

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ	5
1. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ.....	8
1.1 Понятие методических материалов и их роль в образовательном процессе.....	8
1.2 Требования к методическим материалам	9
1.3 Классификация методических материалов	11
1.4 Сравнительный анализ существующих аналогов	16
1.5 Моделирование бизнес-процессов.....	19
Выводы по первой части.....	26
2. МОДЕЛИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ.....	27
2.1 Составление инфологической модели информационной системы	27
2.2 Разработка даталогической модели информационной системы	32
2.3 Клиент–серверная архитектура системы.....	36
Выводы по второй части.....	40
3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ИС.....	41
3.1 Средства разработки информационной системы	41
3.2 Создание базы данных.....	42
3.3 Реализация бизнес-логики сущности “Пользователь”	46
3.4 Реализация бизнес-логики сущности “Материал”	52
3.5 Тестирование и демонстрация работы информационной системы.....	58
3.6 Руководство по работе с информационной системой.....	65
Выводы по третьей главе.....	67
ЗАКЛЮЧЕНИЕ.....	68
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ	69
ПРИЛОЖЕНИЕ	72

СПИСОК СОКРАЩЕНИЙ

ЭОР – электронный образовательный ресурс

ИОМ – интерактивный образовательный модуль

УМК – учебно–методический комплекс

ИС – информационная система

БД – база данных

ПК – первичный ключ

ПНФ – первая нормальная форма

ВНФ – вторая нормальная форма

ТНФ – третья нормальная форма

НФБК – третья частная нормальная форма Бойса–Кодда

ВВЕДЕНИЕ

В современном мире роль информационных технологий в различных сферах деятельности человека становится все более значимой. Не является исключением и работа учителя. В условиях постоянного обновления и совершенствования образовательного процесса, педагогу необходимо использовать различные информационные материалы для повышения своего профессионального уровня и качества процесса обучения.

Одним из наиболее эффективных способов получения информации являются онлайн–ресурсы, предоставляющие доступ к методическим материалам. Такие сайты позволяют учителю не только знакомиться с новыми методиками и подходами, но и обмениваться опытом с коллегами, а также использовать уже существующие наработки для улучшения своей работы.

Возможности онлайн–ресурсов методических материалов:

- доступность: онлайн–ресурсы доступны в любое время и в любом месте при наличии интернета, что позволяет обучающимся изучать материал в удобном для них темпе и в удобное время;
- гибкость: онлайн–ресурсы позволяют обучающимся самостоятельно выбрать интересующую их тему для дальнейшего изучения;
- персонализация: онлайн–ресурсы можно адаптировать под индивидуальные потребности обучающихся, например, выбрать уровень сложности материала или формат представления информации;
- сокращение затрат времени на поиск информации: онлайн–ресурсы вмещают в себя великое множество материалов по различным темам и предоставляют удобную поисковую систему для сортировки материалов по различным критериям.

Темой выпускной квалификационной работы является разработка информационной системы «Методическая копилка учителя–предметника».

Актуальность данной выпускной квалификационной работы заключается в том, что разработка сайта–сборника методических материалов является востребованным направлением в автоматизации современной системы образования. Такой сайт может стать незаменимым помощником для учителей, студентов и всех тех, кто интересуется обучением. Он позволит сконцентрировать в одном месте различные методические материалы, делая их доступными для всех пользователей Интернета, а также обеспечить систематизацию и удобство поиска информации.

Целью данной выпускной квалификационной работы является моделирование и программная реализация веб–приложения сборника методических материалов для учителей–предметников.

Для достижения указанной цели необходимо решить следующие задачи:

1. Провести анализ заданной предметной области, изучить категории методических материалов, их особенности и существующие нормативные требования к их учету и наполнению.
2. Проанализировать существующие сайты–сборники методических материалов и определить основные функции и требования пользователей к таким сайтам.
3. Выполнить моделирование информационной системы в соответствии с базовыми требованиями реализации веб–приложений и полученной информацией из профильных нормативных документов, ГОСТов.
4. Разработать базу данных информационной системы. База данных должна соответствовать третьей нормальной форме, эффективно обрабатывать данные и предоставлять доступ к ним.
5. Выполнить программную разработку информационной системы. Провести тестирование системы по выделенным функциям в соответствии с ролями пользователей системы.

Выпускная квалификационная работа состоит из трех частей, списка используемых источников и приложения.

В первой части работы выполнен анализ предметной области. Здесь определены ключевые объекты будущей информационной системы и выполнен анализ существующих решений автоматизации заданной предметной области. Далее на основе полученных результатов проведено моделирование бизнес-процессов и сделан вывод о проделанной работе.

Во второй части осуществлена разработка инфологической и даталогической моделей информационной системы, удовлетворяющих критериям соответствия третьей нормальной форме Бойса–Кодда.

В третьей части описаны этапы программной реализации информационной системы, включающие в себя выбор средств разработки, создание базы данных, реализацию бизнес-логики и интерфейса информационной системы. Также выполнено тестирование функционала в соответствии с техническим заданием и описаны ограничения функциональности в зависимости от роли пользователя.

В заключении приведены выводы о достижении поставленной цели и решении задач данной выпускной квалификационной работы.

В части «ПРИЛОЖЕНИЕ» приведено техническое задание, на основе которого осуществлялась разработка информационной системы.

1. ИССЛЕДОВАНИЕ ПРЕДМЕТНОЙ ОБЛАСТИ

Исследование предметной области предполагает определение границ этой области с целью последующего моделирования структур, процессов и отношений, характерных для данной исследуемой области. Под моделированием предметной области понимают процесс создания абстрактного представления объекта реального мира. Моделирование предметной области состоит из нескольких этапов:

1. Определение основных элементов системы, их свойств и отношений между собой
2. Оценка отношений элементов системы для реализации возможных действий внутри информационной системы

Целью моделирования предметной области является формирование структурированного представления о предметной области, для последующего использования при разработке информационной системы, удовлетворяющей ряду требований и предоставляющей определенные возможности для решения задач.

1.1 Понятие методических материалов и их роль в образовательном процессе

Методические материалы – это совокупность средств, методов и форм обучения, направленных на достижение образовательных и воспитательных целей. Они являются важным инструментом в работе педагога, позволяющим организовать образовательный процесс максимально эффективно.

В соответствии с Законом об образовании в Российской Федерации, методические материалы должны соответствовать требованиям федеральных государственных образовательных стандартов (ФГОС) и обеспечивать достижение обучающимися планируемых результатов освоения основной образовательной программы.

Роль методических материалов в образовательном процессе:

- обеспечение учебного процесса необходимыми материалами и средствами для организации занятий, контроля знаний, самостоятельной работы учащихся и т. д;

- повышение эффективности и качества обучения за счет использования современных методов, форм и технологий обучения;
- формирование и развитие компетенций учащихся на основе применения разнообразных методических средств и приемов;
- создание условий для индивидуализации и дифференциации обучения, позволяющих учесть особенности каждого учащегося и обеспечить ему оптимальные условия для развития;
- помощь педагогу в подготовке и проведении занятий, выборе методов и средств обучения, анализе результатов своей работы. Таким образом, методические материалы играют важную роль в современном образовательном процессе, поскольку они способствуют повышению качества и эффективности обучения, развитию компетенций учащихся и совершенствованию профессиональной деятельности педагогов.

1.2 Требования к методическим материалам

С точки зрения существующих стандартов, методические материалы должны удовлетворять следующему ряду требований:

- доступность и наглядность: методические материалы должны быть доступны и понятны всем участникам образовательного процесса [1];
- актуальность: содержание методических материалов должно соответствовать современным требованиям и стандартам [2];
- эффективность: методические материалы должны обеспечивать достижение поставленных целей обучения и быть эффективными в использовании [3];
- технологичность: методические материалы должны предусматривать использование современных образовательных технологий и методик [4];

- соответствие образовательным стандартам: методические материалы должны соответствовать федеральным государственным образовательным стандартам и другим нормативным документам [5];
- безопасность: методические материалы не должны содержать информации, которая может нанести вред здоровью или вызвать негативные эмоции у участников образовательного процесса [6];
- контроль качества: методические материалы проходят проверку на соответствие требованиям образовательных стандартов, а также оценку их эффективности [7].

«...в соответствии с ГОСТ Р 52653 – 2006 электронный образовательный ресурс (далее, ЭОР) – образовательный ресурс, представленный в электронно–цифровой форме и включающий в себя структуру, предметное содержание и метаданные о них. В общем случае ЭОР включает в себя образовательный контент, программные компоненты и метаданные» [8]. Образовательный контент – структурированное предметное содержимое, используемое в образовательном процессе, информационно значимое наполнение ЭОР. Программные компоненты обеспечивают предъявление элементов контента пользователю в определенных сочетаниях, а также обеспечивают интерактивный режим работы с контентом. Метаданные – структурированные данные, предназначенные для описания характеристик ЭОР, объекта данных или компонента образовательной технологической системы. ЭОР могут быть классифицированы: по цели создания – учебные, социокультурные и др.; по категории пользователей – учитель, ученик и др.; по форме организации учебного процесса – аудиторные занятия, самостоятельная образовательная деятельность; по специальным потребностям – без ограничений, с ограниченными возможностями здоровья; по природе основной информации – текстографические, элементарные аудиовизуальные, мультимедийные; по технологии распространения – локальные, сетевые, комбинированного распространения; по функции в учебном процессе – информационные, практические, контрольные и др. типы ЭОР; и т.д. ЭОР, обладающий развитой

интерактивностью и мультимедийностью, называют также интерактивным образовательным модулем (далее, ИОМ). Каждый ИОМ автономен: он может использоваться самостоятельно, независимо от других ИОМ. Вариативами называются ИОМ одного типа, имеющие значительные отличия контента и посвященные одному и тому же тематическому элементу. Необходимость создания вариативов определяется заданием на выполнение работ по государственному контракту. К числу основных показателей качества ИОМ относятся: содержательные характеристики – свойства, определяющие качество, достаточность и методическую проработанность представленного учебного материала; интерактивность – свойство, определяющее характер и степень взаимодействия пользователя с элементами ИОМ; мультимедийность – свойство, определяющее количество и качество форм представления информации, используемых в ИОМ; модифицируемость – свойство, определяющее возможность и сложность внесения изменений в содержание и программные решения ИОМ.

1.3 Классификация методических материалов

По своему назначению методические материалы условно можно разделить на 4 группы:

I. Группа – информирующие методические материалы.

- информация (собственно информация, информационный лист, информационный плакат, обзор, графики);
- информационный каталог;
- информационный справочник;
- информационно – методическая выставка.

II. Группа – описывающие методические материалы.

- методическое описание;
- комментарий;
- аннотация;
- реферат.

III. Группа – инструктирующие методические материалы.

- инструктивно – методическое письмо;
- методическая записка;
- памятка;
- инструкция;
- методические советы;
- методические рекомендации.

IV. Группа – прикладные методические материалы.

- сценарии;
- тематическая подборка материалов;
- картотека;
- дидактические пособия;
- задачник, сборник упражнений.

Информирующие методические материалы.

Главная задача информирующих материалов – донести информацию до адресата. Эти методические материалы не предназначены для описания техник или методов, анализа чьего-либо опыта или описания их действий. Как правило, в информационных материалах преобладает нумерационная информация, цифровые отчеты.

Описывающие методические материалы.

Их задача - подчеркнуть опыт и рассказать о деле. Такие материалы не всегда доступны самостоятельно, они обычно доступны как часть предложения или разработки.

Инструктирующие методические материалы.

Назначение этого вида материала – разъяснить цель и порядок действия, методику организации, проведения дела, акции, показать возможные приемы, формы.

Прикладные методические материалы.

Методические материалы, имеющие непосредственное практическое назначение, условно называются прикладными.

Отдельной группой являются электронные учебные материалы.

Самостоятельный электронный материал в определенной области знаний или в ряде областей – это электронный учебный материал. Он предназначен для обучения в классе, автоматического отслеживания информации и индивидуального обучения студентов. Эти материалы были отредактированы и опубликованы на интернет-платформе, на которой размещены цифровые образовательные ресурсы университета.

Выделяют следующие виды электронных учебных материалов:

Электронный учебник представляет собой самостоятельное учебное электронное издание, которое обеспечивает непрерывность и целостность учебного процесса, способствует информационному освоению студентом. В нем содержатся как теоретические, так и практические материалы, соответствующие учебной программе по определенной дисциплине, и используются современные мультимедийные технологии.

Сетевой курс представляет собой электронную образовательную платформу, которая охватывает все основные методы обучения по определенному предмету, учитывая различные технологии и включая контроль за самостоятельной работой студентов. Он предназначен для использования онлайн и хранения на сменных носителях данных. Это система обучающего материала по определенному предмету, разработанная для достижения образовательных и воспитательных целей, определенных учебной программой.

Электронный тестовый комплекс представляет собой набор тестов, организованных в понятной форме для определения степени подготовленности учащихся. Он предназначен для использования в Интернете для отслеживания успеваемости по определенным предметам, нескольким дисциплинам одновременно или в течение всего периода обучения. Электронный набор тестов может использоваться как преподавателями, так и

студентами. Учителя могут использовать его для оценки знаний учащихся, а учащиеся могут использовать его для самоконтроля и самостоятельной подготовки. Кроме того, набор тестов может помочь октябрьскому проведению различных исследований в области образования, поскольку он позволяет получить объективные данные об уровне знаний учащихся.

Электронные справочники – это инструменты, которые предоставляют студентам возможность получить необходимую справочную информацию в любое время. Он содержит информацию, которая многократно дополняет материал учебника. Обычно он представлен в виде списка терминов, слова на изучаемом языке или имени автора. Каждый элемент списка содержит гиперссылки на определение, перевод и грамматические особенности термина. Такая система нужна в педагогико-методическом комплексе, и электронный справочник может быть еще одним элементом комплекса или интегрирован в электронный учебник.

Компьютерные модели, тренажеры и конструкторы используются для закрепления знаний и приобретения практических навыков в условиях, имитирующих реальные ситуации. В отличие от других компонентов, компьютерные модели обычно не универсальны и предназначены для моделирования узкого круга явлений. На основе математических моделей, содержащих управляющие параметры, компьютерную модель можно использовать не только для иллюстрации явлений, которые трудно воспроизвести в области обучения, но и для определения влияния различных параметров на процессы и явления, изучаемые в интерактивном режиме. Это позволяет им использоваться вместо лабораторного оборудования, а также позволяет им разрабатывать методы управления моделируемыми процессами. Компьютерные тренажеры и конструкторы, в отличие от моделей, предназначены для развития практических навыков в определенной области. Его можно использовать в различных тренингах, таких как управление оборудованием в различных ситуациях, отработка навыков решения проблем и принятия решений. Например, симулятор для пилотов позволяет научиться

управлять самолетом, а симулятор хирургии помогает студентам-медикам подготовиться к реальной операции. Компьютерные технологии позволяют не только работать с готовыми объектными моделями, но и создавать их из отдельных элементов.

К тренажерам могут быть отнесены также и компьютерные задачки. Компьютерный задачник позволяет отработать приемы решения типовых задач, позволяющих наглядно связать теоретические знания с конкретными проблемами, на решение которых они могут быть направлены.

Электронный лабораторный комплекс позволяет имитировать реальные процессы или провести эксперимент, который невозможно выполнить в реальных условиях. Тренажер не только имитирует оборудование, но и представляет объекты исследования и экспериментальные условия. Такие тренажеры помогают подобрать оптимальные параметры для эксперимента, получить опыт и навыки на начальном этапе, упростить и ускорить работу с реальным оборудованием. В качестве тренажера может использоваться и компьютерная тестирующая система, которая обеспечивает, с одной стороны, возможность самоконтроля для обучаемого, а с другой – принимает на себя рутинную часть текущего или итогового контроля.

«Компьютерная тестирующая система может представлять собой как отдельную программу, не допускающую модификации, так и универсальную программную оболочку, наполнение которой возлагается на преподавателя. В последнем случае в нее включается система подготовки тестов, облегчающая процесс их создания и модификацию (в простейшем случае это может быть текстовый редактор). Эффективность использования тестирующей системы существенно выше, если она позволяет накапливать и анализировать результаты тестирования. Тестирующая система может быть встроена в оболочку электронного учебника, но может существовать и как самостоятельный элемент УМК. В этом случае тестирующие программы по различным дисциплинам целесообразно объединять в единой базе данных» [8].

1.4 Сравнительный анализ существующих аналогов

В ходе исследования предметной области необходимым является рассмотрение существующих аналогов. Такой анализ позволит выявить обязательные требования пользователей к разрабатываемой системе, сформировать понимание внутренних процессов, минимизировать риск появления возможных недоработок и увеличить общую функциональность, производительность и удобство работы итоговой системы.

Для проведения анализа следует использовать методологию SWOT. Ее суть заключается в исследовании четырёх ключевых параметров [9].

- Сильные стороны – преимущества и уникальные возможности, которые могут быть использованы для достижения успеха.
- Слабые стороны – недостатки, ограничения и проблемы, которые мешают ей достичь поставленных целей.
- Возможности – внешние факторы и условия, которые могут способствовать развитию и достижению целей, например, новые рынки, технологии или изменения в законодательстве.
- Угрозы – события или факторы, которые могут оказать негативное влияние, ограничить развитие, например, появление новых конкурентов, изменение предпочтений потребителей или экономические кризисы.

SWOT–анализ позволяет получить объективную оценку текущего состояния изучаемого объекта, выявить его сильные и слабые стороны, а также определить возможности и угрозы, которые могут возникнуть в будущем. Полученные выводы помогают принимать обоснованные стратегические решения, направленные на укрепление позиций на рынке, повышение конкурентоспособности и улучшение пользовательского опыта.

Достаточно популярным ресурсом является портал Методисты.ру. Он характеризуется простотой работы для пользователя. Проведем SWOT анализ портала Методисты.ру(<http://metodisty.ru/>)

Интерфейс портала Методисты.ру изображен на рисунке 1.

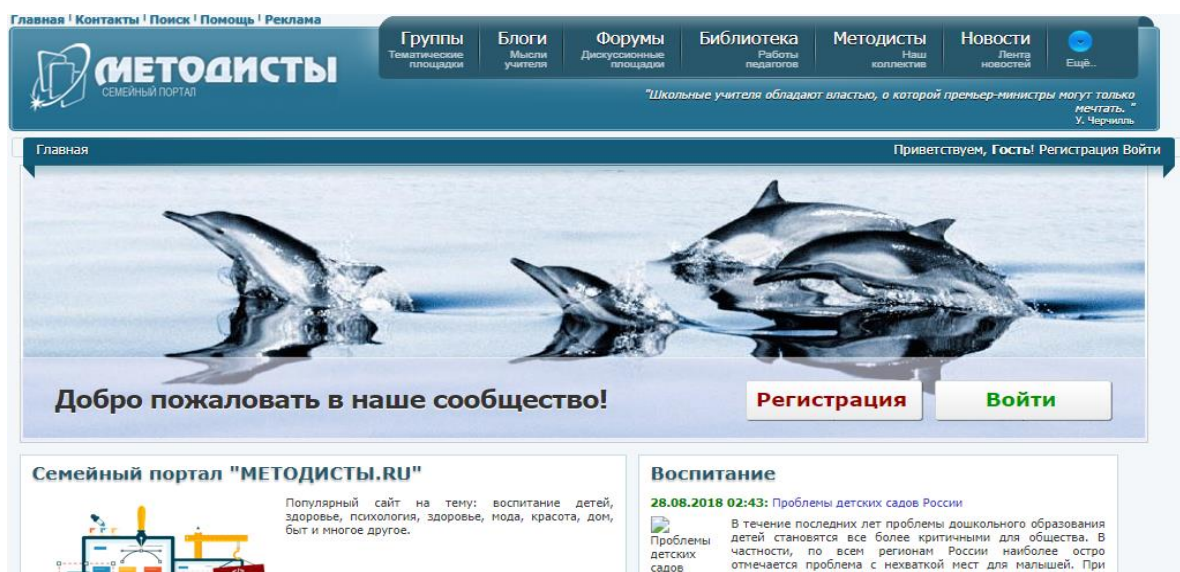


Рисунок 1 – Интерфейс портала Методисты.ру

Сильные стороны:

- большое количество существующих методических материалов;
- предлагаемые методические материалы полностью охватывают содержание школьной программы;
- наличие форума для общения педагогов.

Слабые стороны:

- устаревший дизайн;
- неудобство работы в связи с отсутствием адаптивной верстки для мобильных устройств;
- не все материалы высокого качества – некоторые материалы могут быть устаревшими или некачественными;
- отсутствие предпросмотра файлов перед скачиванием.

Возможности:

- растущий спрос на информатизацию образовательных материалов;
- поддержка образовательных ресурсов со стороны государства;
- разработка мобильного приложения или обновления верстки под мобильные устройства.

Угрозы:

- высокий уровень конкуренции со стороны других сайтов, предоставляющих аналогичные материалы и услуги;
- ddos и другие атаки, влияющие на стабильную работу сайта и безопасность пользовательских данных.

Еще одним популярным решением является сайт nsportal.

Он выделяется широким функционалом и большим количеством методических материалов. Проведем SWOT анализ ресурса nsportal(<https://nsportal.ru/>)

Интерфейс ресурса nsportal изображен на рисунке 2.

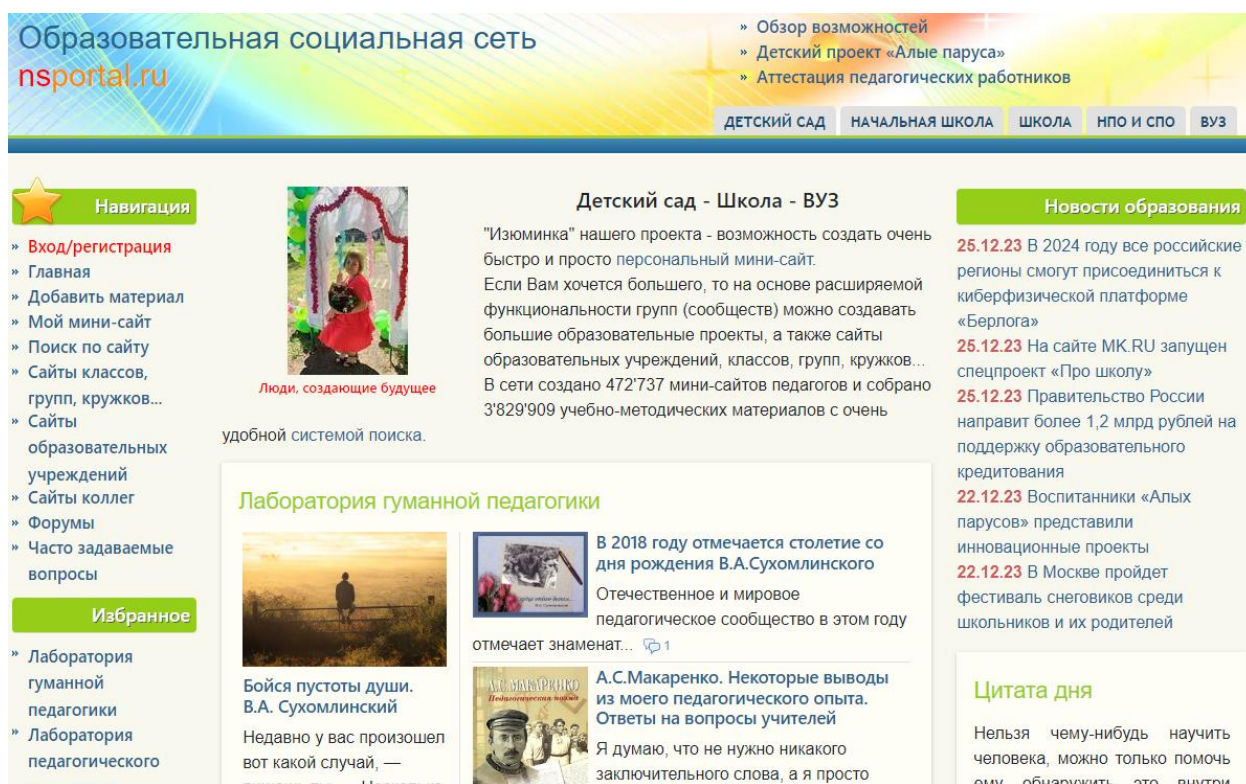


Рисунок 2 – Интерфейс ресурса nsportal

Сильные стороны:

- широкий выбор образовательных и воспитательных материалов, охватывающий все существующие уровни образования;
- встроенный конструктор мини-сайтов.

Слабые стороны:

- перегруженная поисковая система.

Возможности

- перспективная активно развивающаяся область информатизации.

Угрозы:

- высокий риск нарушения работы сайта из-за хакерских атак;
- высокий уровень конкуренции на рынке.

В результате анализа существующих аналогов были выявлены общие функциональные особенности сайтов методических материалов, среди которых авторизация/регистрация пользователя, публикация, изменение, удаление, предпросмотр и скачивание материалов. Также был выполнен анализ слабых сторон существующих решений, которые необходимо устранить, а именно:

- разрабатываемая система должна предоставлять возможность предпросмотра материала перед скачиванием;
- поисковая система должна предоставлять возможности быстрого поиска материалов.
- интерфейс информационной системы должен быть адаптивным.

1.5 Моделирование бизнес-процессов

На основе полученных в ходе SWOT анализа данных можно выделить следующие бизнес-процессы:

- регистрация нового пользователя;
- вход зарегистрированного пользователя в систему;
- добавление материалов на сайт;
- скачивание материалов с сайта;
- изменение добавленных материалов;
- удаление добавленных материалов.
- добавление материала в избранное

Для иллюстрирования бизнес-процессов следует использовать методологию IDEF0.

IDEF0 (Integration Definition for Function Modeling) – это методология функционального моделирования, являющаяся стандартом для моделирования, анализа и описания бизнес-процессов. Целью IDEF0 является создание функциональной модели, которая представляет собой схематическое изображение бизнес-процесса, где функции, из которых состоит процесс, представлены в виде блоков, а связи между ними – в виде стрелок [10]. Использование IDEF0 обусловлено тем, что в результате будет получена модель, которая позволит детально изучить процесс, выявить проблемы и определить направления для оптимизации и улучшения эффективности работы системы.

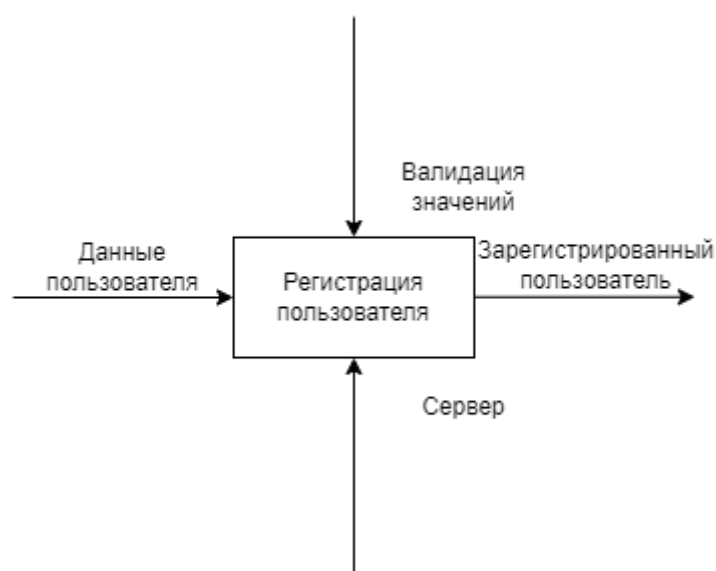


Рисунок 3 – Бизнес-процесс "Регистрация"

В процессе регистрации пользователю необходимо ввести данные, которые в дальнейшем будут использованы им для авторизации на сайте. В дальнейшем сервер после обработки запроса в случае корректного заполнения пользователем полей данных предоставляет ему доступ к сайту.

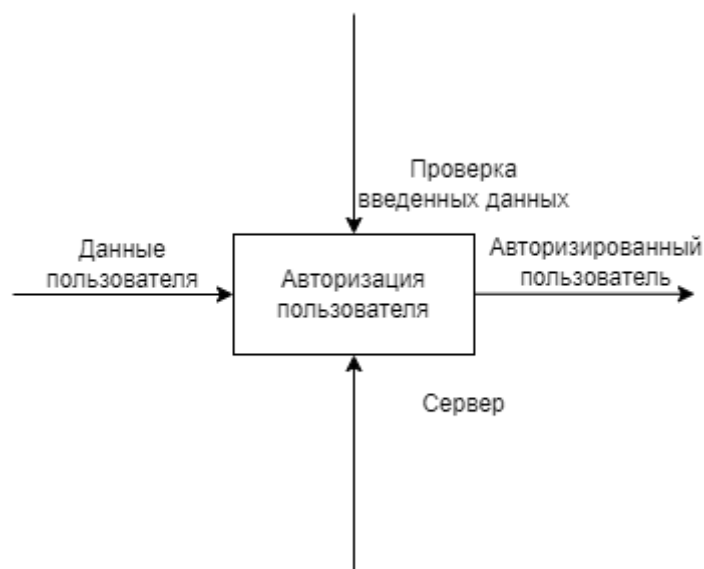


Рисунок 4 – Бизнес-процесс "Авторизация"

Для авторизации на сайте пользователю необходимо ввести данные своей учетной записи, после чего, в случае если введенные данные пройдут проверку на сервере, пользователь перейдет на главную страницу сайта.

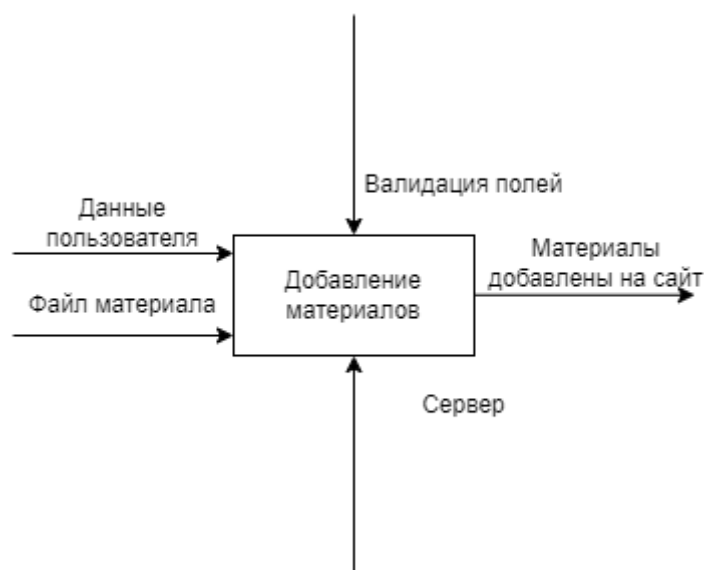


Рисунок 5 – Бизнес-процесс "Добавление материалов"

Чтобы добавить материалы на сайт, пользователь должен сначала авторизоваться на сайте. Для добавления материалов пользователю необходимо заполнить карточку материала согласно предложенной форме.



Рисунок 6 – Бизнес-процесс "Скачивание"

Для скачивания материалов сайта, пользователю необходимо перейти на страницу материала и посредством нажатия на кнопку, отправить запрос на сервер, после чего начнется скачивания материала.

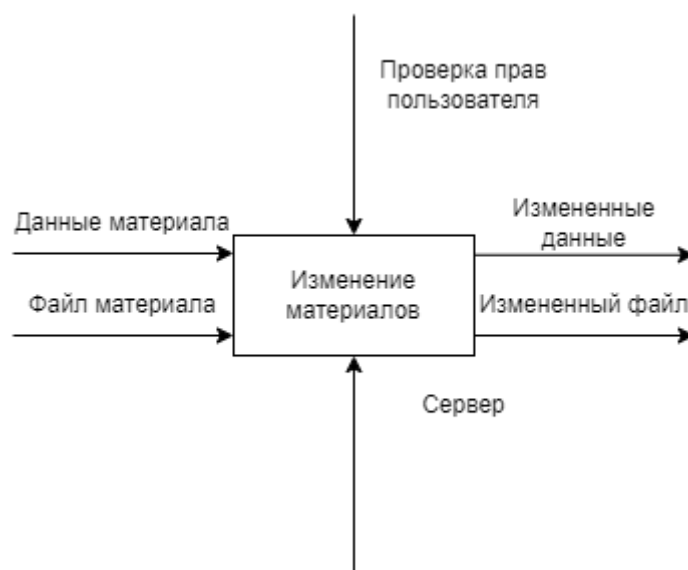


Рисунок 7 – Бизнес-процесс "Изменение материалов"

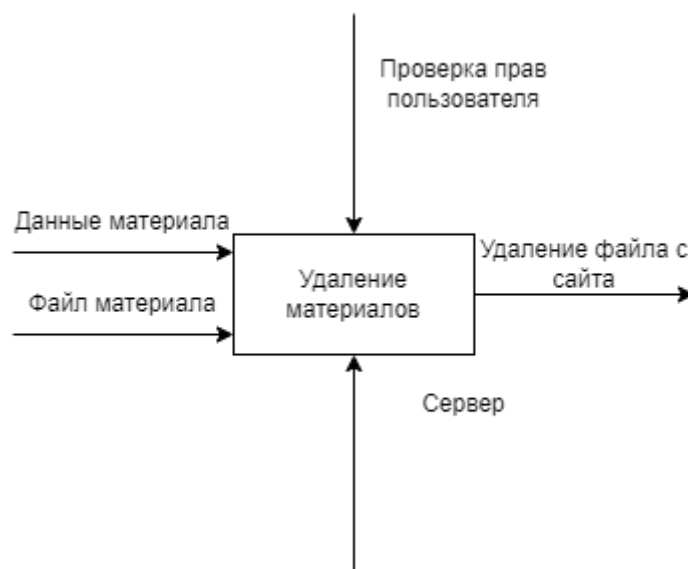


Рисунок 8 – Бизнес-процесс "Удаление"

Для редактирования/удаления материала с сайта, пользователь должен быть авторизован. В случае если материал был добавлен пользователем, ему станут доступны функции редактирования/удаления конкретного материала.

Для бизнес-процессов, требующих дополнительных требований и проверок пользователя следует построить укрупненные модели в нотации IDEF1, которая призвана более детально описать подпроцессы нижнего уровня.

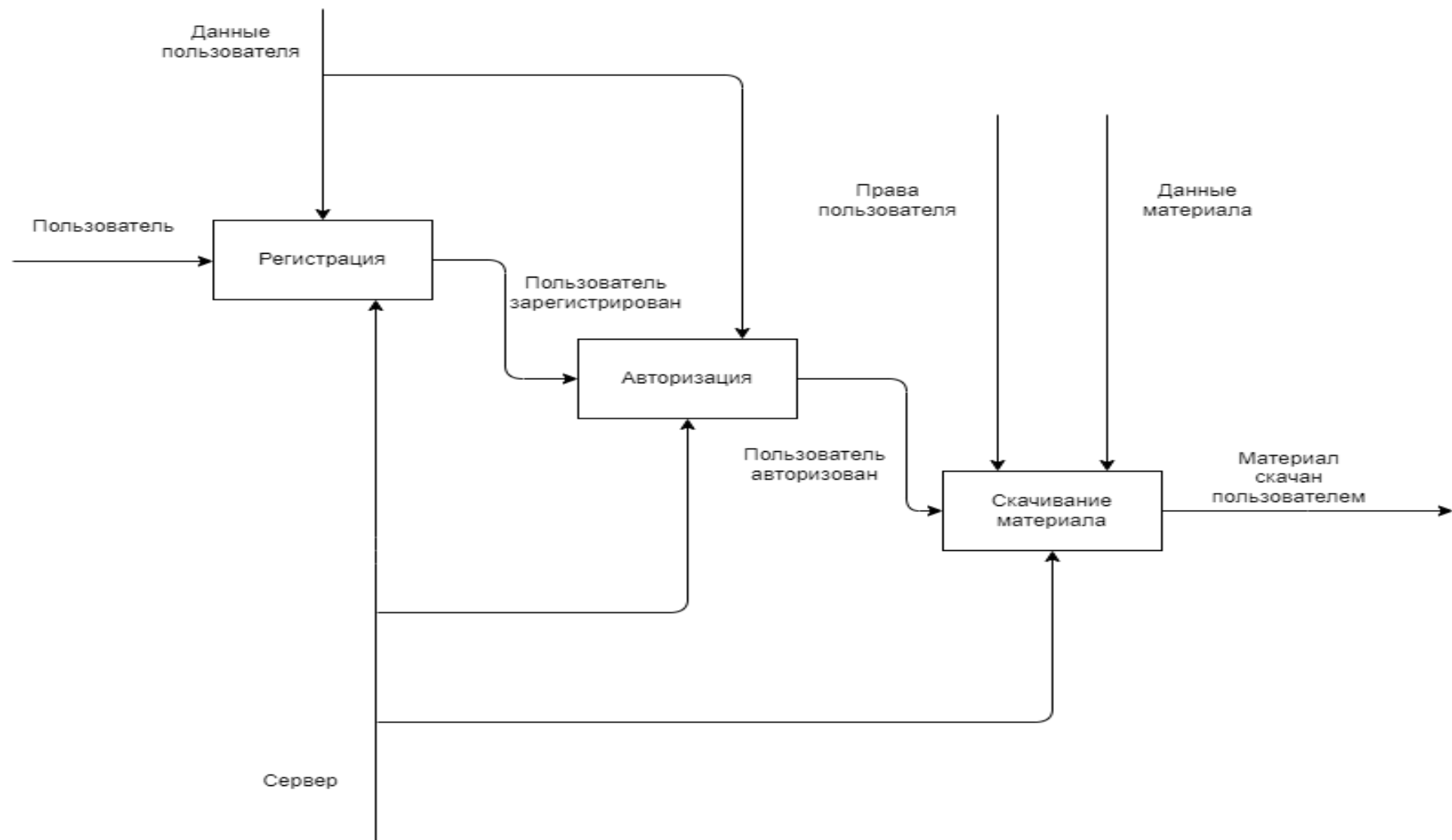


Рисунок 9 – IDEF1–модель процесса скачивания материала

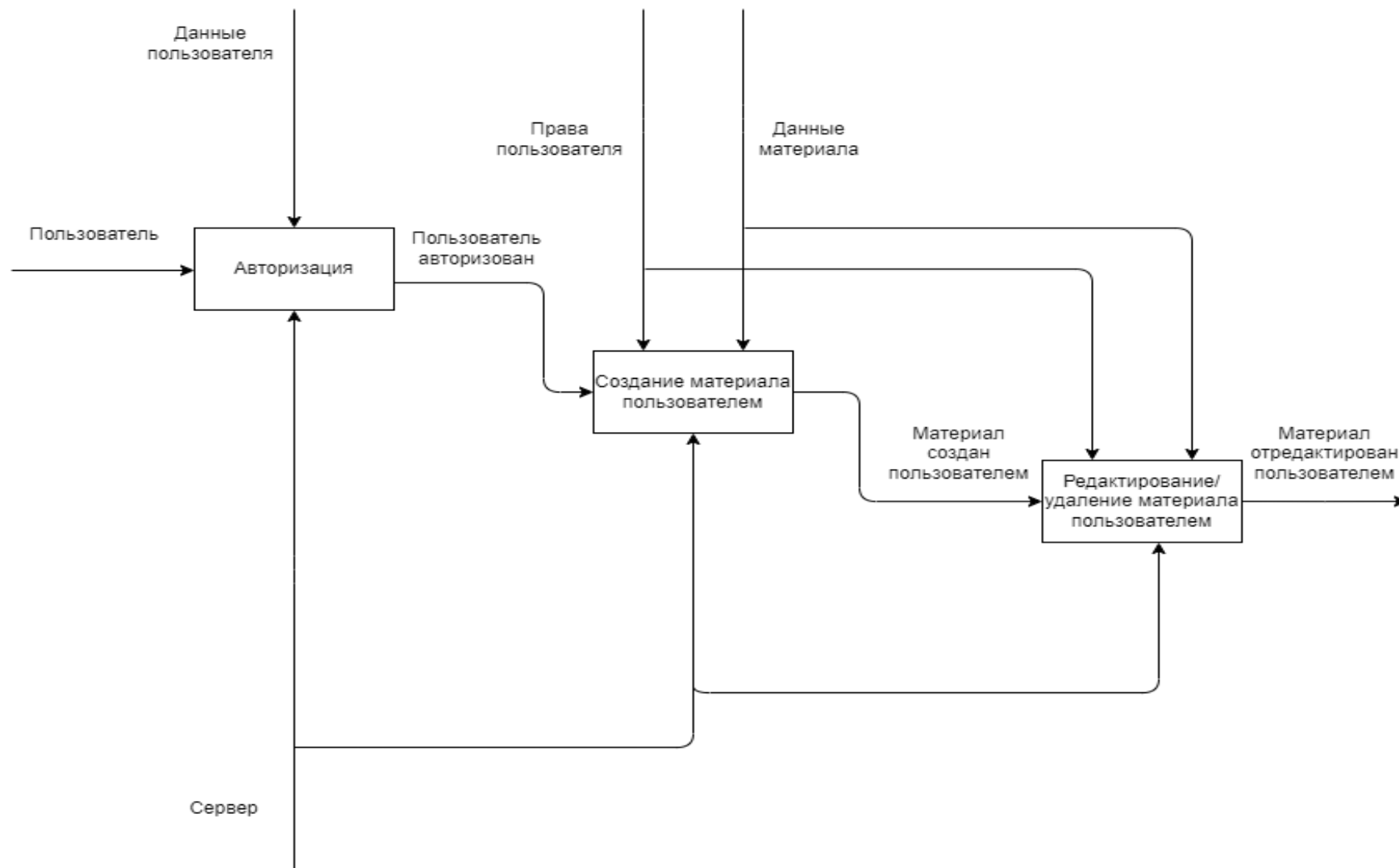


Рисунок 10 – IDEF1–модель процесса редактирования/удаления материала

Выводы по первой части

1. Правильное осознание роли методических материалов в осуществлении и поддержке образовательного процесса в школе определяет возможность и целесообразность разработки информационной системы.
2. Классификация методических материалов по различным признакам позволила систематизировать их для дальнейшего использования в ходе моделирования ИС.
3. Сравнительный анализ существующих аналогов показал, что каждый из них имеет свои преимущества и недостатки, а также позволил выявить совокупность требований пользователей, предъявляемых к сайтам методических материалов, и описать возможности для улучшения и оптимизации рассмотренных подходов в рамках разрабатываемой информационной системы.
4. Моделирование бизнес-процессов позволило структурировать бизнес-логику ИС, которую в дальнейшем необходимо реализовать.

2. МОДЕЛИРОВАНИЕ ИНФОРМАЦИОННОЙ СИСТЕМЫ

2.1 Составление инфологической модели информационной системы

Перед началом проектирования реальной базы данных необходимо составить инфологическую модель информационной системы. «Инфологическая модель – это визуальное представление данных и связей между ними, которое помогает понять и проанализировать структуру информации в определенной области» [10]. Она включает в себя сущности (объекты, факты, понятия), их атрибуты (свойства, характеристики) и отношения между ними (связи, зависимости). Инфологические модели используются для анализа, оптимизации и организации данных, а также для проектирования баз данных и других информационных систем.

Сущность – это основной элемент, который представляет собой объект или явление, описываемое моделью. Сущность имеет определенные атрибуты, которые описывают ее свойства и характеристики, а также связи с другими сущностями в модели. Сущности могут быть как простыми, так и сложными, и могут иметь различный уровень детализации в зависимости от целей моделирования [21].

Атрибут – это характеристика или свойство сущности, которое описывает ее поведение или свойства. Атрибуты могут быть количественными (например, стоимость товара) или качественными (например, цвет товара). Они помогают описать сущность более подробно и точно, а также позволяют сравнивать сущности между собой [21].

Существует несколько основных видов связей между сущностями в базе данных:

- Один к одному (1:1): связь между двумя сущностями, когда каждая запись в одной сущности связана с одной записью в другой сущности.
- Один ко многим (1:M): связь между основной сущностью и зависимой сущностью, когда одна запись основной сущности может быть связана с несколькими записями зависимой сущности.

- Многие ко многим (М:N): связь между двумя сущностями, когда одна запись в каждой сущности может быть связана с множеством записей в другой сущности.

Инфологическая модель послужит основой для составления последующей модели, учитывающей специфику конкретной СУБД.

Описание сущностей информационной системы

сущность ПОЛЬЗОВАТЕЛЬ – содержит следующие поля:

- идентификатор пользователя;
- имя пользователя;
- адрес электронной почты;
- пароль;
- фото профиля;
- идентификатор роли.

Идентификатор пользователя используется для однозначной идентификации записи в таблице базы данных. Имя пользователя содержит имя пользователя на сайте. Адрес электронной почты необходим для возможной коммуникации с автором материала. Пароль используется для авторизации пользователя на сайте. Роль определяет допустимые возможности пользователя.

Материал – сущность включает следующие поля:

- идентификатор материала;
- название материала;
- описание материала;
- файл;
- дата публикации;
- id пользователя;
- id категории;
- id дисциплины;
- id возрастной группы.

Идентификатор материала выполняет функцию аналогичную идентификатору автора для соответствующей таблицы. Название материала определяет наименование материала. Описание материала служит в качестве короткой аннотации, помогающей пользователю решить, подходит ли ему данный материал. Дата публикации отображает время, в которое материал был добавлен на сайт. id пользователя, id категории, id дисциплины, id возрастной группы выполняют роли внешних ключей, ссылающихся на соответствующие таблицы сущностей в базе данных.

Категория – сущность содержит следующие поля:

- идентификатор категории;
- название категории.

Идентификатор категории выполняет функцию аналогичную идентификатору автора для соответствующей таблицы.

Дисциплина – сущность содержит следующие поля:

- идентификатор дисциплины;
- название дисциплины.

Идентификатор дисциплины выполняет функцию, аналогичную идентификатору автора для соответствующей таблицы. Название дисциплины содержит название конкретного учебного предмета.

Возрастная группа – сущность содержит следующие поля:

- идентификатор возрастной группы;
- название возрастной группы.

Идентификатор возрастной группы выполняет функцию, аналогичную идентификатору автора для соответствующей таблицы. Название возрастной группы содержит наименование возрастной группы, для которой предназначен методический материал.

Роль – сущность содержит следующие поля:

- идентификатор роли;
- название роли.

Идентификатор роли выполняет функцию аналогичную идентификатору автора для соответствующей таблицы. Название роли определяет наименование роли, которая наделяет пользователя рядом возможностей.

Избранное – сущность содержит следующие поля:

- идентификатор избранного;
- идентификатор пользователя;
- идентификатор материала.

Идентификатор пользователя и идентификатор материала являются внешними ключами сущностей, зарегистрированных в системе.

Дидактическая единица – сущность содержит следующие поля:

- идентификатор единицы;
- идентификатор материала;
- название.

Идентификатор материала является внешним ключом сущности МАТЕРИАЛ, название определяет наименование дидактической единицы, привязанной к конкретному материалу.

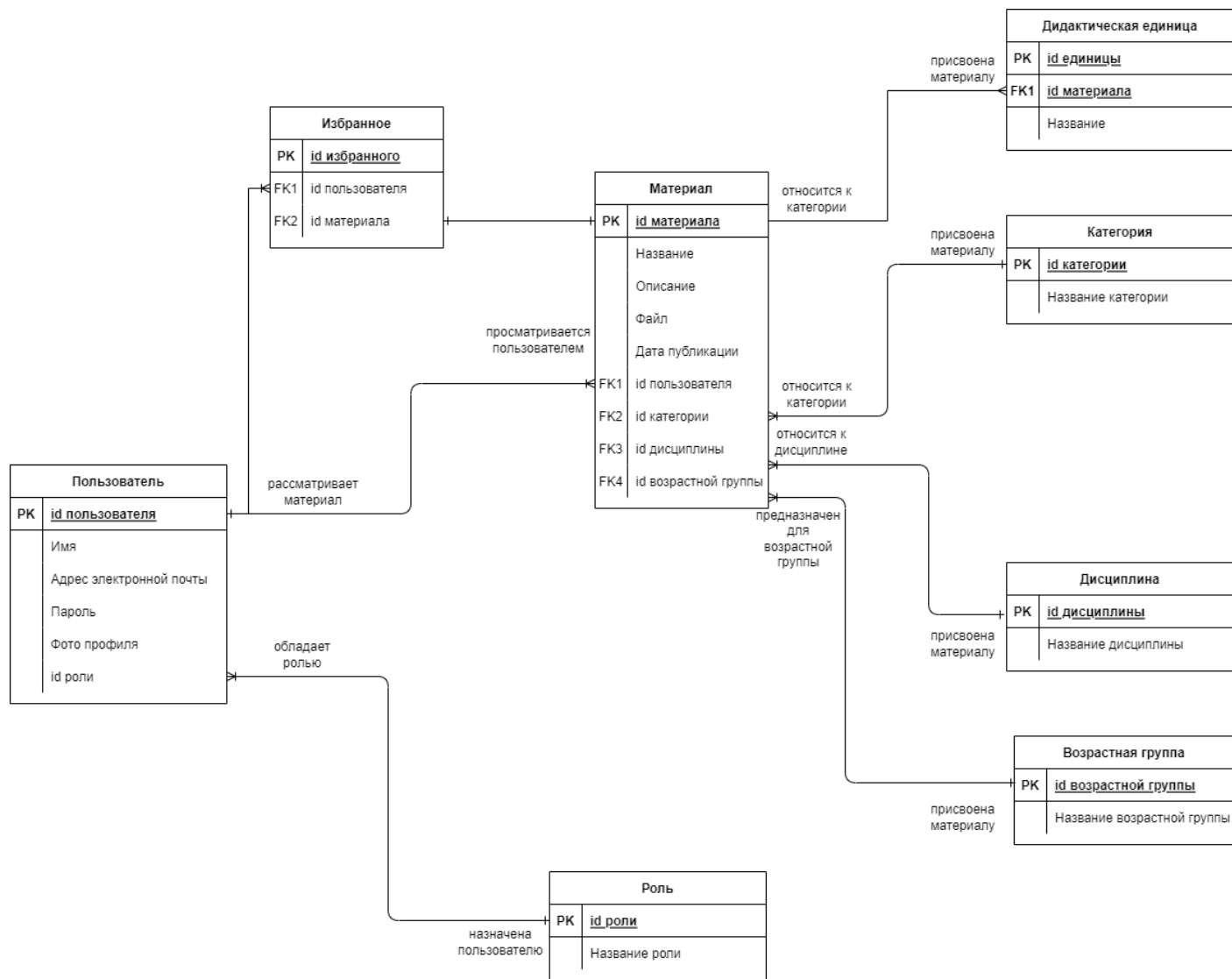


Рисунок 11 – Инфологическая модель

2.2 Разработка даталогической модели информационной системы

Следующей ступенью проектирования структуры базы данных является даталогическая модель. Даталогическая модель, также известная как логическая структура данных, представляет собой более низкоуровневое представление данных, которое учитывает особенности конкретной СУБД и обеспечивает физическую организацию данных. Она определяет типы данных, индексы, ограничения целостности и другие детали реализации, которые необходимы для работы с данными на уровне СУБД. Для отображения взаимосвязей сущностей внутри информационной системы необходимо составить ER–модель. ER–модель для информационной системы должна быть представлена в третьей частной нормальной форме Бойса–Кодда (НФБК).

Причины для этого следующие. Рассмотрим фрагмент базы данных, не приведенной к нормальной форме.

Таблица 1 – Таблица–пример

Материал	Категория
Контрольная работа	Статья, Контрольная работа
Статья	Статья, Контрольная работа

Как видно из таблицы, для назначения категории для материала возникает избыточность данных. Чтобы устранить эту проблему преобразуем таблицу к первой нормальной форме.

Таблица 2 – Приведение к ПНФ

Материал	Категория
Контрольная работа	Контрольная работа
Статья	Статья

В итоге полученная таблица обеспечивает оптимальный доступ к данным. Рассмотрим другой фрагмент таблицы:

Таблица 3 – Таблица–пример

Пользователь	Материал	Категория
Пользователь1	Контрольная работа	Контрольная работа

Полученная таблица содержит в себе поле КАТЕГОРИЯ, не имеющее прямого отношения к пользователю, что негативно влияет на производительность. Верным решением является разделение таблицы, в соответствии со второй нормальной и третьей нормальной формами.

Таблица 4 – Приведение к ТНФ

Пользователь	Материал
Пользователь1	Контрольная работа

Материал	Категория
Контрольная работа	Контрольная работа

Вторая нормальная форма позволит уменьшить количество ошибок при работе с данными посредством добавления первичных ключей, но по-прежнему сохранится избыточность данных. Использование третьей частной нормальной формы Бойса–Кодда (НФБК) позволит сократить количество полей сущности материал и улучшить структуру базы данных, что положительно скажется на производительности и процессе разработки серверной части приложения.

Таблица 5 – Описание сущностей

Сущность	Поле	Тип	Ограничения
Материал	id	int	NOT NULL
	title	string	NOT NULL UNIQUE
	description	string	
	file	string	NOT NULL
	date_publication	date	NOT NULL
	userId	int	NOT NULL
	categoryId	int	NOT NULL
	subjectId	int	NOT NULL
Пользователь	groupId	int	NOT NULL
	id	int	NOT NULL
	name	string	NOT NULL
	email	string	NOT NULL UNIQUE
	password	string	NOT NULL
	profile_img	string	
Роль	roleId	int	NOT NULL
	id	int	NOT NULL
	title	string	NOT NULL UNIQUE

Сущность	Поле	Тип	Ограничения
Категория	id	int	NOT NULL
	title	string	NOT NULL UNIQUE
Дисциплина	id	int	NOT NULL
	title	string	NOT NULL UNIQUE
Возрастная группа	id	int	NOT NULL
	title	string	NOT NULL UNIQUE
Дидактическая единица	id	int	NOT NULL
	id материала	int	NOT NULL
	название	string	NOT NULL UNIQUE
Избранное	id	int	NOT NULL
	id пользователя	int	NOT NULL
	id материала	int	NOT NULL

Описанная выше модель, позволит реализовать элементы бизнес-логики, рассмотренные в пункте 1.5.

2.3 Клиент–серверная архитектура системы

Далее следует определить клиент–серверную архитектуру будущей системы. Существуют различные виды архитектур, обладающих различными преимуществами и недостатками. В рамках реализации данной системы будет использоваться классическая трехуровневая клиент–серверная архитектура.

Трехуровневая клиент–серверная архитектура представляет собой модель организации взаимодействия между клиентом (пользовательским приложением) и сервером (базой данных). Эта модель предполагает наличие промежуточного уровня – бизнес-логики, который обеспечивает разделение функциональных возможностей приложения на отдельные модули.

Причины выбора данной архитектуры, следующие:

- разделение функциональных обязанностей: каждый уровень отвечает за свою часть функциональности, что упрощает разработку, поддержку и масштабирование приложения;
- повышение безопасности: поскольку бизнес-логика отделена от пользовательского интерфейса и базы данных, это снижает риск взлома системы;
- гибкость: возможность изменять или обновлять один из уровней без влияния на другие, что позволяет быстро внедрять новые технологии и улучшать производительность приложения;
- улучшение масштабируемости: разделение компонентов на уровни позволяет легко масштабировать систему по мере увеличения нагрузки на приложение;
- возможность повторного использования кода: компоненты бизнес-логики могут быть использованы в разных проектах, что снижает затраты на разработку.

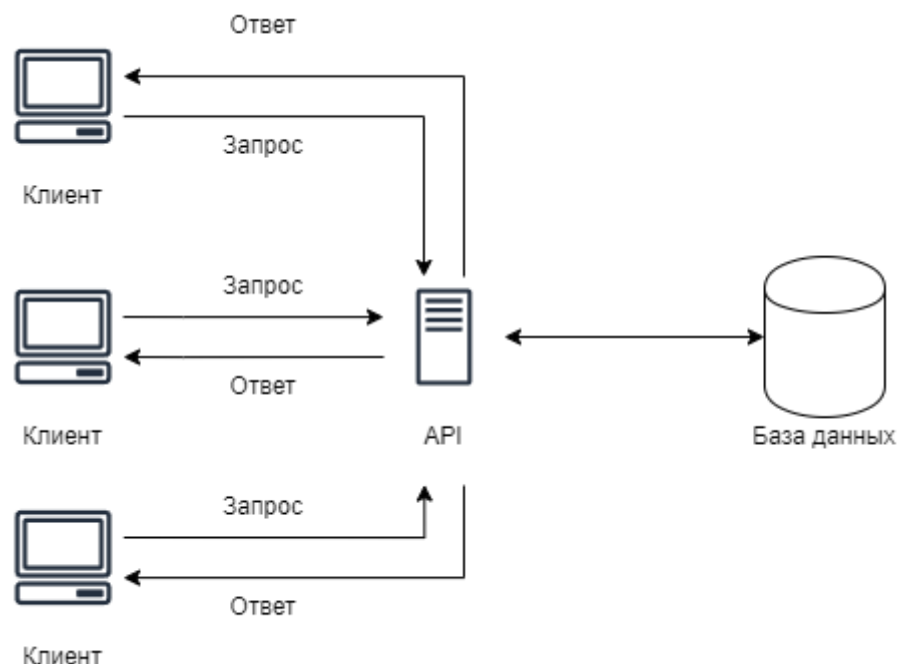


Рисунок 13 – Архитектура системы

Рассмотрим подробнее механизмы работы, описанные выше на примере, представленном на рисунке 13. Изначально клиент отправляет HTTP запрос на сервер, с целью получения данных. Запрос начинает обрабатываться промежуточной прослойкой API. API необходимо для общения с базой данных. После того как API получило данные с базы формируется ответ, который поступает на клиент.

В рамках информационной системы API будет согласно требованиям REST. REST API (Representational State Transfer Application Programming Interface) – это набор правил и форматов, которые определяют, как клиент может взаимодействовать с серверным приложением или веб-сервисом. REST API использует стандартные методы HTTP (GET, POST, PUT, DELETE) и форматы данных (JSON, XML), что позволяет различным платформам и языкам программирования легко интегрироваться с сервером.

Для реализации данной модели система должна быть разделена на слои. Слой “Уровень доступа к приложению” содержит модуль, обеспечивающий взаимодействие пользователя с системой. Слой “Уровень бизнес-логики” содержит модули обработки данных для запрашиваемых или введенных

пользователем. Слой “Уровень доступа к источникам данных” служит для общения предыдущего слоя с базой данных.

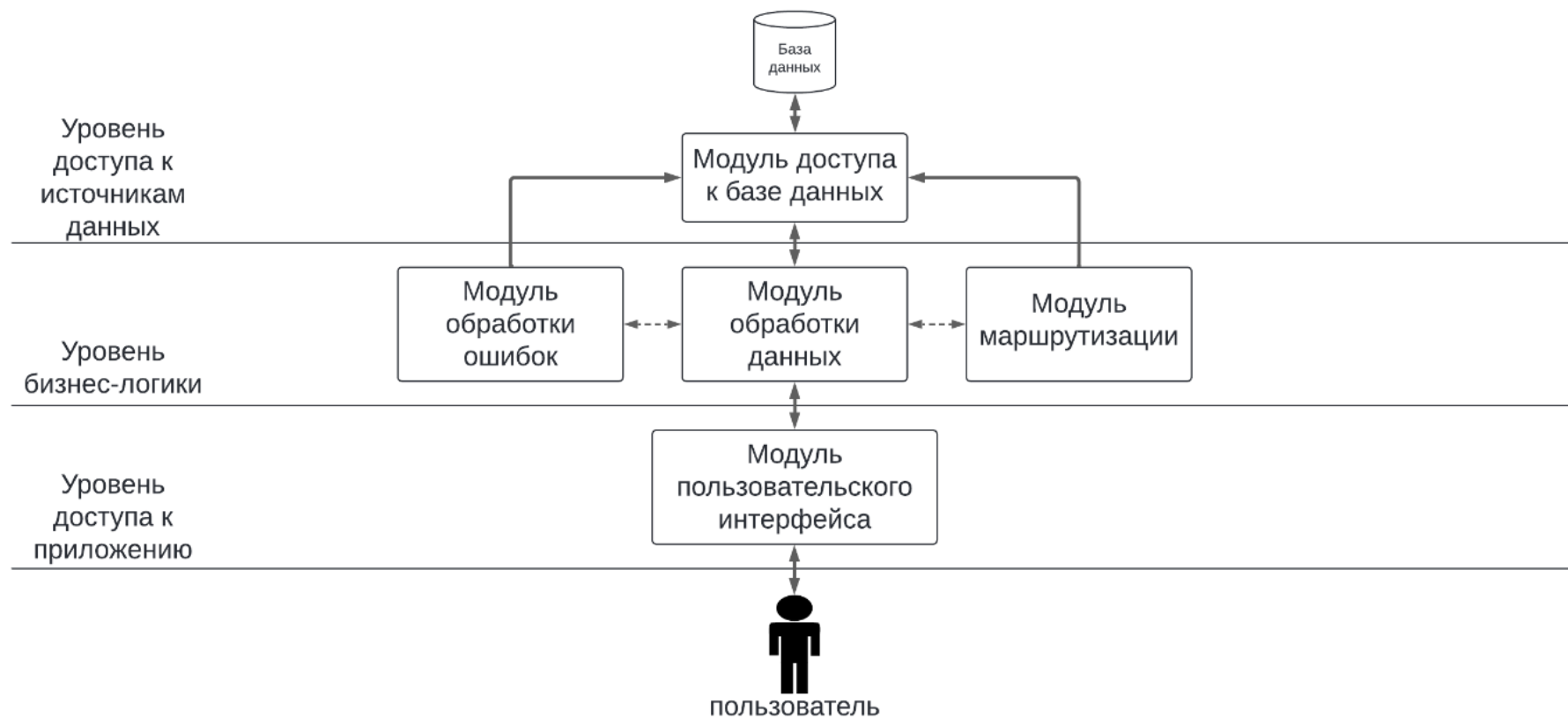


Рисунок 14 – Слоистая архитектура

Выводы по второй части

1. Составленная инфологическая модель определила основные сущности и связи между ними, что позволило сформировать структуру базы данных для дальнейшей разработки системы.
2. Даталогическая модель определила выбор наиболее подходящего типа ключевых атрибутов и их ограничений для каждой сущности, что обеспечит основу для эффективной работы разрабатываемой системы.
3. Выбор клиент–серверной архитектуры позволил разделить систему на две части: клиентскую, которая отвечает за взаимодействие с пользователем, и серверную, которая обрабатывает запросы и выполняет необходимые операции с данными. Это решение способствует оптимизации производительности и масштабируемости ИС.
4. Разработанная модель информационной системы учитывает особенности предметной области и позволяет реализовать поддерживаемое веб–приложение.

3 ПРОГРАММНАЯ РЕАЛИЗАЦИЯ ИС

3.1 Средства разработки информационной системы

Перед началом разработки приложения необходимо рассмотреть выбранный стек технологий.

Для разработки информационной системы были выбраны базовые средства разработки современных веб–приложений:

- HTML (HyperText Markup Language) — это стандартизированный язык гипертекстовой разметки документов для просмотра веб–страниц в браузере [11].
- CSS — формальный язык декорирования и описания внешнего вида документа (веб–страницы), написанного с использованием языка разметки [11].
- JavaScript — мультипарадигменный язык программирования. Поддерживает объектно–ориентированный, императивный и функциональный стили [11].
- PostgreSQL — это бесплатная СУБД с открытым исходным кодом. С помощью PostgreSQL можно создавать, хранить базы данных и работать с данными с помощью запросов на языке SQL [12].
- Express – это минималистичный и гибкий веб–фреймворк для приложений Node.js, предоставляющий обширный набор функций для мобильных и веб–приложений [13].
- React — JavaScript–библиотека с открытым исходным кодом для разработки пользовательских интерфейсов [14].
- Node или Node.js — программная платформа, основанная на движке V8 (компилирующем JavaScript в машинный код), превращающая JavaScript из узкоспециализированного языка в язык общего назначения. Node.js добавляет возможность JavaScript взаимодействовать с устройствами ввода–вывода через свой API, написанный на C++, подключать другие внешние библиотеки, написанные на разных языках, обеспечивая вызовы к

ним из JavaScript-кода. Node.js применяется преимущественно на сервере, выполняя роль веб-сервера. В основе Node.js лежит событийно-ориентированное и асинхронное (или реактивное) программирование с неблокирующим вводом/выводом [13].

- Sequelize — это ORM (Object-Relational Mapping — объектно-реляционное отображение или преобразование) для работы с такими СУБД (системами управления (реляционными) базами данных, Relational Database Management System, RDBMS), как Postgres, MySQL, MariaDB, SQLite и MSSQL [15].

Выбранный стек технологий позволяет реализовать описанную в пунктах 2.2/2.3 модель базы данных и архитектуру системы.

3.2 Создание базы данных

Первоначально необходимо создать проект. Для разработки используется IDE WebStorm. Структура проекта выглядит следующим образом:

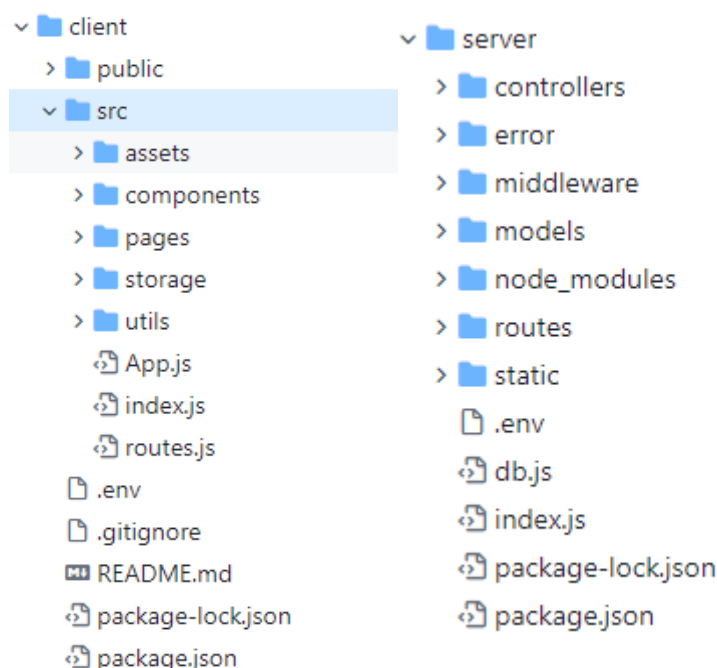


Рисунок 15 – Структура проекта

Далее необходимо настроить подключение базы данных в проекте. Для этого в файле окружения .env запишем ряд переменных, отвечающих за подключение к базе данных:

DB_NAME=VKR – название базы данных
DB_USER=postgres – имя пользователя
DB_PASSWORD=password – пароль
DB_HOST=localhost – хост базы данных
DB_PORT=5432 – порт подключения

Далее создадим отдельный файл db.js где при помощи ORM Sequelize создадим подключение к базе данных:

```
const {Sequelize} = require('sequelize')

module.exports = new Sequelize(
  process.env.DB_NAME,
  process.env.DB_USER,
  process.env.DB_PASSWORD,
  {
    dialect: 'postgres',
    host: process.env.DB_HOST,
    port: process.env.DB_PORT
  }
)
```

После для проектирования базы данных согласно разработанной ER-модели необходимо определить модели сущностей в отдельном файле models.js.

```
const sequelize = require('../db')
const {DataTypes} = require('sequelize')

const User = sequelize.define('user', {
  id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement: true},
  name: {type: DataTypes.STRING, unique: true},
  email: {type: DataTypes.STRING, unique: true},
  password: {type: DataTypes.STRING},
  profile_img: {type: DataTypes.STRING},
})
```

```

const Material = sequelize.define('material', {
  id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement: true},
  title: {type: DataTypes.STRING, unique: true,
allowNull: false},
  description: {type: DataTypes.STRING},
  date_publication: {type: DataTypes.DATEONLY},
  file: {type: DataTypes.STRING}
}, {
  createdAt:false,
  updatedAt:false
})

const Tag = sequelize.define('tag', {
  id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement: true},
  title: {type: DataTypes.STRING, unique: true,
allowNull: false},
})

const MaterialFavorite =
sequelize.define('material_favorite', {
  id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement: true},
})

const Category = sequelize.define('category', {
  id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement: true},
  title: {type: DataTypes.STRING, unique: true,
allowNull: false},
})

const Subject = sequelize.define('subject', {
  id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement: true},
  title: {type: DataTypes.STRING, unique: true,
allowNull: false},
})

const Group = sequelize.define('group', {
  id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement: true},

```

```

    title: {type: DataTypes.STRING, unique: true,
allowNull: false},
  })

const Role = sequelize.define('role', {
  id: {type: DataTypes.INTEGER, primaryKey: true,
autoIncrement: true},
  title: {type: DataTypes.STRING, unique: true,
allowNull: false},
  })

User.hasMany(Material)
Material.belongsTo(User)

Material.hasOne(MaterialFavorite)
MaterialFavorite.belongsTo(Material)

Material.hasMany(Tag, {as: 'tags'})
Tag.belongsTo(Material)

User.hasMany(MaterialFavorite)
MaterialFavorite.belongsTo(User)

Category.hasMany(Material)
Material.belongsTo(Category)

Subject.hasMany(Material)
Material.belongsTo(Subject)

Group.hasMany(Material)
Material.belongsTo(Group)

Role.hasMany(User)
User.belongsTo(Role)

module.exports = {
  User,
  Material,
  Tag,
  MaterialFavorite,
  Category,
  Subject,
  Group,

```

```
    Role
  }
```

Модели User, Material, Tag, MaterialFavorite, Category, Subject, Group и Role полностью копируют типы данных и ограничения сущностей ПОЛЬЗОВАТЕЛЬ, МАТЕРИАЛ, ДИДАКТИЧЕСКАЯ ЕДИНИЦА, ИЗБРАННОЕ, КАТЕГОРИЯ, ДИСЦИПЛИНА, ВОЗРАСТНАЯ ГРУППА и РОЛЬ, описанных во второй части дипломной работы. Функции hasMany и belongsTo – встроенные функции модуля Sequelize связывающие сущности базы данных связями “один ко многим” и “один к одному”.

После необходимо выполнить команду `npm run dev`, все настройки, описанные в моделях, автоматически применятся к базе данных. Далее перейдем к реализации бизнес-логики приложения, рассмотренной в пункте 1.5.

3.3 Реализация бизнес-логики сущности ПОЛЬЗОВАТЕЛЬ

Перейдем к рассмотрению реализации бизнес-логики сущности пользователь. Первая из них – функция registration используется для регистрации нового пользователя(Бизнес-процесс "Регистрация").

```
const ApiError = require('../error/ApiError')
const bcrypt = require('bcrypt')
const jwt = require('jsonwebtoken')
const {User, Subject, Material} =
  require('../models/models')
const uuid = require("uuid");
const path = require("path");

const generateJwt = (id, name, email, roleId) => {
  return jwt.sign(
    {id, name, email, roleId},
    process.env.SECRET_KEY,
    {expiresIn: '1h'}
  )
}

class UserController {
  async registration(req, res, next) {
    const {name, email, password, roleId} =
      req.body
```

```

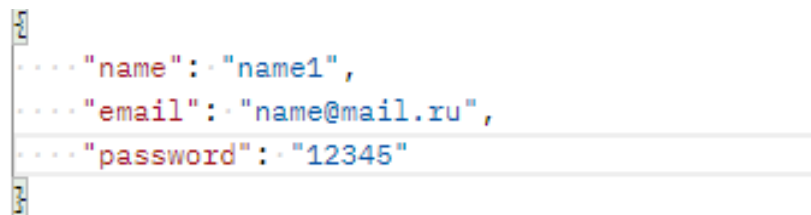
        if (!name || !email || !password) {
            return
        }
        next(ApiError.badRequest('Некоректное заполнение
полей!'))
    }
    const candidate = await User.findOne({where:
{email}})
    if (candidate) {
        return
    }
    next(ApiError.badRequest('Пользователь с таким email
уже существует'))
    }
    const hashPassword = await
bcrypt.hash(password, 5)
    const user = await User.create({name, email,
roleId, password: hashPassword})
    const token = generateJwt(user.id, user.name,
user.email, user.roleId)
    return res.json({token})
}

```

В функции используются асинхронные методы `async/await`, так как функция взаимодействует с базой данных. На вход принимается три параметра:

- `req` – запрос от клиента
- `res` – ответ от сервера
- `next` – функция серверного `middleware`

При срабатывании функции с клиента поступает запрос вида:



```

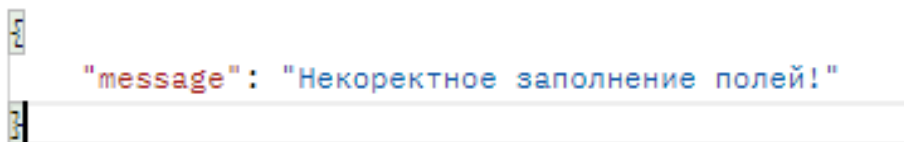
{
  "name": "name1",
  "email": "name@mail.ru",
  "password": "12345"
}

```

Рисунок 16 – Запрос клиента

После чего выполняется его деструктуризация. Далее выполняется проверка на правильность заполнения данных, если данные заполнены

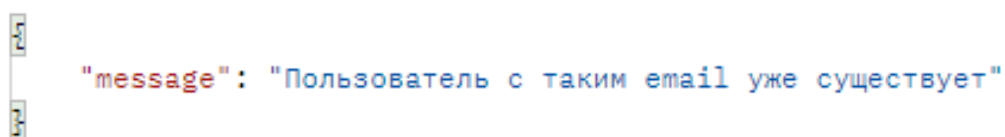
некорректно срабатывает функция серверного middleware и клиент получает следующий ответ от сервера:



```
{  
  "message": "Некорректное заполнение полей!"  
}
```

Рисунок 17 – Ответ сервера в случае ошибки

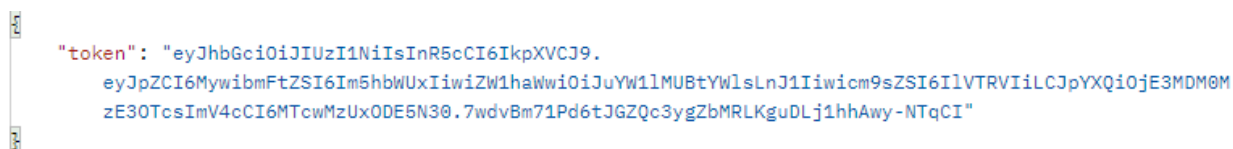
Если данные введены верно, но на почту, введенную пользователем, уже зарегистрирована запись в базе данных сервер отвечает следующим образом:



```
{  
  "message": "Пользователь с таким email уже существует"  
}
```

Рисунок 18 – Ответ сервера, если указана зарегистрированная почта

В случае, когда данные введены корректно выполняется хэширование пароля по алгоритму bcrypt. Далее в базу данных добавляется новая запись, пароль пользователя в которой хеширован. После для пользователя формируется JSON Web Token. JSON Web Token (JWT) — это JSON объект, который определен в открытом стандарте RFC 7519. Он считается одним из безопасных способов передачи информации между двумя участниками. Для его создания необходимо определить заголовок (header) с общей информацией по токену, полезные данные (payload), такие как id пользователя, его роль и т.д. и подписи (signature). JWT токен формируется в отдельной функции generateJwt, в которой данные введенные пользователем, помещаются в payload токена. Далее сервер отправляет на клиент следующие сообщение:



```
{  
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.  
    eyJpZCI6MywibmFtZSI6Im5hbWUxIiwiaWwiOiJ1YUw1MUBtYUwlsLnJ1Iiwicm9sZSI6IlVTRVVIiLCJpYXQiOiJlE3M0M0M  
    zE3OTcsImV4cCI6MTcwMzUxODE5N30u7wdvBm71Pd6tJGZQc3YgZbMRLKguDLj1hhAwy-NTqCI"  
}
```

Рисунок 19 – Ответ сервера при успешной работе функции

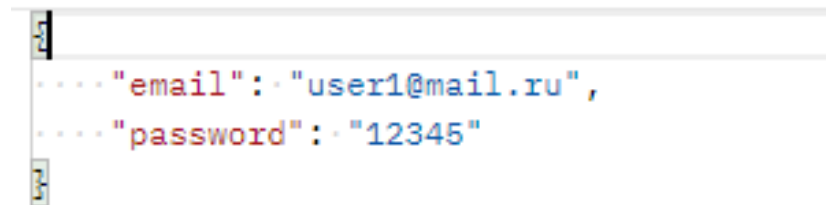
Следующая функция ввода данных — функция авторизации пользователя в системе login(Бизнес-процесс "Авторизация").

```

    async login(req, res, next) {
        const {email, password} = req.body
        const user = await User.findOne({where:
{email}}})
        if (!user) {
            return next(ApiError.internal('Пользователь
не найден'))
        }
        let comparePassword =
bcrypt.compareSync(password, user.password)
        if (!comparePassword) {
            return next(ApiError.internal('Неверный
пароль'))
        }
        const token = generateJwt(user.id, user.name,
user.email, user.roleId)
        return res.json({token})
    }

```

Функция аналогично функции registration принимает три аргумента. Запрос от клиента выглядит следующим образом:



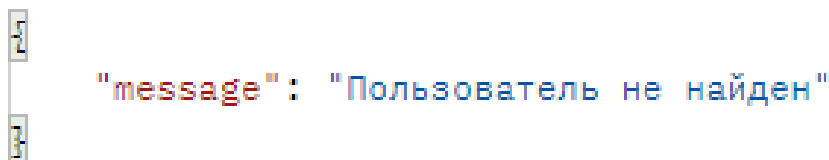
```

{
  "email": "user1@mail.ru",
  "password": "12345"
}

```

Рисунок 20 – Данные клиента

Выполняется деструктуризация запроса клиента. Далее происходит проверка на корректность введенных данных. В случае, если указанный адрес электронной почты не найден в базе данных, сервер отправляет на клиент сообщение:



```

{
  "message": "Пользователь не найден"
}

```

Рисунок 21 – Ответ от сервера для несуществующего пользователя

Если введенный пользователем пароль не является верным, пользователь получит следующее сообщение:

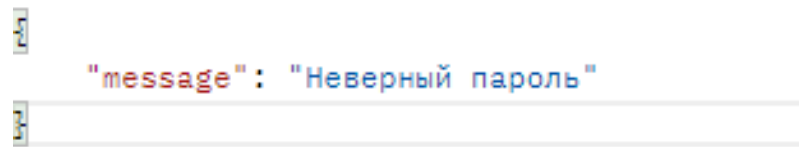


Рисунок 22 – Ответ сервера при вводе неверного пароля

При корректном заполнении данных пользователь получает новый JWT токен и успешно входит в систему.

Далее рассмотрим функцию `check`. Ее функционал заключается в генерации нового токена для пользователя.

```
async check(req, res, next) {  
  const token = generateJwt(req.user.id,  
    req.user.name, req.user.email, req.user.roleId)  
  return res.json({token})  
}
```

Функция принимает запрос, в теле которого находятся данные пользователя, далее данные принимает функция `generateJwt` и после этого пользователю возвращается сгенерированный токен.

На этом функции, реализованные в API, заканчиваются. Рассмотрим использование реализованных функций на стороне клиента. Для выполнения HTTP-запросов используется библиотека `axios`.

```
import axios from "axios";  
const host = axios.create({  
  baseURL: process.env.REACT_APP_API_URL  
})  
  
const hostAuth = axios.create({  
  baseURL: process.env.REACT_APP_API_URL  
})  
  
const authInterceptor = config => {  
  config.headers.authorization = `Bearer  
    ${localStorage.getItem('token')}`  
  return config  
}  
  
hostAuth.interceptors.request.use(authInterceptor)
```



```
export {
  host,
  hostAuth
}
```

Для работы с `axios` необходимо создать два экземпляра класса. Объект `host` будет использоваться для запросов, которые не требуют наличия прав зарегистрированного пользователя, `hostAuth` позволяет выполнять любые запросы.

Далее приведены функции, которые позволяют клиенту получать данные из базы данных посредством использования API, реализованного выше.

```
import {host, hostAuth} from "../index";
import {jwtDecode} from "jwt-decode";
export const registration = async (name, email,
password) => {
  const {data} = await
host.post('api/user/registration', {name, email,
password, roleId:'1'})
  localStorage.setItem('token', data.token)
  return jwtDecode(data.token)
}

export const login = async (email, password) => {
  const {data} = await host.post('api/user/login',
{email, password})
  localStorage.setItem('token', data.token)
  return jwtDecode(data.token)
}

export const fetchUsers = async () => {
  const {data} = await host.get('api/user')
  return (data)
}

export const updateProfileImg = async (user) => {
  const {data} = await host.put('api/user/' +
user.id, user)
  return data
}

export const check = async () => {
```

```

    const {data} = await hostAuth.get('api/user/auth')
    localStorage.setItem('token', data.token)
    return jwtDecode(data.token)
  }

```

3.4 Реализация бизнес-логики сущности МАТЕРИАЛ

Рассмотрим API для взаимодействия с сущностью МАТЕРИАЛ.

Функция `create` позволяет добавить новую запись (Бизнес-процесс "Добавление нового материала").

```

async create(req, res, next) {
  try {
    let {title, description, date_publication,
        userId, categoryId, subjectId, groupId, tags} =
    req.body
    const {file} = req.files
    if (!title || !description) {
      return
    }
    next(ApiError.badRequest('Некорректное заполнение
    полей!'))
  }
  const existing = await Material.findOne({where:
    {title}})
  if (existing) {
    return next(ApiError.badRequest('Материал с
    таким названием уже существует'))
  }
  let fileName = uuid.v4() + ".pdf"
  file.mv(path.resolve(__dirname, '..', 'static',
    fileName))
  const material = await Material.create({title,
    description, date_publication, userId, categoryId,
    subjectId, groupId, file: fileName})

  if (tags) {
    tags = JSON.parse(tags).forEach(async tag
=> {
      const existingTag = await
    Tag.findOne({where: {title: tag.title}})
      if (!existingTag) {
        const newTag = await
    Tag.create({title: tag.title, materialId: material.id})
      }
    })
  }
}

```

```

        return res.json(material)
    } catch (e) {
        next(ApiError.badRequest(e.message))
    }
}

```

Аналогично предыдущим функциям функция принимает три аргумента. Запрос от пользователя выглядит следующим образом:

KEY	VALUE
title	Новый материал
description	Описание материала
date_publication	2023-12-18 20:08:20.563+03
file	<input type="text" value="some_file.pdf"/>
userId	1
categoryId	1
subjectId	2
groupId	1

Рисунок 23 – Запрос от клиента

В блоке try выполняется деструктуризация запроса от клиента req. После принятому от клиента файлу назначается новое уникальное имя для хранения на сервере. При формировании имени файла используется библиотека uuid. После этого в базе данных появляется новая запись для соответствующего материала при помощи функции create из ORM Sequelize. После выполнения функции сервер отправляет ответ:

```

{
  "id": 2,
  "title": "Новый материал",
  "description": "Описание материала",
  "date_publication": "2023-12-18T17:08:20.563Z",
  "userId": 1,
  "categoryId": 1,
  "subjectId": 2,
  "groupId": 1,
  "file": "edda95d0-64a2-4a42-8d92-28798cddff08.pd",
  "updatedAt": "2023-12-25T23:02:08.172Z",
  "createdAt": "2023-12-25T23:02:08.172Z"
}

```

Рисунок 24 – Ответ от сервера

В случае ошибки происходит обработка ошибки middleware, после чего код ошибки и сообщение об ошибке отправляются на клиент.

Далее перейдем к рассмотрению реализации функции получения всех материалов `getAll`.

```
async getAll(req, res, next) {
  try {
    let {userId, subjectId, groupId,
categoryId, title, limit, page} = req.query
    page = page || 1
    limit = limit || 9
    let offset = page * limit - limit
    let where = {}
    if (userId) where.userId = userId
    if (subjectId) where.subjectId = subjectId
    if (groupId) where.groupId = groupId
    if (categoryId) where.categoryId =
categoryId
    let likes = []
    if (title) {

likes.push(Sequelize.where(Sequelize.fn('LOWER',
Sequelize.col('title')), 'LIKE', '%' +
title.toLowerCase() + '%'))
    }
    if (likes.length > 0) {
      where[Sequelize.Op.and] = likes
    }
    const materials = await
Material.findAndCountAll({where, limit, offset, likes})
    return res.json(materials)
  } catch (e) {
    next(ApiError.badRequest(e.message))
  }
}
```

Функция принимает от пользователя query параметры, далее в зависимости от переданных параметров выполняется фильтрация данных, после чего пользователь получает интересующие его данные.

Функция `getOne` выполняет функцию аналогичную `getAll`, с той разницей что пользователь всегда получает один конкретный материал. Это используется в клиентской части приложения, чтобы пользователь мог

рассмотреть страницу с конкретным материалом для возможного скачивания(Бизнес-процесс "Скачивание материала").

```
async getOne(req, res, next) {
  try {
    const {id} = req.params
    const material = await Material.findOne(
      {
        where: {id}
      }
    )
    return res.json(material)
  } catch (e) {
    next(ApiError.badRequest(e.message))
  }
}
```

Следующая функция updateMaterial реализует возможность изменять уже существующий материал(Бизнес-процесс "Изменение материала").

```
async updateMaterial(req, res, next) {
  try {
    const {id, title, description,
date_publication, userId, categoryId, subjectId,
groupId} = req.body
    const {file} = req.files
    if (!title || !description) {
      return
    }
    next(ApiError.badRequest('Некоректное заполнение
полей!'))
  }
  let fileName = uuid.v4() + ".pdf"
  file.mv(path.resolve(__dirname, '..',
'static', fileName))
  const material = await
Material.update({title, description, date_publication,
userId, categoryId, subjectId, groupId, file:
fileName}, {where: {id}})
  return res.json(material)
}
catch (e) {
  next(ApiError.badRequest(e.message))
}
}
```

На входе функция ожидает от пользователя данные по изменяемому материалу, после чего выполняется запрос к базе данных при помощи функции `Material.update`. В случае, если запрос от пользователя был корректным, сервер возвращает ответ с обновленным материалом, иначе будет отправлено сообщение с кодом ошибки.

```
async toggleFavorite(req, res, next) {
  try {
    const {userId, materialId} = req.body
    const candidate = await
MaterialFavorite.findOne({where: {userId, materialId}})
    if (candidate) {
      await MaterialFavorite.destroy({where:
{userId, materialId}})
      return res.json({message: 'удалено из
избранного'})
    } else {
      await MaterialFavorite.create({userId,
materialId})
      return res.json({message: 'добавлено в
избранное'})
    }
  } catch (e) {
    next(ApiError.badRequest(e.message))
  }
}

async getFavorites(req, res) {
  const {id} = req.query
  const favorites = await
MaterialFavorite.findAll({where: {userId: id}})
  if (!favorites) {
    return res.json([])
  }
  const materials = await
Material.findAll({where: {id: favorites.map(favorite =>
favorite.materialId)}})
  return res.json(materials)
}
```

Функции `getFavorite` и `toggleFavorite` отвечают за получение списка избранных материалов пользователем и их добавлением/удалением. `getFavoritess` основана на логике, рассмотренной в функции `getAll`.

toggleFavorite изначально проверяет находится ли материал в избранном, если это так, материал удаляется из списка избранного, иначе будет добавлен в список.

Последней функцией в API является функция deleteMaterial(Бизнес-процесс "Удаление"). Она позволяет пользователю удалять принадлежащие ему материалы с сайта. На входе функция ожидает id материала, после чего соответствующий материал будет удален функцией Material.destroy.

```
async deleteMaterial(req, res, next) {
  try {
    const {id} = req.params
    const material = await Material.destroy({
      where : {id}
    })
  }
  catch (e) {
    next(ApiError.badRequest(req.body))
  }
}
```

Рассмотрим реализацию обращения к API на стороне клиента.

```
import {host, hostAuth} from "../index";

export const createMaterial = async (material) => {
  const {data} = await hostAuth.post('api/material',
material)
  return data
}

export const fetchMaterials = async (userId, subjectId,
groupId, categoryId, page, limit= 5) => {
  const {data} = await host.get('api/material',
{params: {
    userId, subjectId, groupId, categoryId, page,
limit
  }})
  return data
}

export const fetchOneMaterial = async (id) => {
  const {data} = await host.get('api/material/' + id)
  return data
}
```

```

}

export const updateMaterial = async (id, material) => {
  const {data} = await hostAuth.put('api/material/' +
id, material)
  return data
}
export const deleteMaterial = async (id) => {
  const {data} = await
hostAuth.delete('api/material/' + id)
  return data
}

```

Функции `createMaterial`, `fetchMaterials`, `fetchOneMaterial`, `updateMaterial` и `deleteMaterial` описывают обращение клиента, к соответствующим функциям рассмотренного выше API.

3.5 Тестирование и демонстрация работы информационной системы

На завершающем этапе создания информационной системы проводится проверка уже реализованных компонентов и решений. С целью обеспечения корректности и стабильности системы требуется изучить разные сценарии взаимодействия с ней. Проверка помогает найти и исправить ошибки, а также обнаружить недостатки, которые могут сказаться на работоспособности и производительности системы.

Также будут продемонстрированы разделы, которые выполняют исключительно информационную функцию и не нуждаются в тестировании.

Изначально рассмотрим разделы регистрации и авторизации, в которых пользователю предлагается заполнить необходимые данные для получения прав доступа к полной функциональности приложения. Соответствующие формы представлены на рисунках 25 и 26.

Регистрация

Введите имя

Введите email

Введите пароль

Уже есть аккаунт? [Войти](#)

[Зарегистрироваться](#)

Рисунок 25 – Форма регистрации

Авторизация

use23c@mail.ru

...



Нет аккаунта? [Создать](#)

[Войти](#)


Рисунок 26 – Форма авторизации

В случае если какое-либо из полей заполнено некорректно пользователь получит сообщение об ошибке, изображенное на рисунке 27.

Авторизация

user|  

Введен некорректный email!

Введите пароль 

Некорректное заполнение поля!

Нет аккаунта? [Создать](#)

[Войти](#)

Рисунок 27 – Сообщение об ошибке

После авторизации в системе пользователь попадает на главную страницу приложения. В верхней части страницы находится навигационная

панель, содержание которой меняется в зависимости от прав пользователя. В случае, если пользователь не является администратором, пункт “Панель администратора” не будет отображен. В левой части сайта представлено меню для сортировки по предметам. В центральной части сайта отображены кнопки для сортировки по категориям и возрастным группам, кнопка для добавления материалов и список опубликованных материалов. Интерфейс главной страницы представлен на рисунке 28.

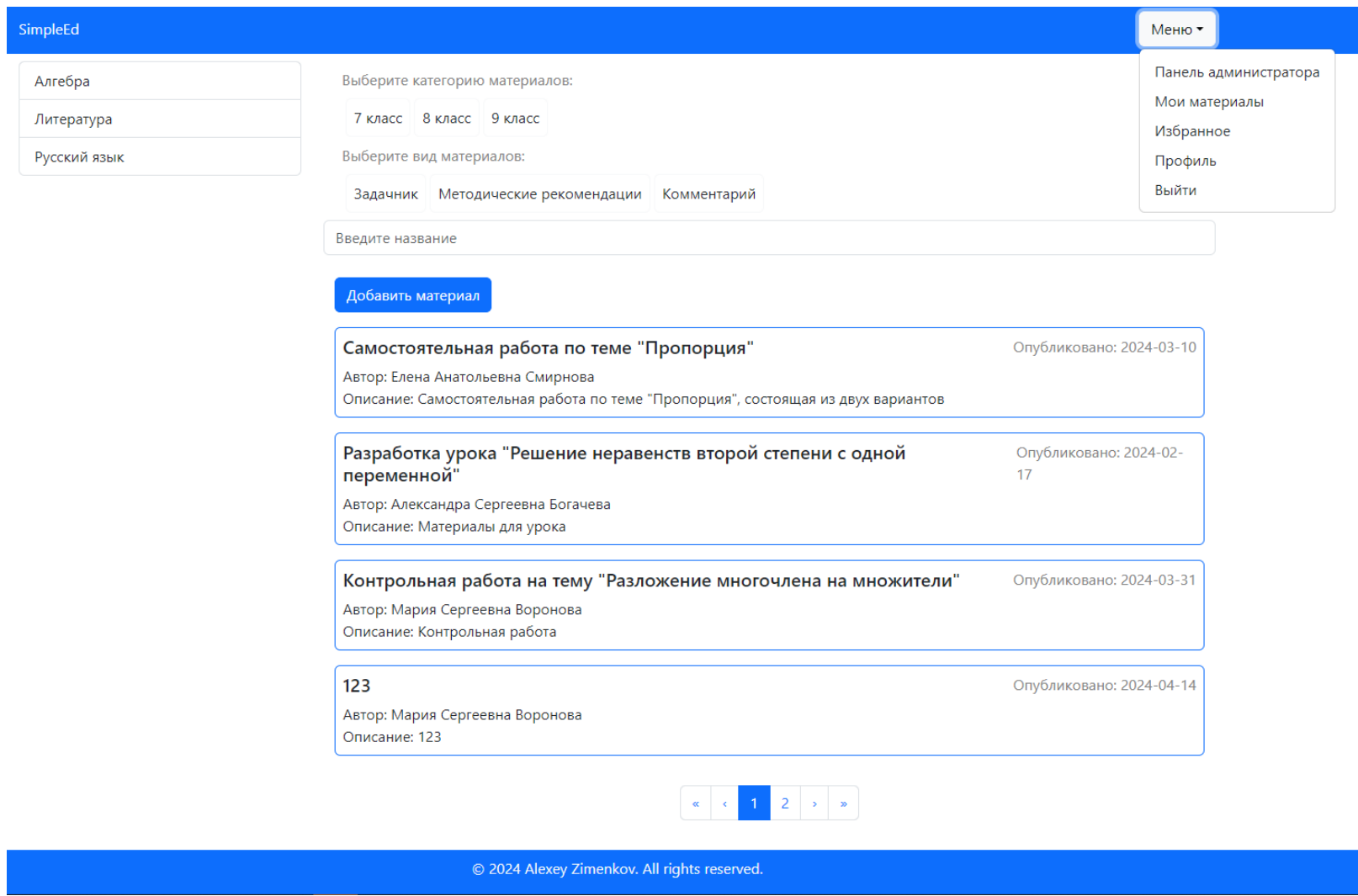


Рисунок 28 – Главная страница

При нажатии на кнопку “Добавить материал” открывается модальное меню, где пользователю необходимо выбрать значения из выпадающих списков для предмета, категории и возрастной группы соответственно, заполнить поля “Название” и “Описание”, а также выбрать файл с методическим материалом. После этого необходимо нажать на кнопку “Добавить” и материал будет опубликован. Интерфейс модального окна добавления нового материала представлен на рисунке 28.

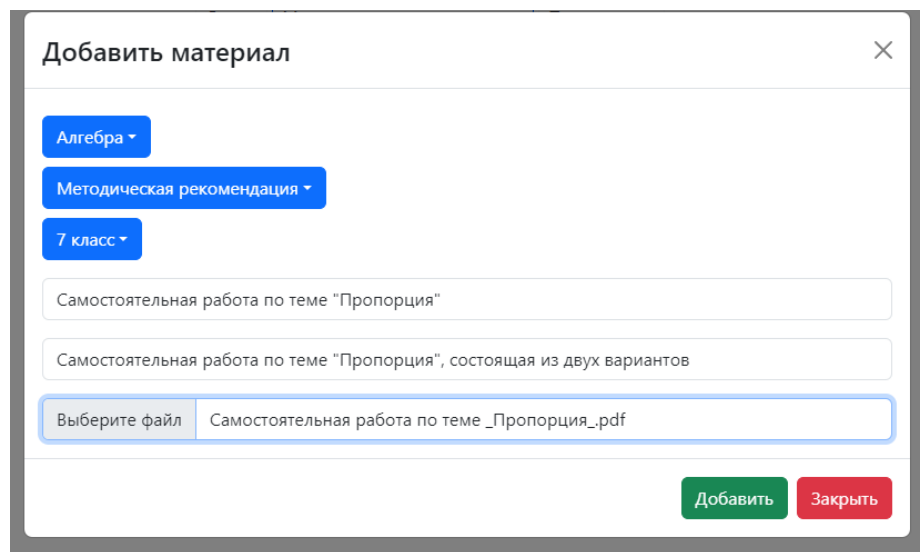


Рисунок 29 – Модальное окно добавления материала

При нажатии на карточку материала, пользователь переходит на страницу конкретного материала, где можно подробнее ознакомиться с содержанием или скачать файл с материалом. Представление страницы конкретного материала изображено на рисунке 30.

Контрольная работа на тему "Разложение многочлена на множители"

Методические рекомендации

9 класс

**Самостоятельная работа «Разложение многочлена на множители»
1 вариант****№1.** Разложить многочлен на множители:

3. $14 - 14m^2$, 2) $3a - 3a^3$, 3) $3x^2 - 24xy + 48y^2$, 4) $-3a^4 - 12a^3 - 12a^2$,

5) $a^2 - 2ab + b^2 - 25$, 6) $a + 5b + a^2 - 25b^2$, 7) $x^2 - y^2 - 6x + 9$.

№2. Решить уравнение:

3. $7x^3 - 63x = 0$, 2) $49x^3 - 14x^2 + x = 0$, 3) $x^3 - 5x^2 - x + 5 = 0$,

4) $x^3 - 3x^2 - 4x + 12 = 0$, 5) $x^4 + 2x^3 + 8x + 16 = 0$.

**Самостоятельная работа «Разложение многочлена на множители»
2 вариант****№1.** Разложить многочлен на множители:

1. $7x^2 - 28$, 2) $3a^3 - 108a$, 3) $3x^2 - 48xy + 192y^2$, 4) $-75a^6 + 30a^4 - 3a^2$,

5) $x^2 + 2xy + y^2 - 64$, 6) $m^2 + 16n^2 + 8mn - b^2$, 7) $a^2 - c^2 - 6a + 9$.

№2. Решить уравнение:

Автор: Мария Сергеевна Воронова

Контрольная работа

Скачать



Рисунок 30 – Страница материала

При нажатии кнопки “Мои материалы” в навигационной панели, пользователь перейдет на страницу, где отображены все материалы, добавленные пользователем. При нажатии кнопки “Редактировать” открывается модальное окно, аналогичное добавления материала, которое позволяет обновить данные материала. Нажатие кнопки “Удалить” удаляет материал с сайта. Интерфейс страницы с материала пользователя отображен на рисунке 31.

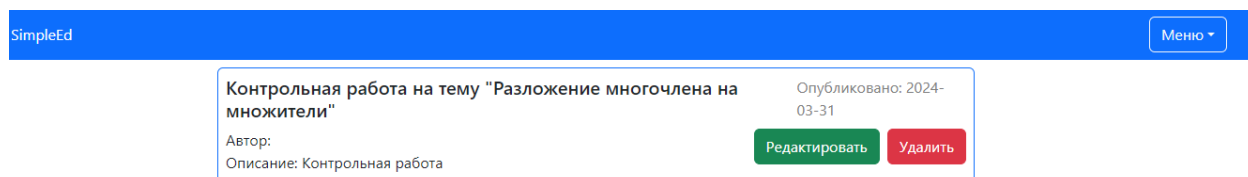


Рисунок 31 – Страница “Мои материалы”

При нажатии кнопки “Профиль” в навигационной панели пользователь переходит в свой личный кабинет, где указана его личная информация и можно изменить фото профиля. Интерфейс страницы профиля показан на рисунке 32.

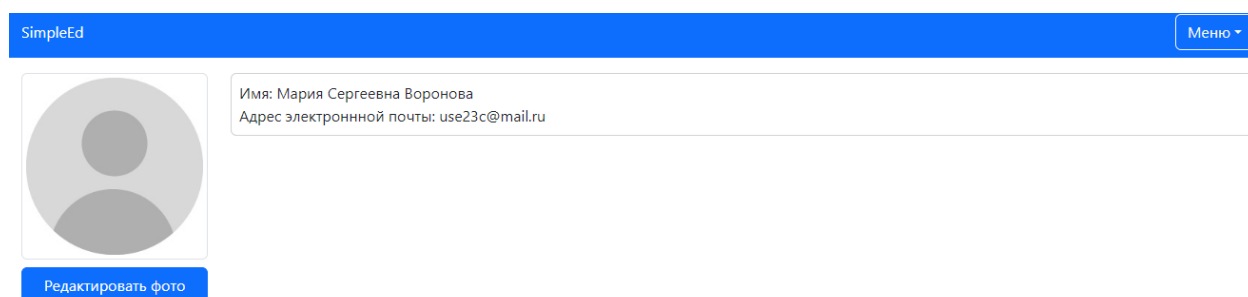


Рисунок 32 – Страница профиля

В случае, если пользователь является администратором в навигационной панели доступна дополнительная кнопка “Панель администратора”. При ее нажатии администратор переходит на страницу, на которой возможно добавлять новые учебные предметы, категории и возрастные группы методических материалов. Интерфейс панели администратора представлен на рисунке 33.

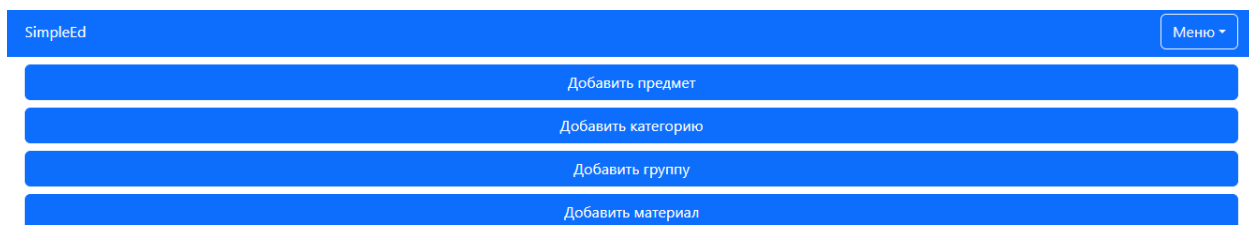


Рисунок 33 – Панель администратора

При нажатии на кнопки открываются модальные окна, аналогичные рассмотренному выше.

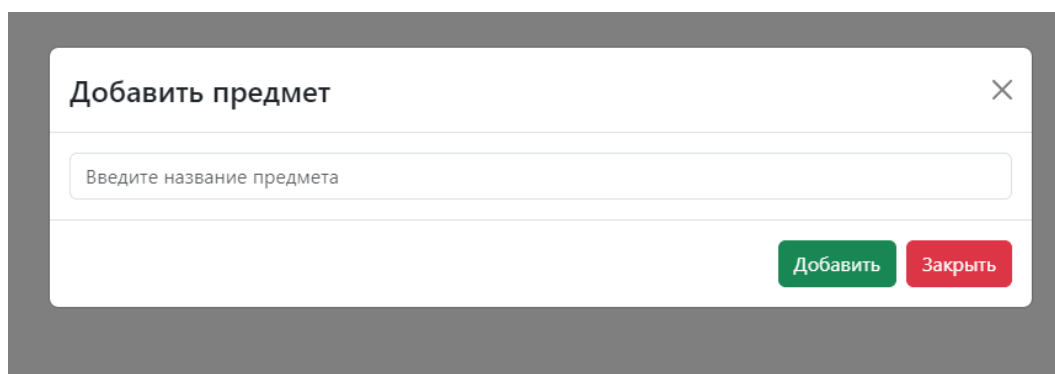


Рисунок 34 – Модальное окно «Добавить предмет»

3.6 Руководство по работе с информационной системой

Информационная система разработана для решения задачи автоматизации системы обучения. В рамках системы заданы три типа пользователей: незарегистрированный посетитель, авторизованный пользователь и администратор. В соответствии с требованием технического задания (п.3.1), каждый из пользователей обладает определенными возможностями по взаимодействию с системой, в соответствии с назначенной ролью.

Администратору доступен полный функционал системы. В перечень его возможностей входит:

- добавление категорий материалов;
- добавление учебных предметов;
- добавление возрастных групп;
- добавление/редактирование/удаление методических материалов;

- скачивание материалов;

Возможности авторизованного пользователя включают:

- добавление/редактирование/удаление методических материалов;
- скачивание материалов.

Возможности неавторизованного пользователя ограничены просмотром материалов и поиском материалов.

Рассмотрим возможности подробнее. При успешной авторизации (рис.25) пользователь получает доступ к своим возможностям. Пользователю, который является администратором, доступно добавление различных новых категорий (рис.32). Для добавления категорий необходимо нажать на кнопку соответствующей категории, после заполнить поле в появившемся модальном окне и нажать кнопку “Добавить”.

Следующие возможности являются доступными пользователям с ролями “Администратор” и “Авторизованный пользователь”. Они могут свободно просматривать и скачивать методические материалы, добавлять понравившиеся материалы в избранное (рис.30), а также добавлять собственные (рис.29). Для скачивания материалов с сайта, на главной странице необходимо кликнуть по карточке, интересующего материала, после на странице материала нажать по кнопке “Скачать”. Добавленные материалы можно редактировать и удалять (рис.31). Для добавления материалов в избранное, пользователь должен нажать по кнопке с иконкой сердца, после чего материал можно будет найти во вкладке “Избранное”. Для создания нового материала, пользователю необходимо кликнуть по кнопке “Добавить материал”. Далее, в появившемся модальном окне заполнить поля указанные поля и прикрепить файл с материалом в формате pdf. Чтобы удалить материал с сайта, пользователь должен нажать на соответствующую кнопку в разделе “Мои материалы”, после чего материал будет полностью удален с сайта.

Возможности неавторизованного пользователя ограничены просмотром и поиском материалов на сайте. Для поиска материалов необходимо указать название материала в строке поиска, после чего на странице автоматически

отобразятся найденные материалы. Описанные возможности соответствуют требованиям технического задания, описанным в пункте 2.1 “Назначение системы”.

Выводы по третьей главе

1. Выбранные средства программной реализации позволили выполнить программную разработку информационной системы в соответствии с требованиями технического задания.
2. Продемонстрирован процесс подключения и разработки базы данных на основе моделей, рассмотренных ранее.
3. Рассмотрена реализации бизнес-логики сущности ПОЛЬЗОВАТЕЛЬ, сформулированной в первой части выпускной квалификационной работы.
4. Описан процесс реализации бизнес-логики сущности МАТЕРИАЛ и рассмотрены конкретные функции в соответствии с бизнес-процессами.
5. Проведено тестирование разработанной информационной системы.
6. Реализовано руководство по работе с реализованной информационной системой.

В соответствии с описанными выше пунктами была выполнена программная реализация информационной системы и устранены недостатки существующих решений, рассмотренных в рамках SWOT анализа.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы была разработана информационная система «Методическая копилка учителя-предметника».

Система разработана в соответствии с техническим заданием.

Разработка информационной системы позволила закрепить навыки, полученные в процессе обучения в высшем учебном заведении, а именно:

1. Подробно рассмотрены этапы создания современных информационных систем.
2. Применены навыки проектирования архитектуры и моделирования базы данных, в соответствии с выбранной предметной областью.
3. Рассмотрены современные программные инструменты и методологии реализации информационных систем.
4. Получен опыт разработки и тестирования полноценного fullstack приложения.

В ходе выполнения выпускной квалификационной работы были решены все поставленные ранее задачи, достигнута цель.

Реализованная система в дальнейшем может быть усовершенствована, так как использованная в ходе реализации архитектура и база данных это позволяют.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. ГОСТ Р 7.0.97–2016 Система стандартов по информации, библиотечному и издательскому делу. Организационно–методические документы. Требования к содержанию и оформлению.
2. ГОСТ Р 53620–2009 “Информационные технологии. Обучение, образование и подготовка. Общие положения”.
3. ГОСТ 34.003–90 “Информационная технология. Комплекс стандартов на автоматизированные системы. Термины и определения.”
4. ГОСТ 2.105–95 “Единая система конструкторской документации. Общие требования к текстовым документам”.
5. ГОСТ Р ИСО 9241–310–2014 “Эргономика взаимодействия человек–система. Принципы и требования”.
6. ГОСТ Р 1.5–2012 “Стандарты национальные. Правила построения, изложения, оформления и обозначения”.
7. ГОСТ 15.013–86 “Система разработки и постановки продукции на производство. Продукция производственно–технического назначения”.
8. Соловьева М. А. Электронные учебные материалы. Виды электронных учебных материалов – ГПОУ ТО 2017 – 6 с.
9. Цуканова О. А. Методология и инструментарий моделирования бизнес – процессов: учебное пособие – СПб.: Университет ИТМО, 2015 – 100 с.
10. Алтынбаев Р. Р. Методология функционального моделирования IDEF0 – ГОССТАНДАРТ РОССИИ 2000 – 75 с.
11. Роббинс Дж. Веб–дизайн для начинающих. HTML, CSS, JavaScript и веб–графика. — 5–е изд.; пер. с англ. — СПб.: БХВ–Петербург, 2021. — 956 с.: ил.
12. PostgreSQL: The World's Most Advanced Open Source Relational Database // URL: <https://www.postgresql.org/>
13. Браун И. Веб–разработка с применением Node и Express. Полноценное использование стека JavaScript — Санкт–Петербург: Питер, 2017. — 336 с.
14. React The library for web and native user interfaces // URL: <https://react.dev/>

15. Руководство по Sequelize // URL: <https://my-js.org/docs/guide/sequelize/>
16. Пластун Е. Г., Пустыльникова Г. В., Пахомова О. Н. Классификация и виды методической продукции: словарь— справочник / сост. —Оренбург: ООДТДМ, 2016 – 24 с.
17. ГОСТ 2.105–95 Общие требования к текстовым документам.
18. ГОСТ 34.320.96 Информационная технология. Система стандартов по базам данных. Концепции и терминология для концептуальной схемы и информационной базы данных.
19. ГОСТ Р ИСО/МЭК 9075–93 Язык баз данных SQL с расширением целостности.
20. ГОСТ Р 52653–2006 Информационно–коммуникационные технологии в образовании. Термины и определения.
21. Сухомлинов А. И. Инфологическое моделирование: учебно–методическое пособие: для студентов, обучающихся по направлению 09.03.03 «Прикладная информатика» / А.И. Сухомлинов. – Владивосток: Издательство Дальневосточного федерального университета, 2021 – [36 с.]. – URL: <https://www.dvfu.ru/science/publishing-activities/catalogue-of-books-fefu/>. – Дата публикации: 09.04.2021. – Текст. Изображения: электронные.
22. Difference between PERN and MERN stack // URL: <https://www.geeksforgeeks.org/difference-between-pern-and-mern-stack/>
23. Чиннатамби К. Изучаем React. — СПб.: «Питер», 2019. — С. 368
24. Беляева И. В. Архитектура информационных системы учебное пособие / И. В. Беляева. — Ульяновск: Известия Ульяновского государственного технического университета, 2005.
25. Звонарёв С. В. Основы математического моделирования / С. В. Звонарёв. — Екатеринбург: Уральский федеральный университет им. Б. Н. Ельцина, 2019.
26. Кара–Ушанов В. Ю. Модель "сущность–связь" / В. Ю. Кара–Ушанов. — Екатеринбург: Уральский федеральный университет им. Б. Н. Ельцина, 2017.

27. Кара–Ушанов В. Ю. Методология функционального моделирования IDEF0 / В. Ю. Кара–Ушанов. — М: Научно–исследовательский Центр CALS–технологий «Прикладная Логистика», 2000.
28. Максимова Н. Н. Математическое моделирование. Учебно–методическое пособие / Н. Н. Максимова. — Благовещенск : АмГУ, 2019.

ПРИЛОЖЕНИЕ

Техническое задание на разработку информационной системы «Методическая копилка учителя–предметника»

1. ОБЩИЕ ПОЛОЖЕНИЯ

1.1. Полное наименование системы и ее условное обозначение

Полное наименование – Информационная система «Методическая копилка учителя–предметника».

Сокращенное наименование – SimpleEd.

1.2. Наименование организации–заказчика

ТГПУ им. Л.Н. Толстого

Тульский государственный педагогический университет имени Льва Николаевича Толстого.

1.3. Наименование организации–участников работ

Исполнитель: студент 4 курса группы 1521601 Зименков Алексей Романович.

1.4. Определения, обозначения и сокращения

Система – разрабатываемая система, сокращенно ИС.

Материал – электронный документ в ИС, фиксирующий всю информацию о методическом материале.

Карточка материала – диалоговая форма материала в ИС.

Уровень доступа – классификатор, используется в электронном документе для ограничения доступа пользователей.

Процесс (обработки документа) – механизм, обеспечивающий обработку документов. Процессы бывают элементарные: Регистрация, Редактирование, Удаление, Хранение.

Файл материала: Хранилище информации определённого типа (содержащее текстовую и/или графическую информацию).

Электронный документ: Документированная информация, представленная в электронной форме, то есть в виде, пригодном для

восприятия человеком с использованием электронных вычислительных машин, а также для передачи по информационно–телекоммуникационным сетям или обработки в информационных системах.

2. НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ СИСТЕМЫ

2.1. Назначение системы

Создаваемая автоматизированная система предназначена для автоматизации следующих функциональных контуров:

- 2.1.1. регистрация материалов;
- 2.1.2. редактирование добавленных материалов – изменение уже внесенной информации в карточку материала;
- 2.1.3. удаление материала и всей информации, связанной с ним;
- 2.1.4. разграничение доступа к работе с материалом;
- 2.1.5. поиск зарегистрированных материалов;
- 2.1.6. скачивание материалов из системы.
- 2.1.7. добавление интересующих материалов в отдельную вкладку, для облегчения поиска;

2.2. Цели создания системы

Целью разработки информационной системы является автоматизация процесса обучения; обеспечение информатизации, уменьшения времени и трудозатрат на подготовку и поиск необходимых материалов.

Система должна решать следующие задачи:

- обеспечить регистрацию и хранение электронных копий входящих материалов;
- организовать надежное, оперативное и долговременное хранение материалов, в рамках веб–приложения с возможностью поиска по различным критериям;
- обеспечить безопасность и доступность сервиса;

3. ТРЕБОВАНИЯ К СИСТЕМЕ

3.1. Требования к структуре и функционирования системы

Система должна обеспечивать возможность управления доступом к документам.

Уровень детализации правил разграничения доступа должен позволять определить права доступа для каждого конкретного пользователя.

Система должна предоставлять пользователю возможность полноценной работы с использованием технологии WEB–доступа (работа с системой, используя возможности интернет–браузеров: Google Chrome; Internet Explorer; Yandex browser; Mozilla Firefox, независимо от версии). Система не должна конфликтовать с известными средствами организации шифрования и частного доступа к данным. Информационная система должна обеспечивать возможность подключаться к интерфейсу программы через WEB приложение (браузер) по протоколу http в пределах локально вычислительной сети и https, как в пределах локально вычислительной сети, так и вне ее, без участия дополнительных плагинов и модулей.

Система должна обеспечивать сохранность и защиту персональных данных, в соответствии с Российским законодательством.

Интерфейс должен быть простым в освоении, удобным, учитывать предпочтения основных пользователей. Во время работы пользователям должны быть доступны только необходимые функции согласно их ролям в рабочих процессах.

3.2. Требования к надежности

Система должна поддерживать ежедневное круглосуточное функционирование. Допускается временная приостановка работы системы для проведения профилактических работ программно–аппаратного обеспечения сервера, на котором располагается система.

Необходимым условием функционирования системы является условие функционирования аппаратной части и сервера, на котором размещено приложение.

Система в целом должна сохранять работоспособность при

некорректных действиях конечных пользователей.

Система должна обеспечивать восстановление работоспособности при появлении сбоев, аварий и отказов, возникающих на сервере и сетевом аппаратном обеспечении.

3.3. Требования к эргономике и технической эстетике

Интерфейс системы должен быть прост, нагляден, интуитивно понятен и легок в освоении.

Система должна позволять просматривать всю основную информацию (список материалов, материал, файл материала).

Интерфейс системы должен отвечать следующим требованиям:

- Единый унифицированный интерфейс, реализованный на русском языке.
- Однозначность в наименовании пунктов меню.
- Сигнализацию об ошибках системы или выполнении ошибочных действий пользователем в виде индикаций на экране с информацией об ошибке и/или подсказкой о дальнейших действиях на русском языке;
- Наличие вспомогательной индикации при выполнении длительных процессов.

Цветовое решение интерфейса должно быть выдержано в спокойных тонах, не вызывающих утомление зрения.

3.4. Требования к защите информации от несанкционированного доступа

Система разграничения доступа к информации должна предусматривать назначение групповых прав доступа к данным.

Информационная безопасность должна обеспечиваться средствами нескольких уровней:

- Средствами базы данных;
- Средствами системы;

3.5. Требования к информационной программной совместимости

Специальных требований не предъявляется.

3.6. Требования к транспортированию и хранению

Специальных требований не предъявляется.

4. ТРЕБОВАНИЯ К ПРОГРАММНОЙ ДОКУМЕНТАЦИИ

Предварительный состав программной документации:

- техническое задание (включает описание применения);
- программа и методика испытаний;
- руководство программиста;
- ведомость эксплуатационных документов;
- формуляр.

Вся документация должна быть подготовлена и передана как в печатном,

так и в электронном виде (в формате Microsoft Word).

5. Стадии и этапы разработки

Разработка должна быть проведена в две стадии:

- 1) техническое задание;
- 2) технический (и рабочий) проекты;
- 3) внедрение.

На стадии «Техническое задание» должен быть выполнен этап разработки, согласования и утверждения настоящего технического задания.

На стадии «Технический (и рабочий) проект» должны быть выполнены

перечисленные ниже этапы работ:

- разработка программы;
- разработка программной документации;
- испытания программы.

Содержание работ по этапам:

- На этапе разработки технического задания должны быть выполнены перечисленные ниже работы:
- постановка задачи;
- определение и уточнение требований к техническим средствам;
- определение требований к программе;
- определение стадий, этапов и сроков разработки программы и документации на нее;
- согласование и утверждение технического задания.

На этапе разработки программы должна быть выполнена работа по программированию (кодированию) и отладке программы.

На этапе разработки программной документации должна быть выполнена разработка программных документов в соответствии с требованиями ГОСТ 19.101-77.

На этапе испытаний программы должны быть выполнены перечисленные ниже виды работ:

- разработка, согласование и утверждение порядка и методики испытаний;
- проведение приемо-сдаточных испытаний;
- корректировка программы и программной документации по результатам испытаний.

5. Порядок контроля и приемки

Контроль и приемка разработанного программного обеспечения будут проводиться следующим образом:

1. Проведение внутренних тестов на соответствие требованиям.
2. Предоставление программного обеспечения для тестирования заказчику.
3. Анализ результатов тестирования и внесение необходимых исправлений.
4. Подписание акта приемки программного обеспечения заказчиком.