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

Разработка веб-платформы управления и публикации научных статей
Тема

Научный руководитель	_____	доцент, канд. техн. наук	А. В. Хныкин
	подпись, дата	должность, ученая степень	инициалы, фамилия
Выпускник	_____		Н. Ю. Беляев
	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 Беляеву Никите Юрьевичу
фамилия, имя, отчество

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

Программная инженерия
наименование

Тема выпускной квалификационной работы Разработка 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, проектирование, разработка, описание результатов разработки, заключение, список использованных источников

Перечень графического материала 39 рисунков

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

Задание принял к исполнению _____ Н. Ю. Беляев
подпись инициалы и фамилия студента

« ___ » _____ 20__ г.

РЕФЕРАТ

Выпускная квалификационная работа по теме «Разработка веб-платформы управления и публикации научных статей» содержит 62 страницы текстового документа, 32 использованных источника, 39 иллюстраций, 6 таблиц.

ВЕБ-ПЛАТФОРМА, ВЕБ-ПРИЛОЖЕНИЕ, АВТОМАТИЗАЦИЯ, НАУЧНАЯ СТАТЬЯ, ПУБЛИКАЦИЯ В НАУЧНОМ ЖУРНАЛЕ.

Целью данной выпускной квалификационной работы является разработка веб-платформы управления и публикации научных статей. Разработанная веб-платформа направлена на повышение престижности журнала и привлечение новой аудитории читателей и учёных, что должно потенциально увеличить уровень цитирования и импакт-фактор научного журнала.

Для достижения поставленной цели были решены следующие задачи:

- проанализированы существующие решения;
- проанализированы технические средства для разработки и определён технологический стек;
- спроектирована архитектура разрабатываемой веб-платформы;
- спроектирована база данных;
- разработана серверная часть веб-платформы;
- спроектирован интерфейс и разработана клиентская часть приложения.

В результате исследования предметной области были изучены и проанализированы аналоги, был сделан вывод, что в настоящее время не существует готовых решений «под ключ» для редакций российских научных журналов. Были определены функциональные требования к веб-платформе.

В итоге была разработана веб-платформа со следующими возможностями:

- просмотр и скачивание архивных и текущих номеров и статей журнала;
- просмотр информационных страниц для потенциального автора;
- поиск по статьям, включая фильтрацию результатов по дате публикации;
- регистрация в системе и вход в неё;

- получение уведомлений на электронную почту согласно роли пользователя в системе (например, о новом вышедшем или сформированном номере);
- реализация функционала главного редактора и членов редакционной коллегии.

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

Первый научный журнал был издан во Франции в XVII веке. Количество оригинальных статей в то время было невелико, стиль и структура работ больше напоминали письма от учёного к ученому и существенно отличались от современных. Однако, начиная с XIX века, журналы становятся источником научных сведений и на сегодняшний день являются основным источником научной информации [1].

В настоящее время, в России наблюдается стабильный рост числа научных журналов, издательств и публикаций, так, например, количество публикаций в начале 2015 года составляло 16850192, однако к июню 2019 года значение выросло до 30202561 [2]. Также, по данным Глобального инновационного индекса, наша страна занимает высокие позиции по уровню генерации и приобретения новых знаний в формате научных публикаций и патентов [3]. Тем не менее, одной из слабых сторон инновационного развития России является низкая эффективность процессов распространения знаний, что связано с невысоким уровнем использования практических результатов научной деятельности.

Востребованность результатов публикаций научных трудов характеризуется уровнем их цитирования. По данным аналитического сервиса Essential Science Indicators, в последнее время среди российских учёных данный показатель растёт [4]. Несмотря на это, уровень цитирования остаётся невысоким в мировом масштабе, так как значительная часть работ исследователей России публикуется в изданиях с невысоким импакт-фактором.

Целью данной выпускной квалификационной работы (ВКР) является разработка веб-платформы управления и публикации научных статей (далее – Journal System). Предлагаемая веб-платформа потенциально способна повысить престижность журнала и привлечь новую аудиторию читателей и учёных, что в свою очередь позволит увеличить уровень цитирования, который напрямую связан с импакт-фактором научного издания. Более того, платформа направлена

на обеспечение автоматизации прикладной логики издания российских научных журналов.

Для достижения поставленной цели необходимо выполнить следующие задачи:

- провести анализ существующих решений;
- провести анализ технических средств для разработки и определиться с технологическим стеком;
- спроектировать архитектуру разрабатываемой веб-платформы;
- спроектировать базу данных;
- разработать серверную часть веб-платформы;
- спроектировать интерфейс и разработать клиентскую часть приложения.

1 Анализ предметной области

1.1 Анализ существующих решений

1.1.1 Open Journal System

Open Journal System (OJS) [5] – это программная система управления электронным журналами с открытым исходным кодом. OJS разрабатывается с 2001 года в рамках проекта Public Knowledge Project группой канадских университетов.

OJS представляет собой единую платформу для управления электронными журналами, которую возможно настраивать и подстраивать под различные бизнес-модели. Система имеет модульную архитектуру и подробную документацию, что позволяет разрабатывать собственные классы и модули. OJS поддерживает несколько языков, в том числе и русский, в комплекте также идут макеты для создания сайта, обеспечивающие корректную работу не только на персональных компьютерам, но и на мобильных устройствах. В системе реализована ролевая модель пользователей с разными правами доступа, многоступенчатый процесс публикации ресурсов и прочее.

Несмотря на большое количество преимуществ, стоит отметить, что настройка системы под нужды определённого российского журнала является непростой задачей, что обусловлено комплексностью OJS.

1.1.2 ePublishing Toolkit

ePublishing Toolkit (ePubTk) [6] – это набор инструментов, написанный на языке программирования Python, для осуществления издательской и управленческой деятельности ведения семейства электронных научных журналов. Разработка ePubTk началась в 2005 году сообществом Max Planck Society.

Архитектура ePubTk представляет собой набор компонентов, способных функционировать независимо друг от друга, а также общих библиотек, содержащих реализацию базовых функций системы. Основными компонентами являются: pubBuilder, предоставляющий функционал создания публикаций, refdb, отвечающий за управление и осуществление запросов к базе данных, EIMS (Editorial Information Management System), реализующий бизнес-процессы и жизненный цикл журнала и register компонент, служащий для авторизации и аутентификации пользователей согласно стандарту OpenID.

К плюсам ePubTk можно отнести: гибкость системы, достигаемой за счёт независимости компонентов и наличие подробной документации. Недостатками являются: отсутствие версий системы, что не позволяет сделать вывод о периодичности обновлений и планах развития, отсутствие отдельного законченного дистрибутива для установки, что делает процесс настройки и введения системы в работу достаточно сложным и требующим высокой квалификации.

1.1.3 Digital Publishing System

Digital Publishing System (DPubS) [7] – это свободно распространяемая система для публикации научных журналов, трудов конференций и монографий онлайн. Разработка DPubS началась в 2004 году в США по инициативе Корнеллского и Пенсильванского университетов. Система не обновлялась с 2008 года.

DPubS спроектирована с учётом проблем, связанных с сохранностью информационных ресурсов, имеется поддержка работы с такими институциональными репозиториями, как DSpace и FEDORA. Система имеет модульную архитектуру и содержит набор взаимосвязанных сервисов. Модули функционально разделены на: модуль объединения в коллекции, редакционный модуль, модуль индексирования, модуль поиска, обратной связи, репозитория, подписки, модули пользовательского и административного интерфейсов.

Редакционный модуль отвечает за загрузку статей авторами и передачу их рецензентам, дальнейшую подготовку и публикацию в выпуске, перемещение выпуска в хранилище.

Система имеет документацию по настройке и поддержке, руководства системного администратора и программиста, однако, в целом, документация не является полной, зачастую описания модулей недостаточны или вовсе отсутствуют. DPubS для функционирования необходим отдельный сервер, на котором не должно быть никаких других веб-приложений.

1.1.4 Elpub

Elpub [8] – это платформа комплексной поддержки и сопровождения научного журнала, которая обеспечивает соответствие всем требованиям и условиям международных баз любого уровня. Elpub была разработана при участии Министерства науки и высшего образования РФ по Госконтракту в 2013 году.

Elpub по умолчанию предоставляет двуязычный сайт журнала, готовые шаблоны ключевых базовых текстов и писем, модули контроля рецензентов и быстрого рецензирования, полный функционал поисковых механизмов, разметку и размещение свежих выпусков, а также интеграцию с системами Антиплагиат, CrossRef и РИНЦ.

Elpub не является бесплатным продуктом и распространяется по подписке от 90000 рублей в месяц, что может являться существенным препятствием для множества небольших научных журналов. Также в стоимость входят другие услуги, такие как техническая поддержка, поддержка редакции, производство выпусков, оформление пакетов документов и набор других прочих услуг, не связанных с самой автоматизированной системой управления и ведения журнала.

1.2 Определение требований к системе

Проанализировав аналоги разрабатываемой веб-платформы в подразделе 1.1, были составлены следующие ключевые функциональные требования к системе:

- регистрация и авторизация в системе посредством логина и пароля, а также через сторонние сервисы, такие как Google и Facebook;
- простая, интуитивная навигация по страницам веб-приложения через навигационную панель;
- наличие страниц, посвящённых правилам и политике журнала, страницы этики и общей информации для потенциального автора;
- наличие страниц с томами журналов, каждый из которых содержит выпуски;
- страница выпуска должна содержать полную информацию о каждой статье на русском и английском языках: наименование статьи, аннотацию, ключевые слова, фамилию, имя и отчество авторов, а также организации, в которых они состоят с занимаемыми ими там должностями;
- пользователи могут выполнять поиск по статьям, опубликованным на сайте;
- зарегистрированные пользователи системы по желанию могут получать информацию о новых выпусках посредством электронной почты;
- пользователь с правами администратора может просматривать информацию об остальных пользователях системы, удалять их и присваивать им роли;
- пользователь с правами главного редактора может подтверждать публикацию номеров, а также просматривать историю своих действий;
- пользователи с правами члена редакционной коллегии могут выносить решение о формирующихся выпусках в форме «да-нет», а также оставлять к ним комментарии.

1.3 Выбор инструментов разработки

1.3.1 Фронтенд

В настоящее время существует множество различных библиотек и фреймворков для реализации клиентской части веб-приложения. Согласно опросу сайта stackoverflow [9] в 2019 году, React получил большее признание, нежели Angular, Vue и прочие библиотеки.

Ключевой концепцией React являются компоненты. Компоненты – это JavaScript функции, которые возвращают HTML элементы, отображающиеся в браузере. Компоненты зачастую независимы друг от друга и поэтому их удобно проектировать и реализовывать таким образом, чтобы их можно было удобно использовать повторно. У каждого компонента есть свой жизненный цикл, он определяет поведение компонента от его рендера в браузере пользователя, до его исчезновения. Также компоненты имеют внутреннее состояние, изменения которого вызывают их полную или частичную перерисовку [10].

React был выбран в качестве библиотеки написания фронтенда по следующим причинам:

- наличие предыдущего опыта разработки на React;
- поддерживает серверный рендеринг;
- имеет большое количество готовых свободно распространяемых компонентов.

Определившись с библиотекой построения фронтенда, необходимо решить какую графическую библиотеку следует использовать. Для React существует большое количество решений, самым популярным из них является Material UI.

Material UI – это набор готовых компонентов, разработанных согласно визуальному языку Material Design компании Google. Из особенностей Material UI можно выделить тот факт, что изначально библиотека создавалась для мобильных устройств, а потом масштабировалась до десктопа. Благодаря

следованию философии Material Design, графические элементы библиотеки выглядят одинаково на всех видах устройств, а изменение темы приложения является простой операцией замены цветов в JavaScript файле.

Было принято решение использовать Material UI как библиотеку для построения пользовательского интерфейса благодаря её хорошей документированности и лаконичности, что важно для тематики разрабатываемой веб-платформы.

1.3.2 СУБД

Система управления базами данных (СУБД) – это «сердце» современных приложений и перед выбором подходящей СУБД следует определиться с требованиями к хранимым данным.

Ожидается, что на разрабатываемой веб-платформе будут размещаться тома и выпуски научного журнала. Каждый том состоит из выпусков, выпуски включают статьи, которые содержат набор метаданных, представленный как на русском, так и на английском языках: название статьи, авторов, аннотацию и ключевые слова. Пользователь может скачать, как и целый выпуск, так и каждую статью по отдельности. По данному описанию функционала, становится понятно, что между сущностями базы данных существуют явные связи, что говорит о том, что выбираемая СУБД должна быть реляционной.

Среди свободно-распространяемых реляционных СУБД существуют два «гиганта»: MySQL и PostgreSQL.

В отличие от MySQL, PostgreSQL является объектно-реляционной системой управления базами данных, что позволяет хранить в ней, к примеру, JSON (англ. JavaScript Object Notation, рус. нотация объектов JavaScript) объект и осуществлять поиск по значениям его ключей. Также PostgreSQL поддерживает индексирование по выражению, возможность реализовать который может оказаться полезным в случае, когда индексировать необходимо

не значение ячейки таблицы, а результат выполнения функции над данным значением.

В качестве СУБД, таким образом, было принято решение использовать PostgreSQL, так как она хорошо документирована, имеет большое сообщество разработчиков [9] и реализует объектно-ориентированный подход в хранении данных, что делает её более удобной для построения веб-приложения в рамках данной ВКР.

1.3.3 Бэкенд

Выбор инструментов бэкенд разработки главным образом был обусловлен наличием опыта разработки на языке программирования JavaScript.

С момента появления язык JavaScript прошёл долгий путь от браузерного языка до одного из важнейших языков во всех областях разработки программного обеспечения, на что существенно повлияло появление таких технологий, как Node.js, React.js и Electron [11]. Именно Node.js был выбран в качестве веб-сервера Journal System.

Node.js – это асинхронная управляемая событиями исполнительная платформа JavaScript. Node.js использует JavaScript-ядро Google V8, основанное на стандарте ES2015. Ядро JavaScript V8 было разработано для повышения скорости выполнения JavaScript за счёт JIT-компиляции (англ. Just-in-time compilation, рус. компиляция «на лету»). Ядро V8 написано на C++.

Одной из сильных сторон Node.js является однопоточная модель программирования: Node.js работает с моделью программных потоков, используемой в браузере. На стороне сервера подобный подход возможен благодаря трём концепциям Node.js: событиям, асинхронным API и неблокирующему вводу/выводу.

В качестве фреймворка для построения API был выбран Express.

Express – это лёгкий и гибкий Node.js фреймворк, предоставляющий широкий набор функций для мобильных и веб-приложений.

Одной из сильных сторон Express является его гибкость и расширяемость. Не существует «правильного способа» построения веб-приложения используя Express: программист сам волен определяться с архитектурой и способом взаимодействия компонентов системы.

Для работы бэкенда с PostgreSQL существует минимум три решения: ORM (англ. Object-Relational Mapping, рус. объектно-реляционное отображение) библиотека Sequelize, коллекция Node.js модулей node-postgres и библиотека pg-promise, разработанная на основе node-postgres. Из всех представленных инструментов Sequelize единственный предоставляет работу с данными не только с помощью стандартных SQL запросов, но и в терминах классов, а не таблиц, что позволяет существенно упростить реализацию MVC (англ. Model-view-controller, рус. модель-представление-контроллер) и разработку в целом.

Так как выбранный в качестве фронтенд-фреймворка React.js полностью исполняется на стороне клиента, а контент разрабатываемой веб-платформы состоит из выпусков журналов и статей, это может вызвать проблемы с краулингом (англ. crawling, т.е. процесс обнаружения и сбора поисковым роботом ссылок на другие страницы сайта) веб-страниц поисковыми сервисами и замедление первоначального рендера страницы [12]. В таком случае целесообразно производить рендер страниц на стороне сервера и отправлять готовый HTML код клиенту. Для решения данной задачи было решено использовать фреймворк Next.js.

Next.js – это фреймворк для серверного рендеринга React.js приложений. Next.js почти не требует изменения уже готовых веб-приложений, написанных на React.js, представляет рендер всех страниц на стороне сервера по умолчанию, автоматическое разделение кода для более быстрой загрузки страниц, простую постраничную навигацию на стороне клиента, интеграцию с Webpack и Express.

Next.js использует технику рендеринга, называемую регидратацией. Такой подход позволяет устранить минусы серверного и клиентского рендеринга, объединяя оба подхода. Сервер выдаёт зарендеренную HTML страницу клиенту, а затем на стороне клиента подгружается JavaScript и данные, необходимые для

работы веб-приложения, что позволяет реализовать интерактивность React.js приложения при серверном рендеринге [13].

1.4 Методология разработки

Гибкая методология разработки (англ. agile software development) является самым популярным подходом к разработке программного обеспечения (ПО) в настоящее время [14]. Одним из способов реализации данного подхода является канбан (англ. kanban). Данный метод использовался при разработке Journal System.

Изначально канбан разрабатывался в компании Toyota в рамках концепции бережливого производства, однако затем нашёл себе применение и в сфере разработки ПО [15].

Канбан базируется на трёх основных принципах:

- визуализация процесса разработки;
- ограничение числа текущих задач;
- измерение и оптимизация жизненного цикла разработки.

Для визуализации канбан-доски удобно использовать программу для управления проектами Trello, которая по умолчанию использует парадигму канбан. Скриншот канбан-доски представлен на рисунке 1.

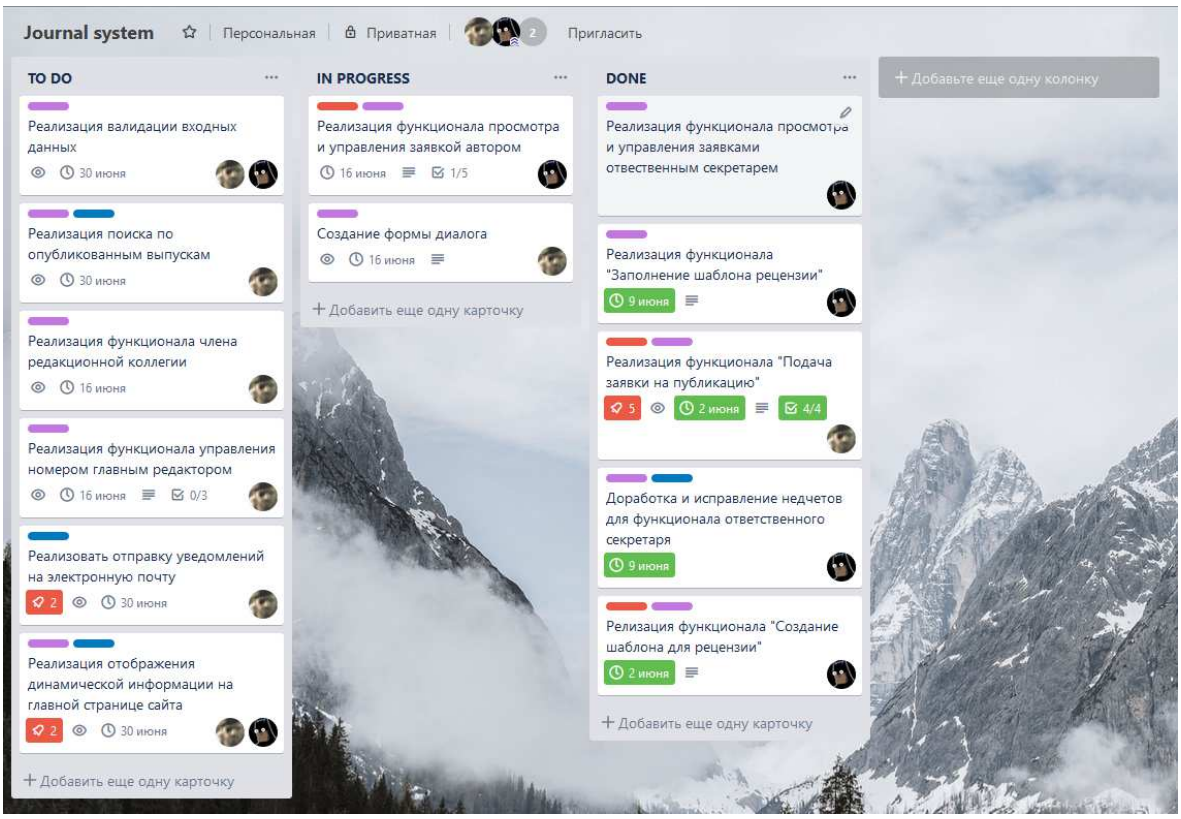

Рисунок 1 – Канбан-доска проекта в системе Trello

При разработке веб-платформы использовалась система контроля версий Git, репозиторий размещался на хостинге Github. Использование Git позволило разрабатывать бэкенд и фронтенд на разных ветвях (англ. branch, указателях на состояние системы) независимо друг от друга, а затем произвести слияние.

Git является распределённой системой, что означает, что в один момент времени главная копия проекта может находиться на нескольких машинах. Git позволяет как легко просматривать изменения между версиями, так и быстро перемещаться между ними, совершать откаты, слияния и прочее [16].

1.5 Выводы по разделу

Целью раздела являлось изучение предметной области, в результате которого были проанализированы аналоги разрабатываемого веб-приложения. Было выявлено, что не существует ни одного бесплатного готового решения «под ключ» для реалий бизнес-логики российских научных журналов.

Были выбраны средства разработки и полностью определён технологический стек: React – библиотека построения фронтенда, Material UI – набор графических компонентов, PostgreSQL – СУБД, Node.js – веб-сервер, Express – фреймворк построения API, Sequelize – ORM библиотека для связи с базой данных и Next.js – фреймворк для серверного рендеринга React приложений.

Также в качестве методологии разработки был выбран канбан. В качестве системы контроля версий было решено использовать Git и разместить репозиторий на платформе Github.

2 Проектирование

2.1 Архитектура информационной системы

Существует множество определений архитектуры, в частности архитектуры программного обеспечения. Хотя определения и отличаются друг от друга, все они в основном указывают на то, что архитектура связана со структурой и поведением только со значимыми решениями, может иметь какой-либо стиль, на неё влияют окружение и заинтересованные лица, она воплощает решения на основе логического обоснования [17].

Архитектура информационной системы – концепция, определяющая модель, структуру, выполняемые функции и взаимосвязь компонентов информационной системы [18]. Говоря об архитектуре веб-приложений, выделяют классическую архитектуру клиент-сервер и многоуровневые архитектуры клиент-сервер.

Клиент-сервер – это архитектура приложения, в которой его функциональные части взаимодействуют по схеме запрос-ответ. Двумя взаимодействующими частями такой архитектуры являются клиент (выполняет активную функцию, т.е. инициирует запросы) и сервер (отвечает на поступающие запросы). Классическая архитектура клиент-сервер подразумевает распределение трёх основных частей веб-приложения (интерфейс взаимодействия с пользователем, серверную логику и хранилище данных) по двум физическим модулям, в чём и заключается основной недостаток данной архитектуры [19]. Архитектура представлена визуально на рисунке 2.

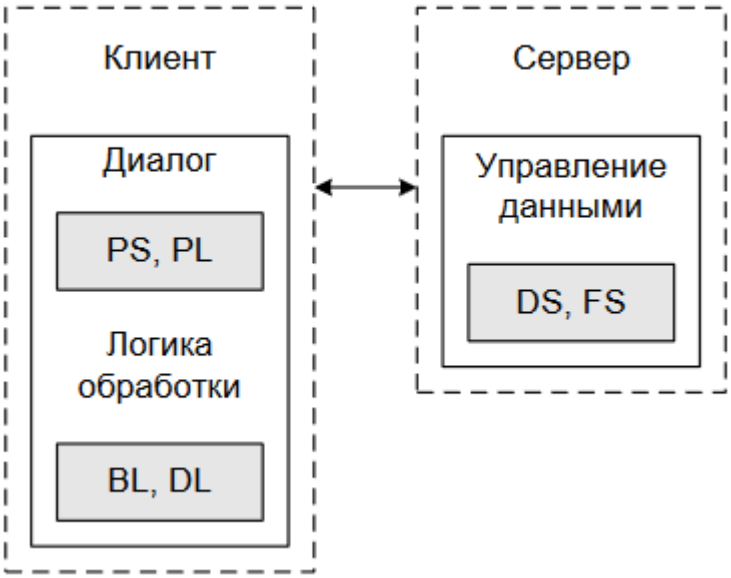

Рисунок 2 – Архитектура клиент-сервер, где PS – средства представления (англ. Presentation Services), PL – логика представления (англ. Presentation Logic), BL –прикладная логика (англ. Business Logic), DL – логика управления данными (англ. Data Logic), DS – операции с базой данных (англ. Data Services) и FS – файловые операции (англ. File Services)

Многоуровневая архитектура клиент-сервер – это разновидность описанной выше двухуровневой архитектуры, в которой обработка данных распределена между одним или несколькими серверами, что позволяет эффективно разделить функции хранения, обработки и представления данных. Данная архитектура реализует более эффективный подход к использованию возможностей как серверов, так и клиентов [20]. Одной из разновидностей многоуровневой архитектуры является трёхуровневая архитектура.

Трёхуровневая архитектура – это архитектура приложения, разделённая на три основных компонента: клиент, сервер приложений и сервер баз данных. Архитектура представлена визуально на рисунке 3.

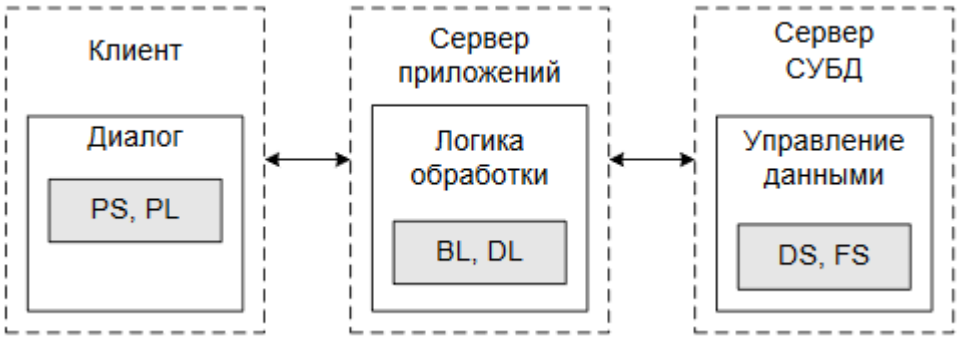

Рисунок 3 – Трёхуровневая архитектура, сокращения соответствуют сокращениям в описании рисунка 2

В рамках разработки Journal System было принято решение разработать платформу в соответствии с принципами трёхуровневой архитектуры по следующим причинам [21]:

- упрощение масштабируемости;
- упрощение реализации полномочий пользователей за счёт переноса прикладной логики на серверную сторону;
- более высокий уровень безопасности по сравнению с двухуровневой архитектурой.

Другой особенностью определения архитектуры разрабатываемой системы является используемый шаблон проектирования. Одним из наиболее распространённых шаблонов в веб-разработке является MVC [22], который было решено использовать при разработке Journal System.

MVC – это шаблон проектирования веб-приложений, включающий в себя несколько более мелких шаблонов. MVC разделяет данные, прикладную логику и пользовательский интерфейс на три компоненты: модель, представление, контроллер [23].

Модель – это центральный компонент шаблона, структура данных, независимая от пользовательского интерфейса. Модель реагирует на запросы из контроллера и возвращает или меняет своё состояние соответствующе.

Представление – это любая визуализация информации, включая таблицы, диаграммы и графики. С представлением взаимодействует пользователь.

Контроллер – это интерпретатор действий пользователя, который принимает запросы пользователя и преобразовывает их в команды для модели или представления. Как правило, фильтрация данных и авторизация пользователя так же является функционалом контроллера.

Общая схема MVC представлена на рисунке 4.

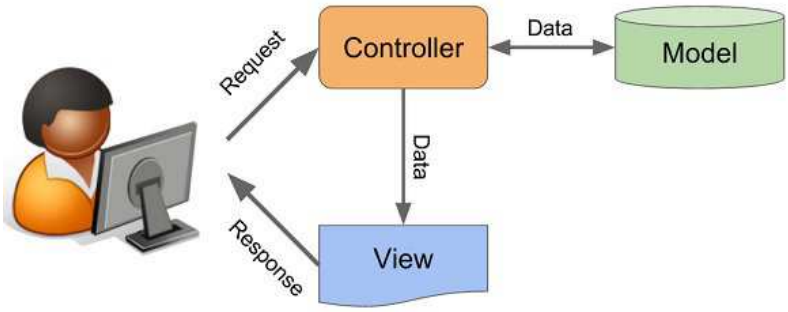

Рисунок 4 – Схема шаблона проектирования MVC

2.2 База данных

Проектирование базы данных является такой же важной и комплексной задачей, как и проектирование остальных компонентов системы. Определение структуры проектируемой базы, является одной из первостепенных задач. Структура базы данных – это принцип или порядок организации записей в базе данных и связей между ними [24].

В разрабатываемой веб-платформе используется реляционная модель представления данных, что означает, что данные имеют предопределённые связи между собой и организуются в виде таблиц, состоящих из столбцов и строк. Таблицы хранят информацию об объектах, представленных в базе данных.

Всего в базе данных насчитывается 30 таблиц. В таблице 1 содержатся наименования и описания таблиц системы.

Таблица 1 – Описание таблиц базы данных

№	Наименование	Описание
1	user	Зарегистрированный пользователь системы.
2	role	Роль пользователя системы.
3	user_role	Таблица, устраняющая отношение «многие ко многим» между пользователем и его ролями.
4	credentials	Персональный данные всех зарегистрированных пользователей системы.
5	country	Список стран на русском и английском языках.
6	submission	Метаданные статьи, поданной в заявке на публикацию.
7	submission_file	Файлы, прикрепленные к заявке на публикацию.
8	file_type	Типы файлов, которые можно прикрепить к заявке на публикацию.
9	submission_history	История заявки на публикацию.
10	submission_status	Статус заявки на публикацию.
11	author_submission	Авторы статьи, поданной в заявке на публикации.
12	user_organization	Организации зарегистрированных пользователей системы, а также авторов статей, не являющихся пользователями системы.
13	user_submission	Таблица, устраняющая отношение «многие ко многим» между пользователем системы и его заявками на публикацию.
14	submission_keyword	Таблица, устраняющая связь «многие ко многим» между статьей и ключевыми словами.
15	keyword	Ключевые слова опубликованной статьи.
16	review	Рецензия на статью, отправленную в заявке на публикацию.
17	review_status	Статус рецензии.

Окончание таблицы 1

№	Наименование	Описание
18	recommendation	Рекомендация рецензента к рецензируемой статье.
19	message	Сообщения между автором (пользователем системы) и ответственным секретарем.
20	message_file	Файлы, прикрепленные к сообщению.
21	issue	Выпуск журнала.
22	issue_status	Статус выпуска журнала.
23	issue_submission	Статьи выпуска журнала.
24	issue_decision	Решение члена редакционной коллегии о пригодности номера к публикации.
25	volume	Том журнала.
26	academic_degree	Список ученых степеней.
27	academic_title	Список ученых званий.
28	review_template	Шаблоны для рецензий.
29	file	Таблица с файлами.
30	language	Язык статьи, также ключевых слов.
31	subscription	Таблица, содержащая информацию о подписке пользователя на определённые почтовые рассылки.
32	user_subscription	Таблица, устраняющая отношение «многие ко многим» между пользователем и его подписками.

Отношения таблиц и их поля удобно представить графически в виде схемы. Схема была логически разделена на 4 части и представлена на рисунках 5–8.

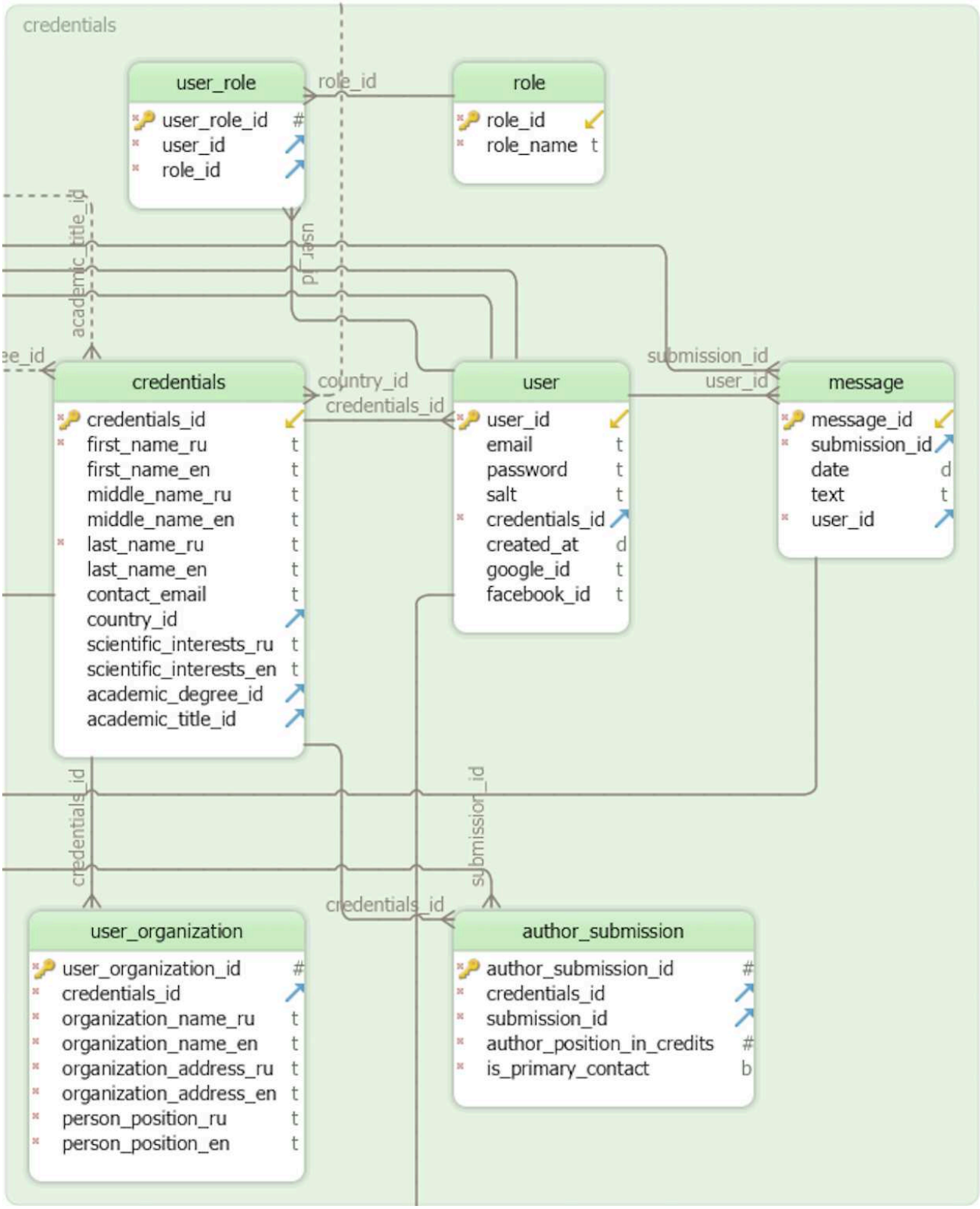

Рисунок 5 – Часть 1 схемы базы данных.

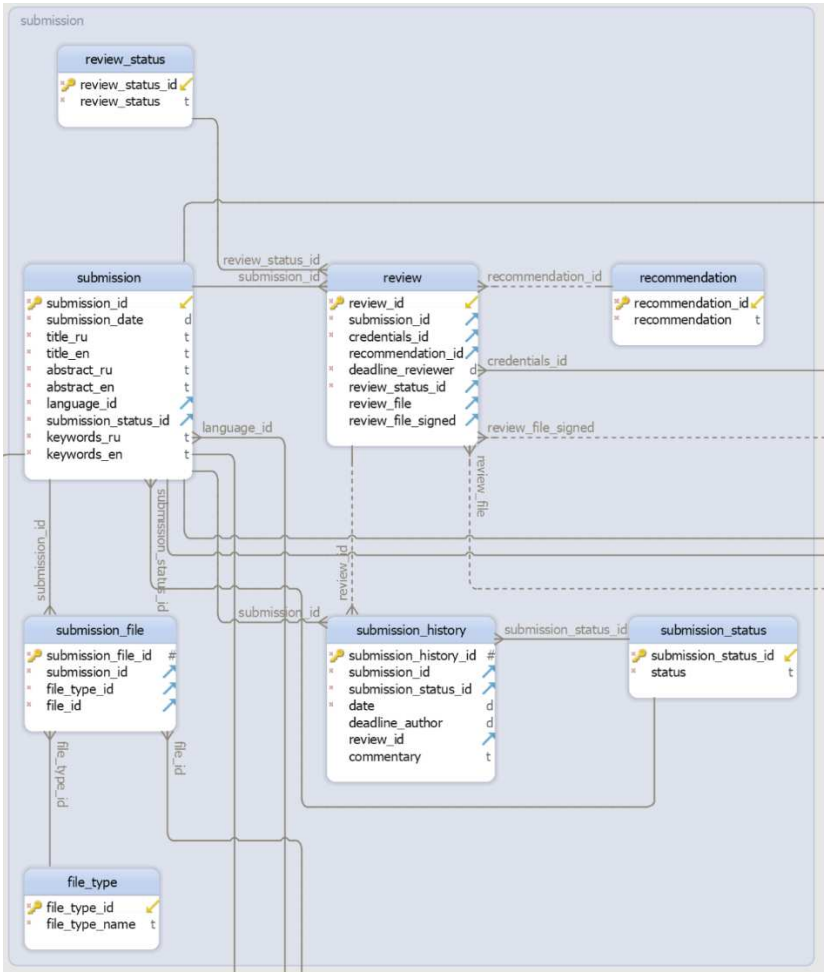

Рисунок 6 – Часть 2 схемы базы данных

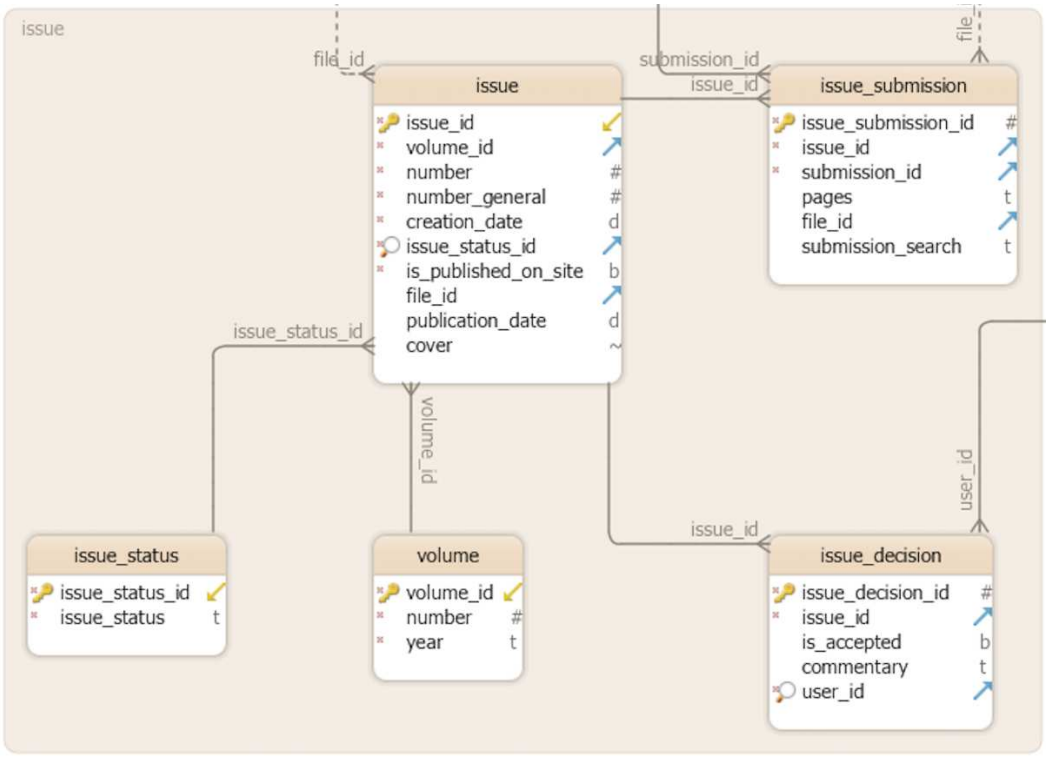

Рисунок 7 – Часть 3 схемы базы данных

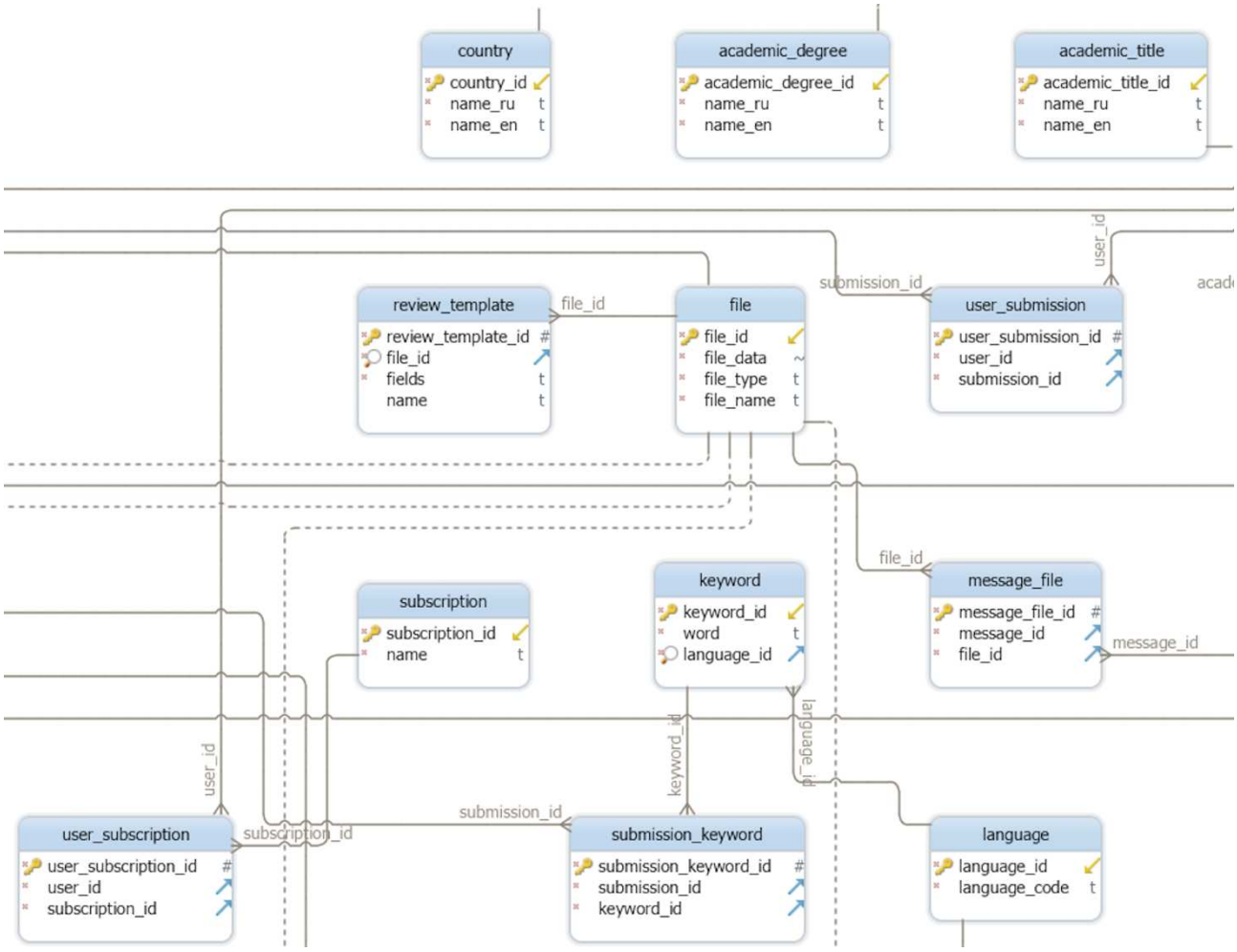

Рисунок 8 – Часть 4 схемы базы данных

2.3 Интерфейс веб-приложения

В настоящее время успех веб-сайта во многом зависит от его интерфейса. Грамотно спроектированный интерфейс лёгок в освоении, его основные функции вынесены на передний план, а лишние спрятаны. Расположение элементов управления влияет на удобство работы с сайтом, а его внешний вид на первое впечатление и удовольствие от использования [25].

С развитием сети Интернет и веб-технологий у пользователей складывались определённые представления работы с веб-сайтами. Подобными представлениями могут быть: расположение навигационной панели, кнопок регистрации и входа в систему в шапке сайта, контактной информации, правил и прочего в футере страницы и многие другие. Данные особенности определённо

необходимо учитывать при разработке такого веб-приложения, как сайт научного журнала.

Так как проблемы с определением целевой аудитории отсутствуют (учёные и образованные люди с различными областями научных интересов), то, основываясь на этом, можно выделить основные требования к интерфейсу:

- лаконичность и простота;
- использование нейтральных цветов, не вызывающих неприятных или тревожных чувств;
- использование ненавязчивой, «мягкой» анимации графических компонентов;
- наличие англоязычной версии сайта;
- адаптация интерфейсов под различные размеры экранов мобильных устройств в следствие роста их распространённости на рынке [26].

Более того, к сайтам научных журналов предъявляются другие разносторонние требования, касающиеся, как содержания веб-страниц журнала, так и их визуального представления [27, 28]. Требования касаются структуры интерфейса сайта, наличия метаданных в HTML-коде страниц, отсутствия рекламы, явных указаний на правила и политику журнала.

В ходе разработки прототипа интерфейса веб-платформы, были созданы наброски страниц приложения. Макет главной страницы представлен на рисунке 9.

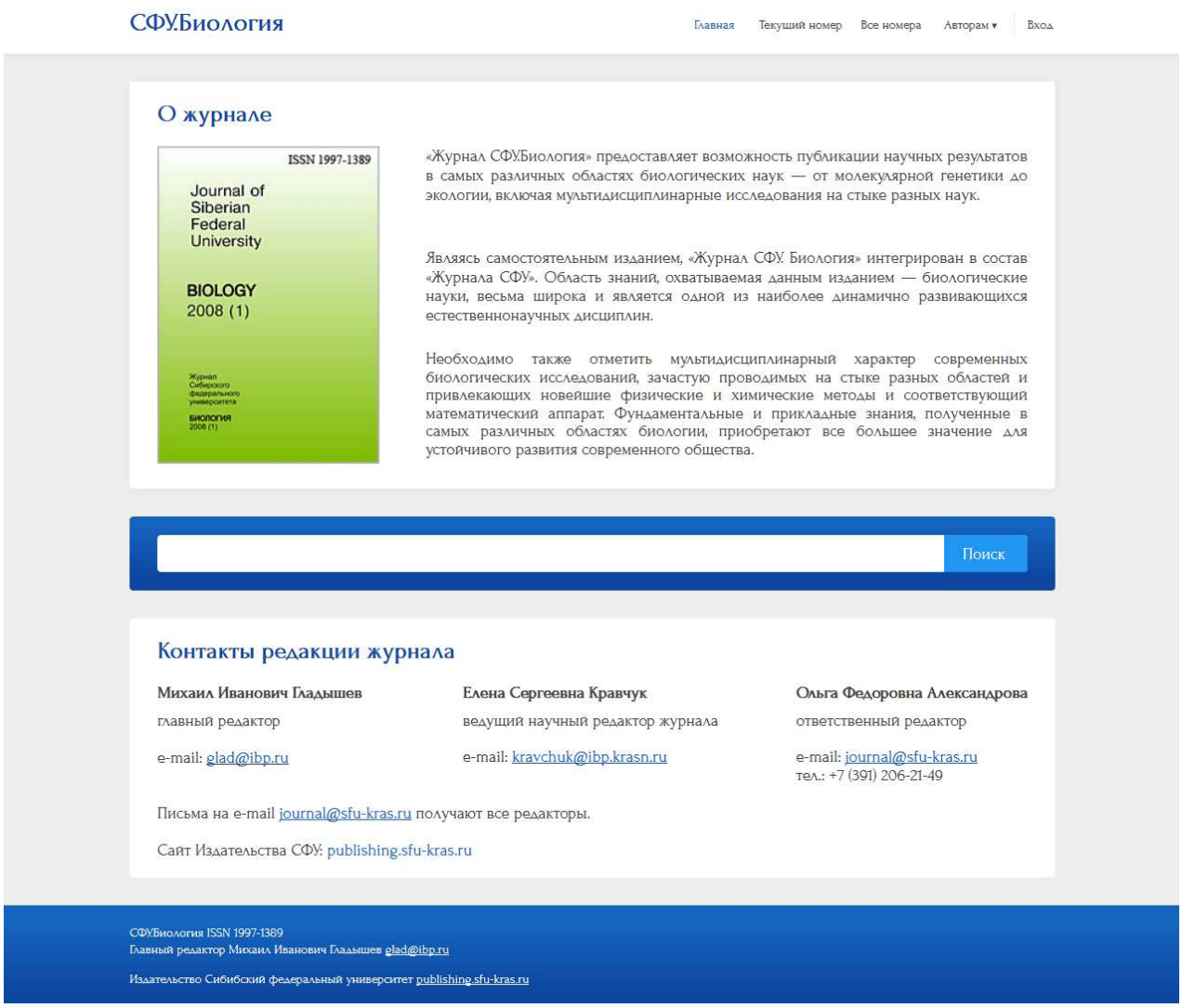

Рисунок 9 – Макет главной страницы сайта

2.4 Диаграммы вариантов использования

Диаграмма вариантов использования (англ. use case diagram) – это диаграмма, на которой изображаются актёры и варианты использования (также называются прецедентами) выполненная с использованием унифицированного языка моделирования UML (англ. Unified Modeling Language). Диаграммы используются для определения взаимодействия пользователя с системой, описывая какие действия может осуществлять актёр, но не каким образом [29].

Разработанные диаграммы вариантов использования Journal System представлены на рисунках 10–13.

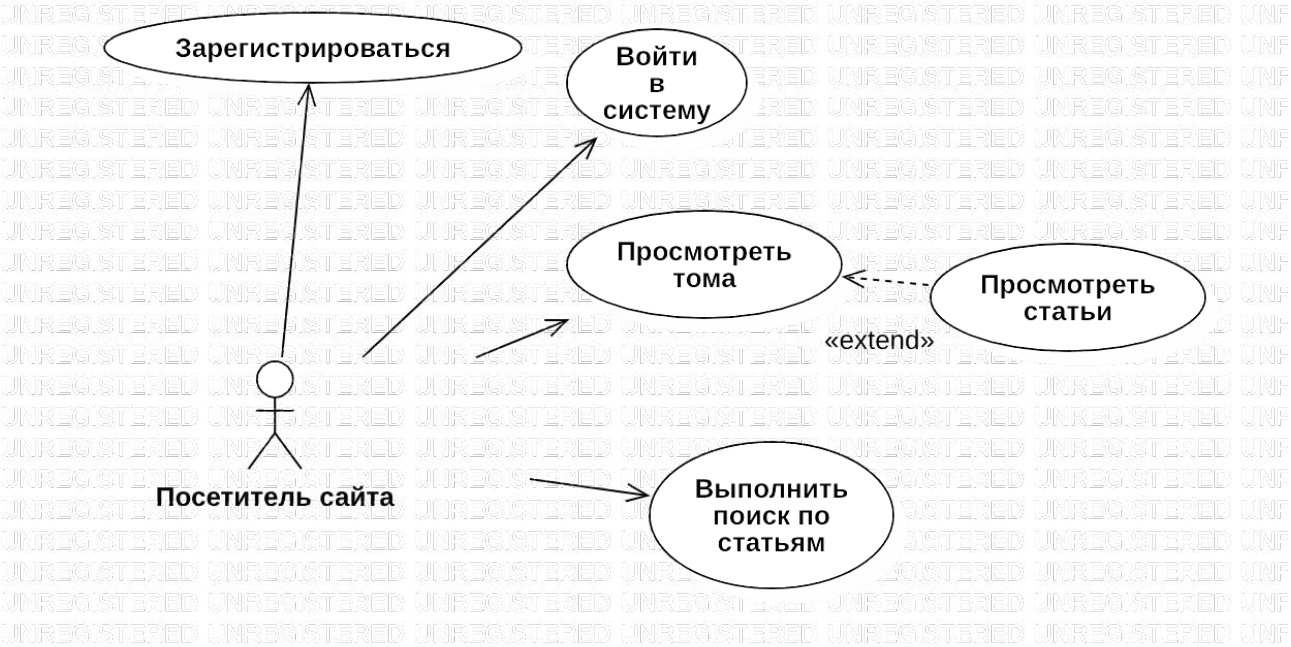

Рисунок 10 – Диаграмма вариантов использования посетителя сайта

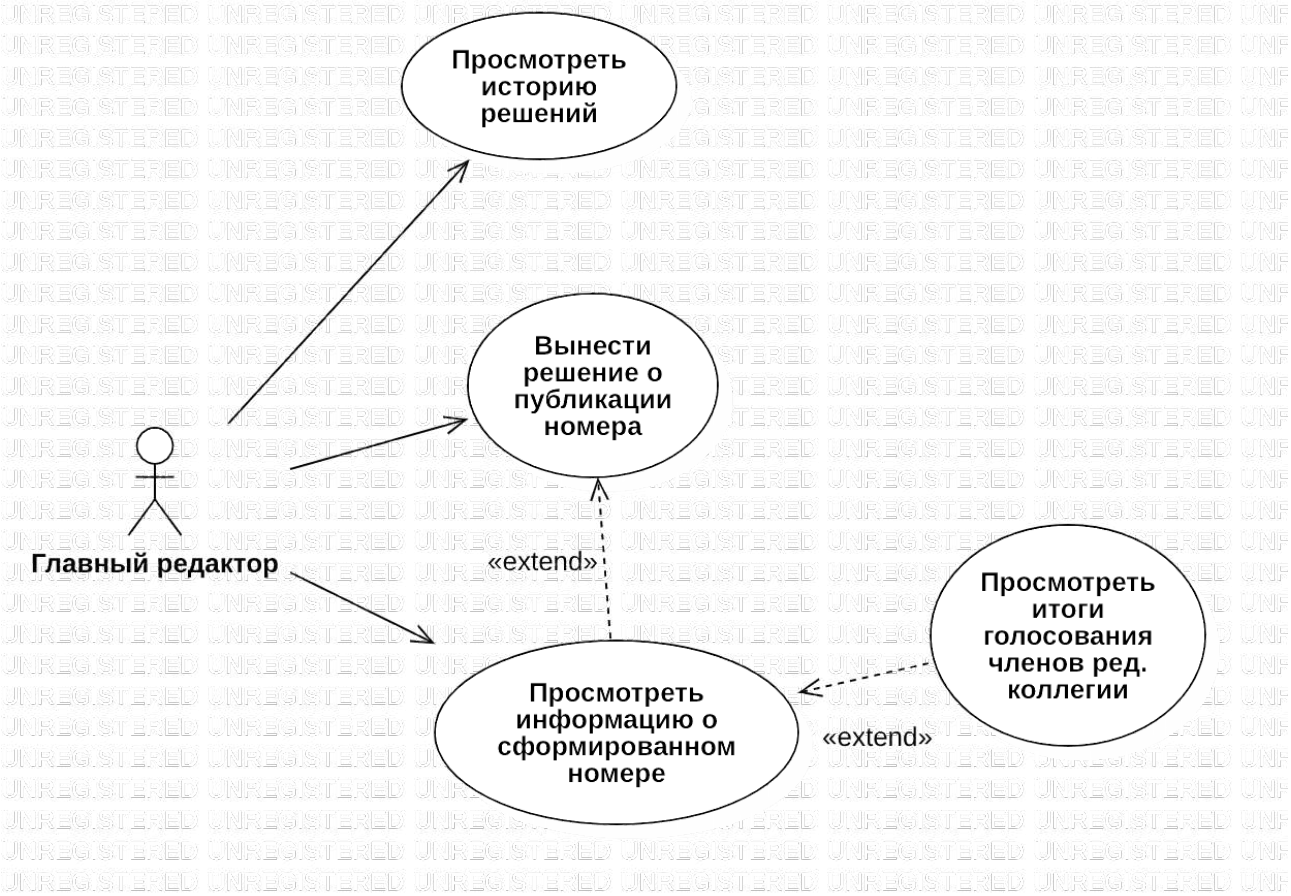

Рисунок 11 – Диаграмма вариантов использования главного редактора

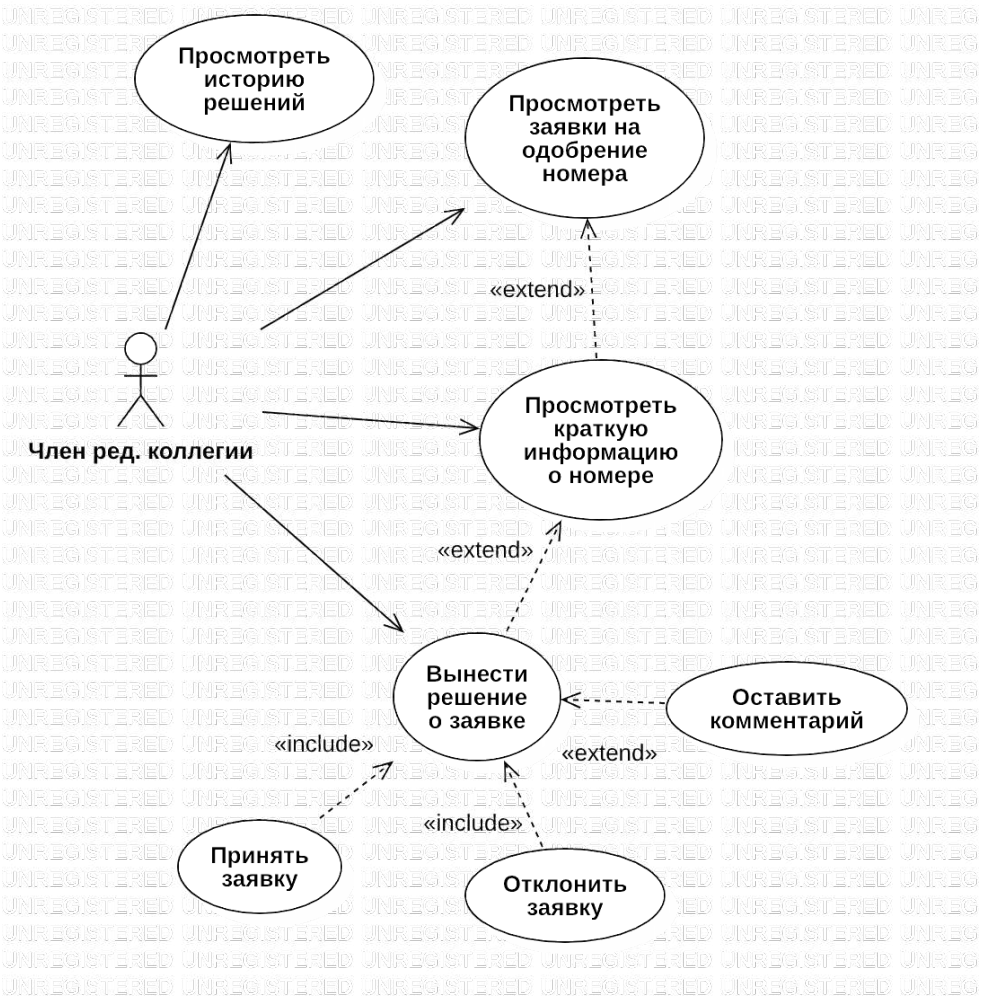

Рисунок 12 – Диаграмма вариантов использования члена редакционной коллегии

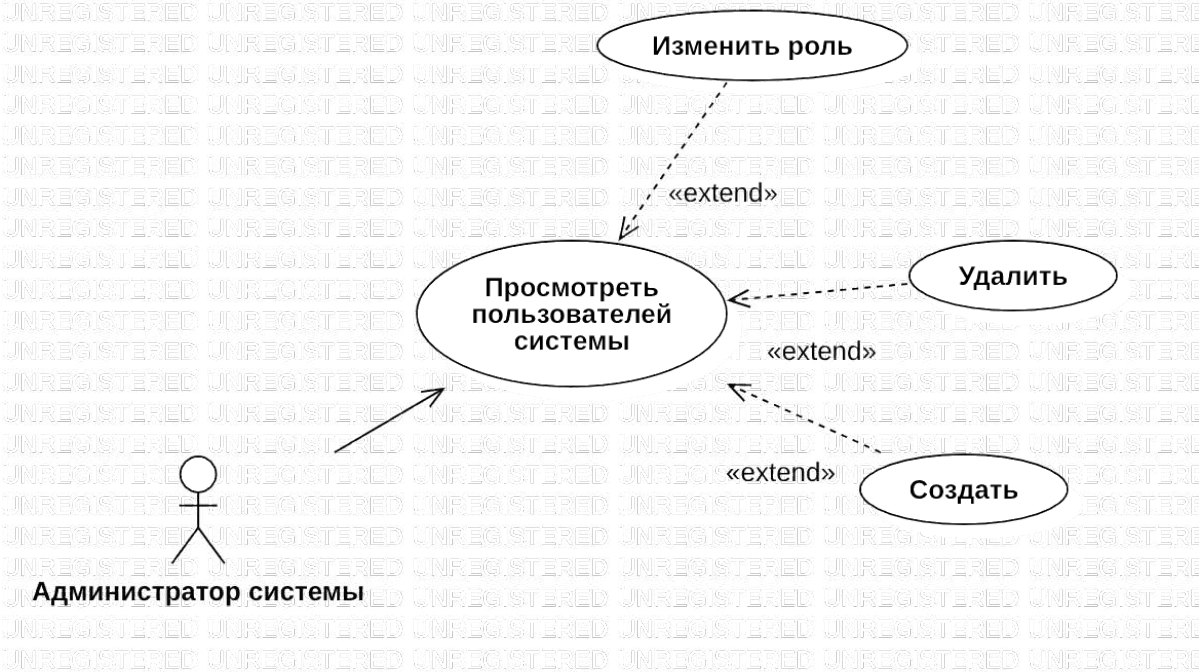

Рисунок 13 – Диаграмма вариантов использования администратора системы

2.5 Выводы по разделу

Целью раздела являлось описание проектирования разрабатываемой веб-платформы. Были описаны архитектуры веб-приложений, представлены их особенности, в результате чего было решено разрабатывать приложение по трёхуровневой архитектуре. Также был описан используемый шаблон MVC.

Была описана база данных, таблицы базы данных и связи между ними.

Также были разработаны модели вариантов использования для посетителя веб-сайта, администратора, главного редактора и члена редакционной коллегии.

Кроме того, были представлены требования к интерфейсу веб-приложения, основанные на современных тенденциях и подходах к построению сайтов научных журналов.

3 Разработка

Разработка клиентской и серверной частей велась в редакторе исходного кода Visual Studio Code (VS Code), так как VS Code является бесплатным, нетребовательным к ресурсам, поддерживает множество языков программирования, автодополнение, подсветку синтаксиса, отладку программ, плагины и прочее.

Частью начала разработки является определение файловой структуры проекта. Структура представлена на рисунке 14.

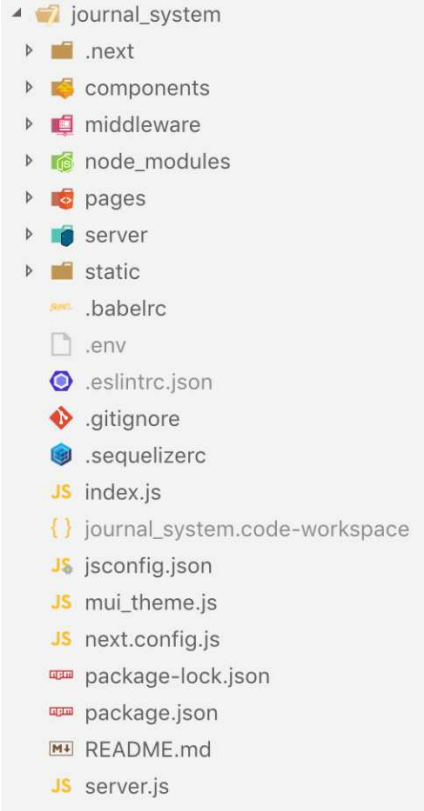

Рисунок 14 – Файловая структура проекта

Файлы инициализации и настроек находятся в корневом каталоге.

Корневой каталог включает следующие директории:

- next – создаётся и управляется одноимённым фреймворком;
- node_modules – создаётся менеджером пакетов npm, хранилище установленных пакетов;

– components – React-компоненты;
– middleware – методы фронтенда, а также промежуточный код, необходимый, как клиентской, так и серверной стороне;
– pages – страницы веб-сайта;
– server – файлы, относящиеся только к серверной стороне, такие как: контроллеры, модели и прочее;
– static – статичные файлы, предназначенные для наполнения веб-страниц.

3.1 Бэкенд

Первым этапом разработки серверной стороны являлось определение объектных моделей с помощью ORM библиотеки Sequelize. Для этого в директории server был создан файл настроек sequelizerc и описаны модели в папке models. Пример модели, основанной на таблице ролей пользователей, представлен на рисунке 15.

```
module.exports = (sequelize, DataTypes) => {
  const Role = sequelize.define('role', {
    role_id: {
      allowNull: false,
      autoIncrement: true,
      primaryKey: true,
      type: DataTypes.SMALLINT
    },
    role_name: {
      type: DataTypes.STRING,
      allowNull: false
    }
  }, {
    freezeTableName: true,
    timestamps: false
  });
  Role.associate = models => {
    Role.belongsTo(models.UserRole, {
      foreignKey: 'role_id',
      targetKey: 'role_id'
    });
  };
  return Role;
};
```

Рисунок 15 – Модель Role

Библиотека Sequelize предоставляет возможность реализации миграций и предварительного заполнения базы данных для упрощения развёртывания приложения.

В директории server также были созданы папки migrations и seeders. Файлы migrations описывают создание таблиц и их полей. Пример файла миграции представлен на рисунке 16.

```
module.exports = {
  up: (queryInterface, Sequelize) => {
    return queryInterface.createTable('role', {
      role_id: {
        allowNull: false,
        autoIncrement: true,
        primaryKey: true,
        type: Sequelize.SMALLINT
      },
      role_name: {
        type: Sequelize.STRING,
        allowNull: false,
      }
    });
  },
  down: (queryInterface, Sequelize) => {
    return queryInterface.dropTable('role');
  }
};
```

Рисунок 16 – Файл миграции для таблицы ролей пользователя

Файлы миграции были написаны для каждой таблицы базы данных, что позволяет описать базу данных одной командой в терминале.

В разрабатываемой веб-платформе существуют такие таблицы, содержание которых должно быть предопределено (например, названия стран, ролей системы). Для выполнения данной задачи была создана директория seeders, содержимое которой описывает содержимое таблицы базы данных. Пример функции, выполняющей предварительное заполнение таблицы ролей, представлен на рисунке 17. Вызов операции автозаполнения также прост, как и миграции.

```
module.exports = {
  up: (queryInterface, Sequelize) => {
    return queryInterface.bulkInsert('role', [
      { role_name: 'Администратор' },
      { role_name: 'Автор' },
      { role_name: 'Ответственный секретарь' },
      { role_name: 'Главный редактор' },
      { role_name: 'Рецензент' },
      { role_name: 'Член редакционной коллегии' }
    ], {});
  },

  down: (queryInterface, Sequelize) => {
    return queryInterface.bulkDelete('role', null, {});
  }
};
```

Рисунок 17 – Функция, заполняющая таблицу ролей системы

Следующий этап заключался в написании функций, реализующих прикладную логику разрабатываемой веб-платформы. Данным функциям была выделена директория methods. Файлы из директории methods содержат экспортируемые функции, используемые контроллерами.

Прежде чем реализовывать контроллеры, было необходимо составить карту API-путей веб-платформы. Каждой роли системы был выделен собственный путь для запросов. При создании названий путей использовались только существительные, так, например, путь получения администратором всех пользователей системы это /api/admin/users, а получение одного пользователя системы это /api/admin/user/:slug, где slug – уникальный идентификатор пользователя. Подобное наименование упрощает понимание и использование API [30].

Работа с созданными путями осуществляется с помощью объекта Router фреймворка Express. В директории controllers был создан файл index.js (рисунок 18), реализующий направление запросов в соответствующие контроллеры согласно API-путям.

```
function api(server) {
  server.use(PUBLIC_BASE_PATH, publicApi);
  server.use(ADMIN_BASE_PATH, authorize(ROLES.ADMIN), adminApi);
  server.use(AUTHOR_BASE_PATH, authorize(ROLES.AUTHOR) ,authorApi);
  server.use(SECRETARY_BASE_PATH, authorize(ROLES.SECRETARY), secretaryApi);
  server.use(EDITOR_BASE_PATH, authorize(ROLES.EDITOR), editorApi);
  server.use(REVIEWER_BASE_PATH, authorize(ROLES.REVIEWER), reviewrApi);
  server.use(EDITORIAL_BOARD_MEMBER_BASE_PATH, authorize(ROLES.EDITORIAL_BOARD_MEMBER), editorialBoardMemberApi);
  server.use(ACCOUNT_BASE_PATH, authorize(), accountApi);
  server.use(COMMON_BASE_PATH, authorize(ROLES.SECRETARY, ROLES.AUTHOR, ROLES.REVIEWER), commonApi);
}
```

Рисунок 18 – Фрагмент кода файла index.js в директории controllers

Контроллеры, в свою очередь, принимают запрос пользователя и выполняют функции, отвечающие за бизнес-логику.

Пример контроллера представлен на рисунке 19.

```
const router = express.Router();

router.get(ADMIN_USERS, async (req, res) => {
  try {
    const result = await getUsers();
    res.status(200).json(result);
  } catch (err) {
    res.status(err.status || 500).json({ message: err.message || err.toString() });
  }
});

router.post(ADMIN_NEW_USER, async (req, res) => {
  try {
    await createUser({ data: req.body, isAdmin: true });
    res.status(200).end();
  } catch (err) {
    res.status(err.status || 500).json({ message: err.message || err.toString() });
  }
});

router.put(ADMIN_USER, async (req, res) => {
  try {
    await editUser(req.params.slug, req.body);
    res.status(200).end();
  } catch (err) {
    res.status(err.status || 500).json({ message: err.message || err.toString() });
  }
});

router.delete(ADMIN_USER, async (req, res) => {
  try {
    await deleteUser(req.params.slug);
    res.status(200).end();
  } catch (err) {
    res.status(err.status || 500).json({ message: err.message || err.toString() });
  }
});
```

Рисунок 19 – Контроллер, отвечающий за функции администратора

Благодаря подобной реализации бэкенда контроллеры выглядят минималистично и понятно, а их добавление и изменения в карте API-путей не требуют большого количества действий.

Так как Journal System разделяет функционал пользователей согласно присвоенным им ролям, было необходимо разработать механизм аутентификации и авторизации. Для выполнения данной задачи, было решено использовать библиотеку Passport.js, которая предоставляет упрощённый подход к реализации так называемых «стратегий» аутентификации.

Для работы с библиотекой в директории server был создан файл конфигурации passport.js, в котором были описаны «стратегии» локальной аутентификации (через электронный адрес и пароль) и через сторонние системы Google и Facebook (для которых потребовалось получить ключи клиента и секретные идентификаторы приложения на порталах разработчика данных систем).

Чтобы идентифицировать клиента, веб-платформа помещает куки в браузер пользователя (что выполнено с использованием библиотеки cookie-session, которая не хранит куки на стороне сервера, а только на клиенте). Для хранения и извлечения информации, хранимой в куки, были разработаны методы serializeUser и deserializeUser. Функция serializeUser, используя заданный секрет, кодирует идентификатор пользователя в куки и сохраняет их в браузере клиента. При обращении клиента к серверу, сервер получает куки, Passport.js с помощью метода deserializeUser извлекает хранимый в куки уникальный идентификатор, проверяет его валидность, а также обращается к базе данных для установления прав пользователя.

Авторизация пользователя выполнена в функции authorize (рисунок 20).

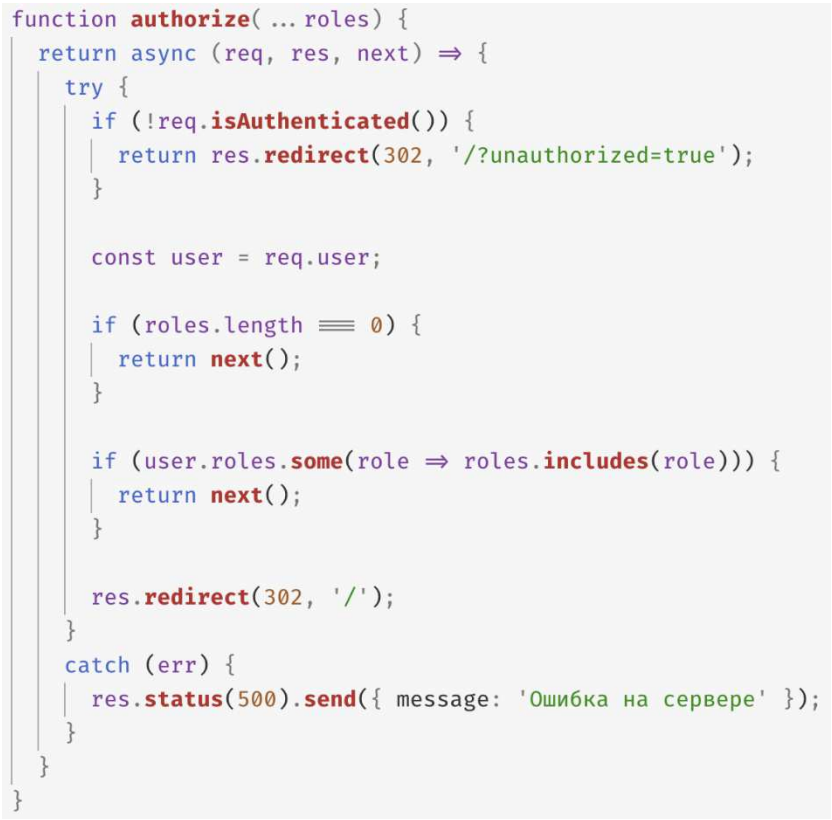

Рисунок 20 – Функция авторизации

Функция авторизации служит как промежуточная функция и получает на входе массив ролей, авторизованных для выполнения операции. Если роли не указаны, то считается, что авторизованы все аутентифицированные запросы. Пример использования данной функции был представлен в фрагменте кода рисунка 18.

3.2 Фронтенд

Начальным этапом разработки клиентской части веб-платформы являлось создание React-компонентов, из которых строятся страницы сайта. В директории components были созданы папки main и dashboard. Первая содержит компоненты, относящиеся только к страницам, видимым всем посетителям (например, главная страница журнала, страница номера, статьи и прочее), вторая – компоненты, используемые на внутренних страницах сайта, доступных только

авторизованным пользователям. Промежуточные компоненты, используемые на всех страницах, располагаются в корневом каталоге папки components.

Пример компонента представлен на рисунке 21. Компонент Layout служит в качестве разметки для основных страниц веб-платформы.

```
const useStyles = makeStyles(theme => ({
  grow: {
    flexGrow: 1
  },
  body: {
    minHeight: '100vh',
    display: 'flex',
    flexDirection: 'column',
    backgroundColor: theme.palette.secondary.main
  },
  content: {
    flex: '1 0 auto'
  }
}));

function Layout(props) {
  const classes = useStyles();
  const { isAuthenticated, value, children, loginOpen } = props;

  return (
    <div className={classes.body}>
      <NavBar isAuthenticated={isAuthenticated} value={value} loginOpen={loginOpen} />
      <div className={classes.content}>
        <Grid container className={classes.grow} justify="center">
          <Grid container item xs={10}>
            { children }
          </Grid>
        </Grid>
      </div>
      <Footer />
    </div>
  );
}
```

Рисунок 21 – Компонент Layout

Следует отметить, что при создании компонентов React использовались «хуки» (англ. Hooks), нововведение библиотеки версии 16.8. Использование «хуков» требует разработки компонентов в виде функций JavaScript, а не классов, что делает код более лаконичным и ёмким [31].

Использование «хуков» позволило просто и ёмко реализовать механизм получения информации об авторизованном пользователе на страницах системы.

Для этого в файле auth.js (рисунок 22) директории middleware был создан объект типа Context.

```
import { createContext } from 'react';

const Auth = createContext();

export default Auth;
```

Рисунок 22 – Файл auth.js

Далее в файл _app.js (файл Next.js для формирования всех страниц сайта) импортировался объект Auth, в параметр value метода Auth.Provider передавался полученный объект пользователя и содержимое страницы оборачивалось в вышеприведённый метод (рисунок 23).

```
<Auth.Provider value={user}>
  <Component {...pageProps} />
</Auth.Provider>
```

Рисунок 23 – Передача данных о пользователе в файле _app.js

Получение данных о пользователе со страниц системы осуществляется простым вызовом «хука» useContext (рисунок 24).

```
const { isAuthenticated, roles } = useContext(Auth);
```

Рисунок 24 – Пример получения информации о пользователе

Также на рисунке 21 продемонстрировано использование функции makeStyles библиотеки Material UI. Объект theme, передаваемый в качестве параметра в функцию обратного вызова, содержит информацию об

установленных значениях отступов, основных и вторичных цветах и прочие косметические особенности. Данный объект описан в файле mui_theme.js (рисунок 25), находящемся в корневом каталоге репозитория Journal System.

```
const theme = createMuiTheme({
  palette: {
    primary: {
      main: '#6f79a8', //#00796b
      light: '#9fa8da',
      dark: '#424d79'
    },
    secondary: {
      main: '#fafafa',
      light: '#ffffff',
      dark: '#c7c7c7'
    },
    error: {
      main: '#e57373',
      dark: '#ef5350'
    },
    status: {
      accepted: green[600],
      rejected: red[600],
      new: blue[800],
      other: indigo[800]
    },
    text: {
      hint: "#c7c7c7"
    },
    icon: {
      main: grey[600]
    }
  },
  typography: {
    body2: {
      fontSize: '1rem',
    },
    body1: {
      fontSize: '0.875rem',
    },
    h6: {
      color: 'rgba(0, 0, 0, 0.87)'
    }
  },
  status: {
    success: green[400],
    error: red[400],
    info: indigo[400],
    warning: orange[400]
  }
});
```

Рисунок 25 – Объект theme файла mui_theme.js, содержащий информацию о косметических параметрах веб-приложения

В случае необходимости, данная особенность делает изменение облика веб-сайта достаточно простым заданием, требующим редактирования нескольких строк файла mui_theme.js.

Чтобы упростить отслеживание ошибок при разработке компонентов, было принято решение использовать библиотеку PropTypes, выполняющую валидацию передаваемых параметров. Пример использования представлен на рисунке 26.

```
Layout.propTypes = {
  value: PropTypes.number,
  children: PropTypes.oneOfType([
    PropTypes.object, PropTypes.array
  ]).isRequired,
  isAuthenticated: PropTypes.bool.isRequired,
  loginOpen: PropTypes.bool
};
```

Рисунок 26 – Использование PropTypes в компоненте Layout

Если получаемые компонентом параметры определены, библиотека отправляет уведомление в консоль браузера при несовпадении полученного типа параметра с описанным.

Для хранения методов обращения к серверной стороне в директории middleware была создана папка api. Пример функции получения персональных данных представлен на рисунке 27.

```
export const getCredentials = cookie =>
  sendRequest(ACCOUNT_BASE_PATH + ACCOUNT_CREDENTIALS, {
    method: 'GET',
    headers: requestHeaders(cookie),
    credentials: 'include'
  });
```

Рисунок 27 – Функция запроса персональных данных со стороны клиента

Динамическое заполнение метаданных страниц в теге <head> было реализовано с помощью методов библиотеки Next.js. Так, например, страница статьи для лучшей индексации поисковыми сервисами, в частности системой Google Scholar должна содержать метатеги с информацией об авторах и статье [32]. Наполнение тега <head> страницы статьи представлено на рисунке 28.

```
<Head>
  <title></title>
  <meta name="citation_title" content={`${article.submission.title_ru} ${general.journalName.value}`} />
  { article.authors.map(author => (
      <meta name="citation_author" content={getFullName(author, 'ru')} />
    ))
  }
  <meta name="citation_publication_date" content={parseDate(article.issue.publication_date)} />
  <meta name="citation_journal_title" content={general.journalName.value} />
  <meta name="citation_volume" content={article.issue.volume.number} />
  <meta name="citation_issue" content={article.issue.number} />
  <meta name="citation_firstpage" content={article.pages.split('-')[0]} />
  <meta name="citation_lastpage" content={article.pages.split('-')[1]} />
  <meta name="description" content={article.submission['abstract_ru']} />
</Head>
```

Рисунок 28 – Заполнение тега <head> на странице статьи

Для удобства управления контентом страниц веб-приложения, в директории static была создана папка page_content, содержащая json-файлы, хранящие содержание страниц системы.

3.3 Выводы по разделу

Целью раздела являлось описание ключевых моментов разработки клиентской и серверной частей веб-платформы.

Была описана работа с библиотекой Sequelize на бэкенде, перенос таблиц в объектное представление, миграции и автозаполнение. Были описаны API-пути и способ реализации запросов с помощью Express, а также механизм аутентификации на основе Passport.js и авторизации.

В подразделе 3.2 был описан подход к разработке клиентской части веб-платформы, включая особенности работы с React-компонентами, стилизацией сайта с помощью Material UI, написание запросов на сервер.

4 Описание результатов разработки

4.1 Бэкенд

В результате разработки серверной части системы была реализована бизнес-логика, составлены API-пути и разработаны соответствующие методы.

Разработанные методы API описаны в таблицах 2–6.

Таблица 2 – Базовый API

№	Метод	Наименование	Описание
1	POST	registration	Регистрация пользователя в системе с помощью, как и связки логин-пароль, так и сторонних сервисов Google и Facebook.
2	POST	login	Вход в систему с помощью связки логин-пароль, Google или Facebook.
3	POST	logout	Выход из системы.
4	GET	getIssues	Возвращает все вышедшие номера.
5	GET	getIssue	Возвращает вышедший номер по ID.
6	GET	getCurrentIssue	Возвращает текущий номера.
7	GET	getArticle	Возвращает статью по ID.
8	POST	search	Возвращает результаты поиска по статьям журнала.
9	GET	downloadFile	Скачивание файла.

Таблица 3 – Пользовательский API

№	Метод	Наименование	Описание
1	GET	getCredentials	Возвращает личные данные пользователя.
2	PUT	editCredentials	Редактирует личные данные пользователя.

Окончание таблицы 3

№	Метод	Наименование	Описание
3	GET	getSettings	Возвращает настройки пользователя, включая подписки.
4	PUT	changeEmail	Изменяет электронный адрес пользователя.
5	PUT	changePassword	Изменяет пароль пользователя.

Таблица 4 – API администратора системы

№	Метод	Наименование	Описание
1	GET	getUsers	Возвращает информацию о пользователях системы.
2	POST	createUser	Создаёт пользователя.
3	PUT	editUser	Изменение информации о пользователе, в том числе его роли.
4	DELETE	deleteUser	Безвозвратно удаляет пользователя из системы.

Таблица 5 – API главного редактора

№	Метод	Наименование	Описание
1	GET	getIssues	Возвращает номера в стадии формирования.
2	GET	getIssue	Возвращает номера в стадии формирования по ID.
3	POST	decide	Выносит окончательное решение о публикации номера.

Таблица 6 – API члена редакционной коллегии

№	Метод	Наименование	Описание
1	GET	getIssues	Возвращает номера в стадии формирования.

Окончание таблицы 6

№	Метод	Наименование	Описание
2	GET	getIssue	Возвращает номер в стадии формирования по ID.
3	POST	vote	Выносит рекомендательное решение о публикации номера.

4.2 Фронтенд

Результатом разработки клиентской части являются страницы:

- главная страница журнала;
- страницы политики журнала (правила публикации, требования к оформлению и т.д.);
- страница номера журнала;
- страница статьи номера;
- страница архива номеров;
- страница поиска по журналу;
- страница голосования члена редакционной коллегии;
- страница голосования главного редактора;
- страница просмотра сформированного номера;
- страница управления пользователями системы администратора;
- страница персональных данных пользователя;
- страница настроек пользователя.

Все страницы веб-приложения также адаптированы под мобильные устройства. Вышеприведённые страницы представлены на рисунках 29–39. За основу контента страниц взято содержимое журнала Сибирского федерального университета.

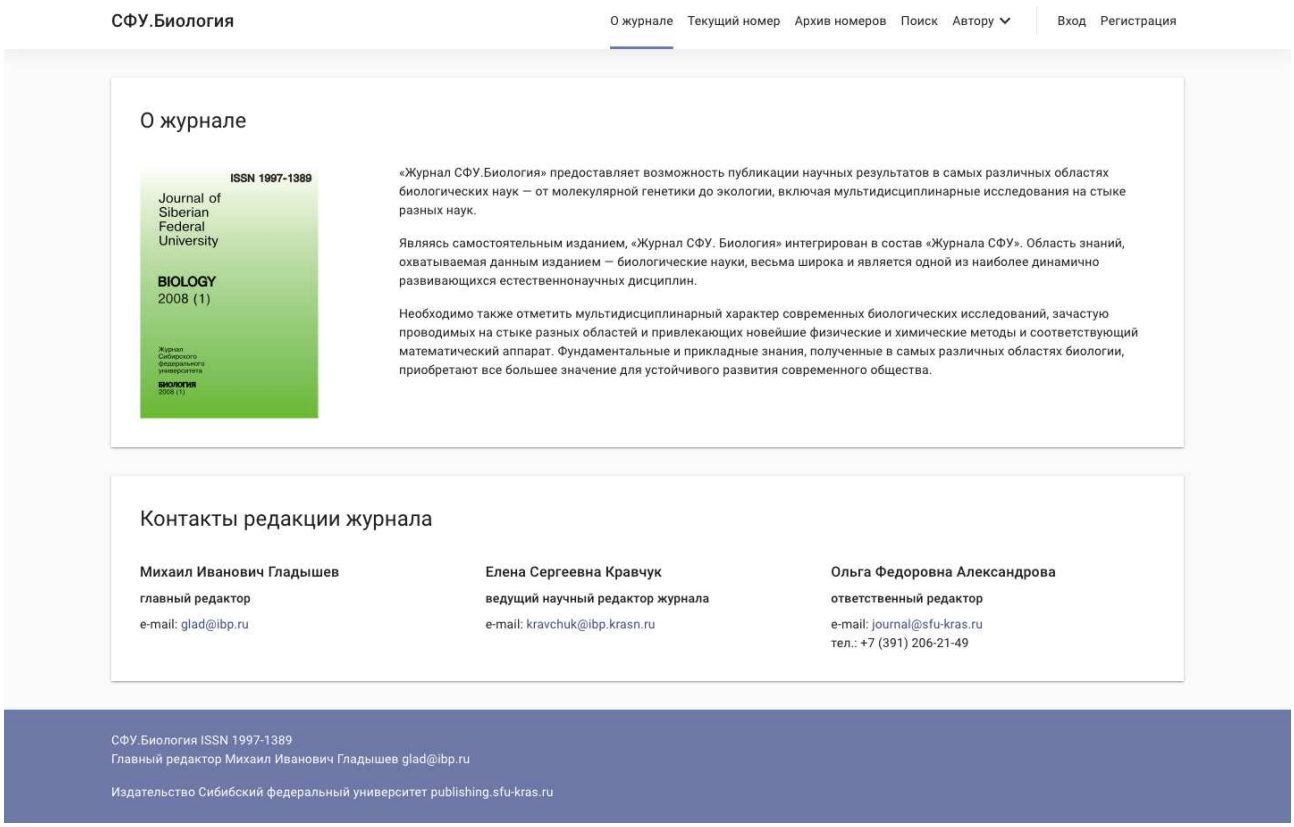

Рисунок 29 – Главная страница журнала

На главной странице журнала расположена основная информация о журнале и контакты редакции. Стоит отметить навигационную панель и футер, присутствующие на всех основных страницах системы.

Навигационная панель отвечает за переход между страницами сайта. Страницы «о журнале», «текущий номер», «архив номеров» и «поиск» подсвечиваются на навигационной панели для указания местоположения пользователя. Кнопка «автору» вызывает выпадающее меню со списком страниц, связанных с политикой и правилами журнала.

Также на навигационной панели располагаются кнопки «вход» и «регистрация», вызывающие открытие модальных окон с соответствующими функциями.

На мобильных устройствах с небольшим размером дисплея, кнопки перехода на страницы пропадают и переходят в отдельный элемент «шторку», вызываемую нажатием на единственную кнопку на панели навигации или свайпом слева-направо на устройстве пользователя.

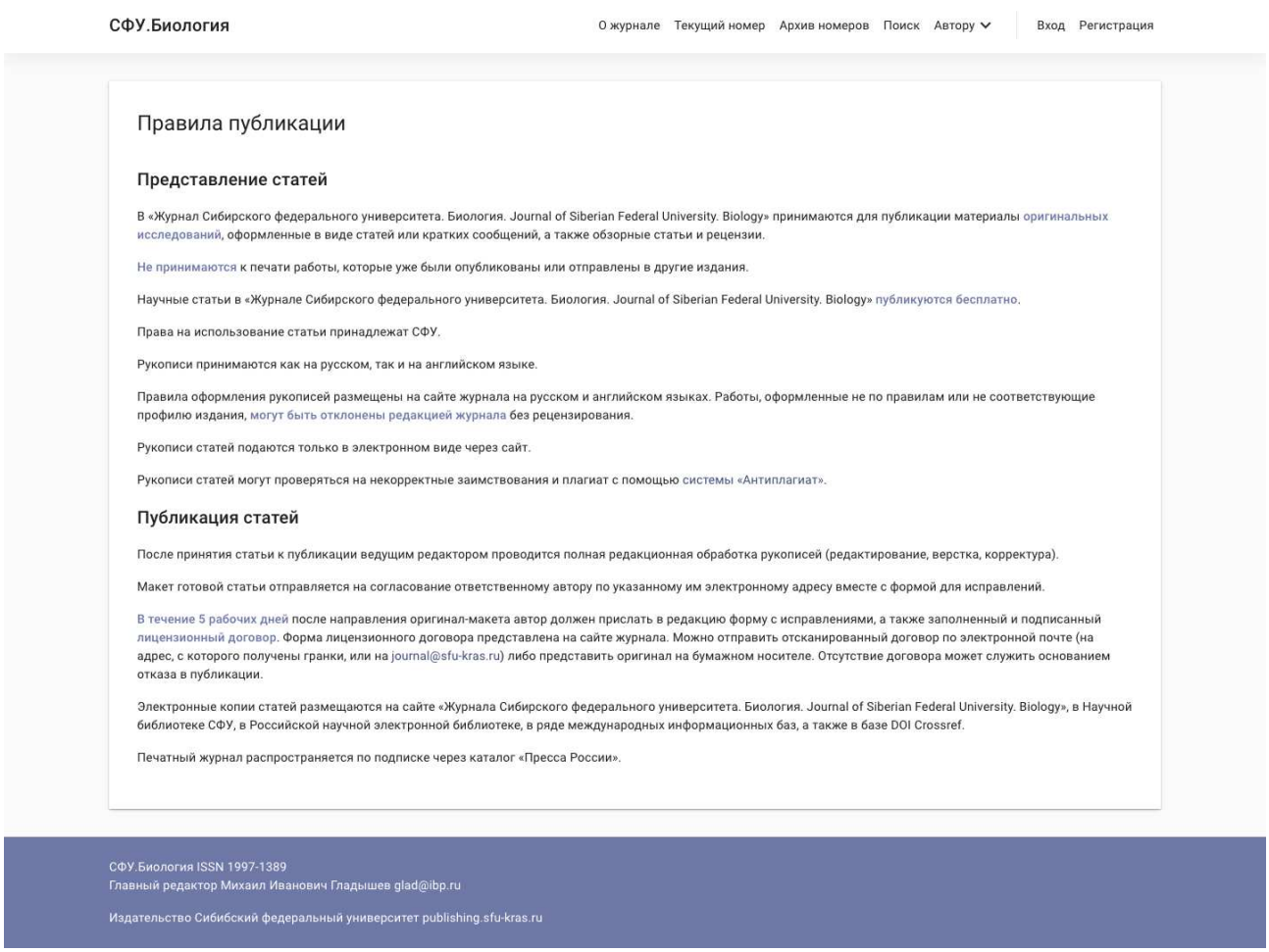

Рисунок 30 – Страница правил публикации

Страница правил публикации и другие страницы, связанные с политикой журнала, имеют одинаковую разметку, поэтому для примера представлена только одна из них.

К подобным страницам также относятся:

– требования к оформлению;
– порядка рецензирования;
– редакционной коллегии;
– публикационной этики.

Данные страницы зачастую содержат только объёмные текстовую информацию, однако средствами библиотеки Material UI представление этой информации удалось выполнить в «негрузном» свободном виде с использованием мягких, не отвлекающих внимание пользователя, цветов.

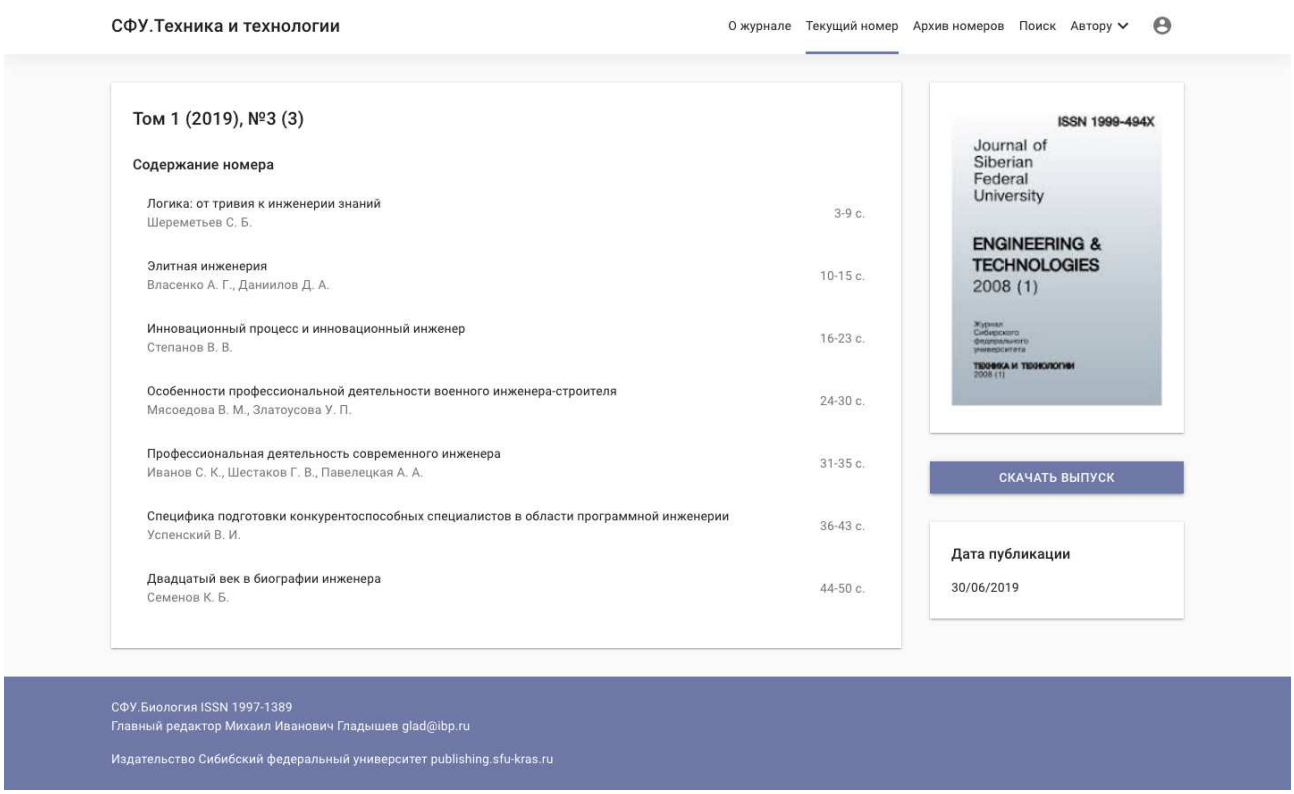

Рисунок 31 – Страница выпуска журнала

Структурно страница выпуска журнала разделена на блок слева, содержащий список всех статей, вошедших в выпуск журнала и на блок справа, в котором располагаются обложка выпуска, дата публикации и кнопка для скачивания всего выпуска.

Левый блок имеет заголовок с номером и годом тома, к которому относится выпуск журнала, а также обычный и сквозной номера выпуска. Ниже располагаются статьи, каждая помимо авторов также имеет соответствующие ей в выпуске страницы. При нажатии на статью происходит переход на её отдельную страницу.

Правый блок имеет, как и косметическую и информативную роли, позволяя левому блоку быть компактнее, так и функциональную, располагая ссылку на скачивание выпуска всегда перед глазами пользователя.

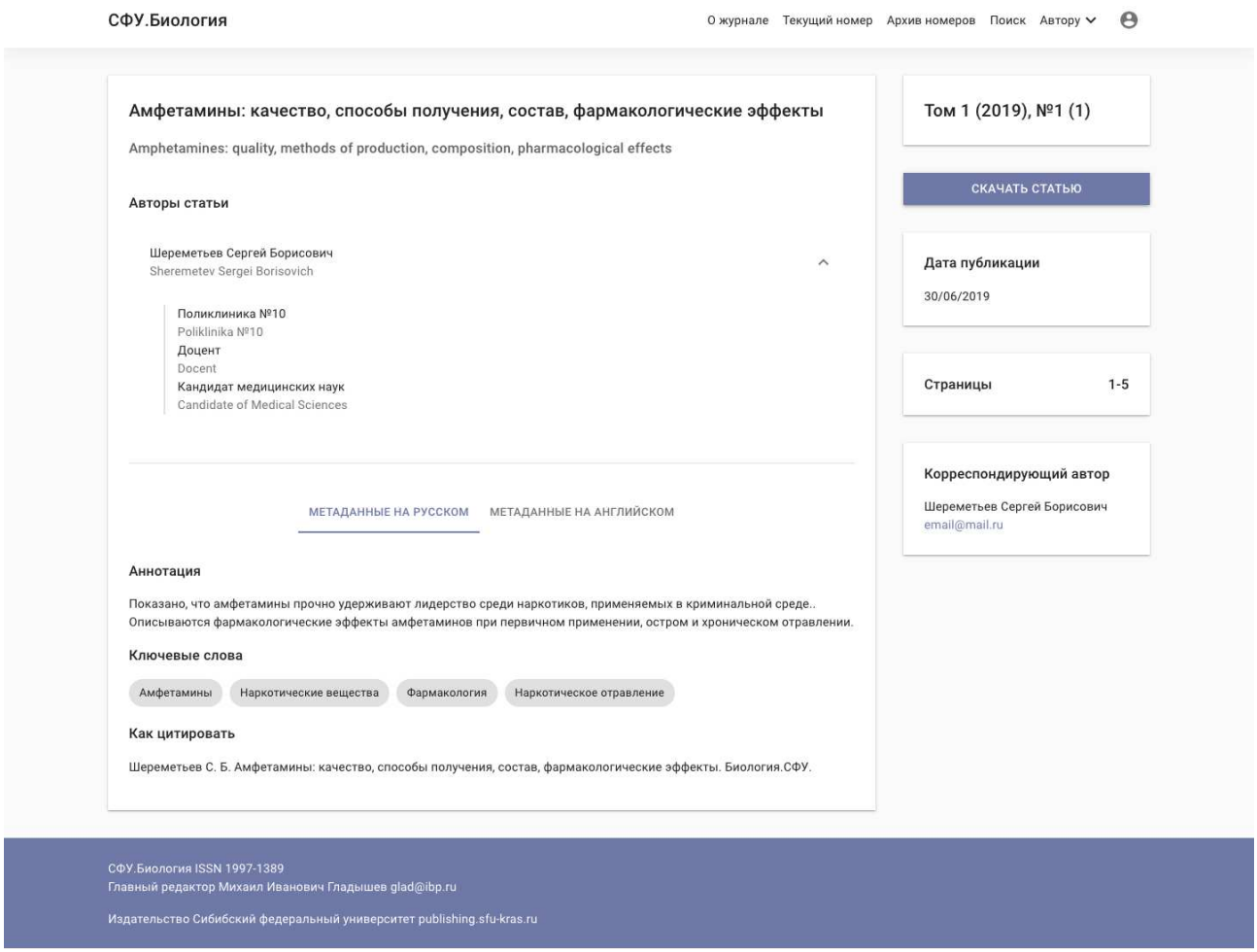

Рисунок 32 – Страница статьи выпуска

Страница статьи выпуска также структурно разделена на два блока.

Левый блок содержит метаданные статьи на русском и на английском языках. Название статьи имеет наибольший размер шрифта и служит заголовком страницы. Англоязычное название дублируется под русскоязычным, но не является столь заметным. Тот же самый подход используется и для остальных метаданных на английском языке. Авторы статьи находятся в списке, нажатие на элемент которого вызывает выпадающее меню, содержащее более полную информацию об авторе. Ниже авторов расположены аннотация, ключевые слова и способ цитирования статьи. Реализовано переключение между русскими и английскими метаданными нажатием соответствующих кнопок. Данное решение призвано сделать страницу просмотра статьи менее загромождённой.

Правый блок носит информативный характер и включает в себя информацию о статье, дате её публикации, страницах статьи в выпуске, а также блок с контактными данными корреспондирующего автора и ссылку на скачивание статьи.

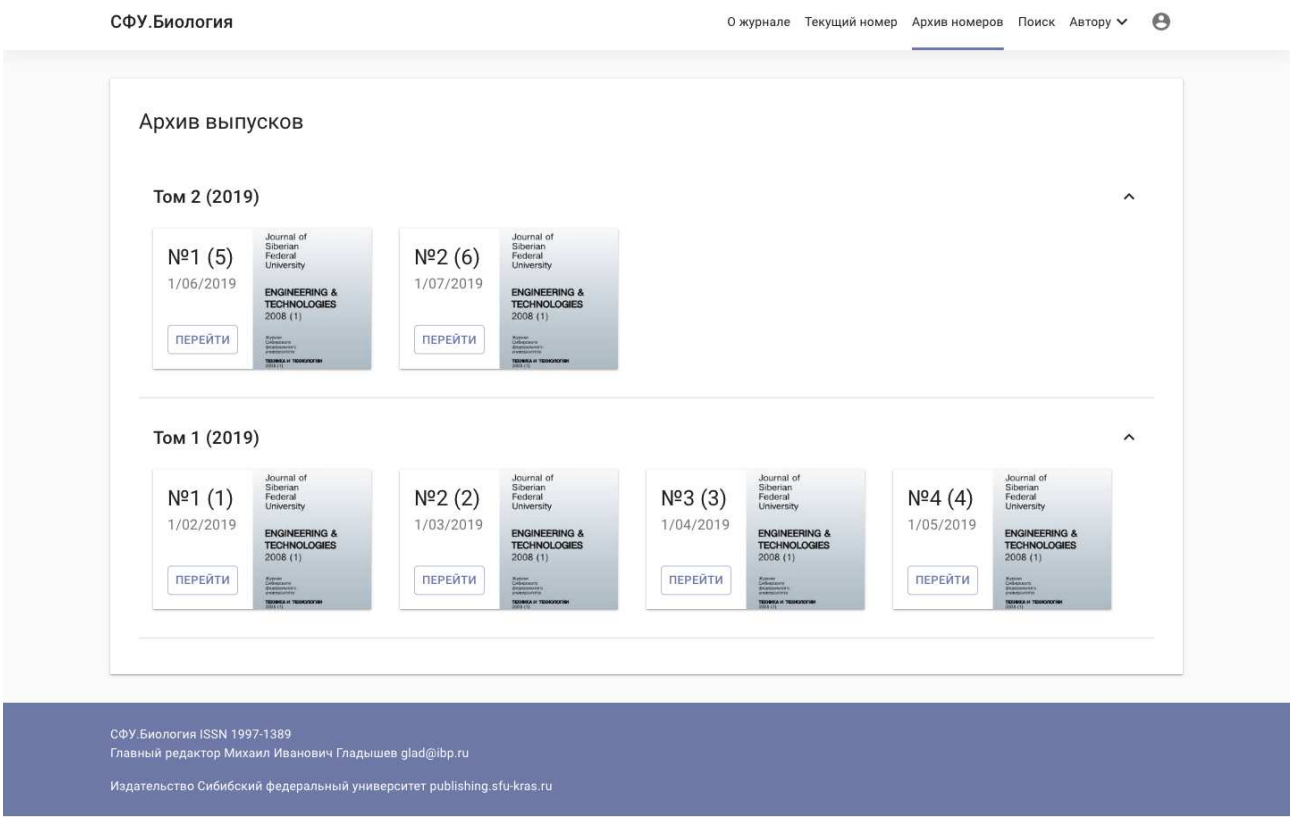

Рисунок 33 – Страница архива номеров

Страница архива номеров предоставляет возможность просматривать все выпуски журнала, сгруппированные по томам. Каждый выпуск содержит свой номер, сквозной номер, дату выхода и обложку. Кнопка «перейти» открывает страницу выбранного выпуска.

Так как со временем томов у журнала становится всё больше, то по умолчанию в развёрнутом виде на странице архива выпусков отображается только последний том, а все остальные компактно располагаются ниже в виде списка, нажатие на элемент которого разворачивает том и представляет его выпуски к просмотру.

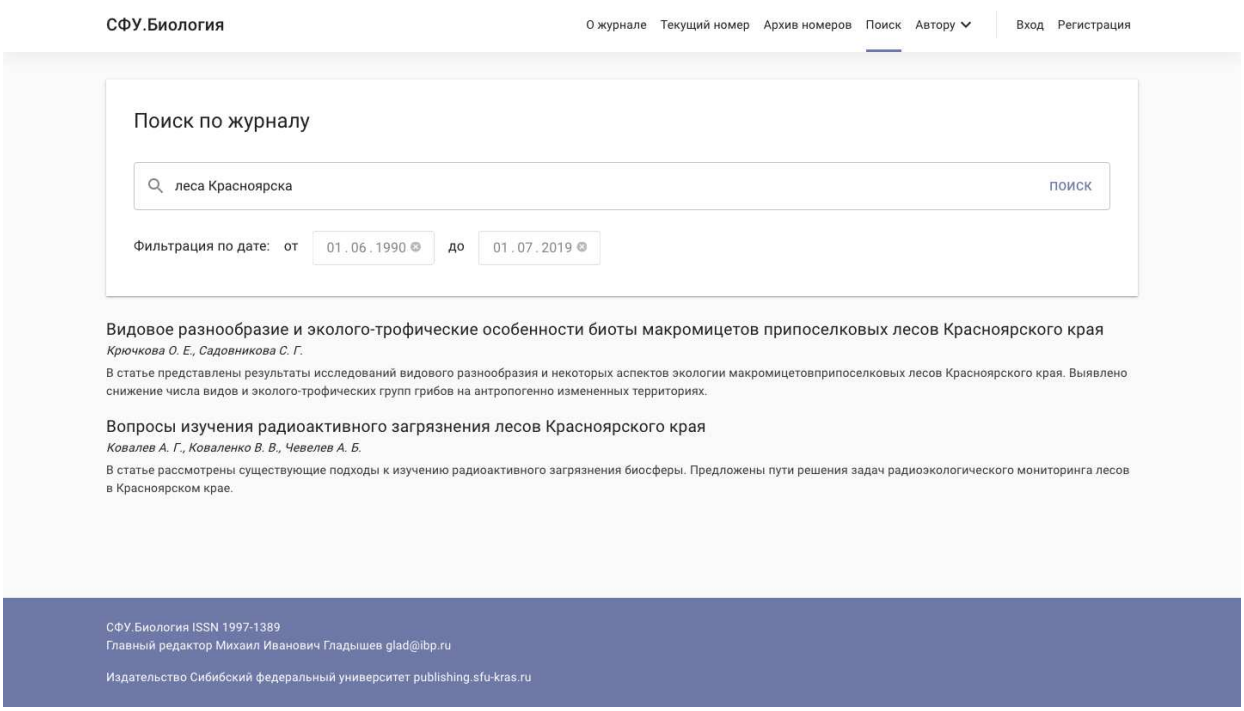

Рисунок 34 – Страница поиска

Страница поиска позволяет искать номера журнала. Поиск осуществляется по всем категориям: названию, аннотации, ключевым словам, ФИО авторов. Учитываются и англофицированные метаданные. Результаты поиска отображаются ниже строки поиска и являются статьями, удовлетворяющими поисковому запросу.

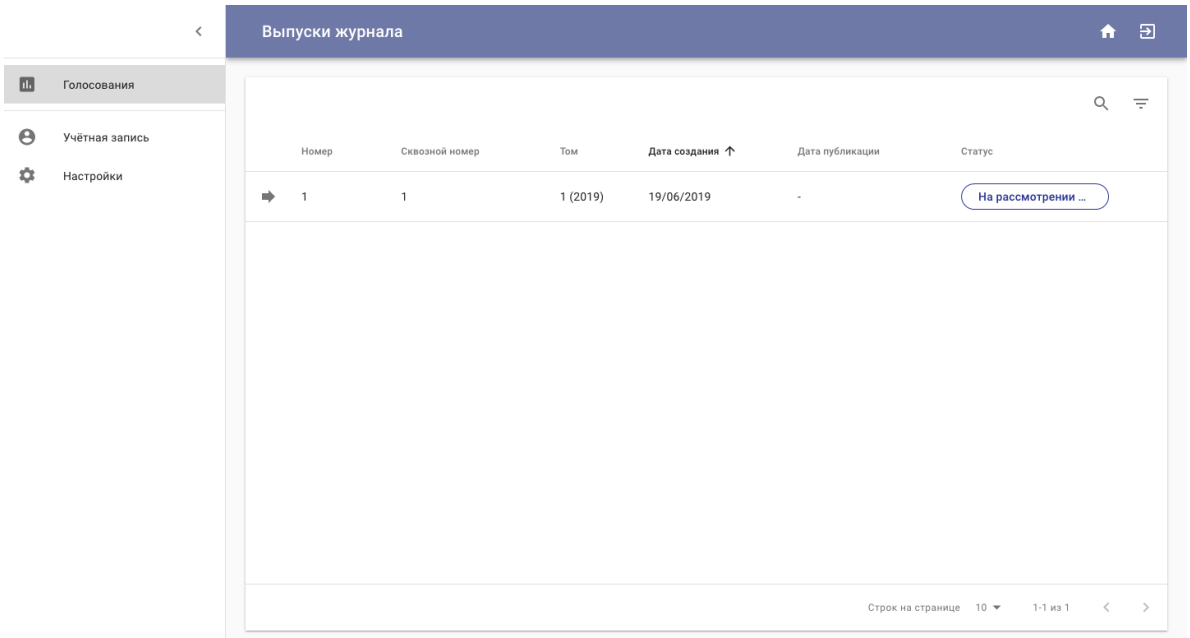

Рисунок 35 – Страница голосования члена редакционной коллегии

Страница голосования члена редакционной коллегии является частью прикладной логики журнала. Пользователь может проголосовать за или против сформированного выпуска и оставить комментарий касательно своего голоса, перейдя на сформированный выпуск в таблице. Таблица представляет возможности по поиску её содержимого и фильтрации выпусков. Страница голосования главного редактора аналогична данной, однако имеет другой функционал при переходе к сформированному номеру и другие статусы номера.

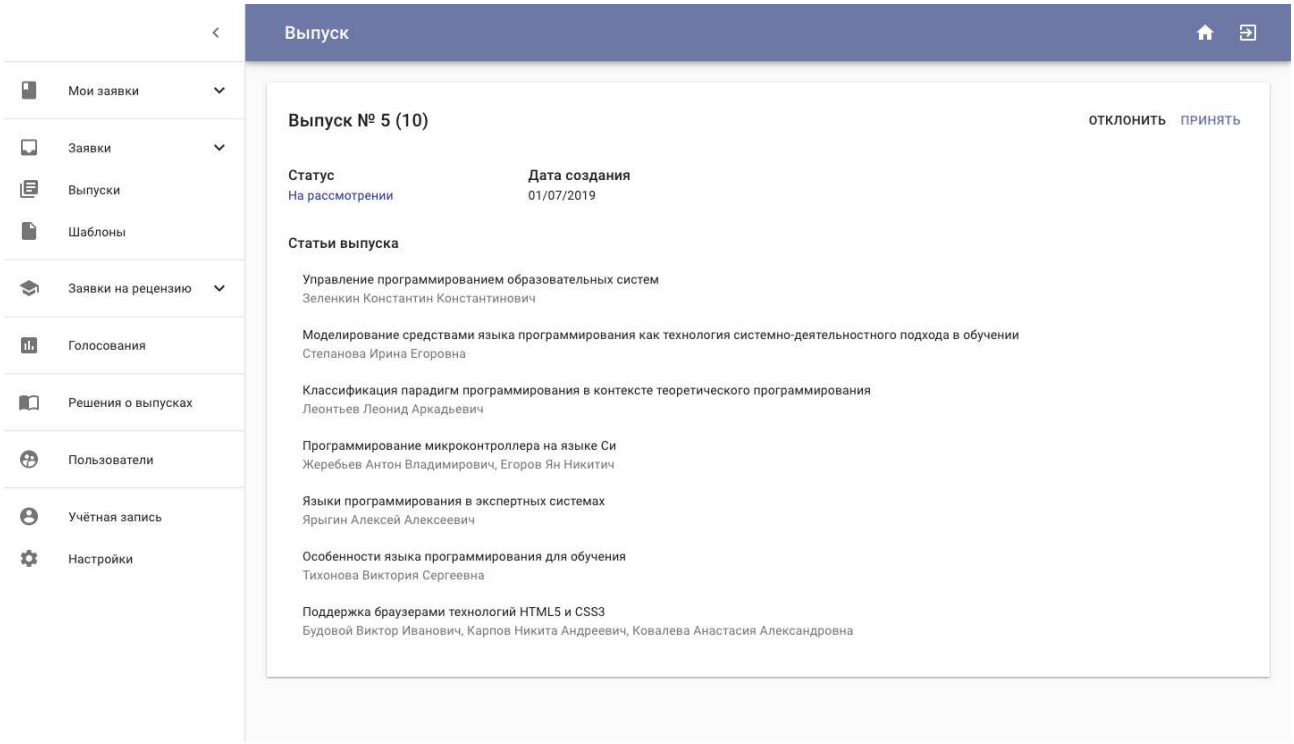

Рисунок 36 – Страница просмотра сформированного номера

Страница просмотра сформированного номера позволяет главному редактору и членам редакционной коллегии голосовать за или против определённого сформированного номера путём нажатия соответствующих кнопок в верхней части страницы. Также на данной странице находятся все статьи, вошедшие в выпуск. Нажатие на статью открывает модальное окно с её метаданными на русском и английском языках.

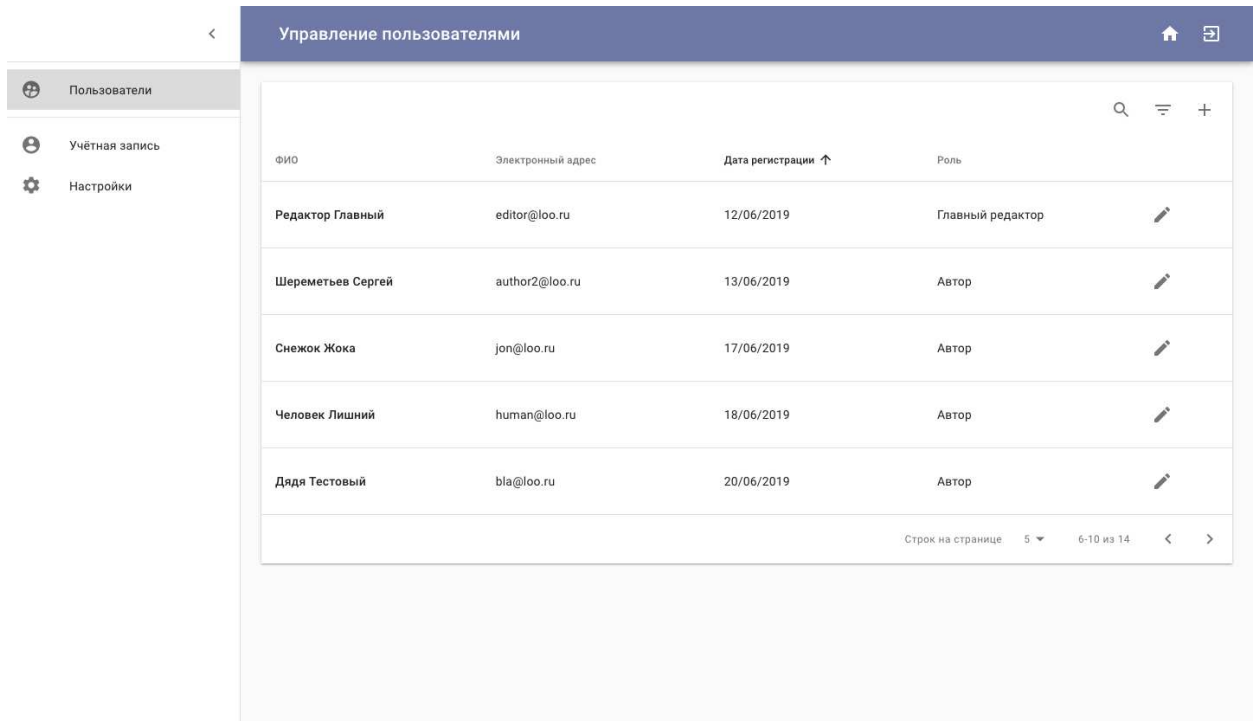

Рисунок 37 – Страница управления пользователями системы

Страница управления пользователями системы предоставляет администратору функционал по управлению пользователями, назначению им ролей или удалению, а также поиск и фильтрацию.

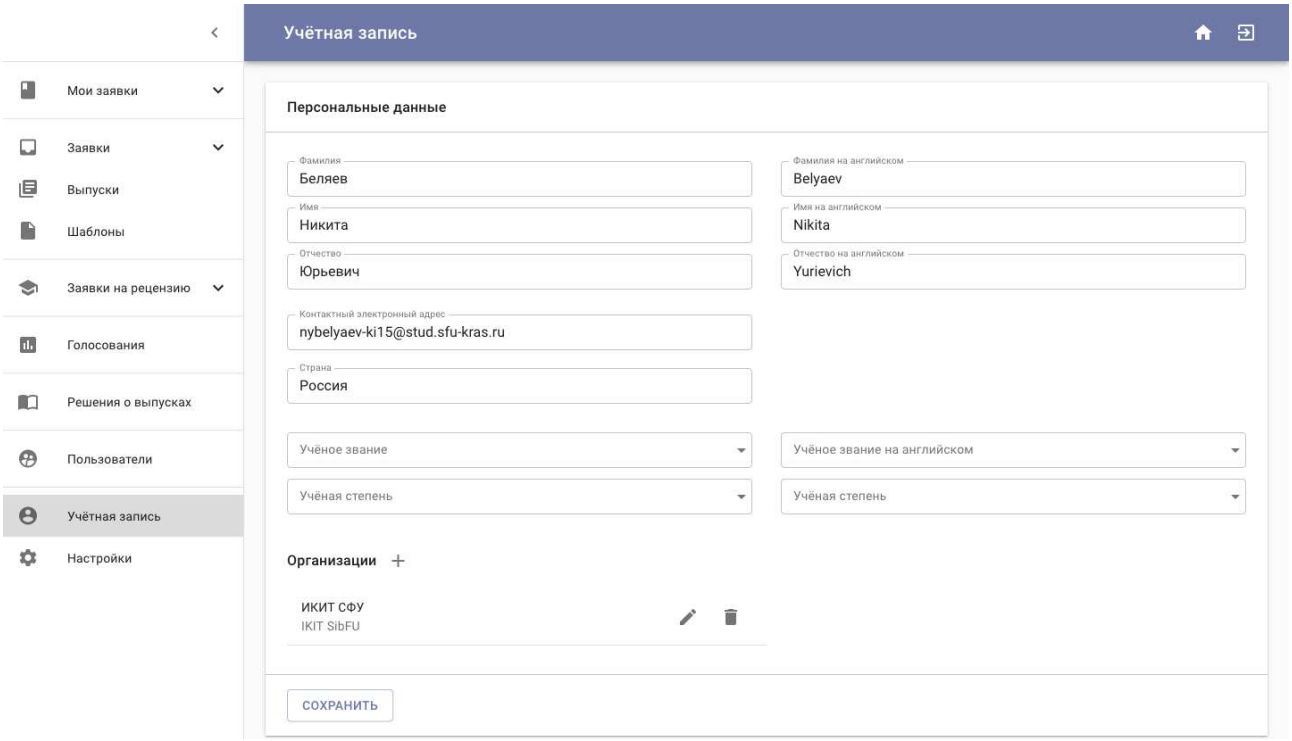

Рисунок 38 – Страница персональных данных пользователя

Страница персональных данных пользователя позволяет пользователю внести различные данные о себе в систему. В дальнейшем эти данные могут использоваться для автозаполнения формы подачи статьи в систему, но данный функционал не разрабатывался в рамках этой ВКР.

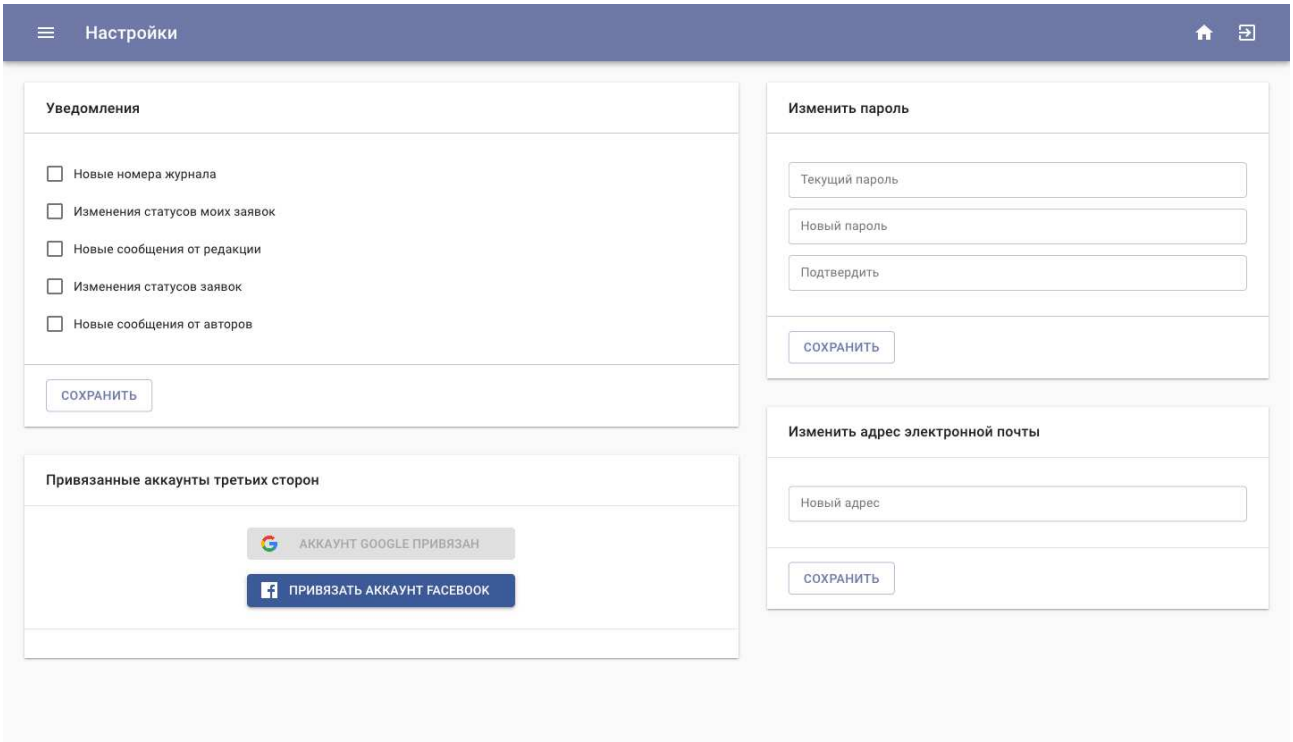

Рисунок 39 – Страница настроек пользователя

Страница настроек пользователя предоставляет возможность изменить электронный адрес или пароль пользователя, подписки на уведомления, а также прикреплять к аккаунту сторонние учётные записи таких систем, как Google и Facebook.

Блок уведомлений содержит только доступные для роли пользователя почтовые уведомления, подписаться на которые можно нажатием на элемент checkbox рядом с наименованием и подтверждение изменений кнопкой «сохранить». Блоки изменения пароля и адреса электронной почты работают схожим образом и также имеют полную проверку вводимых пользователем данных, впрочем, то же самое реализовано и во всех остальных формах системы.

4.3 Выводы по разделу

Целью данного раздела являлось описание результатов разработки веб-платформы.

Были представлены и описаны API-методы серверной части.

Были представлены скриншоты интерфейса клиентской части, а также описан функционал разработанных страниц.

ЗАКЛЮЧЕНИЕ

В ходе выполнения выпускной квалификационной работы был проведён анализ существующих аналогов разработанной веб-платформы, была выявлена её актуальность, были составлены требования, описаны этапы проектирования и разработки серверной и клиентской сторон.

Результатом ВКР является веб-платформа, предлагаемая к использованию редакциям российских научных журналов.

Разработанная платформа позволяет:

- регистрироваться и авторизоваться в ней;
- просматривать и скачивать архивные и текущие номера и статьи журнала;
- просматривать информационные страницы для потенциального автора;
- выполнять поиск по статьям, включая фильтрацию результатов по дате публикации;
- получать на электронную почту уведомления, соответствующие роли пользователя (например, о новом вышедшем или сформированном номере);
- осуществлять деятельность главного редактора и членов редакционной коллегии.

Несмотря на то, что все поставленные задачи были выполнены, существуют дальнейшие способы улучшения системы, например:

- реализация «умного» поиска по тексту статей, который бы учитывал синонимичные слова и выражения;
- создание удобного и многофункционального способа чтения статей прямо со страниц веб-платформы, без использования стандартных встроенных в браузер способов просмотра текстовых файлов и т.п.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Домнина Т. Н. Количество и темпы роста научных журналов / Т. Н. Домнина, О. А. Хачко. – Москва, 2015. – 20 с. – Деп. в ВИНИТИ РАН 21.09.2015, №156-В2015.
2. Статистика по информационным ресурсам [Электронный ресурс] // Сайт научной электронной библиотеки eLibrary. – Режим доступа: https://elibrary.ru/stat_time_items.asp.
3. Отчет по 2018 году [Электронный ресурс] // Официальный сайт Глобального инновационного индекса. – Режим доступа: https://www.globalinnovationindex.org/gii-2018-report#.
4. Российская наука в цифрах [Электронный ресурс] // Сайт ИСИЭЗ. – Режим доступа: https://issek.hse.ru/mirror/pubs/share/215179745.
5. Open Journal System [Электронный ресурс] // Официальный сайт платформы OJS. – Режим доступа: https://pkp.sfu.ca/ojs/.
6. ePublishing Toolkit [Электронный ресурс] // Официальный сайт библиотеки ePublishing Toolkit. – Режим доступа: https://dev.livingreviews.org/projects/epubtk.
7. Digital Publishing System [Электронный ресурс] // Официальный сайт системы DPubS. – Режим доступа: http://dpubs.org/.
8. Система комплексной поддержки и сопровождения научного журнала elpub [Электронный ресурс] // Официальный сайт системы elpub. – Режим доступа: https://elpub.ru/.
9. Результаты опроса разработчиков в 2019 году [Электронный ресурс] // Система вопросов и ответов о программировании Stack Overflow. – Режим доступа: https://insights.stackoverflow.com/survey/2019.
10. Основные концепции React.js, о которых стоит знать [Электронный ресурс] // Библиотека программиста. – Режим доступа: https://proglib.io/p/react-js-concepts/.

11. Node.js в действии. 2-е изд. / А. Янг, Б. Мек, М. Кантелон. – Санкт-Петербург : Питер, 2018. – 432 с.

12. SEO против React: Веб-краулеры умнее, чем вы думаете [Электронный ресурс] // Сайт организации freeCodeCamp. Режим доступа: https://www.freecodecamp.org/news/seo-vs-react-is-it-neccessary-to-render-react-pages-in-the-backend-74ce5015c0c9/.

13. Веб-рендер [Электронный ресурс] // Портал веб-разработки Google. – Режим доступа: https://developers.google.com/web/updates/2019/02/rendering-on-the-web.

14. Agile: самый популярный двигатель инноваций [Электронный ресурс] // Сайт журнала Forbes. – Режим доступа: https://www.forbes.com/sites/stevedenning/2015/07/23/the-worlds-most-popular-innovation-engine/#6cc4481d7c76.

15. Жизненный цикл разработки ПО. Все о Kanban [Электронный ресурс] // Сайт компании XB Software. – Режим доступа: https://xbsoftware.ru/blog/zhiznennyj-tsykl-po-kanban/.

16. ИТ-специалисты сравнивают систему контроля версий Git с конкурентами [Электронный ресурс] // Сайт сообщества разработчиков Search Software Quality. –Режим доступа: https://searchsoftwarequality.techtarget.com/feature/IT-pros-weigh-Git-version-control-against-the-competition.

17. Что такое архитектура программного обеспечения [Электронный ресурс] // Сайт компании IBM. – Режим доступа: https://www.ibm.com/developerworks/ru/library/eeles/index.html#notes.

18. Определение архитектуры информационной системы [Электронный ресурс] // Служба тематических толковых словарей Glossary Commander. – Режим доступа: http://www.glossary.ru/cgi-bin/gl_find.cgi?ph=%C0%F0%F5%E8%F2%E5%EA%F2%F3%F0%E0+%E8%ED%F4%EE%F0%EC%E0%F6%E8%EE%ED%ED%EE%E9+%F1%E8%F1%F2%E5%EC%FB&action.x=23&action.y=10.

19. Многоуровневые системы клиент-сервер [Электронный ресурс] // Сайт издательства «Открытые системы». – Режим доступа: https://www.osp.ru/nets/1997/06/142618/.

20. Многоуровневая архитектура [Электронный ресурс] // Сайт сибирского отделения Российской академии наук. – Режим доступа: http://www-sbras.nsc.ru/Report2006/Report321/node30.html.

21. Архитектура средств интеграции Web и СУБД [Электронный ресурс] // Сайт кафедры «Вычислительная техника» Пензенского Государственного университета. – Режим доступа: http://alice.pnzgu.ru:8080/~zsa/sql/titan_zsa/zsa_site/files/new_conc_files/9.htm.

22. Модель-представление-контроллер [Электронный ресурс] // Документация yiiframework. – Режим доступа: https://www.yiiframework.com/doc/guide/1.1/ru/basics.mvc.

23. MVC – модель-представление-контроллер [Электронный ресурс] // Сайт компании Web creator. – Режим доступа: https://web-creator.ru/articles/mvc.

24. Определение структуры базы данных [Электронный ресурс] // Словарь компании Финам. – Режим доступа: https://www.finam.ru/dictionary/wordf0064A/.

25. Проектирование веб-интерфейсов [Электронный ресурс] // Сайт студии Jazz Pixels. – Режим доступа: https://jazzpixels.ru/blog/8/proektirovanie-interfeisov.

26. Мобильный и десктопный трафик в 2019 году [Электронный ресурс] // Сайт digital-агентства Stone Temple. – Режим доступа: https://www.stonetemple.com/mobile-vs-desktop-usage-study/.

27. Рекомендации по подготовке сайта научного журнала: основные требования для представления издания российскому и международному сообществу / О. В. Кириллова, Н. Г. Попова, А. В. Скалабан, М. М. Зельдина, Т. А. Лоскутова. – Екатеринбург : Изд-во Урал. ун-та, 2018. – 92 с.

28. Сайт научного журнала [Электронный ресурс] // Официальный сайт Института социологии РАН. – 2014. – 12 с. – Режим доступа: http://www.isras.ru/publ.html?id=3058.

29. Диаграммы вариантов использования [Электронный ресурс] // Сайт компании Informicus. – Режим доступа: http://www.informicus.ru/default.aspx?SECTION=6&id=73&subdivisionid=4.

30. Рекомендации по наименованию путей для доступа к ресурсам [Электронный ресурс] // Руководство REST API. – Режим доступа: https://restfulapi.net/resource-naming/.

31. Введение в хуки [Электронный ресурс] // Документация React. – Режим доступа: https://reactjs.org/docs/hooks-intro.html.

32. Рекомендации по включению в систему для вебмастеров [Электронный ресурс] // Справочник Google Scholar. – Режим доступа: https://scholar.google.com/intl/en/scholar/inclusion.html#indexing.

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

Разработка веб-платформы управления и публикации научных статей
Тема

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

Выпускник 03.07.19 Н. Ю. Беляев
подпись, дата инициалы, фамилия

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

Красноярск 2019