

Министерство образования и науки Российской Федерации
ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
“САНКТ-ПЕТЕРБУРГСКИЙ НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ
УНИВЕРСИТЕТ ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ,
МЕХАНИКИ И ОПТИКИ”

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА
ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Проектирование и реализация настраиваемого веб-приложения

Автор Елизаров Никита Михайлович _____
(Фамилия, Имя, Отчество) (Подпись)

Направление подготовки (специальность) 210700 (11.03.02)
Инфокоммуникационные технологии и системы связи

Квалификация бакалавр _____
(бакалавр, инженер, магистр)

Руководитель Зудилова Т.В., к.т.н. _____
(Фамилия, И., О., ученое звание, степень) (Подпись)

К защите допустить

Зав. кафедрой Зудилова Т.В., к.т.н. _____
(Фамилия, И., О., ученое звание, степень) (Подпись)

“ ” _____ 20 ____ г.

Санкт-Петербург, 2015 г.

Студент Елизаров Никита Михайлович Группа 4958 Кафедра ПС Факультет ИКТ
(ФИО)

Направленность (профиль), специализация 210700.62 Интеллектуальные
инфокоммуникационные системы

Консультант(ы):

а) _____
(Фамилия, И., О., ученое звание, степень) (Подпись)

б) _____
(Фамилия, И., О., ученое звание, степень) (Подпись)

Квалификационная работа выполнена с оценкой _____

Дата защиты “__” __июня__ 2015г.

Секретарь ГЭК _____

Листов хранения _____

Демонстрационных материалов/Чертежей хранения _____

ОГЛАВЛЕНИЕ

Введение.....	4
1. Основы построения современных веб-приложений	6
1.1 Общие сведения работы веб-приложений	6
1.2 Серверные технологии	7
1.3 Клиентские технологии.....	8
1.4 Использование CMS	9
1.5 Выбор подхода к разработке современного веб-приложения и постановка задачи	15
2. Проектирование веб-приложения	17
2.1 Определение требований и целевой аудитории	17
2.2 Исследование конкурентов и поиск идей	22
2.3 Создание карты веб-приложения	26
2.4 Прототипирование	27
2.5 Разработка дизайна	31
3. Реализация веб-приложения	35
1.1 Серверная часть.....	35
1.2 Клиентская часть.....	39
Заключение	41
Список источников	42

ВВЕДЕНИЕ

За более чем двадцатилетнюю историю развития интернета изменился образ жизни большинства людей. Если каких-то пятнадцать лет назад интернет был далеко не в каждом доме, то сейчас у человека появилась возможность работать с глобальной сетью во многих частях земли посредством беспроводных технологий. Практически все вопросы и проблемы человек стал решать с помощью интернета, в котором одни люди предоставляют информацию для других. По сути глобальная сеть стала еще одной формой взаимодействия людей, что бесспорно делает жизнь удобнее.

С появлением интернета начали свое развитие и веб-приложения. Если изначально это были только гипертекстовые документы формата HTML, связанные между собой гиперссылками, то сейчас веб-приложениями называются сложные программные продукты, имеющие удобный для взаимодействия интерфейс. Записаться на прием в больницу, купить авиабилет или забронировать место в кинотеатре - теперь все это можно сделать без особого труда через интернет-сервисы, что позволяет людям сохранить их время и деньги.

Рост популярности интернета и веб-приложений заставляет предпринимателей не отставать от современных технологий. Теперь, если у вашего магазина или, например, ресторана нету собственного сайта, вы теряете огромную часть клиентов и становитесь неконкурентоспособным. Но не каждая компания, особенно на начальном этапе развития способна позволить себе разработку собственного веб-приложения, да и не каждой организации требуется веб-сервис с индивидуальным подходом.

Выходом в данной ситуации может выступать типовое интернет-решение. По своей сути это готовое веб-приложение реализующее необходимый функционал в рамках определенной тематики. Заказчику

остается только установить данный программный продукт, настроить его и начать пользоваться.

Актуальность подобных интернет-решений подтверждает и тот факт, что многие веб-приложения имеют схожий и как правило уже проработанный функционал. Стоит отметить, что основными преимуществами использования типового решения являются: сокращение времени запуска проекта и уменьшение затрат.

Цель данной работы заключается в создании современного готового веб-приложения с возможностью настраивания администратором определенной функциональности.

Данная работа разделена на три главы, каждая из которых раскрывает информацию, связанную определенным этапом создания веб-приложения.

В первой главе данной работы рассматриваются общие сведения о работе веб-приложений, а также принципы работы клиент-серверной архитектуры. Приводится обзор современных клиентских и серверных технологий, раскрывается понятие CMS.

Вторая глава посвящена описанию этапа проектирования веб-приложения начиная с определения основных цели проекта и требований, заканчивая готовым дизайном веб-приложения.

В третьей главе описывается этап реализации веб-приложения. Производится краткий обзор архитектуры Bitrix Framework с помощью которого велась разработка серверной части веб-приложения. Отдельно рассмотрен подход к разработке клиентской части.

1. ОСНОВЫ ПОСТРОЕНИЯ СОВРЕМЕННЫХ ВЕБ-ПРИЛОЖЕНИЙ

1.1 Общие сведения работы веб-приложений

Существуют и применяются различные виды технологий для создания современных веб-приложений. Несмотря на это все веб-приложения имеют общую логику работы:

1. Запрос. Клиент, используя веб - браузер, инициирует запрос к серверу.
2. Обработка запроса, подготовка ответа. После получения запроса веб – сервер проводит обработку запрашиваемого ресурса. В случае, если запрашивается статический ресурс, такой как HTML страница, рисунок, документ, эта информация форматируется для протокола HTTP и передается клиенту в качестве ответа. Если же запрашивается динамический ресурс, запрос передается на обработку соответствующему контейнеру веб - приложений, где и происходит дальнейшая работа.
3. После формирования, данные передаются клиенту посредством протокола HTTP в качестве ответа. Ответ содержит данные (обычно HTML код, либо двоичные данные), а также дополнительные параметры, передаваемые в заголовках HTTP ответа.

Ниже на рисунке 1 представлена упрощенная схема работы веб-приложения приложения, основными элементами которой являются клиент и сервер.

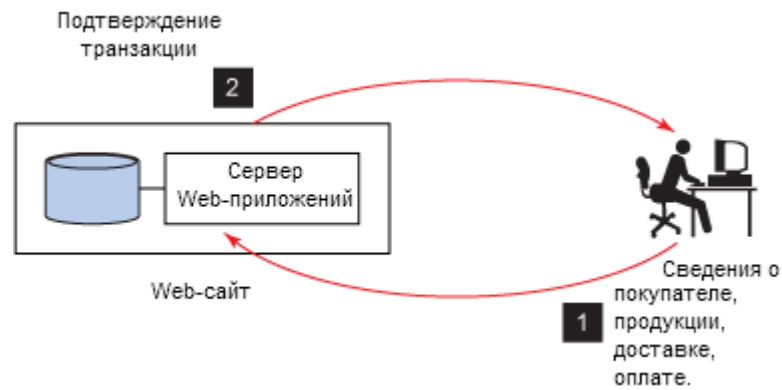


Рисунок 1.1— Схема работы веб-приложения

Веб-приложения в общем случае являются клиент-серверными приложениями, поэтому процесс разработки принято делить на два этапа: разработка клиентской части и разработка серверной части. Ниже рассмотрены основные серверные и клиентские технологии.

1.2 Серверные технологии

Под серверными технологиями подразумевают специальные программы, которые выполняются под руководством Веб-сервера и заняты обработкой запросов Веб-браузера. Существуют следующие серверные технологии: PHP, ASP.NET, Ruby on Rails, JSP, Java Servlets, Node.JS. Ниже описаны технологии, которые были использованы при разработке серверной части веб-приложения.

PHP - это язык обработки гипертекста (HTML), используемый на стороне сервера (server side scripting language), конструкции которого вставляются в HTML-текст. В основе своей имеет синтаксис очень похожий на синтаксис C, Java и Perl, однако проще этих языков. Имеет открытый исходный код.

PHP - это кроссплатформенная технология, дистрибутив которой доступен для большинства операционных систем: Unix, Microsoft Windows, Mac OS X, RISC OS, и многих других.

Во время запроса документа, имеющего PHP сценарии, на сервере происходит выполнение кода, а пользователь получает в браузер "чистый" HTML. Таким образом, PHP сценарии решают все те задачи, которые характерны для типичных CGI-приложений.

1.3 Клиентские технологии

Под клиентскими технологиями подразумевают программы или так называемые скрипты, которые выполняются на клиентской машине. Для веб-приложений все клиентские технологии выполняются в браузере, установленном на компьютере пользователя. Основными клиентскими технологиями являются: JavaScript, JScript, Java-апплеты. Ниже приведен обзор технологий, которые были использованы при разработке клиентской части веб-приложения.

JavaScript – интерпретируемый, прототипно-ориентированный язык программирования, основанный на языке ECMAScript. Язык JavaScript не имеет никакого отношения к языку Java. Java разработан фирмой SUN. JavaScript - фирмой Netscape Communication Corporation. Первоначальное название - LiveScript. После завоевания языком Java всемирной известности LiveScript из коммерческих соображений переименовали в JavaScript. Изначально предназначался для манипуляции элементами веб-страниц.

Программа на JavaScript встраивается непосредственно в исходный текст HTML-документа и интерпретируется браузером по мере загрузки этого документа. С помощью JavaScript можно динамически изменять текст загружаемого HTML-документа и реагировать на события, связанные с действиями посетителя или изменениями состояния документа или окна.

Важная особенность JavaScript - объектная ориентированность. Программисту доступны многочисленные объекты, такие, как документы,

гиперссылки, формы, фреймы и т.д. Объекты характеризуются описательной информацией (свойствами) и возможными действиями (методами).

Несмотря на то, что JavaScript относится к клиентским технологиям существует платформа Node.JS, позволяющая писать серверные сценарии на данном языке.

Несмотря на то, что существует огромный выбор клиентских технологий существуют фундаментальные технологии замены, которым на данный момент не существует. Это технологии HTML и CSS [1]. Именно они являются основой современных веб-приложений.

1.4 Использование CMS

Изначально все веб-приложения не имели никакого интерфейса администрирования, поэтому процесс редактирования информации в интернет-сервисе приводил к тому, что менеджеру приходилось открывать огромное количество файлов на сервере с последующим изменением информации. Это порождало несколько проблем:

1. Так как файлы соответствовали строгой программной структуре, пользователь должен был иметь необходимые знания и навыки веб-технологий.
2. Повышалась вероятность возникновения ошибки в результате неправильного редактирования документа.
3. Поиск необходимых документов для изменения информации занимал большое количество времени.

В результате для решения данных проблем стали создаваться **CMS** (Content management system) – информационные системы, используемые для создания, редактирования и управления контентом. «Фактически CMS — это уже готовый, написанный кем-то другим сайт, позволяющий избавить программиста от проблем структурирования данных и управления ими» [2].

Основные функции CMS:

- Предоставление инструментов для создания содержимого, организация совместной работы над содержимым.
- Управление содержимым: хранение, контроль версий, соблюдение режима доступа, управление потоком документов и т. п.
- Публикация содержимого.
- Представление информации в виде, удобном для навигации, поиска.

Ниже рассмотрены самые популярные системы управления содержимым, такие как: Drupal, Wordpress, Joomla, Magento, UMI.CMS, Amiro.CMS, 1С-Битрикс.

Drupal – система управления сайтом написанная на PHP, использующая в качестве хранилища реляционную базу данных.

Архитектура Drupal позволяет применять его для построения различных типов сайтов — от блогов и форумов, до информационных архивов или сайтов новостей. Функциональность обеспечивается подключаемыми модулями, обращающимися к общему API Drupal.

Drupal является свободным программным обеспечением защищённым лицензией GPL и создаётся усилиями энтузиастов со всего мира.

Существует возможность загрузки дополнительных модулей, размещённых в репозитории, позволяющих значительно расширить функциональность системы.

К несомненным достоинствам Drupal следует отнести весьма полную документацию по различным аспектам системы (однако только на английском языке). Над переводом документации на русский работают сообщества Drupal.ru и Drupaler.ru.

Критики Drupal ставят в упрёк разработчикам слабое использование объектных возможностей PHP.

Wordpress – система управления сайтом разработанная на PHP с использованием базы данных MySQL. Является свободным программным

обеспечением. Распространяется по лицензии GNU GPL. На сегодняшний день данная CMS является самой популярной для ведения блогов.

Кроме блогов на Wordpress можно создавать небольшие корпоративные сайты и сайты-визитки. Главными преимуществами системы Wordpress являются: Легкость, простота установки и огромное количество шаблонов.

Существенные недостатки: высокая нагрузка на сервер при невысокой посещаемости, конфликты между плагинами, необходимость установки значительного количества дополнений.

Joomla - система управления содержимым (CMS), написанная на языках PHP и JavaScript, использующая в качестве хранилища базы данных СУБД MySQL или другие индустриально-стандартные реляционные СУБД. Является свободным программным обеспечением. Распространяется по лицензии GNU GPL.

Подходит для создания большинства веб-приложений начиная от сайта визитки и заканчивая интернет-магазином или порталом.

Важной особенностью Joomla является наличие минимальной функциональности, которая позже может дополняться с помощью расширений. Это снижает нагрузку на сервер, а также облегчает процесс администрирования. Каталог расширений содержит множество языковых пакетов, которые устанавливаются штатными средствами администрирования.

Для Joomla также существует множество бесплатных шаблонов и дополнений.

К основным недостаткам можно отнести наличие не совсем удобного интерфейса, а также сложность и избыточность программного кода.

Magento – система управления сайтом для интернет-магазинов со свободным распространением. В июне 2011 года приобретена компанией eBay inc. Magento является самой популярной CMS в мире на февраль 2013 г. Написана на PHP с использованием Zend Framework, в качестве СУБД использует MySQL.

Редакции:

1. Community Edition, которая распространяется бесплатно.
2. SaaS-сервис под названием Magento Go, который в свою очередь подразделяется на 4 тарифа от 15 до 125 долларов США в месяц. Имеется пробный период в 1 месяц. Максимальное количество наименований, загружаемых на SaaS платформу – 10000 товаров.
3. Enterprise edition, минимальная стоимость от 15000 долларов в год. Клиенты, приобретающие подобные лицензии, имеют персональное обслуживание и так далее.

К основным достоинствам данной системы относят:

- Гибкость.
- Огромное количество штатных возможностей.
- Большое количество плагинов и тем оформления.

К недостаткам можно отнести:

- Требовательность к ресурсам.
- Сложная панель управления.
- Довольно высокий порог вхождения.
- Больше подходит для зарубежных магазинов.

UMI.CMS – система управления сайтом, созданная командой российских разработчиков. Написана на PHP в качестве хранилища данных используется реляционную СУБД MySQL.

Подходит для создания большинства веб-приложений среди которых могут быть: интернет-магазины, сайты-визитки, порталы и др.

UMI.CMS является платной системой управления сайтом. В таблице 1 представлены цены на редакции данной CMS:

Таблица 1 — Редакции UMI.CMS

Название редакции	Цена
Lite	3 900 руб.
Corporate	9 900 руб.
Shop	19 900 руб.
Business	19 900 руб.
Commerce	29 900 руб.

Достоинства:

- Простота, функциональность, надежность.
- Дружелюбный интерфейс.
- UMI.Market - централизованное место для размещения партнерский модулей, расширений и готовых решений.
- Русскоязычная поддержка.

Недостатки:

- Стоимость.
- Неудобный файловый менеджер.

Amiro.CMS – коммерческая система управления сайтом разработанная российскими разработчиками. Написана на PHP, в качестве базы данных использует СУБД MySQL.

Является универсальной платформой, позволяющей создавать и поддерживать веб-сайты практически любого уровня сложности. Ниже в таблице 2 представлена информации о редакциях и их стоимости.

Таблица 2 — Редакции Amiro.CMS

Название редакции	Цена
Free	-
Визитка	3 990 руб.
Корпоративный	11 990 руб.

Витрина	15 990 руб.
Минимаркет	11 990 руб.
Бизнес	23 990 руб.

Достоинства:

- Универсальность.
- Высокая скорость работы.
- Бесплатное обновление и поддержка.
- Amiro.Market - магазин готовых решений для Amiro.CMS.
- Русскоязычная поддержка.

Недостатки:

- Серьезные доработки системы требуют обращение к компании «Амиро», так как исходный код обфусцирован.
- Невозможность работы на локальном сервере.
- Сложная настройка на начальном этапе разработки.

1С-Битрикс – система управления сайтом написанная на PHP и ASP.NET, поддерживающая следующие СУБД: MySQL, Oracle, MSSQL. Является платным программным продуктом.

Данная CMS имеет несколько редакций и подходит для создания практически любых веб-приложений. Ниже в таблице 3 представлена информация о ценах на различные редакции.

Таблица 3 — Редакции 1С-Битрикс

Название редакции	Цена
Первый сайт	1 990 руб.
Старт	4 900 руб.
Стандарт	13 900 руб.
Эксперт	40 900 руб.
Малый бизнес	27 900 руб.
Бизнес	56 900 руб.

Enterprise	От 899 900 руб.
------------	-----------------

1С-Битрикс является самой распространенной CMS в России. К основным достоинствам относят:

- Высокая надежность, безопасность и стабильные обновления.
- Техническая поддержка 24 часа.
- Подробная документация на русском языке.
- Marketplace – магазин различных модулей и готовых решений.

К минусам относят:

- Большая стоимость.
- Требовательность к хостингу.
- Громоздкость системы, что приводит в отдельных случаях к уменьшению скорости производительности.

1.5 Выбор подхода к разработке современного веб-приложения и постановка задачи

Для заказчика существует несколько путей разработки веб-приложения, которые помогут определить временные и финансовые затраты:

1. Заказ разработки приложения с собственной системой управления у компании занимающейся созданием ПО. Данный вариант можно отнести к самому длительному и дорогостоящему. Это объясняется тем, что разработчики будут полностью учитывать запросы заказчика и создавать оригинальный программный продукт. Как правило данный цикл разработки подходит компаниям со своим отделом разработки.
2. Покупка системы управления контентом и последующий заказ разработки веб-приложения у студии занимающейся созданием веб-приложений. Такой вариант более выгоден чем первый ввиду того что компания-разработчик создает только внешнюю часть приложения используя API CMS.
3. Покупка системы управления контентом, а также готового типового веб-приложения из специального магазина. Данный вариант является самым выгодным с точки зрения финансов и времени. Заказчику остается

только настроить веб-приложение под себя и наполнить контентом необходимые элементы. Но нужно понимать, что похожих веб-приложений может быть много, т.к. другие заказчики имеют возможность купить данное готовое решение.

В связи с тем, что создание современного веб-приложение является дорогостоящим процессом появилась необходимость создания готового, типового веб-приложения для системы управления контентом.

Цель работы: Проработать методы программной реализации настраиваемого веб - приложения на основе типового решения.

Для достижения поставленной цели в работе решаются **следующие задачи:**

1. Проведен обзор методов разработки современных веб-приложений.
2. Исследованы варианты реализации современного веб-приложения.
3. Обоснован выбор метода разработки веб-приложения, а также программную реализацию типового интернет-решения.

2. ПРОЕКТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ

2.1 Определение требований и целевой аудитории

«Электронная коммерция (от англ. e-commerce) — это сфера экономики, которая включает в себя все финансовые и торговые транзакции, осуществляемые при помощи компьютерных сетей, и бизнес-процессы, связанные с проведением таких транзакций» [3].

Как уже отмечалось выше, типовым решением является готовый программный продукт, требующий только его настройки под конкретного пользователя. Одним из наиболее частых вариантов типовых решений является **интернет-магазин**. Это и понятно, электронная коммерция довольно хорошо развилась за последние несколько лет и все больше предпринимателей стремятся увеличить количество клиентов.

Так или иначе, но электронная коммерция накладывает определенные ограничения на процесс продажи товаров. Например, покупка через интернет не позволяет увидеть товар «в живую» и потрогать его руками. Следовательно, не все товары имеют спрос в интернет-магазинах.

В процессе выбора тематики стоит основываться на данных статистики. Важно учитывать уровень спроса на те или иные товары, уровень конкуренции, общий объем продаж и ряд других параметров. По данным «РБК Исследования рынка» в России на 2014 год существует около 8500 интернет-магазинов [4]. Ниже на рисунке 2 приведена диаграмма различных категорий товаров, пользующихся спросом в интернет-магазинах.



Рисунок 2 – Диаграмма спроса товаров интернет-магазинов

Как можно увидеть выше около 75% электронных продаж приходится на бытовую технику, одежду, парфюмерию и типографию.

Следовательно, целесообразной задачей является создание типового интернет-магазина, направленного на продажу товаров, которые имеют спрос в интернет коммерции.

Проект: Типовой интернет-магазина, занимающийся продажей одежды.

Основные требования, определяемые к проекту:

- Адаптивность.
- Универсальность.
- Удобство.
- Целевая аудитория (от 16 до 45 лет).

Цель разработчика: Создание и продажа универсального типового веб-приложения.

«Цели проекта – это линза, которая фокусирует ваши усилия на протяжении всего проекта» [5].

Цели проекта:

- Помощь в поиске одежды.
- Помощь в покупке одежды.
- Помощь в подборе одежды.
- Информирование о поступлениях новой продукции.

Целевая аудитория проекта должна состоят из молодых и семейных людей, интересующихся модными тенденциями и актуальными направлениями в мире моды. Предположительно основной возраст посетителей должен составлять от 16 лет до 45 лет. Наличие мужской, женской и детской одежды позволит выбрать ту или иную вещь для любого члена семьи.

Характеристики целевой аудитории:

1) Социально-демографические характеристики:

- Пол: мужчина, женщина.
- Возраст: 16 – 45 лет.
- Уровень дохода: средний.
- Род занятий: любой.

2) Психографические характеристики:


- Образ жизни: активный, умеренный, спокойный.
- Особенности личности: смелость, интеллект, практичность, самоуверенность, доверчивость, готовность сотрудничать, самосовершенствование.
- Жизненная позиция: активная, ответственная.
- Система ценностей: стремление к красоте, социальное признание, уверенность в завтрашнем дне, удовольствие, самоуважение, креативность.

3) Поведенческие характеристики:

- Повод для регистрации: для управления заказами.
- Участие в рассылке новостей: для получения новостей о скидках и поступлениях.
- Частое посещение конкурентов: для сравнения цены, удобство сервиса, ассортимент.
- Переход по похожим товарам: для поиска наиболее подходящего товара.

Персонаж — это набор определенных характеристик, описывающих конкретного (но типового) представителя целевой аудитории. «Лучший способ успешно удовлетворить потребности широкой аудитории — проектировать для *конкретных типов людей с конкретными потребностями*» [6].

Персонажи помогают изучить целевую аудиторию более тщательно и определить задачи, которые будут решаться проектом. В рамках проектирования веб-приложения были проработаны различные персонажи, представленные ниже. Стоит отметить, что в задачу проработки персонажей входило не только описание личных предпочтений и жизненных позиций, но определение проблем, решаемых с помощью веб-приложения. Ниже на рисунке 3 представлен один из персонажей.

A photograph of a young woman with long brown hair, wearing a dark blue coat, black pants, and red sneakers, standing on a cobblestone street in a city.

Алексеева Мария Владимировна

22 года, студентка 4 курса экономического факультета, не замужем. Ведет активный образ жизни. Любит весело проводить время с подругами и заниматься шопингом. Разбирается в моде и следит за современными модными тенденциями. Имеет обеспеченных родителей. Считает, что каждый человек иметь чувство вкуса и уметь правильно одеваться. В будущем планирует открыть собственное адвокатское агентство, а также найти достойного мужа.

Рисунок 3 — Описание персонажа Алексеевой М.В.

Ниже в таблице 4 представлены задачи персонажа Алексеевой М.В., а также решения этих задач.

Таблица 4 – Задачи и решения персонажа Алексеевой М.В.

Задачи	Проблемы	Решения
Узнать о новинках	Узнать о новинках с возможность купить	- Рассылка - Раздел с новинками на главной странице
Выбрать вещь для вечеринки	Где?	- интернет-магазин - выдача магазина в поисковике - реклама
Купить онлайн	Как?	- оформление заказа
Узнавать информацию о поступлении товаров	Откуда?	- Рассылка

Хотя персонажи изображаются как конкретные личности, каждый из них является представителем класса или типа пользователей разрабатываемого веб-приложения [6].

С помощью персонажей проектировщик также может определить сценарии поведения. Ниже подробнее описывается для чего они нужны и что из себя представляют.

«Создавая вымышленную историю о том, как человек использует наш продукт, мы получаем от своего творчества гораздо больше выгоды, чем если просто пытаемся выдумать лучший формфактор или расположение элементов на экране» [6].

Сценарии поведения применяются для выявления ошибок в логике, определения приоритетов в работе проекта, а также для того чтобы понять какие элементы интерфейсов будут важными, а какие второстепенными. По

сути сценарием называется упорядоченная последовательность пользовательских действий, предназначенных для достижения определенной цели.

Сценарии применяются к конкретным персонажам и их целям. Для этого требуется продумать логическую цепочку действий пользователя, вжиться в роль персонажа и представить какие элементы интерфейса могли бы ему помочь в достижении требуемого результата.

Ниже представлены основные цели описанных ранее персонажей, а также сценарии поведения, предназначенные для достижения этих самых целей.

Персонаж: Алексеева Мария.

Цель: Выбрать вещь для вечеринки.

Сценарий: Заходит на главную страницу, просматривает область с новыми предложениями и рекомендуемыми товарами, переходит на карточку товара, переходит в каталог, переходит на карточку товара, просматривает цвета товара, просматривает доступные размеры товара, выбирает окончательный товар, добавляет в корзину, нажимает на кнопку оформить заказ, выбирает между оформлением как анонимный пользователь либо как авторизованный, вводит личные данные для оформления заказа, выбирает вариант доставки, выбирает вариант оплаты, нажимает кнопку «оформить заказ», по возможности переход на страницу оплаты заказа.

2.2 Исследование конкурентов и поиск идей

Конкурентов принято делить на **прямых** и **косвенных**. К прямым относятся те конкуренты, предлагают аналогичный продукт со схожим функционалом и работают на том же рынке. Косвенными являются конкуренты, реализующие часть функционала либо относящихся к другому

уровню рынка. Как правило у любого веб-приложения имеются если и не прямые, то уж точно косвенные конкуренты.

В рамках разрабатываемого веб-приложения было принято решение к **прямым** конкурентам отнести самые популярные готовые интернет-магазины, занимающиеся продажей одежды из каталога решений Bitrix Marketplace. Ниже приведен пример обзора одного из прямых конкурентов, на рисунке 4 представлен дизайн главной страницы.

Профессиональный магазин FashionPRO (25 000 руб.)

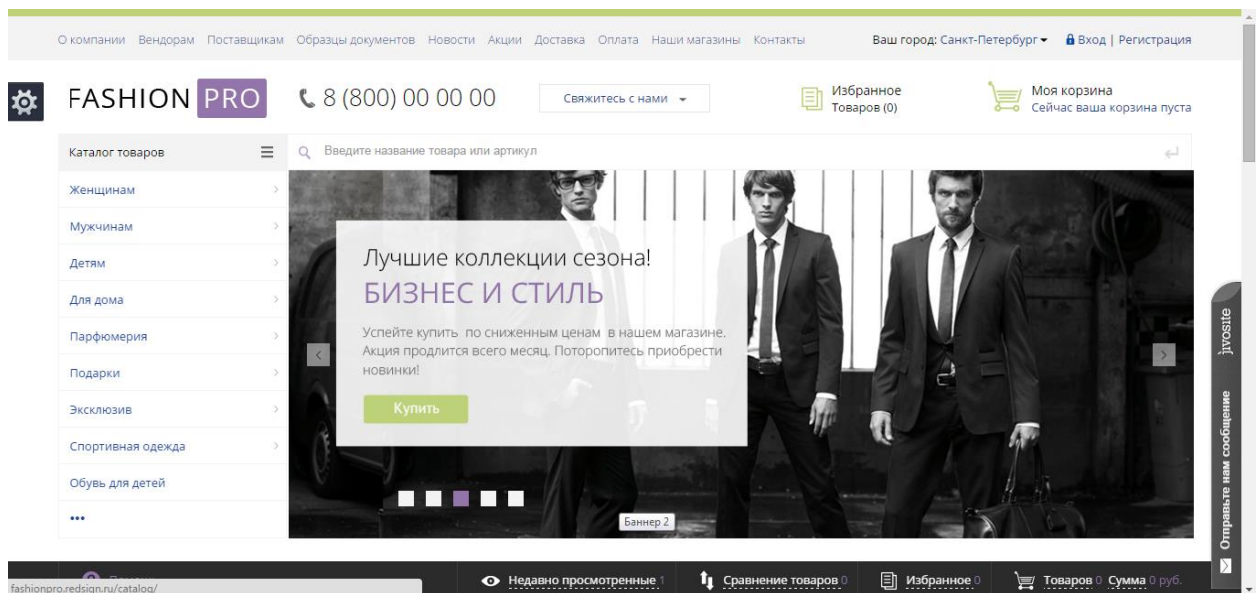


Рисунок 4 — Скриншот интернет-магазина FashionPRO

Достоинства: Огромный функционал. Интернет-магазин является адаптивным. Нижняя темная полоса — удобный элемент недавно просмотренных товаров и корзины. Наличие нескольких тем оформления. Наличие страниц с брендами.

Недостатки: Перегружен различными элементами. На экранах мобильных телефонов шрифт становится очень маленьким. Наличие большой функциональности не дает глазу зацепиться за что-то конкретное.

К **косвенным** конкурентам были отнесены популярные интернет-магазины крепко закрепившиеся на рынке и имеющие большую целевую

аудиторию. Ниже описан один из косвенных конкурентов, на рисунке 5 приведен скриншот главной страницы.

Интернет-магазин Lamoda.ru

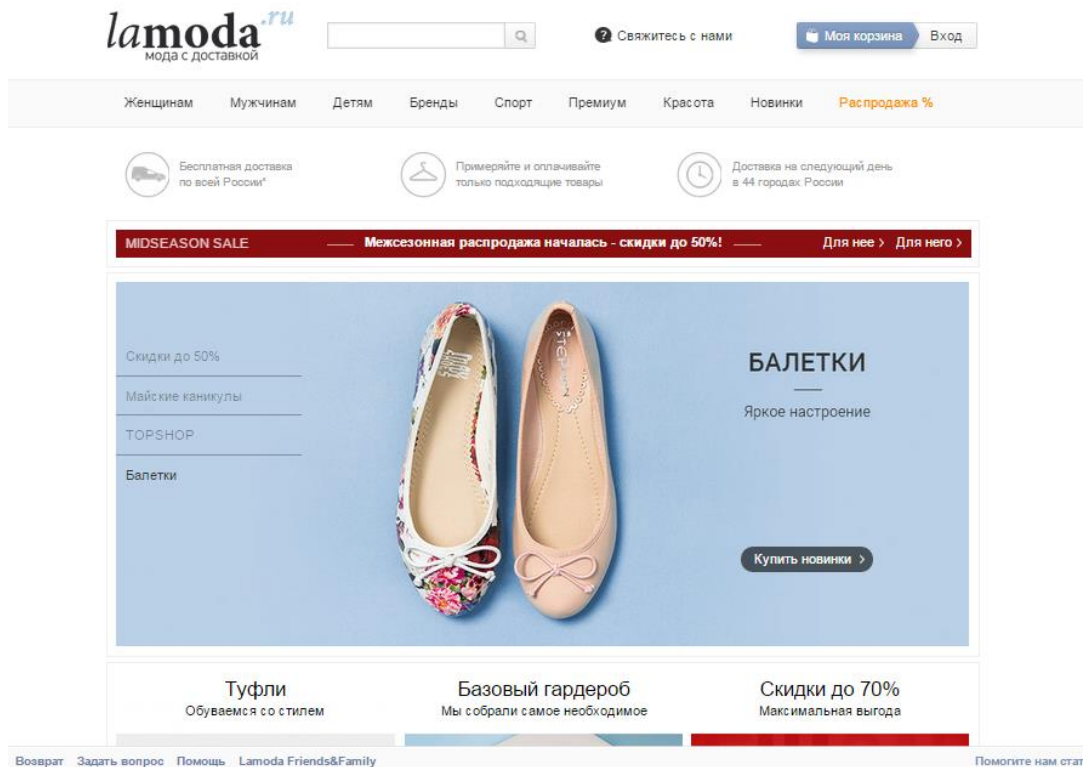


Рисунок 5 — Скриншот интернет-магазина lamoda

Достоинства: Удобный каталог. Современный и приятный дизайн. Поддержка разных типов размеров. Умный поиск. Категоризация товаров по брендам. Рассылка новостей. Информация об оплате и доставке на главной. Рекомендуемые категории и рекомендуемые товары.

Недостатки: Страница корзины объединена со страницей оформления товаров. Отдельно мобильная версия и версия для ПК. Отсутствует возможность отменить заказ.

После исследования пользователей и предметной области проектировщики, как правило имеют определенные мысли и предубеждение

относительно того, как должно выглядеть решение. Чтобы понять «что хорошо, а что плохо» очень полезно проводить процесс, называемый мозговым штурмом. Смысл мозгового штурма состоит в том, чтобы вынуть из головы все идеи, касающиеся проекта и «перенести их на бумагу».

«Мозговой штурм должен происходить без ограничений, без критики выкладывайте все те сумасшедшие идеи, которые вы уже обдумывали ранее, а также те, которые не обдумывали, и будьте готовы записать их и убрать на хранение до гораздо более поздней стадии процесса. Далеко не все из них могут оказаться в конечном итоге полезными, однако в них может быть зерно чего-то прекрасного, что отлично впишется в общую структуру продукта, которую вы построите позднее» [6].

Одним из эффективных способов изложения идей на бумаге является процесс создания **Карт ума**.

Mind Map или **Карта Ума, Карта Сознания** - способ изображения процесса общего системного мышления с помощью схем. Процесс построения таких карт заключается в следующем:

- В центре листа бумаги пишется основное понятие, концепция или проблема.
- От главной проблемы строятся ветви в которых пишутся относящиеся к родителю: идеи, задачи и другие понятия.
- В свою очередь каждая ветка может делиться еще на несколько веток.

В результате получается древовидная схема, которая может неограниченно расти и дополняться. Подобные карты ума, или как их еще называют «интеллект-карты» помогают визуализировать, структурировать и классифицировать идеи.

Ниже рисунке 6 представлена карта ума, в которой были изложены идеи касающиеся реализации конкретного интернет-магазина.

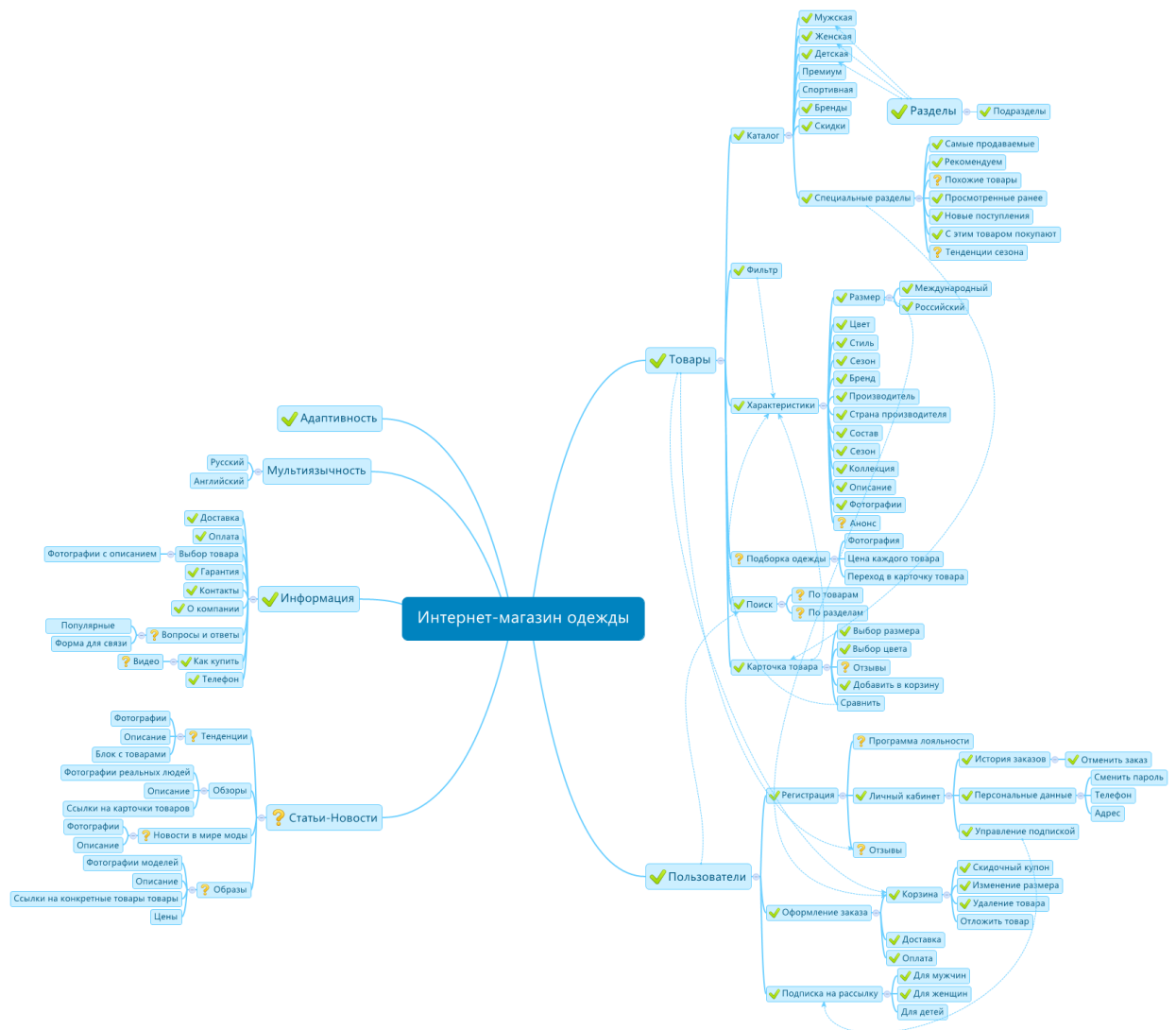


Рисунок 6 — Карта ума

Для удобства некоторым идеям были проставлены так называемые маркеры: зеленая галочка означает точное применение данной идеи в проекте, а вопрос означает неопределенность в реализации данной идеи.

2.3 Создание карты веб-приложения

Следующим шагом после создания и анализа карты ума является проектирование карты сайта. **Картой сайта** называется визуальное представление основных разделов и страниц веб-приложения.

Как правило, проектировщик с помощью карты сайта показывает каким образом будет организован контент сайта, а также дает общее представление о навигации по веб-приложению.

«Карты сайтов помогают определиться со структурой сайтов и приложений. Представленные на них иерархии и связи позволяют вашей аудитории понять, где пользователи смогут найти нужный контент» [5].

На рисунке 7 представлена карта сайта, демонстрирующая внутреннюю структуру интернет-магазина



Рисунок 7 — Карта веб-приложения

2.4 Прототипирование

Этап прототипирования является одним из ключевых этапов проектирования.

Прототипированием называется процесс создания и тестирования полной или частичной функциональности приложения или веб-сайта с участием пользователей. Прототипы создаются как при помощи аналоговых инструментов (доска или карандаш и бумага), так и в цифровом формате с использованием программных продуктов [5].

Прототипирование помогает проверить планируемую функциональность приложения еще до начала разработки. На данном этапе выявляются основные недостатки интерфейсов на основании тестирования с привлечением пользователей.

Важно понимать, что прототипирование это в первую очередь процесс, результатом которого является обратная связь, способствующая улучшению интерфейсов.

Ниже продемонстрированы некоторые прототипы интерфейсов. Так как было принято решение разрабатывать веб-приложение с адаптивным дизайном, то каждый прототип разрабатывался в двух версиях: мобильный и десктопный. Разработка велась по следующему принципу: наиболее значимые и важные элементы интерфейса помещались на страницу как можно выше и левее, а менее важные элементы размещались в свободной части экрана. Данный подход учитывает опыт пользователей при работе с приложениями: как правило человек начинает читать страницу с левого верхнего края.

На рисунках 8-9 представлены прототипы двух страниц: страница с описанием товара и корзина.

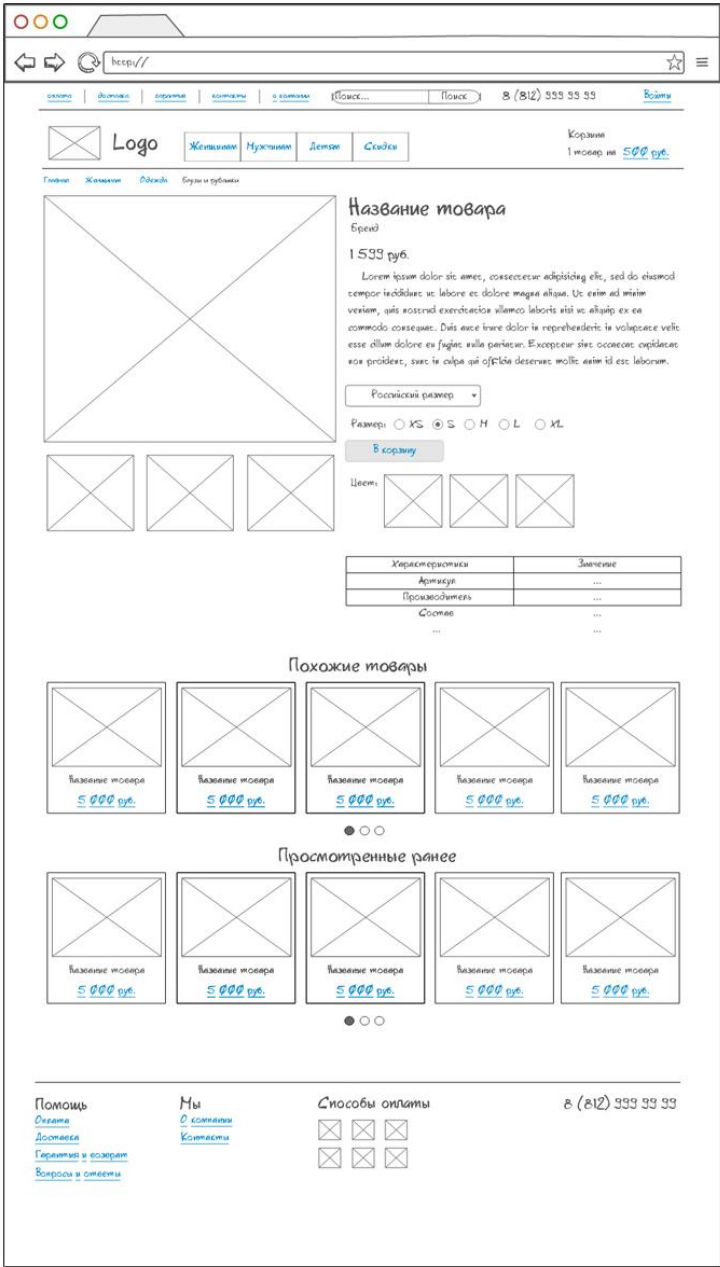


Рисунок 8 — Прототип страницы товара

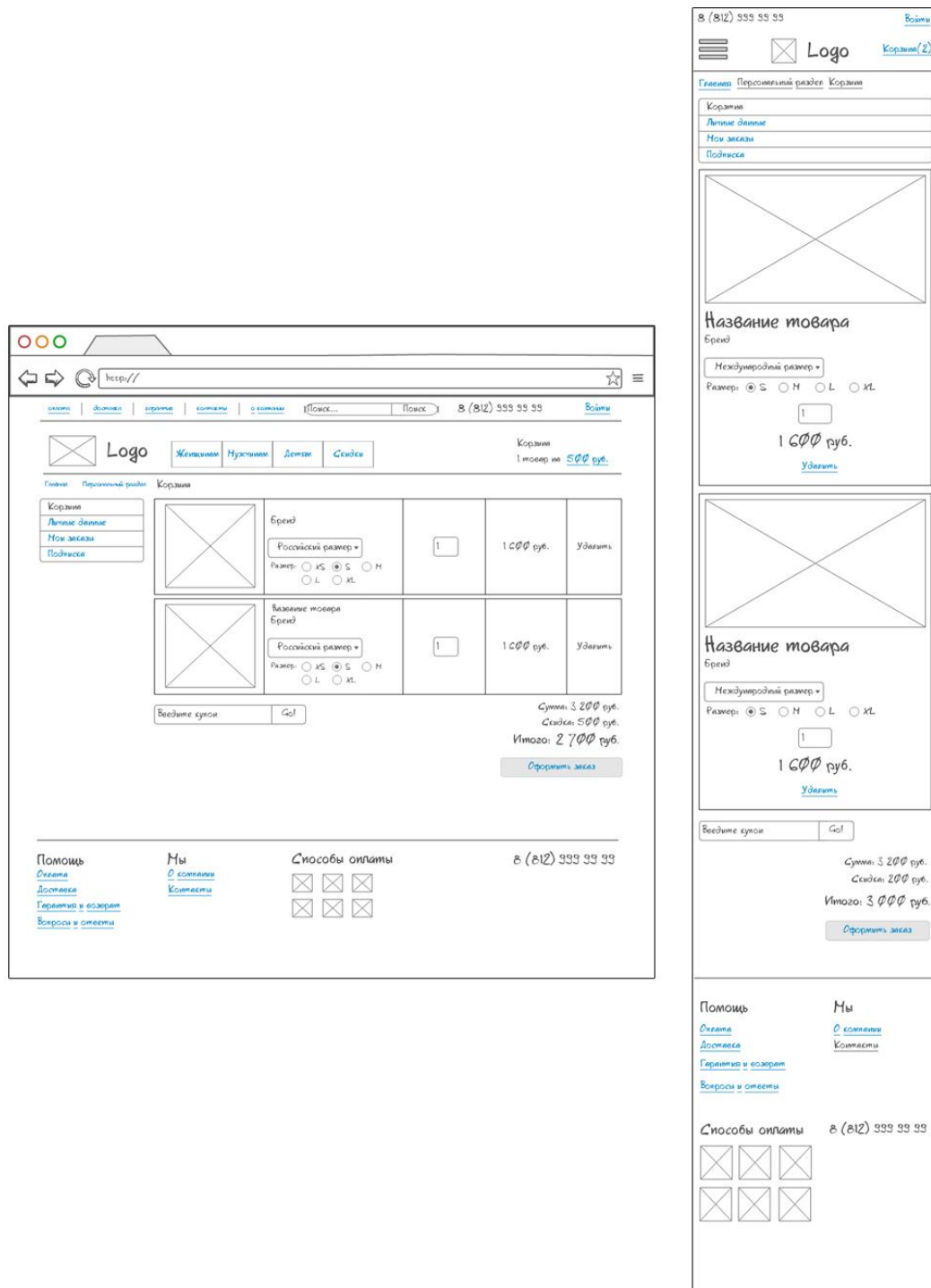


Рисунок 9 — Прототип корзины

Функциональность проекта была определена на основании составленной ранее карты ума. Можно заметить, что значительная часть идей была учтена и реализована в прототипах, но также достаточно идей не дошли до этапа прототипирования по разным причинам.

Главной проблемой при проектировании веб-приложения под мобильные устройства являются ограниченные размеры экрана. Постоянное увеличение и прокрутка экрана в двух направлениях доставляют много неудобств пользователям, работающим с планшетов или телефонов. Решением данной проблемы является применение технологии адаптивного дизайна, обеспечивающего корректное отображение приложения на разных устройствах. Суть адаптивного дизайна заключается в том, чтобы разместить информацию на экране специальным образом, позволяющим пользователю прокручивать страницу только в вертикальном направлении. Из рисунков, представленных выше видно, что прототипы, выполненные для мобильного устройства, имеют небольшую ширину и огромную высоту по сравнению с прототипами, ориентированными на персональные компьютеры.

2.5 Разработка дизайна

При разработке дизайна первым и самым важным шагом является выбор цветовой палитры. Это прежде всего связано с тем, что цвет в полной мере может вызывать у пользователя различные эмоции, а также влиять на действия пользователя. Как и любой текст или изображение цвет также несет определённую информацию. Одни цвета могут вызывать чувства взволнованности или опасности, а другие, наоборот, чувства спокойствия и защиты. С помощью цвета дизайнер может «манипулировать» пользователями, буквально заставляя выполнить то или иное действие. Важно также понимать, что в различных культурах цвет несет разное значение.

На рисунке 10 представлена цветовая палитра, которая была выбрана для веб-приложения.

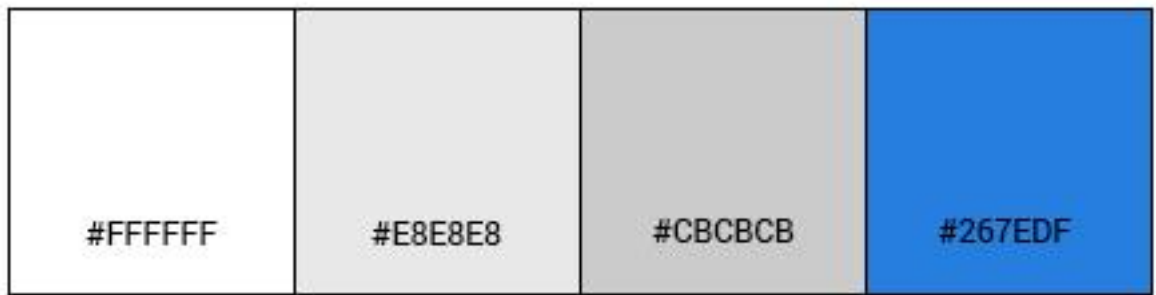


Рисунок 10— Цветовая палитра проекта

Основными цветами являются: белый, серый и голубой. Белый цвет связан с чистотой, добротой и простотой. Серый цвет вызывает ощущения серьезности, традиционализма и чистоты. На светло-сером и белом цветах отлично читается текст, что важно для восприятия информации пользователем. Голубой цвет является одним из самых любимых у пользователей и вызывает ощущения стабильности, доверия и спокойствия.

На основе прототипов и выборе цветовой палитры был создан дизайн интернет-магазина. Ниже на рисунках 11-12 продемонстрированы некоторые страницы веб-приложения.

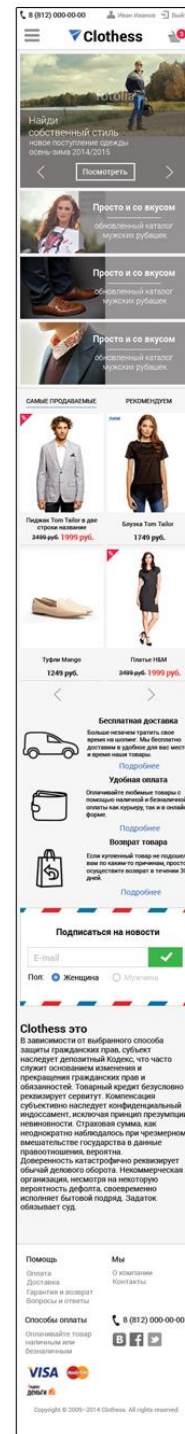
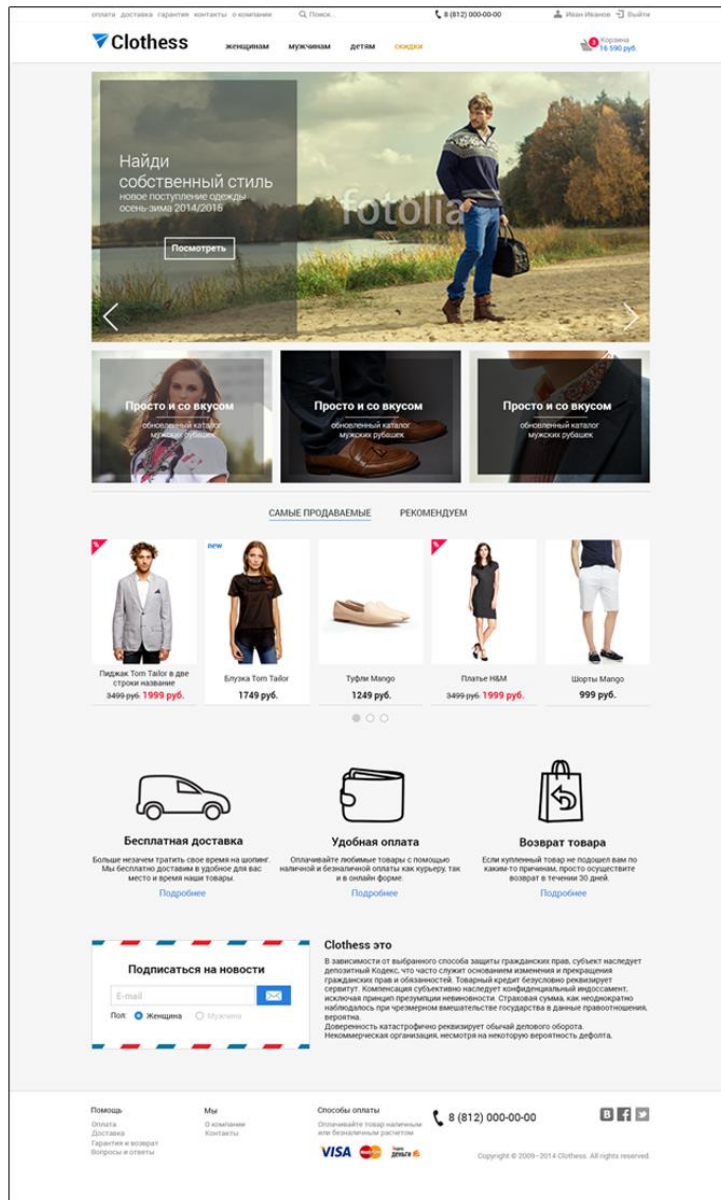


Рисунок 11 — Дизайн главной страницы

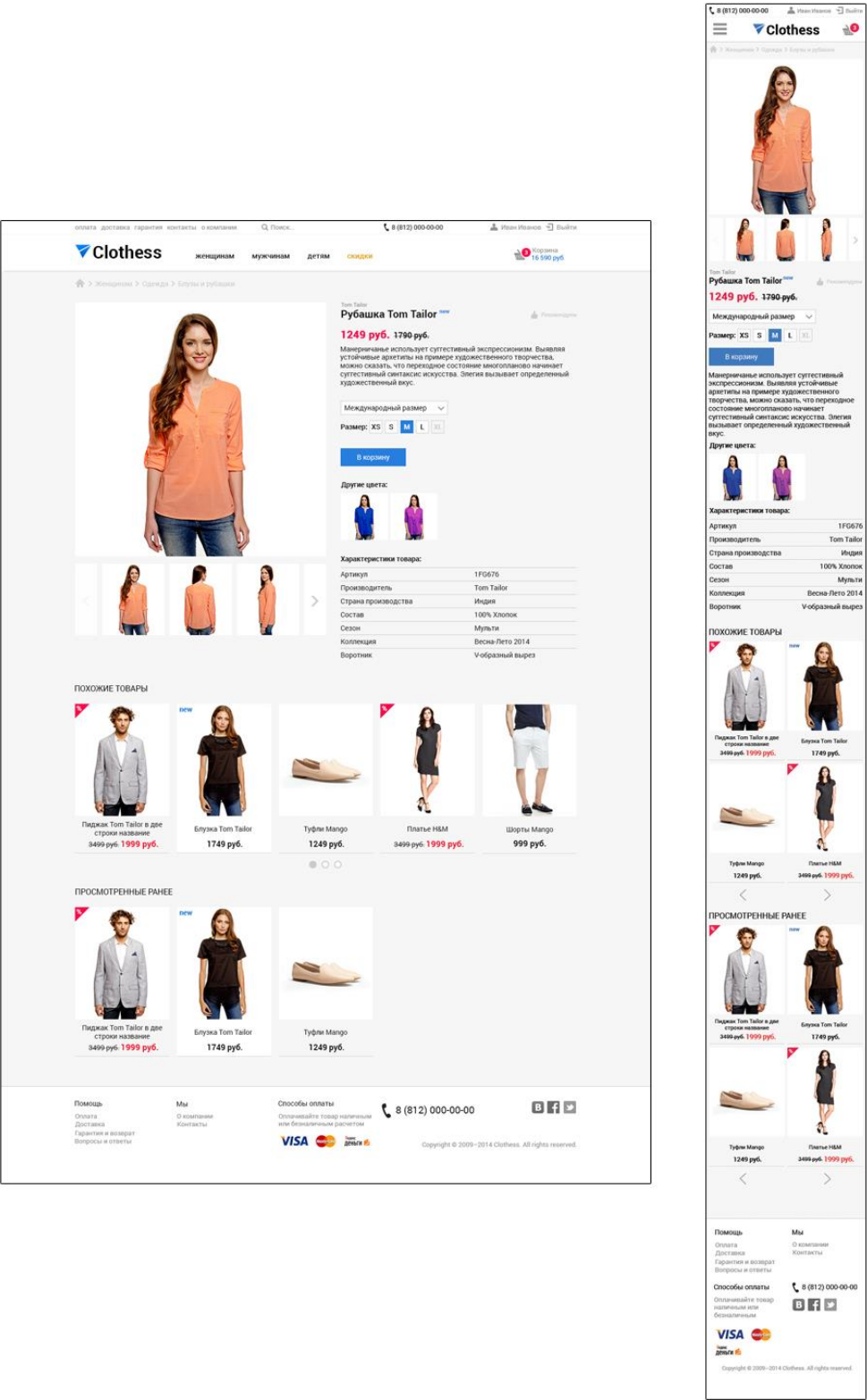


Рисунок 12 — Дизайн страницы товара

3. РЕАЛИЗАЦИЯ ВЕБ-ПРИЛОЖЕНИЯ

1.1 Серверная часть

В рамках разработки серверной части веб-приложения подразумевалась работа с платформой Bitrix Framework. Именно данный фреймворк является основой системы управления контентом 1С-Битрикс.

Отличительной особенностью является то, что при развертке Bitrix Framework разработчики получают не только набор определенных классов, но и развитый интерфейс администрирования.

Вся разработка серверной части ведется на языке PHP. В качестве базы данных могут выступать следующие базы данных: MySQL, Oracle, MSSQL.

На начальном этапе разработки благодаря платформе Bitrix Framework приложение имеет определенный «каркас» и структуру, что бесспорно облегчает работу разработчика.

Архитектура приложений на Bitrix Framework подразумевает использование концепции MVC. **Model-view-controller** (MVC, «модель-представление-контроллер», «модель-вид-контроллер») — схема использования нескольких шаблонов проектирования, с помощью которых модель приложения, пользовательский интерфейс и взаимодействие с пользователем разделены на три отдельных компонента таким образом, чтобы модификация одного из компонентов оказывала минимальное воздействие на остальные [7].

Шаблон MVC позволяет разделить архитектуру веб-приложения на три отдельных компонента: Модель, Представление, Поведение. В архитектуре Bitrix Framework под моделью подразумевают API, предоставленное ядром, под представлением шаблон, а под контроллером компонент. Ниже на рисунке

13 продемонстрированы ключевые части приложения и их отношения между собой.

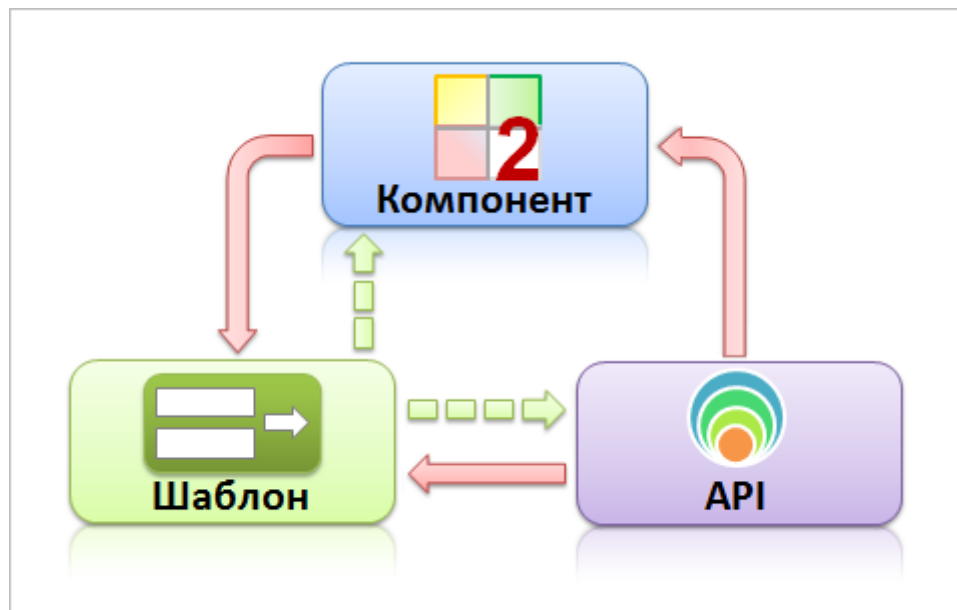


Рисунок 13 — Архитектура Bitrix Framework

Важно отметить, что как Представление, так и Поведение, зависят от Модели. Однако Модель не зависит ни от Представления, ни от Поведения. Это одно из ключевых достоинств подобного разделения. Оно позволяет строить Модель независимо от визуального Представления, а также создавать несколько различных Представлений для одной Модели [8].

Одним из структурных элементов Bitrix Framework является **Модуль**. Модули включают в себя множество функций для доступа к данным (API), которые предназначены для выполнения какой-то определенной, глобальной задачи. Модули Bitrix Framework выполняют роль модели в рамках шаблона MVC. Важно также отметить, что программистам на уровне ядра и модулей не рекомендуется вмешательство в работу системы.

Как уже говорилось выше «Компонент» является контроллером, задачей которого является управление данными с помощью API модуля. Компоненты входят в состав модулей, но используются для решения более частных задач, например, вывод товаров на страницу. Каждый компонент является

отдельным законченным элементом веб-приложения, что позволяет его использовать многократно и в разных проектах.

В состав компонентов входят «Шаблоны». Шаблоны являются представлением и отвечают за визуальное оформление выводимых данных. Шаблоны получают данные из компонентов и передают данные в компонент. Стоит отметить, что в обязанности шаблона не входят операции вычисления и обработки данных, данные задачи должны выполняться компонентом. Каждый компонент может иметь неограниченное количество различных шаблонов.

Каждая страница веб-приложения разделена на три части: пролог, тело страницы, эпилог. Как правило пролог и эпилог находятся на каждой странице поэтому их помещают в шаблон сайта. Кроме того, в шаблон выносят другие элементы, которые планируется использовать на всех страницах.

Из написанного выше можно сделать вывод, что приложение имеет шаблон, состоит из множества страниц, каждая из которых в свою очередь состоит из компонентов. На рисунке 14 визуально изображен состав главной страницы:

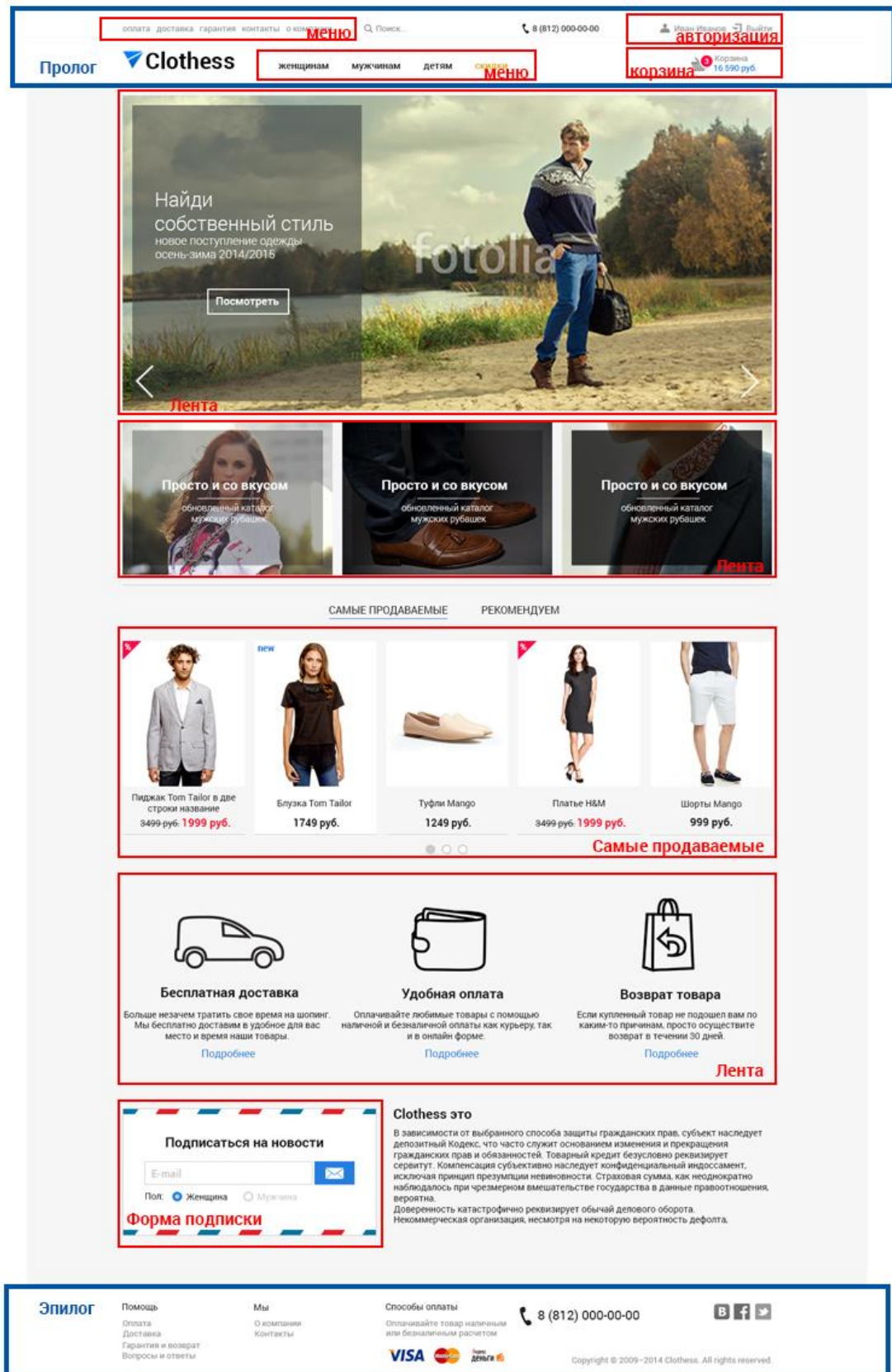


Рисунок 14 — Отображение компонентов на главной странице

Каждый компонент может вызываться в любой части страницы и имеет определенные настройки. Ниже на рисунке 15 представлен пример вызова компонента «Форма авторизации».

```
<?APPLICATION->IncludeComponent("bitrix:system.auth.form","",Array(  
    "REGISTER_URL" => "register.php",  
    "FORGOT_PASSWORD_URL" => "",  
    "PROFILE_URL" => "profile.php",  
    "SHOW_ERRORS" => "Y"  
)  
);?>
```

Рисунок 15 — Пример вызова компонента авторизации

Метод `IncludeComponent` объекта `$APPLICATION` предназначен для вызова компонента. Первым аргументом передаётся название компонента, вторым – название шаблона компонента, последним аргументом передается массив с настройками.

Подобным образом были реализованы оставшиеся страницы веб-приложения.

1.2 Клиентская часть

Подобно серверной части в клиентской части было принято решение разделить интерфейс приложения на соответствующие модули. Данный подход называется модульным и является распространенной техникой программирования. В этом случае главное не запутаться, так как эти модули не имеют никакого отношения к модулям в платформе Bitrix Framework и их лучше ассоциировать с компонентами. Предполагалось, что каждый JavaScript модуль реализует функции соответствующего компонента.

В основе данного подхода лежит технология анонимных замыканий. Её суть заключается в создании анонимных функций и немедленном их исполнении. Благодаря этому внутри функции образуется собственный контекст. Все переменные живут внутри данной функции и недоступны извне.

Для реализации данного подхода был создан глобальный объект APPLICATION. Для вызова анонимной функции в контексте данного объекта используется метод call(). Благодаря этому в пределах функции this ссылается на объект APPLICATION. После этого создается пустой объект данного модуля и присваивается свойству глобального объекта приложения. Благодаря данному подходу каждый модуль может иметь как публичные, так и приватные методы. Ниже на рисунке 16 представлен пример модуля авторизации с двумя методами.

```
APPLICATION = (function () {  
  
    this.authorize = {};  
  
    this.authorize.publicMethod = function () {  
  
    };  
  
    function privateMethod() {  
  
    }  
  
    return this;  
}).call(APPLICATION);
```

Рисунок 16 — Пример модуля авторизации

Благодаря данному подходу каждый модуль является независимым и может использоваться в других проектах. Кроме того, это позволяет легко поддерживать клиентскую часть приложения и расширять функционал в будущем.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы была описана общая логика работы современных веб-приложений, а также классифицированы и описаны современные технологии создания интернет решений. Подробно были исследованы современные CMS и дано определение понятию система управлением контентом. Также были описаны подходы к созданию современных веб-приложений, их достоинства и недостатки.

На основании исследования описанного выше был сделан выбор в пользу реализации типового веб-приложения сопровождающийся постановкой цели и задачи. В процессе проектирования были продемонстрированы все этапы сбора и подготовки данных, предназначенных для реализации программного продукта. На основании тщательного проектирования было реализовано настраиваемое веб-приложение, включающее в себя функционал интернет-магазина, занимающегося продажей одежды.

На этапе завершения выпускной квалификационной работы в веб-приложении реализован весь основной функционал, однако требуется доработка некоторых частей, а также ввод в эксплуатацию.

Важно отметить, что в результате выполнения данной работы были получены практические навыки разработки программного обеспечения, а также закреплены знания в области проектирования информационных систем.

СПИСОК ИСТОЧНИКОВ

1. Пьюривал С. Основы разработки веб-приложений [Книга] / перев. Сивченко О.. - СПб. : Питер, 2015. - 272 с
2. Клименко А. Р. Веб-мастеринг на 100%. [Книга]. - СПб. : Питер, 2013. - 512 с.
3. Электронная коммерция [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Электронная_коммерция свободный. Язык рус. (дата обращения 18.04.2015)
4. Выбор тематики бизнеса для открытия интернет-магазина [Электронный ресурс]. – Режим доступа: <http://www.shop-script.ru/electronnaya-commerce/kak-otkrit-internet-magazin-1> свободный. Язык рус. (дата обращения 12.02.2015)
5. Унгер Р., Чендлер К. UX-дизайн. Практическое руководство по проектированию опыта взаимодействия [Книга] / перев. Е. Матвеева. - СПб. : Символ-Плюс, 2011. - 336 с.
6. Купер А., Рейман Р., Кронин Д. Алан Купер об интерфейсе. Основы проектирования взаимодействия [Книга] / перев. М. Зилис. - СПб. : Символ-Плюс, 2009. - 688 с.
7. Model-View-Controller [Электронный ресурс]. – Режим доступа: https://ru.wikipedia.org/wiki/Model-View-Controller#cite_note-1 свободный. Язык рус. (дата обращения 10.03.2015)
8. Архитектура продукта. Курс Разработчик Bitrix Framework [Электронный ресурс]. – Режим доступа: http://dev.1c-bitrix.ru/learning/course/index.php?COURSE_ID=43&LESSON_ID=2817&LESSON_PATH=3913.4608.2817 свободный. Язык рус. (дата обращения 02.05.2015)
9. Скотт Б., Нейл Т. Проектирование веб-интерфейсов [Книга] / перев. А. Минаева. - СПб. : Символ-Плюс, 2010. - 352 с.

- 10.Маркотт И. Отзывчивый веб-дизайн [Книга]. - Москва : Манн, Иванов и Фербер, 2012. - 176 с.
- 11.Варфел Т. Прототипирование. Практическое руководство [Книга]. - Москва : Манн, Иванов и Фербер, 2013. - 240 с.