

Направление подготовки «09.03.04 – Программная инженерия»

Направление профиля «Программная инженерия»

**ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА  
(БАКАЛАВРСКАЯ РАБОТА)**

На тему «Разработка веб-сервиса для автоматизации составления  
библиографического описания научно-методической литературы»

г. Ханты-Мансийск

2020 год

## АННОТАЦИЯ

на выпускную квалификационную бакалаврскую работу

Тема выпускной квалификационной работы «Разработка веб-сервиса для автоматизации составления библиографического описания научно-методической литературы».

Выпускная квалификационная работа состоит из: введения, аналитической части, проектной части, практической части, заключения, списка использованных источников и приложений.

Во введении обоснована актуальность темы, представлены цели и задачи, определены объект и предмет исследования.

В аналитической главе проанализирована предметная область, проведен обзор существующих сервисов для составления списка источников научно-методической литературы. В проектной главе построена модель процесса по составлению списка источников, сформировано техническое задание, осуществлено проектирование веб-сервиса на основе объектно-ориентированного подхода и проектирование базы данных. В практической главе представлены результаты разработки веб-сервиса по составлению списка источников научно-методической литературы.

В заключении представлены выводы, полученные в ходе выполнения выпускной квалификационной работы.

Дипломная работа включает в себя 6 приложений из 23 страниц, 25 рисунков, 13 таблиц, 25 литературных источников. Общий объем дипломной работы составляет 75 страниц.

**Министерство науки и высшего образования Российской Федерации  
Федеральное государственное бюджетное образовательное учреждение  
высшего образования  
«ЮГОРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ»  
Институт цифровой экономики  
Направление подготовки «09.03.04 – Программная инженерия»  
Направление профиля «Программная инженерия»**

**УТВЕРЖДАЮ**

Руководитель образовательной программы

ст. преподаватель Русанов Михаил

Александрович

(ученая степень, звание, фамилия имя отчество)



« 23 » марта 2020 г.

**ЗАДАНИЕ**

**на выполнение выпускной квалификационной работы**

Студентка: группа

Срок сдачи студентом законченной выпускной квалификационной работы:

1. Тема работы: Разработка веб-сервиса для автоматизации составления библиографического описания научно-методической литературы

2. Исходные данные к выпускной квалификационной работе (организация, отрасль знаний, назначение проектируемой системы/модуля, основные функциональные требования):

Составление списка литературы является обязательным элементом любой научно-исследовательской работы: реферата, статьи, курсовой, дипломной работы и пр. Список включает литературу, имеющую непосредственное отношение к исследуемой теме. Большое значение имеет правильное оформление библиографического описания. В Югорском государственном университете список составляется в соответствии с "Методическими рекомендациями к составлению списков использованной литературы в соответствии с ГОСТ Р 7.0.100-2018 "Библиографическая запись. Библиографическое описание", разработанное Научной библиотекой ЮГУ. При составлении библиографической записи используются обязательные элементы библиографического описания (автор, основное заглавие, первые сведения об

ответственности, первое место публикации и т.д.) и некоторые условно-обязательные элементы (сведения, относящиеся к заглавию; вид содержания; средство доступа). Оформление библиографической записи источника зависит от его типа (книжное издание, многотомное издание, статья из журнала, законодательные материалы, сайты в сети Интернет и т.д.)

Назначение проектируемого и разрабатываемого веб-сервиса состоит в создании правильного библиографического описания без изучения дополнительной информации. Сведения об использованном источнике, введенные пользователем, собираются, обрабатываются и сохраняются в базе данных. На основе этой информации выводится корректная и пронумерованная библиографическая запись.

Регулярная актуализация базы данных источников и ее пополнение обеспечивает организацию и реализацию поиска по сведениям источника для включения его пользователем в формируемый текущий список использованной литературы.

### 3. Планируемые результаты выпускной квалификационной работы:

Объектами автоматизации являются процессы составления библиографической записи и библиографического описания научно-методической литературы.

Проектирование и разработка программного обеспечения в данной области необходимо для решения следующих задач:

- формирование определенного библиографического описания источников научно-методических изданий в соответствии с требованиями методических рекомендаций Научной библиотеки ЮГУ и ГОСТ Р 7.0.100-2018 "Библиографическая запись. Библиографическое описание";
- предоставление возможности пользователям поиска источника по его различным данным (по типу, по автору, по названию) для включения его в формируемый список;
- осуществлять ввод и обработку информации об источниках;
- осуществлять пополнение и актуализацию источников в базе данных;
- сохранение составляемого списка источников во внешний файл;
- транслитерация источника.

Разрабатываемый веб-сервис должен обеспечить получение следующей информации:

1) Перечень типов источников, используемых при формировании списков источников научно-методических разработок пользователя/автора.

2) Отображение данных библиографической записи, включающих заголовки публикации; её основное заглавие; первые сведения об ответственности; сведения об издании (если есть); дополнительные сведения об

издании (если есть); сведения о нумерации (для сериальных ресурсов); первое место публикации, производства и/или распространения; имя издателя, производителя и/или распространителя; дата публикации, производства и/или распространения; специфическое обозначение материала и объем; основное заглавие серии/подсерии или многочастного монографического ресурса (если есть); международный стандартный номер (ISBN, DOI); номер выпуска серии/подсерии или многочастного монографического ресурса; сведения, относящиеся к заглавию; вид содержания: способ и средства доступа.

3) Отображение составляемого списка источников и обеспечение типовых действий с ним (редактирование, удаление, сохранение).

4. Содержание пояснительной записки ВКР (перечень подлежащих разработке вопросов):

- Результаты анализа доступных аналогов программно-информационных систем и модулей, эксплуатируемых в предметной области тематики выпускной бакалаврской работы, и инструментальных средств, использованных при их разработке.

- Разработать техническое задание на проектируемый и разрабатываемый веб-сервис.

- Разработать комплекс моделей предметной среды и сопутствующих информационных моделей, необходимых для проектирования и разработки предполагаемого веб-сервиса.

- Разработать необходимый комплект диаграмм, характеризующий проектируемый веб-сервис.

- Разработать пользовательский интерфейс веб-сервиса и основные формы для заполнения данных, необходимых для его функционирования.

- Разработать базу данных и веб-сервис, обосновав выбор соответствующего инструментария.

5. Сроки отчетности по выпускной квалификационной работе:

## Календарный план выполнения выпускной квалификационной работы

Наименование этапа работы над ВКР	Плановый срок выполнения этапа работы	Фактический срок выполнения этапа работы	Результат выполнения этапа работы над ВКР	Подпись руководи теля ВКР
Обследование предметной области. Изучение и анализ литературы и аналогов веб- сервиса. Формирование технических требований к веб-сервису.	12.05.2020 г. - 19.05.2020 г.	12.05.2020 г. - 19.05.2020 г.	Текст раздела ВКР. Результаты обследования предметной области, постановка задачи на проектирование веб- сервиса. Результаты изучения и анализа литературы и аналогов. Техническое задание на разработку веб-сервиса.	
Проектирование веб-сервиса.	20.05.2020 г. – 27.05.2020 г.	20.05.2020 г. – 27.05.2020 г.	Текст раздела ВКР: комплект диаграмм проектирования веб- сервиса.	
Разработка веб- сервиса.	23.05.2020 г. – 02.06.2020 г.	23.05.2020 г. – 02.06.2020 г.	Текст третьего раздела ВКР.	
Оформление пояснительной записки ВКР	03.06.2020 г. – 07.06.2020 г.	03.06.2020 г. – 07.06.2020 г.	Оформление текста пояснительной записки ВКР.	
Прохождение нормоконтроля	08.06.2020 г. – 20.06.2020 г.	8.06.2020 г. – 20.06.2020 г.	Предварительный вариант пояснительной записки ВКР. Веб-сервис.	
Сдача пояснительной записки ВКР на кафедру	22.06.2020 г.	22.06.2020 г.	Итоговый вариант ВКР.	
Защита ВКР	02.07.2020 г.	02.07.2020 г.	Итоговый вариант ВКР. Доклад, иллюстративные материалы к защите.	

Дата выдачи задания

## СОДЕРЖАНИЕ

ВВЕДЕНИЕ.....	9
---------------	---

1. Аналитическая часть .....	11
1.1. Анализ заполняемых и обрабатываемых данных .....	11
1.2. Обзор аналогов.....	13
1.3 Методологии проектирования ПО .....	21
1.4 Типовые подходы к разработке программного обеспечения.....	24
2. Проектная часть .....	26
2.1 Формирование технического задания на разработку веб-сервиса для автоматизации составления библиографического описания научно-методической литературы .....	26
2.2 Проектирование веб-сервиса для автоматизации составления библиографического описания научно-методической литературы.....	33
2.2.1 Концептуальное проектирование веб-сервиса для автоматизации составления библиографического описания научно-методической литературы.....	33
2.2.2 Логическое проектирование веб-сервиса для автоматизации составления библиографического описания научно-методической литературы.....	36
2.2.3 Проектирование физической структуры веб-сервиса для автоматизации составления библиографического описания научно-методической литературы.....	42
2.3 Проектирование базы данных веб-сервиса для автоматизации составления библиографического описания научно-методической литературы.....	45
3. Практическая часть.....	50
3.1 Выбор модели жизненного цикла программного обеспечения .....	50
3.2 Выбор технологии и средств разработки веб-сервиса.....	54
3.3 Разработка веб-сервиса для автоматизации составления библиографического описания научно-методической литературы.....	57
3.4 Описание интерфейса.....	65
ЗАКЛЮЧЕНИЕ .....	70

СОКРАЩЕНИЯ, ОБОЗНАЧЕНИЯ, ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ.....	72
СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ .....	73
ПРИЛОЖЕНИЯ.....	76
Приложение А .....	76
Приложение Б.....	90
Приложение В.....	93
Приложение Г .....	95
Приложение Д.....	97
Приложение Е.....	98



## ВВЕДЕНИЕ

Научная библиотека Югорского государственного университета (ЮГУ) во многом помогает студентам в процессе их обучения и профессорско-преподавательскому составу университета, включая его филиалы. А также занимается расширением возможностей использования библиотечно-информационных ресурсов, предназначенных для обеспечения учебного, воспитательного и научного процессов университета.

Оформление списка использованных источников обязательный этап при написании студенческих работ, таких как курсовые, дипломы, отчеты, а также научных статей. Список источников, указанный в конце работы, позволяет продемонстрировать умение студента или исследователя к самостоятельной проработке информации по исследуемой предметной области, глубину ознакомления с проблемой, которой посвящена студенческая, либо научная работа. Помимо демонстрации степени ознакомления с проблемой, составление списка источников позволяет не нарушать авторские права тех, чья интеллектуальная собственность использована студентом или исследователем в качестве основы для собственной работы.

Список, оформленный с нарушением правил, является причиной недопуска студенческой работы к защите/проверке, а статьи к рецензированию и публикации.

Согласно статистическим данным за 2016-2019 годы, научными исследованиями в России занимаются порядка семисот тысяч человек, в высших учебных заведениях и средне специальных учебных заведениях обучаются порядка пяти миллионов человек. Всем указанным людям необходимо в ходе своей учебной или научной деятельности оформлять списки источников к работам.

Процесс оформления источников вручную является затратным по времени. Размер списка литературы для курсовой работы/проекта составляет примерно двадцать пунктов, дипломной работы – от тридцати источников (в зависимости

от требований учебного заведения). При оформлении столь объемного перечня источников высокий риск допущения ошибки по причине человеческого фактора. Для устранения влияния человеческого фактора, а также для ускорения процесса формирования списка источников целесообразно использовать программный продукт, позволяющий автоматизировать указанный процесс.

Программный продукт должен запрашивать у пользователя сведения об источнике и затем формировать из них правильную библиографическую запись, построенную в соответствии с «Методическими рекомендациями к составлению списков использованной литературы», разработанное для Югорского государственного университета и в соответствии с ГОСТ Р 7.0.100-2018 «Библиографическая запись. Библиографическое описание».

Объектом выпускной квалификационной работы (ВКР) является процесс формирования списка источников. Предметом исследования – автоматизация процесса по формированию источников.

Целью выпускной квалификационной работы является проектирование и разработка веб-сервиса для автоматизированного составления библиографических описаний научно-методической литературы согласно требованиям Научной библиотеки Югорского государственного университета и ГОСТ.

Для достижения данной цели необходимо выполнить следующие задачи:

- 1) Провести анализ предметной области и доступных аналогов программного обеспечения, эксплуатируемых в предметной области тематики выпускной квалификационной работы. Осуществить выбор методологий проектирования и разработки предполагаемого программного обеспечения.
- 2) Разработать техническое задание на проектируемый программный продукт.
- 3) Осуществить проектирование разрабатываемого программного продукта.
- 4) Разработать программный продукт и его интерфейс.

## 1. Аналитическая часть

### 1.1. Анализ заполняемых и обрабатываемых данных

Оформление списка литературы регламентируется набором государственных стандартов, в которых прописано оформление источников различного типа. В методических требованиях Научной библиотеки указана необходимая структура библиографической записи (БЗ) в учебных работах ЮГУ: порядок следования фамилии и инициалов автора, названия источника, его объем и т.д. Также методические рекомендации содержат примеры оформления источников литературы различного типа.

Пример библиографической записи для монографии: Дейт, Кр. Дж. Введение в системы баз данных / Кр. Дж. Дейт. – М.: Вильямс, 2019. – 1328 с. – ISBN 978-5-907144-17-0. – Текст: непосредственный.

Характеристика составляющих частей библиографической записи приведена в таблице 1.1.

Таблица 1.1 – Характеристика БЗ

Часть записи	Пример
Заголовок	Дейт, Кр. Дж.
Основное заглавие	Введение в системы баз данных
Сведения, относящиеся к заглавию	–
Первые сведения об ответственности	Кр. Дж. Дейт.
Место публикации	М.
Имя издателя	Вильямс
Дата	2019
Объем	1328
Международный стандартный номер	ISBN 978-5-907144-17-0
Вид содержания	Текст
Средство доступа	непосредственный

Сведения, относящиеся к заглавию, вид содержания и средство доступа могут быть опущены при их отсутствии.

БЗ источника меняется в зависимости от его типа. В научно-исследовательских работах часто используются следующие типы:

- книжное издание;
- многотомное книжное издание;
- диссертация;
- законодательные материалы;
- ГОСТы;
- статья из сборника;
- статья из газеты;
- статья из журнала;
- статья с сайта сети Интернет;
- сайт сети Интернет.

Например, для статьи из сборника необходимо дополнительное поле ввода для названия сборника, а для источника в виде сайта с сети Интернет необходимы дополнительные поля в виде ссылки на электронный источник и даты обращения.

Также на БЗ источников, имеющих сведения об авторах, влияет количество авторов. Примеры библиографических записей:

С одним автором: Каменский, П. П. Труды по истории изобразительного искусства : художественная критика / П. П. Каменский. – Санкт-Петербург : БАН, 2017. – 216 с. – ISBN 978-5-336-00204-1. – Текст : непосредственный.

С двумя, тремя авторами: Игнатьев, С. В. Принципы экономико-финансовой деятельности нефтегазовых компаний : учебное пособие / С. В. Игнатьев, И. А. Мешков – Москва : МГИМО, 2017. – 145 с. – ISBN 978-5-9228-1632-8. – Текст : непосредственный.

Более 3-х авторов: Распределенные интеллектуальные информационные системы и среды : монография / А. Н. Швецов, А. А. Суконщиков, Д. В. Кочкин [и др.]. – Курск : Университетская книга, 2017. – 196 с. – ISBN 978-5-9909988-3-4. – Текст : непосредственный.

## 1.2. Обзор аналогов

Проведем анализ существующих программных продуктов, как веб-приложений, так и настольных приложений, предназначенных для формирования списка источников.

Для выполнения поиска аналогов разрабатываемого веб-сервиса и их сравнения были выбраны следующие критерии оценки:

- Стоимость аналога;
- Доступность аналога;
- Понятное использование аналога;
- Составление БЗ по всем интересующим типам источников;
- Составление библиографического описания согласно методическим требованиям Научной библиотеки ЮГУ и ГОСТ 7.0.100-2018 «Библиографическая запись. Библиографическое описание»;
- Наличие базы созданных источников и взаимодействие с ней;
- Проверка на корректность ввода пользователем данных об источнике;
- Возможность сохранения списка источников;
- Реализация транслитерации списка источников.

На обзор были вынесены следующие программные продукты:

- Microsoft Office Word;
- SNOSKA.INFO;
- Zotero;
- Mendeley.

MS Office Word позволяет создавать ссылки на источники, списки литературы в документах. Работа со списками источников осуществляется через пункт меню «Ссылки» (рисунок 1.1).

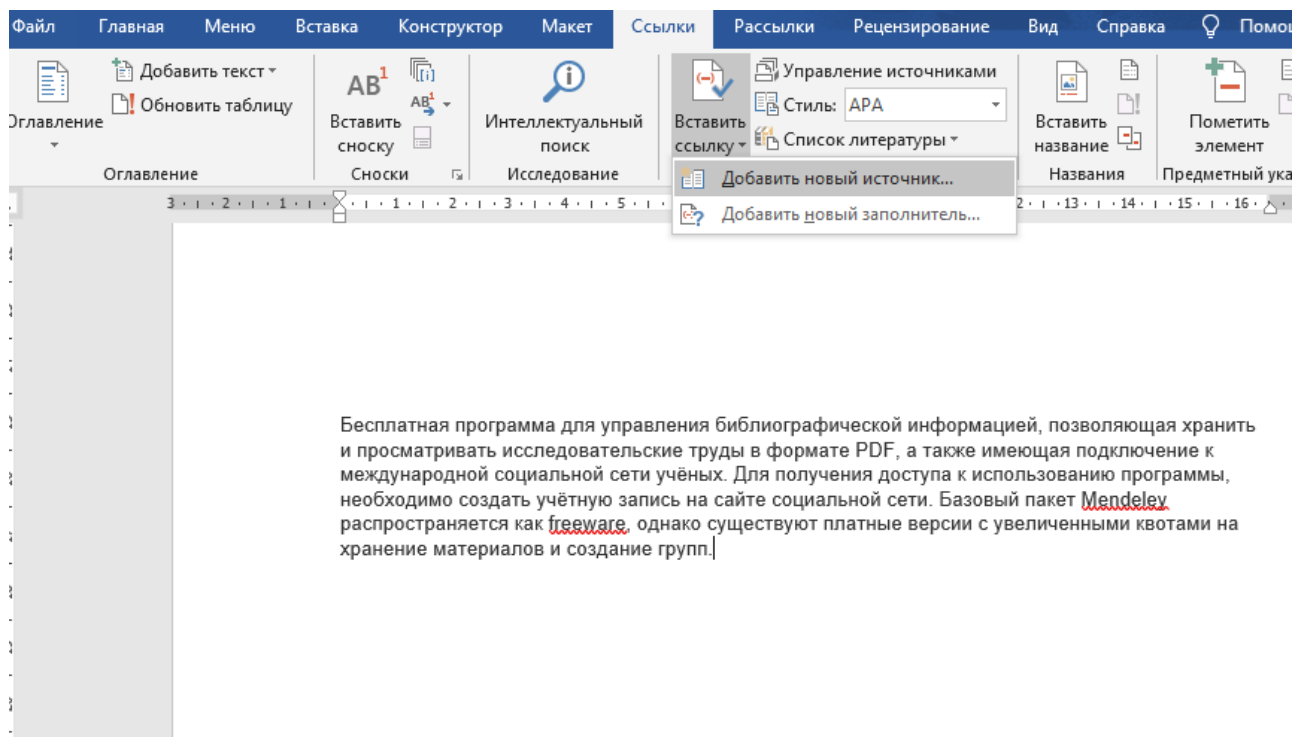


Рисунок 1.1 – Работа со списками источников в MS Office Word

Данные об источниках заносятся в специальную форму (рисунок 1.2). Далее, на основе добавленных источников можно создавать список литературы, либо вставлять ссылки в текст.

Создать источник

Тип источника: Книга | Язык: По умолчанию

Поля списка литературы для APA

\* Автор: Каменский, П. П. | Изменить

☐ Корпоративный Автор

\* Название: Труды по истории изобразительного искусства

\* Год: 2017

\* Город: Санкт-Петербург

Область, край:

Страна или регион:

\* Издательство: БАН

Редактор: П.П. Каменский | Изменить

Том:

Число томов:

☒ Показать все поля списка литературы \* Рекомендованное поле

Имя тега: Пример, Россия

Кам17

OK Отмена

Рисунок 1.2 – Добавление библиографической записи

Стандартный список стилей не содержит стиля, удовлетворяющего требованиям российского ГОСТа, поэтому приходится выбирать из того, что есть. Отсутствуют проверки на вводимые данные.

Из достоинств данного аналога это то, что можно непосредственно в тексте ссылаться на список литературы и результат работы всегда можно сохранить.

SNOSKA.INFO – веб-сервис, позволяющий оформлять библиографические ссылки. Помимо раздела с непосредственным оформлением источников, ресурс содержит ссылки на ГОСТы, раздел с ответами на вопросы, статьи на тему оформления списка источников. Внешний вид веб-сервиса представлен на рисунке 1.3.

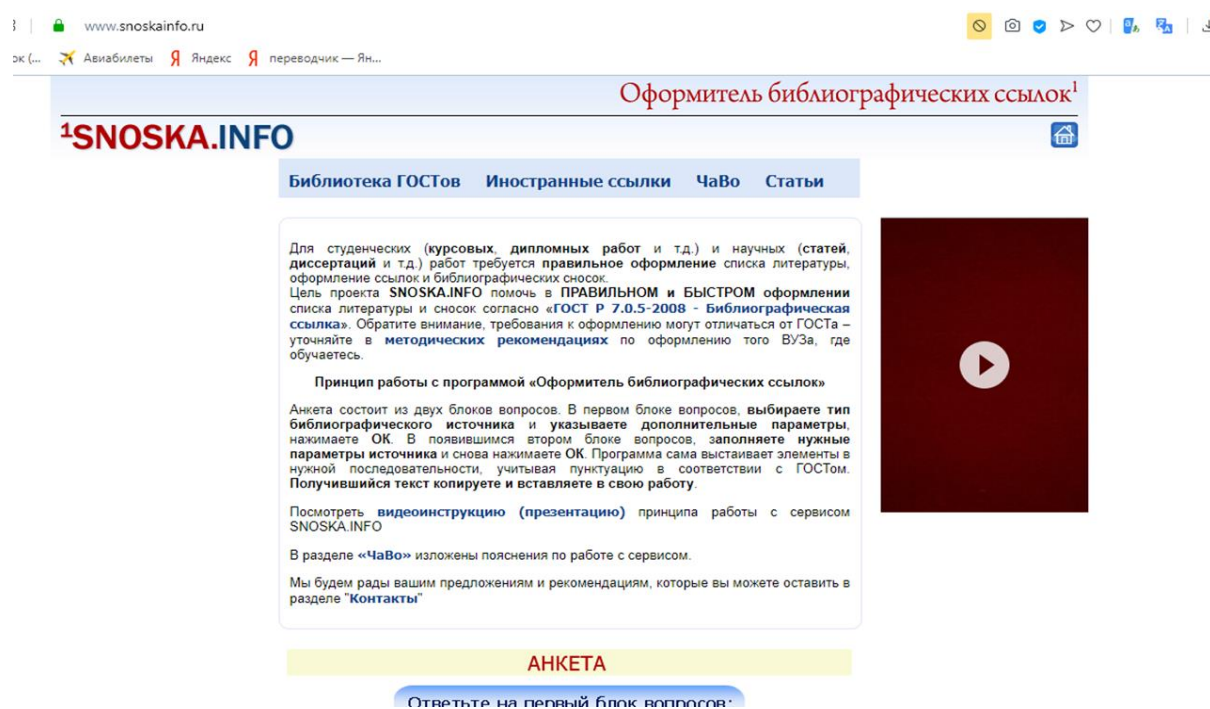


Рисунок 1.3 – Интерфейс SNOSKA.INFO

Для оформления источника с помощью SNOSKA.INFO необходимо ответить на вопросы, разделенные на два блока (рисунок 1.4).

Ответьте на первый блок вопросов:

1. Показывать пример ответа? ☐
2. Выберите тип источника:
  - ☐ Книга
  - ☐ Статья из журнала
  - ☐ Статья из сборника
  - ☐ Статья из газеты
  - ☐ Диссертация
  - ☐ Автореферат
  - ☐ Закон, нормативный акт и т.п.
  - ☐ Интернет-ресурс
3. Выберите нужный вариант, характеризующий источник:
  - ☐ 1-3 автора
  - ☐ Более 3-х авторов и/или под редакцией
4. Нужно ли указывать страницы:
  - ☐ Да
  - ☐ Нет
5. Выберите предназначение библиографической записи:
  - ☐ Для списка литературы
  - ☐ Для ссылки
6. Добавлять тире (–) между элементами библиографической записи?  
 Пример: Пантелеев А.С. Звездин А.Л. Вексель, взаимозачеты: бухгалтерский учет и налогообложение. – 4-е изд. – М.: Омега-Л, 2010. – 176 с.
  - ☐ Да
  - ☐ Нет

ok

Рисунок 1.4 – Заполнение анкеты на SNOSKA.INFO

Вопросы из первого блока являются обязательными и позволяют получить информацию о типе источника, количестве авторов, предназначении источника (ссылка, пункт списка литературы). Второй блок запрашивает информацию об источнике непосредственно. Здесь нужно указать авторов, название, год издания, издательство и т.д. Возможно отображение примеров при ответе на вопросы из второго блока. Результат генерации ссылки отображается в текстовом поле в нижней части страницы.

Вопросы второго блока не являются обязательными, можно заполнять только те поля, информация для которых имеется у исследователя. Но, отсутствие таких важных частей БЗ как авторы или название приведет к генерации некорректной ссылки на источник. Еще одним недостатком сервиса является отсутствие проверок на корректность: например, в полях для введения года и страниц можно указать буквы и специальные символы.



К достоинствам сервиса можно отнести быструю работу, составление библиографической записи близкое к требуемому оформлению, интуитивно понятный интерфейс.

Zotero – свободно распространяемое настольное программное обеспечение для оформления ссылок. Программа обладает простым интерфейсом, освоить ее можно без изучения документации. Внешний вид приведен на рисунке 1.5.

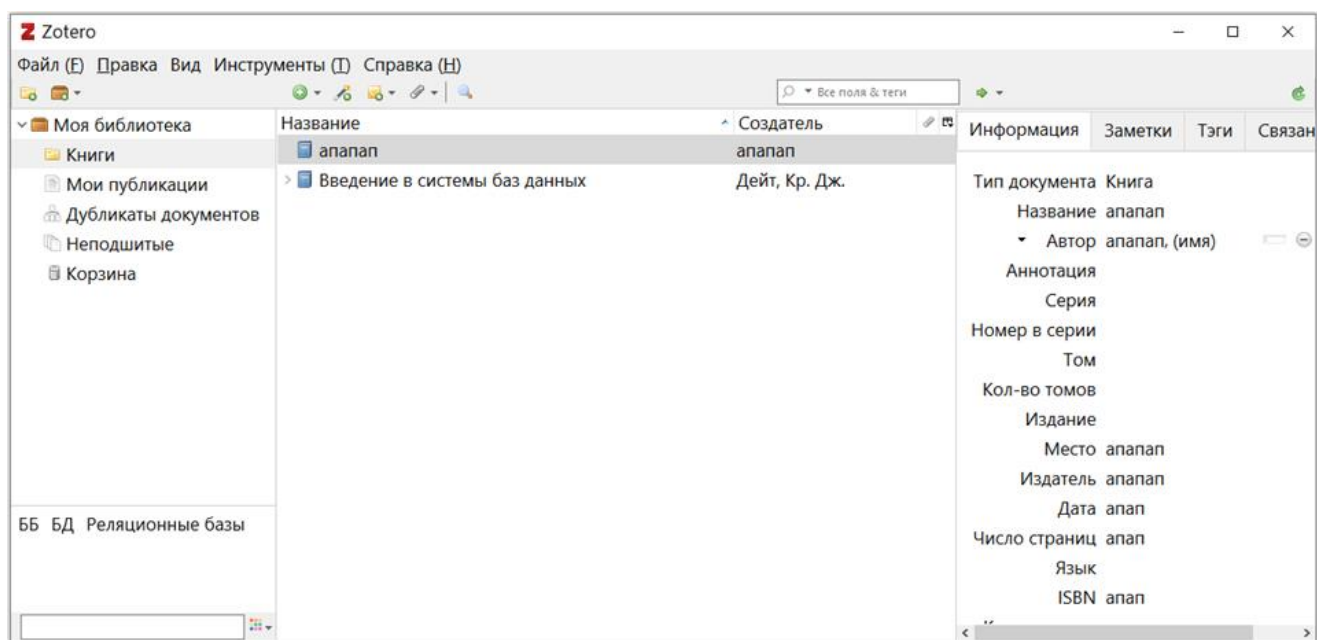


Рисунок 1.5 – Интерфейс Zotero

Zotero содержит обширный список типов источников: книги, статьи, аудиофайлы, фильмы, законопроекты, радиопередачи и т.д. Для создания источника запрашивается стандартный набор данных (рисунок 1.6). При оформлении списка можно выбрать один из стандартных стилей, либо загрузить дополнительный стиль оформления записей.

Информация	Заметки	Тэги	Связанные
<p>Тип документа Книга</p> <p>Название</p> <p>▼ Автор (фамилия), (имя)</p> <p>Аннотация</p> <p>Серия</p> <p>Номер в серии</p> <p>Том</p> <p>Кол-во томов</p> <p>Издание</p> <p>Место</p> <p>Издатель</p> <p>Дата</p> <p>Число страниц</p> <p>Язык</p> <p>ISBN</p> <p>Краткое назв.</p> <p>URL-адрес</p> <p>Дата доступа</p> <p>Архив</p> <p>Место в архиве</p> <p>Библ. каталог</p> <p>Шифр</p> <p>Права</p> <p>Дополнительно</p> <p>Добавлен 31.03.2020, 16:30:06</p> <p>Изменён 31.03.2020, 16:30:06</p>			

Рисунок 1.6 – Заполнение данных в Zotero

Недостаток – отсутствие проверки корректности вводимых данных. Также стиль оформления ГОСТ, который предлагает программа, не полностью соответствует требованиям, необходимо редактировать или создавать специальные стили оформления для каждого типа источника на языке Citation Style Language (CSL).

Для пунктов списка источников можно указывать теги, сортировать по каталогам. Готовый список может быть сохранен как RTF, HTML, скопирован в буфер обмена или распечатан. К источникам можно создавать заметки, прикреплять файлы.

Mendeley – это бесплатное академическое программное обеспечение для организации и обмена исследовательскими работами и создания библиографий, а также имеющее подключение к международной социальной сети учёных.

Работает Mendeley следующим образом: вы заводите свой аккаунт, скачиваете и устанавливаете софт (либо используете как веб-сервис) и заводите свою собственную базу карточек источников.

Программа англоязычная, русификация присутствует частично. Распознавание кириллицы при загрузке файла с книгой может пройти некорректно. Поиск по литературе не работает с русскоязычными источниками. Внешний вид рабочей области Mendeley приведен на рисунке 1.7.

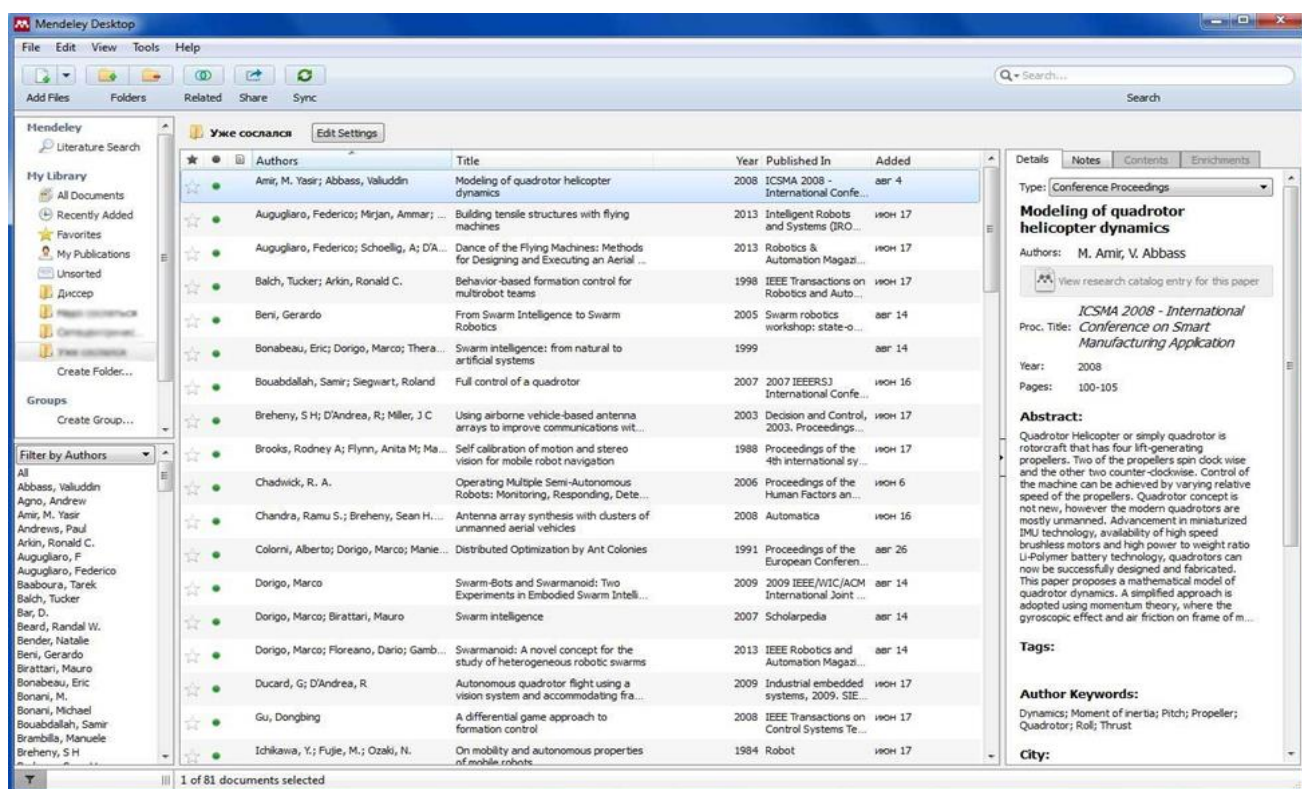


Рисунок 1.7 – Интерфейс Mendeley

Формируемый список может быть отсортирован по дате добавления, либо по алфавиту как в прямом, так и в обратном порядке. Список можно выгружать в вордовский файл. К пунктам списка могут быть добавлены заметки или файлы. Также сервис проверяет на корректность вводимые числовые данные.

Что касается шаблонов форматирования ссылок и самого списка литературы, то помимо огромного количества уже установленных по умолчанию шаблонов, можно добавлять свои. Шаблоны приходится писать на языке стиля цитат Citation Style Language (CSL).

Для обобщения всего выше сказанного была создана таблица с указанием всех критериев оценки и выбранных аналогов (таблица 1.2).

Таблица 1.2 – Результат анализа аналогов по обозначенным критериям

Аналоги Критерии	MS Office Word	SNOSKA.I NFO	Zotero	Mendeley
Стоимость аналога	\$99.99/year	Бесплатно	Бесплатно	Бесплатно
Доступность аналога	Кроссплатформенный	Сервис кроссбраузерный	Программа кроссплатформенна	Программа кроссплатформенна, есть сайт и версия для мобильного приложения
Понятное использование аналога	Удобное и понятное использование	Интуитивно понятный интерфейс, есть раздел с информацией по сайту	Есть русификация, интерфейс удобный и понятный	Нет русификации
Правильное оформление источников	-	+	-	+
Реализуются все интересующие типы источников	-	-	+	+
Наличие базы созданных источников и взаимодействие с ней	-	-	Нет общей базы источников, для каждого пользователя персональная база	Есть база источников. Но поиск по ней не работает с русскоязычными источниками
Проверка на корректность ввода	-	-	-	+
Возможность сохранения списка	Сохраняется	Не сохраняется	Сохраняется локально и онлайн	Сохраняется локально и онлайн
Транслитерация	-	-	-	-

Все рассмотренные аналоги не подходят по причине недостаточной функциональности. Можно заметить, что транслитерация библиографической записи не реализуется ни в одном аналоге.

Необходимо разработать новый продукт конкретно под потребности и задачи Научной библиотеки ЮГУ. В конечном итоге разрабатываемый продукт будет на 100% соответствовать требованиям к функционалу, не имея ничего

«лишнего», что зачастую только мешает работать и препятствует скорейшему освоению программы.

### 1.3 Методологии проектирования ПО

Для создания моделей бизнес-процессов существует достаточно большое количество подходов проектирования. Важнейшими из подходов являются структурный (функциональный) и объектно-ориентированный [1-4]. Главной отличительной чертой этих двух подходов является способ декомпозиции (разбиения) системы:

- в основе структурного подхода – алгоритмическая декомпозиция;
- в основе объектно-ориентированного – объектная декомпозиция.

Выбор того или иного метода проектирования зависит от целей и особенностей проектируемой системы.

Сущность структурного подхода к разработке программных средств заключается в ее декомпозиции (разбиении) на автоматизируемые функции: система разбивается на функциональные подсистемы, которые в свою очередь делятся на подфункции, подразделяемые на задачи и так далее. Процесс разбиения продолжается вплоть до конкретных процедур. При этом автоматизируемая система сохраняет целостное представление, в котором все составляющие компонент взаимосвязаны.

К достоинствам данного подхода относятся, прежде всего:

- возможность проведения глубокого анализа бизнес-процессов, выявления узких мест;
- применение универсальных графических языков моделирования IDEF0, IDEF1X и DFD обеспечивает логическую целостность и полноту описания, необходимую для достижения точных и непротиворечивых результатов;

- большое количество case-средств.

В качестве недостатков можно выделить такие:

- сложность моделирования динамики процесса;
- сложность исправления ошибок проектирования;
- сложность восприятия иерархически упорядоченной информации;
- необходимость следования жесткой структуре.

Объектно-ориентированный подход (ООП) подразумевает разделение системы на компоненты с разной степенью детализации. В основе этой декомпозиции лежат объекты и классы. Классы связаны разнообразными отношениями и обмениваются сообщениями, вызывающие операции над объектами.

ООП к проектированию программного обеспечения (ПО) состоит из нескольких этапов, на каждом из которых строятся определенные группы диаграмм:

1) Концептуальное проектирование программного обеспечения. На данном этапе происходит идентификация вариантов использования ПО с помощью диаграммы прецедентов, определяются потоки событий.

2) Логическое проектирование программного обеспечения. Включает в себя построение комплекса моделей (диаграмм), иллюстрирующих поведение проектируемого ПО в процессе его функционирования при взаимодействии с выявленными объектами/сущностями/системами окружающей его предметной области/внешней среды. Это диаграмма классов, деятельности, диаграмма последовательностей, диаграмма коопераций, диаграмма состояний.

3) Проектирование физической структуры программного обеспечения. К данному этапу относятся диаграммы компонентов и размещения ПО, характеризующие компонентную структуру ПО и размещение его элементов на предполагаемых аппаратных средствах.

ООП обладает следующими преимуществами:

- объектно-ориентированные системы более открыты и легче поддаются внесению изменений;

- объектная декомпозиция позволяет избежать создания сложных моделей, так как она предполагает эволюционный путь развития модели на базе относительно небольших подсистем;

- объектная модель естественна, поскольку ориентирована на человеческое восприятие мира.

К недостаткам ООП относятся:

- высокие начальные затраты;

- сложно осуществлять управление проектом.

После подробного рассмотрения каждого из подходов к проектированию ПО, выбор был сделан в пользу объектно-ориентированного. Это связано с тем, что структурная декомпозиция на основе ООП отличается от функционально-ориентированного подхода лучшей способностью отражать динамическое поведение системы в зависимости от возникающих событий. В этом плане модель проблемной области рассматривается как совокупность взаимодействующих во времени объектов. Тогда конкретный процесс обработки информации формируется в виде последовательности взаимодействий объектов. Одна операция обработки данных может рассматриваться как результат одного взаимодействия объектов.

Конечным результатом процесса ООП должно стать множество классов объектов с присоединенными методами обработки атрибутов. Если в структурном подходе модели данных и операций разрабатываются относительно независимо друг от друга и только координируются между собой, то объектно-ориентированный подход предполагает совместное моделирование данных и процессов. При этом модели проблемной области постепенно уточняются.

В связи с этим система объектно-ориентированных моделей последовательно разворачивается по направлению от модели общего представления функциональности к модели динамического взаимодействия

объектов, на основе которой могут быть сгенерированы классы объектов в конкретной программно-технической среде.

#### 1.4 Типовые подходы к разработке программного обеспечения

Существующие методологии разработки программного обеспечения можно разделить на две большие группы [5, 6]:

1) Традиционные. Традиционные методологии строго стандартизованы. Для них характерно стремление обеспечить разработчиков инструкциями, не требующими обсуждений: нужно выполнять известные предписания, соблюдать регламенты.

2) Гибкие. В гибких методологиях процесс разработки ПО постоянно адаптируется к меняющимся требованиям пользователей. Существует несколько методик, относящихся к классу гибких методологий разработки, в частности экстремальное программирование, DSDM, Scrum, FDD.

Используемая методология разработки должна соответствовать жизненному циклу программного обеспечения. Так как жизненный цикл предполагает ориентацию на предсказуемые процессы разработки программного продукта с четко обозначенными целями и требованиями перед началом разработки, то методология разработки должна относиться к традиционной. Таковыми являются Rational Unified Process (RUP) и Microsoft Solutions Framework (MSF).

Microsoft Solutions Framework – методология разработки программного обеспечения от Microsoft. MSF опирается на практический опыт корпорации Майкрософт и описывает управление людьми и рабочими процессами в процессе разработки решения. Подходит для больших и очень больших проектов. Проект делится на этапы, стадии, восстанавливаются вехи и четко определяются результаты по каждой контрольной точке.

Rational Unified Process – методология разработки программного обеспечения, созданная компанией Rational Software [7]. RUP описывает процесс



разработки, основанный на UML диаграммах. В ходе этого процесса происходит следующее:

- 1) Собираются требования к программной системе в ходе процесса управления требованиями.
- 2) Из требований выбираются нужные, важные, критичные, функциональные и нефункциональные и расставляются приоритеты.
- 3) Составляется документ «технические требования»: что нужно сделать?
- 4) Из него составляется документ «техническое задание»: как решить задачу?
- 5) Составляется набор диаграмм, описывающий техническое решение (как сделано) с нужной степенью детализации.

На новом этапе процессов методологии RUP диаграммы старого этапа перерабатываются в новые диаграммы, пока их набор не будет достаточен для процессов разработки.

RUP является одной из наиболее универсальных и продуманных методологий, позволяет разрабатывать только те артефакты и выполнять только те работы и задачи, которые необходимы в конкретном проекте. Также она базируется на широком использовании UML, что дает основание выбора данной методологии.

Выводы. В данной главе был проведен анализ предметной области и доступных аналогов программного обеспечения, в результате которого была выявлена необходимость в разработке программного продукта. Осуществлен и обоснован выбор методологий проектирования и разработки предполагаемого программного обеспечения, а именно была выбрана традиционная методология Rational Unified Process.

## 2. Проектная часть

### 2.1 Формирование технического задания на разработку веб-сервиса для автоматизации составления библиографического описания научно-методической литературы

Техническое задание (ТЗ) представляет собой документ, в котором формулируют основные цели разработки, требования к программному продукту, определяют сроки и этапы разработки и регламентируют процесс приемно-сдаточных испытаний.

Техническое задание должно содержать следующие разделы:

- общие сведения;
- назначение и цели создания (развития) системы;
- характеристика объектов автоматизации;
- требования к системе;
- состав и содержание работ по созданию системы;
- порядок контроля и приемки системы;
- требования к составу и содержанию работ по подготовке объекта автоматизации к вводу системы в действие;
- требования к документированию.

Полный состав технического задания определяет ГОСТ 34.602-89. Для того чтобы определить какие функции необходимы программному продукту, предназначенному для оформления источников списка литературы, необходимо построить модель процесса по формированию списка источников.

Для моделирования процессов используются CASE (Computer-Aided Software Engineering)-средства – специальное программное обеспечение, предназначенное для проектирования программных продуктов, процессов и т.д. [8, 9].

Для построения модели процесса по формированию списка источников использована структурная методология IDEF0, позволяющая представить

процесс в виде иерархической системы диаграмм [10]. На верхнем уровне иерархии расположена контекстная диаграмма нулевого уровня, отображающая процесс по формированию списка источников как единое целое, взаимодействующее с окружающей средой.

На последующих уровнях расположены диаграммы, представляющие детализацию контекстной диаграммы нулевого уровня сначала на уровне основных подпроцессов, затем, при необходимости, на уровне отдельных работ.

Контекстная диаграмма нулевого уровня для процесса «построение списка источников» показана на рисунке 2.1.

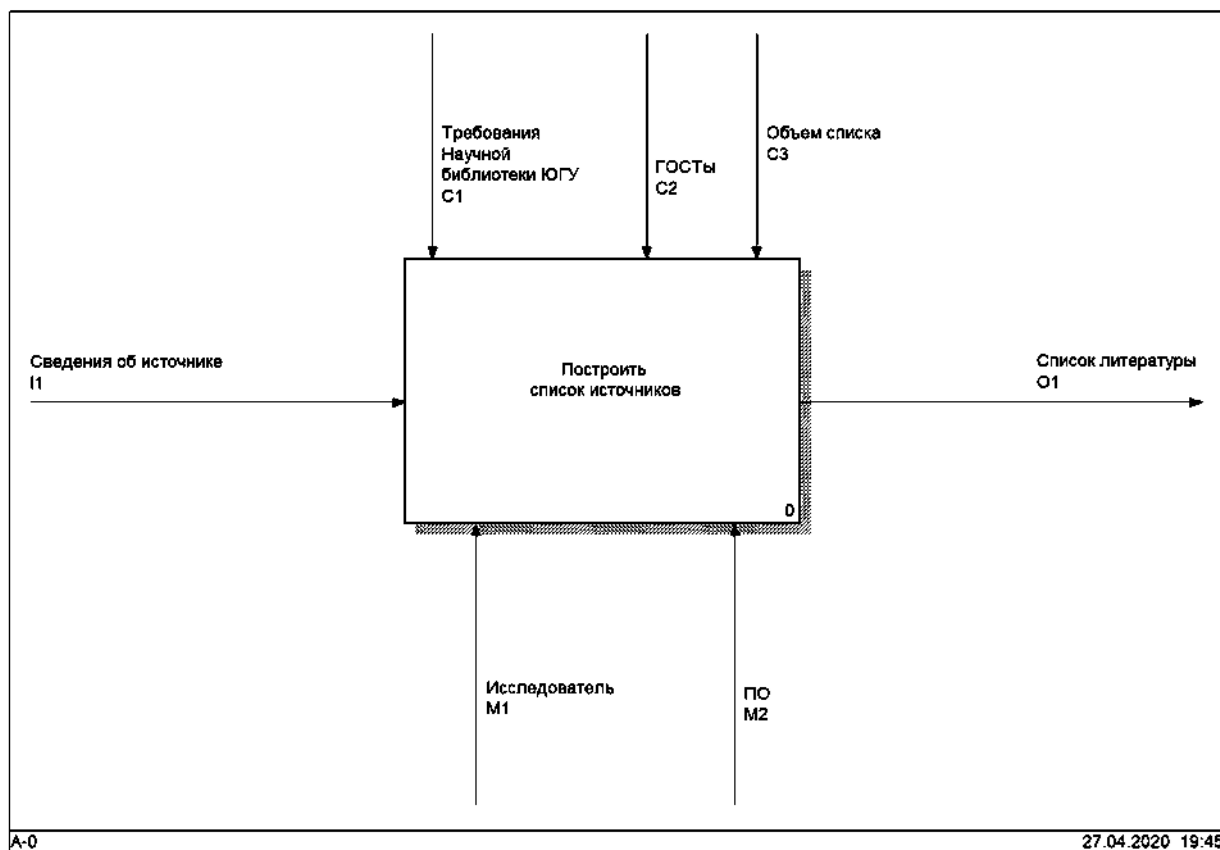


Рисунок 2.1 – Контекстная диаграмма нулевого уровня

В левой части диаграммы расположены входные данные, к которым относятся сведения об источнике, к которым относятся автор, название, издательство, место публикации, объем, тип источника и т.д.

В правой части – выходные данные – результат обработки входных данных. Сверху располагается управление – нормативные документы, регламентирующие ход выполнения работ процесса, объем списка. Внизу расположены механизмы – персонал, выполняющий работы моделируемого процесса и используемые ими инструменты.

Детализация контекстной диаграммы нулевого уровня показана на рисунке 2.2.

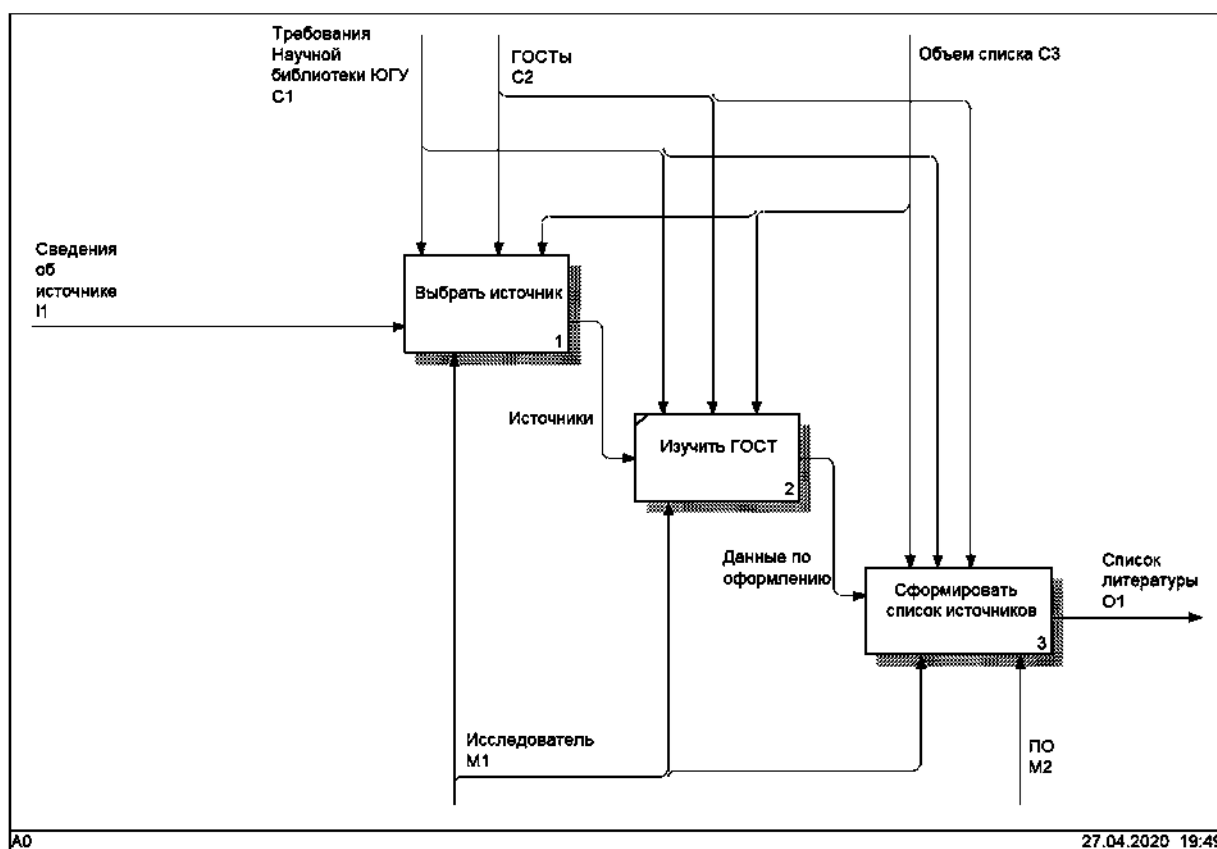


Рисунок 2.2 – Детализация контекстной диаграммы нулевого уровня

Как видно из рисунка 2.2, моделируемый процесс состоит из следующих работ:

- выбор источников – определение перечня источников, содержащих информацию по исследуемой проблеме;
- изучение ГОСТов;
- формирование списка источников согласно требованиям из ГОСТов.

На рисунке 2.3 показана детализация для подпроцесса «Выбрать источник».

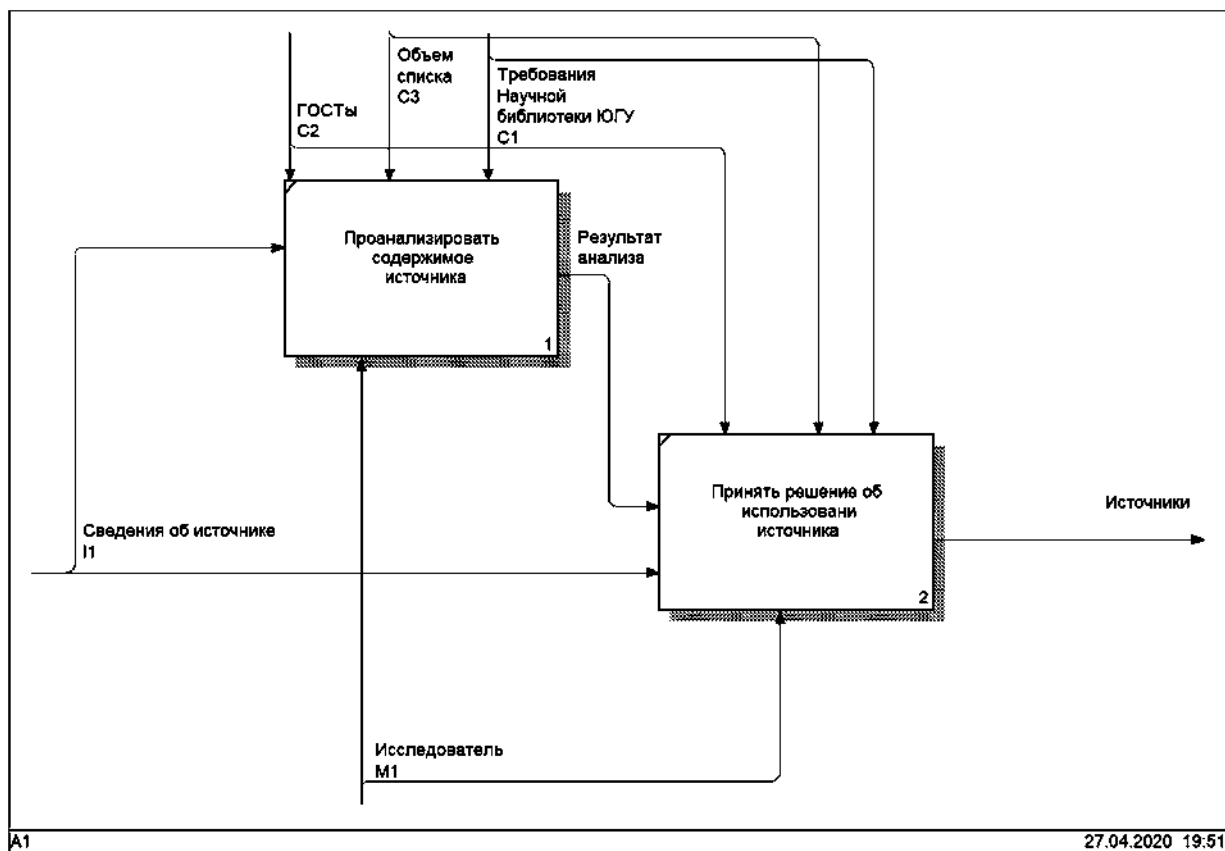


Рисунок 2.3 – Детализация подпроцесса «Выбрать источник»

Подпроцесс выбора источников состоит из анализа содержимого источника литературы с целью определения его соответствия решаемой задаче, а также выбранному процессу и способу его модернизации.

На рисунке 2.4 показана детализация для подпроцесса «Сформировать список источников». Указанный подпроцесс заключается в получении данных об источнике (автор, название, издательство, год издания, количество страниц, тип и т.д.) и последующего оформления согласно требованиям.

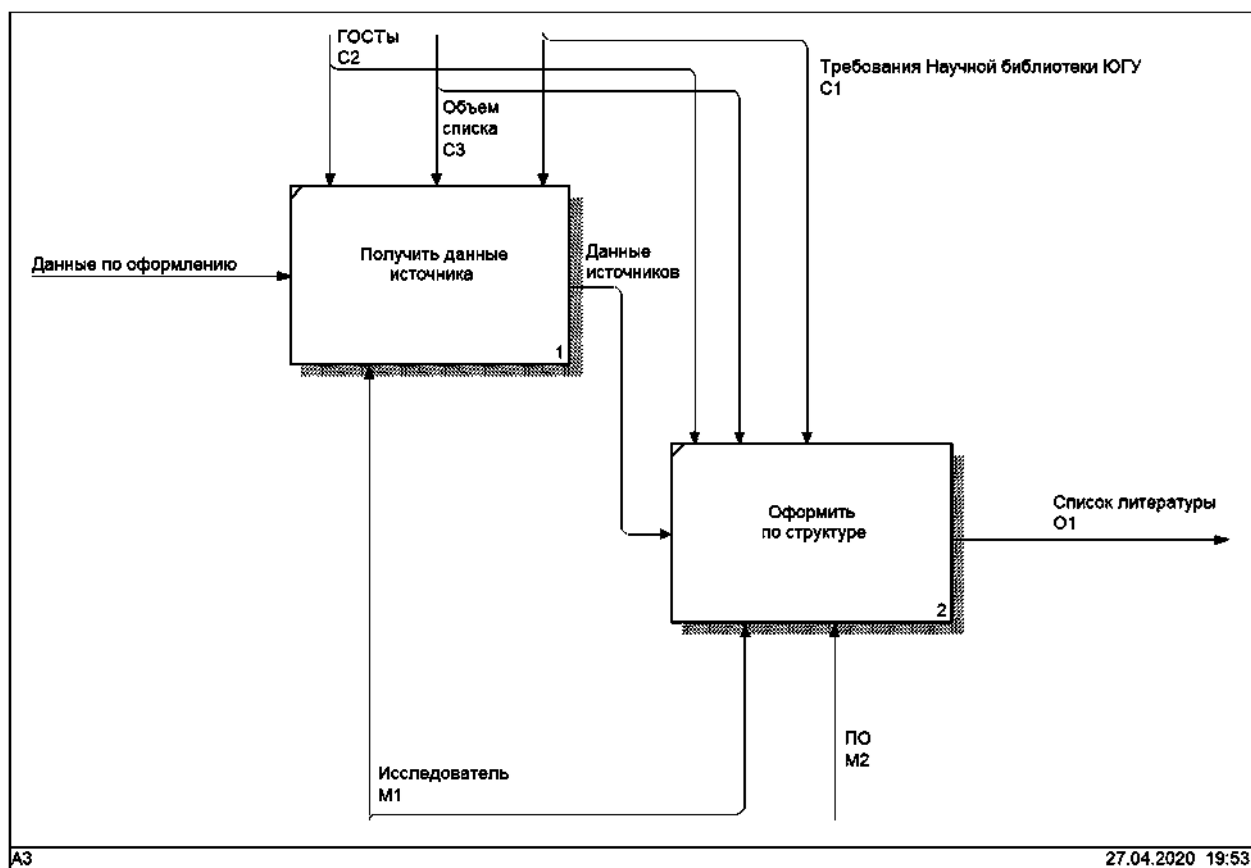


Рисунок 2.4 – Детализация для подпроцесса «Сформировать список источников»

Полное ТЗ на проектирование и разработку веб-сервиса представлено в Приложении А. В веб-сервисе должны быть реализованы следующие функции:

- хранение информации об источниках литературы в базе данных (БД);
- просмотр и последующее добавление источников, хранящихся в БД, в список литературы;
- поиск по ключевым словам среди источников;
- добавление нового источника;
- составление библиографической записи источника согласно требованиям Научной библиотеки и ГОСТ;
- транслитерация библиографической записи источника;

- формирование собственного списка из источников, хранящихся в БД;
- сохранение сформированного списка во внешний файл;
- проверка вводимых пользователем данных на корректность (соответствие типам данных, ограничение по объему).

В качестве входных данных у составителя выступают:

- заголовок (сведения об авторе/авторах);
- основное заглавие;
- первые сведения об ответственности;
- первое место публикации, производства и/или распространения;
- имя издателя, производителя и/или распространителя;
- дата публикации, производства и/или распространения;
- специфическое обозначение материала и объем;
- основное заглавие серии/подсерии или многочастного монографического ресурса (если есть);
- международный стандартный номер (ISBN, DOI);
- номер выпуска серии/подсерии или многочастного монографического ресурса;
- сведения, относящиеся к заглавию;
- средство доступа.

В качестве входных данных администратора выступают: логин/пароль.

Выходная информация представляется отображением составленного списка источников или в виде документа в формате \*.doc.

К системным требованиям к разрабатываемому веб-сервису относятся:

- 1) Необходимо создать две роли пользователей: «Гость» и «Администратор». Гостевому пользователю доступны просмотр и поиск данных об источниках, добавление собственного источника, и формирование собственного списка. Задача администратора – поддержание базы данных в

актуальном состоянии, ему доступны поиск, редактирование и удаление источников.

2) Для решения задач автоматизации средних и малых предприятий с числом одновременно работающих пользователей до 100, имеющих внутренний документооборот значительно ниже уровня 1 млн. транзакций в день рекомендуется использовать двухуровневую архитектуру. Отличительной особенностью данной системы является то, что основная нагрузка во время бизнес-вычислений ложится или на компьютер пользователя, или на сервер баз данных. Преимущества данной архитектуры:

- Простота системы по сравнению с трехуровневой архитектурой;
- Снижение нагрузки на машины сервера и клиентов;
- Пониженные требования к машинам клиентов, так как большая часть вычислительных операций будет производиться на сервере;
- Защищенность БД от несанкционированного доступа.

К требованиям интерфейса разрабатываемого веб-сервиса относятся:

1) Веб-сервис должен иметь простой и понятный интерфейс, рассчитанный на пользователя (в плане компьютерной грамотности) средней квалификации.

2) При построении интерфейса необходимо использовать типовые элементы (кнопки, текстовые поля, выпадающие списки, ссылки и т.д.) со стандартным назначением.

3) Содержать минимум управляющих кнопок и других элементов управления.

4) Страницы веб-сервиса должны корректно отображаться в современных браузерах: Яндекс.Браузер, Google Chrome, Mozilla Firefox, Opera, Safari.



## 2.2 Проектирование веб-сервиса для автоматизации составления библиографического описания научно-методической литературы

В данной работе выбран объектно-ориентированный подход к проектированию веб-сервиса (см. п. 1.2), так как данный подход позволяет строить модели, максимально близкие к сущностям реального мира. Данные и методы, используемые при их обработке, объединяются в единую сущность (класс), что позволяет создавать легко воспринимаемую структуру проектируемой системы.

В качестве средства для проектирования использован унифицированный язык UML (Unified Modelling Language) [11, 12]. Он одновременно является простым и мощным средством моделирования, который может быть эффективно использован для построения концептуальных, логических и графических моделей любого процесса или систем, охарактеризовав деятельность и структуру моделируемого объекта.

### 2.2.1 Концептуальное проектирование веб-сервиса для автоматизации составления библиографического описания научно-методической литературы

Поведение разрабатываемого веб-сервиса (т.е. функциональность, которую он обеспечивает) можно описать с помощью соответствующей функциональной модели, которая отображает системные прецеденты (use cases, случаи использования), системное окружение (действующих лиц, акторов, actors) и связи между ними (use cases diagrams).

Диаграмма прецедентов – диаграмма, отражающая возможные варианты взаимодействия веб-сервиса (программного обеспечения/системы) с его пользователями с учетом возможных связей между ними и имеющимся внешним окружением системы.

Под актором понимают любой объект, субъект или систему, взаимодействующую с моделируемым ПО или системой извне.

Прецедент – это функциональность системы, позволяющая пользователю получить некий значимый для него, осязаемый и измеримый результат. Каждый прецедент соответствует отдельному сервису, предоставляемому моделируемой системой в ответ на запрос пользователя, т.е. определяет способ использования этой системы.

Определим акторов и прецедентов для рассматриваемой предметной области. Взаимодействие с веб-сервисом извне происходит у сотрудника библиотеки (он же администратор) и у составителя списка литературы. Также веб-сервис обращается к базе данных. Таким образом, с рассматриваемым программным обеспечением взаимодействуют: составитель (роль Гостя), сотрудник (роль Администратора), БД.

Детальное описание всех прецедентов/акторов приведено в таблице 2.1. Описание связей между ними приведено в Приложении Б.

Таблица 2.1 – Прецеденты

№	Название прецедента	Описание прецедента
1	Создать список литературы	Формируется пронумерованный список из добавленных источников согласно требованиям
2	Очистить список	Составитель может удалить созданный ранее список
3	Сохранить список в файл	Составитель может сохранить созданный список во внешний файл
4	Провести транслитерацию	Составитель может сделать перевод списка на английский язык
5	Добавить новый источник	Составитель заполняет необходимые данные об источнике, источник выводится в список согласно требованиям и сохраняется в БД
6	Выбор типа источника	Составитель выбирает тип источника, который ему необходим в списке, и в зависимости от типа меняется форма заполнения
7	Поиск источников	Составитель и сотрудник могут по определенным атрибутам осуществлять поиск источников по БД

Продолжение Таблицы 2.1

8	Просмотр данных	Вся необходимая информация об источниках и списке отображается на сайте
9	Авторизация	Только сотрудник библиотеки имеет доступ к данным БД веб-сервиса и авторизуется в нем перед использованием
10	Удалить источник	Сотрудник может удалить источник из БД
11	Редактировать источник	Сотрудник может изменить данные об источнике в БД

На основе получившихся прецедентов была создана диаграмма прецедентов (рисунок 2.5).

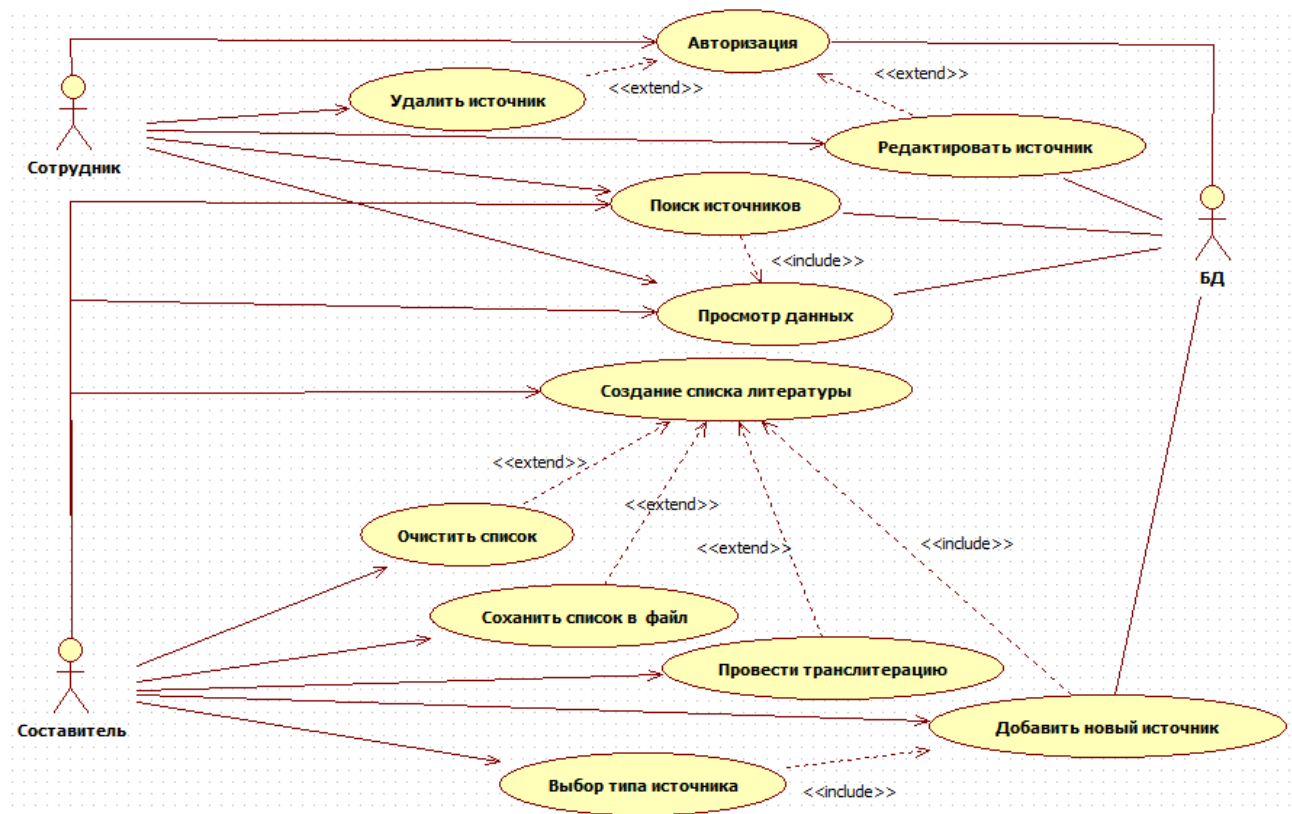


Рисунок 2.5 – Диаграмма вариантов использования

Диаграмму прецедентов согласно рекомендациям стандарта UML сопровождают описанием основного и возможных альтернативных потоков событий. Описание потоков приведено в приложении В.

### 2.2.2 Логическое проектирование веб-сервиса для автоматизации составления библиографического описания научно-методической литературы

После того, как была разработана концептуальная модель ПО, переходим к формированию его логической модели. Логическая модель ПО описывает ключевые абстракции программного обеспечения (классы, интерфейсы и т. п.), т. е. средства, обеспечивающие требуемую функциональность.

Анализ возможных условий функционирования проектируемого веб-сервиса, основанный на построенной диаграмме вариантов его использования (рисунок 2.5) позволил выделить ключевой прецедент – прецедент «Создание списка литературы». Для данного прецедента будут построены все необходимые диаграммы.

Разработка логической модели ПО, в частности, разработка диаграммы классов занимает центральное место в проектирование программного обеспечения.

Диаграмма классов служит для представления статической структуры модели ПО в терминологии классов объектно-ориентированного программирования. Диаграмма классов может отражать различные взаимосвязи между отдельными сущностями предметной области, такими как объекты и подсистемы, а также описывает их внутреннюю структуру и типы отношений. Диаграмма классов является дальнейшим развитием концептуальной модели проектируемого программного обеспечения.

На основании описания предметной области и спроектированной диаграммы вариантов использования были выбраны следующие классы проектируемого веб-сервиса (таблица 2.2).

Таблица 2.2 – Возможные классы

ВАРИАНТ ИСПОЛЬЗОВАНИЯ	ВОЗМОЖНЫЕ КЛАССЫ
Страница формирования списка источников	CreateListSource
Форма добавления нового источника	CreateSource
Авторизация администратора	AuthFormAdmin
Управление источниками, хранящимися в БД	ControllerSource
Содержит информацию об источнике, пункте списка литературы	Source
Содержит информацию о типе источника	TypeSource
Содержит информацию об авторах источника	AuthorSource
Управление данными авторизации пользователя	AuthControllerUser
Содержит информацию о данных администратора	Admin

Тип «entity» отвечает за хранение данных и как правило, описывает структуру БД. В разрабатываемом веб-сервисе к типу «entity» относятся следующие сущности:

- Source;
- AuthorSource;
- TypeSource;
- Admin.

Тип «boundary» необходим для отображения всего того, с чем пользователь может производить манипуляции, то есть пользовательский интерфейс. В разрабатываемом веб-сервисе этот тип представлен классами:

- CreateListSource;
- CreateSource;

- AuthFormAdmin.

Тип «control» это функционал веб-сервиса. Обработка функционала начинается с того момента, как пользователь выбрал его в типе «boundary». Данный тип представляется классами:

- ControllerSource;
- AuthControllerUser.

Эскиз диаграммы классов проектируемого веб-сервиса представлен на рисунке 2.6.

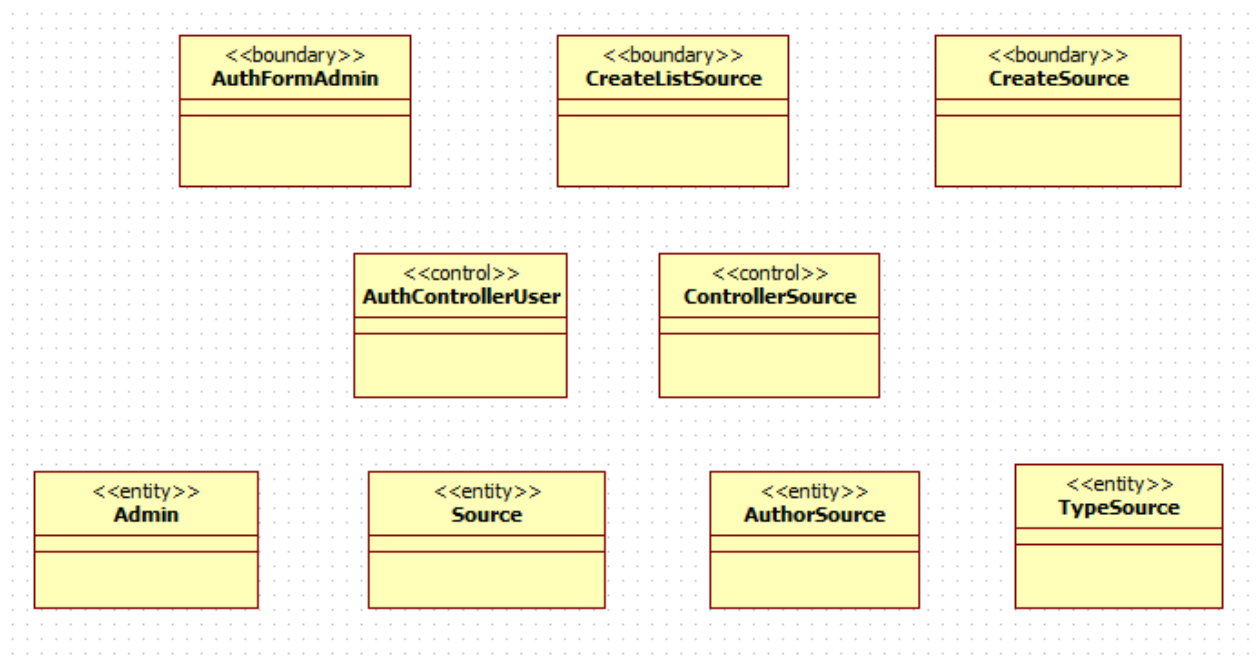


Рисунок 2.6 – Эскиз диаграммы классов

Для отображения всех действий проектируемого веб-сервиса и определения их синхронизации используются диаграммы деятельности.

Каждое состояние на диаграмме деятельности соответствует выполнению некоторой элементарной операции, а переход в следующее состояние выполняется только при завершении этой операции.

Диаграмма деятельности для варианта использования «Создание списка литературы», поскольку в нём, как отмечалось выше, сосредоточен

основной/ключевой функционал проектируемого веб-сервиса приведена на рисунках 2.7 и 2.8 соответственно.

Действия в диаграмме деятельности определяются в соответствии с потоками событий:

- 1) Составитель переходит на сайт.
- 2) Отображается страница для создания списка литературы.
- 3) Составитель выбирает параметры поиска и ищет необходимый источник.
- 4) Веб-сервис составляет поисковый запрос в БД для проверки существования запрашиваемого источника.
- 5) БД обрабатывает запрос и возвращает результат.
- 6) Результат отображается на странице.
- 7) Составитель переходит к созданию источника.
- 8) Составитель выбирает тип источника.
- 9) Составитель выбирает количество авторов у источника.
- 10) Отображается форма, в соответствии с выбранными данными.
- 11) Составитель заполняет форму.
- 12) Веб-сервис составляет запрос в БД для сохранения введенных данных.
- 13) БД обрабатывает запрос.
- 14) Веб-сервис формирует библиографическую запись источника и добавляет ее к списку источников.
- 15) Список отображается на странице.
- 16) Составитель указывает, что нужна транслитерация списка.
- 17) Веб-сервис переводит список источников.
- 18) Пользователь сохраняет список во внешний файл.
- 19) Веб-сервис осуществляет сохранение во внешний файл.

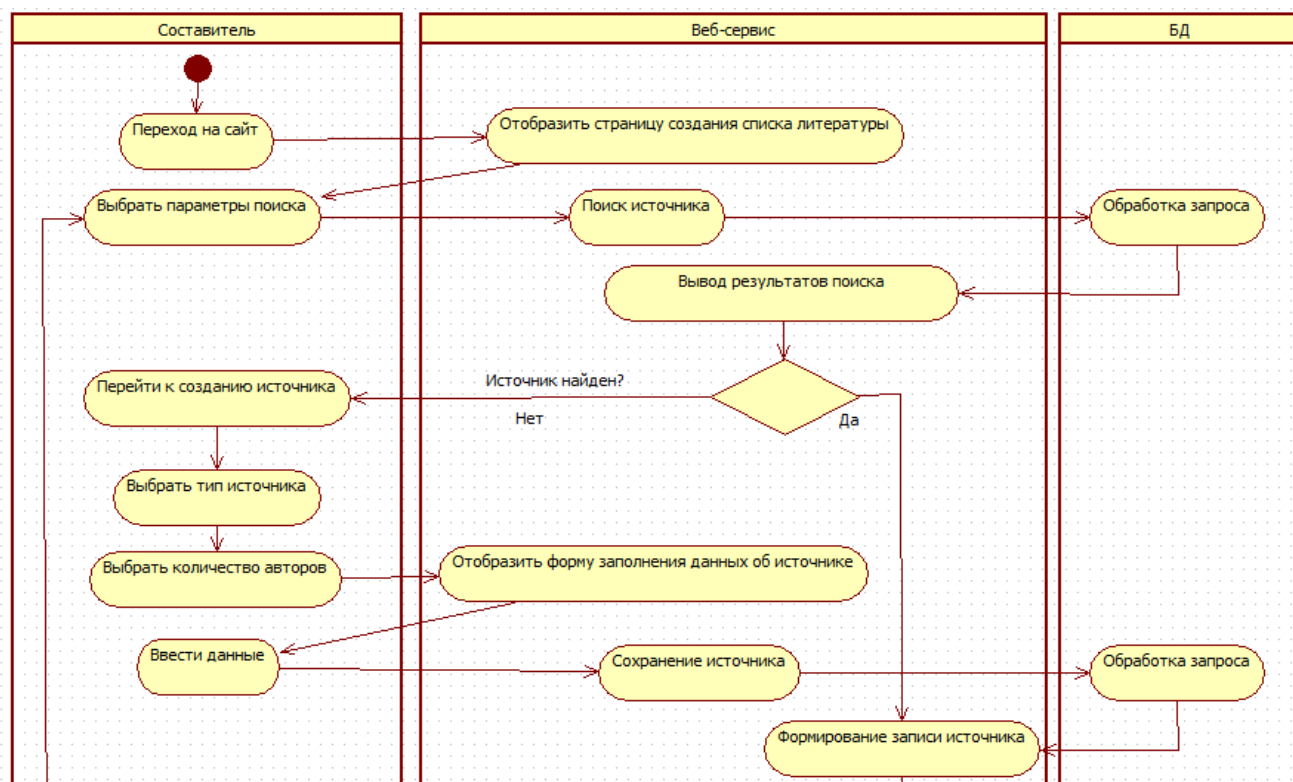


Рисунок 2.7 – Диаграмма деятельности для прецедента «Создание списка литературы» (начало)

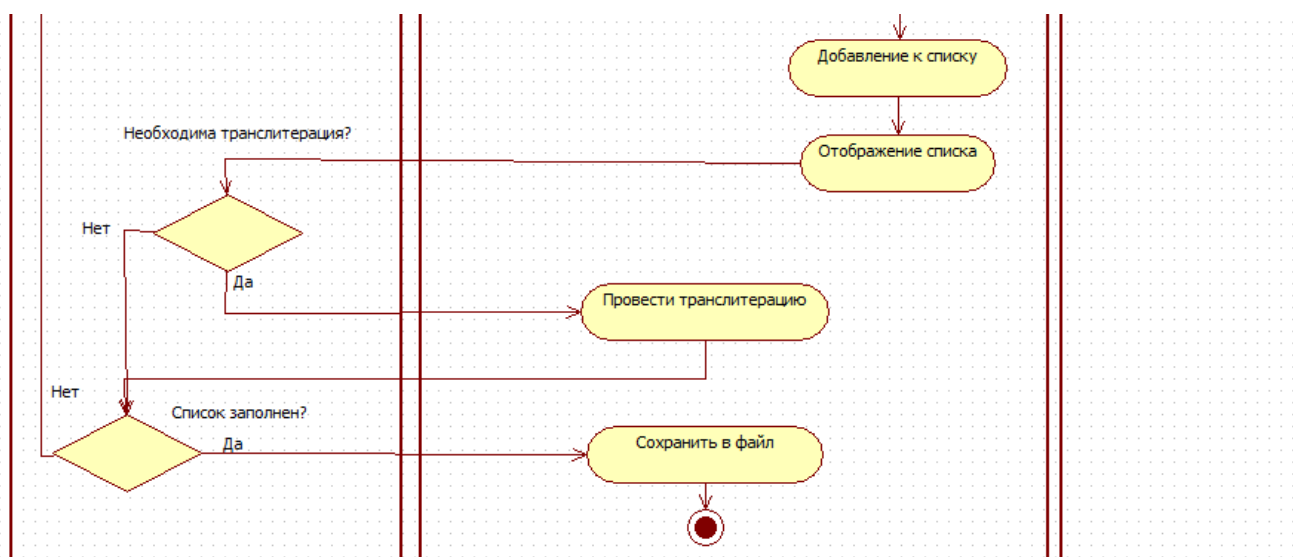


Рисунок 2.8 – Диаграмма деятельности для прецедента «Создание списка литературы» (окончание)

Для моделирования взаимодействия объектов в языке UML используются соответствующие диаграммы взаимодействия. Диаграмма имеет два аспекта взаимодействия. Первый аспект – это взаимодействие объектов можно



рассматривать во времени, и тогда для представления временных особенностей передачи и приема сообщений между объектами используется диаграмма последовательности. Второй аспект подразумевает, что в диаграмме можно рассматривать структурные особенности взаимодействия объектов. Для представления структурных особенностей передачи и приема сообщений между объектами используется диаграмма кооперации.

Диаграммы последовательности (sequence diagram) являются видом диаграмм взаимодействия языка UML, которые описывают отношения объектов в различных условиях. Условия взаимодействия задаются сценарием, полученным на этапе разработки диаграмм вариантов использования.

Разберем каждый элемент диаграммы, по отдельности:

- Объект, Участник (Object, Participant);

Обозначается прямоугольником, в котором указывается информация об участнике действий. Это, как правило, название объекта и его класс, разделенные двоеточием.

- Линия жизни (Life Line);

Линия, идущая вниз от участника, обозначающая отведенное объекту время жизни. Обозначается пунктирной линией.

- Активация, фрагмент выполнения (Activation Bar, Execution Occurances);

Обозначается узким прямоугольником (серого или белого цвета), расположенным на линии жизни. Указывает начало и завершение действия, в котором участвует объект. Поскольку линия жизни – это метафора времени, то прямоугольник на линии жизни указывает на активизацию объекта во времени.

- Сообщение, Стимул (Message, Stimulus) Стрелка от одной жизни к другой. Показывает взаимодействие объектов.

Составленная диаграмма последовательности представлена в Приложении Г.

Объекты, используемые для построения диаграммы:

- объект класса «Составитель»;
- объект класса «БД»;
- объект класса «CreateListSource»;
- объект класса «CreateSource»;
- объект класса «ControllerSource»;
- объект класса «Source»;
- объект класса «TypeSource»;
- объект класса «AuthorSource».

Выше было отмечено, что особенность взаимодействия элементов моделируемого веб-сервиса могут быть представлены на диаграммах последовательности и кооперации.

Диаграмма кооперации предназначена для спецификации структурных аспектов взаимодействия. Главная особенность диаграммы кооперации заключается в возможности графически представить не только последовательность взаимодействия, но и все структурные отношения между объектами, участвующими в этом взаимодействии.

Сформированная диаграмма кооперации проектируемого веб-сервиса для прецедента «Создание списка литературы» представлена в Приложении Д.

### 2.2.3 Проектирование физической структуры веб-сервиса для автоматизации составления библиографического описания научно-методической литературы

В данном разделе приведены результаты проектирования физической структуры проектируемого программного обеспечения и взаимосвязи его компонентов (элементов) с аппаратными средствами среды, где предполагается конечное развертывание и эксплуатация спроектированного программного обеспечения.

Диаграмма компонентов описывает особенности физического представления системы, которая позволяет определить архитектуру разрабатываемой системы, установив зависимости между программными компонентами, в роли которых может выступать исходный, бинарный и исполняемый код. Во многих средах разработки модуль или компонент соответствует файлу. Стрелки, соединяющие модули, показывают отношения взаимозависимости.

Для проектируемого веб-сервиса выбрана двухуровневая архитектура. Обоснование выбора архитектуры было обозначено в 2.1.

Сформированная диаграмма компонентов представлена в Приложении Е.

Диаграмма развертывания разрабатывается для следующих целей:

- определить распределение компонентов программного обеспечения по его физическим узлам;
- показать физические связи между всеми узлами реализации программного обеспечения на этапе его исполнения;
- выявить узкие места программного обеспечения и реконфигурировать его топологию для достижения требуемой производительности.

Узел (node) представляет собой физически существующий элемент системы, который может обладать вычислительным ресурсом или являться техническим устройством. На части этих узлов и развертывается программное обеспечение системы. В рассматриваемом проекте такими узлами являются клиентский компьютер и сервер запросов.

Далее необходимо выбрать соединение между узлами, для этого используется протокол TCP/IP.

Диаграмма развертывания представлена на рисунке 2.9.

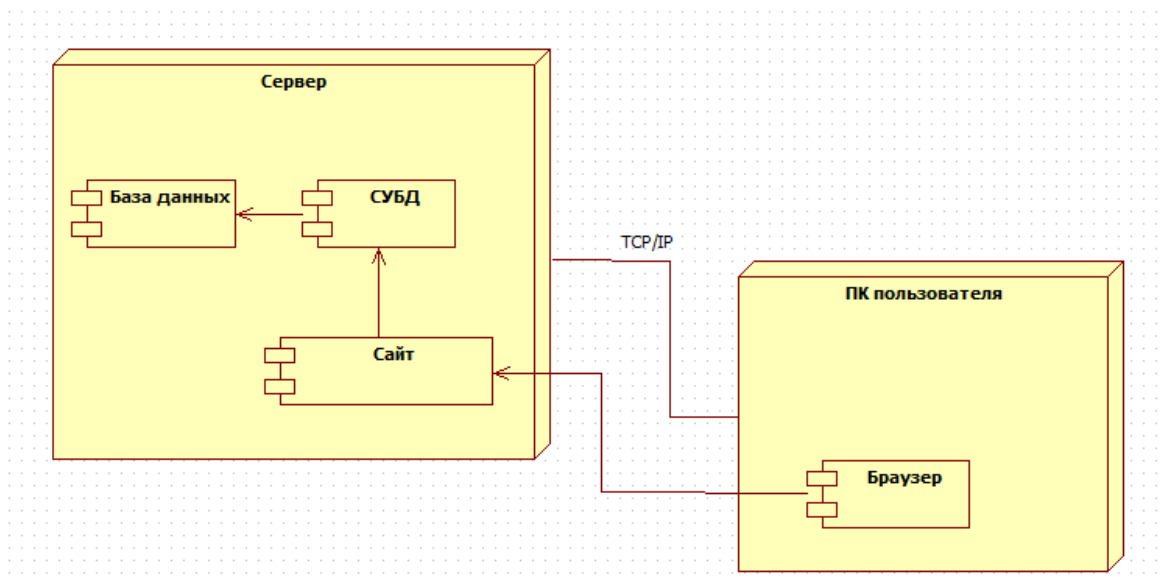


Рисунок 2.9 – Диаграмма развертывания

Как видно из рисунка 2.9, файлы веб-сервиса будут располагаться на сервере, подключенном к интернету. Для получения доступа к сайту пользователю необходим компьютер, подключенный к интернету. Просмотр страниц сайта будет осуществляться через браузер.

Выводы.

Проведенный анализ описания предметной области позволил сформировать требования к проектируемому программному обеспечению, формализованное описание которых приведено в техническом задании (Приложение А). Анализ возможных условий функционирования проектируемого программного обеспечения, основанный на построенной диаграмме вариантов его использования, позволил выделить ключевой прецедент эксплуатации ПО – прецедент «Создание списка литературы»

В рамках логического проектирования программного обеспечения на основании результатов этапа его концептуального проектирования была сформирована логическая модель, представленная его полностью специфицированной диаграммой классов. Построенный комплекс диаграмм, характеризующих поведение проектируемого веб-сервиса, позволяет

охарактеризовать все аспекты его функционирования применительно к рассмотрению ключевого варианта использования веб-сервиса.

В результате реализации этапа физического проектирования с учетом выбора архитектуры, обоснование которого было осуществлено на этапе разработки технического задания (2.1, Приложение А), были сформированы диаграммы, характеризующие компонентную структуру веб-сервиса и размещение его элементов на предполагаемых аппаратных средствах. Построенные диаграммы компонентов и размещения в целом отражают ключевые требования технического задания как по заданным параметрам передачи и хранения данных, так и по ресурсным ограничениям программно-аппаратных средств, на которых предполагается эксплуатация проектируемого веб-сервиса.

### 2.3 Проектирование базы данных веб-сервиса для автоматизации составления библиографического описания научно-методической литературы

Проектирование структуры базы данных происходит в два этапа [14]:

- построение концептуальной модели;
- построение логической модели.

При построении концептуальной модели выявляются основные сущности предметной области на основе ее анализа. Между выделенными сущностями устанавливаются связи, выделяются атрибуты для каждой сущности.

Для построения концептуальной модели использована нотация Чена. Использование нотации Чена позволяет отобразить модель данных в абстрактном виде без привязки к определенным инструментам разработки, типам баз данных. Модель, построенная с помощью нотации Чена позволяет наглядно отобразить связи между сущностями предметной области, для ее понимания не нужны специализированные знания.

Концептуальная модель для базы данных веб-сервиса по формированию списка источников приведена на рисунке 2.10.

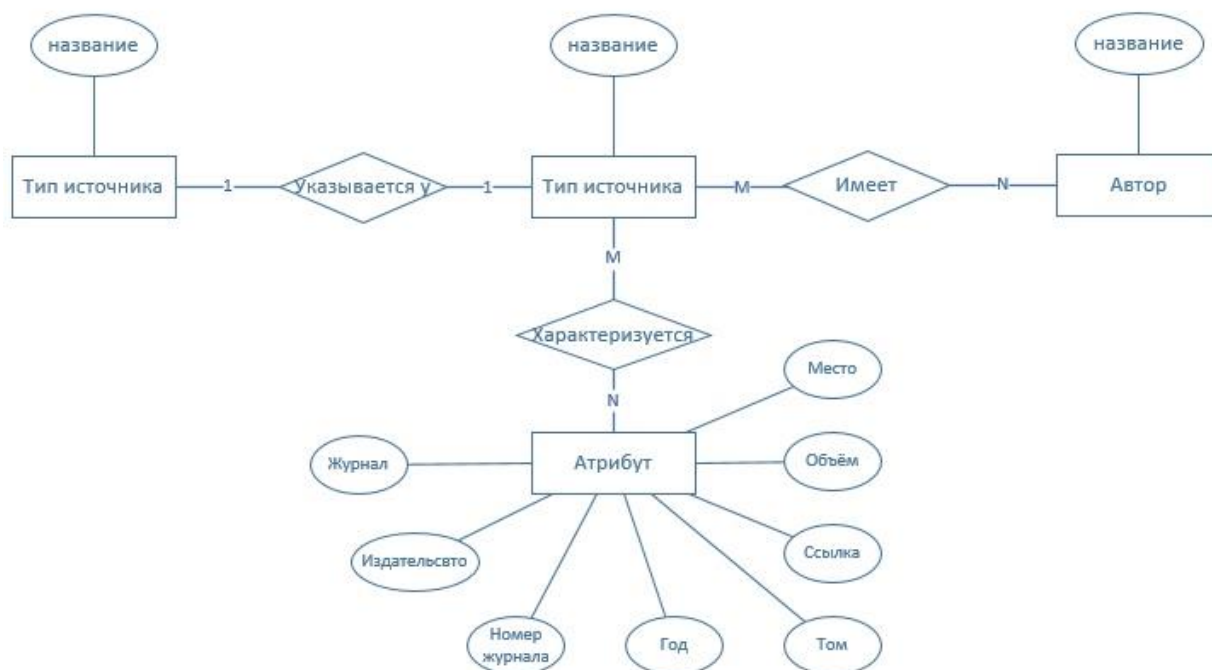


Рисунок 2.10 – Концептуальная модель базы данных

Как видно из рисунка 2.10, основными сущностями предметной области являются:

- Источник – пункт списка литературы в работе;
- Тип источника – книга, статья в журнале, электронный ресурс и другие типы, к которым может быть отнесен источник;
- Автор – автор источника выделен в отдельную сущность, так как авторов может быть несколько, а также, один автор может создать несколько источников;
- Атрибут – характеристика источника: издательство, объем, название журнала, ссылка, вынесен в отдельную сущность, так как разные типы источников имеют разные атрибуты.

Следующий шаг – построение логической модели. На данном этапе необходимо выбрать тип базы данных для построения объектов базы,

соответствующих выбранному типу. В данной работе для построения базы данных выбран реляционный тип. Так как в реляционной базе данных вся информация хранится в двумерных таблицах с фиксированной структурой. Для разработки базы данных будет использована реляционная СУБД MySQL 8.

Логическая модель была разработана с помощью методологии IDEF1X, основанной на концепции «сущность-связь» [15]. Данный стандарт подходит для разработки реляционных баз данных. Основным преимуществом IDEF1X, по сравнению с другими многочисленными методами разработки реляционных баз данных, такими как ER и ENALIM является жесткая и строгая стандартизация моделирования. Установленные стандарты позволяют избежать различной трактовки построенной модели, что является значительным недостатком ER.

Логическая модель базы данных приведена на рисунке 2.11.

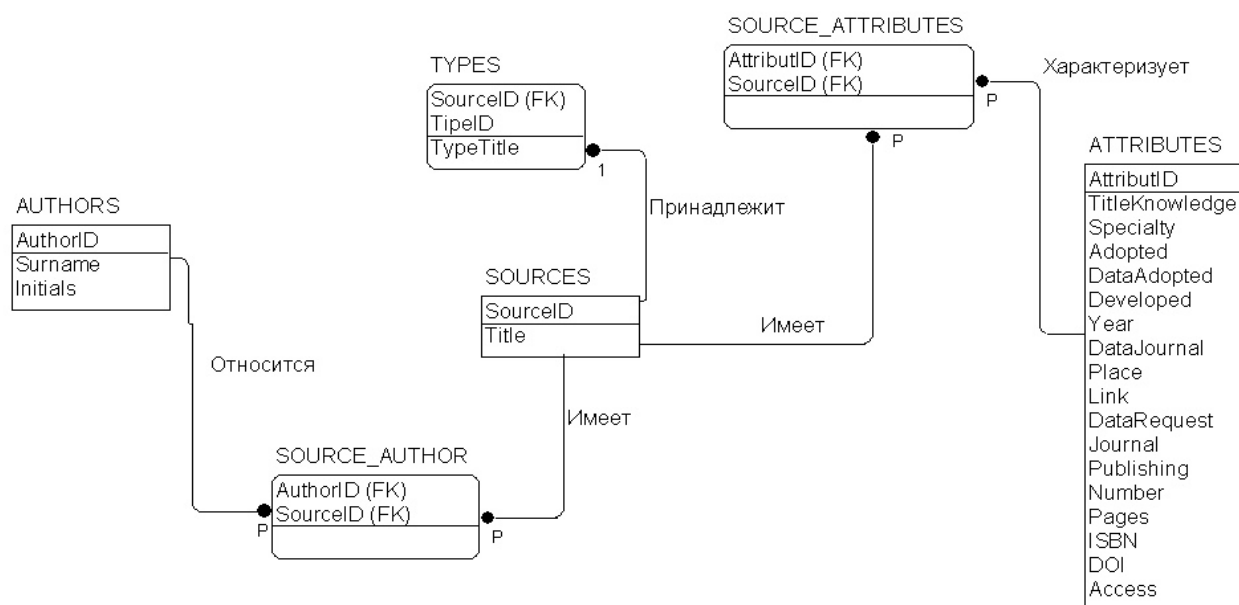


Рисунок 2.11 – Логическая модель базы данных

Как видно из рисунка 2.11, на логической модели разделены связи типа «многие-ко-многим» с помощью создания дополнительных сущностей. Таблица Source\_Author используется для разделения связи «многие-ко-многим» между таблицами Sources и Authors, так как у источника может быть несколько авторов, один автор может относиться к нескольким источникам. Таблица

Source\_Attributes используется для разделения связи типа «многие-ко-многим» между Sources и Attributes: один источник может обладать несколькими атрибутами, один атрибут соответствует нескольким источникам. Перечень атрибутов источника зависит от его типа.

Результаты установления зависимостей между сущностями, т.е. определения мощности отношения между ними представлены в таблице 2.3.

Таблица 2.3 – Мощность связей между сущностями

№	Сущность	Зависимость отношения	Сущность
		Имя отношения	
1	Sources	<u>1: 1 (один к одному)</u> Принадлежит	Types
2	Attributes	<u>1 : P (один ко многим)</u> Характеризует	Source_Attributes
3	Sources	<u>1 : P (один ко многим)</u> Имеет	Source_Attributes
4	Authors	<u>1 : P (один ко многим)</u> Относится	Source_Author
5	Sources	<u>1 : P (один ко многим)</u> Имеет	Source_Author

Характеристики атрибутов, представленных на логической модели, приведены в таблицах 2.4-2.7.

Таблица 2.4 – Author (Авторы)

Название столбца	Тип данных	Описание
AuthorID	Int(10)	Идентификатор автора, первичный ключ
Surname	Varchar(80)	Фамилия автора
Initials	Varchar(80)	Инициалы автора

Таблица 2.5 – Source (Источники)

Название столбца	Тип данных	Описание
SourceID	Int(10)	Идентификатор источника, первичный ключ
Title	Varchar(255)	Название источника



Таблица 2.6 – Type (Тип источника)

Название столбца	Тип данных	Описание
TypeID	Int(10)	Идентификатор типа источника, первичный ключ
SourceID	Int(10)	Идентификатор источника, внешний ключ
TitleType	Varchar(255)	Название типа источника

Таблица 2.7 – Attributes (Характеристики)

Название столбца	Тип данных	Описание
AttributID	Int(10)	Идентификатор характеристики, первичный ключ
TitleKnowledge	Varchar(80)	Сведения, относящиеся к заглавию
Specialty	Varchar(100)	Сведения о специальности
Adopted	Varchar(80)	Когда и кем принят
DataAdopted	Varchar(80)	Дата введения
Developed	Varchar(80)	Кем разработан
Year	Int(10)	Год публикации источника
DataJournal	Varchar(20)	Дата публикации
Place	Varchar(100)	Место публикации
Link	Varchar(300)	Ссылка на электронный источник
DataRequest	Varchar(20)	Дата обращения
Journal	Varchar(300)	Название журнала/сборника
Publishing	Varchar(100)	Издательство
Number	Int(10)	Сведения о нумерации
Pages	Varchar(20)	Страницы (количество или диапазон)
ISBN	Varchar(40)	Международный стандартный номер
DOI	Varchar(80)	Идентификатор цифрового объекта
Access	Varchar(40)	Средство доступа

### 3. Практическая часть

#### 3.1 Выбор модели жизненного цикла программного обеспечения

Жизненный цикл (ЖЦ) программного обеспечения содержит все этапы существования программного продукта, начиная с появления идеи о создании программного продукта и заканчивая выводом его из эксплуатации в связи с моральным устареванием.

Выбор той или иной модели при разработке программного обеспечения оказывает значительное влияние на успешность проекта по разработке. Выбор модели зависит от того насколько подробно сформулированы требования, насколько высока вероятность их изменения, разрабатывается ли система с нуля, либо у нее уже есть прототип. При выборе неподходящей модели жизненного цикла велика вероятность того, что проект будет завершён неудачно.

На данный момент, существуют следующие модели жизненного цикла программного обеспечения:

1) Модель кодирования и устранения ошибок – наименее гибкая модель, применима только для простейших программ.

2) Каскадная модель – процесс состоит из последовательной смены жестко задокументированных этапов, корректировка требований в процессе разработки системы невозможна. Подходит для разработки программы, чьи функциональные возможности четко определены и не будут изменяться.

3) V-модель – разработка программного продукта ведется через тестирование, что требует от разработчика наличия высокоуровневой подготовки.

4) Модель на основе прототипа – на каждом этапе разработки программного продукта создается часть будущей системы, по которой заказчик дает обратную связь. Позволяет работать с неясными, либо постоянно меняющимися требованиями к программному обеспечению. Недостатком является активное вовлечение заказчика в процесс разработки.

5) Инкрементная модель позволяет разрабатывать программный продукт по частям. На каждом новом этапе разработки к программному продукту добавляется небольшой функциональный блок, который сразу же тестируется. Данная модель позволяет получить рабочий вариант приложения за сравнительно небольшой срок, позволяет подстраиваться под новые требования. Одним из немногих недостатков модели является необходимость согласования архитектуры каждого последующего инкремента.

Процедура выбора модели ЖЦ Института SQI базируется на применении четырех таблиц вопросов (таблицы 3.1 – 3.4), соответствующих предложенной данным институтом классификации проектов [16]. По каждому из вопросов необходимо выделить ответы, соответствующие конкретному проекту, и выбрать модель с наибольшим количеством отмеченных ответов.

Таблица 3.1 – Выбор модели ЖЦ на основе характеристик требований

Номер критерия	Критерии категории требований	Ответ на критерий
1	Являются ли требования к проекту легко определяемыми и реализуемыми?	Да
2	Могут ли требования быть сформулированы в начале ЖЦ?	Да
3	Часто ли будут изменяться требования на протяжении ЖЦ?	Нет
4	Нужно ли демонстрировать требования с целью их определения?	Да
5	Требуется ли проверка концепции программного средства или системы?	Нет
6	Будут ли требования изменяться или уточняться с ростом сложности системы (программного средства) в ЖЦ?	Нет
7	Нужно ли реализовать основные требования на ранних этапах разработки?	Нет

Таблица 3.2 – Выбор модели ЖЦ на основе характеристик команды разработчиков

Номер критерия	Критерии категории команды разработчиков проекта	Ответ на критерий
1	Являются ли проблемы предметной области проекта новыми для большинства разработчиков?	Нет
2	Являются ли инструментальные средства, используемые в проекте, новыми для большинства разработчиков?	Нет
3.	Изменяются ли роли участников проекта на протяжении ЖЦ?	Нет
4	Является ли структура процесса разработки более значимой для разработчиков, чем гибкость?	Да
5	Важна ли легкость распределения человеческих ресурсов проекта?	Да
6	Приемлет ли команда разработчиков оценки, проверки, стадии разработки?	Да

Таблица 3.3 – Выбор модели ЖЦ на основе характеристик коллектива пользователей

Номер критерия	Критерии категории коллектива пользователей	Ответ на критерий
1	Будет ли присутствие пользователей ограничено в ЖЦ разработки?	Нет
2	Будут ли пользователи оценивать текущее состояние программного продукта (системы) в процессе разработки?	Нет
3	Будут ли пользователи вовлечены во все фазы ЖЦ разработки?	Нет
4	Будет ли заказчик отслеживать ход выполнения проекта?	Нет

Таблица 3.4 – Выбор модели ЖЦ на основе характеристик типа проектов и рисков

Номер критерия	Критерии категории коллектива пользователей	Ответ на критерий
1	Разрабатывается ли в проекте продукт нового для организации направления?	Нет
2	Будет ли проект являться расширением существующей системы?	Да
3	Будет ли проект крупно- или среднемасштабным?	Нет
4	Ожидается ли длительная эксплуатация продукта?	Да
5	Необходим ли высокий уровень надежности продукта проекта?	Нет
6	Предполагается ли эволюция продукта проекта в течение ЖЦ?	Нет
7	Велика ли вероятность изменения системы (продукта) на этапе сопровождения?	Да
8	Является ли график сжатым?	Нет
9	Предполагается ли повторное использование компонентов?	Да
10	Являются ли достаточными ресурсы (время, деньги, инструменты, персонал)?	Нет

На основании результатов, представленных в таблицах 3.1 – 3.4 можно сделать вывод о том, что для спроектированного веб-сервиса более всего подходит каскадная модель жизненного цикла.

### 3.2 Выбор технологии и средств разработки веб-сервиса

Беря во внимание то, что в будущем имеется возможность внедрения разрабатываемый веб-сервис в информационную систему (ИС) ЮГУ, в частности как модуль системы ЮГУ «ELIOS», веб-сервис разрабатывается на клиент-серверной архитектуре, повторяя архитектуру ИС «ELIOS».

Шаблон проектирования – повторяемая архитектурная конструкция, представляющая собой решение проблемы проектирования в рамках некоторого часто возникающего контекста.

В разрабатываемом приложении используется шаблон проектирования MVC. Структурный паттерн MVC (рисунок 3.1) – Model-View-Controller – позволяет разделять код веб-приложения на следующие компоненты:

Model – модель, содержит данные, которые обрабатываются в приложении, также может содержать ограничения на данные;

View – представление, графический интерфейс пользователя, как правило, веб-интерфейс;

Controller – контроллер, обрабатывает данные, передает результат обработки представлению для отображения данных конечному пользователю.

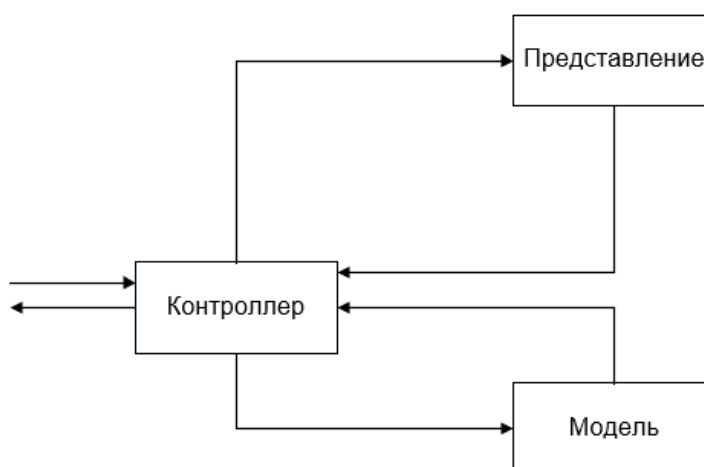


Рисунок 3.1 – Структура MVC

Контроллер получает запрос от браузера пользователя. Для получения данных контроллер вызывает методы модели. Полученные от модели данные передаются методам представления для отображения конечному пользователю.

Использование MVC позволяет разделять код на независимые модули, благодаря чему возможна модернизация одного из модулей, не затрагивающая остальные части приложения.

Используемая методология разработки: ООП – объектно-ориентированное программирование – согласно данной методологии, веб-сервис разрабатывается в виде набора взаимодействующих объектов. Объекты создаются на основе классов – специальных типов данных, содержащих поля, определяющие состояние объекта и методы – поведение объекта.

Классы создаются разработчиками на основе сущностей предметной области. Класс может быть построен на основе уже существующего с помощью наследования. Класс-наследник обладает всеми методами и полями родительского и, при необходимости, дополняет функциональность родительского класса собственными методами и полями. Наследование позволяет избежать дублирования кода.

Еще одной особенностью объектно-ориентированного подхода является полиморфизм – применение одного интерфейса для взаимодействия с объектами различного типа. Использование полиморфизма позволяет избежать создания однотипных методов для работы с различными объектами, имеющими общее основание.

Для построения структуры страниц используется язык гипертекстовой разметки HTML [17]. Структура страницы строится с помощью тегов, которые браузер интерпретирует как элементы страницы. Теги, в большинстве случаев, бывают открывающие и закрывающие. Между открывающим и закрывающим тегами располагается контент – содержимое элемента. Для настройки элементов для тегов указываются атрибуты. Атрибуты бывают общими – используются для всех тегов, и частными – используются только для определенных тегов.

Для оформления страницы используется язык CSS [18, 19]. CSS представляет собой набор стилей, определяющих поведение определенного элемента или группы элементов. Стили могут быть внутренние (наивысший приоритет), внешние (указанные в мета теге head – следующий уровень приоритета, вынесенные в отдельный файл – низший приоритет). CSS-стиль состоит из селектора (или набора селекторов) и списка свойств и их значений, указанных через двоеточие. Стили позволяют задать цвет, размер, положение, толщину границы, отступы и т.д.

Если разрабатываемое веб-приложение предполагает работу с базой данных, необходимо использовать серверный язык программирования, скрипты которого будут генерировать страницы сайта, заполняя их информацией из базы, обрабатывать данные, передаваемые пользователем, изменять содержимое базы данных. Для работы с базой данных выбран язык PHP [20-22]. PHP поддерживает разработку как в рамках процедурного подхода, который целесообразно использовать при разработке небольших приложений, так и в рамках объектно-ориентированного подхода, который целесообразен при разработке средних и крупных приложений, а также приложений, предполагающих расширение функциональности.

При разработке приложений, как правило, не используют исключительно средства языка программирования. Для ускорения процесса разработки используют различные библиотеки и фреймворки. В данной работе использован фреймворк Laravel [23, 24]. Laravel – свободно распространяемый фреймворк, позволяющий строить веб-приложения по структуре MVC, подключать различные компоненты, взаимодействовать с базой данных, оптимизировать загрузку подключаемых файлов.

Для написания динамического фронтэнда используется фреймворк Vue.js [25]. Vue.js является легким и быстрым не только в освоении, но и реализации интерфейса, а также является «реактивным» фреймворком, что позволяет работать с пользовательским интерфейсом, гибко изменяя компоненты и



позволяя расширить функционал веб-сервиса. Помимо этого, в Laravel есть встроенная возможность использовать Vue.js [24].

Для написания кода использовался текстовый редактор PhpShtorm.

### 3.3 Разработка веб-сервиса для автоматизации составления библиографического описания научно-методической литературы

Разработка веб-сервиса начинается с установки Laravel выбранной версии в каталог проекта. Для установки компонентов Laravel используется менеджер пакетов для PHP – Composer. Работа с Composer осуществляется через командную строку. Для создания Laravel-проекта необходимо перейти в каталог на веб-сервере, предназначенный для файлов веб-приложений и выполнить следующую команду: `Composer create-project --prefer-dist Laravel/Laravel Links 5.4.36`

В результате выполнения указанной команды создается каталог с названием Links (каталог проекта) и в него устанавливаются компоненты Laravel. Выбранная версия Laravel: 5.4.36.

Laravel основан на структурном паттерне MVC. Согласно MVC, все запросы, поступающие на сайт, обрабатываются контроллерами. В зависимости от запроса, выбирается определенный контроллер, который обрабатывает внешний запрос, обращаясь к модели и/или представлению.

Выбор контроллера, который должен использоваться для обработки запроса, осуществляет маршрутизатор (route). Маршрутизатор представляет собой php-файл, в котором прописаны зависимости для url-адресов и соответствующих им контроллеров. В Laravel 5.4 маршрутизатор имеет название web.php, располагается в корне каталога routes. Фрагмент web.php приведен в листинге 3.1:

### Листинг 3.1 – Фрагмент web.php

```
Route::get('/', 'PagesController@index');
Route::get('/about', 'PagesController@about');
Route::get('/admin', 'PagesController@admin');
Route::get('/contact', 'PagesController@contact');
Route::get('/newauthor', function() {
    return view('pages.newauthor');
});
Route::post('/saveauthor', 'PagesController@saveauthor');
Route::get('/newsources', 'SourcesController@newsources');
Route::get('/deletesource/{id}', 'SourcesController@deletesource');
Route::get('/editsource/{id}', 'SourcesController@editsource');
Route::post('/editsave', 'SourcesController@editsave');
Route::post('/save', 'SourcesController@save');
Route::resource('sources', 'SourcesController');
```

Для установки соответствия между url-адресом и контроллером используется класс Route. В зависимости от типа запроса (get, post) для установки соответствия используются методы get или post. Метод resource позволяет сгенерировать базовые зависимости для контроллера без необходимости указания маршрута с использованием get или post для каждого метода контроллера.

Строка «'/about', 'PagesController@about'», указанная в качестве списка параметров методов get (а также post) означает, что при переходе по url-адресу «/about» будет выполнен метод about контроллера PagesController. Для resource указывается только имя контроллера, так как маршруты будут автоматически сгенерированы для всех методов контроллера.

Контроллеры проекта располагаются в каталоге app/http/controllers. Все создаваемые контроллеры должны являться наследниками от класса Controller, создаваемого при установке Laravel, для доступа ко всем возможностям, предоставляемым Laravel контроллерам. Количество контроллеров, а также создаваемые в них методы, определяются решаемой задачей. Существует

соглашение по именованию классов-контроллеров. Согласно которому в конце имени каждого контроллера должно указываться слово «Controller».

Для создания контроллеров целесообразно использовать командный интерфейс artisan, включенный в Laravel. Команда для создания контроллера PagesController с помощью artisan выглядит следующим образом: `php artisan make:controller PagesController`

Запуск команд artisan осуществляется через командную строку.

Контроллер содержит методы, которые вызываются при обращении к url-адресу, связанному с данным методом. Метод `about` контроллера `PagesController`, открывающий главную страницу веб-приложения, приведен в листинге 3.2.

Листинг 3.2 – Метод, открывающий страницу с описанием работы сайта

```
Public function about() {  
    $title = 'Как работает сервис';  
    return view('pages.about')->with('title', $title);  
}
```

Метод `about` вызывает представление `about`, расположенное в каталоге `resources/views/pages` и передает параметр `title`, содержащий заголовок страницы.

Источником данных, передаваемых контроллерами представлениям, являются модели. Модели располагаются в корне каталога `app`. Модель представляет собой класс, предназначенный для взаимодействия с соответствующей таблицей базы данных. Для создания моделей используется `artisan`: `php artisan make:model Source -m`

`Source` – имя модели, ключ `-m` позволяет вместе с классом модели создать миграцию – `php`-скрипт, позволяющий создать таблицу в базе данных. Использование миграций позволяет работать с базой данных напрямую из редактора кода, без обращения к клиенту СУБД. Каждая миграция вносит определенные изменения в структуру текущей базы данных. В случае совершения ошибки, миграция может быть отменена. Миграция содержит

методы `up` и `down`, позволяющие выполнить или откатить миграцию соответственно. Методы `up` и `down` для миграции, создающей таблицу `Sources`, приведены в листинге 3.3.

### Листинг 3.3 – Миграция для создания таблицы `sources`

```
public function up() {
    Schema::create('sources', function (Blueprint $table) {
        $table->increments('id');
        $table->integer('source_type_id')->unsigned();
        $table->foreign('source_type_id')->references('id')-
        >on('source_types')->onDelete('cascade');;
        $table->timestamps();
    });
}

public function down() {
    Schema::dropIfExists('sources');
}
```

В методе `create` указываются поля таблицы и их типы данных, название таблицы, включаются/отключаются временные метки и т.д. Для того, чтобы указать связь с другой таблицей, необходимо в классе модели создать метод, указывающий на связываемую таблицу, а также кратность связи. Методы, предназначенные для установления связи «один-ко-многим» между таблицами `sourcetypes` и `sources`, приведены в листинге 3.4.

### Листинг 3.4 – Методы для установления связи между таблицами `sourcetypes` и `sources`

```
// расположен в модели SourceType
public function source() {
    return $this->hasMany('App\Source');
}

// расположен в модели Source
public function source_type() {
```

```
return $this -> belongsTo('App\SourceType');
}
```

Для запуска миграций используется artisan. Прежде чем запускать миграцию, необходимо настроить конфигурационный файл .env, расположенный в корне проекта. Для работы с базой необходимо указать следующие данные:

- DB\_CONNECTION – поставщик драйвера для работы с СУБД, в данном случае, mysql;
- DB\_HOST – адрес хоста, на котором расположен сайт;
- DB\_PORT – номер порта для СУБД, значение для MySQL: 3306;
- DB\_DATABASE – имя базы данных, в данном случае: links;
- DB\_USERNAME – имя пользователя для подключения к базе данных;
- DB\_PASSWORD – пароль для пользователя СУБД.

Команда artisan, предназначенная для запуска миграций, выглядит следующим образом: `php artisan migrate`

В результате выполнения команды будут применены все миграции, созданные с момента последнего обновления базы данных.

Рассмотрим реализацию функции сохранения списка источников в файл, функции поиска данных, функции оформления источника.

Код, отвечающий за работу функции поиска, приведен в листинге 3.5.

### Листинг 3.5 – Поиск данных

```
public function search($search) {
    $sources = DB::table('sources')
    -
    >selectRaw('title * ? || author * ? || publishing * ? || source_type * ?', [$search])
    ->get();
    return view('sources.index')->with('sources', $sources);
}
```

```
}
```

Поисковой запрос – данные из текстового поля, либо из выпадающего списка – передается в функцию в качестве параметра \$search. Если выбраны несколько параметров для поиска, то функция вызывается для каждого из них отдельно.

В листинге 3.6 приведен фрагмент функции getDataList(), отвечающей за оформление пункта списка источников, по шаблону.

### Листинг 3.6 – Реализация шаблонов оформления

```
switch (type) {
case "Книжное издание":
if (moreThan3) {
data = title + " : " + titleknowledge + " / ";
for (let i = 0; i < 3; i++) {
if (i == authors.length - 1) {
data = data + authors[i].initials + " " + authors[i].surname;
} else {
data = data + authors[i].initials + " " + authors[i].surname + ", ";
}
}
data = data + " [и др.]";
} else {
data = authors[0].surname + ", " + authors[0].initials + " " + title + "
: " + titleknowledge + " / ";
for (let i = 0; i < authors.length; i++) {
if (i == authors.length - 1) {
data = data + authors[i].initials + " " + authors[i].surname;
} else {
data = data + authors[i].initials + " " + authors[i].surname + ", ";
}
}
}
data = data + ". - " + place + " : " + publishing + ", " + year + ". - "
+ pages + " - ISBN " + isbn + " - Текст: " + access + ".";
break;
```

В зависимости от типа источника, формирование библиографической записи будет меняться, поэтому было решено использовать конструкцию switch/case для выполнения различных участков кода. Здесь же (листинг 3.6) указаны проверки на количество авторов у источника, в зависимости от их количества форма записи тоже меняется.

Для вывода списка литературы с сохранением всех добавленных библиографических записей источников без перезагрузки страницы необходимо использовать JavaScript, применяя фреймворк Vue.js так как он является частью пакета Laravel. Установить его можно с помощью команды: `npm install`. Для запуска и для возможности в дальнейшем, когда что-либо в коде изменится, автоматически скомпилировать новые файлы, используется команда: `npm run watch`.

Код функции сохранения списка литературы в файл приведен в листинге 3.7.

### Листинг 3.7 – Сохранение в файл

```
exportHTML() {  
  // подготавливаем  
  if (this.transliterate) {  
    this.transliterateFunction();  
  } else {  
    let i = 1;  
    this.dataList.forEach(el => {  
      this.word = this.word + i + ". " + el + "<br><br>";  
      i++;  
    });  
  }  
  // сохраняем  
  var vm = this, word = `
```

```
8'><title>Export      HTML      to      Word      Document      with  
JavaScript</title></head><body>${vm.word}</body></html>`;  
// очищаем (чтобы в следующий раз заново сформировать)  
this.word = "";  
var source = 'data:application/vnd.ms-word;charset=utf-8,' +  
encodeURIComponent(word);  
var fileDownload = document.createElement("a");  
document.body.appendChild(fileDownload);  
fileDownload.href = source;  
fileDownload.download = 'links.doc';  
fileDownload.click();  
document.body.removeChild(fileDownload);  
}
```



### 3.4 Описание интерфейса

При открытии веб-сервиса по формированию списка источников, пользователь попадает на главную страницу (рисунок 3.2).

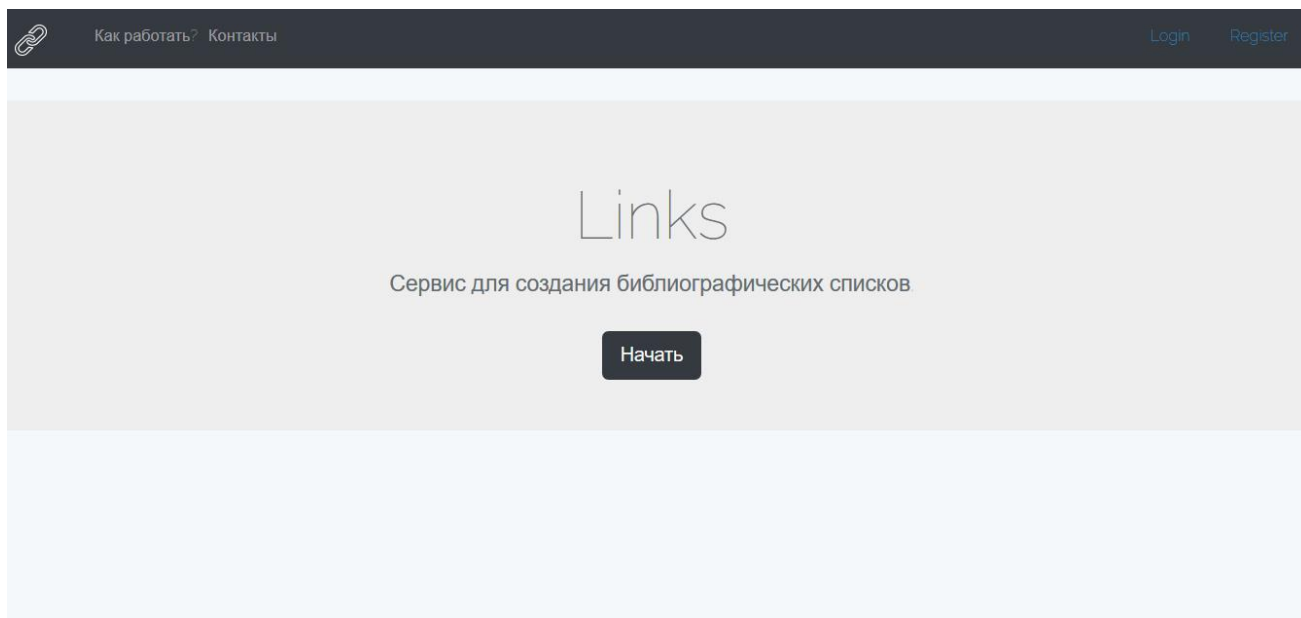


Рисунок 3.2 – Главная страница приложения

Главная страница содержит навигационную панель, краткое описание сервиса и кнопку «Начать», при нажатии на которую пользователь перемещается на страницу со списком источников, находящихся в базе данных на текущий момент (рисунок 3.3).

## Источники

Выберите источник из списка:

--Выбрать все--

Поиск

Поиск

Очистить

Средства моделирования (CASE) и поддержки всех стадий разработки ПО	Нет автора	Добавить
Документация Vue.js	Нет автора	Добавить
Технология разработки программного обеспечения	267 Бахтизин БГУИР	Добавить

**Список**

1. Гамма, Э. Приемы объектно-ориентированного проектирования Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон. - СПб.: Питер, 2009. - 366 с. - ISBN - Текст: непосредственный
2. Методология проектирования программных средств : сайт. - URL: <https://analytics.infozone.pro/methodology-design-software/> (дата обращения 08.03.2020). - Текст : электронный
3. Документация Vue.js : сайт. - URL: <https://ru.vuejs.org> (дата обращения 11.06.2020). - Текст : электронный
4. Бахтизин, В. В. Технология разработки программного обеспечения : учеб. пособие / В. В. Бахтизин. - Минск : БГУИР, 2010. - 267 с. - ISBN 978-985-488-512-4 - Текст: непосредственный

☐ Транслитерация

Сохранить в файл

Нет нужного источника?

Добавьте его

Рисунок 3.3 – Страница с источниками

В верхней части страницы расположена навигационная панель, содержащая ссылки на форму обратной связи и инструкцию по работе с сервисом.

Далее следует форма поиска по издательству, автору, типу источника, а также произвольным ключевым словам. Перечень источников отображается в таблице в центре страницы. Для добавления источника к собственному списку необходимо нажать на кнопку «Добавить» в строке с соответствующим источником.

Если необходимого источника нет в таблице, пользователь может добавить его в базу. Для этого необходимо нажать кнопку в нижней части окна. После этого пользователь перейдет на новую страницу, и появится возможность

выбрать тип источника, в зависимости от которого форма заполнения будет изменяться. Форма для добавления книги приведена на рисунке 3.4.

Книжное издание

Книжное издание

Авторы

Создать нового

Добавить выбранного

Р. Джонсон

- Э. Гамма
- Р. Хелм
- Р. Джонсон

Название (Пример: Труды по истории изобразительного искусства)

Приемы объектно-ориентированного проектирования. Паттерны проектирования

Дополнительные сведения (Пример: учебное пособие)

Место публикации (Пример: Санкт-Петербург)

СПб

Издательство (Пример: БАН)

Питер

Год (Пример: 2017)

2009

Страницы (Пример: 216)

366

ISBN (Пример: 978-5-336-00204-1)

Средство доступа (Пример: непосредственный)

непосредственный

Создать

Рисунок 3.4 – Форма для добавления книги

В полях формы расположены подсказки с примерами для заполнения. Все поля, кроме сведений, относящихся к заглавию, международного номера, вида содержания и средства доступа являются обязательными для заполнения.

Страница с описанием работы с сервисом приведена на рисунке 3.5.

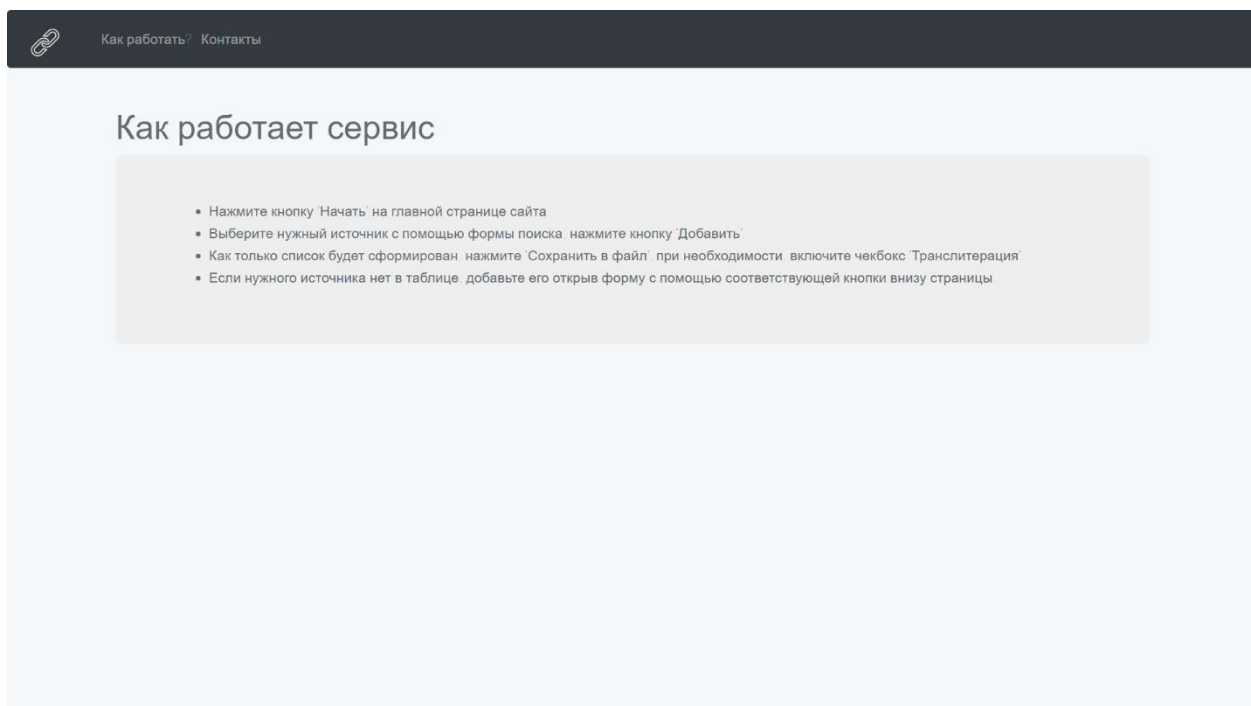


Рисунок 3.5 – Инструкция по работе с сервисом

На рисунке 3.6 показан личный кабинет администратора сервиса. Как видно из рисунка 3.6, помимо поиска и просмотра данных, администратору доступны удаление и редактирование данных источников для сохранения актуальности и достоверности данных в базе.

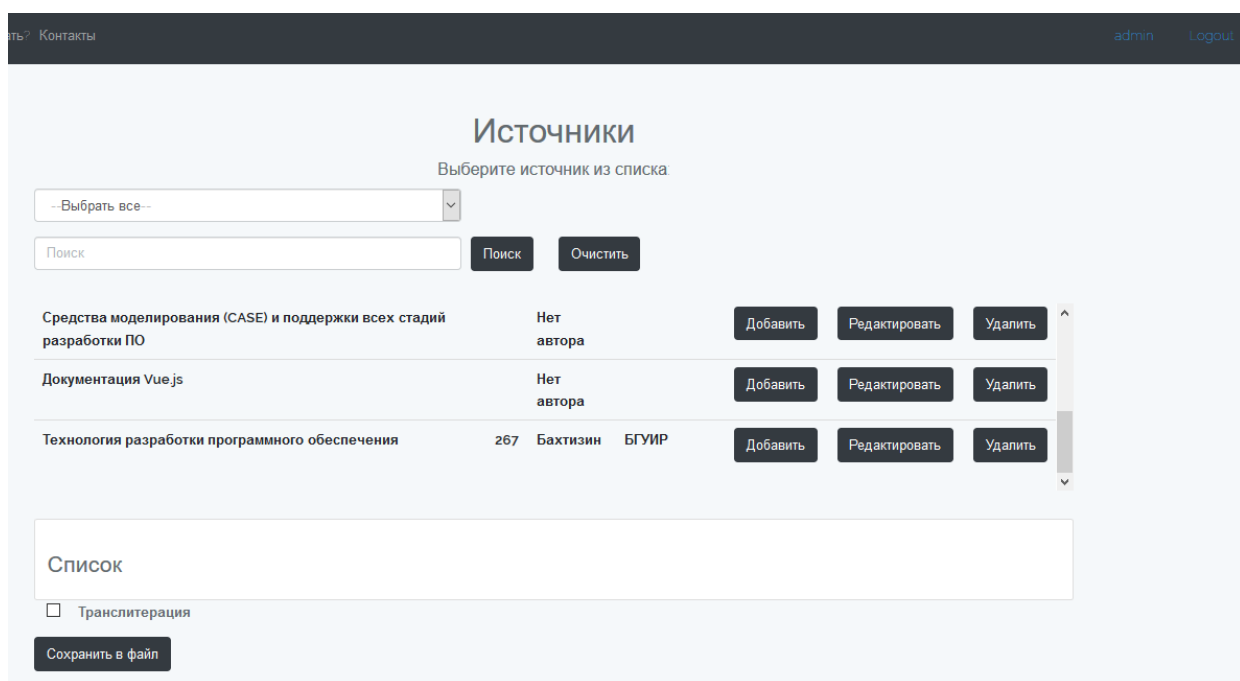


Рисунок 3.6 – Личный кабинет администратора веб-сервиса

Кнопка «Сохранить в файл» позволяет создать текстовый файл .doc с созданным списком источников и, при необходимости, провести транслитерацию.

При сохранении файла со списком, пользователь видит следующее окно (рисунок 3.7).

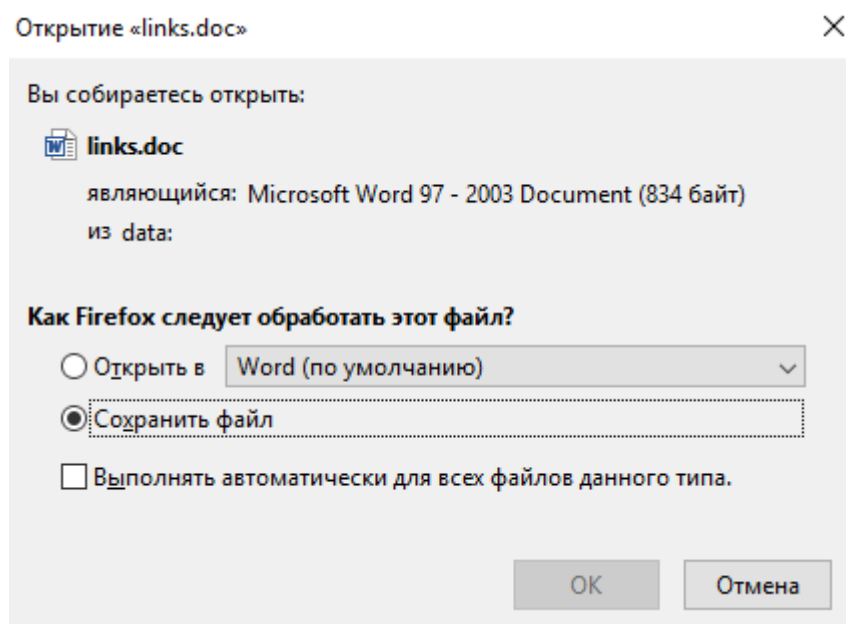


Рисунок 3.7 – Сохранение файла

## ЗАКЛЮЧЕНИЕ

Формирование списка источников для научной/студенческой работы является актуальной и востребованной задачей. Для повышения эффективности и скорости ее решения целесообразно использовать веб-сервис, позволяющий автоматизировать процесс составления списка источников.

В ходе выполнения выпускной квалификационной работы решены следующие задачи:

- 1) Был произведен анализ предметной области.
- 2) Составлен обзор программных продуктов и сервисов для построения списка источников.
- 3) Разработан комплекс моделей предметной области, необходимых для создания технического задания. Разработано техническое задание.
- 4) Спроектирован веб-сервис для составления списка источников на основе объектно-ориентированного подхода.
- 5) Спроектирована и разработана реляционная база данных.
- 6) Разработан веб-сервис с использованием фреймворка Laravel, паттерна MVC.

В данной выпускной квалификационной работе проведена разработка веб-сервиса для автоматизации составления списка источников. Для разработки использованы следующие инструменты:

- язык разметки HTML;
- каскадные таблицы стилей CSS, фреймворк Bootstrap 4;
- язык программирования PHP;
- фреймворк Laravel, версия 5.4;
- структурный паттерн MVC;
- фреймворк Vue.js;
- СУБД MySQL.

Реализован основной функционал, которого достаточно для продуктивного составления списка источников. К основному функционалу относится возможность добавления, поиска, составления, изменения и удаления источника. Так же есть возможность перевода списка и сохранения его во внешний файл. Для различного взаимодействия с источниками не нужно перемещаться по большому количеству страниц. Навигация по различным функциям сайта происходит в верхнем и нижнем меню страницы. Это позволяет максимально эффективно использовать место на экране и концентрировать внимание пользователя на списке источников. Если пользователь сервиса все же затрудняется в выборе действия, хочет узнать какой функционал может предоставить сервис, на этот случай предусмотрен раздел справки.

Среднее время составления библиографического описания, используя веб-сервис составляет от 5 до 10 минут. При осуществлении этого процесса вручную, среднее время составляло 20 до 40 минут. Исходя из этого, можно сделать вывод что разработанный веб-сервис позволяет сократить время на составление списка источников до 4 раз.

В случае дальнейшего развития веб-сервиса для составления списка источников возможно добавление других типов источников, без существенного изменения принципов взаимодействия с сервером, и улучшением процесса транслитерации.

Сайт работает в тестовом режиме и размещен на локальном сервере.

## СОКРАЩЕНИЯ, ОБОЗНАЧЕНИЯ, ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

CSS – Cascading Style Sheets, каскадные таблицы стилей;

DOI – discrete object identifier, идентификатор цифрового объекта;

HTML – HyperText Markup Language, язык гипертекстовой разметки;

ISBN – International Standard Book Number, уникальный номер книжного издания;

MVC – Model-View-Controller, модель-представление-контроллер;

MS – Microsoft;

SQL – structured query language, язык структурированных запросов;

UML – Unified Modelling Language, унифицированный язык моделирования;

БД – база данных;

БЗ – библиографическая запись;

ВКР – выпускная квалификационная работа;

ГОСТ – государственный стандарт;

ЖЦ – жизненный цикл;

ООП – объектно-ориентированный подход;

ПО – программное обеспечение;

СУБД – система управления базами данных;

ТЗ – техническое задание;

ЮГУ – Югорский государственный университет.



## СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Зиглер, К. Методы проектирования программных средств / К. Зиглер. – М. : Мир, 1985. – Текст : непосредственный.
2. Приемы объектно-ориентированного проектирования. Паттерны проектирования / Э. Гамма, Р. Хелм, Р. Джонсон [и др.]. – СПб.: Питер, 2009. – 366 с. – Текст : непосредственный.
3. Методология проектирования программных средств : сайт. – URL: <https://analytics.infozone.pro/methodology-design-software/> (дата обращения: 08.03.2020). – Текст : электронный.
4. Проектирование информационных систем : сайт. – URL: [www.intuit.ru/studies/courses/2195/55/lecture/1640](http://www.intuit.ru/studies/courses/2195/55/lecture/1640) (дата обращения: 08.03.2020). – Текст : электронный.
5. Технология разработки программного обеспечения : конспект лекций / сост. Г. Ф. Довбуш; Петербургский государственный университет путей сообщения. – Санкт-Петербург: Изд-во Петербургского государственного университета путей сообщения, 2010. – 131 с. – Текст : непосредственный.
6. Орлов, С. А. Технологии разработки программного обеспечения : Учебник / С. А. Орлов. – СПб. : Питер, 2002. – 464 с. – ISBN 5-94723-145-X. – Текст : непосредственный.
7. Крачтен Ф. Введение в Rational Unified Process. / Пер. с англ. – 2-е изд. – М.: Вильямс, 2002.
8. Глухова, Л. А. Информационное моделирование с помощью CASE-средства ERwin 3.0 : учеб. пособие по курсу «Технология проектирования программ» для студ. спец. Т.10.02.00 «Программное обеспечение информационных технологий» / Л. А. Глухова, В. В. Бахтизин. – Минск : БГУИР, 1999.
9. Средства моделирования (CASE) и поддержки всех стадий разработки ПО : сайт. – URL: <http://www.interface.ru/home.asp?artId=99> (дата обращения: 15.04.2020). – Текст : электронный.

10. Бистерфельд, О.А. Методология функционального моделирования IDEF0 : учебно-методическое пособие / О.А. Бистерфельд ; Ряз. гос. ун-т им. С.А. Есенина. – Рязань, 2008. – 48 с.
11. Буч, Г. Язык UML. Руководство пользователя. 2-е изд.: Пер. с англ. Мухин Н. / Г. Буч, Д. Рамбо, И. Якобсон. – Москва : ДМК Пресс, 2006. – 496 с. – ISBN 5-94074-334-X. – Текст : непосредственный.
12. Леоненков, А.В. Самоучитель UML : 2-е изд., перераб. и доп. / А.В. Леоненков. – СПб.: БХВ-Петербург, 2004. – 432с. – ISBN 5-94157-342-1. – Текст : непосредственный.
13. Unified Modeling Language (UML) description : официальный сайт. – URL: <http://www.uml-diagrams.org> (дата обращения: 01.05.2020). – Текст : электронный.
14. Конолли, Т. Базы данных. Проектирование, реализация и сопровождение. Теория и практика. / Т. Конолли, К. Бегг – Москва : Вильямс, 2017. – 1440 с. – Текст : непосредственный.
15. Верников, Г. Основы методологии IDEF1X. / Г. Верников. – Текст : электронный // Корпоративный менеджмент : [сайт]. – 1999. – URL: <https://www.cfin.ru/vernikov/idef/idef1x.shtml> (дата обращения: 06.05.2020).
16. Бахтизин, В. В. Технология разработки программного обеспечения : учеб. пособие / В. В. Бахтизин, Л. А. Глухова. – Минск : БГУИР, 2010. – 267 с. – ISBN 978-985-488-512-4 – Текст : непосредственный.
17. Дунаев, В.В. HTML, скрипты и стили / В.В. Дунаев. – БХВ-Петербург – М.: 2017. – 527 с. – Текст : непосредственный.
18. Квинт, И. Создаем сайты с помощью HTML, XHTML и CSS / И. Квинт. – М.: Питер, 2014. – 448 с. – Текст : непосредственный.
19. Фримен, Э. Изучаем HTML, XHTML и CSS / Э. Фримен. – М.: Питер, 2016. – 720 с. – Текст : непосредственный.
20. Веллинг, Л. Разработка веб-приложений с помощью PHP+MySQL. / Л. Веллинг – Москва : Вильямс, 2016. – 848 с. – Текст : непосредственный.

21. Кузнецов, М. PHP 5. Практика создания Web-сайтов / М. Кузнецов. – БХВ-Петербург – М.: 2017. – 960 с. – Текст : непосредственный.
22. Кузнецов, М. Самоучитель PHP 5/ М. Кузнецов. – БХВ-Петербург – М., 2017. – 560 с. – Текст : непосредственный.
23. Документация 5.x | Laravel по-русски : официальный сайт. – URL: <https://laravel.ru/docs/v5> (дата обращения 29.05.2020). – Текст : электронный.
24. Дронов, В.А. Laravel. Быстрая разработка современных динамических Web-сайтов на PHP, MySQL, HTML и CSS / Дронов В.А. – М.: БХВ-Петербург, 2018. – 160 с. – Текст : непосредственный.
25. Документация Vue.js : сайт. – URL: <https://ru.vuejs.org> (дата обращения 11.06.2020). – Текст : электронный.

ПРИЛОЖЕНИЯ  
Приложение А  
Техническое задание

УТВЕРЖДАЮ

Научный руководитель

 /А.В. Кутышкин /

« 19 » мая 2020 г.

ТЕХНИЧЕСКОЕ ЗАДАНИЕ  
«РАЗРАБОТКА ВЕБ-СЕРВИСА ДЛЯ АВТОМАТИЗАЦИИ СОСТАВЛЕНИЯ  
БИБЛИОГРАФИЧЕСКОГО ОПИСАНИЯ НАУЧНО-МЕТОДИЧЕСКОЙ ЛИТЕРАТУРЫ»

Ханты-Мансийск 2020 г.

#### Перечень принятых сокращений

ГОСТ – государственный стандарт

ФГБОУ – Федеральное государственное образовательное учреждение высшего образования

ЮГУ – Югорский государственный университет

БД – база данных

СУБД – система управления базами данных

ПК – персональный компьютер

ЕСПД – единая система программной документации

ВКР – выпускная квалификационная работа

## **1. ОБЩИЕ СВЕДЕНИЯ**

Настоящее Техническое задание разработано в соответствии с ГОСТ 34.602-89 «Техническое задание на создание автоматизированной системы».

Настоящее Техническое задание определяет общие требования и порядок разработки «Веб-сервис для автоматизации составления библиографического описания научно-методической литературы» (далее – Веб-сервис) в целом.

### **1.1 Наименование Веб-сервиса**

#### **1.1.1 Полное наименование Веб-сервиса**

Полное наименование Веб-сервиса: «Веб-сервис для автоматизации составления библиографического описания научно-методической литературы».

#### **1.1.2 Краткое наименование**

Краткое наименование веб-сервиса: «Links».

### **1.2 Основание для проведения работ**

Веб-сервис разрабатывается на основании:

Приказа ФГБОУ ЮГУ о назначении тем выпускных квалификационных работ обучающихся по направлению 09.03.04 «Программная инженерия».

### **1.3 Наименование Заказчика**

Заказчиком разрабатываемого Веб-сервиса является Научная библиотека Югорского государственного университета (ЮГУ) в лице директора библиотеки Кузнецовой Ирины Егоровны.

Юридический адрес: 628012, г. Ханты-Мансийск, ул. Чехова, д. 18

Телефон: +7 3467 35-77-61

E-mail: library@ugrasu.ru

## **2. НАЗНАЧЕНИЕ И ЦЕЛИ СОЗДАНИЯ ВЕБ-СЕРВИСА**

### **2.1 Назначение Веб-сервиса**

Назначение веб-сервиса состоит в автоматическом создании правильного библиографического описания без изучения дополнительной информации со стороны пользователя, воспользовавшимся данным веб-сервисом. Сведения о использованном источнике, введенные пользователем, собираются, обрабатываются и сохраняются в базе данных. На основе этой информации выводится корректная и пронумерованная библиографическая запись. Регулярная актуализация базы данных библиографических записей и ее пополнение обеспечивает организацию и реализацию поиска по сведениям источника для включения его пользователем в формируемый текущий список использованной литературы.

### **2.2 Цели создания Веб-сервиса**

Автоматизация составления списка литературы для студентов и преподавателей ЮГУ.

### **3. ХАРАКТЕРИСТИКА ОБЪЕКТОВ АВТОМАТИЗАЦИИ**

#### **3.1 Объект автоматизации**

Объектами автоматизации являются процессы составления библиографической записи и библиографического описания научно-методической литературы. Данные процессы осуществляются в соответствии с "Методическими рекомендациями к составлению списков использованной литературы в соответствии с ГОСТ 7.0.100-2018 "Библиографическая запись. Библиографическое описание", разработанное для Югорского государственного университета.



## **4. ТРЕБОВАНИЯ К ВЕБ-СЕРВИСУ**

### **4.1 Требования к функциональным характеристикам**

#### **4.1.1 Состав функций**

Веб-сервис должен обеспечивать следующие функциональные возможности:

- просмотр источников, хранящихся в базе данных;
- выполнение поиска, по ключевым словам, среди источников в базе данных;
- создание нового источника;
- оформление источника согласно шаблону;
- сохранение составленного источника в базе данных;
- транслитерация информации об источнике;
- формирование собственного списка из источников, хранящихся в базе данных;
- возможность сохранения собственного списка из источников в формате документа Microsoft Office Word;
- возможность редактирования и удаления источника из БД.

С помощью графического интерфейса веб-сервис должен предоставлять возможности пользователям:

1. Гость:
  - осуществлять навигацию по сайту (переход между страницами);
  - добавлять необходимые составителю источники в составляемый список;
  - выбирать необходимый тип источника и количество авторов у него для добавления его в составляемый список;
  - выполнять поиск источников;
  - вводить необходимый перечень данных по каждому источнику;
  - выполнять изменения в списке в процессе его составления или редактирования.
2. Администратор:
  - авторизоваться в веб-сервисе;
  - выполнять поиск источников;
  - редактировать источники из базы данных;
  - удалять источники из базы данных.

#### **4.1.2 Исходные данные**

- Методические рекомендации к составлению списков использованной литературы в соответствии с ГОСТ Р 7.0.100-2018 «Библиографическая запись. Библиографическое описание».
- ГОСТ Р 7.0.100-2018 «Библиографическая запись. Библиографическое описание».
- Список реализуемых типов источников.

### **4.2 Требования к организации входных и выходных данных**

#### **4.2.1 Входные данные**

В качестве входных данных у гостя выступают:

- заголовок (сведения об авторе/авторах);
- основное заглавие;
- первые сведения об ответственности;
- первое место публикации, производства и/или распространения;
- имя издателя, производителя и/или распространителя;
- дата публикации, производства и/или распространения;
- специфическое обозначение материала и объем;
- основное заглавие серии/подсерии или многочастного монографического ресурса (если есть);
- международный стандартный номер (ISBN, DOI);
- номер выпуска серии/подсерии или многочастного монографического ресурса.
- сведения, относящиеся к заглавию;
- средство доступа.

В качестве входных данных администратора выступают:

- логин/пароль.

#### **4.2.2 Выходные данные**

Выходная информация представляется отображением составленного списка источников или в виде документа в формате \*.doc.

## **4.3 Требования к видам обеспечения**

### **4.3.1 Требования к информационному обеспечению**

#### **Требования к хранению данных**

Все данные сайта должны храниться в структурированном виде под управлением реляционной СУБД.

#### **Требования к языкам программирования**

Для реализации статических страниц и шаблонов должны использоваться языки HTML и CSS. Для реализации динамических страниц должен использоваться язык PHP.

### **4.3.2 Требования к программному обеспечению**

#### **Серверная часть**

- Операционная система – Windows 10
- Веб-сервер – OpenServer
- СУБД – MySQL
- PHP 7.4.2
- PHP-фреймворк: Laravel

#### **Клиентская часть**

Сайт должен быть доступен для полнофункционального просмотра с помощью следующих браузеров:

- Яндекс.Браузер;
- Google Chrome;
- Opera 6.0 и выше;
- Mozilla Firefox 1.0;
- Safari.

### **4.3.3 Требования к разделению доступа**

Все опубликованные разделы сайта должны открываться для доступа на чтение без аутентификации пользователя.

При попытке входа в закрытый раздел у пользователя, не прошедшего аутентификацию, должен быть запрошен логин и пароль.

После прохождения аутентификации система должна проверять полномочия пользователя на доступ к запрошенному разделу. Если доступ запрещен, пользователю должно быть выведено сообщение о невозможности доступа в закрытый раздел.

#### **4.3.4 Требования к лингвистическому обеспечению**

Сайт должен быть на русском языке.

### **4.4 Перечень нефункциональных требований**

#### **4.4.1 Требования к надежности**

- Обеспечить целостность хранимой информации.
- Создание входа в административную форму сервиса только по паролю и логину, заданному администратором.
- Обеспечение непрерывной работоспособности.
- Контроль входной и выходной информации.

#### **4.4.2 Требования к производительности**

Производительность Веб-сервиса напрямую зависит от производительности ПК, на котором запущен Веб-сервис.

#### **4.4.3 Требования к безопасности**

Требуется разграничение доступа. Пароль администратора хранится в зашифрованном виде.

#### **4.4.4 Требования к обработке ошибок**

Для надёжного функционирования веб-сервиса необходимо предусмотреть защиту от некорректных действий пользователя. Веб-сервис должен проверять корректность введенных пользователем данных. В случае возникновения ошибки выдавать соответствующее информационное сообщение.

#### **4.4.5 Требования к интерфейсу**

Интерфейс Веб-сервиса должен удовлетворять следующим требованиям:

- быть простым и интуитивно понятным, рассчитанным на пользователя (в плане компьютерной грамотности) средней квалификации;
- при построении интерфейса необходимо использовать типовые элементы (кнопки, текстовые поля, выпадающие списки, ссылки и т.д.) со стандартным назначением;
- содержать минимум управляющих кнопок и других элементов управления.

## **5. СОСТАВ И СОДЕРЖАНИЕ РАБОТ ПО СОЗДАНИЮ ВЕБ-СЕРВИСА**

### **5.1 Перечень стадий и этапов работ**

Создание Веб-сервиса подразумевает под собой выполнение этапов, описанных в таблице 5.1.

Таблица 5.1 – План-график выполнения работ

Название задачи	Начало	Окончание
1. Разработка технического задания	10.02.20	26.02.20
2. Проектирование веб-сервиса	01.03.20	18.03.20
3. Проектирование базы данных	19.03.20	26.03.20
4. Разработка базы данных	27.03.20	30.03.20
5. Написание SQL запросов к базе данных	01.04.19	16.04.20
6. Разработка серверной части веб-сервиса	17.04.20	30.04.20
7. Разработка клиентской части веб-сервиса	20.04.20	05.05.20
8. Тестирование веб-сервиса	05.05.20	20.05.20
9. Подготовка промежуточного отчета о результатах проекта	26.02.20	27.02.20
10. Подготовка промежуточного отчета о результатах проекта	25.03.19	28.03.20
11. Подготовка промежуточного отчета о результатах проекта	28.04.20	30.04.20
12. Подготовка отчета о результатах проекта	22.05.20	24.05.20
Завершение разработки Веб-сервиса «Links» – 24.05.20		

#### **5.1.1 Подготовка промежуточного отчета о результатах проекта**

Подготовка промежуточного отчета о результатах проекта происходит в конце каждого месяца вплоть до окончания работ над разработкой Веб-сервиса.

#### **5.1.2 Проектирование веб-сервиса**

На данном этапе происходит проектирование концептуальной модели предметной среды, логическое и физическое проектирование веб-сервиса.

### **5.1.3 Проектирование базы данных**

При проектировании базы данных, необходимо определить поля, таблицы и связи между ними содержащие информацию о:

- типах источников;
- авторах источника;
- сведениях об источнике;
- данных авторизации администратора.

### **5.1.4 Написание SQL запросов к базе данных**

SQL запросы к базе данных должны возвращать информацию описанную в пункте анализ базы данных.

### **5.1.5 Разработка серверной части Веб-сервиса**

При разработке серверной части Веб-сервиса необходимо обеспечить возможность:

- агрегирование данных, полученных при помощи SQL запросов;
- формирование библиографического описания источников на основе полученных данных;
- выгрузки документа библиографического описания в формате .docx

### **5.1.6 Разработка клиентской части Веб-сервиса**

При создании интерфейса должны быть выполнены следующие требования:

- русифицированный интерфейс конечного пользователя;
- однозначность в понимании, то есть пункты меню (или их аналоги) и управляющие кнопки называются или изображаются так, чтобы пользователь однозначно понимал их назначение;
- все экранные формы пользовательского интерфейса выполнены в едином графическом дизайне, с одинаковым расположением основных элементов управления и навигации.

## **6. ПОРЯДОК КОНТРОЛЯ И ПРИЁМКИ СИСТЕМЫ**

В рамках выполнения данного проекта – не предусмотрено. В перспективе выполнение этапа возможно в рамках нового проекта.



## **7. ТРЕБОВАНИЯ К ДОКУМЕНТИРОВАНИЮ**

### **7.1 Состав программной документации**

Основными документами, регламентирующими разработку будущей программы, являются документы Единой системы программной документации (ЕСПД):

- Техническое задание (ГОСТ 19.201-78);
- Пояснительная записка (ГОСТ 19.404-79).

### **7.2 Специальные требования к программной документации**


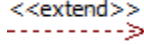
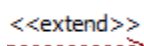
Документы к программе должны быть выполнены в соответствии с ГОСТ 19.106-78 и ГОСТами к каждому виду документа (см. п. 8.1.); Пояснительная записка должна быть загружена в систему Антиплагиат.ВУЗ. Лист, подтверждающий загрузку пояснительной записки, сдается в учебную часть вместе со всеми материалами не позже, чем за день до защиты выпускной квалификационной работы (ВКР).

Техническое задание и пояснительная записка, титульные листы других документов должны быть напечатаны, подписаны академическим руководителем образовательной программы 09.03.04 «Программная инженерия», руководителем разработки и исполнителем перед сдачей ВКР в учебную часть не позже одного дня до защиты.

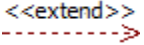
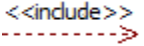
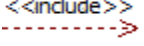
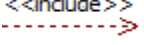

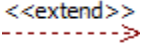
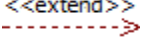
Документация и программа также сдается в электронном виде в формате .pdf или .docx. в архиве формата .zip или .rar.

## Приложение Б

Таблица Б.1 – Отношения между прецедентами и акторами

№	Прецедент/актор (источник)	Вид отношения	Прецедент (приёмник)	Характеристика описание отношения
1	Составитель	 направленная ассоциация	Создать список литературы	Составитель добавляет в список источники, из которых формируется список
			Добавить новый источник	Составитель вносит данные об источнике
			Поиск источников	Составитель выбирает по каким атрибутам искать источник
			Просмотр данных	Составитель видит всю необходимую информацию на веб-сервисе
			Очистить список	Сотрудник отправляет запрос на получение отчёта
			Сохранить список в файл	Составитель может сохранить список литературы во внешний файл
			Провести транслитерацию	Составитель может воспользоваться транслитерацией списка
			Выбор типа источника	Составитель выбирает тип источника для его заполнения
2	Очистить список	 <<extend>>	Создать список литературы	Очистить список можно только в том случае, если он имеется
3	Сохранить список в файл	 <<extend>>	Создать список литературы	Сохранить список можно только в том случае, если он имеется

Продолжение Таблицы Б.1

4	Провести транслитерацию		Создать список литературы	Провести транслитерацию списка можно только в том случае, если он имеется
5	Добавить новый источник		Создать список литературы	В составление списка входит процесс добавления нового источника
6	Выбор типа источника		Добавить новый источник	Прежде, чем добавлять новый источник, необходимо выбрать его тип
7	Поиск источников		Просмотр данных	Просмотр данных об источниках включает в себя поиск
8	Сотрудник	 направленная ассоциация	Авторизация	Сотрудник библиотеки вводит свои данные, необходимые для авторизации
			Редактировать источник	Сотрудник изменяет данные об источнике
			Удалить источник	Сотрудник удаляет источник из БД
			Поиск источников	Сотрудник выбирает по каким атрибутам искать источник
			Просмотр данных	Сотрудник видит всю необходимую информацию на веб-сервисе
9	Редактировать источник		Авторизация	Прецедент не может быть выполнен без авторизации сотрудника
10	Удалить источник		Авторизация	Прецедент не может быть выполнен без авторизации сотрудника

Продолжение Таблицы Б.1

11	БД (база данных)	<hr/> ассоциация	Добавить новый источник	Двунаправленное отношение между веб-сервисом и БД, отправление запросов в БД и получение ответов от неё
			Поиск источников	
			Просмотр данных	
			Авторизация	
			Редактировать источник	

## Приложение В

### Описание потоков

#### Основной поток событий

1. Пользователь заходит на веб-сервис.  
*A1. Пользователь является сотрудником.*
2. Пользователь открывает страницу с составлением списка литературы.  
*E1. Отсутствует подключение к сети.*  
*E2. Сервер недоступен.*
3. Пользователь ищет источники.
4. Пользователь просматривает результат поиска.  
*A2. Пользователь ничего не нашел.*
5. Пользователь добавляет источник в список литературы.
6. Пользователь выбирает тип источника.
7. Пользователь переходит на страницу добавления нового источника.
8. Пользователь выбирает количество авторов у источника.
9. Пользователь заполняет поля необходимыми данными.  
*E3. Поля заполнены неверно.*
10. Пользователь сохраняет источник.
11. Пользователь возвращается к составлению списка литературы.  
*A3. Пользователь решает продолжить составление списка.*  
*A4. Пользователь решает очистить список.*  
*A5. Пользователь решает провести транслитерацию списка.*
12. Пользователь сохраняет список во внешний файл.

#### Альтернативные потоки

- A1. Пользователь является сотрудником.
1. Пользователь переходит на страницу с авторизацией.
  2. Пользователь вводит логин и пароль.  
*E4. Сотрудник не авторизован.*
  3. Пользователь переходит на страницу управления источниками.

4. Пользователь редактирует источники.
5. Пользователь удаляет источники.
- A2. Пользователь ничего не нашел.
  1. Поток переходит к этапу 6 основного потока.
- A3. Пользователь решает продолжить составление списка литературы.
  1. Поток переходит к этапу 3 основного потока.
- A4. Пользователь решает очистить список литературы.
  1. Пользователь очищает список литературы.
  2. Поток переходит к этапу 3 основного потока.
- A5. Пользователь решает провести транслитерацию списка литературы.
  1. Пользователь проводит транслитерацию списка литературы.
  2. Поток переходит к этапу 12 основного потока.

#### Потоки ошибок

- E1. Отсутствует подключение к сети Интернет.
  1. Веб-сервис выводит сообщение об отсутствии подключения к сети Интернет.
  2. Поток возвращается к этапу 1 основного потока.
- E2. Сервер недоступен.
  1. Веб-сервис выводит сообщение о недоступности сервера.
  2. Поток возвращается к этапу 1 основного потока.
- E3. Поля заполнены неверно.
  1. Веб-сервис выводит сообщения о неверном заполнении полей.
  2. Поток возвращается к этапу 9 основного потока.
- E4. Сотрудник не авторизован.
  1. Веб-сервис выводит сообщение об ошибке авторизации.
  2. Сотрудник вводит данные заново логин и/или пароль.
  3. Поток возвращается к этапу 1 альтернативного потока A1.

## Приложение Г

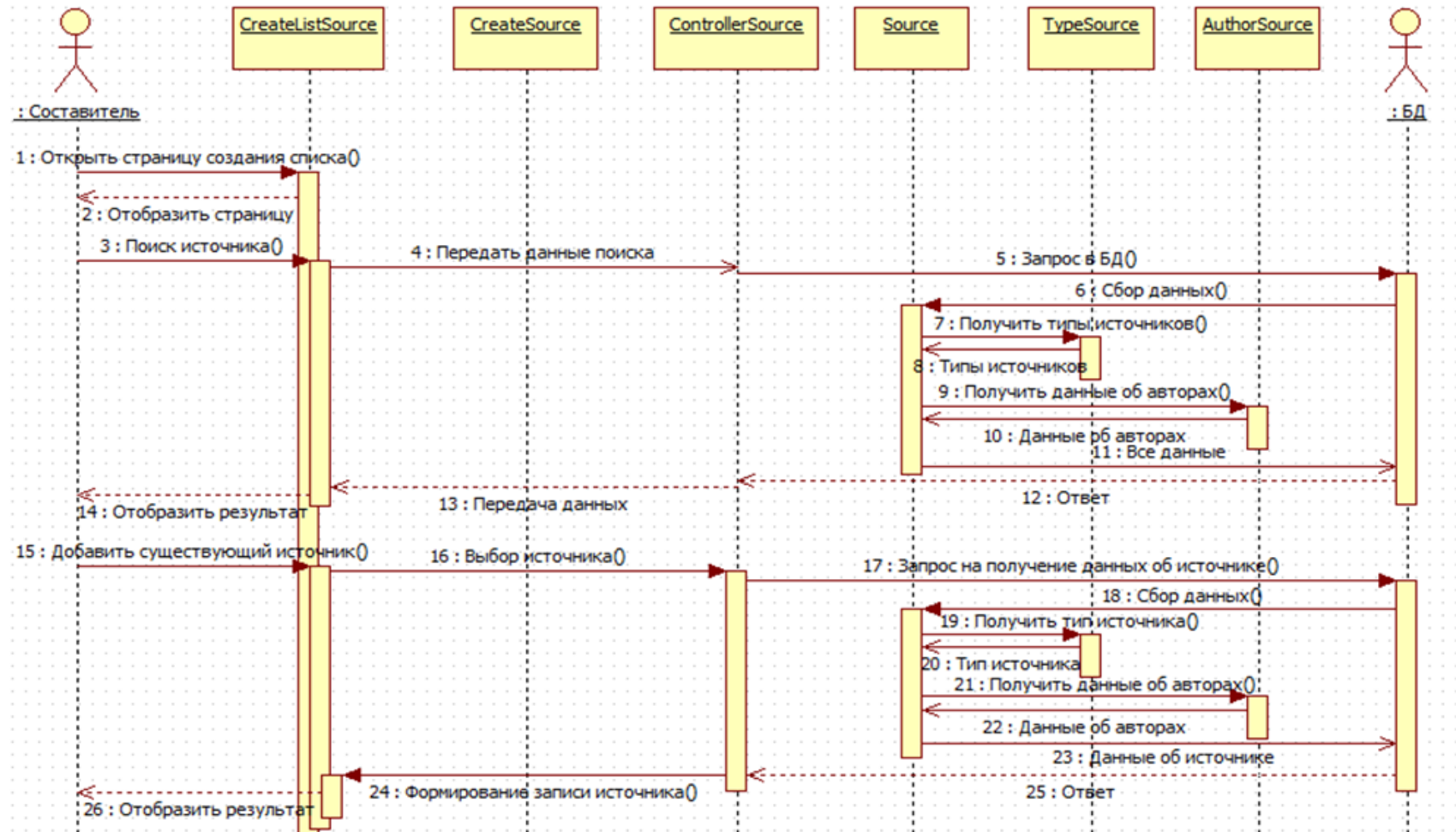


Рисунок Г.1 – Диаграмма последовательности для прецедента «Создание списка литературы» (начало)

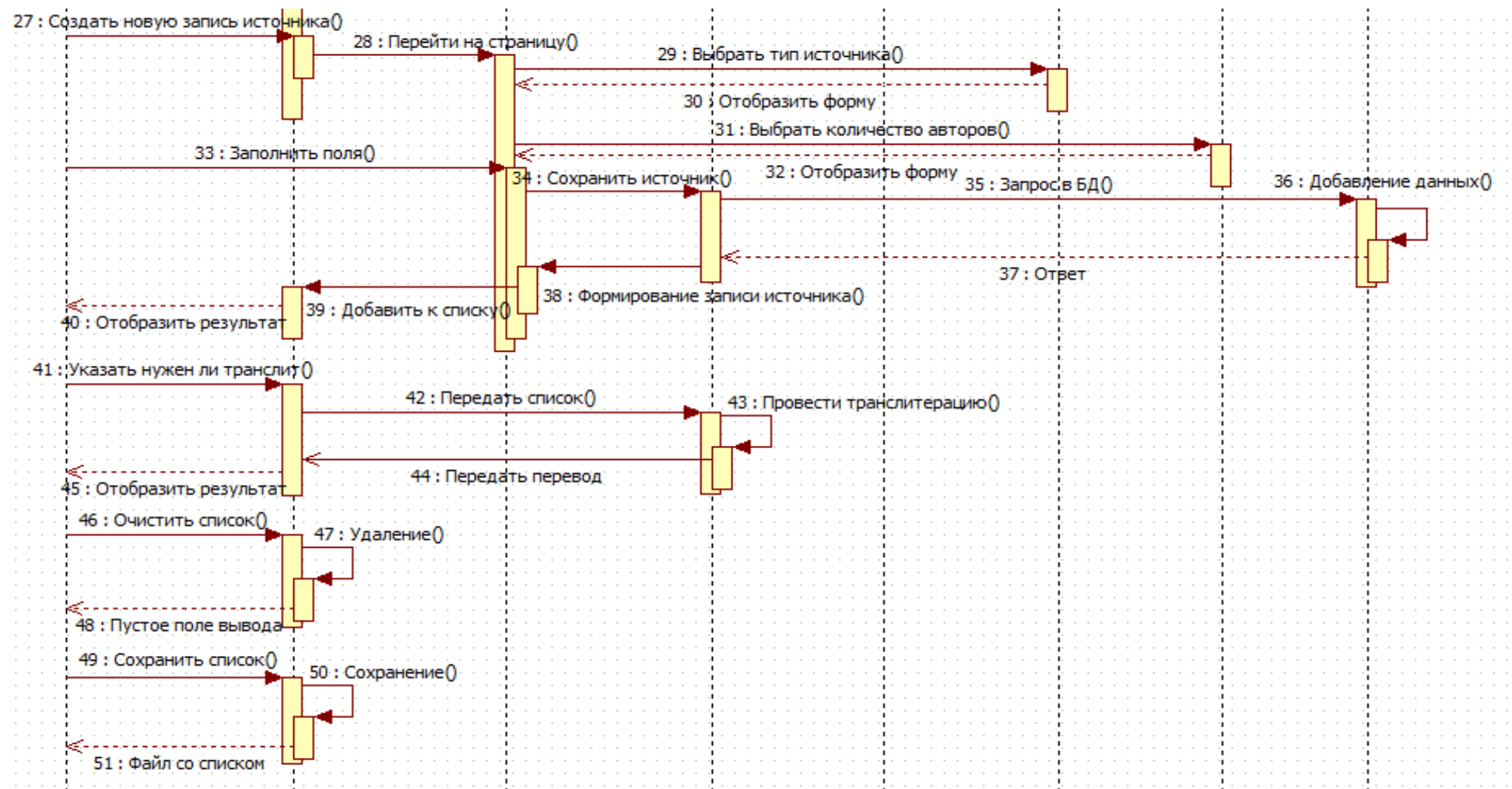


Рисунок Г.2 – Диаграмма последовательности для прецедента «Создание списка литературы» (окончание)



## Приложение Д

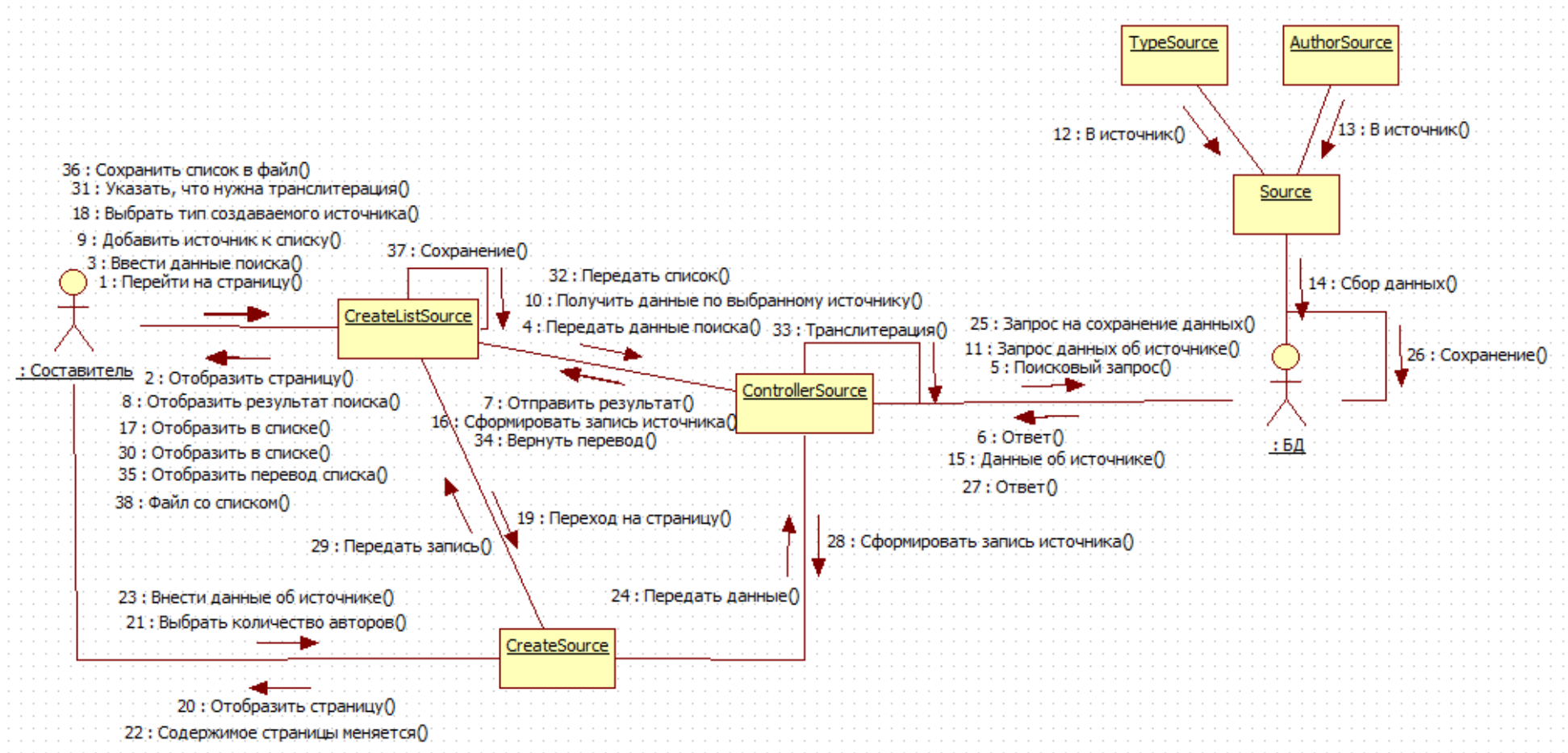


Рисунок Д.1 – Диаграмма кооперации для прецедента «Создание списка литературы»

## Приложение Е

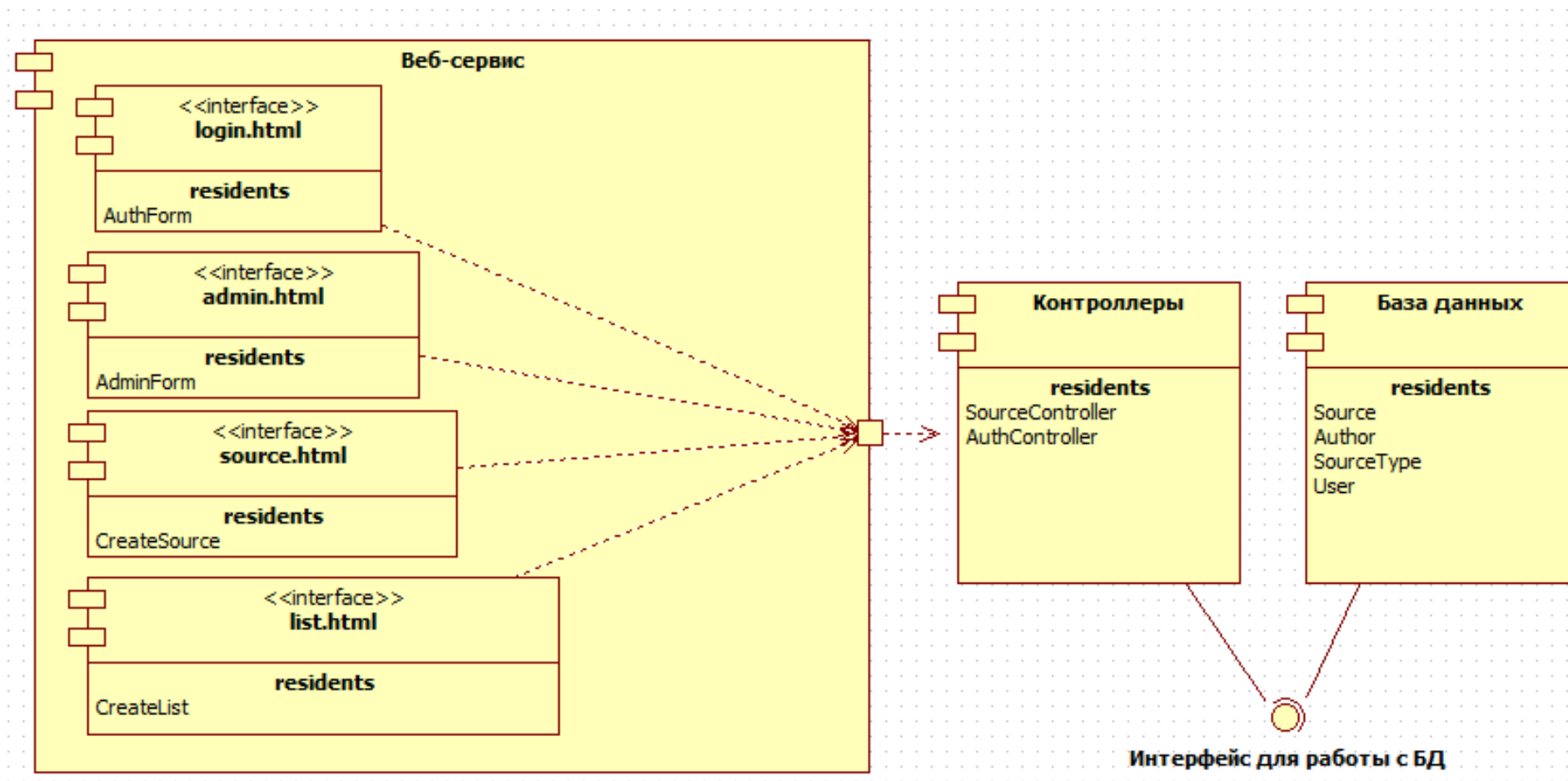


Рисунок Е.1 – Диаграмма компонентов проектируемого веб-сервиса