АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ

Факультет: Информационные технологии и управление

Кафедра: Компьютерная инженерия

Предмет: «Веб-системы и технологии»

Курсовая работа

**Тема:** «Разработка онлайн информационной системы для вокзалов.»

**Группа:** 680.22

**Курc:** 3

**Специальность:** Информационные технологии

**Студент**: Абасова Медина

**Руководитель:** **асс. Халилов М. Э.**

**Зав.кафедрой: доц.Рагимова Н.А.**

**Баку – 2025**

АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ

АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ

ЗАДАНИЕ ПО КУРСОВОЙ РАБОТЕ

**ФАКУЛЬТЕТ**: Информационные технологии и управление

**КАФЕДРА**: Информационные Технологии

**Группа** 680.22 **курс** III

**Специальность** Информационные технологии

**Студент** Асланлы Рашад

**Зачетная книжка**

**Руководитель курсовой работы**  **асс. Халилов М. Э.**

**Срок выдачи**

**Дата сдачи**

**Тема курсовой работы** «Разработка онлайн информационной системы

для вокзалов»

**Отзыв руководителя курсовой**:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Подпись студента**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Подпись рук.курсовой**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Подпись зав.кафедрой**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

**Дата защиты курсовой**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**Оценка**\_\_\_\_\_\_\_\_\_\_

**Пред.комиссии**\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

**Члены комиссии**: 1.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

2.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

3.\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_(\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_)

Оглавление

[АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ 1](#_Toc196660547)

[Факультет: Информационные технологии и управление 1](#_Toc196660548)

[Кафедра: Компьютерная инженерия 1](#_Toc196660549)

[Предмет: «Веб-системы и технологии» 1](#_Toc196660550)

[Курсовая работа 1](#_Toc196660551)

[АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ 2](#_Toc196660552)

[АЗЕРБАЙДЖАНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ НЕФТИ И ПРОМЫШЛЕННОСТИ 2](#_Toc196660553)

[ЗАДАНИЕ ПО КУРСОВОЙ РАБОТЕ 2](#_Toc196660554)

[ВВЕДЕНИЕ 5](#_Toc196660555)

[ТЕОРЕТИЧЕСКАЯ ЧАСТЬ 6](#_Toc196660556)

[1.1 Web-программирование 6](#_Toc196660557)

[1.2 HTML и XHTML 8](#_Toc196660558)

[1.3 Dynamic HTML 14](#_Toc196660559)

[1.3.1 Каскадные таблицы стилей 14](#_Toc196660560)

[1.3.2 JavaScript 17](#_Toc196660561)

[1.3.3 Bootstrap 18](#_Toc196660562)

[2. ПРАКТИЧЕСКАЯ ЧАСТЬ 21](#_Toc196660563)

[2.1 Описание предметной области 21](#_Toc196660564)

[2.2 Создание окружения для проекта 22](#_Toc196660565)

[2.3 Верстка HTML страницы 22](#_Toc196660566)

[2.4 Использование стилей CSS 25](#_Toc196660567)

[3. РЕЗУЛЬТАТ 32](#_Toc196660568)

[4. ЛИТЕРАТУРА 32](#_Toc196660569)

[5. ПРИЛОЖЕНИЯ 33](#_Toc196660570)

ВВЕДЕНИЕ

Веб-системы и технологии относятся к различным инструментам, языкам и платформам, которые используются для разработки, развертывания и обслуживания веб-приложений. Эти технологии развивались на протяжении многих лет, чтобы не отставать от растущего спроса на веб-сервисы, что привело к появлению разнообразных языков программирования, фреймворков, библиотек и баз данных. Когда появились Web-технологии, пользоваться компьютерной техникой начали абсолютно новые категории граждан. Список социальных категорий населения, подсоединяющихся к сети интернет с целью поиска информации во всемирной паутине, постоянно увеличивается за счёт людей, которые не считают себя специалистами в сфере информационных технологий. Web-технология в корне изменила понятия об информационной обработке, как и работе с компьютерной техникой тоже.

Оказывается стандартные характеристики, описывающие прогресс в сфере вычислительного оборудования, а именно, производительность, пропускная способность, объём памяти, практически не включали в свой состав главное системное «узкое место», именуемое пользовательским интерфейсом. Устаревшая техника взаимоотношений пользователя с информационной системой тормозила продвижение передовых технологий и нивелировала преимущества от их внедрения. И лишь когда интерфейс пользователя, обеспечивающий взаимодействие человека и компьютера, разработчики довели до простого и естественного понимания обычными людьми, произошёл невообразимый взрывной рост интереса к достоинствам и возможностям компьютерного оборудования.

Web-технологиями является весь набор средств, позволяющих организовать WWW (World Wide Web), то есть всемирную паутину. Так как каждый сеанс является взаимодействием двух сторон, а именно, сервера и клиента, то и Web-технологии делятся на следующие группы:

1. Технологии серверной стороны (server-side).
2. Технологии клиентской стороны (client-side).

Технологии клиентской стороны включают в свой состав весь набор технологий по созданию веб-страниц (HTML, JavaScript, DHTML и другие), а технологии серверной стороны состоят из технологий доступа к информационным базам данных в сети интернет (CGI, PHP).

Серверные программы обеспечивают предоставление тех или иных ресурсов клиентским программам. Клиенты, когда им требуется какой-либо файл или просто какая-то информация от сервера, вырабатывают специальный запрос клиента и отправляют его серверу. Серверная программа выполняет обработку запроса и отправляет ответ сервера, который содержит запрошенную информацию или же извещение об ошибке, в случае недоступности требуемых данных. Данная компьютерная организация, или по другому, принципы формирования вычислительных систем или сетей, именуется архитектурой «клиент-сервер» или двухзвенной организацией. Как раз на базе двухзвенной архитектуры работают практически все интернет - сервисы, включая и WWW.

Данная курсовая работа рассматривает пример создания информационной системы для продуктовых магазинов.  
Разработка онлайн-информационной системы будет включать в себя следующие этапы:

1. Сбор требований: Первым шагом будет сбор требований от клиента, таких как продукты, которые он хочет продавать, целевая аудитория, функции, которые он хочет видеть на веб-сайте.

2. Дизайн: Следующим шагом будет разработка дизайна веб-сайта, включая создание каркасов и макетов, выбор цветовых схем и шрифтов, а также определение общего макета и пользовательского интерфейса.

3. Разработка: Это предполагает создание веб-сайта с использованием HTML, CSS, JavaScript и других веб-технологий.

4. Тестирование: После завершения разработки веб-сайт необходимо будет протестировать, чтобы убедиться, что он должным образом функционирует в различных браузерах и устройствах.

WEB-сайт Интернет-магазина – это система, ориентированная большей частью на покупателя. Он должен предоставлять сервисы просмотра товара, удобного поиска товара, легкость и удобство при заказе товара.   
 И так, целью моей курсовой работы является создание сайта Интернет-магазина, обеспечивающего удобство, безопасность и интуитивную понятность интерфейса пользователю.

ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

# 1.1 Web-программирование

Web-программирование (Web-разработка) - это бурно развивающийся раздел программирования, ориентированный на разработку динамических Интернет-приложений. Языки Web-программирования делятся на две группы: клиентские и серверные. Клиентские языки обрабатываются на стороне пользователя (в основном в браузере). Соответственно, обработка скрипта зависит от браузера пользователя, и пользователь имеет полномочия настроить свой браузер так, чтобы тот вообще игнорировал скрипты. При этом если браузер старый, он может не поддерживать тот или иной язык или версию языка, на которую опирался разработчик. С современными браузерами таких проблем возникать не должно, к тому же языки программирования не так уж часто кардинально обновляются (раз в несколько лет) и лучшие из них давно известны. Код клиентского скрипта может посмотреть каждый, выбрав в меню своего браузера «Исходный код страницы». Преимущество клиентского языка заключается в том, что обработка скриптов на таком языке может выполняться без отправки документа на сервер. Программа сразу проверит правильное заполнение формы перед отправкой, и, если необходимо, выведет ошибку. Отсюда же вытекает и то ограничение, что с помощью клиентского языка программирования ничто не может быть записано на сервер. Самым распространенным из клиентских языков признан JavaScript, разработчиками которого является компания Netscape совместно с компанией Sun Microsystems. Еще один популярный язык - это VBScript. Помимо этого в последнее время набрали популярность такие технологии, как AJAX, Adobe Flash, Microsoft Silverlight и др. Серверные языки программирования открывают перед программистом большие просторы для деятельности. Когда пользователь делает запрос на какую-либо страницу (переходит на нее по ссылке, или вводит адрес в адресной строке своего браузера), то вызванная страница сначала обрабатывается на сервере (то есть выполняются все программы, связанные со страницей) и только потом возвращается к посетителю по сети в виде файла. Этот файл может иметь расширения: HTML, PHP, ASP, Perl, SSI, XML, DHTML, XHTML.

# 1.2 HTML и XHTML

HyperTextMarkupLanguage (HTML) - это язык разметки документов во Всемирной паутине, принятый за стандартный. Большая доля всех Web-страниц в Интернете создана при помощи языка HTML (или XHTML), поэтому мы рассмотрим его подробно.

Язык HTML позволяет форматировать текст и другие элемента Web-страницы:

*Цвет, жирность, стиль, название шрифта* для текста.

*Позволяет выделять фрагменты и символы* например: ударение в слове, заголовок страницы или абзаца, сам абзац, пункт списка.

*Гипертекстовые ссылки*, позволяют переходить между документами и между фрагментами одного документа.

*Формы* для введения данных, как правило, данные из форм обрабатываются с помощью скриптов на языках программирования, ориентированных на Web,например PHP.

*Отображение мультимедийных файлов*, их может отображать сам браузер - изображения, аудиофайлы, или внешние приложения, взаимодействующие с браузером, например Flash-ролики, Java-апплеты.

HTML - язык разметки документов основанный на тэгах. Документ на языке HTML представляет собой набор элементов, при этом начало и конец каждого элемента обозначается служебными символами - тегами. Все тэги HTML начинаются с «<« (левой угловой скобки) и заканчиваются символом «>« (правой угловой скобки). Завершающий тег выглядит также, как начальный, и отличается от него прямым слэшем перед текстом внутри угловых скобок.

**<HTML></HTML>**

HTML регистронезависимый язык, теги могут быть написаны как строчными, так и заглавными буквами (в отличие от XHTML). Теги могут быть вложенными друг в друга.

<**HTML>**

**<HEAD>**

**<TITLE>**

Заголовок страницы

**</TITLE>**

**</HEAD>**

**</HTML>**

Тэги могут быть пустыми, то есть не содержать текста или других вложенных конструкций (например, <**br>**который переводит строку). Закрывающий тег в таком случае не указывается.

Также, элементы разметки могут иметь атрибуты, задающие их свойства (например, размер шрифта, цвет, расположение). Атрибуты задаются в начале тега.

**<ahref=«**http://www.yandex.ru**»>**Пример элемента с атрибутом **href.</a>**

Теги можно разделить на следующие группы.

*Гиперссылки*

**<AHREF=«**filename**»target=«\_self»>**текст ссылки**</A>**

где filename - имя файла (может быть и локальным) или адрес страницы вInternet, на который нужно совершить переход.

текст ссылки - текст гипертекстовой ссылки, который будет отображаться в браузере, как правило, выделяется подчеркиванием.

target - задает окно или фрейм, в котором будет открыт документ, при переходе по ссылке. Он может принимать значения:

\_top - документ откроется в текущем окне

\_blank - документ откроется в новом окне

\_self - документ откроется в текущем фрейме

\_parent - документ откроется в родительском фрейме

По умолчанию принимает значение \_self.

*Текстовые ссылки.*

**<H1></H1>**,**<H2></H2>**, … ,**<H6></H6>** - заголовки 1-6 уровней. Применяются для выделения частей выводимого текста (заголовок 1 - будет выведен очень большим, 6 - будет размером сопоставимым с обычным текстом).

**<P>** - обозначает начало нового абзаца. Закрывающий тег**</P>**, не является необязательным.

**<BR>** - переход на новую строку. Закрывающий тег**</br>**, отсутствует.

**<HR>** - горизонтальная линия.

**<BLOCKQUOTE></BLOCKQUOTE>** - цитата. Выделение заданного текста как цитаты.

**<PRE></PRE>** - режим предпросмотра. При этом текст заключается в рамку и выводиться не форматированным (то есть все теги, кроме </PRE>, игнорируются, но при этом переводы строки ставятся там, где они присутствуют в исходном документе).

**<DIV></DIV>** - блок текста (как правило, применяется для использования каскадных стилей CSS).

**<SPAN></SPAN>** - строка (как правило, применяется для использования каскадных стилей CSS).

*Теги форматирования текста*

**<EM></EM>** - выделение символа, на который падает ударение (обычно отображается курсивом).

**<STRONG></STRONG>** - выделение символа, на который падает усиленное ударение (обычно отображается жирным текстом).

**<I></I>** - выделение текста курсивом.

**<B></B>** - выделение текста жирным шрифтом.

**<U></U>** - подчёркивание текста

**<S></S>** - зачёркивание текста.

**<STRIKE></STRIKE>** - то же самое, что <S> … </S>

**<BIG></BIG>** - увеличение шрифта.

**<SMALL></SMALL>** - уменьшение шрифта.

**<BLINK></BLINK>** - мигающий текст.

**<MARQUEE></MARQUEE>** - сдвигающийся по экрану текст.

**<SUB></SUB>** - вывод текста под строкой. Например, **H<SUB>2</SUB> О**тобразиться в виде текста H2O.

**<SUP></SUP>** - вывод текста над строкой. Например, **E=mc<SUP>2</SUP>**отобразиться в виде текста E=mc2.

**<FONT атрибуты></FONT>** - задание атрибутов у используемого шрифта. Атрибуты могут быть следующими:

**COLOR=color** - указание цвета. Цвет может быть указан шестнадцатеричным числом в формате #rrggbb (первые 2 шестнадцатеричные цифры указывают интенсивность красного, следующие 2 - зелёного, последние 2 - синего) или названием самого цвета.

FACE= указываем имя шрифта.

SIZE= позволяет изменить размер шрифта. Размеры могут быть от 1 до 7, по умолчанию размер 3.

SIZE=+ размер или SIZE=-размер - размер больше или меньше стандартного. Например,SIZE=+2 указывает размер на 2 больше стандарта, то есть размер 5.

*Списки.*

Данная конструкция

**<UL>**

<LI> первый элемент списка </LI>

<LI> второй элемент списка </LI>

<LI> третий элемент списка </LI>

</UL>

создаёт список вида:

- первый элемент;

- второй элемент;

- третий элемент.

Также стоит отметить, что тегов есть параметры, позволяющие менять вид списка.

*Объекты.*

EMBED - вставка объектов различных типов

APPLET - вставка Java-апплетов

SCRIPT - вставка различных скриптов, например JavaScript

*Изображения.*

IMG - тег для вставки изображения. Это не закрывающийся тег.

SRC - имя локального файла или путь к нему в виде URL

ALT - текст картинки (отобразится, в виде текста, если не удалось отобразить картинку)

TITLE - подсказка (показывается при попадании курсора в область картинки)

WIDTH, HEIGHT - размеры изображения (выводимое изображение будет масштабировано до указанных размеров)

ALIGN - обтекание текста

*Таблицы.*

TABLE - тег создание таблицы. Тег имеет следующие параметры:

BORDER - задает толщину границу таблицы

CELLSPACING - задает расстояние от ячейки до ячейки

CAPTION - задает заголовок таблицы (необязательный тег)

TR - добавление строки в таблицы

TH - задает заголовок столбца (необязательный тег)

TD - добавление ячейки таблицы

WIDTH, HEIGHT - размеры таблицы

*Формы.*Формы ввода данных могут быть самыми разнообразными. Поэтому рассмотрим только основные теги:

FORM - тег для создания формы

INPUT - добавление элемента ввода

TEXTAREA - добавление текстового поля

SELECT - добавление списка (как правило, это выпадающее меню)

OPTION - пункт списка

*Символы.*Некоторые символы не могут быть выведены напрямую. Для их вывода требуется использовать их определения, например, символ апмерсанд& в коде HTML будет иметь вид **&amp**, символ меньше < будет иметь вид **&lt**, символ больше > будет **&gt**. Это ограничение введено, так как эти символы уже используются в языке HTML как служебные.

Любая HTML-страница должна иметь обозначение начала и конца документа обрамленные тегами**<html>** и **</html>** соответственно. Внутри них должны находиться теги заголовка **<head>** и **</head>**, итеги, обозначающие тело документа **<body>** и **</body>**. А внутри них могут быть произвольные комбинации из групп тегов описанных ранее.

Также рассмотрим ExtensibleHypertextMarkupLanguage (XHTML) это расширяемый язык разметки гипертекста. Стоит отметить, что язык XHTML это ни описание самого языка, а список отличий XHTML от HTML. Рассмотрим основные отличия HTML и XHTML. В XHTML все используемые теги должны иметь закрывающий тег. Теги, не имеющие закрывающего тега должны оканчиваться символом /. Например тег **<br>**, должен иметь закрывающий его тег **<br />**. В XHTML допускается писать теги и их атрибуты только строчными буквами. В XHTML очень строгая проверка синтаксиса не допускается использовать< и &, даже в URL, вместо них должны быть **&lt** и **&amp**. Браузеры, обнаружив ошибку синтаксиса в XHTML, должны прекратить его обработку и вывести ошибку на экран. В стандарте HTML браузер должен попытаться отобразить запрашиваемый документ. Стоит отметить, что XHTML расширяемый язык - за счет указания типа документа и возможности использовать свои теги.

Как мы видим, язык разметки HTML предоставляет широкие возможности для отображения информации, для этого в нем содержится большое количество тегов для различного форматирования выводимой информации. Язык XHTML очень похож на HTML, но более строгий, грамматические правила в XHTML менее сложные, и как следствие при созданий Web-страниц будет меньше ошибок.

# 1.3 Dynamic HTML

Dynamic HTML или DHTML - так принято называть связку языка HTML, каскадных таблиц стилей, скриптового языка и объектной модели документов. Скриптовым языком может выступать JavaScript или VisualBasic, но именно первый язык получил большую популярность и сегодня используется повсеместно. При помощи DHTML можно создавать интерактивные Web страницы, он позволяет легко и гибко обрабатывать данные запроса и формировать динамический ответ. DHTML может быть использован для реализации интерфейса Drag'n'Drop. На его основе создаются игры и другие интерактивные сервисы. Стоит отметить, что для реализации принципов DHTML достаточно лишь браузера, который будет обрабатывать содержимое страницы. То есть, нет необходимости, например, в обращениях к базе данных.

## 1.3.1 Каскадные таблицы стилей

Рассмотрим подробнее каскадные таблицы стилей - CSS (CascadingStyleSheets). Это стандарт позволяющий задавать описание внешнего вида некоторых элементов страницы на HTML. CSS используется при создании Web-страниц для определения шрифта, цвета, расположения и прочих атрибутов, используемых в документе, элементов.

Основная цель использования CSS разграничить само содержимое страницы, созданное на языке HTML и описание оформления, которое написано на CSS. Такое разделение упрощает создание и изменение документа, документом более легко управлять, а также снижается избыточность в исходном коде. Также, при помощи CSS можно легко представить один и тот же документ, но с разным форматированием или использовать разные методы вывода, такие как отображение на экране, печать документа, чтение голосом и тому подобных.

Описание стиля на CSS при открытии страницы может быть взято из разных мест: оно может быть встроенным стилем - это блоки CSS внутри страницы на языке HTML.

**<style type=«**text/css**»>**

**body {**

**color:**green;

**}**

**</style>**

Из отдельного файла .css, в котором описывается таблица стиля, на этот файл делается ссылка внутри страницы:

**<link rel=«**stylesheet**» type=«**text/css**» href=«**style.css**» />**

Это может быть Inline-стиль, при этом в HTML документе информация о стиле элемента указывается как атрибут style.

**<p style=«**font-size: 21px; color: green;**»>**Выводимый текст**</p>**

Также в любом браузере есть свой стандартный стиль, используемый по умолчанию.

Таблица стилей это набор описаний форматов. Каждое правило стиля, в свою очередь, имеет один или несколько селекторов, которые разделяются запятыми. В любом стиле также должен присутствовать блок определений, который обозначается фигурными скобками { }, и включает в себя набор свойств и их значений.

селектор, селектор {

свойство1: значение;

свойство2: значение;

свойство3: значение;

}

В стандарте CSS присутствуют приоритеты, согласно которым применяются правила стилей, например, если для элемента подходят свойства сразу нескольких правил. Это и называется каскадом. Стоит обратить внимание и на порядок расположения применяемых свойств - у свойства, указанного позже, будет более высокий приоритет.

До создания CSS оформление стиля Web-страницы производилось непосредственно внутри самого содержимого документа. Однако с использованием CSS появилась возможность принципиально разделить содержания и описание стиля документа. Благодаря этому стало возможным лёгкое применение единого стиля при создании похожих документов, а также это оформления можно изменить легко и быстро. Появилась возможность создать несколько вариантов дизайна страницы для разных устройств, на котором ее можно просмотреть. Например, на мониторе дизайн страницы будет иметь большую ширину, при отправке документа на печать буду отбрасываться такие элементы как меню, а на КПК или сотовом будет масштабирование уменьшающее искажение текста страницы. За счет того, что таблицы стилей хранятся в отдельном файле CSS, уменьшается время загрузки страницы. В этом случае браузер получает только описание структуры документа и данные, находящиеся на Web-странице, а стиль оформления этих данных загружается браузером при первом обращении и потом берется из КЭШа. Последующее изменения дизайна сайта так же значительно упрощается. Нет необходимости изменять каждую страницу, лишь нужно внести изменения в CSS-файл. Но у CSS есть также и недостатки. Различное отображение вёрстки в различных браузерах, которые по-разному отображают одни и те же данные CSS.

Для работы с CSS есть большое количество различных программных продуктов, но они не получили большого распространения. Как правило, объем CSS, кода по соотношению с HTML, не такой большой. И почти все HTML-редакторы имеют поддержку синтаксиса CSS. Из отдельных решений можно выделить TopStyle, данный программный продукт предназначен для создания стилей CSS. Очень полезным в нем является модуль Stylechecker, с помощью него можно не только проверить правильность синтаксиса таблицы стилей, но и проверить формат представления данных в зависимости от браузера. Таким образом, при его помощи можно создавать стили, которые будут одинаково отображаться в разных браузерах. Также существуют визуальные CSS-редакторы, но из них сложно выделить какой-то наиболее достойный.

## 1.3.2 JavaScript

Несмотря на то, что JavaScript не является стандартом W3C, мы рассмотрим его в данном контексте, и как часть DHTML.

JavaScript - это объектно-ориентированный скриптовый язык программирования. JavaScript встраиваемый язык, используемый для доступа к объектам в приложениях. Он нашел очень широкое применение при создании Web-страниц. JavaScript очень похож на язык Си, но все же имеет кардинальные отличия: Структуру объектов и тип объектов (например переменных, функций) можно определить в процессе выполнения кода. В JavaScript действует автоматическое приведение типов данных. Также автоматической является контроль памяти и защита от утечек. Функции в языке программирования JavaScript могут быть без имени. Все описания переменных, функции и других элементов регистр зависимы, названия переменных могут содержать буквы, символ подчёркивания, символ доллара и арабские цифры. При этом названия переменных не должны начинаться с цифры. Стоит отметить, что JavaScript не поддерживает области видимости, не умеет работать с файловой системой и потоками ввода/вывода, не имеет сетевых интерфейсов. Это сделано для обеспечения безопасности, так как скрипт на языке JavaScript легко запустить на любом компьютере, для этого будет достаточным открыть на нем Web-страницу, содержащую скрипт. Также для повышения безопасности скрипт не может получить доступ к свойствам другой страницы.

Использовать JavaScript на странице, можно при помощи тегов**<script></script>.** Например:

<**script type=«**text/javascript**»>**

**alert ('**Hello, World!**');**

**</script>**

Также скрипт может быть сохранен в отдельном файле.

<scripttype=«text/javascript»src=http://Путь\_к\_файлу\_со\_скриптом**></script>**

Язык JavaScriptдает создателю страницы очень широкие возможности для выполнения скриптов и как следствие качественно новые горизонты для создания динамических элементов на Web-страницах. JavaScript позволяет автоматически заполнять данные в формах, изменять форматирование и стиль страницы. Скрывать ненужное содержимое и отображать нужное содержимое, менять поведение клиентской части Web-приложений, изменять элементы управления на странице и так далее. При этом язык довольно простой для понимания, и большая часть функции выполняется автоматически.

Выделить какой-то редактор для JavaScript сложно, но отметим что в среде Eclipse, есть поддержка JavaScript при помощи подключаемого модуля, при этом становиться доступен весь функционал среды Eclipse. А также практически все современные браузеры имеют встроенные средства для отладки JavaScript-скриптов.

## 1.3.3 Bootstrap

Bootstrap — это открытый и бесплатный [HTML-](https://blog.skillfactory.ru/glossary/html/" \t "_blank), [CSS-](https://blog.skillfactory.ru/glossary/css/) и [JS-фреймворк](https://blog.skillfactory.ru/glossary/javascript/), который используют веб-разработчики для быстрой верстки адаптивных дизайнов сайтов и веб-приложений. Включает в себя CSS- и HTML-шаблоны оформления для веб-форм, меток, типографики, кнопок, блоков навигации и других компонентов веб-интерфейса.

Bootstrap используется, когда:

* у сайта много страниц;
* страницы собраны из простых базовых элементов — кнопок или таблиц;
* не будет глобального редизайна;
* шаблонность страниц окупается скоростью внедрения.

По сути, Bootstrap — это набор файлов. После их подключения к странице для верстки станет доступно большое количество готовых компонентов и классов. Они позволяют быстро и качественно создавать адаптивный дизайн сайта.

Классы в Bootstrap делятся на 3 большие группы:

* для создания сетки — адаптивного макета страницы;
* для стилизации контента — текста, изображений, кода, таблиц и прочей информации;
* служебные — для решения популярных вспомогательных задач, таких как отображение и скрытие элементов, выравнивание текста на странице, настройка цвета фона, отступов и пр.

Кроме классов, в Bootstrap есть компоненты (готовые объекты интерфейса). Это хлебные крошки (путь от начального элемента до уровня иерархии, который сейчас просматривает пользователь), кнопки, выпадающие списки и подсказки, модальные окна, формы, навигационные меню, всплывающие панели и пр.

Чтобы начать работать с Bootstrap, нужен текстовый редактор для работы с кодом (Visual Studio Code, Atom, Sublime Text и т.п.) или IDE — интегрированная среда разработки, а также браузер, в котором можно отслеживать изменения. Он должен быть обновлен до версии, которую поддерживает последний вариант Bootstrap.

Существует несколько способов установки Bootstrap. Получить актуальную версию фреймворка можно на [официальном сайте.](https://getbootstrap.com/)

1. Подключение скомпилированных файлов через [BootstrapCDN](https://www.bootstrapcdn.com/" \t "_blank) — общедоступную сеть доставки контента. Достаточно создать шаблон HTML и разместить внутри него ссылку на фреймворк.
2. Скачивание скомпилированных файлов CSS и JS с их подключением к проекту через ссылки.
3. Скачивание исходных файлов. Для работы потребуются [компилятор CSS](https://bootstrap-5.ru/docs/5.1/getting-started/contribute/#sass) и [автопрефиксатор](https://github.com/postcss/autoprefixer" \t "_blank).
4. Установка исходных файлов с помощью менеджеров пакетов yarn, npm, RubyGems, NuGet, Composer.

2. ПРАКТИЧЕСКАЯ ЧАСТЬ

# 2.1 Описание предметной области

Предметная область - область теории, рассматривасмая в пределах отдельного рассуждения, научной теории. Также под предметной областью принято понимать часть реального мира, подлежащего изучению для организации управления и ,в конечном счете, автоматизации.

Предметной областью данной курсовой работы, является онлайн информационной системы для вокзалов.

Архитектура сайта – это систематизация информации и навигации по

ней с целью помочь потенциальным заказчикам находить нужные им данные.

Хорошо продуманная архитектура сайта обеспечивает меньшее время на поиск запрашиваемой информации и гарантирует популярность среди пользователей благодаря простоте использования сайта.

При разработке архитектуры сайта необходимо учитывать то, как

представление информации повиляет на продвижение покупок и услуг на интернет-рынке. В процессе создания структуры нового сайта, либо

оптимизации структуры уже существующего, необходимо представлять

информацию таким образом, чтобы повысить посещаемость сайта и привлечь

внимание потенциальных покупателей к наиболее важным его разделам, исходя их предлагаемых товаров и услуг.

Важным элементом в разработке онлайн информационной системы для вокзалов является интерфейс, так как с помощью него будет происходить взаимодействие с пользователем.

Благодаря грамотному расположению блоков на странице клиент сможет найти интересующую информацию о товарах или же предоставляемых услугах.

Для разработки были использованы следующие технологии:

* HTML – этот язык является базовым в области технологий создания сайтов, так как относительно легок в освоении.
* CSS – формальный язык, преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML.

2.2 Создание окружения для проекта

В качестве редактора кода использован Visual Studio Code. Создаем папку, которую назовем "railway-info", в ней будут размещены все файлы проекта.

Сайт будет состоять из нескольких страниц. Создадим для главной страницы файл index.html. После этого создаем папку "css" и "images". В "css" создаем файл для стилей "style.css", а в "images" будут размещены нужные для сайта изображения.

Окружение готово, и теперь мы можем приступить к самой разработке сайта информационной системы железнодорожных вокзалов.

2.3 Верстка HTML страницы

Открываем ранее созданный "index.html" и пишем стандартный HTML5 код.

<!DOCTYPE html>

<html lang="ru">

<head>

<meta charset="UTF-8">

<meta name="viewport" content="width=device-width, initial-scale=1.0">

<title>Информационная система вокзалов</title>

<link rel="stylesheet" href="style.css">

<script src="site-integrator.js" defer></script>

</head>

<body>

<!-- Содержимое сайта -->

</body>

</html>

Элемент <!DOCTYPE> предназначен для указания типа текущего документа — DTD (document type definition, описание типа документа). Это необходимо, чтобы браузер понимал, как следует интерпретировать текущую веб-страницу, поскольку HTML существует в нескольких версиях.

Тег <html> является контейнером, который заключает в себе все содержимое веб-страницы, включая теги <head> и <body>. Атрибут lang="ru" указывает, что содержимое страницы представлено на русском языке.

Элемент <head> содержит машиночитаемую информацию (metadata) о документе, такую как заголовок, скрипты и стили.

В нашем проекте внутри <head> мы разместили:

* <meta charset="UTF-8"> — указывает кодировку документа
* <meta name="viewport" content="width=device-width, initial-scale=1.0"> — этот код задает параметры масштабирования страницы для устройств с разными размерами экранов
* <title> — заголовок страницы, который отображается в заголовке вкладки браузера
* <link rel="stylesheet" href="style.css"> — подключение внешнего CSS файла
* <script src="site-integrator.js" defer></script> — подключение JavaScript файла с атрибутом defer, чтобы скрипт выполнялся после загрузки страницы

Элемент <body> представляет собой контент (содержимое) документа HTML. В документе может быть только один элемент <body>.

Теперь в <body> пропишем содержимое нашего сайта. Начнем с создания шапки сайта (header).

<header>

<div class="logo">РЖД Информ</div>

<p>Информационная система железнодорожных вокзалов</p>

</header>

В шапке страницы мы разместили логотип и краткое описание сайта. Логотип помещен в div с классом "logo", что позволит нам в дальнейшем стилизовать его с помощью CSS.

Далее мы создаем навигационное меню с помощью тега <nav>.

<nav>

<ul>

<li><a href="index.html" class="active">Главная</a></li>

<li><a href="schedule.html">Расписание</a></li>

<li><a href="stations.html">Вокзалы</a></li>

</ul>

</nav>

HTML-элемент <nav> представляет собой раздел страницы, цель которого — обеспечить навигацию по текущему документу или другим документам.

HTML-элемент <ul> используется для неупорядоченного списка, а элемент <li> для создания элементов этого списка. В каждом элементе списка мы размещаем ссылку <a> на соответствующую страницу сайта. Текущей странице (в данном случае "Главная") присваиваем класс "active" для визуального выделения её в меню.

После навигации создаем блок с приветствием пользователя (hero section).

<div class="hero">

<div class="hero-content">

<h1>Добро пожаловать в информационную систему вокзалов</h1>

<p>Актуальная информация о расписании поездов, вокзалах и дополнительных услугах</p>

</div>

</div>

Блок hero использует класс "hero" для стилизации фонового изображения, и внутри него создается контейнер "hero-content" для размещения текстовой информации. Элемент <h1> используется для основного заголовка страницы, а <p> для дополнительного текста.

Затем создаем основной контейнер для содержимого страницы.

<div class="container">

<!-- Содержимое страницы -->

</div>

Внутри контейнера мы размещаем блок с преимуществами нашего сервиса.

<div class="features">

<div class="feature-card">

<div class="feature-icon">🚆</div>

<h3>Расписание поездов</h3>

<p>Актуальное расписание поездов дальнего и пригородного сообщения. Информация о задержках и изменениях в режиме реального времени.</p>

</div>

<div class="feature-card">

<div class="feature-icon">🏢</div>

<h3>Информация о вокзалах</h3>

<p>Справочная информация о вокзалах, включая инфраструктуру, схемы, услуги и контактные данные.</p>

</div>

<div class="feature-card">

<div class="feature-icon">🎫</div>

<h3>Покупка билетов</h3>

<p>Удобная система бронирования и покупки билетов онлайн. Возможность выбора места и дополнительных услуг.</p>

</div>

</div>

Каждое преимущество представлено в виде карточки (feature-card), которая содержит иконку, заголовок и описание. Для иконок использованы эмодзи, что позволяет не загружать дополнительные изображения.

Далее размещаем призыв к действию (call-to-action).

<div class="cta">

<h2>Путешествуйте с комфортом</h2>

<p>Получите доступ к полному функционалу информационной системы прямо сейчас!</p>

<a href="#" class="cta-button">Купить билет</a>

</div>

Блок CTA содержит заголовок, поясняющий текст и кнопку с призывом к действию. Кнопка представлена в виде ссылки с классом "cta-button" для дальнейшей стилизации.

Затем добавляем раздел с новостями.

<section>

<h2>Новости и объявления</h2>

<div class="news">

<article>

<h3>Обновление расписания пригородных поездов</h3>

<p>С 15 мая изменяется расписание движения пригородных поездов в связи с летним сезоном.</p>

</article>

<article>

<h3>Открытие нового терминала на Центральном вокзале</h3>

<p>Новый терминал предоставит пассажирам дополнительные удобства и сократит время регистрации.</p>

</article>

</div>

</section>

Элемент HTML <section> представляет собой автономный раздел документа, который не имеет более подходящего семантического элемента для определения. Внутри раздела размещается заголовок и контейнер с новостями (<div class="news">).

Каждая новость представлена в виде элемента <article>, который содержит заголовок <h3> и текст <p>.

В конце страницы размещаем подвал (footer).

<footer>

<p>&copy; 2025 Информационная система вокзалов. Все права защищены.</p>

<div class="footer-links">

<a href="#">О нас</a>

<a href="#">Контакты</a>

<a href="#">Помощь</a>

<a href="#">Политика конфиденциальности</a>

</div>

</footer>

В подвале размещается информация об авторских правах и ссылки на дополнительные страницы сайта. Символ &copy; представляет собой специальный HTML-символ для отображения знака авторского права ©.

2.4 Использование стилей CSS

Для стилизации страницы используем CSS. Открываем файл "style.css" и начинаем настройку стилей.

Сначала устанавливаем общие стили для всех элементов страницы.

\* {

margin: 0;

padding: 0;

box-sizing: border-box;

font-family: 'Arial', sans-serif;

}

Селектор \* применяет стили ко всем элементам страницы. Мы устанавливаем следующие свойства:

* margin: 0 и padding: 0 — сбрасываем внешние и внутренние отступы для всех элементов
* box-sizing: border-box — говорит браузеру учитывать любые границы и внутренние отступы в заданных для элемента ширине и высоте
* font-family: 'Arial', sans-serif — устанавливаем основной шрифт для всех элементов

Затем устанавливаем базовые стили для элемента <body>.

body {

background-color: #f5f5f5;

color: #333;

line-height: 1.6;

}

Мы задаем светло-серый фон (#f5f5f5), темно-серый цвет текста (#333) и высоту строки 1.6 для улучшения читаемости.

Далее стилизуем шапку сайта.

header {

background-color: #1a3c8b;

color: white;

padding: 20px 0;

text-align: center;

box-shadow: 0 2px 5px rgba(0,0,0,0.2);

}

.logo {

font-size: 28px;

font-weight: bold;

margin-bottom: 10px;

}

Для элемента <header> задаем темно-синий фон, белый цвет текста, внутренние отступы, выравнивание текста по центру и тень. Для элемента с классом .logo устанавливаем размер шрифта, делаем его жирным и добавляем отступ снизу.

Настраиваем стили для навигационного меню.

nav {

background-color: #0f2557;

padding: 10px 0;

}

nav ul {

list-style: none;

display: flex;

justify-content: center;

}

nav ul li {

margin: 0 15px;

}

nav ul li a {

color: white;

text-decoration: none;

font-weight: bold;

padding: 8px 15px;

border-radius: 4px;

transition: background-color 0.3s;

}

nav ul li a:hover {

background-color: #2e4c8a;

}

nav ul li a.active {

background-color: #3762b3;

}

Для контейнера <nav> задаем темно-синий фон и внутренние отступы. Для списка <ul> удаляем стандартные маркеры, делаем его flex-контейнером и выравниваем элементы по центру. Для элементов списка <li> задаем боковые отступы. Для ссылок задаем белый цвет текста, убираем подчеркивание, делаем текст жирным, добавляем внутренние отступы, скругляем углы и добавляем плавный переход для эффекта наведения.

При наведении на ссылку (a:hover) меняем её фон на более светлый оттенок синего. Для активной ссылки (с классом .active) задаем еще более светлый оттенок синего.

Для основного контейнера устанавливаем максимальную ширину, по центру страницы и внутренние отступы.

.container {

max-width: 1200px;

margin: 0 auto;

padding: 20px;

}

Теперь стилизуем "hero" секцию.

.hero {

background-image: url('https://via.placeholder.com/1200x400?text=Вокзал');

background-size: cover;

background-position: center;

height: 400px;

display: flex;

align-items: center;

justify-content: center;

color: white;

text-align: center;

margin-bottom: 30px;

box-shadow: 0 4px 8px rgba(0,0,0,0.1);

position: relative;

}

.hero::before {

content: '';

position: absolute;

top: 0;

left: 0;

width: 100%;

height: 100%;

background-color: rgba(0, 0, 0, 0.5);

}

.hero-content {

position: relative;

z-index: 1;

padding: 20px;

}

.hero h1 {

font-size: 3rem;

margin-bottom: 20px;

text-shadow: 2px 2px 4px rgba(0,0,0,0.5);

}

.hero p {

font-size: 1.2rem;

max-width: 800px;

margin: 0 auto;

text-shadow: 1px 1px 2px rgba(0,0,0,0.5);

}

Для элемента с классом .hero задаем фоновое изображение, устанавливаем его размер так, чтобы оно покрывало весь контейнер (cover), центрируем его, задаем высоту, делаем flex-контейнером для центрирования содержимого, устанавливаем белый цвет текста, выравниваем текст по центру, добавляем нижний отступ и тень, а также задаем позицию relative для позиционирования псевдоэлемента.

С помощью псевдоэлемента .hero::before создаем полупрозрачный черный слой поверх фонового изображения для лучшей читаемости текста. Для содержимого hero секции (.hero-content) устанавливаем position: relative и z-index: 1, чтобы оно отображалось поверх псевдоэлемента.

Для заголовка h1 внутри hero секции задаем крупный размер шрифта, нижний отступ и тень текста для лучшей читаемости. Для абзаца p задаем размер шрифта, максимальную ширину, выравнивание по центру и тень текста.

Далее стилизуем секцию с преимуществами.

.features {

display: grid;

grid-template-columns: repeat(auto-fit, minmax(300px, 1fr));

gap: 30px;

margin-bottom: 40px;

}

.feature-card {

background-color: white;

padding: 30px;

border-radius: 8px;

box-shadow: 0 4px 8px rgba(0,0,0,0.1);

transition: transform 0.3s;

}

.feature-card:hover {

transform: translateY(-5px);

}

.feature-icon {

font-size: 2.5rem;

margin-bottom: 15px;

color: #1a3c8b;

}

.feature-card h3 {

font-size: 1.5rem;

margin-bottom: 10px;

color: #1a3c8b;

}

Для контейнера .features используем Grid-сетку с автоматическим размещением элементов, где каждый элемент занимает минимум 300px, но может растягиваться пропорционально доступному пространству. Задаем отступы между элементами сетки (gap) и нижний отступ для контейнера.

Для карточек преимуществ (.feature-card) задаем белый фон, внутренние отступы, скругленные углы, тень и плавный переход для эффекта при наведении. При наведении карточка приподнимается на 5 пикселей вверх благодаря свойству transform: translateY(-5px).

Для иконок и заголовков внутри карточек задаем размер, цвет и отступы.

Теперь стилизуем секцию с призывом к действию (CTA).

.cta {

text-align: center;

background-color: #e6f0ff;

padding: 40px 20px;

border-radius: 8px;

margin-bottom: 40px;

}

.cta h2 {

font-size: 2rem;

margin-bottom: 20px;

color: #1a3c8b;

}

.cta-button {

display: inline-block;

background-color: #1a3c8b;

color: white;

padding: 12px 30px;

border-radius: 4px;

text-decoration: none;

font-weight: bold;

font-size: 1.1rem;

transition: background-color 0.3s;

}

.cta-button:hover {

background-color: #0f2557;

}

Для блока .cta задаем выравнивание текста по центру, светло-голубой фон, внутренние отступы, скругленные углы и нижний отступ. Для заголовка внутри блока задаем размер, отступ и цвет. Для кнопки задаем блочное отображение, синий фон, белый цвет текста, внутренние отступы, скругленные углы, убираем подчеркивание, делаем текст жирным, задаем размер шрифта и плавный переход для эффекта при наведении. При наведении кнопка становится темнее.

Стилизуем раздел с новостями.

.news article {

margin-bottom: 20px;

}

.news article h3 {

color: #1a3c8b;

margin-bottom: 5px;

}

Для статей в разделе новостей задаем нижний отступ, а для заголовков статей — синий цвет и небольшой нижний отступ.

В конце стилизуем подвал сайта.

footer {

background-color: #1a3c8b;

color: white;

text-align: center;

padding: 20px;

margin-top: 40px;

}

.footer-links {

display: flex;

justify-content: center;

margin-top: 15px;

}

.footer-links a {

color: white;

margin: 0 10px;

text-decoration: none;

}

.footer-links a:hover {

text-decoration: underline;

}

Для подвала задаем синий фон, белый цвет текста, выравнивание по центру, внутренние отступы и верхний отступ. Для контейнера с ссылками делаем flex-контейнер с выравниванием по центру и верхним отступом. Для ссылок задаем белый цвет, боковые отступы и убираем подчеркивание. При наведении на ссылку добавляем подчеркивание.

Наконец, добавляем адаптивные стили для мобильных устройств.

@media (max-width: 768px) {

nav ul {

flex-direction: column;

align-items: center;

}

nav ul li {

margin: 5px 0;

}

.hero h1 {

font-size: 2rem;

}

.hero p {

font-size: 1rem;

}

.form-row {

flex-direction: column;

}

.schedule-table {

display: block;

overflow-x: auto;

}

.stations-grid {

grid-template-columns: 1fr;

}

.footer-links {

flex-direction: column;

align-items: center;

}

.footer-links a {

margin: 5px 0;

}

}

Медиа-запрос @media (max-width: 768px) применяет стили только на экранах шириной до 768 пикселей. Для таких устройств мы:

* меняем направление навигационного меню на вертикальное
* уменьшаем размер шрифта в hero секции
* адаптируем формы и таблицы для удобного просмотра
* меняем сетку станций на одноколоночную
* меняем направление ссылок в подвале на вертикальное

Таким образом, мы создали информационную систему железнодорожных вокзалов с адаптивным дизайном, который хорошо выглядит как на компьютерах, так и на мобильных устройствах.

3. РЕЗУЛЬТАТ

В процессе выполнения курсовой работы были освоены ключевые этапы создания веб-сайта:

1. Разработана структура веб-страницы с применением языка разметки HTML.
2. Оформление сайта выполнено с использованием каскадных таблиц стилей CSS.
3. Реализована адаптивная верстка с помощью медиа-запросов для корректного отображения на различных устройствах.

В итоге был разработан современный, функциональный и удобный веб-сайт, соответствующий требованиям заказчика и современным стандартам веб-дизайна.

4. ЛИТЕРАТУРА

1. **"HTML и CSS. Разработка и дизайн веб-сайтов" — Джон Дакетт (Jon Duckett)**  
   (Очень доступно изложены основы HTML, CSS, также немного затрагиваются темы динамических страниц.)
2. **"Веб-программирование. HTML, CSS, JavaScript и PHP" — Н. В. Бондаренко, Е. А. Панкратова**
3. **"Самоучитель HTML, CSS и JavaScript" — Джулиан Дакетт (Julian Duckett)**  
   (Пошаговый разбор с объяснением тегов, скриптов, таблиц стилей.)
4. **"HTML5 и CSS3. Разработка сайтов для любых браузеров и устройств" — Брайан Хоган (Brian Hogan)**  
   (Тоже охватывает базовые темы верстки.)
5. **"Основы веб-программирования" — О. В. Петров**  
   (Подробное изложение HTML, XHTML, DHTML.)

5. ПРИЛОЖЕНИЯ

Репозиторий:

<https://github.com/TheRaGGa/Kursovaya>

Страница сайта на хостинге GitHub Pages:

<https://theragga.github.io/Kursovaya/>