/Техническая страница, потом будет удалена/

Текущие задачи по этапам:

1. Исправить мелкие косяки (Дима) +
2. Убрать обрезанности изображения (Дима)
3. Исправить скриншоты на новые с русским названием движка
4. Добавить ссылки на литературу в тексте
5. Добавить ссылки на рисунки
6. Убрать большие пробелы на страницах
7. Уменьшить оглавление до одной страницы
8. Разделить общую записку на две

Оглавление

[Введение 4](#_Toc325408824)

[Постановка задачи 6](#_Toc325408825)

[1. Исследовательская часть 7](#_Toc325408826)

[1.1. Анализ аналогичных программных продуктов 7](#_Toc325408827)

[1.2. Анализ и выбор современных используемых в веб технологий 12](#_Toc325408828)

[1.2.1. Серверные языки программирования 12](#_Toc325408829)

[1.2.2. Клиентские языки программирования 14](#_Toc325408830)

[1.2.3. Способы хранения информации 17](#_Toc325408831)

[1.3. Анализ и выбор программных средств разработки 19](#_Toc325408832)

[1.3.1. Веб-серверы 19](#_Toc325408833)

[1.3.2. Среды разработки 20](#_Toc325408834)

[1.3.3. Средства отладки программного кода 23](#_Toc325408835)

[1.3.4. Система контроля версий 28](#_Toc325408836)

[1.3.5. Вспомогательные средства 35](#_Toc325408837)

[2. Конструкторская часть 36](#_Toc325408838)

[2.1. Анализ и проектирование архитектуры приложения 36](#_Toc325408839)

[2.2. Проектирование и разработка базы данных 44](#_Toc325408840)

[2.2.1. Логическое проектирование 44](#_Toc325408841)

[2.2.2. Физическое проектирование 45](#_Toc325408842)

[2.3. Проектирование программных модулей 50](#_Toc325408843)

[2.4. Проектирование внешнего вида приложения 58](#_Toc325408844)

[2.5. Проектирование безопасности приложения 63](#_Toc325408845)

[3. Технологическая часть 75](#_Toc325408846)

[3.1. Установка и настройка приложения 75](#_Toc325408847)

[3.2. Инструкция пользователя 80](#_Toc325408848)

[3.3. Инструкция администратора 83](#_Toc325408849)

[Заключение 85](#_Toc325408850)

[Список литературы 86](#_Toc325408851)

# Введение

Веб-форум — класс веб-приложений для организации общения посетителей веб-сайта. Термин соответствует смыслу исходного понятия «форум».

Форум предлагает набор разделов для обсуждения. Работа форума заключается в создании пользователями тем в разделах и последующим обсуждением внутри этих тем. Отдельно взятая тема, по сути, представляет собой тематическую гостевую книгу.

Распространённая иерархия веб-форума: Разделы → темы → сообщения.

Обычно сообщения несут информацию «автор — тема — содержание — дата/время». Сообщение и все ответы на него образует «ветку» или «тему».

Отклонение от начальной темы обсуждения часто запрещено правилами форума. За соблюдением правил следят модераторы и администраторы — участники, наделённые возможностью редактировать, перемещать и удалять чужие сообщения в определённом разделе или теме, а также контролировать к ним доступ отдельных зарегистрированных участников, иногда и незарегистрированных,- гостей.

На форумах может применяться чрезвычайно гибкое разграничение доступа к сообщениям. Так, на одних форумах чтение и создание новых сообщений доступны любым случайным посетителям (гостям), на других необходима предварительная регистрация (наиболее распространённый вариант) — те и другие форумы называют открытыми. Применяется и смешанный вариант — когда отдельные темы могут быть доступны на запись всем посетителям, а другие — только зарегистрированным участникам. Кроме открытых, существуют закрытые форумы, доступ к которым определяется персонально для каждого участника администраторами форума. На практике также нередко встречается вариант, когда некоторые разделы форума общедоступны, а остальная часть доступна только узкому кругу участников.

При регистрации участники форума могут создавать профили — страницы со сведениями о данном участнике. В своём профиле участник форума может сообщить информацию о себе, настроить свой аватар (фотографию или картинку для идентификации).

Форум отличается от чата разделением обсуждаемых тем и возможностью общения не в реальном времени. Это располагает к более серьёзным обсуждениям, поскольку предоставляет отвечающим больше времени на обдумывание ответа. Форумы часто используются для разного рода консультаций, в работе служб технической поддержки.

В настоящее время веб-форумы почти полностью вытеснили новостные группы на базе NNTP и являются одним из наиболее популярных способов обсуждения вопросов во Всемирной паутине. На данный момент форумы сосуществуют наравне с блогами. Эти две формы общения в Интернете практически не уступают друг другу по популярности.

Для работы веб-форумов используются различные программные продукты, часто специализированные для конкретных видов форумов. В среде разработчиков для такого программного обеспечения закрепился жаргонный термин «движок форума».

# Постановка задачи

В данной работе необходимо спроектировать и разработать !!!тут вставить!!! с поддержкой идеологии форумного общения. Разрабатываемое веб-приложение должно обладать следующими свойствами:

- поддержка русскоязычных пользователей;

- отсутствие перегруженности пользовательского интерфейса;

- быстрота получения полезного контента (в точности – информации конечным пользователем);

- поддержка всех основных функций форума: создание, редактирования, удаление тем и сообщений;

- идентификация пользователя в системе: регистрация, авторизация;

- ограничение прав пользователей на получение контента (информации);

- возможность контроля за контентом (функции администратора);

- высокая скорость работы;

- возможность расширяемости за счет модулей и визуальных стилевых тем.

# 1. Исследовательская часть

## 1.1. Анализ аналогичных программных продуктов

Плагин для WordPress – Simpleforum

WordPress — система управления содержимым сайта (CMS) с открытым исходным кодом, распространяемая под GNU GPL. Написана на PHP, в качестве базы данных использует MySQL. Сфера применения — от блогов до достаточно сложных новостных ресурсов и даже интернет-магазинов. Встроенная система «тем» и «плагинов» вместе с удачной архитектурой позволяет конструировать практически любые проекты.

В качестве рассматриваемого форума выступает плагин Simpleforum.



Рис. 1. Список категорий Simpleforum

Минус данной CMS – необходима установленная система Wordpress. В ее отсутствие установить форум невозможно.

Достоинством является неперегруженный внешний вид и понятный интерфейс.

Simple Machine Forum (SMF)

Simple Machines Forum (SMF) — бесплатный интернет-форум, написанный на PHP с использованием базы данных MySQL.

У SMF есть базовый репозиторий для бесплатного хранения и отслеживания модификаций через сайт Simple Machines. Множество модификаций, также известных как «моды», создаются и распространяются на бесплатной основе, включая видеоигры, дополнения для профилей, одобрение сообщений модератором, гороскоп, и многие другие, находящиеся в разработке.

Менеджер пакетов, включённый в SMF — одно из его главных достоинств. Он позволяет администратору устанавливать модификации и обновления для SMF без необходимости изменять исходный код скриптов, обычно при помощи всего нескольких кликов мышью.



Рис. 2. Главный вид SMF



Рис. 3. Список тем



Рис. 4. Список групп пользователей в панели администратора

Достоинства данного программного продукта:

* Богатый функционал
* Множество настроек в панели администратора

Недостатки:

* Перегруженность лишней информацией
* Отсутствие русского языка

phpBB

phpBB (PHP Bulletin Board) — популярный бесплатный веб-форум со свободным исходным кодом, разработанный на скриптовом языке PHP, поддерживающий различные СУБД, включая MySQL, PostgreSQL, MS SQL Server, MS Access, SQLite, а также Oracle (при наличии необходимых изменений).

Кроме поддержки различных СУБД достоинствами phpBB являются:

* Несложная в использовании система шаблонов.
* Многоязычный интерфейс: языковые файлы переведены на более чем 50 языков и доступны для свободного скачивания с официального сайта phpBB.
* Большое количество доступных и обновляемых модификаций.

Рис. 5. Список тем форума phBB

Недостатки: изобилие лишней анимации и очень яркие цвета, низкая производительность.

Выводы

На основе рассмотренных cms были выделены следующие положительные качества, которые учитывались при разработке:

- веб-форум должен обеспечивать весь необходимый присущий для данного типа веб-приложения функционал: иерархия Разделы → темы → сообщения, создание, редактирование и удаление каждого элемента иерархии;

- обеспечение безопасности всех выполняемых пользователем функций;

- удобный функционал, позволяющий осуществлять контроль над содержимым сайта и доступом к нему;

- большие возможности наращивания функциональности за счет применения модульной системы.

Также были выделены следующие минусы рассмотренных систем:

- перегруженность интерфейса деталями, слабо связанными или не связанными с основной тематикой сайта. Как следствие, полезная информация (контент) «теряется» и найти нужную информацию становится затруднительно;

- отсутствие русского языка в веб-приложении играет значимую роль, особенно в русскоговорящем сегменте интернета;

- медлительность операций как следствие низкой производительности готовых решений также может оттолкнуть конечного пользователя;

## 1.2. Анализ и выбор современных используемых в веб технологий

### 1.2.1. Серверные языки программирования

Когда пользователь дает запрос на какую-либо страницу (переходит на неё по ссылке или вводит адрес в адресной строке своего браузера), то вызванная страница сначала обрабатывается на сервере, то есть выполняются все программы, связанные со страницей, и только потом возвращается к посетителю по сети в виде файла. Этот файл может иметь расширения: HTML, PHP, ASP, Perl, SSI, XML, DHTML, XHTML.

Работа программ уже полностью зависима от сервера, на котором расположен сайт, и от того, какая версия того или иного языка поддерживается. Список серверных языков программирования: PHP, Perl, Python, Ruby, любой .NET язык программирования (технология ASP.NET), Java, Groovy. Рассмотрим некоторые из них.

PHP

PHP — скриптовый язык программирования общего назначения, интенсивно применяемый для разработки веб-приложений. В настоящее время поддерживается большинством хостинг-провайдеров и является одним из лидеров среди языков программирования, применяющихся для создания динамических веб-сайтов.

Синтаксис PHP подобен синтаксису языка Си. Некоторые элементы, такие как ассоциативные массивы и цикл foreach, заимствованы из языка Perl. Также PHP поддерживает методологию объектно-ориентированного программирования, позволяя описывать классы, их поля и методы.

Исходные коды хранятся на хостинге в файлах с расширением .php и выполняются специальным сервером-интерпретатором, выводя клиенту готовую веб-страницу.

Python

Python — высокоуровневый язык программирования общего назначения с акцентом на производительность разработчика и читаемость кода.

Это активно развивающийся язык программирования, новые версии (с добавлением/изменением языковых свойств) выходят примерно раз в два с половиной года. Вследствие этого и некоторых других причин на Python отсутствуют ANSI, ISO или другие официальные стандарты.

ASP.NET

ASP.NET — технология создания веб-приложений и веб-сервисов от компании Майкрософт. Она является составной частью платформы Microsoft .NET и развитием более старой технологии Microsoft ASP.

Исходные коды могут писаться на любом языке, поддерживаемым .NET технологией. Хранятся они в файлах с расширением .aspx и выполняются сервером на базе Windows, с использованием IIS (Internet Information Services).

Выводы

Любой из рассмотренных серверных языков требует сервера-обработчика на стороне хостинга, хранящего веб-приложения. Язык PHP был выбран по следующим причинам:

- большое количество документации, в т.ч. и на русском языке;

- большое количество уже существующих методик разработки на этом языке;

- самая низкая стоимость сервера с поддержкой этого языка;

- большое количество стабильного инструментария для разработки и отладки приложения;

- отсутствие привязки к конкретной платформе или операционной системе.

### 1.2.2. Клиентские языки программирования

Клинтские языки обрабатываются на стороне пользователя, как правило их выполняет браузер. Это и создает главную проблему клиентских языков — результат выполнения программы (скрипта) зависит от браузера пользователя. То есть если пользователь запретил выполнять клиентские программы, то они исполняться не будут, как бы ни желал этого программист. Кроме того, может произойти такое, что в разных браузерах или в разных версиях одного и того же браузера один и тот же скрипт будет выполняться по-разному. С другой стороны, программист может упростить их работу и снизить нагрузку на сервер за счет программ, исполняемых на стороне клиента, поскольку они не всегда требуют перезагрузку (генерацию) страницы. Самыми распространенными клиентскими языками программирования являются: JavaScript, VBScript, ActionScript, Java, Adobe Flash, Microsoft Silverlight. Рассмотрим некоторые из них.

JavaScript

JavaScript — прототипно-ориентированный скриптовый язык программирования.

JavaScript обычно используется как встраиваемый язык для программного доступа к объектам приложений. Наиболее широкое применение находит в браузерах как язык сценариев для придания интерактивности веб-страницам.

Основные архитектурные черты: динамическая типизация, слабая типизация, автоматическое управление памятью, прототипное программирование, функции как объекты первого класса.

На JavaScript оказали влияние многие языки, при разработке была цель сделать язык похожим на Java, но при этом лёгким для использования непрограммистами. Языком JavaScript не владеет какая-либо компания или организация, что отличает его от ряда языков программирования, используемых в веб-разработке.

ActionScript

ActionScript — объектно-ориентированный язык программирования, который добавляет интерактивность, обработку данных и многое другое в содержимое Flash-приложений. ActionScript исполняется виртуальной машиной, которая является составной частью Flash Player. ActionScript компилируется в байткод, который включается в SWF-файл.

SWF-файлы исполняются Flash Player-ом. Flash Player существует в виде плагина к веб-браузеру, а также как самостоятельное исполняемое приложение. Во втором случае возможно создание исполняемых exe-файлов, когда Flash Player включается в swf-файл.

Microsoft Silverlight

Microsoft Silverlight — это программная платформа, включающая в себя плагин для браузера, который позволяет запускать приложения, содержащие анимацию, векторную графику и аудио-видео ролики, что характерно для RIA (Rich Internet application). Версия 2.0 добавила поддержку для языков .NET и интеграцию с IDE.

HTML

HTML — стандартный язык разметки документов во Всемирной паутине. Большинство веб-страниц создаются при помощи языка HTML (или XHTML). Язык HTML интерпретируется браузерами и отображается в виде документа, в удобной для человека форме.

CSS

CSS — формальный язык описания внешнего вида документа, написанного с использованием языка разметки.

Преимущественно используется как средство описания, оформления внешнего вида веб-страниц, написанных с помощью языков разметки HTML и XHTML, но может также применяться к любым XML-документам, например, к SVG или XUL.

CSS используется создателями веб-страниц для задания цветов, шрифтов, расположения отдельных блоков и других аспектов представления внешнего вида этих веб-страниц. Основной целью разработки CSS являлось разделение описания логической структуры веб-страницы (которое производится с помощью HTML или других языков разметки) от описания внешнего вида этой веб-страницы (которое теперь производится с помощью формального языка CSS). Такое разделение может увеличить доступность документа, предоставить большую гибкость и возможность управления его представлением, а также уменьшить сложность и повторяемость в структурном содержимом.

Выводы

В данном проекте проекте в качестве языка реализации клиентской части был выбран язык JavaScript. При этом учитывалось следующее:

- самая высокая распространённость у языка JavaScript, он поддерживается всеми современными браузерами;

- большое количество документации по этому языку;

- приемлемая производительность при выполнении сценариев;

- в отличие от ActionScript и Silverlight для его работы не нужны отдельные плагины и он работает из под любой операционной системы;

- простота отладки и высокая скорость разработки.

Также используется связка HTML+CSS, ставшая стандартом в веб-программировании.

### 1.2.3. Способы хранения информации

Современные системы управления сайтами используют 2 основных подхода для хранения информации:

- файловый метод

- метод с использованием баз данных.

Различия данных подходов состоят в том, что в первом случае информация о пользователях, об их действиях и контент располагается в отдельных текстовых, либо бинарных файлах, откуда потом считывается во время запросов программным путем на стороне самого приложения. В случае баз данных, информация содержится в хранилище, жестко не связанном с организацией файловой структуры, информация извлекается специальными запросами (как правило на SQL-языке запросов) на стороне системы управления базами данных (СУБД).

Преимущества файлового подхода:

- скорость чтения информации при маленьких объемах во многих случаях будет выше;

- информация, методы ее считывания и хранения не зависят от используемых СУБД, а напрямую зависит от разработчика веб-приложения;

- как следствие неиспользования СУБД уменьшается конечная стоимость разработки и размещения на сторонних хостингах;

- возможность считывать информацию вручную, без веб-приложения, в случае использования текстового формата хранения данных.

Недостатки данного подхода:

- низкая скорость разработки;

- низкая отказоустойчивость;

- низкая производительность при больших наборах данных;

- вероятность увеличения количества файлов и потери возможности организации работы с ними.

Достоинства использования баз данных:

- отказоустойчивость. Низкая вероятность того, что при отказе веб-приложения информация будет испорчена;

- удобство использования. Современные СУБД имеют графический веб-интерфейс, визуальный редактор записей и постоитель диаграмм, а также возможности экспорта и импорта;

- высокая скорость разработки. При использовании языка запроса SQL, возможно получать необходимую информацию из нескольких таблиц в удобном для веб-приложения формате, с использованием фильтров и сортировок.

Недостатки данного подхода:

- необходимость использовать СУБД на стороне сервера, из-за чего повышаются возможные расходы на содержание приложения.

Для данной работы был выбран способ хранения с использованием баз данных. Была выбрана бесплатная СУБД MySQL.

Данная СУБД имеет удобный веб-интерфейс phpMyAdmin.



Рис. 6. Интерфейс phpMyAdmin

Главное преимущество выбранной СУБД в том, что работа с ней поддерживается серверным языком программирования php на уровне языка, главная работа разработчика сводится к написанию SQL-скриптов для чтения необходимой информации.

## 1.3. Анализ и выбор программных средств разработки

### 1.3.1. Веб-серверы

Выбор сервера, исполняющего серверную часть приложения, обусловлен выбранными технологиями: PHP и MySQL.

Для выполнения сценариев на языке PHP необходимо использовать сервер Apache.

Apache является кроссплатформенным ПО, поддерживает операционные системы Linux, BSD, Mac OS, Microsoft Windows, Novell NetWare, BeOS.

Основными достоинствами Apache считаются надёжность и гибкость конфигурации. Он позволяет подключать внешние модули для предоставления данных, использовать СУБД для аутентификации пользователей, модифицировать сообщения об ошибках и т. д.

Apache поддерживает множество современных языков, среди которых PHP, Python, Ruby, Perl, ASP.

Для использования связки PHP+MySQL была использована сборка сервера XAMPP.

XAMPP — кроссплатформенная сборка веб-сервера, содержащая Apache, MySQL, интерпретатор скриптов PHP, язык программирования Perl и большое количество дополнительных библиотек, позволяющих запустить полноценный веб-сервер.

Изначально XAMPP создавался как инструмент для разработчиков, позволяя веб-дизайнерам и программистам тестировать свою работу, не используя Интернет. Для упрощения работы некоторые возможности и настройки безопасности отключены по умолчанию, и в целом XAMPP рекомендуется к использованию только при разработке, локально.

Выбор обусловлен бесплатностью, отсутствием сложной настройки – готовность к использованию сразу после установки.

### 1.3.2. Среды разработки

NetBeans IDE



Рис. 7. Главное окно NetBeans IDE во время разработки

NetBeans IDE — свободная интегрированная среда разработки приложений (IDE) на языках программирования Java, JavaFX, Python, PHP, JavaScript, C++, Ада и ряда других.

Для разработки программ в среде NetBeans и для успешной инсталляции и работы самой среды NetBeans должен быть предварительно установлен Sun JDK или J2EE SDK подходящей версии. Среда разработки NetBeans по умолчанию поддерживала разработку для платформ J2SE и J2EE. Начиная с версии 6.0 Netbeans поддерживает разработку для мобильных платформ J2ME, C++ и PHP без установки дополнительных компонентов.

Проект NetBeans IDE поддерживается и спонсируется компанией Oracle, однако разработка NetBeans ведется независимым сообществом разработчиков-энтузиастов (NetBeans Community) и компанией NetBeans Org.

Notepad++

Notepad++ — свободный текстовый редактор с открытым исходным кодом для Windows с подсветкой синтаксиса большого количества языков программирования и разметки. Базовая функциональность программы может быть расширена как за счёт плагинов, так и сторонних модулей, таких как компиляторы и препроцессоры



Рис. 8. Главное окно Notepad++ во время разработки

Базовые возможности:

- Подсветка синтаксиса;

- Сворачивание кода;

- Автодополнение и автоматическое закрытие скобок и тэгов (если активировано);

- Закладки;

- Регулярные выражения для поиска и замены;

- Запись и воспроизведение макросов;

- Сравнение файлов;

- Переопределение любых горячих клавиш;

- Резервное копирование сохраняемых файлов (включается в настройках);

- Трансформация текста при помощи подключённого плагина TextFX;

- Поддержка и конвертирование кодировок ANSI, UTF-8 и UCS-2;

- Блоковое выделение текста, одновременное выделение нескольких разных мест;

- Мультистрочное редактирование.

Выводы

NetBeans IDE была выбрана в качестве средства разработке программных кодов.

Основное преимущество перед Notepad++ и другими, менее популярными средствами является возможность отладки PHP кода и его запуска из под среды. При этом, вся необходимая функциональность содержалась сразу после установки, без необходимости установки дополнительных плагинов. При этом, по каждой стандартной функции имеется быстрая справка.

Также, особенностью выбранной среды является автодополнение кода, что дополнительно сокращает время разработки и устраняет возможность опечаток.

### 1.3.3. Средства отладки программного кода

Отладка серверной части

Отладка серверной части происходила внутри выбранной среды разработки NetBeans средствами встроенного отладчика Xdebug.

Xdebug - свободная библиотека для разработчиков PHP, написанная одним из разработчиков языка PHP. Используется при отладке программного кода, написанного на языке PHP.



Рис. 9. Главное окно NetBeans IDE во время отладки Xdebug

Встроенный отладчик имеет всю необходимую функциональность:

- точки останова;

- слежение за переменными;

- построчное выполнение;

- запуск произвольного скрипта.

Отладка клиентской части

Отладка клиентской части обусловлена используемым для тестирования браузером. В данном случае была использована встроенная в браузер Google Chrome консоль разработчика (Developer Tools).

Панель Elements в Developer Tools

В рабочем пространстве слева показан исходный код страницы с подсветкой синтаксиса. Причём не только исходный, а отражающий реальную DOM-структуру документа, включая и элементы созданные динамически, которые при обычном просмотре исходного кода не видны. Всё организовано в виде многоуровневого списка, т.е. при нажатии на ► можно увидеть все вложенные элементы (потомки) данного. Путь до выбранного тэга отображается под кодом (прямо над консолью). А вот справа отображены все свойства сгруппированные по категориям выбранного в данный момент элемента.

[](http://u-proga.net/category/internet/browsers/)

Рис. 10. Google Chrome - Панель Elements в Developer Tools

Панель Resources в Developer Tools

Этот инструмент подаёт загруженную страницу в виде составляющих: самого файла html, рисунков, стилей css, скриптов, подключенных. А также проводится разбивка на фреймы, если они используются. Все эти компоненты в виде иерархического списка размещены в панели слева, которая напоминает панель проводника Windows. При выборе конкретного компонента, он показывается в правой части. Так доступны для просмотра кода и редактирования скрипты, таблицы стилей.

Также есть в возможность посмотреть, какие файлы cookies установил сайт, страница которого анализируется.

Панель Networks в Developer Tools

В этом разделе показываются все запросы, выполненные при загрузке данной страницы и её компонентов.

Панель Scripts в Developer Tools

Она очень похожа на Elements, только предназначена для анализа и отладки скриптов. Выбрать конкретный скрипт можно из списка всех загруженных слева вверху. Сразу в окне слева отображается его код. Панель справа по виду и структуре такая как в Elements, но со специализированными вкладками. «Watch Expression» – можно задать выражение (переменную), которую вы хотите наблюдать. В других – задать точки останова, стэк вызовов и другое в этом роде.

[](http://u-proga.net/category/internet/browsers/)

Рис. 11. Google Chrome - Панель Scripts в Developer Tools

Отладка клиентской части

Отладка клиентской части обусловлена используемым для тестирования браузером. В данном случае была использована встроенная в браузер Opera консоль разработчика (Developer Tools). Developer Tools — инструмент для веб-разработчиков. Работает на технологии JavaScript. Код Opera Dragonfly выпущен под свободной лицензией BSD.

Возможности:

* Просмотр DOM;
* Просмотр и редактирование CSS;
* Отладчик JavaScript;
* Просмотр HTTP запросов;
* Отлаживать страницы для мобильных устройств;

Удалённо подключаться к любым компьютерам и устройствам с установленным браузером Opera, поддерживающим данный инструмент, и осуществлять отладку веб-страниц;

Рис. 12. Вид Opera Dragonfly при инспектировании элемента.



Рис. 13. Opera Dragonfly. Область свойств CSS.

### 1.3.4. Система контроля версий

Для организации совместной работы двух разработчиков одного проекта имеет место использование системы управления версиями.

Система управления версиями — программное обеспечение для облегчения работы с изменяющейся информацией. Система управления версиями позволяет хранить несколько версий одного и того же документа, при необходимости возвращаться к более ранним версиям, определять, кто и когда сделал то или иное изменение, и многое другое.

Такие системы наиболее широко используются при разработке программного обеспечения для хранения исходных кодов разрабатываемой программы. Однако они могут с успехом применяться и в других областях, в которых ведётся работа с большим количеством непрерывно изменяющихся электронных документов. В частности, системы управления версиями применяются в САПР, обычно в составе систем управления данными об изделии (PDM).

Ситуация, в которой электронный документ за время своего существования претерпевает ряд изменений, достаточно типична. При этом часто бывает важно иметь не только последнюю версию, но и несколько предыдущих. В простейшем случае можно просто хранить несколько вариантов документа, нумеруя их соответствующим образом. Такой способ неэффективен (приходится хранить несколько практически идентичных копий), требует повышенного внимания и дисциплины и часто ведёт к ошибкам, поэтому были разработаны средства для автоматизации этой работы.

Традиционные системы управления версиями используют централизованную модель, когда имеется единое хранилище документов, управляемое специальным сервером, который и выполняет большую часть функций по управлению версиями. Пользователь, работающий с документами, должен сначала получить нужную ему версию документа из хранилища; обычно создаётся локальная копия документа, т. н. «рабочая копия». Может быть получена последняя версия или любая из предыдущих, которая может быть выбрана по номеру версии или дате создания, иногда и по другим признакам. После того, как в документ внесены нужные изменения, новая версия помещается в хранилище. В отличие от простого сохранения файла, предыдущая версия не стирается, а тоже остаётся в хранилище и может быть оттуда получена в любое время. Сервер может использовать т. н. дельта-компрессию — такой способ хранения документов, при котором сохраняются только изменения между последовательными версиями, что позволяет уменьшить объём хранимых данных. Поскольку обычно наиболее востребованной является последняя версия файла, система может при сохранении новой версии сохранять её целиком, заменяя в хранилище последнюю ранее сохранённую версию на разницу между этой и последней версией. Некоторые системы поддерживают сохранение версий обоих видов: большинство версий сохраняется в виде дельт, но периодически (по специальной команде администратора) выполняется сохранение версий всех файлов в полном виде; такой подход обеспечивает максимально полное восстановление истории в случае повреждения репозитория.

Часто бывает, что над одним проектом одновременно работают несколько человек. Если два человека изменяют один и тот же файл, то один из них может случайно отменить изменения, сделанные другим. Системы управления версиями отслеживают такие конфликты и предлагают средства их решения. Большинство систем может автоматически объединить (слить) изменения, сделанные разными разработчиками. Однако такое автоматическое объединение изменений, обычно, возможно только для текстовых файлов и при условии, что изменялись разные (непересекающиеся) части этого файла. Такое ограничение связано с тем, что большинство систем управления версиями ориентированы на поддержку процесса разработки программного обеспечения, а исходные коды программ хранятся в текстовых файлах. Если автоматическое объединение выполнить не удалось, система может предложить решить проблему вручную.

Часто выполнить слияние невозможно ни в автоматическом, ни в ручном режиме, например, если формат файла неизвестен или слишком сложен. Некоторые системы управления версиями дают возможность заблокировать файл в хранилище. Блокировка не позволяет другим пользователям получить рабочую копию или препятствует изменению рабочей копии файла (например, средствами файловой системы) и обеспечивает, таким образом, исключительный доступ только тому пользователю, который работает с документом.

При выборе системы контроля версий были рассмотрены популярные пакеты: Subversion и Git.

Subversion

Subversion (также известная как «SVN») — свободная централизованная система управления версиями, официально выпущенная в 2004 году компанией CollabNet Inc.

Subversion — централизованная система, то есть данные хранятся в едином хранилище. Хранилище может располагаться на локальном диске или на сетевом сервере.

Работа в Subversion мало отличается от работы в других централизованных системах управления версиями. Клиенты копируют файлы из хранилища, создавая локальные рабочие копии, затем вносят изменения в рабочие копии и фиксируют эти изменения в хранилище. Несколько клиентов могут одновременно обращаться к хранилищу. Для совместной работы над файлами в Subversion преимущественно используется модель копирование — изменение — слияние. Кроме того, для файлов, не допускающих слияние (различные бинарные форматы файлов), можно использовать модель блокирование — изменение — разблокирование.

При сохранении новых версий используется дельта-компрессия: система находит отличия новой версии от предыдущей и записывает только их, избегая дублирования данных.

Возможности:

- хранение полной истории изменений отслеживаемых объектов (файлов, каталогов, символьных ссылок) в централизованном хранилище (репозитории), в том числе при изменении атрибутов («метаданных»), перемещении, переименовании и удалении;

- копирование объектов с разветвлением истории — при копировании в хранилище появляются два отдельных объекта с общей историей;

- поддержка переноса изменений между копиями объектов, в том числе полного слияния копий (в рабочей копии; без объединения истории);

- поддержка ветвления: создания ветвей (копированием директорий) и работы с ними, слияние ветвей (переносом изменений);

- поддержка меток (копированием директорий);

- история изменений и копии объектов (в том числе ветви и метки) хранятся в виде связанных разностных копий — «дешёвых» (не требующих больших временны́х и дисковых ресурсов) при создании и хранении;

- поддержка конкурентной (в том числе одновременной, с изоляцией транзакций) многопользовательской работы с хранилищем и, в большинстве случаев, автоматическим слиянием изменений различных разработчиков (в рабочей копии);

- фиксации изменений в хранилище (в том числе многообъектные) организуются в виде атомарных транзакций;

- сетевой обмен между сервером и клиентом предусматривает передачу только различий между рабочей копией и хранилищем;

- обеспечивается одинаково эффективная работа как с текстовыми, так и с двоичными файлами;

- различные варианты доступа к хранилищу, в том числе: непосредственный доступ на локальной файловой системе; по собственному сетевому протоколу; через веб-сервер по протоколу WebDAV/DeltaV;

- вывод клиента командной строки одинаково удобен и для чтения, и для разбора программами;

- возможность зеркалирования хранилища;

- библиотеки для языков PHP, Python, Perl, Java позволяют встроить функциональность клиента Subversion в программы, написанные на этих языках;

- многоуровневая архитектура библиотек, изначально рассчитанная на клиент-серверную модель.

Git

Git —система управления версиями файлов. Проект был создан Линусом Торвальдсом для управления разработкой ядра Linux, первая версия выпущена 7 апреля 2005 года.

Это система является распределенной. Такие системы используют распределённую модель вместо традиционной клиент-серверной. Они, в общем случае, не нуждаются в централизованном хранилище: вся история изменения документов хранится на каждом компьютере, в локальном хранилище, и при необходимости отдельные фрагменты истории локального хранилища синхронизируются с аналогичным хранилищем на другом компьютере. В некоторых таких системах локальное хранилище располагается непосредственно в каталогах рабочей копии.

Когда пользователь такой системы выполняет обычные действия, такие как извлечение определённой версии документа, создание новой версии и тому подобное, он работает со своей локальной копией хранилища. По мере внесения изменений, хранилища, принадлежащие разным разработчикам, начинают различаться, и возникает необходимость в их синхронизации. Такая синхронизация может осуществляться с помощью обмена патчами или так называемыми наборами изменений между пользователями.

Git поддерживает быстрое разделение и слияние версий, включает инструменты для визуализации и навигации по нелинейной истории разработки. Git предоставляет каждому разработчику локальную копию всей истории разработки, изменения копируются из одного репозитория в другой.

Удалённый доступ к репозиториям Git обеспечивается git-daemon, SSH- или HTTP-сервером. TCP-сервис git-daemon входит в дистрибутив Git и является наряду с SSH наиболее распространённым и надёжным методом доступа. Метод доступа по HTTP, несмотря на ряд ограничений, очень популярен в контролируемых сетях, потому что позволяет использовать существующие конфигурации сетевых фильтров.

Выводы

В данной работе использована система управления версиями Git.

Выбор обусловлен тем, что она является распределенной, а это дает следующие преимущества:

- легкая операция создания ветвей;

- хранение локальной копии всего репозитория, что позволяет каждому разработчику полноценно работать с системой в условиях отсутствия интернета.

Для использования репозитория Git был выбран самый популярный бесплатный сервис GitHub, доступный по адресу: <https://github.com/>. Он предоставляет учетные записи каждому пользователю, 5Гб свободного место под исходные коды и построение различных визуальных диаграмм.

В качестве клиента Git была выбрана графическая оболочка TortoiseGit. Данное ПО выполнено в виде расширения к Windows Explorer, то есть все его функция доступны в контекстном меню директории, в которой размещаются исходные коды.

Утилита поддерживает все основные возможности Git, а также предоставляет статистику использования.

Внешний вид доступных функций показан на рисунке ниже.



Рис. 14. Меню TortoiseGit.

### 1.3.5. Вспомогательные средства

Во время проектирования и разработки понадобились следующие программные средства:

- Браузеры: Internet Explorer 9.0, Google Chrome 19, Opera 11 – для тестирования конечного приложения и адаптации внешнего вида под конкретную платформу.

- Microsoft Visio – для быстрого создания визуальных макетов приложения и диаграмм при проектировании. Редактор диаграмм и блок-схем для Windows. Использует векторную графику.

- Skype – обеспечение мгновенной связи между разработчиками по срочным вопросам. Программа позволяет совершать конференц-звонки, видеозвонки, а также обеспечивает передачу текстовых сообщений и передачу файлов. Есть возможность вместо изображения с веб-камеры передавать изображение с экрана монитора.

# 2. Конструкторская часть

## 2.1. Анализ и проектирование архитектуры приложения

С ростом функциональности программного продукта, растет и его сложность. При этом растет и количество исходных кодов, что может стать большой проблемой при наличии плохо спроектированной структуры.

В нашем проекте изначально решено заранее спроектировать будущую архитектуру приложения, сделав ее в меру сложной, но при этом и легко масштабируемой. Еще одной ключевой задачей было обеспечение структурированности и упорядоченности кода для более легкого дальнейшего сопровождения.

Современные методы проектирования архитектуры представлены паттернами (шаблонами) проектирования MVC, MVP и некоторыми другими.

MVC (модель-представление-контроллер)



Рис. 15. Визуальное представление паттерна MVC

Под Моделью, обычно понимается часть содержащая в себе функциональную логику приложения. Как именно организован этот слой, по большому счету не важно, однако есть ряд ключевых моментов, на которых мы остановимся позднее. Основная цель паттерна - сделать так, чтобы Модель была полностью независима от остальных частей и практически ничего не знала об их существовании, что позволило бы менять и Контроллер и Представление модели, не трогая саму Модель и даже позволить функционирование нескольких экземпляров Представлений и Контроллеров с одной Моделью одновременно. Вследствие чего, Модель ни при каких условиях не может содержать ссылок на объекты Представления или Контроллера.

В обязанности Представления входит отображение данных полученных от Модели. Обычно Представление имеет свободный доступ к Модели и может брать из нее данные, однако это доступ только на чтение, ничего менять в Модели или даже просто вызывать методы, приводящие к изменению ее внутреннего состояния, Представлению позволять нельзя. В случае активной Модели, Представление может подписаться на события изменения Модели и перерисовываться, забрав измененные данные, при получении соответствующего оповещения. Для взаимодействия с Контроллером, представление, как правило, реализует некий интерфейс, известный Контроллеру, что позволяет менять представления независимо и иметь несколько представлений на Контроллер. Вообще, подмена или изменение Представления самая часто встречающаяся задача, по сути это и есть та причина по которой придумывают различные паттерны разделения Модели и Представления.

В задачи Контроллера входит реакция на внешние раздражители и изменение Модели и/или Представления в соответствии с заложенной в него логикой. Один Контроллер может работать с несколькими Представлениями, в зависимости от ситуации, взаимодействуя с ними через некий заранее известный интерфейс, который эти Представления реализуют. Важный нюанс, в классической версии MVC Контроллер не занимается передачей данных из Модели в Представление и не является медиатором между Моделью и Представлениями.

В нашем случае недостатком такого подхода является то, что Модель и Представление слишком связаны между собой, что не применимо к гибким веб-приложениям, поскольку часто приходится менять Представления.

MVP (модель-представление-представитель)

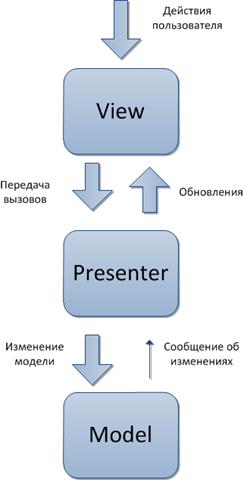


Рис. 16. Визуальное представление паттерна MVP

Паттерн Model-View-Presenter, является модификацией MVC. Вместо Контроллера здесь появляется Представитель, который выполняет немного другую задачу.

Прежде всего, количество связей между сущностями уменьшилось, как правило, в MVP Модель не общается с Представлением даже опосредовано, через механизм оповещений, но это не обязательное условие, более глубокие отличия лежат как раз в области Представителя.

В MVP Представитель, в отличие от Контроллера, общается непосредственно с моделью приложения, а не с неким промежуточным звеном. Это позволяет, в случае необходимости, общаться Представителю с Представлениями, минуя Модель приложения и событийный механизм с этим связанный, так как Представитель в отличии от Контроллера, обладает необходимыми знаниями о Представлениях, а так же удаляет лишнюю логику обработки событий от Модели из Представления.

Вследствие этого в отличии от MVC, Представление в MVP является «ультра-тонким» и не содержит логики, зависимой от конкретного Представителя.

Данный вариант структуры удовлетворяет требованию развязности Модели от реализации внешнего вида, поэтому рассмотрим возможные реализации этой структуры в разрабатываемом веб-приложении.

Централизованная схема

Первоначально была предложена так называемая централизованная схема. Она подразумевает присутствие центрального скрипта-обработчика запросов, который, в зависимости от надобности, вызывал бы компоновщики необходимых страниц. Рассмотрим основные элементы данной структуры:

Представитель:

index.php – центральный скрипт, который разбирает запросы пользователей на сервере и передает их соответствующим компоновщикам, в зависимости от запроса.

Передача происходит с проверкой безопасности. Проверяется корректность запроса, валидность аргументов и права.

forum.php, admin.php и другие – компоновщики конкретных страниц. Их роль заключается в передаче выводимой информации из Базы Данных в специальный шаблонизатор.

Представление:

Шаблонизатор – специальная сущность, главной задачей которой – заполнение шаблонов набором подготовленных данных.

Шаблоны – набор .tpl, .css, .js файлов, определяющих внешний вид страницы и поведение на клиентских машинах. Файлы tpl представляют собой html-страницы, размеченные специальными метками:

<div class="wrapper">

<?=$this->header?>

</div>

На эти метки реагирует шаблонизатор и подставляет необходимые данные.

Модель:

Модель представлена базой данных MySQL и обслуживающими ее скриптами. Главная задачи этих скриптов – обрабатывать основные запросы со стороны компоновщиков: добавление, удаление, изменение и извлечение данных из базы.



Рис. 17. Централизованный вариант реализации архитектуры

Проблемы данного подхода:

- центральный скрипт становится узким местом в приложении, возможны проблемы производительности;

- центральный скрипт должен содержать слишком сложную логику для выборки нужного компоновщика;

- проверка безопасности происходит на начальном этапе. Как следствие, модулю безопасности необходимо знать о корректности запроса, корректности аргументов уже на начальном этапе, что сильно усложняет логику приложения и увеличивает вероятность ошибок;

- конечный пользователь будет видеть адрес страницы, всегда начинающийся с названия центрально скрипта – теряется логический смысл ссылок.

Данные проблемы были решены в следующем варианте.

Распределенная схема

 Рис. 18. Распределенный вариант реализации архитектуры

В отличие от предыдущего варианта, запрос поступает сразу на компоновщик конкретной страницы, без промежуточного центрального скрипта. Это дает возможность запускать напрямую любой компоновщик из строки браузера.

Часть проверок безопасности перенесена в Модель, на уровень скриптов к Базе Данных. На этом уровне легко выделяются аргументы для проверки, и уменьшается количество логики. Остальная часть проверок, такая, как разделение прав пользователей, проверка авторизации перенесена в компоновщики для ограничения вывода информации.

Главной проблемой данного варианта является то, что любой компоновщик может содержать блоки, которые повторяются в других компоновщиках. Отсюда вытекает 2 проблемы:

- проблема повторяемости кода – каждый компоновщик будет содержать проверки авторизации, проверки безопасности и т.д.

- проблема сопровождаемости кода – при обнаружении ошибки, либо изменении логики в одном компоновщике появляется необходимость исправлять соответствующие блоки во всех других компоновщиках.

Данные проблемы были решены в финальном варианте структуры приложения.

Модульная схема

 Рис. 19. Модульный вариант реализации архитектуры

В данном варианте каждый логический блок страницы изолирован от других в отдельном модуле.

Каждый модуль имеет доступ к Модели через скрипты к базе данных и к Представлению, через Шаблонизатор и набор шаблонов. Также, каждый модуль содержит свои необходимые проверки безопасности, отдавая компоновщикам только признаки того, что загрузка модуля прошла успешно, либо не успешно.

Роль компоновщиков в данном случае – подключать необходимые модули для генерации той или иной конечной страницы.

В результате паттерн MVP приобрел следующий вид:

Представитель:

- компоновщики конкретных страниц;

- набор модулей.

Модель:

- скрипты к базе данных и конечные проверки корректности аргументов;

- база данных MySQL.

Представление:

- шаблонизатор;

- набор шаблонов, стилей и клиентских скриптов.

## 2.2. Проектирование и разработка базы данных

### 2.2.1. Логическое проектирование

CMS располагает контентом, который необходимо где-то хранить. Так же в ней содержится информация о пользователях, настройках и т.д. Поэтому для CMS необходима собственная база данных.

На этапе логического проектирования определяются сущности и их взаимосвязи.

Весь контент можно объединить по следующим категориям:

• пользователи

• текстовая информация форума

• права пользователей

• настройки сайта

Рассмотрим группу «пользователи». Каждый участник имеет личную информацию о себе и наделен некоторыми правами. Так как система будет содержать много групп пользователей с одинаковыми правами, так же понадобится сущность «группа пользователей». Пользователь состоит в группе, а группа наделена правами.

Текстовая информация представляет собой структурированные по категориям топики, которые содержат в себе текстовые сообщения. Таким образом, образуются сущности «категория», «тема», «сообщения». Сообщение связано с темой, которой она принадлежит, тема связана с категорией. При этом, необходимо обеспечить такой тип связи, при котором удаление родительской сущности вызывает удаление дочерних сущностей. Так же, к категории может применятся определенное действие пользователя, следовательно, нам необходимо иметь сущность «право категории».

Право в системе – это определенное действие (удаление контента, размещение сообщения). Таким образом, появляется сущность «право». Право может быть групповым, либо право категории. При удалении права, оно удаляется в связанных сущностях. Связанные сущности – «право группы пользователей» и «право категории». Если удаляется группа пользователей – с ней удаляются все связанные с ней права.

### 2.2.2. Физическое проектирование

Физическое проектирование — создание схемы базы данных для конкретной СУБД. Специфика конкретной СУБД может включать в себя ограничения на именование объектов базы данных, ограничения на поддерживаемые типы данных и т. д.

База данных хранится в формате SQL.

Все поля таблиц, включая сами таблицы именуются символами “a-z” а так же нижнее подчеркивание.

Каждое поле имеет свой размер, что выгодно сказывается на производительности базы данных. Если размер не указан, то СУБД выбирает максимальный размер, следовательно, транзакции требуют больше оперативной памяти и процессорного времени.

Для всех числовых значений был выбран тип INT длиной 11 бит. Число является беззнаковым, следовательно, диапазон хранимых чисел равен 0…211.

Для коротких строковых значений был выбран тип VARCHAR с размерностью 255 символов.

Для больших текстовых значений используются другие размерности, но тип так же VARCHAR.

Даты хранятся в формате DATETIME.

Булевые значения хранятся в формате TINYINT.

В каждой таблице выбирается ключевое поле, которое однозначно идентифицирует запись в этой таблице.

Так же устанавливаются связи между таблицами в соответствии с логическим проектированием (см. ПРИЛОЖЕНИЕ 1).



Рис. 20. Поля и типы таблицы users



Рис. 21. Связи таблицы users



Рис. 22. Поля и типы таблицы categories



Рис. 23. Поля и типы таблицы categories\_permissions



Рис. 24. Связи таблицы categories\_permissions



Рис. 25. Поля и типы таблицы general\_permissions



Рис. 26. Поля и типы таблицы menu



Рис. 27. Поля и типы таблицы posts





Рис. 28. Связи таблицы posts



Рис. 29. Поля и типы таблицы settings



Рис. 30. Поля и типы таблицы topics





Рис. 31. Связи таблицы topics



Рис. 32. Поля и типы таблицы user\_groups\_permissions

 Рис. 33. Связи таблицы user\_groups\_permissions



Рис. 34. Поля и типы таблицы users





Рис. 35. Связи таблицы users



Рис. 36. Поля и типы таблицы user\_groups

## 2.3. Проектирование программных модулей

Модуль регистрации

Клиентская часть – это то, что видит пользователь. Клиентская часть регистрационной части – это веб-форма, которая содержит в себе поля ввода, и кнопку «Регистрация»

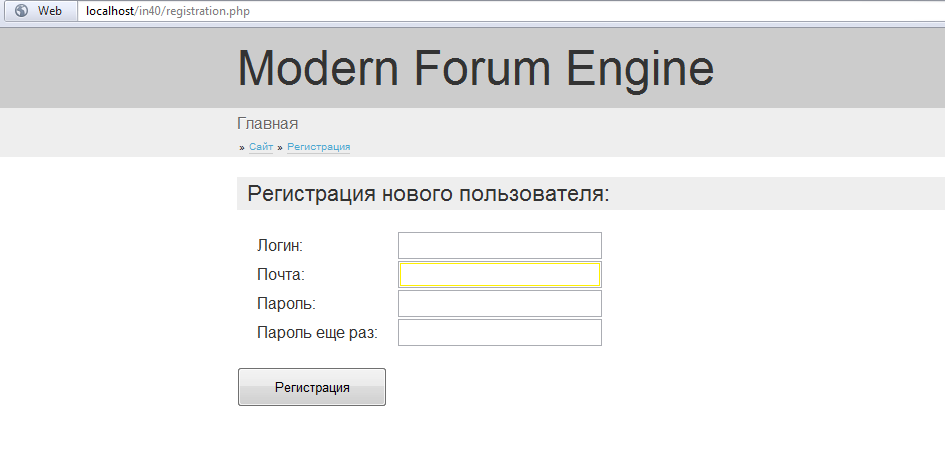


Рис. 37. Страница регистрации нового пользователя.

Все поля должны быть корректны. Алгоритм проверки корректности полей изображен на рисунке ниже.



Рис. 38. Алгоритм проверки корректности полей.

Логин должен быть уникальным, содержать латинские буквы, а так же кириллицу. Данная проверка осуществляется функциями Javascript. Если логин прошел эту проверку, он отправляется ajax запросом на сервер. Сервер извещает клиент о статусе логина – уникален он или нет. После этого происходит проверка почты функциями Javascript. Так же сравниваются пароли на совпадение. Если все проверки пройдены успешно, пользователь может их отправить на сервер, нажав кнопку «Регистрация».

В случае некорректности каких-либо данных выводятся сообщения, которые оповещают пользователя об этом. Примеры этих сообщений изображены на рисунке ниже.

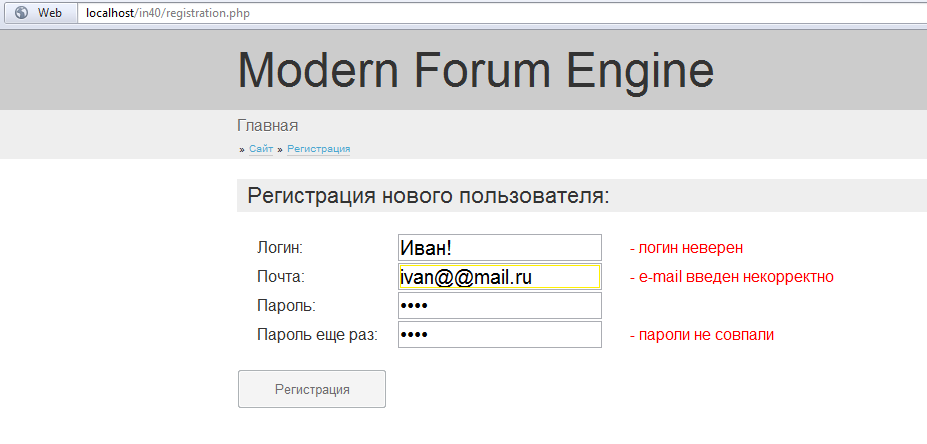


Рис. 39. Пример некорректного заполнения формы

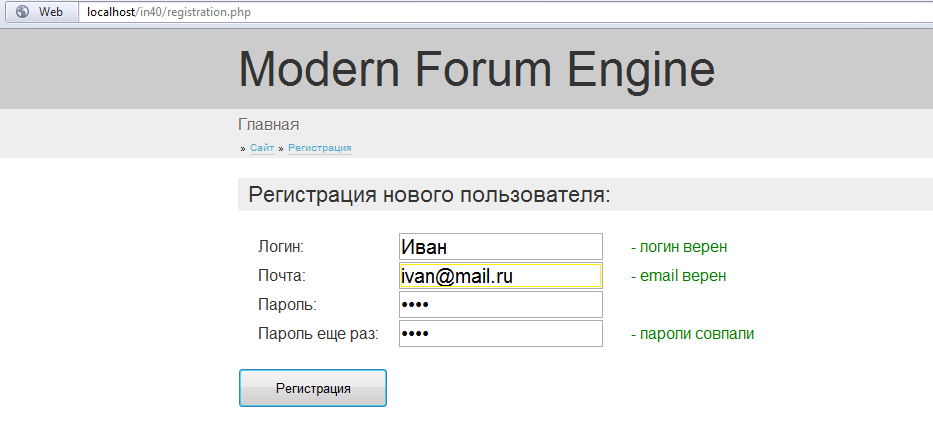


Рис. 40. Пример корректного заполнения формы

Перед отправкой всех данных, клиентская часть шифрует пароль, чтобы он был передан на сервер в зашифрованном виде. Это делается при помощи функций Javascript.

При создании регистрационной части были созданы следующие файлы:

* Registration.php – компоновщик страницы регистрации;
* Registration.js – скрипт, обеспечивающий интерактивность формы, проверку корректности и обмен данными между серверами(ajax).
* Registration.tpl – шаблон формы регистрации

Данные со стороны клиента поступают на сервер и записываются в базу данных. Но прежде чем это будет сделано, данные проходят еще один этап проверки. Это делается для большей безопасности, во избежание атак, цель которых – занести код в базу данных, чем самым получить доступ к ней.



Рис. 41. Серверная проверка корректности данных.

После проверки данные заносятся в базу данных.



Рис. 42. Поля таблицы пользователей в базе данных

Поля имеют следующие предназначения:

* Id – уникальный идентификатор пользователя в системе
* Login – логин пользователя
* User\_group\_id – идентификатор группы пользователей, к которой принадлежит данный пользователь
* Name – имя
* Father\_name – отчество
* Family\_name – фамилия
* Sex – пол
* Country – страна
* City – город
* Avatar – ссылка на сервис gravatar, где хранится фотография пользователя
* Mobile – мобильный телефон
* Icq – сервис icq
* Vk – сервис вконтакте
* Facebook – сервис facebook
* About – информация о пользователе
* Registration\_date – дата регистрации
* Email – почта
* Password – хеш пароля

Если пользователю необходимо заполнить все поля, он может заполнить их после регистрации.

Модуль пользователя

Дополнить или изменить личные данные можно в профиле пользователя.

Заполненные поля отправляются на сервер, где проходят проверку на корректность, во избежание внедрения SQL-кода. После этого данные заносятся в базу данных.

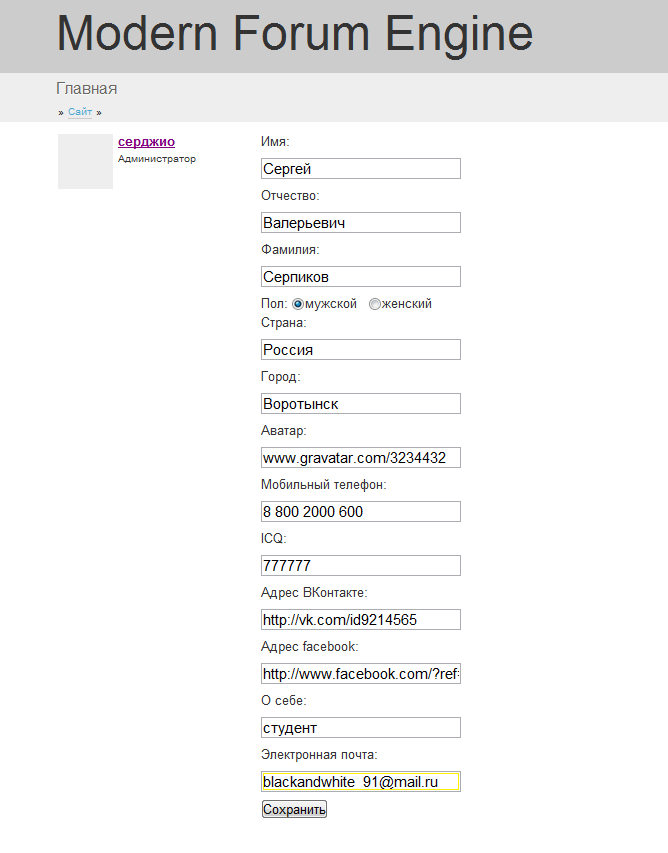


Рис. 43. Редактирование данных пользователя

Модуль навигации

Модуль навигации помогает пользователю перемещаться по разделам сайта.



Рис. 44. Пример навигационного меню

Пункты навигационного меню отображают путь до текущей страницы внутри сайта.

Был созданы файлы:

* bread\_crumbs.php – главный скрипт, собирающий модуль
* nav-item.tpl – шаблон пункта модуля навигации

Чтобы на странице был модуль навигации, необходимо в компоновщике страницы подключить данный модуль и вызвать инициализацию данного модуля:

include\_once 'modules/bread\_crumbs.php';

...

$page\_areas['nav'] = construct\_bread\_crumbs();

Главный скрипт bread\_crumbs.php определяет текущее положение страницы, просматриваемой пользователем, в иерархии сайта.

Модуль логина

Модуль логина выводит поля ввода логина и пароля, а так же ссылку на форму регистрации, чтобы незарегистрированный пользователь мог зарегистрироваться. Если пользователь авторизовался на сайте, то модуль логина выводит текущий логин пользователя и ссылку для сброса авторизации.



Рис. 45. Модуль логина до авторизации



Рис. 46. Модуль логина после авторизации

Модуль логина состоит из файлов

* log\_reg.php – компоновщик
* login-form.tpl – шаблон для ввода пароля и логина
* login-hello.tpl – шаблон для выхода и индикации текущего пользователя

Для того, чтобы включить данный модуль необходимо в компоновщике страницы подключить данный модуль и вызвать инициализацию данного модуля:

include\_once 'modules/log\_reg.php';

...

$page\_areas['login'] = construct\_log\_reg();

## 2.4. Проектирование внешнего вида приложения

Внешний вид, занимаемый в паттерне MVP место Представления (View) представлен следующими Шаблонизатором и набором шаблонов, стилей и скриптов, выполняемых на клиентской машине.

Шаблонизатор

Выполнен отдельным классом TemplateLoader. Этот класс содержит информацию о текущей выбранной теме и директории, где расположены все темы. Потенциально, он может обрабатывать любые шаблоны, созданные для этого веб приложения, поэтому обеспечивается требуемая гибкость.

Главный метод этого класса Load принимает на вход название шаблона, который потребовалось вывести на страницу и набор переменных, которым Шаблонизатор этот шаблон заполнит.

Шаблоны

Каждый шаблон представляет из себя файл .tpl, который внутри содержит html-разметку страницы, либо отдельного модуля. Например, post.tpl содержит шаблон одного сообщения с метками для фото, псевдонима, группы пользователя:

<div class="post">

<div class="post-row">

<div class="user-photo"><img src="<?=$this->user\_photo?>" width=55></div>

<div class="user-info-area">

<a class="user-nickname" href="<?=$this->user\_profile?>"><?=$this->login?></a>

<br>

<span class="user-group"><?=$this->user\_group?></span>

<br>

<a class="post-date" href="#"><?=$this->date?></a>

</div>

<div class="post-body">

<?=$this->body?>

</div>

</div>

<div class="post-row">

<div class="reply-area">

<?=$this->delete\_post\_link?>

</div>

</div>

</div>

В первую очередь был разработан общий макет-правило, который позволил структурировать и упорядочить дальнейший процесс разработки веб-интерфейса.

 Рис. 47. Уровни представления веб-интерфейса

Будущая страница будет генерироваться по следующим уровням:

1) Уровень частей: состоит из верхней части и нижней части. Реализован с чисто технической целью организации «подвала» - нижней части контента

2) Уровень областей: логически более нагруженный уровень. Каждая часть делится на области, созданные для какой-то определенной цели: область заголовка, навигации, меню, контента и т.д.

3) Уровень списков. Чаще всего контент представлен в виде списков логически связанных элементов – пункты меню, уровни навигации, список тем, список сообщений и т.д. Каждый список будет заполнен шаблонизатором соответствующими элементами.

Стили

Файлы стилей написаны с помощью каскадных таблиц стилей CSS и содержит сведения о фоне, цветах, шрифте и других стилевых особенностях тех или иных элементов страницы. Например, для списка тем введено такое правило:

.topic

{

border: 1px solid lightgray;

margin: 0 0 2px 0;

font: 12px sans-serif;

width: 790px;

}

это означает, что каждая тема будет обрамляться тонкой светло-серой рамкой, между темами в списке будет небольшой отступ, сама тема будет набрана шрифтом без засечек и ширина фиксирована – 790px.

Клиентские скрипты

Клиентские скрипты написаны на языке JavaScript с использованием серверного расширения Ajax, позволяющего делать HTTP-запросы не перезагружая страницу.

Клиентские скрипты были широко использованы при реализации каждого модуля.

На примере модуля регистрации (содержатся в фале registration.js):

- реализованы проверки корректности пароля (пароль должен быть от 4х символов и выше и совпадать с полем повтора пароля);

- проверка валидности e-mail;

- проверка корректности введенного псевдонима (псевдоним должен содержать только разрешенные символы, и быть в пределах от 4х до 20 символов);

- также введена проверка на существование пользователя с таким же e-mail и псевдонимом – с помощью Ajax-запроса к Базе данных;

Если введенная пользователем информация корректна – кнопка регистрации станет активной.

Ниже представлен код проверки корректности введенного e-mail.

// проверка e-mail - валидация+совпадение в БД

function check\_email()

{

var a=document.getElementById('email').value;

if (a.length!=0)

{

if (!(/^((([a-z]|[0-9]|!|#|$|%|&|'|\\*|\+|\-|\/|=|\?|\^|\_|`|\{|\||\}|~)+(\.([a-z]|[0-9]|!|#|$|%|&|'|\\*|\+|\-|\/|=|\?|\^|\_|`|\{|\||\}|~)+)\*)@((((([a-z]|[0-9])([a-z]|[0-9]|\-){0,61}([a-z]|[0-9])\.))\*([a-z]|[0-9])([a-z]|[0-9]|\-){0,61}([a-z]|[0-9])\.)[\w]{2,4}|(((([0-9]){1,3}\.){3}([0-9]){1,3}))|(\[((([0-9]){1,3}\.){3}([0-9]){1,3})\])))$/i.test(a)))

{

document.getElementById('correct\_email\_text').style.color="red";

document.getElementById('correct\_email\_text').innerHTML="&nbsp - e-mail введен некорректно";

return false;

}

if (is\_exist(a,"email")=="true") // проверка есть или нет в БД

{

document.getElementById('correct\_email\_text').style.color="red";

document.getElementById('correct\_email\_text').innerHTML="&nbsp - такой почтовый ящик уже зарегистрирован";

return false;

}

document.getElementById('correct\_email\_text').style.color="green";

document.getElementById('correct\_email\_text').innerHTML="&nbsp - email верен";

return true;

}

return false;

}

## 2.5. Проектирование безопасности приложения

При проектировании любого веб-приложения необходимо обеспечивать безопасность по следующим направлениям:

- обеспечивать конфиденциальность информации;

- обеспечивать контроль над приложением;

- предотвращать использование контента в незаконных целях.

Общая защита модулей

Теоретически, для доступа к ограниченному контенту можно использовать некорректно построенные запросы. Например, нет никаких ограничений касательно того, что введет пользователь в строку браузера. Вероятны такие ситуации, когда злоумышленник, в целях обнаружения лазеек и уязвимостей, может запускать произвольные скрипты на выполнение.

В нашем проекте, потенциально уязвимыми скриптами являются все модули и скрипты работы с базой данных, т.к. проверки прав пользователей и некоторые другие реализованы в компоновщиках. Такая уязвимость имеет место быть в случае, если пользователь запросит не компоновщик, который соберет страницу верно, а напрямую уязвимый скрипт.

Во избежание этого, было принято решение ограничить доступ ко всем серверным скриптам, не относящимся к компоновщикам. Был применен следующий прием:

Каждый корректно запускаемый компоновщик содержит следующую строку:

define('Katrin', 1);

что означает, что некая переменная-признак объявлена.

А в начале каждого потенциально уязвимого скрипта производится проверка:

if (!defined('Katrin'))

die ('Access Error');

что означает «если переменная-признак не объявлена, то дальше загрузку файла не производить».

В результате, при корректном запросе, будет объявлена переменная признак, проверка будет пройдена и страница будет успешно загружена. при попытке вызвать модуль с проверкой напрямую, переменная не будет объявлена и проверка пройдена не будет, а значит страница с потенциально ограниченным контентом не будет загружена.

Механизм разделения прав пользователей

Для обеспечения гибкости предоставления контента, необходимо реализовать систему разделения прав. Чтобы грамотно это сделать, необходимо рассматривать всю систему как хранилище контента, где к каждой единице доступ регулируется системой прав. Кроме контента, система содержит действия пользователей, которые тоже должны регулироваться системой прав.



Рис. 48. Место системы прав в разрабатываемой системе

Чтобы упростить использование системы прав, все пользователи были разбиты по группам. Количество групп – произвольное число.



Рис. 49. Группирование пользователей

База данных содержит таблицу «general\_permissions» со списком прав с описаниями.



Рис. 50. Список прав с описаниями

Так же существует таблица групп пользователей, где каждая группа имеет свой уникальный идентификатор.



Рис. 51. Таблица групп пользователей.

Связующим звеном между правами и группами является таблица «user\_groups\_permissions»



Рис. 52. Соответствия прав группам пользователей

Именно в этой таблице и определяется набор прав у групп пользователей. При построении страницы компоновщиком, определяются права, и на основе их наличия подбирается соответствующий контент и выдаются управляющие ссылки. Если неуполномоченный пользователь вдруг получит управляющую ссылку (ссылку с GET-запросом, например, на удаление темы) при отсутствии прав на выполнении данной команды – команда будет проигнорирована компоновщиком.

Преимущество такого подхода в том, что можно определить право один раз и назначить его конкретной группе, каждый раз, при необходимости, проверяя его наличие.

Шифрование MD5

Для обеспечения безопасности хранения паролей, все пароли в системе хранятся в зашифрованном виде. Вводимый в приложении пароль шифруется и сравнивается с паролем, хранимым в базе данных. Для шифрования используется алгоритм MD5.

MD5 позволяет получать относительно надёжный идентификатор для блока данных. Такое свойство алгоритма широко применяется в разных областях. Оно позволяет искать дублирующиеся файлы на компьютере, сравнивая MD5 файлов, а не их содержимое. Как пример, dupliFinder — графическая программа под Windows и Linux. Такой же поиск может работать и в интернете.

С помощью MD5 проверяют целостность скачанных файлов — так, некоторые программы идут вместе со значением хеша. Например, диски для инсталляции.

MD5 используется для хеширования паролей. В системе UNIX каждый пользователь имеет свой пароль и его знает только пользователь. Для защиты паролей используется хеширование. Предполагалось, что получить настоящий пароль можно только полным перебором. При появлении UNIX единственным способом хеширования был DES (Data Encryption Standard), но им могли пользоваться только жители США, потому что исходные коды DES нельзя было вывозить из страны. Во FreeBSD решили эту проблему. Пользователи США могли использовать библиотеку DES, а остальные пользователи имеют метод, разрешённый для экспорта. Поэтому в FreeBSD стали использовать MD5 по умолчанию. Некоторые Linux-системы также используют MD5 для хранения паролей.

Многие системы используют базу данных для хранения паролей и существует несколько способов для хранения паролей.

• Пароли хранятся как есть. При взломе такой базы все пароли станут известны.

• Хранятся только хеши паролей (с помощью MD5, SHA). Найти пароли можно только полным перебором. Но при условии использования несложного, популярного или просто несчастливого пароля (который встречался ранее и занесён в таблицу) такая задача решается за доли секунды. Пароль из таблицы был найден всего за 0,036059 сек.

• Хранятся хеши паролей и несколько случайных символов. К каждому паролю добавляется несколько случайных символов (их ещё называют «salt» или «соль») и результат ещё раз хешируется. Например, md5(md5(pass)+word). Найти пароль с помощью таблиц таким методом не получится.

При регистрации или авторизации, клиентская сторона обрабатывает введенные символы с помощью javascript библиотеки. Функция возвращает md5 хеш значение введенных символов и отправляет данный хеш на сервер для дальнейшей обработки.

Рассмотрим работу алгоритма.

На вход приходит последоватьельность символов. Строка переводится в 16-ричное представление, в конце дописывается 1 бит. Так как один символ имеет размер 1 байт, то последний дописанный байт будет иметь вид 80h. Затем вычисляется размер входной последовательности, в данном случае она равняется 8ми битам. Данное значение записывается в конец таблицы.

В итоге мы имеем таблицу 64 байта, которую в дальнейшем необходимо обработать в 4 раунда.

Устройство шифрования можно представить как процессор, который имеет 4 регистра и умеет выполнять одну команду. Регистры инициализируются следующими значениями:

A = 0x67452301,

B = 0xEFCDAB89,

C = 0x98BADCFE,

D = 0X10325476

Также инициализируется массив констант T

uint[] T = new uint[64]

{ 0xd76aa478,0xe8c7b756,0x242070db,0xc1bdceee,

0xf57c0faf,0x4787c62a,0xa8304613,0xfd469501,

0x698098d8,0x8b44f7af,0xffff5bb1,0x895cd7be,

0x6b901122,0xfd987193,0xa679438e,0x49b40821,

0xf61e2562,0xc040b340,0x265e5a51,0xe9b6c7aa,

0xd62f105d,0x2441453,0xd8a1e681,0xe7d3fbc8,

0x21e1cde6,0xc33707d6,0xf4d50d87,0x455a14ed,

0xa9e3e905,0xfcefa3f8,0x676f02d9,0x8d2a4c8a,

0xfffa3942,0x8771f681,0x6d9d6122,0xfde5380c,

0xa4beea44,0x4bdecfa9,0xf6bb4b60,0xbebfbc70,

0x289b7ec6,0xeaa127fa,0xd4ef3085,0x4881d05,

0xd9d4d039,0xe6db99e5,0x1fa27cf8,0xc4ac5665,

0xf4292244,0x432aff97,0xab9423a7,0xfc93a039,

0x655b59c3,0x8f0ccc92,0xffeff47d,0x85845dd1,

0x6fa87e4f,0xfe2ce6e0,0xa3014314,0x4e0811a1,

0xf7537e82,0xbd3af235,0x2ad7d2bb,0xeb86d391

};

После этого выполняется первый раунд. Он состоит из вызова функции

TransF(ref uint a, uint b, uint c, uint d, uint k, ushort s, uint i)

{

a = b +RotateLeft((a + ((b & c) | (~(b) & d)) + X[k] + T[i - 1]), s);

}

Вызовы первого раунда выглядят так:

TransF(ref A, B, C, D, 0, 7, 1);

TransF(ref D, A, B, C, 1, 12, 2);

TransF(ref C, D, A, B, 2, 17, 3);

TransF(ref B, C, D, A, 3, 22, 4);

TransF(ref A, B, C, D, 4, 7, 5);

TransF(ref D, A, B, C, 5, 12, 6);

TransF(ref C, D, A, B, 6, 17, 7);

TransF(ref B, C, D, A, 7, 22, 8);

TransF(ref A, B, C, D, 8, 7, 9);

TransF(ref D, A, B, C, 9, 12, 10);

TransF(ref C, D, A, B, 10, 17, 11);

TransF(ref B, C, D, A, 11, 22, 12);

TransF(ref A, B, C, D, 12, 7, 13);

TransF(ref D, A, B, C, 13, 12, 14);

TransF(ref C, D, A, B, 14, 17, 15);

TransF(ref B, C, D, A, 15, 22, 16);

При этом, значения регистров с вызовом каждой функции изменяются.

Затем следует второй раунд:

Функция второго рауда:

TransG(ref uint a, uint b, uint c, uint d, uint k, ushort s, uint i)

{

a = b + RotateLeft((a + ((b & d) | (c & ~d)) + X[k] + T[i - 1]), s);

}

Вызовы второго раунда:

TransG(ref A, B, C, D, 1, 5, 17);

TransG(ref D, A, B, C, 6, 9, 18);

TransG(ref C, D, A, B, 11, 14, 19);

TransG(ref B, C, D, A, 0, 20, 20);

TransG(ref A, B, C, D, 5, 5, 21);

TransG(ref D, A, B, C, 10, 9, 22);

TransG(ref C, D, A, B, 15, 14, 23);

TransG(ref B, C, D, A, 4, 20, 24);

TransG(ref A, B, C, D, 9, 5, 25);

TransG(ref D, A, B, C, 14, 9, 26);

TransG(ref C, D, A, B, 3, 14, 27);

TransG(ref B, C, D, A, 8, 20, 28);

TransG(ref A, B, C, D, 13, 5, 29);

TransG(ref D, A, B, C, 2, 9, 30);

TransG(ref C, D, A, B, 7, 14, 31);

TransG(ref B, C, D, A, 12, 20, 32);

Третий раунд:

TransH(ref uint a, uint b, uint c, uint d, uint k, ushort s, uint i)

{

a = b + RotateLeft((a + (b ^ c ^ d) + X[k] + T[i - 1]), s);

}

TransH(ref A, B, C, D, 5, 4, 33);

TransH(ref D, A, B, C, 8, 11, 34);

TransH(ref C, D, A, B, 11, 16, 35);

TransH(ref B, C, D, A, 14, 23, 36);

TransH(ref A, B, C, D, 1, 4, 37);

TransH(ref D, A, B, C, 4, 11, 38);

TransH(ref C, D, A, B, 7, 16, 39);

TransH(ref B, C, D, A, 10, 23, 40);

TransH(ref A, B, C, D, 13, 4, 41);

TransH(ref D, A, B, C, 0, 11, 42);

TransH(ref C, D, A, B, 3, 16, 43);

TransH(ref B, C, D, A, 6, 23, 44);

TransH(ref A, B, C, D, 9, 4, 45);

TransH(ref D, A, B, C, 12, 11, 46);

TransH(ref C, D, A, B, 15, 16, 47);

TransH(ref B, C, D, A, 2, 23, 48);

Четвертый раунд:

TransI(ref uint a, uint b, uint c, uint d, uint k, ushort s, uint i)

{

a = b + RotateLeft((a + (c ^ (b | ~d)) + X[k] + T[i - 1]), s);

}

TransI(ref A, B, C, D, 0, 6, 49);

TransI(ref D, A, B, C, 7, 10, 50);

TransI(ref C, D, A, B, 14, 15, 51);

TransI(ref B, C, D, A, 5, 21, 52);

TransI(ref A, B, C, D, 12, 6, 53);

TransI(ref D, A, B, C, 3, 10, 54);

TransI(ref C, D, A, B, 10, 15, 55);

TransI(ref B, C, D, A, 1, 21, 56);

TransI(ref A, B, C, D, 8, 6, 57);

TransI(ref D, A, B, C, 15, 10, 58);

TransI(ref C, D, A, B, 6, 15, 59);

TransI(ref B, C, D, A, 13, 21, 60);

TransI(ref A, B, C, D, 4, 6, 61);

TransI(ref D, A, B, C, 11, 10, 62);

TransI(ref C, D, A, B, 2, 15, 63);

TransI(ref B, C, D, A, 9, 21, 64);

Затем, после всех раундов, значения промежуточных регистров складываются со значениями:

A = 0x67452301,

B = 0xEFCDAB89,

C = 0x98BADCFE,

D = 0X10325476

А затем байты переставляются в обратном порядке, путем выполнения циклическоо сдвига. Результат шифрования это конкатенация ABCD.

Сервер сравнивает значение хеша с хранимым значением в бд. Если значения совпадают – пользователь ввел верный пароль.

Внедрение SQL-кода

Внедрение SQL-кода — один из распространённых способов взлома сайтов и программ, работающих с базами данных, основанный на внедрении в запрос произвольного SQL-кода.

Внедрение SQL, в зависимости от типа используемой СУБД и условий внедрения, может дать возможность атакующему выполнить произвольный запрос к базе данных (например, прочитать содержимое любых таблиц, удалить, изменить или добавить данные), получить возможность чтения и/или записи локальных файлов и выполнения произвольных команд на атакуемом сервере.

Атака типа внедрения SQL может быть возможна из-за некорректной обработки входных данных, используемых в SQL-запросах.

Разработчик прикладных программ, работающих с базами данных, должен знать о таких уязвимостях и принимать меры противодействия внедрению SQL.

Чтобы исключить внедрение SQL-кода в разрабатываемую cms, на каждом этапе осуществляется проверка корректности данных пользователя, которые он вносит в систему.

Если данные должны быть числом, то они проверяются специальными функциями, которые возвращают положительный или отрицательный результат, в зависимости от того, чем являются данные. Ниже приведен пример кода проверки логина пользователя.

function is\_correct\_login($login)

{;

include\_once("scripts/db/get\_settings.php");

//длина логина

if (strlen($login)<get\_settings("min\_login\_length") || strlen($login)>get\_settings("max\_login\_length"))

{

return false; // Логин должен быть 5-20 символов

}

if (preg\_match("/[^а-яa-z0-9\_]/iu", $login))

{

return false; // недопустимый символ

}

return true;}

# 3. Технологическая часть

## 3.1. Установка и настройка приложения

Приложение поставляется в виде архива формата zip, в котором находятся все необходимые файлы установки. Порядок действий:

1. Распаковать архив setup.zip

2. Переместить папку xxamp в корень диска C

3. Запустить программу C:\xxamp\xampp-control.exe.



Рис. 53. Главное окно xxamp

4. Включить веб-сервер apache, нажав кнопку “Start” напротив надписи “Apache”



Рис. 54. Запуск сервера apache

5. Затем необходимо запустить SQL –сервер.



Рис. 55. Запуск SQL-сервера

6. После этого необходимо в строке браузера прописать путь «http://localhost/phpmyadmin/index.php» и перейти по этому адресу. Откроется окно MyPhpAdmin.



Рис. 56. Вид страницы приветствия программы MyPhpAdmin

7. Нажать на вкладку SQL и прописать туда запрос «create database in40». Выполнить его. В последствии будет создана пустая база данных с назаванием “in40”.

8. Необходимо произвести импорт таблиц. Для этого нажимаем в боковой панели на созданную базу данных, затем вкладку «Импорт»



Рис. 57 Созданная база данных

Затем нажимаем кнопку “choose” для выбора файла. Нужно импортировать файл по адресу C:\xampp\htdocs\in40\!database\in40.sql



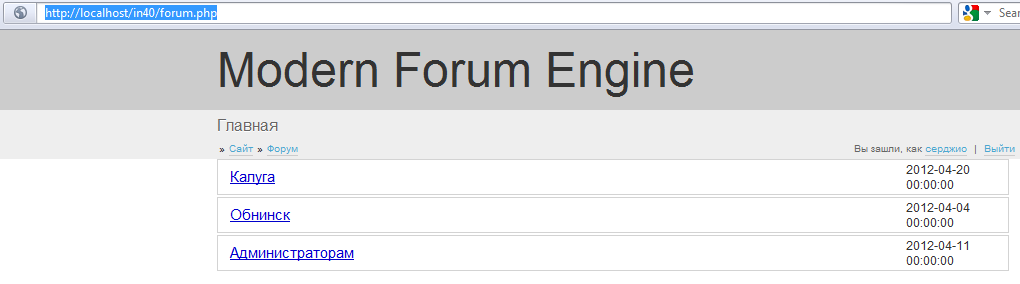
Рис. 58. Окно импорта базы данных.

После успешного импорта появится сообщение:



Рис. 59. Завершение импорта

9. Необходимо набрать в адресной строке браузера «http://localhost/in40/forum.php», после чего появится главная страница форума. Развертывание системы завершено.

Рис. 60. Главная страница форума.

В системе есть файл C:\xampp\htdocs\in40\settings\settings.php, в котором есть настройки:

define("host", "localhost"); (1)

define("user", "root"); (2)

define("pass", ""); (3)

define("site", "http://localhost/in40/"); (4)

define('site\_name', 'Сайт'); (5)

define('current\_theme', 'default') (6)

(1) – имя хоста сервера apache

(2) – имя пользователя в MySQLServer

(3) – пароль

(4) – адрес сайта

(5) – название сайта

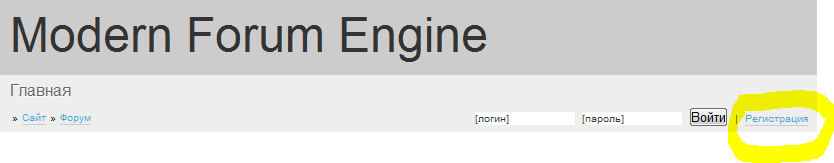
(6) – текущая тема

Все настройки имеют вид «ключ – значение». Чтобы изменить значение настройки, необходимо изменить второй параметр.

## 3.2. Инструкция пользователя

Регистрация

Чтобы зарегистрироваться в системе, необходимо набрать в строке браузера адрес «http://localhost/in40/forum.php» и перейти по нему. Далее нужно перейти по ссылка «регистрация»

 Рис. 61. Ссылка для регистрации

Затем в форму необходимо ввести пользовательские данные: логин, состоящий из букв латинского алфавита и нижнего подчеркивания, электронный ящик и придумать пароль, который необходимо ввести два раза.

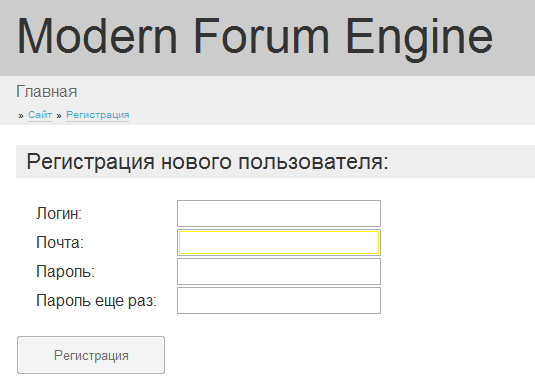
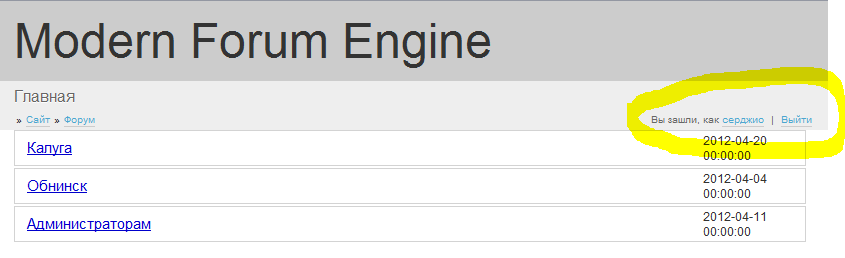


Рис. 62. Форма регистрации

Если все данные корректны, станет активна кнопка регистрации, которую нужно будет впоследствии нажать. Затем, пользователь будет переадресован на главную страницу, где уже будет введен его логин и пароль.

 Рис. 63. Пользователь авторизован

Чтобы учетной записью не смогли воспользоваться посторонние люди, необходимо после завершения работы с системой нажать кнопку «выйти» чтобы удалить свои логин и пароль из cookies браузера.

Авторизация пользователя в системе

Чтобы войти в систему, необходимо ввести логин и пароль в поля ввода и нажать кнопку «войти». Символы пароля будут отображаться точками, чтобы посторонние не смогли увидеть пароль.



Рис. 64. Панель до ввода логина и пароля



Рис. 65. Панель после ввода логина и пароля

Создание, редактирование тем, сообщений

Чтобы создать новую тему, нужно нажать на ссылку «создать тему». Появится форма создания новой темы. Необходимо вписать название и первое сообщение темы, а затем нажать кнопку «создать»

Рис. 66. Форма создания темы

Рис. 67. Заполненная форма

После того, как форма будет заполнена, кнопка «создать тему» станет активной.

Чтобы ответить на сообщение темы, нужно зайти в тему, а потом нажать ссылку «ответить». Появится форма ввода сообщения. После того, как сообщение будет введено форму, кнопка «отправить» станет активной.

Рис. 68. Заполненная форма

## 3.3. Инструкция администратора

Удаление тем

Чтобы удалить тему, нужно зайти в категорию и нажать ссылку «удалить тему».

Рис. 69. Удаление темы.

После удаление темы, текущая страница будет обновлена.

Закрытие темы

Чтобы запретить дальнейшее обсуждение пользователей в теме, необходимо нажать ссылку «закрыть тему».

Рис. 70. Индикация закрытой темы.

Удаление сообщений

Чтобы удалить сообщение, необходимо нажать на ссылку «удалить» рядом с сообщением.



Рис. 71. Удаление сообщения

Редактор категорий

В панели администратора реализован полноценный редактор категорий, который позволяет добавлять, редактировать и удалять необходимые категории. Чтобы открыть редактор категорий, необходимо пройти по ссылке <Название сайта>/admin > Редактор категорий.

Внешний вид и основные элементы редактора указаны на рисунке ниже:



Рис. . Внешний вид и основные элементы редактора категорий

Основные элементы редактора категорий:

1. Удаление категории. При нажатии этой кнопки категория не будет удалена до принятия изменений.

2. Порядок категорий.

3. Отображаемое имя категории.

4. Изменение порядка (2) категорий вверх, либо вниз.

5. Кнопка «Добавить новую категорию» добавляет в конец категорию с пустым именем.

6. Кнопка принятия изменений. При нажатии на эту кнопку, изменения, произведенные в редакторе, будут немедленно приняты и сохранены.



Рис. . Признак успешного сохранения списка категорий

# Заключение

В ходе выполнения работы была поставлена задача реализовать !!! с поддержкой идеологии форумного общения.

Были пройдены следующие этапы:

1. Изучение предметной области. Были рассмотрены основные черты и свойства аналогичных систем, указаны их достоинства и недостатки.

2. Рассмотрены существующие современные технологии, применимые в данной области веб-разработки.

3. Выбран инструментарий, позволивший корректно и максимально быстро выполнить поставленную задачу, с учетом совместной разработки.

4. Были выполнены необходимые этапы проектирования:

- анализ и проектирование архитектуры приложения, что в последствие позволило создать достаточно хорошо структурированное веб-приложение;

- проектирование базы данных, что позволило упорядоченно содержать выводимую информацию;

- проектирование внешнего вида приложения, для повышения удобства пользования приложением конечным пользователем;

- проектирование безопасности приложения, что обеспечило целостность, гибкость использования и конфиденциальность хранимой информации.

5. Было выполнено кодирование всех модулей и связей между ними, разработан интерфейс пользователя.

# Список литературы

1. Announcing the Standard for integration definition for information modeling (IDEF1X), 1993 December 21.

2. HTML 4.01 Specification, W3C Recommendation, 24 December 1999.

3. Htmlbook – справочник по HTML и CSS тегам - <http://htmlbook.ru/>.

4. PHP Manual - <http://php.net/manual>.

5. RSDN Magazine – Российский журнал для программистов - [http://www.rsdn.ru](http://www.rsdn.ru/).

6. Википедия – свободная энциклопедия - [http://ru.wikipedia.org](http://ru.wikipedia.org/).

7. Ларин - Блог о разработке - [http://larin.in](http://larin.in/).

8. Хабрахабр – Российский IT-блог - <http://habrahabr.ru/>.

9. Профессиональное программирование на PHP, Джордж Шлосснейгл, Вильямс, 2006.

10. Справочник по современному JavaScript - <http://javascript.ru/>.