ГОСУДАРСТВЕННЫЙ КОМИТЕТ КОММУНИКАЦИИ, ИНФОРМАТИЗАЦИИ И ТЕЛЕКОММУНИКАЦИОННЫХ ТЕХНОЛОГИЙ УЗБЕКИСТАНА

ФЕРГАНСКИЙ ФИЛИАЛ ПРИ ТУИТ

Кафедра «Информационных технологий»

_____

« ____ »_____2013 г

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ

РАБОТА

На тему: «Создание системы управления содержимым сайта с помощью PHP и нереляционной БД MongoDB

Выпусник: Н.Исмаилов

Фергана – 2013 г.

АННОТАЦИЯ

Аннотация дипломной работы студента Исмаилова Т.Ш. «Создание системы управления содержимым сайта с помощью PHP и нереляционной БД MongoDB». В данной дипломной работе были раскрыты подходы к формированию и созданию системы для управления данными сайтов, также раскрыто комплексное управление структурой и с текстовым содержимым сайта.

Дипломная работа состоит из 4 разделов:

I.Аналитическая часть – в ней приводится краткая характеристика предприятия, рассматривается производственная структура и анализируется деятельность предприятия на внутреннем и внешнем рынке. Так же в этом разделе будет рассмотрены основные принципы производственной деятельности компании. Приведутся обоснования внедрения новых технологий и проведения рефакторинга одного из основных рабочих процессов. В одной из глав данного раздела также будут рассмотрены основные цели и концепция данного дипломного проекта.

II.Теоретическая часть – данный раздел будет посвящен рассмотрению используемых технологий для разработки в сфере создания веб-сайтов и проведению их комплексного обслуживания. Будет проведен анализ внутренних разработок компании, технических наработок и опыта в сфере веб разработок. Также в теоретической части рассмотрим какие программные и аппаратные средства используются для создания и продвижения веб-сайтов.

III.Проектная часть - здесь будет рассмотрены принципы проведения рефакторинга и описан процесс разработки программного обеспечения для компании, с учетом повышения производительности и уменьшения затрат на создание и поддержку работоспособности сайтов. В проектной части будет рассмотрено обоснование выбора программного обеспечения для дальнейшей разработки системы управления данными сайта. Здесь будут описаны

технологии используемые в работе над проектом и весь процесс подготовки, разработки и сдачи в работу создаваемого проекта.

IV.Охрана труда и техника безопасности – в данном разделе будет описаны основные требования охраны труда и соблюдению техники безопасности при проведении работ за компьютером.

В данной дипломной работе будет показан весь процесс разработки комплексного решения по контролю, учету и управлению сайтами и потоковыми данными.

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ.

Общеизвестно высказывание о том, что тот, кто владеет информацией, тот владеет и миром. Иное сообщение стоит дороже жизни. По преданию, 13 сентября 490 года до н.э. греческий воин-гонец, прибежавший из Марафона в Афины, не останавливаясь в пути, упал замертво, но донес весть о победе над персами.

С давних времен сбор и систематизация сведений об окружающем мире помогали человеку выживать в нелегких условиях – из поколения в поколение передавался опыт и навыки изготовления орудий охоты и труда, создания одежды и лекарств. Информация постоянно обновлялась и дополнялась – каждое изученное явление позволяло перейти к чему-то новому, более сложному. Со временем, большие объемы данных об окружающем мире поспособствовали развитию научно-технического прогресса и, как следствие, всего общества в целом – человек смог научится управлять различными видами вещества и энергии.

С течением времени роль информации в жизни человека становилась все существеннее. Нужно было изучать и понимать уже не только законы природы, но и понятия и ценности человеческого общества – литературу, искусство, архитектуру и т.д.Сейчас, в первой половине 21-ого века роль информации в жизни человека является определяющей – чем больше навыков и знаний он имеет, тем выше ценится как специалист и сотрудник, тем больше имеет уважения в обществе.

Познавая окружающий мир, человек постоянно имеет дело с информацией. Она помогает человеку правильно оценить происходящие события, принять обдуманное решение, найти наиболее удачный вариант своих действий. Интуитивно мы понимаем, что информация — это то, чем каждый из нас пополняет собственный багаж знаний. Информация также является сильнейшим средством воздействия на личность и общество в целом. Кто владеет наибольшим объемом информации по какому-либо вопросу, тот всегда находится в более выигрышном положении по сравнению с остальными.

В последние десятилетия настойчиво говорят о переходе от «индустриального общества» к «обществу информационному». Происходит смена способов производства, мировоззрения людей, их образа жизни. Информационные технологии кардинальным образом меняют повседневную жизнь миллионов людей.

Информация стала одним из важнейших стратегических, управленческих ресурсов, наряду с ресурсами - человеческим, финансовым, материальным. Ее производство и потребление составляют необходимую основу эффективного функционирования и развития различных сфер общественной жизни, и, прежде всего, экономики. А это означает, что не только каждому человеку становятся доступными источники информации в любой части нашей планеты, но и генерируемая им новая информация становится достоянием всего человечества. В современных условиях право на информацию и доступ к ней имеют жизненную ценность для всех членов общества. Возрастающая роль информации в обществе явилась предметом научного осмысления. Были выдвинуты теории, объясняющие ее место и значение. Наиболее популярными являются теории постиндустриального и информационного общества.

Мир вступает в новую эру – информационную, в век электронной экономической деятельности, сетевых сообществ и организаций без границ. Приход нового времени радикально изменит экономические и социальные стороны жизни общества. Подобные изменения самым прямым образом касаются места человека в информационном мире. Человек меняется в соответствии с вектором информационно-технических характеристик общества. Однако это совсем не пассивное принятие новых условий производства и потребления. Человек выступает субъектом информационной реальности, далеко выходящей за информационно технические характеристики. Информатизация повседневной жизни и появление нового информационного поля человеческого бытия не проходит бесследно для жизненного мира человека. В электронном пространстве изменяются поведенческие стандарты и ценностные ориентации личности.

Новые условия для мирового человечества в особенной форме проявляются в Узбекистане.Современный Узбекистан еще не является информационным обществом. Прежде всего, потому, что часть информации недоступна широкому кругу пользователей или заменена дезинформацией. Однако информатизация отдельных сегментов социальной жизни, отдельных сфер политики и экономики рано или поздно создаст условия для появления подлинной социальной ткани нового типа, из которой способно вырасти информационное общество. Постиндустриальные тенденции могут быть достаточно органично соединены с особенностями узбекского народа.

Информационное общество нередко называют массовым обществом и обществом потребления. Это связано с такими процессами информатизации как развитие сферы массовых коммуникаций. Глобальные и локальные компьютерные сети, средства сотовой связи, система телевидения и радиовещания, являясь компонентами информационной структуры общества, обеспечивают вместе с этим и коммуникацию между людьми. Массовая коммуникация – одно из важных явлений современного общества, которое заметно сказывается на развитии всяких технологий, информационных технологий в частности как внутри каждой страны, так и между странами. Зачастую процессам информатизации придается негативный оттенок, который присущ обществу потребления. Многие представители общественной и научной мысли видят в информатизации губительные для духовной сферы общества процессы и ассоциируют информационную цивилизацию с антиподом культуры и духовности.

В области теоретического понимания происходящих процессов также до сих пор нет единого мнения относительно путей развития информационного общества, приоритетности того или иного его направления, ясности и четкости формулировок и понятий, выражающих происходящее в информационной сфере. Поэтому теоретическое исследование как концептуальных, так и практических (реальных) предпосылок понимания текущих информационных процессов остается актуальным

В условиях развития современного общества информационные технологии глубоко проникают жизнь людей. Они очень быстро превратились в жизненно важный стимул развития не только мировой экономики, но и других сфер человеческой деятельности. Сейчас трудно найти сферу, в которой сейчас не используются информационные технологии. Так, в промышленности информационные технологии применяются не только для анализа запасов сырья, комплектующих, готовой продукции, но и позволяют проводить маркетинговые исследования для прогноза спроса на различные виды продукции, находить новых партнеров и многое другое.

При этом все бухгалтерские операции на предприятиях и не только, сейчас основываются на применении информационных технологий. Как известно эффективность роботы государственного управления во многом зависит от уровня взаимодействия между гражданами, предприятиями и другими органами управления. Поэтому в государственном управлении информационные технологии позволяют одновременно использовать информационные, организационные, правовые, социально-психологические, кадровые и другие факторы, что значительно облегчает роботу и организацию самого процесса управления. Конечно, применение таких технологий не решает всех проблем, но значительно ускоряют роботу на сложных участках аналитической деятельности, например, во время проведения анализа и оценки оперативной обстановки в сложных ситуациях, подготовки и формирования отчетов и справок.

Применение информационных технологий в научной сфере и в сфере образования сложно переоценить. Сейчас трудно представить себе школу, в которой бы не было компьютерного класса. Сейчас существует масса электронных библиотек, воспользоваться которыми можно не выходя из дома, что значительно облегчает процесс обучения и самообразования. При этом информационные технологии способствуют развитию научных знаний.

Так как увеличивается скорость обмена информацией и появляется возможность проводить сложные математические расчеты за несколько секунд

и многое другое. Информационные технологии это один из современных способов общения, главными преимуществами которого являются общедоступность. Используя информационные технологии можно с легкостью получить доступ к интересующей вас информации, а также пообщаться с живым человеком. С одной стороны это имеет отрицательный эффект, так как люди все меньше общаются «вживую», при непосредственном контакте, но с другой стороны позволят общаться с человеком, который находится на другом конце света, а это согласитесь, имеет огромное значение.

I.АНАЛИТИЧЕСКАЯ ЧАСТЬ

Здесь разссмотрим структуру и анализ производственной деятельности компании “Давр.ру”. Узнаем о предприятии, как оно себя классифицирует, чем занимается и как позиционирует.

1.1 Краткая характеристика предприятия

ООО “Давр.ру” - одна из немногих компаний в Узбекистане, которая не только разрабатывает сайты для своих клиентов, но и предоставляет широчайший спектр разносторонних услуг по обслуживанию сайтов.

Компания была основана в г. Фергана в 2001 году и по сей день продолжает свою успешную деятельность, принося своим клиентам новый рынок для их деятельности и сбыта продукции – Интернет.

Уже на протяжении 14 лет фирма “Давр.ру” помогает своим клиентам легко и успешно создать сайт в сети Интернет, продвигать его и облуживать. Основной целью для компании является создание эффективного сайта требует высокопрофессионального подхода с применением передовых интернет технологий и идей. Веб-студия “Давр.ру” придерживается именно этих критериев в создании и разработке сайтов. Мы на профессиональном уровне занимаемся разработками Интернет-решений и используем все самые передовые технологии в сфере IT-технологий. Особое внимание веб-студия “Давр.ру” уделяет компаниям, только начинающим свой путь в Интернете. В небольших компаниях, как правило, некому заниматься работой с сайтом. Веб-студия “Давр.ру” поддерживает такие компании, беря на себя всю работу по разработке и поддержке сайта.

Развитие Интернет технологий в Узбекистане открывает новые горизонты для эффективных рекламных решений. Ни одно современное начинание или предприятие не обходится без освещения в сети Интернет. Объемы размещения рекламы в Интернете постоянно растут и составляют существенный процент от общей доли рынка рекламы в Узбекистане.

Преимущества сайта как инструмена для рекламы:

- **Долгосрочность** - размещенная в Интернете информация хранится до тех пор, пока владелец сайта сам не решит её удалить.
- **Аудитория** - даже самые массовые издания не в состоянии охватить ту аудиторию, которую охватывает Интернет. Многие издания являются региональными, в то время как рекламу, размещенную на Вашем сайте, увидят не только жители всех регионов Узбекистана, но и потенциальные покупатели в России, странах СНГ и зарубежья.
- **Выгодность** - за каждое обновление рекламного объявления в газетах и журналах надо платить, в то время как обновление и продление Ваших объявлений на сайте Вам ничего не стоит.
- **Направленность** - посещая Ваш сайт, потенциальные клиенты видят только ту информацию, которая им нужна, в то время как в других источниках им приходится искать ваше объявление среди объявлений множества компаний.
- **Поиск** - размещенная на сайте информация отображается в поисковых системах по точным запросам пользователей, что позволяет потенциальному клиенту выйти прямо на Вашу компанию. В СМИ такой функции просто нет.
- **Оперативность** - размещенная на сайте информация круглые сутки доступна пользователям сразу же после её сохранения.

Учитывая приведенные выше факты, можно смело предположить, что уже в самом ближайшем будущем реклама в Интернете по своим объемам обгонит СМИ.

Продвинутые компании Узбекистана уже давно имеют свои представительства в Глобальной сети Интернет. Сайт компании - это эффективный способ заявить о себе, рекламировать продукцию и продвигать бизнес. Для успешного онлайн бизнеса Ваш сайт должен отвечать определенным критериям. Успешный сайт - это прежде всего совокупность стильного дизайна, грамотной структуры, интересного и полезного для посетителей содержания, а также проработанной

концепции. Если это Интернет-магазин, то наличие фотографий и изображений товаров будут весьма кстати. Авторскому сайту помогут уникальные статьи и привлекающий внимание дизайн.

Коммерческие организации должны предоставлять своим потенциальным клиентам полную и удобную информацию о товарах и услугах. Вне зависимости от тематики сайта, дизайн сайта должен быть подобран таким образом, чтобы он ни в коем случае не мешал, а наоборот, помогал посетителям воспринимать информацию. Простая и наглядная навигация способствует долгому нахождению посетителей на сайте. Как следствие, потенциальный клиент без труда ознакомится с Вашим предложением и закажет интересующий его товар (услугу).

1.2 Цели и концепции дипломного проекта

После анализа сервиса для внутренних нужд компании было решено провести рефакторинг и перевод на новые технические “рельсы” проекта по управление содержимым сайтов и контроля и мониторинга потоковых данных для создания тестовых сайтов (далее просто “scm”). Целью дипломного проекта является переделать проект “SCM”, с использованием новых технологий которые помогут:

- существенно снизить нагрузку на сервер проекта
- увеличить время отклика в процессе управления и анализа сайтов
- улучшить отзывчивать проекта
- упростить интерфейс самой системы управления
- снизить ресурсы требуемые для поддержания проекта

1.3 Обоснование внедрения новых технологий и проведения рефакторинга внутренней системы на фирме

Внедрение новых технологий является обоснованной мерой для улучшения произовдства внутри компании. Например вместо используемой раннее SQLite,

предалагается использовать MariaDB в связке с MongoDB для хранения данных и работы системы.

Приведем для сравнения достоинства и недостатки рассматриваемых БД для работы системы:

Преимущества SQLite

- Файловая структура - вся база данных состоит из одного файла, поэтому её очень легко переносить на разные машины
- Используемые стандарты - хотя может показаться, что эта СУБД примитивная, но она использует SQL. Некоторые особенности опущенны (RIGHT OUTER JOIN или FOR EACH STATEMENT), но основные все-таки поддерживаются
- Отличная при разработке и тестировании - в процессе разработки приложений часто появляется необходимость масштабирования. SQLite предлагает всё что необходимо для этих целей, так как состоит всего из одного файла и библиотеки написанной на языке C.

Недостатки SQLite

- отсутствие системы пользователей - более крупные СУБД включают в свой состав системы управления правами доступа пользователей. Обычно применения этой функции не так критично, так как эта СУБД используется в небольших приложениях.
- отсутствие возможности увеличения производительности - опять, исходя из проектирования, довольно сложно выжать что-то более производительное из этой СУБД.

Когда использовать SQLite

- встроенные приложения - если вам важна возможность легкого переноса приложения и не важна масштабируемость. Например однопользовательские приложения, мобильные приложения или игры
- прямой доступ к диску - при необходимости напрямую обращаться к диску вы можете выиграть при переходе на эту СУБД в функционале и

простоте использования SQL языка

- Тестирование - использование дополнительных процессов при тестировании функционала, очень замедляет приложение.

Когда отказаться от SQLite

- Многопользовательские приложения - если вам необходимо обеспечить доступ к данным для нескольких пользователей, да и к тому же различать их по правам доступа, то, наверное, полноценная СУБД (например: MySQL) будет более логичным выбором
- Запись больших объемов данных - одно из ограничений SQLite это операции записи. Разрешен только один процесс записи в промежуток времени, что сильно ограничивает производительность.

Преимущества MySQL

- Простота в работе - установить MySQL довольно просто. Дополнительные приложения, например GUI, позволяет довольно легко работать с БД
- Богатый функционал - MySQL поддерживает большинство функционала SQL.
- Безопасность - большое количество функций обеспечивающих безопасность, которые поддерживается по умолчанию
- Масштабируемость - MySQL легко работает с большими объемами данных и легко масштабируется
- Скорость - упрощение некоторых стандартов позволяет MySQL значительно увеличить производительность.

Недостатки MySQL

- Известные ограничения - по задумке в MySQL заложены некоторые ограничения функционала, которые иногда необходимы в особо требовательных приложениях.
- Проблемы с надежностью - из-за некоторых способов обработки данных MySQL (связи, транзакции, аудиты) иногда уступает другим СУБД по

надежности.

- Медленная разработка - Хотя MySQL технически открытое ПО, существуют жалобы на процесс разработки. Стоит заметить, что существуют другие довольно успешные СУБД созданные на базе MySQL, например MariaDB.

Когда следует использовать MySQL

- распределённые операции - если функционала SQLite не хватает, то стоит рассмотреть MySQL. Так как эта СУБД сочетает в себе продвинутый функционал и свободный доступ к исходному коду.
- высокий уровень безопасности - система безопасности MySQL включает в себе простые и в то же время достойные способы защиты доступа к данным
- Веб сайты и веб приложения - большинство сайтов и онлайн приложений спокойно работают с MySQL несмотря на некоторые ограничения. Будучи легкой в настройке и масштабируемой системой - MySQL проверена временем.
- Индивидуальные решения - если вы работаете с каким либо специфическим проектом, MySQL легко сможет вам помочь благодаря широким возможностям в настройке и функционалом.

Когда лучше отказаться от MySQL

- Соответствие стандартам -Так как MySQL не ставит для себя целью - полностью соответствввать стандртам SQL, то эта СУБД не полностью поддерживает SQL. Если в будущем вы планируете перейти на подобную систему, то MySQL - не лучший выбор.
- Многопоточность - хотя некоторые движки БД довольно легко выполняют параллельное чтение, параллельные операции чтения-записи могут создать проблемы
- Недостаток функционала - некоторые движки MySQL, например, не поддерживают полнотекстовый поиск.

Достоинства PostgreSQL

- Открытое ПО соответствующее стандарту SQL - PostgreSQL - бесплатное ПО с открытым исходным кодом. Эта СУБД является очень мощной системой.
- Большое сообщество - существует довольно большое сообщество в котором вы запросто найдёте ответы на свои вопросы
- Большое количество дополнений - несмотря на огромное количество встроенных функций, существует очень много дополнений, позволяющих разрабатывать данные для этой СУБД и управлять ими.
- Расширения - существует возможность расширения функционала за счет сохранения своих процедур.
- Объектность - PostrgreSQL это не только реляционная СУБД, но также и объектно-ориентированная с поддержкой наследования и много другого

Недостатки PostgreSQL

- Производительность - при простых операциях чтения PostgreSQL может значительно замедлить сервер и быть медленнее своих конкурентов, таких как MySQL
- Популярность - по своей природе, популярностью эта СУБД похвастаться не может, хотя и присутствует довольно большое сообщество.
- Хостинг - в силу выше перечисленных факторов иногда довольно сложно найти хостинг с поддержкой этой СУБД.

Когда использовать PostgreSQL

- Целостность данных - когда надежность и целостность данных - ваши требования, PostgreSQL будет, пожалуй, лучшим выбором
- Сложные пользовательские процедуры - если вам необходимо использовать пользовательские процедуры, то PostgreSQL имеет встроенную поддержку для них
- Интеграция - если в будущем вы планируете переход на платные СУБД,

например Oracle, то сделать это с PostgreSQL будет довольно просто по сравнению с другими бесплатными СУБД

- Сложная структура данных - по сравнению с другими открытими СУБД PostgreSQL предоставляет больше возможностей для создания сложных структур данных без необходимости жертовать какими либо аспектами.

Когда не следует использовать PostgreSQL

- Скорость - если быстрое чтение для вас единственный фактор, то стоит присмотреться к другим СУБД
- Простая настройка - если вам не нужна целостность данных, соответствие ACID или сложные структуры данных, то настройка PostgreSQL может изрядно потрепать вам нервы
- Репликация - если вы не готовы потратить время и энергию на то, что мог бы с легкостью сделать MySQL, то наверное проще было бы на нем и остаться.

В связи с этим, анализируя три представленных БД для хранения больших объемов текстовых данных, была выбрана именно MariaDB (одна из версий MySQL).

В качестве БД для хранения параметрических данных (статистики, результаты анализов и др.) была выбрана MongoDB. Почему именно она, просто других конкурентов при выборе бесплатной, активно развивающейся и стабильно работающей nosql БД не было.

В связи с тем, что рефакторинг принесет очень много пользы и при этом не потребует больших затрат и ресурсов, как финансовых так и временных, то было принято решение о проведении технических работ.

Выводы.

В данной части дипломной работы мы рассмотрели предприятие «Давр.Ру» с нескольких сторон. В этому году компании исполняется 13 лет, за прошедшее время качество работ улучшилось в разы, повысилась сложность выполняемых работ и получилось свести количество допускаемых ошибок к нулю, в связи с

чем они могут называть себя настоящими профессионалами в своем деле.

Из анализа внутренней структуры компании удалось узнать, что несомненно, организационная структура любого предприятия зависит от стиля управления его руководителя. Впрочем, данный фактор является абсолютно индивидуальным, а кроме того, любой руководитель способен создать такую структуру управления, которая бы отвечала цели создания наиболее благоприятного режима для осуществления руководства. Хотя, необходимо отметить, что указанная цель не всегда совпадает с целью создания оптимального механизма функционирования предприятия.

Рассмотрев индивидуальные особенности данной компании и учитывая всю специфику ее работы, теперь мы можем приступить к теоретическому планировнию системы управления содержимым сайта и начинать анализировать список предлагаемых технологий для реализации данной задачи. Наблюдая за деятельностью компании на рынке, ее прогрессом и принципами работы с клиентами можно понять, что для решения поставленной задачи нужно использовать передовые технологии и современный подход к построению высоконагруженных и высокпроизводительных систем.

II.ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

2.1 Анализ существующих разработок в компании

Организационная структура направлена, прежде всего, на установление четких взаимосвязей между отдельными подразделениями организации, распределения между ними прав и ответственности. В ней реализуются различные требования к совершенствованию систем управления, находящие выражения в тех или иных принципах управления.

Организационная структура компании и ее управления находятся в постоянной динамике, совершенствуются в соответствии с меняющимися условиями. Организационные структуры управления промышленными организациями отличаются большим разнообразием и определяются многими факторами и условиями. К ним могут быть отнесены, в частности, размеры производственной деятельности организации; производственный профиль организации; характер выпускаемой продукции; сфера деятельности организации.

Как и любая, организационно - управленческая структура, ООО “Давр.ру” имеет три уровня управления: высший, средний и оперативный. На высшем уровне управления принимаются наиболее общие решения по управлению предприятием и осуществляются функции стратегического планирования, общего контроля и связи с внешними структурами. На среднем уровне решения высшего уровня детализируются, преобразуются в конкретные планы, осуществляется выполнение функций текущего планирования, связи между высшим и низшим уровнями управления, контроля, управления производством и потоками ресурсов. Результатом деятельности работников оперативного уровня является выполнение производственной программы, происходит реализация функций управления основным и вспомогательным производством, оперативного управления и местного контроля.

Во главе компании стоит генеральный директор. Он решает самостоятельно все вопросы деятельности компании. Также он распоряжается в пределах предоставленного ему права имуществом, заключает договора. Издает приказы

и распоряжения, обязательные к исполнению всеми работниками компании. Директор несет в пределах своих полномочий полную ответственность за деятельность комбината, обеспечение сохранности товарно-материальных ценностей, денежных средств и другого имущества компании.

В подчинении директора находятся заместитель директора по экономике и финансам; заместитель директора по коммерческим вопросам; заместитель директора по персоналу и общим вопросам; главный бухгалтер; главный инженер.

Таким образом, можно сделать вывод, что высшее руководство компании ООО “Давр.ру” имеет линейную структуру управления. Это проявляется в непосредственном подчинении по всем вопросам нижестоящих подразделений вышестоящим.

В компании хорошо поставлена практика делегирования полномочий, входящих в непосредственную компетенцию определённого подразделения. Генеральный директор компании сосредоточил всю основную власть в своих руках, при этом он не старается выполнить все поставленные задачи в одиночку, а делегирует часть своих полномочий прямым заместителям. Таким образом, директор управляет своими прямыми заместителями, имея при этом представления о действиях подчиненных.

2.2 Теоретические основы и технологии информационного обеспечения компании “Давр.Ру”

2.2.1 Программные средства

Для создания и управления сайтами в компании “Давр.ру” используется специфическое программное обеспечение, как для дизайнеров, так и для верстальщиков и программистов. Давайте рассмотрим его более подробно ниже:

Sublime Text

Sublime Text — быстрый кроссплатформенный редактор исходных текстов программ. Поддерживает плагины на языке программирования Python.

Sublime Text не является свободным или открытым программным

обеспечением, однако, некоторые его плагины распространяются по свободной лицензии, а также разрабатываются и поддерживаются сообществом разработчиков.

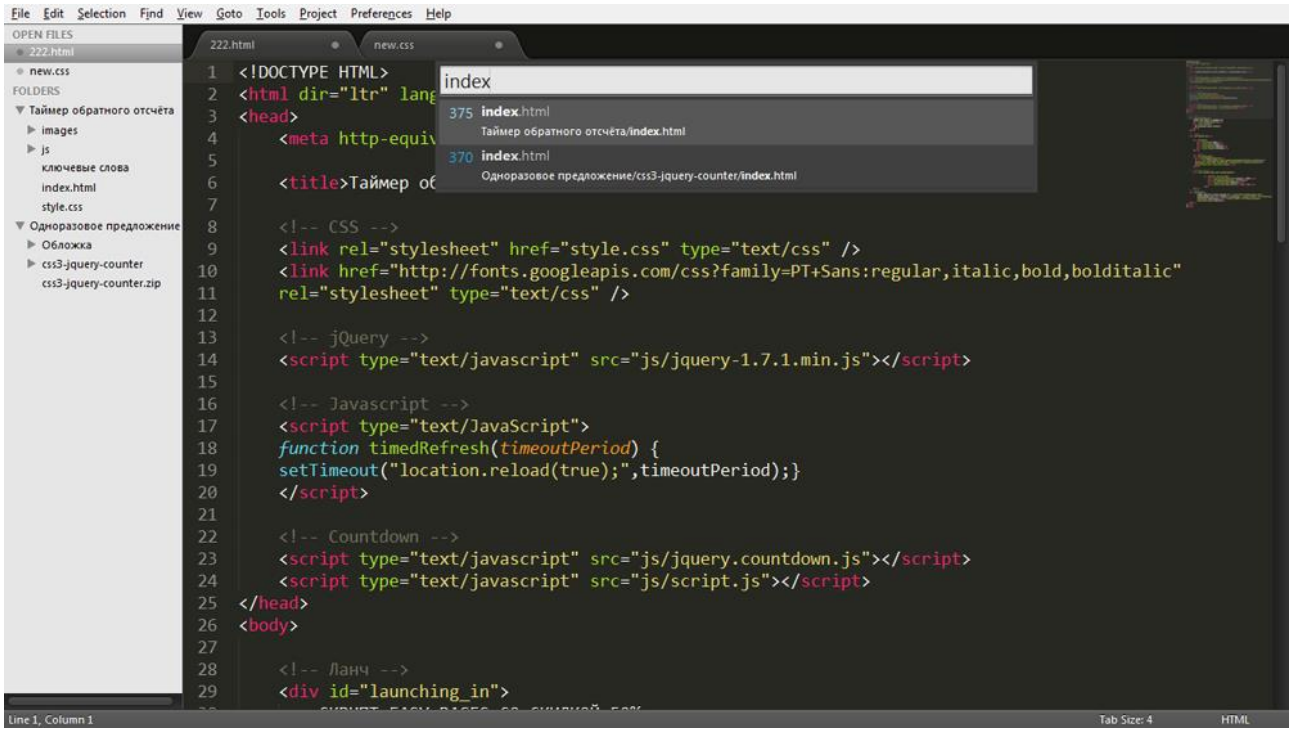

Рис 2.1. Интерфейс редактора Sublime Text

Sublime Text — это платный текстовый редактор, написанный на C++, который:

- Работает в Linux, OS X и Windows
- Обладает приличной скоростью работы
- Приятным интерфейсом (включая всевозможные анимации)
- Гибко настраиваем (правда, не в GUI, а в json-конфигах)
- Имеет множество плагинов, число которых растёт как на дрожжах
- Поддерживает VIM-режим
- Использует fuzzy-поиск

Интерфейс

Одно из первых впечатлений о программе — она красива из коробки. Правильно подобранные шрифты, цветовая схема, плавные анимации (их здесь больше, чем в большинстве текстовых редакторов и IDE). Всё это имеет смысл,

т.к. в итоге радует глаз и не отвлекает внимания. До тех пор, пока редактор не перегружен плагинами, он обладает весьма быстрым откликом, от чего я успел отвыкнуть, используя NetBeans.

Первое что бросается в глаза — отсутствие какой-либо панели инструментов. Также я пока не встретил ни одного диалогового окна, кроме стандартных окон сохранения/открытия файла. Вместо диалоговых окон используются «слои». Символы пробела и tab-а отображаются только при выделении текста, но в настройках можно задать режим «всегда».

Справа по борту расположена **карта кода**. Своеобразный аналог прокрутке страницы в виде pixel-карты, которая представляет из себя сжатый до ~100px по горизонтали код текущего файла (включая подсветку синтаксиса). Помогает в ориентировании по файлу, а также упрощает прокрутку страницы, т.к. действует аналогично scrollbar-у. Сложно наверняка сказать «киллер-фича» это или очередная «свистелка», но в течение всего времени использования у меня так и не возникло желания убрать её.

Слева по борту может располагаться **панель проекта** и **открытых файлов** (View -> Side bar -> Show side bar). Панель проекта — дерево подключённых к проекту директорий с упрощёнными возможностями файлового менеджера (к примеру есть возможность создания новых файлов/папок, переименовывания и удаления, ~~но нет возможности перемещения~~ **#UPD оказывается есть, через переименовывание**). Панель открытых файлов мне не показалась лишней или излишне дублирующей функционал табов. Табы привычнее, но когда их становится слишком много — найти нужный проще по названию именно в этой панели.

Доступны полноэкранный режим (F11) и "**Distraction Free Mode**" (Shift + F11). С первым, я думаю, всё понятно, а вот второй мне был в новинку. Этот режим представляет из себя полноэкранный режим с собственными настройками. Впервые перейдя в него вам доступны лишь сам редактор кода, да строка меню. В ней (во ->View) можно включить/отключить всё нужное/лишнее. Удобный режим для глубокого погружения в работу.

Режим **вертикального выделения** является одной из самых важных функций для продвинутых текстовых редакторов. И sublime не исключение. В Linux-версии он активируется правой кнопки мыши при зажатом шифте. Очень удобно при быстрых правках разного рода списков, разметки и не только. Стоит отметить, что ST2 умеет искать и заменять по регулярным выражениям, без чего было бы сложно рассматривать его всерьёз.
Также стоило бы отметить — **горизонтальный scroll**. Если на вашей мыши его нет, воспользуйтесь shift + вертикальный scroll. Как оказалось — очень удобно.
В статус панели, помимо ошибок и текущей позиции курсора, доступны переключатели текущего синтаксиса файла и размера tab-a.
Иконки-кнопки для раскрытия/сворачивания регионов кода (функции, блоки, теги и т.д.) несколько не очевидны. Дело в том, что хоть они и расположены, как и должны, слева от строки кода, но, по-умолчанию, отображаются лишь по наведению мыши (это настраивается). Доступны для множества структур, в частности очень порадовала возможность «сворачивания» SCSS-селекторов.
Sublime предоставляет массу возможностей для **множественного выделения** и правки. Т.е. можно установить курсор сразу в несколько мест и править код синхронно (при этом будут работать макросы, autocomplete, snippet-ы, буфер обмена и т.д.). Мне очень этого не хватало в Netbeans-е. Установить новую позицию курсора можно через ctrl + left_mouse_click. Или ctrl + left_double/_triple click (выделит слово/абзац целиком). Эти и другие комбинации клавиш и мыши гибко-настраиваемы.

Adobe Photoshop

Adobe Photoshop ([əˈdəʊbɪ ˈfəʊtəʃɒp], Эдóуби Фотошóп) — многофункциональный графическийредактор, разработанный и распространяемый фирмой Adobe Systems. В основном работает срастровыми изображениями, однако имеет некоторые векторные инструменты. Продукт является лидером рынка в области коммерческих средств редактирования растровыхизображений, и наиболее известным продуктом фирмы Adobe. Часто эту программу называют просто **Photoshop**.

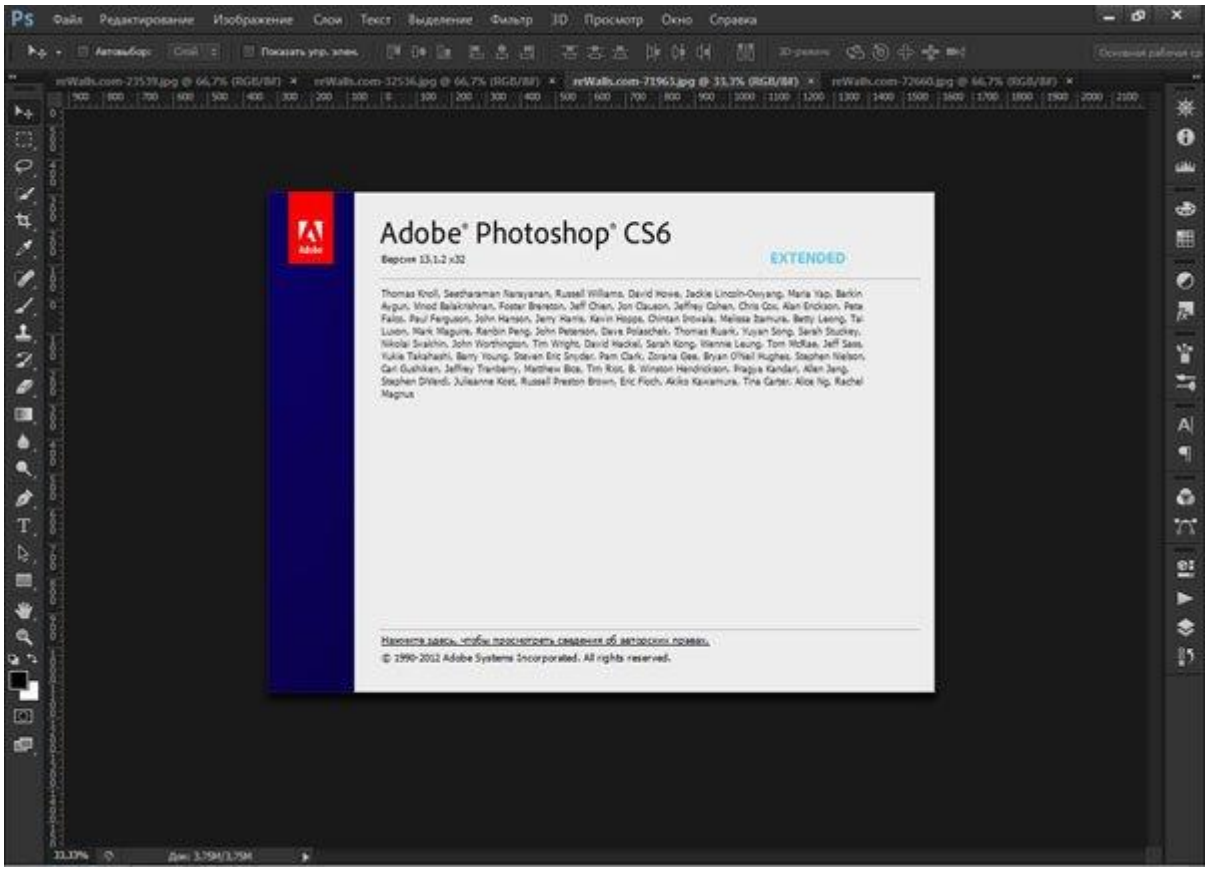

Рис. 2.2. Интерфейс рабочей области в Adobe Photoshop

В настоящее время Photoshop доступен на платформах OS X, Windows, в мобильных системах iOS и Android. Ранние версии редактора были портированы под SGI IRIX, но официальная поддержка была прекращена, начиная с третьей версии продукта. Для версий 8.0 и CS6 возможен запуск под Linux с помощью альтернативы Windows API — Wine

OpenOffice

Apache OpenOffice (ранее OpenOffice.org, OO.org, OO.o, OOo) — свободный пакет офисных приложений. Конкурирует с коммерческими офисными пакетами (в том числе Microsoft Office) как на уровне форматов, так и на уровне интерфейса пользователя. Одним из первых стал поддерживать новый открытый формат OpenDocument (ISO/IEC 26300). Официально поддерживается на платформах Linux, Microsoft Windows, Mac OS XIntel/PowerPC (поддержка оболочки Aqua находится в стадии альфа-тестирования) и раньше поддерживалсяSolaris SPARC/Intel. Существуют порты для OpenSolaris,

FreeBSD, Linux PowerPC и OS/2.

Основан на коде StarOffice, который был приобретён, а затем выпущен с открытым исходным кодом фирмой Sun Microsystems. После покупки последней права на OO.o перешли к компании Oracle.

Ранее распространялся по схеме двойного лицензирования: по лицензиям LGPL и SISSL. Но 3 сентября 2005 года компания Sun Microsystems объявила об отказе от SISSL для всех своих открытых проектов, и пакет с тех пор имеет только лицензию LGPL. В данный момент OpenOffice.org является одним из самых известных приложений среди программ с открытым исходным кодом. Об этом свидетельствует большое количествоответвлений (см. Популярность, Другие проекты).

Существует переносимая версия пакета OpenOffice.org для операционных систем семейства Microsoft Windows с возможностью использования без установки, что позволяет запускать пакет, например, с флеш-накопителя.

Офисный пакет OpenOffice.org согласно решениям Правительства РФ передан в 2008 году во все школы России для обучения информатике и компьютерной грамотности в составе базовых пакетов программ лицензионного и открытого программного обеспечения.

Офисный пакет OpenOffice.org может свободно устанавливаться и использоваться в школах, офисах, вузах, домашних компьютерах, государственных, бюджетных и коммерческих организациях и учреждениях России и стран СНГ согласно лицензии Apache.

28 сентября 2010 года, из-за жёсткого стиля руководства «сверху», некоторые разработчики OpenOffice.org объявили о создании новой некоммерческой организации The Document Foundation с целью продолжения развития офисного пакета в виде проекта LibreOffice, независимого от компании Oracle. В октябре 2010 года было объявлено, что самый популярный дистрибутив на базе Linux — Ubuntu отказывается от OpenOffice и переходит на LibreOffice.

1 июня 2011 года компания Oracle официально объявила о передаче всех прав на OpenOffice.org Apache Foundation. 13 июня фонд принял это предложение, в

результате голосования OpenOffice поступил в Apache Incubator (англ.). После окончательного перехода проекта в руки фонда, лицензия на код OpenOffice.org будет изменена на лицензию Apache 2.0. По этому поводу Фонд свободного ПО выступил с заявлением, в котором выразил огорчение по поводу подобного шага, счёл уход от копилефт лицензии ошибочным шагом и рекомендовал использовать LibreOffice

2.2.2 Аппаратные средства

Основой информатизации является использование электронно-вычислительной техники (ЭВТ) для сбора, накопления, обработки и передачи информации. ЭВМ (англ. - computer) - комплекс технических средств, предназначенных для автоматической обработки информации.

Компьютер - это сложная вычислительная система, каждая часть которой имеет свое функциональное назначение. Рассмотрим структуру персонального компьютера, предназначенного для индивидуального пользования.

Основой конструкции является **системный блок**, в котором размещены: блок питания, обеспечивающий все части компьютера электрическим питанием, системная плата с процессором, памятью и платами расширения, а также устройства записи-считывания информации на гибких и жестких (винчестер) магнитных дисках.

Материнская плата

Центральной частью любого персонального компьютера является системная (материнская) плата. На ней размещаются: базовый микропроцессор; оперативная память; сверхоперативное запоминающее устройство (ЗУ), называемое также кэшпамятью; постоянное запоминающее устройство (ПЗУ) с системной BIOS (базовой системой ввода/вывода); набор управляющих микросхем, или чипсетов (chipset), вспомогательных микросхем и контроллеров ввода/вывода; КМОП-память с данными об аппаратных настройках и аккумулятором для ее питания; разъемы расширения, или слоты

(slot); разъемы для подключения интерфейсных кабелей жестких дисков, дисководов, последовательного и параллельного портов, инфракрасного порта, а также универсальной последовательной шины USB; разъемы питания; преобразователь напряжения с 5В на 3,3 В для питания процессора (некоторым процессорам требуется также и меньшее напряжение); разъем для подключения клавиатуры и ряд других компонентов.

На платах формата ATX и NLX также находятся разъемы мыши и клавиатуры в стандарте PS/2, разъемы параллельного и последовательного портов.

Процессор

Важнейший компонент любого персонального компьютера, его «мозг» - это микропроцессор (в литературе часто его называют CPU, Central Processor Unit - ЦПУ, или центральное процессорное устройство), который управляет работой компьютера и выполняет большую часть обработки информации. Микропроцессор представляет собой сверхбольшую интегральную схему, степень интеграции которой определяется размером кристалла и количеством реализованных в нем транзисторов.

Иногда интегральные микросхемы называют **чипами** (англ, chip). Базовыми элементами микропроцессора являются транзисторные переключатели, на основе которых строятся, например, регистры, представляющие собой совокупность устройств, имеющих два устойчивых состояния и предназначенных для хранения информации и быстрого доступа к ней. Количество и разрядность регистров во многом определяют архитектуру микропроцессора.

Важнейшей характеристикой процессора является тактовая частота - величина, показывающая, сколько элементарных операций - тактов микропроцессор выполняет за одну секунду. Тактовая частота измеряется в мегагерцах (МГц) (1 МГц = 1 млн тактов в секунду) и для разных процессоров имеет следующие значения: 4,77 МГц (Intel 8088), 16-25 МГц (Intel 80286), 25-40 МГц (Intel 80386), 33-100 МГц (Intel 80486), 75-2400 МГц (Pentium). Быстродействие процессора определяется отношением тактовой

частоты к количеству тактов, требующихся процессору для выполнения простой команды, т.е. быстродействие определяется количеством элементарных операций (например, сложение), выполняемых за одну секунду.

В начале 2000 г. был достигнут рубеж тактовой частоты процессора 1 ГГц (109 Гц). Материал данной книги написан на компьютере с процессором Intel Pentium 4 с тактовой частотой 2,4 ГГц.

Память

Память (memory) предназначена для хранения данных и программ их обработки. Различают следующие виды памяти компьютера: внутреннюю и внешнюю. Встроенная в компьютер и непосредственно управляемая им память называется **внутренней**. Она разделяется на постоянное запоминающее устройство (ПЗУ, или ROM - Read Only Memory - память только для чтения) и оперативную память (RAM - Random Access Memory).

Оперативной памятью называется программно-адресуемая память, быстродействие которой соизмеримо с быстродействием процессора. В ней хранятся исполняемые в данный момент программы и оперативно необходимые для этого данные. Недостатком оперативной памяти является ее энергозависимость, т.е. при выключении компьютера все содержимое оперативной памяти стирается. Объем оперативной памяти является одной из важнейших характеристик компьютера.

Постоянная память (ПЗУ - постоянное запоминающее устройство) обычно содержит такую информацию, которая не должна меняться в ходе выполнения микропроцессором различных программ. Постоянная память имеет также название ROM (Read Only Memory), которое указывает на то, что обеспечиваются только режимы считывания и хранения. Постоянная память энергонезависима, т.е. может сохранять информацию и при отключенном питании. В ПЗУ компьютера хранится базовая система ввода-вывода (BIOS - Basic Input Output System), которая состоит из программы тестирования памяти и периферийного оборудования компьютера, а также

программы запуска операционной системы.

Компьютеры на основе процессоров с тактовой частотой 25 МГц и более, чтобы процессор не простаивал в связи с ожиданием получения данных из оперативной памяти, оснащаются кэшпамятью, т.е. «сверхоперативной» памятью относительно небольшого объема (от 64 до 256 Кбайт). **Кэш-память** (cache memory -память немедленного доступа) является «посредником» между процессором и оперативной памятью. В ней хранятся наиболее часто используемые участки оперативной памяти. За счет того, что время доступа процессора к кэш-памяти в несколько раз меньше, чем к обычной памяти, среднее время доступа к памяти значительно уменьшается.

Внешней памятью называются энергонезависимые средства памяти на сменных носителях (магнитные диски, магнитные ленты, перфоленты и другие носители информации: оптические диски, магнито-оптические диски), предназначенные для хранения больших массивов данных. Для хранения архивов, переноса информации с одного компьютера на другой широко используются гибкие магнитные диски (FDD - Floppy Disk Drive). Для работы с ними в системный блок встроен накопитель информации на гибких магнитных дисках (НГМД - дисковод) емкостью от 360 Кбайт до 2,88 Мбайт (1 Мбайт памяти позволяет запомнить около 700 страниц машинописного текста).

Гибкие магнитные диски (дискеты) представляют собой тонкие пластиковые диски, покрытые специальным магнитным материалом и запечатанные в пластиковую обложку. Магнитный материал на диске способен надолго запомнить одно из двух состояний намагниченности, соответствующих двоичным цифрам: 0 или 1. Для считывания или записи информации дискеты вставляются в дисковод. Диск в работающем дисководе вращается, а вдоль радиуса диска около его поверхности перемещается магнитная головка. Информация на дискетах записывается концентрическими кольцами - дорожками, располагаясь небольшими порциями фиксированной длины - секторами. На персональных компьютерах

широко распространены дискеты диаметром 3,5 дюйма (объем памяти от 720 Кбайт до 1,44 Мбайта).

Примечание. 1 дюйм - английская мера длины (примерно равен 2,54 см).

Дискеты бывают с двойной плотностью записи (DS/DD -Double Sided/Double Density - двусторонний гибкий диск двойной плотности) или высокой плотностью записи (DS/HD - Double Sided/High Density - двусторонний гибкий диск высокой плотности). Для предупреждения случайной порчи информации на дискетах предусмотрена защита от записи. На 3,5-дюймовых дисках имеется пластмассовый переключатель, разрешающий или запрещающий запись.

Функции внешней несменной памяти компьютера, предназначенной для долговременного и энергонезависимого хранения информации выполняют **жесткие магнитные диски (HDD-Hard Disk Drive)**. Часто их называют **винчестерами**. Они представляют собой малогабаритный пакет из нескольких жестких магнитных дисков, вращающихся с высокой скоростью на одной оси и размещенных в герметичном корпусе вместе с головками чтения-записи. Информация располагается дорожками и секторами, при этом группа дорожек на всех дисках винчестера, расположенных на одинаковом расстоянии от их общей оси, называется цилиндром. Считывание и запись информации осуществляются сразу несколькими магнитными головками одновременно со всех дисков пакета. Название «винчестер» возникло из-за того, что первые модели таких дисков имели 30 дорожек по 30 секторов; суммарная емкость получаемого накопителя обозначалась цифрами 30/30, подобно калибру старинного охотничьего ружья «Винчестер». Емкость винчестеров значительно больше, чем гибких магнитных дисков, и может иметь значение от 1 Гбайта до 200 Гбайт и более.

В качестве устройств внешней памяти большой емкости все большее распространение получают оптические диски, или CD-ROM (Compact Disk Read Only Memory - компакт-диск, предназначенный только для чтения). Емкость CD-ROM дисков - 600-800 мегабайт. Компакт-диск состоит из

поликарбонатной основы диаметром 5,25 дюйма, отражающего и защитных слоев. В качестве отражающей поверхности обычно используется напыленный алюминий. Цифровая информация представляется здесь чередованием впадин (неотражающих пятен) и отражающих свет островков. Компакт-диск имеет всего одну физическую дорожку в форме непрерывной спирали, идущей от наружного диаметра диска к внутреннему.

Считывание информации с компакт-диска происходит при помощи лазерного луча, который, попадая на отражающий свет островок, отклоняется на фотодетектор, интерпретирующий это как двоичную единицу. Луч лазера, попадающий во впадину, рассеивается и поглощается: фотодетектор фиксирует двоичный нуль.

В последнее время стали широко использоваться и другие устройства внешней памяти: ZIP-диски, флэш-накопители.

Видеомонитор

Результаты обработки информации выводятся из компьютера на экран дисплея (видеомонитора). По устройству и принципу действия видеомонитор похож на цветной телевизор.

Для генерации видеосигнала монитору в компьютере имеется специальное устройство - видеоадаптер . Информация об изображении на экране хранится в видеопамяти - специальных микросхемах видеоадаптера.

Видеомониторы бывают цветными и монохромными. Качество изображения информации на видеомониторе определяется разрешающей способностью - размером минимального элемента изображения или количеством точек, изображаемых на экране, а также воспроизводимой цветовой гаммой.

Наиболее распространены видеомониторы следующих типов:

- CGA (Color Graphics Adapter) имеет разрешающую способность 320x200 точек (320 точек по горизонтали и 200 по вертикали) при воспроизведении 4 цветов.
- EGA (Enhanced Graphics Adapter) имеет разрешающую способность 640x350 точек при воспроизведении 16 цветов.

- VGA (Virtual Graphics Array) имеет разрешающую способность 640x480 точек (минимальный размер изображения 0,31-0,39 мм) при воспроизведении 16 цветов.
- SVGA (Super VGA) в зависимости от графического режима имеет разрешающую способность: 320x200, 640x480,800x600, 1024x768 точек при воспроизведении от 16 до16 млн. цветов.

Видеомонитор может работать в двух, режимах: текстовом и графическом. В текстовом режиме на экране чаще всего располагаются 80 столбцов и 25 строк, но возможны и другие режимы (80x43 и 40x25). В графическом режиме существует доступ к каждой точке экрана, что дает возможность сформировать любое изображение. При работе в графическом режиме необходимо хранить информацию о цвете каждой точки экрана, поэтому требуется значительно больший объем видеопамяти.

Выводы.

Анализируя теоретическую часть выполняемого задания, мы пришли к выводу, что в качестве платформы для сервера нужно использовать операционную систему на базе Linux. В качестве языка программирования, как и изначально планировалось, будет использоваться язык программирования PHP версии 5.4, для хранения данных будет использоваться реляционная база данных MariaDB, которая по сути является идейным продолжением знаменитой базы данных – MySQL. А вот для хранения очереди задач и индивидуальных параметров сайта будет использоваться нереляционная база данных MongoDB, по сути она является - документо-ориентированная система управления базами данных (СУБД) с открытым исходным кодом, не требующая описания схемы таблиц. Написана на языке C++. При разработке авторы исходили из необходимости специализации баз данных, благодаря чему им удалось отойти от принципа «один размер подо всё». За счёт минимизации семантики для работы с транзакциями появляется возможность решения целого ряда проблем, связанных с недостатком производительности, причём горизонтальное

масштабирование, по мнению авторов, становится проще. Используемая модель документов хранения данных (JSON/BSON) проще кодируется, проще управляется (в том числе за счёт применения так называемого «бессхемного стиля» (англ. schemaless style)), а внутренняя группировка релевантных данных обеспечивает дополнительный выигрыш в быстродействии.

MongoDB, по мнению разработчиков, должна заполнить разрыв между простейшими NoSQL-СУБД, хранящими данные в виде «ключ — значение» (простыми и легко масштабируемыми, но обладающими минимальными функциональными возможностями) и большими реляционными СУБД (со структурными схемами и мощными запросами). Также в этом разделе мы рассмотрели основные компоненты аппаратной части используемых ПК. В качестве программного обеспечения мы рассмотрели редактор кода (Sublime Text), графический редактор для построения дизайна и интерфейса (Adobe Photoshop), свободный редактор текста (OpenOffice). Теперь имея четкое представление и имеющихся, как аппаратных, так и программных и языковых ресурсах, можно приступать к выполнению самой задачи.

III.ПРОЕКТНАЯ ЧАСТЬ

3.1.Анализ принципов построения систем потокового управления данными

Для правильного построения любой программы желательно использовать шаблоны проектирования. Что это такое – разберем с вами дальше.

Шаблон проектирования или **паттерн** (англ. design pattern) в разработке программного обеспечения — повторимая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

Обычно шаблон не является законченным образцом, который может быть прямо преобразован в код; это лишь пример решения задачи, который можно использовать в различных ситуациях. Объектно-ориентированные шаблоны показывают отношения и взаимодействия между классами или объектами, без определения того, какие конечные классы или объекты приложения будут использоваться.

«Низкоуровневые» шаблоны, учитывающие специфику конкретного языка программирования, называются идиомами. Это хорошие решения проектирования, характерные для конкретного языка или программной платформы, и потому не универсальные.

На наивысшем уровне существуют **архитектурные шаблоны**, они охватывают собой архитектуру всей программной системы.

Алгоритмы по своей сути также являются шаблонами, но не проектирования, а вычисления, так как решают вычислительные задачи.

В сравнении с полностью самостоятельным проектированием, шаблоны обладают рядом преимуществ. Основная польза от использования шаблонов состоит в снижении сложности разработки за счёт готовых абстракций для решения целого класса проблем. Шаблон даёт решению свое имя, что облегчает коммуникацию между разработчиками, позволяя ссылаться на известные шаблоны. Таким образом, за счёт шаблонов производится унификация деталей решений: модулей, элементов проекта, — снижается количество ошибок. Применение шаблонов концептуально сродни использованию готовых

библиотек кода. Правильно сформулированный шаблон проектирования позволяет, отыскав удачное решение, пользоваться им снова и снова. Набор шаблонов помогает разработчику выбрать возможный, наиболее подходящий вариант проектирования.

Хотя легкое изменение кода под известный шаблон может упростить понимание кода, по мнению Стива Макконнелла, с применением шаблонов могут быть связаны две сложности. Во-первых, слепое следование некоторому выбранному шаблону может привести к усложнению программы. Во-вторых, у разработчика может возникнуть желание попробовать некоторый шаблон в деле без особых оснований.

Также нужно рассмотреть так называемое “обобщеное программирование”, принципы которого я и буду использовать при проектировании своей дипломной работы.

Обобщённое программирование (англ. generic programming) — парадигма программирования, заключающаяся в таком описании данных и алгоритмов, которое можно применять к различным типам данных, не меняя само это описание. В том или ином виде поддерживается разными языками программирования. Возможности обобщённого программирования впервые появились в виде дженериков (обобщённых функций) в 1970-х годах в языках Клу иАда, затем в виде параметрического полиморфизма в ML и его потомках, а затем во многих объектно-ориентированных языках.

Обобщённое программирование рассматривается как методология программирования, основанная на разделении структур данных и алгоритмов через использование абстрактных описаний требований. Абстрактные описания требований являются расширением понятия абстрактного типа данных. Вместо описания отдельного типа в обобщённом программировании применяется описание семейства типов, имеющих общий интерфейс и семантическое поведение (англ. semantic behavior). Набор требований, описывающий интерфейс и семантическое поведение, называется концепцией (англ. concept). Таким образом, написанный в обобщённом стиле алгоритм может применяться

для любых типов, удовлетворяющих его своими концепциями. Такая возможность называется**полиморфизмом.**

Говорят, что тип моделирует концепцию (является моделью концепции), если он удовлетворяет её требованиям. Концепция является уточнением другой концепции, если она дополняет последнюю. Требования к концепциям содержат следующую информацию:

•**Допустимые выражения** (англ. valid expressions) — выражения языка программирования, которые должны успешно компилироваться для типов, моделирующих концепцию.

•**Ассоциированные типы** (англ. associated types) — вспомогательные типы, имеющие некоторое отношение к моделирующему концепцию типу.

•**Инварианты** (англ. invariants) — характеристики типов времени исполнения, которые должны быть постоянно верны. Обычно выражаются в виде предусловий и постусловий. Невыполнение предусловия влечёт непредсказуемость соответствующей операции и может привести к ошибкам.

•**Гарантии сложности** (англ. complexity guarantees) — максимальное время выполнения допустимого выражения или максимальные требования к различным ресурсам в ходе выполнения этого выражения.

В C++ ООП реализуется посредством виртуальных функций и наследования, а ОП — с помощью шаблонов классов и функций. Тем не менее, суть обеих методологий связана с конкретными технологиями реализации лишь косвенно. Говоря более формально, ООП основано на полиморфизме подтипов, а ОП — на параметрическом полиморфизме. В других языках то и другое может быть реализовано иначе. Например, мультиметоды в CLOS имеют сходную с параметрическим полиморфизмом семантику.

Альтернативный подход к определению обобщённого программирования, который можно назвать **обобщённым программированием типов данных**(англ. datatype generic programming), был предложен Ричардом Бёрдом и Ламбертом Меертенсом. В нём структуры типов данных являются параметрами обобщённых программ. Для этого в язык программирования вводится новый

уровень абстракции, а именно параметризация по отношению к классам алгебр с переменной сигнатурой. Хотя теории обоих подходов не зависят от языка программирования, подход Массера-Степанова, делающий упор на анализ концепций, сделал C++ своей основной платформой, тогда как обобщённое программирование типов данных используют почти исключительноHaskell и его варианты.

Средства обобщённого программирования реализуются в языках программирования в виде тех или иных синтаксических средств, дающих возможность описывать данные (типы данных) и алгоритмы (процедуры, функции, методы), параметризуемые типами данных. У функции или типа данных явно описываются формальные параметры-типы. Это описание является обобщённым и в исходном виде непосредственно использовано быть не может.

В тех местах программы, где обобщённый тип или функция используется, программист должен явно указать фактический параметр-тип, конкретизирующий описание. Например, обобщённая процедура перестановки местами двух значений может иметь параметр-тип, определяющий тип значений, которые она меняет местами. Когда программисту нужно поменять местами два целых значения, он вызывает процедуру с параметром-типом «целое число» и двумя параметрами — целыми числами, когда две строки — с параметром-типом «строка» и двумя параметрами — строками. В случае, с данными программист может, например, описать обобщённый тип «список» с параметром-типом, определяющим тип хранимых в списке значений. Тогда при описании реальных списков программист должен указать обобщённый тип и параметр-тип, получая, таким образом, любой желаемый список с помощью одного и того же описания.

Компилятор, встречая обращение к обобщённому типу или функции, выполняет необходимые процедуры статического контроля типов, оценивает возможность заданной конкретизации и при положительной оценке генерирует код, подставляя фактический параметр-тип на место формального параметра-типа в

обобщённом описании. Естественно, что для успешного использования обобщённых описаний фактические типы-параметры должны удовлетворять определённым условиям. Если обобщённая функция сравнивает значения типа-параметра, любой конкретный тип, использованный в ней, должен поддерживать операции сравнения, если присваивает значения типа-параметра переменным — конкретный тип должен обеспечивать корректное присваивание.

Итак что-же такое паттерны программирования?

При создании программных систем перед разработчиками часто встает проблема выбора тех или иных проектных решений. В этих случаях на помощь приходят паттерны. Дело в том, что почти наверняка подобные задачи уже решались ранее и уже существуют хорошо продуманные элегантные решения, составленные экспертами. Если эти решения описать и систематизировать в каталоги, то они станут доступными менее опытным разработчикам, которые после изучения смогут использовать их как шаблоны или образцы для решения задач подобного класса. Паттерны как раз описывают решения таких повторяющихся задач.

Концепция создания программного обеспечения с использованием паттернов, несомненно, очень важная, но относительно молодая, быть может, поэтому до сих пор нет четкого определения, что же такое паттерн. Об этом свидетельствуют непрекращающиеся дискуссии в популярной литературе и на соответствующих форумах в сети.

Например, следует ли считать алгоритмы и структуры данных паттернами? По этому вопросу существуют противоположные мнения. Согласно одному из них, алгоритмы являются вычислительными паттернами, а хорошо известная фундаментальная монография Дональда Кнута "Искусство программирования" по сути, представляет собой каталог таких паттернов. Согласно другому мнению, алгоритмы не являются паттернами, так как решаемые ими проблемы слишком малы (оперируют такими понятиями как вычислительная сложность и потребление ресурсов), а область решения хорошо очерчена. Паттерны же

решают проблемы большего масштаба, при этом паттерн дает не конкретное решение, а некий путь к решению, причем, выбор правильного паттерна - задача нетривиальная, предполагающая от архитектора наличие интуиции, опыта, определенного творчества.

3.2 Обоснование и выбор программного обеспечения

Операционная система является сердцевиной сетевого программного обеспечения, она создаёт среду для выполнения приложений и во многом определяет, какими полезными для пользователя свойствами эти приложения будут обладать.

На основе полученных в результате исследования заданной предметной области данных можно сформулировать ряд требований, которым должна в оптимальной степени удовлетворять ОС:

- совместимость с выбранной архитектурой ЛВС;
- поддержка выбранного аппаратного обеспечения;
- эффективное управление ресурсами;
- достаточная для решения основных задач производительность;
- масштабируемость;
- совместимость;
- довольно высокая надёжность и отказоустойчивость;
- защита данных от несанкционированного доступа;
- поддержка многозадачности;
- поддержка многопользовательского режима;
- приоритет удобства работы пользователя над скоростью вычислений;
- приемлемая стоимость.

Данным требованиям в той или иной мере соответствуют весьма популярные представители семейств Linux и Windows – Ubuntu Server 12.04 и Microsoft Windows XP/2003 соответственно, которые и будут рассмотрены в качестве вариантов при выборе ОС.

Проведём описание выбранных ОС по ряду критериев и выдвинутых ранее

требований, чтобы выявить степень соответствия их данной предметной области.

Для начала рассмотрим особенности ОС Ubuntu Server 12.04.

Fedora – дистрибутив свободной операционной системы Linux. Fedora разрабатывается сообществом в рамках проекта Fedora и спонсируется компанией Red Hat. Проект Fedora является открытым, любой заинтересованный может присоединиться к нему.

Fedora концентрируется на инновациях и последних достижениях в области открытого программного обеспечения. Дистрибутив Fedora является полностью бесплатным для любого применения.

Ubuntu Server – это проект по поддержке пользователей и разработчиков открытого ПО в России. Задача проекта – обеспечить, чтобы Fedora (а, следовательно, и все создаваемые на её основе дистрибутивы) полностью отвечала потребностям российских пользователей «из коробки». За этими простыми словами стоит много аспектов – технологическая готовность дистрибутива, наличие в нём специфичных для России программ, понятность и удобство интерфейса, хорошая документация и учебные пособия для пользователей с самой разной подготовкой, возможность быстро получить советы и ответы на свои вопросы от сообщества пользователей и многое другое.

По критериям эффективности Ubuntu Server 12.04 относится к системам разделения времени, по назначению – к системам общего назначения, поскольку предназначена для решения широкого круга задач, включая запуск различных приложений, разработку и отладку программ, работу с сетью и мультимедиа. Очень важно для сотрудников – перед ними поставлен широкий круг задач.

В Ubuntu Server 12.04 поддерживается создание нескольких пользователей, способных работать в ней как в разные моменты времени, так и в один, и имеет средства защиты информации каждого пользователя от несанкционированного доступа других пользователей. Следовательно, данная ОС поддерживает многопользовательский режим.

Ubuntu Server 12.04 как представитель Linux поддерживает

многозадачность, т.е. способ организации вычислительного процесса, при котором на одном процессоре попеременно выполняются сразу несколько программ. Это важно для рассматриваемой предметной области, поскольку любой сотрудник службы занятости может выполнять одновременно несколько профессиональных задач. Особо стоит отметить, что характер многозадачности вытесняющий, т.е. сама ОС принимает решение о переключении процессора с одного процесса на другой.

По характеру взаимодействия с пользователями Ubuntu Server 12.04 относится к классу диалоговых ОС, которые выполняют команды пользователя в интерактивном режиме. Это благоприятно сказывается на удобстве её использования.

Ubuntu Server 12.04 включает в себя возможность доступа к другим компьютерам локальной сети, работу с файловыми и другими серверами сети, следовательно, она является сетевой ОС, причём поддерживает основные сетевые протоколы, в частности TCP/IP. И сетевой, и терминальный режимы работы поддерживаются в полной степени.

Основанная на дистрибутиве Linux Ubuntu Server 12.04 имеет открытый исходный код, а по лицензии GNU GPL распространяется совершенно бесплатно, что не может не быть положительным моментом.

Данная ОС стандартными средствами поддерживает лишь файловую систему ext4, присущую всем ОС Linux, но, используя специальные приложения, можно добиться поддержки и файловых систем Windows, таких как FAT32 и NTFS.

Существуют как 32-разрядные версии Ubuntu Server 12.04, так и 64-разрядные.

Система защищена от внутренних и внешних ошибок, сбоев и отказов. Её действия предсказуемы, а приложения не наносят вред ОС. Следовательно, она соответствует критерию надёжности.

Ubuntu Server 12.04 соответствует критерию совместимости, поскольку пользовательский интерфейс совместим с существующими системами и

стандартами. Также для неё имеются средства (например, Wine), позволяющие выполнять значительное число основных прикладных программ, написанных для других операционных систем и широко в них используемых.

ОС Ubuntu Server 12.04 является полностью масштабируемой, поскольку её можно настроить для использования в разных вариантах, в зависимости от мощности вычислительной системы, от набора конкретных периферийных устройств, от роли, которую играет конкретный компьютер (сервер, рабочая станция или изолированный компьютер) от назначения компьютера (домашний, офисный, исследовательский и т.п.).

Сотрудники службы занятости любой квалификации смогут работать в данной ОС в качестве пользователей, однако может потребоваться также системный администратор для нужд конфигурирования, причём данный сотрудник может быть и внештатным.

В состав Ubuntu Server 12.04 входят бесплатные прикладные и офисные программы, например, OpenOffice.org, предоставляющие широкий спектр возможностей по работе с текстовой и графической информацией.

Следовательно, ОС Ubuntu Server 12.04 в значительной степени соответствует требованиям, выдвигаемым к ОС для данной предметной области.

Теперь рассмотрим ОС Microsoft Windows XP/2003, причём именно в таком двойном сочетании, поскольку версию XP нельзя использовать для нужд сервера, а лишь для рабочих станций пользователей. Версия 2003 предназначена непосредственно для серверных систем.

В общем же Microsoft Windows относится к системам общего назначения, поскольку предназначена для решения широкого круга задач, включая запуск различных приложений, разработку и отладку программ, работу с сетью и мультимедиа.

По критериям эффективности Microsoft Windows XP/2003 относится к системам разделения времени.

Microsoft Windows XP/2003 также поддерживает создание нескольких

пользователей и имеет средства защиты информации каждого пользователя от несанкционированного доступа других пользователей. – Эта ОС поддерживает многопользовательский режим.

Многозадачность в Microsoft Windows XP/2003 поддерживается, её характер – вытесняющий, т.е. сама ОС принимает решение о переключении процессора с одного процесса на другой.

Так же как и Ubuntu Server 12.04 по характеру взаимодействия с пользователями Microsoft Windows XP/2003 относится к классу диалоговых ОС, которые выполняют команды пользователя в интерактивном режиме. Это значительно повышает эргономичность её использования.

Microsoft Windows XP/2003 включает в себя возможность доступа к другим компьютерам локальной сети, работу с файловыми и другими серверами сети, следовательно, она является сетевой ОС, причём поддерживает основные сетевые протоколы, в частности TCP/IP. Однако и сетевой, и терминальный режимы работы поддерживаются в полной степени лишь в версии 2003. Windows XP поддерживает лишь сетевой режим работы.

Исходный код ОС компании Microsoft закрыт, поскольку это коммерческий проект, причём стоимость дистрибутивов не мала, особенно для версии 2003.

Данная ОС поддерживает лишь собственные файловые системы – FAT, FAT32 и NTFS. И хотя в последнее время появилось значительное число сторонних утилит, позволяющих получить доступ и к разделам с файловой системой Linux-систем, надёжность их работы не позволяет их использовать без риска.

Microsoft Windows XP/2003 выпускаются в 32- и 64-разрядных версиях.

Система соответствует критерию надёжности, потому что она защищена от внутренних и внешних ошибок, сбоев и отказов. Однако надёжность стоит на третьем месте после MacOS и Linux.

Microsoft Windows XP/2003 лишь отчасти соответствует критерию совместимости, поскольку хоть пользовательский интерфейс совместим с

существующими системами и стандартами, но средств, позволяющих выполнять прикладные программы, написанные для других операционных систем, не существует.

ОС от компании Microsoft не является масштабируемой, поскольку каждая из версий предназначена лишь для конкретных целей – рабочая станция или сервер.

Поскольку Windows XP/2003 является наиболее популярной в настоящей момент ОС, сложностей при работе с ней у сотрудников службы занятости не возникнет. Но ввиду наличия отдельного сервера необходима будет должность системного администратора для настройки и конфигурирования первого. Данный сотрудник может быть внештатным.

Дистрибутив Microsoft Windows XP/2003 не содержит мощных пакетов для работы с офисными документами и графическими изображениями. Хотя наиболее популярные пакеты для данной системы являются продуктами той же компании и распространяются на коммерческой основе, в последнее время существует множество альтернативных свободно распространяемых продуктов, обладающих схожим функционалом.

Принимая во внимание рассмотренные критерии, можно сделать вывод о том, что для данной предметной области ОС компании Microsoft подходит, но в меньшей степени, чем Ubuntu Server 12.04. В связи с этим и была выбрана операционная система на базе Linux ядра.

3.3 Информационное обеспечение и разработка интерфейса

Одно из основных требований к задаче являлось то, что нужно разработать отзывчивый и дружелюбный интерфейс программы, давайте разберем что такое интерфейс программы, и с помощью чего мы его разрабатывали.

Интерфе́йс по́льзователя, он же **по́льзовательский интерфейс** (UI — англ. user interface) — разновидность интерфейсов, в котором одна сторона представлена человеком (пользователем), другая — машиной/устройством. Представляет собой совокупность средств и методов, при помощи которых

пользователь взаимодействует с различными, чаще всего сложными, машинами, устройствами и аппаратурой.

Рис. 3.1 Начальное окно разрабатываемой системы управления Scpp

Весьма часто термин применяется по отношению к компьютерным программам, однако под ним может подразумеваться набор средств, методов и правил взаимодействия любой системы, управляемой человеком.

Несколько широко распространённых примеров:

•меню на экране телевизора + пульт дистанционного управления;

•дисплей электронного аппарата (автомагнитолы, часов) + набор кнопок и переключателей для настройки;

•приборная панель (автомобиля, самолёта) + рычаги управления.

Интерфейс двунаправленный (интерактивный) — когда устройство, получив команды от пользователя и исполнив их, выдаёт информацию пользователю наличествующими у неё средствами — визуальными, звуковыми, тактильными и т. п. (приняв которую, пользователь выдаёт устройству последующие команды предоставленными в его распоряжение средствами: кнопки, переключатели, регуляторы, сенсоры, голосом, и т. д.).

Поскольку интерфейс есть совокупность, то есть он состоит из элементов, которые, сами по себе, также могут состоять из элементов (так, экран дисплеяможет содержать в себе другие окна, которые, в свою очередь, могут

содержать панели, кнопки и прочие интерфейсные элементы).

Особое и отдельное внимание в интерфейсе пользователя традиционно уделяется его эффективности и удобству пользования (юзабельности).Понятный, удобный, дружественный — его основные характеристики.

Под совокупностью средств и методов интерфейса пользователя подразумеваются:

Средства:

•вывода информации из устройства к пользователю — весь доступный диапазон воздействий на организм человека (зрительных, слуховых, тактильных, обонятельных и тд.) — экраны (дисплеи, проекторы) и лампочки, динамики, зуммеры и сирены, вибромоторы и тд. и тп.

•ввода информации/команд пользователем в устройство — множество всевозможных устройств для контроля состояния человека — кнопки, переключатели, потенциометры, датчики положения и движения, сервоприводы, жесты лицом и руками, даже съём мозговой активностипользователя.

По наличию тех или иных средств ввода, интерфейсы разделяются на типы — жестовый, голосовой, брэйн, и тд., возможны смешанные варианты. Средства эти должны быть необходимыми и достаточными, быть удобными и практичными, расположенными и скомпонованными разумно и понятно, соответствовать физиологии человека, не должны приводить к негативным последствиям для организма пользователя (всё это входит в понятие**эргономики**).

Методы:

•набор правил, заложенных разработчиком устройства, согласно которым совокупность действий пользователя должна привести к необходимой реакции устройства и выполнения требуемой задачи — т. н. логический интерфейс Правила эти должны быть достаточно ясны для понимания, естественны и легки для запоминания (всё это входит в понятие **юзабилити**)

Увеличение в устройстве (при равной функциональности) средств ввода-вывода даёт упрощение построения методов управления и упрощение правил пользования, но зато приводит к сложности восприятия информации пользователем — интерфейс становится перегруженным. И наоборот — уменьшение средств отображения и контроля приводит к усложнению правил управления — каждый элемент несёт на себе слишком много функций. Потому проектировщики интерфейсов стараются принять компромиссное решение между этими двумя крайностями в каждом отдельном случае.

Безопасность Одним из основных направлений исследований в области обеспечения безопасности пользовательских интерфейсов, и, в частности, визуальных интерфейсов пользователя, является разработка моделей информационной безопасности при условии комплексного учета информационных, функциональных, психофизиологических и экологических аспектов безопасности. Это связано, прежде всего, с включением информационного фактора в состав факторов среды систем человек-компьютер и информационным характером почти всех происходящих в области распространения ИП процессов. Наименее разработанным областям проблематики защиты информации в системе человек-компьютер (СЧК) соответствуют такие угрозы, как:

- искажение воспринимаемой пользователем информации за счет ее зашумления источниками среды на рабочем месте пользователя;
- потеря или искажение воспринимаемой пользователем информации из-за физической, семантической или синтаксической несогласованности ее представления пользователю;
- искажение представлений пользователя о реальном состоянии объекта управления за счет скрытых информационных воздействий и неадекватное принятие им решений в процессе решения задач в рамках СЧК.

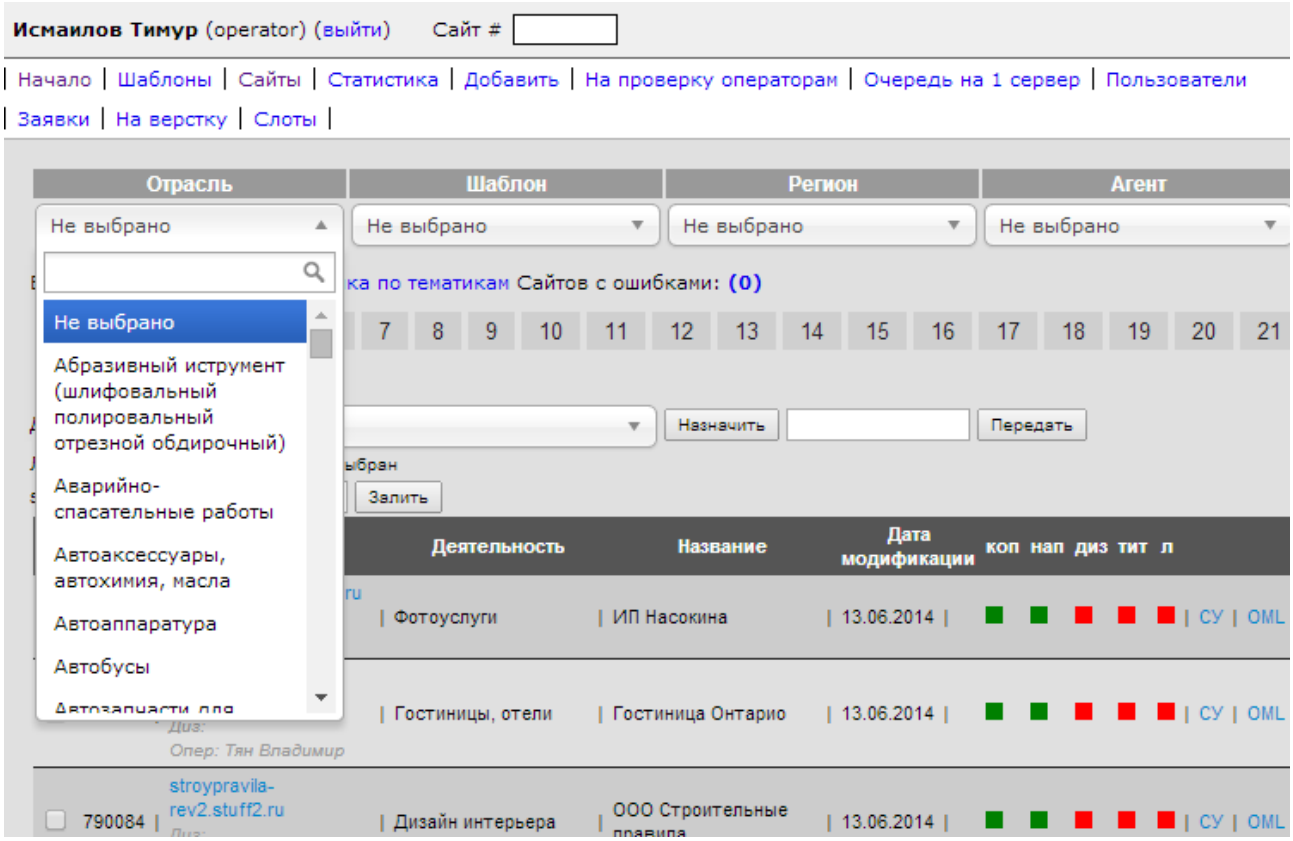

Рис 3.2. Scpp. Окно управления списком сайтов

Интерфейс пользователя компьютерного приложения включает:

•средства отображения информации, отображаемую информацию, форматы и коды;

•командные режимы, язык «пользователь — интерфейс»;

•устройства и технологии ввода данных;

•диалоги, взаимодействие и транзакции между пользователем и компьютером, обратную связь с пользователем;

•поддержку принятия решений в конкретной предметной области;

•порядок использования программы и документацию на неё.

Пользовательский интерфейс часто понимают только как внешний вид программы. Однако, на деле пользователь воспринимает через него всю программу в целом, а значит, такое понимание является слишком узким. В действительности ПИ объединяет в себе все элементы и компоненты программы, которые способны оказывать влияние на взаимодействие

пользователя с программным обеспечением (ПО), это не только экран, который видит пользователь.

К этим элементам относятся:

- набор задач пользователя, которые он решает при помощи системы;
- используемая системой метафора (например, рабочий стол в MS Windows®);
- элементы управления системой;
- навигация между блоками системы;
- визуальный (и не только) дизайн экранов программы;
- средства отображения информации, отображаемая информация и форматы;
- устройства и технологии ввода данных;
- диалоги, взаимодействие и транзакции между пользователем и компьютером;
- обратная связь с пользователем;
- поддержка принятия решений в конкретной предметной области;
- порядок использования программы и документация на нее.

Для упрощения восприятия функции программы пользователем при разработке пользовательского интерфейса желательно использовать метафоры.

Веб-интерфе́йс — это совокупность средств, при помощи которых пользователь взаимодействует с веб-сайтом или любым другим приложением черезбраузер. Веб-интерфейсы получили широкое распространение в связи с ростом популярности всемирной паутины и соответственно — повсеместного распространения веб-браузеров.

Одним из основных требований к веб-интерфейсам является их одинаковый внешний вид и одинаковая функциональность при работе в различных браузерах.

Классическим и наиболее популярным методом создания веб-интерфейсов является использование HTML с применением CSS и JavaScript'а. Однако различная реализация HTML, CSS, DOM и других спецификаций в браузерах

вызывает проблемы при разработке веб-приложений и их последующей поддержке. Кроме того, возможность пользователя настраивать многие параметры браузера (например, размер шрифта, цвета, отключение поддержки сценариев) может препятствовать корректной работе интерфейса.

Другой (менее универсальный) подход заключается в использовании Adobe Flash, Silverlight или Java-апплетов для полной или частичной реализации пользовательского интерфейса. Поскольку большинство браузеров поддерживает эти технологии (как правило, с помощью плагинов), Flash- или Java-приложения могут выполняться с легкостью. Так как они предоставляют программисту больший контроль над интерфейсом, они способны обходить многие несовместимости в конфигурациях браузеров, хотя несовместимость между Java или Flash реализациями на стороне клиента может приводить к различным осложнениям.

В настоящее время набирает популярность новый подход к разработке интерфейсной части веб-приложений, называемый Ajax. При использовании Ajax интерфейсы не перезагружаются целиком, а лишь догружают необходимые данные с сервера, что делает их более интерактивными и производительными.

Веб-интерфейсы удобны тем, что дают возможность вести совместную работу сотрудникам, не находящимся в одном офисе (например, веб-интерфейсы часто используются для заполнения различных баз данных или публикации материалов в интернет-СМИ)

3.4 Реализация программного продукта

Разрабо́тка програ́ммного обеспе́чения (software development) — это род деятельности (профессия) и процесс, направленный на создание и поддержание работоспособности, качества и надежности программного обеспечения, используя технологии, методологию и практики из информатики, управления проектами,математики, инженерии и других областей знания

Сложность разработки ПО

Как и другие традиционные инженерные дисциплины, разработка программного обеспечения имеет дело с проблемами качества, стоимости и надёжности. Некоторые программы содержат миллионы строк исходного кода, которые, как ожидается, должны правильно исполняться в изменяющихся условиях. Сложность ПО сравнима со сложностью наиболее сложных из современных машин, таких как самолёты.

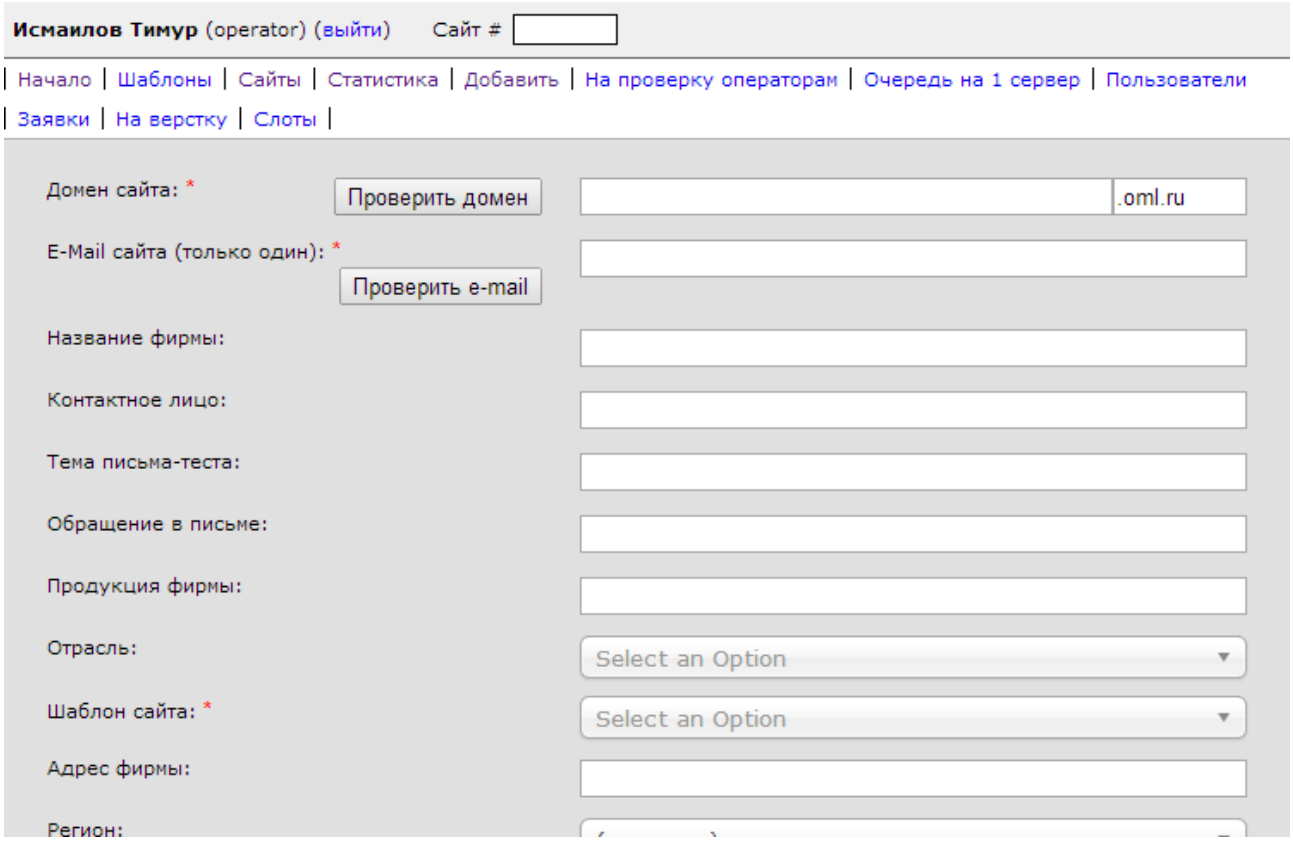

Рис 3.3 Scpp. Окно создания сайта

Разработка программного обеспечения может быть разделена на несколько разделов. Это:

1.Требования к программному обеспечению: извлечение, анализ, спецификация и ратификация требований для программного обеспечения.

2.Проектирование программного обеспечения: проектирование программного обеспечения средствами Автоматизированной Разработки Программного Обеспечения (CASE) и стандарты формата описаний, такие как Унифицированный Язык Моделирования (UML), используя различные

подходы: проблемно-ориентированное проектирование и т.д..

3.Инженерия программного обеспечения: создание программного обеспечения с помощью языков программирования.

4.Тестирование программного обеспечения: поиск и исправление ошибок в программе.

5.Обслуживание программного обеспечения: программные системы часто имеют проблемы совместимости и переносимости, а также нуждаются в последующих модификациях в течение долгого времени после того, как закончена их первая версия. Подобласть имеет дело с этими проблемами.

6.Управление конфигурацией программного обеспечения: так как системы программного обеспечения очень сложны и модифицируются в процессе эксплуатации, их конфигурации должны управляться стандартизированным и структурированным методом.

7.Управление разработкой программного обеспечения: управление системами программного обеспечения имеет заимствования из управления проектами, но есть нюансы, не встречающиеся в других дисциплинах управления.

8.Процесс разработки программного обеспечения: процесс построения программного обеспечения горячо обсуждается среди практиков, основными парадигмами считаются agile или waterfall.

9.Инструменты разработки программного обеспечения, см. CASE: методика оценки сложности системы, выбора средств разработки и применения программной системы.

10.Качество программного обеспечения: методика оценки критериев качества программного продукта и требований к надёжности.

11.Локализация программного обеспечения, ветвь языковой промышленности.

На протяжении нескольких десятилетий стоит задача поиска повторяемого, предсказуемого процесса или методологии, которая бы улучшила продуктивность, качество и надёжность разработки. Одни пытались систематизировать и формализовать этот, по-видимому, малопредсказуемый процесс. Другие применяли к нему методы управления проектами и методы

программной инженерии. Третьи считали, что без постоянного контроля со стороны заказчика разработка ПО выходит из-под контроля, съедая лишнее время и средства.

Опыт управления разработкой программ отражается в соответствующих руководствах, обычаях и стандартах. Если при разработке используется несколько стандартов и нормативных документов, то имеет смысл составить профиль.

Информатика как научная дисциплина предлагает и использует на базе методов структурного программирования технологию надежной разработки программного обеспечения, используя тестирование программ и их верификацию на основе методов доказательного программирования для систематического анализа правильности алгоритмов и разработки программ без алгоритмических ошибок.

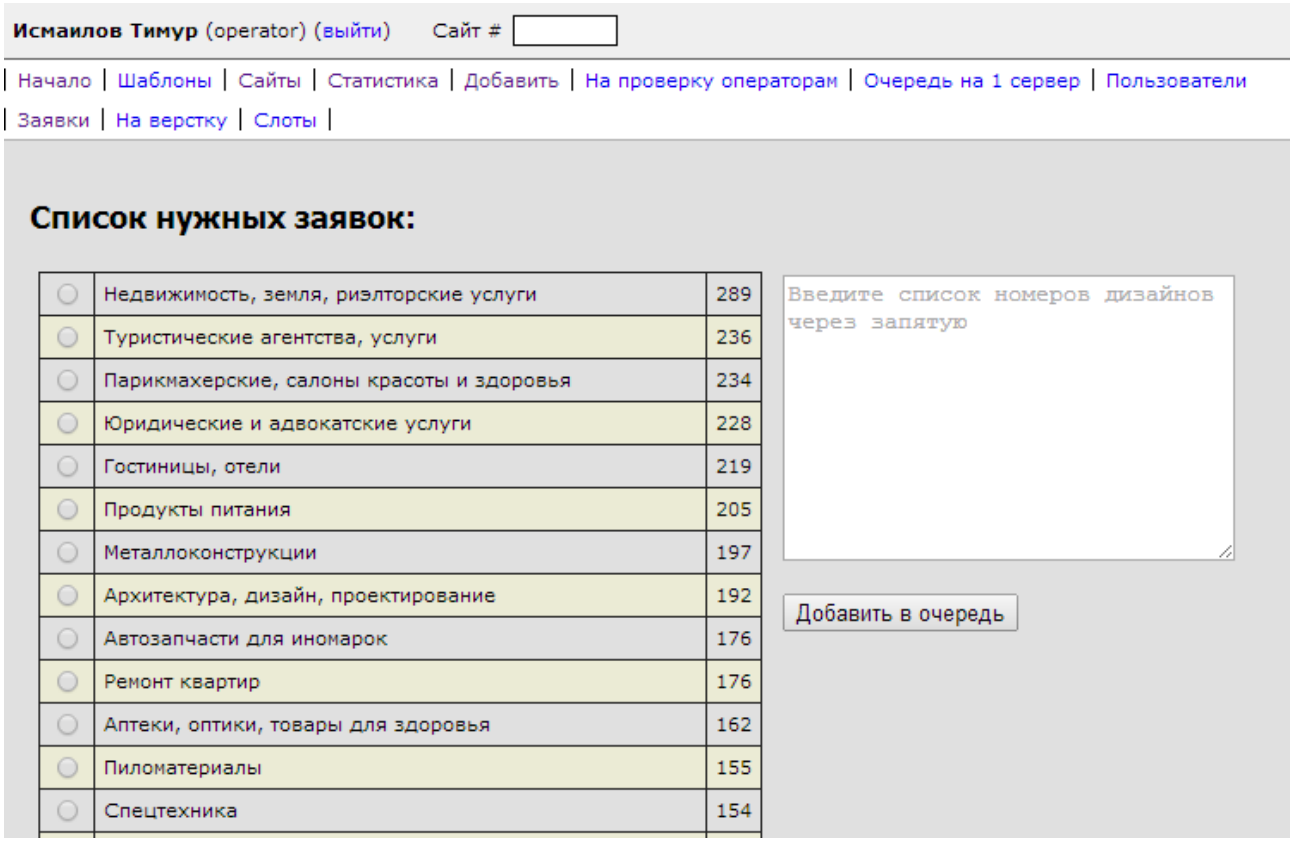

Рис 3.4 Scpp. Список добавления заявок на создание дизайна

Данная методология направлена на решение задач на ЭВМ, аналогичной технологии разработки алгоритмов и программ, используемой на олимпиадах

по программированию отечественными студентами и программистами с использованием тестирования и структурного псевдокода для документирования программ в корпорации IBM с 70-х годов.

Методология структурного проектирования программного обеспечения может использоваться с применением самых различных языков и средств программирования для разработки надёжных программ самого различного назначения. Одним из таких проектов была разработка бортового программного обеспечения для космического корабля «Буран», в котором впервые использовался бортовой компьютер для автоматического управления аппарата, совершившего успешный старт и посадку космического корабля.

При выборе методологии разработки программного обеспечения следует руководствоваться тем, что сложность методологии сравнима со сложностью структуры программного продукта, и неоправданная для продукта данной сложности сложность методологии только неоправданно увеличит стоимость разработки. Примером современной методологии проектирования может быть проблемно-ориентированное проектирование.

Наиболее распространёнными проблемами, возникающими в процессе разработки ПО, считают:

•Недостаток прозрачности. В любой момент времени сложно сказать, в каком состоянии находится проект и каков процент его завершения. Данная проблема возникает при недостаточном планировании структуры (или архитектуры) будущего программного продукта, что чаще всего является следствием отсутствия достаточного финансирования проекта: программа нужна, сколько времени займёт разработка, каковы этапы, можно ли какие-то этапы исключить или сэкономить — следствием этого процесса является то, что этап проектирования сокращается.

•Недостаток контроля. Без точной оценки процесса разработки срываются графики выполнения работ и превышаются установленные бюджеты. Сложно оценить объём выполненной и оставшейся работы. Данная проблема возникает на этапе, когда проект, завершённый более чем

наполовину, продолжает разрабатываться после дополнительного финансирования без оценки степени завершённости проекта.

•Недостаток трассировки.

•Недостаток мониторинга. Невозможность наблюдать ход развития проекта не позволяет контролировать ход разработки в реальном времени. С помощью инструментальных средств менеджеры проектов принимают решения на основе данных, поступающих в реальном времени.
Данная проблема возникает в условиях, когда стоимость обучения менеджмента владению инструментальными средствами сравнима со стоимостью разработки самой программы.

•Неконтролируемые изменения. У потребителей постоянно возникают новые идеи относительно разрабатываемого программного обеспечения. Влияние изменений может быть существенным для успеха проекта, поэтому важно оценивать предлагаемые изменения и реализовывать только одобренные, контролируя этот процесс с помощью программных средств.
Данная проблема возникает вследствие нежелания конечного потребителя использовать те или иные программные среды. Например, когда при создании клиент-серверной системы потребитель предъявляет требования не только к операционной системе на компьютерах-клиентах, но и на компьютере-сервере.

•Недостаточная надёжность. Самый сложный процесс — поиск и исправление ошибок в программах на ЭВМ. Поскольку число ошибок в программах заранее неизвестно, то заранее неизвестна и продолжительность отладки программ и отсутствие гарантий отсутствия ошибок в программах. Следует отметить, что привлечение доказательного подхода к проектированию ПО позволяет обнаружить ошибки в программе до её выполнения. В этом направлении много работали Кнут, Дейкстра и Вирт. Профессор Вирт при разработке Паскаля и Оберона за счет строгости их синтаксиса добился математической доказуемости завершаемости и правильности программ, написанной на этих языках.
Данная проблема возникает при неправильном выборе средств разработки. Например, при попытке создать программу, требующую средств высокого

уровня, с помощью средств низкого уровня. Например, при попытке создать средства автоматизации с СУБД на ассемблере. В результате исходный код программы получается слишком сложным и плохо поддающимся структурированию.

•Неправильный выбор методологии разработки программного обеспечения. Процесс выбора необходимой методологии может проблемно отразиться на всех показателях программного обеспечения - это его гибкость, стоимость и функциональность. Так называемые гибкие методологии разработкипомогают решить основные проблемы, однако, стоит отметить, что и каскадная модель (waterfall) так же имеет свои преимущества

•Отсутствие гарантий качества и надежности программ из-за отсутствия гарантий отсутствия ошибок в программах вплоть до формальной сдачи программ заказчикам.

Данная проблема не является проблемой, относящейся исключительно к разработке ПО. Гарантия качества — это проблема выбора поставщикатовара (не продукта).

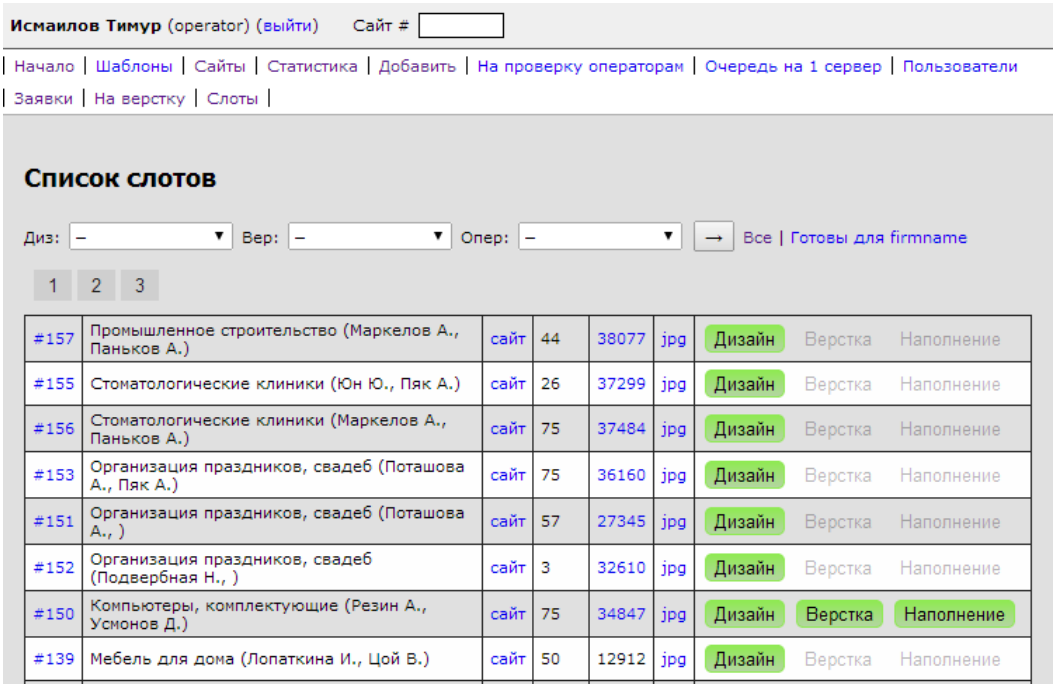

Рис 3.5 Список заявок. Интерфейс для редактирования и контролирования заявок

Для реализации задачи было решено вести структурное и логическое

программирование. **Логи́ческое программи́рование** — парадигма программирования, основанная на автоматическом доказательстве теорем, а также раздел дискретной математики, изучающий принципы логического вывода информации на основе заданных фактов и правил вывода. Логическое программирование основано на теории и аппарате математической логики с использованием математических принципов резолюций.

Самым известным языком логического программирования является Prolog.

Первым языком логического программирования был язык Planner (см. обзор Шапиро (Ehud Shapiro) [1989]), в котором была заложена возможность автоматического вывода результата из данных и заданных правил перебора вариантов (совокупность которых называлась планом). Planner использовался для того, чтобы понизить требования к вычислительным ресурсам (с помощью метода backtracking) и обеспечить возможность вывода фактов, без активного использования стека. Затем был разработан язык Prolog, который не требовал плана перебора вариантов и был, в этом смысле, упрощением языка Planner.

От языка Planner также произошли логические языки программирования QA-4, Popler, Conniver и QLISP. Языки программирования Mercury, Visual Prolog, Oz и Fril произошли уже от языка Prolog. На базе языка Planner было разработано также несколько альтернативных языков логического программирования, не основанных на методе поиска с возвратами (backtracking), например, Ether (см. обзор Шапиро [1989]).

А вот что такое структурное программирование – рассмотрим ниже:

Структу́рное программи́рование — методология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков. Предложена в 70-х годах XX века Э. Дейкстрой, разработана и дополнена Н. Виртом.

В соответствии с данной методологией

1.Любая программа представляет собой структуру, построенную из трёх типов базовых конструкций:

- **последовательное исполнение** — однократное выполнение операций в

том порядке, в котором они записаны в тексте программы;

- **ветвление** — однократное выполнение одной из двух или более операций, в зависимости от выполнения некоторого заданного условия;
- **цикл** — многократное исполнение одной и той же операции до тех пор, пока выполняется некоторое заданное условие (условие продолжения цикла).

В программе базовые конструкции могут быть вложены друг в друга произвольным образом, но никаких других средств управления последовательностью выполнения операций не предусматривается.

2.Повторяющиеся фрагменты программы (либо не повторяющиеся, но представляющие собой логически целостные вычислительные блоки) могут оформляться в виде т. н. подпрограмм (процедур или функций). В этом случае в тексте основной программы, вместо помещённого в подпрограмму фрагмента, вставляется инструкция **вызова подпрограммы**. При выполнении такой инструкции выполняется вызванная подпрограмма, после чего исполнение программы продолжается с инструкции, следующей за командой вызова подпрограммы.

3.Разработка программы ведётся пошагово, методом «сверху вниз».

Сначала пишется текст основной программы, в котором, вместо каждого связного логического фрагмента текста, вставляется вызов подпрограммы, которая будет выполнять этот фрагмент. Вместо настоящих, работающих подпрограмм, в программу вставляются «заглушки», которые ничего не делают. Полученная программа проверяется и отлаживается. После того, как программист убедится, что подпрограммы вызываются в правильной последовательности (то есть общая структура программы верна), подпрограммы-заглушки последовательно заменяются на реально работающие, причём разработка каждой подпрограммы ведётся тем же методом, что и основной программы. Разработка заканчивается тогда, когда не останется ни одной «затычки», которая не была бы удалена. Такая последовательность гарантирует, что на каждом этапе разработки программист одновременно имеет

дело с обозримым и понятным ему множеством фрагментов, и может быть уверен, что общая структура всех более высоких уровней программы верна. При сопровождении и внесении изменений в программу выясняется, в какие именно процедуры нужно внести изменения, и они вносятся, не затрагивая части программы, непосредственно не связанные с ними. Это позволяет гарантировать, что при внесении изменений и исправлении ошибок не выйдет из строя какая-то часть программы, находящаяся в данный момент вне зоны внимания программиста.

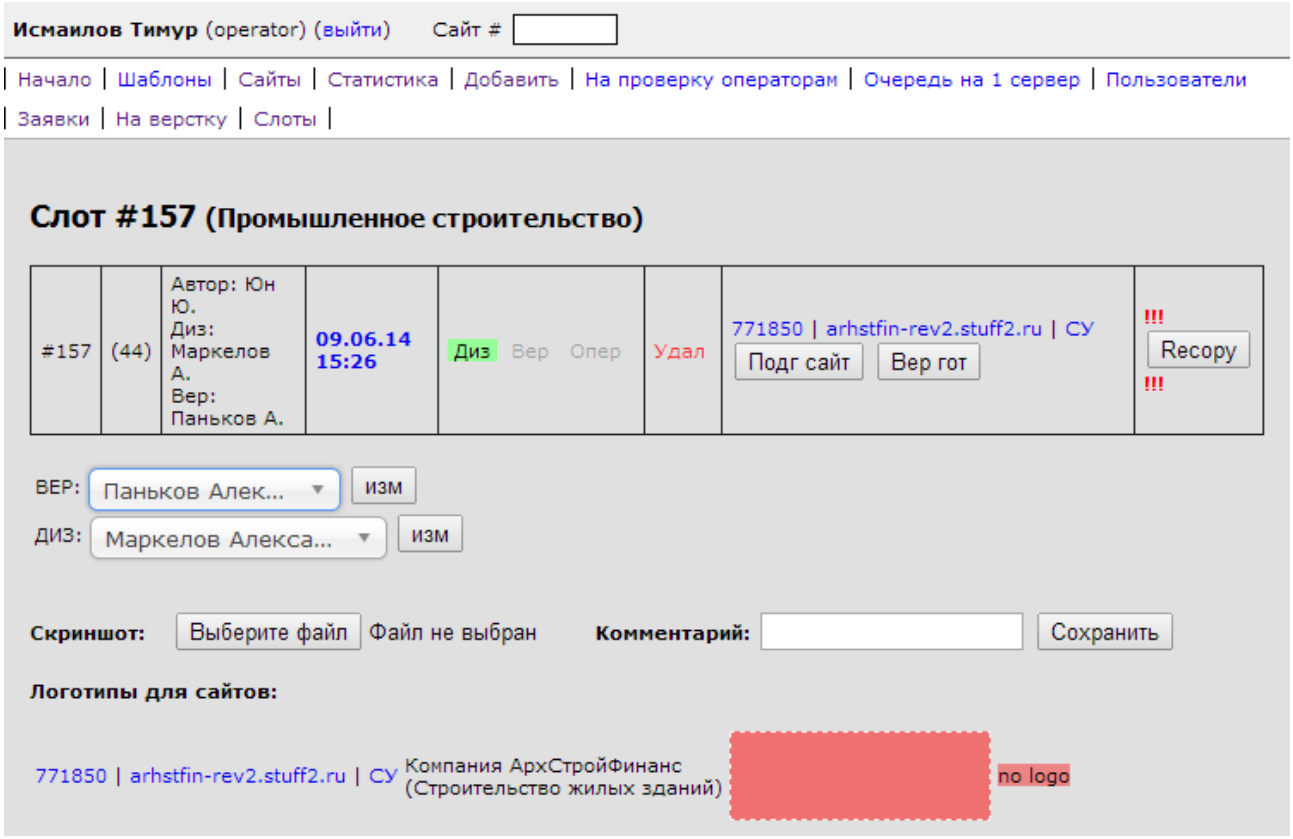

Рис 3.6. Окно редактирования так называемого слота (заполненной заявки)

Методология структурного программирования появилась как следствие возрастания сложности решаемых на компьютерах задач, и соответственного усложнения программного обеспечения. В 70-е годы XX века объёмы и сложность программ достигли такого уровня, что «интуитивная» (неструктурированная, или «рефлекторная») разработка программ, которая была нормой в более раннее время, перестала удовлетворять потребностям

практики. Программы становились слишком сложными, чтобы их можно было нормально сопровождать, поэтому потребовалась какая-то систематизация процесса разработки и структуры программ.

Наиболее сильной критике со стороны разработчиков структурного подхода к программированию подвергся оператор GOTO (оператор безусловного перехода), имевшийся тогда почти во всех языках программирования. Неправильное и необдуманное использование произвольных переходов в тексте программы приводит к получению запутанных, плохо структурированных программ (т. н. спагетти-кода), по тексту которых практически невозможно понять порядок исполнения и взаимозависимость фрагментов.

Следование принципам структурного программирования сделало тексты программ, даже довольно крупных, нормально читаемыми. Серьёзно облегчилось понимание программ, появилась возможность разработки программ в нормальном промышленном режиме, когда программу может без особых затруднений понять не только её автор, но и другие программисты. Это позволило разрабатывать достаточно крупные для того времени программные комплексы силами коллективов разработчиков, и сопровождать эти комплексы в течение многих лет, даже в условиях неизбежных изменений в составе персонала.

Методология структурной разработки программного обеспечения была признана «самой сильной формализацией 70-х годов». После этого слово «структурный» стало модным в отрасли, и его начали использовать везде, где надо и где не надо. Появились работы по «структурному проектированию», «структурному тестированию», «структурному дизайну» и так далее. В общем, произошло примерно то же самое, что происходило в 90-х годах и происходит в настоящее время с терминами «объектный», «объектно-ориентированный» и «электронный».

Перечислим некоторые достоинства структурного программирования:

1.Структурное программирование позволяет значительно сократить число вариантов построения программы по одной и той же спецификации, что

значительно снижает сложность программы и, что ещё важнее, облегчает понимание её другими разработчиками.

2.В структурированных программах логически связанные операторы находятся визуально ближе, а слабо связанные — дальше, что позволяет обходиться без блок-схем и других графических форм изображения алгоритмов (по сути, сама программа является собственной блок-схемой).

3.Сильно упрощается процесс тестирования и отладки структурированных программ.

Вот, придерживаясь именно таких парадигм программирования и было достигнута правильная и быстрая реализация поставленной задачи.

Выводы.

В разделе «Проектная часть», нам удалось составить структуру проекта для работы с сайтами, описать принципы построения и написани высоконагружженых систем управления с использованием современных подходов, шаблонов проектирования. Сдесь же мы рассмотрели сам процесс разработки высокнагруженной системы управления данными. Нам удалось создать удобный и простой интерфейс для довольно-таки сложной системы управления данными, созданию и редактированию сайтов.

С помощью языка программирования PHP удалось быстро создать качественный сервис для внутренних работ компании с учетом всех особенностей и принципов построения бизнес процесса. Благодаря проведенным раннее аналитическому анализу и теоретической подготовки удалось сильно сократить потребовавшееся на реализацию и внедрение проекта.

В итоге мы получили высоконагруженную систему управления содержимым сайта основными особенностиями которой является:

- Отказоустойчивость
- Быстродействие внутренних процессов
- Низкая нагрузка на сервер

•Удобный интерфейс и простое управление основными задачами

Работа над реализацийе поставленной задачи проходила в несколько этапов:

1. Подготовительный этап

На этом этапе необходимо сформировать основную идею системы управления содержимым сайта. На этом же этапе составляется ТЗ (Техническое задание).

2. Разработка макета

После постановки задач будущего сайта, наступает следующий этап — разработка макета. Ориентируясь по пунктам ТЗ разрабатывается будущий шаблон, или шаблоны, из которых можно выбрать нужный вариант. На данном этапе определяется как будет выглядеть дизайн в целом, какие графические элементы будут использованы, а так же какая структура будет у страниц и какие цветовые решения будут на нем присутствовать.

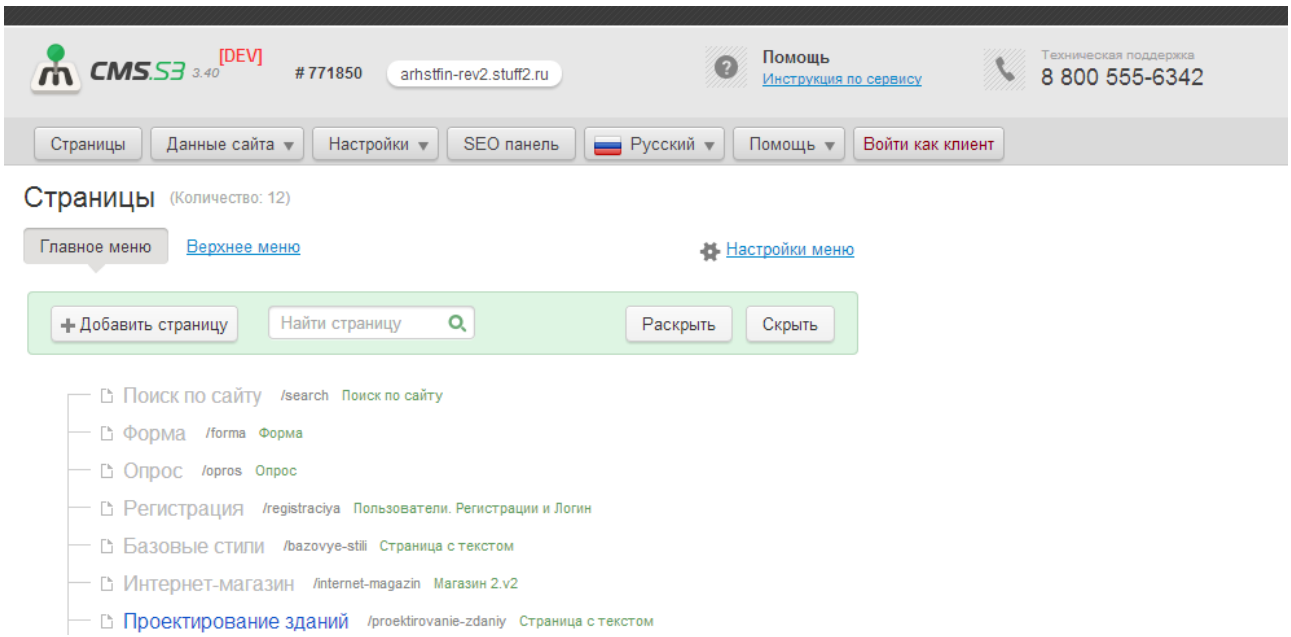

Рис 3.7 Макет системы управления сайтом CMS. Является одним из модулей разрабатываемой системы

После чего предоставляется в формате PSD (со всеми слоями) и в виде картинке, тут можно использовать любой формат (JPEG, PNG), чтобы в дальнейшем передать работу в руки верстальщика. Разработка дизайна сайта

обычно занимает от нескольких дней, до нескольких недель, очень сложные работы могут проводиться даже месяц.

3. Верстка

Чтобы нарезать готовый макет на части, и в дальнейшем прикрепить на сайт, потребуется верстальщик. Что такое верстка? Это процесс написания HTML и CSS кодов для веб-страниц. Каждый код отвечает за то, чтобы все элементы на странице размещались там, где нужно. Верстальщик должен знать все особенности браузеров, так как для каждого браузера необходим свой подход верстки.

Виды верстки при создании сайта:

- Блочная. Преимущества блочной верстки заключается в том, что она правильно отображает все элементы сайта на разных устройствах (например, на мобильных телефонах). Компактный код, все элементы весят меньше, а значит страница будет загружаться быстро. Из недостатков можно отметить, что при использовании разных браузеров верстка может поплыть.
- Табличная. В табличной верстке можно создавать колонки, таблицы в браузерах выглядят почти одинаково. Недостатки: индексация такого сайта очень медленная, долго загружаются страницы и код такой верстки слишком большой.

4. Программирование

На этом этапе идет разработка механизмов системы администрирования сайта, которая в будущем позволит сотрудникам компании менять/обновлять информацию на сайте. То есть создается рабочая версия сайта , готовая к наполнению текстов и графических материалов.

IV. ОХРАНА ТРУДА И ТЕХНИКА БЕЗОПАСНОСТИ

4.1 Основы ТБ при работе с ПК

Требования по электрической безопасности. Персональный компьютер — электроприбор. От прочих электроприборов он отличается тем, что для него предусмотрена возможность длительной эксплуатации без отключения от электрической сети. Кроме обычного режима работы компьютер может находиться в режиме работы с пониженным электропотреблением или в дежурном режиме ожидания запроса. В связи с возможностью продолжительной работы компьютера без отключения от электросети следует уделить особое внимание качеству организации электропитания.

1.Недопустимо использование некачественных и изношенных компонентов в системе электроснабжения, а также их суррогатных заменителей: розеток, удлинителей, переходников, тройников. Недопустимо самостоятельно модифицировать розетки для подключения вилок, соответствующих иным стандартам. Электрические контакты розеток не должны испытывать механических нагрузок, связанных с подключением массивных компонентов (адаптеров, тройников и т. п.).

2.Все питающие кабели и провода должны располагаться с задней стороны компьютера и периферийных устройств. Их размещение в рабочей зоне пользователя недопустимо.

3.Запрещается производить какие-либо операции, связанные с подключением, отключением или перемещением компонентов компьютерной системы без предварительного отключения питания.

4.Компьютер не следует устанавливать вблизи электронагревательных приборов и систем отопления.

5.Недопустимо размещать на системном блоке, мониторе и периферийных устройствах посторонние предметы: книги, листы бумаги, салфетки, чехлы для защиты от пыли. Это приводит к постоянному или временному

перекрытию вентиляционных отверстий.

6.Запрещается внедрять посторонние предметы в эксплуатационные или вентиляционные отверстия компонентов компьютерной системы.

Особенности электропитания монитора. Монитор имеет элементы, способные сохранять высокое напряжение в течение длительного времени после отключения от сети. Вскрытие монитора пользователем недопустимо ни при каких условиях. Это не только опасно для жизни, но и технически бесполезно, так как внутри монитора нет никаких органов, регулировкой или настройкой которых пользователь мог бы улучшить его работу. Вскрытие и обслуживание мониторов может производиться только в специальных мастерских.

Особенности электропитания системного блока.

Все компоненты системного блока получают электроэнергию от блока питания. Блок питания ПК — это автономный узел, находящийся в верхней части системного блока. Правила техники безопасности не запрещают вскрывать системный блок, например при установке дополнительных внутренних устройств или их модернизации, но это не относится к блоку питания. Блок питания компьютера — источник повышенной пожаро-опасности, поэтому вскрытию и ремонту он подлежит только в специализированных мастерских. Блок питания имеет встроенный вентилятор и вентиляционные отверстия. В связи с этим в нем неминуемо накапливается пыль, которая может вызвать короткое замыкание. Рекомендуется периодически (один - два раза в год) с помощью пылесоса удалять пыль из блока питания через вентиляционные отверстия без вскрытия системного блока. Особенно важно производить эту операцию перед каждой транспортировкой или наклоном системного блока.

Система гигиенических требований.

Длительная работа с компьютером может приводить к расстройствам состояния здоровья. Кратковременная работа с компьютером, установленным с грубыми нарушениям гигиенических норм и правил, приводит к повышенному утомлению. Вредное воздействие компьютерной системы на организм человека является комплексным. Параметры монитора оказывают влияние на органы

зрения. Оборудование рабочего места влияет на органы опорно-двигательной системы. Характер расположения оборудования в компьютерном классе и режим его использования влияет как на общее психофизиологическое состояние организма, так и им органы зрения.

Требования к видеосистеме.

В прошлом монитор рассматривали м основном как источник вредных излучений, воздействующих прежде всего на глаза. Сегодня такой подход считается недостаточным. Кроме вредных электромагнитных излучений (которые на современных мониторах понижены до сравнительно безопасного уровня) должны учитываться параметры качества изображения, а они определяются не только монитором, но и видеоадаптером, то есть всей видеосистемы в целом.

1.Монитор компьютера должен удовлетворять следующим между народным стандартам безопасности:

- по уровню электромагнитных излучений — TCO 95;
- по параметрам качества изображения (яркость, контрастность, мерцание, антибликовые свойства и др.) — TCO 99.

Узнать о соответствии конкретной модели данным стандартам можно в сопроводительной документации. Для работы с мониторами, удовлетворяющими данным стандартам, специальные защитные экраны не требуется.

2.На рабочем месте монитор должен устанавливаться таким образом, чтобы исключить возможность отражения от его экрана в сторону пользователя источников общего освещения помещения.

3.Расстояние от экрана монитора до глаз пользователя должно составлять от 50 до 70 см. Не надо стремиться отодвинуть монитор как можно дальше от глаз, опасаясь вредных излучений (по бытовому опыту общения с телевизором), потому что для глаза важен также угол обзора наиболее характерных объектов. Оптимально, размещение монитора на расстоянии 1,5 D от глаз пользователя, где D — размер экрана монитора, измеренный по

диагонали. Сравните эту рекомендацию с величиной 3...5 D, рекомендованной для бытовых телевизоров, и сопоставьте размеры символов на экране монитора (наиболее характерный объект, требующий концентрации внимания) с размерами объектов, характерных для телевидения (изображения людей, сооружений, объектов природы). Завыщенное расстояния от глаз до монитора приводит к дополнительному напряжению органов зрения, сказывается на затруднении перехода от работы с монитором к работе с книгой и проявляется в преждевременном развитии дальнозоркости.

4.Важным параметром является частота кадров, которая зависит от свойств монитора, видеоадаптера и программных настроек видеосистемы. Для работы с текстами минимально допустима частота кадров 72 Гц. Для работы с графикой рекомендуется частота кадров от 85 Гц и выше.

Требования к рабочему месту.

В требования к рабочему месту входят требования к рабочему столу, посадочному месту (стулу, креслу), Подставкам для рук и ног. Несмотря на кажущуюся простоту, обеспечить правильное размещение элементов компьютерной системы и правильную посадку пользователя чрезвычайно трудно. Полное решение проблемы требует дополнительных затрат, сопоставимых по величине со стоимостью отдельных узлов компьютерной системы, поэтому и биту и на производстве этими требованиями часто пренебрегают.

Несмотря на то, что школьники проводят в компьютерном классе сравнительно немного времени, обучить их правильной гигиене труда на достойном примере очень важно, чтобы полезные навыки закрепились на всю жизнь. Это не просто требование гигиены, а требование методики.

1.Монитор должен быть установлен прямо перед пользователем и не требовать поворота головы или корпуса тела.

2.Рабочий стол и посадочное место должны иметь такую высоту, чтобы уровень глаз пользователя находился чуть выше центра монитора. На экран

монитора следует смотреть сверху вниз, а не наоборот. Даже кратковременная работа с монитором, установленным слишком высоко, приводит к утомлению шейных отделов позвоночника.

3.Если при правильной установке монитора относительно уровня глаз выясняется, что ноги пользователя не могут свободно покоиться на полу, следует установить подставку для ног, желательно наклонную. Если ноги не имеют надежной опоры, это непременно ведет к нарушению осанки и утомлению позвоночника. Удобно, когда компьютерная мебель (стол и рабочее кресло) имеют средства для регулировки по высоте. В этом случае проще добиться оптимального положения.

4.Клавиатура должна быть расположена на такой высоте, чтобы пальцы рук располагались на ней свободно, без напряжения, а угол между плечом и предплечьем составлял 100° — 110°. При использовании обычных школьно-письменных столов добиться одновременно правильного " положения и монитора, и клавиатуры практически невозможно. Для работы рекомендуется использовать специальные компьютерные столы, имеющие выдвижные полочки для клавиатуры. Если такой полочки нет и клавиатура располагается на том же столе, что и монитор, использование подставки для ног становится практически неизбежным, особенно когда с компьютером работают дети.

5.При длительной работе с клавиатурой возможно утомление сухожилий кистевого сустава. Известно тяжелое профессиональное заболевание — кистевой туннельный синдром, связанное с неправильным положением рук на клавиатуре. Во избежание чрезмерных нагрузок на кисть желательно предоставить рабочее кресло с подлокотниками, уровень высоты которых, замеренный от пола, совпадает с уровнем высоты расположения клавиатуры.

6.При работе с мышью рука не должна находиться на весу. Локоть руки или хотя бы запястье должны иметь твердую опору. Если предусмотреть необходимое расположение рабочего стола и кресла затруднительно, рекомендуется применить коврик для мыши, имеющий специальный

опорный валик. Нередки случаи, когда в поисках опоры для руки (обычно правой) располагают монитор сбоку от пользователя (соответственно, слева), чтобы он работал вполоборота, опирая локоть или запястье правой руки о стол. Этот прием недопустим. Монитор должен обязательно находиться прямо перед пользователем.

4.2 Расчёт освещения рабочего места оператора

Одним из основных вопросов охраны труда является организация рационального освещения производственных помещений и рабочих мест.

Для освещения помещения, в котором работает оператор, используется смешанное освещение, т.е. сочетание естественного и искусственного освещения.

Естественное освещение – осуществляется через окна в наружных стенах здания.

Искусственное освещение – используется при недостаточном естественном освещении и осуществляется с помощью двух систем: общего и местного освещения. Общим называют освещение, светильники которого освещают всю площадь помещения. Местным называют освещение, предназначенное для определённого рабочего места.

Для помещения, где находится рабочее место оператора, используется система общего освещения.

Нормами для данных работ установлена необходимая освещённость рабочего места E_H=300 лк (для работ высокой точности, когда наименьший размер объекта различения равен 0.3 – 0.5 мм).

Расчёт системы освещения производится методом коэффициента использования светового потока, который выражается отношением светового потока, падающего на расчётную поверхность, к суммарному потоку всех ламп. Его величина зависит от характеристик светильника, размеров помещения, окраски стен и потолка, характеризуемой коэффициентами отражения стен и потолка.

Общий световой поток определяется по формуле:

$$F_{общ} = E_{н} \cdot S \cdot z_1 \cdot z_2 / \eta ,$$

где$E_Н$ – необходимая освещённость рабочего места по норме ($E_Н$=300 лк);

S – площадь помещения, м^2;

z_1 – коэффициент запаса, который учитывает износ и загрязнение светильников (z_1=1.5);

z_2 – коэффициент, учитывающий неравномерность освещения (z_2=1.1, стр. 139 [15]);

η - коэффициент использования светового потока выбирается из таблиц в зависимости от типа светильника, размеров помещения, коэффициентов отражения стен и потолка помещения.

Определим площадь помещения, если его длина составляет $L_д$=6.5 м, а ширина $L_ш$=3.7 м:

$$S = L_{д} \cdot L_{ш} = 6.5 \quad 3.7 = 24 \text{ м}^2$$

Выберем из таблицы коэффициент использования светового потока по следующим данным:

- коэффициент отражения побелённого потолка $R_п$=70%;
- коэффициент отражения от стен, окрашенных в светлую краску $R_{ст}$=50%;

$$i = \frac{L_{д} \cdot L_{ш}}{h_{п} \cdot (L_{д} + L_{ш})} = 0.7,$$

где$h_П$ – высота помещения = 3.5 м. Находим (для люминесцентных ламп i=0.7) η=0.38.

Определяем общий световой поток:

$$F_{общ} = 300 \cdot 24 \cdot 1.5 \cdot 1.1 / 0.38 = 31263 .2 \text{ лм}$$

Наиболее приемлемыми для помещения ВЦ являются люминесцентные лампы ЛБ (белого света) или ЛТБ (тёпло-белого света), мощностью 20, 40 или 80 Вт.

Световой поток одной лампы ЛТБ40 составляет F_1=3100 лм,

следовательно, для получения светового потока $F_{общ}$=31263.2 лм необходимо N ламп, число которых можно определить по формуле

$$N = F_{общ} / F_1$$

Подставим значения, полученные выше:

$$N = 31263 .2 / 3100 = 10 \text{ ламп.}$$

Таким образом, необходимо установить 10 ламп ЛТБ40.

Электрическая мощность всей осветительной системы вычисляется по формуле:

$$P_{общ} = P_1 \cdot N \text{ , Вт,}$$

гдеP_1 – мощность одной лампы = 40 Вт, N – число ламп = 10.

$$P_{общ} = 40 \cdot 10 = 400 \text{ Вт.}$$

Для исключения засветки экранов дисплеев прямыми световыми потоками светильники общего освещения располагают сбоку от рабочего места, параллельно линии зрения оператора и стене с окнами. Такое размещение светильников позволяет производить их последовательное включение в зависимости от величины естественной освещённости и исключает раздражение глаз чередующимися полосами света и тени, возникающее при поперечном расположении светильников.

Расчёт местного светового потока не производится, т.к. в данном случае рекомендуется система общего освещения во избежание отражённой блёсткости от поверхности стола и экрана монитора.

Коэффициент пульсации освещённости:

$$K_{П} = \frac{E_{max} - E_{min}}{2 E_{ср}} \cdot 100 \ \% \text{ ,}$$

гдеE_{max}, E_{min} и $E_{ср}$ показатели освещённости для газоразрядных ламп при питании их переменным током – соответстсвенно максимальная, минимальная и средняя.

Возьмём по аналогии [16], табл. 4 люминесцентную лампу ЛХБ приблизительно той же мощности. Включением смежных ламп в разные фазы

(группы) трёхфазной электрической сети возможно добиться уменьшения коэффициента пульсации $К_П$ с 35 до 3 – т.е. почти в 12 раз (рис. 1). На рис. 1 указаны три выключателя (по одному на каждую фазу – группу ламп) – это необходимо для обеспечения возможности независимого управления группами ламп.

Равномерность распределения яркости в поле зрения. Характеризуется отношением $L_{\text{окр. поля зрения}} / L_{\text{раб. пов-ти}} = 3/5$ (данное отношение считается оптимальным) или $E_{min}/E_{max} \geq 1/3$. В данном случае $E_{min} = E_{max} = 1300$, следовательно отношение $E_{min}/E_{max} = 1$.

Итак, для обеспечения нормальных условий работы программиста, в соответствии с нормативными требованиями, необходимо использовать данное число светильников указанной мощности для освещения рабочего помещения (Рис. 3.1).

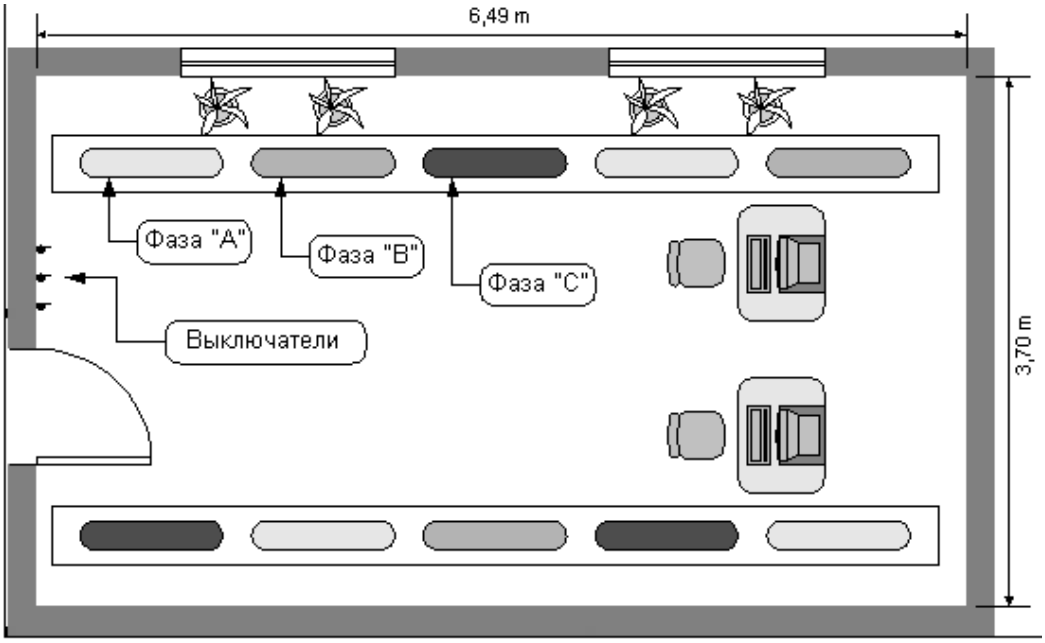

Рис. 3.1. Размещение групп ламп в комнате

4.3 Расчет искусственного освещения

Искусственное освещение в помещениях эксплуатации ВТД и ПЭВМ должно осуществляться системой общего равномерного освещения.

В производственных помещениях в случаях преимущественной работы с

документами допускается применение системы комбинированного освещения.

Освещенность на поверхности стола в зоне размещения рабочего документа должна быть 300 – 500 лк.

Допускается установка светильников местного освещения для подсветки документов.

В качестве источников света при искусственном освещении должны применяться преимущественно люминесцентные лампы типа ЛБ.

Допускается применение лампы накаливания в светильниках местного освещения. Рекомендуется использовать комбинированную систему освещения производственного помещения ($E_{нк}$=500 лк).

Рабочие места с ПЭВМ и ВТД по отношению к световым проемам должны располагаться так, чтобы естественный свет падал сбоку, преимущественно слева. Расстояние между столами в направлении тыла поверхности одного видеомонитора и экрана другого видеомонитора должно быть не менее 2 м, а расстояние между боковыми поверхностями видеомониторов не менее 1,2 м.

Метрологическое обеспечение в области безопасности труда осуществляется на основе положений ГОСТ 12.0.005-84 «Метрологическое обеспечение труда в области безопасности труда. Основные положения»

4.4 Расчёт информационной нагрузки

Расчёт информационной нагрузки оператора необходим для того, чтобы выяснить, будет ли оператор справляться с заданием. Рассчитаем информационную нагрузку оператора. Количество операций, совершаемых оператором за 3 часа (Таблица 3.1):

Таблица 3.1.

Члены алгоритма	**Символ**	**Количество членов**	**Частота повторения** p_i
Афферентные:			1
Наблюдение результатов	F	10	1
Всего:		10	

Эфферентные:			1
Выбор наилучшего варианта из нескольких	C	3	0,04
Исправление ошибок	D	1	0,01
Анализ полученных результатов	M	40	0,54
Выполнение механических действий	K	30	0,41
Всего:		74	
Итого:		84	

Рассчитаем энтропию информации:

$$H_{1F} = -1 \cdot \log_2 1 = 0;$$

$$H_{2C} = -0.04 \cdot \log_2 0.04 = 0.19;$$

$$H_{2D} = -0.01 \cdot \log_2 0.01 = 0.06;$$

$$H_{2M} = -0.54 \cdot \log_2 0.54 = 0.48;$$

$$H_{2K} = -0.41 \cdot \log_2 0.41 = 0.52;$$

$$H_2 = \sum_{i=1}^{4} H_{2i} = 1.25$$

Суммарная энтропия:

$$H = H_{эфф} + H_{афф} + H_{лог} = 1.25 \text{ бит/с.}$$

Поток информационной нагрузки равен $\Phi = \frac{H \cdot N}{t}$, где:

N – суммарное число всех членов алгоритма;

t – длительность выполнения всей работы, мин.

$$\Phi = \frac{1.25 \text{ бит/с}}{180} \cdot 84 = 0.58 \text{ бит/мин} \quad .$$

Вывод: $\Phi_{расч} = 0.58 \frac{бит}{мин} < \Phi_{доп\ max} = 0.8 \frac{бит}{мин}$. Следовательно, информационная нагрузка оператора укладывается в норму.

4.5 Промышленная санитария

Прежде всего, следует отметить тот факт, что предприятие располагает помещением, используемым под офис, где был произведен ремонт и частичная реконструкция, после этого помещение было обследовано представителями

центра стандартизации и метрологии Республики Узбекистан, представителями пожарной охраны и СЭС на соответствие нормативным актам. В офисе находятся рабочие места с использованием ПЭВМ.

Микроклиматические условия соответствуют допустимым нормам и стандартам для офисных помещений. Отклонений от норм в ходе обследования выявлено не было. Основные характеристики микроклиматических условий в офисном помещении сведены в таблице 3.2.

Таблица 3.2. Фактические значения параметров микроклиматических условий в офисе

Период года	Температура воздуха, С°	Относительная влажность воздуха, %	Движение воздуха
Зима	18-24	40-60	Естественное
Весна, осень	18-24	40-60	Естественное
Лето	18-24	40-60	Принудительное

Для обеспечения соответствующего нормам микроклимата в помещениях используются различные средства в зависимости от времени года:

- зимой помещения отапливаются посредством центральной отопительной системы. Обеспечение всех требований по установке и эксплуатации отопительной системы по договору возложено на вышеуказанное предприятие. Использование системы отопления приводит к понижению влажности воздуха;
- летом вместо отопительных систем используются системы охлаждения и принудительной вентиляции. Наиболее эффективными показывают себя так называемые «сплит-системы». В общем случае их использование приводит к повышению влажности воздуха в помещениях.

Вне зависимости от сезонности используется естественная система постоянной вытяжной вентиляции в помещениях предназначенных для санитарно-гигиенических целей, а также местах отведенных для приготовления пищи.

Помещение офиса состоит из нескольких отдельных комнат с естественным боковым освещением (из окон). Общая площадь помещений, используемых для хозяйственной деятельности, составляет 125 м^2. В офисе для искусственного освещения используются люминесцентные лампы. Общая совокупная мощность осветительных приборов верхнего света 1,9 кВт. Дополнительно на каждом рабочем месте установлен светильник мощностью 60 Вт.

Источником шума в офисе является проезжая часть улицы в сторону, которой выходит большая часть окон офиса. Наибольшая интенсивность движения автотранспорта приходится на дневное время. Для снижения шума внутри помещений были использованы шумопоглощающие панели, а также шумо- и теплоизоляционные окна. Кроме того, применение систем охлаждения в летний период избавляет от необходимости открывать окна для проветривания, что также снижает уровень шума в помещении.

Офис оборудован необходимыми средствами санитарно-технического оборудования, которые подключены к соответствующим развязкам городского водопровода и канализации.

Установленные в офисе ПЭВМ имеют в своем составе цветные видеомониторы SamsungLCD17”, которые выполнены в соответствии с международными стандартами радиационной защиты MRP II (Standard) и TCO’92 (Option). Эти мониторы обладают малым уровнем радиации и при этом не устают глаза.

ЗАДАЧА №1

РАСЧЕТ ТРЕБУЕМОГО СНИЖЕНИЯ ШУМА

Рассчитать требуемое снижение шума от технологического оборудования в швейном цехе.

Запроектировать необходимые меры по снижению шума и установить эффективность данных средств.

Указание к решению задачи.

1. Суммарный уровень шума в цехе:

$L_ф = L_1 + \lg \cdot n$, дБ

где: L_1 – уровень шума от одного источника, дБ

n – число источников шума.

При среднегеометрической частоте октавной полосы

125 Гц –L_1 = 94 дБ.

при250Гц – L_1 = 93 дБ.

2. Снижение уровня шума определяют :

$L = L_ф – Lg$, дБ

где $L_Ф$- фактический уровень шума, дБ

Lg- допускаемый уровень шума, дБ

присреднегеометрической частоте октавной полосы

125 Гц –$L_Д$ =92 дБ,

при 250 Гц – $L_Д$ = 86дБ.

3. Шумопоглощение в помещениях может быть достигнуто применением пористых штукатурок, воздушных прослоек в стенах и перегородках. Для снижения уровня шума в цехе выбираем звукопоглощающие прокладки для стен: плиты АГП, гипсовые с заполнением из минеральной ваты с коэффициентом звукопоглощения – 0.09; для потолка – минеральные плиты, сдублированные со стекло тканью Э = 0.1 с коэффициентом звукопоглощения 0.188 .

4. Снижение уровня шума в помещении при этом:

$$\Delta L = 10 \lg \frac{\sum \alpha_2 \cdot S_2}{\sum \alpha_1 \cdot S_1} = 10 \lg \frac{0.09\left(h_1 \cdot l \cdot 2 + h_1 \cdot b \cdot 2\right) + 0.188\ (l \cdot b)}{0.03\left(\left(h_1 - h_2\right) l \cdot 2 + \left(h_1 - h_2\right) + 0.0013\ \left(h_2 \cdot l \cdot 2\right)\right)}$$

где: $\sum \alpha_2 S_2$ - суммарное звуковое поглощение в помещении после устройства звукопоглощающей отделки.

$\sum \alpha_1 \cdot S_1$ - суммарное звуковое поглощение в помещении до устройства звукопоглощающей отделки.

$\alpha_1 \cdot \alpha_2$ - коэффициенты звукового поглощения стен, потолка и панелей

($\alpha_1 = 0.03; 0.02; 0.0015$ соответственно):

S_1/S_2- соответствующие площади стен, потолка и панелей.

Варианты для решения задачи №2

	1	2	3	4	5	6	7	8	9	10
n- число источ. шума Размерипоещения	10	20	40	30	20	50	10	30	40	20
h_1-высота стен, м	6	8	6	10	8	7	5	7	8	9
h_2-высоты панелей, м	3	4	3	5	4	4	2	3	4	5
l-длина, м b-ширина, мм	40 12	45 14	50 13	44 15	46 10	42 12	40 11	38 13	42 15	50 10
Исходные данные	Варианты									
	11	12	13	14	15	16	17	18	19	20
n- число источ. шума Размерипоещения	8	9	10	15	4	7	21	18	11	13
h_1-высота стен, м	5	7	8	6	8	4	7	6	5	4
h_2-высоты панелей, м	2.5	3	4	3	4	2	3	2.5	2.5	2
l-длина, м	25	30	27	24	22	19	20	21	23	24
b-ширина, мм	13	14	12	10	11	9	12	11	13	14

Выводы.

Охрана труда — система сохранения жизни и здоровья работников в процессе трудовой деятельности, включающая в себя правовые, социально-экономические, организационно-технические, санитарно-гигиенические, лечебно-профилактические, реабилитационные и иные мероприятия.[1]

Кроме того, охрана труда рассматривается в юридической литературе ещё с нескольких позиций:

1. Как основной принцип трудового права и трудовых правоотношений

2. Как система законодательных актов, а также предупредительных и регламентирующих социально-экономических, организационных, технических, санитарно-гигиенических и лечебно-профилактических мероприятий, технических средств и методов, направленных на обеспечение безопасных условий труда

Кроме обязанностей, каждый работник имеет права и гарантии права на безопасные и здоровые условия труда, которые сформулированы в узбекском законодательстве.

Гарантии права работника на труд в условиях, соответствующих требованиям ОТ, состоят, в частности, в том, что:

- Государство гарантирует работникам защиту их права на труд в условиях, соответствующих требованиям ОТ;
- Условия труда по трудовому договору должны соответствовать требованиям ОТ;
- На время приостановления работ вследствие нарушения требований ОТ не по вине работника за ним сохраняется место работы и средний заработок;
- При отказе работника от выполнения работ при возникновении опасности для его жизни и здоровья, работодатель обязан предоставить работнику другую работу на время устранения такой опасности. Если предоставление другой работы невозможно, время простоя оплачивается в соответствии с действующим законодательством;

- В случае не обеспечения работника средствами защиты по нормам работодатель не в праве требовать от работника выполнения трудовых обязанностей и обязан оплатить простой;
- Отказ работника от выполнения работ из-за опасности для его жизни и здоровья, либо от тяжёлых работ и работ с вредными или опасными условиями труда, не предусмотренных трудовым договором, не влечёт за собой привлечение его к дисциплинарной ответственности;
- В случае причинения вреда жизни и здоровью работника при исполнении трудовых обязанностей осуществляется возмещение указанного вреда в соответствии с действующим законодательством;

ЗАКЛЮЧЕНИЕ

В ходе выполнения данной работы была создана система управления содержимым сайта с использованием веб-технологий, языка программирования PHP. В качестве базы данных использовались две БД, реляционная – MariaDB и не реляционная MongoDB.

Использование данной системы позволяет:

1. Быстро создавать и редактировать содержимое многих сайтов одновременно
2. Очень легко и просто добавлять новые страницы на сайты
3. Редактировать текстовое содержимое не имея никаких знаний в области веб-программирования
4. Собирать статистику по различным данным и параметрам сайтов
5. В несколько кликов создать несколько сайтов и заполнить их шаблонным содержимым
6. Быдь всегда в курсе о наличии/отсутствии проблем на сервере благодаря системной рассылке прямо на электронную почту
7. Создавать очередь задач внутри системы для потокового выполнения

В соответствии с концепцией повторного использования кода, являющейся одной из основ Open Source идеологии, логика системы опирается , на ранее существовавший - хорошо отлаженный и проверенный временем подход и принципы построения систем управления данными.

Благодаря проведению рефакторинга существующей системы удалось достичь значительного повышения производительности и ускорению работы системы в целом. Получилось изменить существующую нагрузку на сервер:

- Нагрузка на центральный процессор сервера – уменьшилась на 18%
- Нагрузка на оперативную память – уменшилась на 24%
- Нагрузка на жесткий диск (запись/считывание) – увеличилась на 4%

В связи с тем, что раннее жесткий диск был подвержен минимальным нагрузкам мы смогли уменьшить нагрузку на ЦП и ОЗУ за счет добавления

дополнительных операций по работе с файлами на жестком диске, при этом поменяв подход в работе системы нам удалось добиться существенного снижения нагрузки за счет небольшого увеличения нагрузки на жесткий диск, что в нашем случае является допустимым.

Благодаря использованию планировщика задач Cron, удалось достичь равномерной нагрузки на сервер в течении всех 24 часов в сутки, напомню, что раннее сервер подвергался пиковой нагрузке в дневное время и простаивал ночью. Удалось достичь равномерного распределения нагрузки в течении всего дня, в связи с чем занчительно уменьшился износ оборудование, а скорость выполнения задач не пострадала.

На данном этапе развития функционал системы логически завершён. В случае же продолжения разработки в сторону расширения функционала следует добавить возможность большего переопределения исходных настроек администратором системы распределения информации.

СПИСОК ЛИТЕРАТУРЫ

1. И.А.Каримов. **Узбекистан на пороге XXI века**. "Узбекистон",1997
2. И. А. Каримов. **Наша цель: свободная и процветающая родина.** "Узбекистон", 1994
3. И. А. Каримов. Свое **будущее мы строим своими руками,** "Узбекистон", 1999

4. Дональд Эрвин Кнут. **Искусство программирования, том 1: основные алгоритмы**, третье издание. 1997
5. Беседин К.Ю., Костенецкий П.С. **Моделирование обработки запросов на гибридных вычислительных системах с многоядерными сопроцессорами и графическими ускорителями.** Программные системы: теория и приложения. 2014. Т. 5, № 1(19). С. 91-110
6. Торрес Р. Дж. **Практическое руководство по проектированию и разработке пользовательского интерфейса.** СПб.: Вильямс, 2002. - 400 с.
7. Кушнарев Л.И., Хицков Е.А., Гальчич М.А. **Методические рекомендации по дипломному проектированию.** - М.: ФГОУ ВПО МГАУ., 2005. - 114 с.
8. Пауэлл Т.А. **Полное руководство по HTML**. Мн.: Попурри, - 2001. - 912с.
9. Крамер Э. **HTML**. СПб: «Диалектика», 2001. - 426с.
10. Нидерст Дж. **Web - мастеринг для профессионалов. Настольный справочник**. СПб: Изд-во Питер, 2001. - 240с.
11. Абдуллаев С.М., Ленская О.Ю., Гаязова А.О., Иванова О.Н., Носков А.А., Соболев Д.Н., Радченко Г.И. **Алгоритмы краткосрочного прогноза с использованием радиолокационных данных: оценка трасляции и композиционный дисплей жизненного цикла** // Вестник ЮУрГУ. Серия "Вычислительная математика и информатика". 2014. Т. 3. № 1. С. 17-32
12. Стив МакКоннел **Code Complete («Совершенный код»)**, второе издание.

ПРИЛОЖЕНИЕ

```
(function($) {
$(document).ready(function() {
var stuff2 = {};

stuff2.siteid = parseInt($('.stuff2-controller').attr('data-siteid'));
stuff2.verid = parseInt($('.stuff2-controller').attr('data-verid'));

stuff2.container = $('<div>').css({
'position'     : 'fixed',
'left'         : 0,
'top'          : '50%',
'width'        : '250',
'min-height'   : '250',
'color'        : '#222',
'background'   : '#ffffff',
'border'       : '1px solid #222',
'margin-top'   : '-210px',
'padding'      : '10px 20px 10px 0',
'z-index'      : '1000'
});
stuff2.roller = $('<div>').css({
"position"     : "absolute",
"right"        : "0",
"top"          : "0",
"bottom"       : "0",
"width"        : "20px",
"background"   : "#222"
}).click(function() {
```

```
if (stuff2.container.is(':animated')) return false;
if (parseInt(stuff2.container.css('margin-left')) == 0) {
stuff2.container.animate({
'margin-left': -250
}, 500);
} else {
stuff2.container.animate({
'margin-left': 0
}, 500);
}
});

stuff2.iframe = $('<iframe marginheight="0" frameborder="no" scrolling="no"
src="http://stuff2.ru/sites2/moderation/moderation.window.php?site_id=' +
stuff2.siteid + '">').css({
'width'  : '100%',
'height' : '420px',
'border' : 'none'
});

stuff2.container.appendTo('body');
stuff2.roller.appendTo(stuff2.container)
stuff2.iframe.appendTo(stuff2.container);

});
})(jQuery)
```

```
<?php
require_once(__DIR__.'/../stuff2/bootstrap.php');

$q          = "SELECT design_id FROM site_slots WHERE deleted = 0 AND
disabled = 1 LIMIT 20";
$design_ids = \MF\DB\MySQL::fetchCol($q);

if (empty($design_ids)) {
exit;
}

$q          = "SELECT * FROM site_slots WHERE design_id IN (".implode(',',
$design_ids).")";
$slots      = \MF\DB\MySQL::fetchHashAll($q, 'design_id');

$data       =
@file_get_contents('http://srv.megagroup.ru/access/donkey_take_design.php?i
ds='.implode(',', $design_ids));
$data       = json_decode($data, true);


foreach ($data as $v) {
$slot      = $slots[$v['design_id']];
$root_path = __DIR__.'/../public/';
$dir       = '/upload/slots/'.$slot['slot_id'];
$filename  = $v['design_id'].'.jpg';
$filepath  = $root_path.'/'.$dir.'/'.$filename;
\MF\FS::addRoot($root_path);
\MF\FS::makeDirRecursive($root_path.'/'.$dir, 0755);
\MF\FS::put($filepath, @file_get_contents($v['url']));
```

```
$res =
@file_get_contents('http://srv.megagroup.ru/access/donkey_take_design.php?r
eserve_id='.$v['design_id']);
if ($res == 'reserved') {
\MF\DB\MySQL::update('site_slots', [
'filename_orig' => $dir.'/'.$filename,
'disabled'      => 0,
], "slot_id = ".$slot['slot_id']);

\STUFF2\Slot::moderate($slot['slot_id'], 0, 3, 'designer', 'Включение слота');
\STUFF2\Slot::fillSlot($slot['slot_id']);

} else {
\MF\DB\MySQL::update('site_slots', [
'deleted' => 1,
], "slot_id = ".$slot['slot_id']);
}
}
```

==========

```
<?php

require_once(__DIR__.'/../stuff2/bootstrap.php');


$q = "SELECT count(s.site_id) FROM sites s LEFT JOIN stuff_sites ss ON
s.site_id = ss.site_id WHERE s.filled = 1 AND s.designer_id = 0 AND
s.designed = 0 AND ss.is_template = 0";
$sites_for_dist = Db::fetchOne($q);
```

```
$q = "SELECT ag.group_id, ag.agent_id, (SELECT COUNT(*) FROM
agents_groups_links WHERE group_id = ag.group_id) AS k FROM
agents_groups ag WHERE ag.base = 1";
$groups = Db::fetchAll($q);

$koef  = 0;
foreach ($groups as $group) {
$koef = $koef + $group['k'];
}

$sites_per_k = floor((int) $sites_for_dist / (int) $koef);

foreach ($groups as $key => $group) {
$limit = (int) $group['k'] * (int) $sites_per_k;
$q = "SELECT s.site_id FROM sites s LEFT JOIN stuff_sites ss ON s.site_id
= ss.site_id WHERE s.filled = 1 AND s.designer_id = 0 AND s.designed = 0
AND ss.is_template = 0 LIMIT $limit";
$sites = (array) Db::fetchCol($q);
if (count($sites)) {
$site_ids = implode(',', $sites);
$q = "UPDATE sites SET designer_id = ".$group['agent_id']." WHERE site_id
IN ($site_ids)";
Db::query($q);
}
}


<?php
```

```
namespace STUFF2;

class Slot {

public static function fillSlot($slot_id) {

$slot_id = (int) $slot_id;

if (!$slot_id) {
return false;
}

$query = "SELECT *, (usage_limit - sites_count) AS need_sites FROM
site_slots WHERE slot_id = $slot_id LIMIT 1";
$slot  = \MF\DB\MySQL::fetchRow($query);

if (empty($slot)) {
return false;
}

$sites    = \STUFF2\Site::getFreeSites($slot['industry_id'], $slot['need_sites']);
$site_ids = implode(',', array_keys($sites));
$count    = count($sites);

if (!$count) {
return false;
}

if (!$slot['coder_site_id']) {
```

```
if (!count($sites)) {
$query        = "SELECT site_id FROM stuff_sites WHERE slot_id =
".$slot['slot_id'];
$coder_site_id = \MF\DB\MySQL::fetchOne($query);

} else {

$keys = array_keys($sites);
$coder_site_id =  $keys[0];

}

\MF\DB\MySQL::update('site_slots', [
'coder_site_id' => $coder_site_id,
], "slot_id = ".$slot['slot_id']);
/*\MF\DB\MySQL::update('stuff_sites', [
'content_filled' => 1,
], "site_id = ".$coder_site_id);*/

$slot['coder_site_id'] = $coder_site_id;


$query = "SELECT content_template_id, ver_id FROM stuff_sites ss JOIN
version USING(site_id) WHERE ss.site_id = ".$slot['coder_site_id'];
$info  = \MF\DB\MySQL::fetchRow($query);

$res = \STUFF2\Site::copyVersionCron($info['content_template_id'],
$info['ver_id'], [
'callback' => '\STUFF2\Site::setContentFilled',
'params'   => [
```

```
$coder_site_id,
]
]);

}

\MF\DB\MySQL::update('stuff_sites', [
'slot_id' => $slot['slot_id'],
], "site_id IN ($site_ids)");
\MF\DB\MySQL::update('sites', [
'designer_id' => $slot['designer_id'],
], "site_id IN ($site_ids)");
\MF\DB\MySQL::update('site_slots', [
'sites_count' => \MF\DB\MySQL::expr('sites_count + '.$count),
], 'slot_id = '.$slot['slot_id']);

}

public static function moderate($slot_id, $agent_id, $moderated = 3, $type =
'designer', $msg = '') {
$slot_id   = (int) $slot_id;
$agent_id  = (int) $agent_id;
$moderated = (int) $moderated;
$type      = \MF\DB\MySQL::escape($type);
$msg       = \MF\DB\MySQL::escape($msg);

if (!$slot_id) {
return false;
}
```

```
switch ($type) {
case "designer":
$key_moderate      = 'moderate_designer';
$key_date_moderate = 'date_moderate_designer';
$key_error         = 'error_designer';
break;
case "coder":
$key_moderate      = 'moderate_coder';
$key_date_moderate = 'date_moderate_coder';
$key_error         = 'error_coder';
break;
case "oper":
$key_moderate      = 'moderate_oper';
$key_date_moderate = 'date_moderate_oper';
$key_error         = 'error_oper';
break;
default:
return false;
}

$data = [
$key_moderate      => $moderated,
$key_date_moderate => \MF\DB\MySQL::expr('NOW()'),
];

if ($moderated == 2) {
$data[$key_error] = $msg;
} else if ($moderated == 1) {
$data[$key_error] = '';
```

```
if ($type == 'coder') {
$data['moderate_oper'] = 3;
}
}

$query        = "SELECT $key_moderate FROM site_slots WHERE slot_id =
$slot_id";
$old_moderated = \MF\DB\MySQL::fetchOne($query);

\MF\DB\MySQL::update('site_slots', $data, "slot_id = $slot_id");

\MF\DB\MySQL::insert('site_slots_history',[
'slot_id'   => $slot_id,
'agent_id'  => $agent_id,
'message'   => $msg,
'created'   => \MF\DB\MySQL::expr('NOW()'),
'type'      => $type,
'new_value' => $moderated,
'old_value' => $old_moderated,
]);
}

public static function copySitesCron($slot_id, $json = '') {

$slot_id = (int) $slot_id;

if (!$slot_id) {
return false;
}
```

```
$query = "SELECT *, (usage_limit - sites_count) AS need_sites FROM
site_slots WHERE slot_id = $slot_id LIMIT 1";
$slot  = \MF\DB\MySQL::fetchRow($query);

if (empty($slot)) {
return false;
}

$query       = "SELECT ver_id FROM version WHERE site_id =
".$slot['coder_site_id'];
$from_ver_id = \MF\DB\MySQL::fetchOne($query);

if (!$from_ver_id) continue;

$query    = "SELECT site_id FROM stuff_sites WHERE content_filled = 0
AND site_id != ".$slot['coder_site_id']." AND slot_id = ".$slot['slot_id'];
$site_ids = \MF\DB\MySQL::fetchCol($query);
if (!count($site_ids)) {
continue;
}
$site_ids = implode(',', $site_ids);

$query = "SELECT ver_id, site_id FROM version WHERE site_id IN
($site_ids)";
$sites = \MF\DB\MySQL::fetchAll($query);

foreach ($sites as $site) {
$res = \STUFF2\Site::copyVersionCron($from_ver_id, $site['ver_id'], [
'callback' => '\STUFF2\Site::setContentFilled',
'params'   => [
```

```
$site['site_id'],
]
]);
}
//return \STUFF2\Cron::addCronTask('copy.slot.sites', $slot_id, 0, $json);
}

public static function copySites($slot_id) {
$slot_id = (int) $slot_id;

if (!$slot_id) {
return false;
}

$query = "SELECT * FROM site_slots WHERE moderate_coder = 1 AND
moderate_oper = 1 AND moderate_designer = 1 AND slot_id = $slot_id";
$slot = \MF\DB\MySQL::fetchRow($query);

if (empty($slot)) {
return false;
}

$query = "SELECT v.ver_id FROM version v JOIN sites s USING(site_id)
WHERE v.site_id = ".$slot['coder_site_id']." AND s.server_id = 8";
$from_ver_id  = \MF\DB\MySQL::fetchOne($query);

if (!$from_ver_id) {
return false;
}
```

```
\STUFF2\Site::replaceNames($from_ver_id);

$query    = "SELECT site_id FROM stuff_sites ss JOIN sites s USING(site_id)
WHERE ss.content_filled = 0 AND s.server_id = 8 AND ss.site_id !=
".$slot['coder_site_id']." AND ss.slot_id = ".$slot['slot_id'];
$site_ids = \MF\DB\MySQL::fetchCol($query);
if (!count($site_ids)) {
return false;
}
$site_ids = implode(',', $site_ids);

$query = "SELECT ver_id, site_id FROM version WHERE site_id IN
($site_ids)";
$sites = \MF\DB\MySQL::fetchAll($query);

foreach ($sites as $site) {
\STUFF2\Site::copyVersion($from_ver_id, $site['ver_id']);
\STUFF2\Site::setContentFilled($site['site_id']);
}

return true;
}

public static function delete($slot_id) {
$slot_id = (int) $slot_id;

if (!$slot_id) {
die('error');
}
```

```
\MF\FS::addRoot(__STUFF2__.'/public/upload/slots/');
\MF\FS::cleanDir(__STUFF2__."/public/upload/slots/$slot_id/", true, true);

$slot = \MF\DB\MySQL::fetchRow("SELECT * FROM site_slots WHERE
slot_id = $slot_id");
$res =
@file_get_contents('http://srv.megagroup.ru/access/donkey_take_design.php?f
ree_id='.$slot['design_id']);
if ($res != 'free') {
return false;
}

$query = "DELETE FROM site_slots WHERE slot_id = $slot_id";
\MF\DB\MySQL::query($query);

$query    = "SELECT site_id FROM stuff_sites WHERE slot_id = $slot_id";
$site_ids = \MF\DB\MySQL::fetchCol($query);
\STUFF2\Site::freeSites($site_ids);

}
}
```