Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ А. С. Кузнецов
подпись инициалы, фамилия

«_____» _____ 2019 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.04 Программная инженерия
код и наименование специальности

Автоматизация процессов приема и рецензирования для веб-платформы
Тема

управления и публикации научных статей

Научный руководитель	_____	доцент, канд. техн. наук	А. В. Хныкин
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		О. Д. Белецкая
	подпись, дата		инициалы, фамилия
Нормоконтролер	_____	доцент, канд. техн. наук	О. А. Антамошкин
	подпись, дата	должность, ученая степень	инициалы, фамилия

Красноярск 2019

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ А. С. Кузнецов
подпись инициалы, фамилия

«_____» _____ 2019 г.

ЗАДАНИЕ
НА ВЫПУСКНУЮ КВАЛИФИКАЦИОННУЮ РАБОТУ
в форме бакалаврской работы

Студенту Белецкой Ольге Денисовне
фамилия, имя, отчество

Группа КИ15-16Б Направление (специальность) 09.03.04
номер код

Программная инженерия
наименование

Тема выпускной квалификационной работы

Автоматизация процессов приема и рецензирования для веб-платформы управления и публикации научных статей

Утверждена приказом по университету № 6264/с от 13.05.2019

Руководитель ВКР А. В. Хныкин, доцент, канд. техн. наук., ИКИТ СФУ
инициалы, фамилия, должность, ученое звание и место работы

Исходные данные для ВКР описание предметной области

Перечень разделов ВКР введение, анализ предметной области, планирование и проектирование, разработка веб-приложения, описание работы программного продукта, заключение, список сокращений, список использованных источников

Перечень графического материала 40 рисунков

Руководитель ВКР _____ А. В. Хныкин
подпись инициалы и фамилия

Задание принял к исполнению _____ О. Д. Белецкая
подпись инициалы и фамилия студента

« ___ » _____ 2019 г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Автоматизация процессов приема и рецензирования для веб-платформы управления и публикации научных статей» содержит 53 страницы текстового документа, 25 использованных источников, 40 иллюстраций, 4 таблицы.

ВЕБ-ПРИЛОЖЕНИЕ, АВТОМАТИЗАЦИЯ, НАУЧНАЯ СТАТЬЯ, ПУБЛИКАЦИЯ В НАУЧНОМ ЖУРНАЛЕ, БАЗА ДАННЫХ, УПРАВЛЕНИЕ ПРОЕКТОМ, МЕТОДОЛОГИЯ РАЗРАБОТКИ.

Целью данной выпускной квалификационной работы является автоматизация процессов приема и рецензирования научных статей для упрощения подачи заявки на публикацию, предоставления метаданных статьи, взаимодействия автора и редакции журнала.

Для достижения поставленной цели были решены следующие задачи:

- определить технологический стек проекта;
- спроектировать систему приема и рецензирования научных статей;
- спроектировать базу данных;
- спроектировать интерфейс системы;
- разработать спроектированную систему.

В результате исследования предметной области были изучены и проанализированы аналогичные продукты, вследствие чего были определены требования к продукту, в которых упор сделан на соответствие разработанной системы российскому законодательству. Также была определена инфраструктура научного журнала, используемая при составлении бизнес-логики веб-приложения.

В итоге была разработана система, обеспечивающая следующие возможности:

- подача автором заявки на публикацию с последующим отслеживанием и управлением созданной заявкой;
- управление полученными заявками на публикации, а также выпусками журнала со стороны редакции журнала;
- заполнение рецензентом шаблона рецензии и загрузка подписанной версии рецензии в систему, а также управление и просмотр заявок на рецензирование.

Разработанный продукт призван автоматизировать и тем самым улучшить взаимодействие между авторами научных статей и редакцией журнала, за счет предоставления сервиса, в котором можно получить доступ к необходимой информации в удобном виде в любое время.

СОДЕРЖАНИЕ

Введение 4
1 Анализ предметной области 5
1.1 Анализ существующих решений 6
1.2 Выбор инструментов разработки 7
1.3 Определение требований к программному продукту 10
1.4 Выводы 12
2 Планирование и проектирование 13
2.1 Планирование 13
2.1.1 Выбор методологии разработки 13
2.1.2 Управление проектом 14
2.2 Проектирование 15
2.2.1 Проектирование архитектуры информационной системы 15
2.2.2 Проектирование базы данных 17
2.2.3 Проектирование интерфейса веб-приложения 21
2.3 Выводы 23
3 Разработка веб-приложения 24
3.1 Серверная часть веб-приложения 25
3.2 Клиентская часть веб-приложения 29
3.3 Выводы 31
4 Описание работы программного продукта 33
4.1 Функционал автора 33
4.2 Функционал ответственного секретаря 37
4.3 Функционал рецензента 44
4.4 Взаимодействие пользователя с системой 45
4.5 Выводы 49
Заключение 50
Список сокращений 51
Список использованных источников 52

ВВЕДЕНИЕ

В наше время практика публикации своих трудов в научных журналах очень распространена. По данным научной электронной библиотеки eLIBRARY.RU число наименований журналов в данной системе составляет 65481, а общее число публикаций – 29574209 [1]. Причем число опубликованных статей растет не только с каждым годом, но и с каждым месяцем. Данная тенденция объясняется несколькими причинами.

Во-первых, желанием автора поделиться результатами своих исследований, а также своими открытиями. После достижения какого-либо важного результата необходимо донести его как можно большему количеству людей и распространить новую информацию в научных кругах. Это необходимо для того, чтобы результат исследования нашел применение в практических целях. Во-вторых, причиной роста числа публикаций является активный интерес со стороны научного сообщества, ведь работы, изданные в научных журналах, всегда находят своего читателя. И наконец, одни исследования приводят к появлению новых, которые основываются на уже существующих работах. Таким образом, мы видим, что сфера ведения и управления научными журналами является актуальной.

Большинство современных журналов имеют сайты, на которых представлена информация об изданных номерах и статьях. Однако такие сайты чаще всего не обеспечивают автоматизацию многих важных процессов управления научным журналом. Тем временем процессы принятия и публикации статей являются сложными, многосторонними и включают в себя множество сторон, таких так авторы статей, редакторы журналов, рецензенты, редакционная коллегия, литературные редакторы, верстальщики журналов, издательство и читатели научного издания. Более того, путь от подачи статьи к ее публикации не линейный, и имеет множество разветвлений. Это приводит нас к мысли о том, что данные процессы можно автоматизировать, тем самым упростив пользователям предполагаемой системы многие шаги, начиная от подачи заявки на публикацию статьи, заканчивая публикацией изданного выпуска на сайте журнала.

Целью данной выпускной квалификационной работы является автоматизация процессов приема и рецензирования научных статей для упрощения процесса подачи заявки на публикацию, предоставления метаданных статьи, взаимодействия автора и редакции журнала.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- определить технологический стек проекта;
- спроектировать систему приема и рецензирования научных статей;
- спроектировать базу данных;
- спроектировать интерфейс системы;
- разработать спроектированную систему.

1 Анализ предметной области

Научный журнал – это периодическое издание (печатное или электронное), являющееся одним из источников научной информации и средством научной коммуникации [2]. Считается, что первый научный журнал появился в Париже в 1665 году, и издавался он под названием «Журналь де саван» (фр. Journal des savants) [3].

Любое научное издание имеет свою инфраструктуру, однако общая система управления жизненным циклом публикации остается одной (рисунок 1).

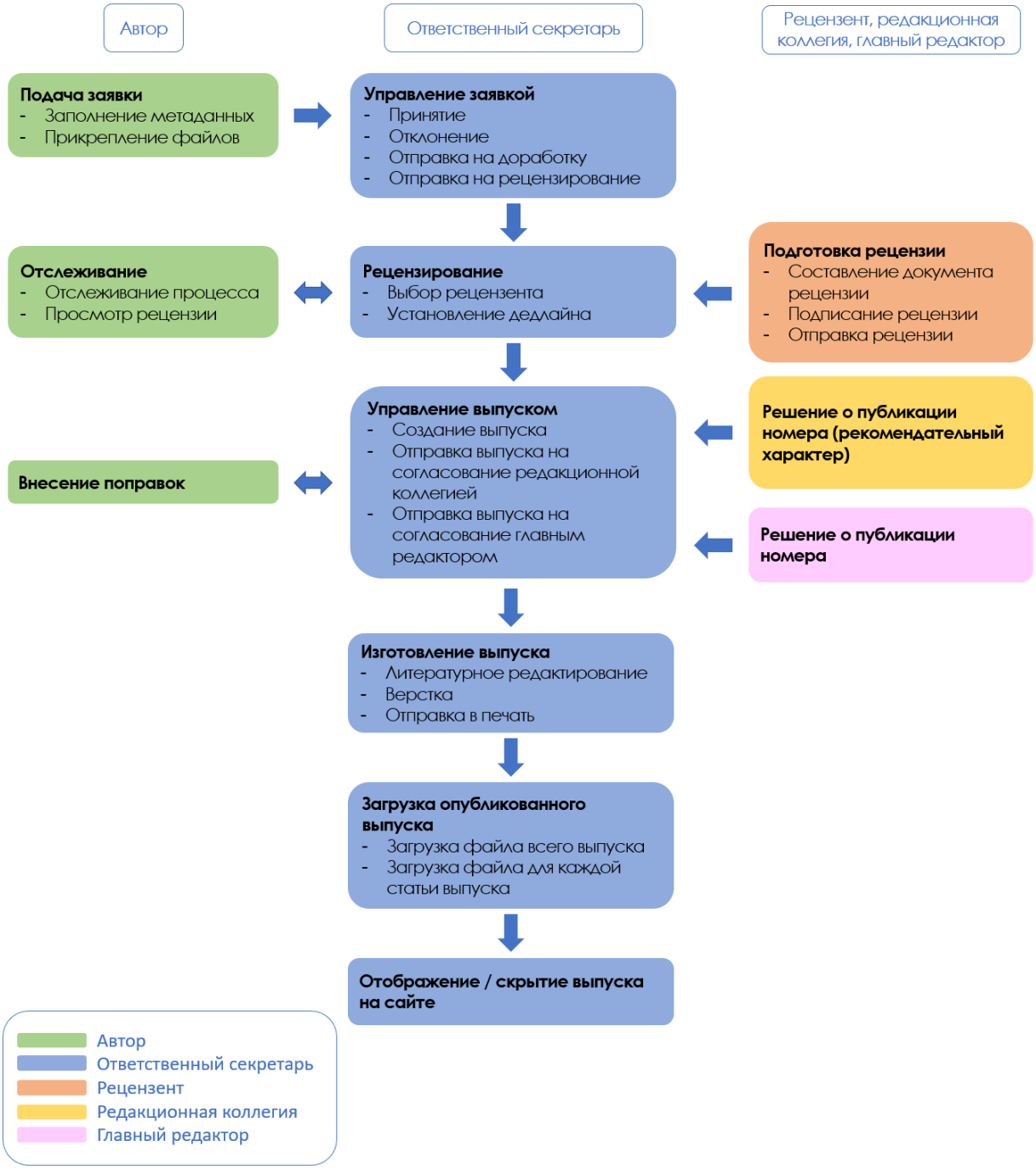

Рисунок 1 – Инфраструктура научного журнала

К сайту журнала также предъявляются особые требования [4], например, такие как:

- на сайте не должно быть размещено информации, не касающейся издаваемого журнала, например, рекламы;
- на сайте должны быть размещены сведения о редакционной коллегии, публикационной этике и описание редакционной политики;
- наличие информации о текущем выпуске и архиве выпусков;
- наличие быстрого и удобного поиска.

Так как научные журналы являются распространителями информации, то существует необходимость в изучении закона РФ от 27.12.1991 N 2124-1 (ред. от 01.05.2019) «О средствах массовой информации» [5].

1.1 Анализ существующих решений

При исследовании предметной области были найдены две крупные системы управления научными журналами: Elpub [6] и Open Journal Systems [7].

Elpub – это платформа комплексной поддержки и сопровождения научного журнала, которая была создана при участии Министерства науки и высшего образования [8]. Данная платформа предоставляет следующие возможности:

- двуязычный сайт журнала;
- шаблоны ключевых базовых текстов;
- разметка и размещение свежих выпусков;
- создание архива журнала;
- работа с системой антиплагиат и другие.

Elpub предлагает несколько вариантов верстки сайта, некоторые из которых не адаптированы под мобильные устройства.

Платформа соответствует всем требованиям, пожеланиям и условиям международных баз любого уровня и ВАК, она является многофункциональной и имеет консультации и поддержку, однако является платной, и стоимость сервиса варьируется от 90 до 150 тысяч рублей в год для одного наименования журнала, имеющего до 30 статей в выпуске.

Open Journal Systems (OJS) – открытое программное обеспечение для организации рецензируемых научных изданий, разработанное канадским некоммерческим исследовательским проектом Public Knowledge Project (PKP).

Данная система поддерживает следующие процессы управления научным журналом:

- публикация статей;
- прием и рецензирование статей;
- каталогизация статей.

OJS предоставляет базовую верстку сайта, которую при желании можно изменить, также, как и различные дизайнерские решения (цветовая палитра, шрифты и другое). Предоставленная базовая верстка частично адаптирована под мобильные устройства.

Устаревшую версию данной системы использует редакция Журнала Сибирского федерального университета (СФУ). Данный факт является одной из основных причин разработки веб-платформы для управления научным журналом.

Минусами используемой СФУ системой являются:

- устаревший дизайн;
- неполная локализация;
- неудобный личный кабинет автора;
- неактуальная форма подачи заявки на публикацию статей;
- отсутствие на сайте обязательной информации о правилах публикации, публикационной этики и редакционной коллегии;
- отсутствие адаптации под мобильные устройства.

1.2 Выбор инструментов разработки

Стек технологий – это набор технологий и инструментов, на основе которых разрабатывается программный продукт.

Так как разрабатываемый продукт является веб-приложением, то обязательными элементами технологического стека являются язык гипертекстовой разметки HTML, формальный язык CSS, необходимый для описания стилей веб-страниц, и язык программирования JavaScript для написания клиентской части приложения.

В качестве веб-сервера была выбрана программная платформа Node.js, обладающая определенным рядом положительных сторон:

- один язык программирования (JavaScript) для клиентской и серверной частей приложения;
- наличие большого количества средств разработки и других инструментов;
- современное и стремительно развивающееся решение;
- в основе лежит асинхронное программирование.

Данное технологическое решение также обладает минусами, среди которых:

- возможны проблемы с хостингом и развертыванием (не все площадки подходят для Node.js);
- плохо работает с тяжелыми и большими вычислениями.

Факт того, что мы можем использовать JavaScript для написания также серверной части веб-приложения, является наиболее весомым, так как JavaScript – самый популярный язык написания исполняемых сценариев на стороне клиента, а это означает наличие крупного и развитого сообщества программистов и большого количества готовых решений. Согласно данным сайта Stack Overflow, площадки для обсуждения и решения вопросов программирования, на 2019 год JavaScript является первым по популярности в рейтинге языков программирования (рисунок 2) [9].

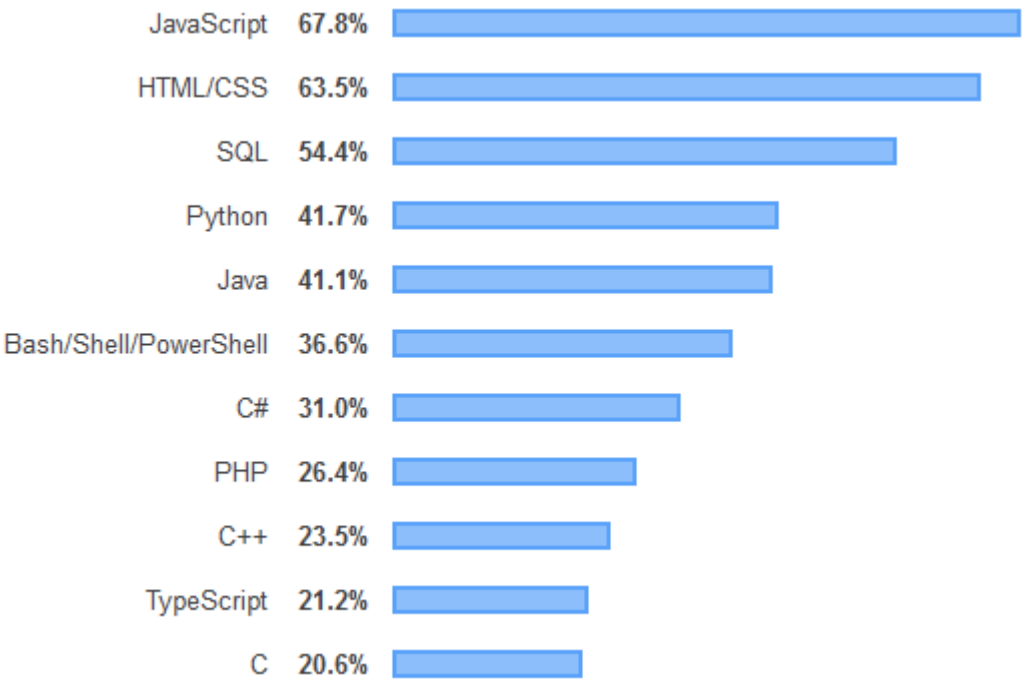

Рисунок 2 – Рейтинг технологий по популярности

Безусловно, у JavaScript есть свои минусы:

- отсутствие типизации данных (контроля типов данных), что на данный момент с легкостью решается применением дополнительных инструментов;
- чувствительность к неуклюжему программированию [10].

С последним помогает справиться Node.js, предоставляя различные фреймворки, библиотеки и другие решения, позволяющие не только ускорить разработку, но и писать хороший программный код.

Одним из таких фреймворков является Express, позиционирующий себя как минималистичное и гибкое решение для построения мобильных и веб-приложений. Данный фреймворк прост в использовании и хорошо масштабируем, что предоставляет разработчику свободу выбора в инструментарии разработки. Он предоставляет минимальный набор необходимых функций, чтобы в дальнейшем разработчик мог добавить то, что ему нужно. Также Express легко настроить на работу с базами данных.

Зачастую с веб-приложениями, реализующимися на Node.js, используется документоориентированная система управления базами данных MongoDB, однако в данном случае была необходимость в использовании реляционной модели данных, что сузило круг возможных решений. Также условием выбора СУБД было наличие триггеров и хранимых процедур. Хранимые процедуры удобны в использовании и более быстродействены, так как хранятся на сервере базы данных. Триггер же вызывается каждый раз при добавлении, редактировании или удалении строки таблицы.

PostrgeSQL – наиболее полнофункциональная, объектно-реляционная свободно распространяемая СУБД [11], которая соответствует стандартам ANSI SQL-92 и SQL-99. Классифицируется как кроссплатформенная, поддерживает большое количество типов данных, в том числе композитных

типов, массивов и сложных типов, таких как xml, json и другие [12].

Несмотря на поддержку PostrgeSQL хранимых процедур, существует необходимость описывать работу с базой данных также на стороне сервера. Для упрощения взаимодействия между сервером веб-приложения и сервером базы данных существует технология ORM (Object-Relational Mapping), которая связывает базу данных с объектно-ориентированной концепцией, то есть позволяет обращаться и работать с таблицами базы данных как с объектами классов. В данной работе была выбрана ORM-библиотека Sequelize.

При разработке веб-приложения важно тщательно выбирать не только инструменты для работы с серверной частью приложения, но также и для работы с клиентской частью. В настоящее время существует множество различных библиотек и фреймворков для написания клиенткой части. Одним из таких примеров, стремительно набирающих популярность, является React.js.

Основные концепции React – это компоненты, жизненный цикл и состояние. Компоненты – это основополагающая концепция React. Ее смысл заключается в возможности повторного использования компонентов как в рамках одного проекта, так и в других проектах. Компоненты представляют из себя функции или классы JavaScript. Плюсом компонентов является хранение некоторой информации о его текущем состоянии, посредством использования Sate. Изменяя состояние компонента, мы можем изменять и обновлять пользовательский интерфейс, используя при этом события, а говоря точнее – обработчики событий. Удобством в использовании компонентов является обмен данными между ними.

React имеет следующие преимущества, значимые для данной работы:

- легкость в изучении и понимании;
- возможность повторного использования кода;
- возможность рендеринга веб-страниц со стороны сервера.

При SSR (server-side rendering) – рендеринге со стороны сервера – пользователю в ответ на запрос возвращается сгенерированный на сервере HTML. Для упрощения конфигурации рендеринга со стороны сервера используется фреймворк Next.js.

Из плюсов SSR можно отметить:

- обеспечивает хорошую поисковую оптимизацию;
- первичная содержательная отрисовка веб-страницы выполняется быстрее.

На рисунке 3 представлена упрощенная диаграмма жизненного цикла веб-страницы при использовании SSR. Согласно данной диаграмме, пользователь сначала получает отрисованное HTML наполнение страницы. Так как библиотека React.js рассчитана на интерактивность, а посылаемый сервером HTML код является статичным, то после его получения загружаются JavaScript файлы, позволяющие реализовать интерактивность на странице.

Минусом данной технологии является тот факт, что после загрузки HTML кода клиентом, существует задержка между первой отрисовкой и возможностью взаимодействовать со страницей.

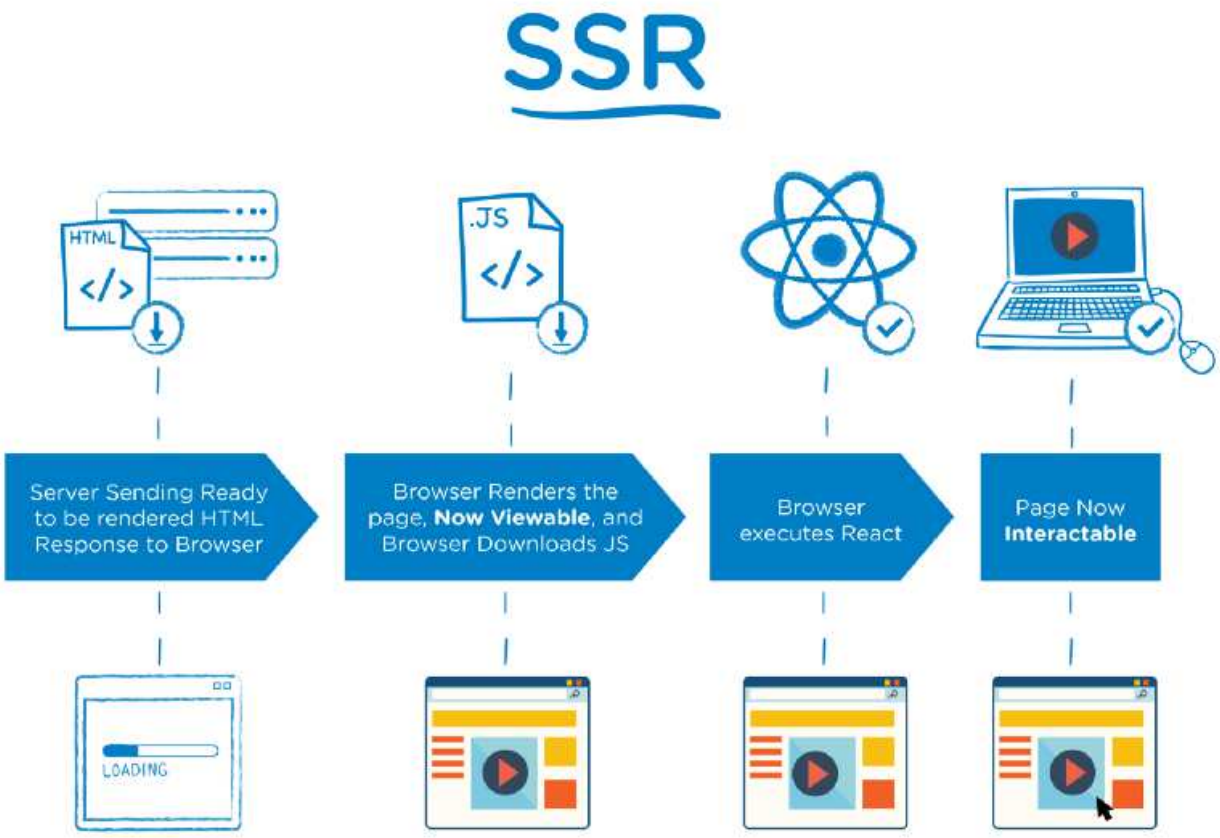

Рисунок 3 – Упрощенная диаграмма жизненного цикла веб-страницы [13]

Последним и немаловажным пунктом является создание дизайна веб-приложения, для чего есть огромное количество фреймворков и библиотек. Для разрабатываемого веб-приложения был выбран стиль дизайна Material Design и фреймворк для React – Material UI.

1.3 Определение требований к программному продукту

В результате сравнения аналогов в пункте 1.2 был составлен ряд требований к разрабатываемому продукту.

Во-первых, процесс оформления заявки на публикацию статьи должен быть адаптирован к российским стандартам:

- все метаданные должны быть представлены на русском и английском языках;
- в список обязательных файлов, прикрепленных к заявке, должны входить текст статьи, «Заключение о возможности открытого опубликования» и «Идентификационное заключение» [5; 14; 15].

Во-вторых, дизайн сайта должен соответствовать следующим требованиям:

- интуитивно понятный интерфейс;
- нейтральная цветовая палитра;
- наличие подсказок и уточняющих надписей;

- весь функционал, предоставляемый веб-приложением, должен быть очевидным, то есть назначение каждого элемента и блока на сайте должно быть понятно из названия или внешнего вида;

- удобное навигационное меню.

В-третьих, информационное наполнение сайта должно быть исчерпывающим, не содержащим каких-либо грамматических ошибок и должно включать в себя следующую информацию:

- правила и порядок опубликования статей;

- порядок рецензирования статей;

- правила оформления статей;

- публикационную этику.

И, наконец, должен быть реализован приведенный ниже функционал сайта.

Для пользователя с ролью «Автор»:

- просмотр текущих заявок на публикацию (находящихся на рассмотрении ответственным секретарем, на рецензировании, отправленных на доработку, рекомендованных к публикации и рекомендованных к публикации в текущем выпуске);

- просмотр архива заявок (отклоненных и опубликованных);

- создание и отправка заявки на публикацию (соглашение с политикой научного журнала, внесение метаданных и прикрепление файлов);

- изменение заявки на публикацию (в случае, если заявка отправлена на доработку);

- просмотр истории заявки на публикации (какое действие произошло с заявкой и когда, просмотр рецензии);

- отправление сообщений с прикрепляемыми файлами ответственному секретарю (по каждой заявке создается отдельный диалог).

Для пользователя с ролью «Ответственный секретарь»:

- просмотр текущих заявок на публикацию;

- просмотр архива заявок;

- управление заявкой (принятие статьи к публикации, отклонение заявки, отправка статьи на доработку, отправка статьи на рецензирование);

- просмотр истории заявки на публикацию;

- отправление рецензии автору;

- отправление сообщений с прикрепляемыми файлами ответственному секретарю (по каждой заявке создается отдельный диалог);

- просмотр рецензий для каждой заявки;

- создание нового тома;

- просмотр выпусков журнала;

- создание нового выпуска;

- загрузка опубликованного выпуска в систему;

- управление выпуском (отправка нового выпуска на рассмотрение редакционной коллегией, отправка рассмотренного редакционной коллегией выпуска на рассмотрение главным редактором);

- создание шаблонов для рецензирования.

Для пользователя с ролью «Рецензент»:

- просмотр заявок на рецензирование и принятых к работе заявок;
- просмотр отклоненных и завершенных заявок на рецензирование;
- заполнение формы шаблона рецензии;
- загрузка подписанной рецензии.

1.4 Выводы

Целью данного раздела являлось изучение предметной области, в результате анализа которого была составлена схема, описывающая инфраструктуру научного журнала. Описанная инфраструктура будет использоваться при создании веб-приложения, учитывать ее ответвления и условия. Также анализ предметной области помог выявить требования к сайту, позиционирующего себя как сайт научного журнала. Это немаловажный пункт, так как к подобным информационным ресурсам предъявляются особенные требования.

Следующий этап – выбор инструментов разработки. Этот шаг был очень важным и требовал особого внимания, так как инструменты определяют дальнейшую стратегию разработки. В итоге был определен следующий технологический стек:

- язык программирования для клиентской и серверной частей JavaScript;
- программная платформа для реализации серверной части Node.js;
- фреймворк веб-приложений для Node.js – Express.js;
- СУБД PostgreSQL;
- ORM-библиотека для Node.js – Sequelize;
- библиотека для разработки пользовательского интерфейса React.js;
- фреймворк для серверной визуализации Next.js;
- фреймворк для React.js – Material UI, воплощающий Material design.

Заключительным этапом работы стало определение функциональных требований к разрабатываемому продукту. Выявленные требования были основаны на проведенном сравнении уже существующих аналогичных программных продуктов, таких как Elpub и Open Journal Systems. Первая система не совсем корректна в данном сравнении, так как представляет собой сервис полной поддержки, сопровождения, продвижения и управления научным журналом, что, соответственно, сказывается на стоимости продукта. Вторая сравниваемая система является универсальной, что объясняет ее громоздкость и сложность во внедрении. Разрабатываемый продукт призван отвечать требованиям российских правил публикования научных работ.

2 Планирование и проектирование

2.1 Планирование

Важным аспектом управления проектом является грамотное распределение работ процесса разработки программного продукта так, чтобы разработка проекта успешно завершилась в установленные строки.

В начале работы над проектом был определен подробный перечень задач, которые необходимо выполнить для достижения поставленной цели:

- изучение предметной области;
- выбор и изучение инструментов разработки;
- проектирование архитектуры информационной системы;
- проектирование БД;
- проектирование интерфейса;
- разработка функционала для пользователя «Автор»;
- разработка функционала для пользователя «Ответственный секретарь»;
- разработка функционала для пользователя «Рецензент»;
- разработка клиентской части веб-приложения.

2.1.1 Выбор методологии разработки

Методология разработки ПО описывает подход к разработке программного продукта и способ организации коллективной разработки. Существует немалое количество таких подходов, каждый из которых учитывает различные аспекты и предназначен для конкретной ситуации. Для выбора методологии для данного проекта сравним некоторые из них.

Одна из самых старых моделей – водопадная, она считается традиционной и подразумевает строгое и последовательное выполнение всех этапов, то есть новый этап должен начаться после завершения предыдущего этапа. Данная модель проста в использовании и дедлайны проекта заранее определены. В тоже время, модель не очень гибкая и не позволяет вернуться на предыдущий этап.

Альтернативой водопадному подходу может стать спиральный подход. Однако в данном случае жизненный цикл разработки продукта имеет достаточно сложную организацию и требует большой внимательности, так как эта методология, в первую очередь, направлена на раннее выявление и уменьшение проектных рисков.

Существует также итеративный подход, решающий проблемы водопадной методологии. Он позволяет учитывать изменяющиеся требования, а также использовать опыт повторно, выделяя некоторые типовые решения. Однако в случае данного проекта, требования точно определены и необходимость в использовании данной методологии отсутствует.

Конечно, существуют более гибкие решения, говоря о которых, нельзя не упомянут об Agile – наборе методов и методологий, которые помогают команде

разработчиков эффективнее мыслить, работать и принимать решения [16]. Agile поддерживает открытое планирование, обсуждение дизайна и других задач всей командой.

Одним их популярных Agile подходов является Scrum [17], концепция которого заключается в делении процесса разработки на спринты, то есть итерации, каждая из которых строго ограничена по времени. Такая методология позволяет проводить регулярный контроль выполнения работ, однако предназначена в основном для больших проектов.

Другим Agile подходом является Канбан, в основе которого заложены следующие принципы [18]:

- визуализация процесса выполнения задач;
- установление ограничений на объем работ на каждом этапе;
- поощрение инициативы.

Канбан – это наглядная система разработки, и для достижения наглядности используются канбан-доски. Такая доска имеет минимум три колонки: «сделать» (to-do), «в процессе» (in progress) и «сделано» (done).

Канбан методология проста в использовании, наглядна, благодаря визуальному отображению информации в формате досок, а также позволяет привнести в разработку большую гибкость. Из минусов можно отметить отсутствие строгих дедлайнов и сложность применения к долгосрочным проектам. Для данного проекта эти минусы не существенны, поэтому методология Канбан выбрана для управления разработкой веб-приложения.

2.1.2 Управление проектом

Для управления работой над проектом был использован инструментарий, предлагаемый системой управления проектами Trello. Данная система используется для небольших проектов, так как обладает для этого всем необходимым функционалом, а также имеет бесплатный доступ к большинству функций. Более того, она позволяет организовать работу по идеям инструмента канбан-досок, применяемых в методологии Канбан, выбранной для данного проекта. Trello имеет простой и понятный интерфейс, десктопную, мобильную и онлайн версии.

В общем структуру Trello можно описать следующими тремя элементами:

- доска – рабочий экран, содержащий списки;
- списки, содержащие карточки;
- карточки – формы для описания задач, которым можно задать название, метку, срок выполнения, ответственных за задачу и подзадачи.

Пример использования сервиса Trello в данном проекте представлен на рисунке 4.

В данном примере используются цветные метки, каждая из которых означает следующее:

- сиреневая – для обозначения работ над клиентской частью приложения;
- синяя – для обозначения работ над серверной частью приложения;

- красная – для обозначения работ, выполнение которых находится в приоритете.

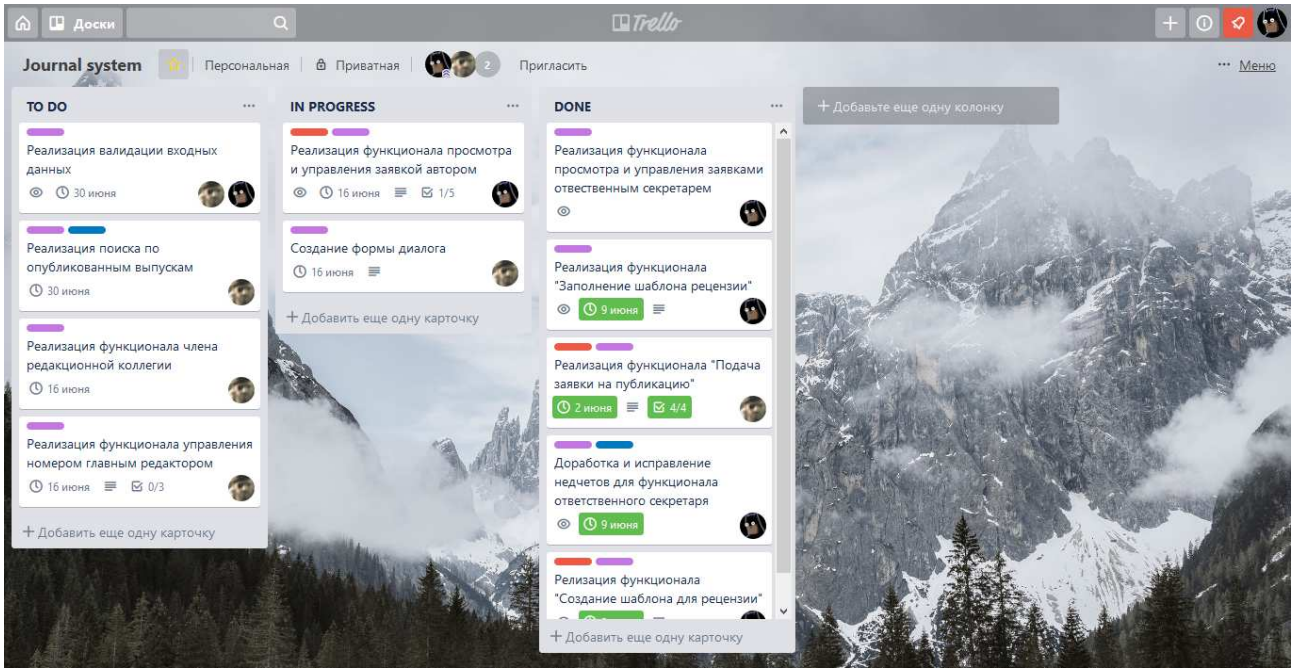

Рисунок 4 – Доска в Trello

2.2 Проектирование

2.2.1 Проектирование архитектуры информационной системы

Архитектура информационной системы – это концепция, определяющая модель, структуру, выполняемые функции и взаимосвязь компонентов системы [19; 20]. Говоря о веб-приложениях, архитектура определяет взаимодействие между приложениями, системами промежуточного программного обеспечения и базами данных, чтобы обеспечить совместную работу нескольких приложений.

При выборе архитектуры для веб-приложения, необходимо понимать, что Интернет – это постоянно изменяющаяся и расширяющаяся среда. Технологии разработки, архитектуры и другие элементы веб-разработки не всегда имеют строгую терминологию, и понятия могут быть размыты. Таким образом, разработчики понимают технологии по-разному, в следствие чего возникает огромное количество вариаций реализаций веб-приложений, что может запутать неопытных программистов на этапе проектирования. Несмотря на количество вариаций и пониманий архитектуры веб-приложений, они могут быть схематично выделены [21].

Классическая клиент-серверная модель. Это двухзвенная архитектура, состоящая из клиента (объект, запрашивающий информацию оп сети, например, персональный компьютер) и сервера. Функциональные части приложения, разработанного согласно данной архитектуре, взаимодействуют по схеме «запрос-ответ», где клиент инициирует запросы, а сервер пассивно на

них отвечает [22]. Минусом данной модели является так называемый «толстый клиент», когда на клиентской части системы реализуется бизнес-логика. Это возлагает на клиент большую нагрузку, которую можно перенести на сервер. Для решения данной проблемы разработчики начали использовать серверные языки программирования, в результате чего выделился еще один уровень и образовался новый вид модели, описываемый далее.

Трехуровневая (трехзвенная) архитектура состоит из трех компонентов: клиент (уровень представления), сервер веб-приложения (уровень бизнес-логики) и сервер БД (уровень хранилища данных). Помимо устранения проблемы «толстого клиента», данная модель представляет возможность масштабируемости и большую конфигурируемость. Промежуточный уровень выполняет основную бизнес-логику, то есть принимает и обрабатывает запросы пользователя, совершает соответствующие запросы к БД, а затем обрабатывает полученный от сервера БД ответ и отправляет его клиенту. Так как данная модель легко масштабируема, то не возникнет трудности в работе с несколькими базами данных одновременно.

Для более комплексных и сложных систем существует сервис-ориентированная архитектура (SOA) [23], суть которой заключается в модульном подходе к разработке. Часто данная архитектура используется для разработки программных комплексов, которые представляют из себя набор-веб служб.

Проанализировав вышеперечисленные модели, было принято решение использовать трехуровневую архитектуру, схема которой представлена на рисунке 5.

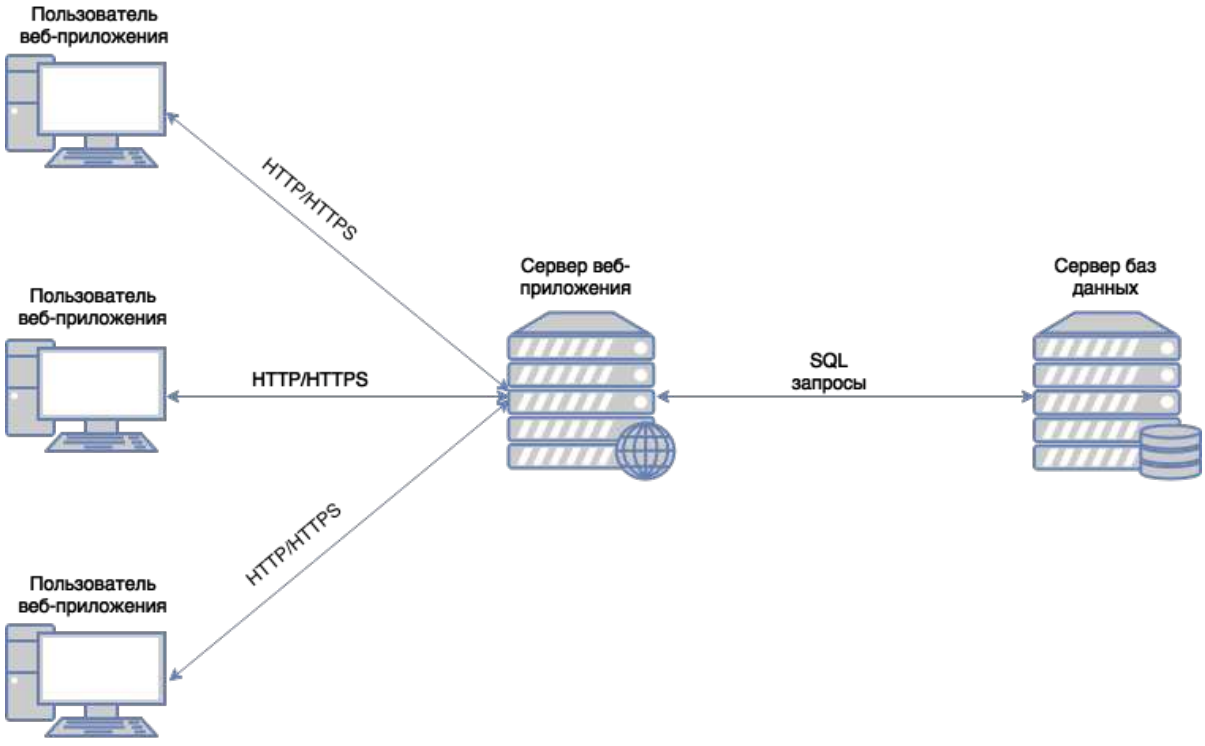

Рисунок 5 – Схема трехуровневой архитектуры

Данная модель была выбрана по следующим причинам:

- масштабируемость;
- возможность многократного повторного использования общих функций обработки данных;
- конфигурируемость;
- высокая безопасность.

При выборе и проектировании архитектуры преследовалась цель максимально снизить зависимость между уровнями. Для этого необходимо убедиться, что каждый уровень оперирует необходимым только ему набором данных.

В итоге веб-приложение приложение разбито на три независимые части:

- клиентская часть, разрабатываемая с использованием HTML, CSS, JavaScript и React.js;
- сервер веб-приложения, в качестве которого выступает Node.js в связке с Express.js;
- сервер БД, в роли которого выступает PostrgeSQL.

2.2.2 Проектирование базы данных

Проектирование БД – одна из наиболее ответственных и трудных задач, связанных с созданием информационной системы. Архитектура системы баз данных – это совокупность ее основных функциональных компонентов, а также средств обеспечения их взаимодействия друг с другом пользователями и системным персоналом [24].

В данной работе БД проектируется на основе реляционной модели данных. Такая модель описывает как необходимо организовывать данные в виде таблиц и связи между этими таблицами.

Реляционная БД характеризуется следующими принципами:

- использование ключей (первичных – идентифицирующих строку таблицы, внешних – позволяющих связывать поля таблиц);
- отсутствие избыточности (дублирования) данных;
- наличие ограничений ввода;
- поддержка целостности данных (установка ограничений при связывании таблиц).

Вышеперечисленные принципы образуют следующие задачи проектирования БД:

- обеспечение хранения в БД всей необходимой информации;
- обеспечение работы всех необходимых запросов, то есть возможность получения данных по всем этим запросам;
- минимизация дублирования данных;
- обеспечение целостности данных.

Для обеспечения целостности используется нормализация данных, целью которой является создание надежной БД за счет разделения данных на логические группы и нахождения связей между ними.

В результате проектирования была составлена схема БД, для удобства разделенная на несколько частей (рисунки 6–9). В итоге БД состоит из 32 таблиц, описание которых представлено в таблице 1.

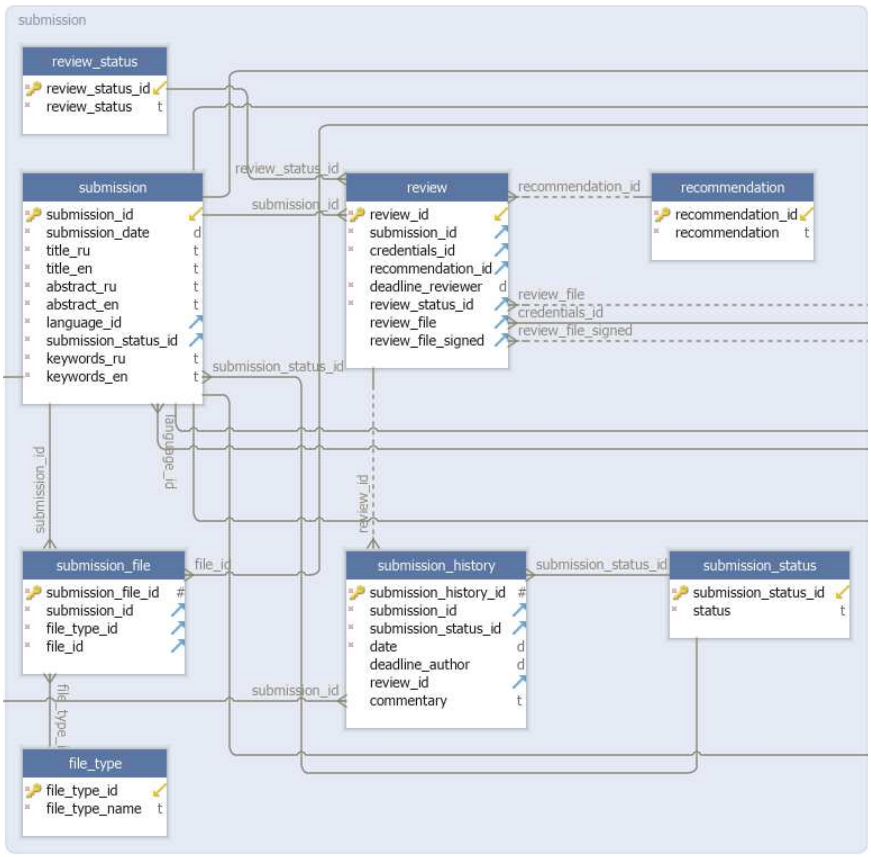

Рисунок 6 – Схема базы данных, часть 1

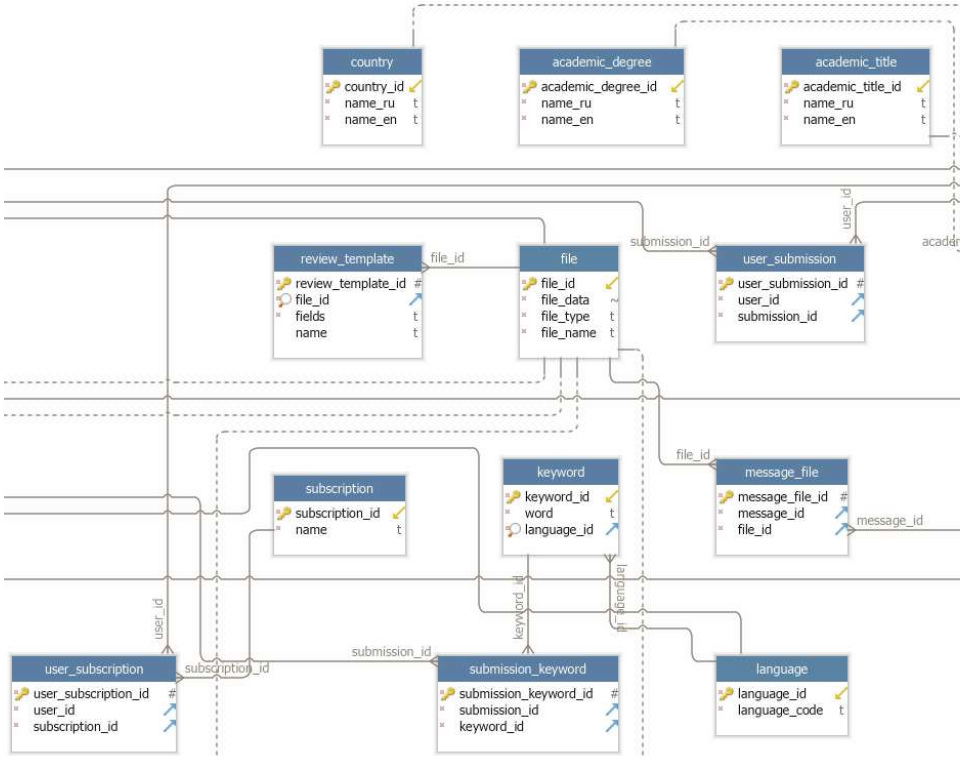

Рисунок 7 – Схема базы данных, часть 2

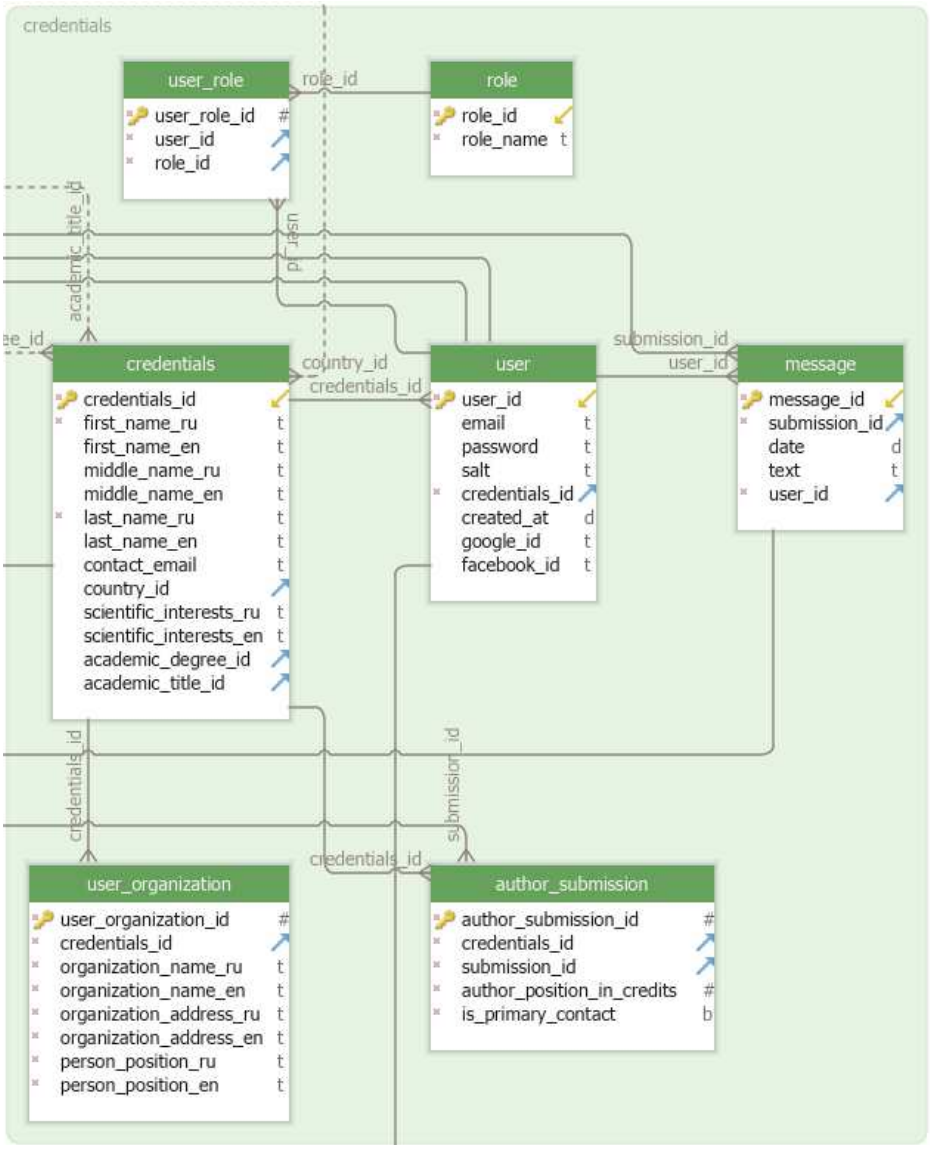

Рисунок 8 – Схема базы данных, часть 3

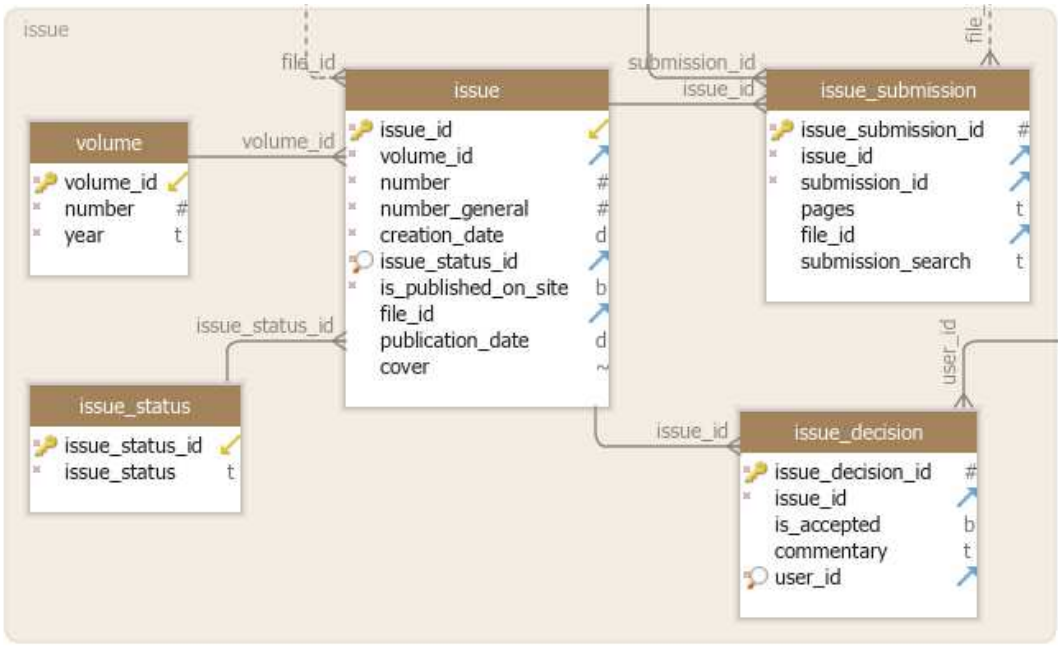

Рисунок 9 – Схема базы данных, часть 4

Таблица 1 – Описание таблиц базы данных

Название таблицы	Описание таблицы
user	Зарегистрированный пользователь системы
role	Список возможных ролей пользователя в системе
user_role	Таблица, обеспечивающая связь между пользователем и ролью (позволяет пользователю иметь несколько ролей в системе)
subscription	Список возможных подписок пользователей
user_subscription	Таблица, обеспечивающая связь между пользователем и подписками
credentials	Персональный данные зарегистрированных пользователей системы, а также авторов статей, не являющихся пользователями системы
country	Список стран на русском и английском языках
submission	Метаданные статьи, поданной в заявке на публикацию
submission_file	Файлы, прикрепленные к заявке на публикацию (таблица, обеспечивающая связь между таблицами submission и file)
file_type	Типы файлов, которые можно прикрепить к заявке на публикацию
submission_history	История заявки на публикацию
submission_status	Статус заявки на публикацию
author_submission	Авторы статьи, поданной в заявке на публикации (таблица, обеспечивающая связь между таблицами submission и credentials)
user_organization	Организации зарегистрированных пользователей системы, а также авторов статей, не являющихся пользователями системы
user_submission	Таблица, обеспечивающая связь между пользователем системы и его заявками на публикацию
submission_keyword	Таблица, обеспечивающая связь между статьей и ключевыми словами
keyword	Ключевые слова опубликованной статьи
review	Рецензия на статью, отправленную в заявке на публикацию
review_status	Статус рецензии
recommendation	Рекомендация рецензента к рецензируемой статье
message	Сообщения между автором (пользователем системы) и ответственным секретарем
message_file	Файлы, прикрепленные к сообщению (таблица, обеспечивающая связь между таблицами message и file)
issue	Выпуск журнала
issue_status	Статус выпуска журнала
issue_submission	Статьи выпуска журнала (таблица, обеспечивающая связь между таблицами submission и issue)
issue_decision	Решение члена редакционной коллегии о пригодности выпуска к публикации
volume	Том журнала
academic_degree	Список ученых степеней

Окончание таблицы 1

Название таблицы	Описание таблицы
academic_title	Список ученых званий
review_template	Шаблоны для рецензий
file	Таблица с файлами
language	Язык статьи

2.2.3 Проектирование интерфейса веб-приложения

Интерфейс приложения является своеобразным «рычагом» взаимодействия пользователя с системой, соответственно, чем лучше интерфейс, тем эффективнее взаимодействие. Интерфейс пользователя характеризуется множеством факторов, от выбора цветовой палитры до распределения элементов интерфейса на странице.

При проектировании интерфейса веб-приложения необходимо выполнить следующие задачи:

- определить общие принципы и правила, которым должен соответствовать выходной продукт;
- определить принципы, которых необходимо придерживаться при создании элементов веб-страниц;
- определить цветовую палитру.

Самый первый и достаточно важный принцип – интуитивно понятный интерфейс. Эта очень распространенная фраза, часто используемая при описании требований к продукту, в то же время является весьма абстрактной и обобщенной. Главной задачей при создании интуитивно понятного интерфейса является ясность. То есть пользователь сразу должен понимать, как пользоваться системой, а также понимать с чем он взаимодействует через предоставленный интерфейс.

Максимальную ясность могут обеспечить подсказки и правильное смысловое расположение элементов интерфейса. Причем подсказки должны быть в буквальном смысле графическими, как, например, всплывающий текст, поясняющий назначение конкретного элемента. Также интуитивность и минимализм интерфейсу могут прибавить различные иконки.

В итоге главная цель – даль пользователю понять, что он управляет программным обеспечением, а не программное обеспечение управляет им [25].

Следующий принцип касается психологии пользователей. Когда пользователь переходит на сайт, он неосознанно смотрит в правый верхний угол в поисках элемента входа или регистрации. Или, например, заходя на какой-либо коммерческий сайт, пользователь ожидает внизу страницы контактную информацию или информацию о доставке. За долгое время пользования Интернетом у людей сформировались некоторые представления, учитывая которые, можно упростить взаимодействие с системой.

Важно, чтобы пользователь совершал важные и критические действия предварительно подтверждая их. Причем диалоги подтверждения и другие различные важные информационные сообщения должны быть заметны на экране, что можно сделать либо выводя сообщение в центре экрана, то есть в активной рабочей зоне пользователя, либо делать акцент на размер элемента или его цвет.

Далее необходимо определить основные дизайнерский решения и принципы создания элементов веб-страницы. Так как в данной работе используется стиль дизайна Material Design, то необходимо придерживаться конкретных правил, таких как:

- тактильные поверхности: Material Design вдохновлен физических миром, а именно его текстурами, объектами и тем, как эти объекты отражают свет и отбрасывают тени;
- плавные переходы и появления объектов;
- адаптивность под различные устройства.

Создаваемые элементы интерфейса не должны быть сложными и разноцветными, в данном случае работает принцип «чем проще, тем понятнее». Также все элементы должны быть выполнены в едином стиле.

При определении цветовой палитры также необходимо учитывать использование Material Design, который устанавливает свой подход к выбору цветовой гаммы. Во-первых, цвета должны быть яркими и сочными. Причем все цвета разделены на три основных типа:

- основной цвет, преобладающий на всех экранах интерфейса и на большинстве его элементов;
- дополнительный цвет, использующийся в основном в блоках, поясняющих и дополняющих основной контент;
- акцидентный цвет, используемый для выделения кнопок и каких-либо важных вещей.

Однако в данном случае, цветовое решение должно обладать спокойными цветами, и иметь несколько акцидентных цветов, для выделения различных действий и состояний.

Разрабатываемое веб-приложение служит для отображения информации о журнале, о выпусках журнала и другой информации, связанной с научным изданием, однако основной функционал в данной работе сосредоточен на процессе подачи и обработки заявки на публикацию. Таким образом, появляется необходимость отделить функционал посетителя сайта и пользователя системы. Данное разделение происходит с помощью применения различного дизайна. Для посетителей сайта, которые желают узнать о журнале, его содержании и правилах существуют информативные страницы сайта с возможностью поиска по выпускам. Для зарегистрированный пользователей системы открывается возможность использования панели управления. Несмотря на то, что разметка информативных страниц сайта отличается от разметки панели управления, это не отталкивает пользователя, так как панель управления более приспособлена к разрабатываемому функционалу.

2.3 Выводы

При работе над подразделом «Планирование» целью было определение методологии разработки программного продукта. Для достижения поставленной цели были проанализированы и сравнены такие модели, как водопадная, спиральная, итеративная, Scrum и Канбан. Так как данный проект не является большим и не имеет строгих ограничений по дедлайнам, то в качестве методологии разработки был выбран подход Канбан, обеспечивающий гибкость и удобство пользования, так как управление проектом происходит с помощью визуализации списка задач. Для удобства применения этой методологии был выбран сервис Trello.

В следующем подразделе были спроектированы архитектура информационной системы, БД и пользовательский интерфейс.

Для данной системы с ее характеристиками было принято решение использовать трехзвенную архитектуру «клиент-сервер-база данных». Это не только снизит объем нагрузки на клиентскую часть веб-приложения, но также позволит при необходимости расширить систему.

В результате работы над проектированием БД была создана ее схема, состоящая из 32 таблиц.

Последним этапом было проектирование интерфейса. Были установлены требования к его реализации и общие принципы, которым необходимо следовать, чтобы сделать систему максимально удобной и понятной конечному пользователю.

3 Разработка веб-приложения

Процесс разработки был разделен на два основных этапа – разработка серверной части и разработка клиентской части. Каждый из представленных этапов был разделен на три части, каждая из которых была посвящен разработке подмодулей для автора, ответственного секретаря и рецензента. Далее будут описаны ключевые моменты разработки веб-приложения.

Разработка проводилась в текстовом редакторе Sublime Text 3, который был выбран по следующим причинам:

- свободно распространяемое ПО;
- наличие огромного количества плагинов – программных блоков, расширяющих основной функционал программы;
- программа не требовательна к ресурсам.

На рисунке 10 представлена структура проекта.

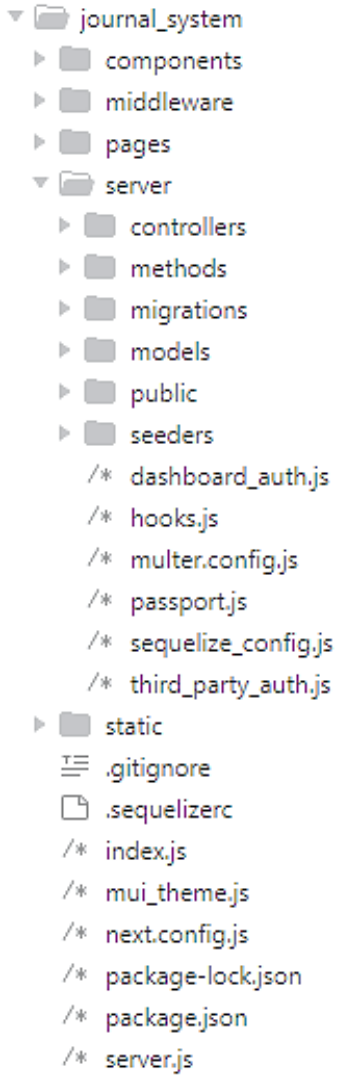

Рисунок 10 – Структура проекта

В таблице 2 представлено описание структурных единиц проекта.

Таблица 2 – Описание структурных единиц проекта

Директория	Содержание
/	Корневой каталог, содержит различные конфигурационные файлы
/components	Компоненты React.js
/middleware	Запросы к серверу, различные вспомогательные функции, используемые как на сервере, так и на клиенте
/pages	Веб-страницы
/server/controllers	Обработчики запросов клиента
/server/methods	Методы для взаимодействия с базой данных
/server/models	Модели Sequelize.js
/server/public	Хранилище различных файлов
/server/migrations /server/seeders	Данные для миграции базы данных
/static	Статические элементы (файлы)

3.1 Серверная часть веб-приложения

Основной целью при работе над серверной частью приложения была разработка API (application programming interface) – программного интерфейса приложения, то есть методов получения данных из запросов. Перед началом разработки были определены следующие требования к API:

- всегда возвращать соответствующие коды состояний;
- названия ресурсов должны быть понятны и при необходимости быть во множественном или единственном числе;
- использовать названия компонентов пути в нижнем регистре (в данном и дальнейшем случаях под путем будет подразумеваться путь для доступа к ресурсу веб-приложения);
- минимальное количество вложений в пути;
- минимизированный JSON в ответе (так как лишние пробелы увеличивают размер ответа).

При разработке серверной части веб-приложения использовался фреймворк Express. Он реализует роль контроллера, то есть получает запросы от пользователя и отправляет данные запроса на обработку. Пример работы данного фреймворка приведен на рисунке 11.

```
router.get(AUTHOR_SUBMISSIONS, async (req, res) ⇒ {
  try {
    const result = await getSubmissions(req.user, req.baseUrl);
    res.status(200).json(result);
  } catch (err) {
    res.status(err.status || 500).json({ message: err.message || err.toString() });
  }
});
```

Рисунок 11 – Получение автором заявок на публикацию

В данном примере показана обработка «GET» запроса автора на получение своих заявок на публикацию. «AUTHOR_SUBMISSIONS» является частью пути, по которому отправляется запрос.

Все пути к ресурсам приложения для удобства описаны в отдельном файле, часть которого приведена на рисунке 12. Данный файл находится в директории «/middleware», называется «api_paths.js» и представляет из себя некую карту путей.

```
const AUTHOR_BASE_PATH = '/api/author';
const AUTHOR_SUBMISSIONS = '/submissions';
const AUTHOR_SUBMISSIONS_ARCHIVED = '/submissions/archived';
const AUTHOR_SUBMISSION = '/submission/:slug';
const AUTHOR_NEW_SUBMISSION = '/submission';
```

Рисунок 12 – Карта путей для автора

На данном рисунке представлены пути для пользователя с ролью «Автор». При отправке запроса используется шаблон «AUTHOR_BASE_PATH» + «AUTHOR_*». Такой подход приводит не только к уменьшению количества повторений в коде, но также позволяет легко контролировать доступ к данным для каждой из ролей и при необходимости легко изменять путь, так как изменение необходимо произвести в одном месте.

После получения запроса сервером данные отправляются на обработку, которая связана со взаимодействием с базой данных. На данном этапе активно применяется ORM-библиотека Sequelize, суть которой заключается в предоставлении возможности обращения к таблицам БД как к объектам.

Для этого необходимо создать модели, каждая из которых связана с определенной таблицей в БД. Пример модели для таблицы «review_template» приведен на рисунке 13.

```
module.exports = (sequelize, DataTypes) => {
  const ReviewTemplate = sequelize.define('review_template', {
    review_template_id: {
      allowNull: false,
      autoIncrement: true,
      primaryKey: true,
      type: DataTypes.INTEGER
    },
    name: {
      type: DataTypes.TEXT,
      allowNull: false
    },
    file_id: {
      type: DataTypes.INTEGER,
      allowNull: false
    },
    fields: {
      type: DataTypes.TEXT,
      allowNull: false
    }
  }, {
    freezeTableName: true,
    timestamps: false
  });
  ReviewTemplate.associate = models => {
    ReviewTemplate.hasOne(models.File, {
      foreignKey: 'file_id',
      sourceKey: 'file_id'
    });
  };
  return ReviewTemplate;
};
```

Рисунок 13 – Модель «ReviewTemplate»

Модель отражает свойства таблицы, которую она определяет. При этом для каждого поля указывается:

- тип (обязательный параметр);
- обязательность, то есть может ли поле принимать нулевое значение (обязательный параметр);
- уникальность;
- указатель на автоинкремент;
- указатель на первичный ключ;
- другие свойства.

Также описываются связи с другими таблицами, с возможностью указания кардинальности (степени связи), которая позволяет описывать:

- связь типа один-к-одному;
- связь типа один-ко-многим;
- связь типа многое-ко-многим.

Пример обращения к таблице базы данных представлен на рисунке 14.

```
let submissions = await Submission.findAll({
  attributes: [
    'submission_id',
    'title_ru'
  ],
  include: [
    {
      model: AuthorSubmission,
      attributes: [ 'author_submission_id' ],
      include: [
        {
          model: Credentials,
          attributes: [
            'first_name_ru',
            'middle_name_ru',
            'last_name_ru'
          ]
        }
      ],
      order: [[ 'author_position_in_credits', 'ASC' ]]
    }
  ],
  where: { submission_status_id: SUBMISSION_STATUS.RECOMMENDED_FOR_PUBLISHING }
});
```

Рисунок 14 – Извлечение из базы данных заявок, рекомендованных к публикации

Данный запрос к базе данных состоит из следующих параметров:

- «attributes» описывает поля базы данных, которые необходимо извлечь;
- «include» позволяется извлечь данные из связанных таблицы;
- «order» позволяет сортировать результат по конкретному полю;
- «where» позволяет указать условие.

Запись, приведенная на рисунке 14, соответствует следующему запросу на языке SQL (рисунок 15).

```
SELECT
  "submission"."submission_id",
  "submission"."title_ru",
  "author_submissions"."author_submission_id" AS "author_submissions.author_submission_id",
  "author_submissions→credential"."credentials_id" AS "author_submissions.credential.credentials_id",
  "author_submissions→credential"."first_name_ru" AS "author_submissions.credential.first_name_ru",
  "author_submissions→credential"."middle_name_ru" AS "author_submissions.credential.middle_name_ru",
  "author_submissions→credential"."last_name_ru" AS "author_submissions.credential.last_name_ru"
FROM "submission" AS "submission"
LEFT OUTER JOIN
  "author_submission" AS "author_submissions" ON
  "submission"."submission_id" = "author_submissions"."submission_id"
LEFT OUTER JOIN
  "credentials" AS "author_submissions→credential" ON
  "author_submissions"."credentials_id" = "author_submissions→credential"."credentials_id"
WHERE "submission"."submission_status_id" = 4;
```

Рисунок 15 – SQL запрос на извлечение заявок, рекомендованных к публикации

Sequelize существенно упрощает написание запросов к базе данных, а также позволяется не «переключаться» с объектно-ориентированного подхода программирования к декларативному.

3.2 Клиентская часть веб-приложения

Для разработки клиентской части использовалась библиотека React.js, что подразумевает создание различных компонентов и их дальнейшее повторное использование. Основные компоненты, используемые для создания интерфейса веб-страниц приведены в таблице 3.

Таблица 3 – Описание основных компонентов

Название	Описание
NextLink	Стилизованный компонент из фреймворка Next.js для осуществления переходов по ссылкам на клиентской стороне
WithMobileDialog	Диалоговое окно, адаптированное под мобильные устройства
CustomTable	Таблица для отображения данных, их сортировки, фильтрации и поиска
EnhancedTableHead	Шапка таблицы для компонента CustomTable, содержит в себе названия столбцов и элемент управления сортировкой данных
EnhancedTableToolbar	Панель управления для компонента CustomTable, содержит в себе такие элементы, как поиск и фильтрация. Функционал данного элемента может быть расширен
SubmissionStepper	Компонент, используемый при создании заявки на публикацию, при ее дальнейшем просмотре и редактировании.
AuthorCard	Компонент, отображающий информацию об авторе при подаче заявке на публикацию (ФИО, позиция в авторах, корреспондирующий автор)
Autocomplete	Компонент для автоматического дополнения вводимых в текстовое поле данных (например, используется при вводе страны)
ConfirmDialog	Диалоговое окно для подтверждения действия пользователя
FilesDropzone	Компонент для загрузки файлов путем их перетаскивания
IssueToolbar	Панель управления для работы с выпуском, содержит элементы для сохранения и отмены изменений
NothingToDisplayText	Компонент для отображения текста при отсутствии данных
ReviewPanel	Компонент для отображения данных о рецензии (статус рецензии, дедлайн, метаданные статьи, а также функциональные элементы для управления рецензией)
StatusChip	Компонент для отображения статуса
StyledSnackbar	Компонент для отображения всплывающего уведомления

Окончание таблицы 3

Название	Описание
SubmissionHistory	Компонент для отображения истории заявки
SubmissionPanel	Компонент для отображения информации о заявке (дата подачи заявки, авторы и название статьи, статус заявки, а также функциональные элементы для управления заявкой)
TransferList	Компонент, реализующий два списка с возможностью переноса записи из одного списка в другой

Для примера на рисунке 16 представлен компонент NextLink, который принимает в себя следующие параметры:

- href – ссылка на ресурс;
- hrefAs – ссылка, отображаемая в строке браузера;
- target – атрибут для определения, где будет открыта ссылка;
- className – атрибут для присвоения css стилей;
- children – дочерние элементы компонента.

```
class NextLink extends React.Component {
  render() {
    const { href, hrefAs, target, className, children } = this.props

    return (
      <Link
        href={href}
        as={hrefAs}
        prefetch
      >
        <a
          className={className}
          target={target}
        >
          {children}
        </a>
      </Link>
    )
  }
}
```

Рисунок 16 – Исходный код компонента NextLink

Данный компоненты был использован при разработке 29 раз, что позволяет сделать вывод об удобстве использования React.js и его концепции создания компонентов.

3.3 Выводы

В результате реализации серверной части веб-приложения был разработан API, описание которого представлено в таблице 4. Каждый путь имеет базовый компонент пути в виде следующего префикса:

- для автора: «/api/author»;
- для ответственного секретаря: «/api/secretary»;
- для рецензента: «/api/reviewer»;
- для общих запросов: «/api/common».

В таблице 4 указаны пути без базового компонента пути.

Таблица 4 – Описание API

Название	Путь	Описание
Автор		
getSubmissions	/submissions	Получение всех текущих заявок
getArchivedSubmissions	/submissions/archived	Получение всех архивных заявок
editSubmission		Редактирование заявки
getSubmission	/submission/:slug	Получение информации о заявке
sendSubmission	/submission	Отправка заявки
Ответственный секретарь		
getSubmissions	/submissions	Получение всех текущих заявок
getArchivedSubmissions	/submissions/archived	Получения всех архивных заявок
getRecommendedSubmissions	/submissions/recommended	Получение заявок, рекомендованных к публикации
getSubmission	/submission/:slug	Получение информации о заявке
manageSubmission		Управление заявкой
getSubmissionReviews	/submission/:slug/reviews	Получения рецензий заявки
getSubmissionReviewers	/submission/:slug/reviewers	Получения всех рецензентов, где отмечены рецензенты, уже работавшие над данной заявкой
sendReviewToAuthor	/review/:slug	Отправить рецензию автору
getVolumes	/volumes	Получить все тома журнала
createVolume	/volume	Создать новый том
deleteVolume	/volume/:slug	Удалить том

Окончание таблицы 4

Название	Путь	Описание
getIssues	/issues	Получить все выпуски
getIssue	/issue/:slug	Получить информацию о выпуске
manageIssue		Управление выпуском
deleteIssue		Удалить выпуск
createIssue	/issue	Создать выпуск
uploadPublishedIssue	/issue/:slug/published	Загрузить опубликованный выпуск в систему
getTemplates	/templates	Получить все шаблоны для рецензирования
createTemplate	/template/:slug	Создать новый шаблон
deleteTemplate	/template	Удалить шаблон
Рецензент		
getReviews	/reviews	Получить все текущие запросы на рецензирование
getArchivedReviews	/reviews/archived	Получить все архивные запросы на рецензирование
getFormedReview	/review/:slug/formed	Загрузить сформированную рецензию
getTemplates	/templates	Получить все шаблоны
sendReview	/review/:slug	Отправить подписанную версию рецензии с рекомендацией
manageReview		Управление заявкой на рецензирование
Общие		
getMessages	/submission/:slug/messages	Получить сообщения, прикрепленные к определенной заявке
sendMessage	/submission/:slug/message	Отправить сообщение
getSubmissionHistory	/submission/:slug/history	Получить историю заявки на публикацию

В результате разработки клиентской части веб-приложения были разработаны:

- графический интерфейс пользователя (с применением разработанных React компонентов);

- логика работы с запросами к серверу и ответами на эти запросы. Подробное описание интерфейса пользователя будет приведено в следующем разделе.

4 Описание работы программного продукта

При описании работы программного продукта все функции будут представлены на примере пользователя, имеющего следующие три роли: автор, ответственный секретарь и рецензент.

При переходе на страницу панели управления пользователь видит боковое меню, соответствующее его роли в системе, и рабочую область.

Пункты бокового меню для автора:

- мои текущие заявки;
- мои архивные заявки.

Пункты бокового меню для ответственного секретаря:

- текущие заявки;
- архивные заявки;
- выпуски;
- шаблоны.

Пункты бокового меню для рецензента:

- текущие заявки на рецензирование;
- архивные заявки на рецензирование.

4.1 Функционал автора

Первым шагом в работе с системой является подача заявки на публикацию. Для этого пользователю необходимо перейти во вкладку «Мои заявки-Текущие» и нажать кнопку «Подать заявку», после чего откроется форма подачи заявки (рисунок 17).

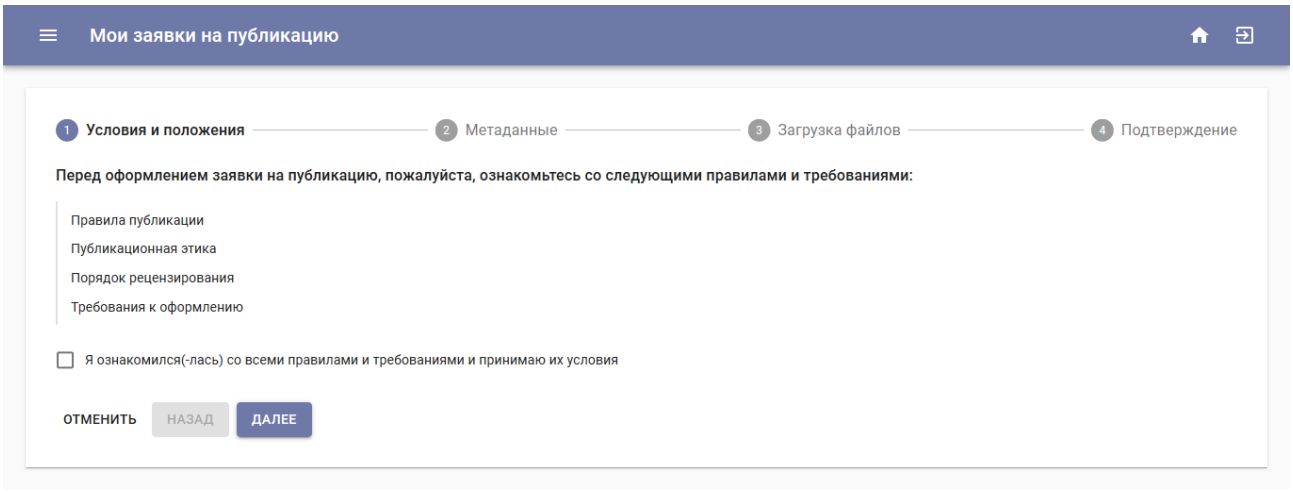

Рисунок 17 – Форма подачи заявки на публикацию

Подача заявки автором состоит из следующих шагов:

- ознакомление с правилами и политикой журнала (здесь пользователю необходимо принять все условия, выдвигаемые со стороны редакции журнала);
- заполнение метаданных, а также информации о всех авторах статьи;
- загрузка файлов статьи и других документов;

- подтверждение введённых данных.

Интерфейс этапа предоставления метаданных представлен на рисунке 18.

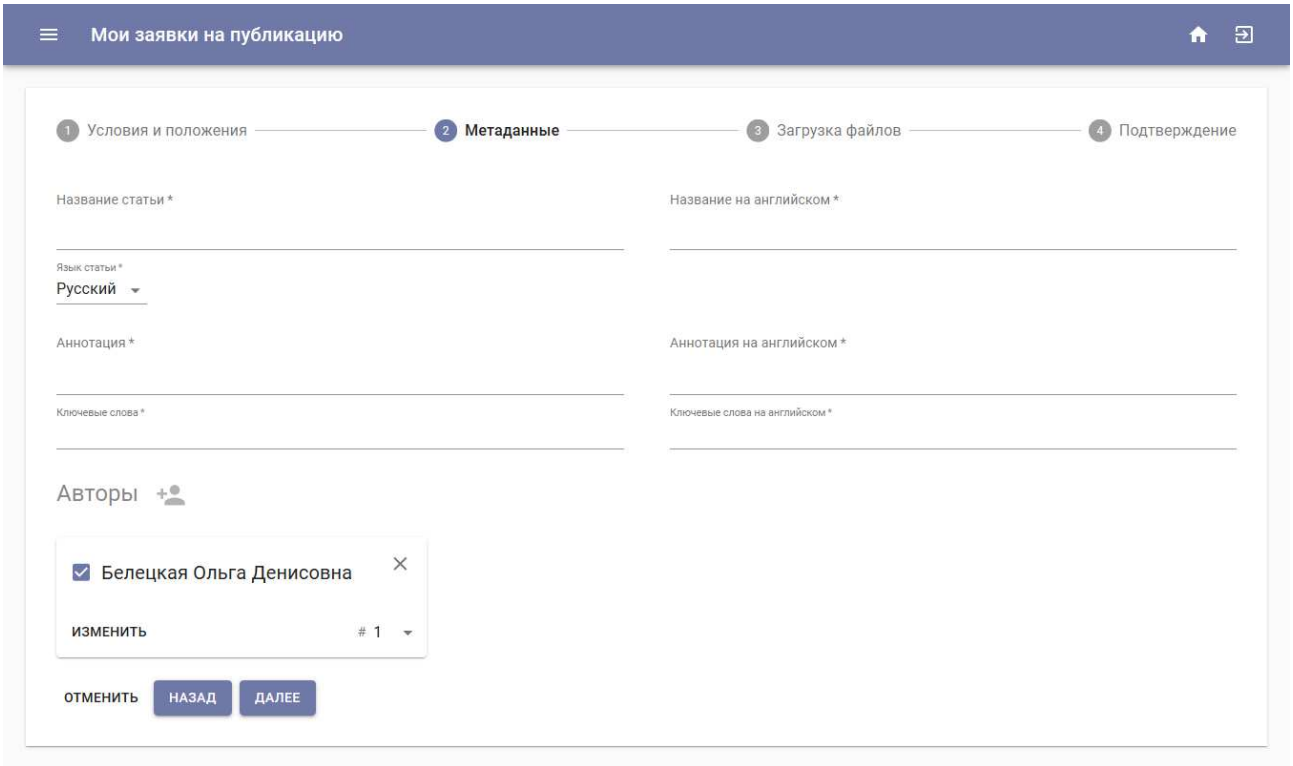

Рисунок 18 – Этап предоставления метаданных

На данном этапе необходимо указать следующие данные о статье:

- название статьи;
- язык статьи;
- аннотация;
- ключевые слова.

Далее необходимо добавить авторов статьи, для которых необходимо указать следующие данные:

- фамилия, имя, отчество;
- ученое звание (необязательно поле);
- ученая степень (необязательно поле);
- адрес электронной почты;
- страна;
- область научных интересов (не обязательно поле);
- организации (название, адрес, должность).

Все данные дублируются на английском языке. Также необходимо указать корреспондирующего автора и порядок авторов в цитировании.

На этапе «Загрузка файлов» необходимо загрузить следующие типы файлов:

- текст статьи;
- идентификационное заключение и заключение о возможности открытого публикования или только экспертное заключение.

Также при необходимости можно загрузить дополнительные файлы.

Отдельно следует отметить разработку компонента, отвечающего за подачу статьи на публикацию. Компонент SubmissionStepper является одним из самых объёмных и многофункциональных компонентов разработанной системы. Для реализации процесса подачи заявки был использован компонент Stepper из библиотеки Material UI. Данный компонент предоставляет визуализацию выполнения последовательных действий, что подходит для решения поставленной задачи.

При разработке компонента возникла сложность с недостаточной производительностью шага заполнения метаданных: поля ввода имели слишком большую задержку, которая не соответствовала поставленным требованиям к системе. Для решения данной проблемы использовались неконтролируемые компоненты – понятие библиотеки React, которое говорит о том, что программное взаимодействие с полями ввода осуществлялось с помощью ссылок на DOM или компоненты React. Принятое решение позволило избавиться от задержки, однако в некоторой степени усложнило логику работы с многочисленными полями ввода, так как, например, количество организаций автора условно не ограничено и пользователь способен добавлять автору столько организаций, сколько требуется, что приносит в код способы динамического добавления элементов и выделения им соответствующих ссылок, а в дальнейшем сбор информации по этим ссылкам.

Также следует отметить, что компонент SubmissionStepper используется не только в ходе подачи заявки пользователем, но и для просмотра и редактирования данной заявки. Для реализации данной возможности были введены дополнительные переменные-индикаторы, отвечающие за состояние компонента. Индикатор readOnly активирует режим «только чтение» заявки, при котором редактирование и изменение введенных данных не является возможным. Индикатор editMode отвечает за режим редактирования заявки, при котором возможно изменить всю предоставленную автором информацию. Процесс сбора и формирования запроса о редактировании заявки на сервер выполнен таким образом, что происходит отправка исключительно измененных пользователем полей, что позволяет не только избавить сервер и БД от выполнения ненужных действий, но и сэкономить клиентский трафик.

После отправки заявка отобразится на странице «Мои заявки-Текущие» (рисунок 19).

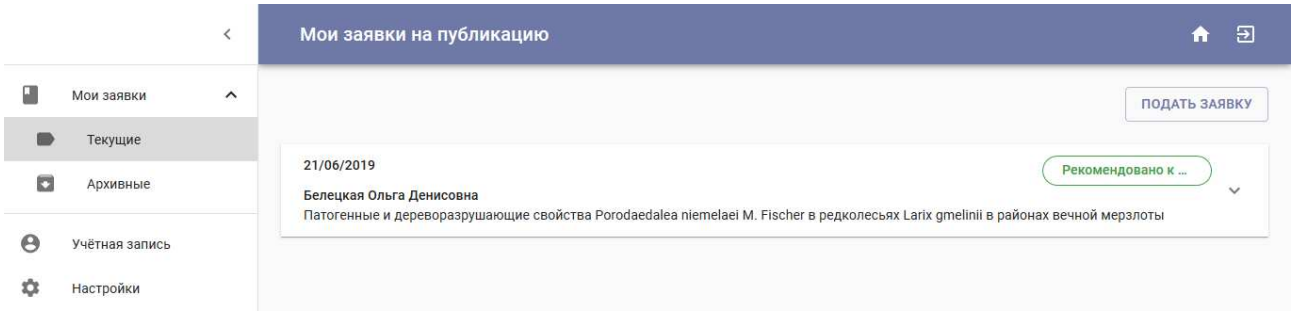

Рисунок 19 – Страницы текущих заявок автора

Опубликованные и отклоненные заявки находятся на странице «Мои заявки-Архивные».

В случае, если поданную на публикацию статью отправили на доработку, автору открывается доступ к редактированию заявки. После того, как автор внес необходимые изменения, статья снова отправляется на рассмотрение редакции журнала.

Автор также имеет возможность отслеживать историю своей заявки на публикацию. Для этого ему необходимо перейти в диалоговое окно с историей заявки, нажав на кнопку «История» панели заявки (рисунок 20).

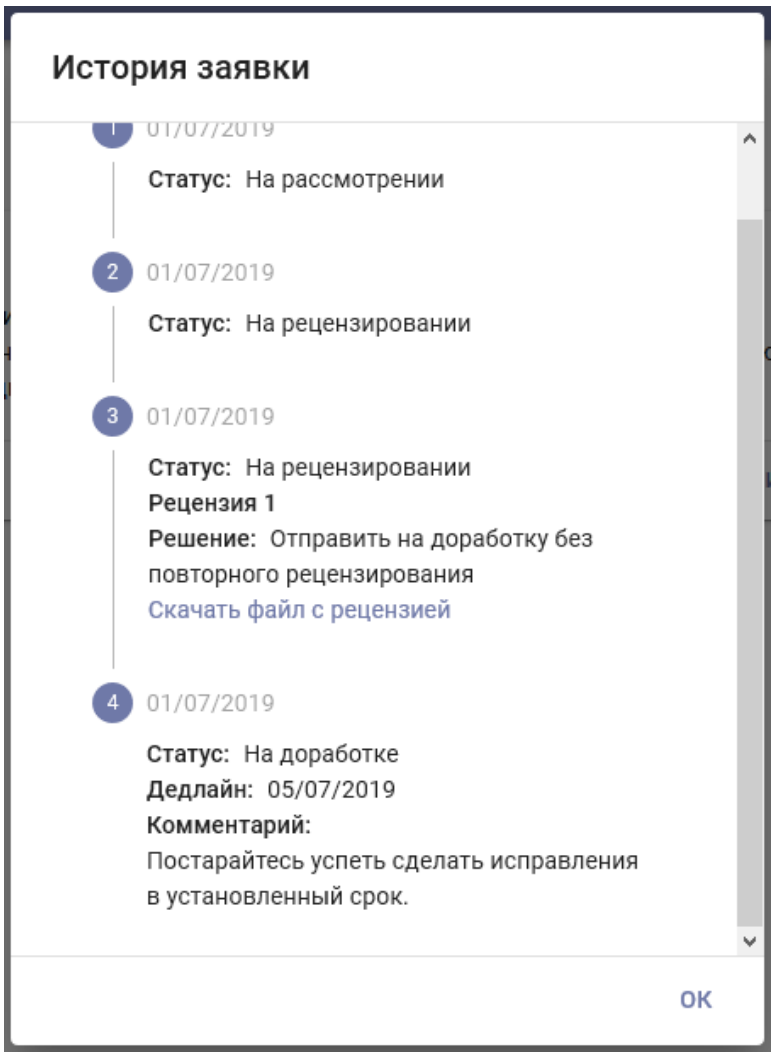

Рисунок 20 – Просмотр истории заявки

Для связи с редакцией в системе предусмотрен чат, а точнее диалог, который связан с определенной заявкой. Диалог автора с ответственным секретарем представлен на рисунке 21.

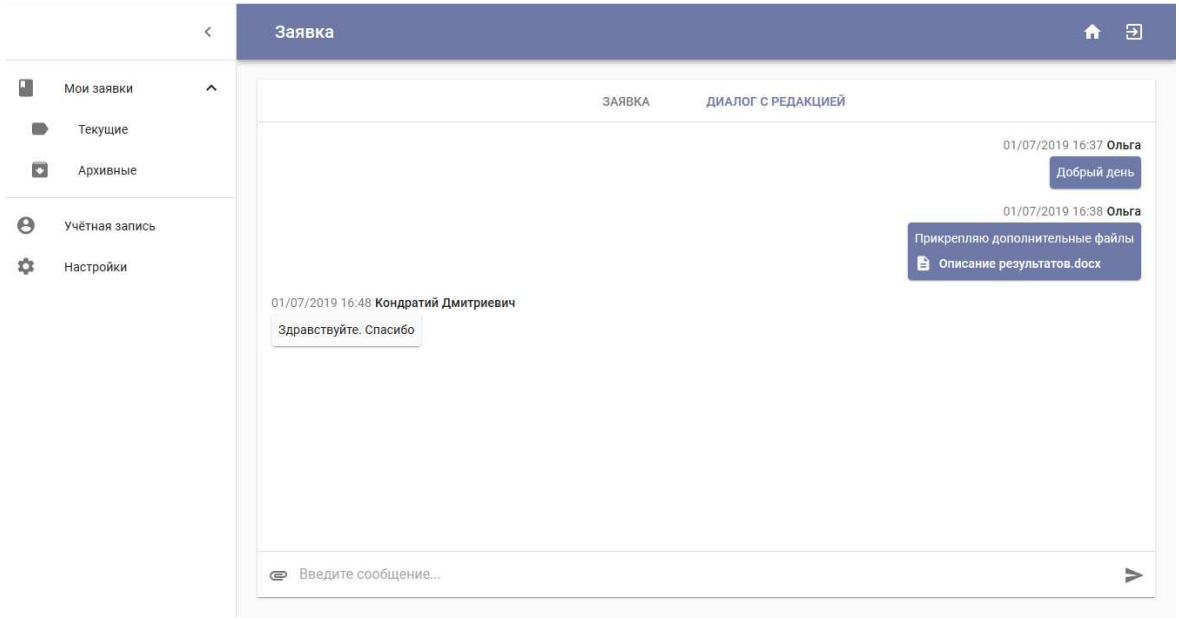

Рисунок 21 – Диалог автора с редакцией журнала

4.2 Функционал ответственного секретаря

Функционал ответственного секретаря разделен на две группы:
- управление заявками на публикацию;
- управления выпусками журнала.

Работа над заявками на публикацию начинается с просмотра новых заявок на странице «Заявки-Текущие» (рисунок 22).

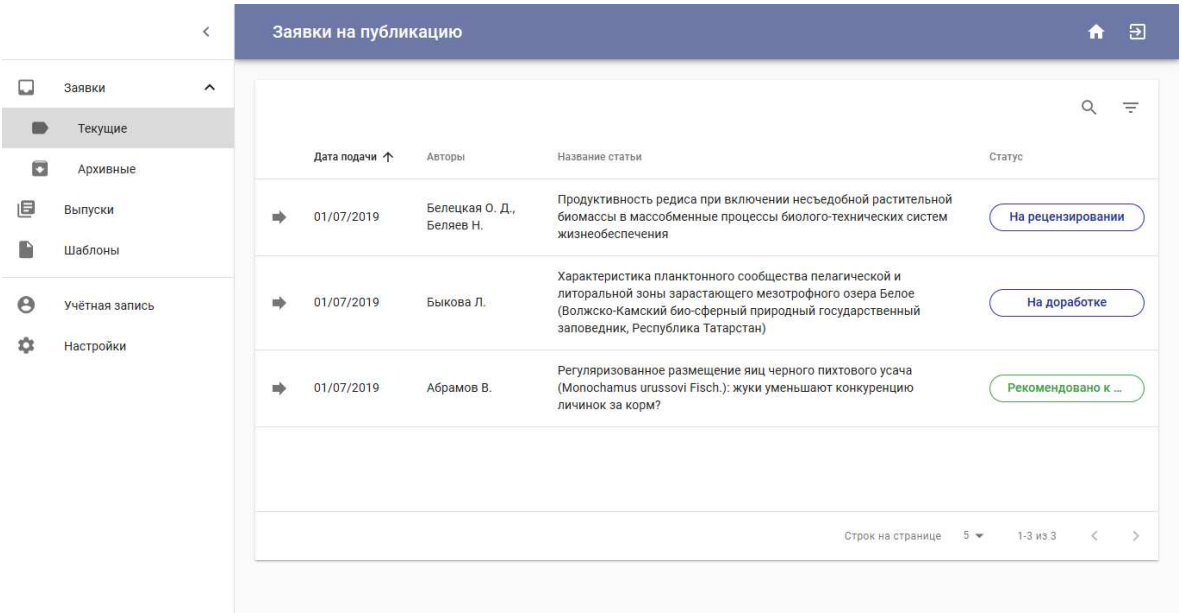

Рисунок 22 – Страница текущих заявок для роли «Ответственный секретарь»

Также ответственный секретарь может просмотреть архивные заявки (опубликованные и отклоненные) на странице «Заявки-Архивные».

На странице заявок секретарь может осуществлять поиск по авторам и названию статьи, а также фильтрацию по статусу заявки.

При переходе к отдельной заявке открывается функционал управления заявкой, а именно:

- принятие заявки к публикации;
- отклонение заявки;
- отправка статьи на доработку;
- отправка статьи на рецензирование.

Также на странице заявки отображается:

- информация о заявке (статус, метаданные, информация об авторах, прикрепленные файлы);
- рецензии на статью;
- диалог с автором.

На рисунке 23 представлена страница для заявки.

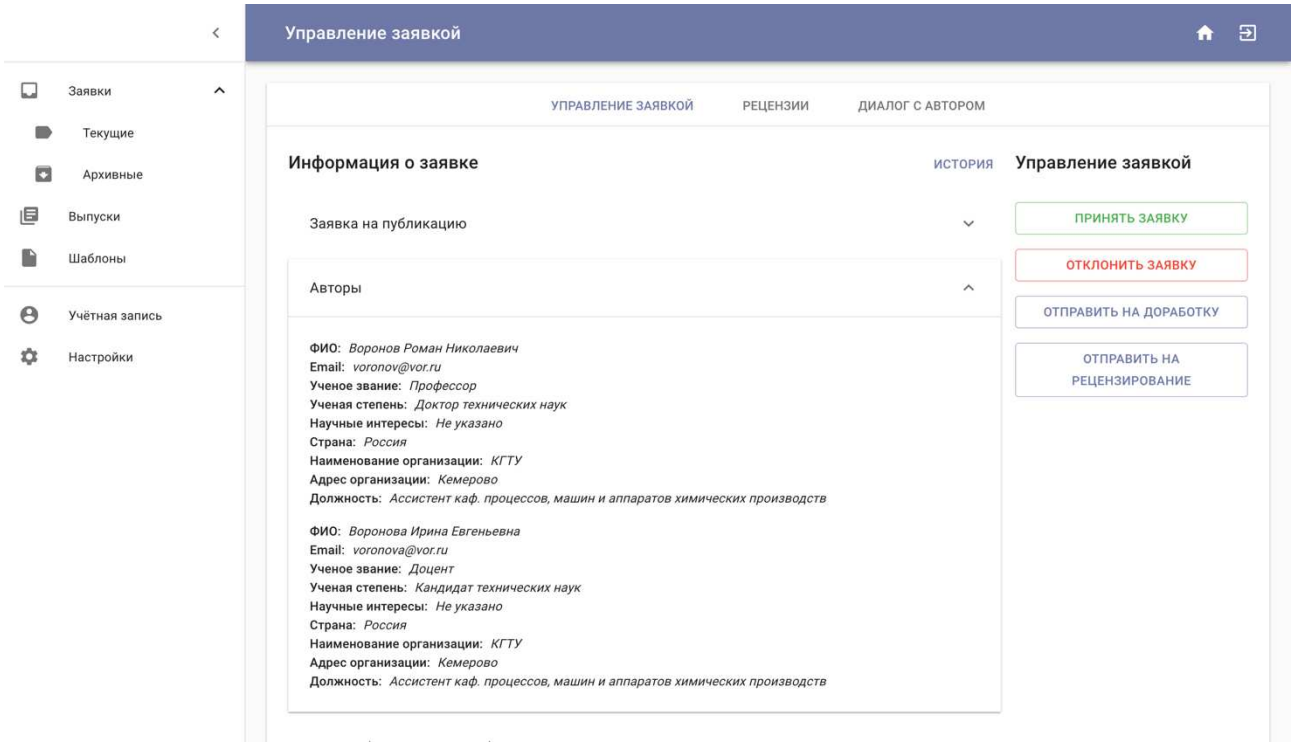

Рисунок 23 – Страница заявки

Во вкладке «Рецензии» представлен список рецензий (рисунок 24), отражающий информацию о каждой рецензии:

- статус;
- рекомендация рецензента;
- ссылки на скачивание подписанной и неподписанной версий рецензии;
- функциональный элемент в виде кнопки для отправки рецензии автору.

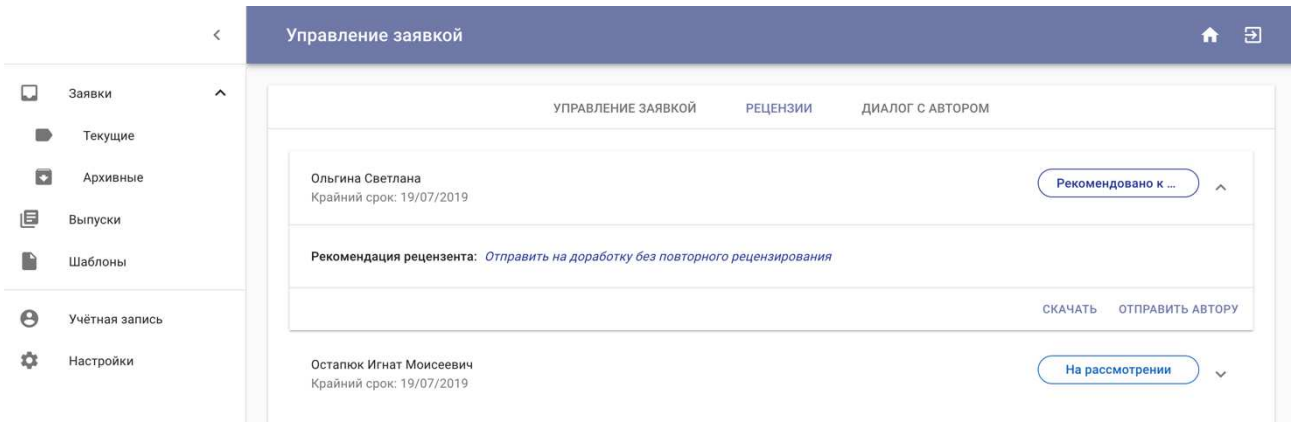

Рисунок 24 – Список рецензий статьи

После того, как заявка на публикацию принята, она получает статус «Рекомендована к публикации» и может быть использована при составлении нового выпуска журнала.

Для создание нового выпуска необходимо перейти на страницу «Выпуски» (рисунок 25).

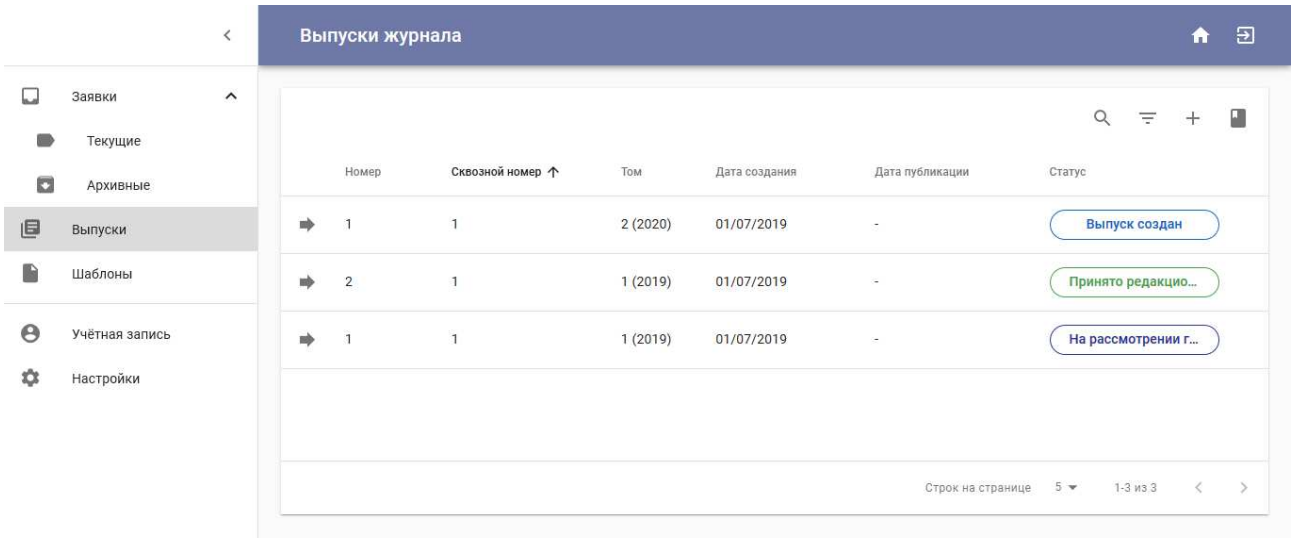

Рисунок 25 – Страницы выпусков журнала

На данной странице отображаются выпуски со следующие информацией:

- номер;
- сквозной номер;
- том;
- дата создания;
- дата публикации;
- статус.

Таблица поддерживает сортировку по всем полям, поиск по номеру, сквозному номеру и тому, фильтрацию по статусу.

Перед созданием нового выпуска необходимо, чтобы в системе были создан как минимум один том, что можно сделать, нажав на кнопку с плюсиком и выбрав пункт выпадающего меню «Создать том». В появившемся диалоговом

окне (рисунок 26) необходимо ввести номер и год тома. Посмотреть созданные тома можно, нажав на кнопку с изображением книги.

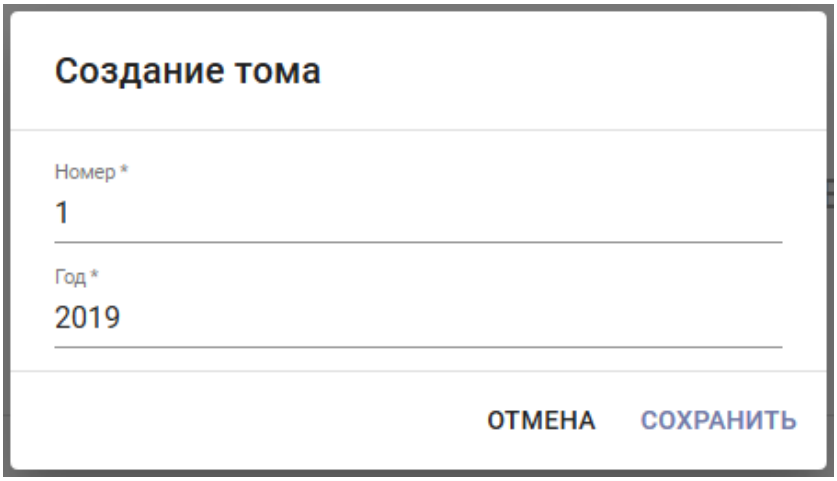

Рисунок 26 – Форма создания нового тома

После того, как в системе есть том, можно приступать к формированию выпуска. Для этого необходимо нажат на кнопку с плюсиком и выбрать пункт «Создать выпуск». В открывшемся диалоговом необходимо выбрать том, указать номер выпуска и нажать на кнопку «Заполнить выпуск», после чего появится форма для формирования выпуска (рисунок 27).

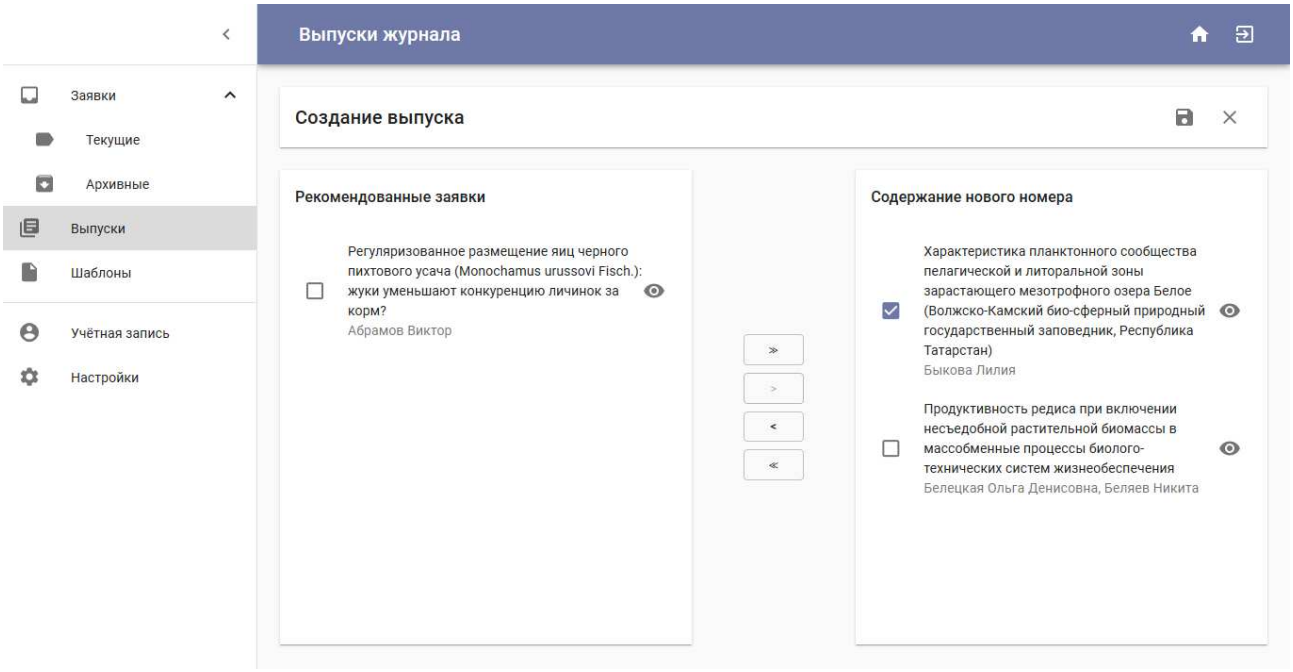

Рисунок 27 – Форма для создания нового выпуска

На форме представлено два списка:

- список с рекомендованными к публикации статьями;
- список со статьями, входящими в новый выпуск.

Путем перемещения элементов списков, пользователь составляет содержание нового выпуска. После завершения формирования выпуска,

необходимо нажать на кнопку со значком «Сохранить», убедиться, что введены верный том и номер и нажать на кнопку «Сохранить».

Просмотреть выпуски журнала можно перейдя по ссылке со страницы «Выпуски». Страница выпуска (рисунок 28) содержит следующую информацию:

- номер, сквозной номер;
- статус;
- дата создания;
- дата публикации;
- содержание выпуска.

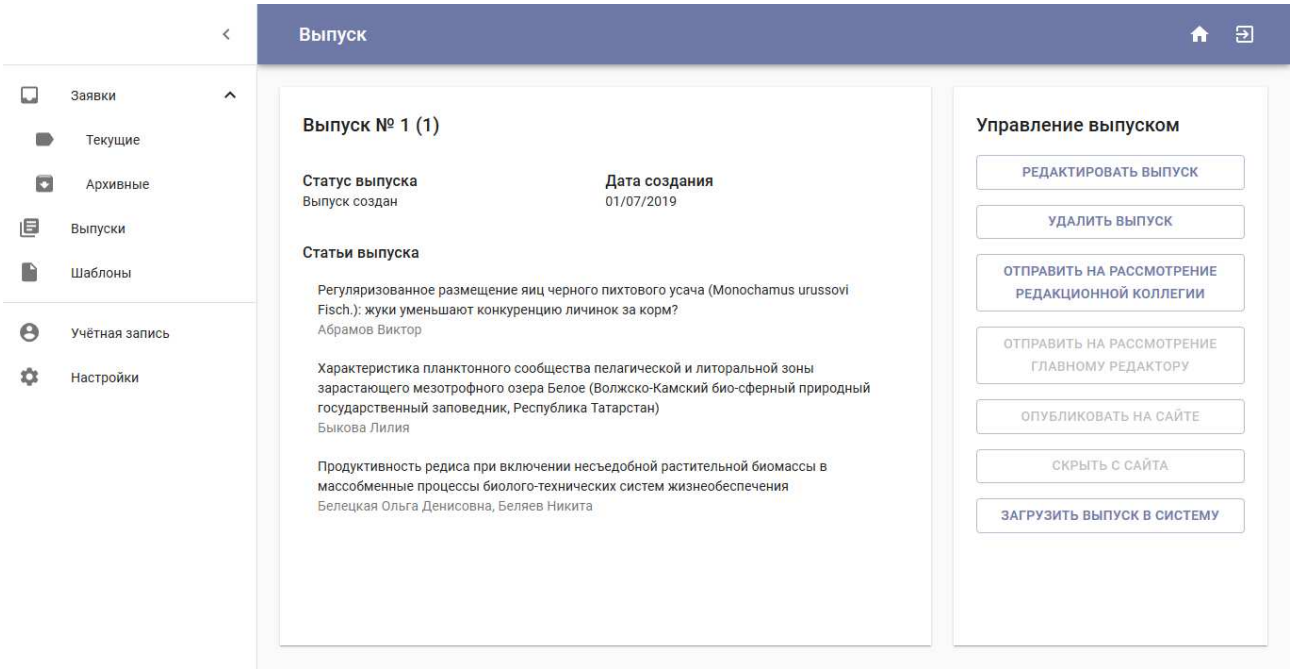

Рисунок 28 – Страница выпуска

Также на странице представлены элементы для осуществления следующего функционала управления выпуском:

- редактирование созданного выпуска;
- удаление выпуска;
- отправка на рассмотрение редакционной коллегии;
- отправка на рассмотрение главным редактором;
- загрузка опубликованного выпуска в систему;
- публикация выпуска на сайте журнала;
- скрытие выпуска с сайта журнала.

Для редактирования выпуска используется такая же форма, как и для его создания. При загрузке выпуска в систему используется форма, представленная на рисунке 29.

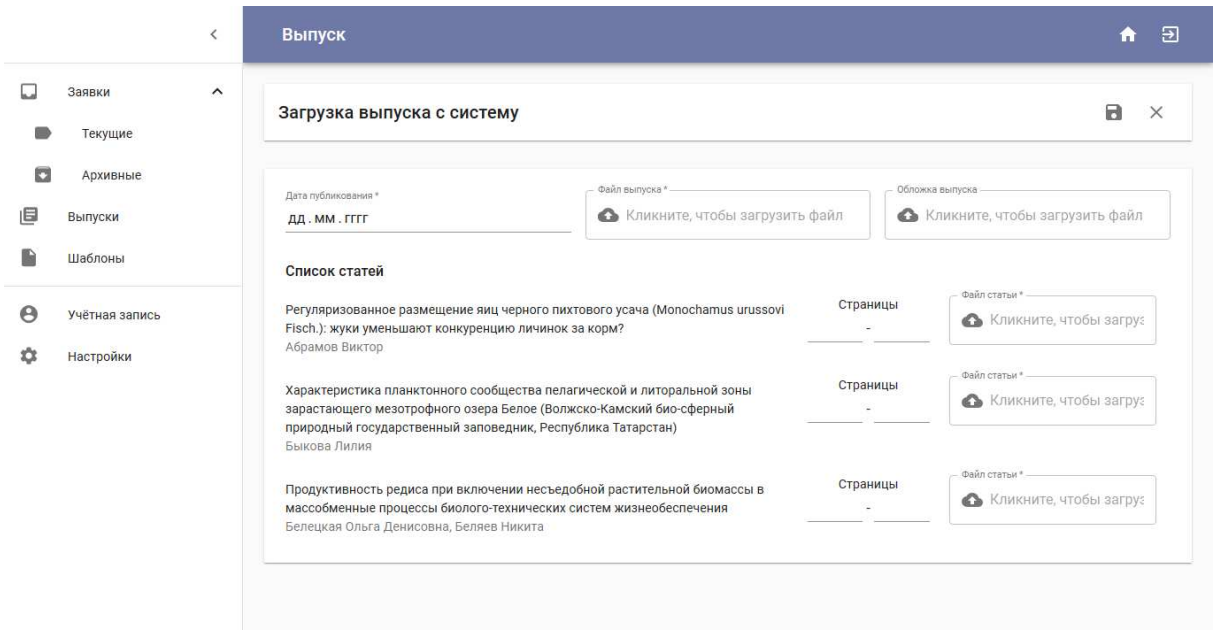

Рисунок 29 – Форма загрузки опубликованного выпуска в систему

Для загрузки выпуска в систему необходимо ввести следующие данные:
- дата публикации;
- файл с полным выпуском;
- обложка выпуска (не обязательно);
- страницы каждой статьи;
- файл с каждой статьей.

После заполнения всех полей необходимо нажать на кнопку с иконкой «Сохранить».

Помимо управления заявками на публикацию и выпусками, ответственный секретарь может создавать шаблоны для рецензирования. На странице «Шаблоны» отображаются все шаблоны для рецензий с возможностью их предпросмотра (рисунок 30).

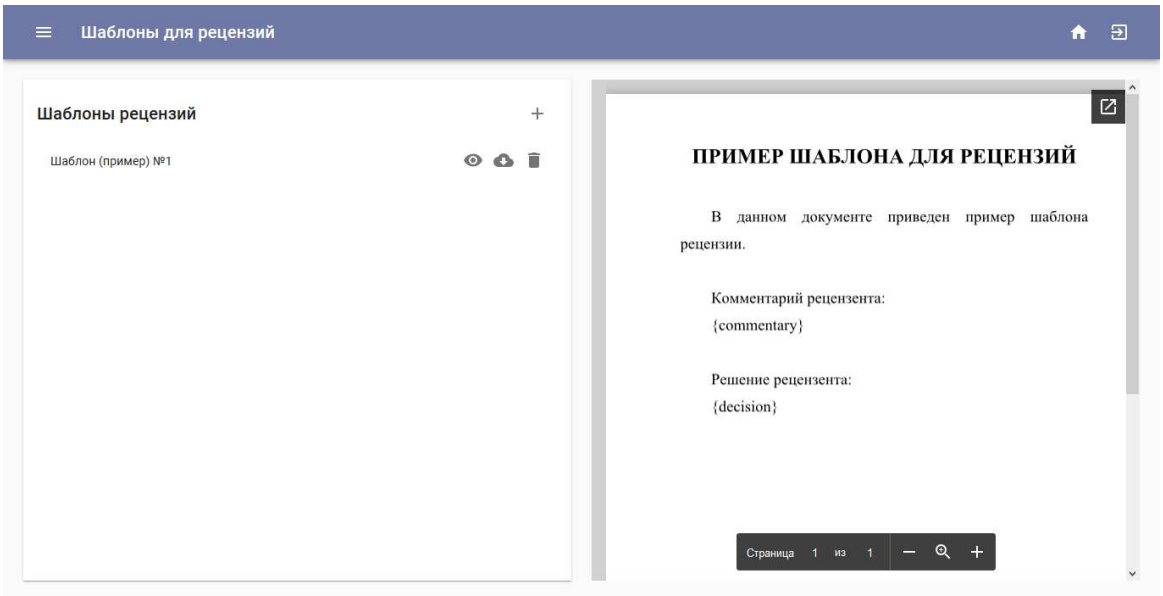

Рисунок 30 – Предпросмотр шаблона

Для создания нового шаблона необходимо нажать на кнопку с плюсиком. Далее в открывшемся окне пользователь должен заполнить данные по следующим правилам:

- ввести название шаблона;
- прикрепить файл с шаблоном;
- заполнить поля шаблона: в текстовом поле «Название поля» необходимо указать название, которое увидит рецензент, в текстовом поле «Обозначение поля» необходимо указать обозначение поля в документе.

На рисунке 31 представлен пример заполнения формы создания шаблона, а на рисунке 32 представлен пример файла шаблона рецензии.

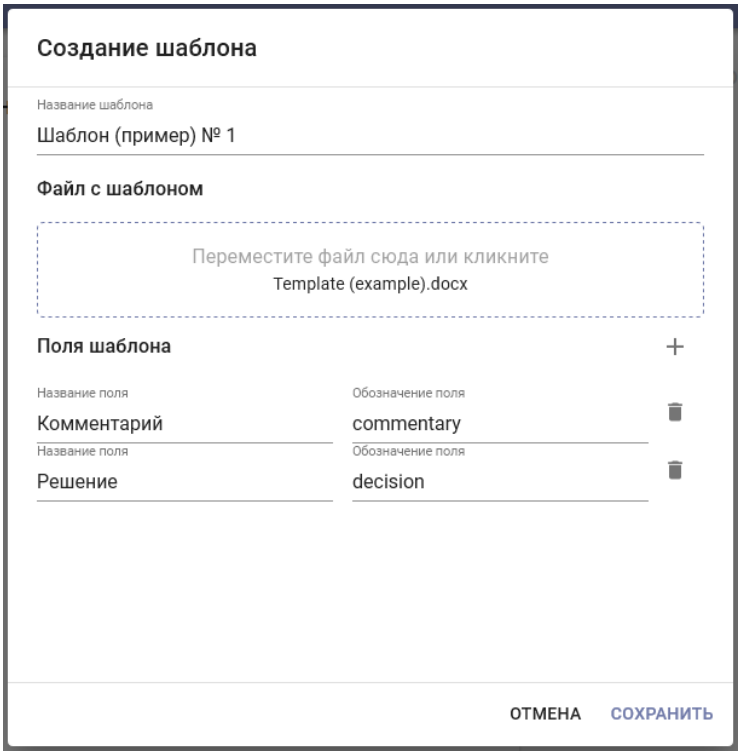

Рисунок 31 – Форма создания шаблона

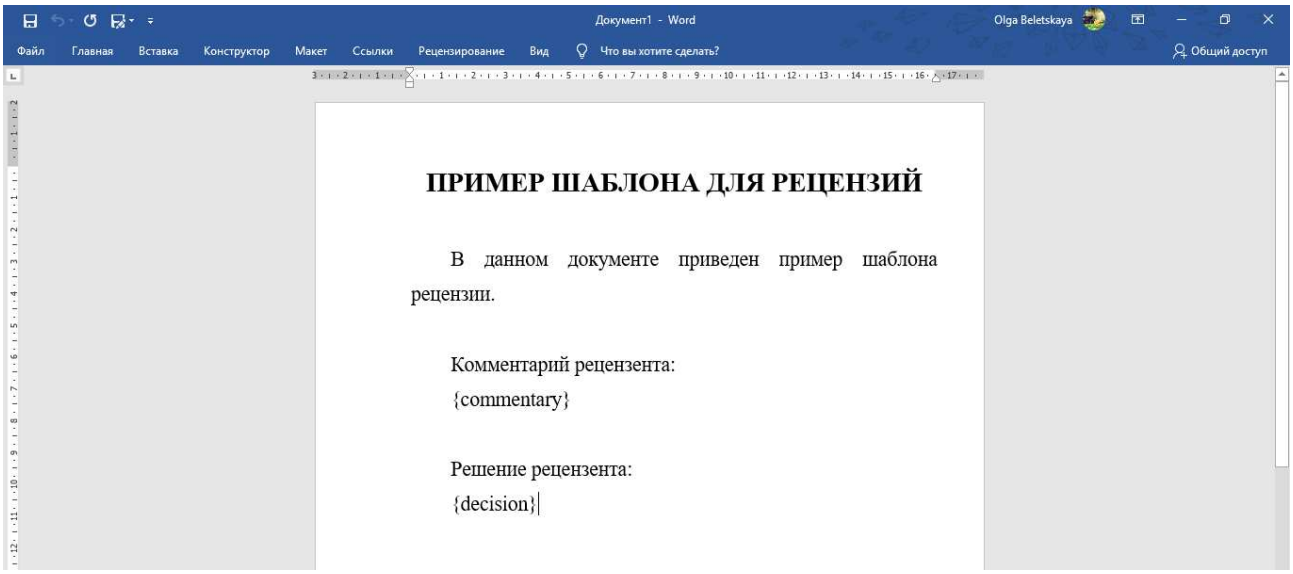

Рисунок 32 – Файл шаблона рецензии

4.3 Функционал рецензента

Как и для предыдущих ролей, для рецензента в первую очередь организован просмотр данных, а именно заявок на рецензирование:

- текущих (находящихся в работе и ожидающих решения рецензента о принятии статьи к рецензированию);
- архивных (отклоненных заявок на рецензирование и отправленных рецензий).

Страница просмотра заявок представляет из себя список с раскрывающимися элементами. Каждый элемент содержит в себе метаданных статьи, которую необходимо прорецензировать, и функциональные элементы в виде кнопок для управления рецензией (рисунок 33).

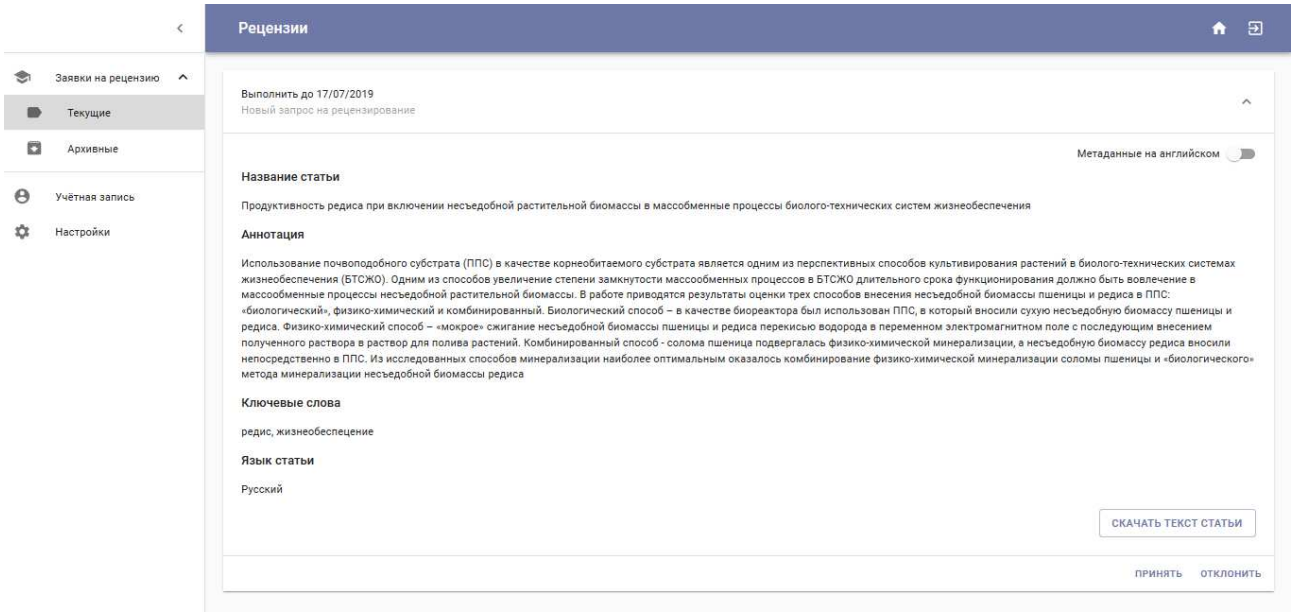

Рисунок 33 – Просмотр заявки на рецензирование

Для начала рецензенту необходимо решить, будет ли он рецензировать статью. Он может просмотреть метаданные, а также максимальный срок выполнения подготовки рецензии, после чего пользователь должен принять решение с помощью кнопок «Принять» и «Отклонить».

В случае, если рецензент отклонил заявку, она перемещается в архив. Иначе, рецензент может приступать к процессу заполнения шаблона рецензии. Для этого необходимо нажать на кнопку «Сформировать рецензию», затем в открывшемся диалоговом окне в выпадающем списке выбрать необходимый шаблон и приступить к заполнению полей. Пример заполненной формы представлен на рисунке 34.

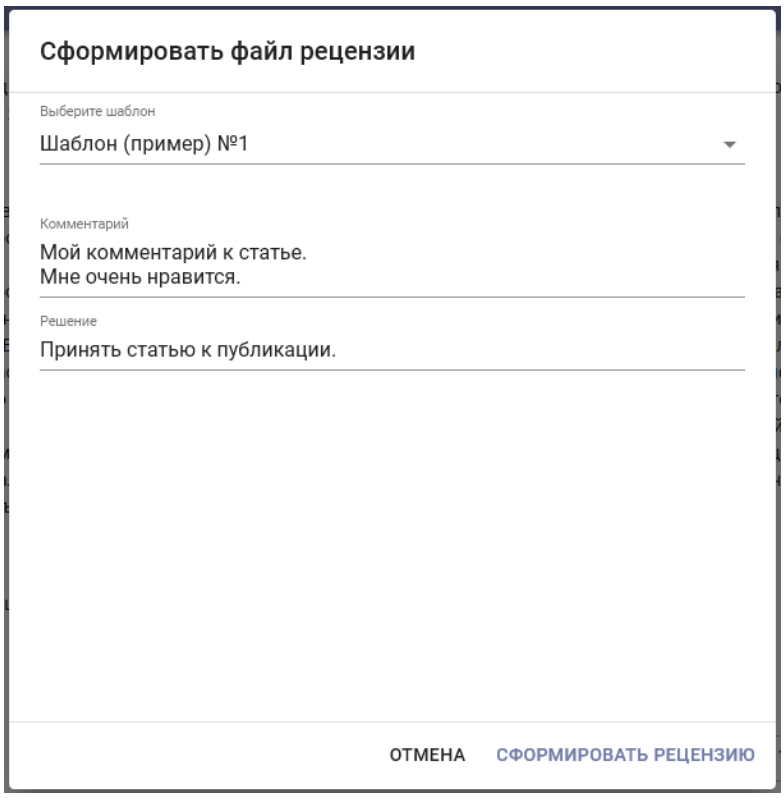

Рисунок 34 – Форма заполнения шаблона рецензии

После заполнения формы и нажатия на кнопку «Сформировать рецензию» начнется скачивание файла. Далее рецензенту необходимо загрузить подписанную версию рецензии, указав рекомендацию.

Возможно 4 вида рекомендации:

- принять;
- отклонить;
- отправить на доработку без повторного рецензирования;
- отправить на доработку с повторным рецензированием.

Отправленная рецензия перемещается в архив.

4.4 Взаимодействие пользователя с системой

Для того, чтобы предотвратить нежелательное поведение системы, все данные, введенные пользователем, проходят проверку не только на стороне сервера, но и на стороне клиента.

Для контроля данных на стороне клиента происходит валидация форм. То есть, когда пользователь вводит данные, веб-приложение проверяет эти данные на корректность. В основном, в веб-приложении проходят следующие проверки:

- проверка на пустоту данных;
- проверка на соответствие введенного формата данных необходимому.

В большинстве случаев, в разработанном веб-приложении происходит динамическая валидация, то есть проверка данных осуществляется по мере ввода этих данных. Однако при использовании неконтролируемых

компонентов, как, например, в компоненте SubmissionStepper, валидация формы проходит после перехода на следующий шаг или при попытке совершить какое-либо действие, требующее наличие корректно введенных данных. Далее будет описан процесс валидации формы подачи заявки на публикацию.

Как уже говорилось ранее, процесс подачи заявки состоит из 4 шагов:

1. условия и положения;
2. метаданные;
3. загрузка файлов;
4. подтверждение.

Таким образом, необходимо, что бы перед подачей заявки все этапы были завершены. Единственный этап, не требующий валидации, это 4 этап, так как он носит информативный характер.

Для того, чтобы первый шаг прошел валидацию, необходимо подтвердить ознакомление со всеми приведенными правилами и требованиями, отметив галочкой пункт «Я ознакомился со всеми правилами и требованиями и принимаю их условия».

Для того, чтобы второй шаг прошел валидацию, необходимо:

- заполнить все поля, отмеченные «*»;
- создать как минимум одного автора;
- каждому автору добавить как минимум одну организацию;
- выбрать корреспондирующего автора;
- определить порядок авторов в цитировании.

Для того, чтобы третий шаг прошел валидацию, необходимо загрузить следующие файлы:

- текст статьи;
- идентификационное заключение и заключение о возможности открытого публикования или только экспертное заключение.

Каждый завершенный этап отмечается галочкой. В случае, если на втором этапе пользователь не заполнил какие-либо обязательные поля, то на этапе 4 будет выведен список незаполненных полей (рисунок 35).

Примером динамической валидации является валидация формы отправки статьи на рецензирование. Здесь пользователю необходимо ввести дедлайн, дата которого обязательно должна быть больше текущей даты, а также выбрать как минимум одного рецензента. В противном случае, пользователь получит сообщения об ошибке, представленные на рисунке 36.

Еще один пример динамической валидации – проверка на пустоту полей формы. На рисунке 37 представлена форма для создания нового тома с сообщениями о некорректности введенных данных.

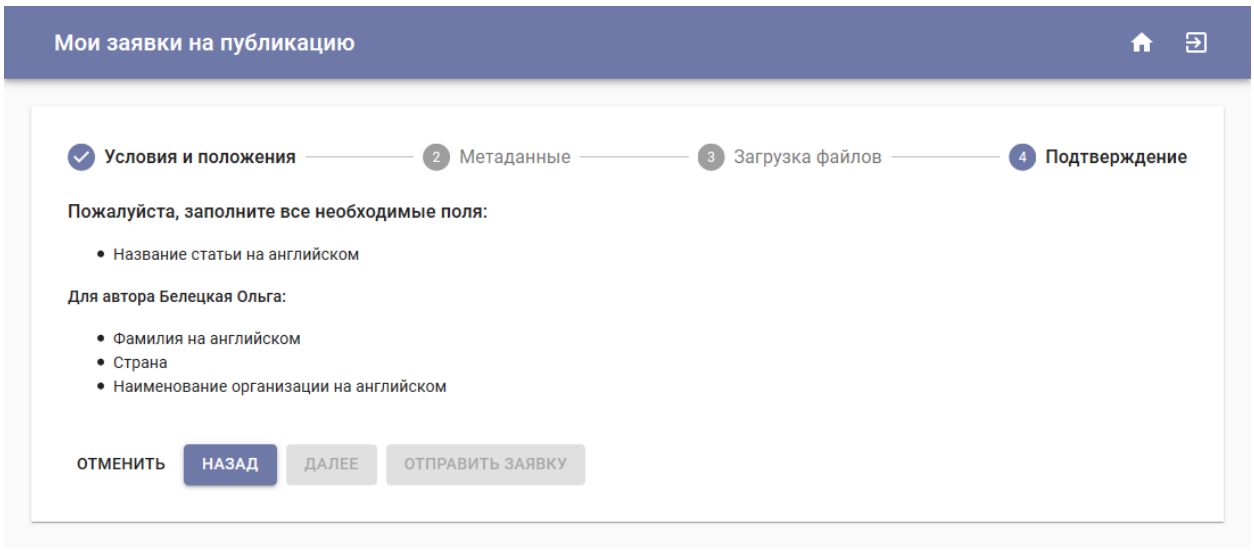

Рисунок 35 – Отображение незаполненных пользователем данных

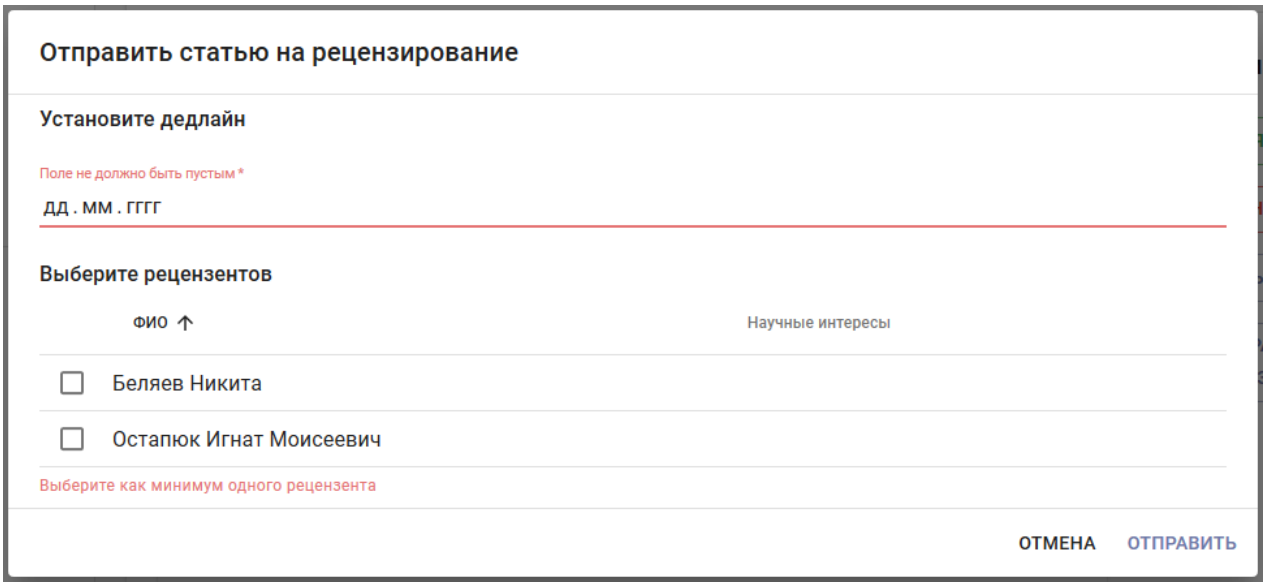

Рисунок 36 – Валидация формы отправки статьи на рецензирование

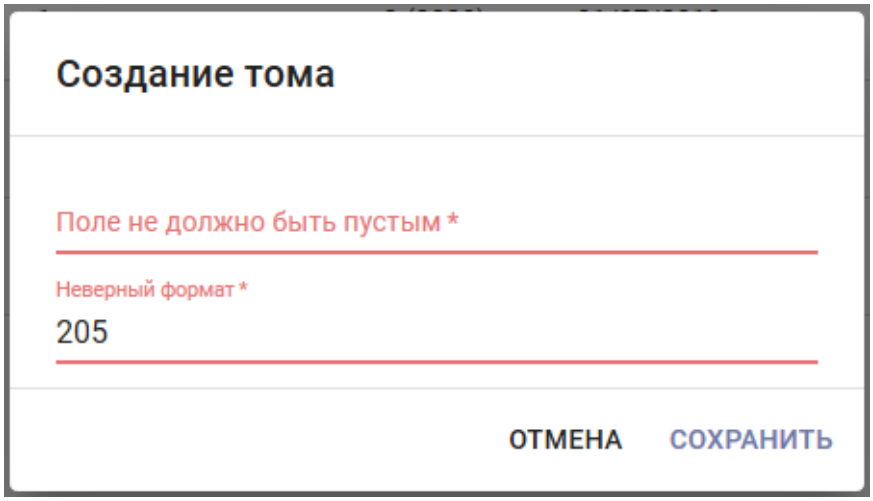

Рисунок 37 – Валидация формы создания тома

Для предотвращения нежелательных действий пользователя также используется блокирование или скрытие элементов, функционал которых не допустим на данном этапе. Например, при принятии статьи ответственным секретарем, кнопка «Принять заявку» становится неактивной. Или, например, после принятия рецензентом решения о том, будет ли он рецензировать статью, кнопки «Принять» и «Отклонить» пропадают, и вместо них появляются кнопки «Сформировать рецензию» и «Отправить рецензию».

Для того, чтобы пользователь не совершил необдуманных или случайных действий, в веб-приложении реализованы диалоговые окна, появляющиеся при совершении важного действия. Например, при попытке принять статью к публикации появляется соответствующее диалоговое окно (рисунок 38).

Рисунок 38 – Диалоговое окно при принятии статьи к публикации

В разработанном интерфейсе встречаются кнопки, представленные не в виде текста, а в виде иконок, значение которых не всегда может быть понятным. По этой причине, к таким кнопкам добавлены всплывающие подсказки (рисунок 39).

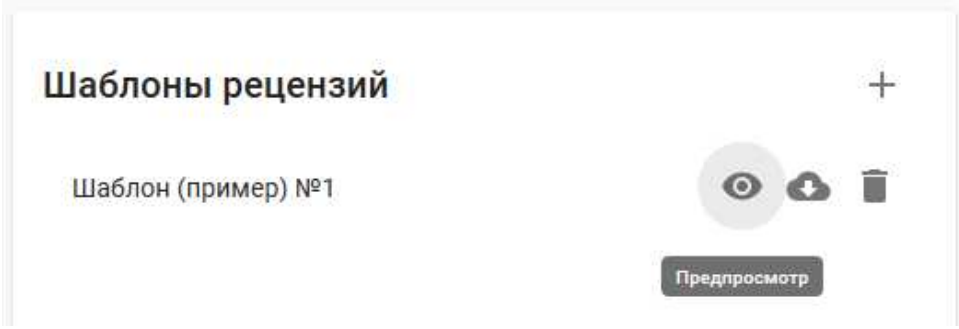

Рисунок 39 – Всплывающие подсказки

Чтобы пользователь понимал, успешно ли завершилось совершенное им действие, система выдает сообщения о том, как оно завершилось. Пример сообщений об успешном завершении действия представлен на рисунке 40.

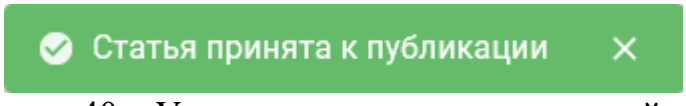

Рисунок 40 – Уведомления о завершении действия

4.5 Выводы

В данном разделе была описано функционирование разработанного программного продукта. Для каждой из ролей (автора, ответственного секретаря, рецензента) был описан функционал и возможности. Также были описаны способы обратной связи с пользователем, среди которых:

- валидация форм и уведомление пользователя о некорректно введенных данных;
- блокирование и скрытие недействительных и ненужных элементов;
- диалоговые окна при совершении важных действий;
- всплывающие подсказки;
- сообщения о результате выполнения запроса на сервер.

В разделе приведены различные скриншоты разработанного интерфейса, а также описаны некоторые особенности разработанных компонентов.

ЗАКЛЮЧЕНИЕ

В итоге выполнения выпускной квалификационной работы были решены все поставленные задачи, а именно:

- определен технологический стек проекта;
- спроектирована система приема и рецензирования научных статей;
- спроектирована база данных;
- спроектирован интерфейс системы;
- разработан функционал пользователя «Автор»;
- разработан функционал пользователя «Ответственный секретарь»;
- разработан функционал пользователя «Рецензент»;
- разработана клиентская часть веб-приложения.

В начале работы над проектом была изучена предметная область, а именно процесс подачи статьи на публикацию в научном журнале, а также процесс рецензирования научной статьи. Были выявлены особенности жизненного цикла научных работ в российских журналах, что определило основной вектор направления разработки, в которой упор делался на требования, предъявляемые к российским публикациям. Также было принято во внимание законодательство Российской Федерации, были рассмотрены следующие законы: «О средствах массовой информации» [5], «О государственной тайне» [14], «Об экспортном контроле» [15].

Разработанное веб-приложение имеет современный дизайн и полностью адаптировано под мобильные устройства. Таким образом, пользователи имеют доступ ко всему функционалу, не зависимо от того, пользуются ли они десктопной версией сайта или мобильной.

Несмотря на то, что все поставленные задачи выполнены, существуют дальнейшие перспективы развития системы, предполагающие следующие улучшения и дополнительный функционал:

- интеграция с системой Антиплагиат;
- улучшение системы рецензирования, путем увеличения ее гибкости (использование таблиц и выпадающих списков, указание размера полей ввода, возможность создания необязательных полей);
- возможность загрузить список литературы для статьи;
- внедрение в систему технологии электронной подписи (для того, чтобы рецензенту не было необходимости вручную загружать подписанную версию рецензии);
- геймификация процесса рецензирования (для поощрения и стимулирования деятельности рецензентов).

Разработанное веб-приложение предоставляет удобный функционал для подачи статьи автором и для последующего отслеживания ее жизненного цикла. Также реализован функционал для управления заявками на публикацию, и для управления выпусками журнала. Данные процессы были автоматизированы полностью, а также был частично автоматизирован процесс рецензирования.

СПИСОК СОКРАЩЕНИЙ

БД – база данных;

ПО – программное обеспечение;

СУБД – система управления базами данных;

API – application programming interface (программный интерфейс приложения);

DOM – document object model (объектная модель документа);

ORM – object-relational mapping (объектно-реляционное отображение);

SSR – server-side rendering (рендеринг со стороны сервера).

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Статистика библиотеки [Электронный ресурс] : статистика по информационным ресурсам // Научная электронная библиотека eLIBRARY.RU. – Режим доступа: https://elibrary.ru/stat_resources.asp.

2. Большая советская энциклопедия : в 30 т. / Глав. ред. А. М. Прохоров. – 3-е изд. – М.: Сов. энциклопедия, 1969 – 1978.

3. Соломонов, Ю.Ю. Региональная пресса Франции История и секреты успеха ежедневных газет / Ю.Ю. Соломонов. – М.: РИП-холдинг, 2003. – 134 с.

4. Григорьева, Е.И. Каким должен быть сайт научного журнала / Е.И. Григорьева, А.С. Кирсанов, И.М. Ситдиков // Полис. Политические исследования. – 2014. – № 5. – С. 177-187.

5. О средствах массовой информации [Электронный ресурс] : закон Российской Федерации от 27.12.1991 № 2124-1 ред. от 06.06.2019 // Справочная правовая система «КонсультантПлюс». – Режим доступа: http://www.consultant.ru.

6. Система комплексной поддержки и сопровождения научного журнала Elbup / НП «НЭИКОН» [сайт]. – Режим доступа: https://elpub.ru.

7. Open Journal Systems / Simon Fraser University Library [Сайт]. – Режим доступа: https://pkp.sfu.ca/ojs.

8. Лицензия Elpub [Электронный ресурс] : система комплексной поддержки и сопровождения научного журнала // НП «НЭИКОН». – Режим доступа: https://elpub.ru/images/files/elpubMainPrez.pdf.

9. Developer Survey Results [Электронный ресурс] : ежегодный опрос разработчиков // Stack Exchange Inc. – Режим доступа: https://stackoverflow.com.

10. Браун, И. Веб-разработка с применением Node и Express. Полноценное использование стека JavaScript / И. Браун. – СПб.: Питер, 2017. – 336 с.

11. Лузанов, П. PostrgeSQL для начинающих / П. Лузанов, Е. Рогов, И. Левшин. – Postgres Professional, 2017. – 118 с.

12. Сорокин, В.Е. Об эффективности наследования таблиц в PostrgeSQL / В.Е. Сорокин // Программные продукты и системы. – 2016. – № 3. – С. 15-23.

13. The Benefits of Server Side Rendering Over Client Side Rendering [Электронный ресурс] : WALMART LABS. – Режим доступа: https://medium.com/walmartlabs/the-benefits-of-server-side-rendering-over-client-side-rendering-5d07ff2cefe8.

14. О государственной тайне [Электронный ресурс] : закон Российской Федерации от 21.07.1993 № 5485-1 ред. от 29.07.2018 // Справочная правовая система «КонсультантПлюс». – Режим доступа: http://www.consultant.ru.

15. Об экспортном контроле [Электронный ресурс] : федеральный закон от 18.07.1999 N 183-ФЗ // Справочная правовая система «КонсультантПлюс». – Режим доступа: http://www.consultant.ru.

16. Грин, Д. Постигая Agile. Ценности, принципы, методологии / Д. Грин, Э. Стеллман. – Манн, Иванов и Фербер. – 2018. – 448 с.

17. Кон, М. Scrum: гибкая разработка ПО / М. Кон. – М.: ООО «И.Д. Вильямс», 2011. – 576 с.

18. Boeg, J. Priming Kanban: A 10 step guide to optimizing flow in your software delivery system [Электронный ресурс] / J. Boeg // Trifork. – 2012. – Режим доступа: https://www.infoq.com/minibooks/priming-kanban-jesper-boeg.

19. Кусов, А. А. Проблемы интеграции корпоративных информационных систем / А. А. Кусов // Управление экономическими системами. – 2011. – № 28. – С. 103-103.

20. Балдин, А. О понятии «Архитектура системы» / А. В. Балдин, А.Н. Данчул // Электронный научно-технический журнал «Инженерный вестник». – 2012. – № 6.

21. Web Architecture [Электронный ресурс] : what is web architecture // Ryte. – Режим доступа: https://en.ryte.com/wiki/Web_Architecture.

22. Коржов, В. Многоуровневые системы клиент-сервер / В. Коржов // Открытые системы. – 1997. – Режим доступа: https://www.osp.ru/nets/1997/06/142618.

23. Krafzig, D. Enterprise SOA: Service-oriented Architecture Best Practices / D. Krafzig, K. Banke, D. Slama. – Prentice Hall Professional, 2005. – 382 pp.

24. Дейт, К. Введение в системы баз данных / К. Дейт. – К.; М.; СПб.: Дом «Вильямс», 1999. – 848 с.

25. Лыгина, Н.И. Информатика : учебное пособие / Н.И. Лыгина, О.В. Лауферман. – Новосибирск: Изд-во НГТУ, 2017. – 84 с.

Федеральное государственное автономное
образовательное учреждение
высшего образования
«СИБИРСКИЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

Институт космических и информационных технологий
институт

Кафедра «Информатика»
кафедра

УТВЕРЖДАЮ
Заведующий кафедрой

_____ А. С. Кузнецов
подпись инициалы, фамилия

«05» 07 2019 г.

БАКАЛАВРСКАЯ РАБОТА

09.03.04 Программная инженерия
код и наименование специальности

Автоматизация процессов приема и рецензирования для веб-платформы
Тема

управления и публикации научных статей

Научный руководитель 03.07.19 доцент, канд. техн. наук А. В. Хныкин
подпись, дата должность, ученая степень инициалы, фамилия

Выпускник 03.07.19 О. Д. Белецкая
подпись, дата инициалы, фамилия

Нормоконтролер 05.07.19 доцент, канд. техн. наук О. А. Антамошкин
подпись, дата должность, ученая степень инициалы, фамилия

Красноярск 2019