**Министерство образования и науки Кыргызской Республики**

**Министерство науки и высшего образования Российской Федерации**

**Государственное образовательное учреждение высшего профессионального образования**

**Кыргызско-Российский Славянский университет**

Естественно – технический факультет

Кафедра информационных и вычислительных технологий

**выпускная квалификационная работа**

на тему:

**Разработка автоматизированной системы для конвертации метаданных из системы Open Journal Systems в систему elibrary.ru**

|  |  |
| --- | --- |
| **Выполнил:**  студент группы ЕПИ 2-16 |  |

Иващенко Лилия Андреевна \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(подпись)

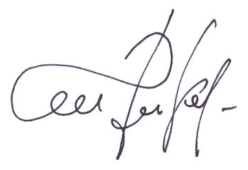
**Руководитель:**

кандидат технических наук,

доцент Верзунов Сергей Николаевич **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(подпись)

Работа к защите допущена:

****Заведующей кафедрой ИВТ

доктор физико-математических наук,

профессор Десятков Геннадий Александрович                **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

(подпись)

**БИШКЕК 2020**

**Аннотация**

Данная выпускная квалификационная работа посвящена разработке системы для конвертации метаданных, которая предназначена для подготовки к выпуску научной статьи.

Целью данной выпускной квалификационной работы является разработка плагина, способного выполнять автоматизированную конвертацию метаданных из системы Open Journal Systems в систему elibrary.ru, а также позволит сделать это в более сжатые сроки.

Для решения поставленной задачи необходимо выполнить следующие работы:

1. Провести анализ предметной области.
2. Реализовать алгоритмы обработки данных.
3. Реализовать создание и сохранение проектов в системе.

Разработанная система выполняет следующие функции:

* Создание новых проектов.
* Метаданные будут автоматически загружаться из файла с расширением .xml.
* Редактирование статьи вручную.
* Сохранение статьи в системе.

Ключевые слова: конвертация метаданных, научная статья, выгрузка метаданных проекта, расширение, библиотека elibrary.ru, система Open Journal Systems.

Объем пояснительной записки: 55 стр.

Количество рисунков: 24 шт.

Количество таблиц: 14 шт.

**Аннотация**

Мета маалыматтын конвертациясынын системасына арналган бүтүрүү квалификациялык иши илимий статьянын басып чыгарылышынын даярдыгына багытталган. Бүтүрүү квалификациялык ишинин максаты болуп Open Journal System системасынан elibrary.ru системасына мета маалыматты автоматтык түрдө конвертациялай алган плагиндин иштелип чыгышы жана ошондой эле эң эле кыскартылган убакта жасай алуу болуп саналат.

Коюлган тапшырманы аткаруу үчүн төмөндөгү ыкмаларды аткаруу зарыл:

1. Предметтик областты анализден өткөрүү.

2. Маалыматты иштеп чыгуучу алгоритмдерди ишке ашыруу.

3. Системадагы проекттерди түзүү жана сактоону ишке ашыруу.

Иштелип чыккан система төмөндөгү функцияларды аткарат:

• Жаңы проекттерди түзүү.

• Мета маалыматтар автоматтык түрдө.xlm. чоңойтуу менен файлдан жүктөлүп турат.

• Статьяны колго оңдоо.

• Статьяны системага сактоо.

Орчундуу сөздөр: мета маалыматты конвертациялоо, илимий статья , проекттин мета маалыматын чыгаруу, elibrary.ru библиотекасы, Open Journal System системасы.

Түшүндүрмө көлөмү: 55 б.

Сүрөттөрдүн саны: 24 даана

Таблицлардын саны: 14 даана

**Annotation**

This final qualification work is devoted to the development of a Systems for converting metadata, which is intended to prepare for the publication of a scientific article.

The aim of this final qualification work is to develop a plug-in that is able to automatically convert metadata from the Open Journal Systems to elibrary.ru, and will also allow this to be done in a shorter time.

To solve this problem, you must perform the following work:

1. Conduct a domain analysis.

2. Implement data processing algorithms.

3. Implement the creation and preservation of projects in the Systems.

The developed Systems performs the following functions:

• Creation of new projects.

• Metadata will be automatically loaded from a file with the extension .xml.

• Editing an article manually.

• Saving an article in the Systems.

Keywords: metadata conversion, scientific article, uploading project metadata, extension, elibrary.ru library, Open Journal Systems.

Volume of explanatory note: 55 pages

Number of drawings: 24 pcs.

Number of tables: 14 pcs.

**Оглавление**

[Введение 6](#_Toc43560108)

[1.2. Позиционирование 8](#_Toc43560109)

[1.3. Описания пользователей 10](#_Toc43560110)

[1.4. Краткий обзор изделия 12](#_Toc43560111)

[1.5. Возможности продукта 13](#_Toc43560112)

[1.6. Ограничения 13](#_Toc43560113)

[1.7. Показатели качества 13](#_Toc43560114)

[1.8. Другие требования к изделию 14](#_Toc43560115)

[1.9. Требования к документации 14](#_Toc43560116)

[2. Спецификация требований к программному продукту 15](#_Toc43560117)

[2.2.  Общее описание 16](#_Toc43560118)

[2.3.  Специфические требования 31](#_Toc43560119)

[3. Проектирование и конструирование программного обеспечения 34](#_Toc43560120)

[4. Разработка тестов и тестирование программного продукта 40](#_Toc43560121)

[4.1. Разработка плана тестирования 40](#_Toc43560122)

[4.2. Модульное тестирование 41](#_Toc43560123)

[5. Руководство пользователя 44](#_Toc43560124)

[5.1. Назначение системы 44](#_Toc43560125)

[5.2. Условия применения системы 44](#_Toc43560126)

[5.3. Подготовка системы к работе 44](#_Toc43560127)

[Заключение 53](#_Toc43560128)

[Список литературы 54](#_Toc43560129)

[Приложение 1. Глоссарий 56](#_Toc43560130)

[Приложение 2. Листинг 58](#_Toc43560131)

# Введение

Компьютеризация издательской деятельности в последние годы привела к существенному увеличению возможностей быстрой публикации результатов научных исследований с помощью различных издательских систем для публикации научных изданий в сети Интернет. Это либо коммерческие решения крупных издательств [1], либо различные самостоятельно разработанные системы, например, электронный научный журнал «Наука, новые технологии и инновации Кыргызстана» [2], или же решения на базе свободно распространяемого программного обеспечения [3].

Традиционно в Кыргызской Республике, в основном используются самостоятельные разработки или, реже, решения на основе свободно распространяемого программного обеспечения, так как большая масса журналов существует при поддержке небольших научных издательств и имеет ограниченные организационные и финансовые возможности. При этом компоновка цифровых научных коллекций происходит зачастую нерегулярно и зависит, в основном, от необходимости обеспечить информационную поддержку научной или образовательной деятельности. Это в частности относится и к сборникам научных статей, а также к материалам научных конференций. По большому счёту статьи из этих источников не имеют возможности попасть даже в такие хранилища научной информации, как Научная электронная библиотека (НЭБ) [4] или Киберленинка [5]. В качестве причин этого можно привести необходимость заключения лицензионного договора с авторами статей, а в основном, – нехватку человеческих и временных ресурсов. Кроме того, в большинстве из создаваемых электронных коллекций не только отсутствует возможность обмена метаданными научных статей, но и сами метаданные в каком-либо из общепринятых форматов их хранения. Это, в свою очередь, делает сомнительным возможность организации взаимодействия таких коллекций с основными хранилищами научной информации и создание условий для индексации ресурсов в ведущих системах поиска и хранения научных публикаций: НЭБ, Google Scholar, Web of Science и других [6].

Поэтому для подавляющего числа научных изданий в Кыргызской Республике актуальной целью является решение проблемы формирования информационного пространства, адекватного современным требованиям, предъявляемым к инновационной научной деятельности.

Одной из важных составляющих этого информационного пространства является интернет-сайт, обеспечивающий решение таких задач как: формирование веб-страниц статей внутри электронного каталога издательства, индексирующихся системами поиска научных публикаций, создание инструментов экспорта статей в электронные коллекции. Особенно важной в настоящее время является проблема загрузки материалов из электронного каталога издательства в каталог НЭБ. В настоящей работе предлагается решение этой задачи путём разработки дополнения к полярной системе автоматизации редакционно-издательстой деятельности.

Пояснительная записка состоит из 5 глав, введения, заключения, списка литературы и приложения.

В первой главе представлено видение проекта. В данном разделе обсуждаются высокоуровневые требования (возможности, свойства) к программному продукту и наиболее существенные ограничения, т.е. производится выявление и анализ бизнес-требований.

Во второй главе спецификация требований к программному продукту. В данном разделе обсуждаются специфические требования, присущие данному программному продукту.

В третьей главе Проектирование и конструирование программного обеспечения. Данный раздел описывает процесс разработки ПО.

В четвертой главе выполнено тестирование программного обеспечения с применением модульного тестирования.

В пятой главе представлено руководство пользователя с описанием основных действий при работе с программой.

### 1. Видение

### 1.1 Введение

В настоящее время для публикации научных работ в национальной академии наук используется система разметки метаданных Articulus.

Программа разметки Articulus предназначена для подготовки выпусков и журналов для размещения в научной библиотеке и в Российском индексе научного цитирования. Обычно эти издания находятся в файлах с расширением .xml [7].

Несмотря на множество подобных систем, все они имеют одни и те же свойства: загрузка статьи и ее разметка по заданным характеристикам.

Под разметкой понимается выгрузка метаданных из файла и добавление пропущенных (имя, фамилия, отчество, год рождения автора, ключевые слова и т.д.). Для разметки метаданных используются плагины, так как это позволяет ускорить время подготовки статьи к публикации и исключает возможность пропуска данных.

### 1.1.1. Цель

Целью создания этой системы является сбор, анализ и определение потребностей и ресурсов высокоуровневой системы преобразования метаданных на основе платформы Open Journal Systems. Документ фокусируется на ресурсах, необходимых совладельцам и целевым пользователям, и на том, почему эти потребности существуют. Подробная информация о том, как система Open Journal отвечает этим требованиям, будет подробно описана в дополнительных спецификациях и вариантах использования.

**1.1.2. Контекст**

Настоящий документ разрабатывается в рамках проекта автоматизации конвертации метаданных из системы elibrary.ru в систему Open Journal Systems.

**1.1.3. Определения, акронимы и сокращения**

Основные определения приведены в документе «Приложение 1. Глоссарий проекта».

**1.1.4. Краткое содержание**

Документ описывает требования высокого уровня для системы Open Journal Systems. Указаны основные коммерческие преимущества решения, предусмотренного в Видении, сформулированы ключевые проблемы и методы их решения, характеристики пользователей системы, возможности системы, ограничения, показатели качества и другие требования продукт указан.

**1.2. Позиционирование**

**1.2.1. Деловые преимущества**

В настоящее время конвертация метаданных отсутствует в системе . Новое решение позволит обеспечить более удобный режим доступа заинтересованных лиц к информации, повысить быстродействие, обеспечить надёжное хранение данных и более полный охват функций, подлежащих автоматизации.

Использование системы автоматизированной конвертации метаданных сократит вовлеченность индексатора в процесс подготовки статьи, сведя его к рабочей системе.

В сравнении с имеющимися, новое решение позволит сократить время работы индексатора для подготовки статьи к дальнейшему рассмотрению и публикации.

**1.2.2. Определение проблемы**

Проблема описана в табл. 1.1.

*Таблица* *1.1*. Определение проблемы

|  |  |
| --- | --- |
| Проблема | Затрудненный доступ рецензента к статье, отсутствие шаблонов для написания рецензии |
| затрагивает | Научных работников, рецензентов |
| Ее следствием является | Задержки выпуска научных статей |
| Успешное решение | Оптимальная организация работы рецензентов за счет конвертации данных в OpenJournalSystems. |
| Проблема | Высокая трудоёмкость процесса обработки метаданных |
| затрагивает | Литературных редакторов |
| Ее следствием является | Более затянутый процесс обработки научных статей, задержка выпусков журналов |
| Успешное решение | Исключение задержек, повышение скорости работы посредством введения новой системы конвертации метаданных |
| Проблема | Сложность назначения реального срока выпуска статей |
| затрагивает | Научных работников, литературных редакторов |
| Ее следствием является | Возможная потеря времени и публикация статей в более поздний срок |
| Успешное решение | При внесении в систему запроса, получаем варианты возможного времени исполнения публикации |

**­­­1.2.3 Определение позиции изделия**

Позиция изделия описана в табл. 1.2.

*Таблица 1.2*. Определение позиции изделия

|  |  |
| --- | --- |
| Для | Редакций научных журналов |
| которой | Требуется оптимизировать процесс работы с метаданными |
| (Название продукта) | Система конвертации метаданных |
| который | Оптимизирована для обработки статей и конвертации метаданных |
| В отличие от | Существующей системы Articulus |
| наш продукт | Позволяет уменьшить трудозатраты для подготовки научного журнала, исключает ошибки конвертации метаданных |

**1.3. Описания пользователей**

**1.3.1. Сведения о пользователях**

У системы существуют два основных пользователя: научный работник (автор статьи), индексатор. Научный работник – предоставляет свои личные данные, а также собственно-разработанные научные технические работы в целях дальнейшей публикации, предоставляет их рецензенту для проверки, затем передает статьи литературному редактору. Индексатор – занимается формированием метаданных для выпуска статьи в онлайн библиотеку или журнал.

**1.3.2. Пользовательская среда**

В настоящее время на предприятии имеется более 50 ученых, предоставляющих свои статьи и 8 системных администраторов.

Система будет использоваться актером индексатором, занимающимся подготовкой к публикации научных статей.

Разумное время обработки данных одной статьи – 40 минут, для статей с большим количеством авторов до 2 часов.

Система будет работать на платформе IBM PC. Операционная система: Microsoft Windows 7, Windows 8, Windows 10. Веб-браузеры: Opera, Mozilla Firefox, Google Chrome.

**1.3.3. Профили пользователей**

Профиль пользователя Научный работник описан в следующей табл 1.3.

*Таблица 1.3*. Профиль пользователя Научный работник

|  |  |
| --- | --- |
| **Типичный представитель** | **Научный работник** |
| **Описание** | Пользователь системы, наделенный правами на чтение публикаций. |
| **Тип** | Пользователь |
| **Ответственности** | Внесение своих личных данных, выполнив регистрацию в системе; предоставление научных статей для дальнейшей обработки литературным редактором, системным администратором. |
| **Критерий успеха** | Сокращение временных затрат на публикацию статьи, успешная регистрация в данной системе. |

Профиль пользователя Индексатор описан в следующей табл. 1.4.

*Таблица 1.4*. Профиль пользователя Индексатор

|  |  |
| --- | --- |
| **Типичный представитель** | **Индексатор** |
| **Описание** | Формирование метаданных |
| **Тип** | Пользователь |
| **Ответственности** | Загрузка метаданных в систему elibrary.ru. |
| **Критерий успеха** | Умение работать с системой. |

**1.3.4. Ключевые потребности пользователей**

Научный работник затрачивает большое количество времени на составление собственной научно-технической статьи. Индексатор, в свою очередь, затрачивает определенное количество времени на обработку данных статей. Большое количество научных работников нуждается в системе, которая бы ускорила и оптимизировала вышеуказанные процессы.

**1.4. Краткий обзор изделия**

**1.4.1. Контекст использования системы**

Система является законченной независимой разработкой. В перспективе возможно использование системы в комплексе с другими системами автоматизации научно-издательской деятельности.

**1.4.2. Сводка возможностей**

Возможности программы описаны в табл. 1.5.

*Таблица 1.5*. Сводка возможностей продукта

|  |  |
| --- | --- |
| **Выгоды заказчика** | **Поддерживающие возможности** |
| Упрощение редактирования метаданных | Автоматическая конвертация метаданных. |
| Ускорение обращения информации в процессе издания журнала | Система позволит ускорить процесс редактирования публикаций, а также оптимизирует взаимодействие автора и редакции журнала. |
| Формирование единой базы публикаций | Все заинтересованные пользователи со своих рабочих мест имеют доступ ко всем сданным в редакцию статьям. |
| Отказ от повторных редактирований для остальных систем конвертации данных | Система позволяет пользователям делать разметку для статьи единожды |

**1.4.3. Предположения и зависимости**

Программа может быть использована при наличии установленного программного обеспечения – веб-браузера(Opera, Mozilla Firefox, Google Chrome) и стабильного интернет соединения.

В случае приобретения или разработки информационных систем, будет необходимо разработать соответствующие средства импорта-экспорта информации.

**1.5. Возможности продукта**

### 1.5.1. Структурированное описание системы

Возможность редактирования метаданных по имеющимся параметрам.

### 1.5.2.  Расчёт нормативного времени выполнения работ

Возможность для каждой из работ автоматически определить, на основании введённых параметров, время исполнения работы.

### 1.5.3.  Передача статьи для публикации

Возможность направить статью в целях дальнейшей ее публикации.

### 1.5.4.  Диспетчеризация работ

Возможность для каждой из работ выбрать требуемый ресурс из возможных ресурсов и установить время начала и окончания выполнения работы.

### 1.5.5.  Назначение исполнителей

Возможность для каждого из системных администраторов произвести публикацию статьи.

### 1.5.6.  Контроль исполнения и оперативная корректировка ошибок

Возможность оперативной корректировки данных статей при возникновении ошибок.

**1.6. Ограничения**

Внедрение системы не должно занимать более 3 месяцев. В ядре системы должна быть представлена промышленная СУБД реляционного доступа.

Все обращения к информации должны осуществляться через сайты, основанные на системе Open Journal Systems.

**1.7. Показатели качества**

**1.7.1. Применимость**

Время, необходимое для обучения обычных пользователей – 2 рабочих дня (16 часов), для обучения продвинутых пользователей – 1 рабочий день (8 часов).

Время отклика для работы пользователя с данными – не более 5 секунд.

**1.7.2. Надежность**

Доступность – время, затрачиваемое на обслуживание программы не должно превышать 3% от общего времени работы.

Среднее время безотказной работы – 10 рабочих дней.

Максимальная норма ошибок или дефектов – 1 ошибка на десять тысяч строк кода.

**1.8. Другие требования к изделию**

**1.8.1. Применяемые стандарты**

Система должна соответствовать всем стандартам интерфейса пользователя Microsoft Windows.

**1.8.2. Системные требования**

Минимальные системные требования:

* 1 GB памяти
* 5 GB свободного дискового пространства
* процессор с тактовой частотой 1 GHz
* Операционная система Windows 7, 8, 8.1, 10.

**1.8.3. Эксплуатационные требования**

Система должна быть способна поддерживать минимум 15 одновременно работающих пользователей, связанных с общей базой данных и иметь возможность увеличить их количество на случай увеличения штата сотрудников предприятия, см. п. 3.2.

**1.9. Требования к документации**

**1.9.1. Руководство пользователя**

В документации должны быть представлены Руководства пользователей (по типам пользователей). Они должны содержать расшифровку всех используемых терминов, описания основных вариантов использования, включая альтернативные сценарии, а также подробный обзор интерфейса программы.

**1.9.2. Интерактивная справка**

Интерактивная справка необходима для разрешения возникших во время работы вопросов. В справке должна быть реализована возможность поиска информации, по ключевым словам, а также вариант представления информации по отдельным позициям меню программы. Справка должна содержать максимально полную и подробную информацию по работе системы.

**2. Спецификация требований к программному продукту**

**2.1. Введение**

**2.1.1. Назначение**

Назначение документа “Спецификация требований к программному продукту «Автоматизированная конвертация метаданных в систему Open Journal Systems из системы elibrary.ru» – определение требований к системе. Документ включает общее видение продукта. Данный документ является достаточным для разработки программного продукта.

**2.1.2. Цели создания системы, решаемые задачи и область применения**

**2.1.2.1. Краткое описание деятельности заказчика и существующей технологии**

Данный документ предназначен для определения, спецификации и согласования с заказчиком требований к системе **Open Journal Systems**. Документ включает общее видение продукта, формализованные требования, описание функционала и объектов системы. Данный документ является достаточным для разработки программного продукта.

**2.1.2.2. Цели создания системы. Эффекты от ее внедрения**

Цель создания этого раздела состоит в том, чтобы собрать, проанализировать и определить высокоуровневые потребности и возможности системы конвертации метаданных на основе платформы Оpen Journal Systems.  Документ акцентирует внимание на возможностях, необходимых совладельцам и целевым пользователям, и на том, почему эти потребности существуют. Подробности того, как система Open Journal Systems выполняет эти потребности, будут детализированы в прецедентах и дополнительных спецификациях.

**2.1.2.3. Назначение системы и область применения**

Программа «Автоматизированная конвертация метаданных в систему Open Journal Systems из системы elibrary.ru» является инструментов для загрузки статей в электронные библиотеки.

**2.1.3. Определения, акронимы и сокращения**

Основные определения приведены в документе «Приложение 1. Глоссарий проекта».

**2.1.4. Ссылки**

Требования к программе «Автоматизированная конвертация метаданных в систему Open Journal Systems из системы elibrary.ru».

**2.1.5. Краткое содержание**

Документ описывает высокоуровневые требования к системе «Автоматизированная конвертация метаданных в систему Open Journal Systems из системы elibrary.ru»*.* Указаны основные деловые преимущества рассматриваемого в Видении решения, сформулированы ключевые проблемы и способы их решения, приведены характеристики пользователей системы, возможности системы, ограничения, показатели качества и другие требования к продукту.

**2.2.** Общее описание

**2.2.1. Позиционирование программного продукта**

Данный программный продукт является расширением для системы Open Journal Systemss. Также данная система зависит от наличия интернет-соедениния, а также веб-браузера.

**2.2.2 Функциональность продукта**

Функционал программы «Автоматизированная конвертация метаданных из системы Open Journal Systems в систему elibrary.ru» представлен в виде диаграммы вариантов использования, которая показана на рис. 2.1.

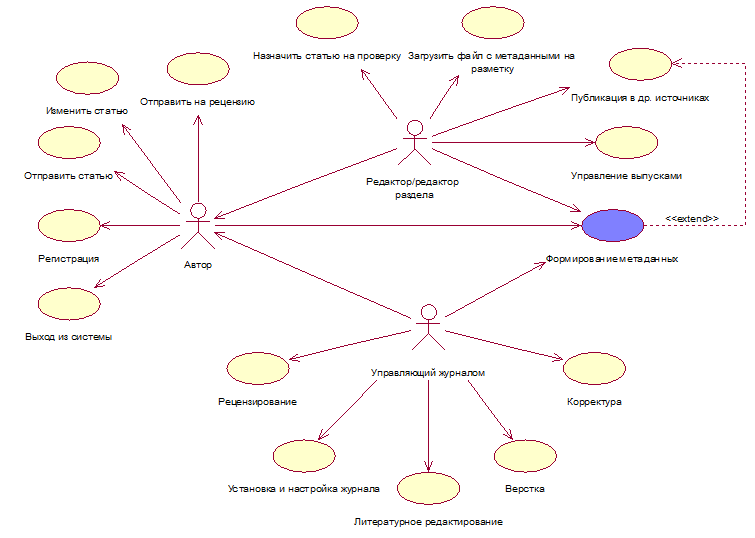


Рисунок 2.1. Диаграмма вариантов использования

Выявлены следующие типы актеров, описание которых будет представлено в табл. 2.1.

*Таблица 2.1*. Актеры

|  |  |
| --- | --- |
| ***Актер*** | ***Краткое описание*** |
| Управляющий журналом | Занимается техническими проблемами, контролирует систему, устанавливает обновления, следит за почтовым сервером. |
| Научный работник | Это автор статьи. Предоставляет свои личные данные, а также собственно-разработанные научные технические работы в целях дальнейшей публикации, предоставляет их рецензенту для проверки, затем передает статьи литературному редактору. Подает метаданные и внешнюю рецензию от независимого лица. |
| Выпускающий редактор | Принимает решение: публиковать статью или нет, назначает литературного редактора, корректора, приглашенных редакторов, редакторов разделов. Отправляет статью рецензенту. |
| Рецензент | Это автор рецензии, которым в изд. практике может быть сотрудник изд-ва или внештатный специалист в той области, к которой относится рецензируемое произведение. Оценивает статью, ее научное содержание, качество. Выдает свое мнение о доработках: аннотации, ключевые слова и т.д. |
| Выпускающий редактор | Знакомится с рецензией. Если все выполнено корректно, отправляет литературному редактору. Если же нет, то автор статьи и выпускающий редактор обращаются к редактору журнала (главный редактор). |
| Главный редактор | Может отказать автору, если что-то выполнено некорректно. Или же дает статью на рассмотрение приглашенным редакторам.  Если со статьей все в порядке, она отправляется литературному редактору, который правит стилистику и орфографию. |
| Литературный редактор | Занимается формированием метаданных для выпуска статьи в онлайн библиотеку или журнал. |
| Верстальщик | Верстает распределение статей по разделам в зависимости от их содержания, форматирует текст, занимается простановкой страниц. И теперь уже есть предварительный макет статьи. |
| Корректор | Просматривает макет. Если что-либо пропущено, то добавляет упущенное. А также смотрит и исправляет ошибки. |
| Управляющий журналом | Публикует статью. |
| Читатель | Читает, ссылается на статьи, оставляет комментарии. |
| Индексатор | Берет информацию с сайта журнала и вносит ее в elibrary.ru, КИНЦ и т.д. Уже имеются название и автор статьи. Далее индексатор должен получить doi. Регистрирует каждую статью на crossruff. Каждой статье присваивает номер. Затем в макет вставляются номера (doi). |
| Дизайнер | Оформляет обложку журнала, интерфейс. |
| Переводчик | Если же автор не владеет тремя языками, осуществляется перевод не всего материала, а только лишь метаданных (аннотации, ключевые слова). Происходит это между вёрсткой и коррекцией. |

В табл. 2.2. представлен перечень вариантов использования модели системы

**Варианты использования**

*Таблица 2.2*. Варианты использования

|  |  |  |  |
| --- | --- | --- | --- |
| ***Код*** | ***Основной актёр*** | ***Наименование*** | ***Формулировка*** |
| **B1** | Научный  работник | Предоставление  личных данных | Научный работник может предоставить свои личные данные для дальнейшей регистрации (ФИО, номер телефона, электронный адрес). |
| **B2** | Научный  работник | Предоставление  научной работы | Научный работник также предоставляет собственно-разработанные научные  технические работы в целях дальнейшей публикации. |
| **B3** | Научный  работник | Предоставление  внешней рецензии | Подает метаданные и внешнюю рецензию от независимого лица. |
| **A1** | Управляющий  журналом | Устранение  технических  проблем | Контролирует систему. |
| **A2** | Управляющий журналом | Устранение технических проблем | Устанавливает обновления. |
| **A3** | Управляющий журналом | Устранение технических проблем | Следит за почтовым сервером. |
| **C1** | Выпускающий редактор | Принятие решения | Публиковать статью или нет. |
| **C2** | Выпускающий редактор | Назначение редакторов | Назначает литературного редактора,  корректора, приглашенных редакторов, редакторов разделов. |
| **C3** | Выпускающий редактор | Отправка статьи | Отправляет статью рецензенту. |
| **D1** | Рецензент | Оценивание статьи | Оценивает статью, ее научное содержание, качество. |
| **D2** | Рецензент | Оценивание статьи | Выдает свое мнение о доработках:  аннотации, ключевые слова и т.д. |
| **C4** | Выпускающий редактор | Проверка рецензии | Знакомится с рецензией. Если все выполнено корректно, отправляет литературному редактору. Если же нет, то автор статьи и выпускающий редактор обращаются к редактору журнала (главный редактор). |
| **E1** | Главный редактор | Проверка статьи на корректность | Может отказать автору, если что-то  выполнено некорректно. Или же дает  статью на рассмотрение приглашенным  редакторам. |
| **F1** | Литературный редактор | Формирование статьи | Правит стилистику и орфографию. |
| **G1** | Верстальщик | Вёрстка | Верстает распределение статей по  разделам в зависимости от их содержания. |
| **G2** | Верстальщик | Форматирование текста | Форматирует текст, занимается  простановкой страниц. |
| **H1** | Корректор | Просмотр макета статьи | Просматривает макет. Если что-либо  пропущено, то добавляет упущенное. |
| **H2** | Корректор | Исправление  ошибок | А также смотрит и исправляет ошибки. |
| **A4** | Управляющий  журналом | Публикация  статьи | Публикует статью. |
| **I1** | Читатель | Чтение и отзывы | Читает, ссылается на статьи, оставляет  комментарии. |
| **J1** | Индексатор | Внесение  информации в  другие научные   библиотеки | Берет информацию с сайта журнала и  вносит ее в elibrary.ru, КИНЦ и т.д. |
| **J2** | Индексатор | Получение doi | Далее индексатор должен получить doi. |
| **J3** | Индексатор | Регистрация   статей | Регистрирует каждую статью на crossruff. |
| **J4** | Индексатор | Присваивание   номера | Каждой статье присваивает номер. Затем  в  макет вставляются номера (doi). |
| **K1** | Дизайнер | Оформление | Оформляет обложку журнала, интерфейс. |
| **L1** | Переводчик | Перевод | Если же автор не владеет тремя языками, осуществляется перевод не всего материала, а только лишь метаданных (аннотации, ключевые слова). |

**2.2.3. Характеристики пользователей**

Характеристика всех пользователей представлена в табл. 2.3.

*Таблица 2.3*. Характеристики пользователей

**Основное действующее лицо: Научный работник.**

|  |  |  |  |
| --- | --- | --- | --- |
| **B1** | Научный  работник | Предоставление  личных данных | Научный работник может предоставить  свои личные данные для дальнейшей  регистрации (ФИО, номер телефона,   электронный адрес). |

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: расширяется прецедентом «B2.

Предоставление научной работы. B3. Предоставление научной работы».

Краткое описание:

Для начала научный работник имеет возможность осуществить регистрацию.

**Основное действующее лицо: Научный работник.**

|  |  |  |  |
| --- | --- | --- | --- |
| **B2** | Научный  работник | Предоставление  научной работы | Научный работник также предоставляет  собственно-разработанные научные  технические работы в целях дальнейшей публикации. |

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: расширяется прецедентом «B1. Предоставление личных данных B3. Предоставление научной работы».

Краткое описание:

После успешной регистрации научный работник должен предоставить статью в целях дальнейшей обработки и публикации.

**Основное действующее лицо: Научный работник.**

|  |  |  |  |
| --- | --- | --- | --- |
| **B3** | Научный  работник | Предоставление  внешней рецензии | Подает метаданные и внешнюю рецензию  от независимого лица. |

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: расширяется прецедентом «B1. Предоставление личных данных. B2. Предоставление научной работы».

Краткое описание:

Литературный редактор получает статью научного работника от рецензента, составившего рецензию, после чего занимается ее редактированием. Затем добавляет и публикует статью в систему.

**Основное действующее лицо: Управляющий журналом.**

|  |  |  |  |
| --- | --- | --- | --- |
| **A1** | Управляющий  журналом | Устранение  технических  проблем | Контролирует систему. |

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: расширяется прецедентом «A2-A3. Устранение технических проблем. A4. Публикация статьи.»

Краткое описание:

Занимается устранением технических проблем. В частности, контролированием системы.

**Основное действующее лицо: Управляющий журналом.**

|  |  |  |  |
| --- | --- | --- | --- |
| **A1** | Управляющий  журналом | Устранение  технических  проблем | Контролирует систему. |

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: расширяется прецедентом «A1-A3. Устранение технических проблем. A4. Публикация статьи.»

Краткое описание:

Занимается устранением технических проблем. В частности, установлением обновлений.

**Основное действующее лицо: Управляющий журналом.**

|  |  |  |  |
| --- | --- | --- | --- |
| **A3** | Управляющий журналом | Устранение технических проблем | Следит за почтовым сервером. |

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: расширяется прецедентом «A1-A2. Устранение технических проблем. A4. Публикация статьи.»

Краткое описание:

Занимается устранением технических проблем. В частности, следит за почтовым сервером.

**Основное действующее лицо: Выпускающий редактор.**

|  |  |  |  |
| --- | --- | --- | --- |
| **C1** | Выпускающий редактор | Принятие решения | Публиковать статью или нет. |

Другие участники прецедента: литературный редактор, корректор, приглашенные редакторы, редакторы разделов.

Связи с другими вариантами использования: расширяется прецедентом «C2. Назначение редакторов. C3. Отправка статьи. C4. Проверка рецензии.»

Краткое описание:

Выпускающий редактор (ответственный секретарь) принимает решение, публиковать ли статью.

**Основное действующее лицо: Выпускающий редактор.**

|  |  |  |  |
| --- | --- | --- | --- |
| **C2** | Выпускающий редактор | Назначение   редакторов | Назначает литературного редактора,  корректора, приглашенных редакторов,  редакторов разделов. |

Другие участники прецедента: литературный редактор, корректор, приглашенные редакторы, редакторы разделов.

Связи с другими вариантами использования: расширяется прецедентом «C1. Принятие решения. C3. Отправка статьи. C4. Проверка рецензии.»

Краткое описание:

Также он имеет право назначать литературного редактора, корректора, приглашенных редакторов, редакторов разделов.

**Основное действующее лицо: Выпускающий редактор.**

|  |  |  |  |
| --- | --- | --- | --- |
| **C3** | Выпускающий редактор | Отправка статьи | Отправляет статью рецензенту. |

Другие участники прецедента: литературный редактор, корректор, приглашенные редакторы, редакторы разделов.

Связи с другими вариантами использования: расширяется прецедентом «C1. Принятие решения. C2. Назначение редакторов. C4. Проверка рецензии.»

Краткое описание:

Если всё выполнено корректно, выпускающий редактор отправляет статью литературному редактору.

**Основное действующее лицо: Рецензент.**

|  |  |  |  |
| --- | --- | --- | --- |
| **D1** | Рецензент | Оценивание статьи | Оценивает статью, ее научное содержание,  качество. |

Другие участники прецедента: выпускающий редактор.

Связи с другими вариантами использования: расширяется прецедентом «D2. Оценивание статьи».

Краткое описание:

Рецензент должен оценить статьи, научное содержание статьи и ее качество.

**Основное действующее лицо: Рецензент.**

|  |  |  |  |
| --- | --- | --- | --- |
| **D2** | Рецензент | Оценивание статьи | Выдает свое мнение о доработках:  аннотации, ключевые слова и т.д. |

Другие участники прецедента: выпускающий редактор.

Связи с другими вариантами использования: расширяется прецедентом «D1. Оценивание статьи».

Краткое описание:

Должен выдать мнение о доработках статьи.

**Основное действующее лицо: Выпускающий редактор.**

|  |  |  |  |
| --- | --- | --- | --- |
| **C4** | Выпускающий редактор | Проверка рецензии | Знакомится с рецензией. Если все  выполнено корректно, отправляет  литературному редактору. Если же нет, то  автор статьи и выпускающий редактор  обращаются к редактору журнала  (главный редактор). |

Другие участники прецедента: литературный редактор, корректор, приглашенные редакторы, редакторы разделов.

Связи с другими вариантами использования: расширяется прецедентом «C1. Принятие решения. C2. Назначение редакторов. C3. Отправка статьи.»

Краткое описание:

Получает статью от рецензента, пишет рецензию по форме. Если все выполнено корректно, отправляет литературному редактору. Если же нет, то автор статьи и выпускающий редактор обращаются к редактору журнала (главный редактор).

**Основное действующее лицо: Главный редактор.**

|  |  |  |  |
| --- | --- | --- | --- |
| **E1** | Главный редактор | Проверка статьи на корректность | Может отказать автору, если что-то выполнено некорректно. Или же дает  статью на рассмотрение приглашенным  редакторам. |

Другие участники прецедента: научный работник, приглашенные редакторы.

Связи с другими вариантами использования: отсутствуют

Краткое описание.

Главный редактор также проверяет статью на корректность. И может отказать автору, если что-то выполнено некорректно. Или же дает статью на рассмотрение приглашенным редакторам.

**Основное действующее лицо: Литературный редактор.**

|  |  |  |  |
| --- | --- | --- | --- |
| **F1** | Литературный редактор | Формирование статьи | Правит стилистику и орфографию. |

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: отсутствуют

Краткое описание:

Литературный редактор должен проверить статью на ошибки, править стилистику и орфографию.

**Основное действующее лицо: Верстальщик.**

|  |  |  |  |
| --- | --- | --- | --- |
| **G1** | Верстальщик | Вёрстка | Верстает распределение статей по  разделам в зависимости от их содержания. |

Другие участники прецедента: литературный редактор.

Связи с другими вариантами использования: расширяется прецедентом «G2. Форматирование текста».

Краткое описание:

Верстальщик верстает распределение статей по разделам в зависимости от их содержания

**Основное действующее лицо: Верстальщик.**

|  |  |  |  |
| --- | --- | --- | --- |
| **G2** | Верстальщик | Форматирование текста | Форматирует текст, занимается  простановкой страниц. |

Другие участники прецедента: литературный редактор.

Связи с другими вариантами использования: расширяется прецедентом «G1. Вёрстка».

Краткое описание:

Верстальщик также должен отформатировать текст и проставить страницы.

**Основное действующее лицо: Корректор.**

|  |  |  |  |
| --- | --- | --- | --- |
| **H1** | Корректор | Просмотр макета статьи | Просматривает макет. Если что-либо пропущено, то добавляет упущенное. |

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: расширяется прецедентом «H2. Исправление ошибок».

Краткое описание:

Корректор просматривает предварительный макет, в котором ошибки всё же имеются. И его задача состоит в просмотре данного макета и добавлении упущенного.

**Основное действующее лицо: Корректор.**

|  |  |  |  |
| --- | --- | --- | --- |
| **H2** | Корректор | Исправление  ошибок | Проверяет на наличие ошибок и исправляет. |

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: расширяется прецедентом «H1. Просмотр макета статьи».

Краткое описание:

Также корректор при обнаружении ошибок, должен их исправить.

**Основное действующее лицо: Управляющий журналом.**

|  |  |  |  |
| --- | --- | --- | --- |
| **A4** | Управляющий  журналом | Публикация  статьи | Публикует статью. |

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: расширяется прецедентом «A1-A3. Устранение технических проблем».

Краткое описание:

Управляющий журналом наконец публикует статью.

**Основное действующее лицо: Читатель.**

|  |  |  |  |
| --- | --- | --- | --- |
| **I1** | Читатель | Чтение и отзывы | Читает, ссылается на статьи, оставляет  комментарии. |

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: отсутствуют

Краткое описание:

Читатель имеет право прочесть заинтересовавшую его статью, ссылаться на другие работы, а также оставлять свои комментарии.

**Основное действующее лицо: Индексатор.**

|  |  |  |  |
| --- | --- | --- | --- |
| **J1** | Индексатор | Внесение  информации в другие научные библиотеки | Берет информацию с сайта журнала и  вносит ее в elibrary.ru, КИНЦ и т.д. |

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: расширяется прецедентом «J2. Получение doi. J3. Регистрация статей. J4. Присваивание номера.».

Краткое описание:

Индексатор берет информацию с сайта журнала и вносит ее в elibrary.ru, КИНЦ и т.д.

**Основное действующее лицо: Индексатор.**

|  |  |  |  |
| --- | --- | --- | --- |
| **J2** | Индексатор | Получение doi | Далее индексатор должен получить doi. |

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: расширяется прецедентом «J1. Внесение информации в другие научные библиотеки. J3. Регистрация статей. J4. Присваивание номера.».

Краткое описание:

Далее индексатору необходимо получить doi.

**Основное действующее лицо: Индексатор.**

|  |  |  |  |
| --- | --- | --- | --- |
| **J3** | Индексатор | Регистрация   статей | Регистрирует каждую статью на crossruff. |

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: расширяется прецедентом «J1. Внесение информации в другие научные библиотеки. J2. Получение doi. J4. Присваивание номера.».

Краткое описание:

Затем индексатор регистрирует каждую статью на crossruff.

**Основное действующее лицо: Индексатор.**

|  |  |  |  |
| --- | --- | --- | --- |
| **J4** | Индексатор | Присваивание   номера | Каждой статье присваивает номер. Затем в  макет вставляются номера (doi). |

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: расширяется прецедентом «J1. Внесение информации в другие научные библиотеки. J2. Получение doi. J3. Регистрация статей.».

Краткое описание:

После всего, каждой статье присваивает номер. Затем в макет вставляются номера (doi).

**Основное действующее лицо: Дизайнер.**

|  |  |  |  |
| --- | --- | --- | --- |
| **K1** | Дизайнер | Оформление | Оформляет обложку журнала, интерфейс. |

Другие участники прецедента: отсутствуют

Связи с другими вариантами использования: отсутствуют

Краткое описание:

Дизайнер должен оформить обложку журнала и поработать над дизайном.

**Основное действующее лицо: Переводчик.**

|  |  |  |  |
| --- | --- | --- | --- |
| **L1** | Переводчик | Перевод | Если же автор не владеет тремя языками,   осуществляется перевод не всего  материала, а только лишь метаданных  (аннотации, ключевые слова). |

Другие участники прецедента: автор статьи.

Связи с другими вариантами использования: отсутствуют

Переводчик переводит статьи на другие языки (русский/кыргызский/английский). Если же автор не владеет тремя языками, осуществляется перевод не всего материала, а только лишь метаданных (аннотации, ключевые слова).

**2.2.4. Ограничения**

Метаданные, которые подготавливаются пользователями, должны быть тщательно подобраны исходя из требований программы для получения корректных результатов.

**2.2.5. Предположения и зависимости**

Система будет функционировать правильно в том случае, когда все необходимое для нее оборудование и программное обеспечение доступно и находится в рабочем состоянии.

**2.3.  Специфические требования**

**2.3.1. Требования**

**Пользовательские интерфейсы**

Для корректной работы автоматизированной системы конвертации метаданных нужно иметь корректный файл с метаданными. Для этого нужно перейти на любой сайт, основанный на системе Open Journal Systems, авторизоваться, перейти в раздел «журнал» и нажать «экспортировать статью».

**2.3.2. Функциональные требования**

**Описание ключевых прецедентов**

**Поиск ключевых вариантов использования**

Анализ сформулированных вариантов использования показывает, что целесообразно выбрать те, варианты использования, которые связаны непосредственно с конвертацией метаданных.

Для дальнейшей детализации выбраны прецеденты:

D2. Создание рецензии по форме

G1. Верстка: распределение статей по разделам в зависимости от их содержания

**Прецедент D2. Создание рецензии по форме**

**Краткое описание**

После того, как рецензент оценивает полученную статью, ее научное содержание и качество, он заполняет рецензию по форме.

**Поток событий**

Прецедент начинается, когда выпускающий редактор отправляет статью выбранному рецензенту.

**Базовый поток** – Создание рецензии по форме

1.  Выпускающий редактор принимает решение - публиковать статью или нет.

2.  Далее выпускающий редактор назначает рецензента для выбранной статьи.

3.  Отправляет статью рецензенту.

4.  Рецензент получает данную статью.

5.  После оценивания полученной статьи, ее научного содержания и качества, он заполняет рецензию по форме.

**Альтернативныепотоки**

* Выпускающий редактор решил не публиковать статью

Если выпускающий редактор принял решение не публиковать статью, но при этом статья соответствует данной научной тематике, то литературный редактор назначает независимого приглашенного редактора, оценивающего статью.

* Статью решили не публиковать

Если же приглашенный редактор после оценивания статьи, утверждает, что она не подходит по научной тематике, то статью не публикуют. Use Case не выполняется.

**Прецедент G1. Верстка: распределение статей по разделам в зависимости от их содержания**

**Краткое описание**

Верстальщик занимается распределением статей по разделам в зависимости от их содержания, форматирует текст, занимается простановкой страниц.

Далее идет работа корректора, который получает почти готовый макет и проверяет его на ошибки. После чего, управляющий журналом публикует статью.

**Поток событий**

Прецедент начинается, когда литературный редактор заканчивает формирование статьи и рецензент заполняет рецензию.

**Базовый поток** – Распределение статей по разделам в зависимости от их содержания

1.  Верстальщик занимается распределением статьи по разделам.

2.  Форматирует текст.

3.  Осуществляет простановку страниц.

4.  Корректор получает статью.

5.  Корректор проверяет статью на корректность (выявление ошибок).

6.  Если же в статье что-либо упустили или имеются ошибки, корректор это исправляет.

7.  Далее статью получает управляющий журналом.

8.  После получения статьи управляющий журналом ее публикует.

**Альтернативные потоки**

* Готовность статьи для корректировки.

В таком случае статья готова для дальнейшей корректировки корректором.

* В случае обнаружения ошибок.

Если же при формировании статьи были допущены некие ошибки, то корректор занимается ее редактированием и добавлением пропущенных действий.

1. **Проектирование и конструирование программного обеспечения**

#### Разработка диаграммы классов

Диаграмма классов (class diagram) используется для представления статической структуры модели системы в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать, в частности, различные отношения между отдельными объектами тематической области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. Эта диаграмма не показывает информацию о временных аспектах работы системы (рис. 3.1). С этой точки зрения диаграмма классов является дальнейшим развитием концептуальной модели проектируемой системы [8].

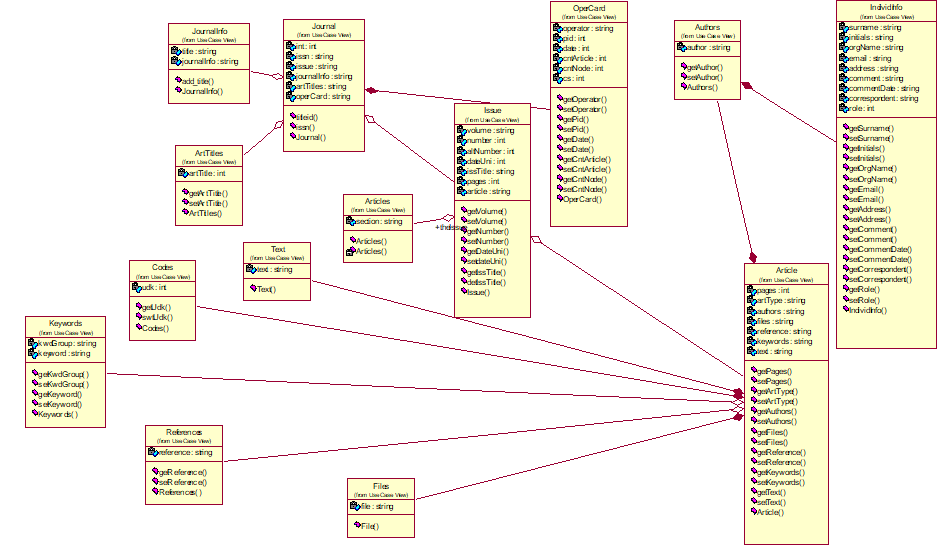


Рисунок 3.1. Диаграмма классов система АКМ

* 1. **Определение классов-сущностей**

Класс - сущность содержит информацию, которая должна храниться постоянно и не уничтожается с уничтожением объектов данного класса или после прекращения работы моделируемой системы. Этот класс соответствует отдельной таблице базы данных. В этом случае его атрибуты являются полями таблицы, а операции – присоединенными или хранимыми процедурами. Этот класс только принимает сообщения от других классов модели [9].

В данной системе в виде классов-сущностей представлены классы, которые отвечают за хранение данных о журнале, авторах и пользователях.

В табл. 3.1 перечислены основные классы сущности и их краткие описания.

*Таблица. 3.1* Основные классы сущности и их краткие описания

|  |  |
| --- | --- |
| **Название класса** | **Краткое описание** |
| ArticleFile | Класс, содержащий все данные о статье. |
| Article | Класс, содержащий все данные статьи. |
| Author | Класс, содержащий все данные об авторах. |
| ArticleAuthorReference | Класс, содержащий данные, связывающие автора и статью. |
| Roles | Класс, содержащий роли пользователей. |
| Permission | Класс, содержащий информацию о правах пользователей. |
| Users | Класс, содержащий информацию о пользователях. |
| RolesPermission | Класс, содержащий данные, связывающий роли и права. |

На рис. 3.2 представлена диаграмма классов-сущностей базы данных и связи и между ними.

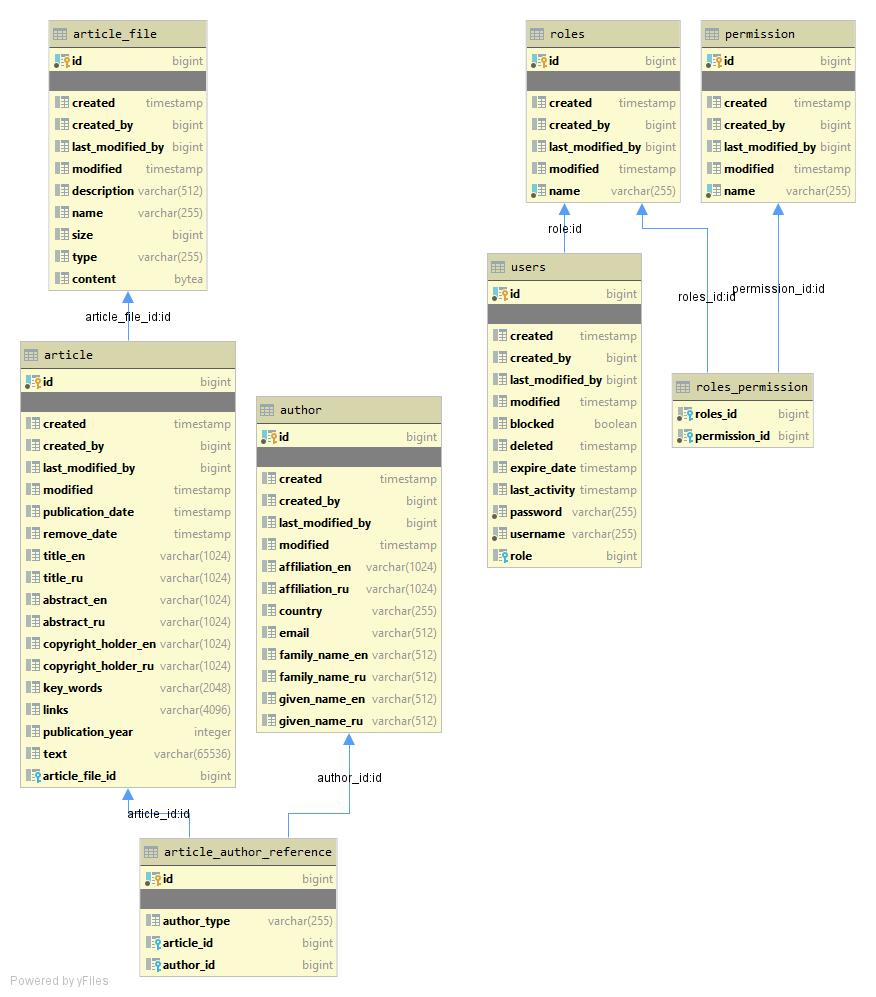


Рисунок 3.2. Классы-сущности и связи между ними

### Разработка диаграмм компонентов

Диаграмма компонентов позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Пунктирные стрелки, соединяющие модули, показывают отношения взаимозависимости, аналогичные тем, которые имеют место при компиляции исходных текстов программ. Основными графическими элементами диаграммы компонентов являются компоненты, интерфейсы и зависимости между ними [10].

Диаграмма компонентов разрабатывается для следующих целей:

* Визуализации общей структуры исходного кода программной системы.
* Спецификации исполнимого варианта программной системы.
* Обеспечения многократного использования отдельных фрагментов программного кода.
* Представления концептуальной и физической схем баз данных.

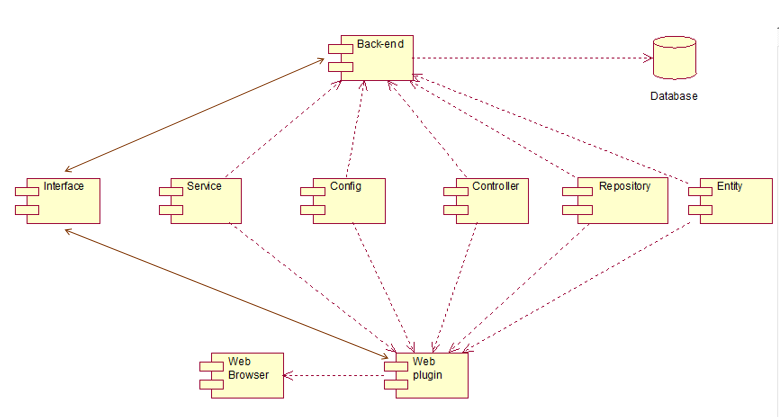


Рисунок 3.3. Диаграмма компонентов системы АКМ

На данной диаграмме (рис. 3.3) указаны компоненты за основу были взяты программные компоненты, которые будут представлены в программной системе, разработанной в IntelliJ IDEA [11].

### 3.4. Разработка диаграмм последовательностей

Диаграмма последовательности отражает сценарий поведения в системе, показывает взаимодействие между объектами и их отношения, а также обеспечивает наглядное представление порядка передачи сообщений, то есть выделяет упорядочение сообщений по времени [12].

На рисунках 3.4, 3.5 и 3.6 представлены диаграммы последовательностей процессов авторизации, поиска статьи по данным и добавления проекта [13].

**Поиск статьи по данным**

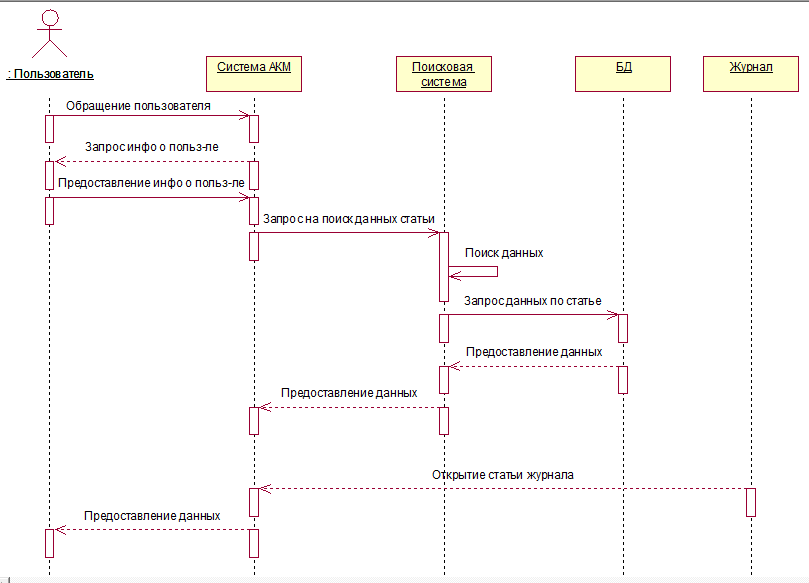


Рисунок 3.4. Диаграмма последовательности поиска статьи

**Описание последовательности:**

1. Пользователь вводит данные для поиска.
2. Системы получает данные и роль пользователя.
3. Идет запрос данных в базе данных.
4. Происходит поиск данных по статье.
5. Данные передаются пользователю.

**Авторизация**

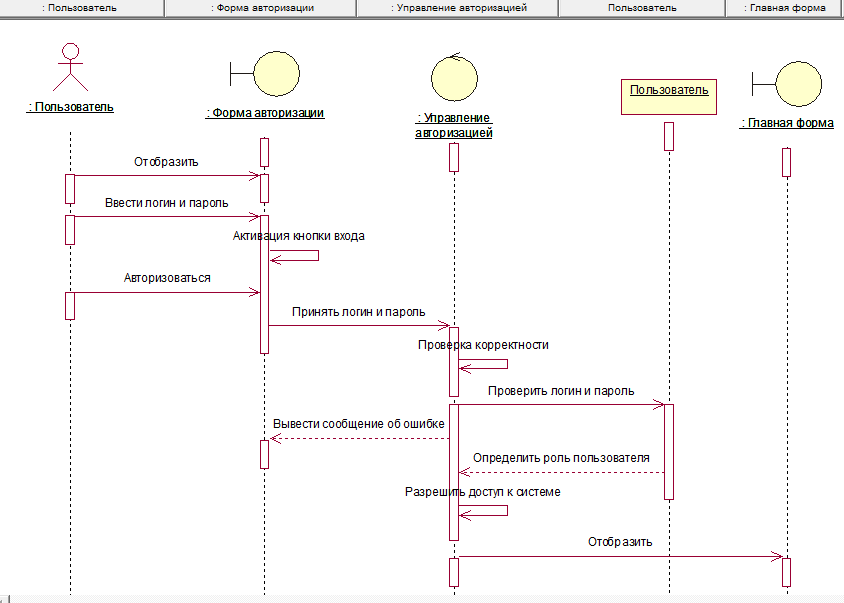
****

Рисунок 3.5. Диаграмма последовательности авторизации

**Описание последовательности:**

1. Отображение формы для входа.
2. Ввод логина и пароля в форму.
3. Проверить корректность данных.
   1. Вернуть сообщение об ошибке.
4. Определение роли пользователя.
5. Переход на главную страницу.

**Добавление проекта**

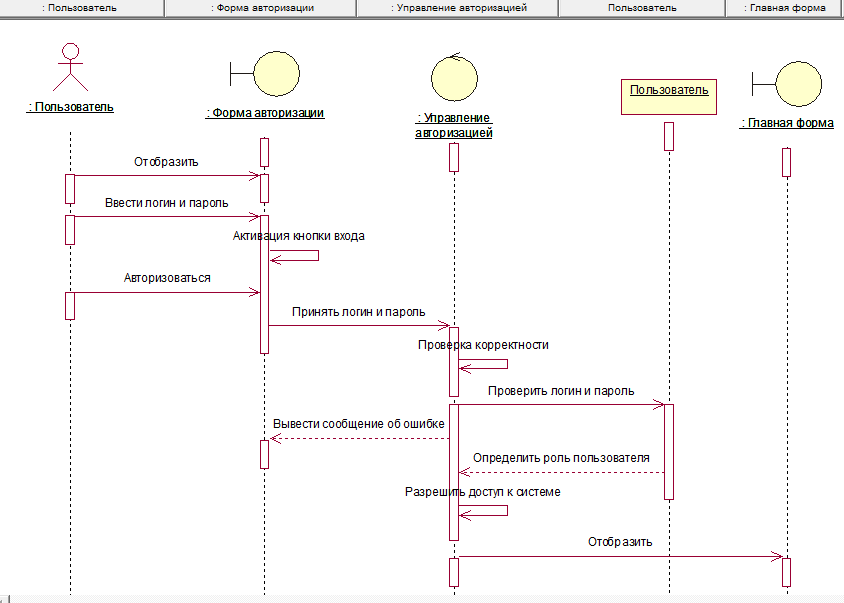


Рисунок 3.5. Диаграмма последовательности авторизации

**Описание последовательности:**

1. Обработать события открытие формы для выбора файла.
2. Обработать выбор файла.
3. Обработать загрузку файла.
4. Проверить наличие в базе данных проекта с таким названием.
   1. Если такой проект существует, то выдать сообщение.
   2. Если такого проекта нет, то добавить.
5. Обработать события о вводе данных
   1. Занести данные
   2. Подтвердить добавление данных

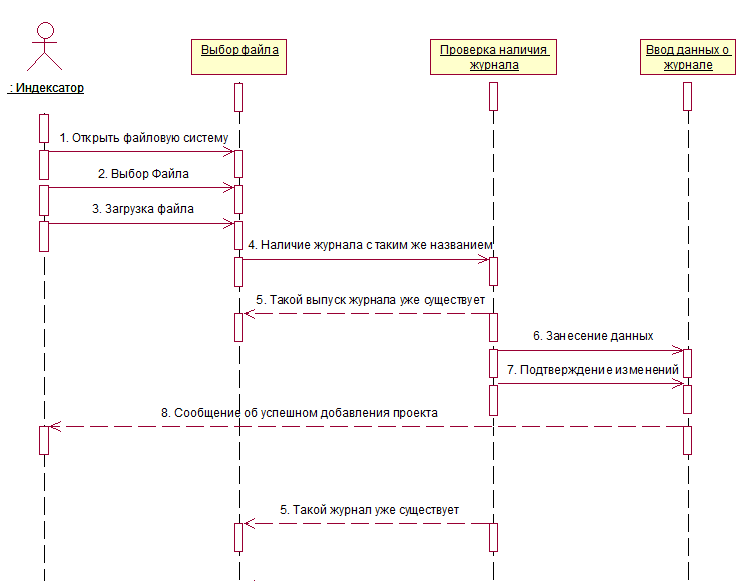
****

Рисунок 3.6. Диаграмма последовательности загрузки файла

# 4. Разработка тестов и тестирование программного продукта

## 4.1. Разработка плана тестирования

### 4.1.1. Введение

Данный документ описывает план тестирования для программы «Автоматизированная конвертация метаданных в систему Open Journal Systems из системы elibrary.ru» согласно спецификации. Полная стратегия тестирования программного обеспечения состоит из следующих типов испытаний и выполняется в следующем порядке:

1. Тестирование компонентов (модульное тестирование). Тестируются все программные компоненты (при этом проверяется покрытие кода тестами). Анализ кода.
2. Тестирование интеграции. Тестируется программное обеспечение, чтобы гарантировать, что компоненты взаимодействуют правильно.

**4.1.2. Область тестирования**

Целью этого тестирования является определить, соответствует ли разработанное программное обеспечение заявленным функциональным требованиям, а также выявить ошибки и представить их исправления, улучшив тем самым качество разработанного программного обеспечения.

**4.1.3. Начальные условия (пререквизиты)**

Задачи, которые должны быть решены перед началом тестирующей деятельности:

1. Имеется законченная программная спецификация.
2. Работающее программное обеспечение.
3. Обнаруженные проблемы фиксируются в журнале тестирования.
4. Все необходимые компоненты находятся в рабочем состоянии.

Составлен следующий набор тестов (табл. 4.1):

Таблица 4.1. Набор тестов для тестирования системы

|  |  |  |
| --- | --- | --- |
| № | Тестовый вариант | Ожидаемый результат |
| 1 | Регистрация пользователя | Пользователь успешно добавлен в базу данных |
| 2 | Авторизация пользователя | Пользователь успешно вошел в систему |
| 3 | Сохранение данных пользователя после редактирования пользователей | Настройки корректно сохранены в базу данных |
| 4 | Применение сохраненных настроек | Результаты успешно отправлены в корректной форме |
| 5 | Сохранение набора данных | Данные корректно сохранены |
| 6 | Добавление проекта | Проект успешно добавлен |
| 7 | Логирование | Успешное отображение логов |

### 4.1.4. Приоритеты тестирования

Проверки перечислены в порядке уменьшения приоритетного уровня:

1. Функциональность – все функции программы выполняются как ожидалось?
2. Удобство и простота использования – является ли программа удобной в использовании и обладает ли программа понятным интерфейсом?
3. Надежность – является ли реакция на некорректное использование адекватной?

### 4.1.5. Методы тестирования

Будут использоваться **тестовые сценарии** – сценарии вариантов использования (с предопределенным вводом и ожидаемыми выходными данными), варианты использования взяты из диаграммы вариантов использования [14].

### 4.1.6. Среда тестирования

Для тестирования данного программного продукта необходимо следующее оборудование:

Персональный компьютер со следующей конфигурацией:

* Intel Core i3, 8Gb
* Операционная система Microsoft Windows 10
* Установленные программы IntelliJ IDEA, Google Chrome

## 4.2. Модульное тестирование

**Модульное тестирование** - это процесс программирования, который позволяет проверять правильность блоков исходного кода, правильную работу программных модулей, а также наборов одного или нескольких программных модулей вместе с соответствующими управляющими данными [15].

Суть метода заключается в создании методов тестирования для тестирования отдельного метода в программе. Таким образом, можно проверить правильность кода при его изменении и добавлении функциональности в программу. Кроме того, модульные тесты позволяют обнаруживать ошибки в работе программных алгоритмов, что, в свою очередь, повышает качество и надежность программного обеспечения [16].

Цель модульного тестирования — выделить отдельные модули программы и проверить работу этих модулей в соответствии с определенными входными данными [17].

### 4.2.1. Тестовые случаи

**Название класса:** AuthControllerRest

**Описание тестового случая**:

Тестирование метода register() – проверка создания нового пользователя в базе данных (табл. 4.2).

*Таблица 4.2*. Тестовые варианты метода register()

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | *Что проверяется* | Ожидаемый результат |
| 1 | Логин: user | Наличие в базе данных пользователя с таким же логином   * Пользователь с таким логином не найден | Сообщение об успешной регистрации. |
| 2 | Логин: user | Наличие в базе данных пользователя с таким же логином   * Пользователь с таким логином найден | Сообщение, что пользователь с таким логином уже существует. |

**Название класса:** AuthControllerRest

**Описание тестового случая**:

Тестирование метода login() – проверка создания нового пользователя в базе данных (табл. 4.3).

*Таблица 4.3*. Тестовые варианты метода login()

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | *Что проверяется* | Ожидаемый результат |
| 1 | Логин: user | Корректность пароля   * Пароль соответствует | Переход на главную страницу. |
| 2 | Логин: user | Корректность пароля   * Пароль не соответствует | Сообщение, что пароль не верный. |

**Название класса: ArticleController**

**Описание тестового случая**:

Тестирование метода submit () – проверка добавления статьи в базу данных (табл. 4.4).

*Таблица 4.4* Тестовые варианты методаsubmit ()

|  |  |  |  |
| --- | --- | --- | --- |
| № | Входные данные | *Что проверяется* | Ожидаемый результат |
| 1 | file | Есть ли в базе данных уже файл с такими же данными   * Такого файла нет | Добавление содержимого в бд. |
| 2 | file | Есть ли в базе данных уже файл с такими же данными   * Такого файл уже есть | Сообщение, что такие данные уже есть |

# 5. Руководство пользователя

## 5.1. Назначение системы

Автоматизированная система конвертации метаданных предназначена для подготовки публикации метаданных из системы Open Journal Systems в elibrary.ru. основным пользователем может быть индексатор, выполняющий данную работу, научный работник, проверяющий корректность данных.

Система представляет собой расширение для системы Open Journal Systems. Программа обладает понятным интерфейсом и удобным в использовании функционалом и может и может извлекать метаданные из файла, добавлять новые проекты, управлять пользователями.

Основная цель системы- оптимизация и автоматизация работы конвертации метаданных.

## 5.2. Условия применения системы

Пользователям системы необходимо иметь персональный компьютер со следующими минимальными системными требованиями:

* 1 GB оперативной памяти
* 10 GB свободного дискового пространства
* процессор с тактовой частотой 1 GHz
* Операционная система Windows 7, 8, 8.1, 10.
* Веб-браузер Opera, Mozilla Firefox, Google Chrome. Версии должны быть не раньше 2019 года.

Также пользователи должны обладать базовыми навыками работы с персональным компьютером

## 5.3. Подготовка системы к работе

**Запуск системы**

Для запуска системы нужно:

* + - 1. Перейти по ссылке localhost:8080.
      2. После этого откроется главная страница веб-приложения (рис. 5.1).

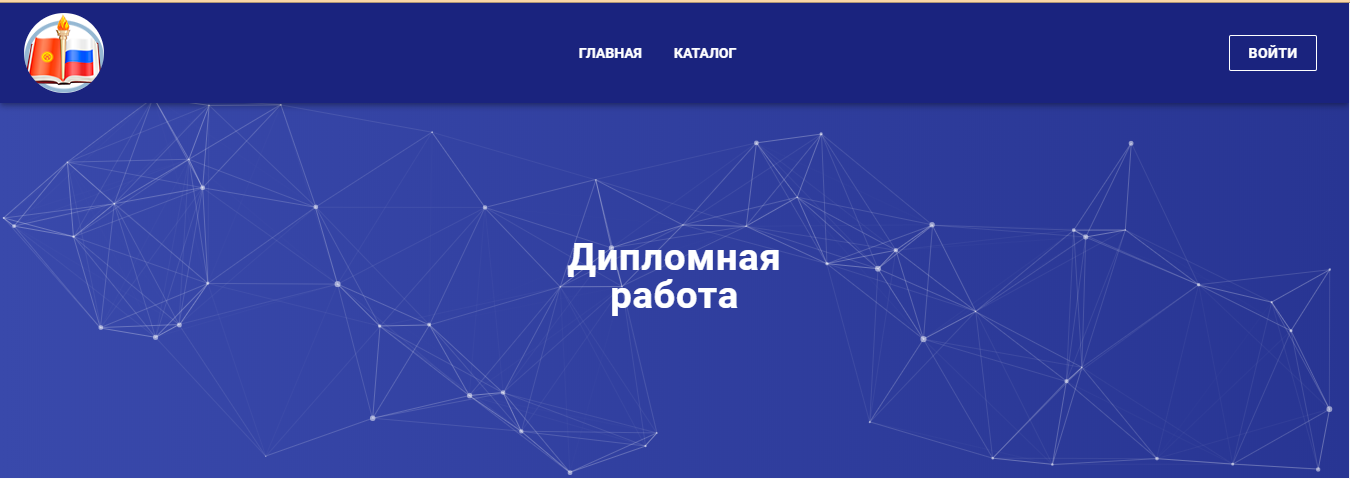


Рисунок 5.1. Главная страница системы

**Вход**

Для входа нужно ввести свои данные: логин и пароль (рис. 5.2), а если нет аккаунта, то нужно создать новый аккаунт.

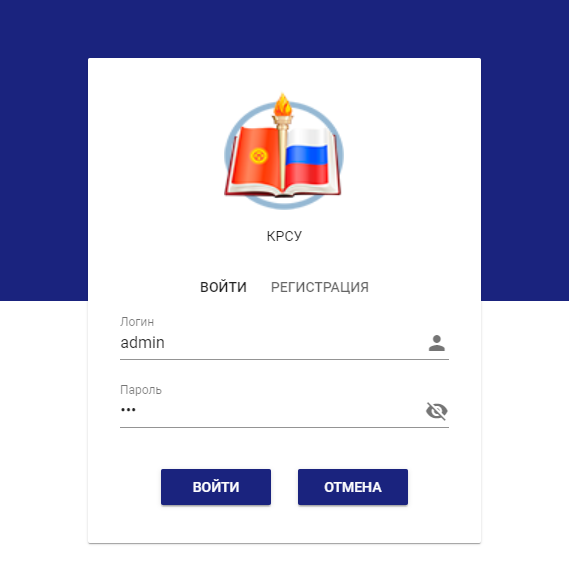


Рисунок 5.2. Вход в систему

**Регистрация**

При регистрации открывается форма регистрации, где нужно ввести свой логин и пароль (рис. 5.3).

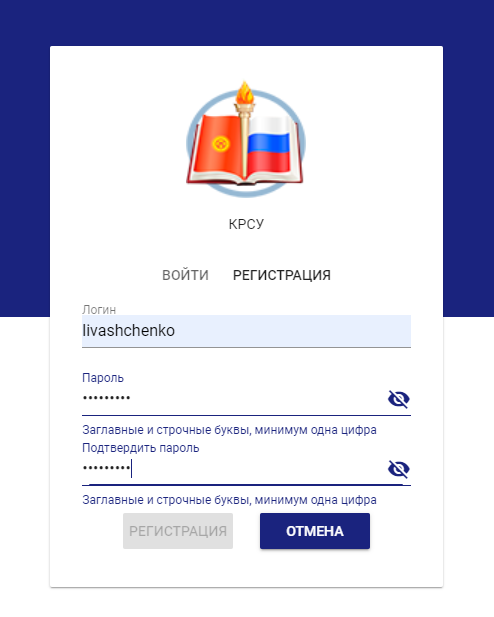


Рисунок 5.3. Регистрация в системе

**Авторизация**

После успешного входа в систему, открывается главная страница сайта, но уже с возможностью просмотра каталога статей (рис. 5.4).

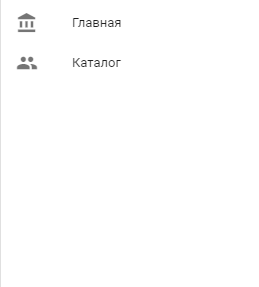


Рисунок 5.4. Главная страница авторизированного пользователя

**Просмотр каталога статей в роли пользователя**

После перехода в каталог, отображаются уже добавленные статьи (рис. 5.5). Также можно скачать исходный файл статьи, нажав на значок загрузки.

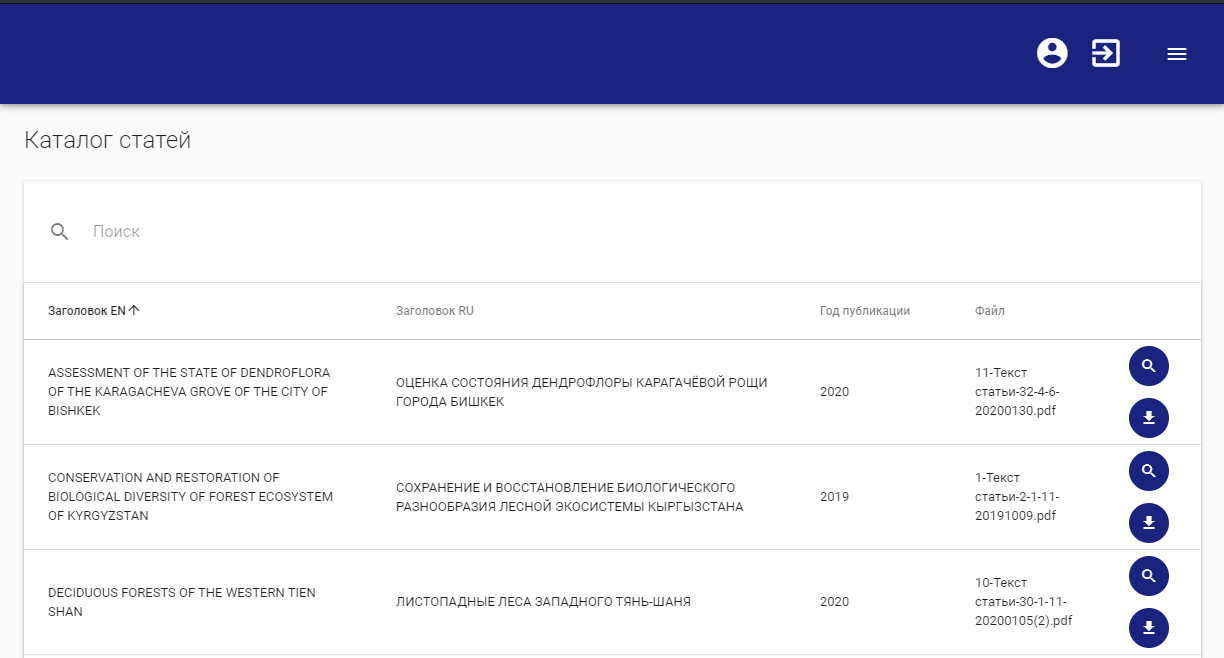


Рисунок 5.5. Отображение каталога статей

**Подробный просмотр статьи**

Для подробного просмотра статьи нужно нажать на значок лупы в каталоге статей, после чего отобразится подробная информация о статье (рис. 5.6).

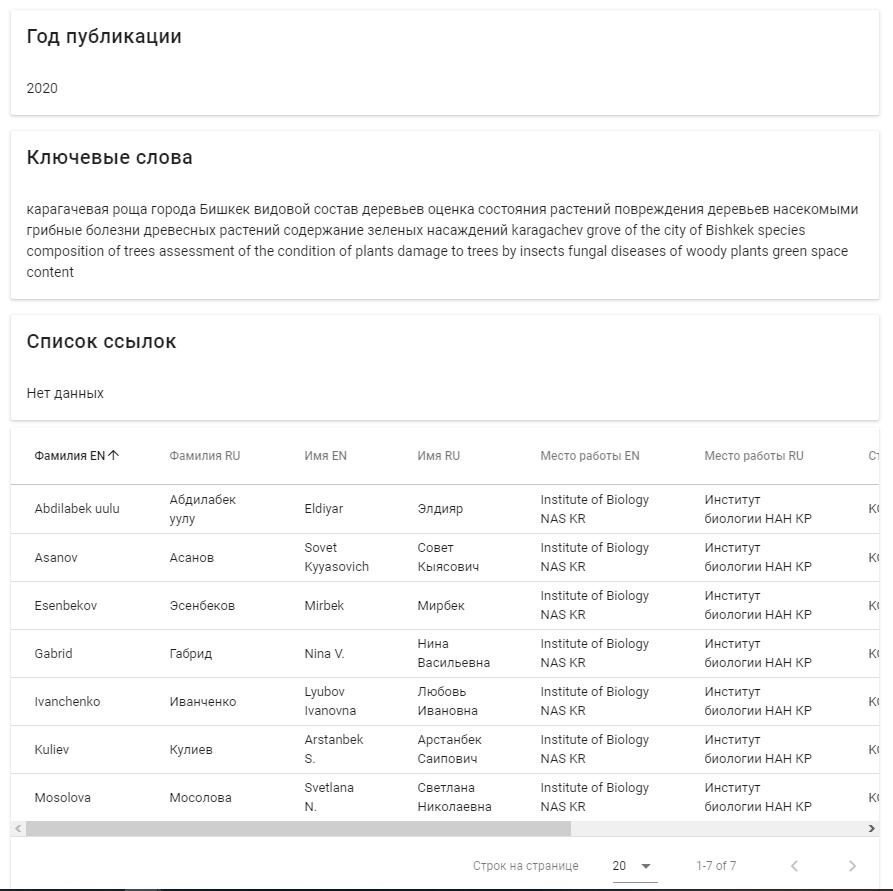
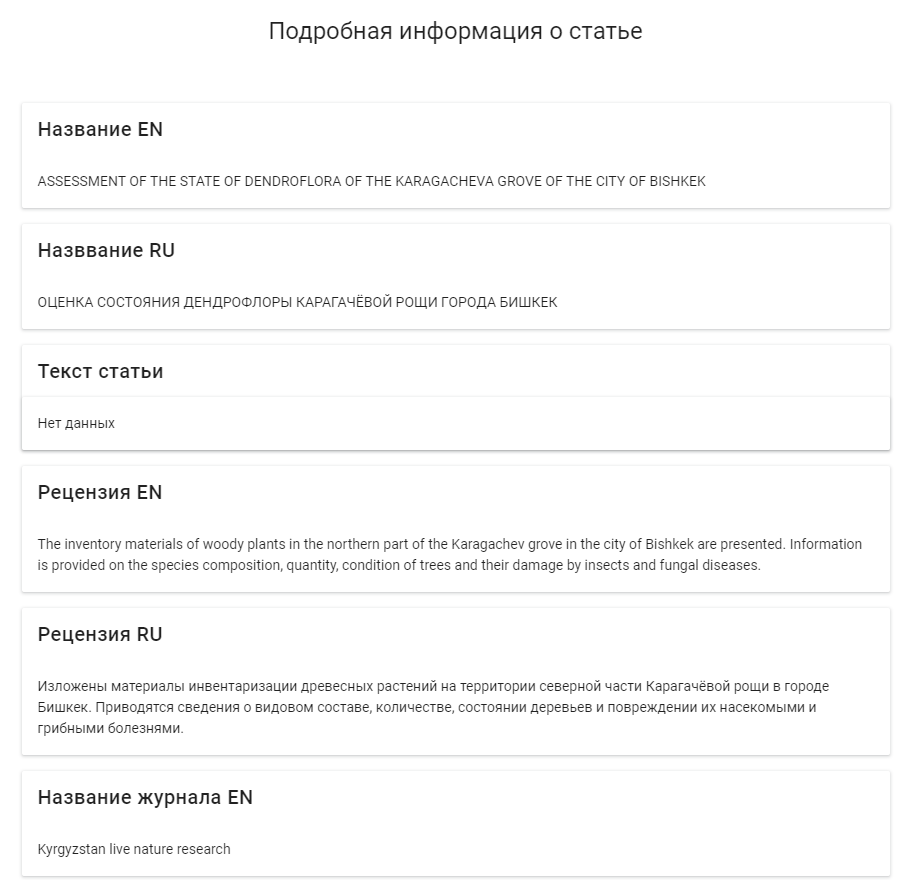


Рисунок 5.6. Подробное отображение информации о статье

**Вход от лица администратора**

Также можно зайти в роли администратора. Только администратор может добавлять проекты и редактировать их (рис. 5.6 и рис. 5.7).

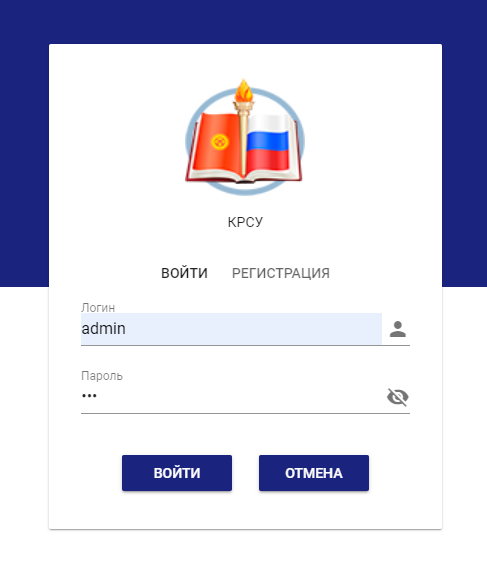


Рисунок 5.6. Форма входа для администратора

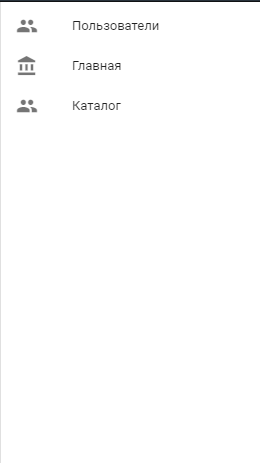


Рисунок 5.7. Главная страница для администратора

**Управление пользователями в роли администратора**

Администратор может управлять пользователями (удалять, блокировать назначать администратором, редактировать данные) (рис. 5.7).

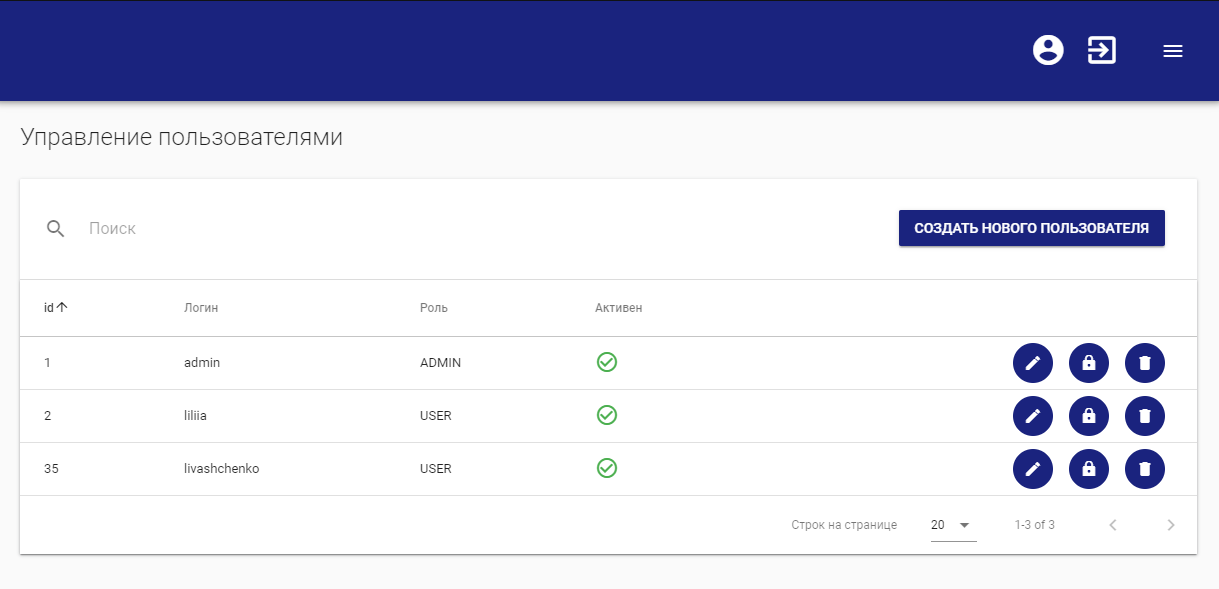


Рисунок 5.7. Страница с отображением всех пользователей в системе

**Добавление нового пользователя**

Пользователь с ролью администратора может сам вручную добавить нового пользователя (рис. 5.8, рис. 5.9).

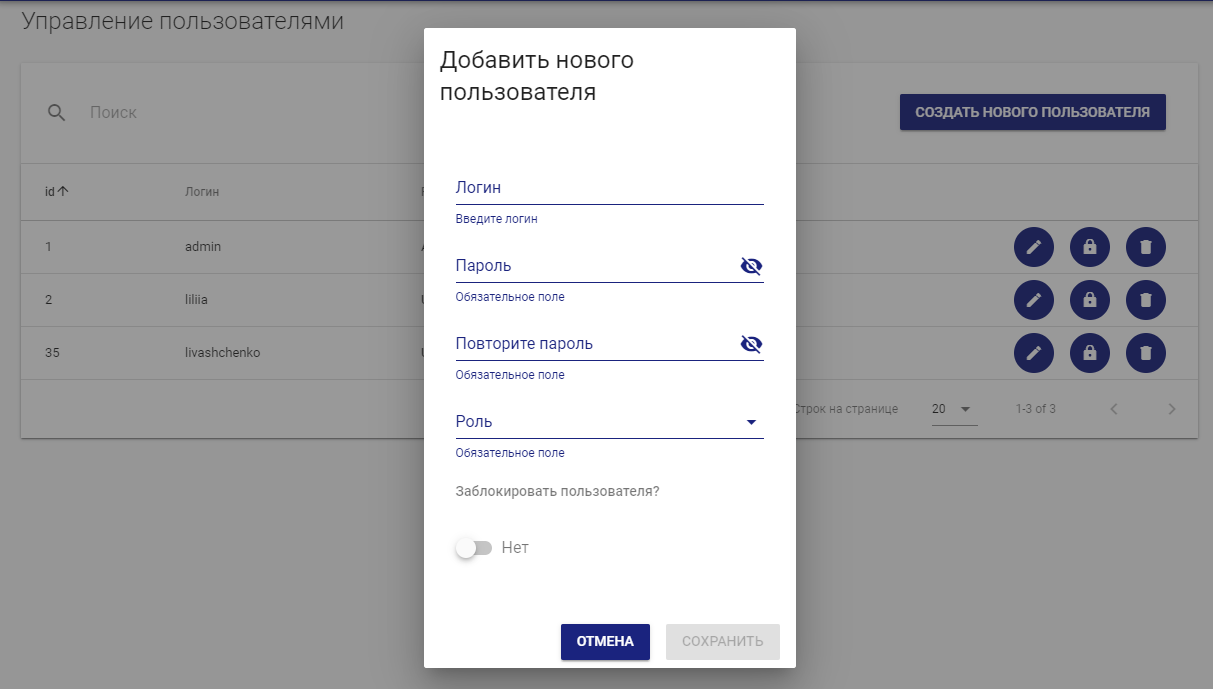


Рисунок 5.8. Добавление нового пользователя администратором

**Редактирование пользователей**

Пользователь с ролью администратора может редактировать данные других пользователей. Он может: изменить логин, изменить роль пользователя, заблокировать и удалить(рис. 5.9).

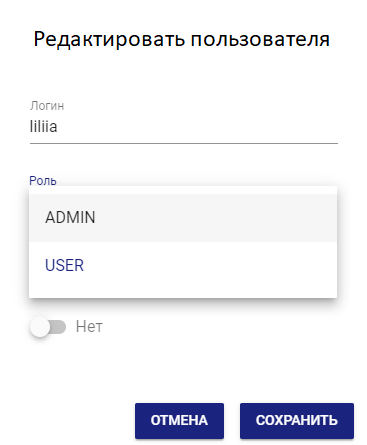
****

Рисунок 5.9. Редактирование данных пользователя

**Добавление новых проектов**

Добавлять новые проекты может только пользователь с ролью администратора. Для этого нужно:

Для автоматического добавленияданных, нужно перейти на страницу с каталогом и нажать на кнопку «Загрузить файл XML» (рис. 5.11).

Для добавления статью вручную, нажать на кнопку «Добавить статью» (рис. 5.12).

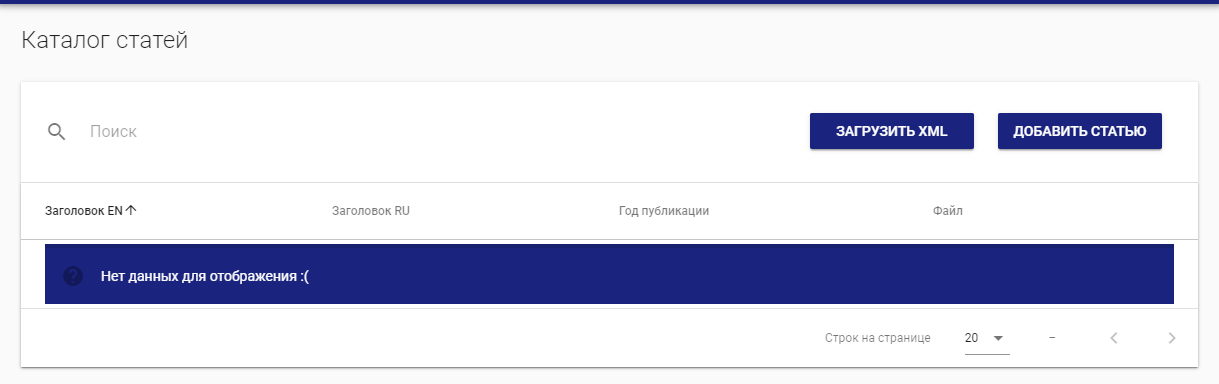


Рисунок 5.10. Отображение каталога статей до загрузки метаданных

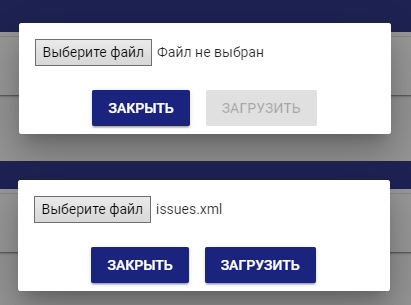


Рисунок 5.11. Загрузка файла

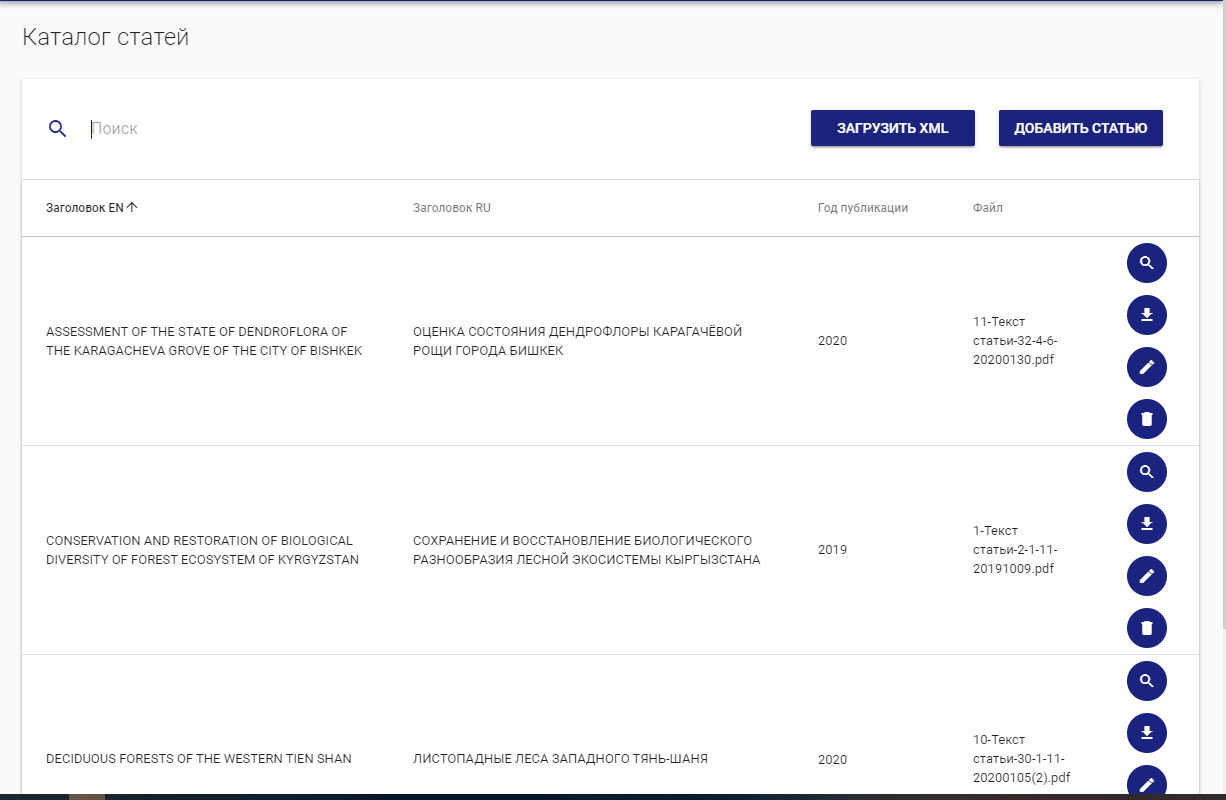


Рисунок 5.12. Отображение данных после загрузки из файла

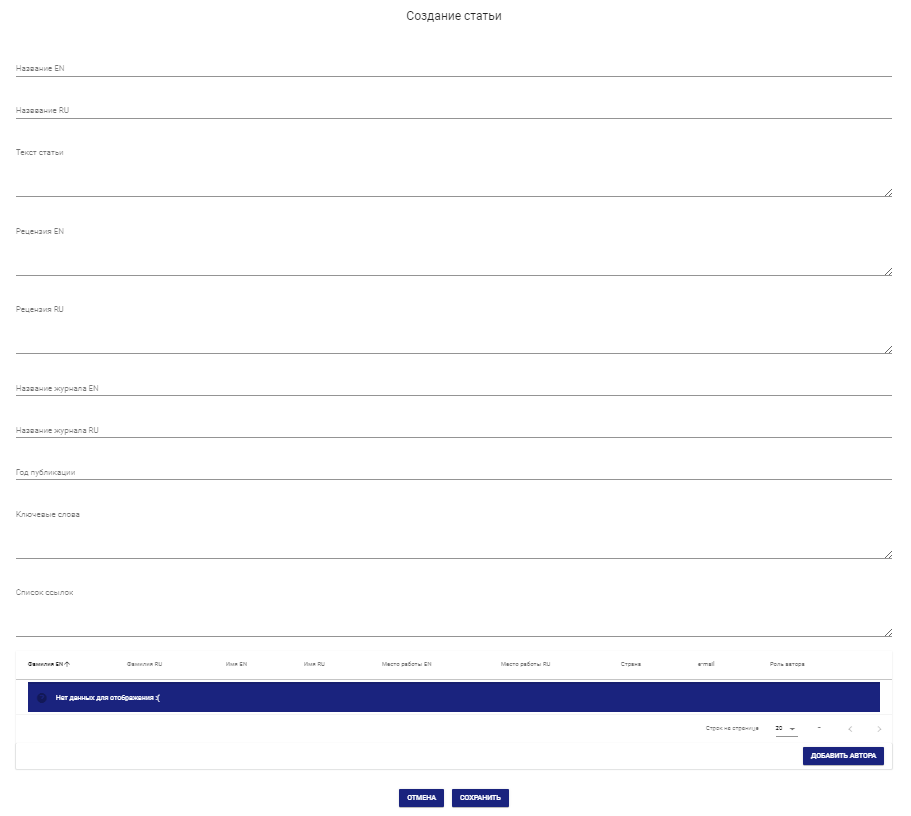


Рисунок 5.13. Разметка метаданных вручную

**Редактирование проектов**

Пользователь с ролью администратора также может редактировать проект (рис. 5.14). Вставка текста и списка ссылок происходит вручную.

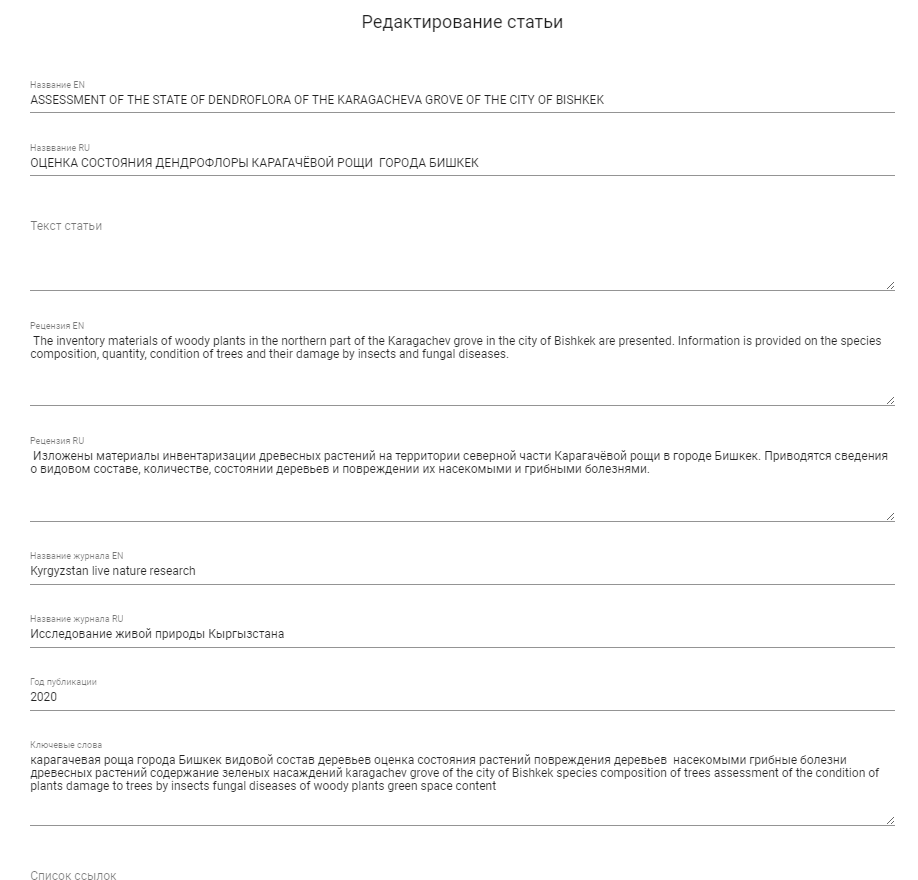


Рисунок 5.14. Редактирование статьи

# Заключение

В результате разработки автоматизированной системы для конвертации метаданных из системы Open Journal Systems в систему elibrary.ru были выполнены следующие задачи:

1. Проведен анализ предметной области;
2. Разработаны интерфейсы;
3. Разработана модель системы;
4. Реализован алгоритм конвертации метаданных из файла в систему.

Разработанная система выполняет следующие функции:

* Регистрация;
* Авторизация;
* Управление пользователями;
* Просмотр каталога проектов;
* Добавление проекта;
* Редактирование данных проекта;
* Редактированных данных пользователей;
* Удаление проекта;
* Удаление пользователя:
* Добавление проекта вручную;
* Просмотр данных статьи;
* Считывание метаданных с файла и отображение их в системе.

Дальнейшее развитие системы включает в себя:

* Внедрение системы в систему Open Journal Systems:
* Реализация передачи подготовленных метаданных на проверку литературному автору (научному работнику);
* Реализация публикации в системе elibrary.ru.

# Список литературы

1. http://ranlib.ru/?page\_id=14775 (дата обращения: 16.05.2020).
2. http://195.38.182.113 (дата обращения: 16.05.2020).
3. http://www.isras.ru/index.php?page\_id=1552 (дата обращения: 16.05.2020).
4. http://elibrary.ru (дата обращения: 16.05.2020).
5. http://cyberleninka.ru (дата обращения: 16.05.2020).
6. И. А. Мбого, Д. Е. Прокудин, А. В. Чугунов Комплексная интеграция цифровых коллекций в информационное пространство научных исследований // Технологии информационного общества в науке, образовании и культуре: Труды XVII Всероссийской объединенной конференции «Интернет и современное общество» (IMS-2014). – СПб: Университет ИТМО, 2014. – C. 48–53.
7. https://vniigis.ru/1\_dlya\_failov/Help/Инструкция по работе с программой Articulus eLibrary НЭБ РИНЦ.pdf#page=2&zoom=100,110,106 (дата обращения: 07.05.2020)
8. <http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl5/gl5.html> (дата обращения: 07.05.2020) https://www.intuit.ru/studies/courses/32/32/lecture/1008?page=4#:~:text=Класс -сущность (entity class),не уничтожаться с выключением системы.&text=Граничный класс может быть изображен,5.3, в) (дата обращения: 07.05.2020) – определение класса-сущности;
9. http://khpi-iip.mipk.kharkiv.edu/library/case/leon/gl10/gl10.html (дата обращения: 07.05.2020)
10. Одиночкина, С. В. Web-программирование PHP / С. В. Одиночкина. — Санкт-Петербург : Университет ИТМО, 2012. — 79 c. — ISBN 2227-8397. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/65750.html (дата обращения: 19.05.2020). — Режим доступа: для авторизир. пользователей
11. Кисленко, Н. П. Интернет-программирование на PHP : учебное пособие / Н. П. Кисленко. — Новосибирск : Новосибирский государственный архитектурно-строительный университет (Сибстрин), ЭБС АСВ, 2015. — 177 c. — ISBN 978-5-7795-0745-5. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/68769.html (дата обращения: 19.06.2020). — Режим доступа: для авторизир. пользователей
12. Использование языка структурированных запросов SQL : методические указания к расчетной работе / составители А. Я. Лахов, К. А. Сафонов. — Нижний Новгород : Нижегородский государственный архитектурно-строительный университет, ЭБС АСВ, 2010. — 38 c. — ISBN 2227-8397. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/15999.html (дата обращения: 19.06.2020). — Режим доступа: для авторизир. пользователей
13. https://ru.wikipedia.org/wiki/Docker(дата обращения: 01.05.2020)
14. https://ru.wikipedia.org/wiki/Spring\_Framework (дата обращения: 01.05.2020)
15. Гутман, Г. Н. Объектно-реляционная СУБД PostgreSQL : учебное пособие / Г. Н. Гутман. — Самара : Самарский государственный технический университет, ЭБС АСВ, 2016. — 125 c. — ISBN 2227-8397. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/90660.html (дата обращения: 19.06.2020). — Режим доступа: для авторизир. пользователей
16. https://ru.wikipedia.org/wiki/Модульное\_тестирование (дата обращения: 07.05.2020)
17. Леоненков, А. В. Объектно-ориентированный анализ и проектирование с использованием UML и IBM Rational Rose. Курс лекций : учебное пособие для студентов вузов, обучающихся по специальностям в области информационных технологий / А. В. Леоненков. — Москва, Саратов : Интернет-Университет Информационных Технологий (ИНТУИТ), Вузовское образование, 2017. — 318 c. — ISBN 978-5-4487-0081-1. — Текст : электронный // Электронно-библиотечная система IPR BOOKS : [сайт]. — URL: http://www.iprbookshop.ru/67388.html (дата обращения: 19.06.2020). — Режим доступа: для авторизир. пользователей

# Приложение 1. Глоссарий

### 1. Введение

### 1.1. Цель

Глоссарий содержит описание терминов, используемых при проектировании автоматизированной системы конвертации метаданных на основе платформы Open Journal Systems. Определяются основные понятия, непосредственно связанные с конвертацией метаданных.

### 1.2. Контекст

Глоссарий создан в рамках разработки системы автоматизированной системы конвертации метаданных на основе платформы Open Journal Systems.

### 2. Определения

### 2.1. Понятия, используемые при описании исходной информации

**Научная статья**

Научная статья – законченное авторское произведение, описывающее результаты оригинального научного исследования (первичная научная статья) или посвящённая рассмотрению ранее опубликованных научных статей, связанных общей темой (обзорная научная статья).

**Конвертация**

**Конвертация данных -** является одним из компонентов технологии обмена данными через формат **EnterpriseData**. Главное назначение **конвертации данных -**это создание программного кода модуля менеджера обмена, состоящего из процедур и функций, в которых реализована логика загрузки данных, представленных в формате **EnterpriseData**, а также логика выгрузки данных в формат.

**Редакция**

Редакция - один из вариантов/версий правки/формулировки [текста](https://ru.wikipedia.org/wiki/%D0%A2%D0%B5%D0%BA%D1%81%D1%82). Это окончательная обработка посторонним лицом ([редактором](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D0%B4%D0%B0%D0%BA%D1%82%D0%BE%D1%80)) текста произведения или собрания произведений какого-либо автора (в данном случае, автора научных статей).

**Научный работник**

Научный работник - пользователь системы, наделенный правами на чтение публикаций. Он предоставляет свои личные данные, а также собственно-разработанные научные технические работы в целях дальнейшей публикации, предоставляет их рецензенту для проверки, затем передает статьи литературному редактору.

**Литературный редактор**

Литературный редактор - занимается формированием метаданных для выпуска статьи в онлайн библиотеку или журнал.

**Системный администратор**

Системный администратор - администратор системы, наделенный правами на обработку, изменение плановой информации в системе. Он обрабатывает предоставленные статьи, добавляет в систему, следит за базой данных пользователей и статьями.

**Open Journal Systems**

Open Journal Systems - [открытое программное обеспечение](https://ru.wikipedia.org/wiki/%D0%9E%D1%82%D0%BA%D1%80%D1%8B%D1%82%D0%BE%D0%B5_%D0%BF%D1%80%D0%BE%D0%B3%D1%80%D0%B0%D0%BC%D0%BC%D0%BD%D0%BE%D0%B5_%D0%BE%D0%B1%D0%B5%D1%81%D0%BF%D0%B5%D1%87%D0%B5%D0%BD%D0%B8%D0%B5) для организации [рецензируемых](https://ru.wikipedia.org/wiki/%D0%A0%D0%B5%D1%86%D0%B5%D0%BD%D0%B7%D0%B8%D1%8F) [научных изданий](https://ru.wikipedia.org/wiki/%D0%9D%D0%B0%D1%83%D1%87%D0%BD%D1%8B%D0%B9_%D0%B6%D1%83%D1%80%D0%BD%D0%B0%D0%BB).

**Метаданные**

Метаданные раскрывают сведения о признаках и свойствах, характеризующих какие-либо сущности, позволяющие автоматически искать и управлять ими в больших информационных потоках. Метаданные в рамках настоящего проекта – это данные, характеризующие статью: заголовок, аннотация, ключевые слова, текст статьи.

**Заголовок**

Заголовок - обозначение структурной части основно­го текста произведения (раздела, главы, параграфа, таблицы и др.) или издания. Основное требование к названию статьи — краткость и ясность. Максимальная длина заголовка — 10—12 слов. Название долж­но быть содержательным, выразительным, отражать содержание статьи.

**Аннотация**

Аннотация - краткая характеристика научной статьи с точки зрения ее назначения, содержания, вида, формы и других особенностей.

Это независимый от статьи источник информации. Рецензию пишут после завершения работы над основным тек­стом статьи. Она включает характеристику основной темы, проблемы, объекта, цели работы и ее результаты. В ней указывают, что нового несет в себе данный документ в сравнении с другими, родст­венными по тематике и целевому назначению. Рекомендуемый объ­ем — 100 – 250 слов на русском и английском языках.

**Ключевые слова**

Ключевые слова выражают основное смысловое содержание статьи, служат ориентиром для читателя и используются для поиска ста­тей в электронных базах. Размещаются после аннотации в количестве 4—8 слов, приводятся на русском и английском языках. Должны от­ражать дисциплину (область науки, в рамках которой написана ста­тья), тему, цель, объект исследования

**Текст статьи**

Содержимое статьи, которое содержит в себе: введение, обзор литературы, основная часть (методология, результаты), выводы и дальнейшие перспективы исследования, список литературы, в соответствии с требованиями конкретной редакции.

# Приложение 2. Листинг

package kg.diplom.server.controllers;

import kg.diplom.server.entities.Article;

import kg.diplom.server.enums.AuthorType;

import kg.diplom.server.enums.PermEnum;

import kg.diplom.server.exceptions.BadRequestException;

import kg.diplom.server.models.ArticleModel;

import kg.diplom.server.models.EnumModel;

import kg.diplom.server.repo.ArticleRepo;

import kg.diplom.server.services.ArticleFileService;

import kg.diplom.server.services.ArticleService;

import kg.diplom.server.services.grant.GrantService;

import lombok.AccessLevel;

import lombok.experimental.FieldDefaults;

import lombok.extern.slf4j.Slf4j;

import org.springframework.http.ResponseEntity;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.web.bind.annotation.\*;

import org.springframework.web.multipart.MultipartFile;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;

import javax.websocket.server.PathParam;

import javax.xml.bind.JAXBException;

import java.io.IOException;

import java.util.List;

import java.util.stream.Collectors;

import java.util.stream.Stream;

@Slf4j

@RestController

@RequestMapping("/api/article")

@FieldDefaults(level = AccessLevel.PRIVATE, makeFinal = true)

public class ArticleController extends BaseCatalogController<Article, ArticleModel, ArticleService, ArticleRepo> {

GrantService grantService;

ArticleFileService articleFileService;

public ArticleController(

ArticleService service,

GrantService grant,

GrantService grantService,

ArticleFileService articleFileService) {

super(service, grant);

this.grantService = grantService;

this.articleFileService = articleFileService;

}

@GetMapping("/canEdit")

@PreAuthorize("isAuthenticated()")

public boolean canEdit() {

return grantService.hasAny(PermEnum.ADMIN);

}

@GetMapping("/authorTypes")

public ResponseEntity<List<EnumModel>> authorTypes() {

return ResponseEntity.ok(Stream.of(AuthorType.values()).map(EnumModel::fromAuthorType).collect(Collectors.toList()));

}

@PostMapping("/uploadXml")

public void submit(@RequestParam("file") MultipartFile file) throws IOException {

try {

service.uploadFile(file);

} catch (JAXBException e) {

log.error("", e);

throw new BadRequestException("Не верный формат xml (" + e.getMessage() + ")");

}

}

@GetMapping("/downloadArticleFile/{id}")

public void download(@PathVariable("id") Long id, HttpServletResponse httpServletResponse) {

articleFileService.download(id, httpServletResponse);

}

}

package kg.diplom.server.controllers;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiImplicitParam;

import io.swagger.annotations.ApiOperation;

import kg.diplom.server.models.User;

import kg.diplom.server.models.UsersModel;

import kg.diplom.server.services.auth.AuthService;

import lombok.AllArgsConstructor;

import lombok.NonNull;

import lombok.experimental.FieldDefaults;

import lombok.extern.slf4j.Slf4j;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestBody;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import java.io.IOException;

import static lombok.AccessLevel.PACKAGE;

import static lombok.AccessLevel.PRIVATE;

@Slf4j

@RestController

@RequestMapping("/auth")

@FieldDefaults(level = PRIVATE, makeFinal = true)

@AllArgsConstructor(access = PACKAGE)

@Api(description = "Контроллер авторизации и регистрации")

public class AuthControllerRest {

@NonNull AuthService service;

/\*\*

\* @param user модель с логином и паролем

\* @return токен

\*/

@PostMapping(value = "/login")

@ApiOperation("Возвращает токен")

@ApiImplicitParam(name = "Authorization", required = false)

public String login(@RequestBody final User user) throws IOException {

log.debug("controller: login with username {} and password {}", user.getUsername(), user.getPassword());

return service.login(user.getUsername(), user.getPassword());

}

/\*\*

\* @param user модель для регистрации пользователя

\*/

@PostMapping("/register")

@ApiOperation("Производит регистрацию пользователя")

@ApiImplicitParam(name = "Authorization", required = false)

public void register(@RequestBody UsersModel user) {

service.registerUser(user);

}

}

package kg.diplom.server.entities;

import kg.diplom.server.entities.base.BaseEntity;

import kg.diplom.server.models.ArticleModel;

import lombok.\*;

import lombok.experimental.FieldDefaults;

import javax.persistence.Column;

import javax.persistence.Entity;

import javax.persistence.OneToOne;

import javax.persistence.Table;

import java.sql.Timestamp;

import java.text.SimpleDateFormat;

@Getter

@Setter

@Entity

@Table

@FieldDefaults(level = AccessLevel.PRIVATE)

public class Article extends BaseEntity<ArticleModel> {

private static final SimpleDateFormat format = new SimpleDateFormat("yyyy-MM-dd");

public Article() {

}

@Builder

public Article(

Long id,

String titleEn,

String titleRu,

String text,

String abstractEn,

String abstractRu,

String copyrightHolderEn,

String copyrightHolderRu,

Integer publicationYear,

Timestamp removeDate,

ArticleFile articleFile,

String keyWords,

String links

) {

super(id);

this.titleEn = titleEn;

this.titleRu = titleRu;

this.text = text;

this.abstractEn = abstractEn;

this.abstractRu = abstractRu;

this.copyrightHolderEn = copyrightHolderEn;

this.copyrightHolderRu = copyrightHolderRu;

this.publicationYear = publicationYear;

this.removeDate = removeDate;

this.articleFile = articleFile;

this.keyWords = keyWords;

this.links = links;

}

@Column(length = 1024)

String titleEn;

@Column(length = 1024)

String titleRu;

@Column(length = 65536)

String text;

@Column(length = 1024)

String abstractEn;

@Column(length = 1024)

String abstractRu;

@Column(length = 1024)

String copyrightHolderEn;

@Column(length = 1024)

String copyrightHolderRu;

Integer publicationYear;

Timestamp removeDate;

@OneToOne

ArticleFile articleFile;

@Column(length = 2048)

String keyWords;

@Column(length = 4096)

String links;

@Override

public ArticleModel toModel() {

return ArticleModel.builder()

.id(getId())

.titleEn(titleEn)

.titleRu(titleRu)

.text(text)

.abstractEn(abstractEn)

.abstractRu(abstractRu)

.copyrightHolderEn(copyrightHolderEn)

.copyrightHolderRu(copyrightHolderRu)

.publicationYear(publicationYear)

.articleFile(articleFile == null ? null : articleFile.toModel())

.keyWords(keyWords)

.links(links)

.build();

}

}

package kg.diplom.server.entities;

import kg.diplom.server.entities.base.BaseEntity;

import kg.diplom.server.models.UsersModel;

import lombok.Builder;

import lombok.Getter;

import lombok.Setter;

import lombok.ToString;

import javax.persistence.\*;

import javax.validation.constraints.NotNull;

import java.sql.Timestamp;

import java.time.LocalDateTime;

import java.util.Date;

@Getter

@Setter

@ToString

@Entity

@Table

public class Users extends BaseEntity<UsersModel> {

@NotNull

private String username;

@NotNull

@ToString.Exclude

private String password;

@Builder.Default

private Boolean blocked = false;

private Timestamp deleted;

public void setDeletedToNow() {

deleted = Timestamp.valueOf(LocalDateTime.now());

}

@ManyToOne(fetch = FetchType.LAZY)

@JoinColumn(name = "role")

private Roles role;

@ToString.Exclude

@Builder.Default

private Timestamp lastActivity = new Timestamp(Systems.currentTimeMillis());

@ToString.Exclude

private Date expireDate;

@Builder

public Users(Long id, @NotNull String username, @NotNull String password, Boolean blocked, Roles role, Timestamp lastActivity, Date expireDate, Timestamp deleted) {

super(id);

this.username = username;

this.password = password;

this.blocked = blocked;

this.role = role;

this.lastActivity = lastActivity;

this.expireDate = expireDate;

this.deleted = deleted;

}

public Users() {}

@Override

public UsersModel toModel() {

return UsersModel.builder()

.id(getId())

.username(username)

.blocked(blocked)

// .role(role.toModel())

.build();

}

}

package kg.diplom.server.controllers;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiOperation;

import io.swagger.annotations.ApiParam;

import kg.diplom.server.entities.Permission;

import kg.diplom.server.entities.Roles;

import kg.diplom.server.enums.PermEnum;

import kg.diplom.server.models.RolesModel;

import kg.diplom.server.services.RolesService;

import kg.diplom.server.services.UsersService;

import kg.diplom.server.services.grant.GrantService;

import kg.diplom.server.util.ListUtil;

import lombok.Getter;

import lombok.Setter;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.http.HttpStatus;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.web.bind.annotation.\*;

import java.util.ArrayList;

import java.util.List;

@Getter

@Setter

@RestController

@RequestMapping("/api/roles")

@Api(description = "Контроллер ролей")

public class RolesControllerRest {

@Autowired

private RolesService rolesService;

@Autowired

GrantService grantService;

@Autowired

UsersService usersService;

@GetMapping

@PreAuthorize("isAuthenticated()")

@ApiOperation("Возвращает список со всеми ролями")

public List<RolesModel> list() {

if (!grantService.hasAny(PermEnum.ADMIN))

return new ArrayList<>();

return ListUtil.map(rolesService.findAllByDeletedIsNull(), Roles::toModel);

}

@PostMapping

@PreAuthorize("isAuthenticated()")

@ResponseStatus(HttpStatus.CREATED)

@ApiOperation("Роут для создания роли, возвращает идентификатор роли в базе")

public Long create(

@ApiParam(value = "модель для создания роли", required = true)

@RequestBody RolesModel rolesModel) {

if (!grantService.hasAny(PermEnum.ADMIN))

return null;

Roles c = Roles.builder()

.id(rolesModel.getId())

.name(rolesModel.getName())

.build();

getPermissionList(rolesModel, c);

return rolesService.save(c);

}

@PutMapping("{id}")

@PreAuthorize("isAuthenticated()")

@ApiOperation("Роут для редактирования роли")

public void update(

@ApiParam(value = "идентификатор роли в базе", required = true, example = "1")

@PathVariable Long id,

@ApiParam(value = "модель для редактирования роли", required = true)

@RequestBody RolesModel rolesModel) {

if (!grantService.hasAny(PermEnum.ADMIN))

return;

Roles c = Roles.builder()

.id(id)

.name(rolesModel.getName())

.build();

getPermissionList(rolesModel, c);

rolesService.save(c);

}

private void getPermissionList(RolesModel rolesModel, Roles c) {

List<Permission> permissionList = ListUtil.map(rolesModel.getPermission(),

permission -> Permission.builder()

.id(permission.getId())

.name(permission.getName()).build());

c.setPermissions(permissionList);

}

}

package kg.diplom.server.controllers;

import kg.diplom.server.enums.PermEnum;

import kg.diplom.server.models.MenuModel;

import kg.diplom.server.models.MenuSubModel;

import kg.diplom.server.services.grant.GrantService;

import lombok.Getter;

import lombok.Setter;

import org.springframework.beans.factory.annotation.Autowired;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.web.bind.annotation.PostMapping;

import org.springframework.web.bind.annotation.RequestMapping;

import org.springframework.web.bind.annotation.RestController;

import java.util.ArrayList;

import java.util.List;

@Getter

@Setter

@RestController

@RequestMapping("/api/mainMenu")

public class MenuController {

@Autowired

GrantService grantService;

@PostMapping("menu")

@PreAuthorize("isAuthenticated()")

public List<MenuModel> menu() {

List<MenuModel> list = new ArrayList<>();

if (grantService.hasAny(PermEnum.ADMIN)) {

list.add(MenuModel.builder()

.title("Пользователи")

.href("/users")

.group("")

.icon("group")

.name("Users").build());

}

return list;

}

private List<MenuSubModel> getCatItems() {

List<MenuSubModel> subItems = new ArrayList<>();

return subItems;

}

}

package kg.diplom.server.controllers;

import com.lambdaworks.crypto.SCryptUtil;

import io.swagger.annotations.Api;

import io.swagger.annotations.ApiImplicitParam;

import io.swagger.annotations.ApiOperation;

import io.swagger.annotations.ApiParam;

import kg.diplom.server.entities.Permission;

import kg.diplom.server.entities.Users;

import kg.diplom.server.enums.PermEnum;

import kg.diplom.server.models.User;

import kg.diplom.server.models.UsersModel;

import kg.diplom.server.services.UsersService;

import kg.diplom.server.services.auth.AuthService;

import kg.diplom.server.services.grant.GrantService;

import kg.diplom.server.util.ListUtil;

import lombok.AllArgsConstructor;

import lombok.NonNull;

import lombok.experimental.FieldDefaults;

import lombok.extern.slf4j.Slf4j;

import org.springframework.security.access.prepost.PreAuthorize;

import org.springframework.web.bind.annotation.\*;

import javax.transaction.Transactional;

import java.sql.Timestamp;

import java.time.LocalDateTime;

import java.util.List;

import java.util.Optional;

import static lombok.AccessLevel.PACKAGE;

import static lombok.AccessLevel.PRIVATE;

@Slf4j

@RestController

@RequestMapping("/api/user")

@FieldDefaults(level = PRIVATE, makeFinal = true)

@AllArgsConstructor(access = PACKAGE)

@Api(description = "Контроллер для управления пользователями")

public class UserControllerRest {

@NonNull UsersService service;

@NonNull AuthService auth;

@NonNull GrantService grant;

@GetMapping

@PreAuthorize("isAuthenticated()")

@ApiOperation("Возвращает список со всеми пользователями")

public List<UsersModel> list() {

grant.checkHasAny(PermEnum.ADMIN);

return service.findAllAsModel();

}

@PostMapping

@PreAuthorize("isAuthenticated()")

@ApiOperation("Роут для создания пользователя (только для админов)")

public void register(

@ApiParam(value = "Модель для создания пользователя", required = true)

@RequestBody final User user) {

grant.checkHasAny(PermEnum.ADMIN);

log.debug("controller: registerAsAdmin with user {}", user);

auth.registerAsAdmin(user);

}

@PostMapping("update/{id}")

@PreAuthorize("isAuthenticated()")

@ApiOperation("Роут для редактирования пользователя")

public void update(

@ApiParam(value = "Идентификатор пользователя в базе", required = true, example = "1")

@PathVariable Long id,

@ApiParam(value = "Mодель для редактирования пользователя", required = true)

@RequestBody UsersModel usersModel) {

grant.checkHasAny(PermEnum.ADMIN);

service.update(id, usersModel);

}

@PostMapping("updatePassword")

@PreAuthorize("isAuthenticated()")

@ApiOperation("Роут для смены пароля текущего пользователя")

public void updateSelfPassword(

@ApiParam(value = "Модель должна содержать поле \"password\"", required = true)

@RequestBody UsersModel usersModel) {

Users users = service.findCurrent();

users.setPassword(SCryptUtil.scrypt(usersModel.getPassword(), 16, 16, 16));

service.save(users);

}

@PostMapping("updatePassword/{id}")

@PreAuthorize("isAuthenticated()")

@ApiOperation("Роут для смены пароля пользователя по идентификатору (только для админов)")

public void updatePassword(

@ApiParam(value = "Идентификатор пользователя в базе", required = true, example = "1")

@PathVariable Long id,

@ApiParam(value = "Модель должна содержать поле \"password\"", required = true)

@RequestBody UsersModel usersModel) {

grant.checkHasAny(PermEnum.ADMIN);

Optional<Users> optionalUsers = service.findById(id);

if (optionalUsers.isPresent()) {

Users users = optionalUsers.get();

users.setPassword(SCryptUtil.scrypt(usersModel.getPassword(), 16, 16, 16));

service.save(users);

}

}

@PostMapping("/delete/{id}")

@PreAuthorize("isAuthenticated()")

@ApiOperation("Роут для удаления пользователя")

public void delete(

@ApiParam(value = "Идентификатор пользователя в базе", required = true, example = "1")

@PathVariable Long id) {

grant.checkHasAny(PermEnum.ADMIN);

Optional<Users> optional = service.findById(id);

Users updateUsers = service.findCurrent();

if (optional.isPresent()) {

Users c = optional.get();

updateUsers.setDeleted(Timestamp.valueOf(LocalDateTime.now()));

service.save(updateUsers);

}

}

@GetMapping("/current")

@PreAuthorize("isAuthenticated()")

@Transactional // небольшое зло во славу САТАНЕ

@ApiOperation("Возвращает текущего пользователя")

public UsersModel getCurrentUser() {

Users users = service.findCurrent();

return users.toModel();

}

@PostMapping("/hasAny")

@PreAuthorize("isAuthenticated()")

@ApiOperation("Роут для проверки разрешений пользователя")

public Boolean hasAny(

@ApiParam(value = "список с резрешениями", required = true)

@RequestBody List<PermEnum> roles) {

return grant.hasAny(roles);

}

@GetMapping("/permissions")

@Transactional

@PreAuthorize("isAuthenticated()")

@ApiOperation("Возвращает все резрешения текущего пользователя")

public List<PermEnum> permissions() {

return ListUtil.map(service.findCurrent().getRole().getPermissions(), Permission::getName);

}

@GetMapping("/exist")

@ApiOperation("Роут для проверки существует ли пользователь с таким именем. Если пользователь с таким именем существует, то вернёт имя пользователя, если нет - пустую строку")

@ApiImplicitParam(name = "Authorization", required = false)

public String checkExist(

@ApiParam(value = "Имя пользователя", required = true)

@RequestParam("user") String username) {

Users u = service.findByUsernameByDeletedDate(username);

if (u != null) {

return u.getUsername();

}

return "";

}

@GetMapping("/full")

@PreAuthorize("isAuthenticated()")

@ApiOperation("Роут возвращает всю информацию о пользователе")

public UsersModel full(

@ApiParam("Имя пользователя (если указано и пользователь имеет пермишн ADMIN, то операция применится к указанному пользователю, иначе параметр игнорируется и берётся имя текущёго пользователя)")

@RequestParam(required = false) String userName) {

if (userName == null || !grant.hasAny(PermEnum.ADMIN)) {

userName = service.findCurrent().getUsername();

}

return service.collectFullUserInfo(userName);

}

}