

Министерство транспорта Российской Федерации
Федерального агентства железнодорожного транспорта
Федеральное государственное бюджетное образовательное учреждение
высшего профессионального образования
«Дальневосточный государственный университет путей сообщения»
Естественно - научный институт
Кафедра «Вычислительная техника и компьютерная графика»

РАЗРАБОТКА WEB-ИНТЕРФЕЙСА САЙТА

Отчет по курсовой работе
КР.09.03.01.РСАПР.01.11.000-943

Исполнитель

студент _____ О. О. Овсейчук

Проверил

доцент _____ Е. В. Фалеева

Хабаровск 2016

СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	3
1 ЭТАПЫ РАЗРАБОТКИ ВЕБ — САЙТА	5
1.1 Сбор информации.....	5
1.2 Проектирование интерфейса сайта.....	7
1.3 Front-end разработка.....	8
1.4 Back-end разработка.....	8
2 ИСПОЛЬЗУЕМЫЕ ТЕХНОЛОГИИ ПРИ WEB-РАЗРАБОТКЕ.....	10
.....	10
2.1 HTML5.....	10
2.2 CSS3.....	11
2.3 JavaScript.....	12
2.4 Ruby on Rails.....	14
3.1 Разработка RoR приложения	18
Рисунок 3.2 – Запуск перво приложения	19
Рисунок 3.3 – application.html.erb.....	19
Рисунок 3.4 – форма регистрации пользователей	21
Рисунок 3.5 – форма регистрации постинга на сайте.....	22
3.2 Разработка шаблона сайта.....	22
Рисунок 3.5 – Макет будущего сайта.....	24
3.3 Проектирование базы данных для сайта	25
ЗАКЛЮЧЕНИЕ.....	28
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ.....	29

ВВЕДЕНИЕ

В настоящее время web интерфейсы получили широкое применение в связи с ростом популярности всемирной паутины и следовательно — повсеместное использование веб-браузеров. Всей сакральностью веб-интерфейса является взаимодействие пользователя с веб-сайтом или другим приложением через браузер.

Основным требованием к веб-интерфейсу является его одинаковый вид и функционал при работе в различных браузерах.

Классическим методом создания веб-интерфейса является использование таких инструментов как HTML, CSS, JavaScript для статических интерфейсов и использования PHP для динамических приложений.

Так же часто для создания веб-интерфейсов используют Adobe Flash, Java-апплетов для полной или частичной его реализации. Так как они дают программисту большой контроль над интерфейсом, а так же позволяют обходить несовместимости конфигураций браузеров. Но большим неудобством является подгрузка плагинов.

В настоящее время не обойтись и без технологии Ajax. Главная суть Ajax'a является в том , что он не перезагружает интерфейс целиком, а всего лишь догрузит необходимые данные с сервера, что делает его интерфейс более интерактивным и производительным.

Другим методом создания веб-интерфейсов является разработка на Ruby on Rails. Rails – Это полноценный многоуровневый фреймворк для постройки веб-приложений, использующих базы данных, который основан на архитектуре Модель — Представление — Контроллер (Model-View-Controller, MVC).

Динамичный AJAX-интерфейс, обработка запросов и выдача данных в контроллерах, предметная область, отраженная в базе данных, — для всего этого Rails предоставляет однородную среду разработки на Ruby.

Так же существуют конструкторы сайтов и готовые cms (Content management system) такие как WIX ,1С-Битрикс, Wordpress.. Зачастую они удобны для создания не большие сайтов визиток и блогов, форумов, но при разработке крупного проекта как социальная сет, сайт организаций, где нужен особый функционал или быстроедействие в отдельных частях сайта Нужно писать использовать чистые языки или фреймворки.

При выборе языка для создания Web-приложений основой идеей является его простота для программиста, быстроедействие и его функционал. Сейчас популярны Php, C#, Ruby, Python. У каждого из них существует свои плюсы и минусы, Так же у каждого языка существуют свои фреймворки которые облегчают написания кода для решения одной и той же задачи. Аутентификация, авторизация, сохранение состояния перегружаемой страницы .

В данной дипломном проекте будет реализован Веб- интерфейс сайта общественного совета министерства промышленности и транспорта Хабаровского края с использованием языка Ruby и фреймворка Ruby on Rails.

1 ЭТАПЫ РАЗРАБОТКИ ВЕБ — САЙТА

Создание качественного интерфейса сайта, не самая легкая задача, зачастую необходимо разделять на этапы.

Но предложенные варианты разделения не являются всеобщим критерием и каждый этап можно разбить на подэтап, а те, в свою очередь, на подподэтапы.

1.1 Сбор информации

Сбор информации необходим для того , чтобы полностью понимать структуру сайта на сегодняшний день, а так же понять, что в нем не нравится заказчику и что он ожидает увидеть по окончании проекта.

Аналитика — одна из главных составляющих успеха будущей разработки. Здесь нужно применить творческий подход, опыт, педантичность, а также умение работать с данными различного качества и характера.

На этом этапе выполняется следующее :

- Проработать все детали с заказчиком, для того что бы четко понять концепцию и цель веб-сайта;
- Просмотр прототипа, наработку сайта;
- Анализ подобных сайтов;

По техническому заданию сайт общественного совета министерства промышленности и транспорта Хабаровского края должен содержать: личный кабинет, новостную страницу, страницу с медиа и публикация о совете, страницу с составом совета и раздел с нормативной документацией регулирующий действия совета.

Для анализа подобных сайтов был выбраны сайты Общественной палаты Хабаровского края (рисунок 1) и сайт Общественной палаты Российской Федерации (рисунок 2)

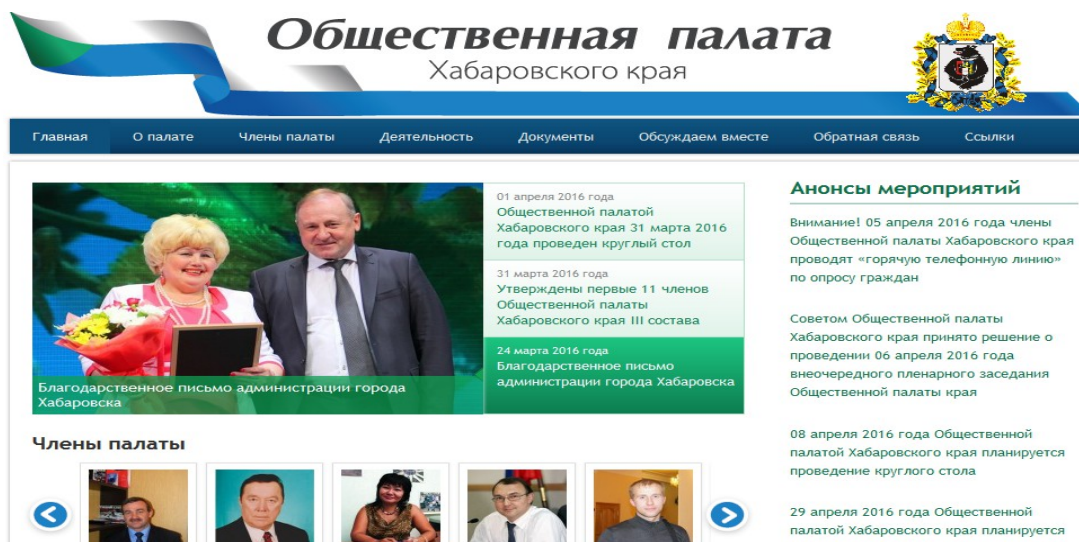


Рисунок 1 — Сайт Общественной палаты Хабаровского края

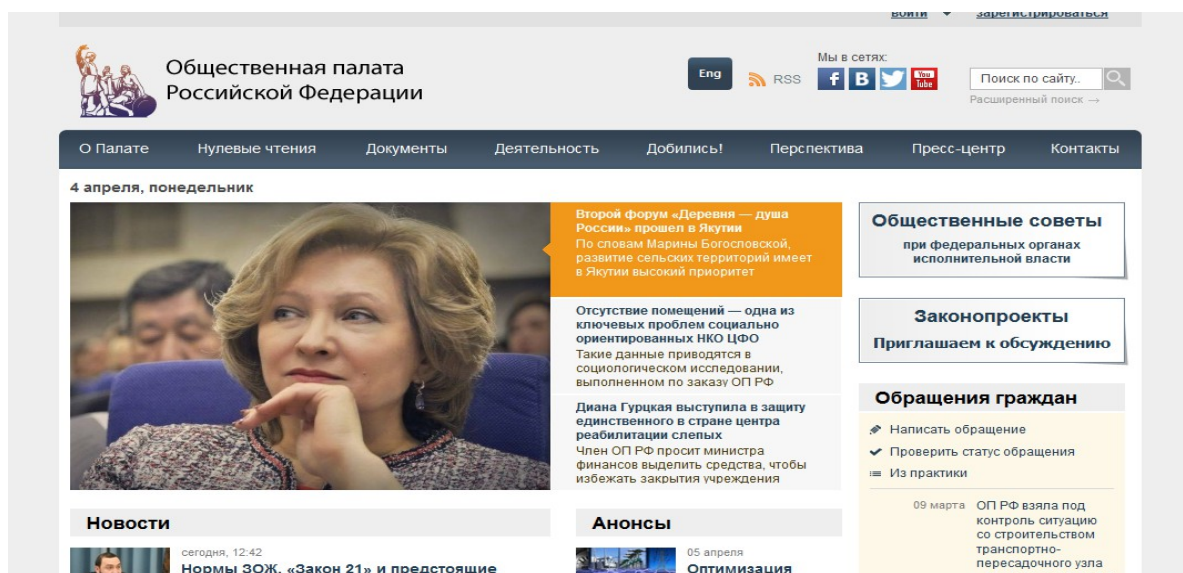


Рисунок 2 — Сайт Общественной палаты Российской Федерации

1.2 Проектирование интерфейса сайта

На данном этапе проектирования создается тестовая версия сайта. Весь ее функционал распределяется по страницам. Разработчик принимает решение относительно того, где будут располагаться, кнопки, формы и тестовая составляющая веб-сайта. А так же определяет, каким образом будут созданы элементы и их поведение.

В итоге на данном этапе мы получаем динамический прототип сайта, который и будет использоваться для разработки проекта.

В ходе разработке был придуман логотип сайта и его визитка(рисунок 3).

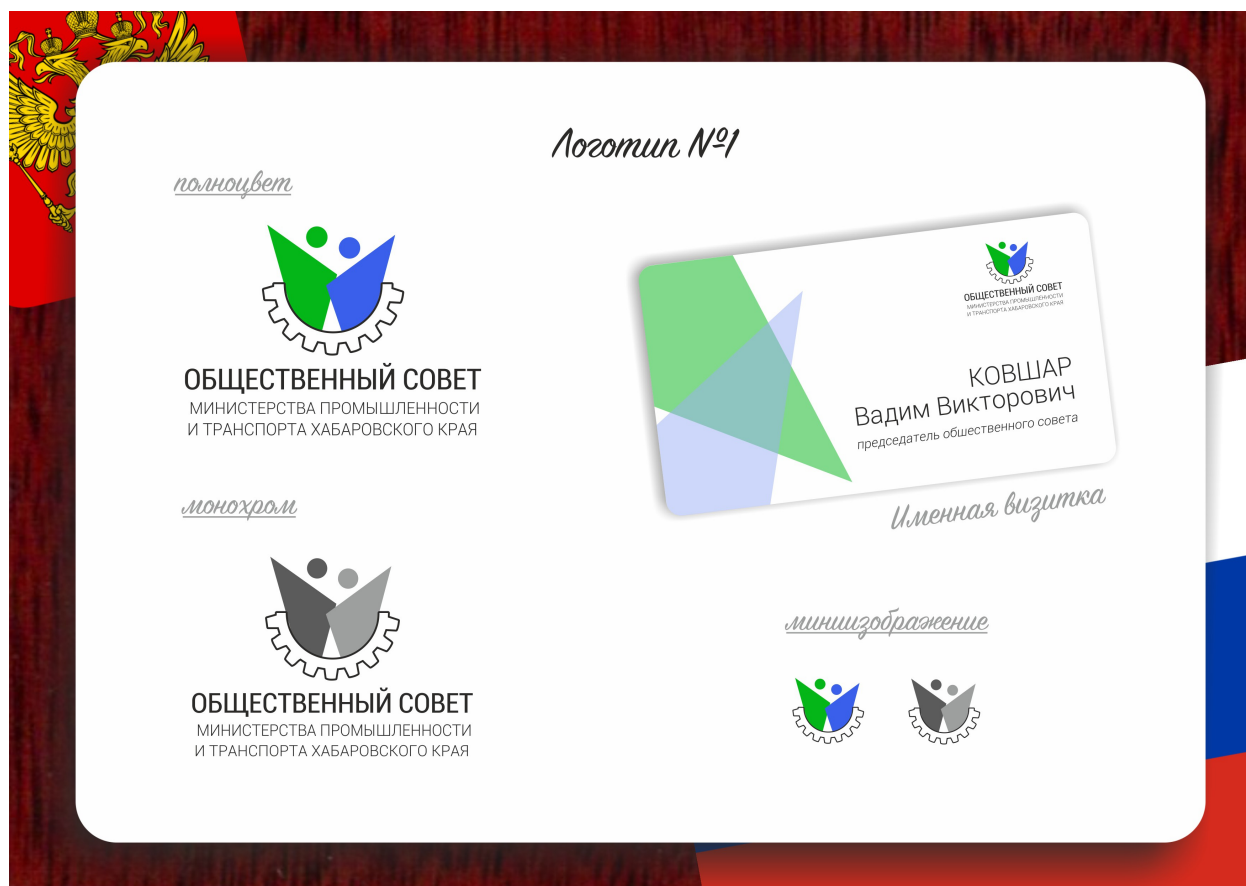


Рисунок 3 — Логотип и визитка сайта.

1.3 Front-end разработка

Front end— это абстракция, которая предоставляет пользовательский интерфейс.

Front-end разработка или программирование интерфейса на клиентском устройстве - один из значимых этапов всей разработки. Современные реалии таковы, что пользователи применяют множество устройств, взаимодействуя с интерфейсами, например: мобильные телефоны, планшеты, ноутбуки, телевизоры. здесь важно создать уникальный дизайн так, чтобы все смогли пользоваться сайтом без ошибок и задержек.

Для разработки на стороне клиента используются такие технологии как язык разметки HTML5, язык разметки CSS3 и сценарный язык программирования JavaScript. Они служат основой качественного и функционального дизайна, а так же удобство пользования сайта для пользователей. Так весь каркас сайта строиться на базовых тегах HTML5, а взаимодействия с элементами сайта проще всего обрабатывать через js скрипты.

1.4 Back-end разработка

Back end – Основная программно-аппаратная часть.

После того как веб-интерфейс был сверстан, необходимо создать серверную часть сайта. В данной части реализуется генерация и вывод необходимого контента из баз данных в нужные участки веб-страницы, заранее определенных на этапе проектирования интерфейса.

Для разработке серверной части чаще всего используются php, ruby, python, но можно использовать языки C#, C++ и тому подобные языки программирования.

Главной задачей серверной части является обработка запросов, работа с базами данных и генерация Html5 страниц. Так же можно писать отдельные скрипты обработки и загрузки файлов, текстов и управления сессиями.

2 ИСПОЛЬЗУЕМЫЕ ТЕХНОЛОГИИ ПРИ WEB-РАЗРАБОТКЕ

Для разработки качественного динамического интерфейса сайта на front end части необходимо использовать язык разметки HTML, в данном случае HTML5, таблицы каскадных стилей CSS3 и встраиваемый язык для программного доступа к объектам приложений JavaScript.

А для разработки Back end части можно использовать PHP , Perl или Ruby или другие языки программирования специализирующиеся на Web приложениях.

2.1 HTML5

HTML5- язык разметки для структурирования и представления содержимого веб- страниц.

Каждый HTML-документ, отвечающий спецификации HTML какой-либо версии, обязан начинаться со строки декларации версии HTML ! DOCTYPE, которая в версии HTML5 выглядит так:

```
<!DOCTYPE html>
```

Эта строка поможет браузеру определить, как правильно интерпретировать данный документ. В данном случае мы говорим браузеру, что HTML соответствует международной спецификации версии 5.

Полностью готовый базовый HTML-файл перед началом работы выглядит как определенный набор открывающихся тегов:

```
DOCTYPE html>  
<html>  
  
<head>
```

```
<meta http-equiv="Content-Type" content="text/html;
charset=UTF-8" />

<title>Заголовок Страницы </title>

</head>

<body>

Тело Страницы

</body>

</html>
```

Текст всего документа заключается в теги<html>, сам документ разбивается на две части - заголовок и тело. Заголовок описывается тегами <head>, в которые могут быть включены название документа (с помощью тегов <title>) и другие параметры, использующиеся браузером при отображении документа. Тело документа заключено в теги<body> и содержит собственно информацию, которую видит пользователь. При отсутствии тегов форматирования весь текст выводится в окно браузера сплошным потоком, переводы строк, пробелы и табуляции рассматриваются как пробельные символы, несколько пробельных символов, идущих подряд, заменяются на один.

2.2 CSS3

CSS3-каскадные таблицы стилей третьего поколения, Представляет собой формальный язык , реализованный с помощью языка разметки. Главной особенностью языка является возможность анимации объектов без использования JavaScript, а также поддержка линейных градиентов, сглаживания и многого другого

Зачастую каскадные таблицы стилей использ не только для определения шрифтов, фона и цвета компонент, но и для определения местоположения блоков при блочной верстке сайтов.

Хорошим правилом является вынесения css структуры в отдельный файл , а подключается он с помощью тега `<link >` и подключение выглядит так:

```
<link rel="stylesheet" type="text/css" href="name_file.css">
```

А сама структура файла состоит из классов, индикаторов.

2.3 JavaScript

JavaScript- сценарный язык программирования.

Основной плюсом JS является его работа с DOM (Document Object Model — «объектная модель документа») , что позволяет добавить интерактива в веб интерфейс, а так же подключения AJAX.

Вставка сценария JavaScript в HTML-документ начинается открывающим тегом `<SCRIPT>` и завершается закрывающим `</SCRIPT>`.

В связи с появлением еще одного скриптового языка VBScript рекомендуется использовать этот атрибут. Кроме того, обратите внимание на использование тегов комментария `<!--` и `-->`. Если страница, содержащая сценарий, будет загружена браузер, не поддерживающий языки сценариев, строки программы, засоряя экран, будут выведены как обычный текст. Если же использовать тег комментария, то такой браузер пропустит текст программы, воспринимая его как комментарий.

Каждый выбирает свой инструмент для создания Web-страниц. Это может быть MS FrontPage или Macromedia DreamWeaver, Allaire HomeSite или 1st Page 2000 или Notepad.

Текстовые редакторы, возможно, использовать только для создания небольших страниц, так как у них есть много минусов: не поддерживаются проекты, отсутствует "подсветка" текста, в общем, работать крайне неудобно.

Библиотека jQuery помогает легко получать доступ к любому элементу DOM, обращаться к атрибутам и содержимому элементов DOM, манипулировать ими. Также библиотека jQuery предоставляет удобный API для работы с AJAX.

jQuery - это JavaScript-библиотека, фокусирующая внимание на взаимодействии JavaScript, HTML и CSS.

jQuery позволяет:

- обращаться к любому элементу DOM (объектной модели документа) и не только обращаться, но и манипулировать ими;
- работать с событиями;
- легко осуществлять различные визуальные эффекты;
- работать с AJAX;
- имеет огромное количество JavaScript плагинов, предназначенных для создания элементов пользовательских интерфейсов.

2.4 Ruby on Rails

Для разработки Back end части будет использован фреймворк Ruby on Rails.

Ruby on Rails- фреймворк, написанный на языке программирования Ruby. Так же RoR представляет архитектуру MVC для веб- приложений , а так же обеспечивает их интеграцию с веб-сервером и сервером базы данных.

Так же RoR можно использовать для создания готовых аутентификации и готовых систем постинга и других готовых моделей.

Для генерации нового приложения нужно ввести в консоль команду `$ rails new name_app` которая автоматически создаст готовую структуру RoR приложение (Рисунок 2.1).

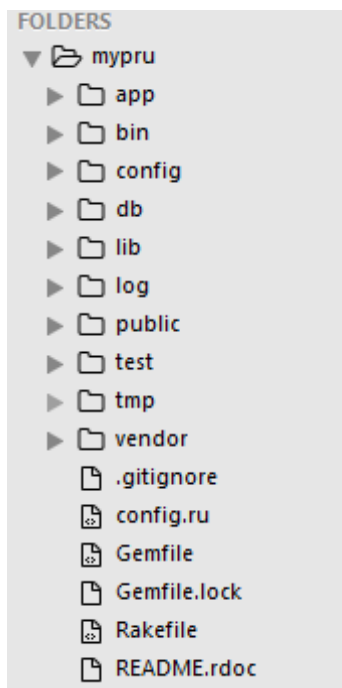


Рисунок 2.1 — структура RoR приложения.

Таблица 2.1

Файл/Директория	Назначение
<code>app/</code>	Основной код приложения (<code>app</code>), включает модели, представления, контроллеры и хелперы
<code>app/assets</code>	"Активы" приложения такие как каскадные таблицы стилей (CSS), JavaScript файлы и изображения
<code>bin/</code>	Бинарные исполняемые файлы
<code>config/</code>	Конфигурация приложения
<code>db/</code>	Файлы базы данных
<code>doc/</code>	Документация для приложения
<code>lib/</code>	Библиотека модулей
<code>lib/assets</code>	Библиотека "активов", таких как каскадные таблицы стилей (CSS), JavaScript файлы и изображения
<code>log/</code>	Файлы логов приложения
<code>public/</code>	Публично доступные данные (например, веб-браузерам), такие как страницы ошибок приложения
<code>bin/rails</code>	Программа для генерации кода, открытия консольных сессий, или запуска локального веб-сервера
<code>test/</code>	Тесты приложения (
<code>tmp/</code>	Временные файлы
<code>vendor/</code>	Код сторонних разработчиков, такой как плагины и геммы
<code>vendor/assets</code>	Сторонние "активы" такие как каскадные таблицы стилей (CSS), JavaScript файлы и изображения
<code>README.rdoc</code>	Краткое описание приложения
<code>Rakefile</code>	Служебные задачи, доступные посредством <code>rake</code> -команды
<code>Gemfile</code>	Геммы необходимые данному приложению
<code>Gemfile.lock</code>	Блокирующий список гемов, обеспечивающий использование всеми копиями приложения абсолютно одинаковых версий гемов
<code>config.ru</code>	Конфигурационный файл для Rack middleware

Начальный Html5 шаблон в RoR состоит из стандартных Html тегов и специальных Ruby вставок. А сам файл имеет двойное расширение Name.html.erb, что означает, что сначала будет обработан серверная чатсь рубина (ERB), и потом данные будут интегрированы в Html-документ(рисунок 5).

```
1 <!DOCTYPE html>
2 <html>
3   <head>
4     <title>Ruby on Rails Tutorial Sample App | <%= yield(:title) %></title>
5     <%= stylesheet_link_tag "application", media: "all",
6       "data-turbolinks-track" => true %>
7     <%= javascript_include_tag "application", "data-turbolinks-track" => true %>
8     <%= csrf_meta_tags %>
9     <!--[if lt IE 9]>
10      <script src="http://html5shim.googlecode.com/svn/trunk/html5.js"></script>
11    <![endif]-->
12  </head>
13  <body>
14    <header class="navbar navbar-fixed-top navbar-inverse">
15      <div class="navbar-inner">
16        <div class="container">
17          <%= link_to "sample app", '#', id: "logo" %>
18          <nav>
19            <ul class="nav pull-right">
20              <li><%= link_to "Home", '#' %></li>
21              <li><%= link_to "Help", '#' %></li>
22              <li><%= link_to "Sign in", '#' %></li>
23            </ul>
24          </nav>
25        </div>
26      </div>
27    </header>
28    <div class="container">
29      <%= yield %>
30    </div>
31  </body>
32 </html>
```

Рисунок 5 — Структура Html.Erb страницы.

Как видно из ресуна в теге `<%= yield %>` будет генерироваться тело документа из созданы файлов views представлений архитектуры модель-представление-контроллер (MVC). (рисунок 2.2)

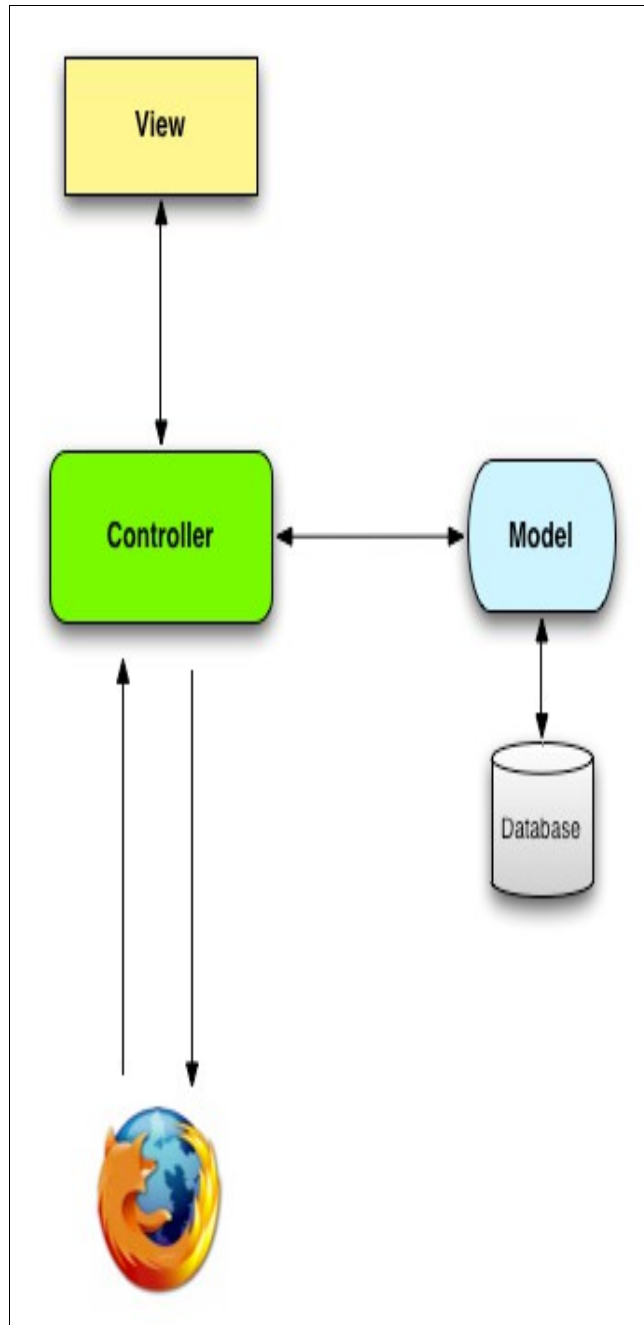


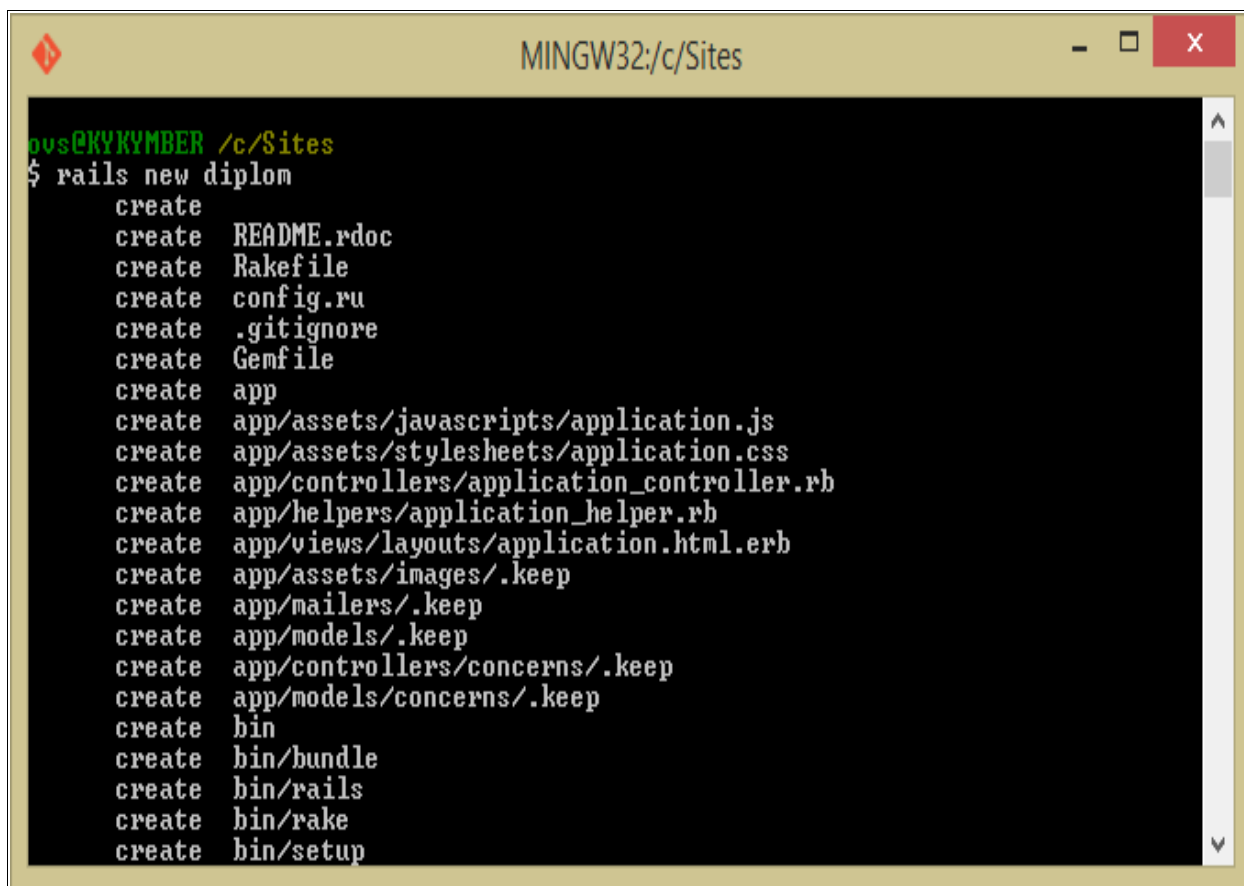
Рисунок 2.2— Схематичное изображение модель-представление-контроллер (MVC) архитектуры.

Данная структура работа шаблона позволяет создавать представления и модели для многих функций не меня структуры шаблона.

3 ПРОЕКТИРОВАНИЕ WEB-ИНТЕРФЕЙСА

3.1 Разработка RoR приложения

Как было сказано из раздела 2 для того , что бы создать новое приложение необходимо воспользоваться командой `rails new name_app` который автоматически создаст все нужные файлы. (рисунок 3.1)



```
ovs@KYKUMBER /c/Sites
$ rails new diplom
create
create  README.rdoc
create  Rakefile
create  config.ru
create  .gitignore
create  Gemfile
create  app
create  app/assets/javascripts/application.js
create  app/assets/stylesheets/application.css
create  app/controllers/application_controller.rb
create  app/helpers/application_helper.rb
create  app/views/layouts/application.html.erb
create  app/assets/images/.keep
create  app/mailers/.keep
create  app/models/.keep
create  app/controllers/concerns/.keep
create  app/models/concerns/.keep
create  bin
create  bin/bundle
create  bin/rails
create  bin/rake
create  bin/setup
```

Рисунок 3.1 – Новое RoR приложение

Для запуска сайта рейлс необходимо выполнить команду `Rails server` который автоматически запустит сервер для интерпретации языка Ruby. А сам сайт будет доступен по ссылке <http://localhost:3000/> (рисунок 3.2)

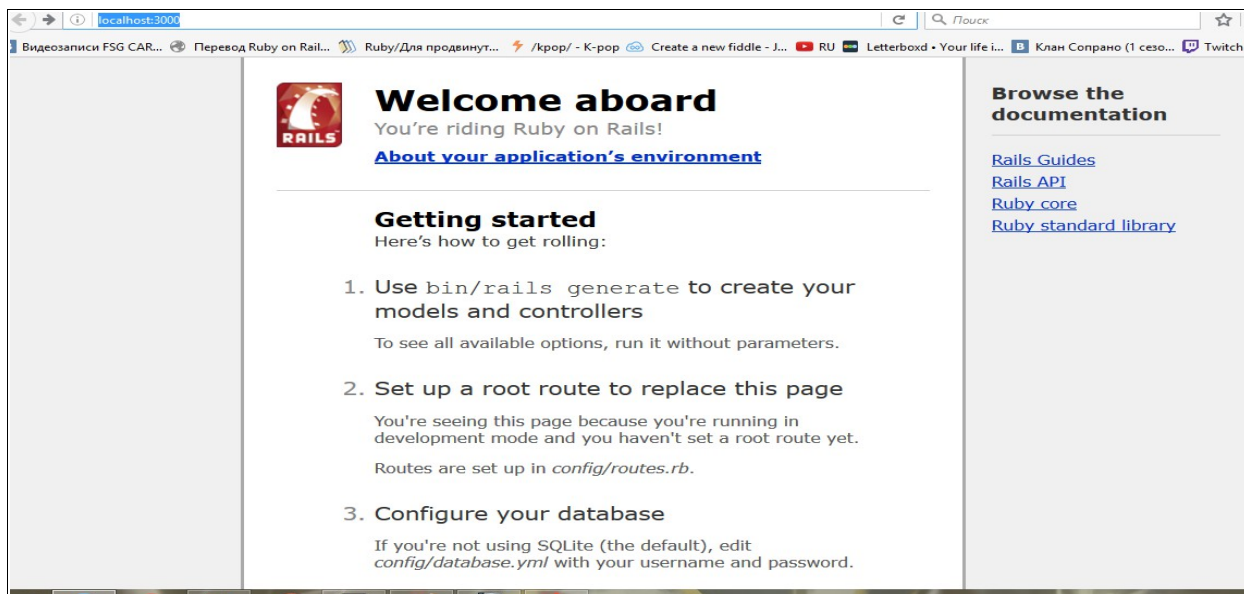


Рисунок 3.2 – Запуск перво приложения

После чего открыть проект в любом удобном текстовом редакторе таком как Notepad ++ или Sublime Text. Для начала необходимо редактировать шаблон нашего проекта, для этого нужно перейти в директиву `app/views/layout` и выбрать файл `application.html.erb` и прописать нужные поля шаблона такие как меню, названия сайта(рисунок 3.3).

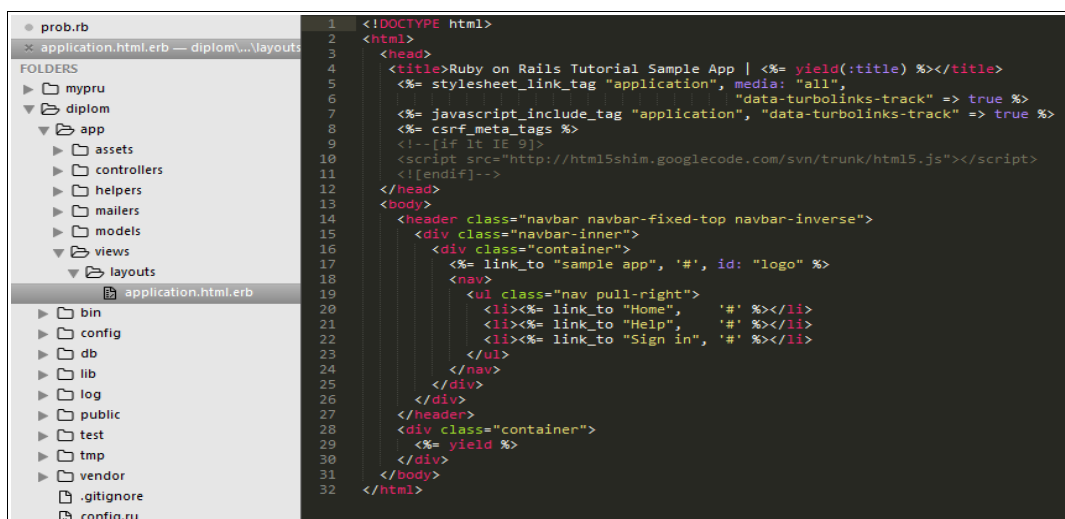


Рисунок 3.3 – application.html.erb

Для того что бы создать нужные представления или модели существует команда rails generate model/view которая, автоматически создает нужные файлы и пропишет к ним пути. Но так же существует технология scaffold с помощью которой можно создать нужные нам готовые реализации аутентификации, постинга и регистрации пользователей.

Так для создания регистрации пользователей с полями *name*, *email* *password* необходимо выполнить следующую команду:

```
rails generate scaffold User name:string email:string pwd: string
```

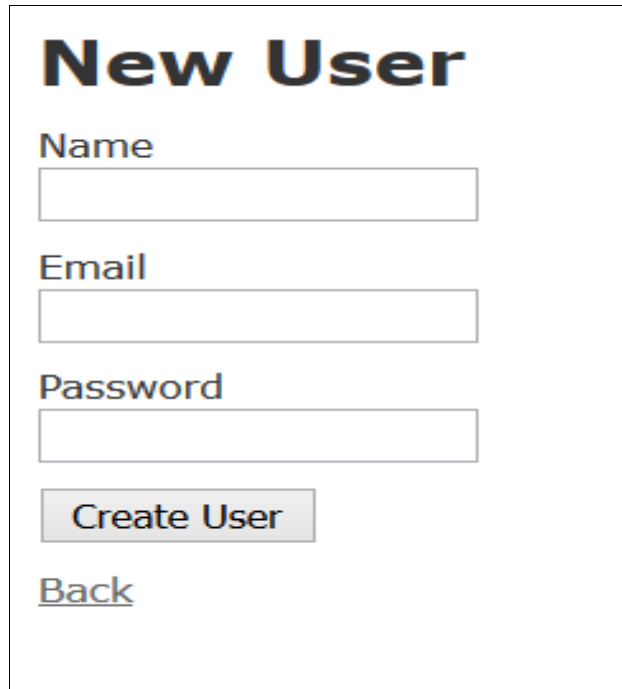
Так же нет надобности включать параметр для *id*; он создается Rails автоматически для использования в качестве *primary key* в базе данных.

Чтобы проект заработал мы сначала должны *migrate* (*мигрировать*, *переместить*) базу данных, используя *Rake* :

```
$ bundle exec rake db:migrate
```

Это функция просто обновляет базу данных с нашей новой моделью данных users.

После перехода по ссылке <http://localhost:3000/users/new> нам будет доступна форма регистрации пользователей (рисунок 3.4)



New User

Name

Email

Password

[Back](#)

Рисунок 3.4 – форма регистрации пользователей

Так же мы получили и другие представления и модели для пользователей (таблица 3.1)

Таблица 3.1

/users	index	страница, для отображения списка всех пользователей
/users/1	show	страница показывающая пользователя с id 1
/users/new	new	страница для создания нового пользователя
/users/1/edit	edit	страница для редактирования пользователя с id 1

Затем пробными командами мы создадим новостную ленту и последующие модель-представление-контроллер для нашего приложения.

Так для создания микростов необходимо выполнить следующую команду : *\$ rails generate scaffold Micropost content:string user_id:integer*.
Которая автоматически создаст ресурс микростов для сайта (рисунок 3.5).

Рисунок 3.5 – форма регистрации постинга на сайте

Как и с примером User для микропостов были созданы и другие представления (таблица 3.2)

Таблица 3.2

HTTP запрос	URL	Действие	Назначение
GET	/microposts	index	страница со списком всех микросообщений
GET	/microposts/1	show	страница показывающая микросообщение с id 1
GET	/microposts/new	new	страница для создания нового микросообщения
POST	/microposts	create	создание нового микросообщения
GET	/microposts/1/edit	edit	страница для редактирования микросообщения с id 1
PATCH	/microposts/1	update	обновление микросообщения с id 1
DELETE	/microposts/1	destroy	удаление микросообщения с id 1

3.2 Разработка шаблона сайта

Для разработки шаблона для начало необходимо разработать в Adobe photoshop готовое представления как будет выглядит сайт. Для начало необходимо выбрать размер нашего сайта, в нашем случае сайт будет разработан сеткой в ширину 960 пикселей. А так же разработать основные элементы сайта, иконки, логотипы, фон, меню, логотип, Необходимо

выбрать шрифты которые будут использоваться на сайте. Хорошим тоном является использования не более двух шрифтов, один для заголовков, а второй для основного текста.

После чего нанесением на каждый новый слой новых элементов сайта формируя полный каркас сайта , а так же и можем выбрать где и какой элемент больше всего подходит для удобства пользователя и смены цветов для гармоничного сочетания его.

В конечном счете мы получим готовый макет сайта(рисунок 3.2.1)



Рисунок 3.5 – Макет будущего сайта

Затем необходимо сверстать данный макет с использованием Html5 и Css3 технологий.

Как видно из макета у нас присутствуют два элемента навигации один элемента формы и четыре элемента контента. Так же необходимо

добавить Ajax технологию для того что бы страница не перезагружалась при выборе следующей новостей, а всего лишь догружались новые новости. После

После того как шаблон закончен необходимо выбрать часто встречаемые элементы и вынести их в файл *application.html.erb* , а элементы для других страниц в необходимые файлы представления.

3.3 Проектирование базы данных для сайта

При разработки Web-приложения не обойтись и без хранения информации о пользователях, новостях, документов и прочего. Проблемы хранения данных позволяют решить реляционные базы данных, таких как: MySQL, NoSql. SQLite

Для того что бы создать базу данных для начала необходимо выбрать нужные таблицы и их элементы которые будут храниться в базе данных. Так для создания хранения пользователь нам достаточно хранить *id*, *name*, *email*, *password* пользователей (рисунок 3.3.1).

users	
id	integer
name	string
email	string

Рисунок 3.3.1 – Модель данных для пользователей

Для модели данных микростов нам необходимо хранить информацию о номер поста *id* , текст поста *content* , а так же номер пользователя который добавил данный пост *user_id* (рисунок 3.3.2)

microposts	
id	integer
content	string
user_id	integer

Рисунок 3.3.2 – Модель данных для микросообщений

Так как пользователей может иметь много сообщений необходимо связать микросообщения и пользователей. Для того что бы это сделать нам необходимо перейти в директорию `app/models/user.rb` и добавить в модель `User` следующий код:

```
class User < ActiveRecord::Base
  has_many :microposts
end
```

Данный код означает что у `У` пользователя есть много микросообщений .

Теперь необходимо сказать, что приложению, что Микросообщения принадлежат пользователю. Для этого нужно в модель `Micropost` в директории `app/models/micropost.rb` добавить следующий код:

```
class Micropost < ActiveRecord::Base

  belongs_to :user

  validates :content, length: { maximum: 140 }
end
```

Мы можем увидеть результат применения этой ассоциации. Из-за `user_id` столбца в таблице `microposts` Rails (используя `Active Record`) может вывести микросообщения, связанные с каждым пользователем. (рисунок 3.3.3).

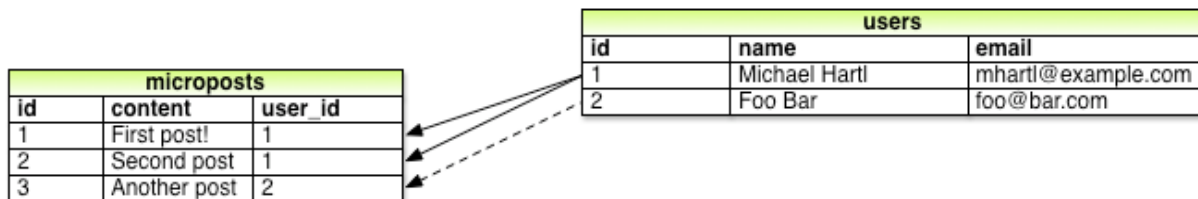


Рисунок 3.3.3 – Связь между микросообщениями и пользователями

Так же создаем и связываем остальные элементы сайта в которых необходимо хранить информацию в базе данных. По итогу у нас получится массивная структура базы данных с десятками таблиц связанных между собой разными структурами.

ЗАКЛЮЧЕНИЕ

В заключение курсовой работы можно сказать, что в разработанном веб-интерфейсе создан отличный дизайн приложения, а так же реализована полная функциональность для сайт общественного совета министерства промышленности и транспорта Хабаровского края.

Использования данного сайта помогает людям узнать о совете, запланированных мероприятиях, а так же прочесть новости и оставить свой отзыв на сайте. А для работников совета разработан личный кабинет, где у каждого работника есть свой личный график занятий и обязанностей, а так же форма размещения новостей или мероприятий.

Данная работа имеет практическое значение, поскольку решает задачи работы общественного совета министерства промышленности и транспорта Хабаровского края, а так же по тому , что при решении данной задачи использованы технологии, которые необходимы в сфере информационных технологий, в частности, сфере web-программирования, которая в настоящее время является одной из наиболее перспективной.

Цель данной работы выполнена, задачи выполнены. Практическое выполнение данной работы было рассмотрено, проект готовится к реализации.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Петюшкин А. В.html Экспресс - курс / Петюшкин А.В. - СПб: БХВ-Петербург, 2003, - 256 с. ил.
2. Леонтьев Б.В. Web-Дизайн: Тонкости, хитрости и секреты / Леонтьев Б.В. - М. Майор, 2001, с.170.
3. Безруков Н.Н. Компьютерные вирусы. - М.: Наука, 2004 г
4. Титоренко Г.А. Информационные технологии управления. - М.: ЮНИТИ-ДАНА, 2003 г.
5. Калина А.В. Организация и оплата труда в условиях рынка (аспект эффективности): Учебно-методическое пособие. - К.: МАУП, 1997. - 300 с
6. Торрес Р. Дж. Практическое руководство по проектированию и разработке пользовательского интерфейса. - СПб: Вильямс, 2002. - 400 с.
7. Кушнарев Л.И., Хицков Е.А., Гальчич М.А. Методические рекомендации по дипломному проектированию. - М.: ФГОУ ВПО МГАУ. 2005. - 114 с.
8. Крамер, Э. HTML: наглядный курс Web-дизайна / Э. Крамер. - Киев: Диалектика, 2001. - 304 с.: ил.
9. Леонтьев, Б. Web-дизайн. Руководство пользователя / Б. Леонтьев. - Киев: 2001. - 384 с, ил.
10. Роббинс, Д. Web-дизайн. Справочник / Д. Роббинс. - "КУДИЦ-ПРЕСС", 2008. - 816 с.

11. Едомский, Ю. Е. Техника Web-дизайна для студента / Ю. Е. Едомский. - СПб: БХВ-Петербург, 2005. - 491 с.
12. Дуванов, А. А. Web-конструирование. HTML / А. А. Дуванов. - СПб: БХВ-Петербург, 2003. - 384 с.
13. Печников, В. Н. Создание Web-страниц и Web-сайтов / В. Н. Печников. - М.; изд. Триумф, 2006. - 370 с.
14. Смирнова, И. Е. Начала web-дизайна / И. Е. Смирнова - СПб: БХВ-Петербург, 2003. - 491 с.
15. Negnevitsky M. Artificial intelligence: a guide to intelligent systems. Addison-Wesley, 2002
16. Выпускная квалификационная работа. Общие требования и правила оформления : метод. пособие. / сост. В.Н. Гопкало, О.А. Графский. – Хабаровск : Изд-во ДВГУПС, 2014. – 44 с. : ил.
- 17.. Исследовано в России [Электронный ресурс]: многопредмет. науч. журн. / Моск. физ.-техн. ин-т. – Электрон. журн. – Долгопрудный: МФТИ, 1998. -. – режим доступа к журн.: <http://zhurnul.milt.rissi.ru>. (дата обращения: 26.02.2016)
- 18.. Розенберг Д. М. Бизнес и менеджмент. Терминологический словарь. - М.: ИНФРА-М, 1997. -464 с.
19. Гольдман И. А., Добробабенко Н. С. Практика рекламы. - Новосибирск, 1991. -141с. Менеджмент и рынок: германская модель: Учебное пособие. Под ред. С. Долгова. - М.: Бек, 1995. - 480 с.

20. Официальные периодические издания: электронный путеводитель / Рос. нац. б-ка, Центр правовой информации. [СПб], 200520076. URL: <http://www.nlr.ru/lawcenter/izd/index.html> (дата обращения: 18.02.2016)
21. Райзберг Б.А. Современный экономический словарь / Б.А. Райзберг, Л.Ш. Лозовский, Е.Б. Стародубцева. – 5-е изд., перераб. и доп. – М.: ИНФРА-М, 2006. – 494 с.
22. Crawford P.J. The reference librarian and the business professor: a strategic alliance that works / P.J. Crawford, T.P. Barrett // Ref. Libr. – 1997. Vol. 3. № 58. – P.75–85.
- 23.. Костромин В. А. Конспект вебмастера. Выбор системы управления содержанием сайта (контентом) // Справочник вебмастера. – 2009-2013.
24. Лабуренко Е. О. Школьный сайт: создание, наполнение и привлечение посетителей / Е. О. Лабуренко, Е. В. Якушина // Народное образование. – 2012. – № 4. – С. 171-181.
25. Маркелов А. О. Разработка Интернет-ресурса “Гид первокурсника института математики, физики и информатики” средствами CMS JOOMLA // Гаудеамус. – 2011. – Т. 2, № 18. – С. 137-138.
26. Норт Б. М. Joomla! практическое руководство / Б. М. Норт; [пер. с англ. А. Киселева]. – СПб: М.: Символ: Символ-Плюс, 2008. – 448 с.
27. Рамел Д. Самоучитель Joomla! / Д. Рамел; [пер. с англ. Д. Колисниченко]. – СПб: БХВ-Петербург, 2008. – 448 с.
28. Севердиа Р. Joomla. Создание сайтов без программирования: [пер. с англ.] / Р. Севердиа, К. Краудер. – М.: Эксмо, 2011. – 382 с.

29. Сычев И. А. Создание сайтов на основе систем управления контентом: электрон. учеб. -метод. пособие / И. А. Сычев, В. Н. Половников. – Бийск: АГАО, 2012.
30. HTML. Просто как дважды два. – М.: Изд-во Эксмо. 2006. - 256с., ил.- (Просто как дважды два).
31. Соглашаемся на некачественный код// Сайт blogspot.ru 3 февраля 2008 г. (<http://denismiller.blogspot.ru/2008/02/blog-post.html>) (Дата обращения 05.03.2016)
32. Системы балансировки нагрузки Web-серверов// Сайт citforum.ru (<http://citforum.ru/internet/webserver/webserverbal.shtml>) (Дата обращения 11.03.2016)
- 33.33 Мержанова М. Уроки Web-мастерства. Урок3 // Мир ПК, № 04.2003.