной прибыли, руб.	8862,44
8. Годовой экономический эффект, руб.	5528,34
9. Срок возврата инвестиций, лет	3,7

ЗАКЛЮЧЕНИЕ

Целью дипломного проекта являлась разработка интернет-магазина компьютерной техники.

В соответствии с поставленной целью были реализованы следующие Задачи:

- возможность просматривать каталоги товара;
- возможность добавлять товары в корзину и заказывать их заполнив соответствующую форму заказа;
- возможность заполнить на сайте специальную форму-заявку и указать в ней свои реквизиты и пожелания. Форма-заявка автоматически должна пересылаться администратору для обработки;
- возможность заходить на страницу, где будут отображаться все вакансии интернет-магазина;
- вход для администратора веб-приложения, который имеет возможность редактировать всю информацию: создавать и удалять продукты, каталоги, список контактов, вакансии, редактировать роли пользователей системы;
- разработка интерфейса;
- разработка базы данных, в которой будет храниться вся информация о магазине;
- регистрация и авторизация пользователей.

По своей структуре проект представляет собой административную и клиентскую часть, которые взаимодействуют через БД.

Разработанный проект обладает списком плюсов по отношению к аналогам, а именно:

- администрирование (вся информация в веб-приложении редактируема);
- возможность выбора товара и заказа;
- простота навигации по приложению.

С целью дальнейшего развития проекта были определены следующие усовершенствования программного продукта:

- улучшения дизайна интерфейса пользователя и администратора;
- создание личного электронного кабинета для пользователей.