

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет

вычислительной техники

Кафедра

«Вычислительная техника»

Направление подготовки: 09.03.01 «Информатика и вычислительная техника»

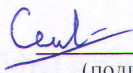
Профиль подготовки: «Вычислительные машины, комплексы, системы и сети»

БАКАЛАВРСКАЯ РАБОТА

на тему

СИСТЕМА ПРИЕМА И ПЕРЕДАЧИ
ДОКУМЕНТОВ НА ОСНОВЕ ТЕХНОЛОГИИ
CMS DRUPAL

Студент

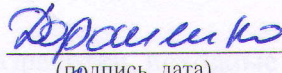


(подпись, дата)

Семозин Никита Романович

(ФИО полностью)

Руководитель

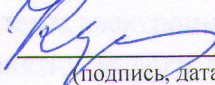


(подпись, дата)

Н.В. Дроздович

(фамилия, инициалы)

Нормоконтролёр

 22.06.19

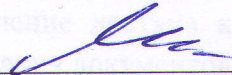
(подпись, дата)

А.В. Кучин

(фамилия, инициалы)

Работа допущена к защите (протокол зас. кафедры от 31.05.2019 г. № 38-09-14)

Заведующий кафедрой



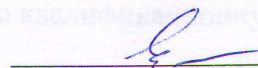
(подпись)

М.А. Митрохин

(фамилия, инициалы)

Работа защищена с отметкой отлично (протокол заседания ГЭК от 24.06.2019 г. № 10)

Секретарь ГЭК



(подпись)

Е.И. Гурин

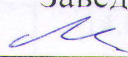
(фамилия, инициалы)

Пенза 2019

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ПЕНЗЕНСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»

Факультет
вычислительной техники

Кафедра
«Вычислительная техника»

«Утверждаю»
Заведующий кафедрой
 М.А. Митрохин
«23» апрель 20 19 г.

**ЗАДАНИЕ
НА БАКАЛАВРСКУЮ РАБОТУ**

1. Студент Семахин Никита Романович гр. 15BB1 факультета вычислительной техники.
2. Тема работы: "Система приема и передачи документов на основе технологии CMS Drupal".

Тема утверждена приказом ПГУ № 554/0 от "23" "апрель" 20 19 г.

3. Руководитель бакалаврской работы к.т.н., доцент кафедры ВТ Дорошенко И.Н.

4. Задание на работу (назначение разработки, исходные данные и т.п.).

Стоит задача разработать сайт системы электронного хранения документации для небольшой частной организации или предприятия малого бизнеса. Назначение разработки – система предназначена для хранения и управления документацией. Разрабатываемая система должна иметь удобный пользовательский интерфейс, поддерживать вход на сайт зарегистрированных пользователей, настройку и добавление новых документов, показ карточки документа, разграничение доступа к документам и обеспечивать возможность создания/изменения/редактирования документов.

5. Перечень подлежащих разработке вопросов.

Требуется определить основные требования к сайту системы, произвести анализ предметной области, изучить основные методы и средства разработки системы, разработать программные модули, обеспечивающие работу системы. Произвести отладку и тестирование, оформить выпускную квалификационную работу.

6. Календарный график выполнения работы

№ п/п	Наименование этапов работы	Объем работы	Срок выполнения	Подпись руководителя
1.	Анализ предметной области	10	07.05.19	Торосиенко
2.	Определение свойств системы	5	12.05.19	Торосиенко
3.	Изучение методов и средств разработки системы	10	12.05.19	Торосиенко
4.	Разработка основных модулей	25	14.05.19	Торосиенко
5.	Отладка и тестирование	20	28.05.19	Торосиенко
6.	Оформление выпускной работы	30	04.06.19	Торосиенко

Дата выдачи задания " 17 " апреля 20 19 г.

Руководитель бакалаврской работы Торосиенко
(подпись, дата)

Задание к исполнению принял студент Сев
(подпись, дата)

Работу к защите допустить
Декан факультета Ст
(подпись, дата)

РЕФЕРАТ

Бакалаврская работа содержит 97 страниц, 26 рисунков, 5 источников, 1 приложение.

Структура работы включает в себя введение, 3 главы, заключение, список использованной литературы и приложение.

Цель работы: разработать сайт системы электронного хранилища документов организации на базе системы управления содержимым Drupal.

Достижение поставленной цели требует решения следующих задач:

- 1) формирование требований к проектируемой системе и определение её функциональных возможностей;
- 2) анализ возможности использования системы управления содержимым Drupal в качестве средства для построения информационных систем и сайтов различной сложности;
- 3) определение основных принципов работы с Drupal и её встроенными средствами по управлению сайтом;
- 4) овладение навыками по созданию программных модулей для системы Drupal;
- 5) разработка необходимых программных модулей, расширяющих базовый функционал системы Drupal до необходимого уровня.

В ходе проделанной работы были изучены основные методы работы с системой управления содержимым Drupal. Были получены навыки как в управлении сайтом с помощью встроенных средств системы, так и в создании собственных программных модулей, расширяющих базовый функционал Drupal.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ	7
1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ	10
1.1 Виды систем электронного документооборота	11
1.1.1 Классификация СЭД по функционалу и решаемым задачам	11
1.1.2 Классификация по типу используемой технологии	4
1.2 Обзор существующих систем документооборота	5
1.3 Требования к проектируемой системе	8
2 СРЕДСТВА РАЗРАБОТКИ САЙТА	11
2.1 Выбор решения	13
2.2 CMS Drupal	13
2.3 Язык программирования PHP	16
2.4 Шаблонизатор TWIG	17
3 РАЗРАБОТКА САЙТА СИСТЕМЫ	19
3.1 Базовые возможности системы	21
3.1.1 Управление контентом	21
3.1.2 Меню	22
3.1.3 Пользователи	22
3.1.4 Темы	23
3.2 Требования к сайту	23
3.3 Разработка типа контента для хранения документов	24
3.4 Определение всех необходимых форм для отображения документов	30
3.5 Разграничение доступа к документам	35
3.6 Работа с сайтом	38
ЗАКЛЮЧЕНИЕ	42
ПРИЛОЖЕНИЕ А: ЛИСТИНГИ ПРОГРАММ	45
Приложение А.1 – Файл «DocumentEntity.php	45
Приложение А.2 – Файл «DocumentEntityType.php	49
Приложение А.3 – Файл «DocumentFormControllerClass.php	50
Приложение А.4 – Файл «DocumentOwnEntityControllerClass.php	53

Приложение А.5 – Файл «DocumentEntityForm.php.....	54
Приложение А.6 – Файл «DocumentEntityPermissionForm.php	56
Приложение А.7 – Файл «DocumentEntityTypeForm.php.....	59
Приложение А.8 – Файл «DocumentEntityListForm.php.....	60
Приложение А.9 – Файл «DocumentCategoriesBlock.php.....	64
Приложение А.10 – Файл «AdminOptionsBlock.php	65
Приложение А.11 – Файл «CabinetBlock.php	66
Приложение А.8 – Файл «DocumentRouteSubscriberBlock.php	67
Приложение А.12 – Файл «DocumentEntityAccessControlHandler.php.....	68
Приложение А.13 – Файл «DocumentEntityPermissionGenerator.php	70
Приложение А.14 – Файл «DocumentEntityListBuilder.php	71
Приложение А.15 – Файл «block.block.administrator.yml.....	72
Приложение А.16 – Файл «block.block.document_categories_block.yml.....	73
Приложение А.17 – Файл «block.block.cabinet.yml	73
Приложение А.18 – Файл «document_module.module	74
ПРИЛОЖЕНИЕ Б: ПРЕЗЕНТАЦИЯ.....	76

ВВЕДЕНИЕ

Период конца XX века для мировой истории характерен высокими темпами технологического прогресса и вступлением человечества в эпоху информационного общества. Мир значительно преобразился. Изменения произошли почти во всех научных, политических и социально-экономических сферах жизни.

Исключением не стала и сфера информационных технологий. Именно здесь произошли самые заметные и глобальные изменения. Список этих изменений очень широк и он не ограничивается только появлением интернета.

Следствием скачка в развитии информационных технологий стал бурный рост общего объема информации, накапливаемого миром. Больше всего эта тенденция стала характерна для сфер бизнеса, производства и торговли. Бумажный документооборот, особенно для крупных предприятий, стал настолько убыточен и неэффективен, что их руководители начали искать альтернативу. Крупные организации столкнулись с проблемой невозможности эффективно работать с информацией старыми «бумажными» методами. Возникла необходимость создания современных способов хранения и передачи данных. Ключевое значение стала играть скорость обмена информацией.

Старые методы показали свою неэффективность по нескольким причинам.

Во-первых, бумажные носители и архивы очень ограничены в объеме информации, которые они способны хранить. Более того, любое обращение к документации, хранящейся подобным образом, требует от человека много времени. Также под большим вопросом стоит надежность и безопасность такого хранилища.

Во-вторых, для современных организаций обременительно расходование огромного количества бумаги на содержание и обновление

архива, да и информация, хранящаяся на таких носителях, менее динамична, чем электронная.

В-третьих, современный ритм жизни и уровень технологий не позволяют человеку проводить много времени в ожидании нужной ему информации. Операции получения и отправки данных должны происходить мгновенно и вне зависимости от расстояния (хорошим примером является электронная почта).

Многочисленные потоки корреспонденции, договоров, заявок и отчетов давно стали неотъемлемой частью любой государственной или частной организации и требуют регулярной обработки и упорядочения. Данную проблему можно решить только путем использования современных электронных информационных систем, а конкретно – информационных систем электронного хранения данных. Их неоспоримыми преимуществами являются мобильность, гибкость и масштабируемость. Настройка и развертывание таких систем хоть и требует финансовых затрат и грамотных специалистов, но в масштабах крупных организаций и общественных учреждений это лишь небольшие издержки.

Концепция безбумажной технологии хранения документов имеет за плечами уже не одно десятилетие, но, несмотря на это, сейчас можно говорить лишь о ее частичной реализации. С одной стороны, ограничителем здесь выступают привычки и техническая грамотность людей, с другой – целесообразность использования. Для большинства крупных и средних компаний использование электронных информационных систем уже давно является обыденностью, но для малого бизнеса такая практика может быть в новинку. Любой организации на определенных этапах ее существования может потребоваться подобная система. Она не обязательно должна быть сложной и дорогой, тут все зависит от конкретных требований заказчиков. В одном случае может потребоваться сложная корпоративная система, а в другом – централизованное хранилище документов, управление которым осуществляет несколько человек. От того, насколько удачно в каждом

конкретном случае будет подобрана система, зависит успех деятельности организации в целом.

На основании сказанного можно заключить, что выбранная тема для дипломного проекта является актуальной и направлена на решение существующих вопросов электронного документооборота.

1 АНАЛИЗ ПРЕДМЕТНОЙ ОБЛАСТИ

Работа современного учреждения, предприятия, организации малого или среднего бизнеса базируется на использовании специальных информационных систем, направленных на повышение эффективности работы персонала с документацией и различного вида отчетностью.

Такие системы, а также системы со схожим функционалом, называют системами электронного документооборота (СЭД). В 70-х годах прошлого века эти системы использовались только крупными компаниями. Их проектирование шло на заказ, а процесс внедрения был дорог и трудоемок. Сейчас ситуация в корне изменилась: на рынке существует масса программных продуктов разного профиля, сложности и стоимости. Широкий выбор позволяет подобрать решение с оптимальным соотношением цены и качества.

Изначально системы электронного документооборота автоматизировали лишь классическое делопроизводство, но к настоящему времени спектр решаемых ими задач значительно расширился. Главное их назначение – это хранение электронных документов, а также работа с ними (редактирование, просмотр, поиск по содержимому и атрибутам и т.д.). Системы электронного документооборота сегодня работают с корреспонденцией, распорядительными и внутренними документами, договорами и проектной документацией. При помощи СЭД решается множество прикладных задач: управление проектами, интеграция с бухгалтерией, работа с клиентами, автоматизация формирования документов. Таким образом, все внутренние процессы, связанные с документооборотом, управляются посредством СЭД.

К полезным функциям также относятся: отслеживание изменений в документах и сроках их выполнения, движение документации по организации, управление бизнес-процессами. Часто бывает так, что архитектура системы повторяет организационно-штатную структуру

предприятия, а правила, по которым работает система, реализуют жесткое разграничение доступа пользователей к документам и функциям в зависимости от их категории.

1.1 Виды систем электронного документооборота

Приведенная ниже классификация систем электронного документооборота не единственная и является условной.

1.1.1 Классификация СЭД по функционалу и решаемым задачам

– Системы делопроизводства. Они же – автоматизированные системы делопроизводства (АСД). Подразделяются на СЭД, ориентированные на автоматизацию бизнес процессов (business-process EDM) и корпоративные СЭД (enterprise-centric EDM). Такие системы являются неотъемлемой частью любого среднего или крупного предприятия, поэтому они и самые многочисленные на рынке. Используются в организациях с жестко формализованными правилами документооборота и вертикальным управлением независимо от их размера, формы собственности и рода деятельности. В своей общей массе решения этого класса не ориентированы на применение в какой-то узкой отрасли или сфере, т.к. принципы организации делопроизводства везде схожи. Основное предназначение АСД состоит в обеспечении корпоративной инфраструктуры и поддержки документа на всех этапах его жизненного цикла внутри организации. Система устанавливает маршрут движения для документа, обусловленный внутренней структурой предприятия. Дополнительные задачи, решаемые данным типом систем характерны и для всех других СЭД. Это, например, организация электронного архива и поддержка групповой работы над документами. АСД системы являются прямыми наследниками концепций бумажного делопроизводства.

– Электронные архивы. Самый простой вид электронных информационных систем, как в плане создания, так и в плане их использования. Хотя электронные архивы и являются пунктом этой классификации, обычно архив не существует как обособленная система, а является частью СЭД. Электронный архив нужен для безопасного и надежного хранения документов и представляет собой информационную систему, призванную обеспечить структурированное хранение электронных документов. Внедрение архива необходимо, если стоит задача долговременного хранения данных. Эта простая мера позволяет снизить риски, связанные с потерей или порчей документов на бумажных носителях, а также повысить общий уровень информационной безопасности предприятия. Причины внедрения могут быть разными. Это, например, ликвидация бумажного архива, издержки, на содержание которого себя не оправдывают или же просто необходимость создания более вместительного хранилища. В простейшем случае информационная система архива должна выполнять следующие функции: перевод в электронный вид уже существующих документов, удобный и быстрый поиск, разграничение прав доступа к документам, удобный и интуитивно понятный интерфейс.

На рисунке 1 и рисунке 2 изображены примеры интерфейсов таких систем.

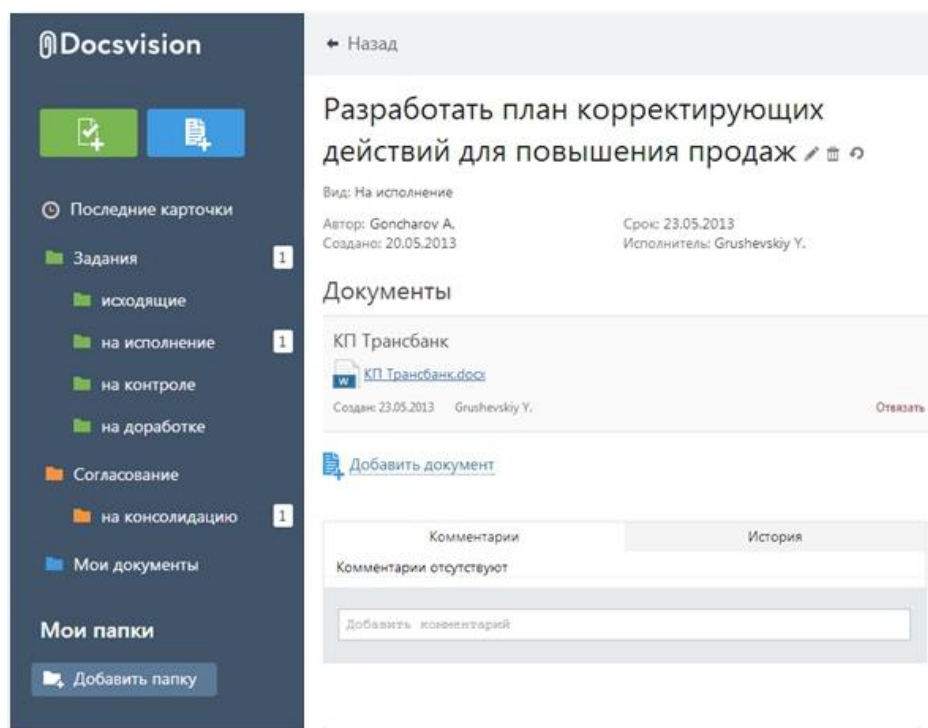


Рисунок 1 – Интерфейс электронной системы управления документами Docsvision

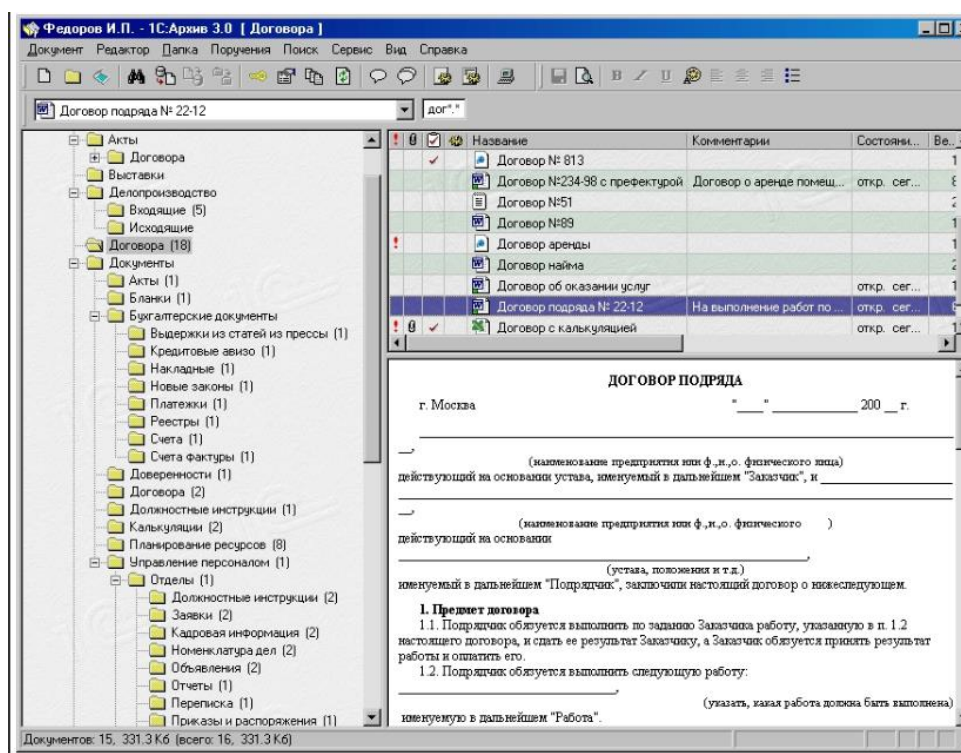


Рисунок 2 – Интерфейса системы электронного архива 1С:Архив

– Workflow-системы. Другое название – системы управления потоками работ (Workflow Management Systems). Вообще, для термина Workflow нет точного эквивалента в русском языке. Концепция Workflow-систем вводит понятие объекта и рассматривает его как движущуюся по определенным

маршрутам абстракцию. Например, простым объектом может быть документ или дело. Система четко определяет маршрутизацию объектов любого типа: что, кто, когда и как делает, откуда получает и куда отправляет. Пользователю не нужно задумываться над тем, как создать документ, как его получить, как обработать, в какие сроки и кому отправить — все уже закреплено в системе. Многие организационные процессы, такие как заполнение документов, напоминания или сообщения о нарушениях происходят под управлением системы, человек же здесь выступает в роли наблюдателя. Такая автоматизация полезна и избавляет персонал от рутинных действий. Больше времени появляется на возможность оценить причины допущения ошибок и своевременно их устранить. Workflow-системы имеют схожие черты с другими видами СЭД и обычно приобретаются как часть более крупного решения. Применяются в компаниях с высокой степенью формализации бизнес-процессов, документооборот в которых при простой структуре имеет массовый характер. Минусом таких систем является сложность и длительность внедрения. Кроме того, они не могут заменить электронный архив, поскольку хранят не все документы, а только используемые в процессе работы.

– Системы управления информацией (Information Management Systems) называемые также порталами, обеспечивают агрегирование, управление и доставку информации через сети internet и intranet. Эти технологии обеспечивают фундамент создания информационных порталов. Такой тип системы, в отличие от других, более удобен для размещения медиа контента или, например, рекламных материалов. Функционал схож с другими системами. Отличительной чертой является большая ориентированность системы на динамическую публикацию контента и использование веб-технологий. Взаимодействие системой осуществляется с помощью веб-браузера, что также очень удобно конечному пользователю.

1.1.2 Классификация по типу используемой технологии

Другая классификация СЭД разделяет их в зависимости от структуры программно-аппаратного комплекса, по принципу которой построена система. Любую СЭД можно рассматривать как совокупность аппаратных и программных компонентов: серверов, клиентских станций, сетевого оборудования и т.д. В этой классификации можно выделить четыре типа систем электронного документооборота:

- Клиент-серверные системы. Система имеет выделенный сервер, на котором размещаются все данные и программные модули управления документацией. Клиентская часть содержит только компоненты, необходимые для обеспечения интерфейса взаимодействия пользователя с системой. Основным преимуществом таких систем является быстрота и надежность. Проектирование такой системы удобно, т.к. требует минимум дополнительных технологий для реализации.

- Системы на основе баз данных. Разновидность систем, интегрированных с базами данных типа SQL и Oracle. Вся информация хранится в этих базах данных. Для обработки информации применяются отдельные модули. Любая информация представляется как совокупность взаимосвязанных таблиц. Это облегчает подход к моделированию контента. Преимуществом таких систем является возможность хранения большого объема информации. Рисунок 3 иллюстрирует пример логической модели базы данных.

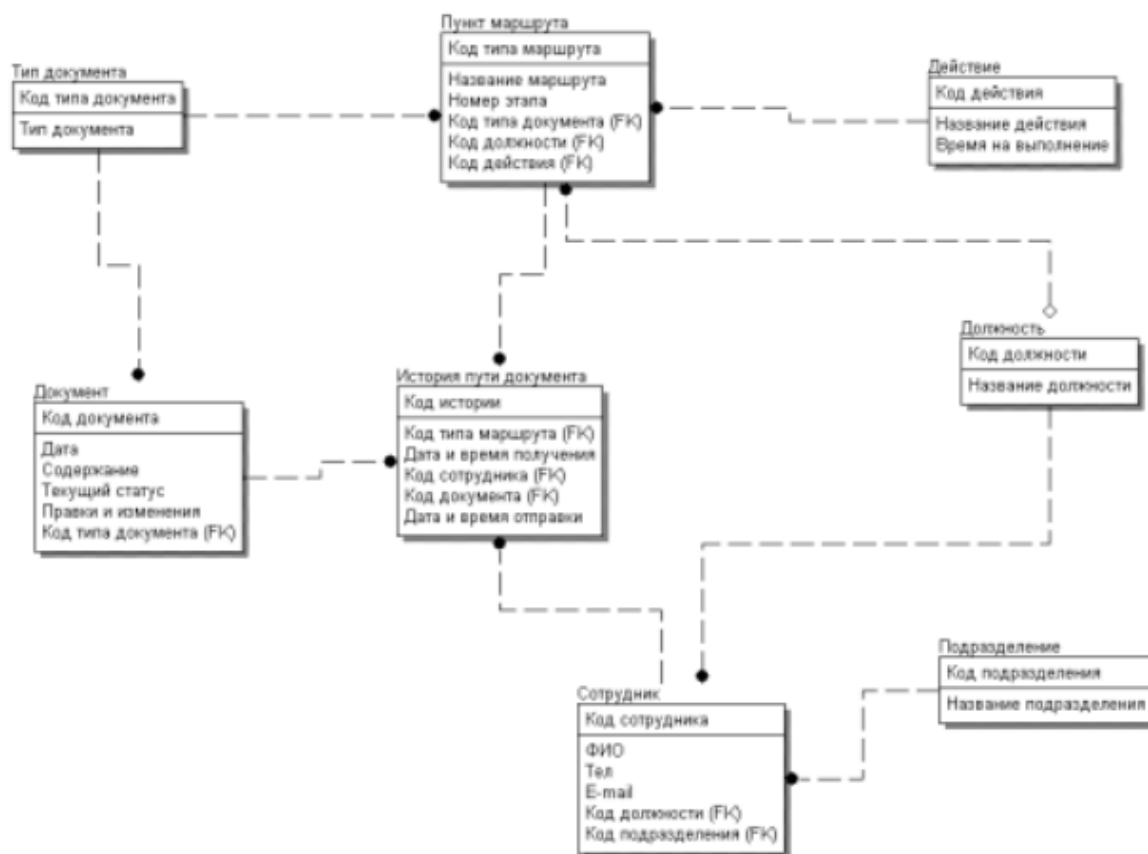


Рисунок 3 – Пример простейшей логической модели базы данных

– Системы на основе web-технологий. Эти системы обеспечивают работу на основе удаленного доступа к серверу. Сама система (ее клиентская часть) является веб-сайтом, предоставляющим пользовательский интерфейс взаимодействия с системой. Данный тип систем удобен, если стоит задача размещения и публикации медиаконтента. Преимуществом является возможность отказаться от проектирования клиентских приложений. Так как система представляет собой веб-сайт, то многие задачи по организации интерфейса перекладываются на веб-технологии. Это удобно как с точки зрения разработчика, так и с позиции пользователя. Доступ к системе документооборота с рабочих мест пользователей может осуществляться посредством любого web-браузера.

1.2 Обзор существующих систем документооборота

В настоящее время отечественный рынок систем электронного документооборота является одним из наиболее динамично развивающихся сегментов ИТ-отрасли. Это обуславливает и плотную конкуренцию среди разработчиков. Примерно половину рынка контролирует несколько компаний: CognitiveTechnologies, «1С», Directum и Microsoft.

Наиболее востребованными на отечественном рынке СЭД традиционно являются продукты российских разработчиков. Это обусловлено тем, что система документооборота в России всегда была вертикально направленной, то есть документ, прежде чем дойти до исполнителя, должен пройти целый ряд согласований и подписаний у вышестоящего руководства.

Приведем список самых перспективных и заслуживающих внимания современных систем электронного документооборота:

– Directum. Высокомасштабируемая ЕСМ-система для крупных и средних предприятий. Одна из наиболее популярных на российском рынке систем. Представляет собой СЭД с возможностями управления документооборотом, бизнес-процессами и web-контентом. Directum полностью соответствует концепции типичной ЕСМ (Enterprise Content Management) системы, одновременно сохраняя в себе все необходимое для традиционного делопроизводства. Directum обеспечивает полный жизненный цикл управления документами. Система включает в себя функции ввода и преобразования документов, управления совместной работой, долговременного хранения документов и обеспечения их целостности. Система Directum предлагает веб-клиент, который, наравне с десктоп-приложением, предоставляет доступ ко всем возможностям системы. Заказчик может самостоятельно определить состав системы. Все интерфейсы системы унифицированы, правила обработки документов и заданий настраиваются под конкретные требования. Стоимость внедрения зависит от вида приобретаемой лицензии и числа приобретаемых дополнительных компонентов. Directum нельзя отнести к числу бюджетных решений.

– E1 ЕВФРАТ. Система «Е1 Евфрат» - мощный инструмент автоматизации бизнес-процессов и оптимизации документооборота для компаний всех типов и размеров. Система эффективно решает задачи как в рамках небольшой структуры, например канцелярии, отдела, департамента или локальной организации в целом, так и в рамках территориально распределенной организации со сложной схемой информационных потоков. При этом «Е1 Евфрат» полностью настраивается в соответствии с требованиями регламентов, положений и инструкций по работе с документами, разработанными и применяемыми организацией. Система занимает лидирующие позиции в классах DMS-, BPM-, и ECM-систем.

– «ДЕЛО». Система, ориентированная на автоматизацию процессов документооборота (от создания документов до передачи в архивное хранение) и поддерживающая интеграцию с существующими информационными системами предприятия. Включает полный набор необходимых для управления делопроизводством и документооборотом инструментов: регистрация корреспонденции, перевод документов в электронный вид, быстрый поиск и надежное хранение, контроль исполнения поручений, построение маршрутов под бизнес-процессы. В зависимости от потребностей компании можно организовать полностью электронный документооборот или его смешанный бумажно-электронный вариант. Программа адаптируется под особенности работы любой организации, при необходимости она масштабируется практически без ограничений. «ДЕЛО» является коробочным продуктом и может быть быстро и легко установлена как собственными силами компании, так и с помощью специалистов разработчика. Интерфейс программы достаточно прост, в нем содержится максимум удобных инструментов для облегчения работы пользователя.

– DocsVision. Представляет собой программный продукт, позволяющий создавать автоматизированные корпоративные решения по управлению бизнес-процессами и документами. В него входит предметно-ориентированная платформа с открытыми интерфейсами для разработки

заказных приложений и готовые типовые приложения с возможностью настройки под параметры заказчика. Система обеспечивает автоматизацию документооборота, поиск и анализ информации, информационную безопасность. В структуру системы входят блоки: электронный архив; делопроизводство; договорная работа; оперативное управление; специализированные решения.

– «1С: Документооборот». «1С: Документооборот» - универсальное решение для автоматизации управления документами, обеспечения полного цикла обработки официальных и внутренних документов организации с поддержкой как электронного, так и бумажного документооборота. «1С: Документооборот» учитывает требования отечественного законодательства, соответствует действующим ГОСТам, стандартам и нормативным актам в области документооборота и делопроизводства. Возможности программы позволяют в комплексе решать различные задачи автоматизации учета документов, взаимодействия сотрудников, контроля и анализа исполнительской дисциплины.

1.3 Требования к проектируемой системе

Стоит задача разработать сайт системы электронного хранения документации для небольшой частной организации или предприятия малого бизнеса.

Примерами таких организаций могут послужить различные магазины, поликлиники или агентства, внутренняя структура которых подразумевает существование нескольких десятков людей из персонала, которые и будут являться пользователями системы. Основные документы, с которыми работают такие организации это, как правило, различная входящая корреспонденция, договоры и отчеты.

Стоит сразу отметить, что система не будет реализовывать концепцию движения документов и бизнес процессов, а являться лишь средством приема, хранения и управления документами.

Разрабатываемая система должна иметь удобный пользовательский интерфейс и реализовывать следующие функции, характерные для любой подобной системы:

- вход на сайт зарегистрированных пользователей
- настройка и добавление новых документов
- показ карточки документа
- разграничение доступа к каждому типу документов
- возможность создания/изменения/редактирования документов
- централизованное хранение документов
- система уведомлений
- система удобного поиска по критериям
- ролевая система пользователей (администратор, пользователь и т.д.).

Интерфейс подобных систем проектируется исходя из следующих принципов:

Левая часть страницы должна представлять собой боковую вертикальную панель элементов (см. рисунок 4). Как правило, это список ссылок для организации доступа к категориям документов (входящие, исходящие, черновики, отработанные и т.д.)

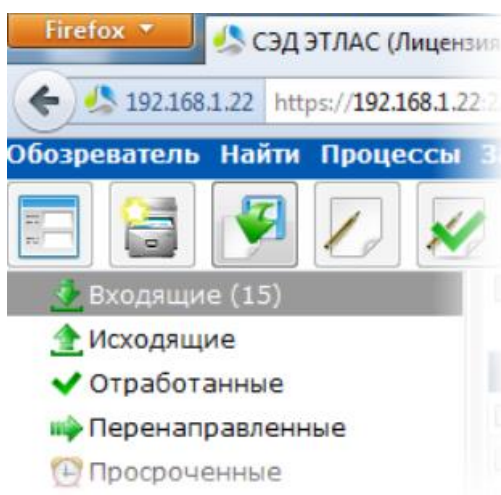


Рисунок 4 – Боковая вертикальная панель инструментов СЭД "ЭТЛАС"

Вся центральная часть страницы используется для отображения документов соответствующей категории, т.е. отведена под отображение самого контента.

В верхней части страницы располагаются всякие элементы панели инструментов, а также меню login/logout.

Ниже приведен пример интерфейса СЭД, найденный на просторах интернета и взятый за основу для разработки собственного проекта.

Страница сайта будет представлять собой шаблон с тремя основными областями: область левой боковой панели, верхняя горизонтальная область (часто эту область называют меню страницы или заголовок страницы) и центральная область, часто называемая областью контента. На рисунке 5 изображено, как должна выглядеть такая страница.

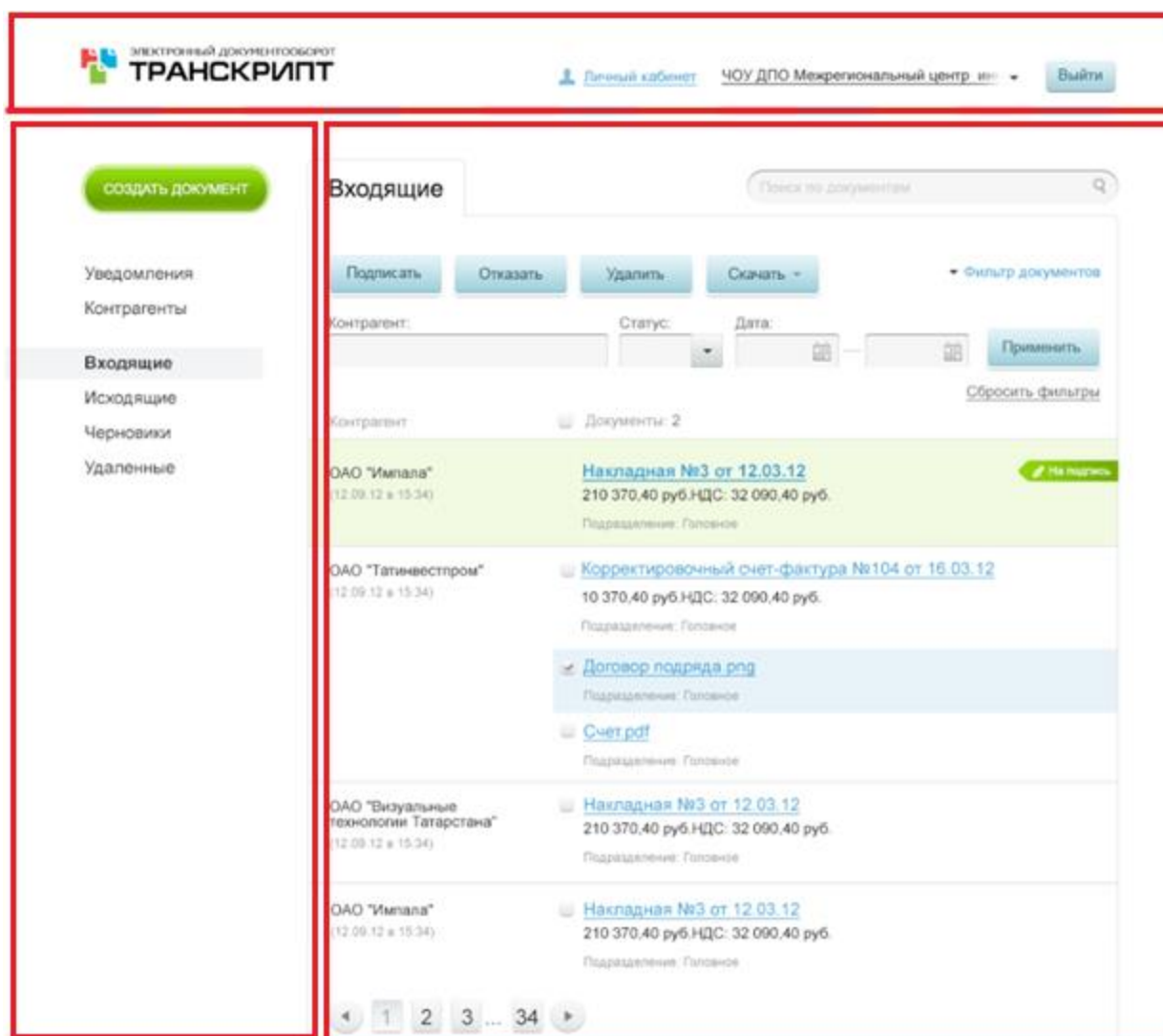


Рисунок 5 – Типичный интерфейс сайта системы на примере СЭД "Транскрипт"

2 СРЕДСТВА РАЗРАБОТКИ САЙТА

На сегодняшний день существует достаточно большое количество технологий, которые используют для создания веб-сайтов. Решение о том, какую именно технологию использовать для разработки сайта принимается исходя из предъявляемых к нему требований. Обычно выбор средства для разработки сайта основывается на следующих критериях: размер и тип проекта, сложность проекта, гибкость решения, наличие подробной документации, требования к безопасности, наличие готовых решений, кроссплатформенность.

Под понятием средств разработки сайта понимается ниша инструментов, помогающая разработчику в процессе создания интернет ресурса. Говоря про средства, имеется ввиду либо просто языки сами по себе, либо разработанное на этих языках дополнительное ПО.

Для дальнейшего анализа разделим все технологии создания сайтов на три абстрактных уровня: чистый язык, веб-фреймворк, система управления контентом (CMS).

Чистый язык — собственно, на этом уровне инструментами разработчика являются языки программирования и разметки текста в своем чистом виде. Это материал, из которого можно сделать все, что угодно. Разработчик ограничен только возможностями самих языков. Почти всегда это следующая четверка технологий: HTML, CSS, PHP, Java-Script.

Но это лишь базовые технологии. Бесспорно, познания в них просто необходимы для любого программиста, но разработка чего-либо только этими средствами неэффективна. Исключением являются, пожалуй, немногие крупнейшие сайты мира с посещаемостью в миллионы пользователей, такие как: Instagram, YouTube, Tumblr, Twitter, Facebook, Amazon и другие. Вначале разработки таких масштабных проектов зачастую нет готовых решений под данную задумку. Более того, крупнейшие проекты

в мире даже создают новые технологии для себя, так как уже существующие их не устраивают.

Веб-фреймворк — среда для разработки с готовыми правилами и инструментами, облегчающая процесс написания и запуска веб-приложения. Избавляет от необходимости написания ненужного кода и призван упростить процесс разработки. Основой программных интерфейсов фреймворка является какой-нибудь язык программирования (Python, PHP, Ruby). Фреймворк, с одной стороны, помогает и ускоряет разработку, а с другой, накладывает определенные ограничения. Архитектура почти всех веб-фреймворков основана на декомпозиции нескольких отдельных слоёв, что означает, что разработчик сможет расширять функциональность, исходя из своих потребностей и использовать изменённую версию вместе с кодом фреймворка. Такая гибкость является ещё одним ключевым преимуществом фреймворков.

Система управления контентом (CMS)— программное обеспечение, предназначенное для создания, организации структуры и управления веб-сайтом. Это уже готовое решение, конструктор, в котором «по частям» собирается нужный проект. Его скорее не программируют, а настраивают. CMS дает возможность добавлять и редактировать контент сайта, не изменяя внутренний механизм организации и вывода страниц. Условно CMS разделена на два хранилища информации: для баз данных с контентом страниц и для элементов визуализации, позволяющих показывать содержимое сайта посетителям (графические элементы, шаблоны и т.д.). Ограничений тут огромное количество. Разрабатывая даже легкий сайт легко выйти за возможности встроенных возможностей CMS. Но это не так страшно, большинство современных CMS предоставляют разработчику механизмы для своего расширения. На CMS делаются сайты разной сложности, здесь все зависит от ее степени расширяемости.

Но, какой бы подход к созданию сайта не был выбран, базовые технологии создания сайта остаются неизменными. Со временем они лишь

улучшаются и дополняют друг друга. Это такие технологии как: HTML, CSS, языки PHP и JavaScript.

2.1 Выбор решения

Задача дипломной работы состоит в разработке сайта системы электронного хранилища документов организации. Задачу можно отнести к задачам средней сложности, поэтому возможностей конструктора сайтов здесь не хватит. Проектируемая система должна иметь свои собственные шаблоны отображения содержимого, типы контента, а также логику доступа к материалам сайта. Оптимальным решением будет создавать сайт на основе CMS. Во-первых, это позволит использовать встроенный функционал выбранной CMS для управления сайтом. Во-вторых, избавит разработчика от «изобретения велосипеда». Многие основополагающие модули любого информационного сайта, такие как система аутентификации и, например, ролевая система пользователей уже включены в состав CMS.

Встает вопрос правильного выбора CMS. По понятным причинам, «базового» функционала, который предоставляют CMS, будет недостаточно для создания полноценного сайта, удовлетворяющего всем требованиям. Сайт будет работать с документами и типами документов, это предполагает создание соответствующего контента на сайте. Хотя многие CMS и обладают встроенными конструкторами контента, на практике их возможности строго шаблонизированы и добавление пары новых полей в структуру контента может оказаться невыполнимой задачей. Поэтому следует выбирать такую CMS, для которой можно будет без проблем создавать свои программные модули. Главным критерием выбора является расширяемость системы.

2.2 CMS Drupal

В поиске подходящей CMS для разработки сайта выбор остановился на технологии CMS Drupal.

Drupal — система управления содержимым(CMS), используемая также как каркас для веб-приложений (CMF), написанная на языке PHP и использующая в качестве хранилища данных реляционную базу данных. В связке с Drupal возможно использование MySQL, PostgreSQL, SQLServer, SQLite и некоторых других реляционных баз данных. Drupal является бесплатным и свободно распространяемым программным обеспечением, защищённым лицензией GPL.

Главной особенностью системы является ее модульная архитектура, позволяющая использовать Drupal для построения различных типов сайтов — от сайтов-визиток до корпоративных приложений и информационных архивов. Модульный принцип построения Drupal подразумевает возможность включения/выключения модулей, отвечающих за отдельные возможности сайта.

Функционал можно расширять практически до бесконечности. Делается это модульным способом построения сайта. Модули помогут не только расширить функциональность сайта, но и кардинально поменять дизайн.

Сам программный продукт представляет собой дистрибутив. Дистрибутив содержит ядро Drupal и некоторые модули, которые включены в Drupal по стандарту. Ядро Drupal – набор PHP скриптов и некоторых других файлов, реализующих базовый слой программного интерфейса Drupal (APIDrupal). На рисунке 6 представлен пример содержимого дистрибутива Drupal.

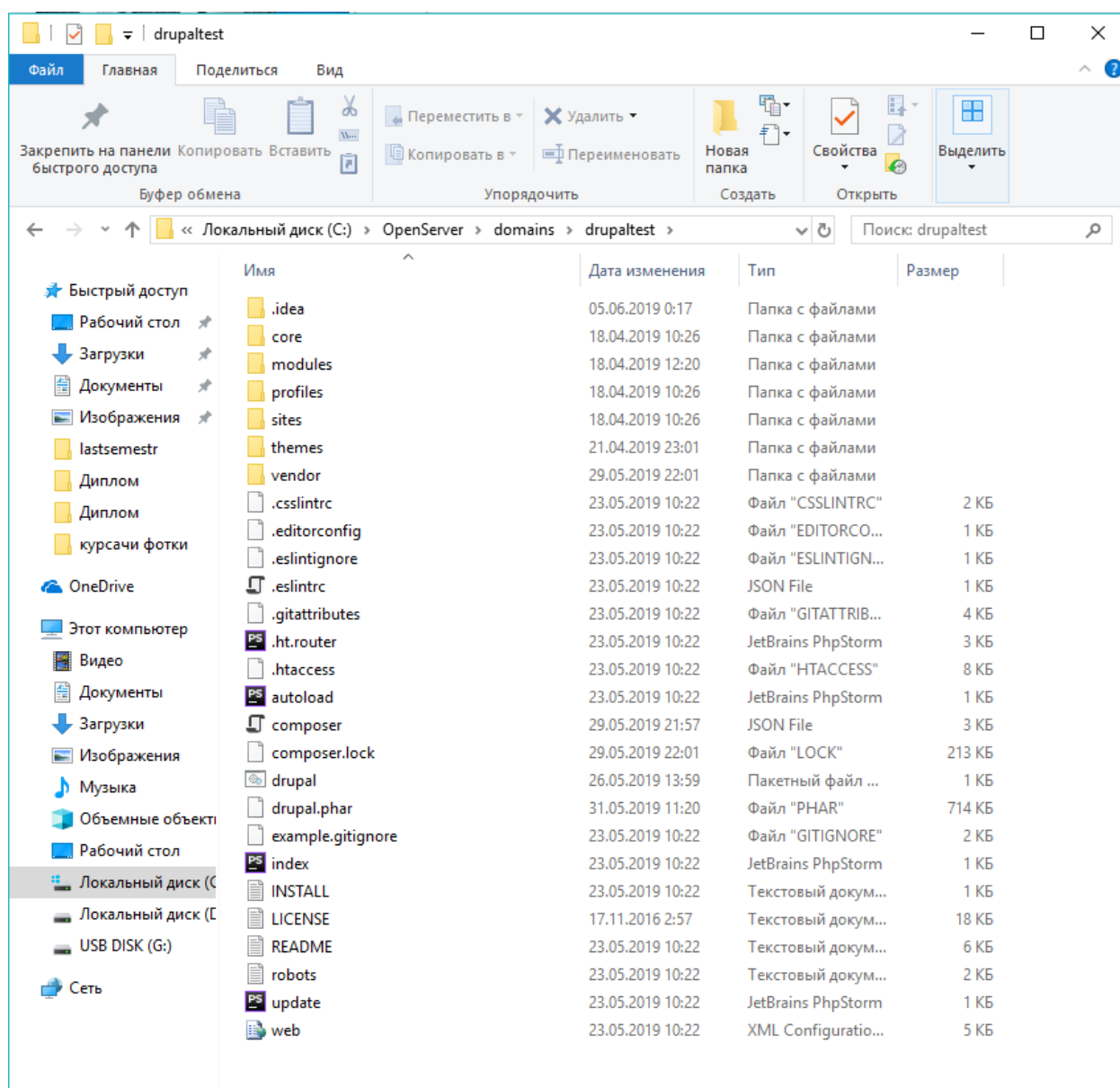


Рисунок 6 – Содержимое дистрибутива Drupal

ПО взаимодействует с веб сервером, базой данных и веб-браузером и обеспечивает работы таких основ CMS как:

- система отправки URL запросов;
- система управления пользователями с настраиваемыми ролями и правами;
- система управления контентом;
- система темизации (шаблонов);
- система блоков, позволяющая группировать контент и располагать его в различных зонах страницы;
- система событий и обработки ошибок.

Поведение ядра Drupal может быть изменено путем создания собственных тем и модулей.

– Темы. Не меняют функциональных возможностей Drupal. Темы изменяют только внешний вид сайта. Тема представляет собой определенную иерархию каталогов. Каталоги могут содержать файлы шаблонов, которые изменяют разметку страниц, CSS-файлы стилей, а также Java-Script файлы и файлы ресурсов темы (изображения, иконки и т.д.). Темы позволяют разработчику полностью переопределить всю разметку и все стили страниц на сайте, необходимо лишь изучить правила создания тем, основанные на определенных соглашениях об именовании файлов и папок, чтобы ядро Drupal смогло распознать тему.

– Модули. Модули добавляют или изменяют поведение и функциональность Drupal. Создаются по такому же принципу, что и темы. Состоят в основном из php файлов и конфигурационных файлов, но могут содержать и другие файлы. Большая часть PHP кода Drupal является объектно-ориентированной. К правилам именования файлов и папок здесь добавляются еще соглашения о правилах объектно-ориентированного программирования внутри модуля. Например, это соглашение затрагивает именование php файлов и то, в каком пространстве имен должен находиться класс, описываемый в этом файле. Полный список правил длинный, и приводиться здесь не будет. Drupal использует ряд расширенных функций языка PHP и библиотек своего ядра, чтобы предоставить разработчикам продвинутый API, средствами которого и создается модуль. Создание собственных базовых классов допустимо, но обычно для реализации чего-либо, достаточно наследоваться от классов Drupal API.

2.3 Язык программирования PHP

PHP – это широко используемый рефлексивный язык программирования, разработанный в качестве инструмента для создания веб-приложений и работы с базами данных.

PHP это язык программирования, специально разработанный для написания сценариев, внедряемых в гипертекстовые страницы и исполняющихся на веб-сервере. Язык в основном предназначен для веб-разработки, но может использоваться и в качестве языка программирования для GUI-приложений.

В области веб-программирования PHP является одним из популярнейших скриптовых языков благодаря своей простоте, скорости выполнения и функциональности.

PHP-скрипты выполняются на стороне сервера. Конструкции PHP выполняются сервером при каждом посещении веб-страницы. Результат их обработки вместе с обычным HTML-текстом передается браузеру. PHP прост для освоения, и вместе с тем способен удовлетворить запросы профессиональных программистов.

Язык PHP постоянно совершенствуется, и ему наверняка обеспечено долгое доминирование в области языков веб-программирования, по крайней мере, в ближайшее время.

2.4 Шаблонизатор Twig

Twig — компилирующий обработчик шаблонов с открытым исходным кодом, написанный на языке программирования PHP.

В Drupal для хранения разметки всех страниц сайта используются файлы с расширением `.html.twig`. Они называются Twig шаблонами. Помимо HTML-разметки они содержат переменные или выражения. Когда браузер запрашивает страницу, все участвующие в отображении этой страницы шаблоны выполняются. Переменные и выражения в шаблонах заменяются вычисленными значениями, пригодными для вывода.

Все Twig шаблоны в Drupal компилируются в `php` файлы и хранятся в одном из каталогов сайта. Файлы шаблонов могут кэшироваться для повторного использования в целях улучшения производительности. Twig

шаблонизатор не всегда использовался в Drupal, а появился только в последней восьмой версии Drupal. На это было несколько причин.

Во-первых, он более безопасный. В шаблонах других версий Drupal часто выводились неочищенные данные. Программист мог засунуть в шаблон какой-нибудь SQL запрос или PHP инструкцию, связанную с логикой работы сайта. Это противоречило концепции отделения логики сайта от его дизайна.

Twig шаблоны исключают возможность добавления в них программной логики. В Twig шаблон возможно передавать только переменные и значения. Таким образом, отображением данных занимается шаблон, а подготовкой данных, соответственно, логическая часть. При этом удобно править каждую часть в отдельности.

Во-вторых, в Twig есть особенность — наследование шаблонов. Теперь достаточно создать базовый родительский шаблон и затем при создании дочернего шаблона указать, какой родительский шаблон он будет использовать.

3 РАЗРАБОТКА САЙТА СИСТЕМЫ

Первым шагом при разработке сайта является установка самого Drupal. Для работы Drupal необходим веб-сервер, интерпретатор языка PHP, а также сервер системы управления базами данных MySQL, хотя возможно использование и другой базы данных. Все эти компоненты можно установить и настроить отдельно, но легче использовать готовые сборки, то есть программные продукты, в которых все вышеперечисленные компоненты установлены и настроены для работы.

Для развертывания сайта остановимся на классической WAMP (Windows + Apache + MySQL + PHP) сборке OpenServer.

OpenServer — это портативный локальный WAMP/WNMP сервер, имеющий многофункциональную управляющую программу и большой выбор подключаемых компонентов.

OpenServer предлагает на выбор сразу два вида HTTP серверов, различные версии PHP и СУБД модулей, а так же возможность быстрого переключения между ними.

После установки OpenServer появится возможность его запустить.

OpenServer удобен тем, что предоставляет удобный интерфейс под нужды веб-разработчика. Так, используя пункт меню «мои сайты» можно сразу просматривать все свои сайты (см. рисунок 7).

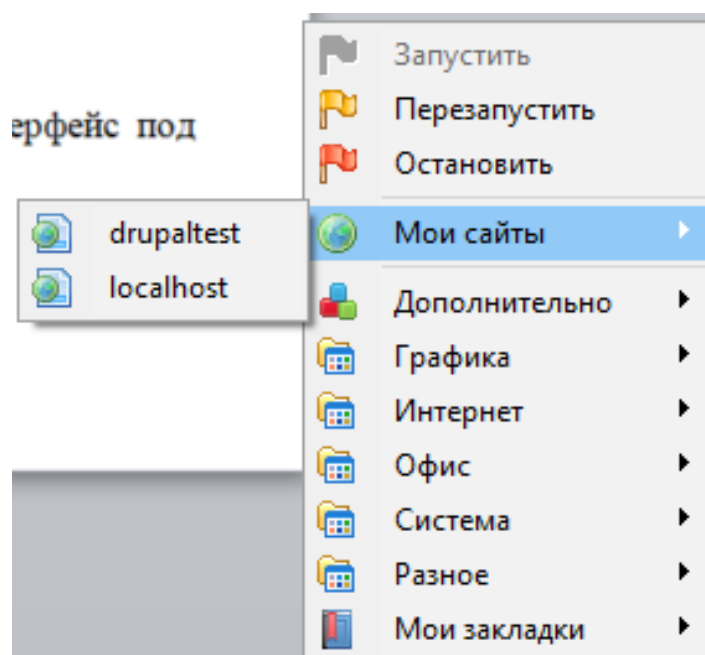


Рисунок7 – Вид меню OpenServer

Следом за OpenServer установим и сам Drupal, скачав дистрибутив с официального сайта и поместив его в директорию OpenServer, предназначенную для хранения сайтов.

Внешний вид главной страницы сайта изображен на рисунке 8. Если на сайт зашел администратор, то для него доступна панель управления сайтом, располагающаяся в верхней части страницы (рисунок 8). Через эту панель осуществляется доступ ко всем базовым механизмам управления сайтом.

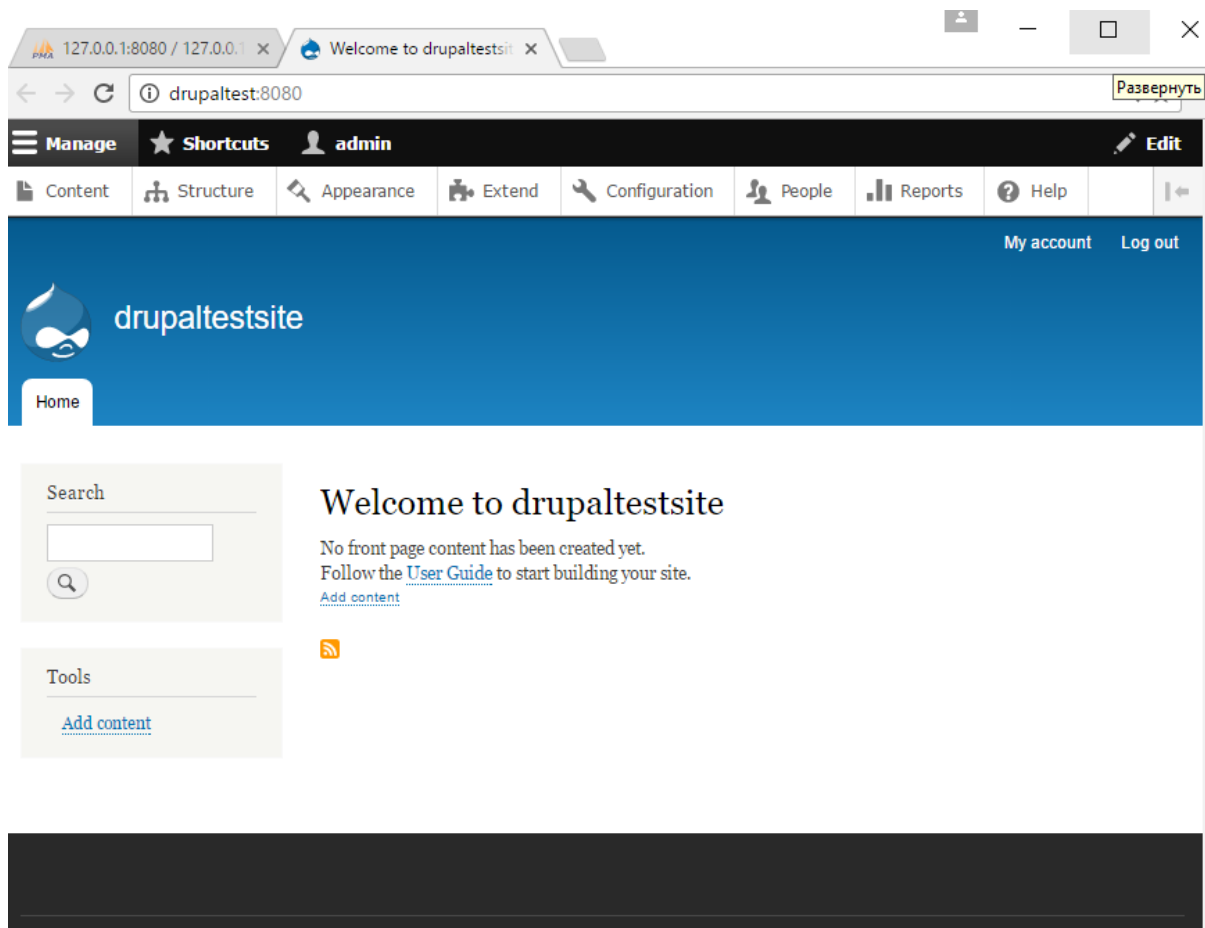


Рисунок 8 – Начальная страница сайта Drupal сразу после установки

3.1 Базовые возможности системы

Рассмотрим некоторые из функциональных возможностей Drupal, которые имеются сразу после установки сайта.

3.1.1 Управление контентом

В Drupal имеется удобная панель администратора сайта. Первым пунктом в ней является «Content». Он открывает формы добавления материалов, также можно создать и свой тип материала. Контент, созданный таким образом, можно публиковать и размещать на главной странице. Это удобно, когда сайт представляет собой простой новостной блог.

Но этот метод не подходит для создания сложных типов контента, которыми и будут являться документы и их типы. Так, понадобится создать

новый тип контента для хранения информации о документах со множеством полей и собственные формы для отображения всего этого.

3.1.2 Меню

Нужные страницы можно вынести в меню. В меню добавляются ссылки на часто используемые страницы сайта. Встроенный блок меню горизонтальной линией размещается сразу под логотипом сайта. Для разрабатываемой системы это ненужная возможность, все нужные пункты будут располагаться в собственных блоках в левой части страницы.

3.1.3 Пользователи

Пункт «People» в панели управления сайтом предоставляет удобный интерфейс добавления пользователей (рисунок 9), настройки их ролей и прав доступа. Здесь ничего добавлять от себя не нужно, ко всем новым ролям и правам, которые будут добавляться в процессе разработки сайта, возможен доступ прямо отсюда.

По умолчанию сайт будет иметь две пользовательские роли: администратор и зарегистрированный пользователь. Администратор сайта может добавлять новые роли и пользователей.

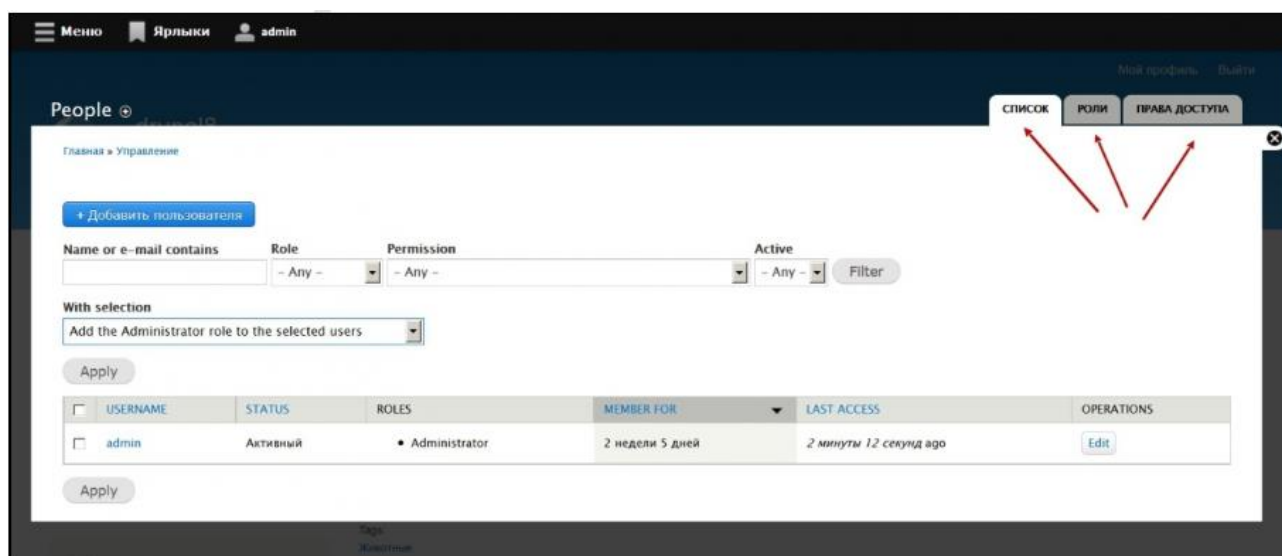


Рисунок 9 - Форма добавления пользователей Drupal

3.1.4 Темы

Настроить текущую или применить новую тему для Drupal можно, воспользовавшись пунктом «Appearance» панели управления. Не содержит ничего лишнего, только список распознанных ядром Drupal тем. Тему можно применить или удалить. Смысла в разработке своей темы с нуля нет, базовая тема содержит много файлов стилей и шаблонов отображения страниц. Достаточным будет создать свою тему на основе существующей, убрав из нее ненужные блоки и переопределив некоторую разметку.

3.2 Требования к сайту

Исходя из сказанного выше, сформулирован перечень того, что необходимо сделать для сайта, чтобы он стал похож на информационную систему хранения и управления документами:

- Разработать новый тип контента «Документы» со множеством полей;
- определить все необходимые формы отображения для документов и блоков меню;
- создать свою тему, исключаящую все лишнее;
- определить все необходимые права, разграничивающие доступ к документам и возможностям по их управлению;
- создать справочники, которые будут хранить некоторую дополнительную информацию, которая может быть указана в карточке документа (отправители, организации и т.д.);

Решать эти задачи будет специальный модуль. Этот модуль будет называться «Модуль документов». Основная задача состоит в разработке этого модуля.

Модуль документов будет иметь типичную для всех модулей Drupal структуру (рисунок 10)

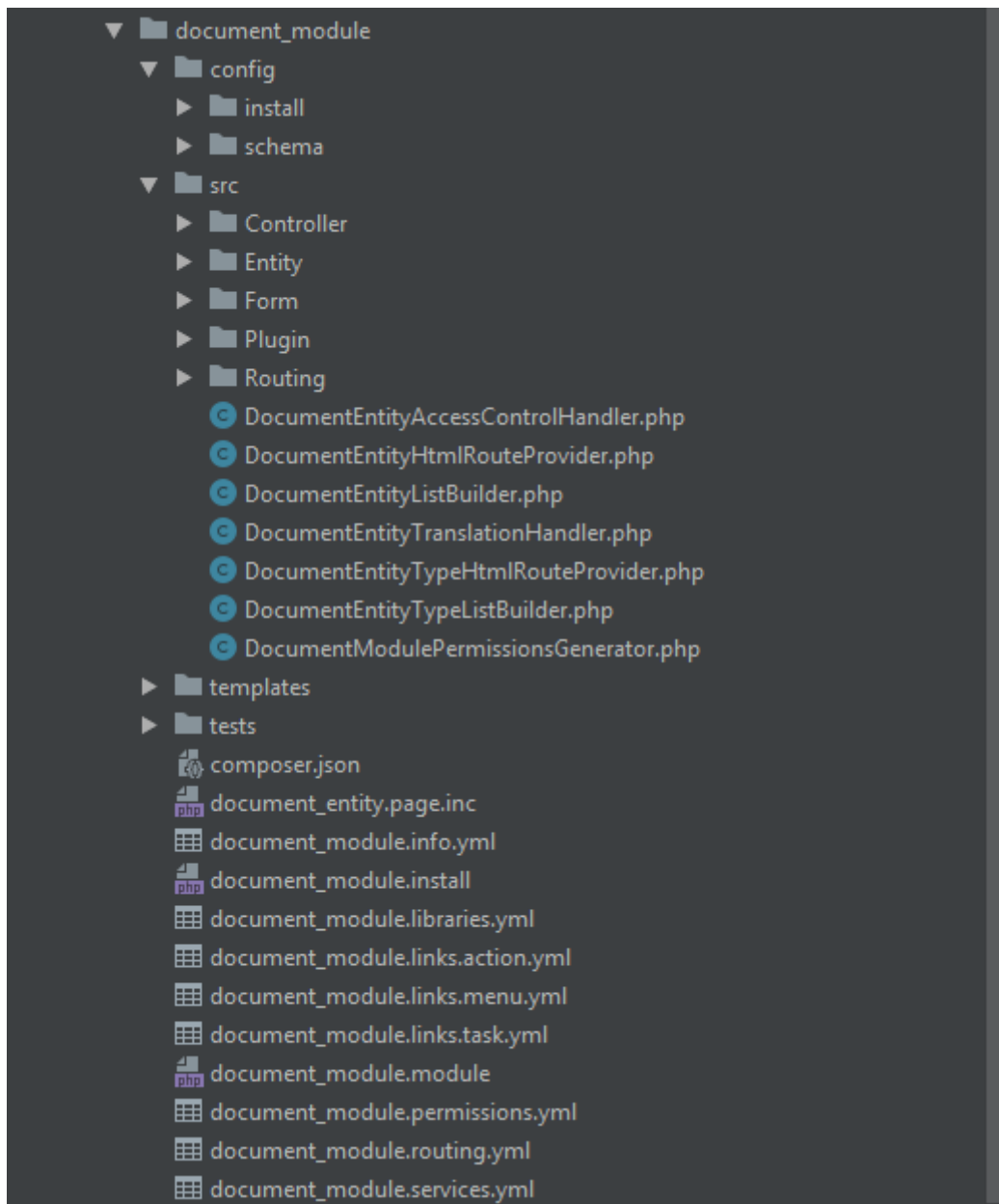


Рисунок 10 - Структура модуля документов

У каждого файла модуля есть свое строго определенное назначение. Если при проектировании модуля была допущена ошибка в любом из файлов, то модуль не установится.

Как говорилось Выше, CMS Drupal построена по модульной структуре – это означает, что есть некое логическое ядро, а весь функционал выносится в отдельные модули.

3.3 Разработка типа контента для хранения документов

В Drupal для хранения информации о контенте используется концепция сущностей (с англ. Entities). Сущность (entity) является основной абстракцией над базой данных в Drupal. Пользователи, статьи, элементы конфигурации и некоторые другие объекты Drupal являются сущностями и поддерживают единый программный интерфейс, называемый Entity API. Сам Entity API представляет собой набор классов и функций ядра Drupal, описывающих принципы объявления и управления сущностями. Сущности в Drupal представляют собой объекты, используемые для постоянного хранения информации. Эта информация хранится в базе данных. Для каждой сущности создается таблица, название которых совпадает с системным именем сущности, а столбцы соответствуют полям сущности.

При этом к одному типу сущности можно прикреплять разные наборы полей, создавая привязки (bundles). Приведу простой пример. Допустим, существует сущность «Документы» с полями идентификатор, имя и дата создания. Тогда для этого документа возможно создать привязку «Входящие» с полем организация – отправитель и привязку «Исходящие» с полем сотрудник – отправитель.

Для организации хранения всех документов необходимо создать новую сущность. Эта сущность будет называться «Документы».

Для того чтобы описать сущность необходимо определить php-класс. Следуя соглашениям Drupal об объектно-ориентированном программировании все классы сущностей должны располагаться в пространстве имен \Drupal\[module_name]\Entity, где module_name – имя модуля (рисунок 11). Это значит, что объявление сущности возможно только внутри модуля. Таким модулем и является «Модуль документов». Подробное описание создания класса сущности здесь приводиться не будет.

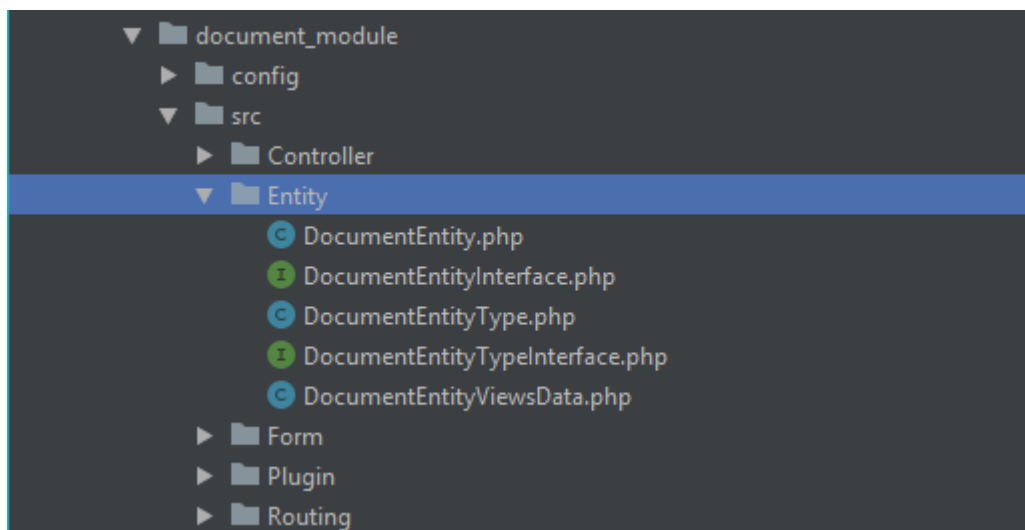


Рисунок 11 - Классы сущностей в структуре модуля

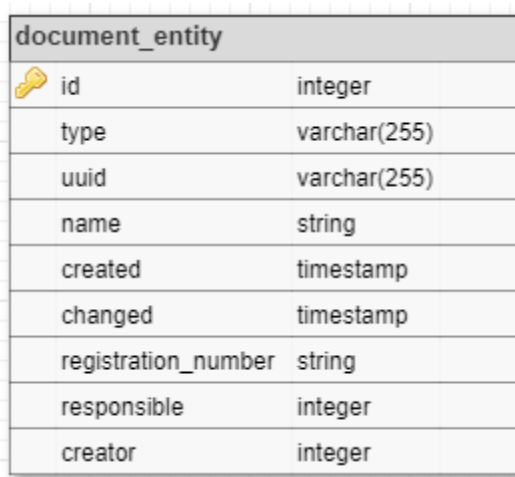
Необходимо определиться со списком базовых полей, которые будут у всех типов документов. Поля в Drupal определяют, какие данные будет хранить сущность. Поля бывают разных типов: строковые, числовые, поля даты, поля электронной почты и т.д. В Drupal существует два типа полей: базовые поля и настраиваемые поля. Первые – это те, которые определены в коде сущности. Они будут общими для всех типов документов. Вторые могут быть прикреплены только к конкретному типу документов. Их определения хранятся в виде конфигурационных файлов `yaml`. Разработчик сайта может создавать и свои типы полей, если существующие не подходят под конкретную задачу.

Документ будет иметь следующие поля: идентификатор (`id`), тип (`type`), `uuid`, имя (`name`), дата создания (`created`), дата редактирования (`changed`), регистрационный номер (`registration_number`), ответственное лицо (`responsible`), кем создан (`creator`), файл (`file`). Поля ответственное лицо (`responsible`) и кем создан (`creator`) будут хранить идентификатор пользователя из таблицы пользователей. Значения полей идентификатор и `uuid` будут автоматически заполняться при создании нового объекта сущности.

Если программно или через пользовательский интерфейс сайта создать новый документ, то в базе данных появится новая запись с уникальным

идентификатором. Таким образом, Drupal обеспечивает хранение объектов сущностей, в нашем случае – документов.

Таблица в базе данных для такой сущности будет выглядеть следующим образом (рисунок 12). Пусть она будет называться document_entity. Все ее столбцы хранят значения базовых полей сущности «Документы».

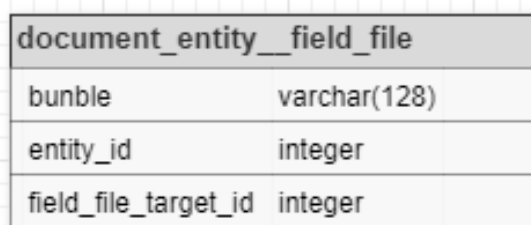


The diagram shows a table named 'document_entity' with the following columns and data types:

document_entity	
id	integer
type	varchar(255)
uuid	varchar(255)
name	string
created	timestamp
changed	timestamp
registration_number	string
responsible	integer
creator	integer

Рисунок 12 - Схема таблицы сущности "Документы" в базе данных

К одному документу возможно прикреплять несколько файлов. В Drupal для полей, которые имеют множественные значения, создаются отдельные таблицы. Например, если бы поле «Файл» использовалось для хранения информации об одном файле, то в таблице document_entity появился столбец с именем file. Но в случае, когда поле используется для хранения нескольких значений, Drupal создает для него отдельную таблицу. Для поля «Файл» создастся таблица document_entity__field_file. На рис. 13 изображена структура этой таблицы.



The diagram shows a table named 'document_entity__field_file' with the following columns and data types:

document_entity__field_file	
bunble	varchar(128)
entity_id	integer
field_file_target_id	integer

Рисунок13–Структура таблицы document_entity__field_file

Для того, чтобы документы могли быть разных типов необходимо создать еще одну сущность «Типы документов». Она будет отличаться от

сущности «Документы». Для нее не будет создаваться никаких новых таблиц в базе данных, ее предназначение будет в другом. Она нужна чтобы создавать новые типы документа и определять, какие у них будут дополнительные поля. Информация о том, какие поля принадлежат конкретному типу документов будет храниться не в базе данных, а в виде конфигурационных файлов. Это файлы формата `yml`, содержащие в себе строки типа ключ – значение.

Пример конфигурационного файла для типа документов «Входящие» приведен ниже. Похожим образом выглядят и конфигурации полей.

Пример 1. Файл `document_entity_type.incoming.yml`

```
uuid: b1379f7a-542a-4b1c-b5d7-8fbfe39b2efc
langcode: ru
status: true
dependencies: {  }
id: incoming
label: Входящие
```

На рисунке 14 изображена структура модуля. Все конфигурационные файлы хранятся в папке `\config\install`.

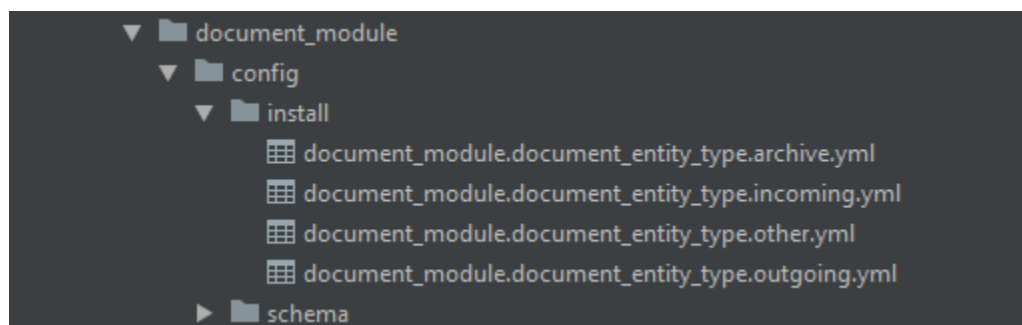


Рисунок 14 - Директория для хранения конфигурационных файлов

При установке модуля к сайту применятся все эти конфигурации. Всего их будет четыре: входящие, исходящие, архивные, другие.

Любой входящий документ помимо базовых полей будет иметь еще следующие поля: организация – отправитель (`sender_organization`), получатель (`recipient`), отработано (`done`).

Поле «организация – отправитель» служит для хранения информации об организации, которая берется из словаря «Организации», поле

«получатель» хранит идентификатор пользователя, которому была адресована эта входящая корреспонденция. Поле «отработано» хранит булево значение. По умолчанию любая входящая корреспонденция не отработана, изменить это значение может либо администратор системы, либо пользователь, которому был адресован этот документ. Отработанная входящая документация помещается администратором в категорию «Архивные».

Для любого исходящего документа добавятся три дополнительных поля: организация – получатель (`recipient_organization`), дата отправки (`outgoing_date`) и отправитель (`sender`).

Архивные документы используют поля от всех типов документов. Это обусловлено тем, что любой тип документа может стать архивным.

Документ из категории «Другие» имеет только базовые поля. Этот тип нужен, чтобы его можно было настраивать через пользовательский интерфейс: добавлять новые поля, редактировать название типа. Сюда можно добавлять документы, не подходящие под другие категории. Диаграмма структуры всех типов документов приведена на рисунке 15.

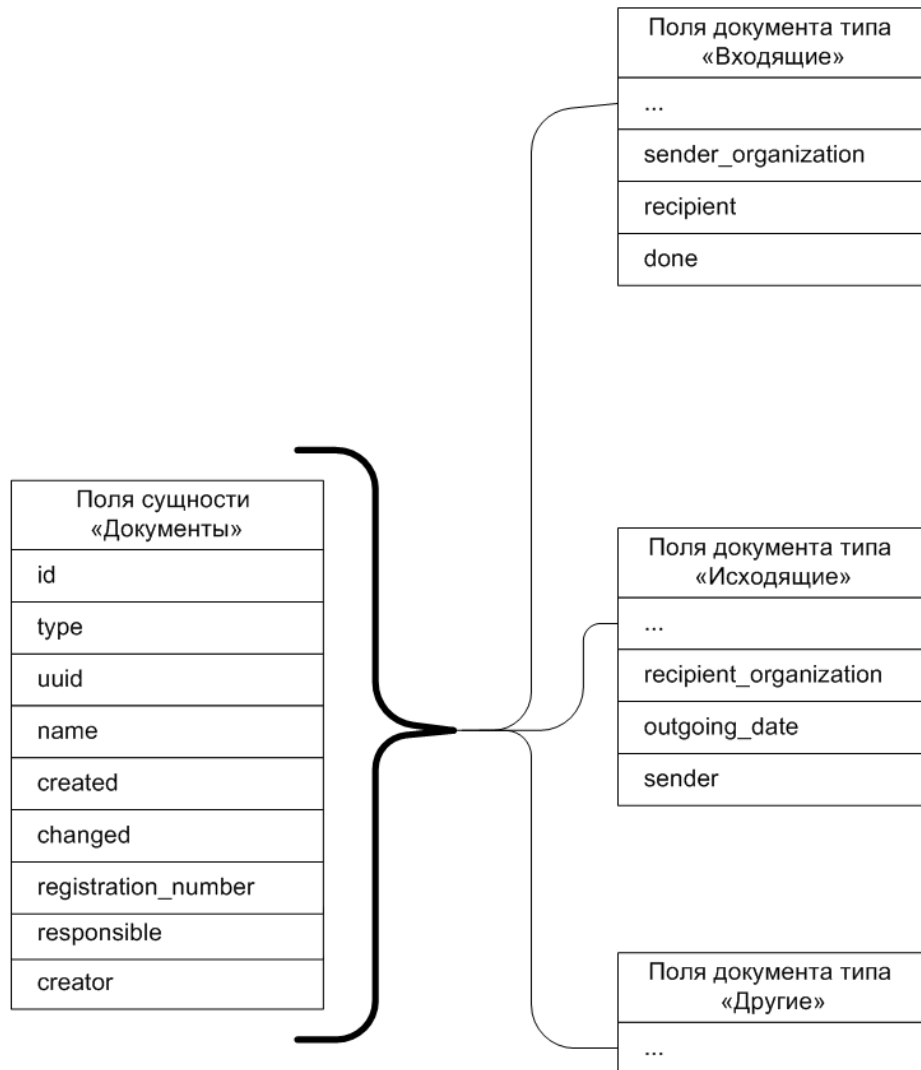


Рисунок 15 - Диаграмма структуры всех типов документов

3.4 Определение всех необходимых форм для отображения документов

Минимальный набор форм, необходимых для функционирования сайта будет содержать: форму добавления документа, форму удаления документа, форму редактирования документа, форму представления документа в виде карточки, табличный список документов. На рис. 16 изображен пример формы добавления входящего документа.

Администратор

Типы документов

Пользователи

Настроить права

Настроить справочники

Документы

Все документы

Входящие документы

Архивные документы

Другие документы

Исходящие документы

Добавить Входящие

Имя *

Первый

Название документа

Регистрационный номер

p-114

Ответственное лицо

Семахин Н. Р. (20)

Организация отправитель

- Не указано -

Оработано

▼ Файл

Показать вес строк

Информация о файле

Operations

+

newfile_0.txt

Удалить

Добавить новый файл

Выбрать файлы

Файл не выбран

Это поле позволяет загрузить неограниченное количество файлов.

Ограничение 100 МБ.

Допустимые типы: txt.

Активация Windows

Чтобы активировать Windows, перейдите в раздел "Параметры".

Сохранить

Рисунок 16 - Форма добавления входящего документа

Концепция форм реализована с помощью специального интерфейса для построения форм (Form API), присутствующего в ядре Drupal. Form API представляет собой набор программных средств, используя которые можно создавать, настраивать и управлять формами Drupal. Любая форма в простом понимании является HTML-кодом, предоставленным браузеру для сбора информации ввода, и логику для обработки этого ввода, когда пользователь отправляет форму.

В Drupal каждая форма представлена классом. Этот класс часто называют контроллером формы. Контроллер формы обеспечивает начальное определение формы, а также проверку и обработку представления с помощью своих методов.

Чтобы определить форму достаточно создать новый класс описания формы и протестировать от одного из базовых классов для создания форм. В теле класса можно переопределять методы базового класса и таким

образом настроить отображение и логику работы формы. На рис. 17 изображено, где располагаются файлы классов форм.

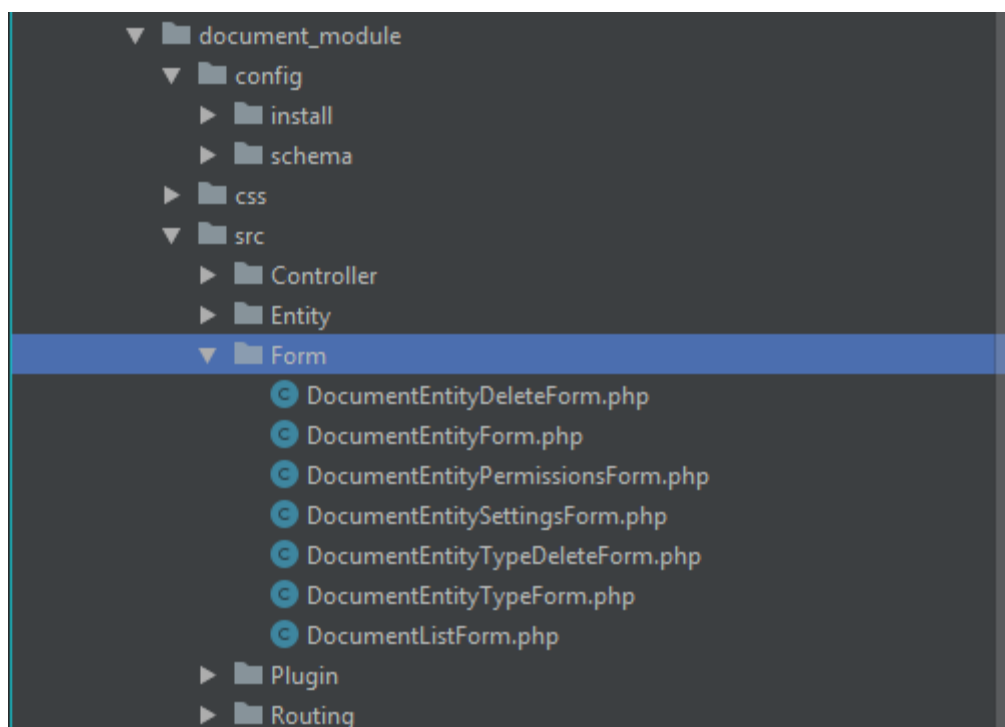


Рисунок 17 - Расположение файлов описания форм

Все формы должны быть описаны внутри модуля. Например, файл `DocumentEntityForm.php` описывает класс формы для добавления и редактирования документов

Пример 2. Файл `DocumentEntityForm.php`

```
<?php
```

```
namespace Drupal\document_module\Form;
```

```
use Drupal\Core\Entity\ContentEntityForm;
```

```
use Drupal\Core\Form\FormStateInterface;
```

```
/**
```

```
 * Form controller for Document entity edit forms.
```

```
 *
```

```
*/
```

```
class DocumentEntityForm extends ContentEntityForm {
```

```
}
```

Аналогичным образом создаются классы для всех других форм.

Все формы будут организовывать свой вывод в область контента страницы. Пользователя, взаимодействуя с сайтом, будет перенаправляться на разные страницы. Каждой странице соответствует своя форма.

В области левой боковой панели будут расположены блоки ссылок (рис. 18). Блок «Администратор» будет доступен только администратору сайта. Блок «Документы» будет доступен всем зарегистрированным пользователям сайта.

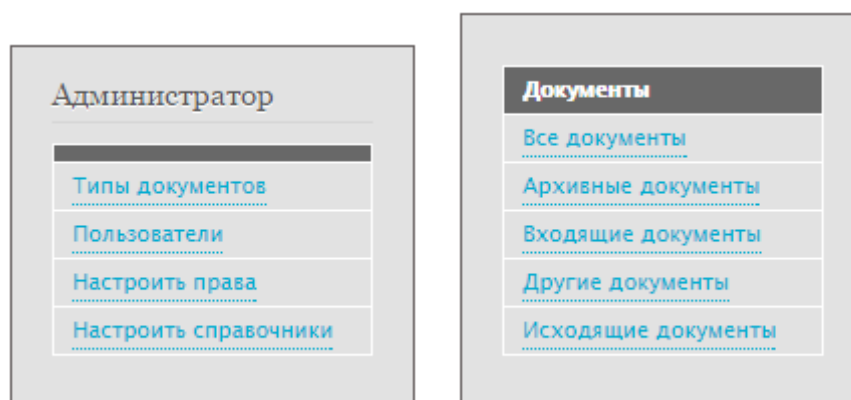


Рисунок 18 - Блоки ссылок

На рисунке 19 изображена диаграмма состояний интерфейса сайта при взаимодействии с блоком «Документы».

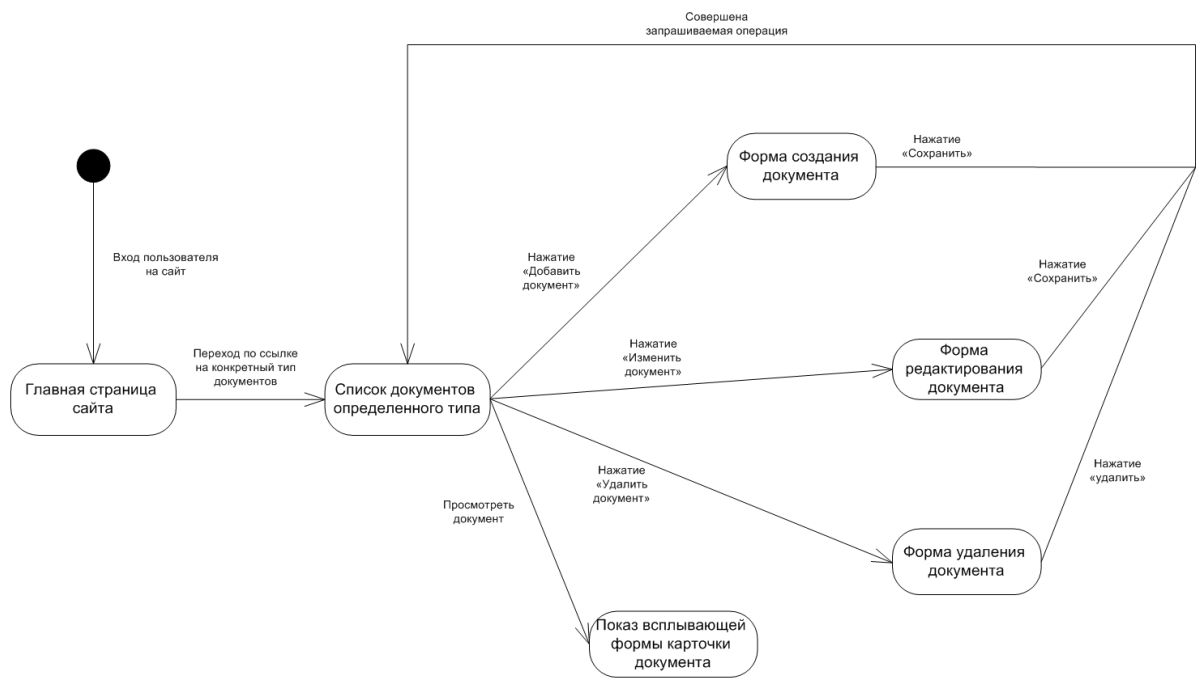


Рисунок 19 - UML-диаграмма состояний интерфейса сайта при работе с блоком «Документы»

На рисунке 20 изображена диаграмма состояний интерфейса сайта при взаимодействии с блоком «Администратор».

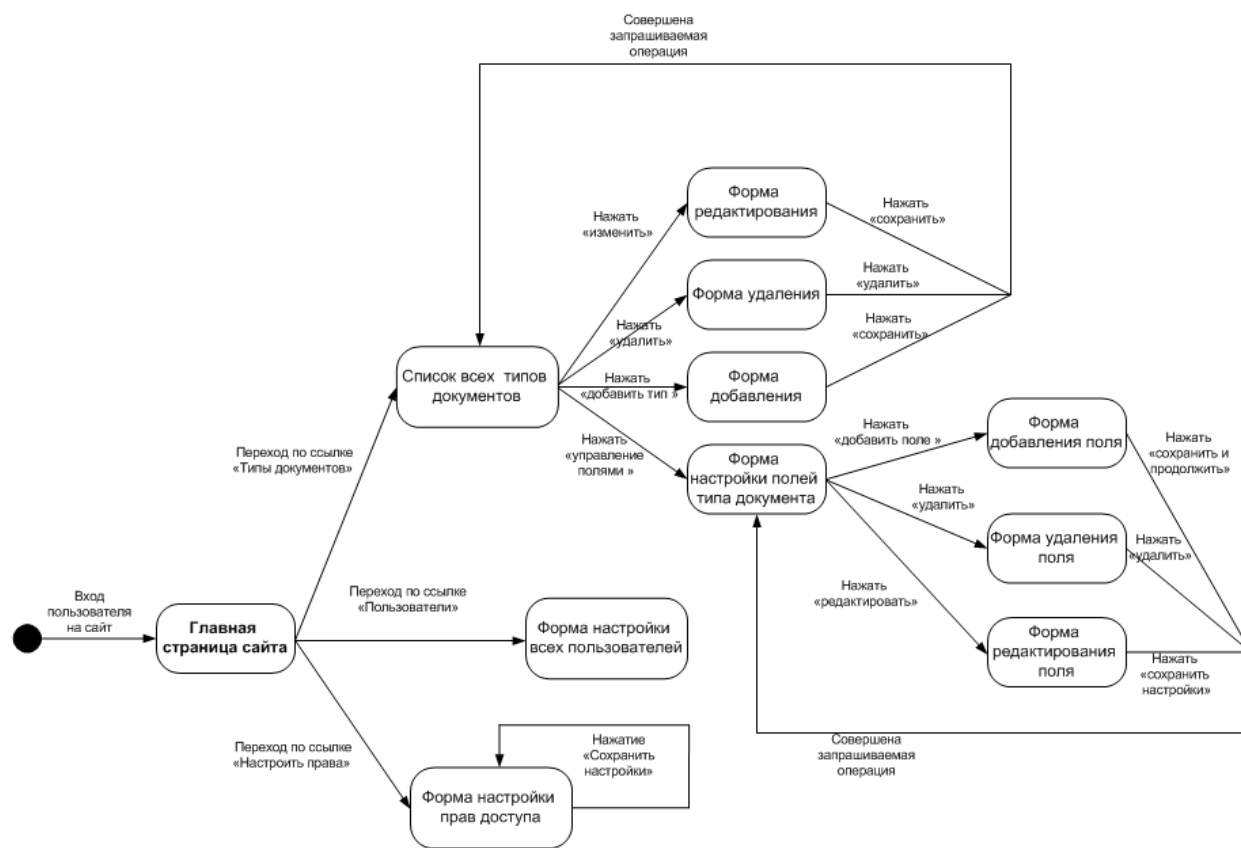


Рисунок 20 – UML-диаграмма состояний интерфейса сайта при взаимодействии с блоком «Администратор»

3.5 Разграничение доступа к документам

Система доступа к контенту в Drupal основана на концепции ролей и прав. Пользователю в Drupal присваиваются роли, а каждая роль, в свою очередь, содержит определенное количество прав на какие-либо действия.

Пользователь (user) – одна из основных сущностей Drupal, являющаяся абстракцией над тем, кто пользуется сайтом. Пользователь может быть либо зарегистрированным, имеющим учетную запись, либо незарегистрированным, анонимным пользователем. Эти два типа соответствуют двум ролям по умолчанию, которые создаются при установке Drupal.

Права (permissions) не являются сущностями и создаются при помощи yml файлов.

В Drupal существует всего четыре операции над сущностями: просмотр (view), удаление (delete), изменение (update), создание (create). Для того, чтобы однозначно определять возможность этих операции для текущего пользователя, необходимо создать хотя бы по одному разрешению на каждое действие.

Перечень прав для модуля управления документами будет выглядеть следующим образом. Для каждого типа документов будут создаваться четыре разрешения, определяющих возможность просмотра, создания, редактирования и удаления этого документа пользователем.

Пользователь с ролью Администратора использует все возможности по управлению сайтом. Ему доступно управление всеми типами контента сайта, включая «Документы» и «Типы документов». Кроме этого, с помощью панели управления администратора он может устанавливать/удалять модули и темы и менять общие настройки сайта.

На рисунке 21 изображена диаграмма вариантов использования сайта для администратора.

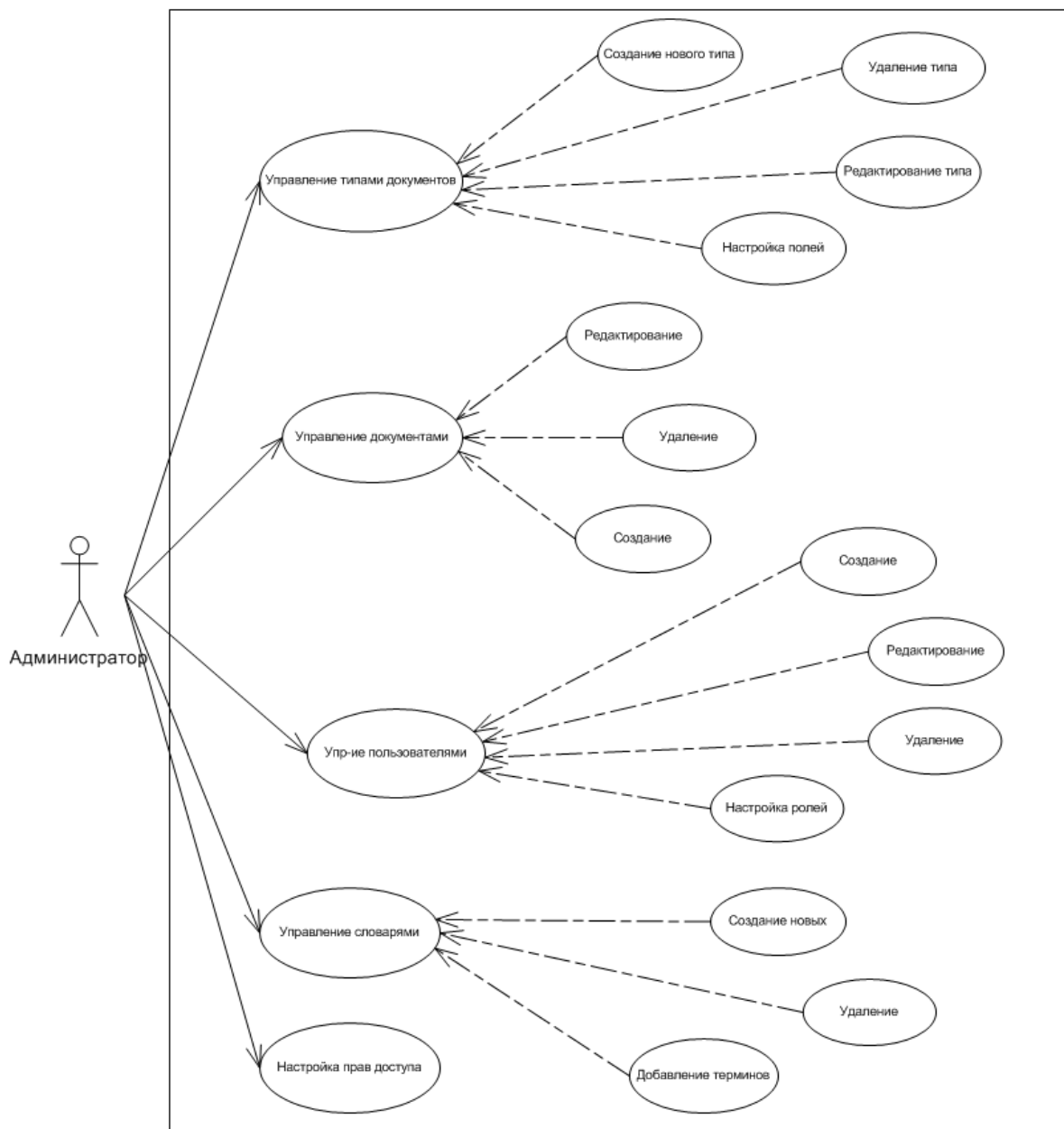


Рисунок 21—UML-диаграмма вариантов использования сайта для администратора

Перечень возможностей любого зарегистрированного пользователя ограничен и определяется конфигурацией, которую для него определил администратор. Такой пользователь не может управлять сайтом. Обычно ему дается доступ к одной или двум категориям документов. На рис. 22 показана форма настройки прав доступа для всех ролей на сайте.

Настроить права доступа

[Hide descriptions](#)

Permission	Anonymous user	Authenticated user	Administrator
Показывать список документов	☐	☐	☒
Показывать список типов документов	☐	☐	☒
Просмотр любого документа категории "Архивные"	☐	☐	☒
Просмотр любого документа категории "Входящие"	☐	☒	☒
Просмотр любого документа категории "Другие"	☐	☐	☒
Просмотр любого документа категории "Исходящие"	☐	☐	☒
Просмотр своих документов категории "Архивные"	☐	☐	☒
Просмотр своих документов категории "Входящие"	☐	☐	☒
Просмотр своих документов категории "Другие"	☐	☐	☒
Просмотр своих документов категории "Исходящие"	☐	☐	☒
Редактирование любых документов категории "Архивные"	☐	☐	☒
Редактирование любых документов категории "Входящие"	☐	☒	☒
Редактирование любых документов категории "Другие"	☐	☐	☒
Редактирование любых документов категории "Исходящие"	☐	☐	☒
Редактирование своих документов категории "Архивные"	☐	☐	☒
Редактирование своих документов категории "Входящие"	☐	☐	☒
Редактирование своих документов категории "Другие"	☐	☐	☒
Редактирование своих документов категории "Исходящие"	☐	☐	☒
Создание нового документа категории "Архивные"	☐	☐	☒
Создание нового документа категории "Входящие"	☐	☒	☒
Создание нового документа категории "Другие"	☐	☐	☒
Создание нового документа категории "Исходящие"	☐	☐	☒
Удаление любых документов категории "Архивные"	☐	☐	☒
Удаление любых документов категории "Входящие"	☐	☒	☒

Рисунок 22 - Форма настройки прав доступа

3.6 Работа с сайтом

Для Администратора главная страница сайта выглядит следующим образом (рис. 23).

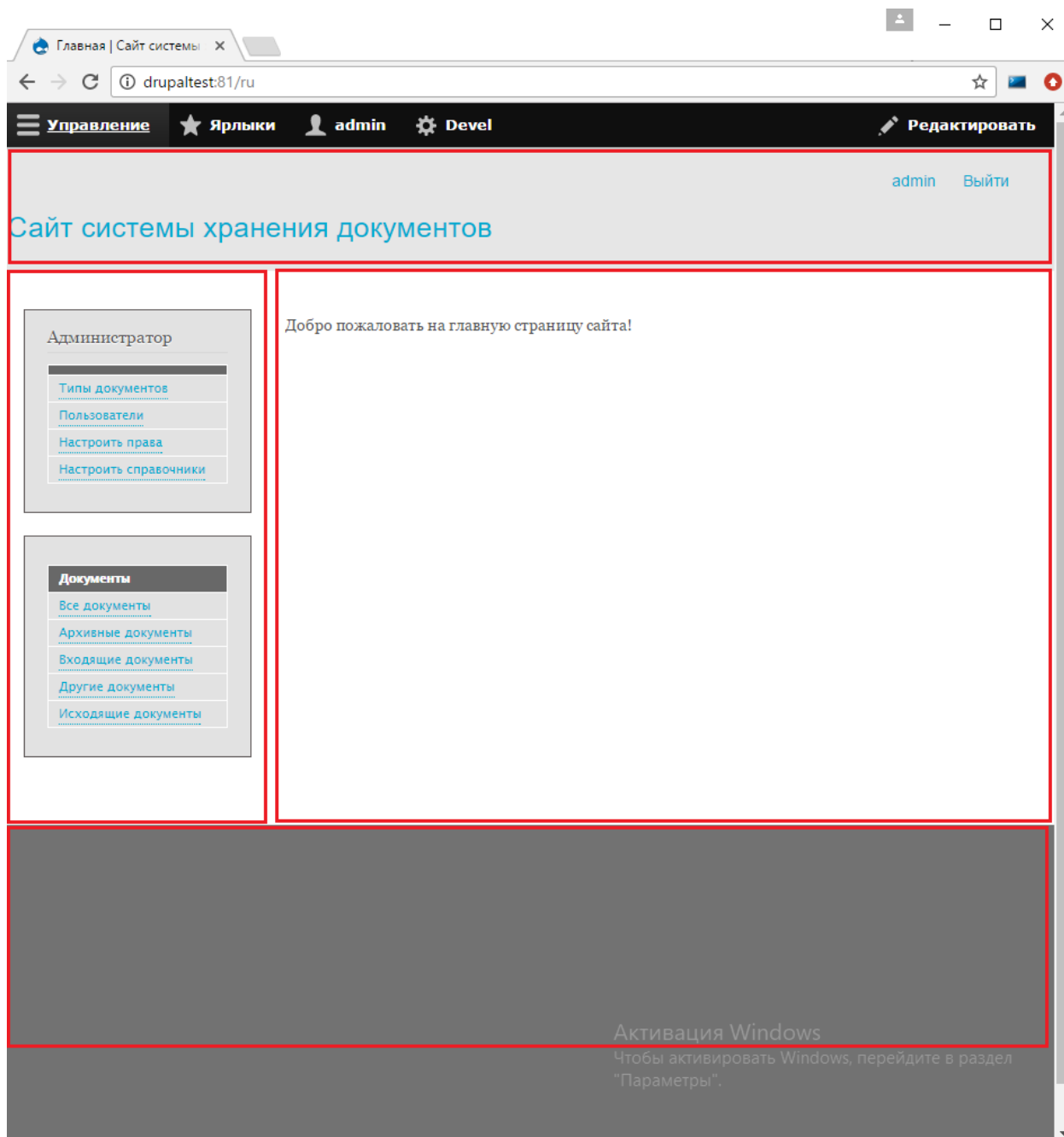


Рисунок 23 - Главная страница сайта

Страница поделена на четыре области: верхняя горизонтальная область (шапка сайта), область левой боковой панели, область контента и область подвала сайта.

Левая боковая панель содержит в себе два блока: блок возможностей администратора и блок быстрого доступа к документам.

На сайт можно зайти любым зарегистрированным пользователем. В зависимости от его прав содержимое левой боковой панели может меняться. Панель управления сайтом исчезнет (рис. 24).

Семахин Н. Р. | Сайт сис

← → ↻ ⓘ drupaltest:81/ru/user/20/edit

Семахин Н. Р. Выйти

Сайт системы хранения документов

Документы

- Все документы
- Архивные документы
- Входящие документы
- Другие документы
- Исходящие документы

Семахин Н. Р.

Текущий пароль

Требуется, если вы хотите изменить *E-mail* адрес или *Пароль* ниже. [Сбросить ваш пароль.](#)

E-mail адрес *

Действительный адрес электронной почты. Все электронные письма, отправляемые системой, будут высланы на данный адрес. Адрес электронной почты не станет публичным и будет использоваться только если Вы захотите получить новый пароль, определенные новости или уведомления на почту.

Пароль

Сложность пароля:

Повторите пароль

Пароли совпадают:
Для изменения текущего пароля, введите новый пароль в обоих полях.

Фамилия *

Инициалы *

Рисунок 24 - Страница сайта для зарегистрированного пользователя

Блок быстрого доступа к документам позволяет открыть список документов нужной категории (рис. 25). Отсюда можно добавить новый документ, редактировать существующий или же открыть представление (карточку) документа (рис.26).

Блок Администратора содержит несколько ссылок на формы, с помощью которых можно управлять сайтом системы. В подробном рассмотрении интерфейса, предоставляемого этими формами, нет необходимости. Он прост и интуитивно понятен.

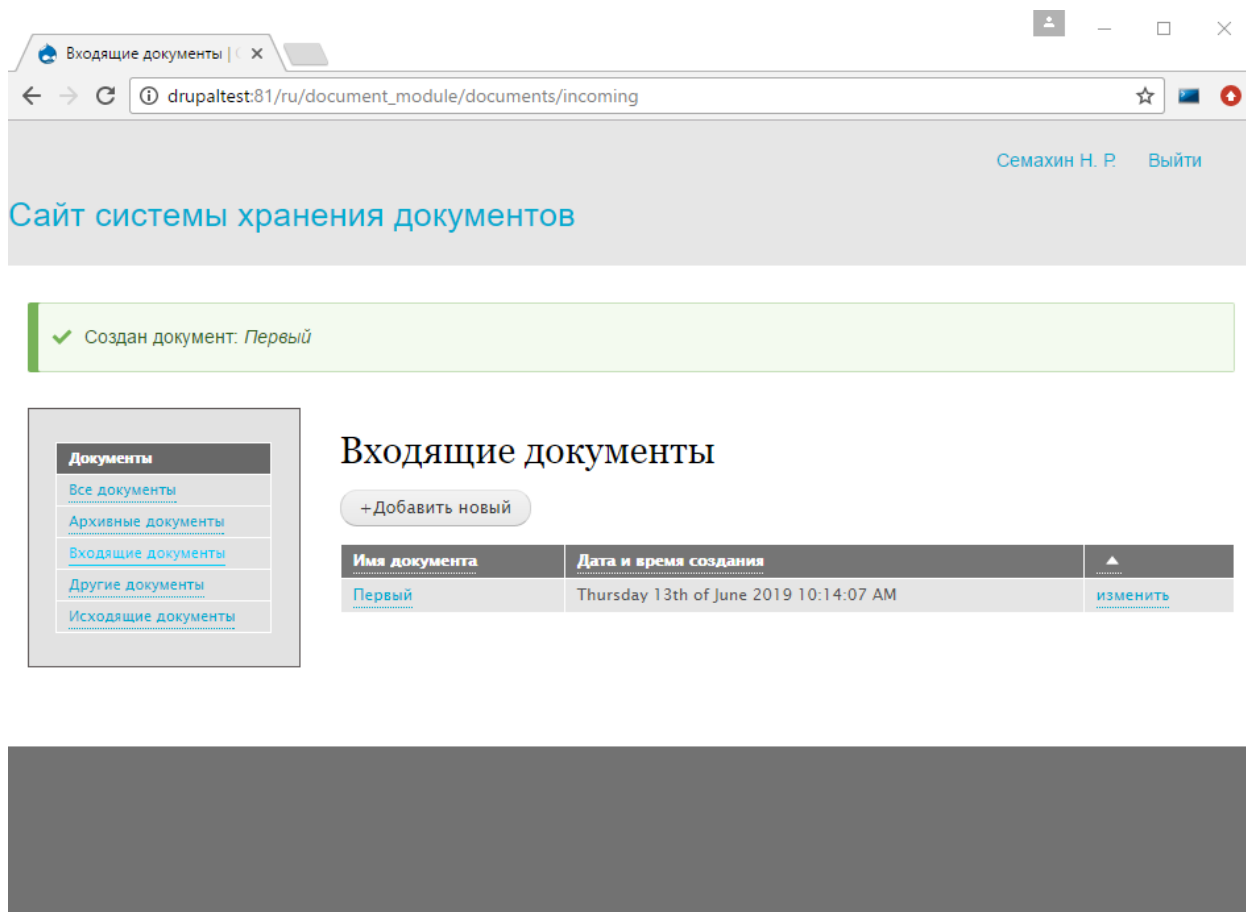


Рисунок 25 - Список всех входящих документов

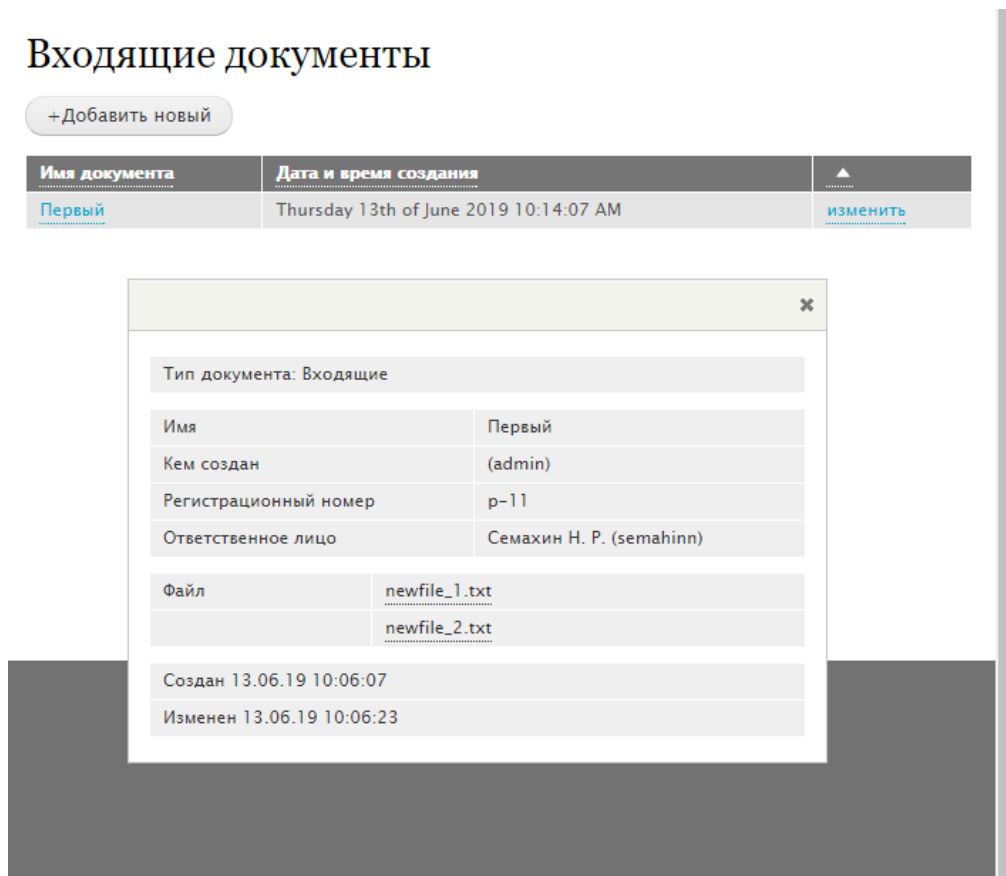


Рисунок 26 - Карточка входящего документа "Первый"

ЗАКЛЮЧЕНИЕ

В данной работе был реализован сайт системы электронного хранения и управления документами.

Поставленная цель была выполнена. Все задачи были решены в полном объеме.

В данной работе:

1) рассмотрены примеры существующих информационных систем электронного хранения документации и систем электронного документооборота, определены их характерные черты и область применения;

2) произведен анализ требований к разрабатываемому сайту системы, определены особенности его внешнего вида и функциональность;

3) рассмотрены существующие технологии и программные средства, используемые при разработке сайта, проанализированы возможности этих средств;

4) дано описание системы управления содержимым Drupal и её возможностей по управлению сайтом;

5) приведено описание хода разработки визуальной и программной составляющих сайта;

6) описана работа с сайтом, основные блоки сайта и навигация по ним.

Получен рабочий вариант сайта информационной системы, который в дальнейшем можно развивать. Расширение функциональных возможностей сайта системы электронного хранения и управления документами возможно в следующих направлениях:

1) внедрение средств для онлайн-просмотра документов;

2) добавление программного модуля для делопроизводства, который будет реализовать движение документа внутри организации;

3) добавление возможности извлечения документа пользователем для монопольного использования;

Основная работа по созданию сайта состояла в программировании на языке PHP и составлении конфигурационных файлов для модуля. В качестве среды разработки использовалась IDE PhpStorm и текстовый редактор SublimeText.

В ходе выполнения работы были получены навыки работы с CMS Drupal. Также получены навыки разработки сайтов. Приобретен дополнительный опыт работы с такими веб-технологиями как HTML, CSS, языком PHP.

СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ

1. Daniel Sipos. Drupal 8 Module Development. M.: Packt Publishing, 2017
2. Шлоссейгл Джордж. Профессиональное программирование на PHP. М.:Вильямс, 2006. - 413 с.
3. Jennifer Hodgdon. Programmer's Guide to Drupal, 2nd Edition. M.: O'Reilly Media, 2015. – 236 с.
4. Matt Glaman. Drupal 8 Development Cookbook.M.: Packt Publishing, 2016
5. Chaz Chumley. Drupal 8 Theming with TWIG. M.: Packt Publishing, 2016

ПРИЛОЖЕНИЕ А: ЛИСТИНГИ ПРОГРАММ

Приложение А.1 – Файл «DocumentEntity.php»

```
<?php

namespace Drupal\document_module\Entity;

use Drupal\Core\Entity\EntityStorageInterface;
use Drupal\Core\Field\BaseFieldDefinition;
use Drupal\Core\Entity\ContentEntityBase;
use Drupal\Core\Entity\EntityChangedTrait;
use Drupal\Core\Entity\EntityTypeInterface;
use Drupal\user\UserInterface;
use Drupal\user\Entity\User;
/**
 * Defines the Document entity entity.
 *
 * @ingroup document_module
 *
 * @ContentEntityType(
 *   id = "document_entity",
 *   label = @Translation("Document entity"),
 *   bundle_label = @Translation("Document entity type"),
 *   handlers = {
 *     "view_builder" = "Drupal\Core\Entity\EntityViewBuilder",
 *     "list_builder" = "Drupal\document_module\DocumentEntityListBuilder",
 *     "views_data" =
"Drupal\document_module\Entity\DocumentEntityViewsData",
 *     "translation" =
"Drupal\document_module\DocumentEntityTranslationHandler",
 *     "form" = {
 *       "default" = "Drupal\document_module\Form\DocumentEntityForm",
 *       "add" = "Drupal\document_module\Form\DocumentEntityForm",
 *       "edit" = "Drupal\document_module\Form\DocumentEntityForm",
 *       "delete" = "Drupal\document_module\Form\DocumentEntityDeleteForm",
 *     },
 *     "access" =
"Drupal\document_module\DocumentEntityAccessControlHandler",
 *     "route_provider" = {
 *       "html" = "Drupal\document_module\DocumentEntityHtmlRouteProvider",
 *     },
 *   },
 *   base_table = "document_entity",
 *   data_table = "document_entity_field_data",
 *   translatable = TRUE,
 *   admin_permission = "administer document entity",
 *   entity_keys = {
 *     "id" = "id",
 *     "bundle" = "type",
 *     "label" = "name",
 *     "uuid" = "uuid",
 *     "uid" = "creator",
 *     "langcode" = "langcode",
 *   },
 *   links = {
 *     "canonical" = "/admin/structure/document_entity/{document_entity}",
 *     "add-page" = "/admin/structure/document_entity/add",
```

```

*      "add-form" =
"/admin/structure/document_entity/add/{document_entity_type}",
*      "edit-form" =
"/admin/structure/document_entity/{document_entity}/edit",
*      "delete-form" =
"/admin/structure/document_entity/{document_entity}/delete",
*      "collection" = "/admin/structure/document_entity",
*  },
*  bundle_entity_type = "document_entity_type",
*  field_ui_base_route = "entity.document_entity_type.edit_form"
* )
*/
class DocumentEntity extends ContentEntityBase implements
DocumentEntityInterface {

    use EntityChangedTrait;

    /**
     * {@inheritdoc}
     */
    public static function preCreate(EntityStorageInterface
$storage_controller, array &$values) {
        parent::preCreate($storage_controller, $values);
    }

    /**
     * {@inheritdoc}
     */
    public function getName() {
        return $this->get('name')->value;
    }

    /**
     * {@inheritdoc}
     */
    public function setName($name) {
        $this->set('name', $name);
        return $this;
    }

    /**
     * {@inheritdoc}
     */
    public function getCreatedTime() {
        return $this->get('created')->value;
    }

    /**
     * {@inheritdoc}
     */
    public function setCreatedTime($timestamp) {
        $this->set('created', $timestamp);
        return $this;
    }

    /**
     * {@inheritdoc}
     */
    public function getOwner() {
        return $this->get('creator')->entity;
    }

    /**
     * {@inheritdoc}

```

```

    */
    public function getOwnerId() {
        return $this->get('creator')->target_id;
    }

    /**
     * {@inheritdoc}
     */
    public function setOwnerId($uid) {
        $this->set('creator', $uid);
        return $this;
    }

    /**
     * {@inheritdoc}
     */
    public function setOwner(UserInterface $account) {
        $this->set('creator', $account->id());
        return $this;
    }

    /**
     * {@inheritdoc}
     */

    /**
     public function isPublished() {
         return (bool) $this->getEntityKey('status');
     }
     */

    /**
     * {@inheritdoc}
     */

    /**
     public function setPublished($published) {
         $this->set('status', $published ? TRUE : FALSE);
         return $this;
     }
     */

    /**
     * {@inheritdoc}
     */
    public static function baseFieldDefinitions(EntityTypeInterface
    $entity_type) {
        $fields = parent::baseFieldDefinitions($entity_type);

        $fields['name'] = BaseFieldDefinition::create('string')
            ->setLabel(t('Имя'))
            ->setDescription(t('Название документа'))
            ->setSettings([
                'max_length' => 50,
                'text_processing' => 0,
            ])
            ->setDefaultValue('')
            ->setDisplayOptions('view', [
                'label' => 'above',
                'type' => 'string',
                'weight' => -4,
            ])
        ];
    }

```

```

->setDisplayOptions('form', [
    'type' => 'string_textfield',
    'weight' => -4,
])
->setDisplayConfigurable('form', TRUE)
->setDisplayConfigurable('view', TRUE)
->setRequired(TRUE);

$fields['creator'] = BaseFieldDefinition::create('entity_reference')
->setLabel(t('Кем создан'))
->setDescription(t('The user ID of author of the Document entity
entity.'))
->setSetting('target_type', 'user')
->setSetting('handler', 'default')
->setDisplayOptions('view', [
    'label' => 'above',
    'type' => 'author',
    'weight' => 0,
])
->setDefaultValue(\Drupal::currentUser()->id())
->setDisplayConfigurable('form', FALSE)
->setDisplayConfigurable('view', TRUE);

/**
 * $fields['status'] = BaseFieldDefinition::create('boolean')
 *   ->setLabel(t('Publishing status'))
 *   ->setDescription(t('A boolean indicating whether the Document entity is
published.'))
 *   ->setRevisionable(TRUE)
 *   ->setDefaultValue(TRUE)
 *   ->setDisplayOptions('form', [
 *       'type' => 'boolean_checkbox',
 *       'weight' => -3,
 *   ]);
 */

/**
 * registration_number (Регистрационный номер)
 */

$fields['registration_number'] = BaseFieldDefinition::create('string')
->setLabel(t('Регистрационный номер'))
->setSettings([
    'max_length' => 255,
    'text_processing' => 0,
])
->setRequired(FALSE)
->setDefaultValue('')
->setDisplayOptions('view', [
    'label' => 'above',
    'type' => 'string',
    'weight' => -4,
])
->setDisplayOptions('form', [
    'type' => 'string_textfield',
    'weight' => -4,
])
->setDisplayConfigurable('form', TRUE)
->setDisplayConfigurable('view', TRUE);

/**
 * responsible (Ответственное лицо) (пользователь или группа)

```

```

*/

$fields['responsible'] = BaseFieldDefinition::create('entity_reference')
  ->setLabel(t('Ответственное лицо'))
  ->setRequired(FALSE)
  ->setSetting('target_type', 'user')
  ->setSetting('handler', 'default')
  ->setDisplayOptions('view', [
    'label' => 'above',
    'type' => 'author',
    'weight' => -4,
  ])
  ->setDisplayOptions('form', [
    'type' => 'entity_reference_autocomplete',
    'weight' => -4,
  ])
  ->setDisplayConfigurable('form', TRUE)
  ->setDisplayConfigurable('view', TRUE);

$fields['created'] = BaseFieldDefinition::create('created')
  ->setLabel(t('Создано'))
  ->setDescription(t('The time that the entity was created.));

$fields['changed'] = BaseFieldDefinition::create('changed')
  ->setLabel(t('Изменено'))
  ->setDescription(t('The time that the entity was last edited.));

return $fields;
}
}

```

Приложение A.2 – Файл «DocumentEntityType.php»

```

<?php

namespace Drupal\document_module\Entity;

use Drupal\Core\Config\Entity\ConfigEntityBundleBase;

/**
 * Defines the Document entity type entity.
 *
 * @ConfigEntityType(
 *   id = "document_entity_type",
 *   label = @Translation("Document entity type"),
 *   handlers = {
 *     "view_builder" = "Drupal\Core\Entity\EntityViewBuilder",
 *     "list_builder" =
"Drupal\document_module\DocumentEntityTypeListBuilder",
 *     "form" = {
 *       "add" = "Drupal\document_module\Form\DocumentEntityTypeForm",
 *       "edit" = "Drupal\document_module\Form\DocumentEntityTypeForm",
 *       "delete" =
"Drupal\document_module\Form\DocumentEntityTypeDeleteForm"
 *     },
 *     "route_provider" = {
 *       "html" =
"Drupal\document_module\DocumentEntityTypeHtmlRouteProvider",
 *     },

```

```

*   },
*   config_prefix = "document_entity_type",
*   admin_permission = "administer document entity types",
*   bundle_of = "document_entity",
*   entity_keys = {
*     "id" = "id",
*     "label" = "label",
*     "uuid" = "uuid"
*   },
*   links = {
*     "canonical" =
"/admin/structure/document_entity_type/{document_entity_type}",
*     "add-form" = "/admin/structure/document_entity_type/add",
*     "edit-form" =
"/admin/structure/document_entity_type/{document_entity_type}/edit",
*     "delete-form" =
"/admin/structure/document_entity_type/{document_entity_type}/delete",
*     "collection" = "/admin/structure/document_entity_type"
*   }
* )
*/
class DocumentEntityType extends ConfigEntityBundleBase implements
DocumentEntityTypeInterface {

  /**
   * The Document entity type ID.
   *
   * @var string
   */
  protected $id;

  /**
   * The Document entity type label.
   *
   * @var string
   */
  protected $label;
}

```

Приложение A.3 — Файл

«DocumentFormControllerClass.php

```

<?php

namespace Drupal\document_module\Controller;

use Drupal\Core\Controller\ControllerBase;
use Drupal\document_module\Entity\DocumentEntityType;
use http\Env\Request;
use Drupal\Core\Field\BaseFieldDefinition;
use Drupal\Core\Field\FieldItemInterface;
use Drupal\user\Entity\User;
use Drupal\document_module\Entity\DocumentEntity;
use Drupal\file\Entity\File;
use Drupal\Core\Url;
use Drupal\Core\Link;

```

```

class DocumentFormControllerClass extends ControllerBase{

    public function getMainPage()
    {

        $form['description'] = [
            '#type' => 'item',
            '#markup' => $this->t('Добро пожаловать на главную страницу сайта!'),
        ];

        return $form;

    }

    public function showCardPropList($docid) {

        /**Загрузка entity document по id*/
        $doccard_storage = \Drupal::entityTypeManager()-
>getStorage('document_entity');
        $doccard = $doccard_storage->load($docid);

        /**Формируем массив рендеринга*/
        $doccard_fields = array();
        $created = NULL;
        $changed = NULL;

        foreach($doccard->getFieldDefinitions() as $field_definition)
        {
            $machine_name = $field_definition->getName();
            $display_view_options = $field_definition->getDisplayOptions('view');

            if ($display_view_options != NULL)
            {
                $field_label = strval($field_definition->getLabel());

                if ($field_definition->getType() == 'entity_reference')
                {
                    $settings = $field_definition->getSettings();
                    if ($settings['target_type'] == "user") {
                        $itemlist = $doccard->get($machine_name);
                        $values = $itemlist->getValue();

                        if (isset($values['0']['target_id'])) {
                            $entity_id = $values['0']['target_id'];
                            $current_user = User::load($entity_id);
                            $account_name = $current_user->getAccountName();
                            $inicialy = $current_user->get('field_inicialy')->getValue();
                            $surname = $current_user->get('field_surname')->getValue();
                            if (isset($inicialy['0']['value'])) {
                                $inicialy = $inicialy['0']['value'];
                            }
                            else {
                                $inicialy = '';
                            }
                        }
                        if (isset($surname['0']['value'])) {

```

```

        $surname = $surname['0']['value'];
    }
    else {
        $surname = '';
    }
    $value = $surname . ' ' . $inicialy . ' (' . $account_name .
')';

    $doccard_fields[$field_label] = $value;
}
}

if ($field_definition->getType() == 'string')
{
    $value = $doccard->get($machine_name)->value;
    if ($value != null)
    {
        $doccard_fields[$field_label] = $value;
    }
}

if ($field_definition->getType() == 'file')
{
    $itemlist = $doccard->get($machine_name);
    $values = $itemlist->getValue();
    foreach ($values as $value)
    {
        $entity_id = $value['target_id'];
        $current_file = File::load($entity_id);
        $uri = $current_file->getFileUri();
        $name = $current_file->getFilename();
        $parts = explode('/:/', $uri);
        $url = 'internal:/file-download/download/' . $parts[0] . '/' .
$entity_id;
        $url = Url::fromUri($url);
        $link = Link::fromTextAndUrl($name, $url);
        $file_links[] = $link;
    }
}

/** выводим несмотря на параметры отображения */
if($machine_name == "created") {
    $created = date("d.m.y H:m:s", $doccard->get($machine_name)->value);
}
if($machine_name == "changed") {
    $changed = date("d.m.y H:m:s", $doccard->get($machine_name)->value);
}
if($machine_name == "type") {
    $itemlist = $doccard->get($machine_name);
    $values = $itemlist->getValue();
    $entity_id = $values['0']['target_id'];

    $current_document_type = DocumentEntityType::load($entity_id);
    $type = $current_document_type->get('label');
}

}

```

```

    $build['#theme'] = 'events_get_card_proplist';
    $build['#created'] = $created;
    $build['#changed'] = $changed;
    $build['#type'] = $type;
    $build['#file_links'] = $file_links;
    $build['#doccardfields'] = $doccard_fields;

    return $build;

}

}

```

Приложение A.4 — Файл

«DocumentOwnEntityControllerClass.php

```

<?php
/**
 * Created by PhpStorm.
 * User: SemakhinPc
 * Date: 01.06.2019
 * Time: 14:41
 */

namespace Drupal\document_module\Controller;

use Drupal\Core\Entity\Controller\EntityController;

use Drupal\Core\Entity\EntityDescriptionInterface;
use Drupal\Core\Entity\EntityInterface;
use Drupal\Core\Entity\EntityRepositoryInterface;
use Drupal\Core\Entity\EntityTypeBundleInfoInterface;
use Drupal\Core\Entity\EntityTypeManagerInterface;
use Drupal\Core\Entity\EntityTypeInterface;
use Drupal\Core\Link;
use Drupal\Core\Render\RendererInterface;
use Drupal\Core\Routing\RouteMatchInterface;
use Drupal\Core\Routing\UrlGeneratorInterface;
use Drupal\Core\Routing\UrlGeneratorTrait;
use Drupal\Core\DependencyInjection\ContainerInjectionInterface;
use Drupal\Core\StringTranslation\StringTranslationTrait;
use Drupal\Core\StringTranslation\TranslationInterface;
use Symfony\Component\DependencyInjection\ContainerInterface;
/**
 * Provides the add-page and title callbacks for entities.
 *
 * It provides:
 * - The add-page callback.
 * - An add title callback for entity types.
 * - An add title callback for entity types with bundles.
 * - A view title callback.
 * - An edit title callback.
 * - A delete title callback.
 */
class DocumentOwnEntityController extends EntityController {

```

```

    public function editTitle(RouteMatchInterface $route_match, EntityInterface
    $_entity = NULL) {
        if ($entity = $this->doGetEntity($route_match, $_entity)) {
            return $this->t('Редактировать "%label"', ['%label' => $entity-
            >label()]);
        }
    }

    public function addBundleTitle(RouteMatchInterface $route_match,
    $entity_type_id, $bundle_parameter) {
        $bundles = $this->entityTypeBundleInfo->getBundleInfo($entity_type_id);
        // If the entity has bundle entities, the parameter might have been
        upcasted
        // so fetch the raw parameter.
        $bundle = $route_match->getRawParameter($bundle_parameter);
        if ((count($bundles) > 1) && isset($bundles[$bundle])) {
            return $this->t('Добавить @bundle', ['@bundle' =>
            $bundles[$bundle]['label']]);
        }
        // If the entity supports bundles generally, but only has a single
        bundle,
        // the bundle is probably something like 'Default' so that it preferable
        to
        // use the entity type label.
        else {
            return $this->addTitle($entity_type_id);
        }
    }

    public function addTitle($entity_type_id) {
        $entity_type = $this->entityTypeManager->getDefinition($entity_type_id);
        return $this->t('Добавить @entity-type', ['@entity-type' => $entity_type-
        >getLowercaseLabel()]);
    }
}

```

Приложение A.5 – Файл «DocumentEntityForm.php»

```

<?php

namespace Drupal\document_module\Form;

use Drupal\Core\Entity\ContentEntityForm;
use Drupal\Core\Form\FormStateInterface;

/**
 * Form controller for Document entity edit forms.
 *
 * @ingroup document_module
 */
class DocumentEntityForm extends ContentEntityForm {

    /**
     * {@inheritdoc}
     */

    public function buildForm(array $form, FormStateInterface $form_state) {
        /* @var $entity \Drupal\document_module\Entity\DocumentEntity */
        $form = parent::buildForm($form, $form_state);

        $entity = $this->entity;
    }
}

```

```

        return $form;
    }

    protected function actions(array $form, FormStateInterface $form_state) {
        // @todo Consider renaming the action key from submit to save. The
        impacts
        //     are hard to predict. For example, see
        //
        \Drupal\language\Element\LanguageConfiguration::processLanguageConfiguration(
        ).
        $actions['submit'] = [
            '#type' => 'submit',
            '#value' => $this->t('Сохранить'),
            '#submit' => [ '::submitForm', '::save'],
        ];

        if (!$this->entity->isNew() && $this->entity->hasLinkTemplate('delete-
        form')) {
            $route_info = $this->entity->toUrl('delete-form');
            if ($this->getRequest()->query->has('destination')) {
                $query = $route_info->getOption('query');
                $query['destination'] = $this->getRequest()->query-
                >get('destination');
                $route_info->setOption('query', $query);
            }
            $actions['delete'] = [
                '#type' => 'link',
                '#title' => $this->t('Удалить'),
                '#access' => $this->entity->access('delete'),
                '#attributes' => [
                    'class' => ['button', 'button--danger'],
                ],
            ];
            $actions['delete']['#url'] = $route_info;
        }

        return $actions;
    }
    /**
     * {@inheritdoc}
     */
    public function save(array $form, FormStateInterface $form_state) {
        $entity = $this->entity;

        $entity_type = $entity->bundle();
        $status = parent::save($form, $form_state);

        switch ($status) {
            case SAVED_NEW:
                drupal_set_message($this->t('Создан документ: %label', [
                    '%label' => $entity->label(),
                ]));
                break;

            default:
                drupal_set_message($this->t('Сохранен документ %label', [
                    '%label' => $entity->label(),
                ]));
        }
        $form_state->setRedirect('document_module.documents', ['document_type' =>
        $entity_type]);
    }

```

}

«DocumentEntityPermissionForm.php

```
<?php

namespace Drupal\document_module\Form;

use Drupal\Core\Extension\ModuleHandlerInterface;
use Drupal\Core\Form\FormBase;
use Drupal\Core\Form\FormStateInterface;
use Drupal\user\PermissionHandlerInterface;
use Drupal\user\RoleStorageInterface;
use Symfony\Component\DependencyInjection\ContainerInterface;

/**
 * Provides the user permissions administration form.
 *
 * @internal
 */
class DocumentEntityPermissionsForm extends FormBase {

  /**
   * The permission handler.
   *
   * @var \Drupal\user\PermissionHandlerInterface
   */
  protected $permissionHandler;

  /**
   * The role storage.
   *
   * @var \Drupal\user\RoleStorageInterface
   */
  protected $roleStorage;

  /**
   * The module handler.
   *
   * @var \Drupal\Core\Extension\ModuleHandlerInterface
   */
  protected $moduleHandler;

  /**
   * Constructs a new UserPermissionsForm.
   *
   * @param \Drupal\user\PermissionHandlerInterface $permission_handler
   *   The permission handler.
   * @param \Drupal\user\RoleStorageInterface $role_storage
   *   The role storage.
   * @param \Drupal\Core\Extension\ModuleHandlerInterface $module_handler
   *   The module handler.
   */
  public function __construct(PermissionHandlerInterface $permission_handler,
    RoleStorageInterface $role_storage, ModuleHandlerInterface $module_handler) {
    $this->permissionHandler = $permission_handler;
    $this->roleStorage = $role_storage;
    $this->moduleHandler = $module_handler;
  }
}
```

```

/**
 * {@inheritdoc}
 */
public static function create(ContainerInterface $container) {
    return new static(
        $container->get('user.permissions'),
        $container->get('entity.manager')->getStorage('user_role'),
        $container->get('module_handler')
    );
}

/**
 * {@inheritdoc}
 */
public function getFormId() {
    return 'user_admin_permissions';
}

/**
 * Gets the roles to display in this form.
 *
 * @return \Drupal\user\RoleInterface[]
 *   An array of role objects.
 */
protected function getRoles() {
    return $this->roleStorage->loadMultiple();
}

/**
 * {@inheritdoc}
 */
public function buildForm(array $form, FormStateInterface $form_state) {
    $role_names = [];
    $role_permissions = [];
    $admin_roles = [];
    foreach ($this->getRoles() as $role_name => $role) {
        // Retrieve role names for columns.
        $role_names[$role_name] = $role->label();
        // Fetch permissions for the roles.
        $role_permissions[$role_name] = $role->getPermissions();
        $admin_roles[$role_name] = $role->isAdmin();
    }

    // Store $role_names for use when saving the data.
    $form['role_names'] = [
        '#type' => 'value',
        '#value' => $role_names,
    ];
    // Render role/permission overview:
    $hide_descriptions = system_admin_compact_mode();

    $form['system_compact_link'] = [
        '#id' => FALSE,
        '#type' => 'system_compact_link',
    ];

    $form['permissions'] = [
        '#type' => 'table',
        '#header' => [$this->t('Permission')],
        '#id' => 'permissions',
        '#attributes' => ['class' => ['permissions', 'js-permissions']],
        '#sticky' => TRUE,
    ];
}

```

```

foreach ($role_names as $name) {
    $form['permissions']['#header'][] = [
        'data' => $name,
        'class' => ['checkbox'],
    ];
}

$permissions = $this->permissionHandler->getPermissions();
$permissions_by_provider = [];
foreach ($permissions as $permission_name => $permission) {
    if($permission['provider'] == 'document_module') {

$permissions_by_provider[$permission['provider']][$permission_name] =
$permission;
    }
}

foreach ($permissions_by_provider as $provider => $permissions) {

    foreach ($permissions as $perm => $perm_item) {
        // Fill in default values for the permission.
        $perm_item += [
            'description' => '',
            'restrict access' => FALSE,
            'warning' => !empty($perm_item['restrict access']) ? $this-
>t('Warning: Give to trusted roles only; this permission has security
implications.') : '',
        ];
        $form['permissions'][$perm]['description'] = [
            '#type' => 'inline_template',
            '#template' => '<div class="permission"><span class="title">{{
title }}</span>{% if description or warning %}<div class="description">{% if
warning %}<em class="permission-warning">{{ warning }}</em> {% endif %}{{
description }}</div>{% endif %}</div>',
            '#context' => [
                'title' => $perm_item['title'],
            ],
        ];
        // Show the permission description.
        if (!$hide_descriptions) {

$form['permissions'][$perm]['description']['#context']['description'] =
$perm_item['description'];
        $form['permissions'][$perm]['description']['#context']['warning'] =
$perm_item['warning'];
        }
        foreach ($role_names as $rid => $name) {
            $form['permissions'][$perm][$rid] = [
                '#title' => $name . ': ' . $perm_item['title'],
                '#title_display' => 'invisible',
                '#wrapper_attributes' => [
                    'class' => ['checkbox'],
                ],
                '#type' => 'checkbox',
                '#default_value' => in_array($perm, $role_permissions[$rid]) ? 1
: 0,
                '#attributes' => ['class' => ['rid-' . $rid, 'js-rid-' . $rid]],
                '#parents' => [$rid, $perm],
            ];
            // Show a column of disabled but checked checkboxes.
            if ($admin_roles[$rid]) {
                $form['permissions'][$perm][$rid]['#disabled'] = TRUE;
                $form['permissions'][$perm][$rid]['#default_value'] = TRUE;
            }
        }
    }
}

```

```

    }
  }
}

$form['actions'] = ['#type' => 'actions'];
$form['actions']['submit'] = [
  '#type' => 'submit',
  '#value' => $this->t('Сохранить настройки'),
  '#button_type' => 'primary',
];

$form['#attached']['library'][] = 'user/drupal.user.permissions';

return $form;
}

/**
 * {@inheritdoc}
 */
public function submitForm(array &$form, FormStateInterface $form_state) {
  foreach ($form_state->getValue('role_names') as $role_name => $name) {
    user_role_change_permissions($role_name, (array) $form_state->
    >getValue($role_name));
  }

  $this->messenger()->addStatus($this->t('Изменения сохранены'));
}
}

```

Приложение А.7 – Файл «DocumentEntityTypeForm.php»

```

<?php

namespace Drupal\document_module\Form;

use Drupal\Core\Entity\EntityForm;
use Drupal\Core\Form\FormStateInterface;

/**
 * Class DocumentEntityTypeForm.
 */
class DocumentEntityTypeForm extends EntityForm {

  /**
   * {@inheritdoc}
   */
  public function form(array $form, FormStateInterface $form_state) {
    $form = parent::form($form, $form_state);

    $document_entity_type = $this->entity;
    $form['label'] = [
      '#type' => 'textfield',
      '#title' => $this->t('Имя'),
      '#maxlength' => 255,
      '#default_value' => $document_entity_type->label(),
      '#description' => $this->t("Название для типа документа"),
      '#required' => TRUE,
    ];
  }
}

```

```

        $form['id'] = [
            '#type' => 'machine_name',
            '#default_value' => $document_entity_type->id(),
            '#machine_name' => [
                'exists' =>
'\Drupal\document_module\Entity\DocumentEntityType::load',
            ],
            '#disabled' => !$document_entity_type->isNew(),
            '#description' => $this->t("Уникальное системное имя типа документа
(только латиница)"),
        ];

        /* You will need additional form elements for your custom properties. */

        return $form;
    }

    /**
     * {@inheritdoc}
     */
    public function save(array $form, FormStateInterface $form_state) {
        $document_entity_type = $this->entity;
        $status = $document_entity_type->save();

        switch ($status) {
            case SAVED_NEW:
                drupal_set_message($this->t('Создан новый тип документа: %label', [
                    '%label' => $document_entity_type->label(),
                ]));
                break;

            default:
                drupal_set_message($this->t('Создан новый тип документа: %label', [
                    '%label' => $document_entity_type->label(),
                ]));
        }
        $form_state->setRedirectUrl($document_entity_type->toUrl('collection'));
    }
}

```

Приложение A.8 – Файл «DocumentEntityListForm.php

```

<?php

namespace Drupal\document_module\Form;

use Drupal\Core\Entity\EntityForm;
use Drupal\Core\Form\FormBase;
use Drupal\Core\Form\FormStateInterface;
use Drupal\document_module\Entity\DocumentEntityType;
use Drupal\document_module\Entity\DocumentEntity;
use Drupal\Core\Link;
use Drupal\Component\Render\FormattableMarkup;
use Drupal\Component\Serialization\Json;
/**
 * Форма для вывода информации о документе определенного типа (не работает)
 */
class DocumentListForm extends FormBase {

```

```

public function buildRow($entity)
{
    $created_date = date("l jS \of F Y h:i:s A", $entity->get('created')->value);

    /**
     * $row['type'] = $entity->bundle();
     *
     */

    /**
     $row['name'] = Link::createFromRoute(
         $entity->name->value,
         'entity.document_entity.canonical',
         ['document_entity' => $entity->id()],
         [
             'attributes' => [
                 'class' => ['use-ajax'],
                 'data-dialog-type' => 'modal',
                 'data-dialog-options' => Json::encode([
                     'width' => 400, 'position' => ['my' => 'center', 'at' =>
'center', 'of' => 'window'],
                 ]),
             ]
         ]
     );
     */

    $row['name'] = Link::createFromRoute(
        $entity->name->value,
        'document_module.showcardproplist',
        ['docid' => $entity->id()],
        [
            'attributes' => [
                'class' => ['use-ajax'],
                'data-dialog-type' => 'modal',
                'data-dialog-options' => Json::encode([
                    'width' => 400, 'position' => ['my' => 'center', 'at' =>
'center', 'of' => 'window'],
                ]),
            ]
        ]
    );

    /**
     * $row['name'] = $entity->name->value;
     *
     */
    $row['RegistrationDate'] = $created_date;

    $row[''] = Link::createFromRoute(
        'изменить',
        'entity.document_entity.edit_form',
        ['document_entity' => $entity->id()]);

    /**
     * используем роут из doccard
     */

```

```

    /**
    $row[''] = Link::createFromRoute(
        'изменить',
        'doccard.editcardproplist',
        ['docid' => $entity->id()]);

    */

    return $row;
}

public function buildHeader()
{
    $header = array(
        // We make it sortable by name.
        /**
        * array('data' => $this->t('Тип'), 'field' => 'type', 'sort' =>
'asc'),
        *
        */
        array('data' => $this->t('Имя документа'), 'field' => 'name', 'sort' =>
'asc'),
        array('data' => $this->t('Дата и время создания'), 'field' =>
'RegistrationDate', 'sort' => 'asc'),
        array('data' => $this->t(''), 'field' => 'Edit'),
    );

    return $header;
}

public function getFormId() {
    return 'document_list_form';
}

public function buildForm(array $form, FormStateInterface $form_state,
$document_type = NULL) {

    $build['#document_type_id'] = $document_type;

    $build['add_button'] = [
        '#type' => 'submit',
        '#value' => t('Добавить новый'),
    ];

    $build['table'] = [
        '#type' => 'table',
        '#header' => $this->buildHeader(),
        '#rows' => [],
        '#title' => '',
    ];

    $table_title = '';
    $document_types = DocumentEntityType::loadMultiple();

    if (isset($document_types[$document_type]))
    {
        $table_title = $document_types[$document_type]->get('label');

        $build['table']['#title'] = $table_title.' документы';

        $dids = \Drupal::entityQuery('document_entity')->condition('type',
$document_type)->execute();
    }
}

```

```

        if ($dids != null)
        {
            $document_storage = \Drupal::entityTypeManager()-
>getStorage('document_entity');

            foreach ($dids as $did)
            {
                $document = $document_storage->load($did);
                $document_fields = $document->getFieldDefinitions();
                foreach ($document_fields as $field)
                {
                    /**
                     * $per_name = $field->getName();
                     * $per_label = $field->getLabel();
                     * $per_decsr = $field->getDescription();
                     * $per_bunble = $field->getTargetBundle();
                     * $per_ttype = $field->getTargetEntityTypeId();
                     * */

                }

                $build['table']['#rows'][$did] = $this->buildRow($document);
            }
        }
        else{

            $build['description'] = [
                '#type' => 'item',
                '#markup' => $this->t('Список пуст'),
            ];
        }

        // Finally add the pager.
        $build['pager'] = array(
            '#type' => 'pager'
        );

    }

    return $build;

}

public function validateForm(array &$form, FormStateInterface $form_state) {

}

public function submitForm(array &$form, FormStateInterface $form_state) {

    $document_types = DocumentEntityType::loadMultiple();
    $sid = $form['#document_type_id'];

    $form_state->setRedirect('entity.document_entity.add_form',
    ['document_entity_type' => $sid]);
}

}

```

Приложение А.9 – Файл «DocumentCategoriesBlock.php»

```
<?php

namespace Drupal\document_module\Plugin\Block;

use Drupal\Core\Access\AccessResult;
use Drupal\Core\Access\AccessResultInterface;
use Drupal\Core\Ajax\AjaxResponse;
use Drupal\Core\Ajax\RemoveCommand;
use Drupal\Core\Controller\ControllerBase;
use Drupal\Core\Session\AccountInterface;
use Drupal\Core\Link;
use Drupal\Core\Url;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Exception\NotFoundHttpException;
use Drupal\Core\Block\BlockBase;
use Drupal\document_module\Entity\DocumentEntityType;
use Drupal\document_module\Entity\DocumentEntity;
/**
 * Блок ссылок на добавление документов
 *
 * @Block(
 *   id = "document_categories_block",
 *   admin_label = @Translation("Категории документов"),
 * )
 */
class DocumentCategoriesBlock extends BlockBase {

  public function build() {

    $build['add_table'] = [
      '#type' => 'table',
      '#caption' => $this->t(''),
      '#header' => [$this->t('Документы')],
      '#rows' => [],
    ];

    $document_types = DocumentEntityType::loadMultiple();

    if ($document_types != NULL)
    {
      $build['add_table']['#rows'][] = [Link::fromTextAndUrl('Все документы',
new Url('entity.document_entity.collection'))];

      foreach ($document_types as $current_document_type)
      {
        $label = $current_document_type->get('label');

        $build['add_table']['#rows'][] = [Link::fromTextAndUrl($label . '
документы', Url::fromRoute('document_module.documents', ['document_type' =>
$current_document_type->get('id')]))];
      }
    }

    return $build;
  }
}
```

```
}
```

Приложение А.10 – Файл «AdminOptionsBlock.php»

```
<?php

namespace Drupal\document_module\Plugin\Block;

use Drupal\Core\Access\AccessResult;
use Drupal\Core\Access\AccessResultInterface;
use Drupal\Core\Ajax\AjaxResponse;
use Drupal\Core\Ajax\RemoveCommand;
use Drupal\Core\Controller\ControllerBase;
use Drupal\Core\Session\AccountInterface;
use Drupal\Core\Link;
use Drupal\Core\Url;
use Symfony\Component\HttpFoundation\Request;
use Symfony\Component\HttpFoundation\Exception\NotFoundHttpException;
use Drupal\Core\Block\BlockBase;
use Drupal\document_module\Entity\DocumentEntityType;
use Drupal\document_module\Entity\DocumentEntity;
/**
 * Блок ссылок на добавление документов
 */
* @Block(
*   id = "admin_options_block",
*   admin_label = @Translation("Администратор"),
* )
*/
class AdminOptionsBlock extends BlockBase {

  protected function blockAccess(AccountInterface $account) {

    // By default, the block is visible.
    return AccessResult::allowed();
  }

  public function build() {

    $build['add_table'] = [
      '#type' => 'table',
      '#caption' => $this->t(''),
      '#header' => [$this->t('')],
      '#rows' => [],
    ];

    $build['add_table']['#rows'][] = [Link::fromTextAndUrl('Типы документов',
new Url('entity.document_entity_type.collection'))];
    $build['add_table']['#rows'][] = [Link::fromTextAndUrl('Пользователи',
new Url('entity.user.collection'))];
    $build['add_table']['#rows'][] = [Link::fromTextAndUrl('Настроить права',
new Url('document_module.permission_form'))];
    $build['add_table']['#rows'][] = [Link::fromTextAndUrl('Настроить
справочники', new Url('entity.taxonomy_vocabulary.collection'))];

    return $build;
  }
}
```

```
}  
}
```

Приложение А.11 – Файл «CabinetBlock.php»

```
<?php  
  
namespace Drupal\document_module\Plugin\Block;  
  
use Drupal\Core\Access\AccessResult;  
use Drupal\Core\Access\AccessResultInterface;  
use Drupal\Core\Ajax\AjaxResponse;  
use Drupal\Core\Ajax\RemoveCommand;  
use Drupal\Core\Controller\ControllerBase;  
use Drupal\Core\Session\AccountInterface;  
use Drupal\Core\Link;  
use Drupal\Core\Url;  
use Symfony\Component\HttpFoundation\Request;  
use Symfony\Component\HttpFoundation\Exception\NotFoundHttpException;  
use Drupal\Core\Block\BlockBase;  
use Drupal\document_module\Entity\DocumentEntityType;  
use Drupal\document_module\Entity\DocumentEntity;  
use Drupal\user\Entity\User;  
/**  
 * Блок ссылок на добавление документов  
 */  
 * @Block(  
 *   id = "cabinet_block",  
 *   admin_label = @Translation("Личный кабинет"),  
 * )  
 */  
class CabinetBlock extends BlockBase {  
  
  public function build() {  
  
    /**  
    $build['add_table'] = [  
      '#type' => 'table',  
      '#caption' => $this->t(''),  
      '#header' => [$this->t('')],  
      '#rows' => [],  
    ];  
  
    $build['add_table']['#rows'][] = [Link::fromTextAndUrl('Типы документов',  
new Url('entity.document_entity_type.collection'))];  
    $build['add_table']['#rows'][] = [Link::fromTextAndUrl('Пользователи',  
new Url('entity.user.collection'))];  
    $build['add_table']['#rows'][] = [Link::fromTextAndUrl('Настроить права',  
new Url('document_module.permission_form'))];  
  
    */  
    $current_user_id = \Drupal::currentUser()->id();  
  
    $user = User::load(\Drupal::currentUser()->id());  
  
    $user_name = $user->get('name')->value;  
    $user_initialy = $user->get('field_initialy')->value;  
    $user_surname = $user->get('field_surname')->value;
```

```

$link_user = '';
$link_logout = '';

if ($current_user_id == 0)
{
    $link_user = [
        '#title' => $this->t('Войти'),
        '#type' => 'link',
        '#url' => Url::fromRoute('user.login'),
    ];
}
else if ($user_initialy == null | $user_surname == null)
{
    $link_user = [
        '#title' => $this->t($user_name),
        '#type' => 'link',
        '#url' => Url::fromRoute('entity.user.edit_form', ['user' =>
$current_user_id]),
    ];
    $link_logout = [
        '#title' => $this->t('Выйти'),
        '#type' => 'link',
        '#url' => Url::fromRoute('user.logout'),
    ];
}
else if ($user_initialy != null | $user_surname != null)
{
    $link_user = [
        '#title' => $this->t($user_surname . ' ' . $user_initialy),
        '#type' => 'link',
        '#url' => Url::fromRoute('entity.user.edit_form', ['user' =>
$current_user_id]),
    ];
    $link_logout = [
        '#title' => $this->t('Выйти'),
        '#type' => 'link',
        '#url' => Url::fromRoute('user.logout'),
    ];
}

$build['#theme'] = 'cabinet_theme_hook';

$build['#link_user'] = $link_user;
$build['#link_logout'] = $link_logout;

return $build;

}
}

```

Приложение A.8 — Файл
«DocumentRouteSubscriberBlock.php»

```

<?php

namespace Drupal\document_module\Routing;

```

```

use Drupal\Core\Routing\RouteSubscriberBase;
use Symfony\Component\Routing\RouteCollection;

/**
 * Listens to the dynamic route events.
 */
class DocumentRouteSubscriber extends RouteSubscriberBase {

    /**
     * {@inheritdoc}
     */
    protected function alterRoutes(RouteCollection $collection) {

        /** Переопределяем заголовки некоторых форм, меняя _title параметр
        маршрута */

        if ($route = $collection->get('entity.document_entity.collection')) {
            $route->setDefault('_title', 'Список всех документов');
        }

        if ($route = $collection->get('entity.document_entity_type.collection')) {
            $route->setDefault('_title', 'Типы документов');
        }

        if ($route = $collection->get('entity.document_entity.add_page')) {
            $route->setDefault('_title', 'Добавить новый:');
            $route->setDefault('_title_callback', NULL);
        }

        if ($route = $collection->get('entity.document_entity.edit_form')) {
            $route->setDefault('_title_callback',
'Drupal\document_module\Controller\DocumentOwnEntityController::editTitle');
        }

        if ($route = $collection->get('entity.document_entity.add_form')) {
            $route->setDefault('_title_callback',
'Drupal\document_module\Controller\DocumentOwnEntityController::addBundleTitle');
        }

        if ($route = $collection->get('entity.taxonomy_vocabulary.collection')) {
            $route->setDefault('_title', 'Словари');
        }
    }
}

```

Приложение
A.12
—
Файл

«DocumentEntityAccessControlHandler.php

```

<?php

namespace Drupal\document_module;

```

```

use Drupal\Core\Entity\EntityAccessControlHandler;
use Drupal\Core\Entity\EntityInterface;
use Drupal\Core\Session\AccountInterface;
use Drupal\Core\Access\AccessResult;

/**
 * Access controller for the Document entity entity.
 *
 * @see \Drupal\document_module\Entity\DocumentEntity.
 */
class DocumentEntityAccessControlHandler extends EntityAccessControlHandler {

    protected function checkAccess(EntityInterface $entity, $operation,
    AccountInterface $account) {

        $entity_type_id = $entity->getEntityTypeId();
        $bundle = $entity->bundle();
        $is_owner = $entity->getOwnerId() === $account->id();

        switch ($operation) {
            case 'view':
                if ($is_owner) {
                    return AccessResult::allowedIfHasPermission($account,
"view own $entity_type_id $bundle");
                }
                return AccessResult::allowedIfHasPermission($account, "view
any $entity_type_id $bundle");

            case 'update':
                if ($is_owner) {
                    return AccessResult::allowedIfHasPermission($account,
"edit own $entity_type_id $bundle");
                }
                return AccessResult::allowedIfHasPermission($account, "edit
any $entity_type_id $bundle");

            case 'delete':
                if ($is_owner) {
                    return AccessResult::allowedIfHasPermission($account,
"delete own $entity_type_id $bundle");
                }
                return AccessResult::allowedIfHasPermission($account, "delete
any $entity_type_id $bundle");
        }

        // Unknown operation, no opinion.
        return AccessResult::neutral();
    }

    /**
     * {@inheritdoc}
     */
    protected function checkCreateAccess(AccountInterface $account, array
    $context, $entity_bundle = NULL) {
        return AccessResult::allowedIfHasPermission($account, "create
{$context['entity_type_id']} $entity_bundle");
    }

}

```

«DocumentEntityPermissionGenerator.php

```
<?php

namespace Drupal\document_module;

use Drupal\document_module\Entity\DocumentEntityType;
use Drupal\Core\StringTranslation\StringTranslationTrait;
use Drupal\document_module\Entity\DocumentEntity;

class DocumentModulePermissionsGenerator {

    use StringTranslationTrait;

    public function documentTypePermissions() {
        $perms = [];
        foreach (DocumentEntityType::loadMultiple() as $entity_type) {
            $perms += $this->buildPermissions($entity_type);
        }

        /**
         * Этот permission будет отвечать за показ списка документов
         (entity.document_entity.collection)
         */

        $perms["show documents list"] = [
            'title' => $this->t('Показывать список документов'),
        ];

        /**
         * Этот permission будет отвечать за показ списка типов документов
         (entity.document_entity_type.collection)
         */

        $perms["show document types list"] = [
            'title' => $this->t('Показывать список типов документов'),
        ];

        return $perms;
    }

    /**
     * Create the permissions desired for an individual entity type.
     *
     * @param DocumentEntityType $entity_type
     *
     * @return array
     */
    protected function buildPermissions(DocumentEntityType $entity_type)
    {
        $type_id = $entity_type->id();
        $bundle_of = $entity_type->getEntityType()->getBundleOf();
        $type_params = [
            '%type_label' => $entity_type->label(),
            '%bundle_of' => $bundle_of,
            '%title_string' => '',
        ];
    }
}
```

```

switch ($type_id)
{
    case 'incoming':
        $type_params['%title_string'] = 'входящего документа';
        break;

    case 'outgoing':
        $type_params['%title_string'] = 'исходящего документа';
        break;

    case 'other':
        $type_params['%title_string'] = 'другого документа';
        break;

    case 'archive':
        $type_params['%title_string'] = 'архивного документа';
        break;
}

return [
    "create $bundle_of $type_id" => [
        'title' => $this->t('Создание нового документа категории
"%type_label"', $type_params),
    ],
    "view any $bundle_of $type_id" => [
        'title' => $this->t('Просмотр любого документа категории
"%type_label"', $type_params),
    ],
    "view own $bundle_of $type_id" => [
        'title' => $this->t('Просмотр своих документов категории
"%type_label"', $type_params),
    ],
    "edit any $bundle_of $type_id" => [
        'title' => $this->t('Редактирование любых документов
категории "%type_label"', $type_params),
    ],
    "edit own $bundle_of $type_id" => [
        'title' => $this->t('Редактирование своих документов
категории "%type_label"', $type_params),
    ],
    "delete any $bundle_of $type_id" => [
        'title' => $this->t('Удаление любых документов категории
"%type_label"', $type_params),
    ],
    "delete own $bundle_of $type_id" => [
        'title' => $this->t('Удаление своих документов категории
"%type_label"', $type_params),
    ],
];
}
}

```

Приложение A.14 – Файл «DocumentEntityListBuilder.php»

```

<?php

namespace Drupal\document_module;

```

```

use Drupal\Core\Entity\EntityInterface;
use Drupal\Core\Entity\EntityListBuilder;
use Drupal\Core\Link;
use Drupal\document_module\Entity\DocumentEntityType;
use Drupal\document_module\Entity\DocumentEntity;

/**
 * Defines a class to build a listing of Document entity entities.
 *
 * @ingroup document_module
 */
class DocumentEntityListBuilder extends EntityListBuilder {

  /**
   * {@inheritdoc}
   */
  public function buildHeader() {
    $header['type'] = $this->t('Категория');
    $header['name'] = $this->t('Имя');

    return $header + parent::buildHeader();
  }

  /**
   * {@inheritdoc}
   */
  public function buildRow(EntityInterface $entity) {
    /* @var $entity \Drupal\document_module\Entity\DocumentEntity */

    $document_types = DocumentEntityType::loadMultiple();

    if (isset($document_types[$entity->bundle()])) {
      $row['type'] = $document_types[$entity->bundle()]->get('label');
    }

    $row['name'] = Link::createFromRoute(
      $entity->label(),
      'entity.document_entity.edit_form',
      ['document_entity' => $entity->id()]
    );
    return $row + parent::buildRow($entity);
  }
}

```

Приложение A.15 – Файл «block.block.administrator.yml»

```

uuid: 158237c4-9dcf-4e84-88a4-c272e0e84d54
langcode: en
status: true
dependencies:
  module:
    - document_module
    - user
  theme:
    - mytesttheme
id: administrator
theme: mytesttheme
region: sidebar_first
weight: -5

```

```

provider: null
plugin: admin_options_block
settings:
  id: admin_options_block
  label: Администратор
  provider: document_module
  label_display: visible
visibility:
  user_role:
    id: user_role
    roles:
      administrator: administrator
    negate: false
  context_mapping:
    user: '@user.current_user_context:current_user'

```

Приложение

A.16

–

Файл

«block.block.document_categories_block.yml

```

uuid: 5639410e-16ec-411a-85af-e18b59843fb7
langcode: en
status: true
dependencies:
  module:
    - document_module
    - user
  theme:
    - mytesttheme
id: document_categories_block
theme: mytesttheme
region: sidebar_first
weight: -4
provider: null
plugin: document_categories_block
settings:
  id: document_categories_block
  label: 'Категории документов'
  provider: document_module
  label_display: '0'
visibility:
  user_role:
    id: user_role
    roles:
      authenticated: authenticated
      administrator: administrator
    negate: false
  context_mapping:
    user: '@user.current_user_context:current_user'

```

Приложение A.17 – Файл «block.block.cabinet.yml

```

uuid: 9410ce68-1645-4d53-bd29-2b44cb7f0cbf
langcode: en
status: true
dependencies:

```

```

module:
  - document_module
theme:
  - mytesttheme
id: lichnyykabinet
theme: mytesttheme
region: secondary_menu
weight: 0
provider: null
plugin: cabinet_block
settings:
  id: cabinet_block
  label: 'Личный кабинет'
  provider: document_module
  label_display: '0'
visibility: { }

```

Приложение A.18 – Файл «document_module.module»

```

<?php

/**
 * @file
 * Contains document_module.module.
 */

use Drupal\Core\Routing\RouteMatchInterface;
use Drupal\Core\Link;
use Drupal\Core\Url;
use Drupal\document_module\Entity\DocumentEntityType;
use Drupal\document_module\Entity\DocumentEntity;

/**
 * Implements hook_help().
 */
function document_module_help($route_name, RouteMatchInterface $route_match)
{
  switch ($route_name) {
    // Main module help for the document_module module.
    case 'help.page.document_module':
      $output = '';
      $output .= '<h3>' . t('About') . '</h3>';
      $output .= '<p>' . t('Module for Document entity type') . '</p>';
      return $output;

    default:
  }
}

/**
 * Implements hook_theme().
 */
function document_module_theme() {
  $theme = [];
  $theme['document_entity'] = [
    'render element' => 'elements',
    'file' => 'document_entity.page.inc',
    'template' => 'document_entity',
  ];
}

```

```

$theme['document_entity_content_add_list'] = [
    'render element' => 'content',
    'variables' => ['content' => NULL],
    'file' => 'document_entity.page.inc',
];
$theme['to_main_page_button'] = [
    'variables' => [],
    'template' => 'to-main-page-button-tpl',
];
$theme['cabinet_theme_hook'] = array(
    'variables' => array(
        'link_user' => NULL,
        'link_logout' => NULL),
    'template' => 'cabinet',
);

$theme['events_get_card_proplist'] = array(
    'variables' => array(
        /**'name' => NULL,*/
        'created' => NULL,
        'changed' => NULL,
        'type' => NULL,
        'file_links' => array('value' => NULL),
        'doccardfields' => array('value' => NULL),
    ),
    'template' => 'doccard-proplist-tpl',
);
return $theme;
}

/**
 * Implements hook_theme_suggestions_HOOK().
 */
function document_module_theme_suggestions_document_entity(array $variables)
{
    $suggestions = [];
    $entity = $variables['elements']['#document_entity'];
    $sanitized_view_mode = strstr($variables['elements']['#view_mode'], '.',
    '_');

    $suggestions[] = 'document_entity__' . $sanitized_view_mode;
    $suggestions[] = 'document_entity__' . $entity->bundle();
    $suggestions[] = 'document_entity__' . $entity->bundle() . '___' .
    $sanitized_view_mode;
    $suggestions[] = 'document_entity__' . $entity->id();
    $suggestions[] = 'document_entity__' . $entity->id() . '___' .
    $sanitized_view_mode;
    return $suggestions;
}

function template_preprocess_to_main_page_button(&$variables)
{
    $variables['link'] = Link::fromTextAndUrl('На главную', new
    Url('document_module.main'));
}

```

ПРИЛОЖЕНИЕ Б: ПРЕЗЕНТАЦИЯ

Система приема и передачи документов на основе технологии CMS Drupal

Разработчик: Семахин Н.Р.

группа 15ВВ1.

Руководитель: Дорошенко И.Н.

Работа современного учреждения, предприятия, организации малого или среднего бизнеса базируется на использовании специальных информационных систем, направленных на повышение эффективности работы персонала с документацией и различного вида отчетностью.

Такие системы, а также системы со схожим функционалом, называют системами электронного документооборота (СЭД).

Главное их назначение - это хранение электронных документов, а также работа с ними (редактирование, просмотр, поиск по содержанию и атрибутам и т.д.). Системы электронного документооборота сегодня работают с корреспонденцией, распорядительными и внутренними документами, договорами и проектной документацией. При помощи СЭД решается множество прикладных задач: управление проектами, интеграция с бухгалтерией, работа с клиентами, автоматизация формирования документов.

Таким образом, все внутренние процессы, связанные с документооборотом, управляются посредством СЭД.

Цель и задачи работы

Цель работы: разработать сайт системы электронного хранилища документов организации на базе системы управления содержимым Drupal.

Достижение поставленной цели требует решения следующих задач:

- ☐ формирование требований к проектируемой системе и определение её функциональных возможностей.
- ☐ анализ возможности использования системы управления содержимым Drupal в качестве средства для построения информационных систем и сайтов различной сложности.
- ☐ определение основных принципов работы с Drupal и её встроенными средствами по управлению сайтом.
- ☐ овладение навыками по созданию программных модулей для системы Drupal.
- ☐ разработка необходимых программных модулей, расширяющих базовый функционал системы Drupal до необходимого уровня.

3

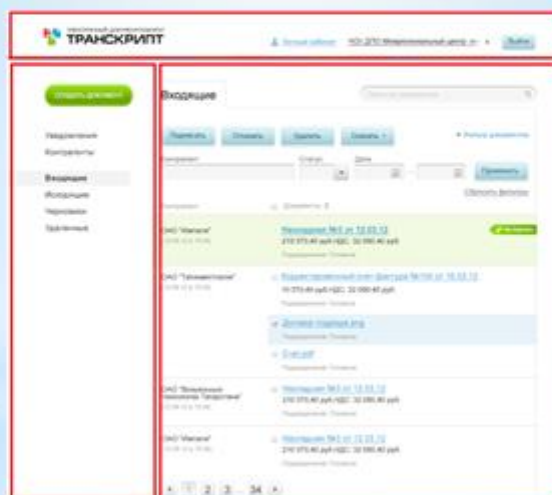
Требования к проектируемой системе

Разрабатываемая система должна иметь удобный пользовательский интерфейс и реализовывать следующие функции, характерные для любой подобной системы:

- * вход на сайт зарегистрированных пользователей
- * настройка и добавление новых документов
- * показ карточки документа
- * разграничение доступа к каждому типу документов
- * возможность создания/изменения/редактирования документов
- * централизованное хранение документов
- * система уведомлений
- * система удобного поиска по критериям
- * ролевая система пользователей (администратор, пользователь и т.д.).

4

Пример интерфейса системы



5

CMS Drupal

В поиске подходящей CMS для разработки сайта выбор остановился на технологии CMS Drupal.

Drupal — система управления содержимым (CMS), используемая также как каркас для веб-приложений (CMF), написанная на языке PHP и использующая в качестве хранилища данных реляционную базу данных.

Модульный принцип построения Drupal подразумевает возможность включения/выключения модулей, отвечающих за отдельные возможности сайта.

6

Требования к сайту

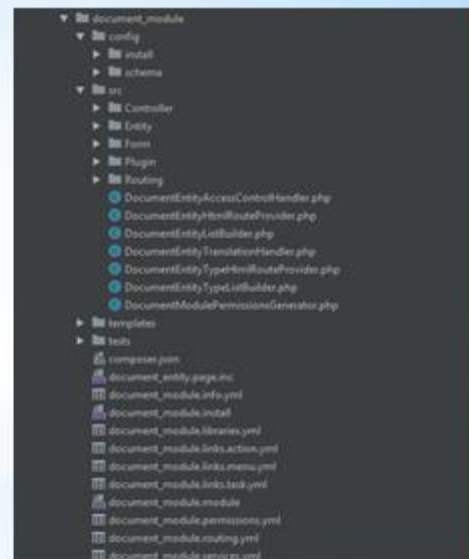
Перечень того, что необходимо сделать для сайта, чтобы он стал похож на информационную систему хранения и управления документами:

- * разработать новый тип контента «Документы» со множеством полей;
- * определить все необходимые формы отображения для документов и блоков меню;
- * создать свою тему, исключая все лишнее;
- * определить все необходимые права, разграничивающие доступ к документам и возможностям по их управлению;

7

Разработка модуля

- * Решать эти задачи будет специальный модуль. Этот модуль будет называться «Модуль документов».



8

Сущность для хранения документов

- * В Drupal для хранения информации о контенте используется концепция сущностей (с англ. Entities).
- * Сущности в Drupal представляют собой объекты, используемые для постоянного хранения информации. Эта информация хранится в базе данных.
- * Поля в Drupal определяют, какие данные будет хранить сущность.
- * При этом к одному типу сущности можно прикреплять разные наборы полей, создавая привязки (bundles).

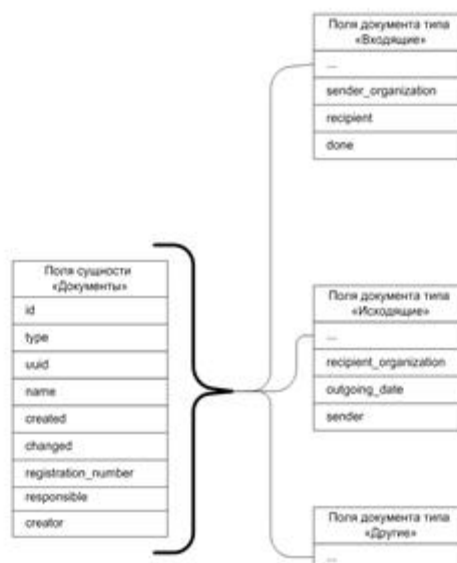
9

Структура документа

- * Документ будет иметь следующие поля: идентификатор (id), тип (type), uuid, имя (name), дата создания (created), дата редактирования (changed), регистрационный номер (registration_number), ответственное лицо (responsible), кем создан (creator), файл (file). Поля ответственное лицо (responsible) и кем создан (creator) будут хранить идентификатор пользователя из таблицы пользователей. Значения полей идентификатор и uuid будут автоматически заполняться при создании нового объекта сущности.
- * Документы будут различаться по типам на входящие, исходящие, архивные и другие.
- * У каждого типа будет свой набор дополнительных полей.

10

document_entity	
 id	integer
type	varchar(255)
uuid	varchar(255)
name	string
created	timestamp
changed	timestamp
registration_number	string
responsible	integer
creator	integer

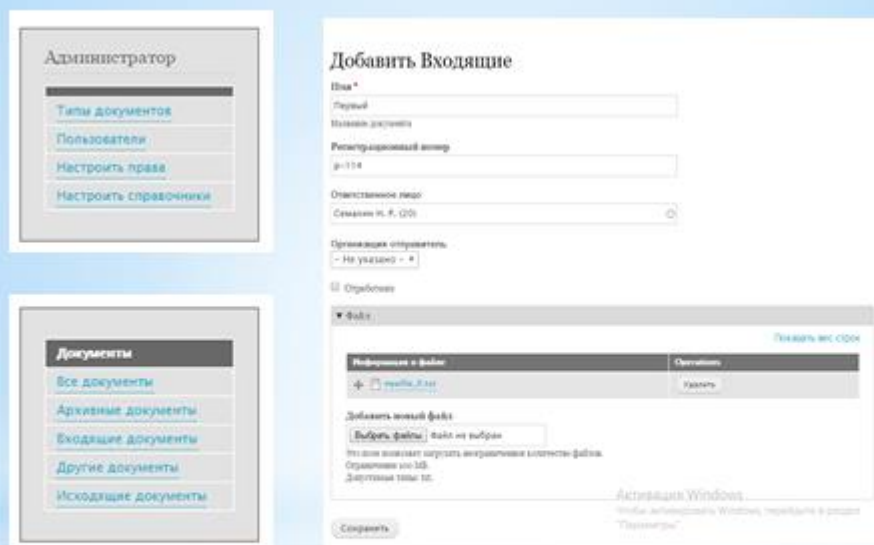


11

Формы и блоки сайта

- * Минимальный набор форм, необходимых для функционирования сайта будет содержать: форму добавления документа, форму удаления документа, форму редактирования документа, форму представления документа в виде карточки, табличный список документов.
- * В области левой боковой панели будут расположены блоки ссылок. Блок «Администратор» будет доступен только администратору сайта. Блок «Документы» будет доступен всем зарегистрированным пользователям сайта.

12



13

Разграничение доступа

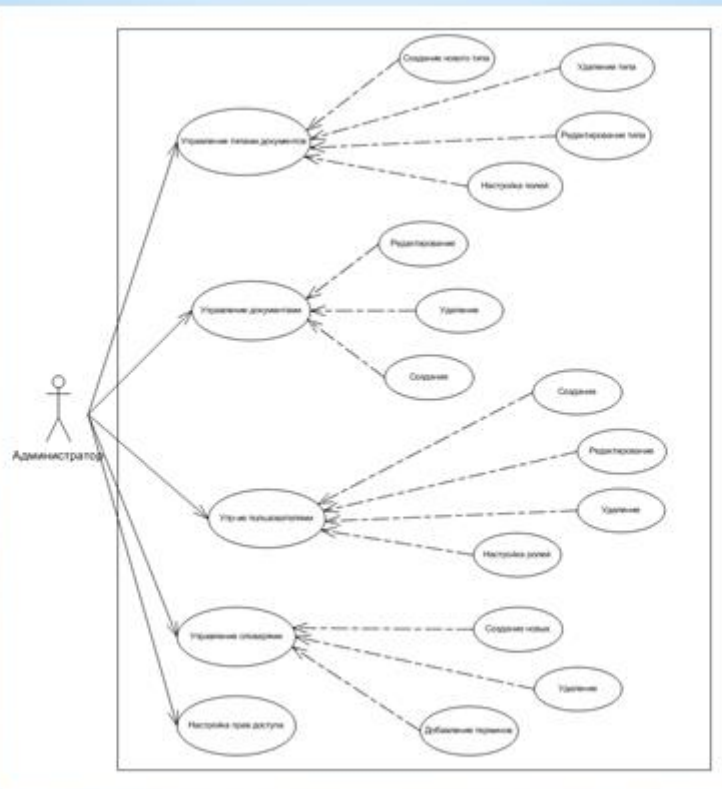
- * Система доступа к контенту в Drupal основана на концепции ролей и прав. Пользователю в Drupal присваиваются роли, а каждая роль, в свою очередь, содержит определенное количество прав на какие-либо действия.
- * В Drupal существует всего четыре операции над сущностями: просмотр (view), удаление (delete), изменение (update), создание (create).
- * Перечень возможностей любого зарегистрированного пользователя ограничен и определяется конфигурацией, которую для него определил администратор.

Настроить права доступа

[Hide description](#)

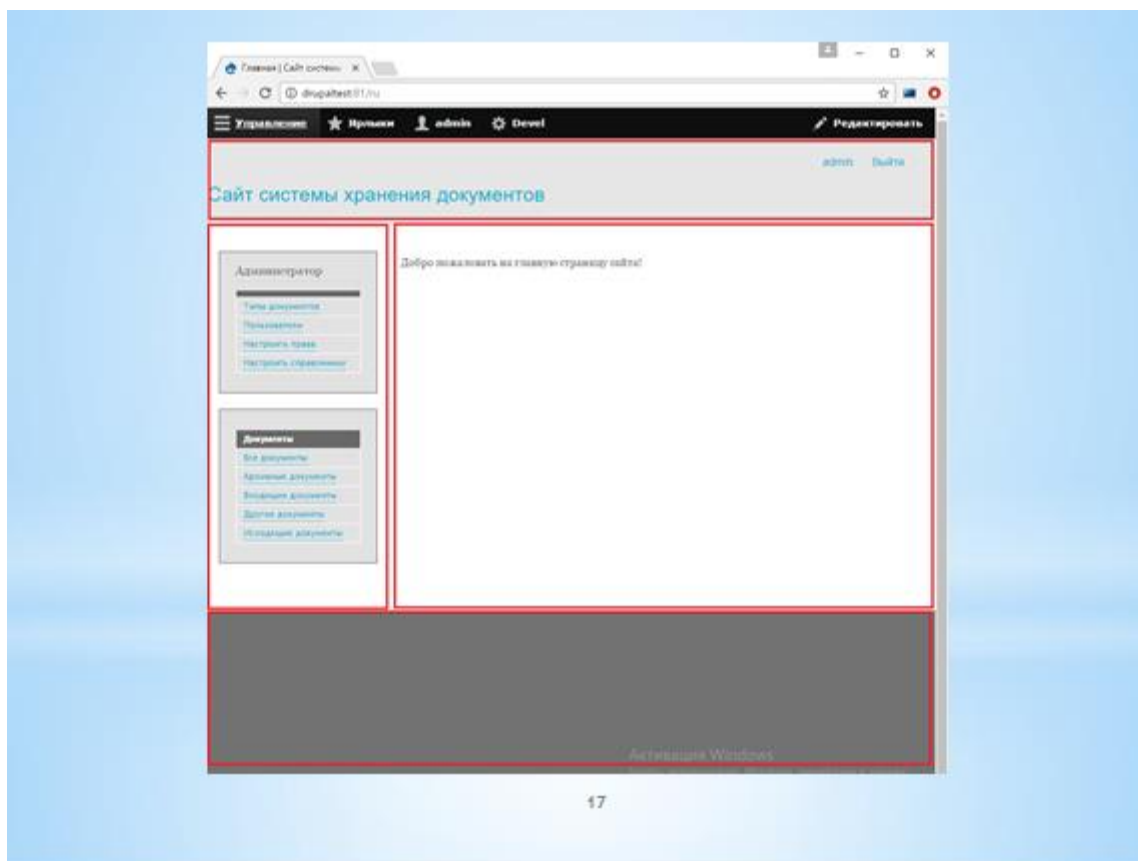
Permissions	Anonymous user	Authenticated user	Administrator
Показывать список документов	☐	☐	☒
Показывать список типов документов	☐	☐	☒
Просмотр любого документа категории "Архивные"	☐	☐	☒
Просмотр любого документа категории "Входящие"	☐	☒	☒
Просмотр любого документа категории "Другие"	☐	☐	☒
Просмотр любого документа категории "Исходящие"	☐	☐	☒
Просмотр своих документов категории "Архивные"	☐	☐	☒

14

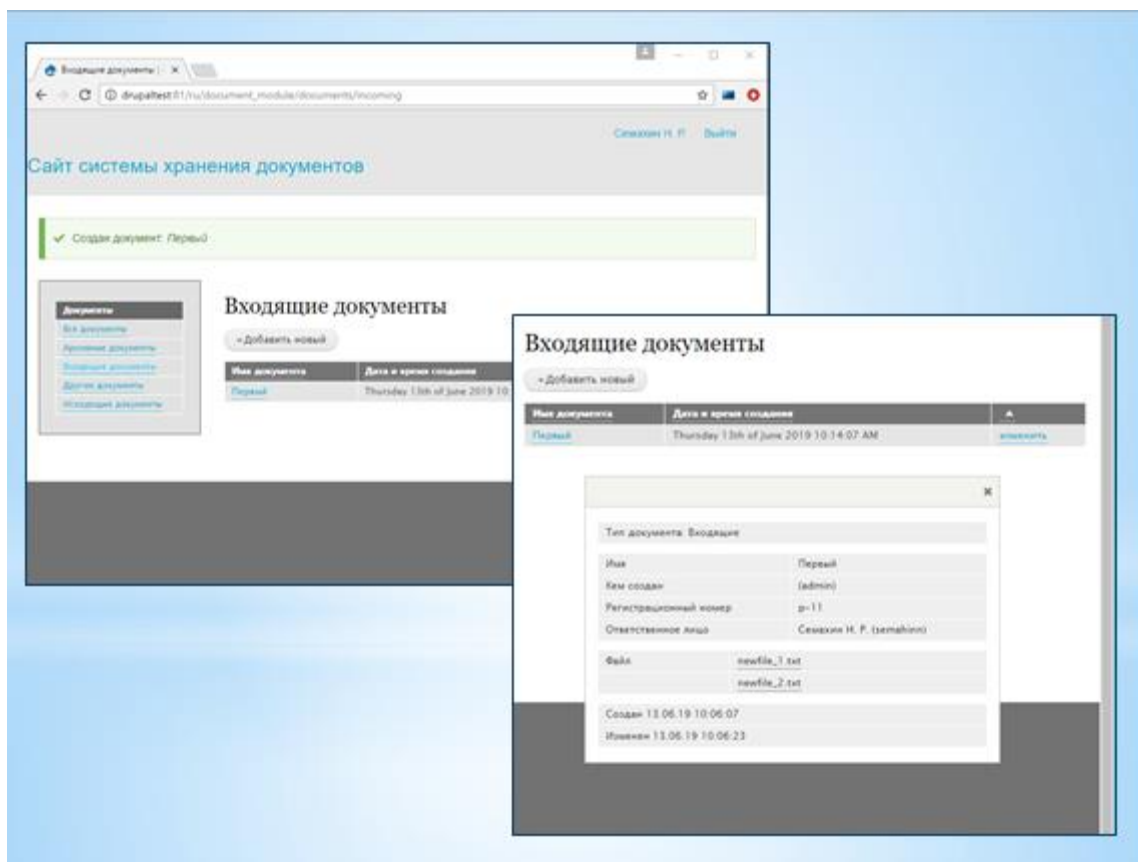


Работа с сайтом

- * Страница поделена на четыре области: верхняя горизонтальная область (шапка сайта), область левой боковой панели, область контента и область подвала сайта.
- * Левая боковая панель содержит в себе два блока: блок возможностей администратора и блок быстрого доступа к документам.
- * На сайт можно зайти любым зарегистрированным пользователем. В зависимости от его прав содержимое левой боковой панели может меняться.



17



Заключение

В данной работе был реализован сайт системы электронного хранения и управления документами.

Поставленная цель была выполнена. Все задачи были решены в полном объеме.

В данной работе:

- * рассмотрены примеры существующих информационных систем электронного хранения документации и систем электронного документооборота, определены их характерные черты и область применения;
- * произведен анализ требований к разрабатываемому сайту системы, определены особенности его внешнего вида и функциональность;
- * рассмотрены существующие технологии и программные средства, используемые при разработке сайта, проанализированы возможности этих средств;
- * дано описание системы управления контентом Drupal и её возможностей по управлению сайтом;
- * приведено описание хода разработки визуальной и программной составляющих сайта;
- * описана работа с сайтом, основные блоки сайта и навигация по ним.

19

Получен рабочий вариант сайта информационной системы, который в дальнейшем можно развивать. Расширение функциональных возможностей сайта системы электронного хранения и управления документами возможно в следующих направлениях:

- * внедрение средств для онлайн-просмотра документов;
- * добавление программного модуля для делопроизводства, который будет реализовывать движение документа внутри организации;
- * добавление возможности извлечения документа пользователем для монопольного использования;

20

Спасибо за внимание

