ГОСУДАРСТВЕННЫЙ КОМИТЕТ СВЯЗИ, ИНФОРМАТИЗАЦИИ И ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ РЕСПУБЛИКИ УЗБЕКИСТАН

САМАРКАНДСКИЙ ФИЛИАЛ ТАШКЕНТСКОГО УНИВЕРСИТЕТА ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ

ФАКУЛЬТЕТ ИНФОРМАЦИОННЫХ И ПЕДАГОГИЧЕСКИХ ТЕХНОЛОГИЙ

КАФЕДРА ОБЩЕПРОФЕССИОНАЛЬНЫХ ДИСЦИПЛИН

ВЫПУСКНАЯ

КВАЛИФИКАЦИОННАЯ РАБОТА

для получения академической степени бакалавриата по направлению 5521900 - “Информатика и информационные технологии”

ТЕМА: **“Создание автоматизированной информационной системы управления в образовании с использованием Web-технологий”**

Работа рекомендовано к
защите заседанием кафедры
№___ от ____ мая 2014 года
И.О. заведующего кафедрой

_____доц. Кубаев С.Т.
“_____” _____ 2014 г.

Выполнил: студент 4-курса

_____ Вафоев Жасур

Научный руководитель:

_____доц. Абдукаримов А.

САМАРКАНД – 2014

СОДЕРЖАНИЕ

Введение

Актуальность темы: 2014 год станет годом развития страны высокими темпами, мобилизации всех возможностей, последовательного продолжения оправдавшей себя стратегии реформ.

Исключительно важное значение в истекшем году придавалась широкому внедрению информационно-коммуникативных технологий во всех сферах экономики и в нашей повседневной жизни [1].

Дальнейшее углубление реформ в системе образования, совершенствование образовательных стандартов и программ, направленных на повышение уровня и качества образовательного процесса, продолжение укрепления материально-технической базы школ, лицеев, колледжей и высших учебных заведений – всем этим вопросам уделялось первостепенное внимание в 2013 году.

В настоящее время подготовка учащихся невозможна без использования современных технологий обучения. Речь, прежде всего, идёт о применении в учебном процессе информационно-компьютерных технологий.

Программа информатизации и компьютеризации учебного процесса предусматривает оснащенность учебного заведения современным компьютерным оборудованием и программным обеспечением. Вычислительные характеристики современного аппаратного обеспечения меняются и совершенствуются практически ежедневно, поэтому любому учебному заведению практически невозможно обновлять свою техническую базу в соответствии с быстро меняющимися вычислительными возможностями современных компьютеров и обеспечить учебный процесс последними новинками компьютерной техники. Такая же ситуация с программным обеспечением, предполагающим немалые материальные затраты на поддержание соответствующего информационного сервиса.

Цель и задачи работы. Программный комплекс “Автоматизированная система управления образованием” представляет собой множество связанных между собой программ, обеспечивающих управление вузом в едином

информационном пространстве, и включает в себя WEB систему (отображение расписания занятий, успеваемости, учебных планов, начислений оплат за общежитие, контроль оплат за обучение и общежитие, тестирование студентов, запись студентов на изучение дисциплин т.д.). Вся информация хранится в одной общей базе данных.

Основными отличительными характеристиками комплекса, является наличие инструмента самостоятельного создания различных печатных форм и статистических экранных форм, что делает автоматизированную систему управления учебным заведением почти не зависимой от фирмы разработчика. Существующие функциональные возможности, как показала практика и анализ, позволяют охватить фактически все индивидуальные особенности вузов без программной доработки кода. Комплекс также позволяет создавать и учитывать индивидуальные траектории обучения студентов, в том числе через Internet.

Объектом исследования является технологии, средства разработки и языки программирования для создания WEB-системы. Методы доступа к базам данных средствами WWW. Возможность создания WEB-сайта, реализующего задачи предметной области на основе технологий Yii Framework, PHP, CSS, языка программирования HTML.

Научная новизна исследования. Люди постепенно перестают пользоваться обычной почтой, так как, посылая письма по Сети они экономят не только время, но и средства. Электронная почта получила такое широкое применение, что совсем скоро каждый человек будет иметь свой собственный электронный почтовый ящик.

Раньше Web-страница выглядела как обычный статичный текст, без какой-либо графики. Теперь же всё иначе: сайты насыщены иллюстрациями и имеют непростую структуру. Иногда на WEB-узлах создавались формы для связи с пользователем, поисковые функции и, возможно, дискуссионные группы, но, как правило, идея WEB-страницы была основана на предоставлении посетителю статической информации. Решать сложные

задачи связанные с конструированием Web-страниц и призваны такие языки программирования, как C++, Perl, PHP, ASP и другие.

Язык PHP был создан осенью 1994 года, программистом Расмусом Лердорфом. Лердорф собирался написать простой «движок» для своей персональной странички и завершил эту работу к началу 1995 года. Движок был написан на языке Perl и умел делать очень немного, так как создавался только для подсчета количества посетителей странички Расмуса. Этот движок был назван Personal Home Page Tools (PHPT), и единственной его возможностью был подсчет посетителей.

Стоит заметить, что в 1994 году никаких инструментов для создания различных приложений для Web еще не было, да и сам Web только еще начинался. Поэтому те задачи, которые решала программа Расмуса, были актуальны для очень многих пользователей сети, и к нему хлынул поток писем с просьбами предоставить свой инструментарий. К концу 1997 года два программиста Зив Сураски и Энди Гутманс переписали первоначальный лексический анализатор, и к лету 1998 года в полной мере увидела свет третья версия языка – PHP 3. Развитие PHP стремительно продолжалось, в язык сотнями добавлялись новые функции, и в 1999 году число разработчиков, использующих PHP, превысило 1 миллион, что сделало PHP одним из самых популярных языков для разработки Web – приложений. К этому времени к разработке языка подключилось большое количество программистов со всего мира.

На настоящий момент используется пятая версия языка PHP. Данный язык набирает все большую популярность, ведь его возможности практически не ограничены. Однако, еще большей его популяризации мешает тот факт, что далеко не все пользователи решившие создать свой сайт или страничку умеют программировать на достаточно высоком уровне. Именно эту проблему частично мы и попытаемся решить.

Практическая значимость. В наш век обмен информацией немыслим без современных средств связи. Одно из таких средств – современные

глобальные компьютерные сети. Сети - важная часть группового взаимодействия, так как они позволяют быстро и эффективно обмениваться информацией.

WEB-система дает доступ только к преподавателям, студентам и их родителям. Они получать информации об учебе в целом, базе данных его исследований. Родители могут наблюдать об успеваемости своих детей.

Структура и объем работы. Данная выпускная квалификационная работа состоит из 84 страниц, включающих в себя введение, три глав, заключение и приложения 1-2, включающего листинг программы, инструкция по администрированию сайта являющегося основой выполненной выпускной квалификационной работы.

Глава 1. Основы автоматизированной информационной системы в образовании. В этой главе объясняются основные понятия и термины, используемые в данной выпускной квалификационной работе. Дается краткая характеристика системы о ее испольнителах, целях и задачах. Приводятся цели и концепции выпускной квалификационной работы. Дается обоснование создания тестовой страницы системы.

Глава 2. Теоретические основы решения задачи. Приводится язык разметки гипертекстовых страниц HTML. Даются основы языка PHP. Описывается технология PHP и ее применение и представляется Yii Framework.

Глава 3. Описание программного обеспечения и порядок его использования. Эта часть посвящена одной из самых главных тем в сфере разработки любого программного обеспечения – описана программная реализация моделей, структуры WEB-сайта и приводится инфологическая модель строящегося WEB – сайта. В приложениях дано короткое описание выпускной работы и листинг.

Глава 1. ОСНОВЫ АВТОМАТИЗИРОВАННОЙ ИНФОРМАЦИОННОЙ СИСТЕМЫ В ОБРАЗОВАНИИ

1.1. Основные понятия и определения

Управление учебным заведением осуществляется на основе сочетания принципов самоуправления коллектива и единоначалия. В основу положена пятиуровневая структура управления.

Первый уровень структуры – уровень директора (по содержанию – это уровень стратегического управления). Директор учебного заведения определяет совместно с Советом учебного заведения стратегию развития учебного заведения, представляет её интересы в государственных и общественных инстанциях. Общее собрание трудового коллектива утверждает план развития учебного заведения. Директор учебного заведения несет персональную юридическую ответственность за организацию жизнедеятельности учебного заведения, создает благоприятные условия для развития учебного заведения.

На втором уровне структуры (по содержанию – это тоже уровень стратегического управления) функционируют традиционные субъекты управления: Совет учебного заведения, методический совет, попечительский совет, Общее собрание трудового коллектива, профсоюзный орган.

Третий уровень структуры управления (по содержанию – это уровень тактического управления) – уровень заместителей директора. Этот уровень представлен также методическим советом. Методический совет – коллегиальный совещательный орган, в состав которого входят руководители отделов учебного заведения.

Четвертый уровень организационной структуры управления – уровень учителей, функциональных служб (по содержанию – это уровень оперативного управления), структурных подразделений учебного заведения. Методические объединения – структурные подразделения методической

службы учебного заведения, объединяют учителей одной образовательной области.

Пятый уровень организационной структуры – уровень учащихся. По содержанию – это тоже уровень оперативного управления, но из-за особой специфичности субъектов, этот уровень скорее можно назвать уровнем «соуправления». Иерархические связи по отношению к субъектам пятого уровня предполагают курирование, помощь, педагогическое руководство.

1.2. Краткая характеристика системы

Web-система является интегрированной системой управления учебное заведение, которая фокусируется на совершенствование и эволюции в области образования. Для достижения этой цели предлагаемая WEB система является гибкой и представляет инновационные новые тенденции в управлении образованием. Web-система является разнообразной, требовательной и обеспечивает высокое качество системы управления образованием, где каждый пользователь может открыть и реализовать свой потенциал для достижения общего развития.

Программный комплекс “Автоматизированная система управления учебой” представляет собой множество связанных между собой программ, обеспечивающих управление вузом в едином информационном пространстве, и включает в себя WEB систему (успеваемости студентов, посещаемость учителей, надсмотр над студентом родителей, контактные данные всех пользователей, отображение расписания занятий, успеваемости, учебных планов, студентов на изучение дисциплин т.д.). Вся информация хранится в одной общей базе данных.

Основными отличительными характеристиками комплекса, является наличие инструмента самостоятельного создания различных печатных форм и статистических экранных форм, что делает автоматизированную систему управления учебным заведением почти не зависимой от фирмы

разработчика. Существующие функциональные возможности, как показала практика и анализ, позволяют охватить фактически все индивидуальные особенности вузов без программной доработки кода. Комплекс также позволяет создавать и учитывать индивидуальные траектории обучения студентов, в том числе через Internet.

Система не только для управления данными колледжей и вузов, его удобные функции обеспечивают инновационные способы дополняют админ-учитель-ученик-родитель взаимодействия для оптимизации использования ресурсов, обмена отчета и сотрудничество. Система осуществляет множество функций, которые облегчают работу учителям и органам управления образованием, предоставляет данные о посещаемости обучающихся, ведет электронный журнал оценок, содержит список изучаемых предметов в данной школе, колледже или институте, позволяет использовать email и вести страничку учебного заведения и класса в Интернете.

Представляет собой программное обеспечение для автоматизации управления учебное заведение и представляет собой единую программно-информационную платформу, позволяющую путем подключения различных модулей создавать единую базу данных учебного заведения и автоматизировать рабочие места директора, завуча, секретаря, кураторов групп и других сотрудников учебного заведения. Использование подобных комплексов позволяет стимулировать процесс информатизации, создавая единое информационное пространство учебного заведения, объединять рабочие места сотрудников в полноценную систему управления учебным заведением. Администратор сети создает и контролирует общие базы данных, подключение к ним учителей, родителей, библиотекарей через локальную или глобальную сеть.

Система позволяет организовать информационное взаимодействие между всеми категориями пользователей, обеспечивает сбор, передачу, обработку информации для органов управления образованием, пересылку сообщений пользователям.

Персонал учебного заведения может использовать электронную базу данных, включающую в себя базы данных обучающихся и персонала учебного заведения, перечень аудиторий. Система позволяет осуществлять не только сбор и хранение данных о студентах, но и обработку этих данных.

Цели появления на рынке любого программного продукта, позволяющего автоматизировать процесс управления учебным заведением: во-первых, облегчить и упростить деятельность администраторов и учителей, с тем чтобы сократить затраты времени на выполнение рутинных операций и увеличить количество времени, отведенного непосредственно на процесс обучения; во вторых, сформировать информационную инфраструктуру учреждения; в-третьих, собирать, обрабатывать и анализировать информацию о ходе учебного процесса и принимать обоснованные решения по управлению учреждением, а самое главное — сформировать информационное образовательное пространство учебного заведения, организовать информационное взаимодействие между всеми сотрудниками учебного заведения и органами управления образованием всех уровней.

Представленные на отечественном и зарубежном рынке программные продукты — это комплексы программ, включающих в себя автоматизированные информационно-аналитические системы для директора, заведующего учебной частью, учителя, работников библиотеки, медицинского персонала учебного заведения и сотрудников бухгалтерии. Выделим основные функциональные возможности ряда отечественных и зарубежных программных продуктов, программных и программно-аппаратных решений для сферы образования.

Система предлагает комплексное решение по автоматизации учебно-воспитательного процесса в целом, а не только отдельных его частей (работу деканата, кураторов групп и т.д.). Система представляет собой сетевое клиент-серверное приложение и имеет Web интерфейс (т.е. на клиентских машинах не нужно ничего, кроме стандартного браузера, поставляемого вместе с операционной системой), что позволяет пользователю не быть

«привязанному» к какому-то определенному компьютеру, включенному в локальную сеть. Система осуществляет сбор и представление информации о сотрудниках, студентах и родителях. В ней содержатся учебные планы, журналы групп, расписание, разнообразные отчеты и другая информация. Система поддерживает информационное взаимодействие всех участников образовательного процесса: учителей, студентов, родителей, администрации учебного заведения (обмен сообщениями, доска объявлений) — как в рамках локальной сети учебного заведения, так и через Интернет. На сайте учебного заведения учителя могут собирать электронные учебные материалы и методические пособия по всем дисциплинам и организовывать дистанционное обучение (например, в случае болезни обучающегося). Система позволяет привлечь к работе не только работников учебного заведения и обучающихся, но и родителей, у которых дома или на работе есть компьютер с выходом в Интернет.

1.3. Система автоматизированного управления учебой

Проявление общих закономерностей процесса автоматизации управления в социально-экономических системах имеет свою специфику в различных сферах общественного производства. Отрасль образования относится к социально-культурной сфере, что накладывает особую специфику на процессы автоматизации. В отличие от производственной сферы, результаты нематериальной деятельности здесь не так очевидны: производимый продукт практически с трудом поддается количественному измерению, критерии эффективности образовательной деятельности не имеют четкого и однозначно понимаемого определения. Это обстоятельство во многом определило тот факт, что существующая теория автоматизированных систем управления (АСУ) разрабатывалась с ориентацией преимущественно на производящие материальный продукт организации.

Все это существенно затрудняет применение теории АСУ в практике построения автоматизированных систем управления в сфере образования. Опыт практической работы по созданию автоматизированной системы управления территориально-распределенной сетью образовательных учреждений позволяет нам сделать попытку выявить то общее, что позволит применять общие положения теории АСУ как для крупной производственной организации, так и для городской системы образования.

Рассмотрим уровни автоматизации крупного производственного предприятия [1]:

Бизнес-уровень - автоматизация этого уровня приводит к созданию АСУ - системы автоматизации управленческой и финансово-хозяйственной деятельности. Традиционно первой создается именно АСУ, хотя появление развитых интегрирующих технологий на рынке программного обеспечения сделало это требование необязательным.

Уровень проектирования - создание САПР - системы автоматизированного проектирования.

Производственно-технологический уровень создание АСУ ТП - системы автоматизации технологических и производственных процессов.

Опыт работы с системами верхнего уровня показывает, что информации, существующей на уровне АСУ, явно недостаточно для принятия оптимальных решений. Серьезный анализ ситуации невозможен без учета параметров основных технологических процессов, сведений о надежности, качестве, реальной себестоимости производства единицы продукции. Без этих данных руководители по-прежнему будут опираться лишь на собственную интуицию и индивидуальный опыт.

Подключение АСУ ТП к АСУ позволяет включить в общий процесс электронного документооборота производственную документацию (технологические карты, карты контроля качества, индивидуальные паспорта на изделия и т.п.) и дать руководству оперативную информацию от всех

подразделений предприятия, в том числе и от производственно-технологических:

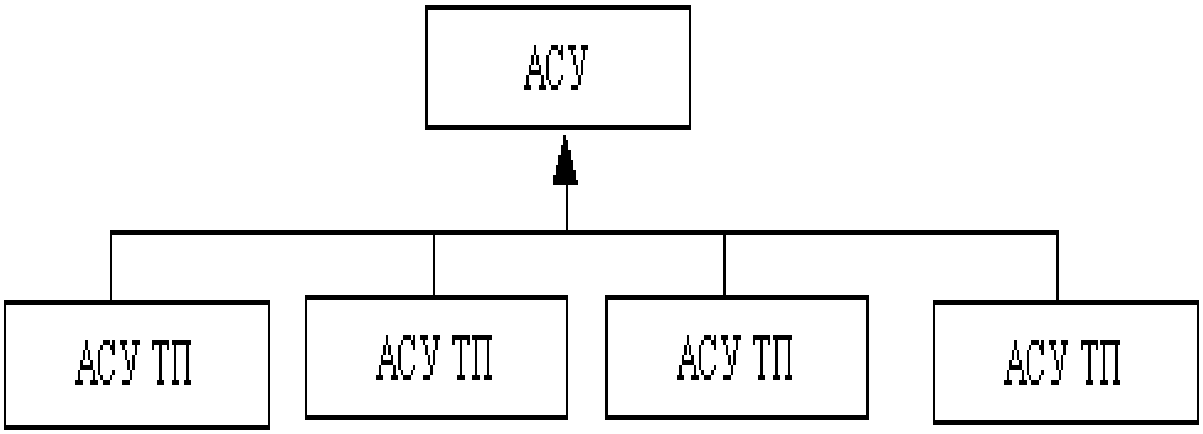

Рис. 1.

До последнего времени в проектировании АСУ ТП участвовали в основном технологи, не знающие досконально сути бизнеса и тенденций его развития и поэтому не умеющие оценить, какая информация и в каком виде должна экспортироваться из АСУ ТП на верхний уровень. В результате даже уже введенная в базы данных АСУ ТП производственно-технологическая информация обычно оставалась невостребованной. Такая ситуация часто приводила (и приводит) к серьезным материальным и моральным потерям.

Для того, чтобы избежать принципиальной "несогласованности" автоматизированных систем на разных уровнях, необходимо непременное непосредственное участие первых лиц предприятия в их проектировании и реализации, особенно на стадии формулирования требований к создаваемым системам (т.е. на стадии постановки).

Автоматизированная система информационного обеспечения управления учебное заведение, колледжем или институтом состоит из следующих обеспечивающих подсистем:

- информационного обеспечения
- технического обеспечения
- математического и программного обеспечения
- методического обеспечения

- организационного обеспечения.

Анализ ее подсистем по содержанию полей баз данных, функциональным возможностям программного обеспечения, содержанию и адресности обучения и консультаций по работе с пользователями, по организационной структуре, обеспечивающей функционирование и развитие системы, показывает, что данная автоматизированная система наиболее близка по своей сущности к АСУ ТП производственного предприятия.

Согласно техническому заданию, она должна прежде всего обеспечивать документооборот в подсистеме общего образования и представлять собой совокупность взаимосвязанных автоматизированных рабочих мест (АРМов) специалистов системы образования:

1. на уровне образовательных учреждений (школ): АРМы директора, заместителей по учебно-воспитательной работе, библиотекаря, психолога;
2. на уровне районных отделов образования: АРМы заведующего отделом, инспекторов школ, инспекторов по охране прав детства, библиотекаря;
3. на уровне Управления образования города: АРМы начальника Управления, инспекторов образовательных учреждений.

Проанализируем причины, по которым создание АСУ ТП в системе образования оказалось первоочередным по сравнению с созданием АСУ, вопреки сложившейся традиции при автоматизации производственных предприятий.

1. Уровень школы

С самого начала школа воспринималась разработчиками как отдельная организация со своей системой управления, однако именно на уровне школы в ходе работ по проектированию программной системы естественным образом произошла переориентация с АСУ на АСУ ТП. По нашему мнению, это обусловлено спецификой информационно-аналитического обеспечения образовательного процесса в школе.

Система информационного обеспечения управления учебным процессом в школе включает, прежде всего, следующие данные:

- списочный состав ученического контингента с данными о ходе обучения школьников: сведения о результатах обучения каждого из них по каждому преподаваемому предмету (итоговые и текущие отметки, данные о переводе на следующую ступень), данные по диагностике качества обучения (тестовый контроль знаний, умений, навыков), а также
- данные, которые можно рассматривать как основные параметры технологического процесса обучения: организация учебного процесса (часы, специфика преподавания предметов, численность классов и т.п.), кадровый состав и квалификация педагогов, материальное, дидактическое и методическое обеспечение учебного процесса и т.д.

Автоматизация работы с информацией именно такого типа требовалась в первую очередь для того, чтобы обеспечить востребованность разрабатываемой системы в школах, и она была проведена, о чем наглядно свидетельствует реестр отчетов АСИОУ "Школа".

Обеспечение востребованности системы на местах было одним из главных условий при ее разработке. Изменения в функционалах специалистов школ (секретаря, завуча и директора), направленные на поддержку автоматизации, также не были произведены, поэтому на местах базы данных велись "в дополнение" к основным обязанностям сотрудников, и их полнота, частота обновления и добросовестность ввода зависели полностью от заинтересованности специалистов в результатах обработки данных.

Выполнение разработчиками условия максимальной востребованности программного продукта привело к тому, что АСИОУ "Школа" как программная система, обеспечивающая управление, прежде всего самим образовательным процессом, принадлежит к виду АСУ ТП.

2. Уровень районного отдела образования (РОО)

В ходе анализа информационных потоков "Школа" , "РОО" , "Управление" выяснилось, что информационное обеспечение деятельности сотрудников РОО не обладает принципиальными

отличиями в сравнении с городским уровнем, поэтому для уровней РОО и Управления было введено общее понятие - "уровень выше школьного".

3. Уровень Управления образования города (УОГ)

На уровне Управления в настоящее время ведется разработка программного обеспечения АРМ ов сотрудников. Таким образом, на уровне выше школьного сохранилось направление автоматизации на создание АСУ "УОГ", состоящей из взаимосвязанных АРМ ов специалистов учреждения.

Анализ реального процесса автоматизации управленческой деятельности в городской системе образования как явления позволяет предположить, что оно укладывается в схему процесса автоматизации крупной территориально-распределенной производственной организации по уровням, о которых говорилось выше:

- *управленческий уровень* - создание АСУ - системы автоматизации управленческой и финансово-хозяйственной деятельности;
- *уровень проектирования* - создание САПП - системы автоматизированного педагогического проектирования;
- *уровень (образовательных) технологий* - *создание* АСУ ОП - системы автоматизации образовательного процесса.

АСУ ТП реализовано программной системой АСИОУ "Школа". Применительно к сфере образования ее можно назвать АСУ ОП, то есть "автоматизированная система управления образовательным процессом", так как именно в школе идет "технологический" процесс обучения. В данном случае школы выступают аналогом территориально-удаленных однородных производственных участков (с точки зрения автоматизации!)

АСУ реализуется программным комплексом АСУ "УОГ", являющейся собственно автоматизированной системой управления городской системой образования, так как именно на этом уровне идут основные процессы принятия управленческих решений.

САПР должен быть реализован программной системой САПП "ЦРО", предназначенной для обеспечения деятельности по проектированию

образовательной среды городской системы образования. Таким образом, для задач автоматизации управления в образовании городская система образования выступает как крупное производственное предприятие с территориально-удаленными технологическими участками и управлением, сосредоточенным в центральном (головном) учреждении.

Именно поэтому попытка создать *АСУ* школы привела к созданию аналога *АСУ ТП* (получившем не совсем соответствующее содержанию название АСИОУ "Школа"), а создание АСУ на уровне Управления образования идет так, как планировалось в техническом задании.

Далее по тексту АСИОУ "Школа" будем называть АСУ ОП "Школа". Связь между АСУ "УОГ" и АСУ ОП "Школа" планируется и обеспечивается на уровне содержания и структур баз данных. Это необходимо, т.к. информации, существующей на уровне АСУ, недостаточно для глубокого анализа ситуации и принятия оптимальных решений.

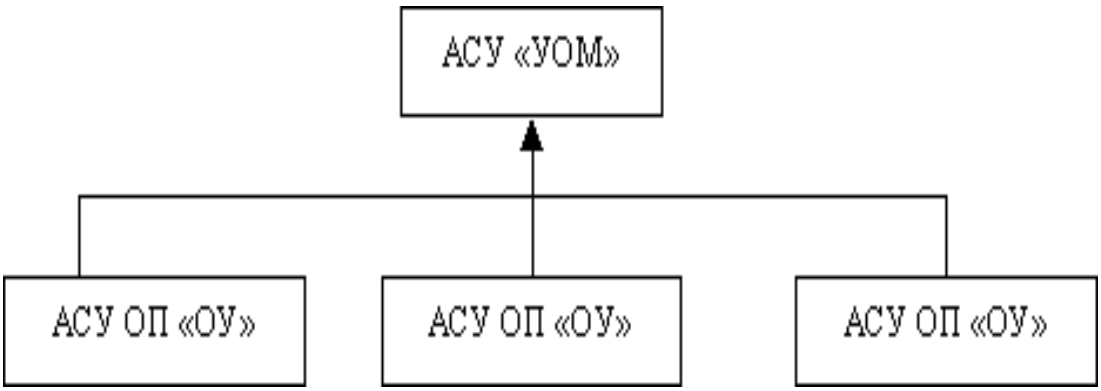

Рис.2

Поэтому необходимо проектирование новых информационных потоков с "технологических участков" городской системы образования в "центр" для учета при выработке управленческих решений параметров образовательного процесса (отличающихся в разных школах), информации по качеству образования (например, результаты тестирования знаний учащихся, результаты диагностики готовности ребенка к школе, диагностика готовности учащихся 9 и 11-х классов к профессиональному самоопределению), данных о себестоимости обучения и т.д.

Для получения информации по различным параметрам состояния управляемого объекта в АСУ ОП "Школа" разработаны различные прикладные подсистемы.

Выявленная нами в ходе практической деятельности модель оптимального решения задачи автоматизации управления городской системой образования представлена на рис. 3.

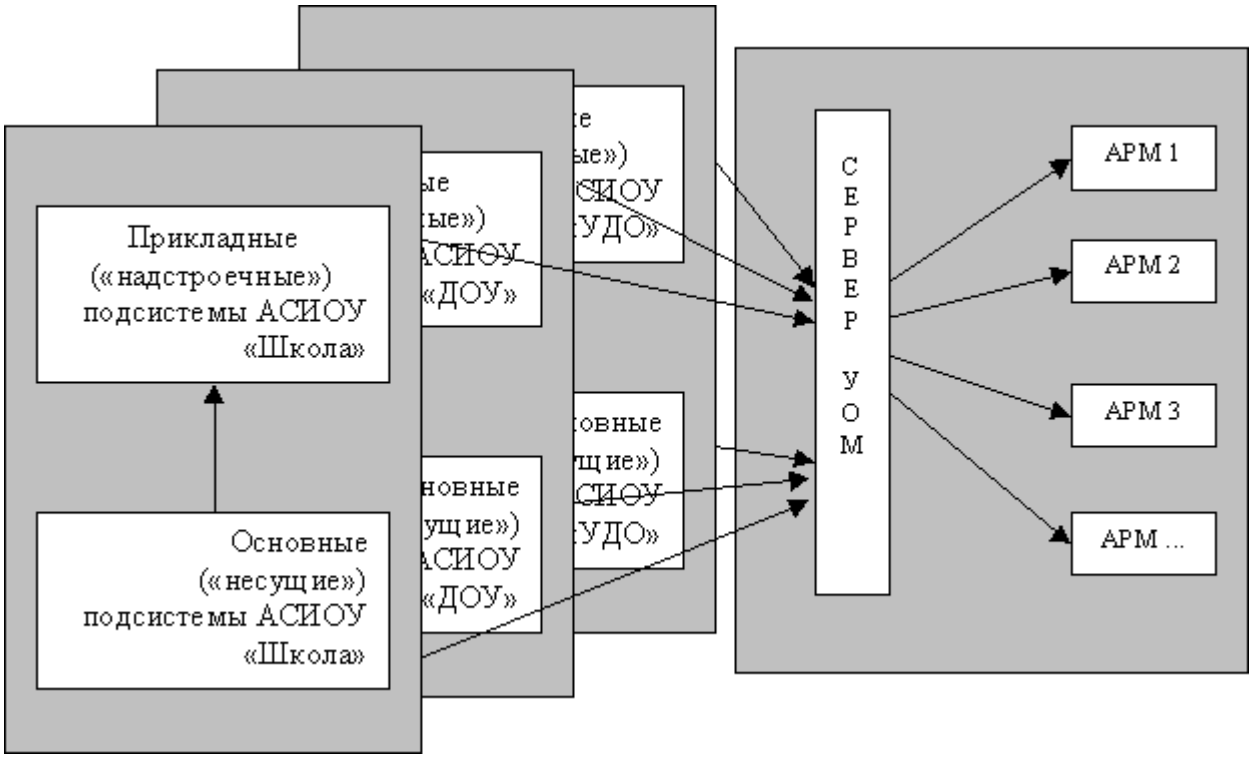

Рис.3. Модель автоматизации системы управления городской системой образования

Для обоснования правомерности привлечения такой модели к решению задач информатизации в сфере образования рассмотрим более подробно понятие АСУ. Прежде всего, дадим определение АСУ и соотнесем ее с СУ - системой управления как таковой.

АСУ представляет собой систему, основанную на использовании современных методов руководства социально-экономическим объектом, применении математических моделей и методов в процессе принятия решений и создании необходимой информационной базы на основе средств вычислительной техники и связи, обеспечивающую достижение нового качества в повышении эффективности управляемой системы [20].

Из этого определения следует, что АСУ, обладая всеми свойствами системы управления, является определенным *этапом развития* системы управления объектом, для которого она создается. Таким образом, речь идет не о параллельном существовании СУ и АСУ рассматриваемым социально-экономическим объектом, а о поэтапном "вживлении" комплекса современных достижений математики, техники и других отраслей науки в процессы функционирования системы управления. Любая реальная СУ, а СУ образования в силу его консервативности - в особенности, обладает определенными инерционными свойствами по отношению к использованию новых методов управления хотя бы потому, что при этом возникает необходимость массовой переподготовки аппарата управления в новых условиях. Поэтому переход от СУ к АСУ осуществляется по этапам, которые в каждом конкретном случае могут иметь различное содержание.

Этапы перехода от СУ к АСУ [20]:

1. **Анализ** системы - это изучение структуры и поведения существующей системы с целью определения ее существенных свойств.
2. **Проектирование (синтез)** системы - это модификация существующей системы (или создание новой), соответствующей определенным требованиям. Предпосылкой успешного проектирования служит качественный анализ первоначальной системы. Само проектирование должно осуществляться при постоянном диалоге с заказчиком.
3. **Внедрение (реализация)** системы - это стадия, когда система начинает функционировать в новом режиме. Существует сильная обратная связь между стадиями описания системы при ее анализе и внедрением, позволяющая корректировать исходные позиции на основании анализа результатов решения.

Рассмотрим этапы перехода от СУ к АСУ применительно к системе образования.

Этап анализа. Проблема целесообразности и глубины анализа управляющей системы, который должен быть предпосылкой для внесения в

нее изменений и ее автоматизации, часто дискутируется. При этом могут проявляться две крайние позиции: первая - без подробного и исчерпывающего анализа нельзя проводить продуманные изменения в системе управления и автоматизировать ее; вторая - учитывая опыт чрезвычайно трудоемкого и при этом малоэффективного анализа систем управления, можно предлагать целесообразное изменение в системе управления без подробного анализа, на основе общих представлений о желаемом состоянии управляющей системы и использовании опыта ее работников.

Опыт автоматизации показывает, что изменения в систему управления необходимо вносить на основе ее анализа, однако такого, при котором задана его цель и который протекает итерационно на нескольких различных уровнях. При этом детализация допускается лишь на тех участках, на которых предполагается проектирование изменений в самом ближайшем будущем.

Этап проектирования. Последние достижения в области проектирования сложных программных систем позволяют говорить об этапах анализа и проектирования как последовательных с определенной долей условности. В реальной деятельности четкого разделения нет, работы по этапам делаются "по блочно", параллельно и на разных участках системы управления - с разной скоростью, причем сам спроектированный "блок" сразу же становится инструментом анализа системы управления и перепроектируется по результатам этого анализа. Это стало возможным благодаря развитию вычислительной техники и созданию средств разработки сложных программных систем.

Этап внедрения. Для территориально-распределенной городской системы образования, обладающей сложной иерархией управления, этот этап представляет наибольшие трудности.

Совершенствование системы управления социально-экономическими объектами всегда вызывает, как следствие, изменения в деятельности

определенного круга работников, что не может не вызывать естественного сопротивления с их стороны. Внедрение АСУ привносит в большинстве случаев новые методы и средства работы, предполагает достаточно высокую работоспособность, приспособляемость работников к деятельности в новых условиях, повышение их квалификации и профессионального образования. Упор в работе переносится на аналитический метод принятия решений. Резко возрастают требования к качеству анализа входных данных и полученных результатов.

Опыт работы по внедрению и сопровождению АСУ ОП "Школа" в общеобразовательных учреждениях городской системы позволяет сформулировать следующие оптимизирующие условия создания реально работающих систем информационного обеспечения управления образовательным процессом в городской системе образования:

1. наличие специализированного подразделения в головном управляющем учреждении (либо в подведомственном учреждении), осуществляющего разработку, внедрение и сопровождение унифицированного программного обеспечения;
2. курирование работы этого подразделения со стороны главного лица управляющего учреждения (либо его заместителя);
3. работающая система методического обеспечения процесса информатизации: постоянно действующие курсы подготовки и переподготовки пользователей разрабатываемой программной системы, методическая литература по работе с ней, семинары, круглые столы и т.п.
4. создание механизма "перехвата", централизации и упорядочивания информационных запросов к школам из различных вышестоящих организаций с целью автоматизации составления отчетов с использованием баз данных АСУ ОП;
5. установление с городского уровня целевой доплаты к ставке школьного делопроизводителя для материального стимулирования качественного ведения баз данных АСИОУ;

6. реализация программного обеспечения АСУ ОП на доминирующей программной платформе для пользовательских программ, обеспечивающей привычный для пользователя интерфейс;
7. наличие централизованной службы технического обслуживания вычислительной техники в школах, предназначающейся для информатизации управленческой деятельности.

Заметим, что условие централизации информационных запросов является одним из самых важных для функционирования и развития АСУ ОП. Как правило, все сведения, необходимые для составления ответа на любой информационный запрос, пришедший из вышестоящей организации, предусмотрены структурой баз данных, однако даже в тех школах, где эти сведения были введены, отчеты формируются и заполняются "вручную". Это объясняется тем, что для автоматического формирования отчета должна быть запрограммирована конкретная заданная запрашивающей организацией форма обработки и вывода данных. Для этого требуется работа программиста, участвующего в сопровождении программного обеспечения АСУ ОП. При условии централизации информационных запросов работа по программированию новой формы и установке программного обеспечения в образовательных учреждениях может быть произведена в оптимальные сроки.

Следует также подчеркнуть важность очевидного, но сложного в практической реализации условия функционального подчинения разработчиков АСУ ОП центральной управляющей организации городской системы образования. Это является прямым следствием принципа первого руководства при разработке и внедрении АСУ в любой производственной организации.

1.4. Цели и концепции выпускной работы

Целью разработки является автоматизация процесса управления образованием, доступа к информации через WEB-сайт:

- к графическим и текстовым материалам;
- к базам данных информаций образовательного учреждения;
- к информации о проводимых мероприятиях и к их материалам;
- возможность осуществлять тестирование студентов;
- возможность самостоятельного создания различных печатных форм и статистических экранных форм;
- иметь возможность информационного обмена через WEB-сайт.

Основанием для разработки явилось:

1) Обращение руководителя ООО «Proper to Expert» о создании WEB-сайта для автоматизации управления образовательного учреждения: отображения расписания занятий, успеваемости, учебных планов, начислений оплат за общежитие, контроль оплат за обучение и общежитие, тестирование студентов, запись студентов на изучение дисциплин т.д., информационного обмена между преподавателями и родителями студентов.

2) Задание на выпускную квалификационную работу, тема «Создание автоматизированной информационной системы управления в образовании с использованием Web-технологий», полученное на кафедре «Компьютерные системы» Самаркандского филиала ТУИТ.

Назначение WEB-сайта. Разрабатываемый программный комплекс “Автоматизированная система управления образованием” предназначен для решения следующих задач:

- Управление вузом в едином информационном пространстве.
- Отображение расписания занятий, успеваемости студентов.
- Учебных планов, учебных программ, календарных планов по предметам.
- Отображать начисление оплат за обучение и за общежитие.

- Тестирование студентов, запись студентов на изучение дисциплин.
- Создания различных печатных форм и статистических экранных форм.
- Создание и учет индивидуальных траекторий обучения студентов, в том числе через Internet.

Требования к WEB-сайту. При разработке WEB-сайта должны учитываться следующие параметры:

- Сайт должен иметь основную тему.
- Сайт должен обладать хорошей информационной базой по теме.
- Сайт должен быстро открываться.
- Сайт должен обладать мощным навигационным меню. В смысле: "мало весит килобайт - быстро открывает сайт".
- Сайт (желательно) должен обладать хорошим дизайном.
- Прежде чем преступить к созданию WEB-сайта, необходимо спланировать его (особенно если он занимает несколько страниц).
- Страницы должны автоматически подстраиваться под различные разрешения экранов, при этом подавляющее большинство посетителей увидит страницу развернутой на весь экран не зависимо от размеров диагонали монитора и установленного на нем разрешения.
- Страницы должны быть информативно независимыми, (пользователь может начать просмотр с любой страницы).
- Должна существовать возможность существует возможность попадать с каждой страницы на каждую.
- Навигационные средства необходимо расположить в верхней и нижней части каждой страницы для удобства перемещения пользователя.
- Начальная страница будет служить входом на сайт.
- Графические файлы загружаются медленно. Поэтому не стоит злоупотреблять графикой.

- Пользователь может использовать браузер только в текстовом режиме, и графические файлы не будут отображаться на странице. Для таких пользователей должен быть предусмотрен альтернативный текст, который будет сообщать им об изображении.
- Изображения и таблицы не должны сливаться с текстом.
- Можно использовать заголовки при организации текста, чтобы проще найти нужную информацию.
- Нежелательно использовать более двух или трех различных гарнитур шрифта.
- Нежелательно использовать много подчеркнутого текста. Подчеркнутый текст ассоциируются со ссылками (гиперсвязь).
- Необходимо придать страницам единый стиль. Обязательно постоянно проверять, что все гиперсвязи ведут пользователя куда надо и следить за этим постоянно.
- WEB-сайт должен обрабатывать сообщения пользователей и результаты обработки заносить в базу данных.
- WEB-сайт должен иметь хорошую навигацию по сайту, в том числе и на больших текстовых страницах с целью перехода в любую часть текста по ключевым словам и ссылкам.

Глава 2. ТЕОРЕТИЧЕСКИЕ ОСНОВЫ РЕШЕНИЯ ЗАДАЧ

2.1. Язык разметки гипертекстовых страниц HTML

Hyper Text Markup Language (HTML) является стандартным языком, предназначенным для создания гипертекстовых документов в среде WEB. HTML-документы могут просматриваться различными типами WEB-браузеров. Когда документ создан с использованием HTML, WEB-браузер может интерпретировать HTML для выделения различных элементов документа и первичной их обработки. Использование HTML позволяет форматировать документы для их представления с использованием шрифтов, линий и других графических элементов на любой системе, их просматривающей.

Большинство документов имеют стандартные элементы, такие, как заголовок, параграфы или списки. Используя тэги HTML можно обозначать данные элементы, обеспечивая WEB-браузеры минимальной информацией для отображения данных элементов, сохраняя в целом общую структуру и информационную полноту документа. Все что необходимо, чтобы прочитать HTML-документ - это WEB-браузер, который интерпретирует тэги HTML и воспроизводит на экране документ в виде, который ему придает автор [19].

В большинстве случаев автор документа строго определяет внешний вид документа. В случае HTML, читатель (основываясь на возможностях WEB-браузера) может, в определенной степени, управлять внешним видом документа, но не его содержимым. HTML позволяет отметить, где в документе должен быть заголовок или абзац при помощи тэга HTML, а затем предоставляет WEB-браузеру интерпретировать эти тэги. Например, один WEB-браузер может распознавать тэг начала абзаца и представлять документ в нужном виде, а другой не имеет такой возможности и представляет документ в одну строку. Пользователи некоторых WEB-браузеров имеют, также, возможность настраивать размер и вид шрифта, цвет и другие параметры, влияющие на отображение документа.

HTML-тэги могут быть условно разделены на две категории:

- тэги, определяющие, как будет отображаться WEB-браузером тело WEB-страницы в целом;
- тэги, описывающие общие свойства WEB-страниц.

Основное преимущество HTML заключается в том, что WEB-страницы могут быть просмотрены на WEB-браузерах различных типов и на различных платформах.

HTML-документы могут быть созданы при помощи любого текстового редактора или специализированных HTML-редакторов и конвертеров.

Все тэги HTML начинаются с "<" (левой угловой скобки) и заканчиваются символом ">" (правой угловой скобки). Как правило, существует стартовый тэг и завершающий тэг. Завершающий тэг выглядит, как стартовый и отличается от него прямым слэшем перед текстом внутри угловых скобок. Некоторые тэги, такие, как <P> (тэг, определяющий абзац), не требуют завершающего тэга. HTML не реагирует на регистр символов, описывающих тэг.

Когда WEB-браузер получает документ, он определяет, как документ должен быть интерпретирован. Самый первый тэг, который встречается в документе, должен быть тэгом <HTML>. Данный тэг сообщает WEB-браузеру, документ написан с использованием HTML.

Тэг заголовочной части документа используется сразу после тэга <HTML> и более нигде в теле документа. Данный тэг представляет собой общее описание документа. Стартовый тэг <HEAD> помещается непосредственно перед тэгом <TITLE> и другими тэгами, описывающими документ, а завершающий тэг </HEAD> размещается сразу после окончания описания документа.

Большинство WEB-браузеров отображают содержимое тэга <TITLE> в заголовке окна. Заголовок, ограниченный тэгами <TITLE> и </TITLE>, размещается внутри <HEAD>-тэгов.

Тело документа находиться между тэгами <BODY> и </BODY>. Это та

часть документа, которая отображается как текстовая и графическая (смысловая) информация документа.

Как любой язык, HTML позволяет вставлять в тело документа комментарии, которые сохраняются при передаче документа по сети, но не отображаются браузером. Синтаксис комментария: <!-- Это комментарий -->.

Гипертекстовые ссылки являются ключевым компонентом, делающим WEB привлекательным для пользователей. Добавляемые гипертекстовые ссылки, создают набор документов связанным и структурированным, что позволяет пользователю получать необходимую ему информацию максимально быстро и удобно.

Ссылки имеют стандартный формат, что позволяет браузеру интерпретировать их и выполнять необходимые функции (вызывать методы) в зависимости от типа ссылки. Ссылки могут указывать на другой документ, специальное место данного документа или выполнять другие функции. URL может указывать на специальное место по абсолютному пути доступа, или указывать на документ в текущем пути доступа, что часто используется при организации больших структурированных WEB-сайтов.

HTML использует URL (Uniform Resource Locator) для представления гипертекстовых ссылок и ссылок на сетевые сервисы внутри HTML-документа. Первая часть URL (до двоеточия) описывает метод доступа или сетевой сервис. Другая часть URL (после двоеточия) интерпретируется в зависимости от метода доступа. Обычно, два прямых слэша после двоеточия обозначают имя машины.

Для того чтобы браузер отобразил ссылку на URL, необходимо отметить URL специальными тэгами в HTML-документе. Синтаксис HTML, позволяющий это сделать - следующий:

<A HREF="URL"> текст-который-будет-подсвечен-как-ссылка </A>

Тэг <A HREF="URL">открывает описание ссылки, а тэг </A> - закрывает его. Любой текст, находящийся между данными двумя тэгами подсвечивается специальным образом Web-браузером. Обычно этот текст

отображается подчеркнутым, и выделенным синим (или другим заданным пользователем) цветом. Текст, обозначающий URL, не отображается браузером, а используется только для выполнения предписанных им действий при активизации ссылки (обычно при щелчке мыши на подсвеченном или подчеркнутом тексте).

В HTML можно делать ссылки на различные участки или разделы одного и того же документа, используя специальный скрытый маркер для этих разделов. Это позволяет быстро переходить от раздела к разделу внутри документа, не используя прокручивание экрана.

Одна из наиболее привлекательных черт WEB - возможность включения ссылок на графические и иные типы данных в HTML-документ. Делается это при помощи тэга <IMG...ISMAP>. Использование данного тэга позволяет значительно улучшить внешний вид и функциональность документов.

Существует два способа использования графики в HTML-документах. Первый - это внедрение графических образов в документ, что позволяет пользователю видеть изображения непосредственно в контексте других элементов документа. Синтаксис тэга:

<IMG SRC="URL" ALT="text" HEIGHT=n1 WIDTH=n2
ALIGN=top|middle|bottom|texttop ISMAP>

Второй - позволяет включать в документ фоновый рисунок, который будет матрицироваться, и отображаться на фоне всего документа. Ненавязчивый полупрозрачный рисунок обычно хорошо выглядит в качестве фона для большинства WEB-страниц. Описание фонового рисунка включается в тэг BODY и выглядит следующим образом:

<BODY BACKGROUND="picture.gif">

Цвет задается шестизначным числом в шестнадцатеричном формате по схеме RGB (Red, Green, Blue). Цвет #000000 соответствует черному, а цвет #FFFFFF - белому.

HTML позволяет использовать различные стили шрифтов для

выделения текстовой информации документах. Список стилей, поддерживаемых большинством браузеров:

- *bold (жирный);*
- *italic (наклонный);*
- *mono spaced (type writer - с использованием фиксированных шрифтов).*

Дополнительные стили:

- big (большой);
- small (маленький);
- sub (подстрочник);
- sup (надстрочник).

HTML позволяет изменять размер шрифта при помощи тэга:

<FONT SIZE=+|- n>

Шрифт может иметь размер от 1 до 7. Можно прямо указать размер шрифта цифрой, или указать смещение относительно базового значения (по умолчанию-3) в положительную или отрицательную сторону. Базовое значение можно изменить при помощи тэга:

<BASEFONT SIZE=n>

Можно изменить цвет шрифта при помощи тэга:

<FONT COLOR="#xxxxxx>

Некоторые браузеры позволяют пользователю, заполнив специальную форму, возвращающую полученное значение, выполнять некоторые действия на WEB-сервере. Когда форма интерпретируется WEB-браузером, создается специальные экранные элементы GUI, такие, как поля ввода, checkboxes, radio buttons, выпадающие меню, кнопки и т.д. Когда пользователь заполняет форму и нажимает кнопку "Подтверждение" (SUBMIT - специальный тип кнопки, который задается при описании документа), информация, введенная пользователем в форму, посылается HTTP-серверу для обработки и передаче другим программам, работающим под сервером.

При описании формы, каждый элемент ввода данных имеет тэг

<INPUT>. Когда пользователь помещает данные в элемент формы, информация размещается в разделе VALUE данного элемента. Все формы начинаются тэгом <FORM> и завершаются тэгом </FORM>. В зависимости от используемого метода можно посылать результаты ввода данных в форму двумя путями:

GET: Информация из формы добавляется в конец URL, который был указан в описании заголовка формы.

POST: Данный метод передает всю информацию о форме немедленно после обращения к указанному URL. Программа получает данные из формы в стандартный поток ввода. Сервер не будет пересылать сообщение об окончании пересылки данных в стандартный поток ввода; вместо этого используется переменная среды CONTENT_LENGTH для определения, какое количество данных необходимо считать из стандартного потока ввода.

ACTION описывает URL, который будет вызываться для обработки формы. Данный URL почти всегда указывает на программу, обрабатывающую данную форму.

Формы можно использовать для отправки не только небольших информационных сообщений в виде параметров, а также и для отправки файлов [20].

HTML позволяет создавать таблицы. Основные тэги таблицы: <TABLE> ...</TABLE>. Все элементы таблицы должны находиться внутри этих двух тэгов. По умолчанию таблица не имеет обрамления и разделителей. Обрамление добавляется атрибутом BORDER.

Строка таблицы: <TR>...</TR>. Количество строк таблицы определяется количеством встречающихся пар тэгов <TR>..</TR>. Строки могут иметь атрибуты ALIGN и VALIGN, которые описывают визуальное положение содержимого строк в таблице.

Ячейка таблицы: <TD>...</TD>. Описывает стандартную ячейку таблицы. Ячейка таблицы может быть описана только внутри строки таблицы. Каждая ячейка должна быть пронумерована номером колонки, для

которой она описывается. Если в строке отсутствует одна или несколько ячеек для некоторых колонок, то браузер отображает пустую ячейку. Расположение данных в ячейке по умолчанию определяется атрибутами ALIGN=left и VALIGN=middle. Данное расположение может быть исправлено как на уровне описания строки, так и на уровне описания ячейки.

Заголовок таблицы: <TH>...</TH>. Ячейка заголовка таблицы имеет ширину всей таблицы; текст в данной ячейке имеет атрибут BOLD и ALIGN=center.

Подпись: <CAPTION>...</CAPTION>. Данный тэг описывает название таблицы (подпись). Тэг <CAPTION> должен присутствовать внутри <TABLE>...</TABLE>, но снаружи описания какой-либо строки или ячейки. По умолчанию <CAPTION> имеет атрибут ALIGN=top, но может быть явно установлен в ALIGN=bottom. ALIGN определяет, где - сверху или снизу таблицы - будет поставлена подпись. Подпись всегда центрирована в рамках ширины таблицы.

Атрибут BORDER используется в тэге TABLE. Если данный атрибут присутствует, граница таблицы прорисовывается для всех ячеек и для таблицы в целом. BORDER может принимать числовое значение, определяющее ширину границы.

Если атрибут ALIGN присутствует внутри тэгов <CAPTION> и </CAPTION>, то он определяет положение подписи для таблицы (сверху или снизу). По умолчанию ALIGN=top. Если атрибут ALIGN встречается внутри <TR>, <TH> или <TD>, он управляет положением данных в ячейках по горизонтали. Может принимать значения left (слева), right (справа) или center (по центру).

Атрибут VALIGN встречается внутри тэгов <TR>, <TH> и <TD>. Он определяет вертикальное размещение данных в ячейках. Может принимать значения top (вверху), bottom (внизу), middle (по середине) и baseline (все ячейки строки прижаты кверху).

Атрибут NOWRAP говорит о том, что данные в ячейке не могут

логически разбиваться на несколько строк и должны быть представлены одной строкой.

Атрибут COLSPAN указывает, какое количество ячеек будет объединено по горизонтали для указанной ячейки. По умолчанию - 1.

Атрибут ROWSPAN указывает, какое количество ячеек будет объединено по вертикали для указанной ячейки. По умолчанию - 1.

Параметр COLSPEC позволяет задавать фиксированную ширину колонок либо в символах, либо в процентах, например COLSPEC="20%".

На данный момент в WEB используется два типа растровых файлов: в форматах JPEG и GIF. JPEG-формат хорошо передает цветовые и тоновые раскаты, размытые границы (например, фото). JPEG-файл хорошо масштабируется в браузере. Плохо передает ровные плоскости цвета, в компрессии уступает GIF-формату. GIF-формат хорошо передает ровные плоскости цвета, жесткие границы (например, векторную графику, логотипы). Имеет максимальную компрессию, допускает прозрачный фон. Плохо масштабируется в браузере, искажает цветовые и тоновые раскаты.

Каскадные таблицы стилей CSS.

Основным понятием CSS является стиль – т. е. набор правил оформления и форматирования, который может быть применен к различным элементам страницы. В стандартном HTML для присвоения какому-либо элементу определенных свойств (таких, как цвет, размер, положение на странице и т. п.) приходилось каждый раз описывать эти свойства, даже если на одной страничке должны располагаться 10 или 110 таких элементов, ничуть не отличающихся один от другого. Необходимо было десять или сто десять раз вставить один и тот же кусок HTML кода в страничку, увеличивая размер файла и время загрузки на компьютер просматривающего ее пользователя.

CSS действует другим, более удобным и экономичным способом. Для присвоения какому-либо элементу определенных характеристик нужно один раз описать этот элемент и определить это описание как стиль, а в

дальнейшем просто указывать, что элемент, который необходимо оформить соответствующим образом, должен принять свойства стиля.

Более того, можно сохранить описание стиля не в тексте странички, а в отдельном файле – это позволит использовать описание стиля на любом количестве Web-страниц. И еще одно, связанное с этим, преимущество – возможность изменить оформление любого количества страниц, исправив лишь описание стиля в одном (отдельном) файле.

Кроме того, CSS позволяет работать со шрифтовым оформлением страниц на гораздо более высоком уровне, чем стандартный HTML, избегая излишнего утяжеления страниц графикой.

Расположение описания стилей в отдельном файле имеет смысл, если планируется применять эти стили к большему, чем одна, количеству страниц. Для этого нужно создать обычный текстовый файл, описать с помощью языка CSS необходимые стили, поместить этот файл на Web-сервер, а в коде Web-страниц, которые будут использовать стили из этого файла, нужно будет сделать ссылку на него. Делается это с помощью тега <LINK>, располагающегося внутри тега <BODY> WEB-страниц.

Первые два параметра этого тега являются зарезервированными именами, требующимися для того, чтобы сообщить браузеру, что на этой страничке будет использоваться CSS. Третий параметр – HREF= «URL» – указывает на файл, который содержит описания стилей. Этот параметр должен содержать либо относительный путь к файлу – в случае, если он находится на том же сервере, что и документ, из которого к нему обращаются – или полный URL («http://...») в случае, если файл стилей находится на другом сервере.

Второй вариант, при котором описание стилей располагается в коде Web-странички, внутри тега <BODY>, в теге <STYLE type="text/css">... </STYLE>. В этом случае можно использовать эти стили для элементов, располагающихся в пределах странички. Параметр type="text/css" является обязательным и служит для указания браузеру использовать CSS.

И третий вариант, когда описание стиля располагается непосредственно внутри тега элемента, который описывается. Это делается с помощью параметра STYLE, используемого при применении CSS с большинством стандартных тегов HTML. Этот метод нежелателен, и понятно почему: он приводит к потере одного из основных преимуществ CSS – возможности отделения информации от описания оформления информации. Впрочем, если необходимо описать лишь один элемент, этот вариант расположения описания стилей также вполне применим.

Также элементы страниц, созданные с использованием CSS, используют механизм наследования: т. е. если вы располагаете изображение внутри тега <P>...</P>, оформленного с помощью CSS, с отступами, так, чтобы параграф занимал только определенную часть ширины страницы, изображение также унаследует значения отступов, указанные для этого параграфа.

CSS реализует возможность присваивать стили не всем одинаковым элементам страницы, а избирательно – для этого используется параметр CLASS = "имя класса" или идентификатор ID=«имя элемента», присваивающиеся любому элементу страницы. Параметр CLASS применяется в случае, если необходимо создать одинаковый стиль для нескольких, но не всех элементов страницы (одинаковых или разных).

Присвоение стилей с помощью идентификаторов применяется в случае, если данному идентификатору соответствует только один элемент на странице. Если элементов, которым необходимо присвоить такой стиль, несколько – это уже класс.

В настоящее время язык CSS насчитывает довольно большое количество свойств элементов HTML, которыми он может управлять [20].

2.2. Основы PHP

PHP (читается как пи-эйч-пи) – один из популярнейших языков программирования в сети Интернет. Дословно аббревиатура переводиться

как Personal Home Page. Данный язык существует с 1994 года. Его создателем является Расмус Лердорф (Rasmus Lerdorf). Первые версии скрипт-движка использовались исключительно только в личных целях автора. PHP начал свою жизнь как ненавязчивая CGI-оболочка, написанная на Perl. Небольшое отступление: CGI (Common Gateway Interface) - общий шлюзовой интерфейс - является стандартом, который предназначен для создания серверных приложений HTTP. Такое приложение, которое называют шлюзом или CGI-программой, запускается www-сервером в реальном времени. Сервер передает запросы пользователя CGI-программе, которая их обрабатывает и возвращает результат своей работы на экран пользователя. Таким образом, Интернет-сервер получает динамическую информацию, которая может изменяться в результате влияния различных факторов. Сам шлюз может быть написан на C/C++, Fortran, Perl, TCL, Unix Schell, Visual Basic, Apple Script и других подобных языках. Но в случае с PHP, для написания его в первоначальном варианте был выбран язык Perl.

Каждая команда PHP обычно начинается с тэга «<?php» и заканчивается «?>». Если вы используете несколько команд подряд, они могут быть объединены внутри одной пары тэгов - в этом случае каждую команду необходимо отделять друг от друга символом «;». В любом месте PHP-скрипта можно размещать комментарии. Для начала комментария используется символы «/*», а для его завершения – «*/». Если комментарий небольшой, удобней использовать символы «//» - тогда все, что следует за ними до конца строки, будет игнорироваться, подобно лишним символам пробела, табуляции и новой строки.

PHP позволяет использовать переменные - при этом их не нужно описывать так, как это делается в Visual Basic или Pascal. Вы просто вводите необходимую переменную там, где вам нужно и тогда, когда вам это нужно. Имена переменных начинаются с символа «$». Переменные могут быть трех типов: целые, с плавающей запятой и символьные строки. А также переменные могут являться массивами. Хотя разделение на типы скорее

условное, и каждая функция стремится использовать правильный тип автоматически.

Для отображения переменных или результатов работы скрипта используется команда echo. Вот небольшой пример рабочей страницы, для того, что бы можно было оценить, как просто включить PHP-скрипт в обычный HTML-документ:

<?php $d = date(d.m.Y); echo "Последние обновления: $d "; ?>

В результате загрузки такой страницы, PHP обработает все команды, которые находятся между специальными тэгами, и вы увидите, в данном случае, строку вроде «Последние обновления: 8.04.2004». Браузер получает чистый HTML-код и если просмотреть его, вы не найдете там никаких следов пребывания PHP.

Циклы в PHP

Циклы в программировании – это повторяющиеся несколько раз операции. Начало (точка отсчета) указывается в начале цикла, а длительность его выполнения ограничивается каким-либо условием. Примером цикла может служить копирование нескольких файлов. Алгоритм выполнения этого задания можно описать так: установить счетчик скопированных файлов в ноль, скопировать файл, проверить закончились файлы или нет, если нет - увеличить счетчик скопированных файлов, вернуться к началу цикла (опять скопировать файл), если да – закончить цикл. Теперь рассмотрим, как циклы реализуются в синтаксисе PHP.

```
<?php
$i = 0; $n = 10;
while ($i <= $n):
echo $i."<br>\n";
$i++;
endwhile;
?>
```

Смысл скрипта очень прост. Присваиваем переменной $i значение, соответствующее началу цикла, а переменной $n – значение конца цикла. Далее открываем цикл оператором WHILE ( ), и внутри его скобок описываем условие, при выполнении которого цикл будет продолжать свою работу. В нашем случае выполнение не прервется, пока $i <= $n. Как только это условие будет нарушено, управление будет передано следующей за циклом операции PHP. Внутри цикла могут быть любые команды PHP (разделенные между собой как обычно – точкой с запятой). Только нужно следить за тем, что бы переменная $i, используемая в цикле, была увеличена (и совсем не обязательно на единицу), иначе цикл станет бесконечным, и интерпретатор PHP будет выполнять его, пока не закроется сессия (окно браузера). Оператор ENDWHILE означает конец цикла. Скрипт, описанный здесь, выводит на экран браузера цифры начиная с 0 до 10. Причем цифры будут выведены в столбик, так как после вывода на экран значения переменной $i мы выводит HTML-тег перевода строки (
). После него идет перевод строки для кода, переданного клиенту (его можно посмотреть, выбрав просмотр в виде HTML в меню Вашего браузера). Это не обязательно, но таким образом достигается удобочитаемость кода. Для примера я привожу еще один вариант выполнения указанной выше задачи, но уже гораздо более правильно в смысле чистоты кода и скорости выполнения.

```
<?php
$i = 0; while ($i <= 10) { echo $i++."<br>"; }
?>
```

Удивительно, но эти два примера абсолютно идентичны в смысле результата. Но сам скрипт уместился в одну строчку! Разница – в стиле применения оператора цикла и в том, что переменная цикла выводится на экран одновременно с увеличением. И в этом – вся прелесть программирования. Иногда бывает что-то простое сделать очень трудно, а сложное – легко. Никогда не стоит останавливаться на уже достигнутом, а

пробовать применять другие алгоритмы и решения. Вот Вам еще один вариант решения. Он основан на применении конструкции PHP DO…WHILE. Это тоже цикл, но отличается он от просто WHILE тем, что значение логического выражения проверяется не до (как в случае с WHILE), а после окончания работы операторов, включенных в сам цикл. Таким образом, DO…WHILE гарантированно будет выполнен хотя бы один раз, что в случае с WHILE совсем не обязательно. Ведь если условие есть ложь, управление сразу будет передано дальше. Для циклов DO..WHILE существует только один вид синтаксиса:

```
<?php
$i = 0; do { echo $i."<br>\n"; $i++; } while ($i <=10);
?>
```

Казалось бы, достаточно вариантов, но это не все - существует еще несколько вариантов цикла. И, как правило, именно они и используются программистами. Циклы FOR - наиболее мощные циклы в PHP. Они работают подобно их аналогам в языке программирования C. Синтаксис цикла FOR:

```
FOR (expr1; expr2; expr3) statement
```

Первое выражение (expr1) безусловно, вычисляется (выполняется) в начале цикла. В начале каждой итерации (проход цикла) вычисляется expr2. Если оно равно TRUE (истина), то цикл продолжается и выполняются вложенный(е) оператор(ы). Если оно равно FALSE (ложь), то цикл заканчивается. В конце каждой итерации вычисляется (исполняется) expr3. Каждое из этих выражений может быть пустым. Если expr2 пусто, то цикл продолжается бесконечно (PHP по умолчанию считает его равным TRUE, как и в языке C). Это не так бесполезно, как могло бы показаться, так как зачастую требуется закончить выполнение цикла используя оператор BREAK в сочетании с логическим условием вместо использования логического выражения в FOR. Если внутри цикла (любого) встречается этот оператор (BREAK), цикл безусловно прекращает выполнение итерации, и управление

передается следующей за циклом команде. Если встречается оператор CONTINUE – управление передается на начало следующего ближайшего цикла.

Массивы в PHP

Массивы и механизмы их использования в значительной степени повышают эффективность программ на PHP. Научившись работать с массивами, мы сможем сохранять и обрабатывать сложные структуры данных. К сожалению, в простой переменной можно хранить только одно значение, однако существует другой тип переменной, которая и называется массивом. Массив даёт возможность хранить сколь угодно значений под одной переменной. Каждое конкретное значение в массиве имеет свой порядковый номер либо свой текстовый указатель, поэтому к нему можно легко обратиться в любой момент. Естественно, если нужно сохранить, к примеру, 5 значений, ничто не мешает завести 5 переменных. Так для чего же тогда нужны массивы? В первую очередь для гибкости. Мы можем сохранить 2 значения, а может и две сотни, не заводя новых переменных. Кроме того массив даёт возможность эффективно обрабатывать эти переменные. Можно просмотреть их все в цикле, выбрать любую нужную или сортировать в числовом или же алфавитном порядке, можно даже указать свой собственный порядок сортировки.

Доступ к любому элементу массива есть возможность получить по его индексу. Индексом может быть номер элемента или строка массива. Как правило, элементы в массиве указываются по номеру, причём нумерация начинается с нуля.

Для создания массива можно воспользоваться функцией array(). К примеру для создания массива с четырьмя элементами- «машина», «дом», «дача» и «гараж», мы должны действовать так:

```
$elements=array(“машина”, “дом”, “дача”, “гараж”);
```

Для вывода элемента массива на экран следует сделать следующее:

```
echo  $elements[2];
```

Программа выведет на экран слово «дача», т.к. хоть он и является третьим элементом, но его порядковый номер равен двум.

Для добавления элемента в массив, необходимо просто присвоить ему значение:

$elements[]=“яхта”;

Следует обратить внимание, что мы не указываем номер нового элемента в массиве, PHP сам вычисляет его.

В PHP существует ещё 2 типа массивов, это ассоциированный и многомерный. Доступ к элементам массива по номеру удобен тогда, когда вам нужно выбирать их в том порядке, в котором они были созданы, или при сортировке массива. Однако бывает необходимо обратиться к элементу массива по его имени.

Ассоциированный массив – это массив, к элементу которого можно обратиться по имени. В других языках программирования аналогичные массивы называют структурами. Разница между ассоциированным массивом и обычным в языке PHP не является принципиальной. Эти массивы не являются объектами разных типов как в Perl. Однако всё же следует обращаться с ними по-разному, потому что они требуют различного подхода и стратегии. Ассоциированный массив также можно создать с помощью функции array(). Для того чтобы это сделать, нужно задать как имя, так и значение для каждого элемента. Фактически массив может состоять из значений, объектов или даже массивов. Многомерный массив как раз и состоит из нескольких массивов, т.е. каждый его элемент является в свою очередь массивом. Для того чтобы обратиться ко второму элементу первого массива нужно написать так:

echo $array[0][1]

Использование многомерных массивов позволяет нам относительно просто создавать довольно сложные структуры данных. Например в следующем примере мы создаём массив, каждый элемент которого является ассоциированным массивом.

```
$characters=array(
                array(
name=>”bob”,
                occupation=>”superhero”,
                age=>”30”,
                “special power”=>”x-ray vision”),
                array(
name=>”sally”,
                occupation=>”superhero”,
                age=>”24”,
                “special power”=>”superhuman strenght”),
                array(
name=>”mary”,
                occupation=>”teacher”,
                age=>”63”,
                “special power”=>”none”),
                );
echo $characters[0][occupation];
```

Данная структура выведет на экран значение строки «occupation» первого ассоциированного массива. Разобравшись с логикой данной программы, мы сможем создавать сложные комбинации массивов, как простых, так и ассоциированных.

Работа с файлами и файловой системой в PHP

Файл представляет собой последовательность байтов, хранящуюся на каком-либо физическом носителе информации. Каждый файл имеет абсолютный путь, по которому определяется его местонахождение. В качестве разделителя пути в Windows может использоваться как прямой (/), так и обратный (\) слеш. В других операционных системах используется только прямой слеш. Открытие файлов в файловой системе сервера производится при помощи функции fopen:

int fopen(*string* filename, *string* mode *[, int* use_include_path*])*

Первый аргумент filename - имя файла или абсолютный путь к нему. Если абсолютный путь не указывается, то файл должен находится в текущем каталоге. Второй аргумент mode говорит о том, для каких действий открывается файл и может принимать следующие значения:

- *r* (Открыть файл только для чтения; после открытия указатель файла устанавливается в начало файла);
- *r+* (Открыть файл для чтения и записи; после открытия указатель файла устанавливается в начало файла);
- *w* (Создать новый пустой файл только для записи; если файл с таким именем уже есть вся информация в нем уничтожается);
- *w+* (Создать новый пустой файл для чтения записи; если файл с таким именем уже есть вся информация в нем уничтожается);
- *a* (Открыть файл для дозаписи; данные будут записываться в конец файла);
- *a+* (Открыть файл для дозаписи и чтения данных; данные будут записываться в конец файла);
- *b* (Флаг, указывающий на работу (чтение и запись) с двоичным файлом; указывается только в Windows).

Третий необязательный аргумент use_include_path определяет должны ли искаться файлы в каталоге include_path. В случае удачного открытия файла, функция *fopen* возвращает дескриптор файла, в случае неудачи - *false*. Дескриптор файла представляет собой указатель на открытый файл, который используется операционной системой для поддержки операций с этим файлом. Возвращенный функцией дескриптор файла необходимо затем указывать во всех функциях, которые в дальнейшем будут работать с этим файлом. Код, приведенный ниже, открывает файл file.txt для чтения:

```
<?
 $file = fopen("c:/www/html/file.txt","r");
 if(!file)
```

```
  {
    echo("Ошибка открытия файла!");
  }
?>
```

Содержимое открытого файла можно отобразить в браузере с помощью функции fpassthru:

int fpassthru *(int* file*)*

Аргумент file представляет собой дескриптор файла.

```
<?
 $file = fopen("c:/www/html/pavlovo.jpg","rb");
 if(!file)
  {
    echo("Ошибка открытия файла!");
  }
  else
  {
    fpassthru($file);
  }
?>
```

Для текстовых файлов существует еще одна функция отображения readfile:

readfile (string *filename*)

в качестве аргумента эта функция принимает имя файла, а не его дескриптор:

```
<?
 readfile ("file.txt");
?>
```

Закрытие файлов осуществляется с помощью функции fclose:

int fclose *(int* file*)*

Аргумент file представляет собой дескриптор файла, который необходимо закрыть.

2.3. PHP и Yii Framework

Спектр возможностей языка PHP чрезвычайно широк. Существуют три основных области, где используется PHP:

- создание сценариев (скриптов), для выполнения на стороне сервера;
- создание скриптов для выполнения в командной строке;
- создание оконных приложений, выполняющихся на стороне клиента.

Однако наибольший интерес представляет именно первая область, так как PHP является языком web-программирования. Об этом свидетельствует то, что PHP поддерживает работу по таким протоколам, как LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (на платформах Windows) и многим другим. Это выгодно отличает его от множества существующих на сегодняшний день языков, таких как C/C++, Fortran, Perl, TCL, Unix Schell, Visual Basic, Apple Script, и других подобных языков.

Данный параграф посвящен использованию PHP непосредственно для построения сайтов и взаимодействия с Web. В ней раскрывается вопрос о том, как легко модифицировать содержимое web-страниц и осуществлять навигацию в Web при помощи ссылок и различных стандартных функций. Данный параграф дополнит изложенный материал - в ней подробно рассматриваются средства взаимодействия с пользователем в формах HTML. Здесь также рассматриваются нетривиальные аспекты web-программирования на PHP.

Простые ссылки

По ссылкам пользователь может переходить как на обычные страницы HTML, так и на страницы, содержащие код PHP:

<View today's date

Если щелкнуть на ссылке, в браузере будет загружена страница с именем date.php. Развивая приведенный пример, можно воспользоваться переменной для построения динамической ссылки:

```
<?
$link = "date.php";
print "<a href = \"$link\">View today's date</a> <br>\n"
?>
```

Вероятно, у вас возник вопрос — почему в коде ссылки перед кавычками (") ставится обратная косая черта (\)? Дело в том, что кавычки в PHP являются специальными символами и используются в качестве ограничителей строк [10].

Файловые компоненты (шаблоны)

Шаблоном (применительно к web-программированию) называется часть web-документа, который используется в нескольких страницах. Шаблоны, как и функции PHP, избавляют от лишнего копирования/вставки фрагментов содержания страницы и программного кода. С увеличением масштабов сайта значение шаблонов возрастает, поскольку они позволяют легко и быстро проводить модификации на уровне целого сайта.

Как правило, общие фрагменты содержания/кода (то есть шаблоны) сохраняются в отдельных файлах. При построении web-документа вы просто «включаете» эти файлы в соответствующие места страницы. В PHP для этого существуют две функции: include( ) и require( ).

Одним из самых выдающихся аспектов PHP является возможность построения шаблонов и программных библиотек и их последующей вставки в новые сценарии. Применение библиотек экономит время и усилия по использованию общих функциональных возможностей на разных web-сайтах. Включение одного или нескольких файлов в сценарий осуществляется стандартными функциями PHP require( ) и include( ).

Функции

В PHP существуют четыре функции для включения файлов в сценарии PHP:

- include( );
- include_once( );

- require( );
- require_once( ).

Несмотря на сходство имен, эти функции решают разные задачи.

include( )

Функция include( ) включает содержимое файла в сценарий. Синтаксис функции include( ):

include (file файл]

У функции include( ) есть одна интересная особенность — ее можно выполнять условно. Например, если вызов функции включен в блок команды if, то файл включается в программу лишь в том случае, если условие if истинно. Если функция include( ) используется в условной команде, то она *должна* быть заключена в фигурные скобки или в альтернативные ограничители.

include_once( )

Функция include_once( ) делает то же, что и include( ), за одним исключением: прежде чем включать файл в программу, она проверяет, не был ли он включен ранее. Если файл уже был включен, вызов include_once( ) игнорируется, а если нет — происходит стандартное включение файла. Во всем остальном include_once( ) ничем не отличается от include( ). Синтаксис функции include_once( ):

include_once (file файл)

require ( )

В целом функция require( ) похожа на include( ) — она тоже включает шаблон в тот файл, в котором находится вызов require( ). Синтаксис функции require( ):

require (file файл)

Тем не менее, между функциями require( ) и include( ) существует одно важное различие. Файл, определяемый параметром require( ), включается в сценарий независимо от местонахождения require( ) в сценарии. Например,

при вызове requi re( ) в блоке if при ложном условии файл все равно будет включен в сценарий!

Во многих ситуациях бывает удобно создать файл с переменными и другой информацией, которая используется в масштабах сайта, и затем подключать его по мере необходимости. Хотя имя этого файла выбирается произвольно, я обычно называю его init.tpl (сокращение от «initializaion.template). С увеличением размеров сайта может оказаться, что некоторые файлы включаются в сценарий по несколько раз. Иногда это не вызывает проблем, но в некоторых случаях повторное включение файла приводит к сбросу значений изменившихся переменных. Если во включаемом файле определяются функции, могут возникнуть конфликты имен. Учитывая сказанное, мы приходим к следующей функции — require_once( ).

require_once( )

Функция require_once( ) гарантирует, что файл будет включаться в сценарий всего один раз. После вызова requi re_once( ) все дальнейшие попытки включения того же файла игнорируются. Синтаксис функции require_once( ): require_once(file файл)

Если не считать дополнительной проверки, в остальном эта функция аналогична **require( ).**

Построение компонентов

При определении структуры типичной web-страницы обычно разбивают ее на три части: заголовок, основную часть и колонтитул. Как правило, в большинстве правильно организованных web-сайтов присутствует заголовок, который практически не изменяется; в основной части выводится запрашиваемое содержание сайта, поэтому она часто изменяется; наконец, колонтитул содержит информацию об авторских правах и навигационные ссылки. Колонтитул, как и заголовок, обычно остается неизменным.

Формы

Получение и обработка данных, введенных пользователем, стали неотъемлемой частью большинства успешных web-сайтов. Бесспорно, возможности накопления статистики, проведения опросов, хранения персональных настроек и поиска выводят Web на принципиально новый уровень — без них эта среда обладала бы минимальной интерактивностью.

Ввод информации в основном реализуется с применением форм HTML. Как правило, пользователь заполняет в форме одно или несколько полей (например, имя и адрес электронной почты), нажимает кнопку отправки данных, после чего получает ответное сообщение.

При вводе данных в форму используются различные управляющие элементы. В одних элементах пользователь вводит информацию с клавиатуры, в других он выбирает нужный вариант, щелкая кнопкой мыши. В формах могут присутствовать скрытые поля, которые поддерживаются самой формой; содержимое скрытых полей не должно изменяться пользователем. Одна страница может содержать несколько форм, поэтому необходимы средства, которые позволяли бы отличить одну форму от другой. Более того, вы должны как-то сообщить форме, куда следует перейти, когда пользователь выполняет действие с формой (как правило, нажимает кнопку отправки данных). Обе задачи решаются заключением форм в следующие теги HTML:

<form action = действие method = "метод" - элементы формы -</form>

Как видно из приведенного фрагмента, в тегах форм указываются два важных элемента: действие и метод. *Действие* указывает, какой сценарий должен обрабатывать форму, а *метод* определяет способ передачи данных этому сценарию. Существует два метода:

- Метод get передает все данные формы в конце URL. Из-за различных ограничений, связанных со спецификой языков и длиной данных, этот метод применяется редко.

- Метод post передает все данные формы в теле запроса. Этот метод используется чаще, чем get[10].

Элементы форм, ориентированные на ввод с клавиатуры

Таких элементов всего два — текстовое поле (text box) и текстовая область (text area). В текстовых полях обычно вводится короткая текстовая информация — скажем, адрес электронной почты, почтовый адрес или имя. Синтаксис определения текстового поля:

<input type="text" nate="имя_переменной" size="N" maxlength="N" value="">

Определение текстового поля включает пять атрибутов:

- type — тип элемента (для текстовых полей — text);
- name — имя переменной, в которой сохраняются введенные данные;
- size — общий размер текстового поля в браузере;
- maxlength — максимальное количество символов, вводимых в текстовом поле;
- value — значение, отображаемое в текстовом поле по умолчанию.

Особой разновидностью текстовых полей является поле для ввода паролей. Оно работает точно так же, как обычное текстовое поле, однако вводимые символы заменяются звездочками. Чтобы создать в форме поле для ввода паролей, достаточно указать type="password" вместо type="text".

Текстовая область (text area) используется для ввода несколько больших объемов текста, не ограничивающихся простым именем или адресом электронной почты. Синтаксис определения текстовой области:

<textarea name="имя_переменной" rows="N" cols="N" value=""></textarea>

Определение текстового поля включает три атрибута:

- name — имя переменной, в которой сохраняются введенные данные;
- rows — количество строк в текстовой области;
- cols — количество столбцов в текстовой области[15].

Элементы форм, ориентированные на ввод с мыши

В других элементах форм пользователь выбирает один из заранее определенных вариантов при помощи мыши. Ограничимся рассмотрением флажков, переключателей и раскрывающихся списков.

Флажки (checkboxes) используются в ситуациях, когда пользователь выбирает один или несколько вариантов из готового набора — по аналогии с тем, как ставятся «галочки» в анкетах. Синтаксис определения флажка:

<input type="checkbox" name="имя_переменной" value="начальное_значение">

Определение флажка включает три атрибута:

- type — тип элемента (для флажков — checkbox);
- name — имя переменной, в которой сохраняются введенные данные (в данном случае — состояние элемента);
- value — значение, присваиваемое переменной по умолчанию. Если флажок установлен, именно это значение будет присвоено переменной с указанным именем. Если флажок не установлен, значение атрибута value не используется.

Переключатель (radio button) представляет собой разновидность флажка; он работает практически так же за одним исключением — в любой момент времени в группе может быть установлен лишь один переключатель. Синтаксис определения переключателя:

<input type="radio" name="имя_переменной" value="начальное_значение">

Как видно, синтаксис почти не отличается от определения флажка. Определение переключателя поля включает три атрибута:

- type — тип элемента (для переключателей — radio);
- name — имя переменной, в которой сохраняются введенные данные (в данном случае — состояние элемента);
- value — значение, присваиваемое переменной по умолчанию. Если переключатель установлен, именно это значение будет присвоено

переменной с указанным именем. Если флажок не установлен, значение атрибута value не используется.

Раскрывающиеся списки особенно удобны в ситуации, когда имеется длинный перечень допустимых вариантов, из которого пользователь должен выбрать один вариант. Как правило, раскрывающиеся списки применяются при работе с относительно большими наборами данных. Синтаксис определения раскрывающегося списка:

<select name="имя_переменной">
<option value="имя_переменной1 ">
<option value="имя_переменной2">
<option value="имя_переменной3">
<option value="имя_переменнойN">
</select>

Скрытые поля не отображаются в браузере и обычно используются для передачи данных между сценариями. Хотя передача в скрытых полях работает вполне нормально, в PHP существует другое, более удобное средство — сеансовые переменные. Впрочем, скрытые поля также используются в некоторых ситуациях и потому заслуживают упоминания.

Синтаксис определения скрытого поля практически идентичен синтаксису текстовых полей, отличается только атрибут типа. Поскольку скрытые поля не отображаются в браузере, привести пример на страницах книги невозможно. Синтаксис определения скрытого поля:

<input type="hidden" name="имя_переменной" value="начальное_значение">

Определение скрытого поля включает три атрибута:

- type — тип элемента (для скрытых полей — hidden);
- name — имя переменной, в которой сохраняются скрытые данные;
- value — значение, по умолчанию сохраняемое в скрытом поле.

Кнопка отправки данных инициирует действие, заданное атрибутом action тега <form>. Синтаксис определения:

<input type="submit" value="текст_на_кнопке">

Определение кнопки включает два атрибута:

- type — тип элемента (для кнопки отправки данных — submit);
- value — текст, по умолчанию отображаемый на кнопке.

Кнопка сброса отменяет все изменения, внесенные в элементы формы. Синтаксис определения:

<input type="reset" value=*" текст _на_кнопке"*>

Определение кнопки включает два атрибута:

- type — тип элемента (для кнопки сброса — reset);
- value — текст, по умолчанию отображаемый на кнопке[10].

Глава 3. ОПИСАНИЕ ПРОГРАММНОГО ОБЕСПЕЧЕНИЯ И ПОРЯДОК ЕГО ИСПОЛЬЗОВАНИЯ

3.1. Инфологическая модель WEB – сайта

Изучив предметную область и собрав исходные данные можно представить предполагаемую модель сайта. Взаимодействие пользователя с сайтом можно представить как технологию «клиент-сервер». Клиент в данном случае это пользователь сети Internet, который осуществляет доступ к серверу посредством браузера. Сервером в данном случае является сайт. WEB-сервер обрабатывает запросы браузера на получение WEB-страниц и отсылает ему требуемые данные. Обмен данными в сети Internet осуществляется на основе коммуникационного протокола TCP/IP и протокола более высокого уровня (приложений) HTTP.

Архитектура WEB-приложений с модулями расширения сервера может включать в себя стандартные модули расширения DLL библиотеки, реализующие технологию ASP. В функцию WEB-сервера входит обработка запросов WEB- браузеров пользователей сети, загрузка соответствующего модуля расширения сервера и передача ему параметров запроса. В результате обработки запроса модулями расширения сервера формируется WEB-документ с использованием HTML-шаблона. Готовый WEB-документ WEB-сервера отсылается обратно WEB-браузеру в формате протокола HTTP .

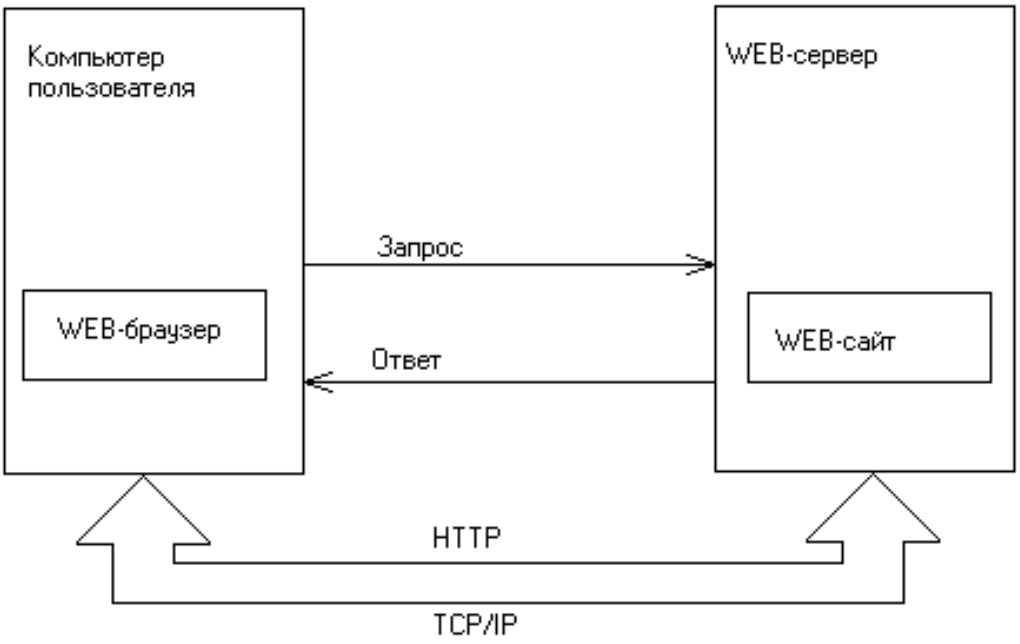

Рис.3.1. Схема функционирования WEB-приложения

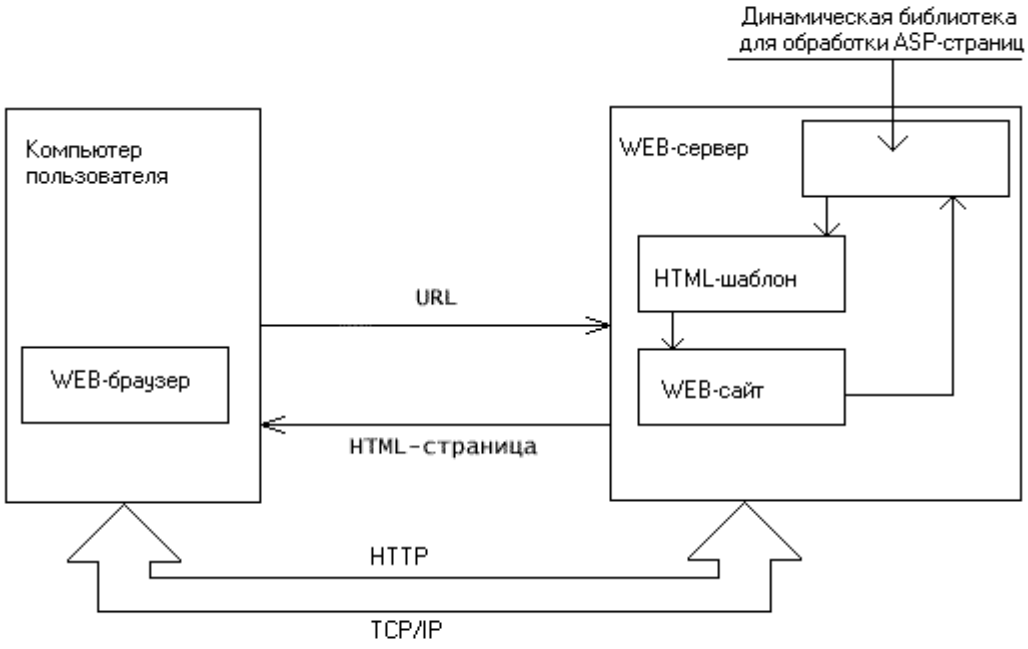

Рис.3.2.Архитектура WEB-приложения с модулем расширения сервера

Средства просмотра Web (например, приложение Internet Explorer) посылают запросы серверу Интернета, используя протокол HTTP. Сервер

Интернета отвечает документом, форматируемым на языке гипертекстовой разметки HTML.

WEB-сайт государственного гранта «Психолого-педагогические основы формирования профессиональной компетентности будущих специалистов (На базе профессиональной подготовки бакалавров и магистров ТУИТ и его филиалов)» организован из 4 основных, 6 тематических, 2-х подключаемых и множества информационных страниц

Основные страницы.

Index.php – главная страница содержит общую информацию о WEB-сайте, который содержить кнопки «Студенты», «Преподаватели», «Факультет и группы», «Настройки».

Подключаемые страницы.

Cap.htm – страница формирующая «шапку» со ссылками по сайту на всех основных страницах.

Bot.htm – страница, формирующая нижнюю часть экрана со ссылками по сайту на всех основных страницах.

Bot_2.htm – страница, формирующая нижнюю часть экрана со ссылками по сайту.

Скрытые страницы.

Order_view.html – страница, пополняющаяся в результате выполнения запросов на исследование по компетентности.

Repair_view. html – страница, пополняющаяся в результате выполнения заявок на мониторинг и тренинг по компетентности.

3.2. Описание программного обеспечения

Система имеет 4 типа пользователя, каждый из которых имеет свой доступ к системе и эти доступы назначаются администратором. В систему могут входить только ограниченные пользователи: сам администратор, студент, родитель студента, преподаватель.

Администратор
Администратор имеет доступ к настройки и конфигурации коммутаторов с помощью простых меню установок

Учитель
Учитель видит только студентов в его / ее класса и имеет доступ к посещаемости, зачетную книжку и приемлемости особенностей

Студент
Студент может видеть только свою собственную информацию

Родитель
Родитель видит своего ребенка или детский информацию вместе с автоматические уведомления отправлены по электронной почте

Отроем сайт webschool.tuit.uz

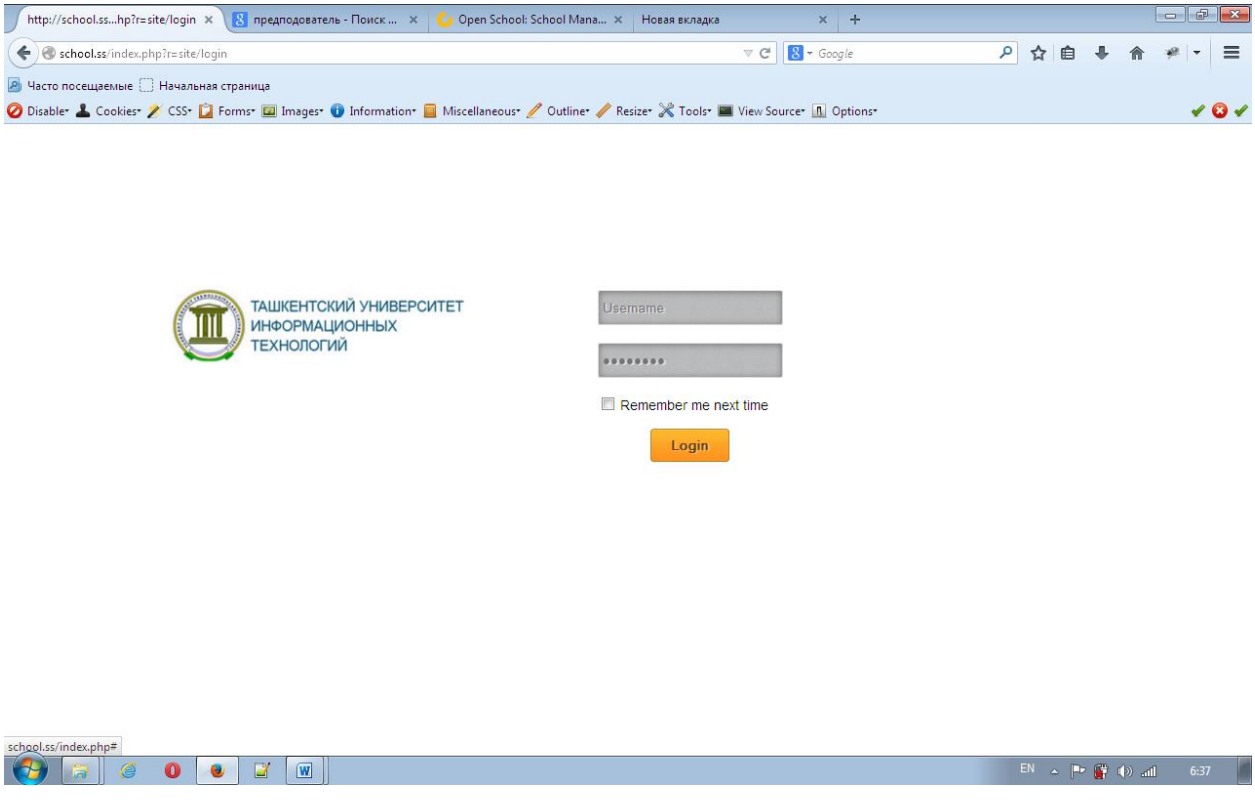

рис. 1

Пользователи которые зарегистрированы администратором могут использовать свой пароль и логин при входе на сайт.

Зайдем с именем администратора или же с любого зарегистрированного пользователя увидим первую страницу.

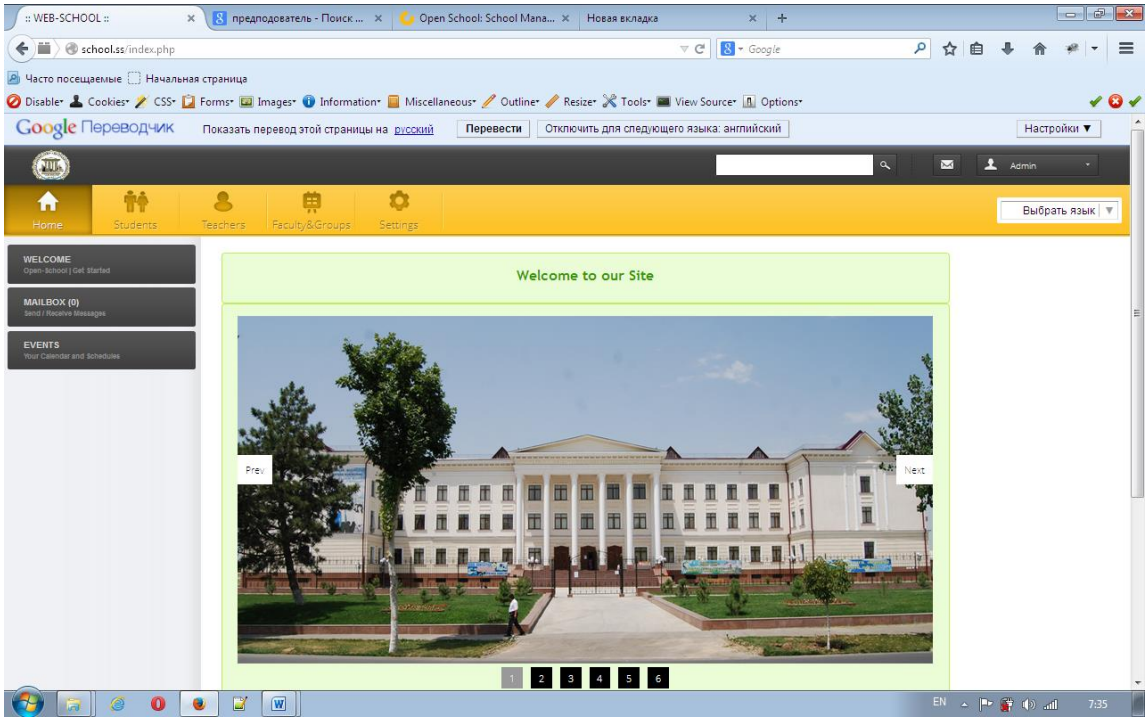

рис. 2

На верхнем меню видно пять разделов: *home*, *student*, *teacher*, *faculty* and *group* и последний *settings*.

В разделе *home* то есть домашняя страница для каждого пользователя существует этот раздел. Тут пользователь может отправить письмо и получать письма от другого пользователя, имеется календарь и список событий.

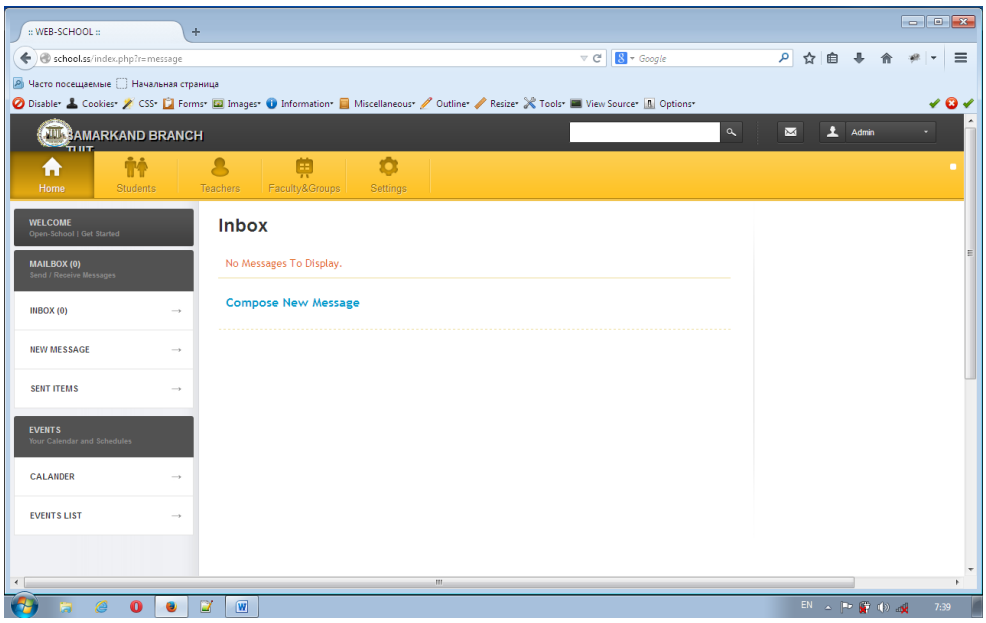

рис. 3

В списке событий будет видно в каком числе будет экзамен, контрольная работа, встречи или же мероприятия.

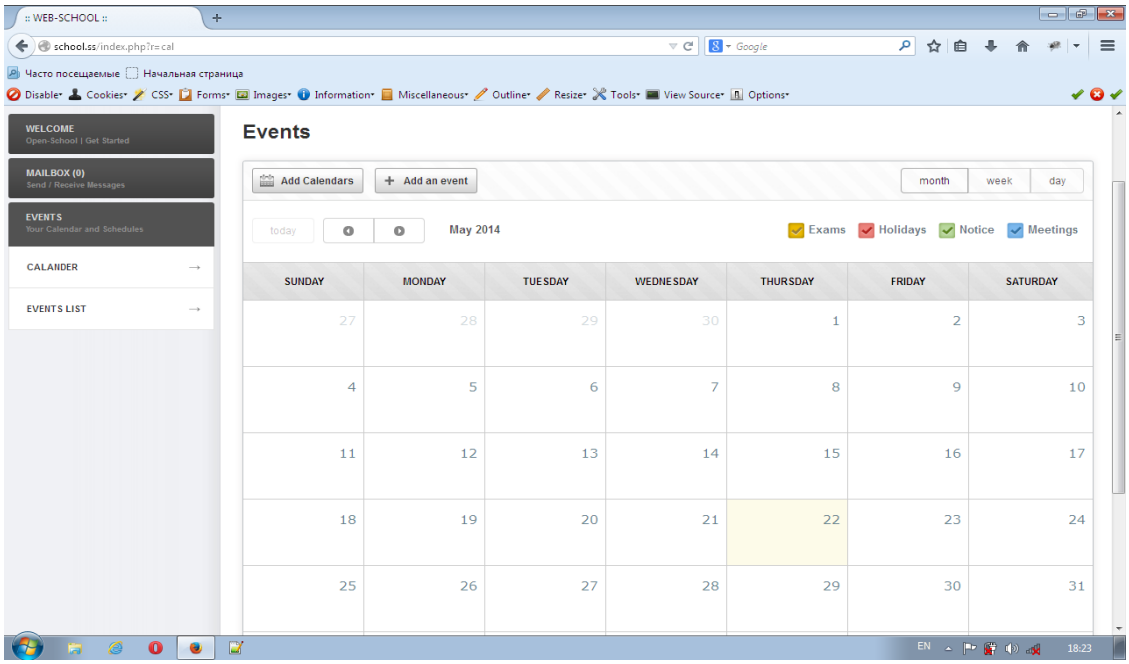

рис. 4

Если мы зашли с именем администратора тогда мы может добавить события выбрав какой не будь число из календаря. Например, пусть будет у нас в 14.06.2014 экзамен и на верхней части есть раздел massage тут надо сообщение для пользователей и описание события, указываем с кого часа до кого часа. В последнем можно указать all days если мы хотим чтобы наша

событие продлилась несколько дней, галочку поставим на editable если мы в дальнейшем редактируем наше событие. Когда закончим можно отметить дни, сдвинув событие смотрим рисунок 5-6.

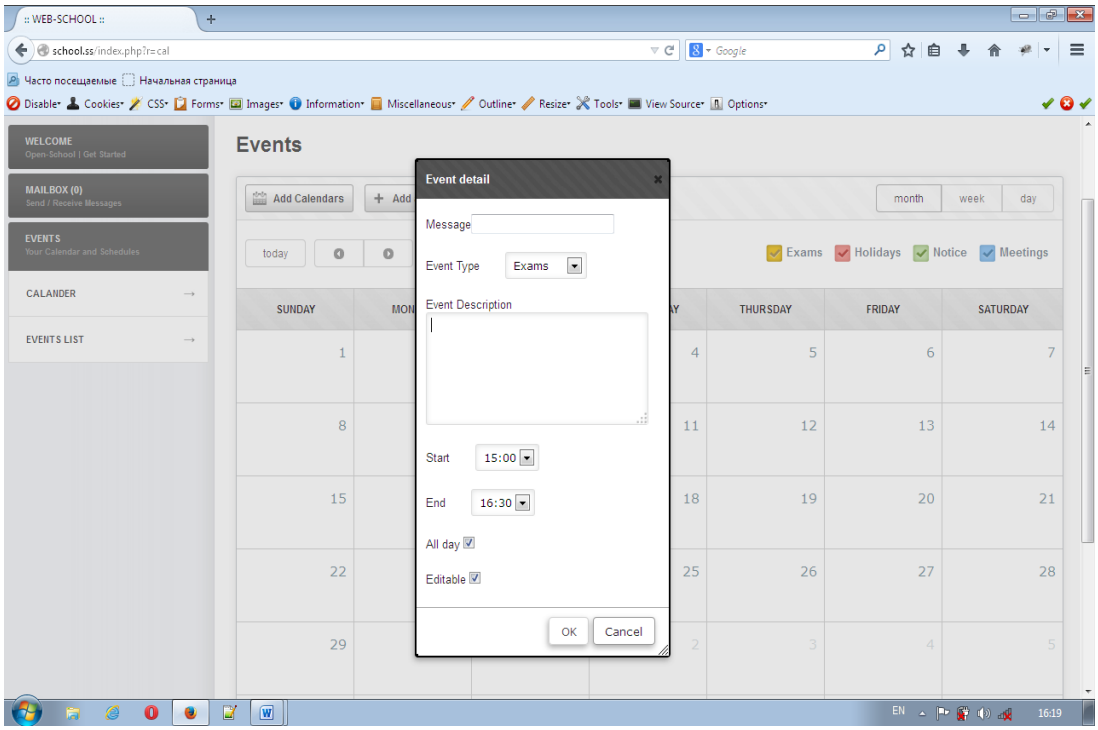

рис. 5

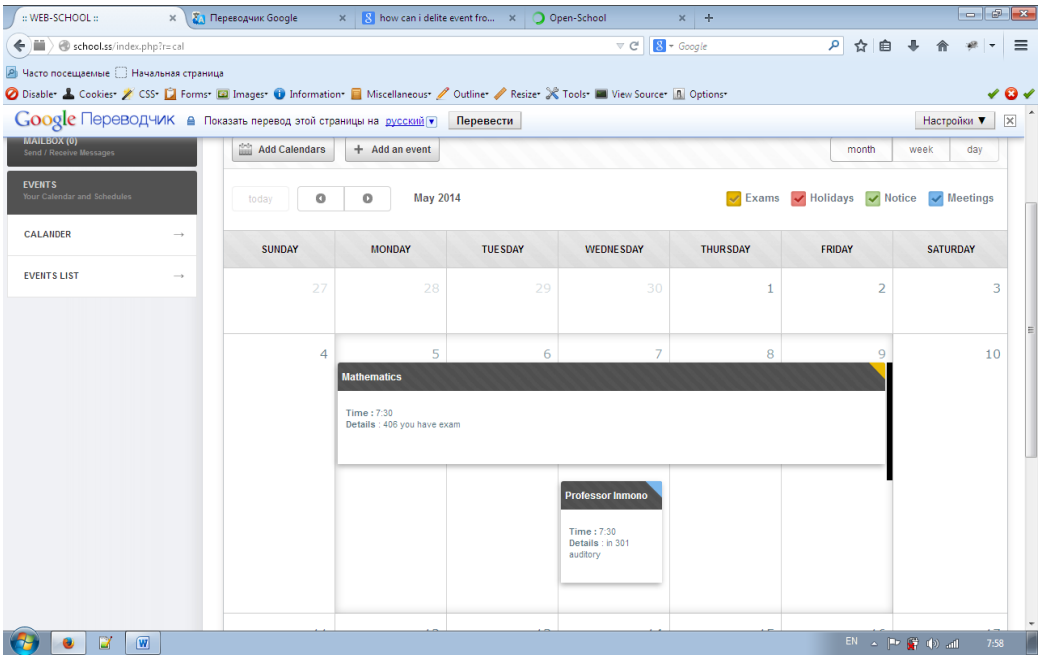

рис. 6

Переходим из меню с первого раздела на второй раздел. Этот раздел для студентов. В разделе управляется студенты и тут можно добавить студента, можно увидит список учащих студентов, набор студентов, отчисленных студентов, список родителей студентов, полная информация студентов и т.д. Кроме этого его оценки и посещаемость студента показана на этом странице.

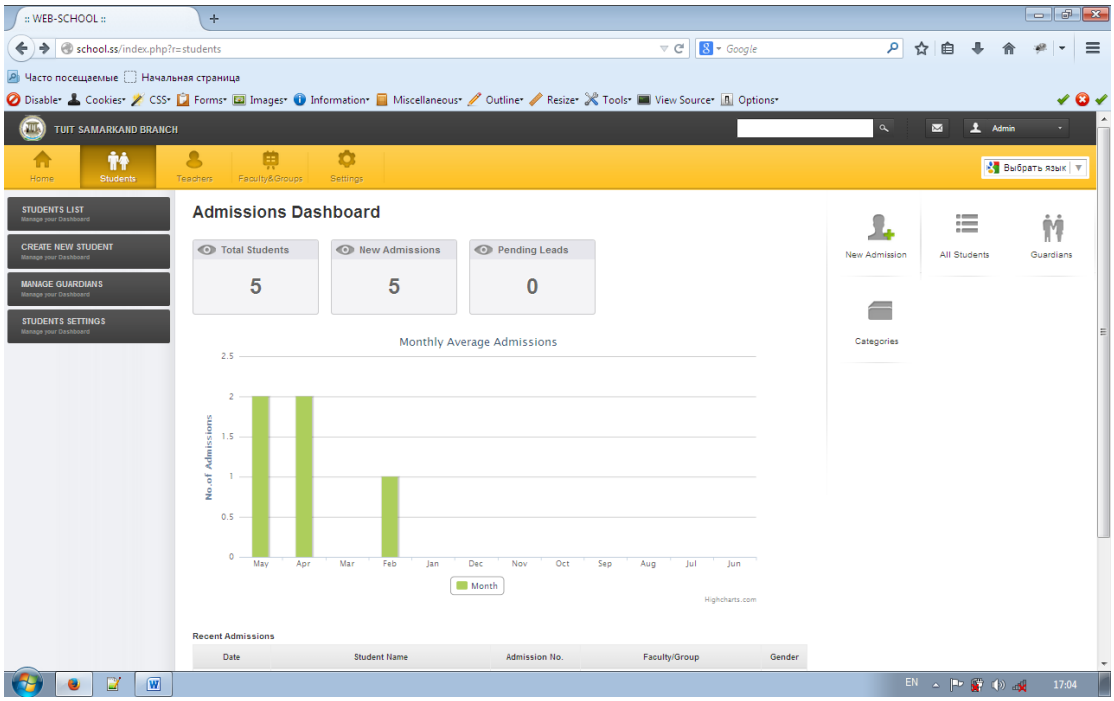

рис. 7

На верхнем правом углу видно *New admission*, нажав на него добавляем студента.

Вводим все его данные в положенные под полной информации понимается все данные, начиная с имени до группа кровы, где он учился до этого, его религия и после этого данные родителя, контактные и его средняя заработная плата.

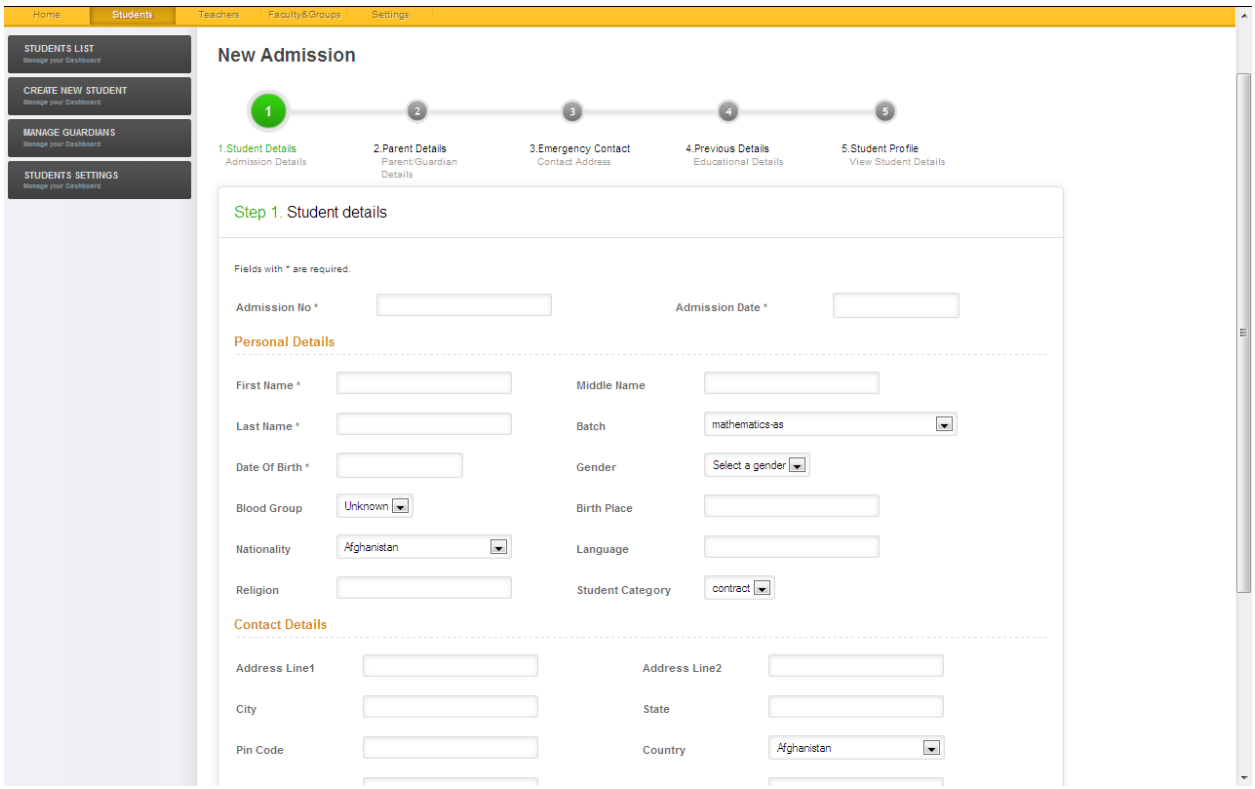

рис. 8

Страница студентов по написанным словам и рисункам можно понять без проблем, теперь переходим на следующую страницу, страница преподавателей, вообще можно было назвать этот раздел страницей сотрудников (рис. 9).

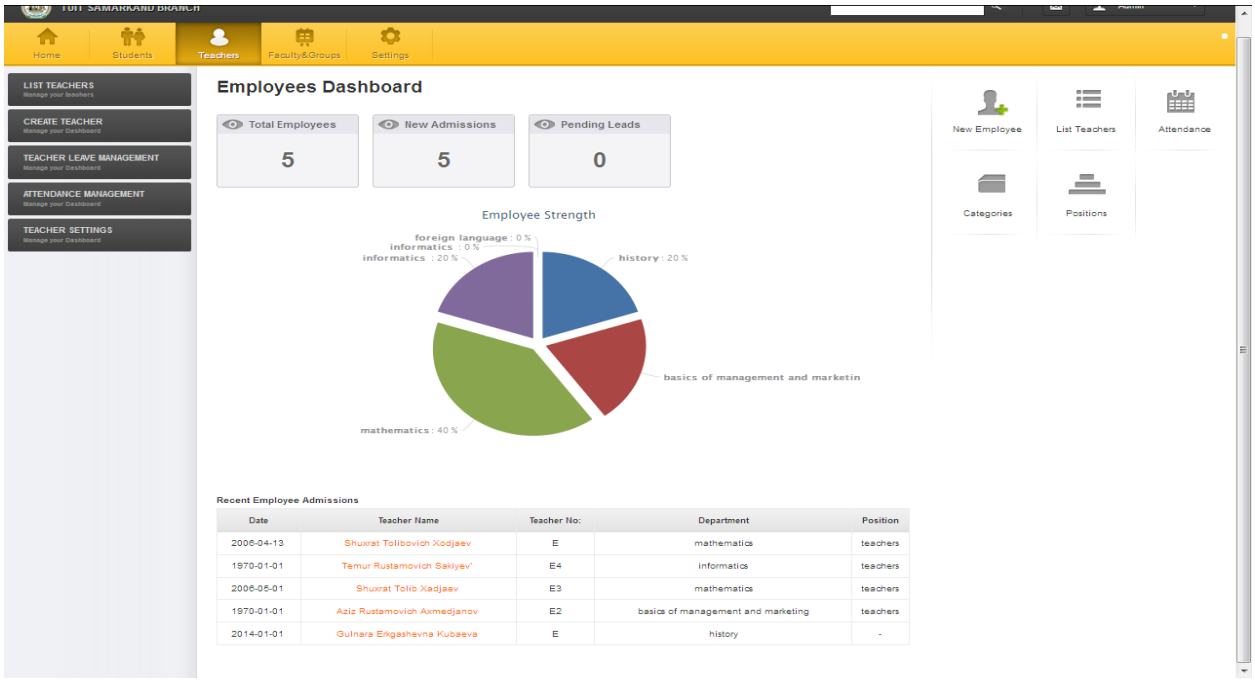

рис. 9

На этом странице управляет преподавателей, даёт полную информацию о них даже посещаемость преподавателей в каких числа х они отсутствовали. В этом разделе с лево видно много под разделов:

Нажимая Listteachers мы можем увидит список преподавателей и по каким предметам они преподают, если выберем какого не будь преподавателя то получим полную информацию о нем.

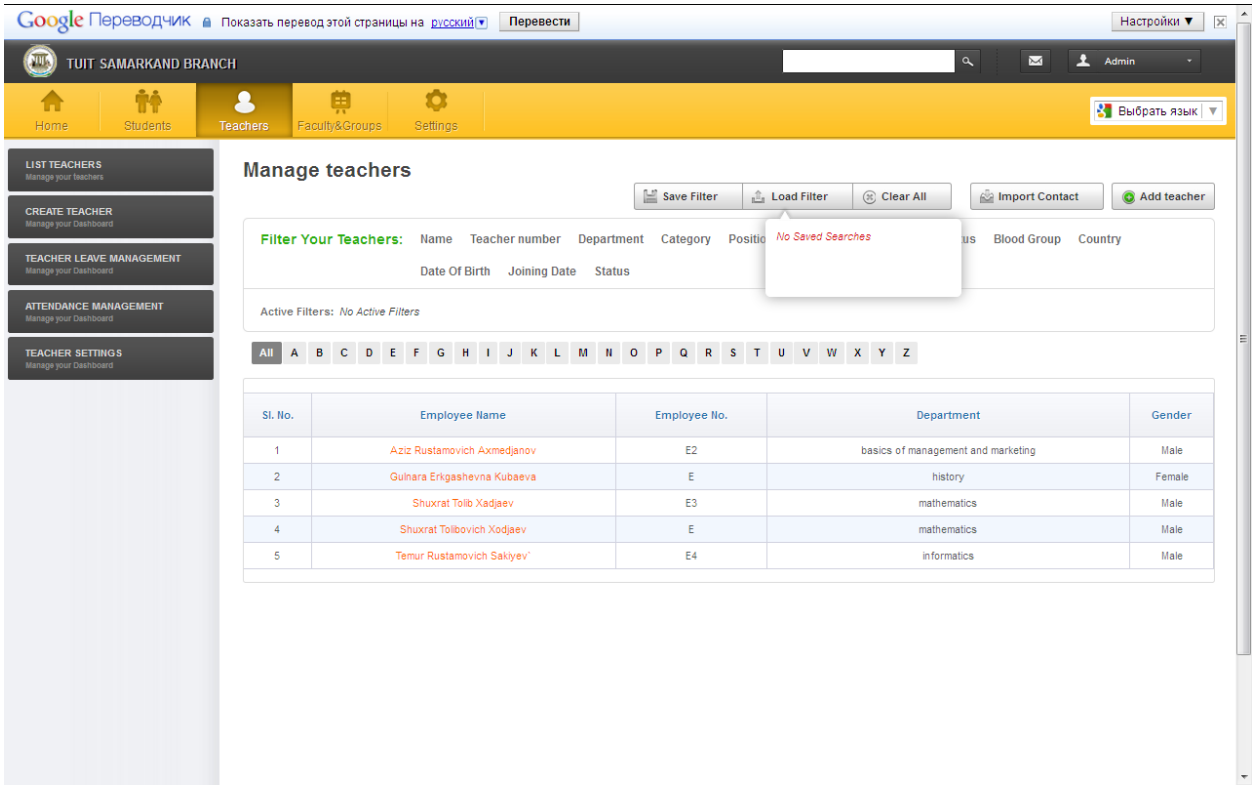

рис. 10

Пока в сайте добавлено только 5 учителей, но в дальнейшем тут минимум 20 имен сотрудников этого ВУЗа будет. Во втором подразделе можно добавлять сотрудника или преподавателя и вообще всех, кто работает на этом университете. Этот раздел легко и всем понятен без проблем можно заполнить анкету добавляющего пользователя,

Теперь переходом на третий подраздел Teacher Leave Management, то есть Управление Отпуском учителей. Когда нажимаем тогда выйдет add leave type то есть тип добавления типа отпуска. Короче тут можно добавить типы отпусков учителей и работников, например отпуск по беременности и родам, трудовой отпуск за проработанные время, ежегодный дополнительный

оплачиваемый отпуск, ежегодный основной оплачиваемый отпуск, отпуск с последующим увольнением и т.д.

Заполняем нужные нам места и добавим тип отпуска.

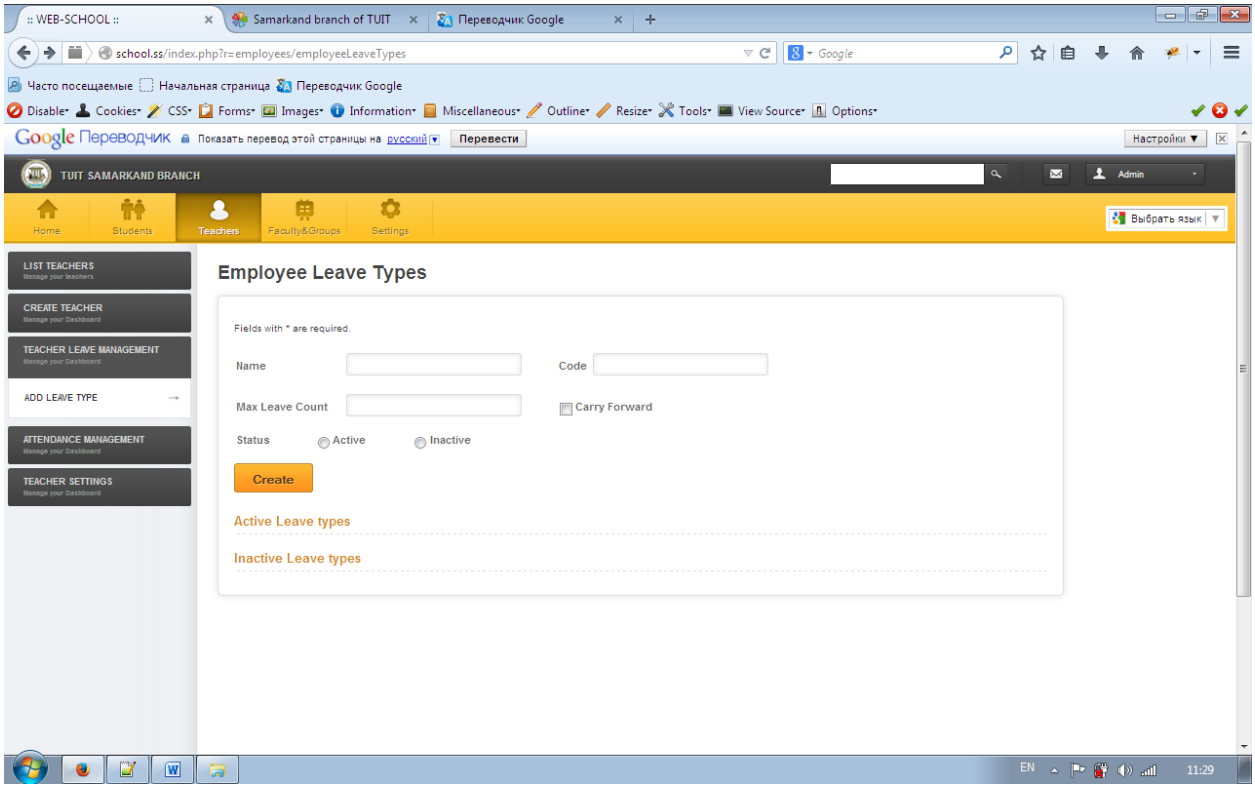

рис. 11

Следующий подраздел называется Attendance Register (запись не присутствие преподавателя, или же выходные дни преподавателя и работника). Чтобы добавить выходного дня какому не будь преподавателя сначала надо выбирать ее предмет потом из списка найти его и отметим за месяц какие же дни он не будет присутствовать на парах посмотрим рисунок 12.

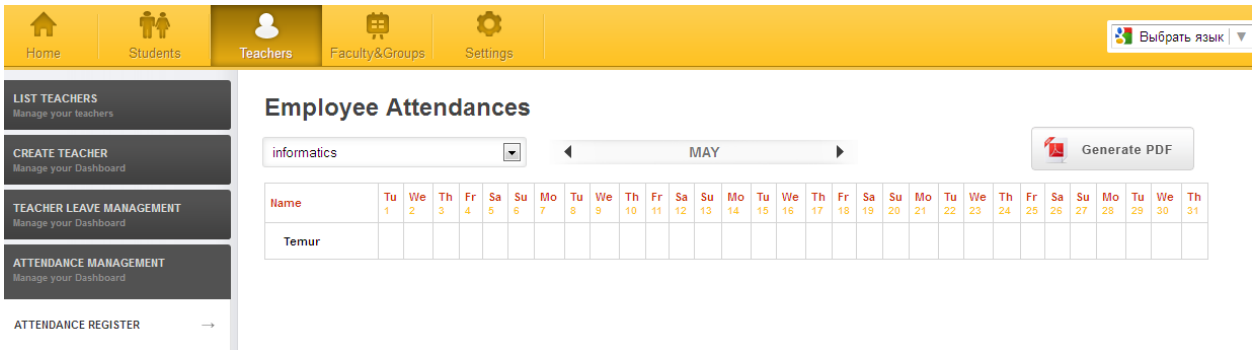

рис. 12

Из календаря выберем какое число тогда у нас выйдет вот так например я выбрал 1 мая (смотрим рис. 13)

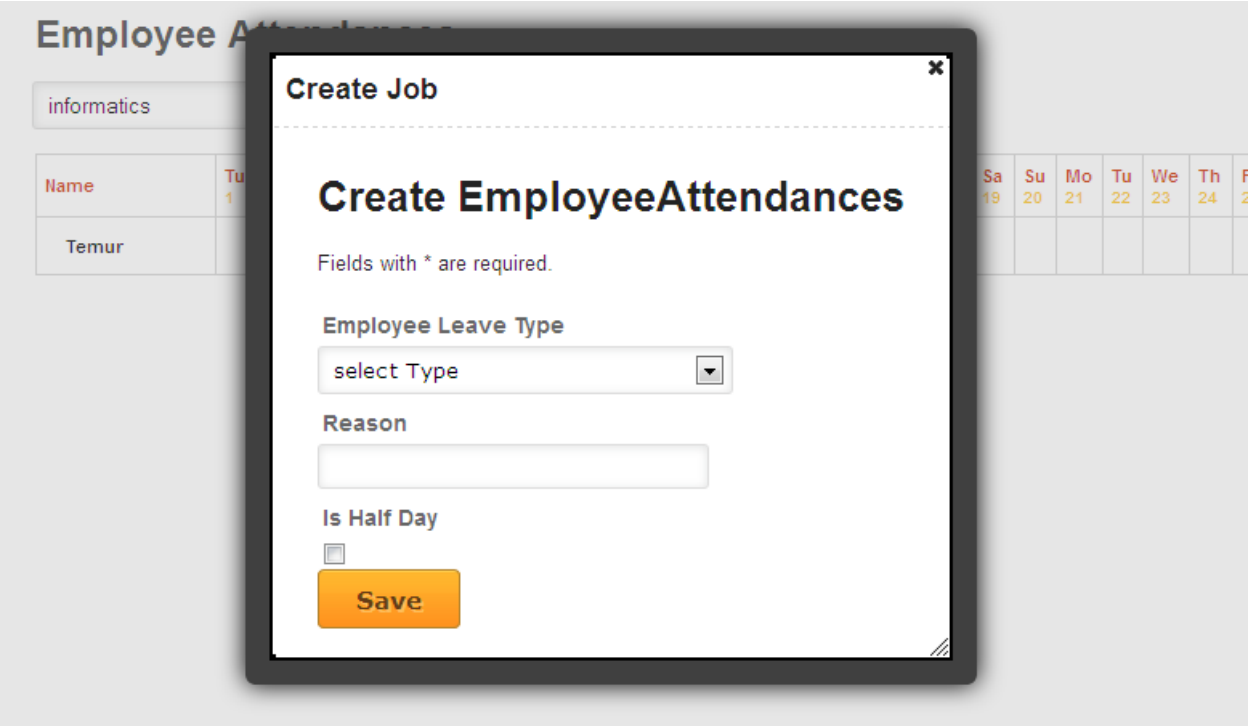

рис. 13

Укажем тип отсутствия и причину если проставим галочку на is half day то можно понят что работник или преподаватель будет отсутствовать пол дня.

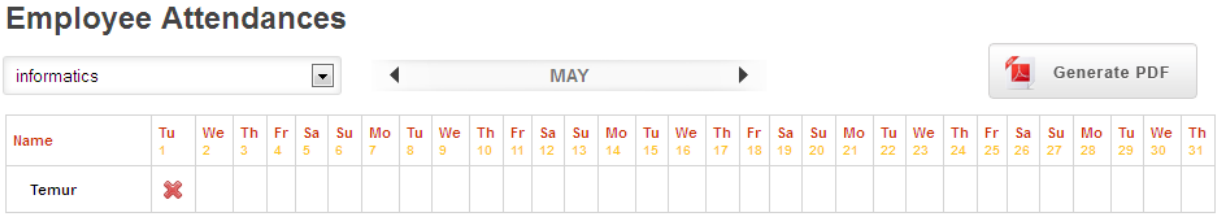

рис. 14

Переходим на другой подраздел и тут мы увидим Subject Association тема объединение то есть, какой не будь преподаватель прикрепляется какому не будь факультету чтобы по глубже провести свои пары.

Следующий подраздел *Manage Employee Categories* то есть управления категориями работников.

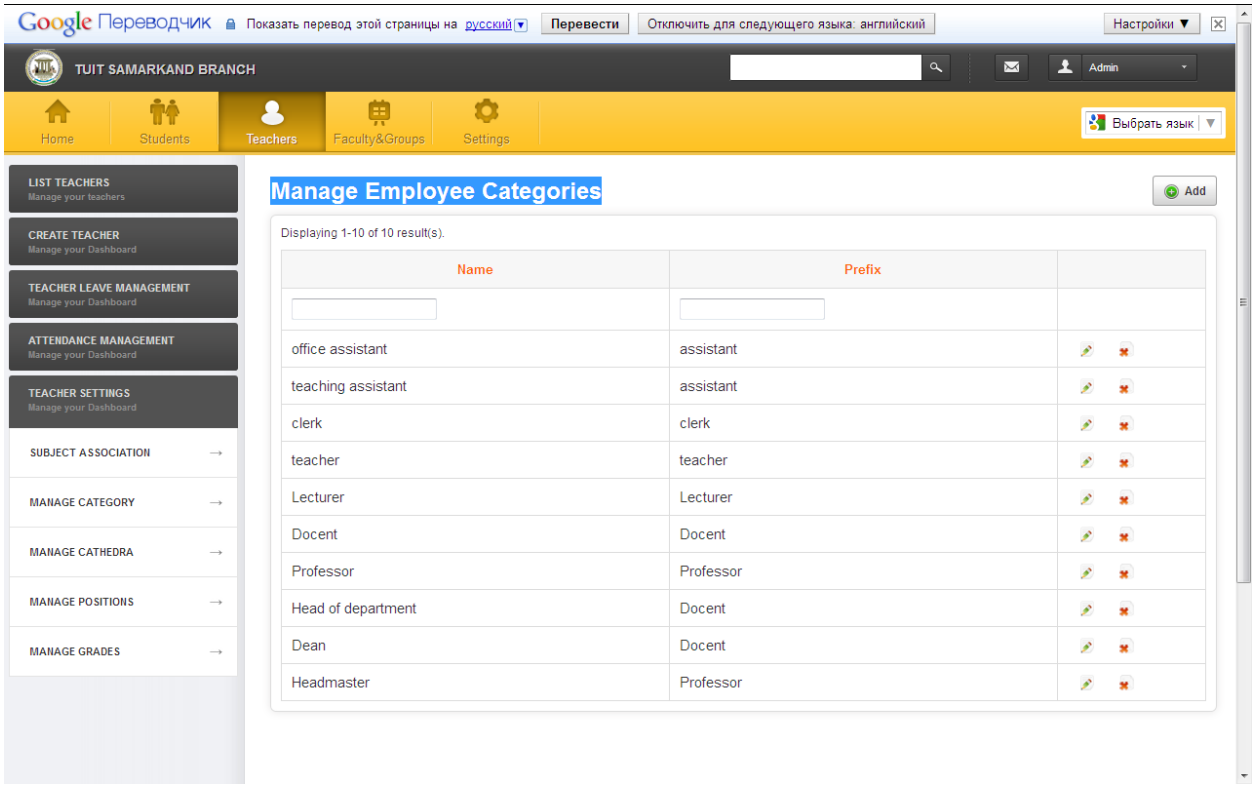

рис. 15

Тут показано список работников ВУЗа и их категории, можно добавить или удалить какую не будь категорию или же редактировать его.

Когда откроем из главной меню раздел учителя то нам на глаза попадется диаграмма. Диаграмма указывает численный состав преподавателей по проценту. Кроме этого тут на верху информация как сколько тут всего работников, новые приемы и ожидании приема сотрудника.

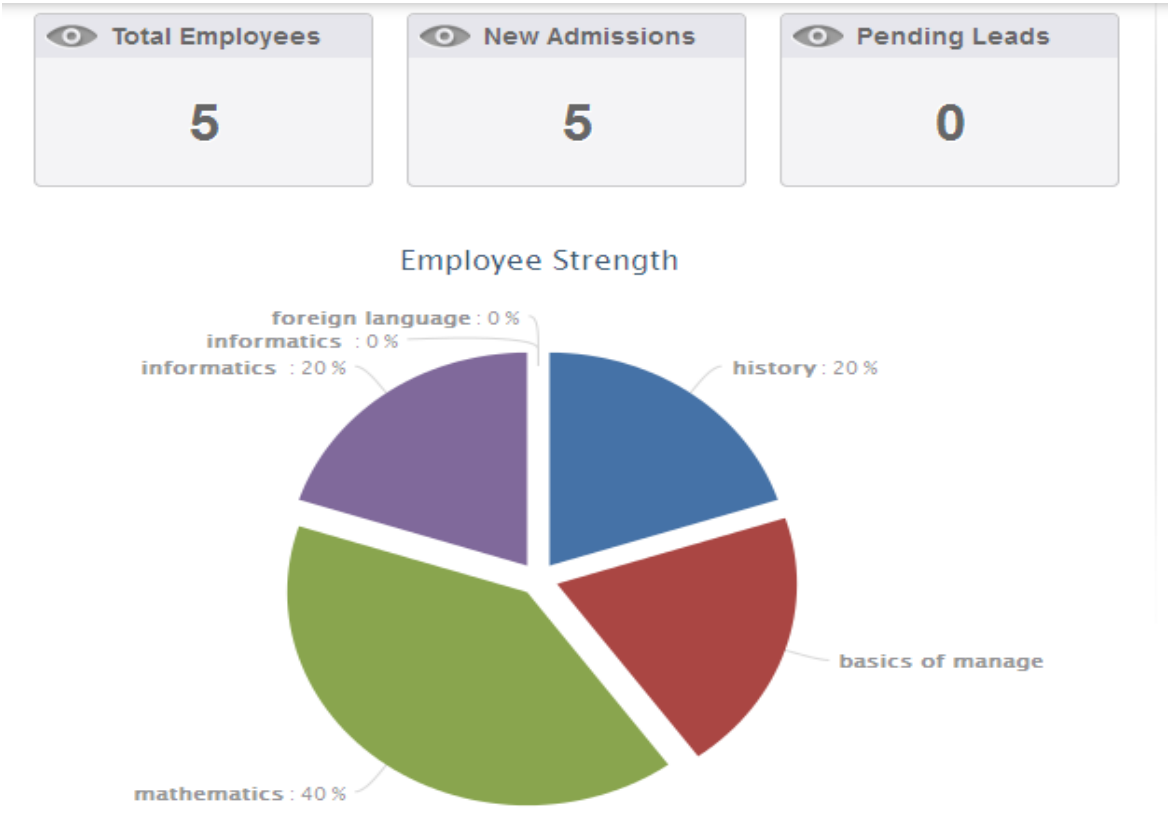

рис. 15

Из меню переходим на страницу факультет и группы. В этой странице по факультетам раздели группы и студенты, выбираем факультет нажимаем на его и выйдет группы которые принадлежать этому факультету. Нажимаем на выбранную группу увидим список учащихся студентов в этой группе. В этом разделе дана полная информация, о группе начиная с оценок до посещаемости и расписания. Кроме всех этих можно распечатать оценок студентов, посещаемости студентов, расписания, короче можно распечатать всю информацию о группе, что облегчает работу деканата. Смотрим рисунки 16-17.

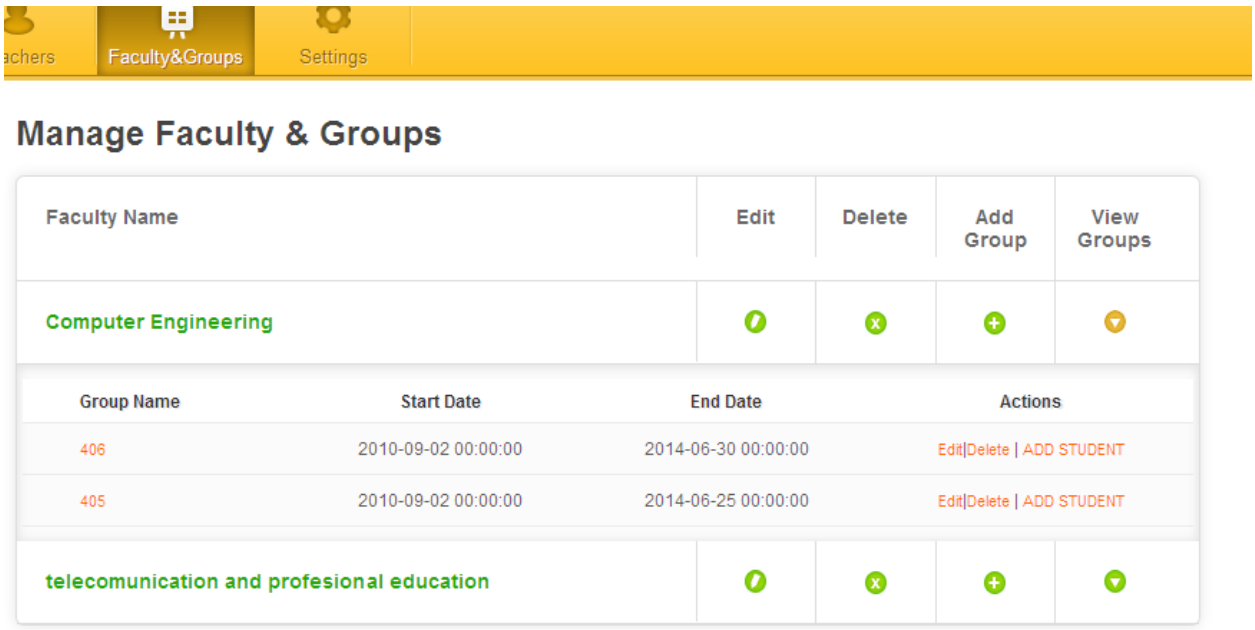

рис 17

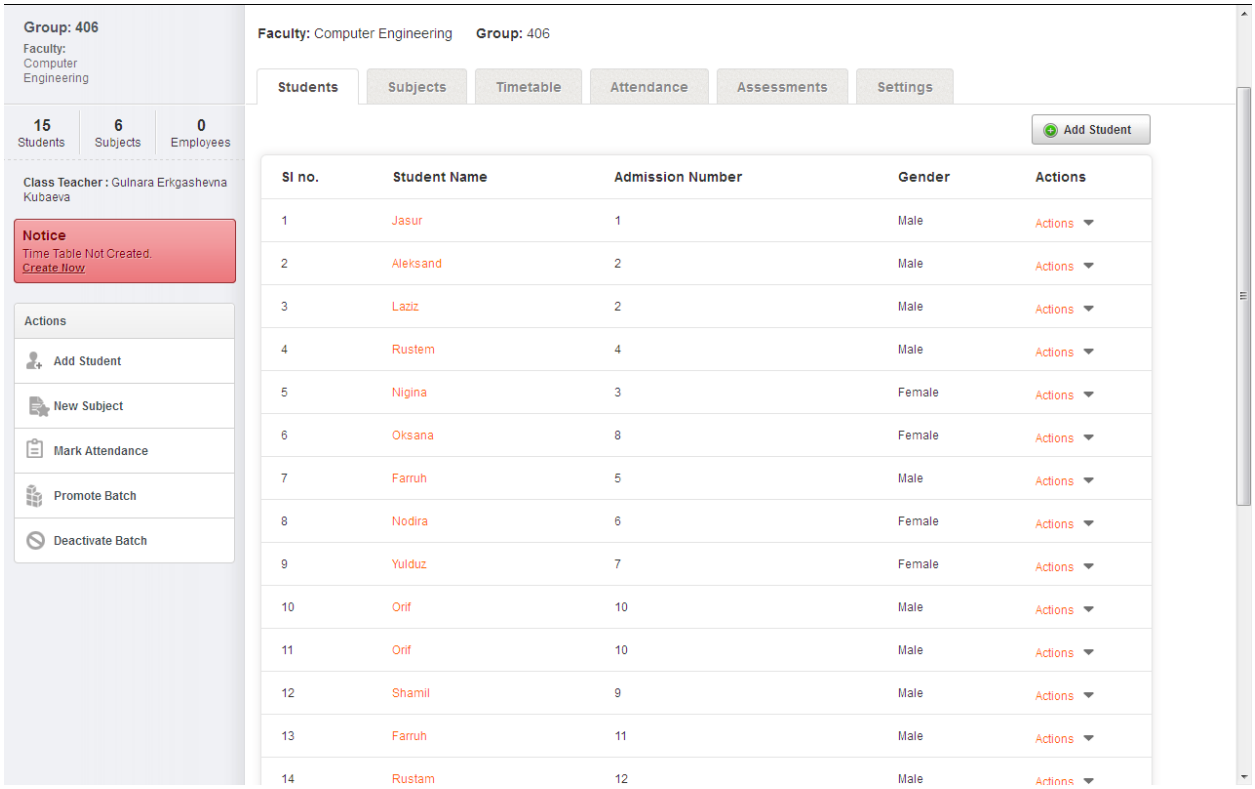

рис. 17

3.3. Обеспечение безопастности жизнедеятельности при работе на компьюторе

При производстве любого изделия возникает достаточно большой спектр ОВПФ, зависящий о технологии, организации, применяемых

материалов и т.д. Однако, при правильной организации производства и контроле за состоянием окружающей среды количество возникающих ОВПФ уменьшается. Весь технологический процесс должен быть построен в соответствии с ГОСТ 12.3.002-75 «Процессы производственные. Общие требования безопасности».

Провести анализ всех возможных проявлений ОВПФ не представляется возможным в рамках данной работы, поэтому ограничимся рекомендациями по устранению некоторых наиболее опасных на данном производстве ОВПФ. Кроме того, в связи со всё возрастающим использованием ЭВМ в производственных процессах, становится актуальным вопрос о безопасности работ с применением ПЭВМ.

При организации работ на ПЭВМ необходимо учитывать особенность данного процесса. Режимы труда и отдыха при работе с ПЭВМ должны организовываться в зависимости от вида и категории трудовой деятельности.

Виды трудовой деятельности разделяются на три группы:

- группа А – работа по считыванию информации с экрана ПЭВМ с предварительным запросом;
- группа Б – работа по вводу информации;
- группа В – творческая работа в режиме диалога с ЭВМ.

Время регламентированных перерывов в течение рабочий смены следует устанавливать в зависимости от ее продолжительности и категории тяжести (см. табл. 1).

Таблица 1

Категория работы	Уровень нагрузки за рабочую смену			Суммарное время регламентированных перерывов, мин.	
	группа А, количество знаков	группа Б, количество знаков	группа В, час	при 8-ми часовой смене	при 12-ти часовой смене
I	до 20000	до 15000	до 2	30	70
II	до 40000	до 30000	до 4	50	90
III	до 60000	до 40000	до 6	70	120

Продолжительность непрерывной работы на ПЭВМ без перерыва не должна превышать 2 часов.

На данном предприятии при 8- часовой рабочей смене производятся работы группы В. Суммарное время регламентированных перерывов в работе можно принять равным 70 минутам.

Во время перерывов необходимо для снятия напряжения и усталости применять рекомендованные СанПиН комплексы физических упражнений.

При организации работ необходимо учитывать следующие требования:

1) Площадь на одно рабочее место для взрослых операторов должна составлять не менее 6 кв.м., а объем – не менее 20 куб.м.

2) Схемы размещения рабочих мест должны учитывать расстояния между рабочими столами: в направлении тыла одного монитора и экраном другого должно быть не менее 2 м, между боковыми поверхностями мониторов – не менее 1,2 м.

3) Экран монитора должен находиться от глаз на расстоянии 60 -70 см, но не менее 50 см.

4) При выполнении основной работы во всех помещениях с ПЭВМ уровень шума не должен превышать 50 дБА. В помещениях инженерно-технического персонала уровень шума не должен превышать 60 дБА. На рабочих местах в помещениях с принтерами, АЦПУ и пр. уровень шума не должен превышать 75 дБА.

5) Естественное освещение должно обеспечивать КЕО не ниже 1,2 в зонах с устойчивым снежным покровом, и ни ниже 1,5 на остальной территории.

6) Искусственное освещение должно осуществляться системой общего равномерного освещения. Освещенность на поверхности стола с клавиатурой и рабочими документами должна быть в пределах 300 - 500 лк.

7) Коэффициент пульсации люминесцентных ламп не должен превышать 5 %.

Параметры микроклимата в помещениях с ПЭВМ и ВДТ (видеодисплейными терминалами), а так же эргономические параметры рабочих мест и визуальные эргономические параметры ВДТ, должны соответствовать СанПиН № 2.2.2.542-96.

Длительное неизменное положение тела. Работа с компьютером характеризуется значительным умственным напряжением и нервно-эмоциональной нагрузкой операторов, высокой напряженностью зрительной работы и достаточно большой нагрузкой на мышцы рук при работе с клавиатурой . Большое значение имеет рациональная конструкция и расположение элементов рабочего места, что важно для поддержания оптимальной рабочей пользы человека-оператора.

Постоянное напряжение глаз. Работа с компьютером характеризуется высокой напряженностью зрительной работы. В выполняемом исследовании значительный объем информации на разных стадиях обработки представлен в графической форме с большим количеством мелких деталей, что дает серьезную нагрузку на зрение. Постоянное напряжение глаз может привести к снижению остроты зрения. Экран видеомонитора должен находиться от глаз пользователя на оптимальном расстоянии 600...700 мм, но не ближе 500 мм с учетом размеров алфавитно-цифровых знаков и символов. Также для снижения утомляемости рекомендуется делать 15-минутные перерывы в работе за компьютером в течение каждого часа.

Техника безопасности при работе на ПК.

1. К самостоятельной работе на ПК допускаются лица не моложе 18-ти лет, прошедшие медицинское освидетельствование, специальное обучение, инструктаж по охране труда на рабочем месте, изучившие "Руководство по эксплуатации" и усвоившие безопасные методы и приемы выполнения работы.

Персонал, допущенный к работе на ПК по наладке, эксплуатации PR-нию обязан:

· получить инструктаж по охране труда;

· ознакомиться с общими правилами эксплуатации и указаниями по безопасности труда, которые содержаться в "Руководстве по эксплуатации";

· познакомиться с предупреждающими записями на крышках, стенках, панелях блоков и устройств;

· познакомиться с правилами эксплуатации электрооборудования.

2. ПК должен подключаться к однофазной сети с нормальным напряжением 220 (120) В, частотой 50 (60) Гц и заземленной нейтрально. Заземляющие контакты розеток должны быть надежно соединены с контуром защитного заземления помещения. В помещении должен быть установлен автомат аварийного или рубильник общего отключения питания.

3. Запрещается самостоятельно производить ремонт ПК (его блоков), если это не входит в круг ваших обязанностей.

4. При эксплуатации ПК должны выполняться следующие требования, правила:

· не подключать и не отключать разъемы и кабели электрического питания при поданном напряжении сети;

· не оставлять ПК включенным без наблюдения;

· не оставлять ПК включенным во время грозы;

· по окончании работы отключить ПК от сети;

· устройства должны быть расположены на расстоянии 1 м от нагревательных приборов; рабочие места должны располагаться между собой на расстоянии не менее 1,5 метров;

· устройства не должны подвергаться воздействию прямых солнечных лучей; непрерывная продолжительность работы при вводе данных на ПК не должна превышать 4 часов при 8-часовом рабочем дне, через каждый час работы необходимо делать перерыв 5-10 минут, через 2 часа на 15 минут;в помещении, где расположена компьютерная техника, должен быть оборудован уголок пожаротушения.

Системный блок следует включать как можно реже (обычно включается в начале рабочего дня и выключается, выключается работа на нем — в конце дня). Для того, чтобы не выгорал экран и не расходовалась лишняя энергия, в компьютере предусмотрен специальный режим гашения экрана — через определенное время, если никто не работает на нем, т.е. нет обращения к клавиатуре или мыши, он выключается. Если монитор получает питание от системного блока, включая системный блок, включаем и монитор. Если соединение монитора и системного блока параллельно, то сначала необходимо включить монитор, потом системный блок. Выключать в обратной последовательности.

Экран монитора стеклянный, а потому и хрупкий, и поэтому надо обращаться с ним осторожно. Недопустимо попадание жидкости за заднюю часть экрана может замкнуть проводка, что выведет его из строя и может привести к возникновению пожара. В случае попадания жидкости следует отключить электропитание.

Защита от излучения расположена только на экране, поэтому, находясь прямо перед экраном, пользователь наиболее защищен от вредного воздействия излучения. На заднюю и боковые части монитора в целях экономии защиту не устанавливают. Следовательно, находясь сбоку или сзади монитора, можно получить максимально вредное воздействие.

При работе с клавиатурой стоит придерживаться следующих правил:

1) сильно не ударять по клавишам, это приводит к быстрой изнашиваемости прибора.

2) не распивать напитки над клавиатурой, так как попадание жидкости в нее приводит к короткому замыканию и выводит из строя клавиатуру, в случае попадания необходимо обесточить компьютер.

3) не кушать над клавиатурой бутерброды, семечки, так как крошки, попадающие в клавиатуру, нарушают ее работу.

4) при наличии защитной панели следует закрывать клавиатуру, тем самым, защищая ее от пыли.

Заключение

За время работы над выпускной квалификационной работой по теме «Создание WEB сайта по «Профессиональной компетентности будущих специалистов»» были изучены теоретические основы построения систем управления.

Результатом выпускной квалификационной работы является создание WEB-сайт для государственного гранта на тему «Психолого-педагогические основы формирования профессиональной компетентности будущих специалистов (На базе профессиональной подготовки бакалавров и магистров ТУИТ и его филиалов)», занимающейся исследованием профессиональной компетентностью будущих специалистов, проведением мониторинга и тренингов по компетентгостьи. Разработанный WEB-сайт позволяет автоматизировать процессы доступа к информационным ресурсам гранта через Интернет и делает возможным осуществление заказов на исследование по профессиональной компетентности, на проведение мониторинга и тренингов по компетентности, информационный обмен между грантом и потенциальными клиентами.

2. В процессе исследования предметной области определена область деятельности прикладног научного гранта «Психолого-педагогические основы формирования профессиональной компетентности будущих специалистов (На базе профессиональной подготовки бакалавров и магистров ТУИТ и его филиалов)», которая может быть автоматизирована и принято решение о создании WEB-сайта способного донести необходимую информацию до многочисленной публики. В ходе постановки задачи определена цель разработки, назначение и требования к WEB-сайту. Разработан предполагаемый вариант функционирования, определены цели, которые могут быть достигнуты с его помощью, разработана концепция и сценарий WEB-сайта. Рассмотрены вопросы регистрации доменного имени и

публикации WEB-сайта.

3. Для публикации информационных материалов и создания структуры сайта использован язык разметки гипертекстовых страниц HTML – как самый распространенный в среде Интернет. Инфологическая модель обеспечивает высокую эффективность хранения и доступа к информации, информационный обмен, ссылочную целостность, делает возможным дальнейшее редактирование WEB-сайта.

4. Созданная структура WEB-сайта на языке HTML предоставляет конечному пользователю эффективную навигацию по сайту благодаря интуитивно понятному интерфейсу. Применение CSS делает редактирование сайта удобным, облегчает программный код, создает привлекательный дизайн. Применение метода подключения страниц с часто повторяющимся фрагментом кода делают общий размер WEB-сайта намного меньше, позволяют изменять дизайн сайта и назначение ссылок, на всех страницах внося изменения лишь в один подключаемый файл.

5. Определение методов тестирования для каждой задачи позволило создать механизм испытаний, охватывающих все аспекты работы WEB-сайта. Положительные результаты испытаний показали готовность WEB-сайта к внедрению в опытную эксплуатацию, по всем задачам были получены положительные результаты. Тестовые испытания показали хорошую пригодность WEB-сайта для реализации поставленных задач, а также к условиям функционирования.

6. Предложения по использованию продукта.

Для продвижения WEB-сайта нужно зарегистрировать его на поисковых серверах, что позволит пользователям заходить на него по ключевым словам. Сайт должен быть «живым» т.е. он должен обновляться хотя бы 1 раз в неделю. Перед публикацией на сайте графических материалов желательно их оптимизировать для уменьшения размера файлов. Перед публикацией любых материалов на WEB-сайте сначала необходимо их протестировать в локальной сети.

7. Перспективы развития.

Разработанный в ходе выпускной квалификационной работы WEB-сайт является «быстрым вариантом» т.е. заключает в себе те немногочисленные функции, которые присущие профессиональным сайтам. Созданный WEB-сайт является творением одного разработчика – автора данной выпускной квалификационной работы, в то время как профессиональной разработкой сайтов занимаются группы людей WEB-дизайнеры, маркетологи, психологи, программисты и четко разграничивают функции каждого участника. Поэтому в течение тестовой эксплуатации WEB-сайта необходимо проанализировать эффективность его работы и определить слабые места для исправления в будущем. В качестве перспектив развития WEB-сайта можно порекомендовать следующие направления:

- для удобства нахождения необходимой информации на WEB-сайте разработать поисковую систему;
- для осуществления контроля корректности информации, введенной пользователем в отправляемые формы, предусмотреть возможность проверки данных, на стороне клиента используя технологии языка Jscript;

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Доклад Президента Республики Узбекистан Ислама Каримова на заседании Кабинета Министров, посвященном итогам социально-экономического развития в 2013 году и важнейшим приоритетным направлениям экономической программы на 2014 год.
2. И.А. Каримов «Мировой финансово-экономический кризис, пути и меры по его преодолению в условиях Узбекистана», Ташкент, Узбекистон, 2009.
3. Ўзбекистон Республикаси қонун ҳужжатлари тўплами, 2012 й., 5-сон, 47-модда. [http://lex.uz]
4. Замонавий ахборот-коммуникация технологияларини янада жорий этиш ва ривожлантириш чора-тадбирлари тўғрисида. Ўзбекистон Республикаси Президентининг 21.03.2012 й.даги №ПҚ-1730 сонли қарори. //Ўзбекистон қонун ҳужжатлари тўплами, 2012й., 13-сон, 139-модда. [www.lex.uz]
5. Каримов И.А. Асосий вазифамиз – ватанимиз тараққиёти ва халқимиз фаровонлигини янада юксалтиришдир.Т.: Ўзбекистон. 2010.-80 б.
6. Федорчук А. "Как создаются Web-сайты". СПб, Питер. 2001, -180с.
7. А. Мазуркевич. PHP: Настольная книга программиста. – М.: Новое знание, 2004.
8. В. Дунаев. Сам себе Web-дизайнер. – СПб.: БХВ-Петербург; Арлит 2002
9. Кузнецов М.В., Симдянов И.В. Самоучитель PHP. – 2-е изд., перераб. и доп. – СПб.: БХВ-Петербург, 2006.
10. Уильман Л., Основы программирования на PHP: Пер с англ. –М.: ДМК Пресс, 2001.
11. Электронная встроенная гипертекстовая справочная система «Информационные технологии», 1999 г.
12. Байков В. "Интернет: поиск информации и продвижение сайтов". СПб, 2000, -367с.

13. Синенко О., Леньшин В. Автоматизация предприятия вчера, сегодня, завтра. PC Week. 2000. № 29. С.15-16.
14. Айзенмегер Р. HTML 3.2/4.0. Справочник. - СПб: Бином, 1998, 249с.
15. Гофман В., Хомоненко А. Delphi 6: СПб, - :БХВ-Петербург, 2001, - 1152с.
16. Каймин В.А. "Интернет-Технологии". Учебное пособие. М., WDU. 2001, - 236с.
17. Создание Intranet: Официальное руководство Microsoft / Под ред. В.Сергеева. пер. с англ.- Спб.: BHV – Санкт-Петербург, 1998, - 672с.
18. Электронная встроенная гипертекстовая справочная система «HTML и CSS», 2001 г.
19. Спецификация CSS. http://www.w3.org/TR/REC-CSS2
20. Архив статей портала PHP.SU (http://www.php.su/)
21. Грег Салливан, Дон Бенаж Microsoft BackOffice в подлиннике БХВ-Санкт-Петербург, 1997 г., т.2 – 640с.
22. Мешков А.В., Тихомиров Ю.В. Программирование для Windows NT и Windows 95 в 3т.: БХВ-Санкт-Петербург, 1997 г., т.1 – 464с.; т.2 – 464с, т.3 – 460с.
23. Хитрости WEB-дизайна, http://www.khv.ru/redirect.cfm?CFID=136786&CFTOKEN=78992983&CFApp=2&Message_ID=1245
24. И.Рахманов, Г.Искадария, К.Худойкулов, «Первая медицинская помощь в чрезвычайных ситуациях». Ташкент, «Фан» 2005.
25. А.Кудратов, Т.Ганиев и др. «Безопасность жизнедеятельности». Ташкент, Алокачи, 2005.

Ссылки на использованные источники в Интернете.

- http://www.gotdotnet.ru/
- http://www.sql.com/
- http://dotsite.ru/

- http://www.rsdn.ru/
- http://www.math.nsc.ru./LBRT/k5/knapsack_problem.
- http://www.math.nsc.ru./LBRT/k5/or.html/
- http://www.citforum.ru/database/case/glava3_2.shtml

Приложение 1

Процесс работы программы

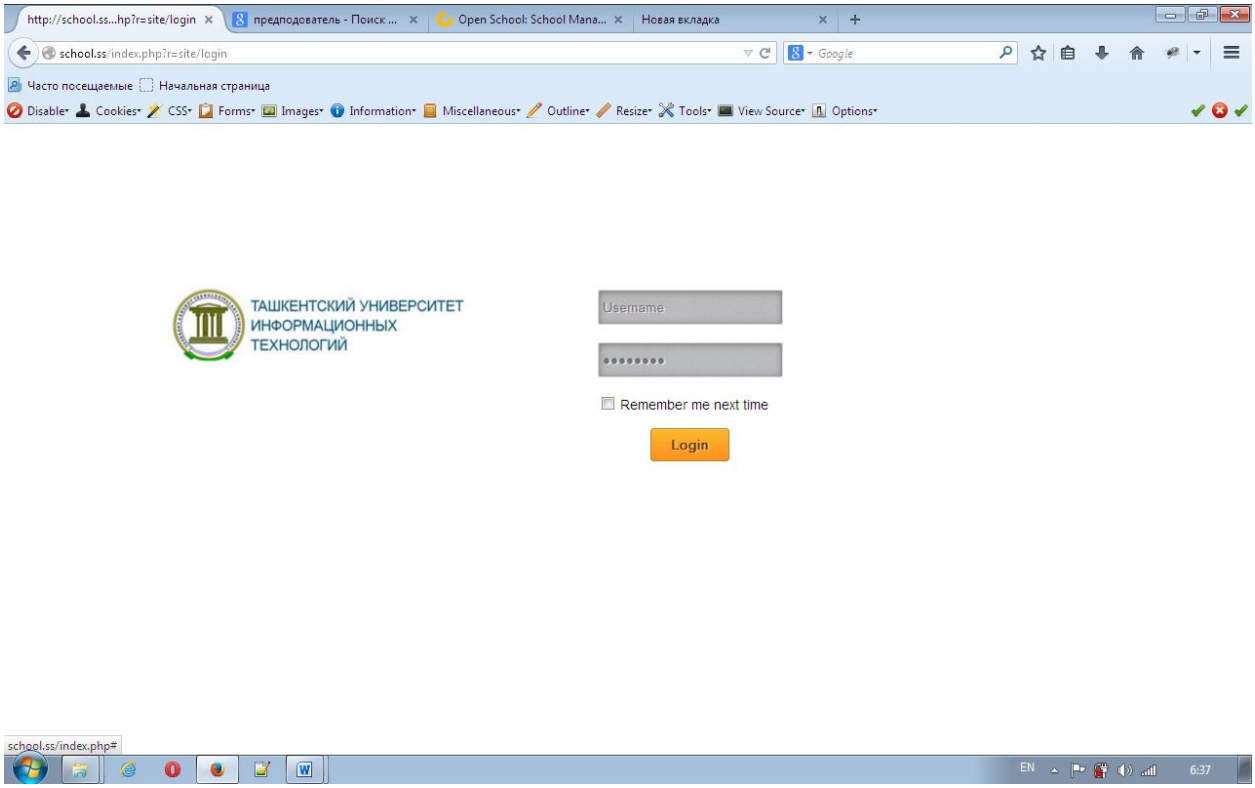

Рис. 1. Вход в систему.

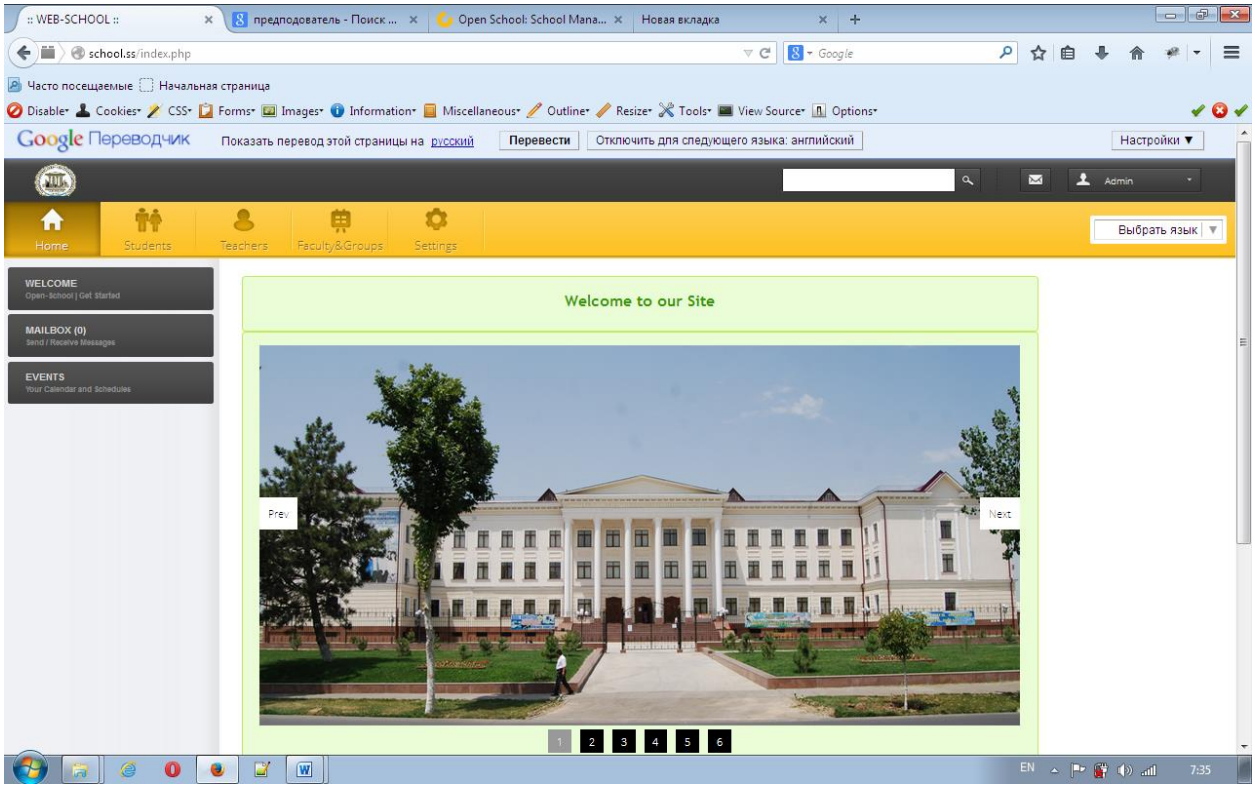

Рис. 2. Главная страница.

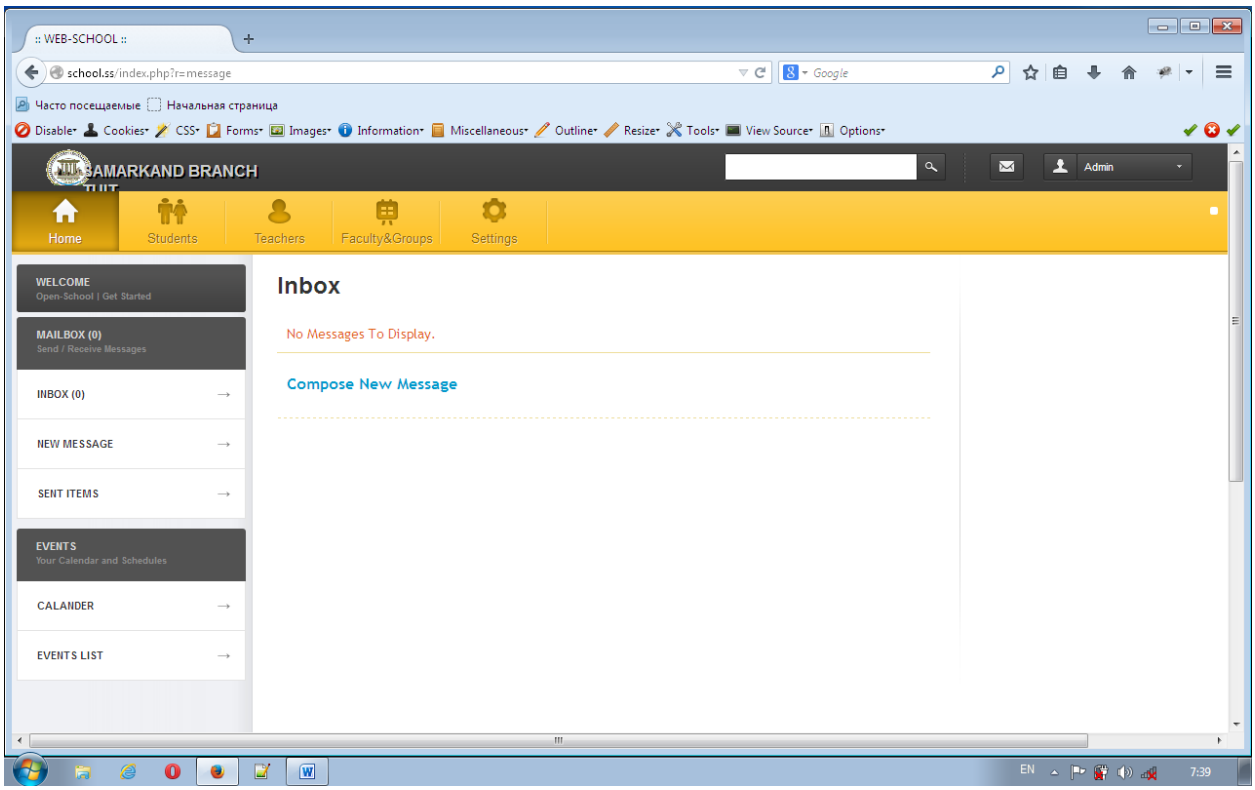

Рис. 3. Страница почты.

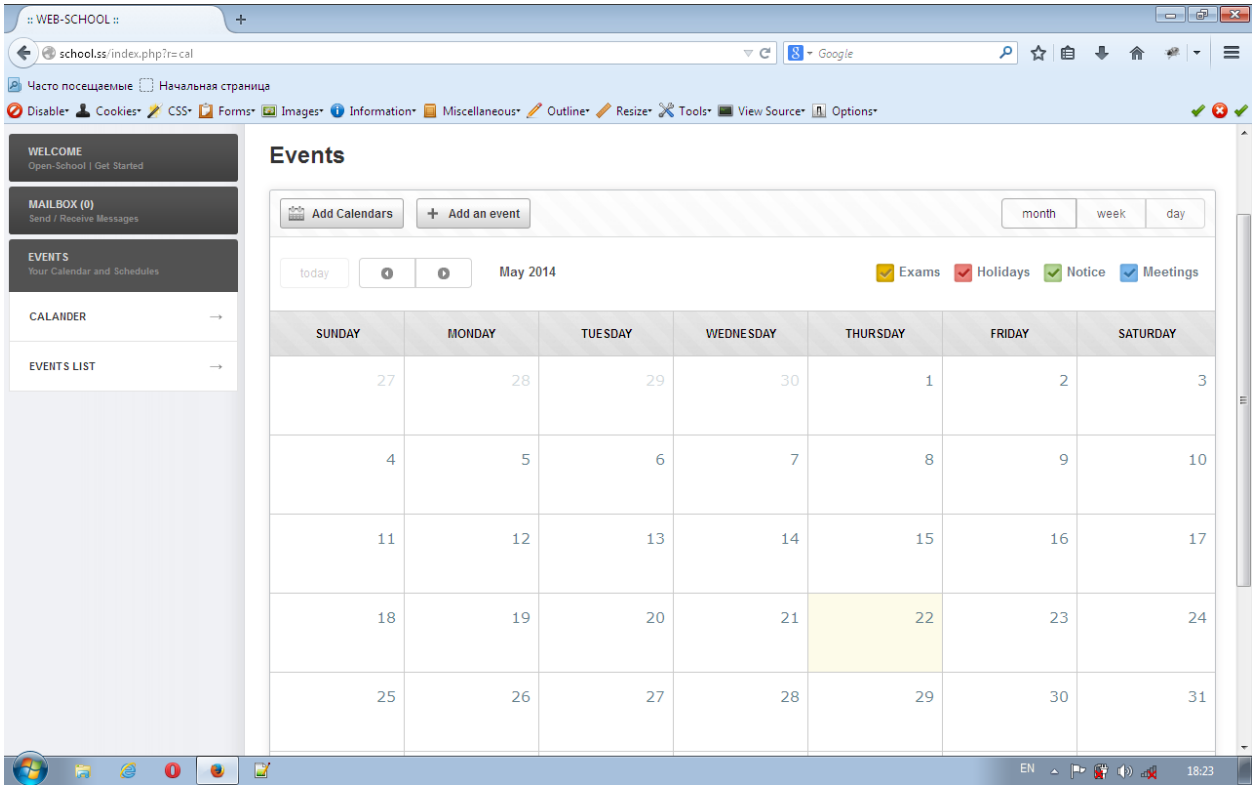

Рис. 4. Страница событий.

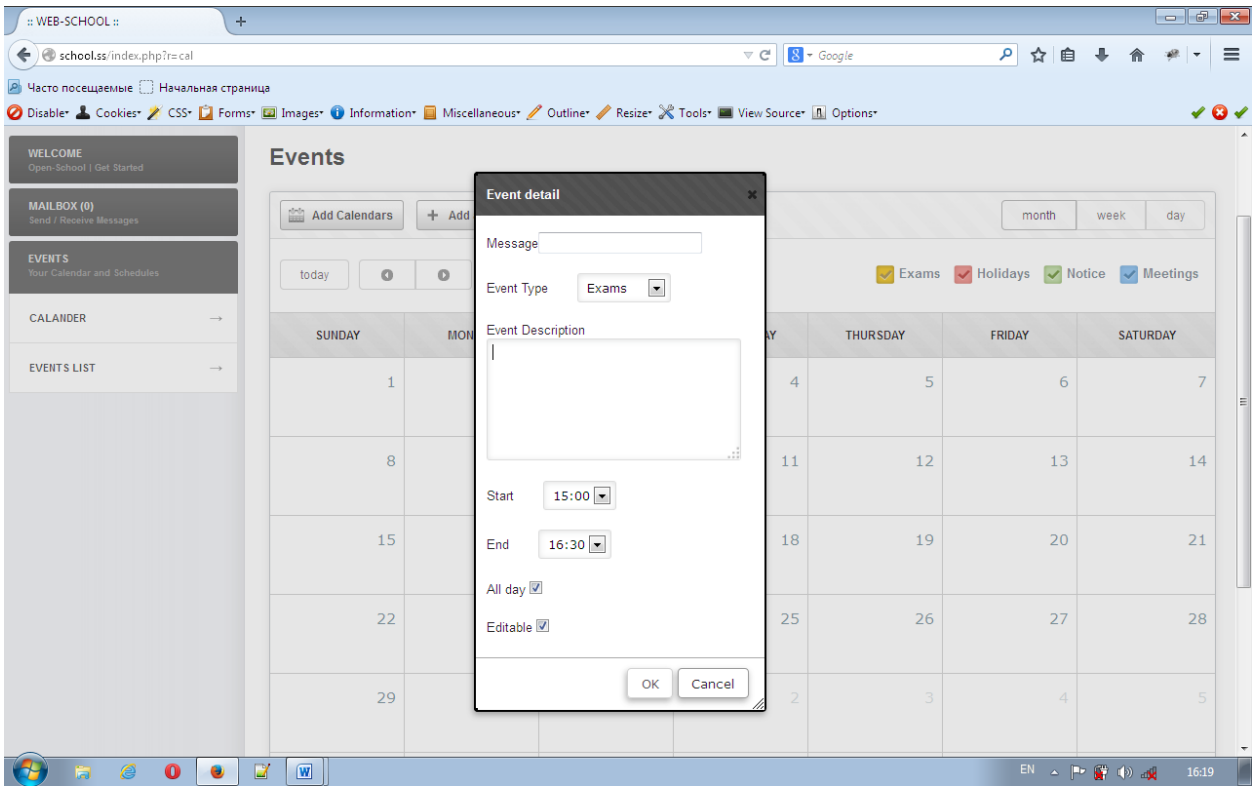

Рис. 5. Пример ввода событий.

Рис. 6. Страница Опубликованные работы участников гранта

Приложение 2

Код программа