ФЕДЕРАЛЬНОЕ АГЕНТСТВО СВЯЗИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ «САНКТ–ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕЛЕКОММУНИКАЦИЙ им. проф. М.А. БОНЧ–БРУЕВИЧА»

Факультет _ИСиТ_____

Кафедра информационных управляющих систем

РАЗРАБОТКА СИСТЕМЫ УПРАВЛЕНИЯ КОНТЕНТОМ РЕСУРСОВ НАУЧНО-ОБРАЗОВАТЕЛЬНОЙ СРЕДЫ

Санкт–Петербург

СОДЕРЖАНИЕ

ОБОЗНАЧЕНИЯ И СОКРАЩЕНИЯ

API – Application programming interface – интерфейс программирования приложений;

AJAX – Asynchronous JavaScript and XML – асинхронный JavaScript и XML;

ALU – Arithmetic logic unit – Арифметико–логический компонент;

BBcode – Bulletin Board Code – язык разметки, используемый для форматирования сообщений на многих электронных досках объявлений и форумах;

CMS – Content Management System – система управления контентом;

CMA – Content Management Application – приложение управления контентом;

CDA – Content Delivery Application – приложение передачи контента;

CDN – Content Delivery Network или Content Distribution Network – географически распределенная сетевая инфраструктура предоставления доступа к контенту;

CRUD – Create Retrieve Update Delete – создание получение обновление удаление – модель взаимодействия пользователя с СУБД;

CRM – система взаимодействия с клиентом;

CSS – Cascading Style Sheets – каскадные таблицы стилей;

CGI – Common Gateway Interface – межплатформенная низкоуровневая технология создания динамических веб–страниц;

DOM – Document Object Model – объектная модель документа;

DFD – Data Flow Diagrams – диаграммы потоков данных;

DKIM – DomainKeys Identified Mail – метод e–mail аутентификации подписи в электронной почте;

ETL – Extract, Transform, Load – извлечение, преобразование, загрузка;

GPL – GNU General Public License – лицензия свободно распространяемого программного обеспечения;

GNU – GNU Project – организация, идеология свободного программного обеспечения;

HTML – Hyper Text Markup Language – язык гипертекстовой разметки текста;

HTTP – HyperText Transfer Protocol – протокол передачи гипертекста;

HTTPS – HyperText Transfer Protocol Secure – расширенная версия протокола HTTP, поддерживающее шифрование;

IIS – Internet Information Server – проприетарный набор серверов от компании Майкрософт;

JS – JavaScript – объектно–ориентированный, скриптовый язык программирования;

JSON – JavaScript Object Notation – открытый текстовый формат обмена данными, основанный на принципе объектов в JavaScript;

IDEF0 – методология функционального моделирования и графическая нотация, предназначенная для формализации и описания бизнес–процессов;

IDEF1X – методология разработки реляционных баз данных;

LIFO – Last In, First Out – последним пришел, первым ушел – способ организации и манипулирования данными относительно времени и приоритетов;

MVC – Model–view–controller – модель–представление–контроллер;

MIME – Multipurpose Internet Mail Extensions – многоцелевые расширения электронной почты;

NTLM – NT LAN Manager – протокол сетевой аутентификации для Windows NT;

PHP – Hypertext Preprocessor – препроцессор гипертекста;

PDO – PHP Data Objects – расширение для соединения с СУБД, используемое при объектно–ориентированном подходе;

QR – quick response – быстрый отклик – матричный код (двумерный штрих код);

RAID – redundant array of independent disks – избыточный массив независимых дисков;

RFC – Request for Comments – документ содержащий технические спецификации и стандарты, применяемые во W3;

RTP – Real–time Transport Protocol – протокол прикладного уровня, используется при передаче трафика реального времени;

RSS – Rich Site Summary – семейство XML–форматов, предназначенных для составления новостных лент;

SQL – Structured Query Language – язык структурированных запросов;

SEO – search engine optimization – поисковая оптимизация;

SSL – Secure Sockets Layer – уровень защищенных сокетов) – криптографический протокол;

SSH – Secure Shell – сетевой протокол прикладного уровня, для удаленного управление операционной системой и туннелирования;

SMTP – Simple Mail Transfer Protocol – простой протокол передачи почты;

SQRL – Secure Quick Reliable Login – технология безопасного и быстрого осуществления аутентификации используя QR код;

S/MIME – Secure/Multipurpose Internet Mail Extensions – стандарт для шифрования и подписи в электронной почте с помощью открытого ключа;

SASL – Simple Authentication and Security Layer – простой уровень аутентификации и безопасности – технологический метод предоставления аутентификации и защиты данных в протоколах на основе соединений;

TLS – Transport Layer Security – криптографический протокол безопасности транспортного уровня;

TCP – Transmission Control Protocol – основной протокол управления передачей данных в TCP/IP;

URL – Uniform Resource Locator – единообразный локатор электронного ресурса;

VPS – Virtual Private Server – виртуальный частный сервер;

VDS – Virtual Dedicated Server – виртуальный выделенный сервер;

W3 – WWW – World Wide Web – всемирная паутина;

WYSIWYG – What You See Is What You Get – свойство программ или веб–интерфейсов, в которых результатирующее содержание отображается в процессе редактирования;

XML – eXtensible Markup Language – расширяемый язык разметки;

ZIP – формат сжатия данных и архивации файлов;

БД – база данных;

СУБД – система управления базой данных;

ПО – программное обеспечение;

ЗУН – Знания, Умения, Навыки.

ВВЕДЕНИЕ

Не так давно было трудно даже представить степень влияния глобальной паутины на информационное развитие общества. Однако, благодаря ей, современное общество имеет доступ к невообразимым архивам, содержащих данные касающиеся всех аспектов жизни человека. Не малую часть изобилия занимают учебно–методические комплекты разного рода, вида и уровня, что позволяет развивать знания, умения и навыки специалистам любых сфер деятельности, вне зависимости от опыта или возраста.

Структурирование и организация информации – пройденный этап. Для повышения общеобразовательной компетенции необходимо создавать новые, динамичные и автоматизированные решения для совершенствования членов информационного общества, предоставляя необходимые и достаточные инструменты для распространения знаний и развития наук. Образовательная и научная сферы страдают от недостатка внимания и весьма ограниченных ресурсов. Автоматизация образовательных процессов с использованием растущих возможностей глобальной паутины позволит обеспечить новый уровень качества знаний.

1 Анализ современных тенденций совершенствования архитектур систем управления контентом

1.1 Эволюция систем управления контентом

С развитием технологического прогресса в окружении человека появляется все больше и больше платформ информатизации. Значительные потребности профессиональной и социальной деятельности человека осуществляются в процессе взаимодействия с привычными в обиходе устройствами, чьи вычислительные мощности разрастаются в геометрической прогрессии.

Во всех вариантах жизнедеятельности наблюдается потребление возрастающих объёмов информации, представляющих эмоции и знания, а в последствие позволяющих обрести новые умения и навыки.

Информационным ресурсам отводится одна из решающих ролей в жизни, а возможно и в существовании человека. На создание огромных массивов информационных данных тратятся многие годы. Производимая информация и генерируемые знания должны обрабатываться, храниться и распространятся для блага и жизни последующих поколений.

Разумеется, чем больше собирается и создаётся информации, тем больше труда затрачивается на поддержание ее в надлежащем, структурированном и доступном виде. Публикации, библиотеки, архивы, хранилища не справляются с постоянно растущим потоком данных.

В связи с этим сформировались объективные предпосылки появления систем управления контентом CMS (Content Management System). В настоящее время распространяются тысячи CMS решений, направленных на обеспечение тех или иных нужд, связанных с хранением информации. Однако автоматизация процессов обработки получаемой информации находится в непрерывном развитии, так как большинство предлагаемых

средств предназначается для частных решений отдельных, узконаправленных задач.

Специалистами отмечаются территориальные осложнения широко внедрения в эксплуатацию автоматизированных систем управления контентом. В России, в отличие от стран активно продвигающих автоматизацию и машинные средства принятия решений, процесс скорее является преемственным. Крупнейшие компании развиваются, в основном, за счет копирования готовых решений зарубежных коллег.

В настоящее время популярность готовых систем управления контентом, большую часть из которых представляют собой решения с открытым исходным кодом, среди российских компаний и независимых разработчиков разрастается достаточно высокими темпами. Рейтинг рассматриваемых систем, опубликованный порталом «Рейтинг Рунета», представляется на рис. 1.1.1 и 1.1.2 [1, 2].

Во многом подобная ситуация обуславливается повышением требований потребителя. Однако и вопросы с доступностью CMS и их развитием в сторону большей производительности и простоты эксплуатации не оставляются без внимания, ведь именно, благодаря этим качествам, системы управления контентом становятся доступны не только крупным компаниям, но и менее известным брендам с пониженным уровнем капитала, направленным на оснащение и организацию своей работы.

Высокий результат в рейтинге, показанный 1С–Битрикс, не является показателем качества данной отечественной системы управления контентом, поскольку данный эффект достигнут путем монополизации.

Системы управления контентом широко применяются в различных отраслях, так или иначе связанных с информацией и где есть необходимость отделить эксплуатирующий персонал, не компетентный в низкоуровневом использовании вычислительных систем, и базу знаний.

#	CMS	Проектов	Балл	Тренд
1	1С-Битрикс	10 435	62.52	—
2	NetCat	1 929	9.44	—
3	UMI.CMS	3 301	8.44	—
4	HostCMS	1 315	4.55	—
5	CS-Cart	1 199	3.35	new
6	AMIRO.CMS	928	2.41	↘
7	CMS S.Builder	381	1.98	↘
8	ABO.CMS	211	1.42	↘
9	DIAFAN.CMS	342	1.06	—
10	DJEM	102	0.99	↘
11	SiteEdit	458	0.81	↘
12	WebAsyst Shop-Script	209	0.60	↗
13	DataLife Engine	214	0.55	↗
14	PHPShop	124	0.49	↘
15	АТИЛЕКТ	80	0.41	↘
16	CMS BS	72	0.25	↘
17	Astra.CMS	54	0.25	↘
18	F-CMS	39	0.25	↘
19	ImageCMS Shop	107	0.17	new

Рисунок 1.1.1 – Рейтинг коммерческих систем управления контентом

Соответственно основное назначение систем управления контентом заключается в обеспечении многоуровневой абстракции, скрывающей техническую часть диалога между пользователем и машиной, чаще используемой как визуальный, автоматизированный генератор документов,

визуально доступных широкому потребителю. Но наиболее широкое применение CMS наблюдается в секторе услуг – бронирование мест, оформление заказов, организация связи с субъектами и просто поддержка представления субъекта, и, разумеется, предоставление информации о том или ином объекте, его использовании и событиях ему предшествующих.

#	CMS	Проектов	Балл	Тренд
1	Drupal	2 594	36.00	↗
2	Joomla!	3 248	22.56	↘
3	MODX	2 504	21.33	—
4	WordPress	1 761	15.66	—
5	TYPO3	157	2.30	—
6	Opencart	182	0.98	new
7	ImageCMS Corporate Free	102	0.59	↘
8	Concrete5	31	0.35	new
9	PrestaShop	58	0.24	new

Рисунок 1.1.2 – Рейтинг свободно распространяемых систем управления контентом

В одних источниках утверждается, что первая CMS создана компанией IBM в 1963 году и представлена в 1964 как программное обеспечение к IBM System/360, в других указывается на RAINMAN (Remote Automated Information Network Manager), созданную в 1992 году компанией AOL. Основными особенностями обеих CMS считается хранение и отображение данных. В то время не ставились задачи интерактивных преобразований и автоматизированной обработки данных, конференцсвязи и конвертации

файлов через удаленный сервер. По мере развития возможностей информационных технологий и аппаратного обеспечения начался этап роста интерактивности. В системы управления контентом стали добавляться API. Первоочередным параметром CMS, определяющим ее структуру роль в индустрии, становиться функционал, как то интеграция средств работы с удаленными серверами, авторизация, групповая политика, персонализация данных.

Крупнейшим достижением в развитии CMS являются социальные сети. Любой простой «Like» реализуется как запрос, активирующий ряд операций по взаимодействию с СУБД и иными типами хранения данных. Первой социальной сетью считается Classmates.com, в последствие (в 2003) известный как Facebook, созданный в 1995 году выпускником Гарварда, Марком Цукербергом.

Однако самым серьезным прорывом в интерактивности CMS становится появление технологии Ajax. Технология, созданная в 2005 году Джесси Джеймсом Гарреттом, основанная на идеях Google по использованию ActiveX технологий компании Microsoft, появившихся в 1999 году. Именно благодаря этой технологии, веб–разработчикам предоставляется возможность подгружать и обновлять определенные, небольшие части веб–документов, а также делать фоновые запросы и при этом не требовать от пользователя установку отдельного программного обеспечения, как то Adobe Flash Player.

Благодаря Ajax появились самые популярные многофункциональные системы управления контентом. Даже при наличии альтернатив Ajax стал необходимым шагом на пути к интерактивному, многофункциональному инструментарию взаимодействия, созданного специалистами W3c, получившего название HTML5.

Точная дата создания HTML5 не фиксируется. Разработчики, работающие с глобальной паутиной, получали и теряли части инструментария и его стандарта на протяжении нескольких лет. Ещё до

появления HTML5, в конце 90–х, разработчиками, работающими с Adobe Flash, уже использовались анимация, векторные форматы изображений, видео и аудио, производились манипуляции с веб–камерой на стороне клиента и создавались мощнейшие API, эмулирующие программное обеспечение, ранее представляемое как исключительно локальное.

Однако появление HTML5 подтолкнуло разработчиков систем управления контентом на создание не монолитных генераторов HTML–документов, заполненных текстом и статичными изображениями (идеальным прообразом таких систем является MVC, показанный на рис. 1.1.3),

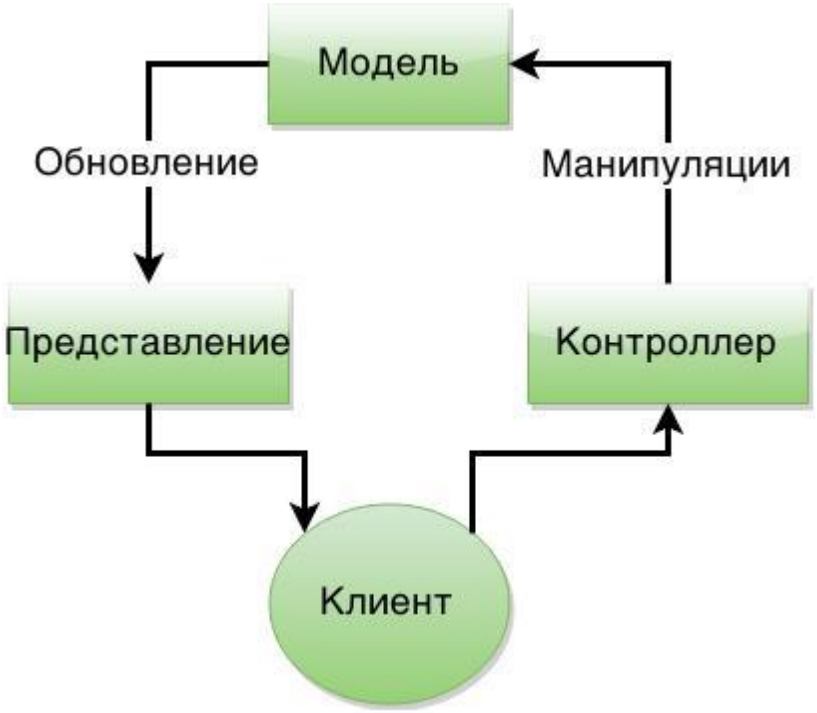

Рисунок 1.1.3 – Концепт модели MVC (Model–View–Controller)

а модульных, расширяемых систем, работающих и взаимодействующих с удаленными системами или данными, а также имеющих библиотеки, поддающихся изменениям вплоть до самого ядра системы управления. Более того, большая часть операций проводится на стороне клиента, что позволяет не только повысить производительность системы и уменьшить нагрузку на аппаратное обеспечение, но обеспечить иной уровень абстракции

функционала и более устойчивую модель обеспечения безопасности, как отдельных персональных данных, так и системы в целом.

1.2 Назначение и виды систем управления контентом

CMS (Content Management System) – это информационная система или программа, используемая для обеспечения и организации совместного процесса создания, редактирования, организации и управления информационным содержимым (то есть контентом) [7, 8]. Управление информационным содержимым может осуществляться либо путем предоставления графического интерфейса как средства проведения манипуляций, либо автоматически.

Появление CMS обуславливается эволюционным разрастанием объёмов данных в информационных инфраструктурах и усилением практической необходимости своевременного предоставления результатов их обработки для сопровождения технических объектов, профессиональной или социальной деятельности. Прочие средства систематизации, хранения и управления базами данных не справляются с перечисленными выше задачами.

Как правило, CMS состоит из двух элементов: приложение управления контентом (CMA – Content Management Application) и приложение передачи контента (CDA – Content Delivery Application) [7]. Соответственно система CMA предназначается для управления операциями над контентом, а системой CDA осуществляется предоставление контента конечному пользователю в необходимом ему виде. Системой CMA предоставляется необходимый инструментарий эксплуатирующей группе, возможно не владеющей техническими навыками и познаниями в языках программирования, для управления процессами заполнения, модификации или удаления контента из информационного пространства используемой среды для выполнения научной, профессиональной, образовательной или социальной деятельности. Системой CDA компилируется и используется

подготовленная системой СМА информация для исполнения задачи, поставленной перед системой пользовательским запросом.

Функционал систем управления контентом, как и отмечалось ранее в параграфе 1.1, варьируется в зависимости от поставленных разработчиком условий и задач, определяемых заказчиком. В связи с этим характер требований к функционалу систем управления контентом не регламентируется определёнными нормативами. Однако большинство веб–ориентированных систем управления контентом специализируется на решении задач публикации, управлении форматами, контроле доступа, индексации и поиске информации.

Веб–ориентированный функционал системы, направленный на осуществление публикации, ориентируется не только на предоставление материала пользователю, но и на обеспечение возможности разработчику создавать шаблоны для эксплуатирующего персонала. Тем самым не только разделяются уровни доступа к системе, но и реализуется необходимый и достаточный интерфейс для осуществления той или иной конкретной задачи, входящей в сферу компетенции работника, с изоляцией от иных функциональных возможностей системы.

При управлении форматами представляется пакет средств, позволяющих осуществление компиляции или конвертации документов в формат, обеспечивающий должное представление востребованного контента. Например, отсканированный документ, состоящий из десятков страниц, может являться пользователю как единичное изображение либо, при запросе всего документа, конвертироваться в Portable Document Format (PDF). Чаще всего, при поддержке веб–ориентации, указанная возможность используется для преобразования отсортированного массива строк из базы данных в Hyper Text Markup Language (HTML).

Благодаря контролю доступа, руководством или администрацией проекта осуществляется выборка пользователей, имеющих возможность на ту или иную операцию, в том числе и сам доступ к организации и контролю

доступа. Базовой имплементацией данного функционала предполагается две группы пользователей: администраторы, управляющие системой и заполняющие ее базу данных, и клиенты, пользователи которым эта информация предоставляется. При ограничении одной группой, считается, что система является ограниченно доступной, однако групповой политики не имеется. В подобных случаях контроль доступа осуществляется средствами входа в систему с использованием имени пользователя и пароля.

Функционал индексации трактуется как автоматизированное структурирование данных для упрощенного и более быстрого доступа к интересующей информации. Традиционно указанная задача выполнялась сервером базы данных, однако с ростом автоматизации обработки и генерации данных стали применяться механизмы сокращения нагрузки на аппаратное обеспечение с целью повышения общей производительности. Наиболее распространённый механизм опирается на кэширование. При востребованности одних и тех же данных делается копия результата запроса, после получения данных из базы, и затем передаётся этот результат без новой сборки остальным пользователям. В этом случае сокращается нагрузка на аппаратное обеспечение и повышается общая производительность.

Другой механизм базируется на разделении контента на смысловые категории, маркировании отдельных материалов ключевыми словами. Благодаря представленным приёмам устанавливается иерархический метод структурирования и обеспечивается реализация более быстрых и качественных средств поиска.

По функционалу поиска системы управления контентом приближаются к CRM–системам (Customer Relationship Management), для которых центр бизнес–проекций сосредотачивается на клиентах.

В связи с этим CMS–система может рассматриваться как простейшая модификация подсистемы CRM–системы, например, в интернет магазинах или сферах предоставления услуг. Наряду с этим, CRM–система также может трактоваться как подсистема для CMS–системы, например, компонент 1С–

Битрикс24, позволяющий проводить интеграцию в проект, работающий на системе управления контентом 1С–Битрикс. Кроме того, все чаще появляются примеры создания средств веб аналитики как части CMS–систем, осуществляющих сбор и структурирование данных о пользователях и их действиях внутри системы.

В системах управления контентом, использующих частичные возможности CRM–систем, предоставляются инструменты для осуществления персонализированного метода маркетинга, называемого «один на один маркетинг». Благодаря указанной способности системы, выполняется преобразование её контента исходя из предпочтений пользователя и его прошлого опыта работы с системой или схожими проектами. Типичным примером использования подобного метода является разделение по интересам. Например, пользователю, указавшему в профиле интерес «технологии», будет отправляться оповещения о новых материалах по электронике вне зависимости от рейтинга данной информации. По принципам выделенного механизма, обычно, проводится подбор рекламных баннеров. При представленном подходе учитываются персональные сведения клиента и история его взаимодействий с информационными ресурсами.

При широком разнообразии подходов и возможных функциональных особенностей, выделяется множество классификаций систем управления контентом с многообразием целей, качеств, глубины архитектуры и перспектив. Одним из признаков классификации CMS–систем является вид лицензии [7, 8]:

1) Открытые. Имеют открытый исходный код, предлагаемый для ознакомления, редактирования. Разрешается создание нового программного продукта на базе данной системы. Типичные представители: Wordpress, Drupal, Umbraco, Joomla;
2) Проприетарные. Системы, являющиеся закрытыми или частично закрытыми для сторонних разработчиков и пользователей.

Обычно платные. Например, 1С–Битрикс, Microsoft SharePoint, UlterSuite, Sapiens ECMP, NetCat.

В качестве признака классификации используется и способ работы с шаблоном:

1) Автономная обработка данных. Предназначены для создания статичных страниц. Например, Octopress;
2) Интерактивная система. Предназначены для создания динамических документов. Например, WooCommerce, TYPO3, Wordpress;
3) Гибридная модель обработки. Сочетают функции и тех и других. Например, Google Blogger.

При классификации учитываются виды функциональных задач:

1) Информационные. CMS–системы общего назначения. Например, MaxSite, 1С–Битрикс, Joomla, Drupal;
2) Блоги. Например, Wordpress, Google Blogger, Octopress;
3) Галереи. Например, Coppermine, Zenphoto;
4) Социальные сети. Например, iCore, LiveStreet, vBulletine;
5) Новостные порталы. Например, DataLife Engine;
6) Форумы. Например, IPB, phpBB;
7) Интернет–магазины. Например, WooCommerce, osCommerce, OpenCart.

Для систем управления контентом, в зависимости от ресурсов и схем ее функциональной реализации, предусматриваются следующие варианты размещения:

1) В защищенном центре хранения и обработки данных;
2) На удаленном сервере, предоставляемом компанией;
3) На сервере компании;
4) На облачном сервере;
5) На кластере;
6) На VPS (Virtual Private Server);

7) На VDS (Virtual Dedicated Server);
8) На NAS сервере.

1.3 Принципы организации систем управления контентом

Исходя из идеологии, сформированной в ходе постановки задач организации систем управления контентом, и идиоматической концепции CRM, первоочередным является осуществление процесса взаимодействия с базой данных. Ввиду этой связи, наиболее распространенный принцип реализации ориентируется на осуществление максимально эффективной ротации и архитектуры организационных моделей, приближенных к тем или иным аспектам жизни человека. К примеру, системы CMS, функционально направленные к модели блога, являются концептуальными последователями журнала. В таком случае записи, представляющие блог, отделяются от записей, представляющих монетизированные отношения со спонсорами. При этом осуществляется хранение и сортировка записей по представляемым категориям или сферам.

В силу набираемых оборотов формирования обширной и открытой базы знаний в W3 (World Wide Web – всемирная паутина), часто используется клиент–серверный принцип взаимодействия [3].

Клиентская часть, как правило, реализуется по модели «тонкий клиент», то есть пользователь, использующий инструменты системы и получающий доступ к информации, содержащейся в файлах на сервере или в базах данных, представляется как субъект, не имеющий должной технической компетенции. Однако с развитием систем управления контентом и ростом потребности в информационной безопасности (в особенности персональной информации), появляется потребность в учёте и противоположной модели, «толстого клиента», в которой пользователь потенциально может быть связан с источником угроз. В подобном контексте проводится разделение пользователей на внешних и внутренних по отношению к проекту и самой системе, а также учитываются и

ограничиваются их функциональные возможности и условия доступности данных.

Для реализации поставленных задач и общей функциональной логики работы системы, а также для проведения манипуляций в файловой системе сервера, используются серверные приложения, чаще создаваемые как «веб сервера». Среди множества подобных приложений выделяются два наиболее популярных: Apache и IIS (Internet Information Server).

Сервер Apache является кроссплатформенным программным обеспечением и имеет открытый исходный код. По общедоступным данным собранным компанией Netcraft, на март 2015 года, свыше 337 млн. сайтов и веб приложений обслуживаются серверами Apache, что составляет 38.39% рынка подобных услуг [6].

Основными преимуществами использования сервера Apache считаются надежность и гибкость конфигурации. Сервером обеспечивается подключение внешних модулей предоставления данных непосредственно во время эксплуатации, а также использование СУБД для аутентификации сервисных пользователей. Имеется ряд инструментов по модификации и организации системного журнала [4].

Ядро веб–сервера Apache, написанное на языке программирования C, организуется согласно модульной архитектуре. При этом поддерживается симметричная мультипроцессорность. Благодаря открытому коду, имеется ряд возможностей как по созданию дополнительных модулей для HTTP/HTTPS–сервера, так и по полной конвертации и распределению внутрисерверных процессов [4].

В отличие от своего всеоткрытого конкурента, сервер IIS является проприетарным набором серверов, а не единым, специализированным программным продуктом. Из–за авторства компании Microsoft, сервер IIS распространяется сугубо с операционными системами семейства Windows, что и является главной проблемой при образовании информационных инфраструктур. [5].

По общедоступным данным, собранным компанией Netcraft, на март 2015 года, более 245 млн. сайтов и веб приложений обслуживаются веб–сервером IIS, что составляет 27.85% рынка подобных услуг [6]. Во многом, такой рост популярности (в сравнении с прошлыми годами), объясняется немалым числом эксплойтов, возникших в 2014 году (самым крупным из которых считается HeartBleed, скомпрометировавший OpenSSL, являющегося компонентом Apache).

Основным компонентом данного набора является веб сервер, позволяющий проводить динамическую обработку данных и размещение в глобальной паутине. Основной особенностью сервера IIS считается архитектура службы WWW, разделяющая и организующая процессы. Например, протокол SSL (Secure Sockets Layer) поддерживается отдельным процессом HTTP SSL, который соединяет протокол TCP и драйвер HTTP.sys. Наряду с этим, отмечается особенно тесное взаимодействие с операционной системой Windows, к примеру функционал обеспечения безопасности в IIS интегрируется с системой в Windows [5].

При реализации корпоративных решений для получения доступа клиента к закрытому или защищенному ресурсу требуется ввести пароль и имя пользователя, существующего в системе Windows, на которой установлен сервер IIS, после чего клиентом продолжается работа с ресурсом тем же образом, как и при выполнении интерактивного входа в систему на сервере. Однако эта особенность, при всем удобстве функционирования, признаётся абсолютно неприемлемой для открытых веб порталов.

В качестве главного функционального недостатка сервера IIS выделяется необходимость поддержки технологии ASP.NET для создания веб–приложений. Даже учитывая тот факт, что сервер IIS поставляется вместе с операционными системами, в базовый пакет которых уже входит .NET Framework, требуемый для работы ASP.NET, и тем самым уже осуществляется встроенная поддержка ASP в наборе IIS, создаются серьезные и лишние нагрузки на аппаратное обеспечение при использовании

альтернативных технологий, в том числе и технологии CGI (Common Gateway Interface), являющейся межплатформенной низкоуровневой технологией создания динамических веб–страниц и отличающейся своей производительностью, за что и получившая признание среди веб разработчиков. Другой недостаток связывается с привязанностью сервера IIS к специфическому протоколу сетевой аутентификации NTML (NT LAN Manager) для работы с электронной почтой, который используется исключительно на операционных системах семейства Windows.

IT–индустрией предоставляется масса вариантов развертывания инфраструктуры систем управления контентом. Одной из самых популярных и доступных форм является аренда удаленного аппаратного обеспечения с сопутствующим пакетом услуг и сервисов, таких как резервное копирование на удаленный сервер, интеграция с социальными сетями, осуществление новостных рассылок и так далее. При использовании данной формы функционирования, большая часть программного обеспечения, требуемого для работы той или иной популярной CMS, часто представляемой на выбор, предоставляется поставщиком, находится на его аппаратном обеспечении и под его ответственностью.

Кроме программного обеспечения поставщиком услуг предоставляется и серверная платформа, сетевая инфраструктура и обеспечивается минимальный уровень надежности проекта. Последнее, чаще всего, характеризуется как вид RAID (Redundant Array of Independent Disks) базового уровня, чем не гарантируется абсолютная защита от потери данных, но предотвращаются негативные последствия при нарушении процесса работы жесткого диска.

Доступ же осуществляется заказчиком посредством удаленного доступа, чаще имеющего web–интерфейс с повышенным уровнем абстракции, не требующим углубленных знаний в архитектуре серверов, в соответствии с чем, эксплуатирующий персонал заказчика не может в полной

мере использовать средства и возможности как при использовании собственного оборудования.

При этом учитывается множество тонких различий. Однако наиболее серьезным фактором выбора хостера является предлагаемый язык программирования, используемый на серверной части проекта. Наиболее популярным, и от того часто представляемым в качестве минимального требования для веб–серверов, является язык PHP.

Как результат огромного количества ограничений, данная форма реализации является наиболее доступной и экономичной. Однако зачастую задачи, предъявляемые системой или проектом, требуют большего, потому и существуют различные варианты размещения, представленные в параграфе 1.2. Наиболее сложным в использовании, но самым просторным на предмет возможностей функционирования, является форма развертывания системы управления контентом с использованием инфраструктуры, проектируемой и создаваемой эксплуатирующим персоналом самой организации. В таком случае командой обеспечивается не только поддержка пользователей и контроль ликвидности состояния представляемой информации, но и осуществляется администрирование серверов, мониторинг процессов амортизации оборудования, а также организуются и проводятся необходимые работы по поддержанию программного и аппаратного обеспечений в должном состоянии в соответствии с обновлениями и новыми разработками и требованиями, предоставляемыми в IT–индустрии на текущий момент.

В общем виде, программная архитектура большинства реализаций систем CMS сводится к модели взаимодействия, показанной на рис. 1.3.1. Пользователь получает доступ к системе посредством использования веб–браузера, поддерживающего диалог по протоколам передачи гипертекстовых сообщений. Изначально, по протоколам HTTP (HyperText Transfer Protocol — протокол передачи гипертекста) или HTTPS (HyperText Transfer Protocol

Secure – расширенная версия протокола HTTP, поддерживающая шифрование) отправляется запрос на сервер.

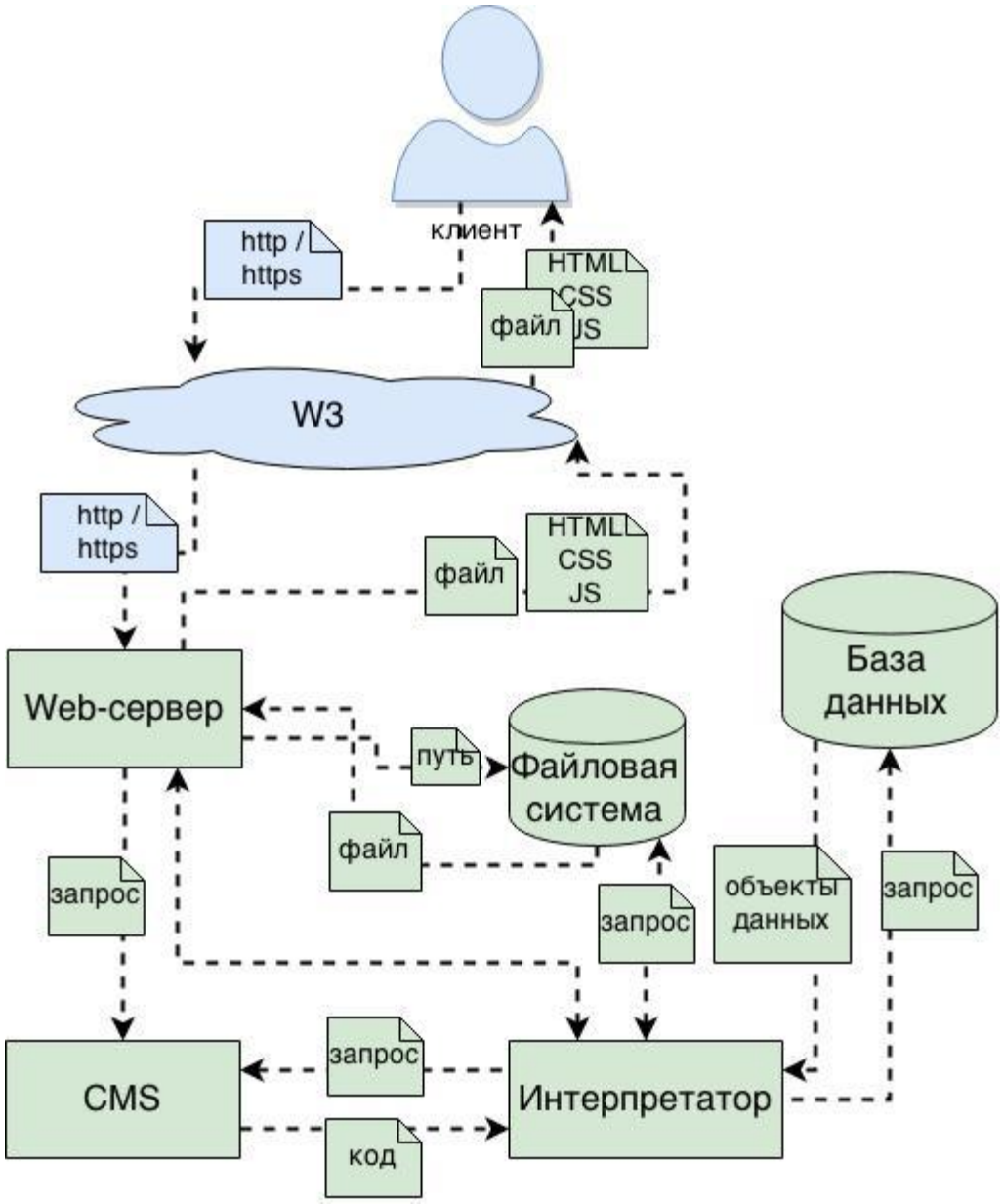

Рисунок 1.3.1 – Базовая модель взаимодействия клиента с веб–сервером, использующим CMS

Запрос проходит через абстрактную модель W3, организующую как маршрутизацию запроса и пакетов, так и определение IP адреса сервера, закрепленного за доменным именем, указанным в запросе. Так запрос доходит до веб–сервера, который, исходя из установленных правил и

конфигурационных данных, создает процессы по формированию запросов и организации исполнения требуемых задач, из чего складывается комплексный запрос к CMS либо запрос на файл в общем доступе.

Получив запрос, система CMS организует двусторонний диалог с интерпретатором, что и позволяет исполнять функциональные задачи на сервере, будь то работа с файлами или базой данных, используя заданные разработчиком компоненты инструментария системы управления контентом.

Основными компонентами базового инструментария, определяющего стартовые задачи и функциональные особенности систем CMS, являются [9, 10, 17]:

1. Средства обеспечения аутентификации пользователей.

Средства аутентификации зачастую сводятся лишь к предоставлению пароля и логина. Однако существует масса вариаций того и другого, к примеру, при новом направлении реализаций предусматривается использование QR (quick response, матричный код) кодов, в частности для технологии аутентификации SQRL (Secure Quick Reliable Login) [11].

2. Организация и поддержка персональных данных пользователей.

Аутентификационные данные, предоставляемые пользователем – лишь вершина результатов сбора данных. Однако, исходя из требуемого минимума, эти данные необходимо держать в защищенном виде и всегда готовыми к подтверждению.

3. Средства ограничения доступа.

Средства аутентификации используются для определения допустимых предоставляемых функциональных задач, а также организации групповой политики, ведущей к запретам и разрешениям конкретизированных запросов.

4. Обеспечение базовых норм информационной безопасности.

Помимо ограничений доступа, необходимо учитывать массу возможных атак злоумышленников, направленных на несанкционированное получение персональных данных либо защищаемого контента–продукции.

5. Организация и соблюдение правил групповой политики.

В качестве средства определения допустимости тех или иных действий пользователя в системе, используются инструменты формирования групповой политики, чаще всего представляемые в виде классификаций, приписываемых нескольким пользователям посредством таблицы пользователей в базе данных.

6. Генерация мета данных о публикациях и медиа контенте.

При получении файлов и текстовых массивов от пользователя необходимо структурировать и организовывать их внутри хранилища данных для обеспечения высокой скорости получения при соответствующем запросе.

7. Средства фильтрации допустимого ввода данных.

Поскольку в задачи системы входит сбор информации, предоставляемой пользователями зачастую не имеющими должного уровня технической компетенции, то необходимо проверять ликвидность вводимых данных, а также производить фильтрацию данных, потенциально опасных для системы или ведущих к утечке информации.

8. Упрощенные средства сбора информации.

Не весь эксплуатирующий персонал и сторонние пользователи имеют необходимые навыки для работы с программным обеспечением, либо отдельные категории персонала не должны иметь к нему непосредственный доступ. По этим причинам система должна иметь необходимый и достаточный веб–интерфейс.

9. Сбор и организованное хранение истории взаимодействия пользователей с системой.

Аналитические данные о просмотре страниц, веб документов, личных страниц пользователей, а также о числе публикуемых материалов – вот минимум требуемый от системы. Однако данные должны быть оптимизированы для экономии пространства и времени, занимаемого на их учет и возможное предоставление.

10. Генерация и предоставление базовых аналитических отчетов.

При осуществлении сбора аналитических данных необходимо организовывать интуитивно понятные формы отчетности об их изменениях с течением времени, чаще представляемые в виде таблиц или графиков.

11. Контроль версий.

Система управления должна обеспечивать мета сопровождение публикуемых материалов, как то дата и время публикации и последнего редактирования, либо, в более детализированных случаях использования и частого редактирования, перечень вводимых изменений по каждой публикуемой версии представляемого материала.

12.Средства обеспечения обратной связи.

Система личных сообщений и комментарии к публикациям позволяют обеспечить должный уровень потока информации в виде персонального мнения пользователей, существующих для формирования и постановки целей и задач в ходе развития проекта, а также выявления возможных проблем в работе системы и проекта в целом.

13. Автоматизированные функции оповещения.

При формировании задач предъявляемых к проекту существует множество возможных осложнений, выявляемых лишь во время непосредственной эксплуатации системы, поэтому существует потребность своевременно уведомлять пользователей о проблемах или новых изменениях, а также сообщать эксплуатирующему персоналу о проблемах и действиях пользователей.

1.4 Перспективные направления развития систем управления контентом

Во временной кривой развития концепции систем управления контентом наблюдаются различные периоды. Системы непрерывно изменяются, так как касаются расширяющегося множества сфер деятельности человека, и обновляются согласно появляющимся технологическим веяниям в жизненном цикле информационных

инфраструктур. В связи с этим актуализируется сфера образования. С расширением технологического пространства больше требуется квалифицированных специалистов.

Компетентностный подход к реализации профессионального образования, будь то повышение качества профессиональной подготовки, обучение молодых специалистов, повышение престижа высококвалифицированного труда работников массовых профессий и специальностей, а также популяризация их достижений и публикация передового опыта неразрывно связывается с использованием IT–технологий (Information technology – информационные технологии) в образовательном процессе. Обучающиеся не только активно пользуются интернет–ресурсами, но и свободно ориентируются в цифровой информационно–коммуникационной среде [16].

Онлайн–технологии, которые в большинстве случаев применяются для дистанционного обучения, становятся понятными и привычными и в аудиторных занятиях. При ориентации на новые цели развития образовательной компетенции требуются изменения не только содержания изучаемых предметов и учебно–методических комплектов для их освоения, но и методов и форм организации образовательного процесса, активизации деятельности обучающихся в ходе занятия, приближения изучаемых тем к реальной жизни и поисков путей решения возникающих проблем обработки и освоения знаний, умений и навыков (ЗУН) [16].

Необходимо отметить, что применение онлайн–технологий в сфере образования – это не просто перенос привычных практик, в виде отсканированных методических пособий и рекомендаций, а так же подобранных на скорую руку контрольно–оценочных средств, отличающихся от бланков тестирования лишь малым набором интерфейсных особенностей кое как связанных друг с другом, но не с тематикой проводимого испытания, но прежде всего онлайн–технологии это новая, еще не раскрытая, модель учебного процесса, требующая немалых усилий со

стороны преподавателей и специалистов технологической сферы деятельности, вместе с полным переосмыслением прошлых методик обучения [20].

Благодаря применению систем управления контентом обеспечивается возможность создания новых интерактивных методов проведения учебных занятий, а также возможность ориентации процесса обучения на более широкое взаимодействие учащихся не только с преподавателем, но и друг с другом, доминирования активности слушателей, получения практического опыта как в виртуализированной информационной среде, так и в более привычных условиях.

В профессиональной деятельности немало специалистов, занимающихся различными наработками и небольшими изменениями в процессе и средствах обучения. Системы управления контентом ресурсов научно–образовательной среды стремятся оградить клиентов от необходимости веб–разработки для установления связи с коллегами и профессионалами из других сфер, а также обеспечиваются мощным инструментарием и возможностями усовершенствования общей структуры образовательных процессов [18, 20].

Обеспечение средств организации информационной безопасности также становится все актуальнее, однако по прежнему отодвигается на второй план и даже накаляющиеся баталии за защиту персональных данных и информационной собственности не приносят видимых результатов.

Как и в большинстве CRM–решений, в CMS бытует устаревшие средства аутентификации пользователей и эксплуатирующих групп, представляемых собой веб интерфейс для ввода имени пользователя и пароля. Горький опыт таких компаний как Yandex и Вконтакте, заставил пойти по организации более дорогого и зарубежного средства, ранее созданного совместными силами Google и Facebook, «двух этапной аутентификации» используя мобильный телефон пользователя. Увы, данное

решение доступно лишь крупным компаниям, да и они позиционируют его как опциональное.

Однако, какие бы современные и высокотехнологичные решения бы не создавались, пользователь слишком ленив чтобы повторять одни и теже процедуры для каждого своего аккаунта в той или иной системе. Для чего существуют методы простого поддержания авторизированных сессий, чаще всего для этой цели используют, генерируемый системой алгоритмами хеширования, электронный ключ, содержащий закодированный сегмент уникального идентификатора пользователя. Клиентское приложение, имеющее данный электронный ключ, при следующем обращении к системе, также хранящей ключ (часто тот же самый), сразу же распознается системой как ее пользователь и получает требуемые привилегии и соответственно организуется сессия взаимодействия между клиентским приложением и информационным порталом, с предполагаемым уровнем закрытости [19].

Минус данного метода в том, что система может «запомнить» лишь одно приложения (один ключ) на пользователя, что с точки зрения информационной безопасности правильно, однако число устройств и приложений электронного доступа растет с каждым днем. На сегодняшний день, более универсальным, простым в использовании, доступным для разработчиков и компаний, а главное действительно надежным способом обеспечения аутентификации может стать упомянутый ранее SQRL, разработанный Стивом Гибсоном, широко известным специалистом по техническим аспектам обеспечения информационной безопасности. Базовая концептуальная модель работы SQRL представлен ниже, на рис. 1.4.1.

Пока что остается следить за развитием данной технологии, так как ее идея была описана в конце 2014 года, однако большая часть документации по–прежнему находится на стадии осмысления, и пока нет примеров ее реализации как и более детализированных аспектов стандарта.

Таким образом, можно утверждать, что системы управления контентом развиваются по направлениям совершенствования информационного,

математического, программного и технологического обеспечения. Детальные особенности этих направлений описываются в последующем параграфе.

Рисунок 1.4.1 – Концептуальная модель базового принципа работы механизма аутентификации SQRL

1.5 Определение основных особенностей лидирующих систем управления контентом

В обширном инфокоммуникационном пространстве веб технологий каждая система управления контентом занимает определённую специализированную нишу. Однако большинство подобных систем особо не выделяются, ввиду закрытости проектов и разного рода корпоративной конфиденциальности. Из известных, наиболее популярными становятся лишь системы, имеющие наиболее обширный набор функциональных возможностей. К ним, например, относятся следующие системы:

1) Wordpress;
2) Drupal;
3) Joomla;
4) 1С–Битрикс.

Система Wordpress является бесспорным лидером большинства. Признание обеспечивается множественными успехами в блогосфере и областях, связанных с открытой журналистикой. Система базируется на открытом исходном коде и распространяется под лицензией GPL (GPLv1 – GNU General Public License) и Apache License, чем обеспечивается

разностороннее развитие, отчасти в виде плагинов. По заявлению компании, 23% веб–сайтов в глобальной паутине создаются на основе системы Wordpress [12].

В системе имеется гибкий визуальный инструментарий по изменению и реконфигурации установленных шаблонов графического представления веб–сайта. Наличие встроенного функционала интеграции с плагином к браузеру для быстрой публикации и скорого написания заметок и статей, признаётся в качестве одного из важных факторов, укрепляющих позиции данной системы управления контентом, как самой удобной для блогеров и журналистов.

Благодаря использованию системы Wordpress, достигается высокая продуктивность и расширение границ свободной журналистики. К тому же, в последние годы наблюдается повышение уровня качества публикаций профессиональной журналистики, в чём прослеживается и заслуга рассматриваемой CMS.

Архитектурное развитие системы Wordpress осуществляется в сторону гибкой и модульной модели функциональности. Во многом благодаря этому, система поддерживается веб–серверами Apache, Nginx и IIS, а также не только привычной СУБД MySQL Server, но и MariaDB [12]. В систему включается обширный интерфейс программирования приложений API (Application programming interface), архитектура которого позволяет создавать разные по функционалу и архитектурным особенностям проекты (базовая концепция показана на рис. 1.5.1). При данном подходе сторонние плагины являются изолированными друг от друга, что улучшает общую картину информационной безопасности системы, но в то же время исключает все возможности взаимодействия с плагинами одного производителя.

При всём многообразии особенностей система не отличается высоким уровнем защиты информации и персональных данных.

Попытки формирования разветвленной групповой политики, закрытого доступа к определенным разделам или защитного уровня во взаимодействии

с базой данных усложняются из–за огромного количества несостыковок в предусмотренной архитектуре и несовершенного кода. Даже при всей открытости система Wordpress не сопровождается должной документацией по работе как на нижнем уровне, так и на уровне профессиональных веб–разработок.

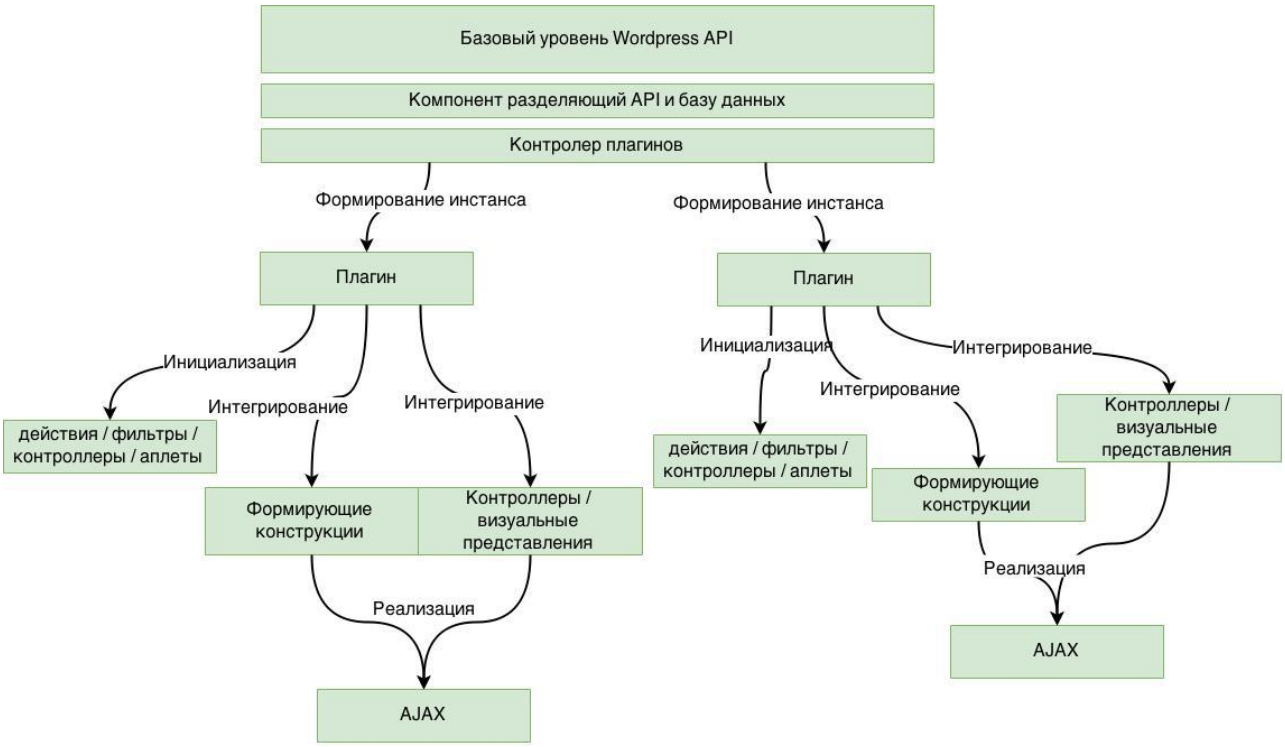

Рисунок 1.5.1 – Схема базового представления архитектуры Wordpress API

В концепции и реализации системы предусматриваются лишь две роли – автор, он же администратор, и незарегистрированные пользователи, у которых есть полный доступ к публикациям. Средства взаимодействия с пользователями, или читателями, ограничиваются комментариями, работающими на стороннем обеспечении Wordpress, как и большая часть API, привязанная к серверам авторов–разработчиков плагинов.

Система Drupal считается системой управления контентом, архитектура и функциональные особенности которой разворачиваются на средствах предоставления больших объемов информации по принципу

новостных порталов. Имеется открытый исходный код и распространяется под лицензией GPL (GPLv1). В систему включается отлаженный и оптимизированный механизм структурирования данных и удобные средства масштабирования. Данной системой пользуются такие ресурсы, как сайт Белого дома США, BBC, NBC и Harvard. Полноценная CMS Drupal появилась в 2005 году на основе интеграции новейших технологий (к примеру, AJAX). Она быстро стала применяться web–мастерами в IT–индустрии [14].

Считается, что в сфере IT–индустрии, система Drupal является наиболее серьезным решением в области обеспечения информационной безопасности. Имеющиеся инструменты организации групповой политики и специализации контента рекомендуются для решения массы возможных проблем. Однако наиболее уникальной и сильной особенностью системы является механизм осуществления поиска, что очевидно и повлияло на успешное распространение CMS. Базовый концепт архитектуры подсистемы поиска представляется на рис. 1.5.2.

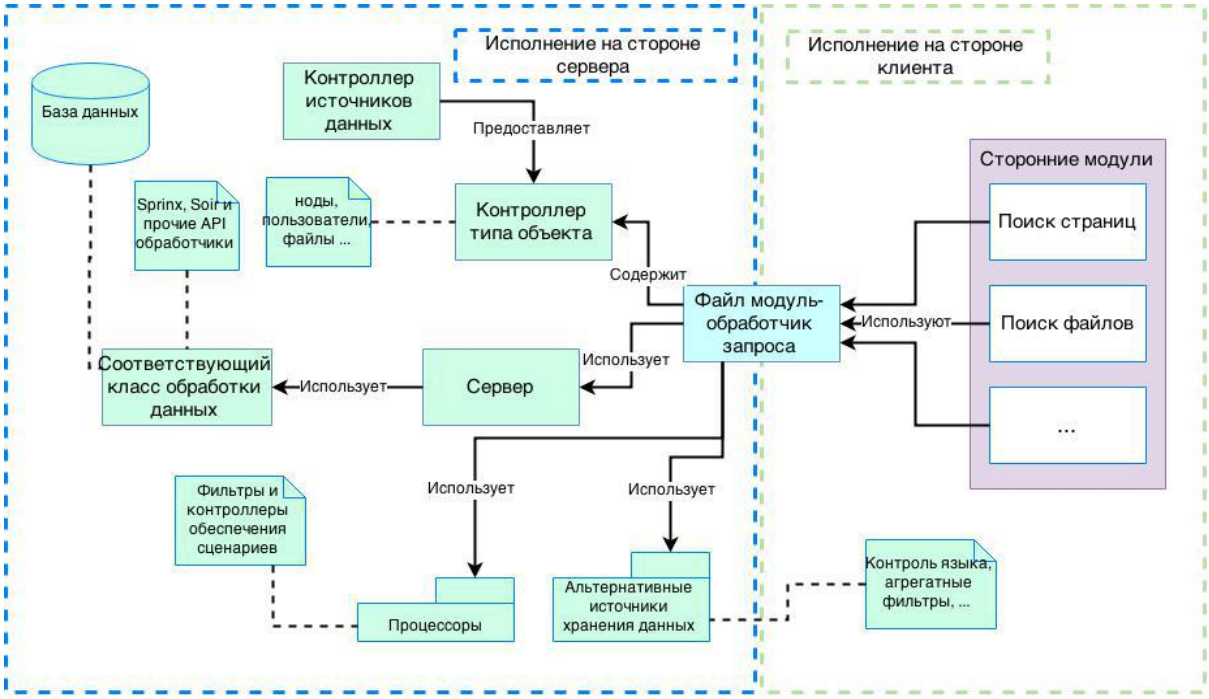

Рисунок 1.5.2 – Базовый концепт архитектурных особенностей подсистемы поиска системы управления контентом Drupal

К немаловажной особенности системы относится поддержка трех СУБД, а именно MySQL, PostgreSQL и SQLite в базовой комплектации системы. В отличие от выше рассмотренной системы Wordpress, подключаются неидентичные СУБД. К примеру, SQLite не является полноценным сервером базы данных. В этой системе осуществляется хранение информации в виде файлов, что подводит к абсолютно иному принципу построения архитектуры данного менеджера баз данных, а значит его поддержка, на фундаментальном уровне, заметно отличается от указанных SQL–схожих СУБД. Что же касается веб–серверов, то в системе используются Apache и IIS серверы.

Система Joomla считается редким примером зарубежной системы управления контентом, поддерживающей кириллицу без интеграции дополнительного программного обеспечения или плагинов. Первая версия данной системы выпускается с 16 сентября 2005 года, и уже к 2008 году стала считаться второй по популярности системой управления контентом, после Wordpress. Распространяется ограниченно бесплатно, используется лицензия GPLv2, то есть базовая комплектация и сторонние дополнения могут приобретаться абсолютно бесплатно. Функционал вроде легкой интеграции с облачными сервисами представляется как CDN (Content Delivery Network – географически распределенная сетевая инфраструктура предоставления доступа к контенту). У бесплатной версии имеется ряд функциональных ограничений, зависящих от используемой комплектации [15].

В базовый пакет дистрибутива не включается ряд интересных особенностей, отличающих Joomla от прочих CMS решений. Поддерживается лишь MySQL сервер работы с базой данных, в качестве веб–сервера могут использоваться Apache или IIS. Используется ряд компонентов, работающих через сторонние HTTP запросы, чем осложняется решение задач информационной безопасности.

Стабильностью работы и огромным количеством дополнительных компонентов объясняется популярность системы Joomla [15]. Особое внимание обращается на расширения поддержки менеджеров баз данных, как то Microsoft SQL Server, SQLite, Oracle Database, так и на CRM решения. Во многом именно малая комплектация базового пакета является основанием для создания дополнительных расширений, не предусматриваемых в Wordpress, или Drupal. Базовый концепт архитектурных особенностей подсистемы организации расширений раскрывается на рис. 1.5.3.

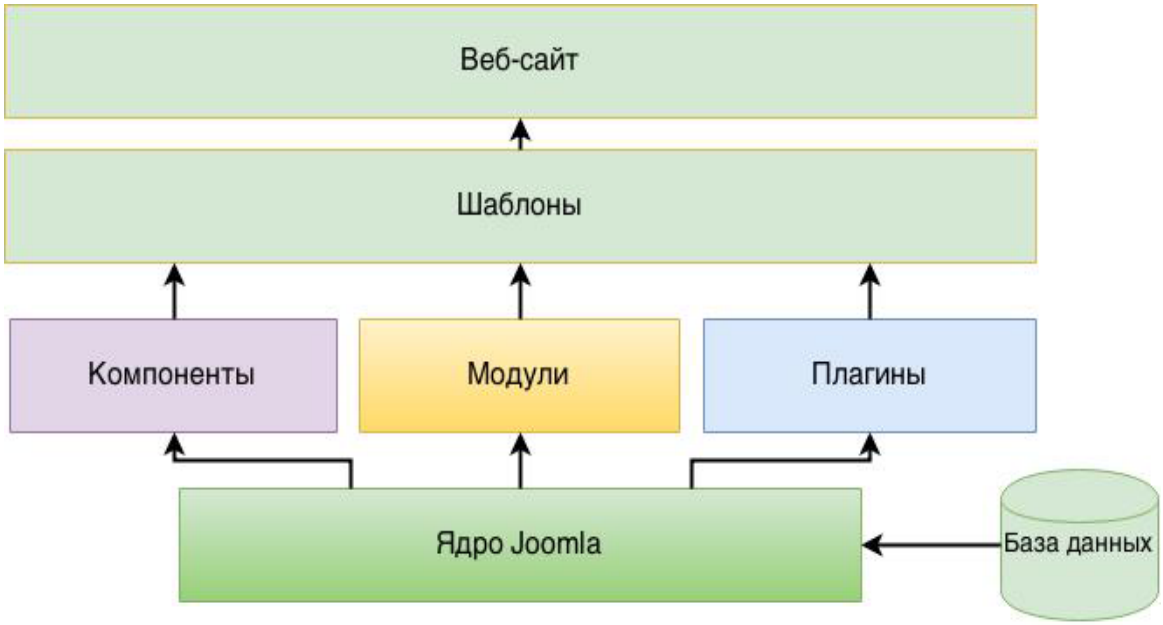

Рисунок 1.5.3 – Базовый концепт архитектурных особенностей подсистемы организации расширений системы управления контентом Joomla

Система 1С–Битрикс представляется как единственный артефакт из списка анализируемых систем, имеющий статус проприетарного программного обеспечения. Испытания системы закончились в 2001 году. Указанная разработка является отечественной системой управления контентом. Однако за пределами страны не пользуется популярностью. К системе 1С–Битрикс относятся по–разному. Многими специалистами считается, что цена себя не оправдывает. Большинство предлагаемых дополнительных пакетов характеризуется необъективной полезностью.

Кроме того, их использование не оправдывается в ресурсном контексте. Система не отличается стабильной работой и высокой производительностью. Минимальные системные требования для базового комплекта определяются в 2гб оперативной памяти при 128мб для Wordpress. Однако, если существует потребность в обширной бухгалтерии и высокодетализированной отчетности, с которыми уже не справляются средства 1С–Бухгалтерии или 1С–Предпирятия (в России, налоговая требует предоставление бухгалтерской отчетности в формате 1С), то система1С–Битрикс становится необходимым инструментарием. При этом учитывается готовая интеграция с бухгалтерией, возможность создания веб интерфейса для отдела кадров, поддержки удаленных взаимодействий, и даже возможность интеграции CRM, но только созданной 1С [13].

Наиболее интересным решением в рассматриваемой системе считается многосайтовость. При этом свойстве под руководством одной системы, с одной панелью управления, курируется сразу несколько информационных порталов. Системой обеспечивается возможность непосредственного использования ядра Битрикс, входящего в базовый пакет программного обеспечения. И, если первое рассматривается лишь как возможность экономии денежных средств на лицензии, то благодаря второму появляется масса интересных возможностей, как то создание собственной CMS с использованием готовых функций, модулей и механизмов. Однако это ядро является закрытым, как и вся система, таким образом, разобраться в коде и построить что–то отличное от комплекса веб–интерфейсов взаимодействия с 1С–Бухгалтерией не получится, да и на рынке есть более интересные, бесплатные и открытые фреймворки с куда более стабильной реализацией и интересными возможностями [13].

У платных систем управления контентом поддерживается действующая служба технической поддержки, которая помогает осуществить комплекс операций и найти решение для любой, даже неординарной проблемы. Вставая на путь использования бесплатного программного

обеспечения, придется брать во внимание факт того, что искать ответы на возникающие вопросы придется на форумах и разнообразных порталах, где советы дают не только опытные и профессиональные пользователи, но и такие же новички, что часто приводит к заблуждениям и как следствие – проблемам [17]. Однако благодаря открытому коду удаётся изменить всю структуру системы так, как никогда не позволит сделать даже самый детализированный интерфейс проприетарных систем.

При выборе платных или бесплатных CMS учитывается то, что их разработка, обычно, предполагает реализацию более конкретизированных задач, что позволяет подобрать наиболее рациональное решение и тратить средства на обеспечение, в том числе и аппаратное, в котором не нуждается предполагаемый проект. В свою очередь, 1С–Битрикс нацеливается на все и сразу.

В отличие от вышеупомянутых систем, в данной системе имеется крайне скудная документация, что делает невозможным описание архитектурных особенностей того или иного решения, а также выявление отличительных качеств в работе модулей системы. Однако, исходя из манеры взаимодействия с панелью управления и непосредственным интерфейсом пользователей, специалистами делается заключение, что 1С–Битрикс использует упоминавшуюся ранее модель MVC (рис. 1.1.3), от которой уже отказались прочие популярные CMS–решения, в том числе и упомянутые выше.

Всеми ведущими разработчиками систем управления контентом ставится первоочередные задачи по стиранию разделяющих границ концептуальной модели MVC и развитию асинхронного ведения диалога системы с клиентом (приложением), а также по установлению приемлемой планки организации информационной безопасности, которую, в последние годы не удаётся поддерживать на должном уровне.

Наряду с этим, развивается методология единой архитектуры ведения данного диалога, предполагающая взаимодействие, как с рабочими

станциями, так и с мобильными устройствами, в одной информационно–программной среде, то есть без переадресаций и функциональных ограничений.

Начало подобного направления связывается с появлением концепции отзывчивого дизайна, входящего в концептуальную группу «Responsive Design» (адаптивный дизайн) и являющегося потомком адаптивного дизайна. Ранее, адаптивный дизайн, представлялся как средство адаптации интерфейса к клиентскому обозревателю, как то ширина страницы или размер шрифта, ширина таблиц или размер изображений.

Однако теперь отзывчивый дизайн, представляется как комплекс визуальных мер, принимаемых при изменениях клиентского обозревателя. Таким образом, даже после загрузки, при уменьшении или увеличении ширины окна обозревателя скрывают или показываются элементы дизайна, меняются и структурируются по новому. К примеру, при уменьшении большой ширины окна, детализированное горизонтальное меню меняется до вертикального списка, а далее и до выпадающего списка для мобильного телефона.

Список подобных электронных ресурсов ещё не столь велик, но с каждым днем появляются новые шаблоны оформления для популярных CMS–решений, показывающие подобную реализацию интерфейса. Следовательно, намечается и тенденция стремлений к адаптационному функционалу и подобной архитектуре.

1.6 Постановка задач разработки

В большинстве известных современных технических реализаций, связанных с обеспечением базового функционала и архитектуры систем управления контентом, прослеживается четкая направленность на воплощение ряда оригинальных решений, позволяющих создать отличительные от конкурентных решений средства взаимодействия. В связи с этим разрабатываемый прототип системы управления контентом ресурсов

научно–образовательной среды должен не только соответствовать общепринятым качественным характеристикам, но и предоставлять необходимый функционал, отвечающий современным требованиям, предъявляемым к системам управления контентом, и быть конкурентоспособным в выбранном классе реализаций.

Для достижения поставленной цели необходимо решить следующие задачи:

– сформировать требования к прототипу системы управления контентом ресурсов научно–образовательной среды;

– выполнить концептуальное моделирование базового функционала прототипа системы управления контентом ресурсов научно–образовательной среды;

– построить алгоритмы работы базовых подсистем прототипа системы управления контентом ресурсов научно–образовательной среды;

– разработать основные компоненты пользовательского интерфейса для прототипа системы управления контентом ресурсов научно–образовательной среды;

– проанализировать и выбрать модель подсистемы виртуализации;

– спроектировать прототип базы данных;

– разработать базовые составляющие серверной и клиентской частей прототипа системы управления контентом ресурсов научно–образовательной среды;

– проанализировать состоятельность выбранных механизмов обеспечения информационной безопасности прототипа;

– выполнить проверку корректности функционирования прототипа системы управления контентом ресурсов научно–образовательной среды.

2 Концептуальное моделирование базового функционала прототипа системы управления контентом ресурсов научно–образовательной среды

2.1 Формирование требований к прототипу системы управления контентом научно–образовательной среды

Требования к прототипу системы управления контентом научно–образовательной среды формируются в результате анализа эволюции систем управления контентом, их назначения и классификаций, принципов организации функциональных спецификаций, перспективных направлений развития и основных особенностей лидирующих реализаций. Сформированные требования представляются в табл. 2.1.1, где для поддержки сравнительного контекста приводятся и функциональные возможности известных систем управления контентом.

Таблица 2.1.1

Сравнение требований к прототипу и основных функциональных возможностей систем управления контентом

	Требования к прототипу	Функциональные возможности известных систем управления контентом			
	Разрабатывае–мый прототип системы	Система Wordpress	Система Drupal	Система Joomla	Система 1С–Битрикс
Open Source	Да	GPLv1 и Apache License	GPLv1	GPLv2	Нет
Распростра–няется	Не распространяет–ся	Бесплатно	Бесплатно	Ограни–ченно бесплат–но	Платно
Возможность смены СУБД во время эксплуатации	Да	Нет	Нет	Нет	Нет

Продолжение табл. 2.1.1

	Требования к прототипу	Функциональные возможности известных систем управления контентом			
	Разрабатывае–мый прототип системы	Система Wordpress	Система Drupal	Система Joomla	Система 1С–Битрикс
Поддержи–ваемые СУБД	MySQL, PostgreSQL, Ms SQL Server, SQLite, Oracle DB	MySQL или MariaDB	MySQL или PostgreSQL или SQLite	MySQL	MySQL или Ms SQL Server или Oracle DB
Максималь–ное число поддержи–ваемых СУБД в одном проекте	Не ограничено	1	1	1	1
Поддержка кириллицы	Да	Плагины	Плагины	Да	Да
Поддержка плагинов	Нет	Да	Да	Да	Нет
Автоматичес–кое обновление	Нет	Да, плагины отдельно	Да, плагины отдельно	Да, плагины отдельно	Да, платно
Поддержи–ваемые веб–серверы	Apache	Apache, Nginx, IIS	Apache, IIS	Apache, IIS	Apache, IIS
Используе-мый язык программи-рования	PHP	PHP	PHP	PHP	PHP
Модель защиты персональ–ных данных	Динамический и уникальный вектор–ключ для каждого пользователя	3 статичны х мастер–ключа до 16 символов каждый	1 статичный мастер–ключ до 64 символов	1 статичны й мастер–ключ до 64 символов	Двухэтапная авторизация по однора–зовому паролю через email

Продолжение табл. 2.1.1

	Требования к прототипу	Функциональные возможности известных систем управления контентом			
	Разрабатывае–мый прототип системы	Система Wordpress	Система Drupal	Система Joomla	Система 1С–Битрикс
Возможность изменения алгоритмов хеширования	Да	Да	Нет	Нет	Нет
Возможность изменения алгоритмов хеширования во время эксплуатации без конвертации	Да	Нет	Нет	Нет	Нет
Возможность восстановле–ния пароля пользователя	Нет	Да	Да	Да	Да
Модель взаимодейст–вия с СУБД	Абстрактная, разделенная модель с модульным пакетом взаимодействи я	Объектно – ориенти–рованный подход	Подгото–вленные запросы[2]	Подгото–вленные запросы[2]	
3rd party CDN	Нет	Google IE HTML5	Google IE HTML5	Google IE HTML5	Нет
3rd party API	Нет	В плагинах	В плагинах	В плагинах	
Представляе–мые JS инструмента–рии	Собственный JS Framework	jQuery, jQueryUI, jQuery Mobile	jQuery, jQueryUI, Lightbox	jQuery, jQueryUI, Lightbox	jQuery, jQueryUI

Продолжение табл. 2.1.1

	Требования к прототипу	Функциональные возможности известных систем управления контентом			
	Разрабатывае–мый прототип системы	Система Wordpress	Система Drupal	Система Joomla	Система 1С–Битрикс
Конфликты с JS инструмента–риями	Нет	Dojo Toolkit	Dojo Toolkit, ThreeJS	Dojo Toolkit, ThreeJS	Dojo Toolkit
Привязки к API производите–ля CMS	Нет	Да	Да	Да	Да
Внешние HTTP	Нет	Обязатель ны для работы системы		Обязатель ны для работы системы	
Возможность интеграции API социальных сетей	Да, собственный, асинхронный toolbar	Плагины	Плагины или AddThis или Qip toolbar’ы	Плагины или AddThis	Лишь самостоя–тельная интеграция
Инструмента –рии для аналитики	Да	Плагины	Плагины	Плагины	Интеграция с 1С CRM
Подсистема работы с SMTP	Да	Плагины	Плагины	Плагины	Да
Поддержка NTLM	Да	Плагины	Плагины	Плагины	Нет
Поддержка SASL	Да	Плагины	Плагины	Плагины	Дополни–тельные модули
Поддержка HTTPS	HTTPS ready	HTTPS ready	Плагины	HTTPS ready	Нет

Продолжение табл. 2.1.1

	Требования к прототипу	Функциональные возможности известных систем управления контентом			
	Разрабатывае–мый прототип системы	Система Wordpress	Система Drupal	Система Joomla	Система 1С–Битрикс
Возможность оповещения по электронной почте без SMTP сервера	Да	Плагины	Плагины	Плагины	Нет
ЧПУ	Да	Плагины	Плагины	Плагины	Нет[1]
Целевая платформа	Научно–образовательные ресурсы	Блог, портфо–лио	Информационный портал	Информационный портал	Интернет–магазин

Метка 1 в табл. 2.1.1 связана со ссылкой на официальный сайт:

http://www.1c–bitrix.ru/bitrix/rk.php?id=1826&event1=banner&event2=click&event3=1+%2F+%5B1826%5D+%5BHOME_PAGE_TOP_BANNER_NEW%5D+%CF%F0%E5%E7%E5%ED%F2%E0%F6%E8%FF+%C124&goto=http%3A%2F%2Fwww.bitrix24.ru%2Fevents%2Fbroadcast_081014.php

Метка 2 представляет тот факт, что непредусмотренные CMS задачи могут быть невозможны для реализации.

Разработанная система требований детально описывается в техническом задании, представленном в приложении 1.

В качестве языка программирования, используемого системой на веб–сервере, был выбран PHP, поскольку данный язык является более подходящим для реализации проекта в соответствии с предъявляемыми требованиями. Сравнительные характеристики популярных языков программирования, чаще всего используемых на веб–серверах, представлены ниже в табл. 2.1.2. Соотносительный расчет баллов по уровню безопасного взаимодействия и готовности языка к W3, наиболее важных для

разрабатываемых решений, представлен в табл. 2.1.3. Сравнение языков проводится в их базовой комплектации, то есть без использования сторонних модулей и расширений.

Таблица 2.1.2

Сравнение качественных характеристик языков в аспектах систем управления контентом

	PHP	Ruby	Python	C#	Java
Уровень взаимодействия с Apache при W3 (Максимальный – «из коробки»)	Максимальный	Максимальный	Максимальный	Требует массу сторонних компонентов	Требует Tom Cat
Степень качества взаимодействия с СУБД (соотносительно)	Высокая[1]	Средняя	Средняя	Средняя	Средняя
Активность развития	Высокая	Высокая	Высокая	Развиваются только компоненты и framework'и	Высокая, при каждом обновлении эксплойтов приходит больше, чем исправлений
Модульность	Частичная	Полная	Частичная	Хаотичная (большинство модулей конфликтует)	Частичная
Степень открытости сообщества (соотносительно)	Средняя	Высокая	Высокая	Слабая	Средняя

Продолжение табл. 2.1.2

	PHP	Ruby	Python	C#	Java
Контроль крупными корпорациями и зависимость от них	Нет	Нет	Нет	Microsoft (полная зависимость)	Oracle (высокий контроль)
Основной сегмент и направление работы	Автоматиза –ция работы СУБД	Многопоточ– ные API	API	Разработан в компании Microsoft как язык разработки приложений для платформы Ms .NET	Вездесущ– ность[2]

Метка 1 в табл. 2.1.2 связана с тем, что самой причиной создания, как и главной задачей языка PHP, является работа со строками.

Метка 2 в табл. 2.1.2 связана с фактом того, что изначально язык Java разрабатывался как язык для широкого применения. Привязка к байт–коду, обрабатываемому виртуальной машиной, рассматривалась как неоспоримое преимущество. Однако в настоящее время IT–специалистами отмечается уязвимость самого языка Java и его компонентов по части информационной безопасности.

Таблица 2.1.3

Соотносительный расчет баллов по уровню безопасного взаимодействия и готовности языка к W3

	PHP	Ruby	Python	C#	Java
Уровень открытости компонентов	5	5	5	0	3

Продолжение табл. 2.1.3

	PHP	Ruby	Python	C#	Java
Требования к пользователю (установленные компоненты – 0)	5	5	5	3	0
Зависимость компонентов языка от операционных систем (5 – не зависит)	5	5	5	0	5
Серверная кроссплат–форменность (5 – да)	5	5	5	0	5
Технологии связи с СУБД (в скобках указаны рабочие варианты)	5 (OOP, PDO)	5 (OOP, Rails, gems и родное соединение с hashing'ом)	4 (OOP)	4 (Silverlight, .Net)	4 (OOP)
Обширный набор компонентов	3	5	4	5	3
Результаты:	28/30	30/30	28/30	12/30	20/30

2.2 Определение состава модулей прототипа

В результате анализа современных тенденций совершенствования архитектур систем управления контентом выбирается состав модулей прототипа системы управления контентом ресурсов научно–образовательной среды, представленный в табл. 2.2.1.

Таблица 2.2.1

Состав модулей прототипа

	Разрабатывае–мый прототип системы	Система Wordpress	Система Drupal	Система Joomla	Система 1С–Битрикс
Новости	Да	Да	Да	Да	Да

Продолжение табл. 2.2.1

	Разрабатывае–мый прототип системы	Система Wordpress	Система Drupal	Система Joomla	Система 1С–Битрикс
Профиль пользователя (персональная страница)	Да	Только автор	Да	Да	Да
RSS генератор	Да	Да	Да	Да	Да
Форма обратной связи	Да	Да	Да	Да	Да
Страницы	Да	Да	Да	Да	Да
Комментарии	Да	Внешние	Да	Да	Да
Ротатор баннеров	Нет	Нет	Нет	Да	Нет
Генерация меню	Нет	Да	Да	Да	Да
Генерация форм	Нет	Нет	Да	Нет	Нет
Генератор опросов	Да	Нет	Нет	Да	Нет
Подсистема работы с электронной почтой	Да	Нет	Нет	Нет	Нет

Сравнительная технологическая среда выдерживается в пределах конкретизированной сферы реализации, определяемой как заказчиком, так и самим производителем в течение некоторого интервала времени направленного на формирование технического задания, представленного в приложении 1.

2.3 Разработка концептуальных моделей подсистем прототипа

Архитектура всей системы и подсистем сопровождается базовым комплексом мер и функциональных особенностей, так или иначе связанных с обеспечением информационной безопасности. В связи с этим разработка моделей начинается с выделения основных уязвимостей систем управления контентом. Предлагаемая базовая концептуальная модель возможных уязвимостей CMS систем представляется на рис. 2.3.1. Расширенная концептуальная модель с приведенными мерами их устранения и более детальным описанием приводится на рис. 2.3.2. Красным треугольником в расширенной концептуальной модели обозначается потенциально опасный сегмент, который не разрабатывается применительно к создаваемой системе управления контентом ресурсов научно–образовательной среды ввиду жёстких временных ограничений на длительность выполнения квалификационной работы. Синим треугольником указывается то, что решение широко известно и его необходимо вводить со стороны аппаратного обеспечения либо операционной системы, что не распространяется на компетенцию CMS.

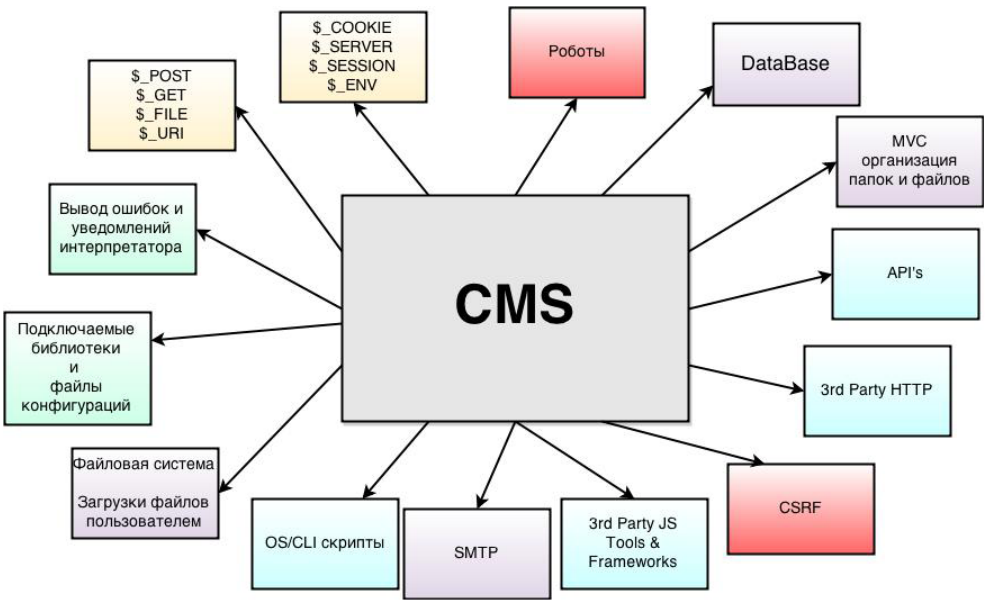

Рисунок 2.3.1 – Базовая концептуальная модель возможных уязвимостей системы управления контентом

Как было написано выше, в параграфе 1.3, одной из основных и определяющих функциональных особенностей систем управления контентом, являются средства обеспечения аутентификации пользователей в данной системе. В разработке концептуальной модели аутентификации, представленной на рис. 2.3.3, учитывается ряд исторических примеров профессиональной несостоятельности решений крупных компаний, связанных с реализацией и хранением персональных данных пользователей. В соответствии с характером поставленной задачи, концепцией сохраняется преемственность по отношению к механизмам хеширования и использованию эллиптических поверхностей для генерации векторов. Благодаря этому, обеспечивается создание системы проверки пароля пользователя, в которой участвуют не только средства фильтрации данных и алгоритмы конвертирования, но и не отслеживаемый пароль, генерируемый самой системой. Указанная особенность является инновацией по отношению к большинству сторонних схожих случаев, где данный пароль фиксируется.

До недавнего времени не отслеживаемый вектор считался бесполезным. Однако в 2014 году мнение изменяется на альтернативное после выявления и исправления уязвимости в механизме OpenSSL, известной как HeartBleed и представляющей собой возможность удалённой фиксированной случайной выборки данных из оперативной памяти сервера.

В качестве главного плюса данной модели считается отсутствие механизмов хранения пользовательского пароля в каком–либо виде. В этом случае реализация функционала «напомнить пароль» становится не возможной. Однако упраздняется ответственность за его хранение, и что самое важное, исключается какая–либо возможность утечки этого пароля.

В концептуальном плане предусматривается возможность усложнения алгоритмов. Ввиду своих особенностей, алгоритм конкатенации «соли» и пароля пользователя, может меняться. В предлагаемом решении не требуется полный перебор данных.

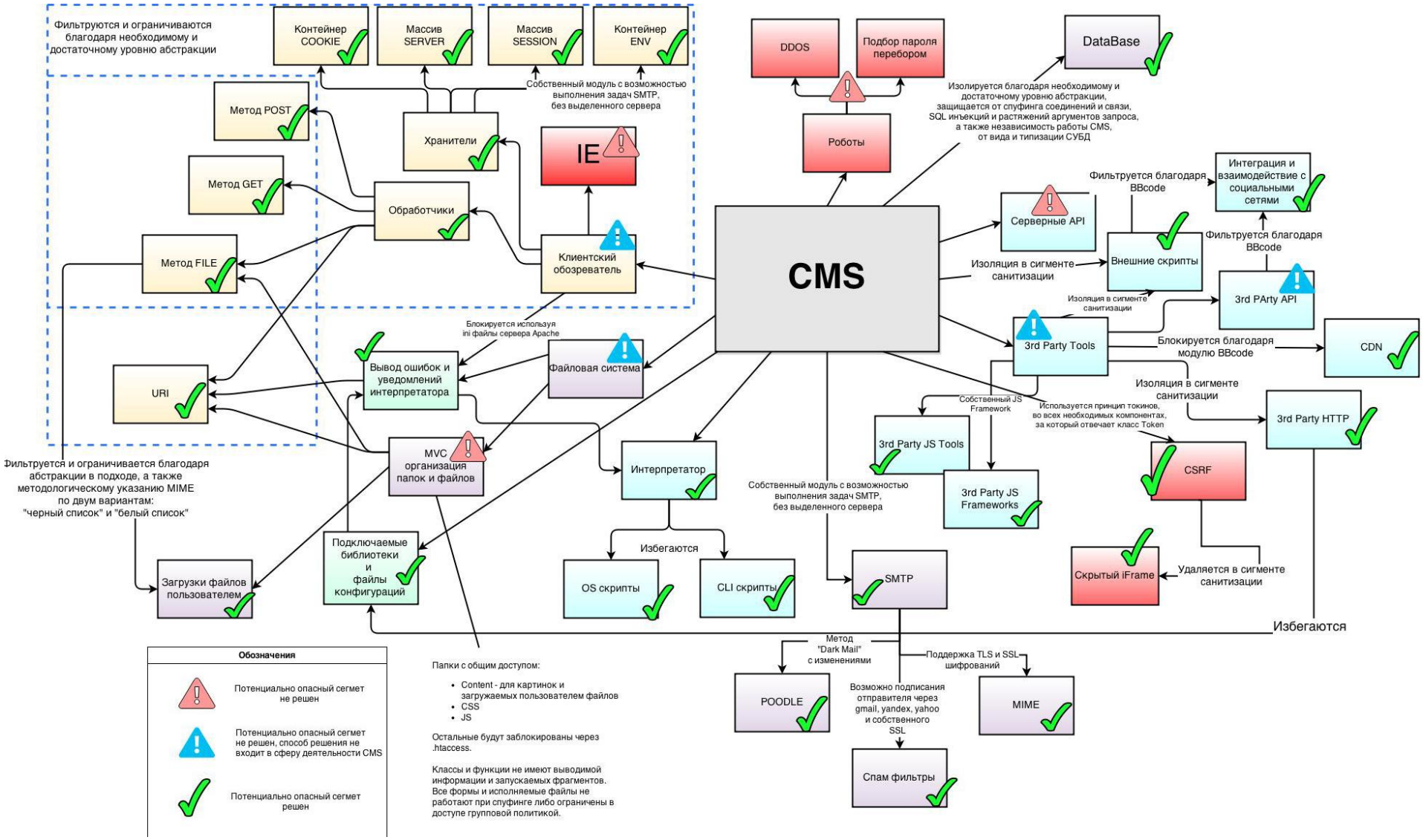

Рисунок 2.3.2 – Расширенная концептуальная модель с приведенными мерами устранения уязвимостей системы управления контентом

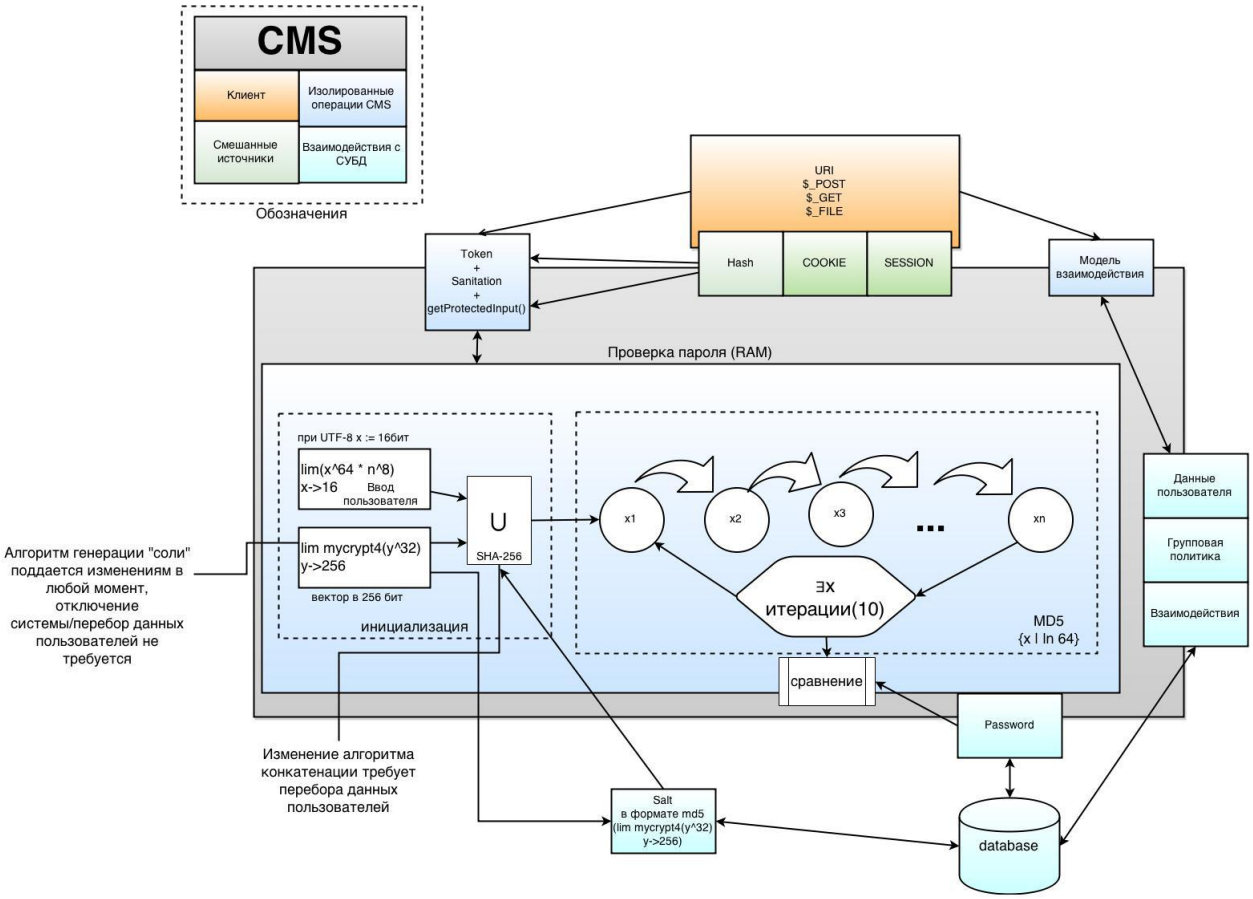

Рисунок 2.3.3 – Концептуальная модель аутентификации в системе

Алгоритм генерации «соли» может изменяться даже при непосредственной работе системы. Для поддержки изменений генератор вектора размещается отдельно. При подобном приёме механизм может усложняться по мере появления новых технологий или требований, либо регулярно, что позволит обеспечить пользователей абсолютно разными паролями вплоть до механизмов кэширования, поскольку сгенерированный хеш пароля в базе данных, заменяется лишь при смене пароля пользователем.

Что касается использования несостоятельного в плане информационной безопасности алгоритма хеширования MD5, то в данной модели он используется лишь как быстрое и простое средство преобразования результата конкатенации в вид и тип данных, который можно записать в базу данных.

Для реализации средств защиты от подмены пользовательских адресов в запросе предусматривается прием генерации токенов. Механизмы организации информационной безопасности базы данных рассматриваются далее. Представленный модуль «санитизации» трактуется как механизм фильтрации ввода, реализуемый в виде алгоритма, работающего со строками Этот модуль описывается в параграфе 2.4.

После получения доступа к данным пользователь пользуется визуализацией данных. За всю работу с базой данных в системе отвечает семейство модулей, собранных в одной подсистеме работы с базой данных и созданных как для повышения абстракции работы с базой данных, так и для реализации средств защиты, ограничивающих как пользователей так и изолирующих сами модули друг от друга и системы в целом. Концептуальная модель процесса функционирования модулей базы данных раскрывается на рис. 2.3.4. Вводимые обозначения поясняются на рис. 2.3.5.

Данная подсистема не только обладает серьезными показателями защищенности от спуфинга соединения и связи, SQL инъекций и растяжений аргументов запроса, но также, благодаря необходимым и достаточным уровням абстракции, которых в подсистеме шесть, позволяет создавать запросы общего синтаксического уровня, а также использовать сколь угодное количество СУБД, автоматически получая доступ к общепринятым преимуществам реализаций. К тому же, данная подсистема поддерживает работу с любыми СУБД, имеющими SQL–Like синтаксис. Однако данная особенность ограничивается возможностями Apache сервера по установлению соединений с базами данных. В связи с этим СУБД, требующие дополнительных и сторонних модулей в данной реализации не рассматриваются. В качестве механизма осуществления соединения выбирается PDO (PHP Data Objects – расширение для соединения с СУБД, используемое при объектно–ориентированном подходе).

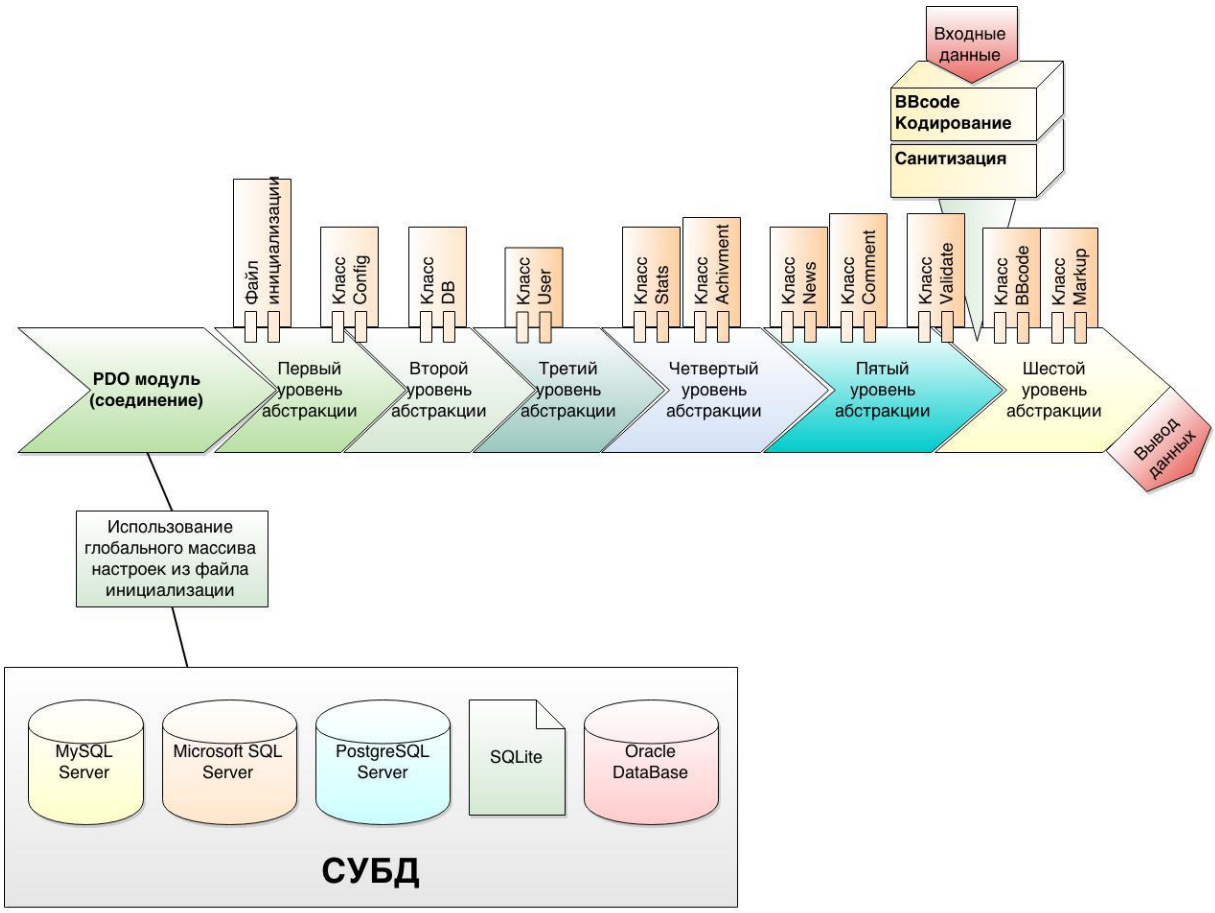

Рисунок 2.3.4 – Концептуальная модель работы подсистемы базы данных

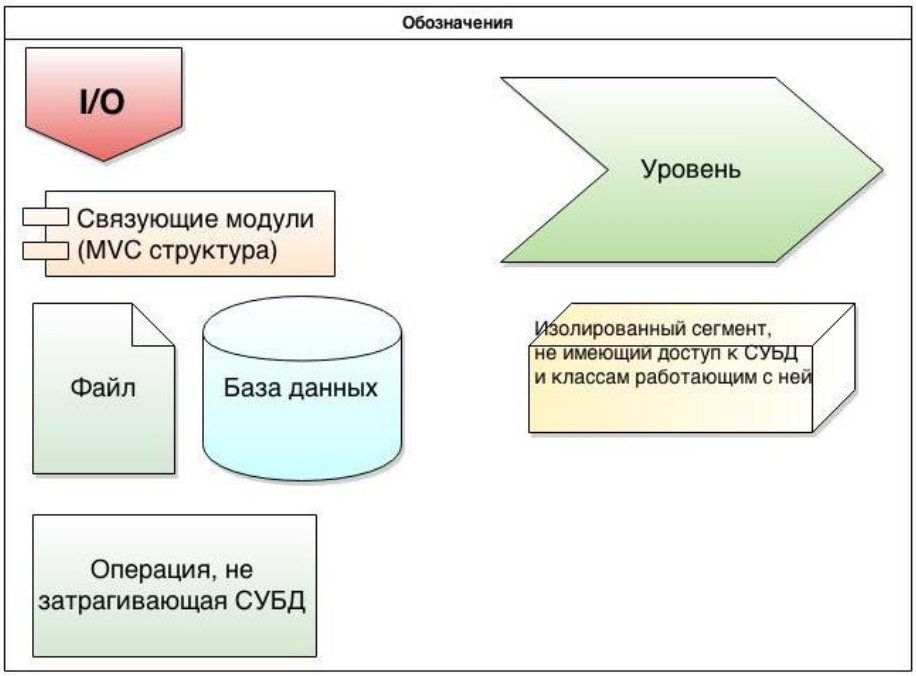

Рисунок 2.3.5 – Обозначения для концептуальной модели работы подсистемы базы данных

Выбор обуславливается не только набираемой популярностью и новизной, но и тем, что PDO позволяет системе управления контентом устанавливать свои правила на низком уровне взаимодействия. Благодаря представленному выбору, создаётся механизм работы с базой данных и организуются инстансы соединения с базой под конкретизированные модели взаимодействия CMS системы с пользователем. Таким образом, подсистемой определяются следующие ситуации: нужно ли создавать инстанс при запросе пользователя, нужно ли его закрыть или использовать тот, что использовался ранее другим или тем же самым пользователем. Посредством определения перечисленных ситуаций в значительной степени ускоряется работа системы, связанная с базой данных, а также организуется диалог обрабатываемого запроса, чем в значительной степени облегчается дальнейшее развитие системы и обеспечивается необходимый уровень защищенности в автоматическом режиме.

В предлагаемой модели предусматривается использование сильных сторон широко известных СУБД решений. К примеру, при получении результатов запроса от MySQL сервера в объекте также передается количество записей в результате. Однако SQLite такие данные не добавляет, что может вызвать путаницу при смене базы, СУБД или создании дополнительных модулей. По этой причине подсистема наделяется конкретным функционалом, позволяющим получить принятое значение от MySQL. При этом в том же синтаксисе может подсчитываться число полученных записей при отсутствии таковых данных в результате.

В представленном концепте связующие модули подсистемы имеют собственную организованную иерархию, дабы не допускать возможностей превышения допустимых методов тем или иным модулем, а также облегчить дальнейшее развитие системы. Концептуальная модель организации связующих модулей показывается на рис. 2.3.6.

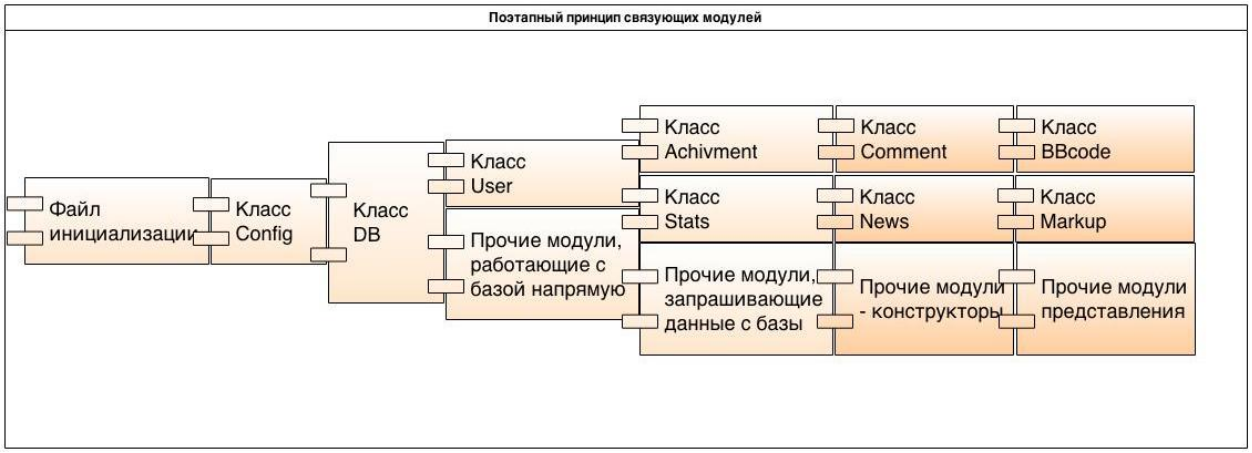

Рисунок 2.3.6 – Модель организации связующих модулей подсистемы с базой данных

Каждым модулем подсистемы поддерживается взаимодействие лишь с соседними модулями, чем абстрагируются механизмы формирования запросов, либо вовсе устраняется их необходимость. Таким образом, до тех пор пока не нарушается иерархия модулей, при создании новых модулей высокого уровня абстракции не нужно акцентироваться на базовых принципах безопасности системы и функциональных особенностях используемой СУБД, что позволяет в полной мере сосредоточиться на решении непосредственной функциональной задачи. При этом не требуется знания SQL синтаксиса.

Таким образом, после получения запроса пользователя с данными он проводится системой через этап конвертации допустимого HTML кода в BB–code, после чего данные фильтруются в блоке «санитизации», избавляясь от потенциально опасного кода. Данный процесс описывается в параграфе 2.4. Далее, переходя из шестого уровня в пятый, данные проходят процесс валидации дабы удостоверится, что пользователь ввел правильные данные, как то e–mail или дата рождения. На пятом уровне в процесс включаются модули–конструкторы, занимающиеся сбором данных и формированием их вывода в виде передачи на шестой уровень. Данными модулями запрашиваются данные с четвертого уровня, необходимые для формирования вывода. В связи с этим лишь с четвертого и третьего уровней обращения

передаются на второй уровень, где с помощью класса DB структурируются и интерпретируются запросы модулей в понятный для СУБД язык. После описанных событий с помощью модуля работы с конфигурационными файлами создается PDO объект или начинается работа с имеющимся PDO объектом, в процессе которой передаются необходимые ему команды.

Как одно из наиболее опасных функциональных требований, предъявляемых к системам управления контентом, связанных с организацией работы баз данных, является то, что средства создания резервных копий баз данных, находятся в данной подсистеме. Однако средства создания резервных копий баз данных реализуются не обычным способом. Чаще всего для быстрого создания резервной копии базы данных используются средства командной строки, из–за чего системой создается инстанс, к которому, при должной сноровке, может получить доступ злоумышленник. Более того, во время создания резервных копий в большинстве систем останавливается вся деятельность проекта. Подобное событие сопровождается потерей пользователей, а иногда и некоторых данных, не успевших сохранится.

В разрабатываемой системе управления контентом использование системных команд является недопустимым, поэтому создание резервных копий реализуется путем построения SQL запроса, также являющегося резервной копией всех данных в указываемой СУБД либо отдельной базы, либо отдельной таблице. Данным методом удаётся не только избежать использования системных команд, но и предоставляется системе возможность осуществлять работу в штатном режиме. Недостатком данной реализации является затрачиваемое время на создание резервной копии, для чего и создаётся функционал, позволяющий делать копии отдельных сегментов базы данных.

Особо выделяется модуль User, занимающийся хранением и формированием запросов к модулю DB, касающегося пользователей. Когда пользователь авторизуется в системе, его персональные данные по типу дата рождения, логин, имя, а также данные о возможностях и ограничениях,

связанных с групповой политикой, попадают в данный модуль, что позволяет не запрашивать их снова и снова. Таким же образом модулем User могут храниться данные других пользователей, материалы или профили которых в данный момент просматривается. Концептуальная модель модуля User представляется на рис. 2.3.7.

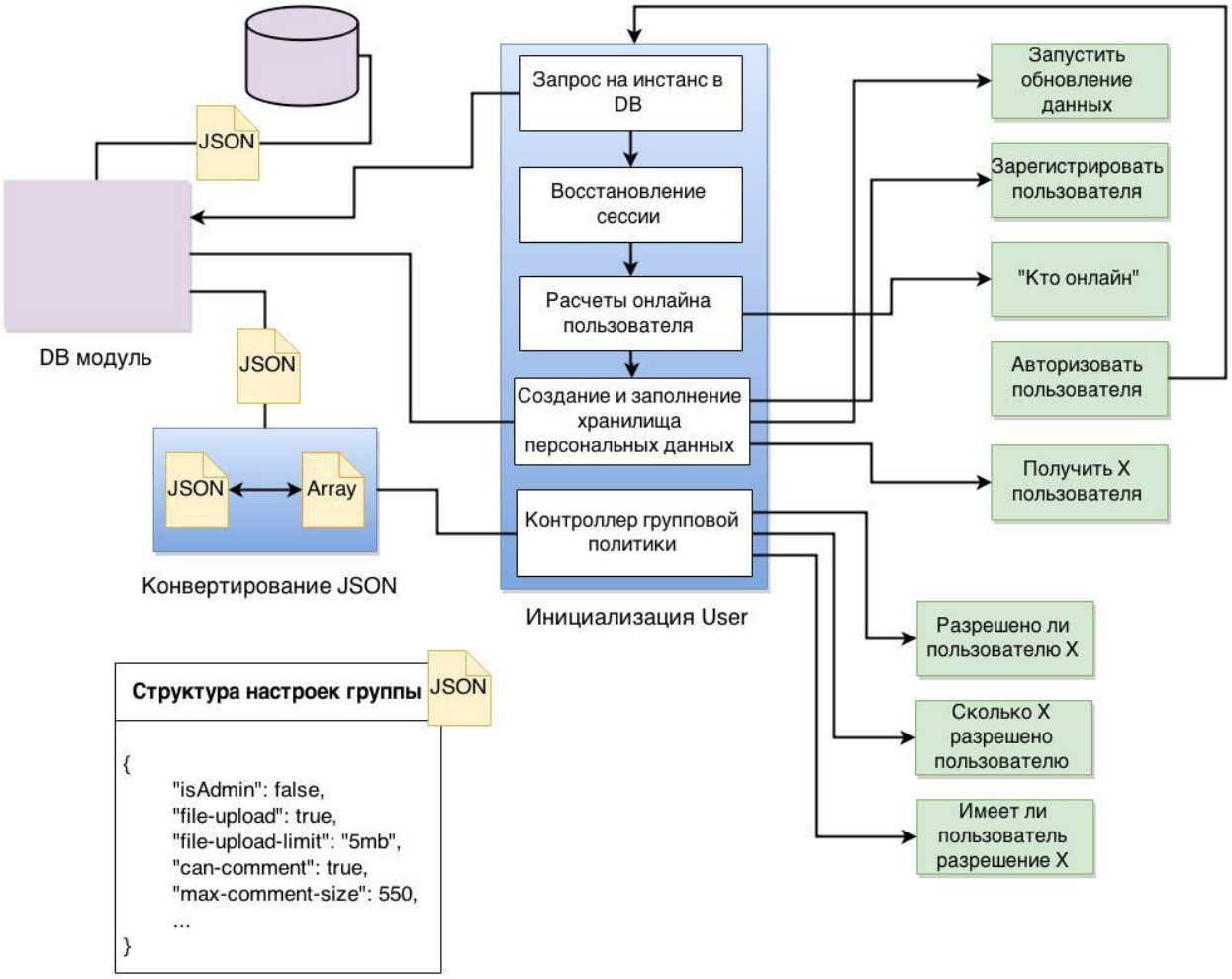

Рисунок 2.3.7 – Концептуальная модель модуля User

Благодаря хранению данных о разрешениях группы в формате JSON (JavaScript Object Notation – открытый текстовый формат обмена данными, основан на модели инициализации объектов в JavaScript), все настройки групповой политики содержатся в одном столбце базы данных, и что самое важное – легко редактируются. Каждая группа может иметь свои, особые настройки, поскольку их перечень может отличаться по порядку. Система

управления контентом действует по принципу – что не разрешено, то запрещено. Данные групповой политики, также как и персональные данные, хранятся внутри модуля на протяжении сессии или до внесения изменений.

Далее моделируется подсистема работы с электронной почтой, позволяющая организовывать базовые средства оповещения как пользователей, так и эксплуатирующего персонала. Эта подсистема характеризуется уникальными функциональными особенностями в реализации. Концептуальная модель подсистемы работы с электронной почтой изображается на рис. 2.3.8.

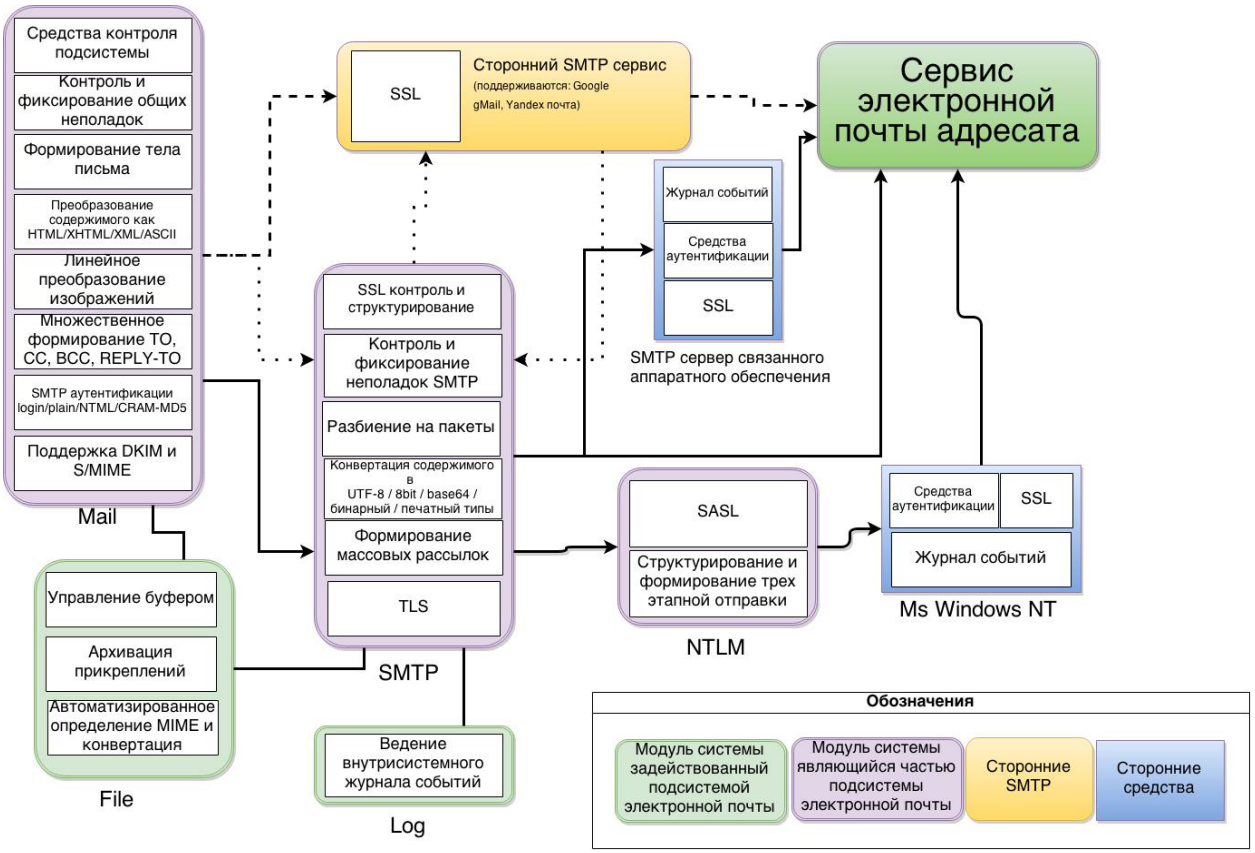

Рисунок 2.3.8 – Концептуальная модель подсистемы работы с электронной почтой

На изображении модели светло–зелеными блоками показываются лишь необходимые в данном контексте части возможностей модулей Log и File. Первый занимается ведением журнала событий во всей системе управления контентом. В данной подсистеме создается отдельный специализированный

журнал, детально описанный в приложении 2 – Руководство системного программиста, параграф 4.1.2. Второй модуль, File, занимается всей работой CMS системы, так или иначе связанной с файлами и файловой системой, будь то работа с мета данными файлов или создание/удаление папок. В данном решении модулем File исполняется роль буфера для файловых прикреплений, определение их типа в соответствии с MIME (Multipurpose Internet Mail Extensions – многоцелевые расширения электронной почты).

На рис. 2.3.8 стрелками показываются пути принятия решений подсистемой в зависимости от тех или иных конфигурационных настроек, детально разбираемых в руководстве системного программиста в приложении 2 к главе 3. В качестве наиболее значимого достоинства предлагаемой концепции является возможность отправки электронных писем без использования SMTP серверов. Модулями Mail и SMTP при необходимости осуществляется отправка электронных писем на уровне формирования пакетов. SMTP модуль, представленный на рис. 2.3.8, это класс системы управления контентом, который не является сторонним программным обеспечением, просто имеет схожее название из–за исполняемой роли в данной реализации.

В настоящее время существует бесчисленное множество вендоров, предоставляющих пользователям услуги для работы с электронной почтой. Для осуществления поддержки большинства данная подсистема и ее функционал тесно связанны с существующими стандартами RFC. Автоматизированный процесс формирования заголовочных данных письма формируется исходя из содержания и прикрепляемых файлов, а также адресов получателей, что определяет и типизацию конвертирования прикрепляемых файлов, осуществляя поддержку MIME стандарта типизации файлов, описанную в стандартах RFC 4289 и RFC 4021. Благодаря этому определяется процесс конвертирования, необходимый для конкретного прикрепления и непосредственного текстового сообщения. Последнее

осуществляется посредством поддержки форматов, описанных в стандартах RFC 2822 и RFC 2616, а именно:

1) 8bit;
2) Base64;
3) бинарный;
4) HTML;
5) печатный.

Работая с учетом стандартов RFC 6376, RFC 5863, и RFC 5585, подсистемой осуществляется полная поддержка DKIM (DomainKeys Identified Mail – метод E–mail аутентификации подписи в электронной почте) заголовков и осуществляется отправка писем в соответствии с особенностями, поддерживаемыми клиентским приложением пользователя для просмотра и организации электронной почты. В зависимости от того поддерживается ли, к примеру, HTML или клиент требует не размеченный текст, письмо будет распознаваться и предоставляться пользователю в требуемом виде.

Подсистемой также обеспечивается поддержка S/MIME (Secure/Multipurpose Internet Mail Extensions – стандарт для шифрования и подписи в электронной почте с помощью открытого ключа) заголовков для прикрепляемых файлов, формируемых в соответствии со стандартами RFC 2183, RFC 2047, RFC 2046, RFC 2045 и RFC 1341. Таким образом, к примеру, gMail принимая письмо, определяет типы прикрепленных файлов отдельно от графических изображений, являющихся элементами дизайна непосредственно самого письма, и предлагает пользователю открыть прикрепленный файл в соответствующем приложении на смарт устройстве или в подходящем и имеющемся у пользователя плагине браузера. Исходя из чего, создается поддержка следующих типов прикреплений:

1) стандартное прикрепление – удаленный файл, контекстное описание и мета данные которого прилагаются к письму;

2) линейное прикрепление – построчное разбиение файла с конвертированием содержимого, после чего осуществляется закрепление строковых данных в дата типе (шестнадцатеричном формате содержимого файла с соответствующими подписями по MIME);
3) альтернативное прикрепление – тоже что и стандартное прикрепление, но в виде альтернативного перехода на удаленный сервер;
4) альтернативное линейное прикрепление – тоже что и линейное прикрепление, но в виде блока данных – заглушки.

Распознавание типов самих файлов и установка соответствующих данных определяется по стандарту MIME, описанному в RFC 4288, RFC 4289, и RFC 4855, что позволяет осуществлять поддержку общепринятых форматов файлов и в соответствии с этим их конвертировать надлежащим образом.

Поскольку электронные письма, создаваемые на уровне пакетов, не имеют SSL (Secure Sockets Layer – уровень защищенных сокетов) – криптографический протокол) подписи, то высоки шансы, что такое письмо будет рассмотрено как спам и отправлено в соответствующую папку сервиса электронной почты пользователя. В связи с этим предусматривается не менее интересный вариант формирования пакетов системой, с последующим подписанием письма сторонним SMTP сервисом, таким как Google gMail или Yandex почта. Таким образом, письмо получит требуемый уровень доверия весьма серьезных вендоров, и в то же время получит все плюсы формирования и производительности аппаратного обеспечения эксплуатирующего ресурса, ведь без отправки gMail и яндекс–почта не будут собирать письмо, конвертировать содержимое, и разбивать на пакеты. Однако и этот вариант возможен, можно переложить все трудности и работу на эти сервисы.

При разработке системы учитываются особенности сторонних сервисов. Во–первых они ограничивают количество отправляемых пользователем писем, в то время как от систем управления контентом обычно требуется довольно много. Во–вторых, если отправлять одно письмо по многим адресатам, то получатели увидят персональные данные других пользователей в поле с пометкой «кто еще получил такое письмо», а беспочвенное раскрытие персональных данных всегда сопровождается рисками.

В подсистеме также учитывается метод работы с локальным или удаленным SMTP сервером, при должной настройке которого CMS система возьмет на себя большую часть функций, что позволит повысить производительность. На тот случай, если сервер находится под управлением операционной системы Windows NT, в подсистеме имеется отдельный модуль NTLM, позволяющий работать по протоколу NTLM (NT LAN Manager – протокол сетевой аутентификации для Windows NT). В этом случае аутентификация осуществляется согласно стандарту RFC 4422, что позволяет осуществлять поддержку технологии SASL (SASL – Simple Authentication and Security Layer – простой уровень аутентификации и безопасности – технологический метод предоставления аутентификации и защиты данных в протоколах на основе соединений), используемой в SMTP серверах на базе Windows NT.

Подводя итоги изложенных выше концепций, стоит отметить, что в подсистеме поддерживается три механизма шифрования электронной почты, вне зависимости от SMTP сервера, такие как SSL, TLS и отсутствие шифрования. Для шифрования файловых прикреплений может использоваться модуль File, позволяющий проводить архивацию файлов с последующей защитой паролем, поддерживаемым в ZIP (формат сжатия данных и архивации файлов). Модулем файл может самостоятельно проводиться шифрование файлов, подробнее об этом излагается в приложении 2 – Руководство системного программиста.

Поскольку любой выбираемый путь использования подсистемы Mail является трудоёмким, то в подсистеме реализуется механизм приоритетов писем с пятью уровнями. Подробно данный механизм и его использование описываются в приложении 2 – Руководство системного программиста, параграф 5.4.

Данная система управления контентом наилучшим образом показывает свой потенциал при реализации ресурсов научно–образовательной среды. Как представлялось выше, во многом по этой причине упор делался на производительность системы, что также сыграло свою роль в отказе от сторонних библиотек, фреймворков и компонентов. Однако отличительной особенностью системы, обеспечивающей конкурентоспособность CMS–решения в данной узкоспециализированной среде, является виртуальная машина.

Среди общепринятых и устоявшихся понятий виртуальной машины не найти абсолютно точной аналогии с предлагаемым решением. Наиболее часто фигурируют два варианта понятий. Первый вариант понятия: виртуальная машина представляет собой виртуализированную среду, имитирующую работу аппаратного обеспечения. Яркими примерами могут служить машины, создаваемые в Oracle VirtualBox или VMware. Данное решение имитирует лишь модель работы процессора и оперативную память, поэтому установить на нее серверную операционную систему, настроить сетевое подключение и периферийные устройства – не получится.

Второй вариант понятия: «виртуальная машина» является программным обеспечением, проводящим интерпретационные действия над абстрактным языком программирования через преобразования его в байт–код. Вследствие чего, та же машина предоставляет среду для исполнения данного байт–кода и параллельного тестирования. Примерами данного решения служат виртуальные машины Java и Python.

Обсуждаемый вариант машины имеет ряд средств, в особенности транслятор для перевода языка программирования Assembler в байт–код, позволяющий собирать машину посредством добавления сторонних средств.

Поскольку перед системой стояла задача обеспечить пользователей необходимым инструментарием для создания и решения комплексов упражнений с целью повышения образовательной и технической компетенции через практическое изучение, то рассматривались два возможных варианта:

1) создание интерпретатора–симулятора для конкретного языка программирования. Этот вариант отклонён из–за возникающих сложностей в процессе расширения спектра поддерживаемых языков и усложненного процесса обновления спецификаций;
2) установка отдельной «виртуальной машины» на сервере, к которой бы предоставлялся удаленный доступ через веб интерфейс по протоколу SSH (Secure Shell — сетевой протокол прикладного уровня для удаленного управления операционной системой и туннелирования), где можно было бы использовать уже существующие средства программного обеспечения для компиляции и интерпретации кода. Данный вариант признан недопустимым из–за высокой нагрузки и возможных осложнений в информационной безопасности.

Таким образом, оба варианта признаются не пригодными для решения поставленной задачи. Однако посредством сочетания выборочных решений обоих подходов создаётся предлагаемый вариант виртуальной машины, концептуальная модель которой представлена на рис. 2.3.9. Предлагаемый вариант машины представляет собой DLL библиотеку, подключаемую в качестве модуля веб–сервера Apache. Предложенный приём позволяет осуществлять ее работу в отдельном процессе, изолированно от системы управления контентом, и держать неизменную архитектуру в виде модели черного ящика.

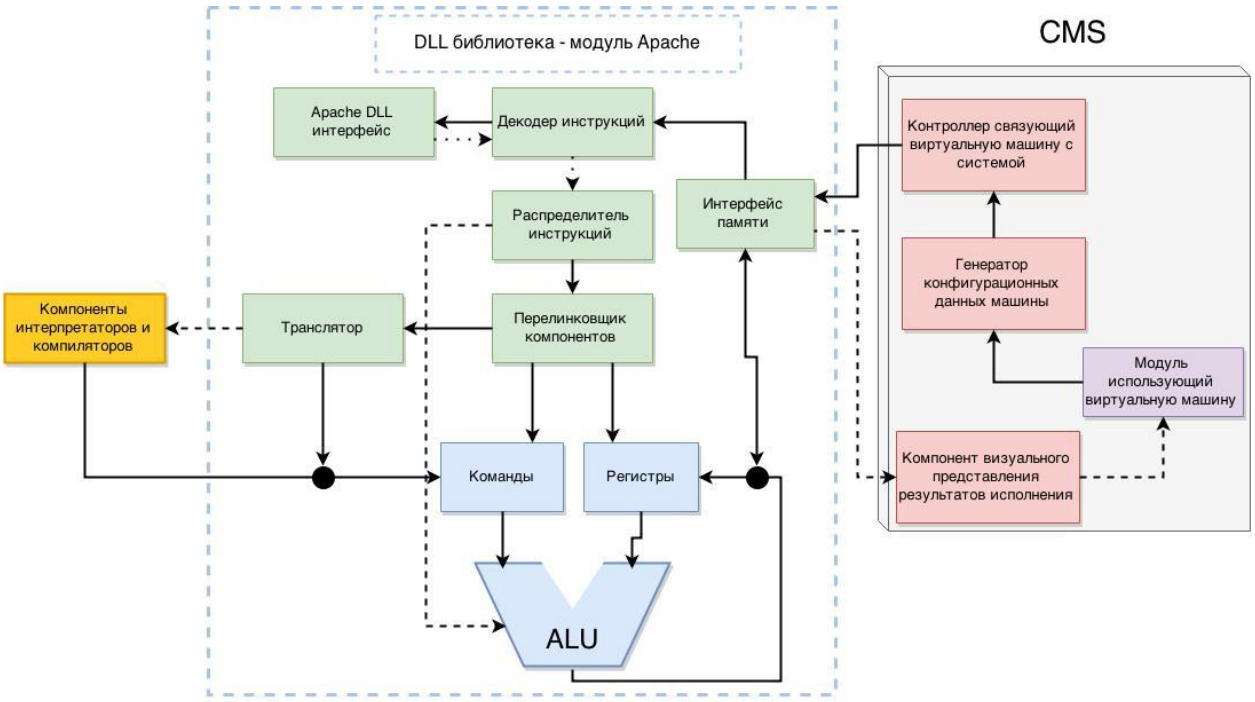

Рисунок 2.3.9 – Общий концепт виртуальной машины

Модули сервера Apache используют язык программирования C++. За счёт выбора указанного языка программирования удалось создать компонент столь простой и производительный (в рамках современного мира информационных технологий), что модули CMS создают комплекс с конфигураций, исходя из требований стоящих перед пользователем.

В конфигурации учитывается и допустимый размер стека, и число ячеек в памяти, и разумеется перечень компонентов, необходимых для подключения. Ради упрощения процесса реализации в качестве встроенного транслятора выбран Assembler. Поскольку данный язык по уровню абстракции весьма близок к машинному коду и имеет не богатый синтаксис (в сравнении с высокоуровневыми языками программирования), то реализация транслятора не требует больших объемов ресурсов. К тому же, как указывалось ранее, большинство программных средств можно дизассемблировать, то есть, если пользователю понадобится компилятор C++, то можно не искать решений на языке Ассемблер, а просто заранее

дизассемблировать подходящий компилятор. Данный приём широко используется в инфосфере.

Блок ALU (Arithmetic logic unit – арифметико–логический компонент) представляет собой комплекс битовых операций над машинным кодом. В машине эмулируется работа центрального процессора, в сущности, также как и в физических эквивалентах.

На данном этапе реализации машина использует 16–битную архитектуру, во многом для малого (по современным меркам) числа адресов, что ведет к понижению требований к аппаратному обеспечению. Однако в дальнейшем возможно добавление поддержки другой архитектуры, устанавливаемой в конфигурациях. Данное расширение функциональности не должно быть сложным в плане реализации. Предлагаемый вариант виртуальной машины нуждается в полевом тестировании, проводимом перед осуществлением подобных изменений. Обсуждаемое расширение потребуется для использования средств параллельного программирования.

Эмуляция работы стека осуществляется через расширяющийся массив, сама же логика представляет собой список элементов, организованных по методологическому принципу LIFO (Last In, First Out – последним пришел, первым ушел – способ организации и манипулирования данными относительно времени и приоритетов). Данный процесс проиллюстрирован на рис. 2.3.10.

Представленные на рисунке регистры работают идентично с использующимися в ассемблере. Кроме того, данное решение связанно с упрощением осуществления поддержки транслятора ассемблера.

Как и в физическом аппаратном обеспечении, данные хранятся в оперативной памяти вместе с инструкциями и контролируются единым интерфейсом памяти. Поскольку виртуализация центрального процессора осуществляется максимально идентично с физическим, данный функционал детально не описывается в данной работе.

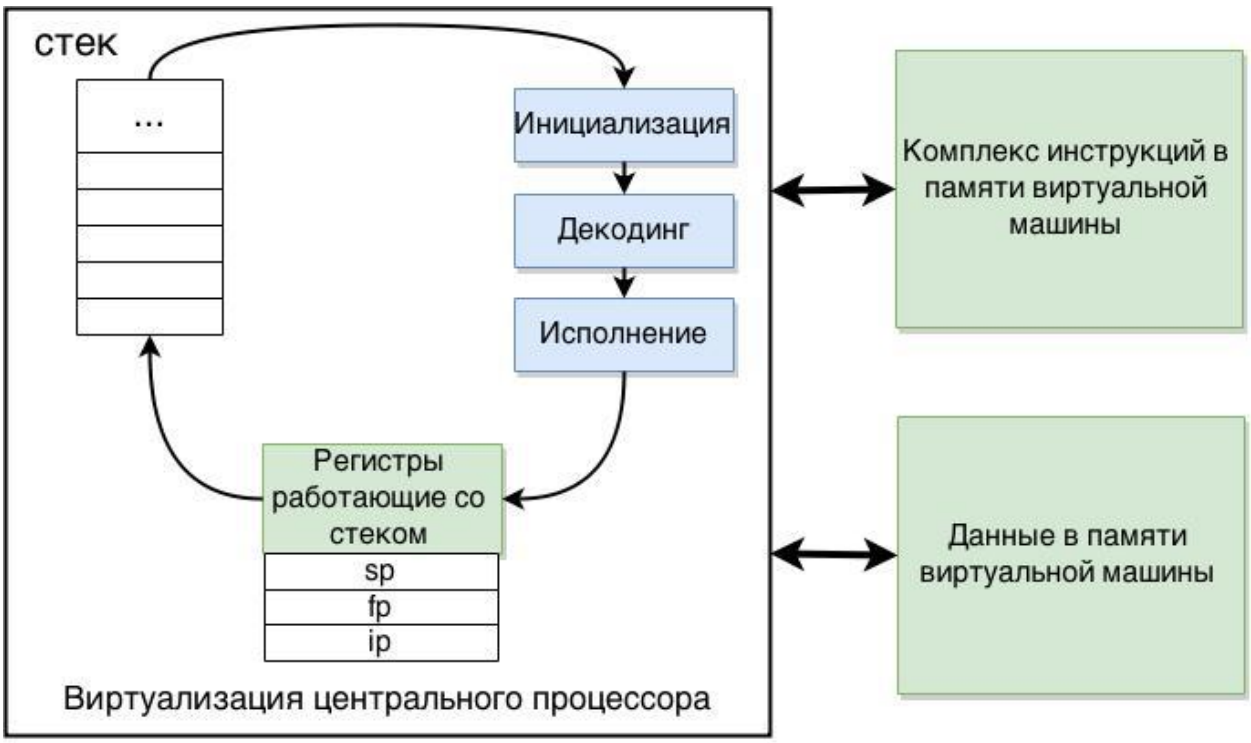

Рисунок 2.3.10 – Концептуальная модель процесса работы стека виртуальной машины

2.4 Разработка алгоритмов работы подсистем

При первой же обработке запроса пользователя к системе управления контентом срабатывает главный алгоритм инициализации всей системы. Его архитектура представлена на рис. 2.4.1. В его задачу входит проведение входных настроек установленных системным программистом и введение их в систему, а также базовое определение сессии пользователя используя функционал модуля User, что позволяет с помощью зашифрованных данных файла COOKIE, узнать был ли этот пользователь раньше и в случае указанной опции запоминания сессии в системе решить тот ли это пользователь и надо ли восстанавливать его сессию.

Справа на рисунке изображен комплект прилагаемых настроек, которые иерархически распределены так, чтобы не разделять функциональные настройки модулей, однако, чтобы имелось должное логическое разбиение на категории исходя из их представляемой роли в операция. Что позволяет использовать одни и те же настройки в множестве

методов и функций не связанных друг с другом, но объединенных общей концепцией функционирования системы.

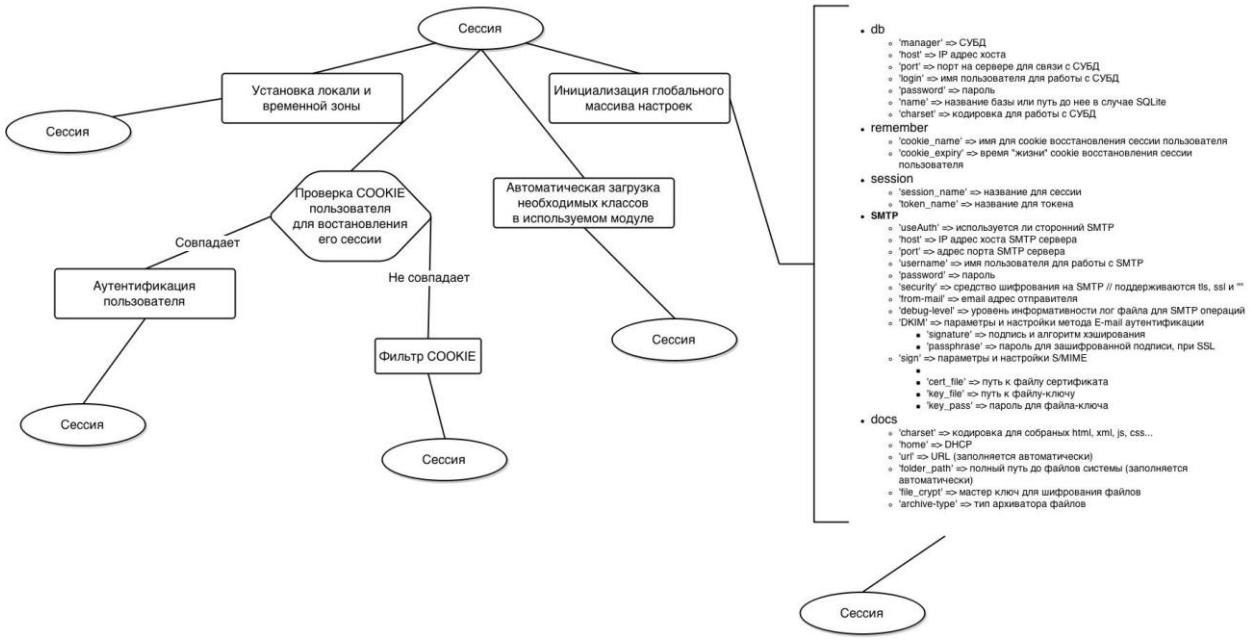

Рисунок 2.4.1 – Алгоритм инициализации системы управления контентом

Также стоит отметить, что алгоритм инициализации включает в себя механизм автоматического подключения и курирования программных компонентов, по мере их вызова в тех или иных модулях. Благодаря чему изрядно уменьшается время отклика и подключения не дублируются, обеспечивая должный уровень простоты и абстрактного взаимодействия с модулями, при расширении функционала.

Поскольку любая процедура ввода данных пользователем, требует проверки на вредоносный или недопустимый код, в системе реализован механизм санитизации ввода работающий по алгоритму представленному ниже на рис. 2.4.2. Однако, данный функционал был бы не полным без предоставления пользователям альтернативных методов иллюстрирования и придания интерактивности к публикуемым материалам, чем занимается модуль BBcode, переводящий допустимый HTML в иные лингвистические единицы предусмотренные и одобренные эксплуатирующей группой, которые в последствии, автоматически интерпретируются в необходимый

HTML. Модуль BBcode не имеет отличительных или важных особенностей так как работает как парсер строковых данных при помощи регулярных выражений, и потому в данной работе детально не описывается.

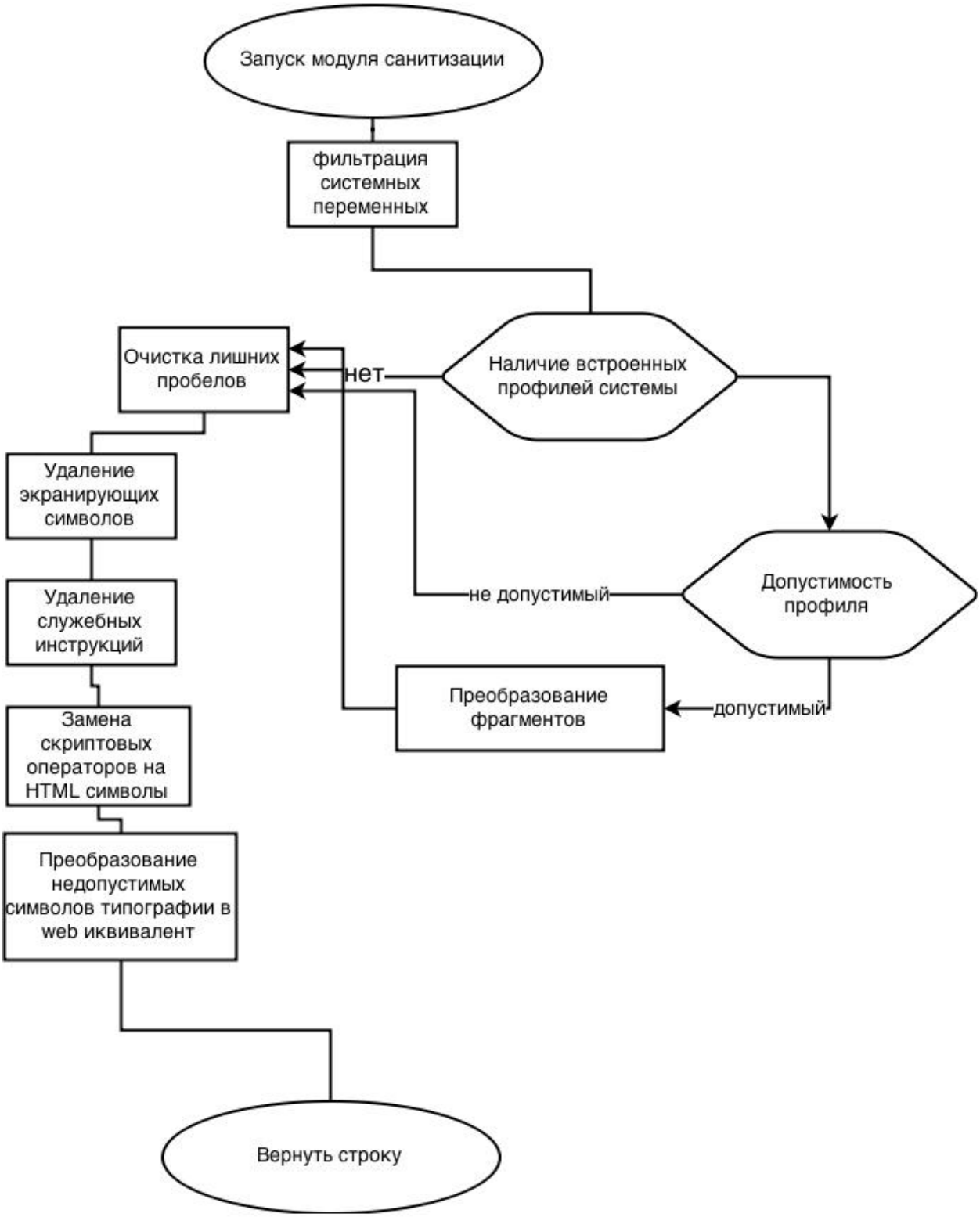

Рисунок 2.4.2 – Алгоритм санитизации ввода

Алгоритм обрабатывает ввод в строковой типизации данных, убирая возможные служебные инструкции, в процессе интерпретируя элементы и специальные символы в их печатный эквивалент, что исключает возможность прохождения любого кода, от которого может пострадать функциональность системы или опыт взаимодействия пользователей.

На клиентской стороне, функционал системы управления контентом, во многом связанный с интерактивностью интерфейса и взаимодействием с пользователем, базируется на собственном JavaScript фреймворке без использования сторонних библиотек и готовых модулей. Данное решение обусловлено потребностью предоставления необходимого инструментария пользователям создающим интерактивные обучающие задачи, упражнения и компоненты, в специализированном виде, связанным со спецификой решения. Соответственно, это исключает любые возможные конфликты компонентов клиентской части системы со сторонними библиотеками, которые, возможно, будут использовать пользователи.

Чтобы снизить время загрузки страниц с пользовательским контентом, требующим ту или иную библиотеку или модуль, в фреймворке имеется механизм асинхронной загрузки компонентов, таких как JavaScript или CSS. Алгоритм работы представлен ниже на рис. 2.4.3. Несмотря на скромный дизайн, асинхронный загрузчик представляет собой многофункциональный компонент с широким спектром возможностей. Помимо загрузки файлов и исполнения кода после осуществления полной загрузки страницы, загрузчик имеет комплект средств валидации как URL (Uniform Resource Locator – единообразный локатор электронного ресурса) так и данных запросов с непосредственного адреса, что позволяет вводить ограничения на не доверенные CDN источники.

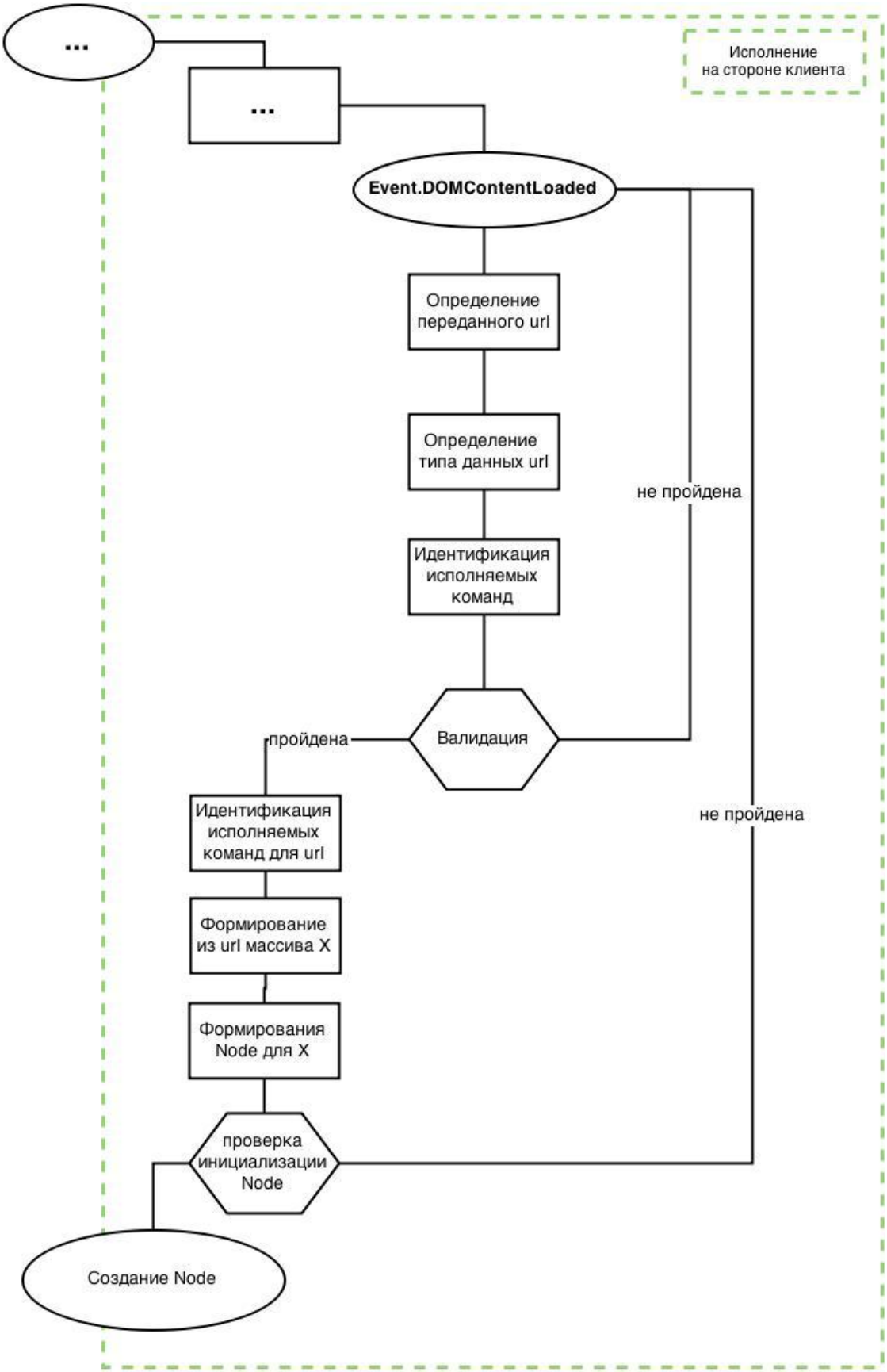

Рисунок 2.4.3 – Алгоритм работы асинхронного загрузчика

Также асинхронный загрузчик автоматически определяет тип запрашиваемого файла с учетом прописанных в нем внутренних мета данных. Нельзя не отметить расширяемость и масштабируемость компонента, ввиду представляемых возможностей загрузчика было создано два дополнительных компонента, первый довольно прост, загружает несколько файлов требуемых для функционала API социальных сетей, после чего добавляет необходимые элементы интерфейса, алгоритм работы показан на рис. 2.4.4.

Главной особенностью данного расширения является не создание Node на готовой HTML–странице, и даже не соответствующий HTML–код элементов для разных API с разным синтаксисом. Наиболее интересна его возможность ожидания ответов удаленных API, о завершении загрузки необходимых файлов с последующим позиционированием элементов, поскольку загрузка осуществляется единовременно со всех сервисов, в то время как концепция «базового» асинхронного загрузчика заставляет ожидать полной загрузки каждого файла в отдельности. Однако в случае CDN социальных сетей, данное свойство не требуется.

Однако в качестве первого, формат XLS выбран не случайно, это обусловлено его широким распространением в информационной среде, а также поддержкой различных производителей программного обеспечения. Плюс ко всему, данный формат логически сочетается с представлением рядовых пользователей о таблицах, что позволяет использовать имеющийся у них опыт.

Учитывая тот факт, что система не использует сторонних компонентов, в клиентской части реализации, для проведения диалогов с пользователями, а также для визуального отклика инструментариев, создан собственный механизм генерации и вызова диалоговых и модальных окон. Алгоритм его работы показан на рис. 2.4.6.

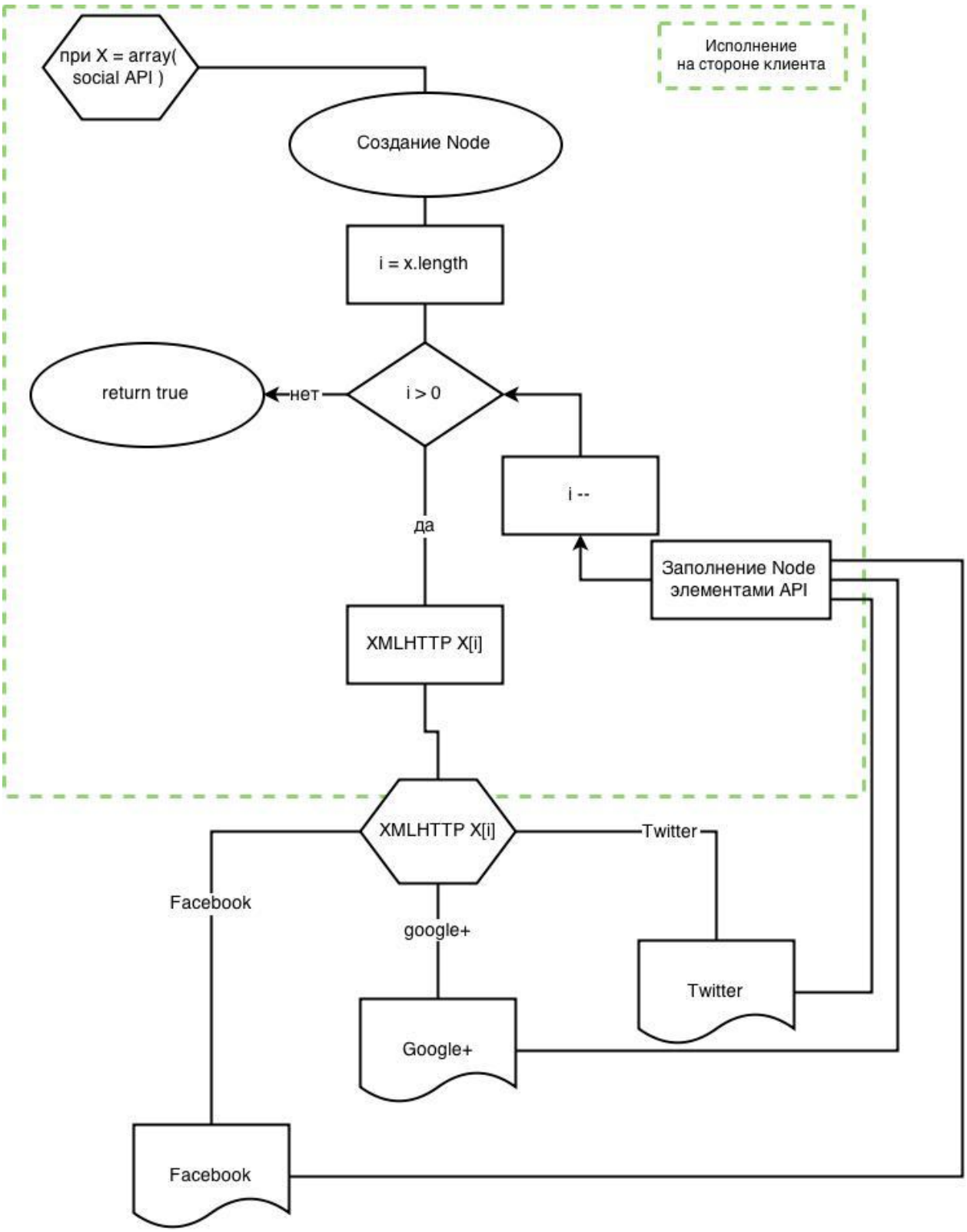

Рисунок 2.4.4 – Алгоритм работы расширения асинхронного загрузчика для API социальных сетей.

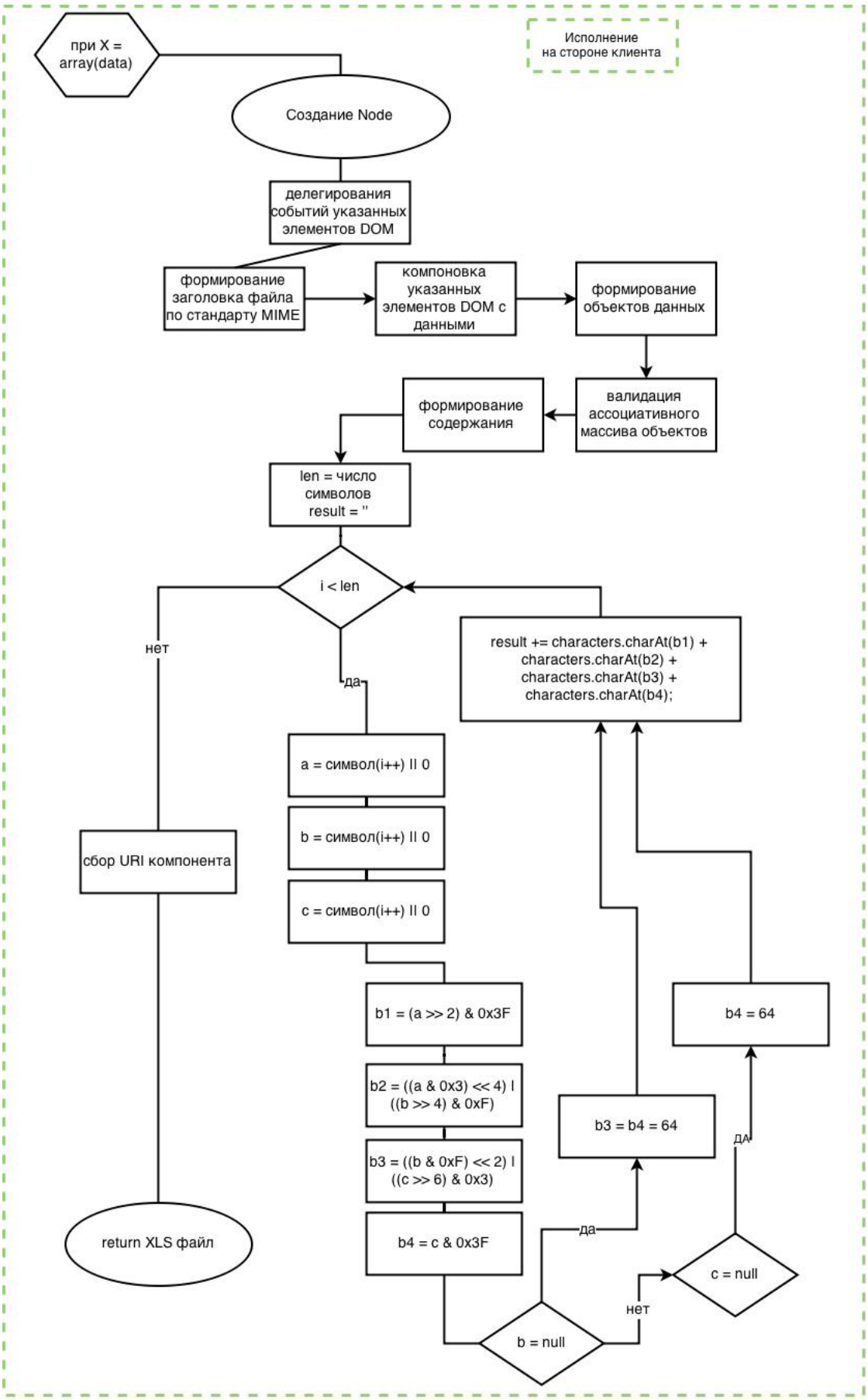

Рисунок 2.4.5 – Алгоритм расширения для генерации XLS файлов на стороне клиента

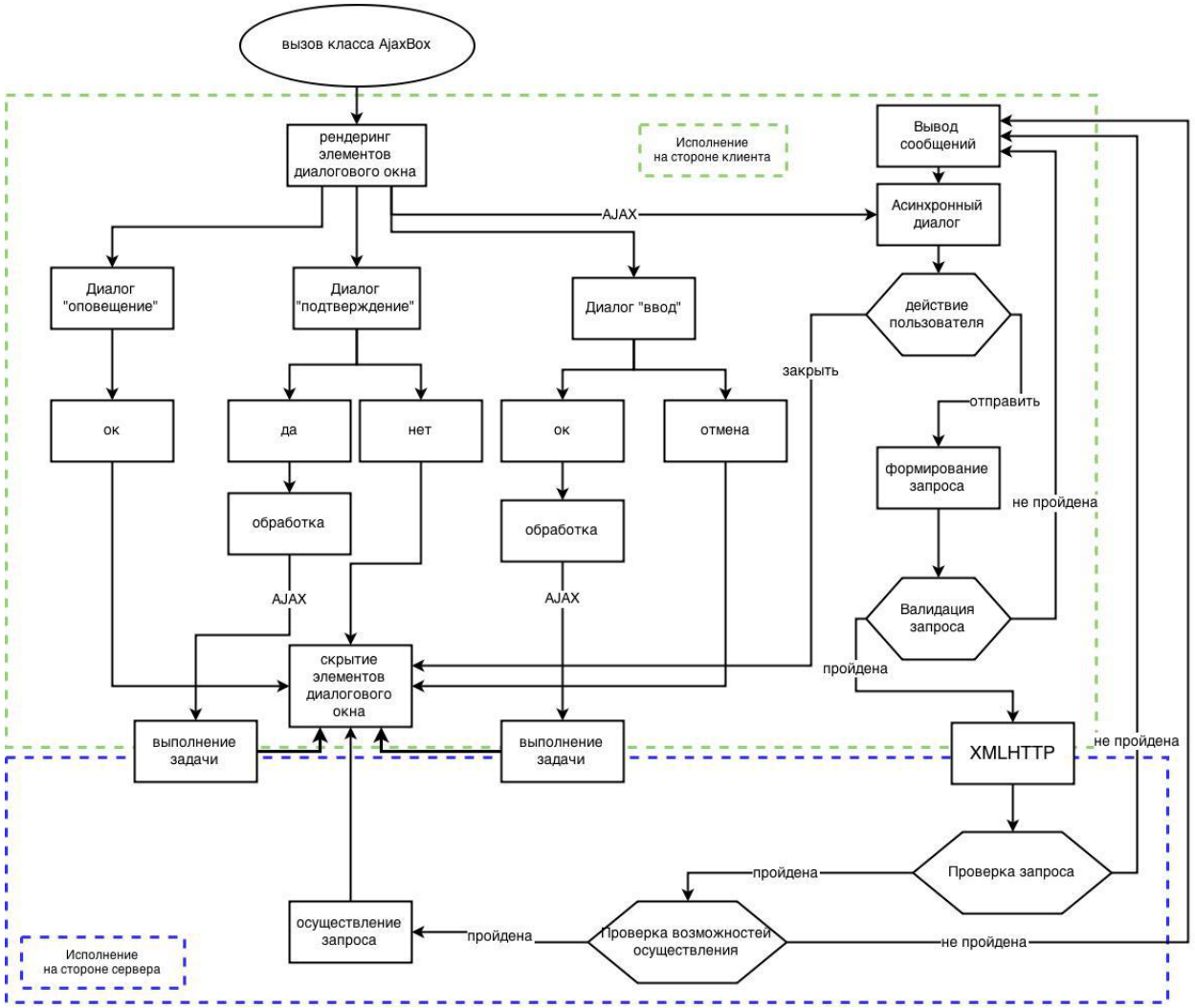

Рисунок 2.4.6 – Алгоритм работы генератора диалоговых и модальных окон

Отличительной особенностью такого решения является использование серверных компонентов системы управления контентом AJAX технологию, что позволяет не только не перезагружать страницу при исполнении того или иного действия, но и вести непрерывный диалог исполняя как простые задачи вроде валидации формы регистрации, так и сложные комплексные запросы как аутентификация, голосование в опросе или мини–чат. Из–за столь обширных возможностей, на стороне сервера всегда осуществляется простой процесс валидации запросов передаваемых через AJAX XMLHTTP. Однако данный механизм, как и общая концепция системы, имеет повышенный уровень абстракции и предполагает возможности масштабирования и расширения, к примеру он участвует в работе

компонента загрузки файлов, что разумеется требует наличие собственных средств валидации и работы с файлами, но при включенном JavaScript на стороне пользователя, загружать файлы намного проще и быстрее. Алгоритм работы загрузчика файлов представлен на рис. 2.4.7.

Алгоритм имеет клиентскую часть занимающуюся валидацией запроса на загрузку и допустимость файла к загрузке, исходя из характеристик последнего, как то размер, тип и количество файлов. Однако поскольку результаты данной валидации могут быть скомпрометированы, на стороне сервера они не имеют никакого значения и служат лишь для повышения производительности системы, чтобы не отправлять лишние запросы на сервер.

На стороне сервера реализация алгоритма представляет собой единый класс для валидации файлов и работы с файловой системой, все настройки и требования указываются вне файла в исполняющих сценариях, что позволяет использовать механизм при разных условиях и с разными требованиями, в том числе и групповой политики, как то загрузка нескольких файлов, общий размер загрузки и каждого файла. Файл помещается во временный каталог Apache сервера, после чего процесс валидации осуществляет проверку файловой системы на готовность принятия файла, где также проверяется разрешенность данного типа файла по стандарту MIME, для чего предусмотрено два варианта: черный список и белый список. В случае использования черного списка разрешается загрузка любого типа файлов, кроме определенного списка, в случае использования метода белого списка разрешается загружать лишь файлы того формата, который разрешен. Оба списка также могут редактироваться в исполняющих сценариях.

В качестве альтернативного варианта, можно разрешить загрузку любых файлов, но файлы запрещенные к загрузки, используемым методом разрешения типов (будь то черный или белый список), получат новое разрешения, то есть суффикс.

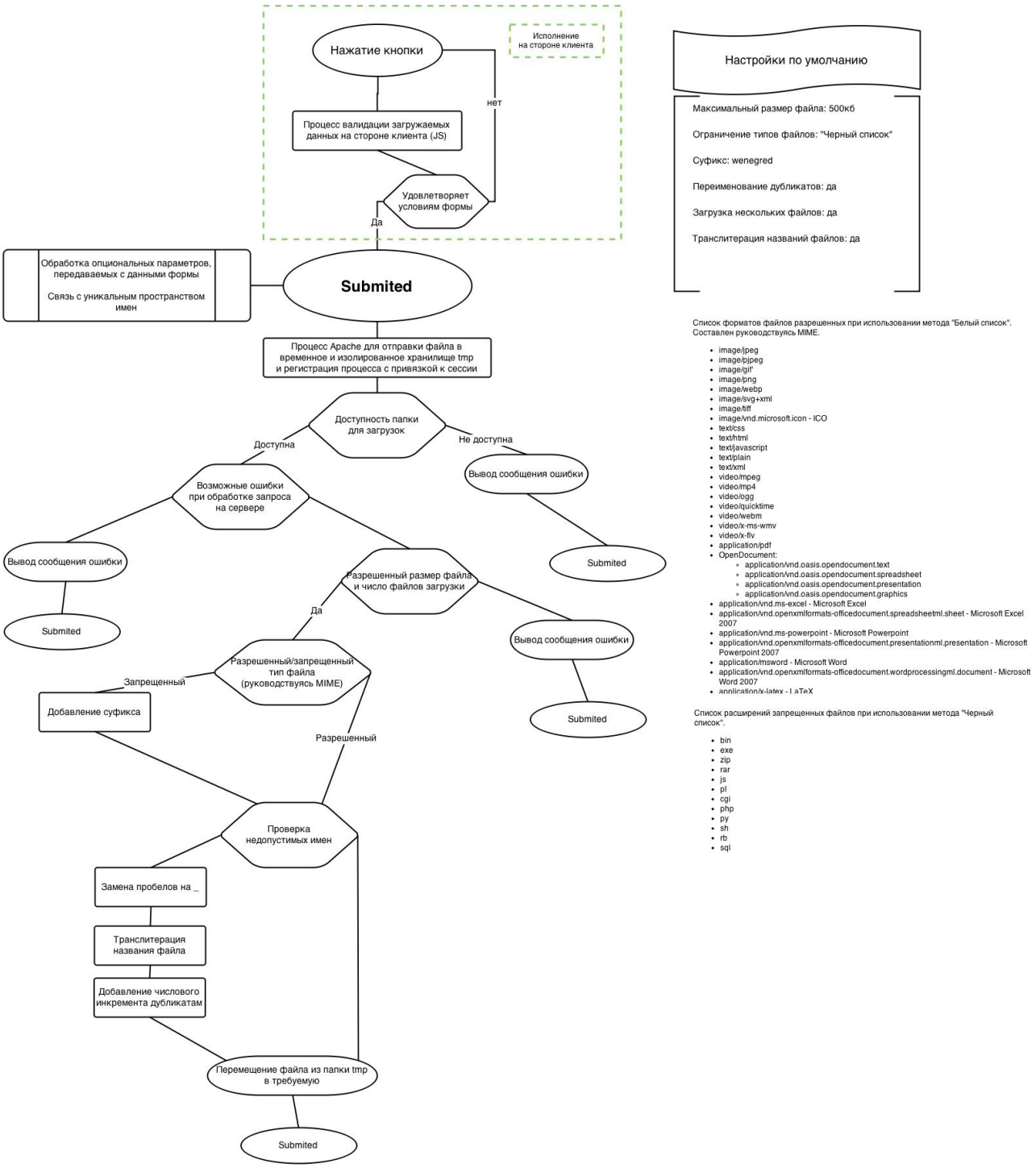

Рисунок 2.4.7 – Алгоритм работы файлового загрузчика

В механизме имеется комплекс средств для транслитерации имен файлов, добавления числового инкремента — индекса, при наличии такого названия, а так же замены пробелов на подчеркивания, для URL. В случае каких либо нарушений выдаются сообщения об ошибках, при успешной загрузке выдается оповещающее сообщение, а также перечень мер

повлиявших на имена файлом, с их указанием. При загрузке комплекта файлов, если возникли проблемы или нарушения требований с каким либо файлом, загрузка не осуществляется, а пользователь получает список файлов с указанием причин по которым тот или иной файл не был загружен.

2.5 Организация файлов системы

В соответствии с особенностями архитектуры системы, описанными выше, разработана организация файлов, распространяющаяся как на внутренние файлы CMS, так и на генерируемые системой файлы. При организации сохранилась логика модульной архитектуры системы и абстрактное представление образующих компонентов.

В корневом каталоге системы, представленном на рис. 2.5.1, находятся все папки проекта, а также компоненты, организующие виртуальные разделы для обработки пользовательских запросов, что позволяет изолировать аспекты представления пользователю от исполняемых файлов–инструкций и модулей подсистем.

Служебный файл .htaccess, представляет собой комплекс конфигурационных инструкций для веб–сервера Apache, работающий со всеми файлами и каталогами системы. В данном файле при помощи регулярных выражений формируются интуитивно понятные ссылки (ЧПУ).

На рис. 2.5.2 представлена структура трех каталогов:

1) actions – содержит файлы–обработчики форм для определения необходимых мероприятий, связанных с вводимыми пользователем данными, а также исполняемые файлы для Ajax диалогов;
2) blocks – содержит файлы, используемые в качестве сокращений повторяемых статичных блоков системы;
3) core – содержит файл инициализации системы и файлы–журналы событий.

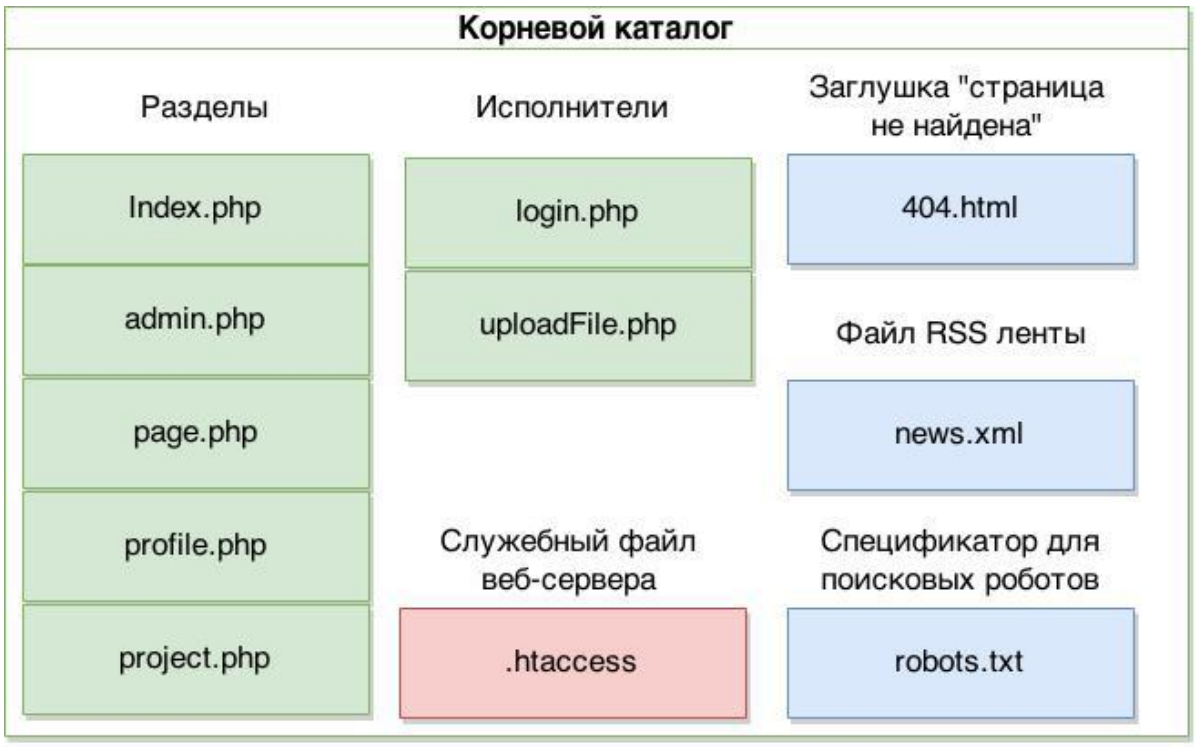

Рисунок 2.5.1 – Структура корневого каталога системы

Содержащиеся файлы .htaccess блокируют внешний доступ по любому запросу, который не содержит необходимый генерируемый токен, определяющий то, что данный запрос является ликвидным и является технически относящимся к форме, заполняемой пользователем, направившим запрос, или Ajax диалогом, совпадающим с пользователем, создавшим сессионный запрос на Ajax.

На рис. 2.5.3 показана структура каталога content, содержащего файлы загружаемые пользователем, распределенные по папкам в зависимости от раздела, использующего загружаемые файлы, и их типов. Файлы, содержащиеся в данных папках, являются общедоступными, и могут быть запрошены по средствам прямых ссылок.

На рис. 2.5.4 раскрыты особенности структуры каталога functions, содержащего php файлы, где каждый файл представляет лишь одну одноименную функцию. Интерпретатор исполняет меры по нахождению и загрузке файла лишь один раз, что улучшает производительность, так как механизмы манипуляции с файлами исполняются долго. Данное решение

связано с модулем Load, позволяющим осуществлять контроль над загруженными функциями и не загружать новые, в связи, с чем функции запрашиваются лишь один раз и только в случае исполнения тех или иных условий, связанных с запросом и исполнением операций модулей. Благодаря чему, код остается чистым и модульным, а производительность не страдает.

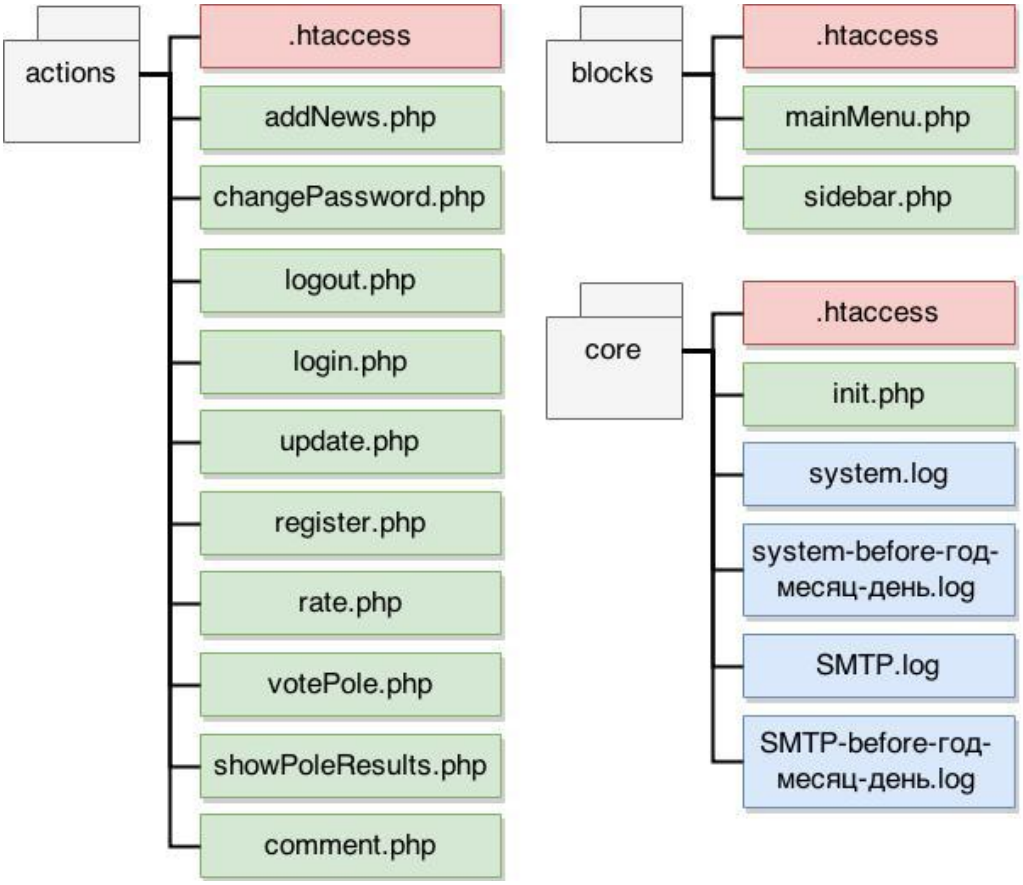

Рисунок 2.5.2 – Структура каталогов: actions, blocks и core

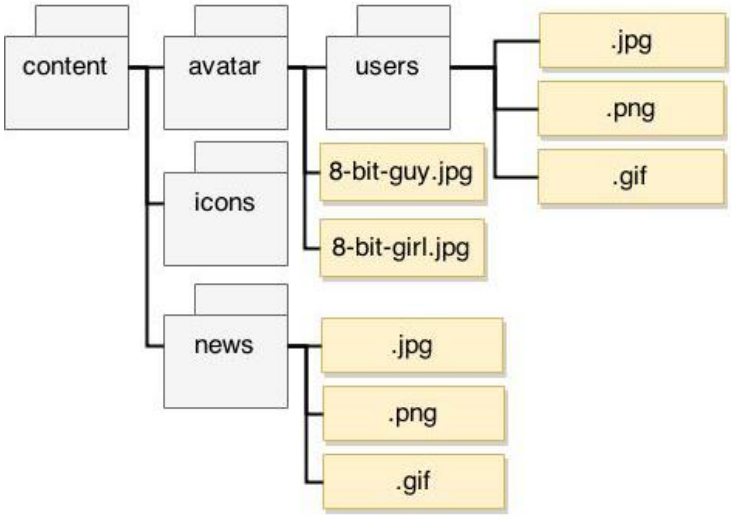

Рисунок 2.5.3 – Структура каталога content

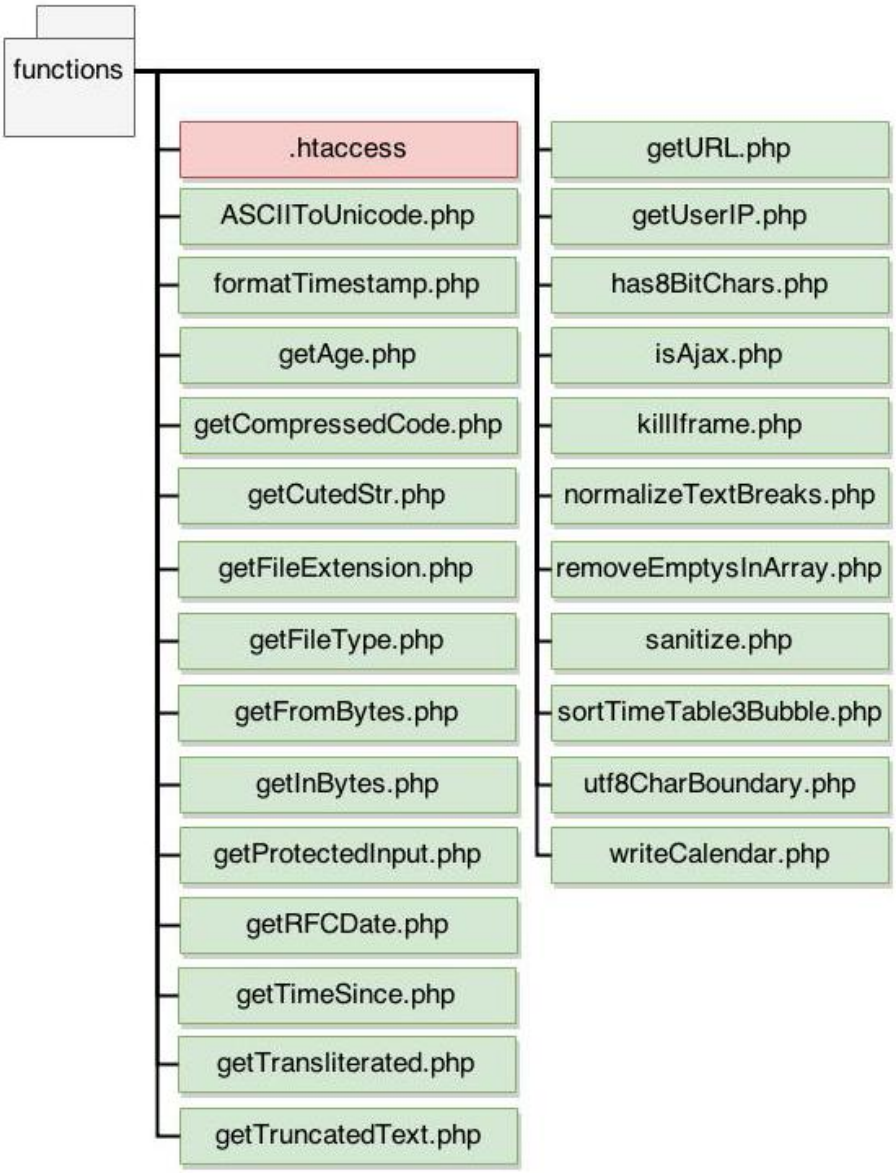

Рисунок 2.5.4 – Особенности структуры каталога functions

Наиболее широко применяемые механизмы работы системы и модулей технически представляют собой классы. Каждый класс представлен в одноименном файле для удобства введения коррекций, а также для механизма автоматической загрузки требуемых и задействованных классов. Структура каталога приведена на рис. 2.5.6.

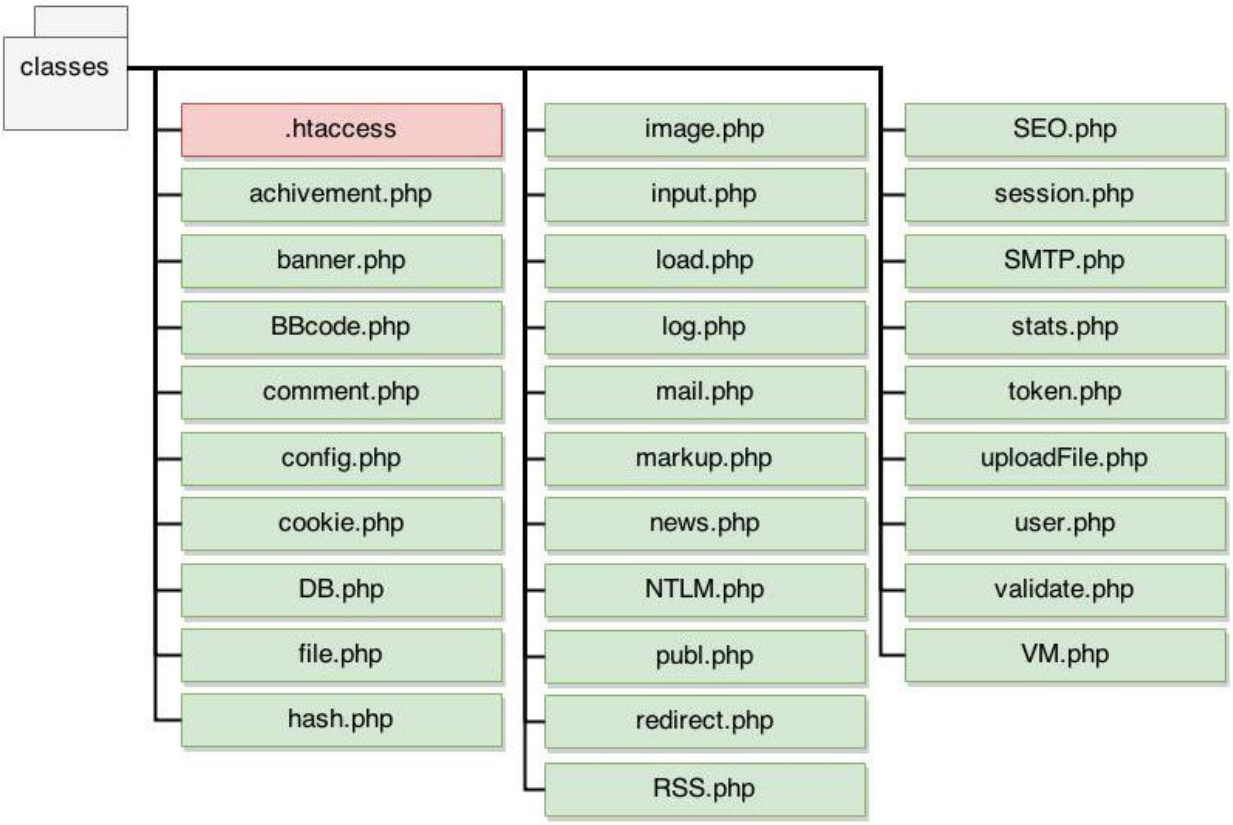

Рисунок 2.5.6 – Структура каталога classes

Каталог JSON, показанный на рис. 2.5.7, содержит общедоступные данные, структурирование, организация и представление которых осуществляется посредством компонентов системы управления контентом. Указанные объекты используются для цитирования комплектов ссылок на материалы, схожими с представленными, а также для быстрого и простого редактирования материалов, расположенных в разных разделах сайта или частично представленных в публикациях.

Рис. 2.5.8 показывает структуру каталога JS, созданного для организации файлов JavaScript. В корне каталога содержится файл с автоматически собранным и сжатым кодом, находящимся внутри папки src. Данный подход позволяет не только уменьшить размер файла передаваемого пользователю, но и сократить число символов, что ускоряет обработку инструкций браузером клиента.

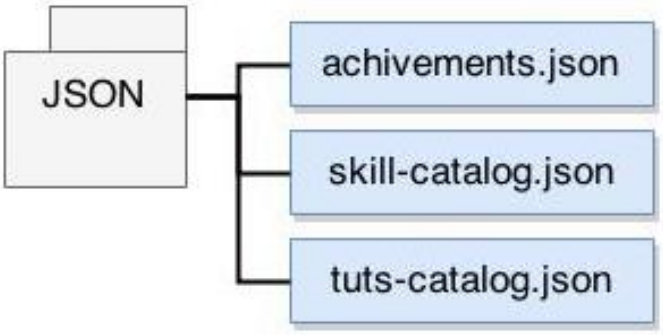

Рисунок 2.5.7 – Структура каталога JSON

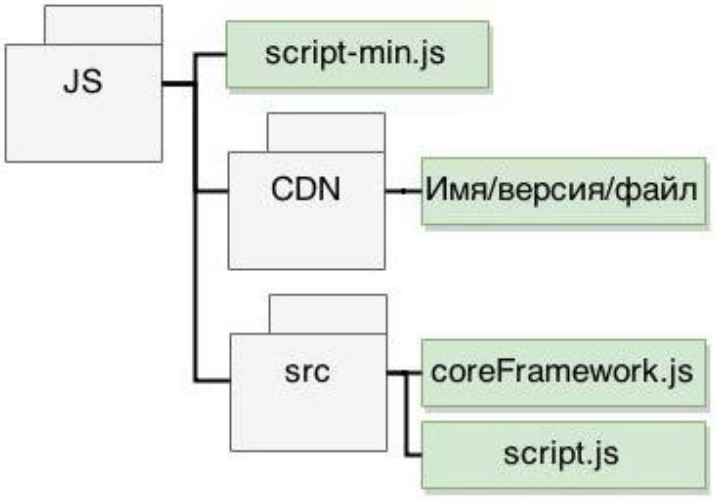

Рисунок 2.5.8 – Структура каталога JS

В данном каталоге содержится папка CDN, в которую помещаются файлы, связанные со сторонними фреймворками и библиотеками, необходимыми пользователю для тех или иных динамичных примеров и инструкций. Файлы структурируются, сортируясь по названию, версии, где уже находятся необходимые файлы. В этом случае предоставляется защищенная альтернатива сторонним CDN репозиториям, а также обеспечивается использование различных версий файлов.

На рис. 2.5.9 представлена структура каталога CSS, структура которого идентична каталогу JS.

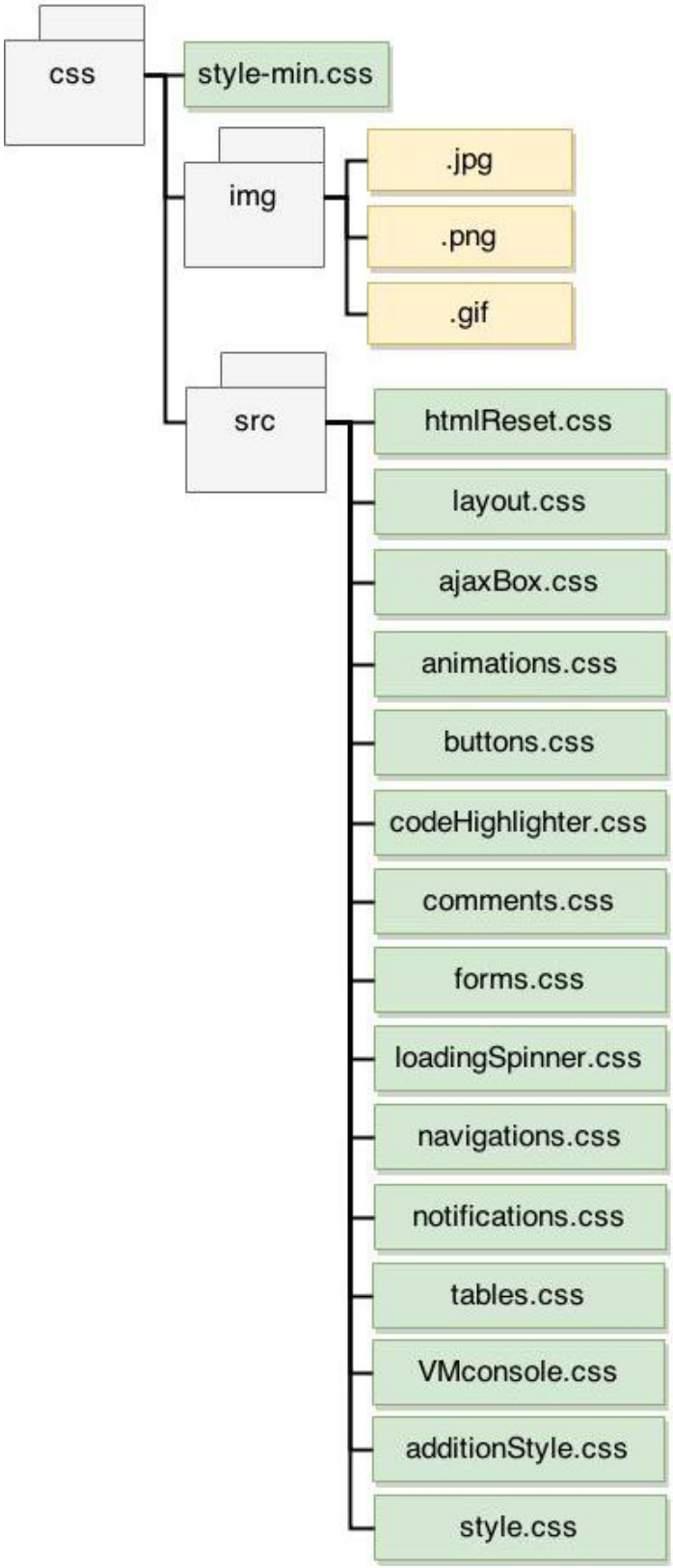

Рисунок 2.5.9 – Структура каталога CSS

2.6 Проектирование базы данных

Разрабатываемая система управления контентом рассчитана на содержание огромного количества пользовательских данных. Ввиду чего спроектирована реляционная база данных.

Логическая модель представлена ниже на рис. 2.6.1.

Для хранения и организации информационных данных спроектировано 14 таблиц:

1) users – содержит персональные данные пользователей;
2) users_session – хранит данные сессии пользователей;
3) groups – хранит названия групп пользователей и данные групповой политики;
4) news – содержит информационный контент раздела новости;
5) pages – хранит выводимую разметку для специализированных страниц веб–документов создаваемых пользователями системы;
6) tips – содержит информационный контент публикаций;
7) comments – содержит комментарии пользователей на разных разделах;
8) projects – содержит информационный контент раздела «проекты»;
9) stats – хранит данные статистики использования системы пользователями;
10) rates – хранит данные рейтингов материалов;
11) polls – содержит необходимые организационные данные для осуществления опросов;
12) polls_choices – содержит варианты ответов для опросов;
13) polls_answers – содержит данные о собранных результатах опросов;
14) user_achivements – хранит персонализированные идентификаторы достижений пользователя.

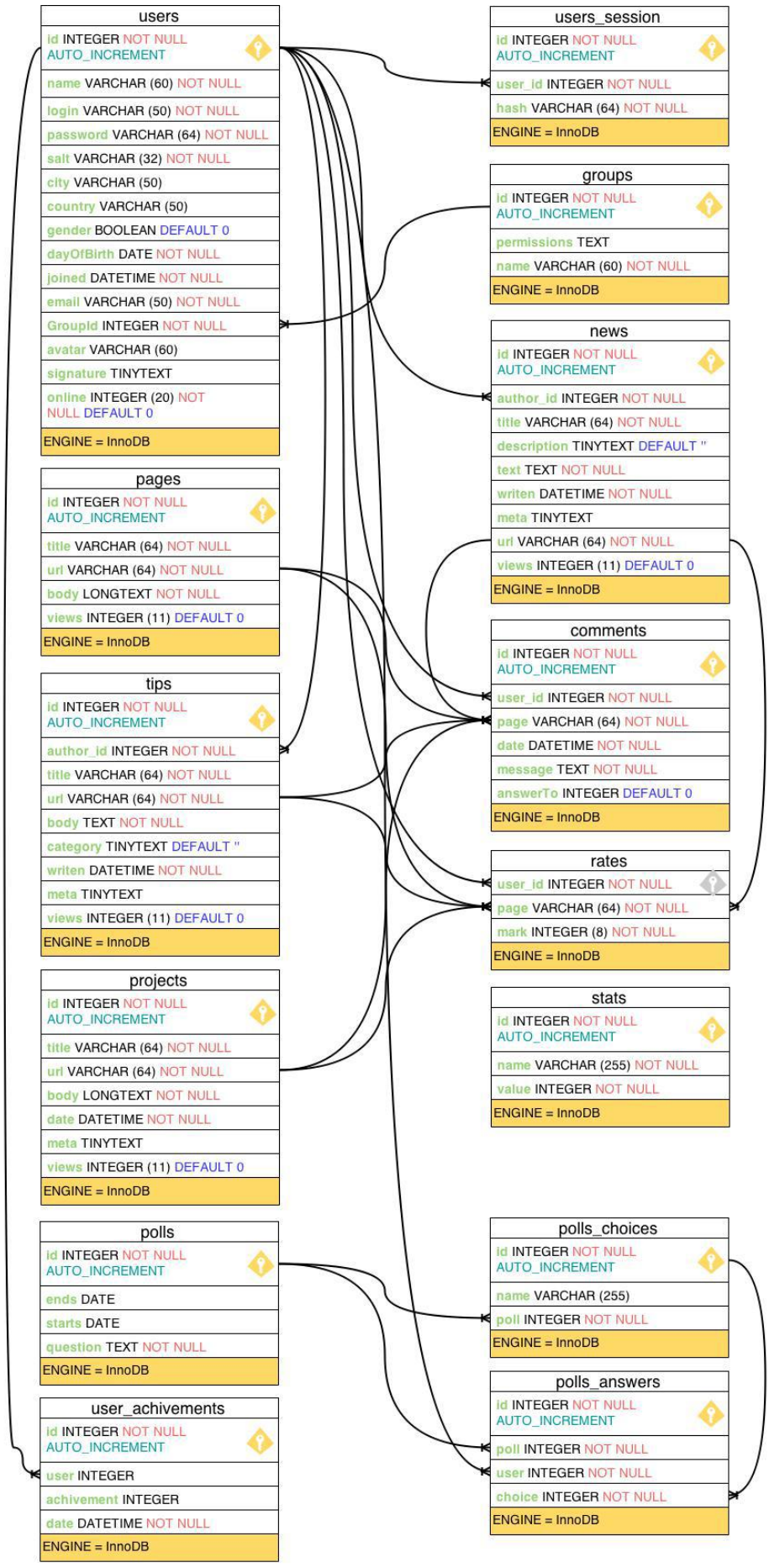

Рисунок 2.6.1 – Логическая модель базы данных

3 Разработка базовых составляющих серверной и клиентской частей прототипа системы управления контентом ресурсов научно–образовательной среды

3.1 Разработка базы данных прототипа

В соответствии с прототипом базы данных, указанным выше в параграфе 2.6, разработан SQL–запрос, представленный ниже. В качестве примера использована кодировка utf8mb4, что соответствует клиентской информационной составляющей для поддержки UTF–8.

```
SET NAMES utf8mb4 COLLATE utf8mb4_unicode_ci;
CREATE TABLE users(
    id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR (60) NOT NULL,
    login VARCHAR (50) NOT NULL,
    password VARCHAR (64) NOT NULL,
    salt VARCHAR (32) NOT NULL,
    city VARCHAR (50),
    country VARCHAR (50),
    gender BOOLEAN DEFAULT "0",
    dayOfBirth Date NOT NULL,
    joined DATETIME NOT NULL,
    email VARCHAR (50) NOT NULL,
    groupId INTEGER NOT NULL,
    avatar VARCHAR (60),
    signature TINYTEXT,
    online INTEGER (20) NOT NULL default "0"
)ENGINE = InnoDB;
CREATE TABLE users_session(
    id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    user_id INTEGER NOT NULL,
```

```
    hash VARCHAR(64) NOT NULL
)ENGINE = InnoDB;
CREATE TABLE groups(
    id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    name VARCHAR (60) NOT NULL,
    permissions TEXT
)ENGINE = InnoDB;
CREATE TABLE news(
    id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    author_id INTEGER NOT NULL,
    title VARCHAR(64) NOT NULL,
    description TINYTEXT DEFAULT '',
    text TEXT NOT NULL,
    writen DATETIME NOT NULL,
    meta TINYTEXT,
    url VARCHAR(64) NOT NULL,
    views INTEGER(11) DEFAULT 0
)ENGINE = InnoDB;
CREATE TABLE comments(
    id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    user_id INTEGER NOT NULL,
    page VARCHAR(64) NOT NULL,
    date DATETIME NOT NULL,
    message TEXT NOT NULL,
    answerTo INTEGER DEFAULT 0
)ENGINE = InnoDB;
CREATE TABLE rates(
    user_id INTEGER NOT NULL,
    page VARCHAR(64) NOT NULL,
    mark INTEGER(8) NOT NULL
```

```
)ENGINE = InnoDB;
CREATE TABLE pages(
        id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
        title VARCHAR(64) NOT NULL,
        url VARCHAR(64) NOT NULL,
        body LONGTEXT NOT NULL,
        views INTEGER(11) DEFAULT 0
)ENGINE = InnoDB;
CREATE TABLE tips(
        id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
        author_id INTEGER NOT NULL,
        title VARCHAR(64) NOT NULL,
        url VARCHAR(64) NOT NULL,
        body TEXT NOT NULL,
        category TINYTEXT DEFAULT '',
        views INTEGER(11) DEFAULT 0,
        meta TINYTEXT,
        writen DATETIME NOT NULL
)ENGINE = InnoDB;
CREATE TABLE projects(
        id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
        title VARCHAR(64),
        url VARCHAR(64) NOT NULL,
        body TEXT NOT NULL,
        meta VARCHAR(64),
        views INTEGER (11) DEFAULT 0,
        date DATETIME NOT NULL
)ENGINE = InnoDB;
CREATE TABLE stats(
        id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
```

```
    name VARCHAR(255) NOT NULL,
    value INTEGER NOT NULL
)ENGINE = InnoDB;
CREATE TABLE polls(
    id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    question TEXT NOT NULL,
    starts DATE,
    ends DATE
)ENGINE = InnoDB;
CREATE TABLE polls_choices(
    id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    poll INTEGER NOT NULL,
    name VARCHAR(255)
)ENGINE = InnoDB;
CREATE TABLE polls_answers(
    id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    user INTEGER NOT NULL,
    poll INTEGER NOT NULL,
    choice INTEGER NOT NULL
)ENGINE = InnoDB;
CREATE TABLE user_achivements(
    id INTEGER NOT NULL AUTO_INCREMENT PRIMARY KEY,
    user INTEGER,
    achievement INTEGER,
    date DATETIME NOT NULL
)ENGINE = InnoDB;
```

3.2 Разработка базовых составляющих серверной части прототипа

Ниже представлено содержание файла .htaccess корневого каталога, обеспечивающего настройку директив веб-сервера Apache для всех файлов и каталогов системы:

```
AddDefaultCharset UTF-8
RewriteEngine on
RewriteRule ^([^/.]+)/?$ $1.php [NC,L]
RewriteRule ^profile/([0-9a-zA-Z]+) profile.php?user=$1 [NC,L]
RewriteRule ^page/([-0-9a-zA-Z]+) page.php?u=$1 [NC,L]
php_flag display_startup_errors on
php_flag display_errors on
php_flag html_errors off
php_flag ignore_repeated_errors off
php_flag ignore_repeated_source off
php_flag report_memleaks on
php_flag track_errors on
php_flag file_uploads on
php_value max_file_uploads 20
php_value upload_max_size 4194304
php_value post_max_size 16777216
php_value docref_root 0
php_value docref_ext 0
php_flag short_open_tag off
DirectoryIndex index.php
Options -Indexes FollowSymLinks
ErrorDocument 404 /404.html
```

Для осуществления процессов инициализации компонентов системы и настроек (описание и возможные параметры настроек представлены в приложении 2) разработан файл инициализации «init.php», следующего содержания:

```
<?php
$wenegred_start = microtime(true);
session_start();
error_reporting(E_ALL & ~E_NOTICE);
$GLOBALS['config'] = array(
	'db' => array(
		'manager' => 'mysql',
		'host' => '127.0.0.1',
		'port' => '3306',
		'login' => 'root',
		'password' => '',
		'name' => 'wenegred',
		'charset' => 'utf8mb4'
	),
	'remember' => array(
		'cookie_name' => 'hash',
		'cookie_expiry' => 2629743
	),
	'session' => array(
		'session_name' => 'user',
		'token_name' => 'token'
	),
	'SMTP' => array(
		'isDebuging' => true,
		'useAuth' => true,
		'host' => 'smtp.gmail.com',
		'port' => 587,
		'username' => 'test@wenegred.ru',
		'password' => 'SuperSecretPassword',
		'security' => 'tls',
```

```
                'from-mail' => 'test@wenegred.ru',
                'debug-level' => 4,
                'log-size' => '1048576',
                'DKIM' => array(
                        'signature' => 'rsa-sha1',
                        'passphrase' => ''
                ),
                'sign' => array(
                        'cert_file' => '',
                        'key_file' => '',
                        'key_pass' => ''
                )
        ),
        'docs' => array(
                'charset' => 'utf-8',
                'home' => 'http://localhost/',
                'url' => (isset($_SERVER['HTTPS']) && $_SERVER['HTTPS'] == 'on' ? 'https' : 'http') . '://' . $_SERVER['HTTP_HOST'] . dirname($_SERVER['PHP_SELF']),
                'folder-path' => getcwd(),
                'log-size' => '1048576',
                'file-crypt' => 'TESTINGKEYFORFILEENCRIPTION',
                'archive-type' => 'zip'
        )
);
spl_autoload_register(function($class){require $_SERVER['DOCUMENT_ROOT'] . 'classes/' . $class . '.php';});
Load::func('sanitize');
setlocale(LC_ALL, 'ru_Ru.' . Config::get('docs/charset'));
date_default_timezone_set('Europe/Moscow');
```

```
if(Cookie::exists(Config::get('remember/cookie_name')) && !Session::exists(Config::get('session/session_name'))){
    $hashCheck = DB::getInstance()->get('*', 'users_session', array('hash', '=', Cookie::get(Config::get('remember/cookie_name')) ));
    if($hashCheck->count()) (new User($hashCheck->first()->user_id))->login();
}
```

Для осуществления функционирования компонентов системы и ее подсистем разработан ряд классов, представленных ниже:

Класс Config предоставляет методы назначения и получения конфигурационных параметров:

```
class Config{
    public static function get($path = null){
        if($path){
            $config = $GLOBALS['config'];
            $path = explode('/', $path);
            foreach($path as $bit)
                if(isset($config[$bit]))
                    $config = $config[$bit];
            return $config;
        }
        return false;
    }
    public static function set($path, $set){
        $config = $GLOBALS['config'];
        $path = explode('/', $path);
        $GLOBALS['config'][$path[0]][$path[1]] = $set;
    }
}
```

Класс Cookie, содержит методы работы с COOKIE:

```
class Cookie{
	public static function exists($name){
		return (isset($_COOKIE[$name])) ? true : false;
	}
	public static function get($name){
		return $_COOKIE[$name];
	}
	public static function put($name, $value, $expiry){
		if(setcookie($name, $value, time() + $expiry, '/'))
			return true;
		return false;
	}
	public static function delete($name){
		self::put($name, '', time() - 1);
	}
}
```

Класс Log, разработан для ведения системных журналов:

```
class Log{
	private static function writeLn($txt){
		$LogFile = Config::get('docs/folder-path') . '/core/system.log';
		$time = date('Y-m-d H:i:s');
		if(filesize($LogFile) >= Config::get('docs/log-size')){
			file_put_contents($LogFile, "\n//----------
----------\n----------Was closed at {$time}\n----------
----------//", FILE_APPEND);
			rename($LogFile, rtrim($LogFile, ".log") . "-before-" .
date('Y-m-d') . ".log");
		}
		$txt = $time . " --> " . $txt . "\n";
		file_put_contents($LogFile, $txt, FILE_APPEND);
```

```
        }
        public static function write($action, $user = ''){
            if(!$user)
                $user = 'Wenegred';
            self::writeLn("IP {$_SERVER["REMOTE_ADDR"]} | {$user} | {$action}");
        }
        public static function error($txt){
            self::writeLn($txt . " <-- ERROR");
        }
    }
```

Класс DB позволяет осуществлять контроль и фундаментальные операции с СУБД:

```
    class DB{
        private static $_instance = null;
        private $_pdo;
        private $_query;
        private $_error = false;
        private $_results;
        private $_count = 0;
        private function __construct(){
            try{
                $this->_pdo = new PDO($this->genDSN(), Config::get('db/login'), Config::get('db/password'));
            }catch(PDOException $e){die($e->getMessage());}
        }
        private function genDSN(){
            $m = Config::get('db/manager');
            $host = Config::get('db/host');
            $port = Config::get('db/port');
```

```
                $base = Config::get('db/name');
                $char = Config::get('db/charset');
                switch($m){
                        case 'pgsql': case 'mysql':
                                return
"{$m}:host={$host}:{$port};dbname={$base};charset={$char}";
                        case 'oci':
                                return
"oci:dbname={$host}:{$port}/{$base};charset={$char}";
                        case 'sqlite':
                                return "sqlite:{$base}";
                        case 'sqlsrv':
                                return
"sqlsrv:Server={$host},{$port};Database={$base};charset={$char}";
                        }
        }
        public static function getInstance(){
                if(!isset(self::$_instance)) self::$_instance = new DB();
                return self::$_instance;
        }
        public function query($sql, $params = array()){
                $this->_error = false;
                if($this->_query = $this->_pdo->prepare($sql)){
                        $x = 1;
                        if(count($params)){
                                foreach($params as $param){
                                        $this->_query->bindValue($x, $param);
                                        $x++;
                                }
                        }
```

```
                    if($this->_query->execute()){
                        $this->_results = $this->_query->fetchAll(PDO::FETCH_OBJ);
                        $this->_count = $this->_query->rowCount();
                    }else{
                        $this->_error = true;
                    }
                }
                return $this;
            }
            private function validOperator($operator){
                return in_array($operator, array('=', '>', '<', '>=', '<='));
            }
            private function action($action, $table, $where = array()){
                if(count($where) === 3){
                    $field = $where[0];
                    $operator = $where[1];
                    $value = $where[2];
                    if($this->validOperator($operator)){
                        $sql = "{$action} FROM {$table} WHERE {$field} {$operator} ?";
                        if(!$this->query($sql, array($value))->error())
                            return $this;
                    }
                }
                return false;
            }
            public function getSorted($what, $table, $where = array(), $sorter, $DESC = false){
                if(count($where) === 3){
```

```
                $field = $where[0];
                $operator = $where[1];
                $value = $where[2];
                if($this->validOperator($operator) && !preg_match('/\s/', $sorter)){
                        $sql = "SELECT {$what} FROM {$table} WHERE {$field} {$operator} ? ORDER BY {$sorter}";
                        if($DESC)
                                $sql .= ' DESC';
                        if(!$this->query($sql, array($value))->error())
                                return $this;
                }
        }
        return false;
    }
    public function getLimited($what, $table, $where = array(), $sorter, $DESC = false, $limit = 10){
        if(count($where) === 3){
                $field = $where[0];
                $operator = $where[1];
                $value = $where[2];
                if($this->validOperator($operator) && !preg_match('/\s/', $sorter)){
                        $sql = "SELECT {$what} FROM {$table} WHERE {$field} {$operator} ? ORDER BY {$sorter}";
                        if($DESC)
                                $sql .= ' DESC';
                        if(is_numeric($limit)){
                                $sql .= ' LIMIT . $limit;
                        }else{
```

```
                                $limit = explode('-', $limit);
                                if(count($limit) === 2)
                                        $sql .= ' LIMIT ' . $limit[0] . ',' .
$limit[1];
                                }
                                if(!$this->query($sql, array($value))->error())
                                        return $this;
                        }
                }
                return false;
        }
        public function getBetween($what, $table, $where = array()){
                if(count($where) === 3){
                        $sql = "SELECT {$what} FROM {$table} WHERE ?
BETWEEN {$where[1]} AND {$where[2]}";
                        if(!$this->query($sql, array($where[0]))->error())
                                return $this;
                }
                return false;
        }
        public function get($what, $table, $where){
                return $this->action('SELECT ' . $what, $table, $where);
        }
        public function insert($table, $fields = array()){
                if(count($fields)){
                        $keys = array_keys($fields);
                        $values = '';
                        $x = 1;
                        foreach($fields as $field){
                                $values .= "?";
```

```
                    if($x < count($fields))
                        $values .= ', ';
                    $x++;
                }
                $sql = "INSERT INTO {$table} (`" . implode('`, `', $keys) . "`) VALUES ({$values})";
                if(!$this->query($sql, $fields)->error()) return true;
            }
            return false;
        }
        public function update($table, $id, $fields){
            $set = '';
            $x = 1;
            foreach($fields as $name => $value){
                $set .= "{$name} = ?";
                if($x < count($fields))
                    $set .= ', ';
                $x++;
            }
            $sql = "UPDATE {$table} SET {$set} WHERE id = {$id}";
            if($this->query($sql, $fields)->error())
                return true;
            return false;
        }
        public function backup(){
            $LogMessage = "Резервное копирование базы данных -->";
            $result = $this->query('SHOW TABLES')->results();
            $return = "USE " . Config::get('db/name') . ";\nSET NAMES " . Config::get('db/charset') . " COLLATE " . Config::get('db/charset') . "_unicode_ci;\n";
```

```
                if($result){
                    foreach($result as $table){
                        $table = $table->{'Tables_in_'.Config::get('db/name')};
                        $return .= 'INSERT INTO '. $table . '(';
                        $fieldsQ = $this->query('SHOW FIELDS FROM ' . $table)->results();
                        $fields = array();
                        foreach($fieldsQ as $value)
                            $fields[] = $value->Field;
                        for($i = 0; $i < count($fields); $i++)
                            $return .= $fields[$i] . ', ';
                        $return = rtrim($return, ', ') . ')VALUES(';
                        $content = $this->get('*', $table, array('1', '=', '1'))->results();
                        foreach($content as $value){
                            for($i = 0; $i < count($fields); $i++)
                                $return .= '"' . str_ireplace('', '\"', str_ireplace('"', "\"", $value->{$fields[$i]})) . '", ';
                            $return = rtrim($return, ', ') . '),\n';
                        }
                        $return = rtrim($return, '),\n') . ');\n';
                    }
                }
                $filename = 'test/'. date("Y-m-d-H-i-s", time()) . '-backup.sql';
                fopen($filename, 'w+');
                file_put_contents($filename, $return);
                fclose($filename);
                Log::write($LogMessage . 'успешно.');
```

```
	}
	public function delete($table, $where){
		return $this->action('DELETE', $table, $where);
	}
	public function results(){
		return $this->_results;
	}
	public function first(){
		return $this->results()[0];
	}
	public function error(){
		return $this->_error;
	}
	public function count(){
		return $this->_count;
	}
}
```

Класс Hash, определяет все процессы хеширования в CMS:

```
class Hash{
	public static function make($string, $salt = ''){
		return hash('sha256', $string . $salt);
	}
	public static function salt($length){
		return hash('md5', mcrypt_create_iv($length));
	}
	public static function unique(){
		return self::make(uniqid());
	}
}
```

Класс Validate, содержит методы валидации вводимых данных и комплекс фильтров:

```
class Validate{
    private $_passed = false;
    private $_errors = array();
    private $_db = null;
    public function __construct(){
        $this->_db = DB::getInstance();
    }
    public function check($source, $items = array()){
        foreach($items as $item => $rules){
            foreach($rules as $rule => $rule_value){
                $value = trim($source[$item]);
                $item = escape($item);
                if($rule === 'required' && empty($value)){
                    $this->addError("требуется ввести {$item}.");
                }else if(!empty($value)){
                    switch($rule){
                        case 'min':
                            if(strlen($value) < $rule_value)
                                $this->addError("{$item} менше требуемых {$rule_value} символов.");
                            break;
                        case 'max':
                            if(strlen($value) > $rule_value)
                                $this->addError("{$item} больше допустимых {$rule_value} символов.");
                            break;
                        case 'matches':
```

```
                                        if($value != $source[$rule_value])
                                            $this->addError("{$rule_value} не совпадает с {$item}.");
                                        break;
                                    case 'unique':
                                        $check = $this->_db->get('*', $rule_value, array($item, '=', $value));
                                        if($check->count())
                                            $this->addError("Пользователь {$value} уже существует.");
                                        break;
                                    case 'email':
                                        if(!filter_var($value, FILTER_VALIDATE_EMAIL))
                                            $this->addError("{$value} не похож на электронную почту.");
                                        break;
                                }
                            }
                        }
                    }
                    if(empty($this->_errors))
                        $this->_passed = true;
                    return $this;
                }
                private function addError($error){
                    $this->_errors[] = $error;
                }
```

```
        public function errors(){
            return $this->_errors;
        }
        public function passed(){
            return $this->_passed;
        }
    }
```

Класс User разработан для осуществления комплексных мероприятий, связанных с персональными данными пользователей как авторизованных, так и представляемых в документе. Он предоставляет методы работы с данными групповой политики:

```
    class User{
        private $_db;
        private $_data;
        private $_sessionName;
        private $_cookieName;
        private $_isLoggedIn;
        public function __construct($user = null){
            $this->_db = DB::getInstance();
            $this->_sessionName = Config::get('session/session_name');
            $this->_cookieName = Config::get('remember/cookie_name');
            if(!$user){
                if(Session::exists($this->_sessionName)){
                    $user = Session::get($this->_sessionName);
                    if($this->find($user)){
                        $this->_isLoggedIn = true;
                        $this->_db->update("users",     $this->data()-
>id, array('online' => date('U') + 50));
                    }
                }
```

```
            }else{
                $this->find($user);
            }
        }
        public function update($fields = array(), $id = null){
            if(!$id && $this->isLoggedIn())
                $id = $this->data()->id;
            if($this->_db->update('users', $id, $fields))
                throw new Exception('Произошла проблема с обновлением данных.');
        }
        public function create($fields = array()){
            if(!$this->_db->insert('users', $fields))
                throw new Exception('Проблема с регистрацией пользователя.');
        }
        public function find($user = null){
            if($user){
                $field = (is_numeric($user)) ? 'id' : 'login';
                $data = $this->_db->get('*', 'users', array($field, '=', $user));
                if($data->count()){
                    $this->_data = $data->first();
                    return true;
                }
            }
            return false;
        }
        public function login($login = null, $password = null, $remember = false){
```

```
                if(!$login && !$password && $this->exists()){
                    Session::put($this->_sessionName, $this->data()->id);
                }else{
                    $user = $this->find($login);
                    if($user){
                        if($this->data()->password === Hash::make($password, $this->data()->salt)){
                            Session::put($this->_sessionName, $this->data()->id);
                            if($remember){
                                $hashCheck = $this->_db->get('*', 'users_session', array('user_id', '=', $this->data()->id));
                                if(!$hashCheck->count()){
                                    $hash = Hash::unique();
                                    $this->_db->insert('users_session', array(
                                        'user_id' => $this->data()->id,
                                        'hash' => $hash
                                    ));
                                }else{
                                    $hash = $hashCheck->first()->hash;
                                }
                                Cookie::put($this->_cookieName, $hash, Config::get('remember/cookie_expiry'));
                            }
                            return true;
                        }
                    }
```

```
            }
            return false;
        }
        public function getPermission($key){
            $group = $this->_db->get('permissions', 'groups', array('id', '=', $this->data()->GroupId));
            if($group->count()){
                $permissions = json_decode($group->first()->permissions, true);
                if(isset($permissions[$key]))
                    return $permissions[$key];
            }
            return false;
        }
        public function hasPermission($key){
            $has = $this->getPermission($key);
            if($has)
                return true;
            return false;
        }
        public function getGroup(){
            $group = $this->_db->get('name', 'groups', array('id', '=', $this->data()->GroupId));
            if($group->count())
                return $group->first()->name;
            return false;
        }
        public function exists(){
            return (!empty($this->_data)) ? true : false;
        }
```

```
        public function logout(){
            $this->_db->delete('users_session', array('user_id', '=', $this->data()->id));
            Session::delete($this->_sessionName);
            Cookie::delete($this->_cookieName);
        }
        public function data(){
            return $this->_data;
        }
        public function isLoggedIn(){
            return $this->_isLoggedIn;
        }
    }
```

Класс Token, разработан для функционирования средств токенизации, к примеру для подтверждения подлинности пользователя осуществлявшего действие.

Класс Session, представляет собой абстрактный пакет методов для работы с пользовательскими сессиями.

Класс Input, содержит методы для работы с пользовательским вводом, обеспечивая должный уровень абстракции.

Класс Load, осуществляет контроль подключаемых функций.

Класс MarkUp представляет собой контроллер HTML разметки.

Класс File, содержит методы для работы с файловой системой.

Класс UploadFile, позволяет автономным подходом реализовывать работу форм пользовательской загрузки файлов. Для генерации формы загрузки файла разработан одноименный компонент uploadFile.

Класс BBcode, предоставляет функционал для преобразования и конвертации сокращенных и конвертированных строковых данных. Данная система осуществляет поддержку дополнительного языка разметки, используемого для формирования сообщений. Теги BB-кода заключаются в

квадратные скобки. В последствии использования данных кодов, система перед отображением страницы, то есть при генерации содержания документа, преобразует BB-коды и осуществляет ряд необходимых операций для их функционала. Данная модель используется в системе не только ради предоставления пользователям возможности использования HTML компонентов, ограниченных для обеспечения безопасности, но и ради упрощенного использования функционала системы. Реализованные в системе теги и получаемые эффекты представлены ниже в табл. 3.2.1.

Таблица 3.2.1

BB-коды реализованные в сиситеме

BB-код	Эффект	Преобразования
[b]Полужирный текст[/b]	Полужирное очертание текста	Полужирный текст
[i]Курсив[/i]	Курсивное очертание текста	Курсив
[u]Подчеркнутый текст[/u]	Подчеркивает текст внутри	<u>Подчеркнутый текст</u>
[s]Зачеркнутый текст[/s]	Зачеркивает текст внутри	<s>Зачеркнутый текст</s>
[size=14]Текст[/size]	Изменяет размер текста внутри, размер можно указывать в px, pt, em или процентах, без указания будет использоваться pt	<span style=’font-size: 14pt’>Текст
[color=red]Цветной текст[/color] или [color=#FF0000]Цветной текст[/color]	Окрашивает текст внутри, можно указывать как web наименования, так и HEX	<span style=’color: #FF0000’>Цветной текст
[sub]Нижний индекс[/sub]	Отображает текст в нижнем индексе	_{Нижний индекс}
[sup]Верхний индекс[/sup]	Отображает текст в верхнем индексе	^{Верхний индекс}
[img]URL[/img] или [image]URL[/image]	Вставляет изображение по URL	<img src=’URL’ alt=’’>
[url]URL[/url] или [url=URL]Текстовое представление[/url]	Преобразует содержимое в ссылку	URL или Текстовое представление

Продолжение табл. 3.2.1

BB-код	Эффект	Преобразования
[q]Цитируемый текст[/q] или [quote]Цитируемый текст[/quote]	Оформляет содержимое как цитату	<blockquote>Цитируемый текст</blockquote>
[code]web код[/code]	Позволяет отображать web код, в исходном виде	<pre><code>web код<code></pre>
[hr]	Вставляет разделительную линию	<hr>
[hN]Заголовок[/hN] или [headingN]Заголовок [/headingN]	Определяет внутренний текст как заголовок, где N – номер уровня	<hN>Заголовок</hN
[p]Текст параграфа[/p] или [paragraph]Текст параграфа[/paragraph]	Определяет внутренний текст как параграф	Текст параграфа
[ul][li]Элемент 1[/li] [li]Элемент 2[/li][/ul] или [unordered_list] [li]Элемент 1[/li] [li]Элемент 2[/li][/unordered_list] или [list][li]Элемент 1[/li] [li]Элемент 2[/li][/list]	Преобразует содержимое как неупорядоченный список, для обозначения элемента списка ([li]) может также использоваться тег [item]	- Элемент 1 - Элемент 2
[ol][li]Элемент 1[/li] [li]Элемент 2[/li][/ol] или [ordered_list] [li]Элемент 1[/li] [li]Элемент 2[/li][/ordered_list]	Преобразует содержимое как упорядоченный список, для обозначения элемента списка ([li]) может также использоваться тег [item]	- Элемент 1 - Элемент 2
[left]Текст[/left]	Выравнивает текст по левому краю	Текст
[right]Текст[/right]	Выравнивает текст по правому краю	<div align=” right”>Текст
[center]Текст[/center]	Выравнивает текст по центру	<div align=” center”>Текст
[justify]Текст[/justify]	Выравнивает текст по ширине	<div align=” justify”>Текст
@ЛогинПользователя	Создает ссылку на профиль указанного пользователя, открывающуюся в новой вкладке	<a href=’Ссылка на профиль’ target=’_blank’ >@ЛогинПользователя<a>

Продолжение табл. 3.2.1

BB-код	Эффект	Преобразования
[comment=N]	Вставляет комментарий пользователя, с оформлением и установленными дополнительными данными. Где N – идентификационный номер комментария в базе данных	См. рис. 3.2.2

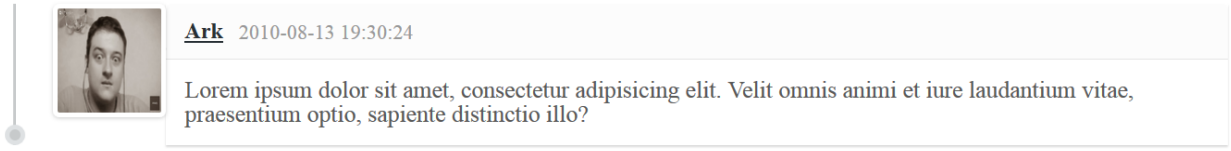

Рисунок 3.2.2 – Пример комментария, получаемого при использовании BB-кода

Класс VM, предоставляет средвства контроля функциональных связей с виртуальной машиной и генерацией конфигурационных запросов.

Класс RSS, осуществляет генерацию RSS лент.

Класс SEO, предоставляет инструментарий для осуществления связи с компонентами Google аналитики, и эмулирует панель SEO с удаленным сбором информации.

Класс Redirect, позволяет осуществлять перенаправления клиента, в абстрактном виде.

Класс Image, содержит средства редактирования файлов-изображений.

Класс News, добавляет абстрактный уровень для работы с новостными блоками.

Класс Comment, создает абстрактный уровень для работы с комментариями пользователей.

Класс Stats, обеспечивает систему средствами ведения статистики о действиях пользователей.

Класс Banner, предоставляет необходимый функционал для средств ротации рекламных блоков и объявлений.

Класс Achivement, содержит комплекс методов для работы с пользовательскими достижениями.

Классы Mail, SMTP и NTLM, совместно представляют собой подсистему работы с электронной почтой, используя разные вариации конфигурационного модуля.

Для сохранения модульной архитектуры, разработанны отдельные, небольшие функциональные компоненты используемые в различных частях системы, представленные в виде функций.

Файл addNews, разработан для формы добавления новостных блоков.

Файл changePassword, разработан для осуществления функции смены пароля пользователя.

Файл logout, служит для вывода пользователя из системы.

Файл register, разработан для осуществления регистрации пользователей.

Файл update, разработан для предоставления пользователям возможности изменения данных своего профиля.

Файл votePoll разработан для обработки функционала участия пользователей в проводимых опросах.

Файл showPollResults разработан для осуществления вывода результатов голосования в проводимом опросе.

Для хранения главного меню, дублируемого на всех страницах, был разработан файл MainMenu.

Для генерации боковой панели, дублируемой на всех страницах, был разработан файл Sidebar.

Для функциональной части административной панели управления, был разработан компонент admin.

Для генерации главной страницы, разработан файл index.

Для осуществления авторизации пользователей разработан компонент login.

Для автоматической сборки отдельных страниц разработан компонент page.

Для генерации персональных страниц пользователей разработан компонент profile.

Для управления и правильной индексации поисковыми системами создан файл robots.txt.

3.3 Разработка базовых составляющих клиентской части прототипа

Графический интерфейс системы активно использует возможности каскадных таблиц стилей, для хранения общих правил созданы раздельные файлы стилей.

HtmlReset – осуществляет общий возврат визуальных особенностей отображения интерфейсов разными браузерами.

Layout – содержит стили используемые для позиционирования и разметки страниц и организации объектов.

AjaxBox – содержит стили оформления диалоговых окон.

Animations – содержит стили для анимационных эффектов.

Buttons – сожержит стили оформления кнопок.

CodeHighlighter – содержит стили для оформления фрагментов кода:

Comments – содержит стили оформления блоков комментариев пользователей.

Forms – содержит стили оформления элементов форм.

LoadingSpinner – содержит стили для индекатора загрузки.

Navigations – содержит стили для панелей навигации.

Notifications – содержит стили для уведомляющих сообщений.

Tables – содержит стили оформления таблиц.

AdditionStyle и style содержат дополнительные стили.

JavaScript фреймворк функции которого используются по всей системе управления контентом, разработан специально для системы и содержится в едином файле.

Для вывода страницы с ошибкой 404 создан файл 404.html.

4 Тестирование и анализ выбранных механизмов в процессе реализации прототипа системы управления контентом ресурсов научно–образовательной среды

4.1 Проверка корректности функционирования прототипа системы

Для осуществления контроля функционирования системы, в реализованном CMS-решении используются внутренние механизмы выдающие сообщения о неполадках.

Поскольку ключевые указатели ошибок генерируются либо вводятся в настройках, на местах возможного представления данных вариаций использованы переменные X, Y, Z. Сообщения выдаваемые пользователю в случае нарушения тех или иных правил групповой политики или осуществления эксплуатационных мероприятий – в данном разделе не указаны. Перечень сообщений об ошибках в работе системы управления контентом:

1) Соединение с СУБД X прервано;
2) Ошибка от СУБД с кодом X;
3) Резервное копирование базы данных X прервано на таблице Y;
4) Произошла проблема с обновлением данных записи X;
5) Нарушение валидации: разрешения X группы Y нарушает разрешение Z;
6) Найденный и сохраненный в буфер пользователь X, не найден;
7) Разрешение X группы Y превышает лимит указанный в конфигурации веб-сервера;
8) Нет доступа к папке X для записи файла Y;
9) Нет доступа к папке X для загрузки файла Y;
10) Запрашиваемый файл X не обнаружен;
11) Директория X не может быть удалена;
12) Заканчивается место на жестком диске (X мб);
13) Запись X с идентификатором Y не найдена, при запросе;

14) Нарушение логики или иерархии в указании прав группы пользователей – возможны неправомерные действия;
15) RSS файл X недоступен;
16) Ошибка при типизации компонента X при интерполяции аргумента Y в методе w_Object;
17) Microsoft.XMLHTTP нарушил работу, операция AjaxBox заблокирована.

Перечень сообщений об ошибках связанных с работой подсистемы электронной почты. Подробные отчеты о работе подсистемы сохраняются в журнале генерируемым системой.

1) X - нет доступа к временной директории сервера;
2) Ошибка SMTP: ошибка авторизации;
3) Ошибка SMTP: не удается подключиться к серверу SMTP;
4) Ошибка SMTP: данные не приняты;
5) Превышен лимит времени запроса (X сек.);
6) Соединение с данным SMTP сервером уже установлено;
7) Соединение: stream_socket_client не доступен, система произвела сброс до fsockopen;
8) На данном этапе аутентификация запрещена;
9) Не поддерживаемый для удаленного SMTP метод аутентификации;
10) Запрос аутентификационного метода X не поддерживается сервером;
11) Срыв соединения с сервером;
12) SMTP ошибка с кодом: X;
13) Пустое тело сообщения;
14) Неизвестный вид кодировки: X;
15) Невозможно выполнить команду: X, Y, Z;
16) Нет доступа к файлу: X, Y, Z;
17) Файловая ошибка: не удается открыть файл: X, Y, Z;

18) Неверный адрес отправителя: X, Y, Z;

19) Ошибка массива получателей: X, Y, Z;

20) Невозможно запустить функцию mail;

21) Неправильный формат email адреса: X;

22) X - почтовый сервер не поддерживается;

23) Необходимо ввести хотя бы один адрес e-mail получателя;

24) Ошибка SMTP: отправка по следующим адресам получателей не удалась: X, Y, Z;

25) Ошибка подписания: X;

26) Ошибка соединения с SMTP-сервером;

27) Ошибка SMTP-сервера: X;

28) Невозможно установить или переустановить переменную: X;

29) Необходимо активировать NTLM функции в php.ini;

30) SMTP ЗАМЕЧАНИЕ: во время проверки соединения получен EOF;

31) Объявлена команда X без установленного соединения;

32) Использовался метод связи HELO. Клиент ничего не знает о расширениях сервера;

33) NTLM аутентификация находится не на фазе старта;

34) Расширение X требуемое NTLM SASL клиентом, недоступно в конфигурации Apache;

35) NTLM аутентификация была завершена с ошибкой от Windows: X;

36) В конфигурации SMTP Windows NT используется неверная фаза NTLM аутентификации.

4.2 Автоматизированное тестирование совместимости компонентов прототипа со стандартизированными параметрами программного обеспечения в реализациях

Автоматизированное тестирование компонентов системы управления контентом проводилось на виртуальных машинах. Базовый функционал тестировался на машине с характеристиками представленными ниже в табл. 4.3.1. Для тестирования отдельного компонента NTLM, для осуществления функционирования подсистемы работы с электронной почтой на операционной системе Windows NT, использовалась виртуальная машина идентичной конфигурации, но с ОС Windows Server 2008 R2.

Таблица 4.3.1

Сведения и характеристики виртуальной машины, используемой для тестирования базового функционала системы

Параметр	Значение
Операционная система	Ubuntu Server
Версия операционной системы	14.04.2 LTS
Разрядность системы	64–bit
Интерфейс взаимодействия	SSH
Процессоры	2
Тактовая частота	3.4 GHz
Чипсет	PIIX3
Кэш процессора	12 Mb
Оперативная память	2048 Mb
Частота оперативной памяти	2400 MHz
VT–x	включено
Использование видеокарты родителя	включено
Объем видеопамяти	6144 Mb
Частота видеопамяти	7 GHz
Разрядность шины видеопамяти	384
Количество GPU	1
Частота ядра GPU	1116 MHz
Версия веб сервера Apache	2.4.9
Версия PHP интерпретатора	5.5.12
Версия MySQL сервера	5.6.17

Поскольку в разработанной системе активно используются алшаритмы хеширования, как для паролей хранимых в базе данных, COOKIE, механизмов токенизации и так далее. Леквидность выбираемого алгоритма для той или иной задачи определяется в соответствии с его

производительностью, в чем необходимо найти баланс. Для этого был разработан скрипт автоматизированного тестирования поддерживаемых системой алгоритмов хеширования. При тестировании, были получены результаты представленные в табл. 4.3.2.

Таблица 4.3.2

Результаты тестирования производительности алгоритмов хеширования в системе, по 1000 на алгоритм

алгоритм	среднее время на hash (мс)	общее время (мс)
adler32	0.0000060000	0.0060000420
fnv132	0.0000060000	0.0060000420
fnv164	0.0000060010	0.0060009956
joaat	0.0000079999	0.0079998970
md4	0.0000080011	0.0080010891
md5	0.0000090001	0.0090000629
tiger128,3	0.0000090001	0.0090000629
tiger192,3	0.0000090001	0.0090000629
tiger160,3	0.0000090010	0.0090010166
tiger160,4	0.0000110002	0.0110001564
sha1	0.0000110009	0.0110008717
tiger128,4	0.0000110009	0.0110008717
tiger192,4	0.0000110009	0.0110008717
crc32b	0.0000140009	0.0140008926
haval224,3	0.0000150008	0.0150008202
sha384	0.0000150011	0.0150010586
crc32	0.0000150011	0.0150010586
haval128,3	0.0000150011	0.0150010586
haval160,3	0.0000150011	0.0150010586
haval192,3	0.0000150011	0.0150010586
haval256,3	0.0000150011	0.0150010586
sha512	0.0000160010	0.0160009861
ripemd128	0.0000160010	0.0160009861
ripemd256	0.0000160010	0.0160009861
haval160,4	0.0000190010	0.0190010071
haval128,4	0.0000200009	0.0200009346
haval224,4	0.0000200009	0.0200009346
haval256,4	0.0000200009	0.0200009346
haval192,4	0.0000200012	0.0200011730
sha224	0.0000210011	0.0210011005
sha256	0.0000220010	0.0220010281
ripemd160	0.0000220010	0.0220010281
ripemd320	0.0000220010	0.0220010281
haval192,5	0.0000230012	0.0230011940
haval160,5	0.0000240009	0.0240008831

haval256,5	0.0000240011	0.0240011215
haval224,5	0.0000240018	0.0240018368
haval128,5	0.0000240021	0.0240020752
алгоритм	среднее время на hash (мс)	общее время (мс)
whirlpool	0.0000400028	0.0400028229
gost	0.0000630031	0.0630030632
snefru256	0.0001350079	0.1350078583
snefru	0.0001350081	0.1350080967
md2	0.0003620210	0.3620209694

Результаты проверки функционирования компонентов TLS, используемых системой управления контентом средствами шифрования файлов и электронных писем (в вариации использования TLS методов шифрования содержимого или сертификации SSL), представлены в табл. 4.3.2.

Таблица 4.3.2

Результаты тестирования совместимости TLS компонентов

Значение регистра	Описание	TLS-OK	Указатель
0	Не назначено		
1	rsa_sign	Да	[RFC5246]
2	dss_sign	Да	[RFC5246]
3	rsa_fixed_dh	Да	[RFC5246]
4	dss_fixed_dh	Да	[RFC5246]
5	rsa_ephemeral_dh_RESERVED	Да	[RFC5246]
6	dss_ephemeral_dh_RESERVED	Да	[RFC5246]
7-19	Не назначено		
20	fortezza_dms_RESERVED	Да	[RFC5246]
21	alert	Да	[RFC5246]
22	handshake	Y	[RFC5246]
23	application_data	Y	[RFC5246]
24	heartbeat	Y	[RFC6520]
22	handshake	Y	[RFC5246]
25-63	Не назначено		
64	ecdsa_sign	Да	[RFC4492]
65	rsa_fixed_ecdh	Да	[RFC4492]
66	ecdsa_fixed_ecdh	Да	[RFC4492]
67-223	Не назначено		
224-255	Зарезервированно		[RFC5246]

ЗАКЛЮЧЕНИЕ

Разработанная система управления контентом ресурсов научно–образовательной среды отвечает современным требованиям информационной безопасности и может быть использована в проектах, реализованных в глобальной паутине.

Все функциональные аспекты системы реализованы в соответствии с предъявляемыми требованиями, описанными в техническом задании. Однако столь массивный и динамичный объект разработки требует дальнейшего тестирования в долговременных полевых условиях, использующих весь потенциал системы.

Пользовательский интерфейс и анимационный компонент еще далеки от задуманного и будут дорабатываться, что в рамках проекта не займет много времени учитывая созданную базу для реализаций. Но их разработка не входит в рамки системы управления контентом, а относится к сфере веб–дизайна и потому не рассматривается в данной работе.

БИБЛИОГРАФИЧЕСКИЙ СПИСОК

1. Интернет портал Рейтинг Рунета [Электронный ресурс] // Рейтинг коммерческих систем управления контентом. – 2014. – Режим доступа: http://www.ratingruneta.ru/cms/commercial/, свободный (дата обращения: 23.03.2015).
2. Интернет портал Рейтинг Рунета [Электронный ресурс] // Рейтинг свободно распространяемых систем управления контентом. – 2014. Режим доступа: http://www.ratingruneta.ru/cms/opensource/, свободный (дата обращения: 23.03.2015).
3. Интернет энциклопедия Wikipedia [Электронный ресурс]. – Режим доступа: http://wikipedia.org/, свободный (дата обращения: 24.03.2015).
4. Интернет портал The Apache Software Foundation [Электронный ресурс]. – Режим доступа: https://www.apache.org/, свободный (дата обращения: 25.03.2015).
5. Интернет портал Microsoft IIS [Электронный ресурс]. – Режим доступа: http://www.iis.net/, свободный (дата обращения: 25.03.2015).
6. Интернет портал Netcraft [Электронный ресурс]. – Режим доступа: http://www.netcraft.com, свободный (дата обращения: 26.03.2015).
7. Boiko B. Content Management Bible (2nd Edition) [Текст]. John Wiley & Sons. Ноябрь 2004.
8. Addey D. Content Management Systems [Текст]. Glasshaus. 2002 г.
9. Boag P. 10 Things To Consider When Choosing The Perfect CMS // Smashing magazine. – Март 2009. – P. 32–45.
10. Petr Passinger. 15 Crucial Features when choosing a Content Management System for an Enterprise Website [Текст]. Kentico Devnet. Апрель 2012.
11. SQRL – Secure Quick Reliable Login [Текст]. – Введ. 2014–12. – Gibson Research Corporation, 2014.

12. Интернет портал Wordpress [Электронный ресурс]. – Режим доступа: https://wordpress.org/, свободный (дата обращения: 1.04.2015).
13. Интернет портал 1С–Битрикс [Электронный ресурс]. – Режим доступа: https://www.1c–bitrix.ru/, свободный (дата обращения: 2.04.2015).
14. Интернет портал Drupal [Электронный ресурс]. – Режим доступа: https://drupal.org/, свободный (дата обращения: 2.04.2015).
15. Интернет портал Joomla [Электронный ресурс]. – Режим доступа: https://joomla.org/, свободный (дата обращения: 2.04.2015).
16. Двуличанская Н.Н. Интерактивные методы обучения как средство формирования ключевых компетенций [Текст] // Наука в образовании : электрон. науч.–тех. изд. 2011 4 апр.
17. Есиков А.В. Сравнительный анализ систем управления контентом (Content Management System – CMS) [Текст] // Перспективы развития информационных технологий : науч.–тех. номер 13 изд. 2013 с. 62–65.
18. Моисеев А. Совершенствуем технологию CMS [Текст] // Системный администратор : науч.–тех. номер 8 (33) изд. 2005 с. 82–84.
19. Беданоков М.К. Использование CRM–систем в маркетинговой деятельности автотранспортных предприятий [Текст] // Новые технологии : науч.–тех. номер 3 изд. 2007 с. 122–125.
20. Ясавнина С. Н. Использование CMS и LMS для подготовки аудиторных учебных занятий [Текст] // Грани познания : науч.–тех. номер 5 (25) (25) изд. 2013 с. 79–82.
21. ГОСТ 2.105–95. Общие требования к текстовым документам [Текст]. – Взамен ГОСТ 2.105–79, ГОСТ 2.906 –71; введ. 1995 –04–24. – Минск : Межгос. совет по стандартизации, метрологии и сертификации; М. : Изд–во стандартов, 2005–09.
22. ГОСТ 7.1–2003. Библиографическая запись. Библиографическое описание. Общие требования и правила составления [Текст]. –

Взамен ГОСТ 7.1–84, ГОСТ 7.16–79, ГОСТ 7.18–79, ГОСТ 7.34–81, ГОСТ 7.40–82; введ. 2003–07–02. – Москва : ИПК Издательство стандартов; М. : Изд–во стандартов, 2004–07–01.

23. ГОСТ 34.602–89. Информационная технология. Комплекс стандартов на автоматизированные системы. Техническое задание на создание автоматизированной системы [Текст]. – Введ. 1990–01–01. – Москва : ИПК Издательство стандартов; М. : Изд–во стандартов, 2004.
24. ГОСТ 24.701–86. Надежность автоматизированных систем управления. Основные положения [Текст]. – Взамен ГОСТ 24.701–83; введ. 1986–03–31. – Москва : Министерство приборостроения, средств автоматизации и систем управления; М. : Гос. комитет СССР по стандартам, 1987–07–01.
25. ГОСТ 15150–69. Машины, приборы и другие технические изделия. Исполнения для различных климатических районов. Категории, условия эксплуатации, хранения и транспортирования в части воздействия климатических факторов внешней среды [Текст]. – Введ. 1969–12–29. – Межгосударственный совет по стандартизации, метрологии и сертификации; М. : Гос. комитет СССР по стандартам, 1999–05–28.
26. ГОСТ 21958–76. Система "Человек–машина". Зал и кабины операторов. Взаимное расположение рабочих мест. Общие эргономические требования [Текст]. – Введ. 1976–06–23. – Москва : Государственный комитет стандартов совета министров СССР; М. : Гос. комитет СССР по стандартам, 1977–07–01.
27. ГОСТ 12.1.004–91. ССБТ. Пожарная безопасность. Общие требования [Текст]. – Взамен ГОСТ 12.1.004–85; введ. 1991–06–14. – Москва : Гос. комитет СССР по управлению качеством продукции и стандартам, 2006–09.

28. ГОСТ 12.2.003–91. ССБТ. Оборудование производственное. Общие требования безопасности [Текст]. – Взамен ГОСТ 12.2.003–74; введ. 1991–06–06. – Москва : Гос. комитет СССР по управлению качеством продукции и стандартам, Советом всеобщей конфедерации профессиональных союзов СССР, 2006–12.
29. ГОСТ Р 50571.22–2000. Электроустановки зданий. Часть 7. Требования к специальным электроустановкам. Раздел 707. Заземление оборудования обработки информации [Текст]. – Введ. 2000–12–18. – Всероссийский научно–исследовательский институт электрификации сельского хозяйства, Всероссийский научно–исследовательский институт стандартизации и сертификации в машиностроении, 2002–01–01.
30. ГОСТ 21552–84. Средства вычислительной техники. Общие технические требования, приемка, методы испытаний, маркировка, упаковка, транспортирование и хранение [Текст]. – Взамен ГОСТ 21552–76; введ. 1986–01–01. – Москва : Гос. комитет СССР по стандартам, 2010–04–19.
31. ГОСТ 34.201–89. Виды, комплектность и обозначения документов при создании автоматизированных систем [Текст]. – Взамен ГОСТ 24.101–80, ГОСТ 24.102–80, РД 50–617–86; введ. 1989–03–24. – Москва : Гос. комитет СССР по стандартам, 1990–01–01.
32. ГОСТ 19.001–77. Единая система программной документации [Текст]. – Взамен ГОСТ 19781–83, ГОСТ 19.004–80; введ. 1990–08–27. – Москва : Гос. комитет СССР по управлению качеством продукции и стандартам, 2005–04–01.
33. ГОСТ 19.101–77. Единая система программной документации. Виды программных документов [Текст]. – Введ. 1977–05–20. – Москва : Гос. комитет стандартов Совета Министров СССР, 1980–01–01.
34. ГОСТ 19.503–79. Руководство системного программиста. Требования к содержанию и оформлению [Текст]. – Введ. 1979–01–

12. – Москва : Гос. комитет стандартов Совета Министров СССР, 1980–01–01.
35. Bodo Moller, Thai Duong, Krzysztof Kotowicz . This POODLE Bites: Exploiting The SSL 3.0 Fallback [Текст] / Google Security Advisory // – 2014–09.
36. RFC 6838. Media Type Specifications and Registration Procedures [Текст]. – Взамен RFC 4288, RFC 2048; введ. 2013–01. – Internet Engineering Task Force (IETF), Oracle, AT&T Laboratories, 2013.
37. RFC 6376. DomainKeys Identified Mail (DKIM) Signatures [Текст]. – Взамен RFC 5672, RFC 4871, RFC 4870; введ. 2011–09. – Internet Engineering Task Force (IETF), Brandenburg InternetWorking, AT&T Laboratories, Cloudmark, 2011.
38. RFC 5863. DomainKeys Identified Mail (DKIM) Development, Deployment, and Operations [Текст]. – Введ. 2010–05. – Internet Engineering Task Force (IETF), AT&T Laboratories, Default Deny Security, Brandenburg InternetWorking, 2010.
39. RFC 5585. DomainKeys Identified Mail (DKIM) Service Overview [Текст]. – Введ. 2009–06. – Network Working Group, AT&T Laboratories, Brandenburg InternetWorking, Default Deny Security, 2009.
40. RFC 5322. Internet Message Format [Текст]. – Взамен RFC 2822; введ. 2008–10. – Network Working Group, Qualcomm Inc., 2008.
41. RFC 4855. Media Type Registration of RTP Payload Formats [Текст]. – Взамен RFC 3555; введ. 2007–02. – Network Working Group, Packet Design Inc., 2007.
42. RFC 4422. Simple Authentication and Security Layer (SASL) [Текст]. – Взамен RFC 2222; введ. 2006–06. – Network Working Group, Isode Limited, OpenLDAP Foundation, 2006.
43. RFC 4289. Multipurpose Internet Mail Extensions (MIME) Registration Procedures [Текст]. – Взамен RFC 2048; введ. 2005–12. – Network Working Group, Sun Microsystems, 2005.

44. RFC 4021. Registration of Mail and MIME Header Fields [Текст]. – Введ. 2005–03. – Network Working Group, University of Oxford, Stockholm University/KT, 2005.
45. RFC 2822. IETF Tools – Internet Message Format [Текст]. – Взамен RFC 822; введ. 2001–04. – Network Working Group, Qualcomm Inc., 2001.
46. RFC 2821. Simple Mail Transfer Protocol [Текст]. – Взамен RFC 821, RFC 974, RFC 1869; введ. 2001–04. – Network Working Group, AT&T Laboratories, 2001.
47. RFC 2616. Hypertext Transfer Protocol – HTTP/1.1 [Текст]. – Взамен RFC 2068; введ. 1999–06. – Network Working Group, UC Irvine, Compaq/W3C, Compaq, W3C/MIT, Xerox, Microsoft, 1999.
48. RFC 2392. Content–ID and Message–ID Uniform Resource Locators [Текст]. – Взамен RFC 2111; введ. 1998–08. – Network Working Group, 1998.
49. RFC 2183. Communicating Presentation Information in Internet Messages: The Content–Disposition Header Field [Текст]. – Введ. 1997–08. – Network Working Group, New Century Systems, Qualcomm Inc., University of Tennessee, 1997.
50. RFC 2047. Multipurpose Internet Mail Extensions: Message Header Extensions for Non–ASCII Text [Текст]. – Взамен RFC 1521, RFC 1522, RFC 1590; введ. 1996–11. – Network Working Group, University of Tennessee, 1996.
51. RFC 2046. Multipurpose Internet Mail Extensions: Media Types [Текст]. – Взамен RFC 1521, RFC 1522, RFC 1590; введ. 1996–11. – Network Working Group, Innosoft, First Virtual, 1996.
52. RFC 2045. Multipurpose Internet Mail Extensions: Format of Internet Message Bodies [Текст]. – Взамен RFC 1521, RFC 1522, RFC 1590; введ. 1996–11. – Network Working Group, Innosoft, First Virtual, 1996.

53. RFC 1341. Multipurpose Internet Mail Extensions: Mechanisms for Specifying and Describing the Format of Internet Message Bodies [Текст]. – Введ. 1992–06. – Network Working Group, Bellcore, Innosoft, 1992.
54. Louis Davidson, Jessica M. Moss. Pro SQL Server 2012 Relational Database Design and Implementation. Apress. 2013.
55. Jim DeMarco. Pro Excel 2013 VBA. Apress. 2014.
56. Danny Brian. The Definitive Guide to Berkeley DB XML. Apress. Август 2006.
57. Научная электронная библиотека [Электронный ресурс]. – Режим доступа http://elibrary.ru/, свободный (дата обращения: 6.04.2015).