МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Южно-Уральский государственный университет»
(национальный исследовательский университет)
Высшая школа экономики и управления
Кафедра «Информационные технологии в экономике»

РАБОТА ПРОВЕРЕНА
Рецензент, к.т.н. доцент

_____/Е.Ю. Алексеева/
«_____» _____2019 г.

ДОПУСТИТЬ К ЗАЩИТЕ
Заведующий кафедрой, д.т.н., с.н.с

_____/Б.М. Суховилов/
«____» _____2019 г.

Разработка интернет – площадки по торговле подарками

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
ЮУрГУ–09.03.03.2019.032 ВКР

Руководитель, ст.преп.
_____/А.Г. Палей/
«_____» _____2019 г.

Автор
студент группы ЭУ-438
_____/Д.С. Алиев/
«_____» _____2019 г.

Нормоконтролер, доцент
_____/Е.А.Конова/
«_____» _____2019 г.

Челябинск 2019

АННОТАЦИЯ

Алиев Д.С. Разработка онлайн площадки для торговли подарками. – Челябинск: ЮУрГУ, ЭиУ-438, 48 с., 17 ил., 6 табл., библиогр. список – 6 наим., 4 прил.

Выпускная квалификационная работа выполнена с целью создания онлайн площадки для торговли подарками.

В работе обоснована актуальность выбранной темы, сформулирована цель и задачи. Кроме того, проанализированы аналогичные существующие информационные системы, выявлены их достоинства и недостатки.

Описана структура приложения. Разработана система для онлайн торговли подарками, которая успешно отлажена и протестирована.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

В настоящее время актуальной является задача, объединения и координирования агентов рынка через интернет. На данный момент в подарочном бизнесе нет сервисов, эффективно связывающих покупателей с оптимальным поставщиком услуги или товара в этой области. Существует несколько решений, помогающих пользователю найти нужный подарок, соответствующий его требованиям, но эти сервисы имеют ряд существенных недостатков.

Так же существует большое количество компаний и предпринимателей, которые предлагают множество разнообразных товаров в сфере подарков. Сейчас, у них, основным способом поиска клиентов в интернете является контекстная реклама и реклама в соц. сетях. Эти методы поиска клиентов имеют недостатки: высокая стоимость, сложность настройки и большое количество клиентов, которые не являются целевыми.

Целью дипломного проекта является разработка онлайн сервиса для торговли подарками.

Задачи дипломного проекта:

- Анализ существующих решений;
- Разработка архитектуры приложения;
- Проектировка база данных;
- Выбор инструментов для разработки системы;
- Проектировка интерфейса;
- Написание кода приложения;
- Отладка приложения.

В дипломной работе разработано web-приложение, которое позволяет продавать и находить подарки.

1 ПОСТАНОВКА ЗАДАЧИ

1.1 Технические требования к системе

Главным недостатком сервисов, на которых можно выбрать подарок, является отсутствие фильтра по городам. Вследствие этого основная масса продавцов находится в Москве, а покупатели из регионов вынуждены пользоваться доставкой в свой город. Это создаёт проблемы, как покупателям, так и продавцам:

- Доставка в город происходит в среднем в течение 2-5 дней.
- Не все товары из сегмента подарков подлежат транспортировке
- Доставка требует дополнительных трат.

Таким образом, большое количество людей вынуждены отказываться от использования сервисов для онлайн поиска подарков.

Существующие сервисы подбора подарков не являются торговыми площадками, и лишь ссылаются на онлайн-магазины. У этого подхода есть несколько недостатков:

- Для торговли нужно иметь веб-сайт;
- Нет возможности быстро разместить свои товары на платформе;
- Нет возможности отследить совершил ли человек покупку или нет;
- Не всегда на сторонних сайтах удобный процесс оформления покупки;
- Чтобы совершить покупку человеку необходимо покинуть сайт.

Существующие торговые площадки не являются узкоспециализированными именно на подарках. Вследствие чего отсутствуют удобные инструменты для поиска подарков и не все товары подходят под категорию подарков.

Основные требования к системе:

- Возможность регистрации и авторизации;
- Добавление, изменение и удаление товаров, авторизированными пользователями;
- Возможность загружать изображения товаров;
- Понятный интерфейс;
- Возможность поиска товаров, соответствующих заданным требованиям, для всех пользователей.

1.2 Анализ существующих информационных систем

В настоящее время существует несколько онлайн-сервисов предоставляющих возможность найти либо продать подарок. Все они имеют ряд преимуществ и недостатков.

Список рассматриваемых конкурентов:

- Ярмарка мастеров (livemaster.ru);
- Ламбада маркет (lmbd.ru);
- Подарки.ру (podarki.ru);
- Идеи Подарков (ideipodarkov.net);
- Яндекс Маркет Подарки (market.yandex.ru/gifts).

1.3 Сервис «Ярмарка мастеров»

Торговая площадка для покупки и продажи авторских handmade-работ и дизайнерских вещей: украшений, одежды, аксессуаров, предметов интерьера и прочего. Большое количество товаров на сайте попадают под категорию «Подарки». На сайте есть раздел «Сувениры и подарки ручной работы».

«Ярмарка мастеров» является крупнейшим порталом на Российском рынке в своей сфере и является аналогом международного портала «Etsy». Посещаемость данного сервиса доходит до 300 000 посетителей в сутки.

Монетизация происходит через продажу «Клубных Карт», которые позволяют продавать на сайте определенное количество товаров. Главная страница сайта изображена на рисунке 1.

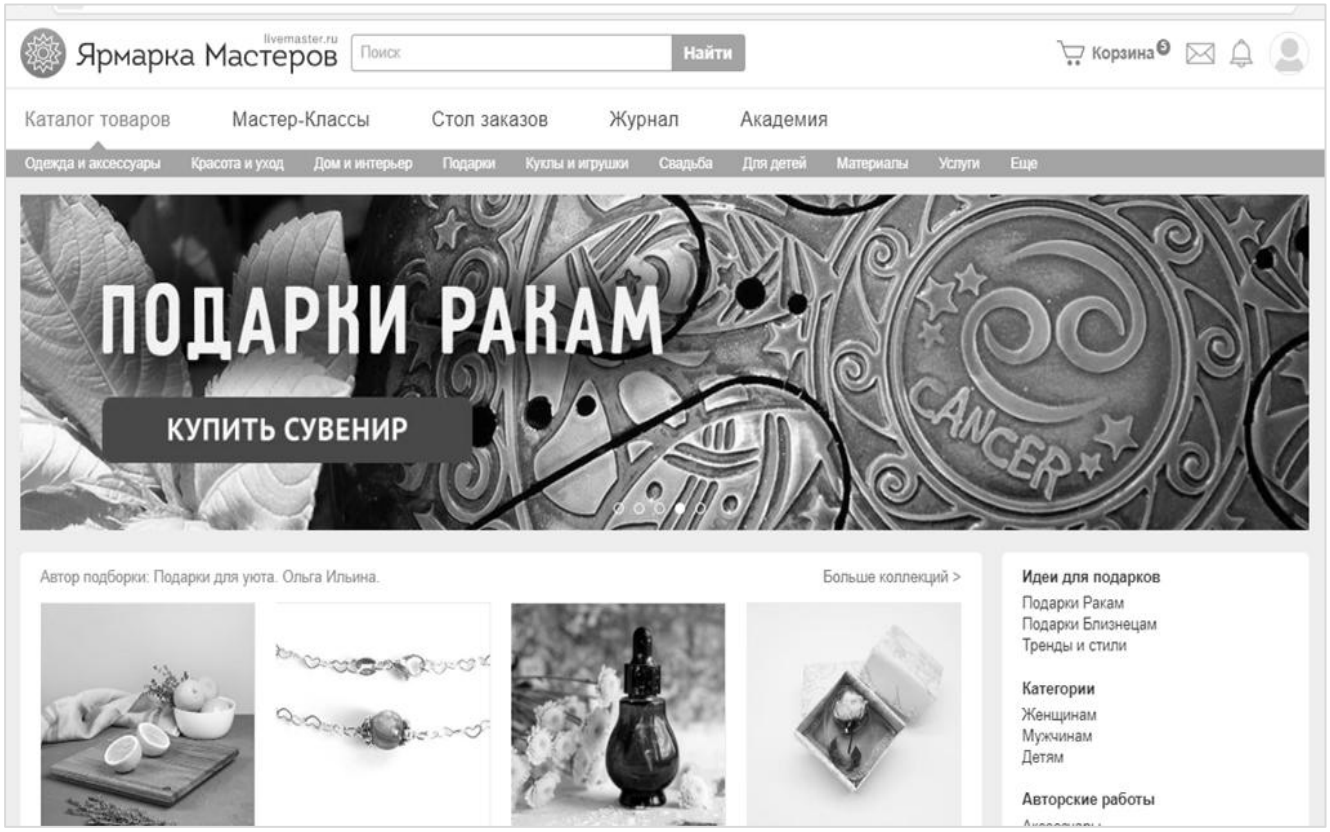

Рисунок 1 – Главная страница портала «Ярмарка мастеров»

На главной странице портала располагаются: логотип, строка поиска, ссылка на страницу «корзина», ссылка на страницу «сообщения», кнопка подписки на рассылку, ссылка на личный кабинет, категории товаров.

Сильные стороны сервиса

- Высокая посещаемость;
- Давно на рынке;
- Большой ассортимент;
- Возможность онлайн оплаты;
- Большое количество категорий;
- Присутствие в большом количестве соц. Сетей.

Слабые стороны

- Сложно торговать в регионах;
- Трудно найти товар в своем городе, если это не Москва;
- Нет фильтров для поиска подарков;
- Почти нет охвата мужской аудитории;
- Платное размещение больше 3-ех товаров.

1.4 Сервис «Ламбада маркет»

«Ламбада-маркет» — это онлайн площадка для торговли уникальными товарами ручной работы. На этом портале находят друг друга покупатели и продавцы со всей России и не только, которых объединяет интерес к модной и винтажной одежде, к маленьким тиражам, к аксессуарам и обуви, к эко-косметике, к уютным предметам быта и интерьера, к антикварной и вручную сделанной мебели, к небольшим семейным бизнесам и магазинам, которые созданы друзьями и единомышленниками.

Данный сервис отличается строгой модерацией товаров и их фотографий. Поэтому продаваемые товары являются качественными и представлены в «Выгодном свете». Но также из этого следует снижение ассортимента.

На сайте нет раздела «Подарки» в отличии от «Ярмарки мастеров», это говорит о том, что на данный момент этот портал не уделяет достаточно внимания данному сегменту рынка.

Товары на данной площадке представлены из многих городов России. Также можно найти товары из Украины, Беларуси, Германии и Бельгии.

Монетизация происходит через покупку подписки на определенное количество мест для продажи. Стоимость одного места — 8 рублей на 3 месяца. Минимальный платёж — 64 рубля.

Главная страница сайта изображена на рисунке 2.

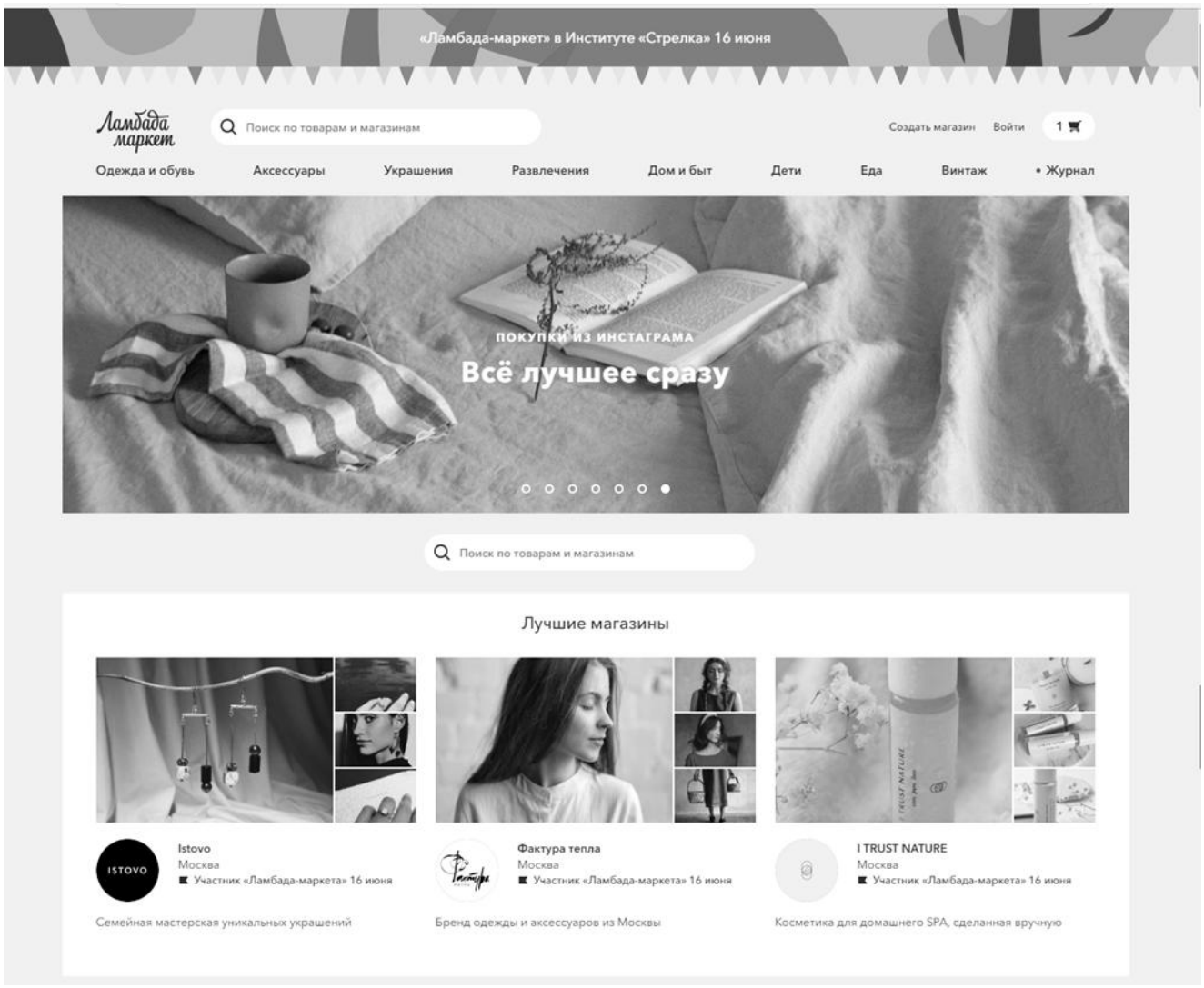

Рисунок 2 – Главная страница портала «Ламбада маркет»

На главной странице портала располагаются: логотип, строка поиска, ссылка на страницу «корзина», кнопка «создать магазин», кнопка входа в личный кабинет, категории товаров, слайдер, список лучших магазинов.

Сильные стороны сервиса:

- Высокое качество товаров;
- Сформированное и растущее сообщество;
- Присутствие во многих соц. Сетях;
- Удобный процесс оформления покупок ;
- Фотографии высокого качества;
- Возможность найти товар в своём городе;
- Лаконичный, современный дизайн.

Слабые стороны:

- Высокие цены
- Долгая загрузка страниц сайта;
- Нет фильтров для поиска подарков;
- Почти нет охвата мужской аудитории;
- Платное размещение;
- Нет возможности онлайн оплаты;
- Маленький ассортимент.

1.5 Сервис «Подарки ру»

Подарки.ру — это сервис подборки подарков по поводам, интересам, профессиям, праздникам и другим критериям. Данный сервис помогает выбрать подарок и подскажет где его можно купить.

Для размещения товаров нужно иметь свою страницу с возможностью оформить покупку.

Сайт имеет множество категорий, но отсутствует функция поиска по заданным критериям.

Главная страница сайта изображена на рисунке 3.

Рисунок 3 – Главная страница портала «Подарки ру»

На главной странице портала располагаются: логотип, кнопка поиска, описание сайта и категории товаров.

Сильные стороны сервиса

- Много категорий товаров
- Пользуется популярностью в своей нише
- Понятные условия партнёрской программы
- Фиксированная оплата за переходы с сервиса

Слабые стороны

- Сложно найти товар в своем городе, если это не Москва
- Нельзя оформить покупку на сайте, без перехода на сайт продавца
- Не удаляются нерабочие ссылки
- Для размещения на сайте нужна своя интернет страница для товара

1.6 Сервис «Идеи подарков»

Сайт «ideipodarkov.net» создан, чтобы помочь выбрать оригинальный и уместный подарок по любому поводу. Сервис предоставляет поиск, учитывающий особенности человека и позволяющий подобрать самые подходящие подарки для нужного случая.

Есть возможность указать для поиска примерную цену подарка, возраст, пол и особенности характера одариваемого, а сервис выдаст список подарков, подходящий запросу. Для каждого подарка выводится также список страничек в интернет-магазинах, где вы его можно заказать.

Так же на сайте предусмотрена возможность выбрать город, но эта функция никак не учитывается при поиске.

На сайте отсутствует ручная модерация и возможность ранжирования по популярности, что затрудняет поиск нужного товара.

Главная страница сайта изображена на рисунке 4.

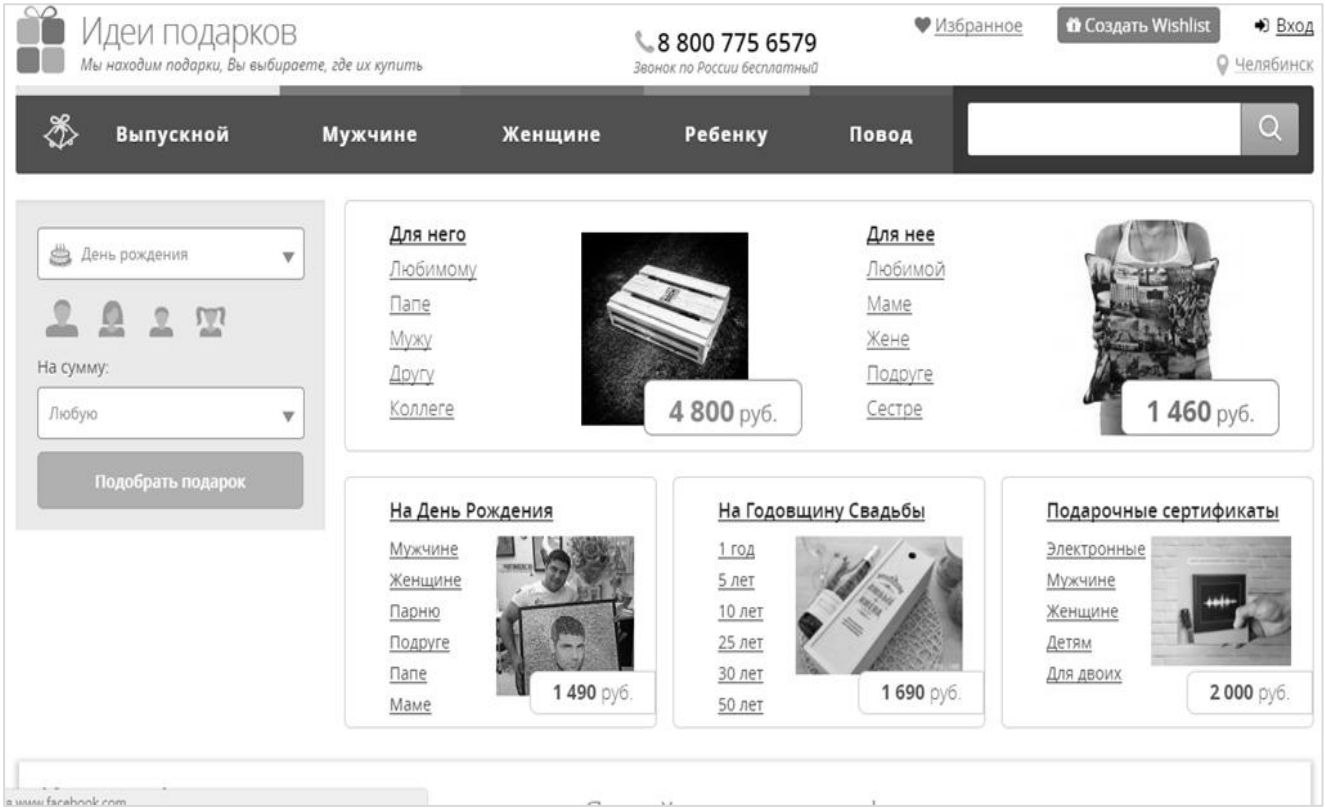

Рисунок 4 – Главная страница портала «Идеи подарков»

На главной странице портала располагаются: логотип, номер телефона, ссылка на страницу «избранное», кнопка для создания «wishlist», строка поиска, категории товаров, блок поиска подарков с возможностью задать характеристики.

Сильные стороны сервиса:

- Удобный поиск с возможностью задать критерии;
- Широкий ассортимент;
- Большое количество категорий товаров.

Слабые стороны:

- Сложно найти товар в своем городе, если это не Москва;
- Нужно переходить на сайт продавца для совершения покупки;
- Нет условий сотрудничества в публичном доступе;
- Нет мобильной версии сайта;
- Для размещения на сайте нужна своя интернет страница для товара.

1.7 Сервис «Яндекс.Маркет - подарки»

Сервис «Яндекс.Маркет - подарки» является подразделением сервиса «Яндекс.Маркет» с возможностью поиска подарков среди основного ассортимента товаров сервиса.

Получить идею для подарка помогают привычные инструменты для выбора товаров на Маркете. Можно выбрать подарок по параметрам (указать, кому и по какому поводу он предназначен и какие свойства важны: например, забавный сувенир коллеге на день рождения или романтичный подарок любимой на Новый год. А если не хочется долго выбирать, помогут короткие рецепты — например, «Приятные мелочи» или «Детские игрушки».

Многие подарки, например, коньки, можно сразу купить на Маркете, а более экзотические идеи — например, небесный фонарик, — найдутся на поиске Яндекса.

Главная страница сайта изображена на рисунке 5.

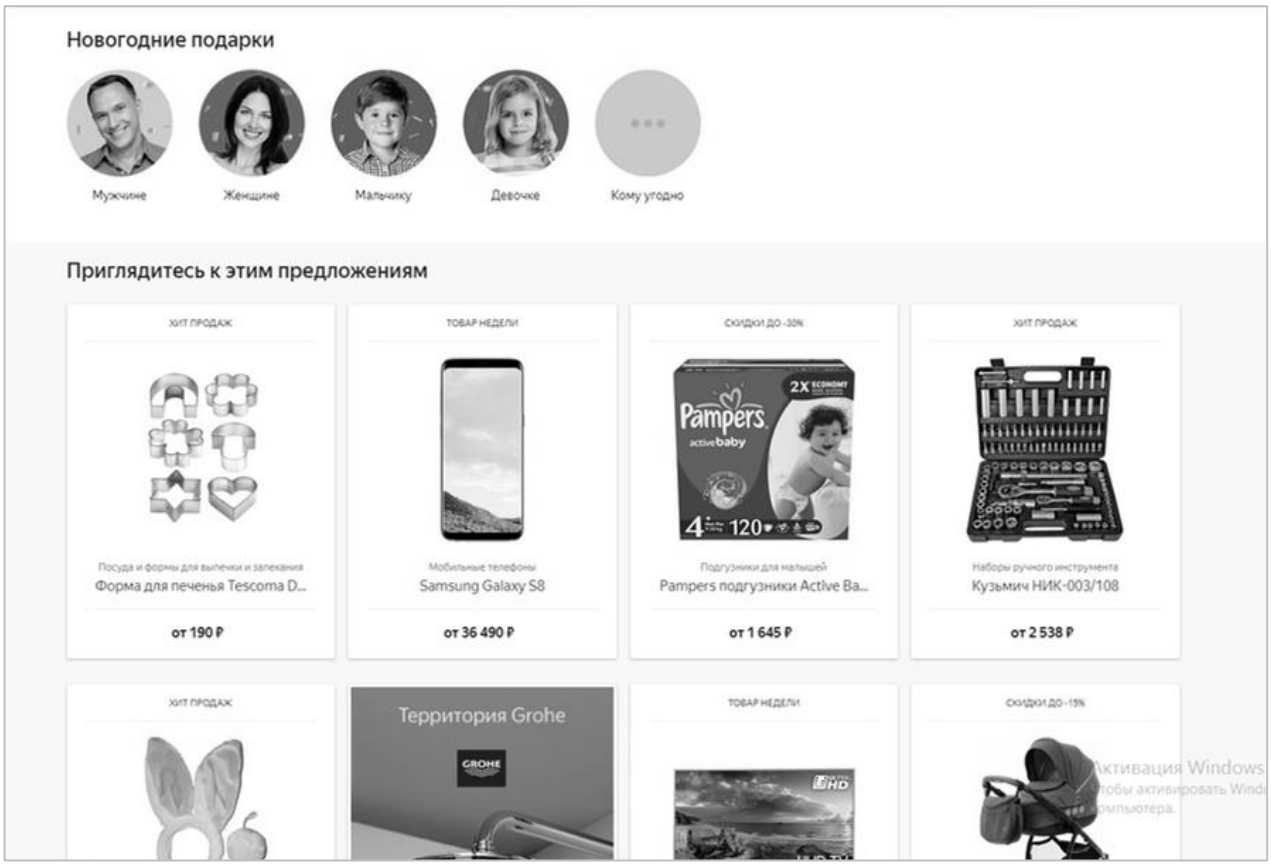

Рисунок 5 – Главная страница портала «Яндекс.Маркет - подарки»

На главной странице портала располагается список категорий, по которым можно отфильтровать подарки и блок с рекомендациями.

Сильные стороны сервиса:

- Удобный поиск с возможностью задать критерии
- Известный бренд
- Большая база отзывов о магазинах

Слабые стороны:

- Нет возможности выбрать город
- Маленький ассортимент
- Не прозрачный процесс отбора товаров, которые попадут в раздел «Подарки»

1.8 Сравнительный анализ

Проанализированы пять основных сервисов, которые смогли бы составить конкуренцию на рынке поиска подарков. Самые высокие оценки получили две торговые площадки для покупки и продажи авторских handmade-работ и дизайнерских вещей, «Ярмарка мастеров» и «Ламбада маркет». Специализированные сервисы для онлайн подбора подарков «Подарки.ру», «Яндекс подарки» и «Идеи подарков» в сравнении с рассмотренными торговым площадками имеют ряд недостатков. Наглядные различия характеристик данных порталов можно увидеть на лепестковой диаграмме, изображенной на рисунке 6.

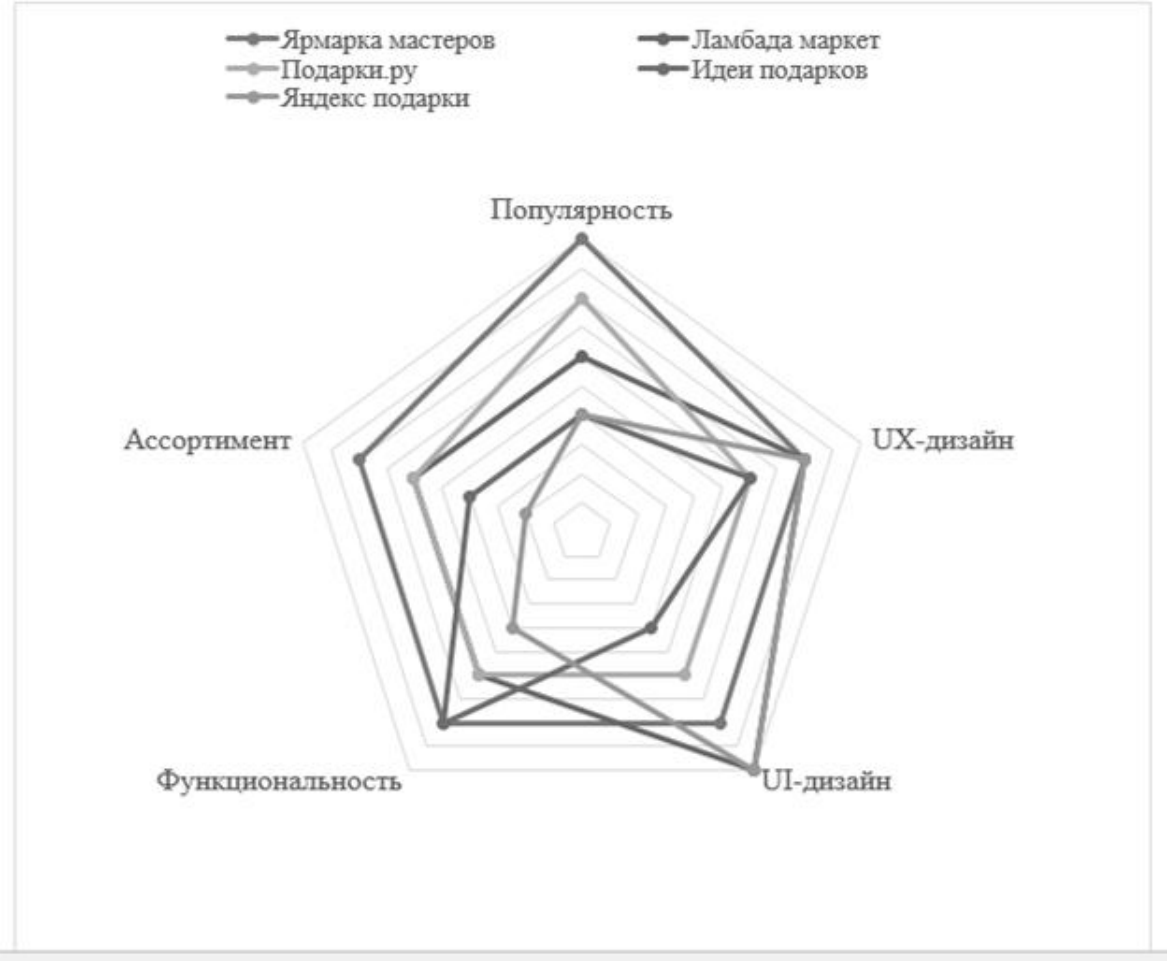

Рисунок 6 – Лепестковая диаграмма сравнения конкурентов

На рисунке показано соотношение пяти характеристик сервисов: популярность, ux-дизайн, ui-дизайн, функциональность, ассортимент.

Критерии оценивания характеристик

Оценка популярности сервисов основана на сведениях о их посещаемости.

Критерии оценки ux-дизайна:

- Эффективность – время, затрачиваемое на выполнение стандартных манипуляций;
- Наглядность – возможность безошибочно найти расположение нужных элементов интерфейса. [1]

Критерии оценки ui-дизайна:

- Ясность используемых графические символов;
- Наличие группировки элементов по логическим блокам;

- Наличие единой стилистики всех элементов сайта;
- Простота распознавания всех элементов интерфейса.

Критерием оценки функциональности сервиса является количество функций, которыми может воспользоваться пользователь.

Критерием оценки ассортимента является количество представленных товаров на сайте.

Вывод по разделу один

В первом разделе представлены технические требования к разрабатываемой системе, приведено описание аналогичных существующих систем, определены возможности этих систем.

2 ПРОЕКТИРОВАНИЕ ПРОГРАММНОГО ПРОДУКТА

2.1 Выбор программных средств

В качестве системы управления базами данных выбрана MySQL. MySQL — это самая популярная в мире база данных с открытым кодом. Благодаря своей производительности, надежности и простоте использования эта база данных чаще всего используется для веб-приложений.

Обычно MySQL используется в качестве сервера, к которому обращаются удалённые клиенты. В клиент-серверной архитектуре клиентское приложение выполняется на персональном компьютере пользователя, который в свою очередь является частью локальной сети. [5]

Для программирования на стороне сервера выбран скриптовый язык общего назначения – PHP. В настоящее время он поддерживается подавляющим большинством хостинг-провайдеров и является одним из лидеров среди языков, применяющихся для создания динамических веб-приложений. Популярность в области веб-программирования объясняется большим количеством встроенных средств и дополнительных модулей для разработки веб-приложений. К крупнейшим сайтам, использующим PHP, относятся Facebook и Wikipedia.

Чтобы структурировать и организовать сам процесс разработки, решено использовать веб-фрэймворк. Для этой задачи подошел бесплатный фреймворк с открытым исходным кодом, предназначенный для разработки с использованием архитектурной модели MVC – Laravel. Он имеет собственный обработчик шаблонов «Blade», понятный синтаксис, упрощающий выполнение рутинных операций, таких как авторизация, управление сессиями, очередями, кэшированием и маршрутизацией.

В качестве среды разработки был выбран редактор исходного кода Visual Studio Code. Он имеет возможности IDE и позиционируется как «лёгкий» редактор кода для кроссплатформенной разработки веб и облачных приложений. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса,

IntelliSense и средства для рефакторинга. Имеет широкие возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации.

Для разработки пользовательского интерфейса было решено использовать javascript, css, blade, jquery и bootstrap.

JavaScript чаще всего используется в качестве встраиваемого языка для доступа к объектам приложений. В подавляющем большинстве находит применение в браузерах как язык сценариев для создания интерактивных интерфейсов. JavaScript – это объектно-ориентированный язык, но используемое в языке прототипирование обуславливает отличия в работе с объектами в сравнение с традиционными класс-ориентированными языками. Помимо этого, JavaScript имеет свойства присущие функциональным языкам — функции как объекты первого класса, объекты как списки, анонимные функции, замыкания. Перечисленные функции придают языку дополнительную гибкость.

jQuery — кроссплатформенная библиотека JavaScript, которая содержит готовые функции языка JavaScript, созданная для упрощения клиентских сценариев.

CSS – это язык таблиц стилей. Он позволяет добавлять стилизацию к структурированным документам, таким как HTML и XML. Чаще всего CSS-стили нужны для создания и изменения стилей элементов веб-страниц и пользовательских интерфейсов. Каскадные таблицы стилей описывают правила форматирования элементов с помощью свойств и допустимых значений этих свойств.

Blade – простой и мощный шаблонизатор, входящий в состав Laravel. [4] Blade основывается на концепции наследования шаблонов и секциях. Его основная цель — это отделение представления данных от исполняемого кода. Использование шаблонbзаторов улучшает читаемость кода и упрощает внесение изменений.

2.2 Проектирование диаграммы потоков данных приложения

DFD — диаграммы потоков данных. Это методология графического структурного анализа, предназначенная для моделирования информационный систем через описание источников и адресатов данных.

Спроектированная диаграмма потоков данных приложения имеет три уровня. Первый уровень изображен на рисунке 7.

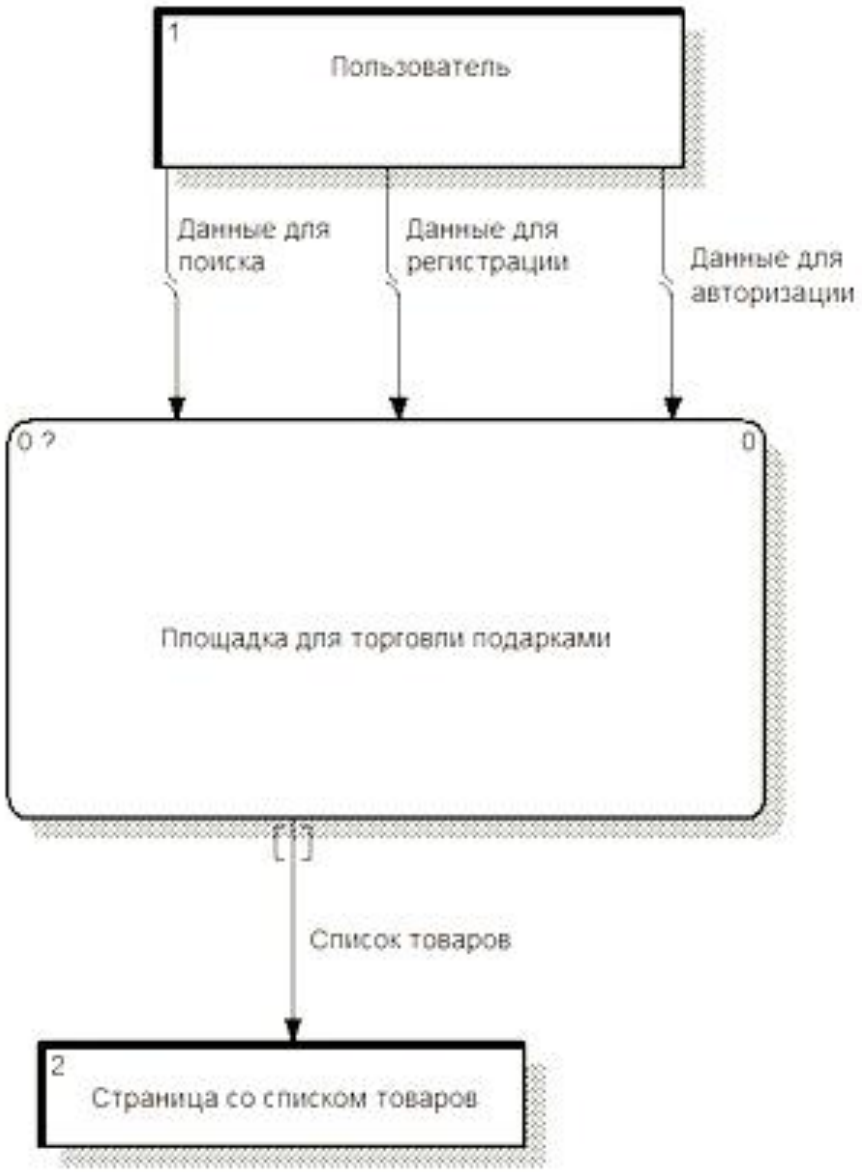

Рисунок 7 – Первый уровень DFD-модели

На рисунке 7 показано разделение данных, вводимых пользователем. Если пользователь вводит данные для поиска, то приложение возвращает ему страницу со списком товаров соответствующих его запросу. Потоки данных для регистрации и авторизации рассмотрены на втором уровне. Второй уровень изображен на рисунке 8.

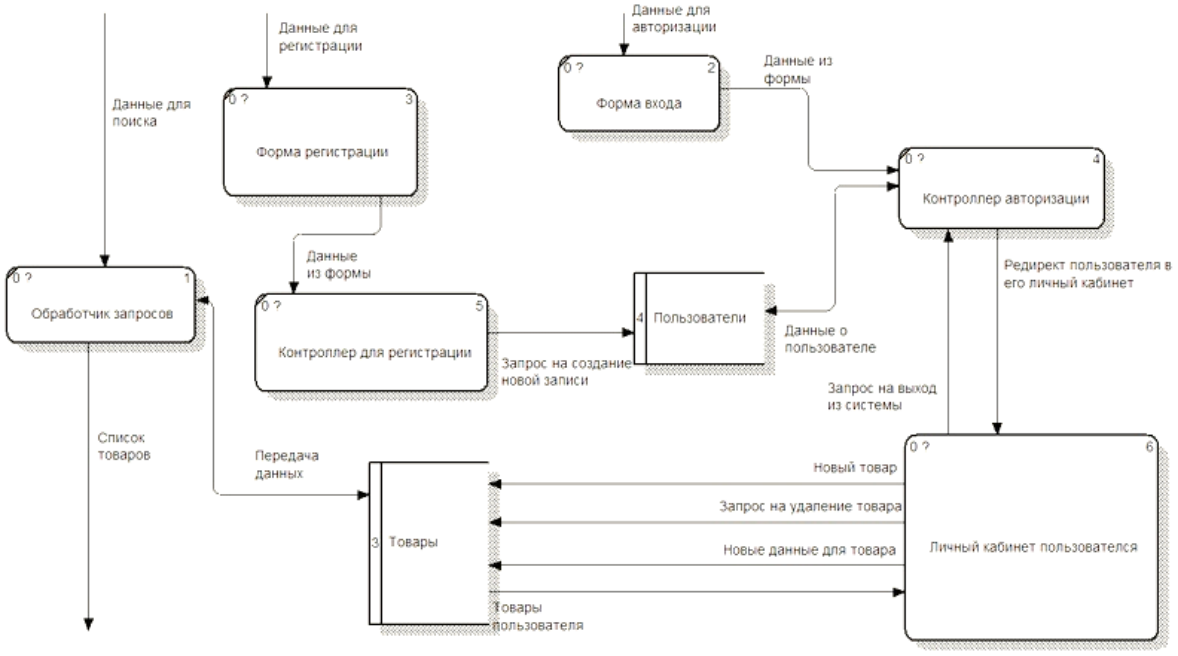

Рисунок 8 – Второй уровень DFD-модели

На рисунке 8 показано, что данные для поиска попадают в контроллер, который формирует запрос к базе данных и страницу со списком товаров. Помимо этого на рисунке изображена работа с базой данных и личным кабинетом. Данные для регистрации попадают в таблицу «Пользователи». Данные для авторизации попадают в соответствующий контроллер и сравниваются с данными в таблице «Пользователи», если данные верны, происходит перенаправление в личный кабинет. Потоки данных в личном кабинете отражены на третьем уровне, который изображен на рисунке 9.

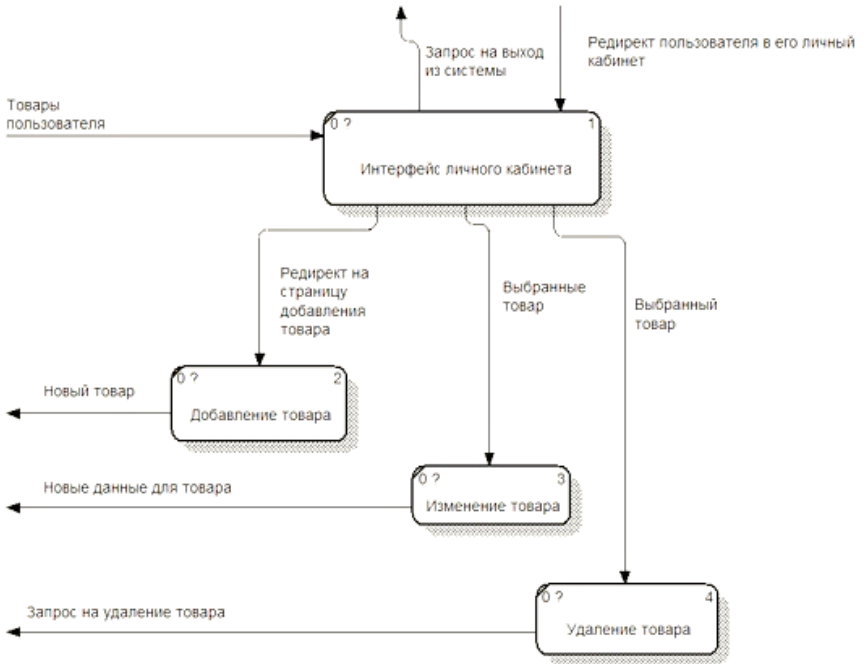

Рисунок 9 – Третий уровень DFD-модели

На рисунке 9 показана работа с базой данных из личного кабинета. Изображены процессы добавления, изменения и удаления товаров. Показан процесс выхода из авторизации.

2.3 Проектирование и применение базы данных

База данных – это система, позволяющая упорядоченно хранить данные и управлять ими.

Для хранения информации и сведений, которые требуются для web-сервиса, разработана база данных «joymarket». На ней расположены списки полей таблиц. Наименования таблиц находятся в заглавных частях списков. Так же на схеме данных имеются линии, которые соединяют списки полей. Они показывают каким образом таблицы связанны друг с другом.

Так как, для удобства работы с базой данных, была использована система объектно-реляционного отображения «EloquentORM», то названия таблиц должны соответствовать некоторым правилам наименования этой системы. В соответствии с принятым соглашением, имена таблиц должны использовать именам классов, соотносящихся с представленными таблицами, в нижнем

регистре и во множественном числе. На рисунке 6 показана схема данных базы данных «joymarket».

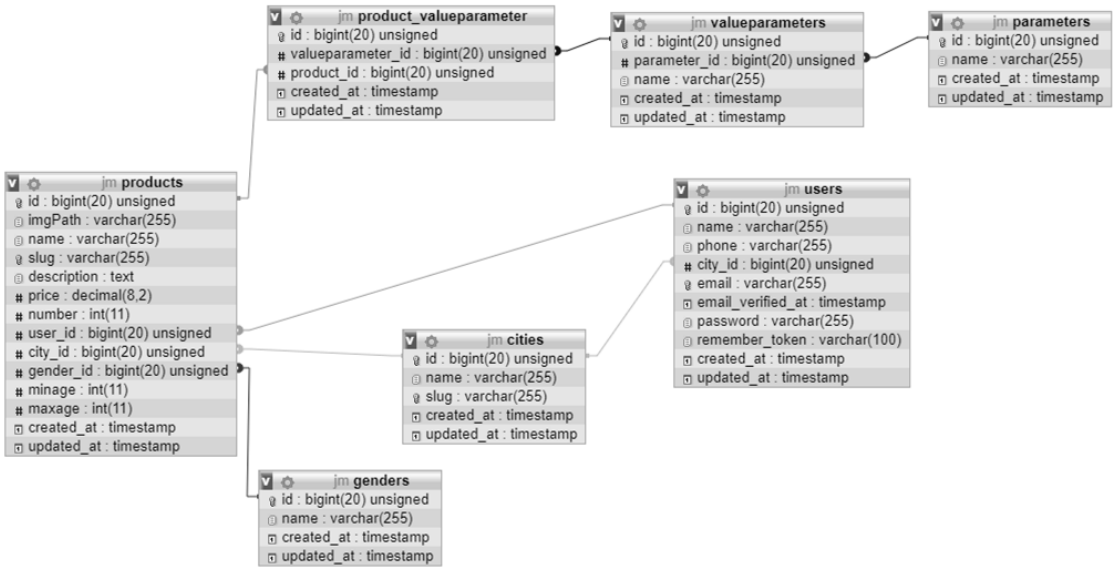

Рисунок 6 – Схема базы данных «joymarket»

На схеме базы данных «joymarket» показаны таблицы, в которых содержится информация о зарегистрированных пользователях, о городах, о товарах и их характеристиках.

2.4 Описание таблиц базы данных

В таблице «cities» хранится информация о городах. Список её полей приведен в таблице 2.

Таблица 1 – Поля таблицы «cities»

Название поля	Описание	Тип поля
id	Уникальный код города	Идентификатор
name	Название города	Текстовый
slug	Названия города после транслитерации английским алфавитом	Текстовый
created_at	Дата создания записи	Целое
updated_at	Дата последнего изменения записи	Целое

В таблице «users» содержатся данные обо всех зарегистрированных на сайте пользователях. Список её полей приведен в таблице 1.

Таблица 2 – Поля таблицы «users»

Название поля	Описание	Тип поля
id	Уникальный код пользователя	Идентификатор
name	Имя пользователя	Текстовый
phone	Номер телефона пользователя	Текстовый
city_id	Уникальный код города пользователя	Целое
email	Адрес электронной почты пользователя	Текстовый
email_verified_at	Информация о том подтвердил ли пользователь свой e-mail	Временная метка
password	Идентификатор версии алгоритма шифрования, количество итераций функции для получения ключа, конкатенированные соль и хеш.	Текстовый
remember_token	Уникальный токен пользователя для его аутентификации	Текстовый
created_at	Дата создания записи	Дата/Время
updated_at	Дата последнего изменения записи	Дата/Время

В таблице «products» хранится информация о товарах на сайте. Список её полей приведен в таблице 3.

Таблица 3 – Поля таблицы «products»

Название поля	Описание	Тип поля
id	Уникальный код товара	Идентификатор
imgPath	Путь к изображению товара	Текстовый
name	Название товара	Текстовый
slug	Названия товара после транслитерации английским алфавитом	Текстовый
description	Описание товара	Текстовый
price	Стоимость товара	Десятичное число
user_id	Уникальный код пользователя товара	Целое
city_id	Уникальный код города товара	Целое
gender_id	Уникальный код гендерной принадлежности товара	Целое
minage	Минимальный возрастное ограничение	Целое
maxage	Максимальное возрастное ограничение	Целое
created_at	Дата создания записи	Дата/Время
updated_at	Дата последнего изменения записи	Дата/Время

В таблице «parameters» хранится информация о свойствах товаров. Список её полей приведен в таблице 4.

Таблица 4 – Поля таблицы «parameters»

Название поля	Описание	Тип поля
id	Уникальный код свойства	Идентификатор
name	Наименование свойства	Текстовый
created_at	Дата создания записи	Дата/Время
updated_at	Дата последнего изменения записи	Дата/Время

В таблице «valueparameters» хранится информация о возможных значениях свойств товаров. Список её полей приведен в таблице 5.

Таблица 5 – Поля таблицы « valueparameters »

Название поля	Описание	Тип поля
id	Уникальный код значения свойства	Идентификатор
parameter_id	Уникальный код свойства	Целое
name	Наименование значения свойства	Текстовый
created_at	Дата создания записи	Дата/Время
updated_at	Дата последнего изменения записи	Дата/Время

Все поля таблицы «product_valueparameter», хранящей данные о соответствии товаров значениям их свойств, приведены в таблице 6.

Таблица 4 – Поля таблицы «product_valueparameter»

Название поля	Описание	Тип поля
id	Уникальный код записи	Идентификатор
valueparameter_id	Уникальный код значения свойства	Целое
product_id	Уникальный код товара	Целое
created_at	Дата создания записи	Дата/Время
updated_at	Дата последнего изменения записи	Дата/Время

2.5 Архитектура приложения

В качестве способа построения структуры приложения выбрана концепция MVC. MVC – это архитектурный паттерн, применяемый в web-разработке, суть которого состоит в том, что данные приложения, пользовательский интерфейс и управляющая логика разделены на три отдельных компонента: «Модель», «Представление» и «Контроллер». Преимущество данного подхода заключается в том, что таким образом, изменение каждого компонента осуществляется независимо от других. В результате, код приложения становится проще писать, масштабировать, сопровождать и тестировать, так как все модули приложения являются слабо связанными. [2]

В архитектуре MVC модель включает в себя данные и правила бизнес-логики, представление отвечает за отображение интерфейса приложения, а взаимодействие модели и представления обеспечивается контроллером. [3]

Концепция MVC в web-приложении была реализована средствами фреймворка «Laravel».

Для работы с данными использована система «Eloquent ORM». У каждой таблицы имеется соответствующий ей класс, который позволяет работать с данной таблицей. Все модели таблиц наследуют базовый класс «Model».

Для формирования web-страниц используется шаблонизатор «blade», код которого компилируются на сервере в PHP-код и отправляется клиенту. Соответствующий код программы для реализации модели предоставлен в приложении В.

Для обеспечения взаимодействия пользователя с базой данных через интерфейс используются контроллеры. Каждый написанный контроллер наследуют базовый класс «Controller». Соответствующий код программы для реализации контроллера предоставлен в приложении Г.

Для направления запроса пользователя к соответствующему контроллеру используется роутинг. Все маршруты приложения определены в файлах

маршрутов. Соответствующий код программы для реализации маршрутизации предоставлен в приложении А.

2.5 Структура и описание работы приложения

Структура web-приложения представлена на рисунке 10.

Рисунок 10 – Структура web-приложения

На рисунке 10 показана иерархическая структура приложения. На верхнем уровне располагается главная страница сайта, содержащая форму поиска, форму регистрации, форму входу, список последних добавленных товаров. На нижних уровнях показаны дочерние страницы.

Главная страница приведена на рисунке 11.

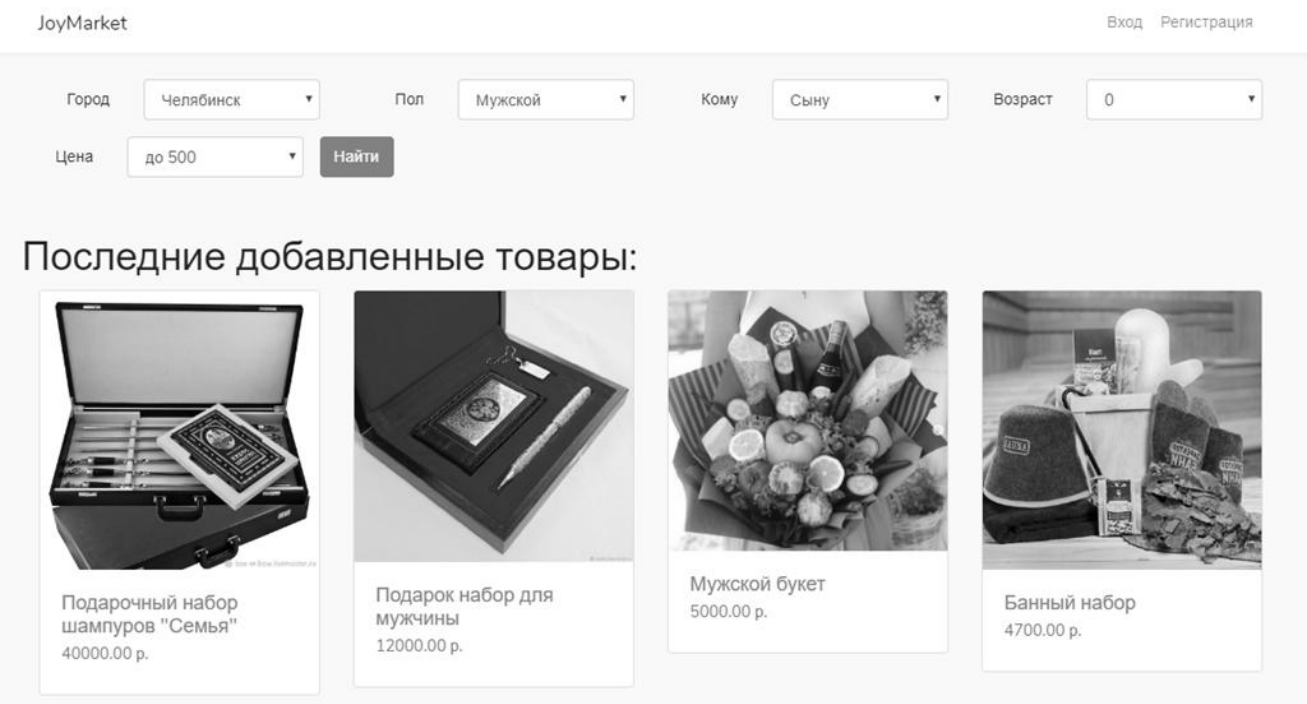

Рисунок 11 – Главная страница web-приложения

На главной странице приложения расположено несколько различных элементов:

- Ссылки на страницу регистрации и страницу входа (если пользователь авторизован, то вместо них располагаются ссылка на личный кабинет и кнопка выхода);
- Логотип, ссылающийся на главную страницу сайта;
- Параметры поиска, представленные в виде выпадающих списков;
- Кнопка поиска;
- Список последних добавленных на сайт товаров.

Для позиционирования и стилизации элементов интерфейса использован фреймворк bootstrap4 [6]. Соответствующий код программы для формирования страницы предоставлен в приложении Б.

2.5.1 Форма поиска

При нажатии на кнопку «Найти», формируется запрос к базе данных, на основе выбранных значений в выпадающих списках, и клиенту возвращается сформированная страница со списком товаров, соответствующих его запросу. Пример сформированной страницы по запросу «Мужской подарок, для брата, которому 20 лет, дороже 10000р» показан на рисунке 12.

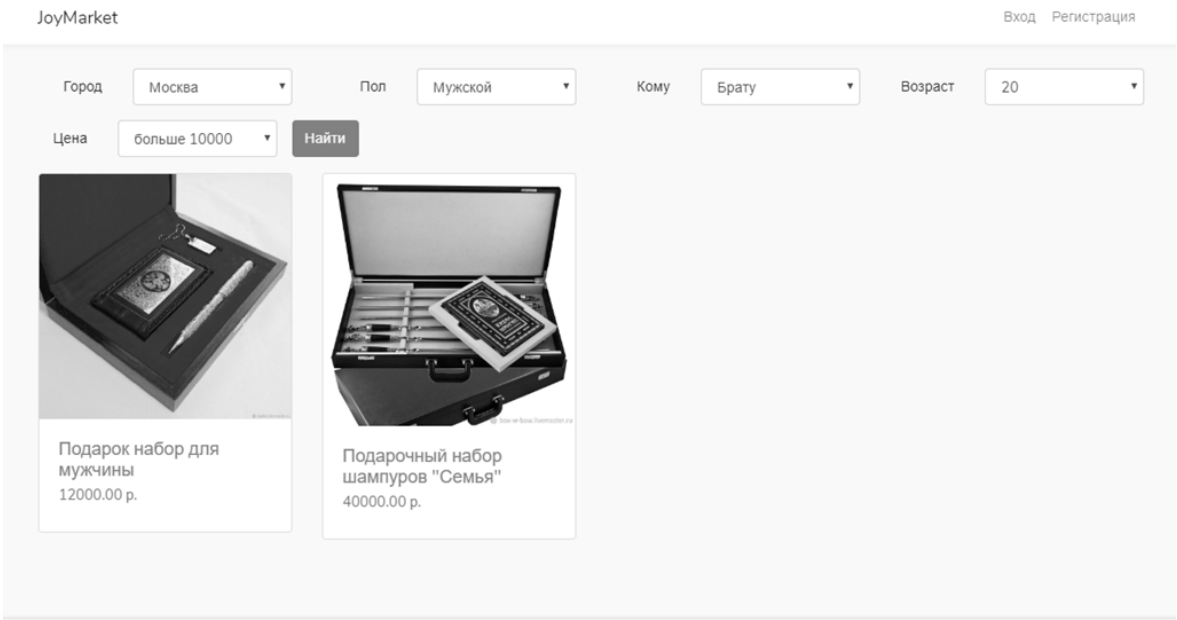

Рисунок 12 – Страница найденных товаров

На форме поиска находятся пять выпадающих списков, для выбора характеристик искомого товара:

- Список «Город», содержит названия доступных городов. Предназначен для выбора товаров по конкретному городу;
- Список «Пол», содержит значения, для фильтрации по гендерному предназначению товара. При выборе значения, в списке «Кому» отображаются только те социальные роли, которые соответствуют выбранному полу;
- Список «Кому» содержит список социальных ролей. Предназначен для выбора товаров, которые соответствуют выбранному социальному статусу;

- Список «Возраст», содержит значения возрастов. Предназначен для выбора товаров, которые соответствуют выбранному возрасту;
- Список «Цена», содержит список диапазонов цен. Предназначен для выбора товаров, попадающих в выбранный диапазон.

2.5.2 Форма регистрации

При нажатии на кнопку «Регистрация» происходит отправка введенных данных в контроллер, отвечающий за валидацию и если данные верны, то происходит создание новой записи в базе данных и перенаправление на страницу личного кабинета. Форма регистрации изображена на рисунке 13.

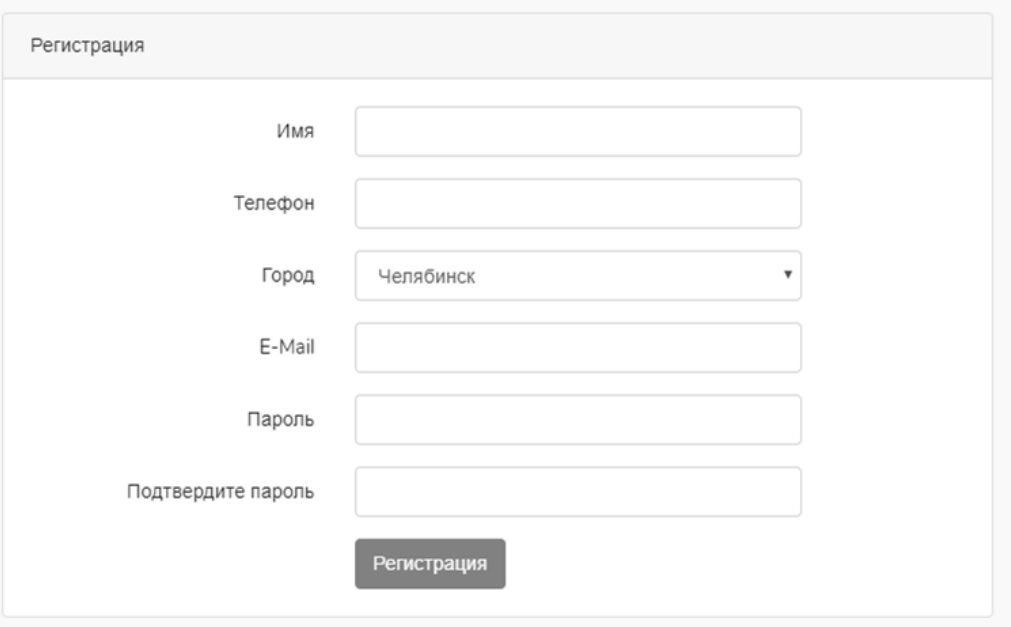

Рисунок 13 – Форма регистрации

На форме регистрации расположены выпадающий список, содержащий названия доступных городов и поля для заполнения: «имя», «телефон», «e-mail», «пароль», «подтверждение пароля».

2.5.3 Форма входа

При нажатии на кнопку «Войти» происходит поиск записи в таблице «users» по адресу электронной почты, затем происходит хеширование введенного пароля и сравнение его значения со значением в таблице. Если данные совпали,

происходит перенаправление на страницу личного кабинета, если нет, то появляется сообщение об ошибке. Форма входа изображена на рисунке 14.

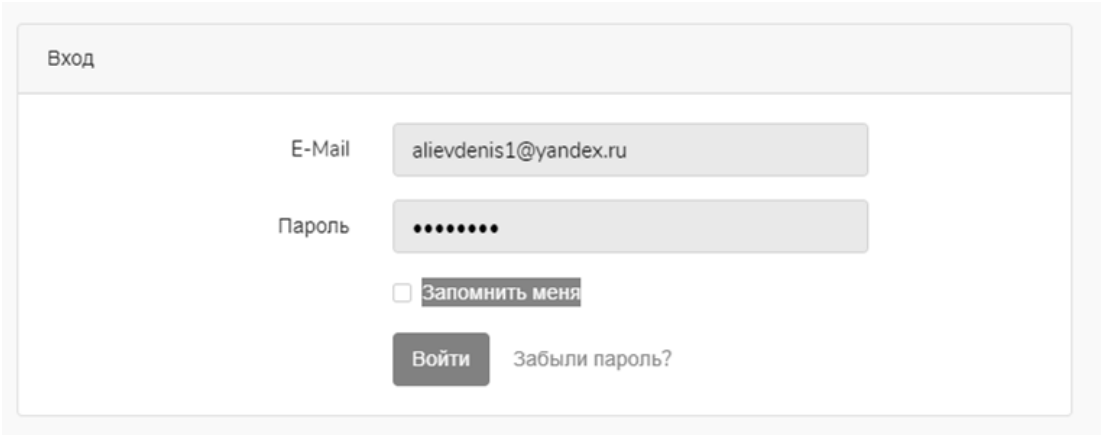

Рисунок 14 – Форма входа

На форме входа расположены чек-бокс «Запомнить меня», поля для заполнения «e-mail» и «пароль». «e-mail», «пароль», «подтверждение пароля».

2.5.4 Личный кабинет

При нажатии кнопки «изменить» открывается форма изменения товара, заполненная текущими данными о товаре. При нажатии кнопки «удалить» появляется диалоговое окно для подтверждения удаления товара. Диалоговое окно изображено на рисунке 16. Страница личного кабинета показана на рисунке 15.

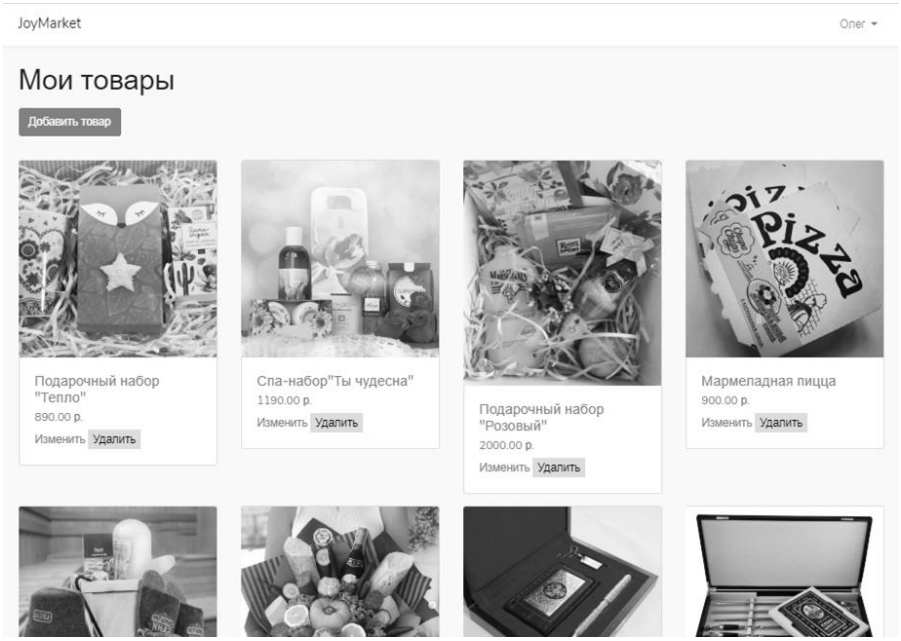

Рисунок 15 – Страница личного кабинета

На странице личного кабинета расположены кнопка «Добавить товар», при нажатии на которую открывается форма добавления товара, список товаров пользователя с кнопками «изменить» и «удалить».

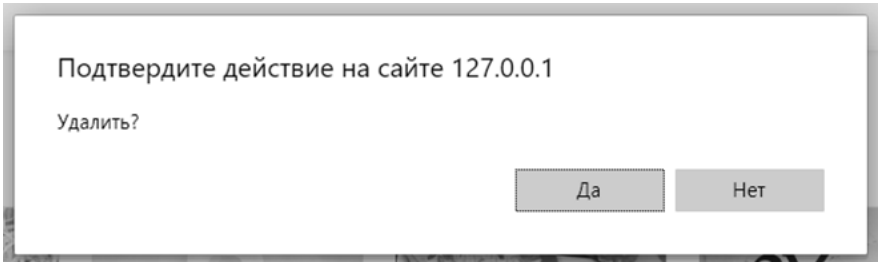

Рисунок 16 – Диалоговое окно подтверждения удаления

На диалоговом окне расположен текст «Удалить?» и кнопки «Да» и «Нет».

2.5.5 Форма добавления товара

При нажатии кнопки «выбрать файл», открывается окно для загрузки изображения товара.

При выборе значения в списке «пол», происходит обновление списка чек-боксов с соответствующими полу социальными ролями.

При нажатии кнопки «добавить» происходит проверка введенных данных и если данные введены верно, то отправляется запрос на создание новой записи в таблице «products» с введенными данными, иначе возвращается сообщение со списком неверно заполненных полей.

Форма добавления товара показана на рисунке 17.

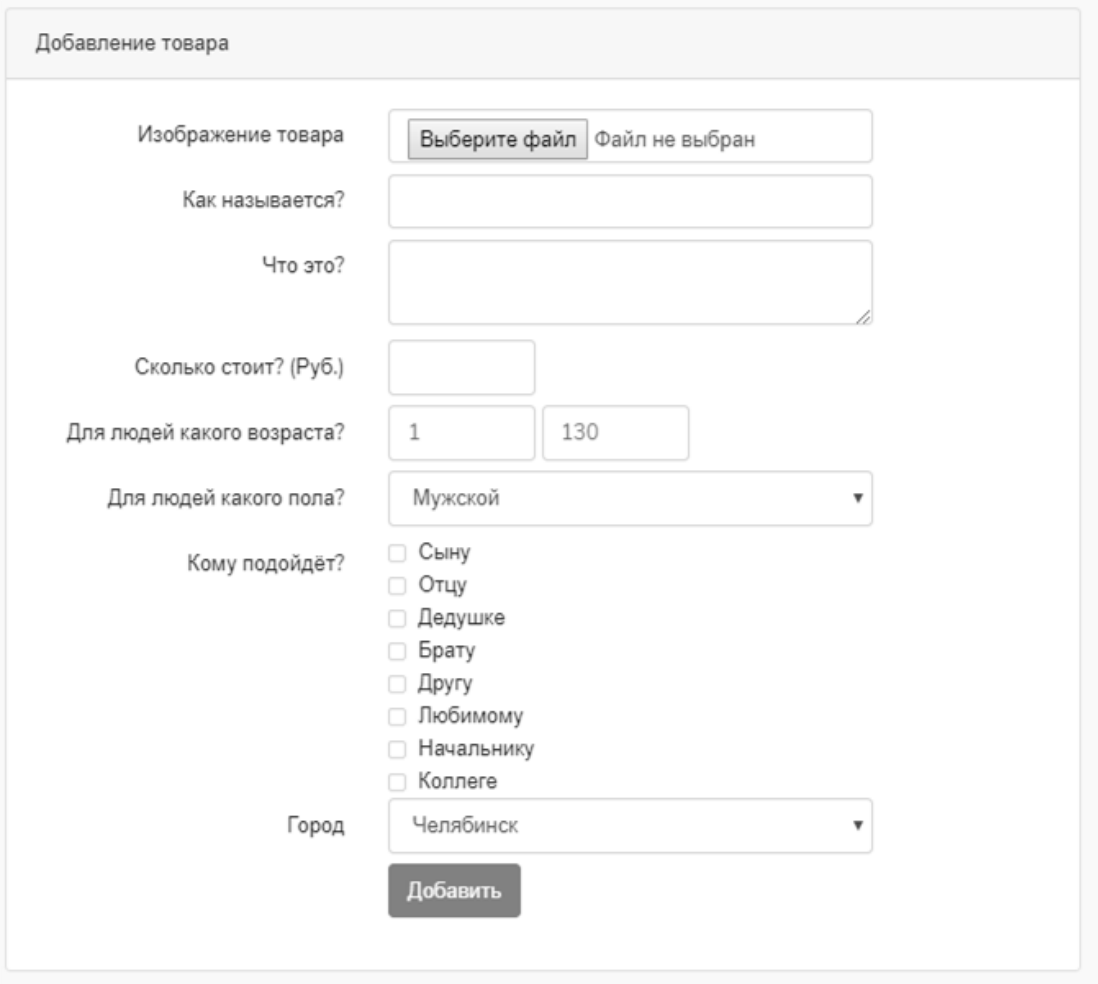

Рисунок 17 – форма добавления товара

На форме добавления товара расположены кнопка «выбрать файл», выпадающие списки «Пол», «Город», поля «название», «описание», «стоимость», «минимальный возраст», «максимальный возраст», кнопка «добавить».

2.5.6 Форма изменения товара

На форме изменения товара расположены те же элементы, что и на форме добавления товара, но уже с заполненными данными выбранного товара.

2.6 Хранение и шифрование пароля пользователя

Для обеспечения безопасности пароль пользователя хранится в зашифрованном виде. Для хеширования пароля используется функция «bcrypt». Использование «bcrypt» является общепринятым и одним из лучших способов для хеширования паролей. Это адаптивная криптографическая хеш-функция формирования ключа, основанная на криптографическом алгоритме,

реализующем блочное симметричное шифрование, который называется «Blowfish».

Для защиты от атак с помощью радужных таблиц bcrypt использует соль. Функция принимает пароль в формате строки и возвращает строку длинной 60 символов, содержащую версию алгоритма, стоимость вычислений, конкатенированные соль и хеш, закодированные через base64. Первые 22 символа — 16 байт соли. Оставшееся — хеш. Пример такой строки изображен на рисунке 18.

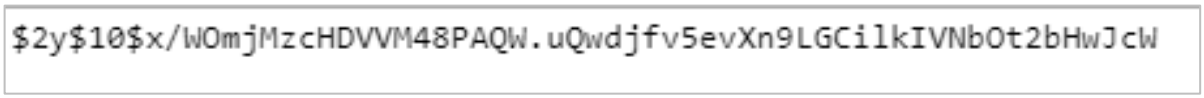

Рисунок 18 – пароль пользователя в зашифрованном виде.

Вывод по разделу два

В данном разделе приведены выбранные инструменты разработки. Описаны таблицы базы данных «joymarket», используемые при разработке web-приложения. Разработанна архитектура приложения и методы, использованные при её создании. Показана созданная диаграмма потоков данных. Рассмотрен способ защиты аккаунтов пользователей с помощью хеширования данных. В разделе отображена структура разработанного приложения и описание работы с сервисом.

ЗАКЛЮЧЕНИЕ

Цель данного дипломного проекта – разработка платформы для онлайн торговли подарками.

Для решения поставленной задачи использовались следующие программные инструменты –Visual Studio Code, MySQL, HTML, CSS, JavaScript, Jquery, PHP, Laravel.

В процессе выполнения дипломного проекта решены задачи:

- Анализ существующих решений;
- Разработка архитектуры приложения;
- Проектировка база данных;
- Выбор инструментов для разработки системы;
- Проектировка интерфейса;
- Написание кода приложения;
- Отладка приложения.

Итогом данного дипломного проекта является готовое интернет приложение для онлайн торговли подарками. Таким образом, поставленные задачи выполнены, а цель дипломного проекта достигнута.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Якоб Нильсен, Web-дизайн. Удобство использования Web-сайтов 2007.

2. Мэтт Зандстра PHP: объекты, шаблоны и методики программирования 4-е издание.

3. Изучаем PHP 7: руководство по созданию интерактивных веб-сайтов. : Пер. с англ. — СпБ. : ООО «Альфа-книга», 2017. — 464 с.

4. Installation - Laravel - The PHP Framework For Web Artisans – https://laravel.com/docs/5.8

5. Ржеуцкая С.Ю. - Базы данных. Язык SQL

6. Documentation for bootstrap4 – https://getbootstrap.com/docs/4.3/getting-started/introduction/

ПРИЛОЖЕНИЕ А

Фрагмент кода файла маршрутов.

```
        Auth::routes();
        Route::get('/', 'PagesController@index')->name('home');
        Route::get('/myproducts', 'ProductsController@myProducts')->name('account')-
>middleware('auth');
        Route::post('{city}/products', 'ProductsController@viewsProducts')->name('Products');

        Route::get('{city}/products/{id}', 'ProductsController@viewProduct')->name('Product');
        Route::delete('{city}/products/{id}', 'ProductsController@deleteProduct')-
>name('deleteProduct');
        Route::get('{city}/products/{id}/edit', 'ProductsController@editProduct')-
>name('editProduct');
        Route::post('{city}/products/{id}/edit', 'ProductsController@putEditProduct')-
>name('putEditProduct');
        Route::get('/addproduct', 'ProductsController@addProduct')->name('addProduct')-
>middleware('auth');
        Route::post('/addproduct', 'ProductsController@postAddProduct')-
>name('postAddProduct')->middleware('auth');

        Route::get('/searchProduct', 'ProductsController@viewSerchProducts')-
>name('searchProduct');
```

ПРИЛОЖЕНИЕ Б

Фрагмент кода базового шаблона

```
<!DOCTYPE html>
<html lang="{{ str_replace('_', '-', app()->getLocale()) }}">
<head>
    <meta charset="utf-8">
    <meta name="viewport" content="width=device-width, initial-scale=1">
    <!-- CSRF Token -->
    <meta name="csrf-token" content="{{ csrf_token() }}">
    <title>{{ config('app.name', 'jm') }}</title>
    <!-- Scripts -->
    <script src="{{ asset('js/app.js') }}" defer></script>
    <script src="{{ asset('js/common.js') }}" defer></script>
    <!-- Fonts -->
    <link rel="dns-prefetch" href="//fonts.gstatic.com">
    <link href="https://fonts.googleapis.com/css?family=Nunito" rel="stylesheet"
type="text/css">
    <!-- Styles -->
    <link href="{{ asset('css/app.css') }}" rel="stylesheet">
    <link href="{{ asset('css/common.css') }}" rel="stylesheet">
</head>
<body>
    <header id="app">
        <nav class="navbar navbar-expand-md navbar-light navbar-laravel">
            <div class="container">
                <a class="navbar-brand" href="{{ url('/') }}">
                    {{ config('app.name', 'JoyMarket') }}
                </a>
                <button class="navbar-toggler" type="button" data-toggle="collapse" data-
target="#navbarSupportedContent" aria-controls="navbarSupportedContent" aria-expanded="false" aria-
label="{{ __('Toggle navigation') }}">
                    <span class="navbar-toggler-icon"></span>
                </button>
                <div class="collapse navbar-collapse" id="navbarSupportedContent">
                    <!-- Left Side Of Navbar -->
                    <ul class="navbar-nav mr-auto">
                    </ul>
                    <!-- Right Side Of Navbar -->
                    <ul class="navbar-nav ml-auto">
                       <!-- Authentication Links -->
                        @guest
                            <li class="nav-item">
                                <a class="nav-link" href="{{ route('login') }}">{{
__('Вход') }}</a>
                            </li>
                            @if (Route::has('register'))
                                <li class="nav-item">
```

```
          <a class="nav-link" href="{{ route('register') }}">{{ __('Регистрация') }}</a>
                                    </li>
                                @endif
                            @else
                                <li class="nav-item dropdown">
                                    <a id="navbarDropdown" class="nav-link dropdown-toggle"
href="#" role="button" data-toggle="dropdown" aria-haspopup="true" aria-expanded="false" v-pre>
                                        {{ Auth::user()->name }} <span class="caret"></span>
                                    </a>
                                    <div class="dropdown-menu dropdown-menu-right" aria-
labelledby="navbarDropdown">                                    <a class="dropdown-item" href="{{
route('account') }}">
                                         {{ __('Мои товары') }}
                                    </a>
                                    <a class="dropdown-item" href="{{ route('logout') }}"
                                       onclick="event.preventDefault();
                                                    document.getElementById('logout-
form').submit();">
                                        {{ __('Выход') }}
                                    </a>

                                    <form id="logout-form" action="{{ route('logout') }}"
method="POST" style="display: none;">
                                        @csrf
                                    </form>
                                </div>
                                </li>
                            @endguest
                        </ul>
                    </div>
                </div>
            </nav>
        </header>
            <main class="py-4 main-content">
                    @yield('content')
                </main>
            <footer class="footer navbar">
                    <div class="container">
                        <div class="row">
                            © 2019 Денис Алиев
                        </div>
                    </div>
            </footer>
        </body>
        </html>
```

ПРИЛОЖЕНИЕ В

Фрагмент кода модели «Product»

```
class Product extends Model
{
  use Sluggable;

    public function user()
    {
      return $this->belongsTo('App\User');
    }

    public function valueparameters()
    {
      return $this->belongsToMany('App\Valueparameter');
    }

    public function city()
    {
      return $this->belongsTo('App\City');
    }

    public function gender()
    {
      return $this->belongsTo('App\Gender');
    }

    protected $fillable = [
      'name', 'description', 'number', 'price', 'user_id', 'city_id', 'minage', 'maxage',
'gender_id','imgPath'
    ];

    public function sluggable()
    {
        return [
            'slug' => [
                'source' => 'name'
            ]
        ];
    }

}
```

ПРИЛОЖЕНИЕ Г

Фрагмент кода контроллера «ProductsController»

```
class ProductsController extends Controller
{
    public function viewSerchProducts(Request $request)
    {
        $cities = City::all();
        $genders = Gender::all();
        $rolesWoman = Parameter::find(2)->valueparameters()->get();
        $rolesMan = Parameter::find(1)->valueparameters()->get();
        $roles = Parameter::find(3)->valueparameters()->get();


       $products = Valueparameter::find($request['role'])->products()-
>where('city_id', $request['city'])->whereIn('gender_id', array($request['gender'],
3))->where('minage', '<=' , (int)$request['age'])->where('maxage', '>=' ,
(int)$request['age']);

        if((int)$request['price']==0){
         $products = $products->where('price', '<=' ,500);
        }else if((int)$request['price']==1){
         $products = $products->where('price', '>=' ,500)->where('price', '<='
,1000);
        }else if((int)$request['price']==2){
         $products = $products->where('price', '>=' ,1000)->where('price', '<='
,5000);
         }else if((int)$request['price']==3){
             $products = $products->where('price', '>=' ,5000)->where('price',
'<=' ,10000);
         }else if((int)$request['price']==4){
             $products = $products->where('price', '>=' ,10000);
         };

         $products =  $products->get();

         return view('searchproducts', compact('products', 'cities', 'genders',
'rolesWoman', 'rolesMan', 'roles'));
    }

    public function viewProduct($city, $productName)
    {
        $product = Product::whereSlug($productName)->firstOrFail();

        return view('product',  compact('product'));
    }
```

```
        public function deleteProduct($city, $productName)
        {
            $product = Product::whereSlug($productName)->firstOrFail();
            if(Auth::check() &&  Auth::user()->id == Product::where('slug',
$productName)->firstOrFail()->user_id){
                $product->delete();
            }
            else{
                return redirect()->route('account');
            };

            return redirect()->route('account');
        }


        public function editProduct($cityName, $productName)
        {

            $cities = City::all();
            $genders = Gender::all();
            $rolesWoman = Parameter::find(2)->valueparameters()->get();
            $rolesMan = Parameter::find(1)->valueparameters()->get();
            $roles = Parameter::find(3)->valueparameters()->get();

            $productSelect = Product::whereSlug($productName)->firstOrFail();


            return view('editproduct', compact('cities', 'genders', 'rolesWoman',
'rolesMan', 'roles', 'productSelect' ));
        }



        public function myProducts()
        {
            $products = Auth::user()->products()->get();
            $cities = City::all();
            return  view('account', compact('products', 'cities'));
        }
        public function addProduct()
        {
            $cities = City::all();
            $genders = Gender::all();
            $rolesWoman = Parameter::find(2)->valueparameters()->get();
            $rolesMan = Parameter::find(1)->valueparameters()->get();
```



```
        $roles = Parameter::find(3)->valueparameters()->get();
```

```
            return view('addproduct', compact('cities', 'genders', 'rolesWoman',
'rolesMan', 'roles' ));
        }

        public function postAddProduct(Request $request)
        {
            //загрузка изображения
            $imgPath = $request->image->store('upload', 'public');

            //получаем id всех ролей
            $rolesId = array();
            if( $request['gender'] == 1 ) {
                foreach (Parameter::find(1)->valueparameters()->select('id')-
>get() as $value){
                    array_push($rolesId,  $value->id);
                };

                foreach (Parameter::find(3)->valueparameters()->select('id')-
>get() as $value){
                    array_push($rolesId,  $value->id);
                };

            };
            if( $request['gender'] == 2 ) {
                foreach (Parameter::find(2)->valueparameters()->select('id')-
>get() as $value){
                    array_push($rolesId,  $value->id);
                };

                foreach (Parameter::find(3)->valueparameters()->select('id')-
>get() as $value){
                    array_push($rolesId,  $value->id);
                };
            };

            if( $request['gender'] == 3 ) {
                foreach (Parameter::find(1)->valueparameters()->select('id')-
>get() as $value){
                    array_push($rolesId,  $value->id);
                };

                foreach (Parameter::find(2)->valueparameters()->select('id')-
>get() as $value){
                    array_push($rolesId,  $value->id);
                };
```

```
       foreach (Parameter::find(3)->valueparameters()->select('id')->get() as
$value){
                        array_push($rolesId,  $value->id);
                    };
                };


                //Проходимся по всем ролям и если роль выбрана пользователем добавляем
в массив для связывания с товаром
                $valueparameters = array();
                foreach ($rolesId as $x){
                    if( $request['role' . $x] ) array_push($valueparameters,
Valueparameter::find($x));
                };



                Validator::make($request->all(), [
                    "image" => ['required', 'image']
                ])->validate();

                //Создаём товар и связываем с выбранными ролями
                $product = Product::create([
                    'imgPath' => $imgPath,
                    'name' => $request['name'],
                    'description' => $request['description'],
                    'number' => 1,
                    'price' => $request['price'],
                    'user_id' => Auth::user()->id,
                    'city_id' => $request['city'],
                    'gender_id' => $request['gender'],
                    'minage' => $request['age'],
                    'maxage' => $request['age2']
                ])->valueparameters()->saveMany($valueparameters);

                return redirect()->route('account');
            }


            public function putEditProduct(request $request, $city, $productName)
            {

                if(Auth::check() && Auth::user()->id == Product::where('slug',
$productName)->firstOrFail()->user_id){
                //загрузка изображения
                if( $request->image != null){
```

```
        $imgPath = $request->image->store('upload', 'public');
                }
           else{
                    $imgPath = Product::where('slug', $productName)->firstOrFail()-
>imgPath;
                }
                //получаем id всех ролей
                $rolesId = array();
                if( $request['gender'] == 1 ) {
                    foreach (Parameter::find(1)->valueparameters()->select('id')-
>get() as $value){
                        array_push($rolesId,  $value->id);
                    };
                    foreach (Parameter::find(3)->valueparameters()->select('id')-
>get() as $value){
                        array_push($rolesId,  $value->id);
                    };

                };
                if( $request['gender'] == 2 ) {
                    foreach (Parameter::find(2)->valueparameters()->select('id')-
>get() as $value){
                        array_push($rolesId,  $value->id);
                    };

                    foreach (Parameter::find(3)->valueparameters()->select('id')-
>get() as $value){
                        array_push($rolesId,  $value->id);
                    };
                };
                if( $request['gender'] == 3 ) {
                    foreach (Parameter::find(1)->valueparameters()->select('id')-
>get() as $value){
                        array_push($rolesId,  $value->id);
                    };
                    foreach (Parameter::find(2)->valueparameters()->select('id')-
>get() as $value){
                        array_push($rolesId,  $value->id);
                    };

                    foreach (Parameter::find(3)->valueparameters()->select('id')-
>get() as $value){
                        array_push($rolesId,  $value->id);
                    };
                };
```

```
                //Проходимся по всем ролям и если роль выбрана пользователем добавляем
в массив для связывания с товаром
                $valueparameters = array();
                foreach ($rolesId as $x){
                    if( $request['role' . $x] ) array_push($valueparameters,
Valueparameter::find($x));
                };

                Validator::make($request->all(), [
                    "image" => ['image']
                ])->validate();


                //Создаём товар и связываем с выбранными ролями
                $product = Product::where('slug', $productName)->update([
                    'imgPath' => $imgPath,
                    'name' => $request['name'],
                    'description' => $request['description'],
                    'number' => 1,
                    'price' => $request['price'],
                    'user_id' => Auth::user()->id,
                    'city_id' => $request['city'],
                    'gender_id' => $request['gender'],
                    'minage' => $request['age'],
                    'maxage' => $request['age2']
                ]);

                Product::where('slug', $productName)->firstOrFail()-
>valueparameters()->detach();
                Product::where('slug', $productName)->firstOrFail()-
>valueparameters()->saveMany($valueparameters);

                return redirect()->route('account');

                }
                else{
                    return redirect()->route('account');
                };

            }


        }
```