МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество
АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

кафедра Компьютерных технологий

«Допущен к защите»
Заведующий кафедрой д.ф.-м.н.
(Ф.И.О., ученая степень, звание)
(подпись) «_____» **20__ г.**

ДИПЛОМНЫЙ ПРОЕКТ

На тему: Разработка информационного сайта с использованием технологий ASP.NET

Специальность 5В070400 - Вычислительная техника и программное обеспечение

Выполнил (а) Муратова А. (Фамилия и инициалы) ВТ-10-4 (группа)

Научный руководитель Турганбаев Е.С., к.ф.-м.н., доцент (Фамилия и инициалы, ученая степень, звание)

Консультанты:

по экономической части:
Ергешева З.Д., ст. преп.
(Фамилия и инициалы, ученая степень, звание)
Ергешева (подпись) «12» 05 2014 г.

по безопасности жизнедеятельности:
Прокофьева Н.Г., д.х.н., профессор
(Фамилия и инициалы, ученая степень, звание)
(подпись) «04» 05 2014 г.

по применению вычислительной техники:
Турганбаев Е.С., к.ф.-м.н., доцент
(Фамилия и инициалы, ученая степень, звание)
(подпись) «29» 05 2014 г.

_____
(Фамилия и инициалы, ученая степень, звание)
(подпись) «_____» 20__ г.

Нормоконтролер: Тусупова Д.М.
(Фамилия и инициалы, ученая степень, звание)
(подпись) «31» мая 2014 г.

Рецензент: _____
(Фамилия и инициалы, ученая степень, звание)
(подпись) «_____» 20__ г.

Алматы 2014 г.

МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РЕСПУБЛИКИ КАЗАХСТАН

Некоммерческое акционерное общество

АЛМАТИНСКИЙ УНИВЕРСИТЕТ ЭНЕРГЕТИКИ И СВЯЗИ

Факультет Информационных технологий
Специальность Вычислительная техника и программное обеспечение
Кафедра Компьютерных технологий

ЗАДАНИЕ

на выполнение дипломного проекта

Студент Муратова Айман
(фамилия, имя, отчество)

Тема проекта Разработка информационного сайта с использованием технологии ASP.NET

утверждена приказом ректора №115 от «24» сентября 2013 г.
Срок сдачи законченной работы «___» _____ 20___ г.
Исходные данные к проекту требуемые параметры результатов проектирования (исследования) и исходные данные объекта
разработать информационной сайт с использованием технологии ASP.NET, на котором располагается информация о городе, истории и т.д.

Перечень подлежащих разработке дипломного проекта вопросов или краткое содержание дипломного проекта:
Дипломная работа имеет целью разработку информационного сайта с использованием технологии ASP.NET.
Для достижения указанной цели необходимо:
1. Анализ предметной области
2. Обзор технологий для разработки веб-сайта
3. Проектирование сайта.

Перечень графического материала (с точным указанием обязательных чертежей)

Схема этапов разработки сайта
Гибридная структура сайта
Структура XML-файла
Предварительный макет страницы сайта
Итоговый макет сайта

Рекомендуемая основная литература

1. Горлаков С.Г. Осваиваем популярные системы управления сайтами
2. Ганеев Р.М. Проектирование интерактивных Web-приложений
3. Кузнецов М.В., Симдянов И.В. PHP на примерах.
4. Гончаров А.Н. Самоучитель HTML.

Консультанты по проекту с указанием относящихся к ним разделов

Раздел	Консультант	Сроки	Подпись
БЖД	Трифонова И.Г.	11.04 – 04.05.14	
Экономика	Евсеева З.Д.	15.04. – 12.05.14	Евсеева
Нормоконтроль	Гусейнов Д.М.	31.05.14	
Основная часть	Нуртазабаев Е.С.	3.03 – 29.05.14	

Г Р А Ф И К
подготовки дипломного проекта

№ п/п	Наименование разделов, перечень разрабатываемых вопросов	Сроки представления руководителю	Примечание
1.	Сбор и анализ данных	3.03 - 15.03	
2.	Моделирование предметной области		
3.	Обзор технологий для разработки вэб-сайта		
4.	Проектирование сайта		
5.	Безопасность жизнедеятельности		
6.	Экономическое обоснование		

Дата выдачи задания «3» марта 2014 г.

Заведующий кафедрой _____ Курманбаев З. К
(подпись) (Фамилия и инициалы)

Руководитель _____ Курманбаев Е. С
(подпись) (Фамилия и инициалы)

Задание принял к исполнению
студент _____ Мурзагова А.
(подпись) (Фамилия и инициалы)

Аннотация

В данном дипломном проекте исследуются технологии, предназначенные для разработки сайтов. В процессе исследования нужно провести сравнение двух технологий, ASP.net и PHP, и выделение одной из них. Используя одну из технологий, требуется разработать информационный сайт города Зайсан. Так же, в данной работе, нужно использовать XML-файлы, для хранения информации. Сайт должен содержать актуальную информацию о городе и его окрестностях. Сайт должен быть ориентирован на пользователей, которым требуется корректная и подробная информация о городе. Разработка веб-сайта позволит пользователям находить правильную информацию.

Аңдатпа

Берілген дипломдық жобада Зайсан қаласының ақпараттық сайты құрастырылады. Сайт қала тұрғындары мен туристтерге арналған. Соған қоса, Зайсан қаласының сайтын ғаламторға орналастырылғандығынан, ақпарат жетіспеулік мәселесі шешіледі.

Проект барысында ASP.net және PHP серверлық технологияларын зерттеліп, салыстырылады. Салыстырудан кейін ең қолайлы технология таңдалып, сол технология негізінде сайт құрастырылады. Соған қоса, сайтты құрастыру кезінде ақпараттарды сақтау үшін XML файлы пайдаланылады.

Abstract

In this degree project the technologies intended for development of sites are researched. In the course of research it is necessary to carry out comparing of two technologies, ASP.net and PHP, and separation of one of them. Using one of technologies, it is required to develop information site of the city of Zaisan. As, in this operation, it is necessary to use XML files, for information storage. The site shall contain up-to-date information about the city and its neighborhoods. The site shall be oriented on users who need the incorrect and detailed information about the city. Development of the web site will allow users to find the correct information.

Содержание

Введение

С каждым годом Интернет становится доступнее и надежнее, в связи с этим количество его пользователей увеличивается. Интернет являет собой один из самых активных и процветающих средств информации. Интернет используется в разных сферах жизни человека и для различных целей. Одни используют его для связи с близкими людьми, находящимися далеко, на большом расстоянии. Другие для того, чтобы быть в центре последних новостей, третьи для обучения и т.п. Сферы использования Интернета весьма разнообразны.

Самой используемой сферой Интернета является справочно–информационный поиск. Существует большое количество разнообразных справочно–информационных ресурсов, предоставляющих подробную информацию практически по любому запросу. Пользователи всемирной сети предпочитают зайти на пойсковый сайт и найти необходимую информацию, нежели искать ее в газетах и журналах.

На сегодняшний день, создать сайт можно различными способами. Также можно воспользоваться услугами веб–студий или независимых разработчиков, которые предоставляют свои услуги по разработке и в продвижении сайтов. Разработка и продвижение сайта – это два различных понятия, которые рассматриваются поотдельности.

Главная задача информационного сайта – это предоставление корректной информации. В чем же приемущества размещения и передачи информации через Интернет? Во–первых, это доступность информации, любой пользователь Интернета имеет доступ к размещенной на сайте информации вне зависимотсти от времени. Во–вторых, скорость размещения информации, администратор способен быстро поместить ее на странице сайта, а также быстро ее откорректировать. В–третьих, это удобство при пользовании сайтом и информацией, поскольку размещаются только актуальные и точные сведения. Поэтому неудивителен тот факт, что большинство людей обращается за помощью на различные сайты.

Для разработки сайтов используются различные технологии и инструменты. Как правило, это – такие языки как HTML, CSS, JavaScript и др. Также для проектирования взаимодействия сайта с сервером используются различные технологии. В данном дипломном проекте исследуются и сравниваются две технологии: ASP.net и PHP. А также на основе сравнения будет выбрана подходящая для данного проекта.

Город Зайсан – административный центр Зайсанского района, с населением более 40 000 человек. Окрестности имеют прекрасную природу и хороший климат. Город имеет развитую инфраструктуру, в лице различных развивающихся предприятий и промышленности. Вблизи города находится озеро Зайсан, место отдыха и туризма, которое привлекает большое количество приезжих людей. Район имеет большие потенциалы на развитие.

Город имеет официальный государственный сайт. Были изучены все материалы и сведения находящиеся в данном ресурсе, и было сделано заключение, что этого недостаточно для пользователей Интернета.

Все вышеизложенные аспекты и стали причиной выбора темы данного дипломного проекта, а именно – разработки информационного сайта. Соответственно, целью проекта является разработка иныормационного сайта города Зайсан, для получения информации, так же в рамках дипломного проекта – исследование двух серверных технологий. В итоге должен, получиться сайт, на котором располагается информация о городе, истории его создания, достопримечательностях, природе и климате. А также служить источником всех последних и важных мероприятий и новостей города и района. Данная цель определила структуру саму дипломного проекта. Для достижения цели необходимо изучение языков программирования сайтов, которые являются самыми популярными, изучить структуры аналогичных, провести параллель и выявить то, что будет новшеством в данной области. Далее определение содержания сайта и его дальнейшая разработка.

1 Анализ предметной области

Интернет – всемирная компьютерная сеть, которая заключает в себе почти все компьютеры со всего мира. В современном мире ежедневно интернетом пользуются около 16 миллионов людей, более чем в 150–ти странах мира. С каждым годом размеры и пользователи всемирной сети увеличиваются. Интернет образует некое подобие архива данных, которое так же обеспечивает связь между различными информационными сетями, принадлежащими различным учреждениям во всем мире, одну с другой.

Если раньше всемирная паутина использовалась только для передачи файлов и сообщений электронной почты, то на сегодняшний день решаются более сложные задачи распределенного доступа к ресурсам, а так же Интернет используется для хранения данных, обмена информацией, живой связи между пользователями и для многого другого.

Сайт – это место во всемирной сети (интернете), которое имеет свой адрес, собственного владельца и построено из одиночных веб–страниц, которые представлено нам как один каталог информации.

Всемирная сеть состоит из различных сайтов, которые доступны для пользователей. Сайты состоят из нескольких веб–страниц, каждый сайт имеет свой собственный уникальный корневой адрес, содержание, систему и дизайн. Каждая веб–страница сайта это текстовый документ, написанный на различных языках программирования, обычно на HTML, CSS, JavaScript, PHP и других.

По запросу браузеров данные веб–страницы в виде текстовых файлов, загружаются на компьютер пользователя, обрабатываются браузером и появляются на компьютере пользователя как сайт. Веб–страницы сайтов имеют возможность содержать не только тексты и картинки, но так же и сложные виды информации, с огромным количеством функций, таких как скрипты на различную анимацию и др.

Обычно, для разработки сайтов, привлекается некоторое количество людей, таких как веб–разработчики, веб–дизайнеры и другие. Процесс разработки сайта делится на несколько этапов. Для начала аналитиком проводится планирование структуры будущего сайта (разделы, содержание и др.). Далее веб–дизайнер создает дизайн сайта, после этого за проект берется верстальщик, который создает верстку разработанного макета сайта. Веб–разработчик же, разрабатывает систему управления сайтом, устанавливает программные модули, наполняет сайт информацией, проводит предварительное тестирование. После прохождения этапа тестирования, сайт размещают в интернете.

Первым сайтом в мире, считается интернет страница с описанием технологии World Wide Web, которая была создана 6 августа 1991 года. Автором данного проекта является Тим Бернерс Ли.

В наше время существует огромное количество различных типов сайтов, начиная от простых страничных сайтов–визиток, включая различного вида

социальные сети, интернет–магазины, форумы, заканчивая, крупными энциклопедиями вроде Википедии. С помощью сайта можно распространять информацию, быстро её корректировать, таким образом, люди, посетившие сайт будут обладать самыми последними данными. При помощи сайта существует возможность обмена огромным количеством информации с другими пользователями сети. Никакие другие средства массовой информации не могут дать такого же эффекта и такой быстрой реакции. К примеру, если рассматривать различные печатные издания, то получается, что пользователь ограничен количеством информации. Если у вас есть свой сайт, то вы имеете неограниченные возможности по его управлению и управлению информацией в ней, и можете размещать все что вы хотели бы видеть. Любой человек, интересующийся определенной информацией с интернета, может в любое время суток зайти на сайт и просмотреть все, что ему нужно. Главная задача сайта – это предоставление корректной информации пользователю и потенциальному клиенту. По статистике и прогнозам на будущее, можно уверенно сказать, что интернет ресурсы имеют прочную связь с общественной жизнью людей. Почти у каждого человека есть своя страница в социальных сетях, у каждой второй компании имеется свой собственный веб–сайт. Телефонная связь, газеты, телевидение уходят на задний план. Люди научились пользоваться интернетом в своих целях, обычно для поиска необходимой информации. Имея свой сайт, вы могли бы быть ближе к своим потребителям и или вносить личный вклад в общественную жизнь.

С каждым годом количество пользователей всемирной паутины увеличивается. Общее количество пользователей составляет более 2,5 млрд., это около 34% всех жителей Земли. Более 62% населения Казахстана пользуются Интернетом. Согласно данным веб–мониторинговой компании Netcraft, по состоянию на 1 января 2013года, в интернете работали 861 379 152 сайта и персональных блога. Большинство сайтов разработаны на основе платформ Apache (64,92%), Microsoft (14,39%), Nginx (9,89%), Google (3,16%). Если учитывать то, что процентная доля разработчиков Apache, Microsoft, Nginux возрастает, а Google уменьшается (в феврале – на 0,09%), то актуальность изучения этих платформ пропорционально увеличивается.

В настоящее время около 90% компаний в США и более 80% компаний в Западной Европе имеют собственные сайты в Интернете. У большинства из них даже нет физического офиса, так как их бизнес на 100% создаётся в сети.

Сайт – это зачастую первое, что видит человек при знакомстве с Вашей компанией. Поэтому разработка сайта – это очень важное и ответственное решение. Если учитывать то, что современный мир полон различных технологий, к которым так же можно отнести и интернет ресурсы, которые совершенствуются и изменяются очень быстро, то можно сделать вывод, что использование веб–сайтов может оказать положительное воздействие на уровень и размер его аудитории и пользователей. К примеру, если допустить что пользователю требуется определенная информация по вашей теме (по тематике вашего сайта), то в первую очередь он обратится за помощью к

поисковым сайтам, которые выведут топ сайтов с необходимой информацией. И при правильной организации содержимого вашего сайта ваш сайт может быть в начале поискового результата. Тем самым поднимая рейтинг вашего веб–сайта и его посещаемость.

После разработки и размещения сайта во всемирной паутине, перед владельцами сайтов стоит вопрос его поддержки и продвижения. В таком случае стоит поближе ознакомиться с SEO оптимизацией сайтов.

Если вы хотите притянуть на свой сайт больше пользователей, чтобы иметь шанс на демонстрацию своих продукций и услуг или же просто увеличить посещаемость, то вам нужно находиться в первой десятке результатов поисковых запросов. А чтобы сайт находился в списке заветной десятки, нужно знать основы оптимизации и продвижения сайта в поисковых системах и уметь ими пользоваться.

SEO расшифровывается как «Search Engine Optimization», что в переводе с оригинала дает нам понятие «поисковой оптимизации» или же «оптимизации под поисковые машины». Значение этих слов – это оптимизация сайта для его быстрого продвижения в рейтинге поисковых систем.

Чем ближе позиция сайта к первой десятке выводимых запросов, тем больше возможность активного посещения данного сайта. Например, если сайт находится в первом списке, то это означает, что он будет на 100% просмотрен пользователем.

Если рассматривать статистику по поисковикам и по действию пользователей по переходу по ссылкам можно сделать такой практичный прогноз. Чем выше позиция сайта в выдаче поисковика, тем больше вероятнее, что пользователь просмотрит сайт, так как, по статистике, около 100% заходят по первым трем ссылкам поискового результата, далее тенденция к уменьшению – до десятой ссылки первой страницы выдачи добираются 20–50% пользователей.

На вторую страницу выдачи поисковика заходят 10–20%. Следовательно, для всякого ресурса наиболее желанно место именно в первой десятке поисковой выдачи. Но, к сожалению, сайтов–конкурентов слишком много, а десятка всего одна. И тут SEO оптимизация сайта приходит на помощь.

Поисковую SEO–оптимизацию условно можно разделить на три части:

В первую часть входит работа с содержимым сайтов. В нее входит исправление всевозможных ошибок, добавление и изменение содержимого, программного кода веб–страниц сайта, перепланирование ссылок и так далее. Так называемая внутренняя оптимизация. Насколько действенным будет первый шаг, зависит только от усилий и знаний в этой области. Нужно так же учитывать, что алгоритмы поисковиков немного различаются, поэтому оптимизация для Google–поиск будет немного отличаться от оптимизации для Яндекс–поиска.

Во вторую часть оптимизации сайта относится его раскрутка (реклама). В этот период, нужно размещать рекламные ресурсы на ваш сайт. Все это выполняется с помощью различных мероприятий, которые нужно проводить

вне сайта (на других сайтах, в каталогах статей, форумах, закладках и прочих площадках). Задачей этих мероприятий является увеличение ссылок на ваш сайт и продвижение веб–сайта по поисковым запросам, а так же увеличение его рентабельности. Это называется продвижением сайта или внешней оптимизацией (раскруткой).

Третья часть заключается в поддержке достигнутых позиций и улучшении получаемых результатов. Постоянное наблюдение за своими результатами и показателями конкурентов и их анализ, постоянное изменение и пополнение ключевых слов, текстов для ссылок, поддержка сайта, корректировка площадок – все это должно выполняться своевременно, чтобы сохранять занятую позицию.

1.1 Город Зайсан

Зайсан – город в Казахстане, административный центр Зайсанского района Восточно–Казахстанской области.

Расположен в предгорьях хребта Саур, в 378 км на юго–восток от железнодорожной станции Жангизтобе (на линии Семипалатинск – Алма–Ата). Имеется аэропорт.

Зайсанский район занимает юго–восточную часть территории области. На юге и востоке он граничит с Китайской Народной Республикой, на западе с Тарбагатайским районом. На севере граница проходит по Чёрному Иртышу и Бухтарминскому водохранилищу. Большая часть района расположена в Зайсанской и Чиликтинской впадинах, которые представляют собой обширный межгорный прогиб. Менее половины территории района носит горный характер. Горы тянутся в широтном направлении, занимая всю южную часть района и представлены хребетами Саур и Манрак.

Хребет Саур расположен в юго–восточной части территории района. Высшие точки Саурского хребета, горы Музтаг (ледяная гора), достигают 3500–3816 м, с которых спускаются небольшие ледники. По направлению к северо–западу высоты постепенно снижается до 1500–1800 м.

Климат района резко континентальный с большими суточными амплитудами температуры воздуха. По климатическим условиям территория района относится к пустынно–степной сухой и альпийской тундрово–луговой зонам, которые характеризуется следующими показателями.

Пустынно–степная луговая зона. Лето сухое и жаркое, Зима малоснежная и суровая. Среднегодовое количество осадков составляет 281 мм. Среднегодовая температура воздуха – 40.

Снежный покров устанавливается во второй половине ноября, сходится в первых числах апреля. Средняя высота снежного покрова к концу зимы достигает 20–30 см, с колебанием в отдельные годы от 5 до 40 см.

Население города составляет – более 40 000 жителей.

1.2 История города

Город возник в 1868 году, с 1897 года – уездный город Семипалатинской области. Статус города получн 1941 года.

В верховьях Жеменейки среди густых лесов в 1830–1840 годах был заложен первый камень города Зайсана. Раньше город назывался "Джеменейка". Примерно в 1830 годах географ–геодезист Маяковский приезжает в эти места для того, чтобы найти удобное и безопасное место для постройки поста на востоке Иртыша. Его сопровождал житель крепости Базарка Серменулы Сырымбай из рода Таукен.

Джеменейка расширялась и стала крупным населенным пунктом. Начиная с 1860 года жизнь жителей города стала тесно связана с озером Зайсан и рекой Иртыш. Многочисленные путешественники из Семипалатинска, Павлодара, Каркаралы, Усть–Каменогорска и других городов стали называть этот край "Зайсан".

В официальных архивных данных указывается, что город получил название "Зайсан" между 1864–1868 годами. В 18 томе географической энциклопедии, выпущенной в 1903 году в Санкт–Петербурге издательством А.Ф. Дербиена указано, что "Первый камень города Зайсан заложен в 1864 году, когда большая часть территории уезда принадлежала России". А в 1864 году город Зайсан по указу царя стал называться станицей и был укреплен как крепость воинскими подразделением под руководством генерал–майора инфантерии Л.Ф.Бакова. Крепость обрела большое значение, когда через таможенный пост Зайсан стали вывозить из Казахстана в Китай, Тибет, Синьзян, шерсть, шкуры, лес и другое сырье, а завозить оттуда ткани, одежды и ручные изделия.

В 1870–1880 годах в городе проживало 2663 русских, 122 казаха, а в 1900 годах казахов стало 4 тысячи. Население стало заниматься полеводством, скотоводством, растениеводством.

Окружные указы стали губернскими, в 1857 году во время образования уездов был создан Зайсанский уезд, а Зайсанский пост стал административно–политическим центром уезда. В городе находилось уездное начальство, таможня, казначеиство, почта–телеграф, личное подсобное хозяйство. В 1880 году была протянута телеграфная линия, связывающая город с Семипалатинском.

Восточная часть реки Жеменей, делящей пост Зайсан на две части, стала называться "Казачья слобода" и состояла из большой улицы, а в западной части было 3–4 улицы, начальная школа для азиатов, деревянная церковь, дом приставского начальства, торговая площадка, 110 кирпичных и 7 деревянных домов – писал Потанин.

Удобный для внутренней и внешней торговли город Зайсан быстро расширился, укрепилась взаимная торговля с западным Китаем, Монголией. Ежегодно проводимая в мае Никольская ярмарка увеличила объем торговли. Стали открываться такие предприятия, как мыловаренный, пивоваренный,

кирпичный заводы, завод по обработке шерсти и кожи, имевшие важное местное значение. В городе Зайсан богатый – дешевым сырьем, стало приезжать много купцов.

К концу 19 века Зайсан стал известен как купеческий город. Для детей купцов и чиновников, кроме азиатской начальной школы для мальчиков, открыли в 1883 году начальное училище для девочек.

О районе Зайсанский уезд стал многонациональным. Здесь проживали казахи, русские, татары, узбеки, туркмены, киргизы, китайцы, и представители других народов.

В Зайсане, который был центром Зайсанского уезда Семипалатинской губернии было 2 мечети и 2 церкви. В то время 6 районов уезда были объединены и созданы 17 самостоятельных волостей. В уезде открылось 12 школ, где училось 350 детей и преподавали 17 учителей. В 1896 году в городе открылась семиместная больница, где медицнскую услугу оказывали два врача, предприниматели А.С.Хохлов, Титов, Собачкин открыли шахту, наладили угледобычу в кендырлыке. В 1880–1890 годах город стал крупным центром, об этом свидетельствовало и его архитектура. В городе выстроились рядами магазины, дома из обоженного кирпича, созданные по плану архитектура Баязита Сатпаева.

В городе, ставшем крупным культурным центром где работали известные поэты Султуанмахмут Торайгыров, Асет Найманбайулы, Ногайбаев. Качество и предпопложительно объем Кендырлыкского угля изучил Павел Васильев Михаэлис. Н.Бухман изобрел солнечную энергию, которую на деле применил Коншин, а ссыльные Ватсон, Томашевский, Худыковский и дригие внесли свой вклад в развитие культуры края.

В городе было два больших кожевенных завода. Работал кирпичный завод Фунтикова, расположенные вдоль Жеменейки мельницы обеспечивали город мукой. Пивовареный завод Дображанского обеспечивал баварским пивом не только уезд, но также экспортировал в приграничные районы. В городе было два мыловаренных завода, и работал завод по обработке привозимой железной руды, которую экспортировали в Китай, Тибет и Монголию. Кинотеатр Сорокина демонстрировал народу фильмы и информировал населения о жизни и событиях планеты. Большое значение имел завод Асташева изготовляющий валенки из собранной шерсти.

В 1922 году в городе была открыта уездная больница.

В 1938 году было открыта противомалярийная, а позднее санитарно–эпидемиологияческая станция. В 1937 году был открыт родильный дом. На 7 мест, где работала врач Щедрина. В 1945 году стала оказывать услуги первая аптека. В 1950 году родильный дом был открыт в отдельном здании, койко–мест было увеличино до 100. в 1952 году был открыт кожно–венерологический диспансер на 25 койко–мест, в 1958 году туберкулезный диспансер. На 25 койко–мест.

В 1926 году в Зайсан был создан городской Совет.

Осенью 1928 года в республике были распущенны уезды по решению исполнительного Комитета ЦК ВКП Казахстана, вместо них были созданы районы. Зайсанский уезд был разделен на Зайсанский, Маркакольский, Кокпектинский, Кызылтас, Аксутаский районы. Зайсанский район был создан 3 сентября 1928 года по Указу ВЦИК. Территория района сейчас равна по объему трем волостям: Манырак, Черный Иртыш, Кендырлык. До 1930 года Зайсанский район был составе Семипалатинской области, начиная с этого года стал относиться вновь образованной Восточно–Казахстанской.

В годы Великой Отечественной войны из района ушли воевать 6 559 человек, из них погибло 2500. В настоящее время в городе осталось 29 ветеранов войны.

1.3 Природа, ресурсы и климат

Поистине безграничны возможности для развития туризма.

Горы покрыты лесами, в которых водятся дикие звери, птицы. Озера и реки изобилуют рыбой. А всего сто–сто пятьдесят лет назад край Зайсанский был буквально "перегружен" зверем и дичью. Обитали здесь тигры, дикие верблюды, куланы, сайгаки, лошади Пржевальского. Водились дрофы, рябки, орлы, а уж прочей птицы на озере было – как мошки в вечерний час. Климатические условия этой удивительной земли позволяют выращивать виноград, яблоки, арбузы, овощи. Склоны гор изобилуют барбарисом, смородиной, боярышником, здесь растет более 100 видов лекарственных трав. Минеральная вода артезианского источника Джеменейского месторождения славится своим качеством.

Вода – основа жизни. Построенное в 1967 году водохранилище Уйдене позволяет орошать 13 тысяч гектаров земли, а две мощные горизонтальные турбины надежно снабжают значительную часть района дешевой электроэнергией. Наличие воды, электроэнергии, природных ресурсов, упорство людей сделали этот край привлекательным для интенсивного сельского хозяйства, предприятий по переработке его продуктов.

Зайсанский район занимает юго–восточную часть территории области. На юге и востоке он граничит с Китайской Народной Республикой, на западе с Тарбагатайским районом. На севере граница проходит по Черному Иртышу и Бухтарминскому водохранилищу. Большая часть района расположена в Зайсанской и Чиликтинской впадинах, которые представляют собой обширный межгорный прогиб. Менее половины территории района носит горный характер. Горы тянутся в широтном направлении, занимая всю южную часть района, представлены хребтами Сауыр и Манрак.

Хребет Сауыр расположен в юго–восточной части территории района. Высшие точки Сауырского хребта, горы Муз–Тау (ледяная гора), достигают 3500–3816 м, с которых спускаются небольшие ледники. По направлению к северо–западу высоты постепенно снижается до 1500–1800 м.

Климат района резко континентальный с большими суточными амплитудами температуры воздуха. По климатическим условиям территория района относится к пустынно–степной сухой и альпийской тундрово–луговой зонам.

Пустынно–степная луговая зона. Лето сухое и жаркое. Зима малоснежная и суровая. Среднегодовое количество осадков составляет 281 мм. Среднегодовая температура воздуха –40. Абсолютный минимум температуры приходится на январь месяц – 50, абсолютный минимум на июль + 46. Продолжительность безморозного периода 130–150 дней.

Снежный покров устанавливается во второй половине ноября, сходится в первых числах апреля. Средняя высота снежного покрова к концу зимы достигает 20–30 см, с колебанием в отдельные годы от 5 до 40 см.

Зайсан (также Зайсан–Нор) – озеро на востоке Казахстана, в открытой высокой и плоской долине между горными хребтами: с северо–востока Алтайским, с северо–запада – Колбинским и с юга. Тарбагатайским. Китайская граница проходит на расстоянии 60 км от восточного берега озера, с китайской стороны течёт и впадает в Зайсан река Чёрный Иртыш.

Озеро находится на высоте 420 м, его длина 105 км, а ширина – 22–48 км, максимальная глубина 15 м.

Зайсан находится между 47°60' и 48°30' с. ш. и между 83° и 85° в. д.

В прежнее время озеро было глубже и имело большее протяжение, о чём свидетельствуют старые прибои, находящиеся на низменных берегах вдали от уреза воды. Вода пресная, мягкая и здоровая. Озеро покрывается льдом в ноябре и вскрывается в конце апреля. Дно Зайсана иловато, местами песчано и покрыто мелкой галькой.

Берега низкие, заросшие на большом пространстве от воды камышом, только около Бакланьяго и Бархоцкого мысов берег чист. В некоторых местах от горных возвышенностей входят в озеро мысы, из которых, кроме двух вышеупомянутых, наиболее известны: Вершинин, Голодаевский, Тополевый, Песчаный, Голый. В середине озера островов нет, только при впадении Чёрного Иртыша находятся два маленьких острова Канинских и близ выхода Белого Иртыша – Кылинский остров. В Зайсан впадают реки: с восточной стороны – Чёрный Иртыш, Кендырлык, с западной – Кокпектинка, Бугаз и Базар, с северной – Черга, Арасан, Терс–Арлык и пр., вытекает на севере Белый, или собственно Иртыш. Озеро славится обилием рыбы. Здесь ловятся: судак, таймень (Salmo fluviatilis), ускуч (Salmo coregonoides), щука, налим, окунь, язь, линь и карась.

Горы Саур–Тарбагатай

Саур–Тарбагатай расположен на востоке Сарыарки. К северу от него лежит Зайсанская котловина, к югу – Алакольская. Близко расположенные к озеру Зайсан горы Саур, начинаясь в Китае, с западной стороны Улюнгирского озера, протянулись с востока на запад между глубокими, большими котловинами. Южная сторона котловин – Конырские горы (на территории

Китая). Котловины образовались в результате тектонических движений земной коры. Тарбагатай в виде нескольких невысоких гор расположился по соседству с хребтом Саур. Их разделяют реки Кандысу и Верхний Емель (на территории Китая). Протяженность Саура 110 км. На территории Казахстана находится только его сравнительно короткая северная часть длиной 60–65 км. Саур соединен с горным хребтом Маныpак. Северное подножье Саура упирается в озеро Зайсан и Черный Иртыш, а южное граничит с впадиной Шиликты. Тарбагатай расположен на западе Саура и соединяется с хребтом Чингизтау (250 км). Он ниже, но длинее Саура (300 км). Его ширина 30–50 км.

Рельеф и геологическое строение, полезные ископаемые. Наиболее высокая точка горной системы Саур–Тарбагатай находится в Сауре – пик Музтау (3816 м). Северный склон Саура по геологическому строению схож с Алтайскими горами, а южный склон крутой и скалистый – со среднеазиатскими горами.

На территорию Казахстана заходит западная часть Тарбагатая (от перевала Хабар Асу до реки Аягуз), а также северный склон его восточной части. Южный склон восточной половины Асу остается на китайской стороне. Тарбагатай не очень высок. Его средняя высота над уровнем моря 2000–2200 м. Самая высокая точка – Тастау – достигает 2992 м. По структуре Тарбагатай сложен складчатыми блоками. Вершины гор сильно сглажены. Им свойственны выровненные плоские участки. Склоны гор слабо расчленены ущельями. Там нет ледников.

Саур–Тарбагатай, в основном, сложен породами палеозоя (глинистыми и кристаллическими сланцами, песчаником, известняком и конгломератом). В Сауре широко распространены эффузивные породы, которые представлены порфирами и порфиритами. А в Тарбагатае часто встречаются известняки, много гранитов, которые отсутствуют на Сауре.

Уйденeнское водохранилище, созданное руками человека на высоте 820 м над уровнем моря.

Уйденeнское водохранилище построено в 1967 году всего в 14 км от города в каньоне горной реки Уйдене протекающей по северным отрогам Саурского хребта. Плотина Уйдене является одновременно моделью плотины Нурекской гидроэлектростанции, что на реке Вахш в Таджикистане. Каменно–набросная плотина высотой с 20–этажный дом производит огромное впечатление, особенно зрелищно выглядит вырывающийся из–под плотины мощный напор воды, играющий на солнце всеми цветами радуги. Шум падающей воды слышен далеко от этого места.

1.4 Инфраструктура и туризм

В Зайсанском районе 28 школ из них: 18 средних, 8 основных, 2 начальные.

Количество учащихся – 6103, учителей – 834.

В Зайсанском районе в селе Сартерек ведет работу оздоровительный центр «Байтерек».

Основная отрасль экономики – сельское хозяйство.

Сельхозформирования занимаются поливным и богарным земледелием, выращиванием традиционных видов скота, птиц, мараловодством.

В структуре промышленности района:

– горнодобывающая промышленность составляет 62%;

– обрабатывающая промышленность – 29%;

– производство и распределение электроэнергии, воды – 9%.

Горнодобывающая промышленность района представлена предприятием ТОО «Сайкан», которое занимается добычей угля и с 2011 года.

ТОО «Тарбагатай Мунай», которое занимается разведкой месторождения нефти и газа. ТОО «Сайкан» обеспечивает твердым топливом все бюджетные учреждения и население района.

Основу обрабатывающей промышленности составляют следующие отрасли:

– производство пищевых продуктов, включая напитки – (29,1%);

– производство прочих неметаллических минеральных продуктов – (70,9%).

Производством строительных материалов занимается ТОО «СМУ–Шыгыс». Предприятием производится плитки, плиты, кирпичи и изделия аналогичные из цемента, бетон или камня искусственного.

В производстве пищевых продуктов, включая напитки, участвуют малые предприятия, крестьянские хозяйства и индивидуальные предприниматели. Численность работающих в промышленности 279 человек (5,7% от занятого населения).

Земельные ресурсы:

– земли сельскохозяйственного назначения – 425057 га;

– промышленности – 3385 га;

– земли запаса – 404428 га.

Трудовые ресурсы

Численность трудовых ресурсов, занятых во всех сферах экономической деятельности.

На крупных, средних предприятиях в организациях и учреждениях – 2471 человек, в малых предприятиях – 131 человек, в фермерских хозяйствах – 1 человек, в крестьянских хозяйствах – 1248 человек, в общественных организациях – 0 человек, количество ИП – 1140 человек, количество наемных работников у ИП – 1167 человек, зарегистрированных безработных – 226 человек, незанятое население – 17900 человек.

В КГУ «Зайсанский технический колледж» обучаются 294 студента по специальностям:

– тракторист–машинист сельскохозяйственного производства – 123;

– электросварщик – 24;

– электромонтер по обслуживанию электрооборудования – 20;
– санитар ветеринарный – 53;
– бухгалтер – 25;
– штукатур – 24;
– каменщик – 25.

Инвестиционный потенциал

В 2013 году планируется реализация следующих инвестиционных проектов Зайсанского района:

– «Строительство магистрального газопровода Сарыбулак–Майкапчагай» (ТОО «Тарбагатай Мунай»). Стоимость проекта – 27 млрд. 500 млн.тенге. В проекте магистрального газопровода предусмотрен отвод для подачи газа в населенные пункты Зайсанского района – г. Зайсан и 9 населенных пунктов Зайсанского района.

– «Модернизация и расширение железобетонных изделий» 2–ая очередь (ТОО «СМУ Шыгыс»). Стоимость проекта 434,65 млн. тенге. Цель проекта: выпуск напорных железобетонных труб большого диаметра. Проект реализуется в рамках Государственной программы «Производительность 2020».

– «Строительство гостиничного комплекса, приобретение оборудования» (ИП Келгенбаев Ш.), стоимость проекта 50,09 млн. тенге.

Туристический потенциал

В Зайсанском районе на учете состоит 6 исторических, 21 культурно–архитектурных памятников культуры. На сегодняшний день мест для отдыха, санаториев, заповедников имеющие историческое значение не существует. В районе имеются заповедники: «Маныракские горы» – Кенсайском сельском округе, Каратальские пески «Бозайгыр» – Каратальском округе, Чиликтинские курганы – Чиликтинском сельском округе и ледник «Муз–тау» – Сартерекском сельском округе Зайсанского района.

В районе функционирует 9 гостиниц, 1 спортивно–оздоровительный лагерь «Байтерек» и 6 парков отдыха и развлечений.

1.5 Процесс создания сайта

Процесс создания сайта – очень трудоемкая вещь. Разработка проекта – первый, и один из самых важных этапов. В процессе создания сайта участвуют специалисты совершенно разных направлений, на основе проекта каждый участник процесса точно определит объем и поставленные перед ним задачи. Процесс написания проекта подразумевает активное участие разработчика и заказчика. Только после разработки проекта можно приступать к работе по созданию сайта. Ниже приведена наглядная поэтапная схема (Рисунок 1.1).

Первый этап – постановка задачи.

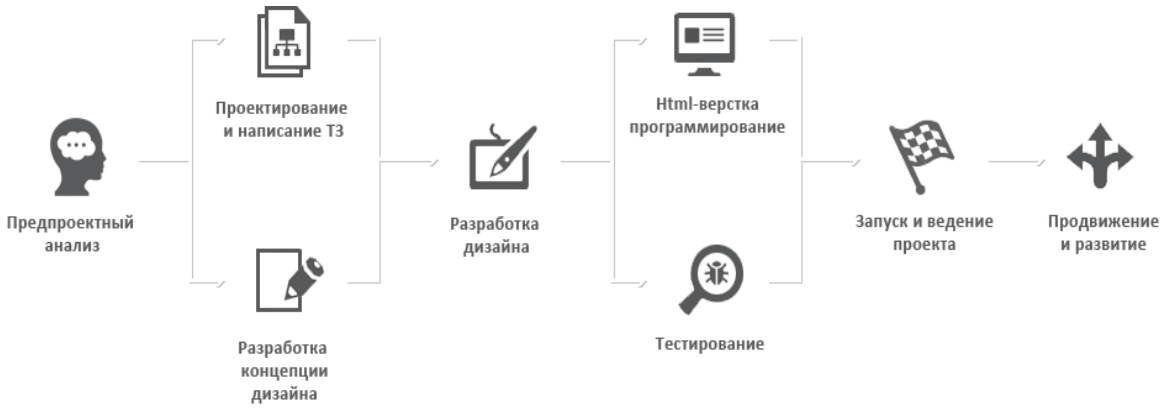

Рисунок 1.1 – Схема этапов разработки сайта

Перед тем как начать разработку сайта или перед тем как заказывать его, нужно определиться с его тематикой, с целью его создания и ожиданиями от его запуска во всемирную сеть.

Сайты создаются для разных целей и для достижения этих целей существуют разные способы. После определения того, что же вам необходимо, можно правильно подобрать специалиста для создания вашего сайта. На этом этапе создается структура будущего сайта и проектируется бизнес–логика.

В рамках обсуждения проекта выяснятся, за какой срок это можно реализовать решение задачи и какой бюджет потребуется. В результате выполнения 1–го этапа вы получаете описание структуры сайта, схему навигации по сайту и технические требования к сайту.

Обычно 1–й этап требует 8–15 дней на выяснение необходимых деталей и получение материалов, необходимых для постановки задачи на разработку сайта и написание полноценного технического задания.

Иногда сама постановка задачи является серьезной проблемой, решение которой требует времени и ресурсов. Например, такая ситуация возникает, если вам хочется сайт, который автоматизирует ваше производство или включает специфический Интернет–магазин и т.д. Таким образом, в случаях, когда для постановки задачи на разработку сайта требуется предварительное исследование и описание бизнес–процессов, этот этап вносится в договор и требует дополнительных ресурсов [1].

Последующий, второй этап – это разработка дизайна веб–сайта.

После утверждения постановки задания на разработку сайта наступает второй этап – разработка дизайна сайта. Для начала можно воспользоваться идеями заказчика, взяв ее за основу. Далее с течением обстоятельств и направлением изменения тематики сайта можно и подгонять к нему дизайн.

Дизайн должен учитывать тематику сайта. Чтобы все было в взаимодействии и было направлено на правильное восприятие пользователем. Дизайн веб–ресурсов имеет ряд особенностей и кардинально отличается от всех остальных направлений визуального оформления. Для разработки качественного дизайна, обязательно нужно знать основные языки, используемые при написании сайтов и технологии продвижения в поисковых

системах, только при соблюдении всех правил можно рассчитывать на успешное развитие и качественную работу веб–ресурса.

Дизайном сайта занимается веб–дизайнер. Который создает «нарезку сайта» (первоначальную верстку) в виде картинок и схемами с направлениями ссылок и нужных эффектов. Этот этап обычно занимает 1–2 недели.

Так же можно воспользоваться готовыми шаблонами, коих очень много в бесплатном варианте в интернете. Если постараться, то можно найти очень хороший дизайн, подходящий по цветовой гамме и навигации к вашей тематике. Преимуществом выбора готового дизайна является то, что вы экономите очень много времени и денег. Если учитывать то, что вы могли бы нанять дизайнера, которому нужно было бы оплачивать почасовую работу. Так же вы можете приобрести готовый дизайн–шаблон в любое время суток.

Если же посмотреть на вопрос немного шире, то можно сказать, что возможность выбирать дизайн помогает лучше представить, какой будет ваш сайт, что вы от него ждете, какая будет структура и для чего он вам вообще. Вы заранее будете представлять, как примерно будет выглядеть веб–сайт. От дизайна сайта зависит многое, например, восприятие пользователями важной информации.

Третьим, самым ответственным этапом является интеграция дизайна с системой управления.

Этот этап возникает, если вы хотите получить сайт, который можно отнести к категории "продвигающих" (подробнее о типах сайтов). Этот сайт по определению требует некой программной системы управления, поскольку при большом количестве обновляемого материала ручная публикация будет съедать массу времени вашего персонала. Как правило, сроки программирования зависят от технических особенностей и функциональных возможностей сайта (чем больше функциональные возможности сайта, тем сложнее его программировать). Качество программирование зависит от уровня специалиста.

Хорошая система управления сайтом помимо этого также решает задачи анализа аудитории на вашем сайте, задачи целевого воздействия на конкретные части этой аудитории, а также предоставляет другие функциональные возможности (подробнее о системах управления сайтами).

Систему управления сайтом так же можно написать самостоятельно, не надеясь на готовые решения. Поскольку готовые решения обладают рядом минусов, таких как: необходимость в подгонке кода под ваш сайт или же наоборот, подгонке вашего сайта к данной системе управления сайтом; необходимость изучения степени защищенности от хаккерских атак; так же требуется хорошее ознакомление с контентом и возможностями данной системы управления. Если же вы решились на разработку собственной системы управления, как это делают крупные компании, то для этого бросаются основные силы и бюджет. Разработка принципиально новой системы – очень трудоемкий процесс. Он охватывает многие задачи и решения.

Третий этап обычно занимает от 5–6 дней. В сложных случаях, когда помимо интеграции дизайна приходится заниматься определенной

автоматизацией работы сайта, это может занять более длительный срок, который оговаривается заранее.

Четвертый этап – это размещение готового сайта в Интернете.

Перед публикацией сайта, необходимо провести его тестирование на ошибки. Основной задачей, перед размещением сайта, является выбор его адреса в сети (домена). Обычно, при выборе домена, заказчики основываются на названии компании либо организации (торговой марки и т.д.). Поскольку это может оказать огромную помощь при ассоциации пользователей между вашим сайтом и продукцией (услугами). Так, например, поскольку наша компания называется "Инфо–Эксперт", то и название домена было выбрано info–expert.ru.

Ключевой задачей периода «перед размещением сайта в интернете» является поиск и места для размещения вашего готового к использованию сайта. Место для размещения сайта, которое имеет сервер, постоянно подключенный к интернету, называется хостингом.

Физическое расположение сервера не имеет значения, поэтому выбор хостинга зависит только от его цены, условий использования и длительности пребывания вашего сайта в сети.

Четвертый этап занимает от 2–3 дней. За это время происходит внедрение сайта и исправление его недостатков и различных ошибок. Четвертый этап очень ответственный, поскольку если неправильно настроить сайт, то это может привести к большим ошибкам и неправильному функционированию сайта. Или же, если не заметить ошибки при размещении сайта, то это может привести к плачевным последствиям.

Предпоследним этапом является заполнение сайта материалами.

Помимо разработки дизайна сайта и наладки системы управления содержимым сайта возникает задача наполнения сайта материалами. Иногда какие–то материалы у заказчика уже есть (готовые рекламные буклеты, статьи, материалы для выставок и т.д.) и их только надо привести в форму, предназначенную для публикации на сайте, а иногда никаких готовых материалов нет, а есть только общее представление и пожелания о таких материалах. Материалы для сайта должны быть уникальными, чтобы не быть однообразными и похожими на другие сайты, чтобы сайт мог конкурировать в предоставлении информации с другими похожими сайтами. Так же это может привлечь разнообразную аудиторию. Срок исполнения опять же зависит от сложности проекта, объема информации, который надо расположить на сайте, и того вида, в котором Заказчик ее представил. Если информация представлена в электронном виде, а графика не требует дополнительной обработки (например, изменение размера, добавление дополнительных элементов и проч.), то наполнение сайта происходит достаточно быстро. Если необходима дополнительная работа, как например, поиск или набор текста, сканирование фотографий или создание рисунков, то этап наполнения сайта может стать одним из самых длительных.

Если же нет готовой информации, для вставки в сайт, то нужно будет нанимать редактора, для составления правильных статей и корректного

предоставления информации. Естественно это будет стоить дополнительных затрат. Если же заказчик сам будет писать все статьи и прочую информацию, то можно будет ограничиться лишь ее редактированием и правильным размещением на сайте. Правильное содержание сайта и хорошая подача информации может привлечь больше клиентов и пользователей сайта.

Последний и самый продолжительный этап – это поддержка сайта.

Поддержка сайта, самый продолжительный по времени этап. Который занимает весь жизненный цикл сайта. Обычно поддержкой сайта занимается сам заказчик, но в некоторых случаях он обращается к веб–разработчику, который будет следить за правильной работой сайта и отвечать за ее работоспособностью. В поддержку сайта входит и ее продвижение. К нему можно отнести создание новых разделов сайта, ее автоматизация, увеличение производительности, привлечение новых клиентов и расширение существующей аудитории. Обычно все это проводится за счет рекламы сайтов, а так же размещением названий сайтов на заметных местах связанных с самой организацией. Все зависит от типа сайта, например: сайт визитка, после разработки и размещения на хостинге не требует особой поддержки, так как информация на нем обновляется не очень часто. Интернет магазин наоборот, ежедневно нужно заниматься его продвижением, и удержанием позиций в поисковых системах. Так же под поддержкой иногда подразумевается несущественное изменение (усовершенствование) дизайна или технической части сайта. Простыми словами к поддержке относятся все вопросы по содержанию сайта после его разработки и первичной настройки.

Для того чтобы сайт могли находить пользователи по определенным запросам его необходимо внести в поисковые системы и каталоги сайтов. Но не все так просто как кажется на первый взгляд, на сегодняшний день конкуренция в интернете очень большая, и для того чтобы Ваш сайт занял наивысшие места в поисковиках ему необходимо поисковое продвижение. На основе многочисленных исследований можно сделать вывод, что основная масса пользователей не просматривает сайты, которые находятся на второй странице выдачи поисковых систем, соответственно чтоб сайт был максимально эффективен, он должен находиться в первой десятке выдачи. Даже если сайт очень хорошо оптимизировать то, вероятность мала, того, что он займет нужные позиции в поисковых системах, в этом случае выход только один дополнительный комплекс мер по его продвижению.

2 Обзор технологий для разработки веб–сайта

Как было раскрыто в предыдущей главе, можно точно сказать, что процесс создания и разработки веб–сайтов включает в себя:

1 Утверждение первоначального технического задания на создание веб–сайта.

2 Определение структурной схемы сайта – расположение разделов, контента и навигации.

3 Веб–дизайн – создание графических элементов макета сайта, стилей и элементов навигации.

4 Разработку программного кода, модулей, базы данных и других элементов веб–сайта, необходимых в проекте.

5 Тестирование и размещение сайта в сети Интернет.

Эти этапы разделены между собой по типу работы и по персоналу, ответственному за их реализацию. К каждому этапу привязывается один сотрудник, в целом весь процесс будет на руках у группы сотрудников, разрабатывающих сайт совместными усилиями. Самым ответственным моментом при разработке веб–сайта представляется первый этап, потому, что от него зависит весь ход дальнейшего проектирования сайта. От этого этапа зависит выбор технологий, серверов, хостинга, первичного дизайна.

Немаловажную роль при разработке веб–сайта, занимает написание программного кода, как отмечалось выше, от этого зависит работоспособность сайта и его безопасность. Этот этап тоже является ответственным и требует особого внимания именно в отношении безопасности. Для целостности данных и правильного написания кода как минимум нужен разработчик и тестировщик. Первый будет писать весь программный код, тогда как второй будет тестировать его на ошибки и исправлять их. Это является примером эффективного использования ресурсов во время проектирования сайта. Как говорится одна голова – хорошо, а две еще лучше. Если же разработкой и тестированием проекта занимается один человек, то ему следует разделить этап разработки на два: написание кода и его тестирование.

При разработке некоторых проектов основной упор падает именно на технологии разработки сайтов. Во всех этапах используются разнообразные программы и технологии. Все технологии, которые используются для разработки веб–сайтов, имеют свои исключительные преимущества и недостатки. Среди них существуют незаменимые программы и языки, которые используются для всех существующих и разрабатываемых сайтов.

Также не стоит игнорировать и технологии связанные с дизайном сайтов. Они могут обеспечить хорошее продвижение сайта в рейтинге поисковых запросов, что соответственно увеличивает количество пользователей. Сравнение и определение наилучшего и выигрышного варианта связки технологий очень трудоемкий процесс, который занимает большое количество времени и требует особого внимания. Их описание и сравнение приводятся в

данной главе.

2.1 Краткая характеристика используемой ОС

В данном дипломном проекте была выбрана и использована операционная система Windows 7 версии. Windows 7 и Windows 8 – последнее воплощение графических операционных систем для использования на компьютерах типа IBM PC и совместимых с ним. По мере проникновения на рынок, за последнее десятилетие, Windows XP, Windows Vista, Windows 7 почти полностью вытеснили всех имевшихся конкурентов и стали, фактическими, эталонами операционной системы для персональных компьютеров. К сожалению, последняя версия операционной системы, Windows 8, кардинально отличается от предыдущих, таким образом, приводя пользователей Windows в замешательство. Последняя версия операционной системы показывает насколько отличаются представления разработчиков Microsoft и их пользователей в отношении направления улучшения системы.

Действительно, Windows 8 не лучшая версия системы. По замечаниям пользователей не многие решились на освоение в корне нового подхода к размещению контента. Дизайн больше напоминает операционные системы, созданные для Windows Phone. Большинство пользователей Windows переходили обратно на Windows 7. На сегодняшний день Windows 7 является самой качественной и удобной версией ОС Windows.

Операционная система не может реализовывать многозадачность без управления памятью. Так как одни программы запускаются, а другие завершаются, память фрагментируется. Система должна быть способной объединять свободное пространство. Для этого требуется, чтобы система перемещала в памяти блоки программ и данных.

Windows – это графический интерфейс, и программы для Windows могут полностью использовать графику и форматированный текст, как на дисплее, так и на принтере.

Графический интерфейс не только более удобный для восприятия, но он может также обеспечить пользователю высококачественное отображение информации.

У программ, написанных для Windows, нет прямого доступа к аппаратной части устройств отображения информации, таких как экран и принтер. Вместо этого Windows включает в себя язык графического программирования, называемый графическим интерфейсом устройства, который облегчает создание графики и форматированного текста. Windows абстрагируется от конкретного устройства отображения информации. Программы, написанные для Windows, будут работать с любым типом дисплея и любым типом принтера, для которых имеется в наличии драйвер Windows. В программе нет необходимости задавать тип используемого в системе оборудования. Доступность всей оперативной памяти, динамическое подключение библиотек дают преимущества программирования для Windows.

Эта операционная система была выбрана по многим критериям. Во–первых, она является доступной для нас. Так же ее интерфейс нам знаком с самого первого времени работы с компьютером. Во–вторых, буквально все существующие и необходимые программы написаны именно под данную операционную систему. В–третьих, работа с этой ОС очень проста и логична. Она подходит под любой компьютер. И последним аргументом, в пользу этой операционной системы является то, что это продукция Microsoft. Так же следует учитывать то, что все выбранные в дальнейшем технологии связанны именно с этой корпорацией, и взаимодействуют между собой на ура.

2.2 Стандартный набор: HTML, CSS, JavaScript и jQuery

Все сайты, которые находятся в интернете, созданы на основе языка разметки HTML (hyper text markup language). Поскольку он является основой, так называемым каркасом и связующей нитью всех страниц и разделов сайта. Без него невозможно создать сайт.

Так же к основным языкам, требуемым для разработки сайтов, относится CSS (cascade sheets style). Главной задачей, которой является создание единого стиля и оформления сайта, связующего все его разделы и отражающего тематику сайта.

Для хорошего отображения содержимого вашего сайта, а так же для правильной или необычной подачи информации для пользователей можно использовать различные анимации. К анимации можно отнести: открытие изображений в большем размере на одной и той же странице, проигрывание музыки, создание других разнообразных эффектов. Такие эффекты, зачастую привлекают внимание аудитории и оставляют хорошую связь именно с вашим сайтом. Чтобы сделать эффекты, без которых уже не обходятся современные сайты, необходимо использовать скриптовый язык программирования – JavaScript. В недавнее время в пользование веб–разработчикам прочно вошла независимая библиотека, содержащая различные виды готовой анимации. Эта библиотека была разработана специально, для упрощения работы с JavaScript. jQuery содержит функциональность, которая широко используется при разработке.

2.2.1 Язык гипертекстовой разметки HTML

Язык HTML (Hyper Text Markup Language – это язык гипертекстовой разметки), который лежит в основе всех существующих сайтов. Без всяких сомнений можно сказать, что HTML – это главное незаменимое и связующее звено при сайтостроении. В настоящее время используется последняя версия языка – HTML 5.

При помощи HTML можно создавать два вида веб–страниц: статические и динамические. Статическим (неизменным) сайтом можно назвать сайт, у которого все его содержимое остается неизменным во времени до и после загрузки и обновления веб–страниц, а так же содержимое сайта остается таким

же неизменным при навигации и передвижению мыши по странице. Однако, если рассматривать сайты, имеющиеся в пользовании во всемирной паутине, можно сделать вывод, что статичных сайтов довольно так и мало. Побывав на любом крупном сайте, можно вмиг заметить, что на его страницах есть элементы, которые реагируют на щелчки либо движения мыши, на некоторых сайтах даже можно встретить анимацию с изображениями, всплывающие окна и пр. Такие страницы называются динамическими. Для их создания используются небольшие программные коды, которые внедряются в HTML–код данной веб–страницы. Эти программные коды называются сценариями (скриптами). Самыми распространенными скриптовыми языками считаются JavaScript VBScript.

При написании кода на языке HTML–документов существуют определенные правила. Основой языка является элемент. Элементы можно назвать основой построения веб–сайта. Все, что создается на странице веб–сайта, будет сделано при помощи элементов. HTML документы могут быть созданы с помощью различных текстовых редакторов или специальных HTML–редакторов. Выбор редактора, который будет использоваться для создания HTML–документов, зависит исключительно от понятия удобства и личных пристрастий каждого автора. Например, HTML–редакторы, такие, как NVU, позволяют создавать веб–документы не только глядя на код и генерируя его на странице браузеров, а наблюдая всю ситуацию и происходящие изменения сразу же, по изменении кода. В последние годы такие технологии получили широкое распространение так называемые визуальные редакторы WYSIWYG. С другой стороны, большинство традиционных средств, для создания документов имеют конвертеры, позволяющие преобразовывать документы к формату HTML.

Тег (дескриптор) – это специальный код, вставляемый в программный текст, для форматирования элементов HTML–документа. Это основной элемент кодирования, принятый в стандарте HTML. Теги заключаются в угловые скобки <>. Например, тегом абзаца является <P>, а тегом горизонтальной линии – <HR>.

Все теги HTML начинаются с "<" и заканчиваются символом ">". Название элемента помещается в угловые скобки, например <h>. Полученное выражение называется тегом. В данном случае это открывающий тег. Иногда необходимо задать парный закрывающий тег, который записывается так: </h>.

В основном парные теги используются при форматировании текста, они задают начало и конец блока форматирования. В языке есть свои исключения, при которых местами закрывающие теги не требуются, а иногда их можно просто опускать, то есть пропустить, однако для корректной обработки документа рекомендуется всегда дописывать закрывающий тег.

Существует два типа тегов: теги–контейнеры и одинарные теги.

Контейнеры. Это пара тегов, которая состоит из открывающего и конечного (закрывающего) тегов. Начальный тег имеет вид <тег>, где вместо «тег» вставляется имя реального HTML–тега. Конечный тег имеет вид </тег>.

Контейнеры предназначены для хранения некоторой информации, например текста или других HTML–тегов. Поэтому между начальным и конечным тегами заключено содержимое контейнера. Например, элемент, представляющий собой отформатированный текст, заключается между тегами <PRE> и </PRE>.

Пустой тег. Отличается от контейнеров в том, что они не содержат никакой информации. У них имеется лишь начальный тег. Пустой дескриптор обычно выполняет самостоятельную задачу, не связанную с конкретным текстом. Например, тег <HR> создаёт горизонтальную линию.

Элемент HTML–документа – это открывающий и закрывающие теги контейнера вместе с находящимся между ними содержимым. Элементом может быть изображение, фрагмент текста, форма, таблица, список, ссылка, текстовое поле, кнопка и даже заголовок документа или его основная часть (тело).

Если рассматривать функции тегов, то их можно объяснить таким образом: открывающий тег форматирование, а закрывающий выключает. При этом отличием в записи тегов, кроме постановки символа «/» в закрывающем теге, является присутствие различного рода атрибутов у первого.

Примером использования закрывающего тега является работа с элементом h, который обозначает заголовок второго уровня:

```
<h2> Какой либо текст </h2>
```

Элемент <IMG>, который добавляет картинку на сайт, не требует наличия закрывающего тега. По назначению элемента зачастую можно догадаться, требуется ли ему закрывающий тег.

Элементы применяются для того, чтобы сказать браузеру, какой блок вы хотите видеть в определенном месте страницы, а также какую информацию этот блок должен содержать. Кроме того, браузеру нужно сообщить, как отображать эту информацию. Для этого используют атрибуты элементов.

В начале каждого HTML–документа нужно помещать строку объявления такого рода:

```
<! DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01 Transitional//EN" "http://www.w3.org/ TR/html4/loose.dtd">
```

Эта строка дает браузеру общую информацию об HTML–документе.

Каждый гипертекстовый документ начинается и заканчивается тегом <HTML>. Все элементы относящиеся к разметке должны находиться внутри тегов <HTML> и </HTML>. После того как для документа создана основа, внутри нее нужно создать заглавную область.

Вторым тегом, как правило, следующим после тега <HTML> является теги <HEAD> и </HEAD>. Информация, вводимая в элемент HEAD, не отображается в окне браузера, а помогает ему в обработке страницы. Элементы, находящиеся внутри элемента HEAD, играют немаловажную роль: данные,

содержащиеся в них, помогают браузеру в обработке страницы, а поисковым системам – в индексации документа.

Внутри тегов TITLE должно находиться название страниц веб–сайтов. В переводе слово TITLE значит «название». Согласно спецификации HTML в содержимом элемента HEAD обязательно наличие элемента TITLE, причем в единственном числе. Элемент требует наличия закрывающего тега </TITLE>. Текст, содержащийся между открывающими и закрывающими тегами, и будет отображаться в строке заголовка окна браузера.

После тегов <TITLE> и </TITLE> следует тег STYLE, в котором определяются стили различных элементов веб–страницы. Таких элементов внутри заголовка страницы может быть несколько. Для задания стилей в документе HTML 5 применяется язык CSS. Он привязывается к данному гипертекстовому документу через ссылку.

Когда браузер получает веб–документ, он определяет, каким образом этот веб–документ должен быть интерпретирован. Простейший HTML–документ должен выглядеть так:

```
<HTML> ...тело документа... </HTML>
```

Заголовочная часть документа <HEAD>.

Как и в любом языке, HTML тоже позволяет вставлять в тело документа комментарии, которые сохраняются при передаче документа по сети, но не отображаются браузером. Комментарии могут встречаться в документе где угодно и в любом количестве. Синтаксис комментария:

```
<!--текст комментария-->
```

Теги тела документа идентифицируют отображаемые в окне компоненты HTML–документа. Тело документа может содержать ссылки на другие документы, текст и другую форматированную информацию.

Тело документа должно находиться между тегами <BODY> и </BODY>. Это та часть документа, которая отображается как текстовая и графическая (смысловая) информация вашего веб–документа.

В отличие от большинства текстовых процессоров, в HTML–документе обычно игнорируются символы возврата каретки. Физический разрыв абзаца может находиться в любом месте исходного текста документа (для удобства его читаемости). Однако браузер разделяет абзацы только при наличии тега <P>. Если вы не разделите абзацы тегом <P>, ваш документ будет выглядеть как один большой абзац.

Гипертекстовые ссылки являются ключевым компонентом, делающим веб–сайт привлекательным для пользователей и более функциональным. Добавляя гипертекстовые ссылки (далее – ссылки), делается набор документов связанным и структурированным, что позволяет пользователю получать необходимую ему информацию максимально быстро и удобно [6].

Существует два распространенных способа вставки графической информации в HTML–документы. Первый – это внедрение графической информации образов в документ, что позволяет пользователю видеть изображения непосредственно внутри других элементов документа, таких как тексты, ссылки и пр. Вторым способом вставки графической информации считается ее размещение в виде ссылки, отдельно от текстов и другой информации.

Ниже приведен пример структуры HTML–документа:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 3.2 Final//EN">
<HTML>
<HEAD>
<TITLE> Название документа, его заголовок
</TITLE>
</HEAD>
<BODY>
Тело документа, различные вставки
</BODY>
</HTML>
```

Если принять к сведению выше изложенную схему документа, можно сделать заключение, что HTML–документ содержит два основных блока – "заголовок" и непосредственно "тело документа". Заголовок HTML–документа размещается внутри тегов «HEAD», а тело документа в элементе «BODY».

Заголовок содержит программную часть HTML–документа, хотя чаще всего используется только для содержания его названия.

Тело документа – самая важная часть. Именно в нем находится все то, что отображается на странице: текст, картинки, таблицы. Соответственно, делаем вывод: большинство ваших HTML–экспериментов будет проводиться в пространстве между тегами <BODY> и </BODY>.

HTML является языком, используемым для структурирования содержимого страницы. По мере его развития, веб–дизайнеры начали поиски возможностей форматирования онлайн–документов. Чтобы удовлетворить возросшим требованиям пользователей, производители браузеров (тогда – NetScape и Microsoft) изобрели новые HTML–теги, такие, например, как <size>, которые отличались от оригинальных HTML–тегов тем, что они определяли внешний вид. Таким образом, в этапы разработки сайтов прочно вошел CSS. Он используется для форматирования структурированного содержимого HTML–документов.

2.2.2 Таблицы стилей CSS

Cascading Style Sheets (CSS) – язык таблицы стилей, используемый для описания отображения и форматирования документа, написанного на языке разметки. Используется для разработки интерфейса веб–страницы, написанного на HTML и XHTML, язык может быть применен к любому виду документа

XML, включая простой XML, SVG и XUL. CSS – спецификация краеугольного камня сети, и почти все веб–страницы используют таблицы стилей CSS, чтобы описать их дизайн.

CSS разработан, прежде всего, для того, чтобы позволять разделять содержания документа от его стиля. Включающие такие элементы как расположение, цвет и шрифты. Это разделение может улучшить доступность содержания документа, обеспечить гибкость и контроль над особенностями дизайна. Так же позволяет многостраничным веб–сайтам разделить форматирование страниц и уменьшить сложность и повторение в стилевом содержании (такое как описывание стилей).

CSS также позволяет одной и той же странице быть представленной в различных стилях для различных методов предоставления, такими как на экране, в печати, озвучивание информации.

CSS определяет приоритетную схему, какие стили будут применяться первыми, если имеется больше чем одно описание дизайна данного элемента. В этом так называемом каскаде приоритеты вычисляются и назначаются на элементы, так, чтобы результаты были предсказуемы.

Таблицы стилей (CSS) представляют собой огромный прорыв для проектировщиков веб–страниц, расширяя их способность улучшения вида их страниц. В среде, в которой была создана глобальная сеть, пользователей больше беспокоит содержание их веб–документов, чем их представление. В то время как намерения были хороши – чтобы улучшить представление Веб–страниц – методы для того, чтобы сделать, так имели неудачные побочные эффекты.

Таблицы стилей решают проблемы со стилями в то же время, они заменяют ограниченный диапазон механизмов представления в HTML. Таблицы стилей облегчают определение краев неиспользуемых пространств между текстовыми строками, цвета, используемые для текста и фона, размера шрифта и стиля и массы других деталей.

Например, следующая короткая таблица стилей CSS (сохраненная в файле "стиль.css"), выбирает текстовый зеленый цвет абзаца и чертит вокруг него красную границу:

```
P.стиль
{
color: green;
border: red;
}
```

Разработчики могут связать эту таблицу стилей со своим исходным HTML–документом посредством ссылок:

```
<!DOCTYPE HTML PUBLIC "-//W3C//DTD HTML 4.01//EN"
"http://www.w3.org/TR/html4/strict.dtd">
<HTML>
```

```
        <HEAD>
          <LINK href="стиль.css" rel="stylesheet" type="text/css">
        </HEAD>
        <BODY>
          <P  class="special">This  paragraph  should  have  special
green text.
        </BODY>
      </HTML>
```

HTML 5 поддерживает следующие особенности таблицы стилей.

Гибкое размещение информации о стиле.

Размещение таблиц стилей в отдельных файлах делает их доступными для повторного использования. Иногда полезно включать инструкции по представлению в сам HTML–документ, к которому они применяются, или сгруппировывать в начале документа. Чтобы облегчить управление стилем таблицу стилей выносят в отдельный документ.

Независимость таблицы стилей от определенных языков.

Эта особенность, не связывает HTML ни с каким другим языком таблицы стилей. Допускается диапазон только таких языков, которые будут использоваться, например простые для большинства пользователей и намного более сложные для меньшинства пользователей с узкоспециализированными потребностями.

Каскадирование

Эта способность позволяет информации о стиле из нескольких источников, применяться вместе. Храня их отдельно, таблицы стилей могут быть снова использованы, упрощая их написание и делающее более эффективным использование сетевого кэширования. Каскад определяет последовательность таблиц стилей, определяя их очередность. Права есть у тех документов, которые прилинкованы первыми, чем последующие. Не все виды каскадирования поддерживает таблица стилей.

Преимущества CSS

CSS позволяет уменьшить размер программного кода и сделать его более понятным.

CSS позволяет задавать такие параметры, которые нельзя задать только языком HTML. Например, убрать подчеркивание у ссылок.

CSS позволяет легко изменять внешний вид страниц. Представьте, вы сделали сайт из 50 страниц, на которых все заголовки синего цвета. Через какое–то время, вы захотели поменять синий цвет на зеленый. Вам придется пройтись по всем 50 страницам и поменять цвет в соответствующем атрибуте. С CSS вам придется сделать это только один раз, в таблице стилей. С CSS связана так называемая блочная верстка сайта.

Как и любой другой язык программирования, CSS имеет свой собственный синтаксис, т.е. правила по которым создаются таблицы стилей. В

CSS в отличие от HTML нет ни элементов, ни атрибутов, ни тегов. Основной структурной единицей здесь является правило, которое определяет, как будет выглядеть тот или иной элемент в документе.

Сначала записывается так называемый селектор, показывающий к какому HTML тегу (тегам) применяется то или иное свойство.

Далее, непосредственно за селектором, пишется блок объявления стилей, который обязательно заключается в фигурные скобки.

Каждое объявление в свою очередь состоит из свойства и его значения. После свойства ставится двоеточие. Правило может содержать в себе несколько объявлений. В таком случае они должны быть отделены друг от друга точкой с запятой, причем после последнего объявления точку с запятой можно не ставить.

2.2.3 JavaScript и jQuery

JavaScript (JS) – это динамический язык программирования. Который, обычно используется в качестве вспомогательной технологии, с помощью которой идет отображение на веб–браузерах, внедрение которой позволяет серверной стороне клиента взаимодействовать с пользователем. Управлять браузером, общаться асинхронно и изменять содержание документа, которое показано. Он также используется в программировании стороны сервера, для анимации и создании настольных и мобильных приложений.

Название и логотип «JavaScript» – это зарегистрированный товарный знак Oracle Corporation.

JavaScript – основан на прототипе скриптового язык с динамическим программированием и имеет первоклассные функции. Его синтаксис был похож на C. Ключевые принципы разработки в JavaScript заимствованы с различных языков программирования. Это – язык мультипарадигмы, поддерживающий объектно–ориентированное программирование, императив, и функциональные программные стили.

JavaScript был сформирован языковом стандартом ECMAScript и прежде всего, используется в качестве части отображения веб–страниц (сторона клиента JavaScript).

JavaScript был внедрен Бренданом Эйком, в то время работавшим в Netscape Communications Corporation. Кроме Брендана Эйка в разработке были задействованы: один из основателей Netscape Communications Марк Андресен и основатель Sun Microsystems Билли Джой. Группа была создана по причине ограниченности времени на реализацию проекта внедрения. Разработчики поставили себе цель, создать «язык для склеивания» составляющих частей веб–страницы: изображений, плагинов, различных апплетов, который был бы по душе веб–дизайнерам и веб–программистам, не обладающим высокой квалификацией в области программирования. Можно с уверенностью сказать, что разработчики языка полностью добились поставленной цели.

Изначальным названием языка было «LiveScript», и этот язык предполагал свое использование на обеих сторонах, как на стороне клиента, так же и для программирования со стороны пользователя.

На синтаксис нового языка оказали большое влияние языки C и Java. Так же одной из причин переименования названия языка на JavaScript явилось модное в то время, слово Java . Таким образом, 4 декабря 1995 года «LiveScript» был переименован в JavaScript и получил лицензию у корпорации Sun. Анонсирование нового языка, Netscape и Sun провели накануне выпуска второй бета–версии браузера NetscapeNavigator.

К сведению, раньше разработчики, впервые слышавшие о новом языке объектного скриптового программирования, ошибочно полагали, что это некоторая видоизмененная версия известного языка – Java. Действительно многие даже сейчас связывают их друг с другом. Но популярность языка в современном веб–программировании просто поражает. Почти все сайты используют данный язык.

JavaScript является объектно–ориентированным языком, но используемое в языке прототипирование обуславливает отличия в работе с объектами по сравнению с традиционными класс–ориентированными языками. Кроме того, JavaScript имеет ряд свойств, присущих функциональным языкам – функции как объекты первого класса, объекты как списки, карринг, анонимные функции, замыкания – что придаёт языку дополнительную гибкость.

Несмотря на схожий синтаксис с языком C, JavaScript имеет коренные отличия:

- объекты, с возможностью интроспекции;
- функции как объекты первого класса;
- автоматическое приведение типов;
- автоматическая сборка мусора;
- анонимные функции.

В языке отсутствуют такие полезные инструменты как:

- модульная система: JavaScript не предоставляет возможности управлять зависимостями и изоляцией областей видимости;
- стандартная библиотека: в частности, отсутствует интерфейс программирования приложений по работе с файловой системой, управлению потоками ввода–вывода, базовых типов для бинарных данных;
- стандартные интерфейсы к веб–серверам и базам данных;
- система управления пакетами, которая бы отслеживала зависимости и автоматически устанавливала их.

Синтаксис языка JavaScript очень похож на синтаксис C и Java, семантически язык гораздо ближе к Self, Smalltalk.

В JavaScript все идентификаторы зависимы от регистра, в названиях переменных можно использовать буквы, подчёркивание, и различные символы. Названия переменных не должны начинаться с цифр, для оформления

однострочных комментариев используются «//», многострочные и внутристрочные комментарии начинаются с «/*» и заканчиваются «*/».

JavaScript игнорирует все пробелы, символы табуляции и переход на новую строку, если только они находятся не внутри текстовой строки.

Все простые операторы должны заканчиваться символом «;» (точкой с запятой). Он служит для отделения операторов друг от друга, но может быть опущен, если операторы находятся в отдельных строках. Но советуется все–таки использовать этот символ «точки с запятой».

Есть несколько способов добавления кодов программы на JavaScript (JS) в HTML–документ. Первый способ – написание кодов на JS между тегами <SCRIPT> и </SCRIPT>. Другим способом является подключение внешнего файла с кодами JS с помощью тега <SCRIPT> и указания ссылки.

Третий способ – это использование кодов JS непосредственно в тегах HTML при задании обработчиков событий. И последним вариантом считается использование псевдо–протокола JavaScript: URL в тегах HTML.

Можно рассматривать первые два способа встраивания кода на JS в HTML–документ.

Включение JavaScript между тэгами <SCRIPT> и </SCRIPT>.

Для включения фрагментов программы на JavaScript или другом скрипте (объявлений переменных, описаний функций, операторов, вызовов функций и др.) обычно используется следующий шаблон:

```
SCRIPT TYPE="text/javascript" >
<!--Запись сценария
.   .   .
Фрагменты кодов JavaScript
.   .   .
// комментарий
-->
</SCRIPT>
```

Параметр TYPE задает скрипт (язык программирования), используемый в тэге–контейнере <SCRIPT>. Кроме этого, согласно рекомендациям организации W3C с помощью тэга <META> должен быть задан скрипт, принимаемый браузером по умолчанию. Поэтому, если при разработке сценария используется язык JavaScript, то параметр LANGUAGE можно не указывать.

Использование HTML–комментария (<!— —>) в шаблоне предназначено для маскирования JavaScript–сценария для тех браузеров, которые его не поддерживают (версии Netscape Navigator до 2.0 и Microsoft Internet Explorer до 3.0). Иначе Web–страница будет воспроизведена неправильно. При этом тэги <SCRIPT> и </SCRIPT> этими браузерами будут пропущены, поскольку браузеры игнорируют тэги, которые не могут распознать. Браузеры, которые позволяют интерпретировать JavaScript, HTML–комментарий распознают иначе: его начало (<!—) считается однострочным комментарием, а окончание (—>) игнорируется.

Для браузеров, которые не поддерживают JavaScript и другие скрипты, можно воспользоваться тэгом–контейнером <NOSCRIPT> </NOSCRIPT> для того, чтобы сообщить об этом пользователю:

```
<NOSCRIPT>
<B> Эта Web-страница содержит фрагменты JavaScript. Используйте браузер, поддерживающий этот язык.
</B>
</NOSCRIPT>
```

Хотя эти браузеры сейчас практически не используется, тем не менее, рекомендуется использовать при работе со сценариями на JavaScript предложенный шаблон встраивания программ. В первом примере реализован именно этот подход. Хотя в дальнейшем в целях более компактного изложения программ на JavaScript будет использоваться сокращенный вариант. Приведенный ниже:

```
<SCRIPT>
.  .  .
Фрагменты сценария JavaScript
.  .  .
</SCRIPT>
```

Подключения внешнего файла с JavaScript.

Для подключения внешнего файла с JavaScript–кодами используются те же самые тэги <SCRIPT> и </SCRIPT>, но в отличие от внутреннего встраивания тэг <SCRIPT> содержит параметр SRC, который задает URL–адрес внешнего файла с фрагментами JavaScript.

Отметим, что в случае использования параметра SRC в тэге <SCRIPT> закрывающий тег </SCRIPT> необходим, хотя данные, которые находятся между этими тегами – игнорируются. Их нельзя совместить в одном тэге–контейнере <SCRIPT> </SCRIPT> сразу и внутреннее и внешнее подключение кодов JavaScript.

jQuery – кросс–платформенная библиотека JavaScript, разработанная, для упрощения программирования на стороне клиента.

Она была разработана и выпущена в январе 2006 года в BarCamp Нью–Йорке, независимым разработчиком Джоном Резигом. В настоящее время развитием языка занимается команда разработчиков во главе с Дэйвом Метвином.

Используемый более чем в 80% сайтов. jQuery – самая популярная библиотека JavaScript, которую используют сегодня.

jQuery – бесплатное, общедоступное программное обеспечение, лицензируемое в соответствии с лицензией MIT.

Синтаксис jQuery разработан, для облегчения работы при разработке веб–страниц, для легкого и быстрого выбора элементов DOM. Для создания мультипликации, которые обращаются за событиями к приложениям на AJAX.

jQuery также обеспечивает разработчикам возможность, для создания программного расширения поверх библиотеки JavaScript. Что позволяет разработчикам создавать абстракции для простого взаимодействия с мультипликацией, новейшие эффекты и различные виджеты. Модульный подход в jQuery библиотеке, позволяет ей создавать сильные и динамические веб–страницы и веб–приложения.

Microsoft и Nokia применяют jQuery в связке с их платформами. Microsoft включает его в Visual Studio, для использования в пределах структуры ASP.NET, AJAX и Microsoft ASP.NET MVC.

jQuery имеет следующие особенности:

- выборка элементов DOM;
- пересечение DOM и ее модификация (включая поддержку CSS 1–3);
- манипуляция DOM, основанная на CSS, которое использует имя элементов узла и признаки элементов узла (id и класс) как критерии, чтобы построить селекторы;
- события;
- визуальные эффекты и мультипликации;
- AJAX–дополнения;
- парсинг JSON;
- JavaScript–плагины;
- различные утилиты.

В настоящее время существует огромное количество программных расширений jQuery, имеющихся в сети, которые открывают широкий диапазон функциональности, такие как дополнения AJAX, веб–сервисы, datagrid–ы, динамические списки, XML и инструменты XSLT, события, обработка cookies.

Существует одна примечательная и полезная функция – функция поддержки, которая проверяет, доступны ли определенные функции на браузере пользователя.

Традиционно, разработчики обращаются к запросам браузеров – для определения, какой веб–браузер конечный пользователь использует. Они основаны на информации, предоставленной самим браузером – чтобы исправить найденные проблемы. Используя функцию сервисной поддержки jQuery, есть возможность проверки, доступности определенных функций для пользователей, и легко создайте приложения, которые ухудшаются изящно на более старых браузерах или не послушных со стандартами.

jQuery UI – библиотека JavaScript, которая обеспечивает эффекты для взаимодействия простого уровня и мультипликации, передовые эффекты и высокого уровня, темы и виджеты, построенные поверх библиотеки jQuery JavaScript, который может использоваться для построения интерактивных веб–приложений. Эта библиотека была выпущена в сентябре 2007. О ней сообщил в своем в блоге Джон Резиг, на jquery.com.

Все виджеты jQuery UI – полностью используют объединенный, скоординированный механизм темы.

Draggable – предоставляет элементу возможность быть перемещенным с помощью мыши. Droppable – контролирует, где перетаскиваемый элемент может быть брошен. «Resizable» – предоставляет возможность динамически изменять размеры элемента с помощью мыши. «Selectable» – предоставляет возможность выделять один или несколько элементов пользовательского интерфейса из группы. «Sortable» – предоставляет возможность сортировки для группы элементов.

Виджеты: «accordion» – виджет «аккордеон»; «autocomplete» – поле ввода с автозаполнением; «button» – улучшенная кнопка, может также быть флажком (check box) или радио–кнопкой (radio button); все виды кнопки могут располагаться на панели инструментов (toolbar); «datepicker» – виджет для выбора даты или диапазона дат; «dialog» – диалоговое окно, которое может иметь любое содержимое; «progressbar» – полоса прогресса; «slider» – слайдер; «tabs» – вкладки.

Эффекты

Color Animation – анимирует изменение цвета компонента.

Toggle Class, Add Class, Remove Class, Switch Class – анимируют изменение набора класса стилей компонента.

Effect – множество эффектов связанных с появлением и исчезновением компонентов интерфейса.

Toggle – функция переключения между режимами видимости компонентов с использованием эффектов.

Hide – функция исчезновения компонента с использованием эффектов.

Show – функция появления компонента с использованием эффектов.

Position – установка положения элемента относительно позиции другого элемента (выравнивание).

Для всех компонентов интерфейса jQuery UI используется единая тема оформления, которая может быть выбрана из нескольких существующих, либо легко создана разработчиком.

Ниже приведен пример использования данной библиотеки:

```
<script type="text/javascript">
// #draggable draggable
   $(function () {
   $("#draggable").draggable();
    });
</script>
  <div id="draggable" class="ui-widget-content">
   <p>Drag me around</p>
  </div>
```

Код, описанный выше, определяет действие объекта, во время указания на него мышью.

jQuery Mobile – сенсорно–ориентированный веб–фреймворк, который был разработан командой jQuery, создателей jQuery. Разработка сфокусирована на кросс–браузерности с уклоном в сторону смартфонов и планшетов. JQuery Mobile совместим с прочими мобильными фреймворками, такими как PhoneGap, Worklight и другими. Мобильный jQuery служит сильной тематической основой, которая позволяет разработчикам настраивать цветовые схемы и определенные CSS особенности UI.

2.3 XML–файлы

Первоначально разработанный для решения крупномасштабных задач электронных изданий, язык XML, также играет важную роль при обмене самых разнообразных данных в Интернете. Спецификация XML описывает язык и ряд вопросов, касающихся кодировки и обработки документов.

Система на основе XML может обеспечить использование большего количества типов данных в большем количестве программ и на большем количестве компьютеров. Она способна автоматизировать процедуры ввода данных, помогая быстрее создавать документы.

XML–систему нельзя просто купить и установить. Для развертывания XML–системы потребуется провести совместную работу с ИТ–подразделением или кем–то, кто разбирается в проектировании и создании различных компонентов XML–системы.

Типичная XML–система состоит из файлов трех типов:

– XML–данные – это данные пользователя плюс XML–теги, которые описывают смысл и структуру данных;

– XML–схемы задают правила, определяющие, какие объекты могут, а какие не могут присутствовать в файлах данных. Например, схема может запрещать пользователям вводить слова в поле даты;

– XML–преобразования позволяют использовать данные в разных программах или файлах. Например, одно преобразование может добавлять данные по продажам к электронной таблице, в то время как другое будет вставлять те же данные в текстовый документ.

Главным компонентом системы XML являются данные. Файлы данных XML содержат ваши данные и набор кодов, называемых тегами, которые поясняют значение данных. Это одна из причин адаптивности и полезности языка XML, по которой его называют расширяемым.

2.3.1 Язык разметки XML

XML (англ. eXtensible Markup Language – расширяемый язык разметки; произносится [экс–эм–эл]) – рекомендованный Консорциумом Всемирной паутины (W3C) язык разметки. Спецификация XML описывает XML–документы и частично описывает поведение XML–процессоров (программ, читающих XML–документы и обеспечивающих доступ к их содержимому).

XML разрабатывался как язык с простым формальным синтаксисом, удобный для создания и обработки документов программами и одновременно удобный для чтения и создания документов человеком, с подчёркиванием нацеленности на использование в Интернете. Язык называется расширяемым, поскольку он не фиксирует разметку, используемую в документах: разработчик волен создать разметку в соответствии с потребностями к конкретной области, будучи ограниченным лишь синтаксическими правилами языка. Сочетание простого формального синтаксиса, удобства для человека, расширяемости, а также базирование на кодировках Юникод для представления содержания документов привело к широкому использованию как собственно XML, так и множества производных специализированных языков на базе XML в самых разнообразных программных средствах.

XML является подмножеством SGML. В принципе, XML является базой данных с древовидной структурой. Реляционная модель данных в этой структуре не поддерживается. База данных в роли XML можно использовать только с человеком–оператором.

XML представляет собой гибкий способ создания общих информационных форматов. XML, рекомендация от World Wide Web Consortium (W3C). Язык похож на язык программирования современных веб–страниц, язык гипертекстовой разметки (HTML). Оба, и XML и HTML содержат разметку символов, используемых для описания содержимого страницы или файла. Однако, HTML описывает содержание веб–страницы (в основном текстовое и графическое отображение) только в плане того, как это должно быть отображено. Например, буква "p" помещенная в теги разметки начинает новый абзац . XML описывает содержание с точки зрения того, что содержится в данных. Например, слово " phonenum" помещенное в теги разметки может означать, что данные, которые находятся внутри это номера телефонов. Это означает, что файл XML может быть обработан вместе с программой как данные или же его можно хранить с аналогичными данными на другом компьютере или, как HTML файл.

XML является "расширяемым" языком, потому что, в отличие от HTML, символы разметки неограниченны. XML самом деле, проще и легче в использовании, чем другие языки (SGML).

Самый простой XML–документ состоит из двух частей: пролога и корневого элемента. Пролог содержит объявление XML, указывающее на то, что это XML–документ, и содержит номер версии XML. Пролог может также содержать необязательные компоненты:

- объявление типа документа;
- одну или несколько инструкций по обработке.

XML решает ряд проблем, которые не решает HTML.

Представление документов любого (не только текстового) типа, например, музыки, математических уравнений и т.д.

Сортировка, фильтрация и поиск информации.

Представление информации в структурированном (иерархическом) виде.

В зависимости от уровня соответствия стандартам документ может быть "верно сформированным" ("well–formed"), либо "валидным" ("valid"). Вот несколько основных правил создания верно сформированного документа:

– каждый элемент XML должен содержать начальный и конечный тэг (либо пустой тэг типа <TAG/>, который может нести информацию посредством своих атрибутов);

– любой вложенный элемент должен быть полностью определён внутри элемента, в состав которого он входит;

– документ должен иметь только один элемент верхнего уровня;

– имена элементов чувствительны к регистру.

Есть три основных способа сообщить браузеру, как отображать каждый из созданных вами XML–элементов:

– каскадная таблица стилей (Cascading Style Sheet – CSS) или расширяемая таблица в формате языка стилевых таблиц (Extensible Stylesheet Language – XSL);

– связывание данных. Этот метод требует создания HTML–страницы, связывания с ней XML–документа и установления взаимодействий HTML–элементов с элементами XML. В дальнейшем HTML–элементы автоматически отображают информацию из связанных с ними XML–элементов;

– написание сценария. Этот метод требует создания HTML–страницы, связывания с ней XML–документа и получение доступа к XML–элементам с помощью кода сценария JavaScript или VBScript.

Индивидуальный пользователь, компания или комитет по стандартам может определить необходимый набор элементов XML и структуру документа, которые будут применяться для особого класса документов. Подобный набор элементов и описание структуры документа называют XML–приложением или XML–словарём.

XML–приложение обычно определяется созданием описателя типа документа (DTD), который является допустимым компонентом XML–документа. DTD устанавливает и определяет имена элементов, которые могут быть использованы в документе, их порядок, в котором элементы могут появляться, и доступные к применению атрибуты элементов. DTD обычно включается в XML–документ и ограничивает круг элементов и структур, которые будут использоваться. Примечание: приложение XML Schema позволяет разрабатывать подробные схемы для ваших XML–документов с использованием стандартного синтаксиса XML и является альтернативой DTD.

XML–документ может содержать комментарии, начинающиеся с символов "<!—" и заканчивающиеся символами "—>". Комментарий может содержать любой текст, за исключением символов "—". Тексты комментариев доступны для написанного внутри HTML–страницы кода сценария.

XML–документ можно набрать в любом текстовом редакторе, сохранив документ как текстовый файл с расширением «.xml». В дальнейшем такой

документ будет открываться двойным щелчком в Internet Explorer. Вот пример простейшего XML–документа:

```
<?xml version="1.0"?>
<!-- Комментарий -->
<PRODUCTS>
      <PRODUCT>
     <TITLE> Продукт 1 </TITLE>
     <PRICE> 50 </PRICE>
     </PRODUCT>
     <PRODUCT>
          <TITLE> Продукт 2 </TITLE>
          <PRICE> 100 </PRICE>
     </PRODUCT>
</PRODUCTS>
```

Для возможности использования кириллицы в документах такого формата, нужно указывать требуемую кодировку в прологе (начале кода):

```
<?xml version="1.0" encoding="windows-1251"?>
```

По определению, XML документ является строкой, содержащей всевозможные различные символы. Почти каждый символ Unicod–а может быть задействован в XML–документе.

XML был собран рабочей группой из одиннадцати участников. Так же в проекте были заинтересованы и участвовали около 150 участников. Проектная группа, на протяжении долгого времени, вела дебаты и споры по поводу новой разработки. Все решения по этому поводу должны были быть приняты по общему согласию, но это никак не достигалось. Поэтому проектную часть возглавил Майкл Сперберг–Маккуин. Он собрал весь отчет по проекту 4 декабря 1997 года. Джеймс Кларк служил техническим руководителем рабочей группы, особенно внося пустой элемент" <пустой/>" синтаксис и имя "XML". Другие имена, которые были выдвинуты для рассмотрения, включали "MAGMU" (Минимальная Архитектура для Обобщенных Приложений Повышения), "SLIM" и "MGML" (Минимальный Обобщенный Язык Повышения).

Соредакторами спецификации первоначально были Тим Брей и Майкл Сперберг–Маккуин. Но на полпути разработки проекта, Майкл Брей принял предложение от компании Netscape, по которой он должен был консультировать их, тем самым вызывая недовольству у корпорации Microsoft. Брея попросили оставить должность главного редактора на некоторое время. Что привело к многочисленным спорам и разногласиям в рабочей группе, что, в конечном счете, привело к решению на назначение Джин Пэоли на роль третьего, главного редактора Microsoft.

Рабочая группа XML никогда не встречалась лицом к лицу; дизайн языка был создан, благодаря использованию комбинации электронной почты и

телеконференций. Основные проектные решения были приняты в период интенсивной работы между августом и ноябрем 1996 года, когда первый рабочий проект спецификации XML уже был издан. Дальнейшая проектная работа продолжалась до 1997, и 10 февраля 1998 года XML 1.0 стал рекомендацией W3C.

Есть две версии XML. В 1998 была принята первая версия языка (XML 1.0). С тех пор язык видоизменялся несколько раз. В конечном счете, это привело к его улучшению и выпуску следующей версии. 26 ноября 2008 была выпущена текущая версия. Были обсуждения по поводу выпуска еще одной версии, более продвинутой версии языка. Но пока разработчики остановились на текущей версии.

2.3.2 Использование XML–файлов для хранения данных

Данные в дата–центричных документах могут иметь свое происхождение из базы данных (в этом случае их нужно трансформировать в XML) или из внешней базы данных (в этом случае вам нужно сохранить их в базе данных). Примером первого типа может являться огромное количество различных данных, хранящихся в реляционных базах данных; примером второго типа можно считать научные данные, собираемые некой измерительной системой и конвертируемые в XML.

Таким образом, в зависимости от ваших нужд, вам может понадобиться программное обеспечение, которое переносит данные из XML–документа в базу данных, из базы данных – в XML–документ, или и то и другое. Это программное обеспечение может либо быть встроено в базу данных, в этом случае говорится, что такая база данных "оснащена средствами XML", или этот перенос может осуществляться с помощью специально написанного промежуточного программного обеспечения.

При переносе данных из XML–документа в базу данных, часто можно опускать информацию о документе, например, его имя или DTD. Кроме того, можно опускать информацию о физической структуре документа, например, определение сущностей и параметр usage, который задает способ хранения бинарных данных (Base64, как непарсируемая сущность или как–то еще), секции CDATA и информацию о кодировке. Иногда даже можно опустить информацию о логической структуре, в частности, процессуальные инструкции и порядок, в котором следуют значения атрибутов или соседние элементы.

Подобным образом, при переносе данных из базы данных в XML–документ, результирующий XML–документ, скорее всего, не будет содержать CDATA или сущностей (кроме изначально заданных сущностей lt, gt, amp, apos, и quot), а порядок, в котором будут следовать соседние элементы или атрибуты будет определяться порядком, в котором эти данные были выданы базой данных.

Хотя, сперва, это может показаться удивительным, все это вполне разумно. Например, в случае, когда XML используется для переноса торговых

заказов из одной базы данных – в другую. В этом случае не имеет значения, хранится в документе номер заказа до или после даты заказа, не имеет значения и как хранится имя заказчика: как секция типа CDATA, как внешняя сущность, как атрибут или прямо как секция PCDATA. Имеет значение только правильный перенос важных данных из одной базы в другую. Таким образом, программное обеспечение для переноса данных должно учитывать только иерархический порядок, в котором группируется информация в каждом торговом заказе и ничего больше.

Одним из последствий игнорирования информации относительно документа и его физической структуры является отсутствие обратимости – то есть, сохранение данных из документа в базу данных, а затем обратное восстановление документа из базы часто приводит к другому документу, даже в каноническом смысле слова. Допустимо ли это – зависит от ваших требований, и ответ на этот вопрос может повлиять на выбор базы данных и программного обеспечения для переноса данных.

При отображении на основе шаблона между документом и базой данных не существует какого–то изначально заданного соответствия. Вместо этого в шаблон каждый раз вставляются специально подобранные команды, которые обрабатываются программами переноса данных. Например, следующий шаблон имеет встроенные выражения SELECT в виде элементов <SelectStmt>:

```
<?xml version="1.0"?>
<FlightInfo>
<Introduction>Есть свободные места на
следующие рейсы:</Introduction>
<SelectStmt>SELECT Airline, FltNumber,
Depart, Arrive FROM Flights</SelectStmt>
<Conclusion>Мы надеемся, вы найдете
что–то подходящее</Conclusion>
</FlightInfo>
```

Отображения на основе таблиц применяются во многих программах переноса данных между XML–документами и реляционными базами данных. В них XML–документ представляется как таблица или множество таблиц. То есть, структура документа должна быть примерно такой, как в этом примере, при этом элемент <database> и дополнительный элемент <table> отсутствуют в одно–табличном случае:

```
<database>
<table>
<row>
<column1>...</column1>
<column2>...</column2>
...
</row>
<row>
...
```

```
</row>
...
</table>
<table>
...
</table>
...
</database>
```

В некоторых программных продуктах можно конкретизировать, сохраняются ли данные в колонке как дочерние элементы или как атрибуты, так же, как и имена, используемые для каждого элемента и атрибута. Кроме того, в продуктах на основе табличных отображений часто реализована возможность включения табличных и колоночных метаданных либо в начале документа, либо в виде атрибутов каждой табличного или колоночного элемента. Заметьте, что термин "таблица" обычно понимается весьма вольно. То есть, при переносе данных из базы данных в XML, "таблицей" может быть результирующее множество, а при переносе данных из XML в базу данных, "таблицей" может являться реальная таблица или другое модифицируемое отображение.

Отображения на основе таблиц полезны для сериализации реляционных данных, например, при переносе данных между двумя реляционными базами данных. К очевидному недостатку таких систем можно отнести то, что они не могут работать с XML–документами, которые не удовлетворяют приведенному выше формату.

Объектно–реляционное отображение применяется всеми реляционными базами, оснащенными средствами XML, и некоторыми программами переноса данных между XML–документами и реляционными базами данных. В этом отображении данные в XML–документе моделируются деревом объектов, специфичных для данного типа документов. В этой модели типы элементов с атрибутами, содержимое элементов или смешанное содержимое (сложные типы элементов) обычно моделируются классами. Типы элементов, чье содержимое является чистыми секциями PCDATA (простые типы элементов), атрибуты и секции PCDATA моделируются скалярными свойствами. Затем модель отображается в реляционную базу данных с помощью традиционных техник объектно–реляционного отображения или с помощью объектных представлений SQL 3. То есть, классы отображаются в таблицы, скалярные свойства – в колонки, а объектно–означенные свойства отображаются в пару первичный ключ / внешний ключ.

Термин "объектно–реляционное отображение" не совсем верен, поскольку дерево объектов может быть прямо отображено в объектно–ориентированную или иерархическую базу. Однако, мы его все равно будем использовать, поскольку огромное количество продуктов, которые используют этот вид отображений, действуют на основе реляционных баз данных, и термин "объектно–реляционное отображение" широко используется.

Инициализируются ли на самом деле объекты этой модели – зависит от продукта. Некоторые программные продукты позволяют вам порождать классы в модели, а затем использовать объекты этих классов в своих приложениях. С помощью таких продуктов данные переносятся между XML–документом, этими объектами и базой данных. В других продуктах объекты используются лишь как инструмент, который помогает вам визуализировать отображение, а перенос производится прямо между XML–документом и базой данных. И полностью от вашего приложения зависит, есть ли смысл в инициализации этих промежуточных объектов.

Привязка XML–документов к объектам обычно называется привязкой XML–данных (XML data binding) – после реализации компанией Sun технологии. В некоторых программных продуктах реализуется привязка XML–данных; большинство из них могут переносить также и данные между объектами и базой данных.

Способы осуществления объектно–реляционного отображения варьируются.

Во всех программных продуктах поддерживается отображение сложных типов элементов в классы, а простых типов элементов и атрибутов – в свойства.

Все продукты позволяют нам указывать корневой элемент, который не отображается в объектную модель или базу данных. Этот несущий элемент полезен в тех случаях, когда вам нужно в одном XML–документе хранить несколько объектов верхнего уровня. Например, если вы хотите хранить в одном XML–документе несколько торговых заказов, вам потребуется охватить их единым корневым элементом.

Корневой элемент эквивалентен элементу <database>, он используется, когда XML–документ, использующий табличное отображение содержит множество таблиц и присутствует только для того, чтобы выполнить требование XML (документ должен иметь единственный корневой элемент). В некоторых программных продуктах также можно обозначить элементы более низкого уровня как несущие.

Большая часть продуктов позволяет определить, отображаются ли свойства в атрибуты или дочерние элементы XML–документа. Некоторые продукты позволяют смешивать атрибуты и дочерние элементы, другие разрешают либо одно, либо другое.

Большая часть продуктов позволяет использовать различающиеся имена в XML–документе и базе данных, но в некоторых требуется использовать одинаковые.

Большая часть продуктов позволяет определить порядок, в котором следуют дочерние элементы в родительском элементе, хотя это реализуется так, что невозможно построить много моделей контента. К счастью, поддерживаемые модели контента годятся для большей части дата–центричных документов (Порядок следования дочерних элементов важен при валидации документа).

Некоторые продукты позволяют отображать комплексные типы элементов в скалярные свойства. Это полезно, когда тип элемента содержит смешанное содержимое, такое, как элемент <Description> в приведенном выше примере торгового заказа. Хотя элемент <Description> имеет дочерние элементы и текст в форме XHTML, гораздо полезнее рассматривать этот элемент как единое свойство, нежели разбивать его на части.

Некоторые продукты поддерживают отображение PCDATA в смешанное содержимое.

Также можно хранить данные XML–документов в истинной XML–базе данных (native XML database). Есть несколько причин для этого. Первая – полу–структурированность данных. То есть, они обладают регулярной структурой, но эта структура достаточно сильно варьирует и отображение ее в реляционную базу данных приводит к большому числу колонок с пустым значением (пустая трата места) или к большому количеству таблиц (что неэффективно). Хотя полу–структурированные данные можно хранить в объектно–ориентированной или иерархической базе данных, можно хранить их в истинной XML–базе данных в виде XML–документов.

Вторая причина хранить данные в истинной XML–базе данных – это скорость доступа. В зависимости от того, как физически истинная XML–база данных хранит данные, она может получать доступ к данным гораздо быстрее, чем реляционная база данных. Причина в том, что некоторые стратегии хранения, применяемые в истинных XML–базах, хранят целые документы физически вместе или используют физические (а не логические) указатели между частями документа. Это позволяет извлекать документы либо без объединений, либо с помощью физических объединений, и то и другое быстрее, чем логические объединения, которые применяются в реляционных базах данных.

Рассмотрим, например, приведенный выше документ, содержащий торговые заказы. В реляционной базе данных его можно было бы хранить в четырех таблицах: SalesOrders, Items, Customers, и Parts, и извлечение документа могло бы потребовать объединения по всем этим таблицам. В истинной же XML–базе данных весь документ мог бы храниться в одном месте на диске, так что получение доступа к документу или к его фрагменту потребовало бы единственного обращения к диску и единственного чтения данных. Реляционная база потребовала бы четыре поиска индексов и по меньшей мере четырех чтений с диска.

Очевидное соображение в этом случае: увеличение скорости будет ощущаться только в тех случаях, когда данные на выходе должны быть организованы именно так, как они хранятся на диске. Если вы хотите получать данные в различных представлениях, например, в виде списка заказчиков и относящихся к ним заказов, производительность была бы, по всей видимости, меньше, чем в реляционной базе данных. Таким образом, хранение данных в истинной XML–базе из–за стремления к большей производительности

оправдано лишь в тех случаях, когда в вашем приложении доминирует какое–то одно представление данных.

Третья причина хранить данные в истинной XML–базе данных – если вам нужны некие XML–специфичные возможности, например, выполнение XML–запросов. С учетом того, что немногие дата–центричные приложения нуждаются сегодня в этом и многие реляционные базы данных могут выполнять XML–запросы, эта причина становится менее существенной.

Одна из проблем хранения данных в истинной XML–базе заключается в том, что большинство истинных XML–баз данных могут возвращать данные только в виде XML. (слабо поддерживается привязка элементов или атрибутов к переменным приложения.) Если вашему приложению понадобятся данные в другом формате (часто так и происходит), приложению придется сначала отпарсить XML и только после этого оно сможет использовать эти данные. Это очевидный недостаток для локальных приложений, которые вместо реляционой базы данных используют истинную XML–базу, поскольку требуется дополнительные ресурсы, в которых нет нужды, например, в ODBC–приложениях. Но это не проблема в распределенных приложениях, которые используют XML для транспортировки данных, поскольку дополнительные ресурсы понадобятся вне зависимости от того, какая используется база данных.

XML не поддерживает типизацию данных в любом значимом смысле этого слова. Кроме непарсируемых сущностей, все данные в XML–документе являются текстом, даже если он представляет другие типы данных, например, даты или целые числа. В общем случае программы переноса конвертируют данные из текста (в XML–документе) в другие типы (в базе данных) и наоборот. Однако, число текстовых форматов, в которых распознаются данные того или иного типа, ограничено, например, теми, которые заданы в JDBC–драйвере. Представление дат – наиболее проблемное место, поскольку диапазон возможных форматов очень велик. Числа, которые также имеют несколько национальных форматов, тоже могут вызвать проблемы.

Более серьезная проблема состоит в том, что "модель" данных, используемая в XML–документе часто отличается (и обычно более сложна), чем большинство эффективных моделей хранения данных в базе данных. Например, рассмотрим следующий фрагмент XML–кода:

```
<Customer>
      <Name>ABC Industries</Name>
      <Address>
         <Street>123 Main St.</Street>
         <City>Fooville</City>
         <State>CA</State>
         <Country>USA</Country>
         <PostCode>95041</PostCode>
      </Address>
</Customer>
```

Процедура генерации реляционной схемы из определения DTD могла бы в этом случае создать две таблицы: одну для заказчиков и одну для адресов. Однако, в большинстве случаев было бы разумнее хранить адрес в таблице заказчиков, а не в отдельной таблице.

Элемент <Address> – это хороший пример корневого несущего элемента. Несущий элемент обычно нужен по двум причинам. Во–первых, он предоставляет дополнительную структуру, которая облегчает понимание документа. Во–вторых, он часто используется в качестве некой формы типизации данных, например, элемент <Address> можно передавать процедуре, которая конвертирует все адреса в объект Address, вне зависимости от того, где находится этот элемент.

Хотя корневые элементы полезны в XML, они обычно становятся причиной проблем в форме излишней структуры при отображении в базу данных. По этой причине перед генерацией реляционной схемы их обычно удаляют из определения DTD. Поскольку вряд ли хорошим решением является радикальное изменение DTD, это приводит к расхождению между реальными документами и документами, подходящими для обработки программами переноса, поскольку корневой элемент не вовлекается в отображение. Эту проблему можно решить, преобразуя документы на лету, например, с помощью XSLT: корневые элементы удаляются перед переносом данных в базу и снова вставляются после переноса данных из базы [7].

2.4 Язык программирования C#

C# относится к семье языков с C–подобным синтаксисом, из них его синтаксис наиболее близок к C++ и Java. Язык имеет статическую типизацию, поддерживает полиморфизм, перегрузку операторов (в том числе операторов явного и неявного приведения типа), делегаты, атрибуты, события, свойства, обобщённые типы и методы, итераторы, анонимные функции с поддержкой замыканий, LINQ, исключения, комментарии в формате XML.

Переняв многое от своих предшественников – языков C++, Java, Delphi, Модула и Smalltalk – C#, опираясь на практику их использования, исключает некоторые модели, зарекомендовавшие себя как проблематичные при разработке программных систем: так, C# не поддерживает множественное наследование классов (в отличие от C++).

2.4.1 История языка

C# (C Sharp) – объектно–ориентированный язык программирования. Который был представлен в 1998–2001 годах группой инженеров, возглавляемых Андерсеном Хейлсбергом, разработчиком компании Microsoft. Этот язык являлся языком разработки приложений для платформы Microsoft .NET Framework и впоследствии был стандартизирован как ECMA–334 и ISO/IEC 23270 [10].

Разработанный язык многое позаимствовал от своих конкурентов и в частности у Java. C# в отличие от C++ не поддерживает множественное наследование классов (между тем допускается множественное наследование интерфейсов).

Первая версия C# 1.0 была представлена летом 2000 года, а в феврале 2002 года с Microsoft Visual Studio вышла окончательная версия языка. Версия 1.0 вобрала в себя лучшие стороны предшествующих ей популярных языков программирования C, Java и C++. Это позволило программистам легко осуществить переход на C#, основываясь на знаниях любого из перечисленных раннее, языков.

Основной синтаксис и операторы перешли в C# из C. Объектная модель, реализованная в C++, в C# гораздо усовершенствована. Если, по отношению к C и C++, первая версия C# является прямым потомком, то для Java он будет, скорее всего, кровным братом. Конечно, Java тоже происходит от семейства C, и имеет так же много общего с C#, как и много отличий.

Отличием от предшественников является то, что в C# включено поддержание написания компонентов программ. Подобной поддержкой, например, выступают такие средства, как события, методы и свойства. Но главным ключом в компонентной ориентированности C# является, конечно же, обеспечение безопасной работы в среде мультиязыкового программирования.

В 2005 году вышел окончательный релиз версии 2.0 C#. Она укрепила позиции этого языка программирования и окончательно убедила в его эффективности и дальнейшем развитии. Добавление новых возможностей, таких как анонимные методы, обобщения, частичные и параметризованные типы значительно расширили возможности применения C#. Во второй версии была добавлена поддержка 64–х разрядных вычислений, что открыло возможности в увеличении адресного пространства. Также было реализовано создание триггеров, хранимых процедур и типов данных на .NET языках.

Следующая версия C# 3.0, окончательно вышедшая уже в 2008 году, полностью изменила мир программирования. Множество важных новшеств позволили C# стать флагманом в эволюционном развитии языков программирования. К ним относятся такие нововведения как: язык интегрированных запросов (LINQ), позволивший осуществлять запросы изXML из SQL; не явно типизированные переменные и методы расширения; лямбда–выражения, которые назначают реализацию кода делегатам с помощью нового, более простого синтаксиса. Лямбда –выражения вместе с языком интегрированных запросов выразили и акцентировали C# в качестве ведущего языка программирования. Анонимные типы переменных позволили избежать громоздкости и откровенного неудобства при описании переменных, дав возможность объявлять новый тип непосредственно при ее создании. Интересной новинкой в C# 3.0 стали так называемые «ленивые вычисления», которые производят необходимые вычисления только при запросе нужных соответствующих данных.

Кроме того, в C# было добавлено несколько основных опций, для приспособления программирования функционального стиля, достигающего хорошего уровня при взаймодействии с LINQ, выпущенным с C# 3.0 и его структурой поддержки лямбда–выражений, дополнительных методов и анонимных классов. Эти опции позволяют программистам C# использовать функциональные программные методы, такие как закрытия, когда это выгодно для их применения. Расширения LINQ и функциональный импорт помогают разработчикам уменьшить сумму кодекса «газетного материала», который включен в общие задачи как сомнение базы данных, парсинг xml файла или осуществление поиска структуры данных, перемещая акцент на фактическую логику программы, чтобы помочь улучшить удобочитаемость и ремонтопригодность.

В 2010 году была выпущена текущая версия C# 4.0. Ее главным дополнением к предыдущим версиям стали именованные и необязательные аргументы. Именованные дают возможность привязки аргумента и параметра по имени, а неименованные позволяют указать аргумент, который используется по умолчанию, для каждого параметра. Не менее важное новшество – тип «dynamic». Он объявляет объекты, проверка на соответствие типов которых происходит не при компиляции, а непосредственно во время выполнения программы.

Другие важные средства C#, представленные в версии 4.0 значительно упрощают процесс создания программ. Правда, реализованы они не языком программирования, а средой Framework 4.0. Это библиотека распараллеливания задач (TPL) и параллельный вариант языка интегрированных запросов (PLNQ). Их поддержка позволяет осуществить параллельность выполнения кода в компьютерах с многоядерными процессорами или несколькими одно ядерными [2].

2.4.2 Объектно–ориентированное программирование

Объектно–ориентированное программирование (ООП) – парадигма программирования, которая представляет собой смесь понятий «объектов», у которых есть поле данных (признаки, которые описывают объект), и связанные с ними процедуры, известные как методы. Объекты, которые обычно являются случаями классов, используются, для взаимодействия их друг с другом, для проектирования компьютерных программ. C ++, Objective–C, Smalltalk, Java, C#, Perl, Phyton, Ruby и PHP являются примерами языков объектно–ориентированного программирования.

Объектно–ориентированное программирование – это подход к модульному проектированию, повторно использующему систему программного обеспечения. Хотя при обсуждении объектно–ориентированной технологии могли быть найдены несоответствия в деталях одного языка против другого. Реальным ключом к объектно–ориентированному подходу является то, что это – подход моделирования сначала. Объектно–ориентированный подход считают

логическим расширением хороших методов дизайна, которые возвращаются к самому началу программирования вместо того, чтобы быть революционными, как требуется некоторыми сторонниками. Объектное–ориентирование – это простое логическое расширение более старых методов таких, как структурированное программирование и абстрактные типы данных. Объект – это абстрактный тип данных с добавлением полиморфизма и наследованием.

Вместо того, чтобы разделять программу отдельно по коду и по данным, объектно–ориентированное программирование объединяет два концепта использования понятия «объекта». У объекта есть содержание (данные) и поведение (код). Объекты соответствуют вещам, находящимся в реальном мире. Так же, например, у графической программы будут объекты, такие как круг, квадрат, меню. У системы онлайн–покупок будут объекты, такие как магазинная тележка, клиент, продукт. Система покупок поддержит поведения те, которые размещают заказ, осуществляют платеж и предлагают скидку. Объекты разработаны как иерархии классов. Так, например, у системы покупок могли бы иметься классы высшего уровня, такие как электроника, кухонные продукты и книги. Могут быть дальнейшие разделения, например, под электронной продукцией понимаются: CD–плееры, DVD–плееры, и т.д. Эти классы и подклассы соответствуют наборам и подмножествам как в математической логике.

Цели объектно–ориентированного программирования:

- увеличение понимания;
- легкое обслуживание;
- быстрое и легкое развитие.

Полное понимание системы постоянно расширяется, потому что семантический промежуток – расстояние между языком, на котором говорят разработчики и, на котором говорят пользователи – уменьшено. Вместо того, чтобы говорить о таблицах базы данных и программировании подпрограмм, разработчики задумывались о вещах, с которыми пользователь знаком: объекты от их прикладной области.

Объектно–ориентированное программирование ослабляет обслуживание при помощи информационного сокрытия и инкапсуляции. Один из наиболее распространенных источников ошибок в программах – когда одна часть системы случайно вмешивается в другую часть. Например, в очень самые ранние дни программирования, разработчикам было свойственно использовать, "идти в" заявления, чтобы подскочить к произвольным местоположениям в пределах только нескольких установленного порядка и функций. Критики назвали этот "код спагетти", потому что он дезорганизован. Структурированное программирование обращается к этому, поощряя использование процедур и подпрограмм. Соответствующие секции использования от ответственности за отдельные блоки, чтобы осуществить отдельную функциональность. Так, например, можно было бы знать, что функция квадратного корня была отдельной от ракетной функции запуска и изменения, нельзя было затронуть другой.

Объектно–ориентированное программирование оставляет это на следующий шаг. Это по существу сливает абстрактные типы данных со структурированным программированием и делит системы на модульные объекты, которые владеют их собственными данными и ответственны за их собственное поведение. Эта особенность известна как инкапсуляция. С инкапсуляцией, не только может, "квадратный корень" и "запускает ракеты" функции не, вмешиваются друг в друга, но также и данные для этих двух разделены так, чтобы изменения одного объекта не могли затронуть другой. Обратите внимание на то, что все это полагается на различные языки, используемые соответственно, который, конечно, никогда не является бесспорным.

В дополнение к обеспечению легкого обслуживания, инкапсуляция и информационное сокрытие обеспечивают также непринужденность развития. Программное обеспечение обозначают как модульные компоненты, которые поддерживают наследование, облегчает и снова использовать существующие компоненты и расширять компоненты по мере необходимости, определяя новые подклассы со специализированными поведениями.

Объектно–ориентированное программирование позволяет программисту помещать данные там, где это не доступно для остальной части системы (кода). Вместо этого к данным получают доступ, вызывая специально письменные функции, названные методами, которые связаны данными. Они действуют как посредники для восстановления или изменения данных, которыми они управляют. Программную конструкцию, которая объединяет данные с рядом методов для доступа и управления теми данными, называют объектом. Практика использования подпрограмм, для исследования или изменения определенных видов данных также использовалась не в ООП.

Объектно–ориентированное программирование обычно содержит различные типы объектов, каждый соответствующий реальному объекту или понятию, такие как банковский счет, хоккеист или бульдозер. Программа могла бы содержать многократные копии каждого типа объекта, один для каждого из реальных объектов соглашения о программе. Например, мог быть один объект банковского счета для каждого реального счета в настоящем банке. Каждая копия объекта банковского счета была бы подобна в методах, которые это предлагает для управления или чтения его данных, но данные в каждом объекте отличались бы, отражая различную историю каждого счета.

Объекты могут считаться заключенными в капсулу их данных в пределах ряда функций, разработанных, для гарантий, что данные используются только по назначению, и помогают им в использовании. Методы объекта, как правило, включают проверки, и охраняет определенный для типов данных, которые содержит объект. Объект может также предложить простые к использованию, стандартизированные методы для выполнения особых операций на его данных, скрывая специфические особенности того, как те задачи выполнены. Таким образом изменения могут быть сделаны к внутренней структуре или методам объекта, не требуя что остальная часть программы быть измененными. Этот

подход может также использоваться, чтобы предложить стандартизированные методы через различные типы объектов. Как пример, несколько различных типов объектов могли бы предложить методы печати. Каждый тип объекта мог бы осуществить тот метод печати по–другому, отражая различные виды данных, которые каждый содержит, но все различные методы печати можно было бы назвать тем же самым стандартизированным способом откуда–либо в программе. Эти особенности становятся особенно полезными, когда больше чем один программист вносит кодекс в проект или когда цель состоит в том, чтобы снова использовать кодекс между проектами.

Обзор Деборы Дж. Армстронг почти 40 лет вычислительной литературы идентифицировал много фундаментальных понятий, найденных в значительном большинстве определений ООП.

Не все эти понятия появляются на всех языках объектно–ориентированного программирования. Например, объектно–ориентированное программирование, которое использует классы, иногда называют основанным на классе программированием, в то время как основанное на прототипе программирование как правило не использует классы. В результате существенно отличающаяся все же аналогичная терминология используется, чтобы определить понятие объекта и случая.

Бенджамин К. Пирс и некоторые другие исследователи рассматривают любую попытку дистиллировать ООП к минимальному набору особенностей как бесполезное. Он, тем не менее, идентифицирует фундаментальные особенности, которые поддерживают программный стиль ООП на большинстве объектно–ориентированных языков:

– динамическая отправка – когда метод призван на объект, сам объект, определяет, какой кодекс выполнен, ища метод во время, которым управляют, в столе, связанном с объектом. Эта особенность отличает объект от абстрактного типа данных (или модуль), у которого есть фиксированное (статическое) внедрение операций для всех случаев. Это – программная методология, которая дает модульное составляющее развитие в то же время, будучи очень эффективной;

– инкапсуляция (или мульти–методы, когда классы разделены);

– подтип полиморфизма;

– наследование объекта (или делегация);

– открытая рекурсия – специальная переменная (синтаксически это может быть ключевое слово), обычно называемый этим или сам, который позволяет телу метода призывать другое тело метода того же самого объекта. Эта переменная последняя направляющаяся; это позволяет методу, определенному в одном классе призывать другой метод, который определен позже в некотором подклассе этого.

Точно так же в его книге 2003 года, Понятиях на языках программирования, Джон К. Митчелл идентифицирует четыре главных особенности: динамическая отправка, абстракция, подпечатает полиморфизм и

наследование. Майкл Ли Скотт в Прагматике Языка программирования рассматривает только инкапсуляцию, наследование и динамическую отправку.

Дополнительные понятия, используемые в объектно–ориентированном программировании, включают:

- классы объектов;
- случаи классов;
- методы, которые действуют на приложенные объекты;
- прохождение сообщения;
- абстракция.

В последние годы объектно–ориентированное программирование стало особенно популярным для динамических языков программирования. Python, Ruby и Groovy, являются динамическими языками, основанными на принципах ООП, в то время как Perl и PHP добавили в свои языки объектно–ориентированные опции начиная только с Perl 5 и PHP 4, а ColdFusion начиная с 6 версии.

2.5 ASP.net и PHP

Как правило, для создания крупного сайта с многочисленными разделами и поддоменами, необходимо использовать не только HTML, CSS и JavaScript.

Для хорошего функционирования сайта нужен как минимум ответственный подход к написанию бэк–кода или же хорошая система управления. Код, который доступен только на серверной стороне, должен описывать правильное функционирование страниц между собой, правильной подачи, загрузки и добавления информации на сайте.

Программирование сайтов, точнее, их программирование со стороны северов, построено на различных технологиях, таких как php, ASP.net и др. Так же как и технологий разработки сайтов, языков для ее программирования тоже очень много: PHP, C#, Java, C++, VBscript, Perl, ASP, XML, ColdFusion, JavaScript и др. Выбор той или иной технологии/языка должен исходить от задачи построения данного сайта.

Сегодня существует много различных технологий, позволяющих создавать веб–приложения разной степени сложности. К ним относятся ASP, ASP.NET, PHP, JSP и многие другие. Наиболее популярной и распространённой технологией на сегодняшний день является PHP, во многом благодаря своей бесплатности. Однако с появлением технологии ASP.NET у PHP появился серьёзный конкурент. В данной дипломной работе некоторые фрагменты текста отсутствуют или замещены на текст не соответствующий теме этого диплома.

2.5.1 Технология ASP.net

ASP.NET(Active Server Pages) – технология создания веб–приложений и веб–сервисов от компании Майкрософт. Она является составной частью

платформы Microsoft .NET и развитием более старой технологии Microsoft ASP. На данный момент последней версией этой технологии является ASP.NET 4.5.1.

ASP.NET внешне во многом сохраняет схожесть с более старой технологией ASP, что позволяет разработчикам относительно легко перейти на ASP.NET. В то же время внутреннее устройство ASP.NET существенно отличается от ASP, поскольку она основана на платформе .NET и, следовательно, использует все новые возможности, предоставляемые этой платформой.

После выпуска сервера Internet Information Services 4.0 в 1997 году, компания Microsoft начала исследовать возможность новой модели веб–приложения, которая удовлетворит жалобы на ASP, особенно связанные с отделением оформления от содержания, и которая позволит писать «чистый» код. Работа по разработке такой модели была поручена Марку Андерсу, менеджеру команды IIS, и Скотту Гатри, поступившему на работу в Microsoft в 1997. Андерс и Гатри разработали первоначальный проект в течение двух месяцев, и Гатри написал код первоначального прототипа во время рождественских каникул 1997 года.

Первоначальный проект назывался «XSP». Гатри объяснил в интервью 2007 года что, «всегда спрашивают, что означает буква X. В то время она ничего не значила. XML начинается с нее; XSLT начинается с нее. Все клевое начинается с X, поэтому мы его так и назвали». Прототип XSP был написан на Java, так как на тот момент у Microsoft не было Java–подобной технологии. В то время уже предполагалось (небезосновательно, как выяснилось в дальнейшем), что лицензирование Java для Microsoft не будет продлено в 2003 году (в 2003 истекал срок выданной Sun Microsystems лицензии). В 1999 было решено построить новую платформу на основе Common Language Runtime (CLR), так как в нем, как и в Java наличествовало программирование по принципам ООП, сборка мусора и другие возможности. Гатри описал это решение как «огромный риск», так как успех новой разработки был связан с успехом CLR, которая, как и XSP, находилась на ранней стадии разработки [3].

Веб–технология стороны сервера Microsoft. ASP.NET проявляет подход объектно–ориентированного программирования при выполнении веб–страницы. Каждый элемент на ASP.NET странице рассматривается как объект на сервере.

ASP.NET используется, для создания веб–страниц и веб–сервисов и является неотъемлемой частью .NET Framework Microsoft.

ASP.NET – часть Microsoft.NET. Таким образом, у него имеются все преимущества объектно–ориентированных программных возможностей. Веб–страницы, созданные с ASP.NET, являются в основном объектами, у которых имеются свои сценарии (события в пределах страницы), команды (методы в кодексе), и разделы (иллюстрировавшие примерами объекты).

WebForms – наиболее распространенный способ работы с ASP.NET. С WebForms любой пункт на странице программируем, и имеет события.

Главное, Bochicchio, Мострада и Де Санктис говорят, что, для использования модели WebForms, все что нужно знать пользователю – это его шаги как сделать, куда поместить. Например, Button объект на странице, который имеет событие на случай Click–а. Разработчику требуется только поместить объект на страницу и настроить его действия.

15 августа 2012 была выпущена структура.NET 4.5. Ряд новых или улучшенных компонентов было добавлено в эту версию. Некоторые из них (связанные с этим проектом):

– поддержка нового HTML5, форматирование типов;

– поддержка новых шаблонов связки на WebForms;

– поддержка невидимого JavaScript в подлинниках проверки стороны клиента;

– интегрированные режимы кодирования из библиотеки AntiXSS (ранее внешняя библиотека);

– поддержка протокола WebSockets;

– поддержка чтения и письма запросов HTTP и ответов асинхронно;

– поддержка асинхронных модулей и укладчиков;

– поддержка отступления распределительной сети содержания (CDN) в контроле ScriptManager.

2.5.2 PHP

PHP – это скриптовый язык программирования, который создан для генерации HTML–страниц на веб–сервере и его взаймодествия с базами данных.

На сегодняшний день, этот язык имеет поддержку почти во всех доступных хостингах. Главным преимуществом этой технологии является то, что она распространяется бесплатно. Благодаря своей простоте, скорости выполнения, богатой функциональности, распространению исходных кодов на основе лицензии PHP, этот язык является, чуть ли не самым популярным в области технологий создания сайтов. Отличается наличием ядра и подключаемых модулей, «расширений»: для работы с базами данных, сокетами, динамической графикой, криптографическими библиотеками, документами формата PDF и т.п. Есть возможность разработать, а также подключить дополнительное расширение.

Возможности PHP очень обширны. Главным образом, PHP применяется при написании скриптов, работающих на стороне сервера; таким образом, PHP способен выполнять всё то, что выполняет любая другая программа CGI (например, обрабатывать данных форм, генерировать динамические страницы, отсылать и принимать cookies). Но PHP дает возможность выполнять также множество других задач.

Существуют три основных области, где используется PHP:

1) создание скриптов для выполнения на стороне сервера;

2) создание скриптов для выполнения в командной строке;

3) создание приложений GUI, выполняющихся на стороне клиента.

Помимо этого PHP:

– доступен для большинства операционных систем, включая Linux, многие модификации Unix (такие, как HP–UX, Solaris и OpenBSD), Microsoft Windows, Mac OS X, RISC OS, и многих других;

– включает поддержку большей части веб–серверов (для большинства серверов PHP поставляется в качестве модуля, для других, поддерживающих стандарт CGI, PHP может функционировать в качестве процессора CGI);

– поддерживает обширный круг баз данных;

– поддерживает DBX для работы на абстрактном уровне (таким образом можно работать с любой базой данных, использующих DBX);

– ODBC (т.е. вы можете работать с любой базой данных, поддерживающей этот стандарт);

– поддерживает стандарт обмена сложными структурами данных WDDX;

– поддерживает объекты Java, дает возможность использовать их в качестве объектов PHP;

– дает возможность формировать изображения, файлы PDF, ролики Flash, создаваемые "на лету";

– способен выдавать любые текстовые данные (XHTML, другие XML–файлы);

– автоматически генерировать и сохранять в файловой системе сервера;

– включает средства обработки текстовой информации, начиная с регулярных выражений Perl или POSIX Extended и заканчивая парсером документов XML;

– поддерживает многие другие расширения (функции поисковой машины mnoGoSearch, функции IRC Gateway, функции для работы со сжатыми файлами (gzip, bz2), функции календарных вычислений, функции перевода и др.).

2.5.3 Сравнение ASP.net и PHP

Сравнение PHP (распространимого на безвозмездной основе) с (платным) ASP.net включает в себя следующие пункты сравнения: работа, стоимость, масштабируемость, поддержка и уровнень сложности.

Если поискать в интернете запросы по сравнению этих технологий, то можно наткнуться на многочисленные статьи и форумы, в которых сторонники обеих технологий, бурно спорят между собой, доказывая, какая технология лучше.

Кроме того, если учитывать дату написания статей, то можно заметить, что большинство из них устарело. Поэтому не следует доверять и основывать предпочтетиня на их примере, так как та информация уже не действительна.

Ведь нужно учитывать все улучшения этих двух технологий, произведенных за последнее время.

Можно точно сказать, что на сегодняшний день, обе технологий имеют широкое распространение и одинаково используются для разработки как крупных и известных веб–приложений и веб–сайтов, так и для более мелких. Не может быть никаких сомнений в способности работы обеих технологий с крупномасштабными веб–приложениями, в одинаковой степени.

В данном сравнении, будет проведено некое сопоставление работоспособности этих технологий, по их результатам, скорости выполнения и большинстве кодов.

Что касается масштабируемости – будут рассмотрены, такие факты, как правильное программирование и выявление ошибок.

Если рассматривать стоимость данных продуктов, то учитывая, бесплатное распространение PHP и платную основу распространения ASP.net, то будет разумным, включение в цену стоимости всех затрат, связанных с реализацией, поддержкой и использованием различных ресурсов.

При сравнении времени разработки, определенно, PHP выполняеся быстрее, когда как при работе с ASP.net, требеутся вдвое больше кода. Но это только на начальном этапе, поскольку в ASP.net, после описания нескольких кодов веб–страниц, остальные генерируются автоматически, или же можно пользоваться мастером страниц.

Рассмотрим все пункты сравнения по отдельности.

Масштабируемость и непринужденность обслуживания

Масштабируемость и простота сервиса обслуживания не имеет никакого отношения к самому выбору, будь это PHP или платформа ASP.net.

Масштабируемость веб–приложения и простота обслуживания, прежде всего, зависит от:

- опыта программиста;
- использование лучших способов программирования;
- использование твердой программной структуры;
- от программных рекомендации и стандартов.

Компиляция и скорость

Есть и другие факторы, которые нужно рассматривать, при сравнении скорости генерации (работы) технологий, поскольку оно зависит так же от языка программирования. Но нужно иметь в виду что скорость генерации и исполнения не имеет никакого значения и эффекта на скорость работы большинства веб–сайтов сегодня.

Однако, если язык программирования должен выполнить огромные задачи, подобные таким, которые выполняют такие, крупные веб–приложения, как Google или Yahoo. Google и Yahoo используют несколько языков программирования (главным образом бесплатные источники), каждый из

которых специально отобран, для обращения к различным задачам, так что язык программирования должен быть лучшим во время выполнения.

PHP, сервер MySQL, сервер PostgreSQL, сервер апачи и Linux, который идет как бесплатная ОС, они все бесплатны. Кроме того, нет никакой дополнительной стоимости лицензирования для того, чтобы иметь другой горячий резервный сервер как резервную копию, или должной быть управлять многократными серверами для балансировки нагрузки или кластеризации серверов.

LAMP (Linux, апач, MySQL и PHP) также намного более популярна среди хостинговых компаний и его результатов популярности в более низкой ежемесячной принимающей стоимости для оказания гостеприимства ЛАМПЫ по сравнению с оказанием гостеприимства Windows.

ASP.net и IIS имеют свободное распространение, если покупается Windows OS. Есть существенная стоимость лицензирования для Microsoft Windows Server, Microsoft SQL Server и их модернизации. Например, Microsoft Server 2008 R2 Standard – 64 первоначальная стоимость эксплуатации составляло приблизительно 1029$, а Microsoft SQL Server 2008 Standard Edition For Small Business стоил приблизительно 1038$.

Вышеупомянутые затраты лицензирования для Microsoft могут существенно увеличиться, если место становится популярным и есть потребность управлять местом на многократных серверах или требует особенностей сервера, таких как балансировка нагрузки, кластеризация серверов или горячий резерв. Но поддержка сайта на PHP, его хостинг и прочее, в целом составляет большую цифру затрат, нежели ASP.net.

Поддержка и ресурсы

Так как LAMP – открытый источник, есть огромное количество преданных и дружелюбных разработчиков во всем мире, которые непрерывно делают улучшения и обновления, и оказывают поддержку для платформы. Кроме того, есть много ресурсов поддержки и разработок, доступных для Платформ LAMP и PHP.

ASP.net полагается на разработчиков в Microsoft для того, чтобы сделать улучшения и обновления. Существует меньшее количество участников поддержки, доступных, для решения проблемы с ASP.net.

Кроме того, PHP интерпретируется в сервере, поэтому изменяя функциональность, никакие дополнительные шаги не требуются, чтобы видеть изменения. С другой стороны, ASP.net должен быть собран каждый раз, когда кодекс изменен. Снова, процесс развития отнимает много времени, если используется ASP.net.

Редакторы и инструменты

PHP & MySQL – имеют многочисленные, независимые редакторы.

Большинство программистов ASP.net полагается на Microsoft Visual Studio, чтобы помочь им разработать.NET приложения.

Это – различный стиль программирования – PHP и общедоступные разработчики склонны использовать редакторы текста в качестве редакторов.

Существуют очень продвинутые и независимые редакторы, и программисты, которые изучают и используют тех редакторов для самых полных возможностей, могут выполнить очень сложное программирование быстрым, эффективным и независимым способом. Те программисты имеют больше контроля и гибкости, и когда дело доходит до потребности использования и интеграции других существенных платформ, таких как JavaScript, Аякс, JQuery, и т.д., у разработчиков PHP есть лучшее преимущество из–за их знакомства с общедоступной окружающей средой и кодированием руки.

Независимая платформа

PHP – независимая платформа и может работать на любой платформе–сервере – Linux, Unix, Mac OS X, Windows.

ASP.net построен так, чтобы работать только на платформе Windows.

В следующей таблице перечислены главные, популярные места и платформу и языки, которые они используют.

Популярность

Платформа LAMP намного более популярна, чем платформа Windows. Основанный на обзоре веб–сервера июля 2010 Неткрэфта 205,714,253 мест, 112,945,968 (54.90%) приняты на Apache, и 53,217,620 (25.87%) приняты на Windows; остальные приняты на других платформах. Но с каждым годом процент сайтов на серверах Windows и на технологии ASP.net увеличиваются.

Сторонников языка программирования PHP гораздо больше, чем ASP.NET. «PHP–шники» взахлеб доказывают преимущества их технологии: скорость, производительность, абсолютная бесплатность (платформа, хостинг), возможность создать любой проект – от мала до велика.

Не стоит долго ломать голову, почему программирование на PHP получило столь широкое распространение. А большая часть армии программистов представлена, прежде всего, «пхп–шниками». Чтобы научиться писать на этом языке, достаточно прочесть "один–единственный учебник" и немного попрактиковаться. В то время, как с ASP.NET все намного труднее. Да, это сложная технология, но овладев данным «набором концепций», вы без проблем, не тратя много времени и усилий сможете добавлять нужные строчки, таблицы, оперативно проводить отладку и т.п. Усидчивость и желание понять окупятся сторицей.

Для начала разберемся с понятиями. PHP (Personal Home Page Tools) – открытая и бесплатная технология. PHP переводится как «Инструменты для создания персональных веб–страниц». Стоит знать, что пхп – это скриптовый язык (скрипт – это сценарий, последовательность операций), предназначенный для динамического вывода HTML (язык разметки документов, с его помощью создано большинство веб–страниц). А это для владельца сайта будет означать,

что создавать на PHP крупные проекты – достаточно дорогостоящий и трудозатратный процесс. В итоге, получается один плюс (бесплатность) и один минус (сложность, с которой можно столкнуться при создании крупного проекта, особенно, если программист всего один).

ASP.NET – технология, предназначенная для создания веб–сайтов, веб–сервисов и приложений. Разработанная корпорацией Microsoft. Знакомить с этим мировым лидером в области программного обеспечения сейчас уже не приходится никого. Что это означает для будущего владельца сайта? Надежность, безопасность, устойчивость к атакам. В ASP.NET встроена защита от хакерских действий. Это большой плюс, особенно для крупных компаний и проектов.

Но это еще не все параметры, по которым можно сравнивать сайты на PHP и сайты на ASP.NET. Существует много споров относительно скорости работы. Большинство специалистов приходит к выводу, что PHP работает значительно быстрее. Обсуждение можно продолжить. Если это MS SQL, то по быстродействию ASP.NET не будет уступать связке PHP + MySQL.

В качестве минуса создания сайта на ASP.NET некоторые отмечают дорогой хостинг, в то время, как для PHP можно найти и бесплатный. Но если хорошо искать, то в сети можно обнаружить доступный, экономичный шаред хостинг (Shared Hosting), от слова share – доля, часть. Его плюс – провайдер не будет отрубать вас за превышение нагрузки, как, например, это часто бывает на PHP–хостинге. Бесплатный сыр только в мышеловке.

Фактор надежности, безопасности и стабильной работы сайта, построенного на ASP.NET, важен для стабильной работы всей компании.

2.6 Преимущества выбранной технологии

ASP.NET стал большим и коммерчески выгодным проектом, но, как уже говорилось, остальной мир веб–разработки также развивался. Пока Microsoft сдувал пыль с WebForms, ее основные конструкции начали выглядеть весьма устаревшими.

В октябре 2007 года на первой конференции по ASP.NET в Остине, штате Техас, вице–президент Microsoft Скотт Гатри объявил и продемонстрировал новую платформу по разработке MVC, построенную на базовой платформе ASP.NET, явно задуманную как прямой ответ на эволюцию технологий, таких как Rails и как реакцию на критику WebForms. В следующих разделах описывается, как эта новая платформа преодолела ограничения WebForms и снова возвеличила ASP.NET.

Архитектура MVC

Важно различать архитектурный паттерн MVC и ASP.NET MVC Framework. MVC паттерн не является новым, его корни уходят к 1978 году и проекту Smalltalk в Xerox PARC, но он завоевала огромную популярность

сегодня в качестве паттерна для веб–приложений по ниже описанным причинам.

Взаимодействие пользователя с MVC приложением следует естественному циклу: пользователь совершает действие, в ответ на это приложение меняет свою модель данных и предоставляет пользователю обновленный вид. А затем цикл повторяется. Это очень удобно для веб–приложений, предоставляемых в виде серии HTTP запросов и ответов.

Необходимость веб–приложению объединять несколько технологий (например, базы данных, HTML и исполняемый код), как правило, разбивается на множество уровней или слоев. Моделей, которые вытекают из этих комбинаций, естественны для концепции MVC.

ASP.NET MVC Framework реализует MVC паттерн и, тем самым, обеспечивает значительно улучшенное разделение концепций. На самом деле ASP.NET MVC реализует современный вариант MVC паттерна, который особенно хорошо подходит для веб–приложений.

Применяя и адаптируя MVC паттерн, ASP.NET MVC Framework сильно конкурирует с Ruby on Rails и аналогичными платформами, и переносит MVC паттерн в основное русло мира .NET. Суммируя опыт и лучшую практику разработчиков, использующих другие платформы, можно сказать, что ASP.NET MVC может предложить даже больше, чем Rails.

Расширяемость

Внутренние компоненты настольного ПК являются независимыми частями, которые взаимодействуют только через стандартные, публично документированные интерфейсы. Вы можете легко вынуть видеокарту или жесткий диск и заменить его другим от другого производителя и будете уверены, что он впишется в слот и будет работать. MVC Framework также построен как ряд независимых компонентов, удовлетворяющих .NET интерфейс или построенных на абстрактном базовом классе, так что вы можете легко заменить компоненты, такие как система маршрутизации, движок для просмотра и так далее другими.

Контроль над HTML и HTTP

ASP.NET MVC признает важность получения чистой, соответствующей стандартам разметки. Его встроенные методы HTML помощника предоставляют соответствующие стандартам выходные данные, но есть и более значительные философские изменений по сравнению с Web Forms. Вместо того чтобы плодить огромные участки HTML, котором нам сложно управлять, MVC Framework рекомендует вам выработать простой, элегантный стиль разметки с помощью CSS.

Конечно, если вы хотите использовать некоторые готовые виджеты для сложных элементов пользовательского интерфейса, такие как выбор даты или каскадное меню, то вам стоит знать, что подход ASP.NET MVC к разметке упрощает использование лучших в своем роде UI библиотек, таких как JQuery

UI или библиотеки Yahoo YUI. Разработчики JavaScript будут рады узнать, что ASP.NET MVC так хорошо сработался с популярной библиотеки JQuery, что Microsoft сделал JQuery встроенной частью шаблона проектов ASP.NET MVC и даже позволяет напрямую ссылаться на .js файл jQuery на собственных CDN серверах Microsoft.

Страницы, сгенерированные ASP.NET MVC, не содержат никаких данных View State, поэтому они могут быть в сотни килобайт меньше, чем обычные страницы, созданные при помощи ASP.NET Web Forms. Несмотря на современную широкополосную связь и быстрые подключения, эта экономия пропускной способности до сих пор чрезвычайно притягательна для конечных пользователей.

Как Ruby on Rails, ASP.NET MVC работает в гармонии с HTTP. Вы полностью контролируете запросы, проходящие между браузером и сервером, поэтому вы можете подогнать настройки под себя, на сколько вам это нравится. AJAX сделан просто, и нет никакого автоматического обратного вмешательства в состояния на стороне клиента. Любой разработчик, который в первую очередь фокусируется на веб программировании, почти наверняка посчитает это освобождением и будет наслаждаться рабочим процессом.

Архитектура MVC дает вам отличную возможность создавать ваше приложение таким, чтобы его можно было легко сопровождать и тестировать, потому что вы, естественно, захотите разделить логические блоки приложения по независимым частям программного обеспечения. Тем не менее, создатели ASP.NET MVC на этом не остановились. Для поддержки модульного тестирования они приняли компонето–ориентированный дизайн фреймворка и убедились, что каждая отдельная часть построена так, чтобы отвечать требованиям модульного тестирования.

Тестируемость – это не только вопрос модульного тестирования. ASP.NET MVC приложения также хорошо работают с инструментами автоматического тестирования. Вы можете написать тестовые скрипты, которые имитируют взаимодействие с пользователем, без необходимости гадать, какие структуры HTML элементов, CSS классы или ID будет генерировать фреймворк, и вам не придется беспокоиться о структуре, если она вдруг неожиданно изменится.

Мощная система маршрутизации (роутинга).

Стиль ссылок изменился, поскольку технология веб–приложений улучшилась. Такие ссылки, как эта, можно встретить довольно редко:

```
/App_v2/User/Page.aspx?action=show%20prop&prop_id=82742
```

Теперь они заменены более простым и чистым форматом:

```
/to-rent/chicago/2303-silver-street
```

Есть несколько веских причин для заботы о структуре URL. Во–первых, поисковые системы придают значительный вес ключевым словам, находящимся в URL. Поиск "аренда в Чикаго" (rent in Chicago) имеет гораздо больше шансов с простым URL. Во–вторых, многим пользователям Интернета теперь хватит навыков и знаний, чтобы понять URL, и оценить возможности навигации, набрав его в адресной строке своего браузера. В–третьих, когда кто–то понимает структуру URL, он, скорее всего, будет ссылаться именно на него, поделится этой ссылкой с другом или даже продиктует ее вслух по телефону. В–четвертых, такая ссылка не предоставляет технические подробности, папки, имена файлов и структуру приложения на весь общественный Интернет, так что вы можете изменить внутреннюю реализацию, не нарушая ссылки.

В более ранних фреймворках было сложно реализовать чистые ссылки, но ASP.NET MVC использует возможность System.Web.Routing, которая по умолчанию создает чистые URL–адреса. Теперь вы можете контролировать схему ссылок и ее связь и отношение к приложению, то есть вы свободны в создании шаблона URL–адресов, которые являются значимыми и полезными для пользователей, без необходимости соответствовать предопределенному шаблону.

Возможность разрабатывать в лучших сегментах платформы ASP.NET.

Существующая платформа Microsoft ASP.NET предоставляет зрелый, хорошо зарекомендовавший себя набор компонентов и средства для разработки эффективных и действенных веб–приложений.

Во–первых, и что наиболее очевидно, поскольку ASP.NET MVC основан на. NET платформе, у вас есть возможность писать код на любом .NET языке и иметь доступ к тем же API функциям, не только к MVC, но и к обширной .NET библиотеке классов и огромной экосистеме сторонних .NET библиотек.

Во–вторых, готовые возможности платформы ASP.NET, такие как мастер–страницы, аутентификация, роли, профили и интернационализация, могут уменьшить количество кода, который нужно писать и поддерживать для любых веб–приложений, и эти функции так же эффективны при использовании в MVC Framework, как и в классических проектах WebForms. Вы можете заново использовать некоторые встроенные серверные элементы управления WebForms, а также свои собственные элементы управления из предыдущих проектов ASP.NET в приложениях ASP.NET MVC (если они не зависят от некоторых конкретных возможностей WebForms, таких как View State).

С момента своего создания в 2002 году .NET платформа Microsoft неумолимо развивалась, поддерживая и даже определяя аспекты современного программирования.

ASP.NET MVC 4 был создан для .NET 4.5, поэтому его API в полной мере принял преимущества самых современных языков и технологий, в том числе ключевое слово await, методы расширений, лямбда–выражения, анонимные и динамические типы и LINQ (Language Integrated Query). Многие из методов API MVC Framework и паттерны кодирования следуют более чистым, более выразительном композициям, чем это было доступно в ранних платформах.

ASP.NET MVC имеет открытый исходный код.

В отличие от предыдущих платформ веб–разработки от Microsoft, вы можете загрузить исходный код для ASP.NET MVC и даже изменить и скомпилировать собственную версию. Это имеет неоценимое значение, когда ваша отладка касается системы компонентов, и вы хотите зайти в код (и даже прочитать комментарии программистов–создателей). Это также полезно, если вы создаете «продвинутые» компоненты и хотите посмотреть, какие возможности существуют для их развития или как действительно работают встроенные компоненты [4].

2.7 Microsoft Visual Studio 2013 Express for Web

Visual Studio – это полный набор инструментов и служб для создания различных приложений как для платформы Microsoft, так и для других платформ. Visual Studio также позволяет связать все проекты, группы и всех заинтересованных лиц. Группа проектов может работать более гибко практически где угодно независимо от используемого средства разработки (в том числе Eclipse и Xcode). Вы сможете разрабатывать важные приложения .NET, писать невероятно быстрый код с помощью C++ AMP или тестировать и отлаживать облачное приложение на HTML или JavaScript, которое работает на множестве устройств.

Visual Studio Express 2012 for Web – для создания приложений, работающих во всемирной паутине. Ниже приведено изображение окна Visual Studio Express 2012 for Web (Рисунок 2.1).

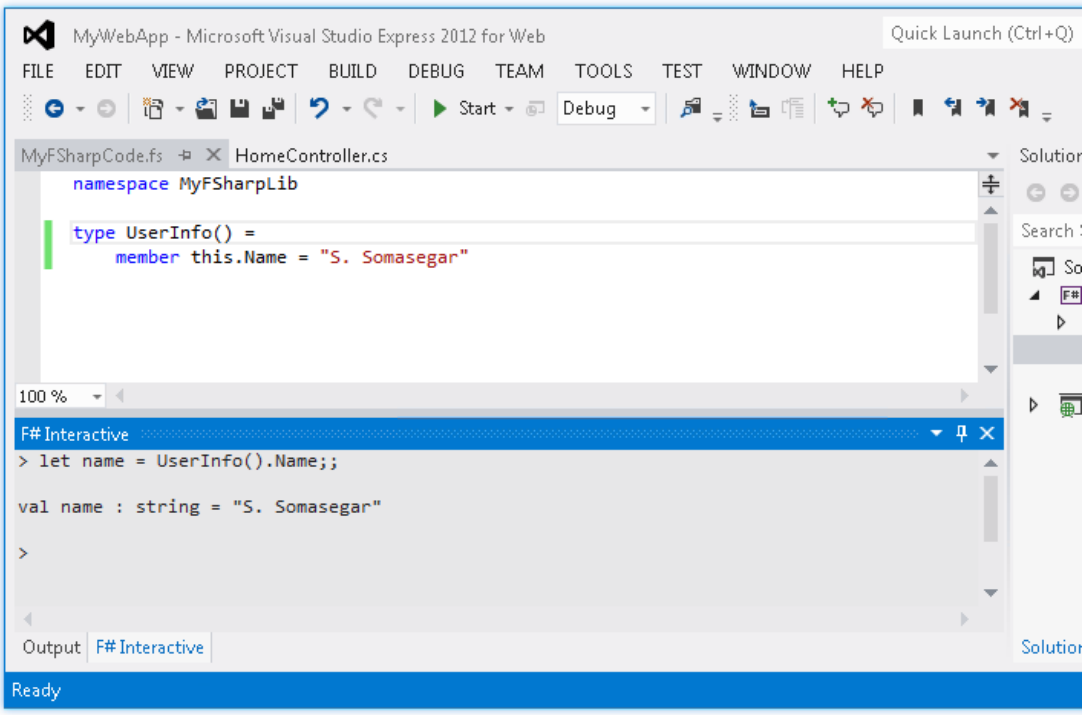

Рисунок 2.1 – Окно Visual Studio Express 2012 for Web

3 Проектирование сайта

3.1 Постановка задачи

В процессе проектирования дипломного проекта необходимо разобраться в действии серверной технологии ASP.net, создать информационный сайт города Зайсан, который будет предоставлять подробную и корректную информацию о городе, его окрестностях, природе, местах отдыха и инфраструктуре, для тех пользователей, которых интрересует информация, связанная с изложенной тематикой.

Конкретной задачей при разработке информационного сайта является его создание с использованием серверной технологии ASP.net, а также нужно использовать такие языки и технологий, как: HTML, CSS, JavaScript, jQuery, C#, XML–данные. Дизайн сайта должен быть привлекательным, а также соответсвовать контенту сайта. Стоит использовать неброскую цветовую схему. Не стоит перегужать веб–страницы различными гаджетами, достаточно и одного. Сайт должен быть легким для навигации и его контент должен часто обновляться.

3.2 Алгоритм реализации проекта

1) Определение цели создания сайта. Поскольку это влияет на многое. Надо выбрать технологии для использования. Платный или бесплатный будет вариант его создания. Начинаем делать черновой вариант.

2) Построение структур и схем. Построение схемы XML–файлов для хранения данных (статей). Создание предварительного шаблона сайта. Структурное расположение всех блоков сайта, а так же определение всех направлений и ссылок.

3).Регистрация доменного имени сайта, которое будет отличительной чертой сайта и будет нести информацию о содержимом сайта. Лучше приобрести домен, чтоб избежать проблем с бесплатным. Также, лучше арендовать хостинг, чем использовать бесплатный на популярных платформах. Он обеспечит круглосуточное нахождение сайта на сервере.

4) Выбор дизайна сайта. Выбор стилевого оформления.

5) Web–программирование – разработка серверного кода на ASP.net, создание и маркировка страниц на HTML, создание хранилища данных с XML–файлами, разработка функций на JS, подключение плагинов библиотеки jQuery.

6) Раскрутка и продвижение сайта не имеет своего окончания. Чтоб сайт имел посетителей необходимо пополнение вашего сайта свежей, интересной информацией. С помощью рекламы сети сайт раскручивается. Это может быть и обмен ссылками с иными сайтами, собственная рассылка, участие в системах рекламы в сети и т.д.

3.3 Структурирование и создание макета

3.3.1 Структура веб–сайта

Для начала нужно определиться с содержимимым сайта. Какие будут страницы, с какими данными и т.д.

После анализа содержимого большого количества информационных сайтов, были определены основные страницы сайта. Это:

– главная страница – отображение базовых данных о городе;
– статьи – доступ к статьям (страница с кратким описанием);
– Зайсан – справочная информация о городе, окрестностях и природе;
– галерея – каталог изображений;
– контакты – данные для связи с администратором сайта.

Предполагается, что содержимое страницы Зайсан, будет иметь несколько подразделов. К этим подразделам относятся подразделы «Природа», «Места» и «История». Такая структура имеет начало – первую страницу, корень дерева.

Так как структура нашего сайта не древовидная, а совмещенная с последовательной, ее можно назвать смешанной или гибридной. На рисунке 3.1, представлена структура сайта.

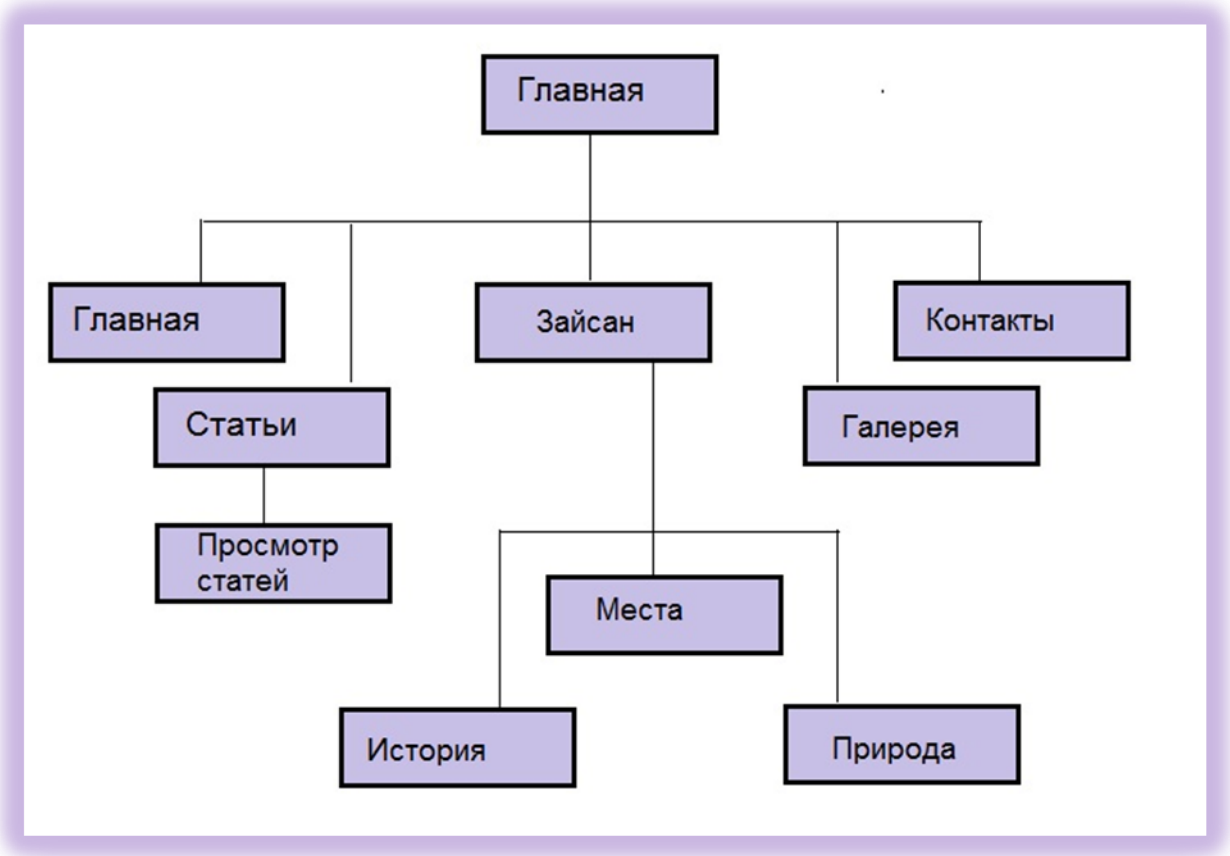

Рисунок 3.1 – Гибридная структура сайта

3.3.2 Структура XML–данных

Рассмотрим атрибуты значений для XML–файлов. В данном файле будут находиться статьи. Все статьи сосредоточены в одном файле, их пополенение будет осуществляться там же. Каждая статья, находящаяся в файле, имеет свои значения атрибутов, такие как номер статьи, используемый для правильного перехода к статье, так же дата размещения статьи на странице, заголовок статьи, несущий в себе краткое описание содержимого статьи, изображения, рассматриваемые в статье, и сам текст статьи.

Атрибуты сущности «Статьи»:

– articleID – идентификационный номер статьи, так называемый уникальный ключ;

– date – дата создания статьи;

– topic – заголовок статьи;

– img – изображения используемые в статье;

– content – сам текст статьи.

Специально, для раздела статьи, будет работать поиск, который находит искомое слово в статьях. Ниже представлена структура XML–файла (Рисунок 3.2).

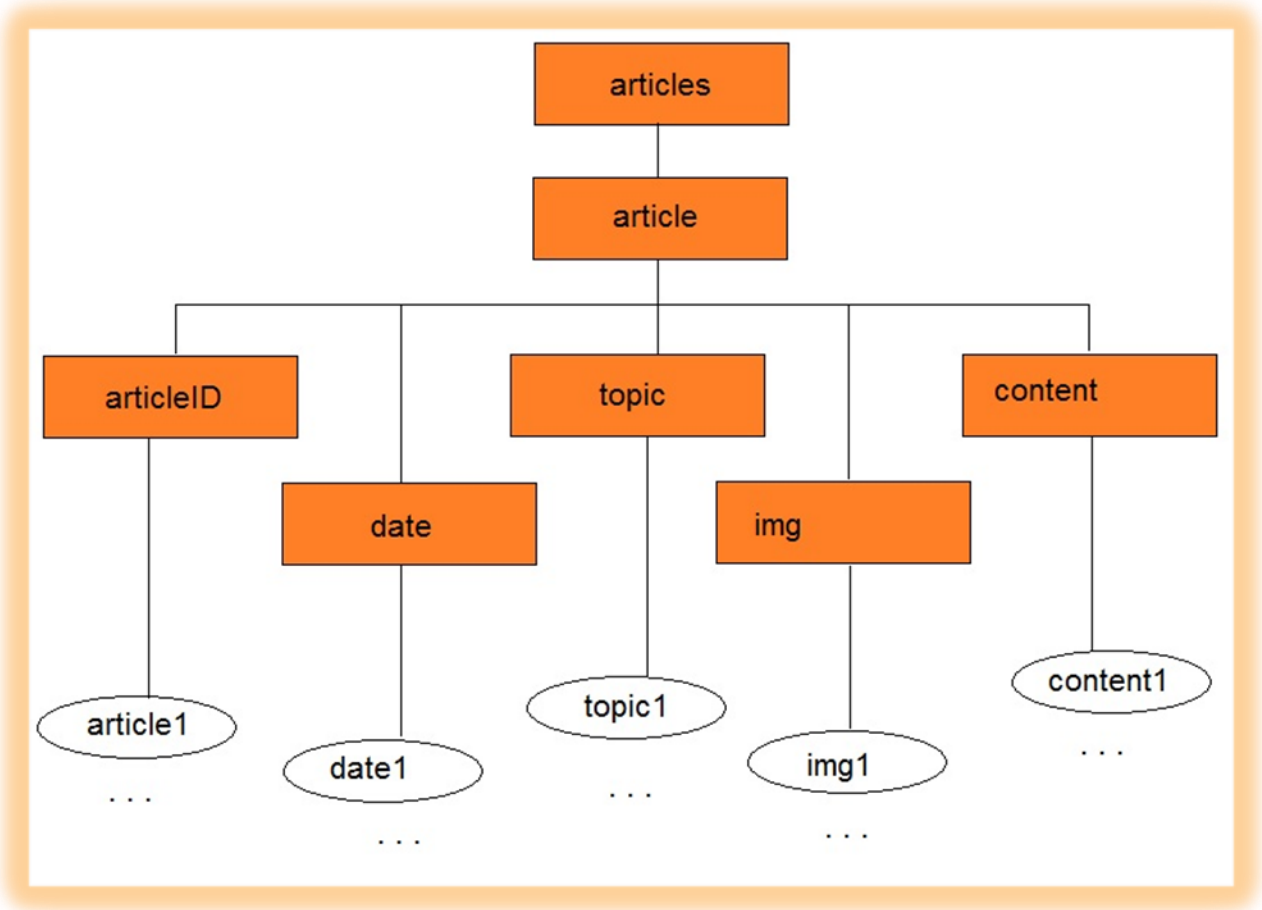

Рисунок 3.2 – Структура XML–файла

3.3.3 Создание макета сайта

Дизайн макета сайта – это графический образ представления будущего сайта. Макет сайта – это далеко не сам сайт, это только лишь приблизительная картинка, наиболее близко представляющая внешний вид страниц будущего сайта. Для начала требуется создать представление сайта. Предварительный макет сайта сделан на Paint и представлен на рисунке 3.3.

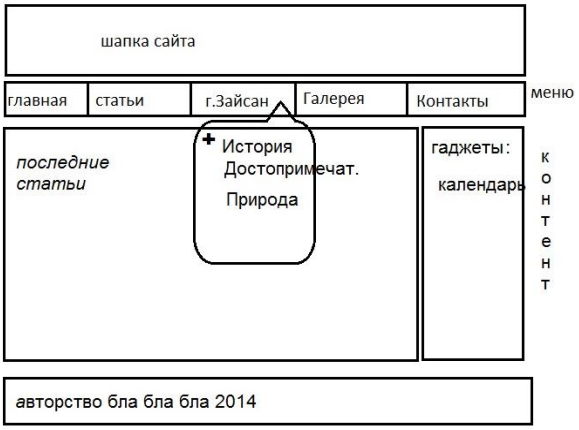

Рисунок 3.3 – Предварительный макет страниц сайта

Так как, вышерасположенный макет сайта, точнее его главной страницы был предварительным, то со временем разработк он подвергся значительным изменениям. Главным образом расположением панели навигации (меню), она получила свое раположение в правом углу шапки сайта. Стоит отметить, что на главной странице не будут отображаться статьи, вместо них будет расположена подробная информация о городе, на сегодняшний день. Настоящий макет сайта расположен ниже (Рисунок 3.4).

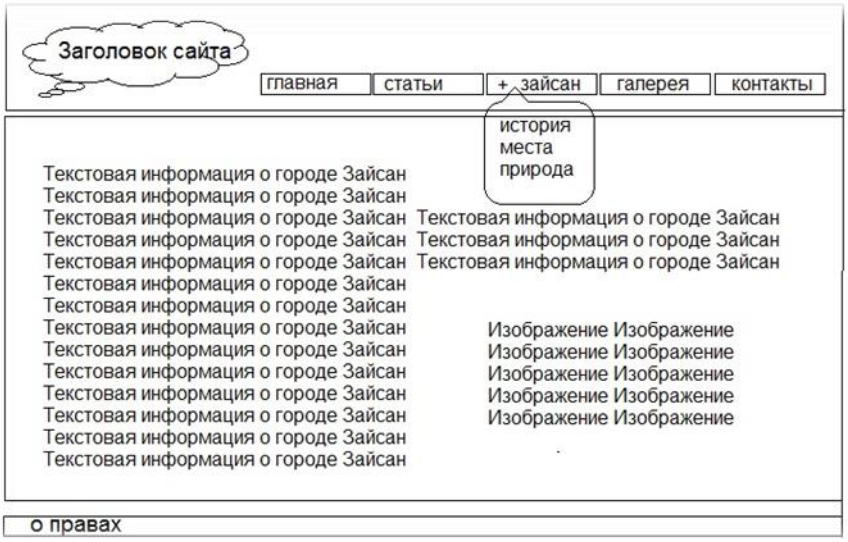

Рисунок 3.4 – Используемый макет сайта

3.3.4 Дизайн сайта

Adobe Photoshop – самый популярный графический пакет для профессионального редактирования любых форматов изображений. Художники, работающие в области компьютерного дизайна, могут бесконечно спорить о том, что лучше Mac или PC, но все они едины в одном – без Adobe Photoshop практически никто из них не обходится. Потрясающие инструменты для работы с растровой графикой уже давно возвели Adobe Photoshop в ранг короля графических редакторов. Изначально программа была разработана как редактор изображений для полиграфии. В настоящее время Adobe Photoshop широко используется и в веб–дизайне. Нижерасположено изображение рабочего окна программы (Рисунок 3.5).

Рисунок 3.5 – Рабочее окно Adobe PhotoShop CS6.

Дизайн сайта будет реализован в определенной цветовой схеме, представленной на рисунке 3.6, использованы приглушенные оттенки синего цвета, а для фона выбрана смесь из белого и серого цветов.

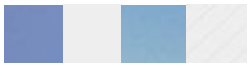

Рисунок 3.6 – Используемая цветовая схема

Шапка сайта была сделана на программе Adobe Photoshop CS6. При ее создании было использовано три изображения окрестностей города Зайсан. Изображения были соеденены в одну панораму. Процесс создания шапки сайта на рисунке 3.7.

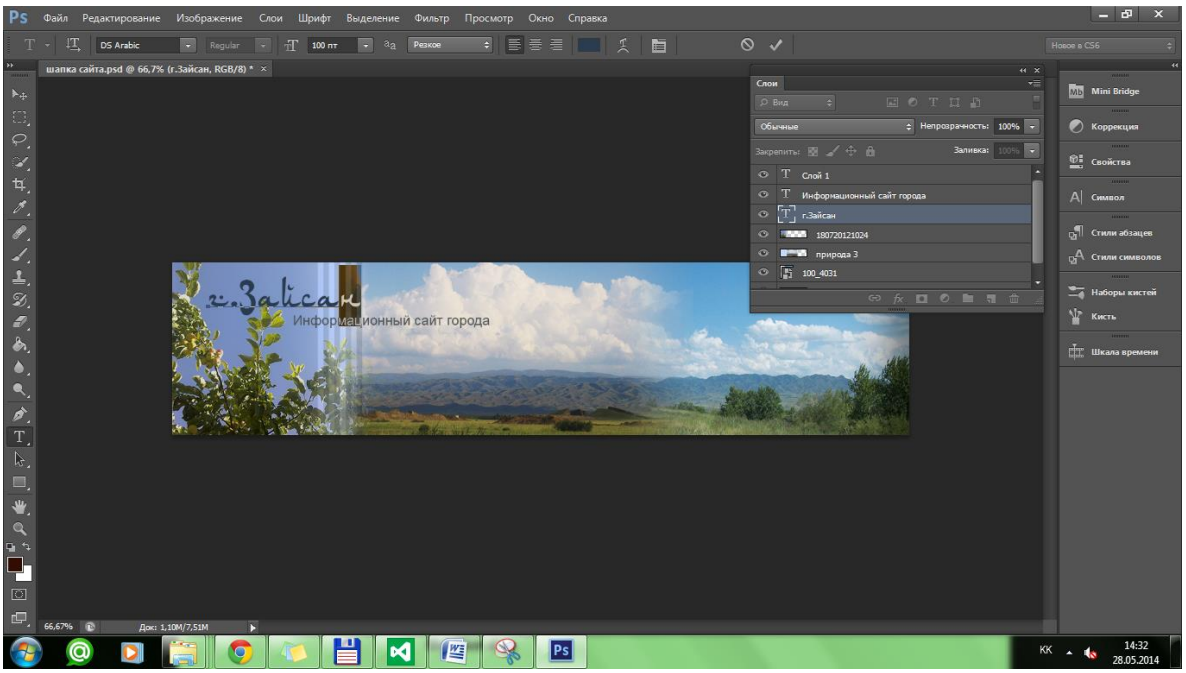

Рисунок 3.7 – Процесс создания шапки сайта

Для создания надписи была использована функция ввода текста, а также был использован специальный шрифт – «DC Arabic». Заголовок был расположен в правом верхнем углу шапки сайта (Рисунок 3.8).

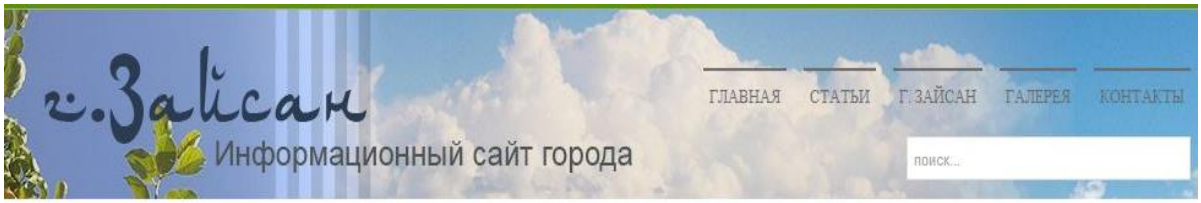

Рисунок 3.8 – Шапка информационного сайта г.Зайсан

Созданная шапка также является фоном для расположения панели навигации и поля для поиска (Рисунок 3.9).

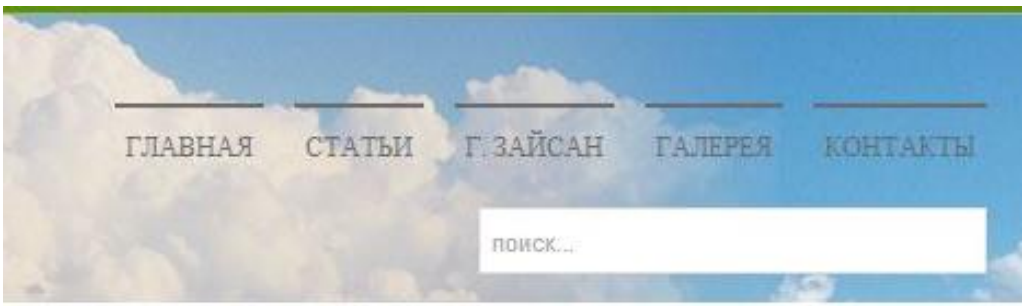

Рисунок 3.9 – Панель навигации по сайту (меню)

3.4 Реализация проекта

Реализация инфомацииононного сайта была разделена на несколько этапов. Во–первых, это составление цепочки целей и задач, для правильного расставления акцентов сайта и корректной подачи дизайна. Далее идет составление структуры сайта, схематизированное представление ее работы. Следом стоит необходимсоть предварительного проектирования макета сайта, создание отдельных элементов дизайна сайта. Далее верстка на языке гипертекстовой разметки, и предпоследним идет написание программного кода, для работоспособности веб–страниц и их ваймодействия. Во время проектирования была использована технология ASP.net WebForms.

3.4.1 Создание веб–страниц

Первой спроектированной страницей, является мастер –страница. Мастер–страница является основой, так называемым каркасом всех страниц. В этой странице размещаются все элементы, которые располагаются на всех страницах. Ниже представлено изображение написания кода в среде программирования Visual Studio Express 2012 for Web (Рисунок 3.10).

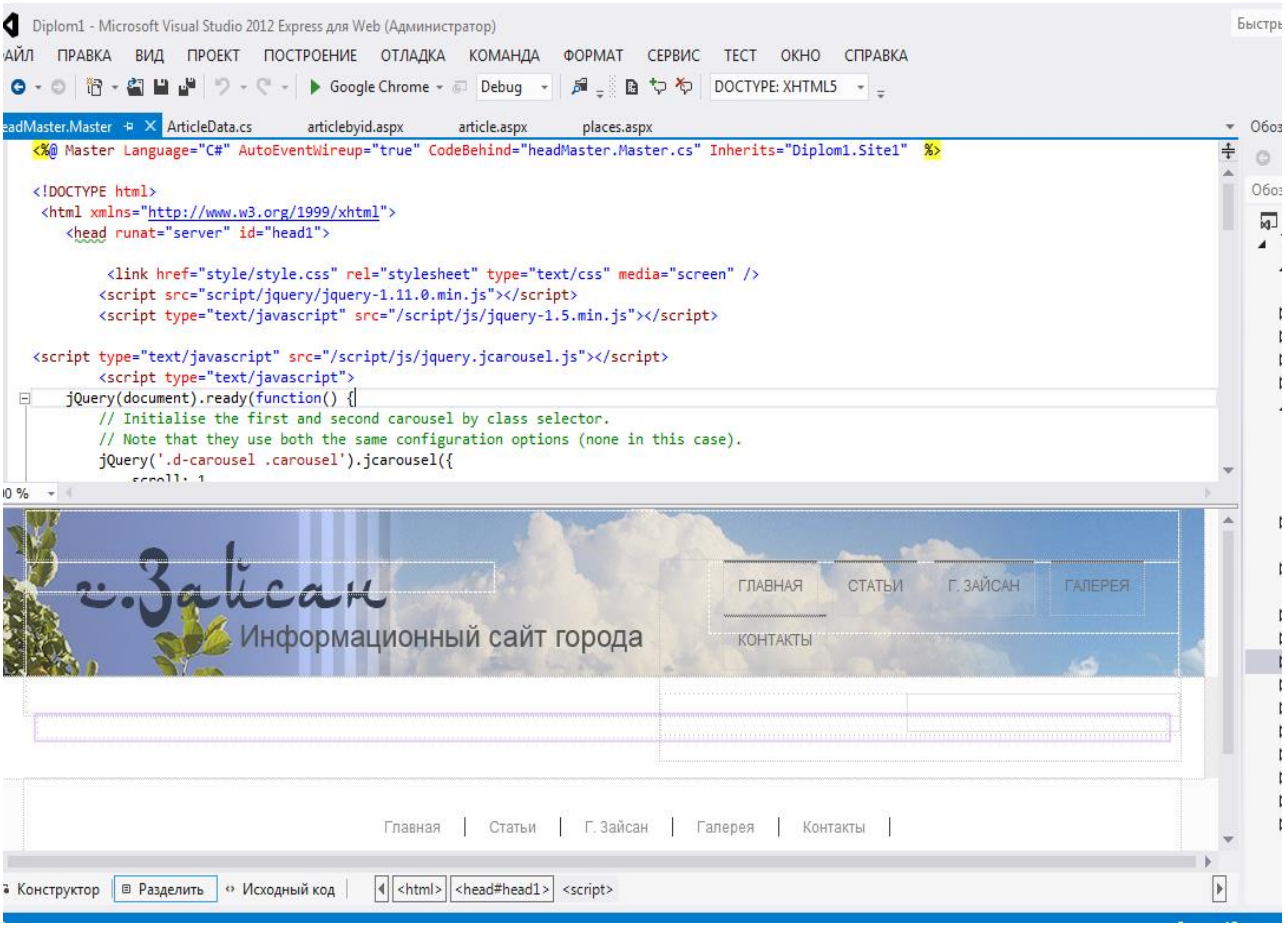

Рисунок 3.10 – Мастер – страница

На рисунке 3.11 представлена структура проекта Diplom1, в среде Visual Studio Express 2012. В нем содержатся все файлы, привязанные с данным проектом.

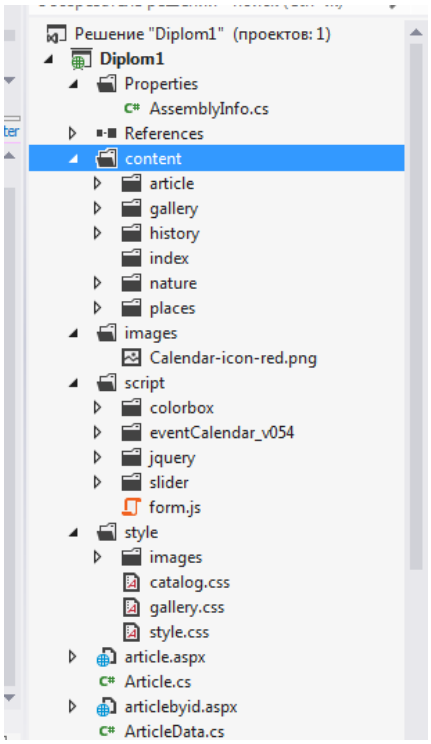

Рисунок 3.11 – Структура проекта

Последующей страницей, спроектированной после мастера–страниц, является главная страница сайта (Рисунок 3.12).

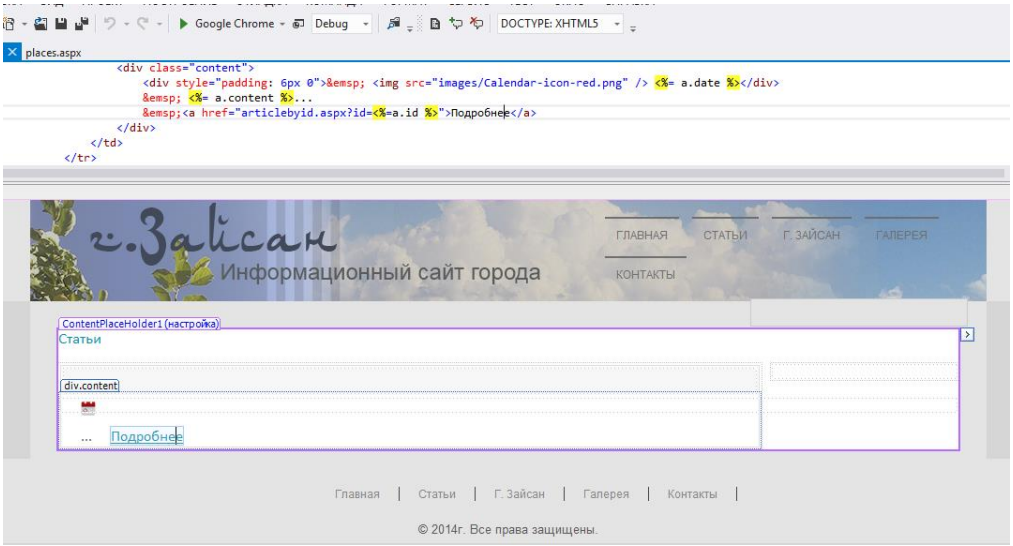

Рисунок 3.12 – Создание главной страницы

Таким же образом были созданы все последующие страницы сайта.

3.4.2 Работа с JavaScript jQuery

Идея JavaScript очень проста. Все операции, которые можно исполнять в программе на JavaScript, описывают действия над хорошо известными и понятными объектами, которыми являются элементы рабочей области программы Netscape Navigator и контейнеры языка HTML. Собственно объектная ориентированность JavaScript на этом и кончается. Есть только объекты с набором свойств и набор функций над объектами. Последние называются методами. Кроме методов существуют и другие функции, которые больше похожи на функции из традиционных языков программирования и позволяют работать со стандартными математическими типами или управлять процессом выполнения программы. Еще в JavaScript есть события – аналог программных прерываний. Эти события также ориентированы на работу в World Wide Web, например, загрузка страницы в рабочую область Navigator'a или выбор гипертекстовой ссылки. Используя события, автор гипертекстовой страницы и программы ее отображающей может организовать просмотр динамических объектов, например, бегущая строка, или управление многооконным интерфейсом [12].

В данной работе были использованы готовые плагины из библиотеки jQuery. Первый плагин это – jCarousel.js. jCarousel – это jQuery–плагин для реализации вертикальной или горизонтальной "карусели" изображений или данных. Он использован на странице с изображением достопримечательностей города (Рисунок 3.13).

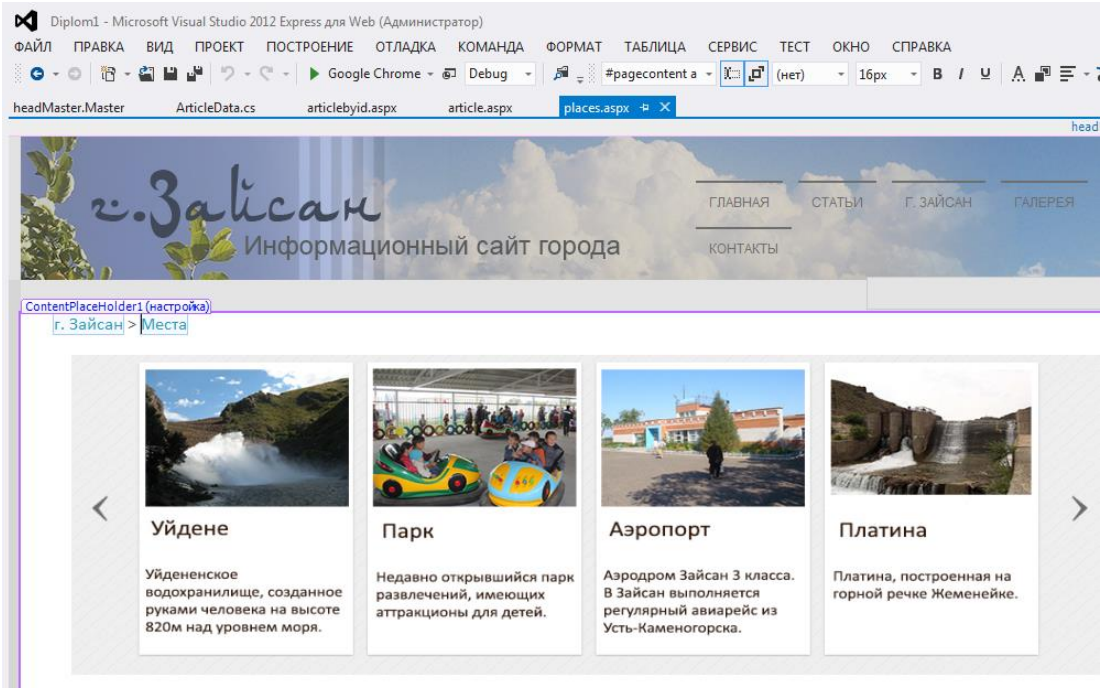

Рисунок 3.13 – Использованный плагин jCaruosel.js

На следующем изображений можно увидеть привязку к библиотеке jQuery (Рисунок 3.14).

```
<%@ Page Title="Галерея" Language="C#" MasterPageFile="~/headMaster.Master" AutoEventWireup="true" CodeBehind="galler

<asp:Content ID="galleryHead" ContentPlaceHolderID="head_ContentPlaceHolder" runat="server">
    <link href="script/colorbox/colorbox.css" type="text/css" rel="stylesheet" />
    <link href="style/gallery.css" type="text/css" rel="stylesheet" />
    <script src="script/colorbox/jquery.colorbox.js"></script>
     <script >
         jQuery(document).ready(function ($) {
             $(".gallery_img").colorbox({ rel: 'gallery_img', width: '50%' });
         });
    </script>
</asp:Content>
<asp:Content ID="Content1" ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
     <a href="gallery.aspx">Галерея</a>
    <br/>
     <table id="gallery-table">
        <tr>
            <td>
                <img class="gallery_img"  src="content/gallery/Акимат.jpg" title="Акимат" />
                <div class="title">Акимат</div>
            </td>
            <td>
                <img class="gallery_img"  src="content/gallery/Байтерек на площади.jpg" title="Байтерек на площади" /
                <div class="title">Байтерек на площади</div>
            </td>
            <td>
                <img class="gallery_img"  src="content/gallery/Вид с гор.jpg" title="Вид с гор" />
```

Рисунок 3.14 – Привязка библиотеки jQuery

Следующим, использованным плагином библиотеки jQuery, был «слайдер». Слайдер позволяет открыть изображение в полном размере, осуществляется пролиствание направо и налево (Рисунок 3.15).

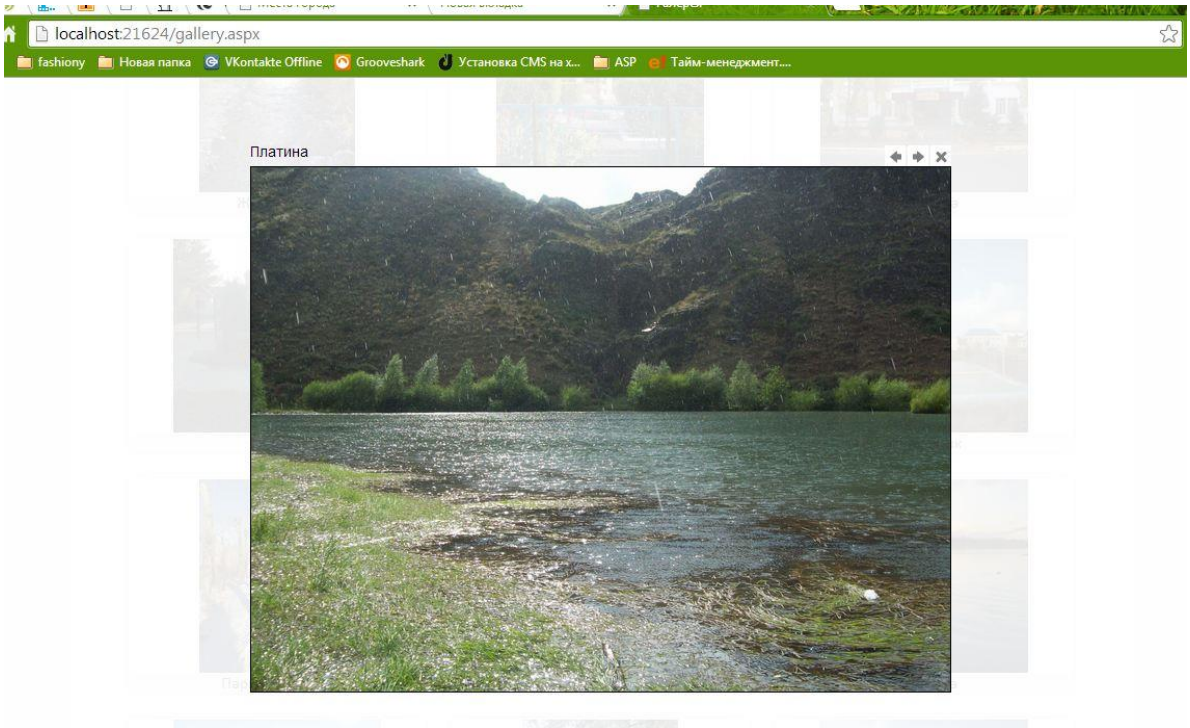

Рисунок 3.15 – Увеличение изображения с помощью jSlider.js

3.4.3 Написание кода для страниц

Для реализации функионала сайта пишется код для страниц. Все страници имеющиеся в сайте рассмотрены на рисунке 3.16.

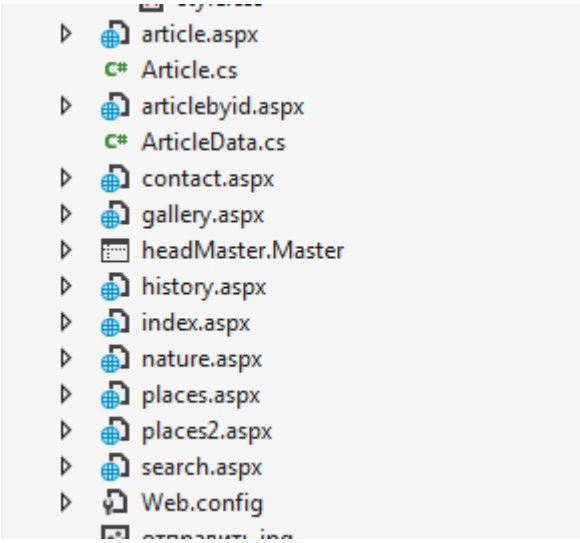

Рисунок 3.16 – Веб–страницы сайта

Код для функционирования страниц написан на языке C# с использованием объектно–ориентированного программирования. Код может ыноситься в отдельный файл, с расширением .aspx.cs. Ниже изображено окно с кодом для страницы со статьями (Рисунок 3.17).

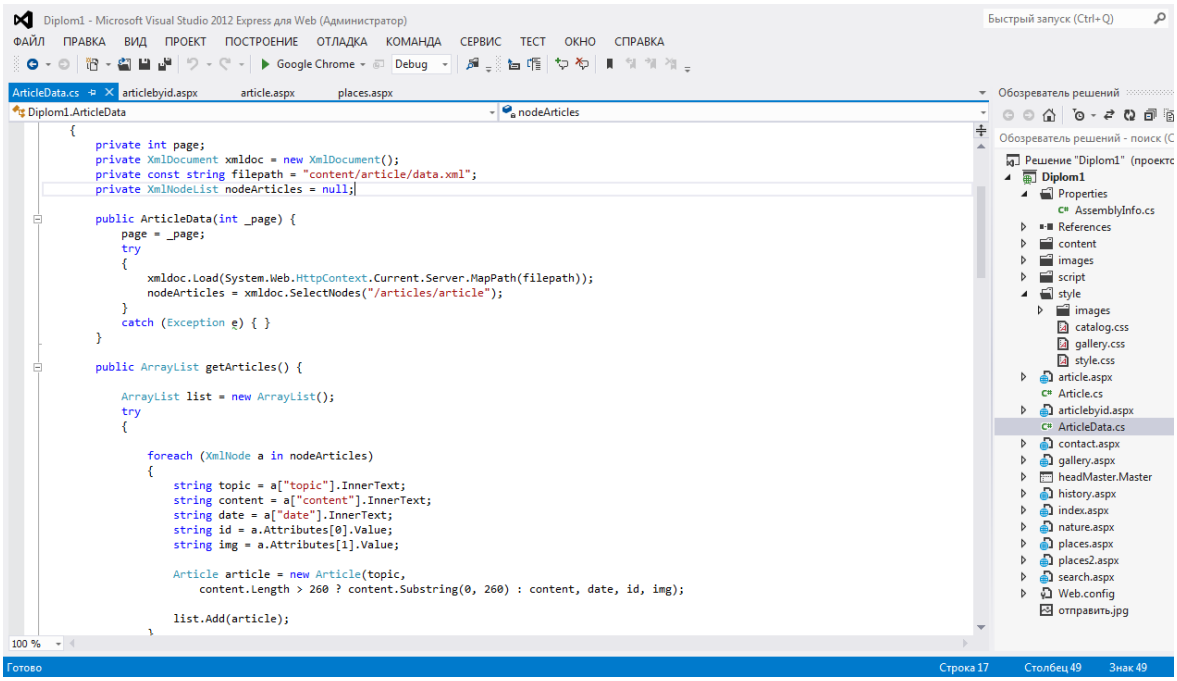

Рисунок 3.17 – Код функционирования со стороны сервера

Так же программный код можно писать прямо внутри кода страниц .aspx (Рисунок 3.18).

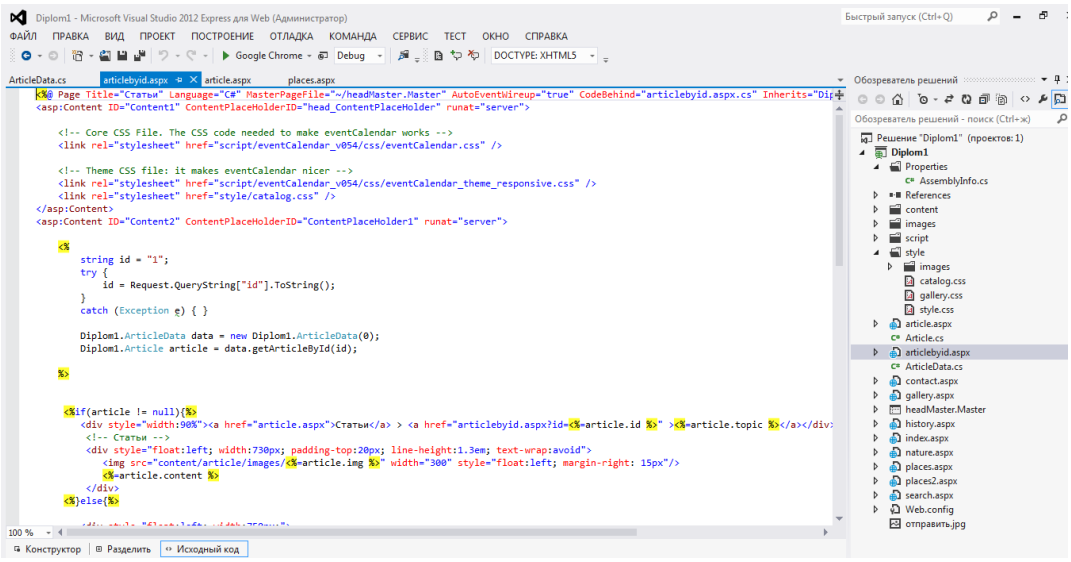

Рисунок 3.18 – Смешанный код .aspx и .cs

3.5 Результаты реализации проекта

Результатом проектирования является разработка сайта. Сайт был полностью спроектирован, все недочеты, и ошибки были исправлены.

В данном разделе представлены скриншоты всех страниц сайта.

Главная страница сайта содержит информацию о городе (Рисунок 3.19).

Рисунок 3.19 – Главная страница сайта

Внизу страниц так же имеется навигация по сайту (Рисунок 3.20).

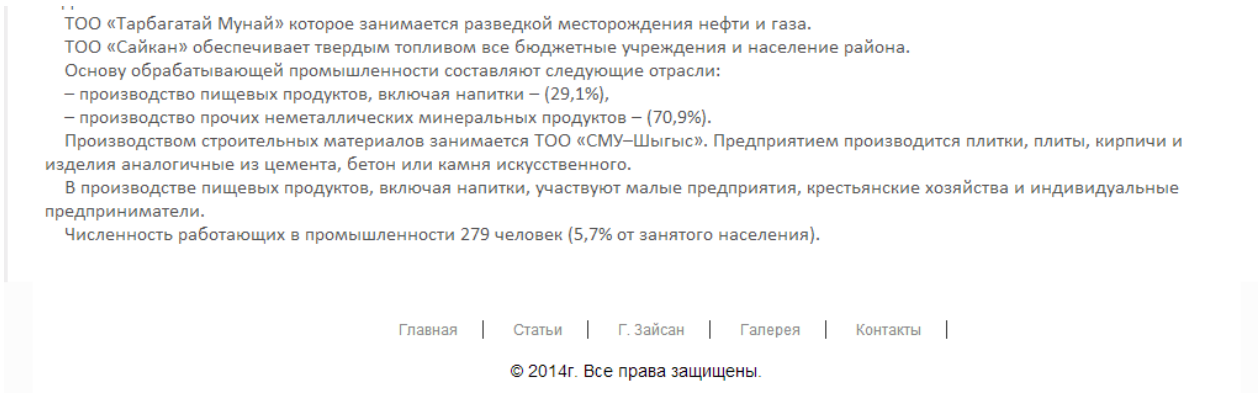

Рисунок 3.20 – Низ страницы сайта

Статьи расположены в разделе «Статьи». В данной странице рапологаются 10 последних статей. В правом блоке сраницы имеется гаджет – «Календарь» (Рисунок 3.21).

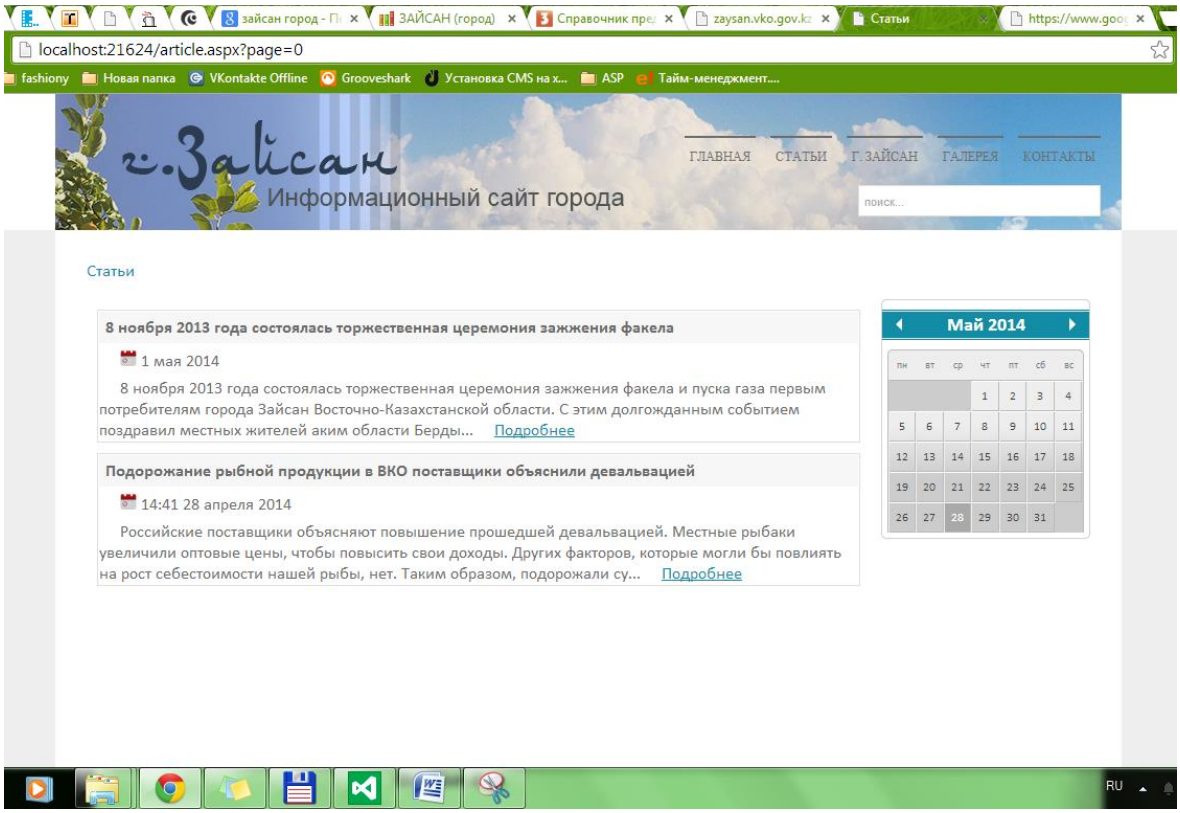

Рисунок 3.21 – Раздел «Статьи»

При нажатии на ссылку «Подробнее» открывается сама статья на этой же странице и в этом же блоке (Рисунок 3.22).

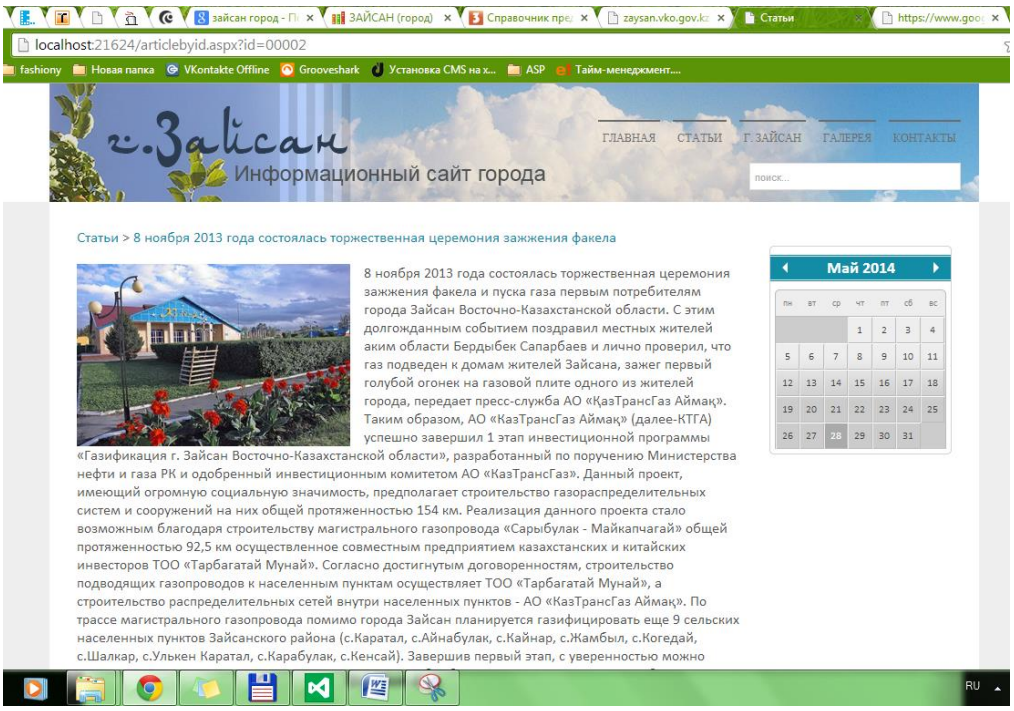

Рисунок 3.22 – Просмотр отдельной статьи

В разделе «г.Зайсан» имеется три подраздела. Подраздел «История» раскрывает собой историю города Зайсан, начиная со дня его основания (Рисунок 3.23).

Рисунок 3.23 – Подраздел «История»

Подраздел «Места» предоставляет пользователям перечень достопримечательностей города Зайсан и их описания (Рисунок 3.24).

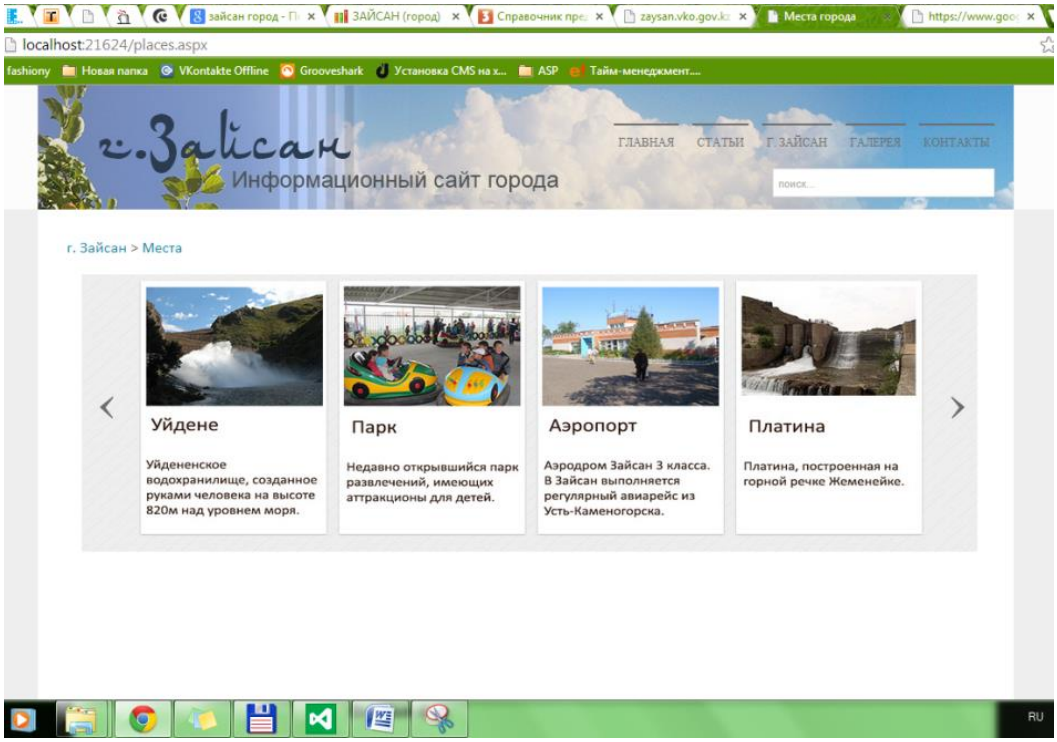

Рисунок 3.24 – Раздел «Места»

Подраздел «Природа» предоставляет пользователям информацию о природе, климате и ресурсах окрестностей города Зайсан (Рисунок 3.25).

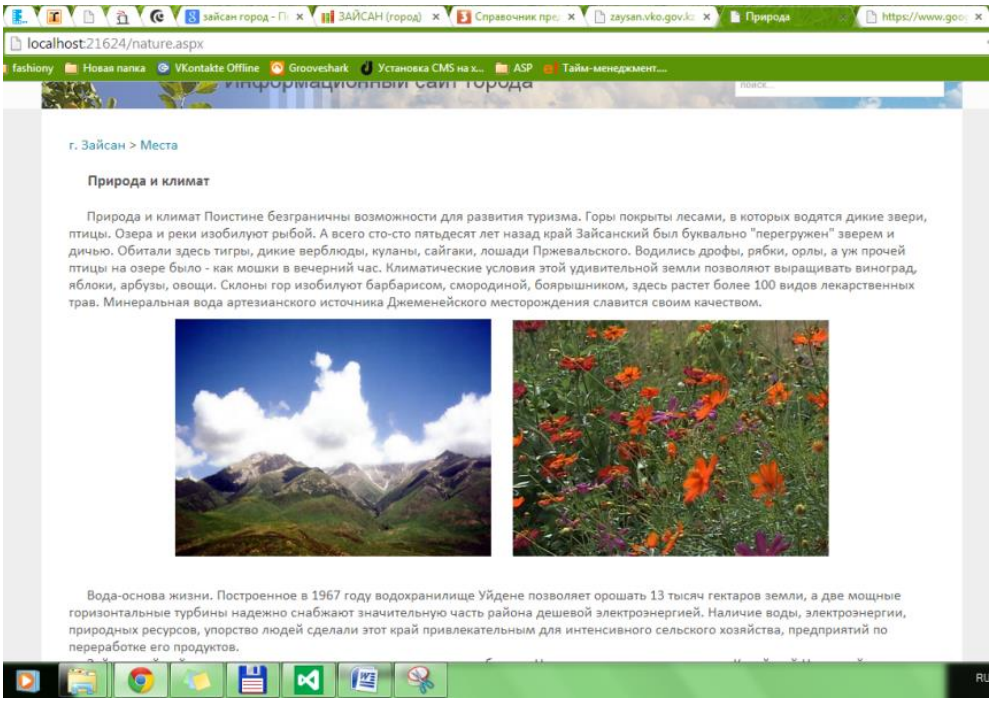

Рисунок 3.25 – Раздел «Природа»

Раздел сайта – «Галерея». Здесь расположены качественные фотографии достопримечательностей и природы города и его близлежащих окрестностей (Рисунок 3.26).

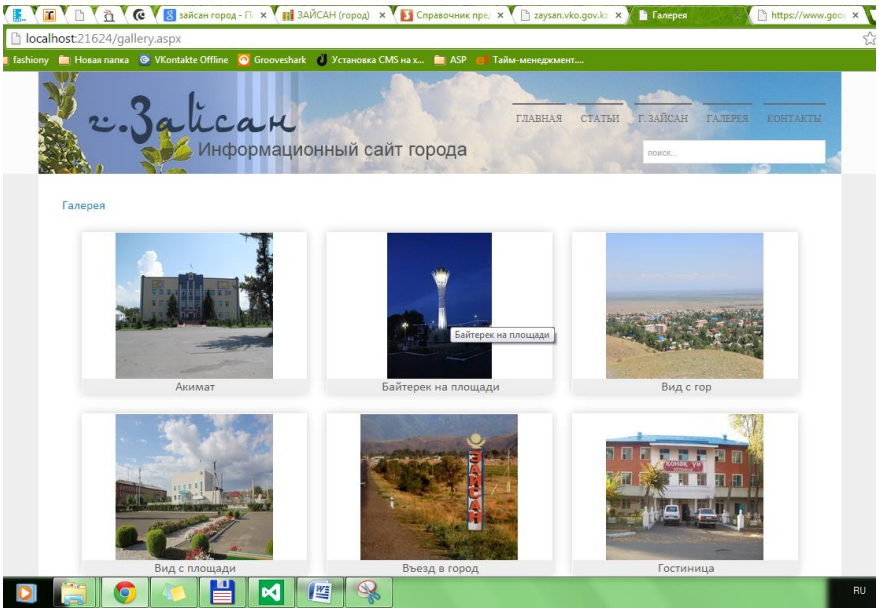

Рисунок 3.26 – Раздел «Галерея»

Так же в разделе можно просматривать фотографии по отдельности, виде слайдов. Эта функция была сформирована благодаря плагину jslider.js (Рисунок 3.27).

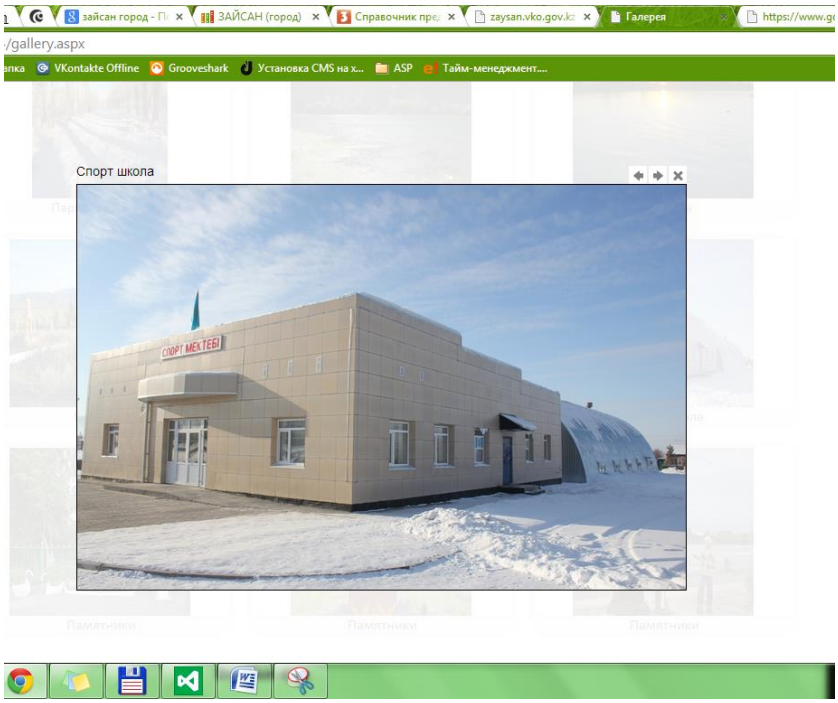

Рисунок 3.27 – Просмотр изображений в разделе «Галерея»

Также, для удобства пользователей, имеется поиск по статьям (Рисунок 3.28).

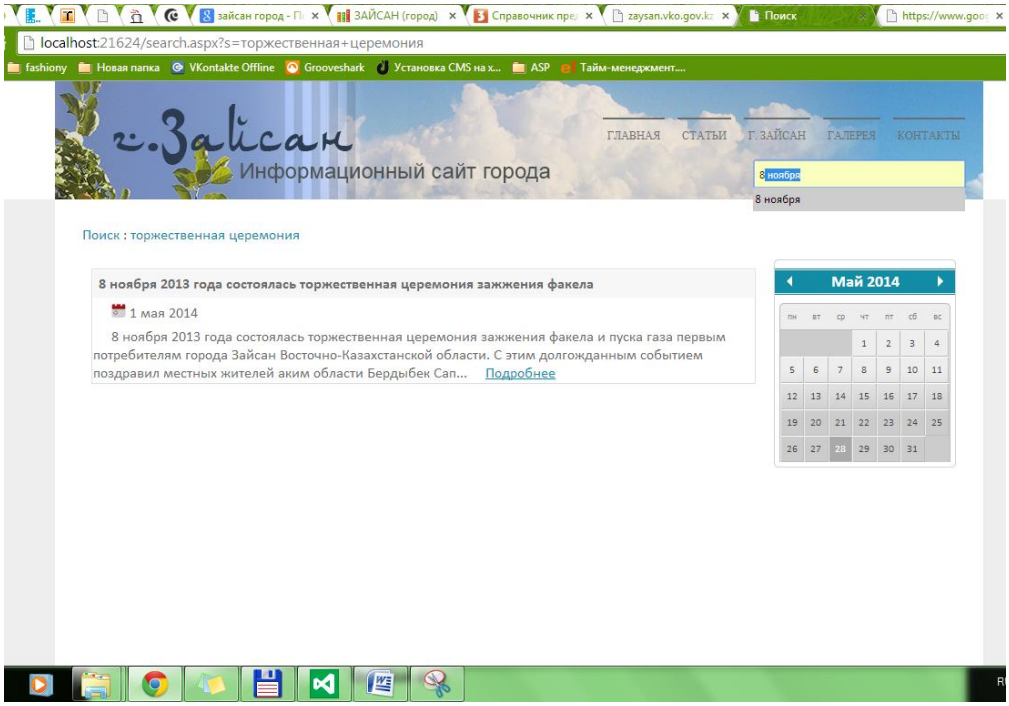

Рисунок 3.28 – Выполнение поиска в разделе «Статьи»

4 Безопасность жизнедеятельности

4.1 Анализ условий труда программиста при разработке веб–сайта

В процессе разработки веб–сайта используется компьютерная техника.

При работе с компьютерной техникой пользователь может быть подвержен такими опасными и вредными производственными факторами как: электромагнитное излучение, статическое напряжение, поражение электрическим током, недостаточная освещенность рабочего места, понижение ионизации воздуха и др. Данные вредные производственные факторы могут сказаться на здоровье пользователя компьютерной техники (в данном случае персонального компьютера).

Компьютерная техника является потенциальным источником формирования поражения электрическим током и пожарной опасности. К мерам обеспечения пожарной безопасности при работе с персональным компьютером относятся: правильное размещение оборудования и электрических кабелей. В виде профилактических мероприятий необходимо использовать скрытую электросеть, надежные розетки, рассчитывать нагрузку на сеть и соблюдать правила пожарной безопасности. Так же следует регулярно проводить чистку внутренних частей оборудования от пыли. Для предотвращения искрения не стоит часто беспокоить штепсельные вилки из розеток.

Компьютеры при работе создают электростатические поля, в зону действия которых попадают самые разные предметы. Во время работы вентиляторов, гоняющих воздух в системных блоках, выдуваются наружу наэлектризованные пылинки, которые оседают не только на нашей коже, но и в дыхательных путях. Экран монитора так же является накопителем зарядов статического электричества.

На сегодняшний день влияние статического электричества на организм человека малоизученная область. Большинство исследователей считают, что при воздействии статического электричества на человека, раздражаются нервные окончания кожи, а так же происходит изменение в ионном составе тканей. Все эти изменения приводят к повышенной утомляемости, раздражительности и плохому сну.

К мерам защиты от статического электричества стоит включить: влажную уборку, дополнительное увлажнение воздуха в рабочем помещений (относительная влажность должна быть больше 50%), проветривание помещения, заземление техники.

Стоит отметить, что статическое электричество является причиной деионизации воздуха. Деионизация воздуха объясняется притяжением отрицательно заряженных ионов к экрану монитора. При маленьком уровне ионизации падает работоспособность, увеличивается количество головных болей, понижается тонус, ухудшается умственная и физическая активность.

Внедрение системы искусственной ионизации и очистка воздуха поможет стабилизировать ситуацию.

Наряду с электрической опасностью существует так же электромагнитное влияние на пользователя ПК. Компьютер имеет два источника электромагнитного излучения (монитор и системный блок).

Большинство пользователей ноутбуков ошибочно полагают, что благодаря ЖК–мониторам проблема связаная с электромагнитным излучением становится менее опасной. Но в результате проведенных исследований ЖК–мониторы так же как и ЭЛТ–мониторы являются источниками опасного для здоровья электромагнитного излучения, но в меньшей степени. При работе с ПК в тесном, непроветренном помещений ситуация более усугубляется.

Более подверженными к электромагнитному излучению являются глаза и мозг, желудочно–кишечный тракт и мочеполовая система, кроветворные органы и имунная система. В целом состояние организма ухудшается.

В качестве защитных мер от электромагнитных излучений компьютера, стоит назвать систематические прогулки на свежем воздухе, проветривание помещение, физические занятия, соблюдение правил работы с компьютером, выбор и пользование хорошей техникой, отвечающим стандартам безопасности и санитарным нормам.

Уровень шумопроизводства ПК зависит от его мощности. Шум имеет свойство вызвать акустические раздражения. Человек подвергающийся систематическому шумовому воздействию, быстро устает, становится раздражительным, забывчивым. Источниками шума в ПК может являться системный блок, вентеляционные системы охлаждения устройства, процессор, а так же CD или DVD–приводы. Необходимо делать перерывы.

При работе за компьютером освещение играет большую роль. Свет не должен быть ярким, оптимальным вариантом при работе за современным ПК является рассянный приглушенный свет. Расположение компьютера относительно окна немаловажно. Компьютер стоит расположить таким образом, чтобы свет из окна не падал прямо на пользователя. Так как это может привести к переутомлению глаз при работе. Чтобы избежать этого, можно приобрести жалюзи или плотные шторы, которые будут защищать от прямх солнечных лучей.

Кроме описанных выше вредных и опасных факторов, влияющих на пользователя ПК во время работы, нужно отметить и другие вредные факторы, спровоцированные неправильной организацией работы за компьютером.

Организации рабочего места стоит уделять максимальное внимание, ведь от этого зависит здоровье пользователя. Если верить недавним исследованиям, то около 20% нарушений здоровья, связанных с работой за компьютером, вызванны незнанием основных правил работы с ним, а также неправильной организацией рабочего места.

Сидячая работа сама по себе является вредной для человека. Длительное прибывание в одно положнеии заставляет мышцы работать непрерывно без отдыха. Малоподвижность главная проблема пользователей ПК и

программистов. При снижении физической активности, вызванной продолжителным сидением, возрастает риск заболевании вроде ожирения, гемороя, остеохандроза. Если, работая за компьютером, человек сидит в неправильной позе, сутулится ил поддается вперед, его позвоночник деформируется и травмирует диски.

Необходимо правильно организовать рабочее место, постоянно следить за осанкой и делать перерывы на отдых и на физические упражнения.

Исходя из всех раскрытых вредных факторов, следует выделить требования по организации работы за компьютером:

– в рабочем помещений должно иметься как естественное так и искусственное освещение;

– помещение нужно оборудовать системами кондиционирования воздуха или эффективной вентиляцией;

– помещение должно проветриваться ежечасно;

– в помещений нужно проводить ежедневную влажную уборку;

– желательно закрывать окна шторами или жалюзи, чтобы не было прямых попаданий лучей солнца;

– искусственное освещение должно быть общим и равномерным;

– на рабочем столе должны свободно помещаться монитор, клавиатура, мышь, а также документы, книги, бумаги;

– стул должен регулироваться по высоте и углам наклона сиденья и спинки;

– экран монитора должен быть расположен на расстоянии не менее 55–60 см от глаз, так же он должен быть установлен так, чтобы не отражать посторонний свет. Следует правильно выбрать яркость экрана;

– рекомендуется хотя бы один–два раза в день выполнять гимнастику для глаз;

– правильное положение рук при работе с клавиатурой и мышью: локти располагаются параллельно поверхности стола и под прямым углом к плечу;

– необходимо постоянно следить за положением тела в процессе работы, то есть за осанкой;

– время работы непосредственно за компьютером не должно превышать шести часов за смену. На протяжении рабочего дня следует устраивать перерывы продолжительностью 10–20 минут, и выполнять комплексы физических упражнений.

Характеристики рабочего помещения.

Рабочее помещение оборудовано на одно рабочее место.

Рабочее помещение расположено в здании, отдаленном от железнодорожных путей, крупных автомагистралей, аэропорта и прочих источников шума, способных оказывать влияние на процесс работы. Рассмотрим план рабочего помещения (Рисунок 4.1).

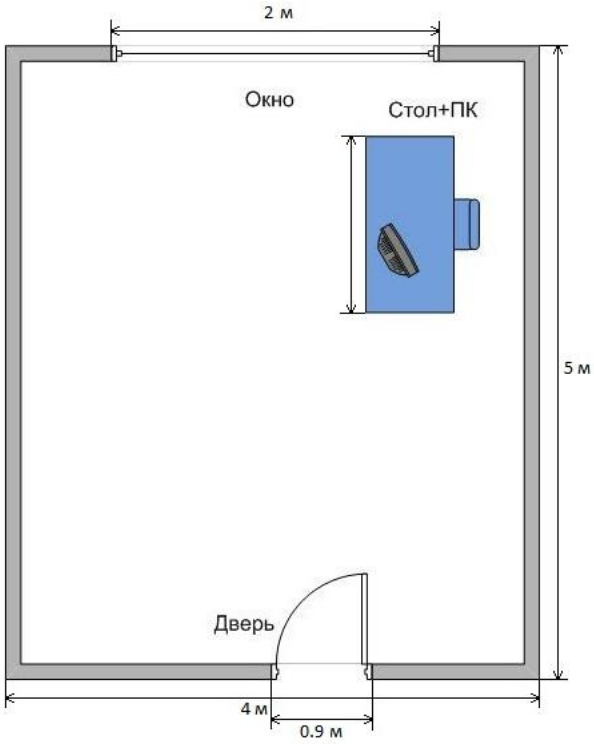

Рисунок 4.1 – План рабочего помещения

Помещение имеет нижеперечисленные параметры:

– находится на четвертом этаже пятиэтажного здания;
– размеры помещения: длинна 4 м, ширина 5м, высота 3м;
– большое окно, размером 2*2;
– вид светопропускающего материала – стеклопакет однокамерный;
– солнцезащитные устройства – шторы из плотной ткани;
– внутренняя отделка стен – светлая;
– боковое естественное освещение (свет из окна падает слева);
– искусственное освещение – один потолочный светильник с линейной люминесцентной лампой.

Характеристики используемой техники:

– Intel(R) Core(TM)2 Duo CPU T6600 @ 2.20GHz 2.20 GHz, 4 ГБ RAM / HDD 300 Gb;
– ЖК–экран Samsung, LED подсветка, (1366*768) 16:9 глянцевый, диагональ 15.6”;
– 3 линейных люминесцентных ламп;
– электропитание: переменное напряжение 220–250 В, частотой 50 Гц, мощность светильника 40 Вт.

4.2 Расчет воздухообмена в рабочем помещении

Рассмотрим расчет воздухообмена, необходимый для удаления избыточного тепла и очистки воздуха от вредных паров.

Потребный воздухообмен в помещении определяется по формуле

$$Q = \frac{g}{X - X_n} \qquad (4.1)$$

где Q – потребный воздухообмен, м^3/ч;

g – количество вредных веществ, л/ч;

X – предельно допустимая концентрация вредных веществ в помещения, л/м^3;

X_n – предельно допустимая концентрация вредных веществ в наружном воздухе, л/м^3.

Кроме того, можно рассчитать кратность воздухообмена n. Величина, показывающая сколько раз в течение одного часа воздух должен полностью смениться

$$n = \frac{Q}{Q_{пот}} \qquad (4.2)$$

где $Q_{пот}$ – объем помещения;

$Q_{пот}$=60 м^3.

Количество углекислоты, выделяемой человеком при легком труде, равняется 23 л/ч, предельно допустимая концентрация в воздухе помещения 1 л/м^3, предельно допустимая концентрация в наружном воздухе 0,5 л/м^3.

Определим потребный воздухообмен по формуле (4.1) при одном рабочем

$$Q = \frac{23}{1 - 0{,}5} = 46 \text{ (м}^3\text{/ч)}$$

Потребная кратность воздухообмена по формуле (4.2)

n = 46 / 60 = 0,77 (1/ч)

Расчет воздухообмена для удаления избыточного тепла производится по формуле

$$Q = \frac{L_{ИЗБ}}{G_B \cdot C_B \cdot dt} \qquad (4.3)$$

где $L_{изб}$ – избыточное тепло, ккал/ч;

G_B = 1,206 кг/м³ – удельная масса приточного воздуха;

C_B = 0,24 ккал/кг·°С – теплоемкость воздуха;

dt – разность температур удаленного и приточного воздуха.

Величина *dt* при расчетах выбирается в зависимости от теплонапряжённости воздуха – L_H

$$L_H = \frac{L_{ИЗБ}}{Q_{ПОТ}} \quad (4.4)$$

Если L_H больше 20 ккал/ч, то *dt*=8°С.

Если L_H меньше 20 ккал/ч, то *dt*=6°С.

Количество избыточного тепла определим по формуле

$$L_{изб}=L_{об}+L_{ос}+L_л+L_р+L_{отд} \quad (4.5)$$

где $L_{об}$ – тепло от оборудования, ккал/ч;

$L_{ос}$ – тепло от системы освещения, ккал/ч;

$L_л$ – тепло, выделяемое людьми, ккал/ч;

$L_р$ – тепло от солнечной радиации, ккал/ч;

$L_{отд}$ – теплоотдача естественным путем, ккал/ч.

Тепло от оборудования найдём по формуле

$$L_{об}=860 \cdot P_{об} \cdot f \quad (4.6)$$

где $P_{об}$ – номинальная мощность оборудования, Вт;

f – коэффициент передачи, *f* = 0,25.

$L_{об}$ = 860 ·1 ·0,25=215 (ккал/ч)

$$L_{ос}=860 \cdot P_{ос} \cdot a \cdot b \cdot cos(f) \quad (4.7)$$

где $P_{ос}$ – номинальная мощность освещения, кВт;

a – коэффициент перевода электрической энергии в световую, *a*=0,46;

b – коэффициент одновременной работы ламп, *b*=1;

cos(f) – коэффициент мощности, *cos(f)*= 0,3.

$L_{ос}$=860 · (0,04 ·3) ·0.46 ·1 ·0.3=14,24 ккал/ч

$$L_л=n \cdot g \quad (4.8)$$

где *n* – количество человек;

g – тепловыделение одного человека, *g*=50 ккал/ч

$L_л$ =1 ·50=50 (ккал/ч)

$$L_р=m \cdot F \cdot D_{ос} \quad (4.9)$$

где m – количество окон;

F – площадь окна, м^2;

$D_{ос}$ – солнечная радиация, $D_{ос}$=65 ккал/ч.

$L_р$=1 ·4 ·65 = 260 (ккал/ч)

Теплоотдачу естественным путем можно приравнять к $L_р$ в зимнее время года и считать равной нулю в летнее, тогда по формуле (4.6)

$L_{изб}$ = 215+14,24+50+260 = 539,24 (ккал/ч)

Теплонапряженность воздуха определяется по формуле (4.5)

$L_н$ = 539,24 /46=11,72 (ккал/ч), что меньше 20, тогда dt=6°С

Подставляя полученное значение в формулу (4.1), получаем значение потребного воздухообмена для удаления избыточного тепла

Q = 539,24 /(1,206 · 0,24 · 6)=310,5 (м^3/ч)

Таким образом, для удаления избыточного тепла и очистки воздуха от вредных паров следует применять систему вентиляции, которая обеспечивает требуемую подачу воздуха Q = 310,5 м^3/ч.

4.3 Расчет естественного освещения

Одним из наиболее важных гигиенических показателей рабочего места является освещенность помещения.

Рабочая зона или рабочее место освещается в такой степени, чтобы можно было хорошо видеть процесс работы, не напрягая зрения, и исключалось прямое попадание лучей источника света в глаза. По нормам освещенности СНиП 11–4–79 [11] и отраслевым нормам, работа инженера относится к четвертому разряду зрительной работы.

Основной задачей светотехнических расчетов является определение требуемой площади световых проёмов при естественном освещении и потребляемой мощности осветительных приборов при искусственном.

Требуемую площадь светового проема при боковом естественном освещении определяют по формуле

$$S = \frac{Sn \cdot en \cdot K_3 \cdot h_0 \cdot K_{30}}{100 \cdot t_0 \cdot r_1} \qquad (4.10)$$

где S_n – площадь помещения, м^2;

e_n – нормированное значение КЕО, %;

$K_з$ – коэффициент запаса, принимаемый из таблиц;

h_o – световая характеристика окон (6,5 – 29);

$K_{зо}$ – коэффициент, учитывающий затемнение окон противостоящими зданиями (1,0 – 1,7);

r_1 – коэффициент, учитывающий повышение КЕО за счет отраженного света от поверхности помещения (1,05 – 1,7);

t_o – общий коэффициент светопропускания, определяемый из СНиП 11–4–79 (0,1 – 0,8).

Коэффициент $K_з$ определяется: $K_з = 1{,}5$.

Учитывая, что длина пола помещения равна 4 м, а ширина равна 5 м, находим площадь пола

$$S_n = 4 \cdot 5 = 20 \text{ (м}^2\text{)}$$

Нормированное КЕО определяется: $e_n = 1{,}5$

Значение остальных коэффициентов примем равными следующим

$$h_o = 29;\ K_{зо} = 1;\ r_1 = 1{,}05;\ t_o = 0{,}7$$

Итак, при подсчете получим следующее значение требуемой площади

$$S = (20 \cdot 1{,}5 \cdot 1{,}5 \cdot 29 \cdot 1)/100 \cdot 0{,}7 \cdot 1{,}05 = 10{,}65 \text{ (м}^2\text{)}$$

Учитывая, что в помещении площадь оконного проема составляет около 4 м^2, применение одного бокового освещения недостаточно для данного помещения.

4.4 Расчет искусственного освещения

Целью данного расчета является проверка соответствия освещенности помещения, норме освещенности согласно СНиП 11–4–79 для персонала, осуществляющего эксплуатацию ЭВМ. Согласно этой норме для четвертого класса зрительных работ освещенность должна быть не менее 200 Лк. Проведем проверочный расчет освещенности.

Определим световой поток, который необходим для создания освещения в лаборатории на рабочем месте по формуле

$$E = \frac{\Phi_{СВ} \cdot n \cdot N \cdot I}{K_З \cdot s \cdot z} \qquad (4.11)$$

где E – номинальная освещенность рабочего места;

$\Phi_{св}$ – световой поток от ламп, лк;

N – количество светильников;

$K_з$ – коэффициент запаса, учитывающий запыленность и износ светильников;

n – коэффициент использования светильников;

s – площадь помещения, м^2;

z – коэффициент неравномерности освещения.

Согласно СНиП 11–4–79 для используемого типа ламп в ЛАЗ КДП (светильник типа УСП–35) [11]:

– $K_з$ =1,4÷1,5 при нормальной эксплуатации светильников;

– z =1,1÷1,2 при оптимальном размещении светильников.

Коэффициент n зависит от типа светильника, коэффициентов отражения светового потока от стен – p_1, потолка – p_2, пола – p_3, которые в свою очередь зависят от геометрических размеров помещения, учитывающихся величиной I – индекс помещения

$$I = \frac{A \cdot B}{h_C \cdot (A + B)} \qquad (4.12)$$

где A – ширина, м, A =5 м;

B – длина, м, B=4 м;

h_c – высота светильников над рабочей поверхностью, h_c=3 м

I = (3 ·4) / (2,3 ·(3 + 4)) = 0,75

По таблице 4.1 определим коэффициент использования светового потока n с учетом Рп = 50%, Рс = 30%, Рпола = 10%

Т а б л и ц а 4.1 – Значения коэффициента использования светового потока в зависимости от показателя помещения:

Показатель помещения, I	0,5	1	2	3	4
Коэффициент использования светового потока, h	0,22	0,36	0,48	0,54	0,59

Коэффициент n = 0,3.

Световой поток от лампы типа ЛБ–40 равняется 3120 Лк, световой поток от трех ламп светильника равняется 9360 Лк. Определим номинальную освещенность рабочего места по формуле (4.10)

$$E = \frac{9360 \cdot 0{,}3 \cdot 3 \cdot 0{,}75}{1{,}4 \cdot 12 \cdot 1{,}2} = 313{,}39 \text{ Лк}$$

Полученное значение соответствует условиям нормальной работы.

Вывод

В этом разделе был произведен анализ условий труда в данном рабочем помещении, в частности расчет естественного и искусственного освещения и расчет воздухообмена.

Расчет показал, что для естественного освещения не достаточно одного окна площадью 4 $м^2$.

Также для освещения рабочего места вполне достаточно общего освещения, при котором рабочее место освещает один светильник, в светильнике имееются две лампы со световым потоком излучения 3120 лкм каждая, поэтому в этом помещении можно работать и в темное время суток.

В результате проделанного расчета, мы убедились, что требования, предъявляемые СНиП II–68–75, обеспечивают все нормируемые параметры микроклимата в помещении для эффективной организации работы.

5 Технико–экономическое обоснование

5.1 Описание работы и обоснование необходимости

Тема данного дипломного проекта «Разработка информационного сайта с использованием технологии ASP.NET».

Основная цель – разработка сайта, предоставляющего пользователям корректную информацию о городе Зайсан.

Главная задача сайта – осуществление помощи пользователям при поиске необходимой информации о городе.

На сегодняшний день, не существует других аналогичных сайтов с информацией о городе. В социальных сетях имеются различные страницы, так же существует государственный сайт, но для получения большей информации этого не достаточно. Исходя из этого можно полагать, что сайт будет интересным как для жителей, так и для других пользователей.

В данном разделе приводится рассмотрение экономической составляющей реализации данного проекта, отражающей временные, трудовые и финансовые затраты на проект.

5.2 Трудовые ресурсы, используемые в работе

План разработки программного продукта:

– для того, чтобы определить затраты на разработку и внедрение, необходимо составить план проведения работы и смету затрат;

– жизненным циклом программы считается весь цикл от принятия решения о проведении разработок до его реализации.

Разработка и внедрение программного продукта состоит из 4 основных этапов:

– этап моделирования;
– этап написания кода – программирование;
– этап тестирования;
– этап внедрения.

Моделирование

Составление модели программного обеспечения. Этап включает в себя разработку программного обеспечения: определение алгоритмов, основных компонентов и оценку системных. Длительность этапа: 5 рабочих дней.

Программирование

На данном этапе будет осуществляться описание разработанной на первом этапе модели с помощью алгоритмических языков программирования. Участники этапа: web–программист. Длительность этапа: 15 рабочих дней.

Тестирование ПО

На данном этапе будет производиться тестирование разработанного ПО, выявление ошибок реализации. Длительность этапа: 3 рабочих дня.

Внедрение ПО

Внедрение разработанного программного обеспечения в сеть Интернет. Длительность этапа: 1 рабочий день.Сводные данные по плану проведения работ по разработке программного продукта представлены в таблице 5.1.

Т а б л и ц а 5.1 – План проведения работ по разработке ПО

Наименование этапов и содержание работ	Исполнитель	Количество исполните–лей	Длительность цикла, дн.	Норма –часы
Все этапы	web–разработчик	1	24 (20.04.14–22.05.14)	192

5.3 Программное обеспечение, используемое в работе

Стоимость использованного программного обеспечения при разработке сайта предоставлено в таблице 5.2.

Т а б л и ц а 5.2 – Перечень использованного программного обеспечения

№	Программное обеспечение	Стоимость, тенге
1	Хостинг + доменное имя	8 500
3	Microsoft Visual Studio 2010	80 500
Итого		89 000

Цены на программное обеспечение приведены без учета НДС.

5.4 Расчет стоимости работы по проектированию и разработке

Расчет полных затрат на разработку проектного решения включает следующие статьи расходов:

- заработную плату исполнителей;
- отчисления на социальный налог;
- амортизационные отчисления;
- расходы на электроэнергию;
- накладные расходы.

Таким образом, себестоимость разработки проекта определяется по следующей формуле

$$С = ФОТ + Сн + А + Сэ + Н \quad (5.1)$$

где ФОТ – фонд оплаты труда;
Сн – социальный налог;
А – амортизационные отчисления;
Сэл – расходы на электроэнергию;
Н – накладные расходы.

5.4.1 Расчет затрат на оплату труда

В процессе разработки данного программного обеспечения участвует 1 человек – web–разработчик.

Месячная заработная плата web–разработчика – 125 000 тенге.

Дневной заработок рассчитываем по формуле [1]

$$D = \frac{ЗПм}{Др} \tag{5.2}$$

где $ЗПм$ – ежемесячный размер заработной платы;
$Др$ – количество рабочих дней в месяце (21 рабочих дня).

Заработная плата web–разработчика за один день составляет

$$D = \frac{125000}{21} = 5952{,}4 \text{ тенге/час}$$

Сводные данные расчета заработной платы представлены в таблице 5.3.

Таблица 5.3 – Расчет основной заработной платы персонала.

Наименование содержания работ	Исполнитель	Трудоёмкость, дней	Зар. плата за день (тг/дн)	Сумма заработной платы (тг.)
Моделирование Программирование Тестирование Внедрение	Web–разработчик	24	5952,4	142 857
Итого (ФОТ)				142 857

5.4.2 Расчет затрат по социальному налогу

Согласно Налоговому Кодексу РК, социальный налог составляет 11 % [3] от дохода работника и рассчитывается по формуле [1]

$$Сн = (ФОТ–ПО)\cdot 11\% \quad (5.3)$$

где ФОТ – фонд оплаты труда.

ПО – пенсионные отчисления, которые составляют 10% от ФОТ и социальным налогом не облагаются, рассчитываются по формуле

$$ПО=ФОТ\cdot 10\% \quad (5.4)$$

ПО=142 857·0,1=14 285,7

Таким образом, отчисления на социальный налог составляют

Сн=(142 857–14 285,7)·0,11=14 142,8 тенге

5.4.3 Расчет амортизационных отчислений

Амортизационные отчисления вычисляются по формуле [2]

$$A = \frac{H_A \cdot C_{пер} \cdot N}{100 \cdot 365} \quad (5.5)$$

где H_A – норма амортизации;

$C_{пер}$ – первоначальная стоимость оборудования;

N – количество дней на выполнене работ.

Норма амортизации H_A на компьютерную технику составляет 40% от стоимости всего оборудования [2].

Амортизационные отчисления за весь период разработки ПО составляют

$$A=\frac{40\cdot 89000\cdot 24}{100\cdot 365}=2340,8 \text{ тенге}$$

5.4.4 Расчет затрат на электроэнергию

Поскольку в процессе производства используется электрооборудование, важной статьей затрат являются затраты на потребляемую электроэнергию. Затраты на электроэнергии на оборудование и дополнительные нужды рассчитывается по следующей формуле [2]

$$C_{эл} = З_{об} + З_{доп..нуж.} \quad (5.6)$$

где $З_{об.}$ – затраты на электроэнергию оборудования;

$З_{доп.нуж}$ – затраты электроэнергии на дополнительные нужды.

$$З_{об.} = W \cdot T \cdot S \cdot К_{исп} \quad (5.7)$$

где W– потребляемая мощность, кВт;

T – количество часов работы;

S – стоимость киловатт–часа электроэнергии;

$К_{исп}$ – коэффициент использования ($К_{исп}$=0,8).

Виды используемого оборудования, а так же потребляемая ими мощность представлены в таблице 5.4.

Т а б л и ц а 5.4 – Потребляемая мощность оборудования

№	Наименование	W, Вт	$К_{исп}$	T, час
1	Монитор Samsung SyncMaster LS24	40		
2	Процессор	450		
3	Модем	50	0,8	192
	Итого:	540	0,8	192

Согласно установленному тарифу по энергопотреблению пользователей для I–го уровня стоимость 1 кВт составляет 14,36 тенге с НДС [6].

С учетом длительности восьмичасового рабочего дня и длительности разработки, внедрения проекта, количество часов работы составило

T=8·24=192 часа

В соответствии с формулой (5.7) расходы на электроэнергию составят

$З_{об.}$=0,54·192·14,36·0,8=1191,1 тенге

Затраты на дополнительные нужды берутся по укрупненному показателю в размере 5% от затрат на оборудование и составляют

$З_{доп.нуж.}$=1191,1·0,05=59,5 тенге

Таким образом суммарные затраты на электроэнергию составляют

$С_{эл}$=1191,1+59,5=1250,65 тенге

Это было вычислено при помощи формулы (5.6), которая была предоставлена выше.

5.4.5 Расчет затрат на накладные расходы

Накладные расходы на разработку проекта составляют от 50% от общей суммы понесенных затрат и рассчитываются по формуле [2]

$$Н = (ФОТ + O_c + А + Сэл) \cdot 0,5 \quad (5.8)$$

Тогда, согласно формуле (5.8), накладные расходы будут равны

$$Н = (142857 + 14142,8 + 2340,8 + 1250,65) \cdot 0,5 = 78499,9 \text{ тенге}$$

5.3.6 Расчет стоимости по всем статьям затрат

В соответствии с произведенными расчетами по статьям затрат себестоимость проекта, согласно формуле 5.1, будет равна

$$C = 142857 + 14142,8 + 2340,8 + 1250,65 + 78499,9 + 89000 = 328091 \text{ тенге.}$$

Сводные результаты расчета затрат по разработке сайта и их структура представлены в таблице 5.5 и на рисунке 5.1.

Таблица 5.5 – Стоимость разработки информационного сайта

Наименование статей затрат	Сумма, тенге
ФОТ	142 857
Отчисления на социальные нужды	14142,8
Амортизационные отчисления	2 340,8
Затраты на электроэнергию	1 250,65
Программное обеспечение	89 000
Накладные расходы	78 499,9
Итого	328 091

Рисунок 5.1 – Структура затрат по разработке информационного сайта г. Зайсан

5.5 Цена интеллектуального труда

Цена реализации программного продукта складывается из себестоимости и чистого дохода, и вычисляется по формуле

$$Ц = С + П \tag{5.9}$$

где С – себестоимость продукта;
П – чистый доход.

Первоначальная цена рассчитывается через рентабельность проекта. Учитывая, что желаемый уровень рентабельности для отрасли телекоммуникации составляет 20%, применим следующую формулу

$$Ц_{П} = С \cdot \left(1 + \frac{Р}{100}\right) \tag{5.10}$$

где Р – рентабельность (20%).

Согласно формуле (5.10) первоначальная цена равна

$$Ц_{П} = 328\,091 \cdot \left(1 + \frac{20}{100}\right) = 393709{,}38 \text{ тенге}$$

Цена реализации готовой продукции рассчитывается по формуле

$$Ц_{Р} = Ц_{П} + НДС \tag{5.11}$$

Поскольку на сегодняшний день размер НДС в РК составляет 12%, следовательно

$$НДС = \frac{12}{100} \cdot Ц_{П} \quad (5.12)$$

НДС = $\frac{12}{100} \cdot 393709{,}38 = 47245{,}12$ тенге

Тогда согласно формуле 5.12 цена реализации составит

$Ц_{р}$ = 393709,38 + 47245,12 = 440 954,5тенге

Вывод

Разработка инфорамционного сайта – сложный и трудоемкий процесс, требующий наряду с интеллектуальными, техническими затратами и финансовых затрат.

Себестоимость разработки сайта составила 328 091 тенге, наибольшую долю в себестоимости составляют затраты на оплату труда 142 857 тенге (43%). Цена реализации составила 440 954 тенге. Средняя цена на рынке на аналогичные продукты составляет от 100 000 до 500 000 тенге. Анализируя полученные данные, можно сделать вывод, что цена на продукт находится в допустимых пределах.

Заключение

По статистике количество пользователей всемирной сети постоянно увеличивается. На сегодняшний день эта цифра составляет около 38,4%.

Город Зайсан является административным центром Зайсанского района. Население района составляет около 40 000 человек. Инфраструктура района развивается и имеет огромные потенциалы. Тем самым обращая внимание на город и его окрестности, становясь привлекательным местом для туристов и предпринимателей со всего Казахстана, и не только.

Для достижений поставленных целей и решения задач, были выполнены следующие мероприятия:

– была проанализирована предметная область проекта;

– .досконально изучены программные средства для разработки сайтов;

– .были изучены и сопоставлены серверные технологии, такие как ASP.net и PHP;

– .выбраны наиболее подходящие и актуальные программные средства и инструменты при разработке сайта;

Практическая ценность проекта заключается в том, что:

– была изучена такая непростая технология как asp.net;

– так же был получен опыт в разработке сайта на asp.net и реализации использования xml–файлов, вместо реляционной БД;

– так же был запущен проект, который будет использоваться по назначению.

В ходе выполнения дипломного проекта был создан полнофункциональный веб–сайт, полностью готовый к применению.

Данный сайт ориентирован для уведомления посетителей о новостях, связанных с городом и самим районом. С его помощью пользователи смогут получать необходимую информацию о проводимых мероприятиях, актуальных афишах и предстоящих действиях, связанных с городом.

В ходе разработке сайта были проанализированы современные технологии, которые позволяют создавать динамические веб–страницы.

В технико–экономическом разделе были определены затраты на разработку информационного сайта. Себестоимость продукта составила 328 091, а цена реализации продукта – 440 954 тенге.

С использованием обозначенных формул и уравнений, а также информации из трудового кодекса республики Казахстан, были вычислены все неизбежные и предполагаемые затраты, которых стоит ожидать при приведении в жизнь задуманного проекта. В результате расчета затраты на создание данного программного продукта составили 328 091тенге.

Стоит отметить, что определение цены и затрат проекта, предназначены только для сравнения с ценами на рынке на подобные товары. Поскольку проект является личным вкладом в общество.

В разделе безопасности жизнедеятельности были проанализированы условии труда в данном рабочем помещении, а также был выполнен расчет составляющих микроклимата для проверки их текущего состояния. Условия труда признаны допустимыми, а данные, полученные в результате проведенных расчетов, полностью удовлетворяют требованиям в стандартах безопасности жизнедеятельности.

Расчёты показали, что одно окно размером 1,5х1,2м не соответствует нормативам естественного освещения рабочего помещения. Для обеспечения необходимой освещенности в дневное время суток используется искусственное освещение.

По результатам расчета, чтобы обеспечивать расход воздуха L=145,9 $м^3$/ч, можно использовать 1 кондиционер фирмы Samsung серии HA 85 с максимальным расходом воздуха 195 $м^3$/ч, модель R22.

Разработанный сайт удовлетворяет всем требованиям, поставленным на этапе постановки задачи. При разработке сайта были использованы различные инструменты, такие как JavaScript, HTML, CSS, jQuery.

Сайт будет функционировать исправно, его содержимае будет постоянно обновляться. В дальнейшем возможна доработка сайта, а именно создания системы ее управления, для упрощения написании статей.

Список литературы

1 Горнаков С.Г. Осваиваем популярные системы управления сайтом. – М.: Наука, 2009.

2 Ганеев Р.М. Проектирование интерактивных WEB–приложений. – М.: Горячая Линия – Телеком, 2001. – 272 с.

3 Кузнецов М.В, Симдянов И.В. PHP на примерах. – 2–е изд., – СПб.: БХВ–Петербург, 2011. – 505 с.

4 Косентино К. PHP. Web – профессионалам. – М.: Издательская группа BHV, 2001.

5 Ли Дж., Уэр Б. Использование Linux, Apache, MySQL и PHP для разработки Web–приложений. – М.: Вильямс, 2010. – 432 с.

6 Гончаров А.Н. Самоучитель HTML. – СПб.: Питер, 2002.

7 Норт Б. Joomla.: Практическое руководство. – М.: Символ–плюс, 2008.

8 Рамел Д. Самоучитель Joomla. – СПб.: Питер, 2008.

9 А. В. Белозубов, Д. Г. Николаев. Основы работы с HTML–редактором Abobe Dreamweaver CS3. – СПб.: СПбГУ ИТМО, 2007. – 112 с.

10 Сайт www.adilet.kz.

11 Экономика от А до Я: Тематический справочник/ Г.М. Гукасьян. – М.: ИНФРА–М, 2009. – 480 с.

12 Экономика промышленного предприятия: Учебник / И.Н. Иванов. – М.: ИНФРА–М, 2011. – 395 с.

13 Экономика труда: Учебное пособие / Ю.М. Остапенко. – М.: ИЦ РИОР, 2010. – 160 с.

14 Экономика труда: Учебник / А.И. Рофе. – М.: КиноРус, 2010. – 400 с.

15 Сайт www.coolreferat.com.

16 Дюсебаев М.К., Бегимбетова А.С. Методические указания к выпускной работе (для студентов всех форм обучения специальностей 050719 – Радиотехника электроника и телекоммуникации, 050704 – Вычислительная техника и программное обеспечение). – Алматы: АИЭС, 2008. – 10 с.

17 ГОСТ 4.02–05–2001: Отопление, вентиляция и кондиционирование. – М.: Изд–во стандартов, 2001.

18 Абдимуратов Ж.С, Мананбаева С.Е. Безопасность жизнедеятельности. Методические указания к выполнению раздела «Расчет производственного освещения» в выпускных работах для всех специальностей. Бакалавриат. – Алматы: АИЭС, 2009.

Приложение А

Листинг сайта

headmaster.Master

```
<%@ Master Language="C#" AutoEventWireup="true"
CodeBehind="headMaster.Master.cs" Inherits="Diplom1.Site1"  %>
<!DOCTYPE html>
 <html xmlns="http://www.w3.org/1999/xhtml">
    <head runat="server" id="head1">

         <link href="style/style.css" rel="stylesheet"
type="text/css" media="screen" />
        <script src="script/jquery/jquery-1.11.0.min.js"></script>
        <script type="text/javascript" src="/script/js/jquery-
1.5.min.js"></script>
<script type="text/javascript"
src="/script/js/jquery.jcarousel.js"></script>
        <script type="text/javascript">
    jQuery(document).ready(function() {
        // Initialise the first and second carousel by class
selector.
        // Note that they use both the same configuration options
(none in this case).
        jQuery('.d-carousel .carousel').jcarousel({
            scroll: 1
        });
    });      </script>
        <script type="text/javascript"
src="https://maps.googleapis.com/maps/api/js?sensor=false"></scrip
t>
        <asp:ContentPlaceHolder ID="head_ContentPlaceHolder"
runat="server">
        </asp:ContentPlaceHolder>
    </head>
<body>
    <div id="header">
        <div id="menu">
           <div id="logo">

            </div>
            <div  style="float: right;text-align:right;">

                <ul id="menu_main_block_1">
                    <li ><a href="index.aspx">Главная</a></li>
                    <li ><a
href="article.aspx?page=0">Статьи</a></li>
                    <li id="aspaptar"><a href="history.aspx">г.
Зайсан</a>
                        <ul id="sub-menu">
```

```
                                <li><a
href="history.aspx">История</a></li>
                                <li><a
href="places.aspx">Места</a></li>
                                <li><a
href="nature.aspx">Природа</a></li>
                            </ul>
                        </li>
                        <li><a href="gallery.aspx">Галерея</a></li>
                        <li><a href="contact.aspx">Контакты</a></li>
                    </ul>

                    <div style="margin-top:45px">
                        <form method="get" action="search.aspx" >
                            <div>
                                <input type="text" name="s"
id="search-text" placeholder="Поиск..." value="" />
                            </div>
                        </form>
                    </div>
                </div>
            </div>
        </div>

        <div id="content" style="background:#efeeef;">
            <section style="margin:0 auto; width:1020px; background-
color:white; min-height:600px;">
                <div style="padding: 30px;" id="pagecontent">
                    <asp:ContentPlaceHolder ID="ContentPlaceHolder1"
runat="server">

                    </asp:ContentPlaceHolder>
                </div>
            </section>
        </div>
        <!-- end #page -->
        <div id="footer-menu">
            <ul>
                <li ><a href="index.aspx">Главная</a></li>
                <li ><a href="article.aspx?page=0">Статьи</a></li>
                <li ><a href="history.aspx">г. Зайсан</a>
                </li>
                <li><a href="gallery.aspx">Галерея</a></li>
                <li><a href="contact.aspx">Контакты</a></li>
            </ul>
                <p>© 2014г. Все права защищены.</p>
        </div>
    </body>
     <asp:contentplaceholder id="bottom_ContentPlaceHolder"
    runat="server">
```

```
 </asp:contentplaceholder>
</html>
```

Article.aspx

```
<%@ Page Title="Статьи" Language="C#"
MasterPageFile="~/headMaster.Master" AutoEventWireup="true"
CodeBehind="article.aspx.cs" Inherits="Diplom1.article" %>
<asp:Content ID="Content1"
ContentPlaceHolderID="head_ContentPlaceHolder" runat="server">

    <!-- Core CSS File. The CSS code needed to make eventCalendar
works -->
     <link rel="stylesheet"
href="script/eventCalendar_v054/css/eventCalendar.css" />

     <!-- Theme CSS file: it makes eventCalendar nicer -->
     <link rel="stylesheet"
href="script/eventCalendar_v054/css/eventCalendar_theme_responsive
.css" />
    <link rel="stylesheet" href="style/catalog.css" />
</asp:Content>
<asp:Content ID="Content2"
ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
     <a href="article.aspx">Статьи</a><br/><br/>
     <div style="float:left; width:750px">

         <!-- Статьи -->
         <table id="catalog_table">
         <%
             int page = 0;
             try {
                 page =
Convert.ToInt16(Request.QueryString["page"].ToString());
             }
             catch (Exception e) { }

             Diplom1.ArticleData data = new
Diplom1.ArticleData(page);
             ArrayList articles = data.getArticles();
             foreach (Diplom1.Article a in articles)


             {%>
                <tr>
                    <td>
                        <div class="title"
onclick="javascript:window.location='articlebyid.aspx?id=<%=a.id
%>'"><%= a.topic %></div>
                        <div class="content">
```

```
                                <div style="padding: 6px 0"> 
<img src="images/Calendar-icon-red.png" /> <%= a.date %></div>
                                  <%= a.content %>...
                                 <a
href="articlebyid.aspx?id=<%=a.id %>">Подробнее</a>
                            </div>
                        </td>
                    </tr>
                <%}%>
            </table>
        </div>
        <div style="float:right; width:200px">
            <div id="eventCalendarHumanDate"></div>
            <script>
                $(document).ready(function () {
                    $("#eventCalendarHumanDate").eventCalendar({
                    });
                });
             </script>
        </div>
       <div>   </div>
       <br style="clear:both"/>
</asp:Content>
<asp:Content ContentPlaceHolderID="bottom_ContentPlaceHolder"
runat="server" ID="content3">
    <script
src="script/eventCalendar_v054/js/jquery.eventCalendar.js"
type="text/javascript"></script>
</asp:Content>
```

articlebyId.aspx

```
<%@ Page Title="Статьи" Language="C#"
MasterPageFile="~/headMaster.Master" AutoEventWireup="true"
CodeBehind="articlebyid.aspx.cs" Inherits="Diplom1.article" %>
<asp:Content ID="Content1"
ContentPlaceHolderID="head_ContentPlaceHolder" runat="server">

    <!-- Core CSS File. The CSS code needed to make eventCalendar
works -->
     <link rel="stylesheet"
href="script/eventCalendar_v054/css/eventCalendar.css" />

     <!-- Theme CSS file: it makes eventCalendar nicer -->
     <link rel="stylesheet"
href="script/eventCalendar_v054/css/eventCalendar_theme_responsive
.css" />
    <link rel="stylesheet" href="style/catalog.css" />
</asp:Content>
```

```
<asp:Content ID="Content2"
ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

    <%
        string id = "1";
        try {
            id = Request.QueryString["id"].ToString();
        }
        catch (Exception e) { }

        Diplom1.ArticleData data = new Diplom1.ArticleData(0);
        Diplom1.Article article = data.getArticleById(id);

    %>


     <%if(article != null){%>
        <div style="width:90%"><a href="article.aspx">Статьи</a> >
<a href="articlebyid.aspx?id=<%=article.id %>" ><%=article.topic
%></a></div>
         <!-- Статьи -->
         <div style="float:left; width:730px; padding-top:20px;
line-height:1.3em; text-wrap:avoid">
            <img src="content/article/images/<%=article.img %>"
width="300" style="float:left; margin-right: 15px"/>
            <%=article.content %>
         </div>
     <%}else{%>

        <div style="float:left; width:750px;">
            <a href="article.aspx">Статьи</a> > <a
href="article.aspx">Назад</a>
            <br/><br/><br/><span style=" font-size:24px">Статья не
найдена</span>
        </div>
     <%} %>
     <div style="float:right; width:200px">
         <div id="eventCalendarHumanDate"></div>
         <script>
             $(document).ready(function () {
                 $("#eventCalendarHumanDate").eventCalendar({
                 });
             });
          </script>
     </div>
    <div>   </div>
    <br style="clear:both"/>
</asp:Content>
<asp:Content ContentPlaceHolderID="bottom_ContentPlaceHolder"
runat="server" ID="content3">
```

```
    <script
src="script/eventCalendar_v054/js/jquery.eventCalendar.js"
type="text/javascript"></script>
</asp:Content>


Contacts.aspx
<%@ Page Title="Контакты" Language="C#"
MasterPageFile="~/headMaster.Master" AutoEventWireup="true"
CodeBehind="contact.aspx.cs" Inherits="Diplom1.index" %>

<asp:Content ID="Content2"
ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
       <a href="contact.aspx">Контакты</a><br/><br/>
      <p style="padding-left:20px">Хотите сделать заказ рекламного
текста или обратиться с другим предложением?</p>
      <div style="float:right" id="contact-content">
            <p>Адрес:</p>
            <span>Республика Казахстан, г. Алматы ул. Шашкина
14</span>
            <p>Телефон:</p>
            <span>+7 (702) 629 33 99</span>
            <p>Электронный адрес:</p>
            <span><a href="mailto:dombyra-azat@mail.ru"
style="color:blue">muratova1093@gmail.com</a></span>
            <p>skype:</p>
            <span>aiman.muratova</span>

            <br/>
            <br/>
        </div>

    <script type="text/javascript" src="script/jquery/gmap-
master/jquery.gmap.min.js"></script>
    <div id="map" style="float:left;margin:10px; width:
400px;height:400px" />
    <script>
        $(document).ready(function () {
            var lat = 43.223255;
            var lon = 76.932225;
            $("#map").gMap({
                latitude: "fit",
                longitude: "fit",
                zoom: 15,
                markers: [{ latitude: lat, longitude: lon }]
            });
        })
    </script>

</asp:Content>
```

Gallery.aspx

```
<%@ Page Title="Галерея" Language="C#"
MasterPageFile="~/headMaster.Master" AutoEventWireup="true"
CodeBehind="gallery.aspx.cs" Inherits="Diplom1.gallery" %>

<asp:Content ID="galleryHead"
ContentPlaceHolderID="head_ContentPlaceHolder" runat="server">
    <link href="script/colorbox/colorbox.css" type="text/css"
rel="stylesheet" />
    <link href="style/gallery.css" type="text/css"
rel="stylesheet" />
    <script src="script/colorbox/jquery.colorbox.js"></script>
     <script >
         jQuery(document).ready(function ($) {
             $(".gallery_img").colorbox({ rel: 'gallery_img',
width: '50%' });
         });
    </script>
</asp:Content>
<asp:Content ID="Content1"
ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
     <a href="gallery.aspx">Галерея</a>
    <br/>
     <table id="gallery-table">
        <tr>
            <td>
                <img class="gallery_img"
src="content/gallery/Акимат.jpg" title="Акимат" />
                <div class="title">Акимат</div>
            </td>
            <td>
                <img class="gallery_img"
src="content/gallery/Байтерек на площади.jpg" title="Байтерек на
площади" />
                <div class="title">Байтерек на площади</div>
            </td>
            <td>
                <img class="gallery_img"  src="content/gallery/Вид
с гор.jpg" title="Вид с гор" />
                <div class="title">Вид с гор</div>
            </td>
        </tr>
        <tr>
            <td>
                <img class="gallery_img"  src="content/gallery/Вид
с площади.jpg" title="Вид с площади" />
                <div class="title">Вид с площади</div>
            </td>
            <td>
```

```
<img class="gallery_img"  src="content/gallery/въезд в город.jpg"
title="Въезд в город" />
   <div class="title">Въезд в город</div>
            </td>
             <td>
                 <img class="gallery_img"
src="content/gallery/Гостиница.jpg" title="Гостиница" />
                 <div class="title">Гостиница</div>
             </td>
        </tr>
        <tr>
             <td>
                 <img class="gallery_img"
src="content/gallery/Жеменейка.jpg" title="Жеменейка" />
                 <div class="title">Жеменейка</div>
             </td>
              <td>
                 <img class="gallery_img"
src="content/gallery/Жулдыз Мерамханасы.jpg" title="Жулдыз
Мерамханасы" />
                 <div class="title">Жулдыз Мерамханасы</div>
             </td>
              <td>
                 <img class="gallery_img"
src="content/gallery/казпочта.jpg" title="Казпочта" />
                 <div class="title">Казпочта</div>
             </td>
        </tr>
        <tr>
             <td>
                 <img class="gallery_img"
src="content/gallery/Музей.jpg" title="Музей" />
                 <div class="title">Музей</div>
             </td>
              <td>
                 <img class="gallery_img"
src="content/gallery/Памятник у аллеи.jpg" title="Памятник у
аллеи" />
                 <div class="title">Памятник у аллеи</div>
             </td>
              <td>
                 <img class="gallery_img"
src="content/gallery/Памятник.jpg" title="Памятник" />
                 <div class="title">Памятник</div>
             </td>
        </tr>
        <tr>
             <td>
                 <img class="gallery_img"
src="content/gallery/Парк Абая зима.jpg" title="Парк Абая зима" />
```

```
                <div class="title">Парк Абая зима</div>
            </td>
             <td>
                <img class="gallery_img"
src="content/gallery/Платина (2).JPG" title="Платина" />
                <div class="title">Платина</div>
            </td>
             <td>
                <img class="gallery_img"
src="content/gallery/природа (2).JPG" title="Природа" />
                <div class="title">Природа</div>
            </td>
        </tr>
        <tr>
            <td>
                <img class="gallery_img"
src="content/gallery/природа (6).JPG" title="Природа" />
                <div class="title">Природа</div>
            </td>
            <td>
                <img class="gallery_img"
src="content/gallery/природа (9).JPG" title="Природа" />
                <div class="title">Природа</div>
            </td>
            <td>
                <img class="gallery_img"
src="content/gallery/Спорт школа.jpg" title="Спорт школа" />
                <div class="title">Спорт школа</div>
            </td>
        </tr>
         <tr>
            <td>
                <img class="gallery_img"
src="content/gallery/дост (3).jpg" title="Памятники" />
                <div class="title">Памятники</div>
            </td>
             <td>
                <img class="gallery_img"
src="content/gallery/дост (5).jpg" title="Памятники" />
                <div class="title">Памятники</div>
            </td>
             <td>
                <img class="gallery_img"
src="content/gallery/дост2 (2).jpg" title="Памятники" />
                <div class="title">Памятники</div>
            </td>
        </tr>
    </table>
</asp:Content>
```

Search.aspx

```
<%@ Page Title="" Language="C#"
MasterPageFile="~/headMaster.Master" AutoEventWireup="true"
CodeBehind="search.aspx.cs" Inherits="Diplom1.article" %>
<asp:Content ID="Content1"
ContentPlaceHolderID="head_ContentPlaceHolder" runat="server">
    <title>Поиск</title>
    <!-- Core CSS File. The CSS code needed to make eventCalendar
works -->
     <link rel="stylesheet"
href="script/eventCalendar_v054/css/eventCalendar.css" />

     <!-- Theme CSS file: it makes eventCalendar nicer -->
     <link rel="stylesheet"
href="script/eventCalendar_v054/css/eventCalendar_theme_responsive
.css" />
    <link rel="stylesheet" href="style/catalog.css" />
</asp:Content>
<asp:Content ID="Content2"
ContentPlaceHolderID="ContentPlaceHolder1" runat="server">

    <%
        string s = "";
        try {
            s = Request.QueryString["s"].ToString();
        }
        catch (Exception e) { }

        Diplom1.ArticleData data = new Diplom1.ArticleData(0);
        ArrayList articles = data.searchArticles(s);
    %>

    <a href="#">Поиск</a> : <a href="#"><%=s %></a> <br/><br/>

     <div style="float:left; width:750px">
         <!-- Статьи -->
         <table id="catalog_table">
         <%
             foreach(Diplom1.Article a in articles){%>
                <tr>
                    <td>
                        <div class="title"
onclick="javascript:window.location='articlebyid.aspx?id=<%=a.id
%>'"><%= a.topic %></div>
                        <div class="content">
                            <div style="padding: 6px 0"> 
<img src="images/Calendar-icon-red.png" /> <%= a.date %></div>
                              <%= a.content %>...
```

```
                                 <a
href="articlebyid.aspx?id=<%=a.id %>">Подробнее</a>
                            </div>
                        </td>
                    </tr>
                <%}%>
            </table>
        </div>
        <div style="float:right; width:200px">
            <div id="eventCalendarHumanDate"></div>
            <script>
                $(document).ready(function () {
                    $("#eventCalendarHumanDate").eventCalendar({
                    });
                });
             </script>
        </div>
       <div>   </div>
       <br style="clear:both"/>
</asp:Content>
<asp:Content ContentPlaceHolderID="bottom_ContentPlaceHolder"
runat="server" ID="content3">
    <script
src="script/eventCalendar_v054/js/jquery.eventCalendar.js"
type="text/javascript"></script>
</asp:Content>
```

Places.aspx

```
<%@ Page Title="Места города" Language="C#"
MasterPageFile="~/headMaster.Master" AutoEventWireup="true"
CodeBehind="places.aspx.cs" Inherits="Diplom1.index" %>

<asp:Content ID="Content2"
ContentPlaceHolderID="ContentPlaceHolder1" runat="server">
    <a href="history.aspx">г. Зайсан</a> > <a
href="places.aspx">Места</a>
   <br/><br/>    <img src="images/бэкап.jpg"
style="height: 290px; width: 936px; text-align: center"  />
<table id="placetable">
        <tr>
            <td>
                Платина<br/><br/><img src="content/places/pl.jpg"
height="250" />
            </td>
            <td>
                Телевышка<br/><br/><img
src="content/places/tele.jpg" height="250" />
            </td>
            <td>
```

```
                Аэродром Зайсан 3 класса, способен принимать
воздушные суда Ан-24, Ан-26
                <br/><img src="content/places/аэропорт.jpg"
height="250" />
            </td>
        </tr>
        <tr>
            <td>
                Уйдененское водохранилище<br/><img
src="content/places/uidene.jpg" height="250" />
            </td>
            <td>
                Парк «Закиев»<br/>
                Адрес: г. Зайсан, ул. Желтоксана
                <br/><img src="content/places/закиева.jpg"
height="250" />
            </td>
            <td>
                Парк развлечений<br/><img src="content/places/В
парке развлечений.jpg" height="250" />
            </td>
        </tr>
         <tr>
            <td>
                Зайсан  мешіті<br/><br/><img
src="content/places/Зайсан  мешіті.jpg" height="250" />
            </td>
            <td>
                Парк «Абай»<br/>
                Адрес: г. Зайсан, ул. Спамбетова
                <br/><img src="content/places/Парк Абая.jpg"
height="250" />
            </td>
            <td>
                Сквер «Тауелсіздіктін 20 жылдыг аланы»<br/>
                Адрес: г. Зайсан, ул. Жангельдина
                <br/><img src="content/places/Аллея
Независимости.jpg" height="250" width="300" />
            </td>
        </tr>
    </table>

    <br />
    <br />
    <br />
</asp:Content>
```

Article.cs

```
using System;
```

```
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace Diplom1
{
    public class Article
    {
        public string topic;
        public string content;
        public string date;
        public string id;
        public string img;

        public Article(string _topic, string _content, string
_date, string _id, string _img) {
            topic = _topic;
            content = _content;
            date = _date;
            id = _id;
            img = _img;


        }
    }
}
```

articleData.cs

```
using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Xml;
using System.Xml.Linq;
using System.Xml.XPath;

namespace Diplom1
{
    public  class ArticleData
    {
        private int page;
        private XmlDocument xmldoc = new XmlDocument();
        private const string filepath =
"content/article/data.xml";
        private XmlNodeList nodeArticles = null;

        public ArticleData(int _page) {
            page = _page;
```

```
            try
            {

xmldoc.Load(System.Web.HttpContext.Current.Server.MapPath(filepath
));
                nodeArticles =
xmldoc.SelectNodes("/articles/article");
            }
            catch (Exception e) { }
        }

        public ArrayList getArticles() {

            ArrayList list = new ArrayList();
            try
            {

                foreach (XmlNode a in nodeArticles)
                {
                    string topic = a["topic"].InnerText;
                    string content = a["content"].InnerText;
                    string date = a["date"].InnerText;
                    string id = a.Attributes[0].Value;
                    string img = a.Attributes[1].Value;

                    Article article = new Article(topic,
                        content.Length > 260 ?
content.Substring(0, 260) : content, date, id, img);

                    list.Add(article);
                }
            }
            catch (Exception e) { }

            return list;
        }

        public int getTotalPageNumber() {
            int nodeCount = nodeArticles.Count;
            return nodeCount % 2 == 1 ? nodeCount / 2 + 1 :
nodeCount / 2;
        }

        public Article getArticleById(string _id) {

            Article article = null;
            string topic = "";
            string content = "";
            string date = "";
            string id = "";
```

```
            string img = "";

            XElement root =
XElement.Load(System.Web.HttpContext.Current.Server.MapPath(filepa
th));
            IEnumerable<XElement> list =
root.XPathSelectElements("//article[@id='"+_id+"']"); ;

            foreach (XElement elarticle in list)
            {
                foreach (XElement el in elarticle.Elements())
                {
                    if (el.Name == "topic")
                        topic = el.Value;
                    else if (el.Name == "content")
                        content = el.Value;
                    else if (el.Name == "date")
                        date = el.Value;
                }

                foreach (XAttribute a in elarticle.Attributes())
                {
                    if (a.Name.LocalName =="id")
                        id = a.Value;
                    else if (a.Name.LocalName.Equals("img"))
                        img = a.Value;
                }
            }
            if (id != "")
                article = new Article(topic, content, date, id,
img);

            return article;
        }

        public ArrayList searchArticles(string s) {

            ArrayList list = new ArrayList();

            XDocument root =
XDocument.Load(System.Web.HttpContext.Current.Server.MapPath(filep
ath));
            IEnumerable<XElement> ar =
               from el in root.Root.Elements("article")
               where  el.Element("content").Value.Contains(s)
               select el;

            foreach (XElement elarticle in ar)
            {
                string topic = "";
```

```
                string content = "";
                string date = "";
                string id = "";
                string img = "";

                foreach (XElement el in elarticle.Elements())
                {
                    if (el.Name == "topic")
                        topic = el.Value;
                    else if (el.Name == "content")
                        content = el.Value;
                    else if (el.Name == "date")
                        date = el.Value;
                }

                foreach (XAttribute a in elarticle.Attributes())
                {
                    if (a.Name.LocalName == "id")
                        id = a.Value;
                    else if (a.Name.LocalName.Equals("img"))
                        img = a.Value;
                }
                Article article = new Article(topic,
                    content.Length > 260 ? content.Substring(0,
260) : content, date, id, img);

                list.Add(article);
            }

            return list;
        }
    }
}
```