

**Направление 09.03.02 «Информационные системы и технологии»**

**Профиль Информационно-управляющие системы**

**Кафедра Информационные системы**

*К защите допустить:*

**Заведующий кафедрой**

**Цехановский В.В.**

**ВЫПУСКНАЯ  
КВАЛИФИКАЦИОННАЯ РАБОТА  
БАКАЛАВРА**

***Тема: “Разработка системы ввода и учета библиографической информации”***

**Студент**

\_\_\_\_\_

*подпись*

***Бестужев М.П.***

\_\_\_\_\_

*Фамилия И.О.*

**Руководитель**

***К.т.н., доцент***  
*(Уч. степень, уч. звание)*

\_\_\_\_\_

*подпись*

***Шилов Н.Г.***

\_\_\_\_\_

*Фамилия И.О.*

**Консультант по  
экономическому  
обоснованию**

***К.э.н., доцент***  
*(Уч. степень, уч. звание)*

\_\_\_\_\_

*подпись*

***Ширяева Т.П.***

\_\_\_\_\_

*Фамилия И.О.*

**Нормоконтроль**

***Ст. преп.***  
*(Уч. степень, уч. звание)*

\_\_\_\_\_

*подпись*

***Коробкин В.П.***

\_\_\_\_\_

*Фамилия И.О.*

**Антиплагиат**

***К.т.н., доцент***  
*(Уч. степень, уч. звание)*

\_\_\_\_\_

*подпись*

***Назаренко Н.А.***

\_\_\_\_\_

*Фамилия И.О.*

## ЗАДАНИЕ

Утверждаю

Зав. кафедрой АСОИУ

Цехановский В. В.

«      » 2017 г.

Студент                      Бестужев М.П.

Группа 3373

Тема работы: Разработка системы ввода и учета библиографической информации

Место выполнения ВКР: Санкт-Петербургский государственный электротехнический университет «ЛЭТИ» им. В.И. Ульянова (Ленина)

Исходные данные (технические требования):

Создание базы данных для поддержки и сохранения данных, разработка программного средства для ввода библиографической информации и её отображения

## Содержание ВКР:

Постановка задачи и анализ предметной области, база данных для хранения информации, программный интерфейс для взаимодействия с базой данных, язык разработки интерфейса

Перечень отчетных материалов: пояснительная записка, иллюстративный материал, иные отчетные материалы

## Дополнительные разделы: технико-экономическое обоснование

Дата выдачи задания	Дата представления ВКР к защите
«    »                      2017 г.	«    »                      2017 г.

Студент Бестужев М.П.

Руководитель К.т.н., доцент Шилов Н.Г.  
(Уч. степень, уч. звание)

## КАЛЕНДАРНЫЙ ПЛАН ВЫПОЛНЕНИЯ ВЫПУСКНОЙ КВАЛИФИКАЦИОННОЙ РАБОТЫ

Утверждаю

Зав. кафедрой АСОИУ

Цехановский В. В.,

«      »      2017 г.

Студент                      Бестужев М.П.

Группа 3373

Тема работы: Разработка системы ввода и учета библиографической информации

№ п/п	Наименование работ	Срок выполнения
1	Обзор литературы по теме работы	02.02 – 10.03
2	Постановка задачи и анализ предметной области	15.03 – 26.03
3	База данных для сохранения информации	19.03 – 24.04
4	Язык разработки интерфейса	10.05 – 28.05
5	Оформление пояснительной записки	11.04 – 30.05
6	Оформление иллюстративного материала	04.06 – 11.06

Студент

Бестужев М.П.

Руководитель *К.т.н., доцент*  
(Уч. степень, уч. звание)

Шилов Н.Г.

## Реферат

Пояснительная записка: 50 с., 14 рис., 5 табл., 8 источников.

Объектом исследования являются веб-технологии разработки интерфейсов.

Цель работы – разработка системы для ввода и учета библиографической информации.

В ходе работы получены следующие результаты. Разработан интерфейс для системы ввода. Создана база данных для учета приходящей информации. Предложены пути развития системы в качестве основы для анализа и сбора статистических данных. Исследованы технологии клиент-серверного взаимодействия как основа разработанной платформы.

Область применения. Полученные результаты могут быть использованы при разработке системы анализа и сбора статистических данных.

Значимость работы. Полученные результаты способствуют разработке новых систем, расширяющих возможности исходной, добавляющие новые инструменты при работе с полученной информацией. Также способствуют повышению комфортности работы с разработанной платформой, уменьшению трат времени на рутинные операции.

## **Annotation**

The aim of the project is to develop a web-interface for a database of bibliographic content. The development is carried out on an engineer's laptop using software that is freely available. All activity is concentrated in writing the web application code for database queries and does not imply any other kind of expression of the total, except as demonstrating the functionality of the interface.

The developed model can be used to develop a system for analyzing and collecting statistical data.

## Содержание:

Введение .....	7
Глава 1. Постановка задачи и анализ предметной области .....	9
1.1. Техническое задание .....	9
1.2. Индексирование публикаций .....	11
1.3. Актуальность задачи .....	15
Глава 2. Модель данных и инструментарий для системы ввода и учета библиографической информации .....	17
2.1. Модель данных хранения информации .....	17
2.1.1. Варианты использования системы пользователем.....	17
2.1.2. Разработка модели классов .....	18
2.2. Обзор используемых программных продуктов.....	22
2.3 Дополнительно используемые библиотеки.....	25
Глава 3. Реализация.....	27
3.1 Сценарии использования системы .....	27
3.2 Используемые формы.....	34
3.3 Сценарии выполняемые системой .....	36
3.4 Удобство работы .....	37
Глава 4. Техничко-экономическое обоснование.....	39
4.1 Этапы технико-экономического обоснования .....	39
4.2 Трудоемкость выполнения работ .....	40
4.3 Сырье и материалы .....	45
4.4 Амортизационные отчисления .....	46
4.5 Накладные расходы .....	47
4.6 Совокупные расходы на ВКР.....	47
Заключение .....	50
Список литературы .....	51
Приложение А. Листинг основных сценариев, исполняемых системой.....	52

## **Введение**

В наше время все больше внимания уделяется области автоматизации ручного труда, также, как и его упрощению. После того как ПК вошли в нашу жизнь, люди задумались, что для обычного пользователя и повседневных нужд нужны более дружелюбные программы. Используемые программы, которые создавались в самом начале пути, проектировались разработчиком только для себя и не имели удобного интерфейса. Поэтому в наше время большое внимание уделяется областям упрощения и оптимизации интерфейсов для новых решений в области информационных технологий. Это происходит не только для удобства пользователя, но и по причинам увеличения трудовых затрат при работе с многофункциональным программным продуктом, поэтому разработчики стремятся к уменьшению времени на выполнение даже самых сложных ранее манипуляций.

В период всемирного охвата беспроводных технологий, массового распространения сети Интернет и связи всех способных на это устройств и ПК становится более привлекательны способы удаленного доступа и управления к повседневным системам. Поэтому продукт ВКР будет иметь потенциальные возможности по коммуникации через сеть Интернет.

В данной ВКР преследуются цели разработки удобного интерфейса для учёта научных публикаций, статей, изданий. Разрабатываемая система будет представлена как web-приложение, что дает возможность оперативного и беспроблемного доступа к базе данных. Будущая система в дальнейшем может быть полезна для использования в сборе статистики или аналитики в данной области.

Систем подобных этой существуют в небольшом количестве, но они не все занимаются учетом научных публикаций в полной мере с удобным интерфейсом добавления этих публикаций и не имеют необходимого функционала. Подобные задачи решались и ранее, но в комплексе они позволяют стать этой системе

удобным инструментом и базой данных для доступа к любой научной информации, внесенной в неё, так и её необходимым сопровождением.

В работе были поставлены следующие задачи:

- Изучить информацию по разработке интерфейсов для баз данных.
- Выбрать наиболее удобную архитектуру и язык программирования.
- Спроектировать базу данных, дающую возможность сохранить более полную информацию вводимых данных.
- Разработать интерфейс и логику работы системы ввода и учета информации библиографического содержания.



## **Глава 1. Постановка задачи и анализ предметной области**

### **1.1. Техническое задание**

Появилась необходимость в разработке системы для ввода и учета различных научных изданий и рукописей, которая позволит решать следующие задачи:

- Предоставлять пользователю удобную форму для занесения информации о своей публикации.
- Передавать и сохранять все данные в специально подготовленную базу данных.
- Представлять пользователю информацию о имеющихся публикациях, сохраненных в базе данных.
- Редактировать информацию об уже занесенных публикациях.
- Предоставить возможность интеграции и улучшения системы для добавления новых алгоритмов фильтрации и поиска публикаций, дающих возможность более полно анализировать информацию, сокращающих время на поиск необходимой информации и предоставляющих возможность формирования отчетов.

Необходимо разработать интерфейс системы с расчетом что ей сможет пользоваться человек без профессиональных навыков в области компьютерных технологий. Главные критерии при разработке – наглядность и комфорт при работе с системой.

Задача разработки такой системы не нова, но актуальна. Существует множество методов для решения для её решения, что ставит перед вопросом о выборе одного метода из множества. Анализ множества методик в этой сфере поможет рассмотреть различные возможные варианты реализации системы, которые смогут решить поставленные задачи.

Целью данной выпускной квалификационной работы является разработка интерфейса, обработка и передача данных от интерфейса, создание базы данных для содержания всей информации.

## **1.2. Индексирование публикаций**

Значимость процесса индексирования может быть неочевидна. Простой публикаций в каком-либо печатном издании недостаточно для признания и известности. Публикация будет доступна только на страницах издания, а затем и вовсе потеряется по происшествии времени. Для того, чтобы научные труды были сохранены, а также более распространены и доступны для быстрого поиска существует индексирование. Оно позволяет резко увеличить популярность и доступность изданий во всем мире. При этом выявление плагиата в таких трудах станет на порядок легче, что подразумевает оперативную защиту авторских прав.

Благодаря помощи в индексировании публикаций стало возможно быстро и удобно создать индекс научного цитирования. Признанный во всем научном мире критерий «значимости» трудов ученого. Суть заключается в числе ссылок на публикацию ученого в индексируемых научных периодических изданиях. Существование в научно-образовательных учреждениях ученых, имеющих высокий индекс цитируемости, говорит о высокой эффективности и результативности деятельности организации. Составляются библиографические базы научных публикаций, проводятся их индексирование, определяются индексы цитируемости и другие статистические показатели научных работ. Начиная с 1975 года, индекс значимости научных изданий стали рассчитывать ежегодно. Ранее они проводились для журналов, которые индексировались в Journal Citation Reports (JCR). Это отличный от других междисциплинарный список научных журналов. Со временем JCR была интегрирована в Web of Science.

Существует несколько способов расчета индекса. Для примера остановимся на одном из них называемый импакт-фактор. Данный индекс рассчитывается как число ссылок, полученные журналом в текущем году на статьи, опубликованные в этом журнале за два предыдущих года, к числу статей, опубликованных в этом журнале за этот же период. Это наглядно продемонстрировано на рисунке 1.1.

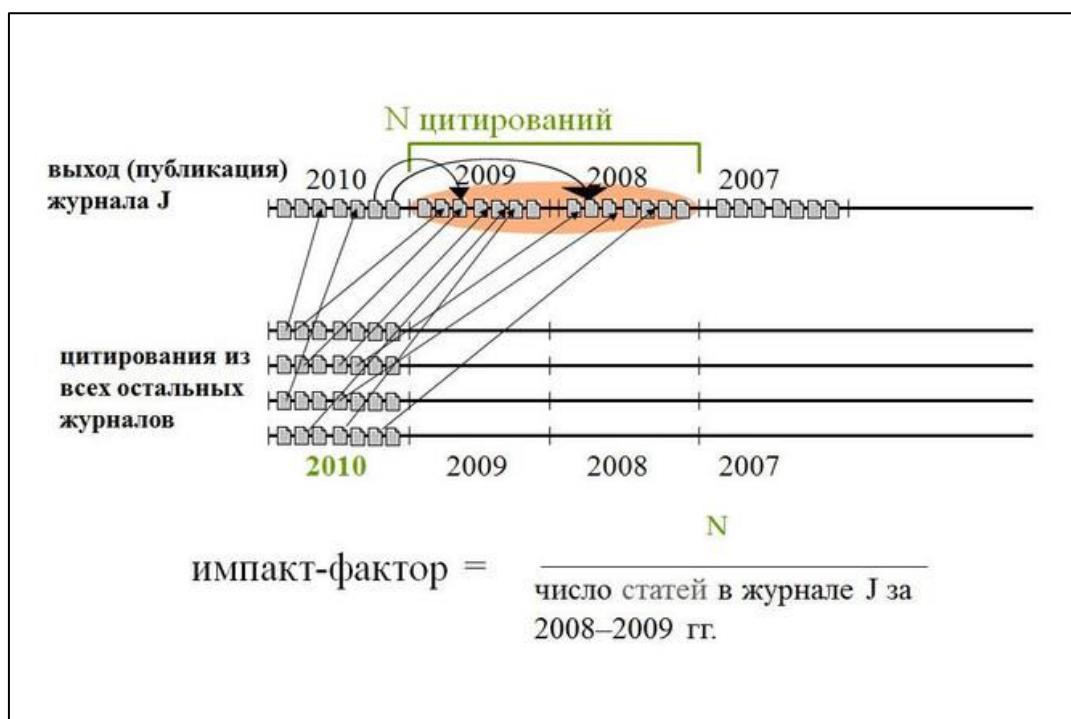


Рисунок 1.1 – Расчет импакт-фактора

Этот показатель используется для сравнения журналов, близких по тематике, однако не дает возможности сравнивать качество журналов из разных научных областей. Основными недостатками импакт-фактора являются большая зависимость от области науки, непредсказуемый временной отрезок «публикационного окна» и различие типов документов в числителе и знаменателе формулы расчета: в числителе учитываются ссылки из всех типов документов, а в знаменателе учитываются только статьи и обзоры.

Несовершенство порядка выбора публикаций или неразборчивость в этом процессе незамедлительно сказываются на изменении импакт-фактора. Для авторов коэффициент важности издания помогает принять решение, в какой журнал лучше отправить публикацию. Высокий импакт-фактор издания, конечно, частично является критерием его качества, но иногда даже в таких изданиях неплохая статья не пройти через строгие рамки отбора. Для научных учреждений количество публикаций сотрудников в журналах с высоким импакт-фактором дает сведения о высоком качественно-научном уровне исследований, проводимых в них.

Существует некоторое количество систем задач, которых является пополнение своих баз данных научными работами, обеспечение доступа к ним, предоставление метаданных о них, расчет оценки цитируемости и прочее.

Одна из них это платформа eLIBRARY.RU – крупнейшая в России электронная библиотека научных публикаций, имеющая в наличии широкие возможности поиска и анализа научной информации. Библиотека связана с одной из разновидностей индексов цитирования Российским индексом научного цитирования (РИНЦ) - бесплатным общедоступным инструментом вычисления публикационной активности ученых и организаций путем подсчета количества упоминаний научных трудов автора в публикациях других ученых.

Платформа была создана в 1999 году по инициативе Российского фонда фундаментальных исследований для обеспечения доступа российских ученых к наиболее авторитетным иностранным научным изданиям. С 2005 года начала работу с русскоязычными публикациями и до сегодняшнего дня является наиболее обширной электронной библиотекой научной периодики на русском языке в мире. На сегодня посетителям доступны полные тексты более 24 млн научных статей и публикаций, в том числе электронные версии более 5300 российских научно-технических журналов. Общее число зарегистрированных институциональных пользователей (организаций) - более 2800. В системе зарегистрированы 1,7 миллиона индивидуальных пользователей из 125 стран мира. Ежегодно читатели получают из библиотеки более 12 миллионов полнотекстовых статей и просматривают более 90 миллионов аннотаций. Свыше 4500 российских научных журналов размещены в бесплатном открытом доступе.

[1]

Существует в том числе и Русское агентство цифровой стандартизации (РАЦС) – это организация, ставящая себе цель стандартизировать научные труды в России и СНГ. РАЦС поспособствовал становлению в России и СНГ стандарта DOI (Digital Object Identifier) — идентификатор цифрового объекта. DOI применяется для упрощения процесса цитирования, поиска и локализации

научной информации, повышает авторитетность журнала и свидетельствует о его качестве как источника информации технологического характера, является необходимым атрибутом системы научной коммуникации за счет оперативного обеспечения процессов обмена научной информацией, способствует гарантированному переходу на актуальное местонахождение отдельно взятой публикации, это стандарт, принятый всеми ведущими издательствами. Организация РАЦС присваивает DOI академическим цифровым данным для улучшения их распознавания и последующего цитирования. Данные размещаются в базах данных РАЦС.

Европейской разработкой в области учета публикаций является European Reference Index for the Humanities and the Social Sciences (ERIH PLUS) – реферативная база по гуманитарным и социальным наукам. Публикации представлены как на английском, так и на ряде основных европейских языков. Реферативная база данных и цитирования научных материалов ERIH PLUS является наиболее крупной в мире системой по публикации и отслеживанию работ в самых различных сферах на нескольких основных языках. Многие специалисты из таких областей науки, как антропология, археология, гендерные исследования, история, философия, лингвистика, психология, литературоведение, музыковедение и многих других регулярно публикуют собственные рукописи в базе данных организации. Это позволяет им демонстрировать результаты своей работы международному научному сообществу, находить единомышленников, получать гранты на дальнейшие исследования и продвигать тем самым свою карьеру.

ERIH не является библиографическим или рейтинговым инструментом. Цель создания этого индекса - повышение доступности ведущих европейских исследований в области гуманитарных наук, а также облегчение доступа к научно-исследовательским журналам, изданных на всех европейских языках. Включение издания в этот индекс позволяет исследователям всего мира пользоваться научными публикациями, продемонстрированных в журнале,

повышает цитируемость авторов журнала в научных трудах ученых разных стран.

Одна из старейших основанных библиографических баз данных называется WorldCat. Является крупнейшей в мире сетью, предоставляющей библиотечный контент и услуги по работе с ним. База преследует цель обеспечения доступа к ресурсам в сети Интернет, где большинство пользователей начинают свой поиск информации. Это объединенный электронный каталог библиотек всего мира, участвующих в OCLC, насчитывает более 35 миллионов библиографических записей и 600 миллионов адресных списков. Ежегодно количество повышается на 2 млн. записей. Особенностью является, что пользователи не могут добавить свою публикацию в базу. Платформа сама просматривает библиотеки и индексирует в своей базе данных. Пользователи могут только искать необходимое и обсуждать публикации.

### **1.3. Актуальность задачи**

Ранее упомянутые системы подобны разрабатываемой и можно утверждать, что необходимость в такой платформе минимальна. Все платформы, описанные ранее, безусловно, дают возможность добавления публикации или индексирования уже добавленной в другой базе данных и занесения в свою. У некоторых упомянутых есть специальные инструменты для поиска и необходимого анализа. Но задача проектируемой платформы состоит не только в удобной работе с занесением необходимой информации. Также необходимым условием является предоставление возможности впоследствии провести собственный анализ этой информации, не ограниченным рамками предлагаемых инструментов и позволяющий работать непосредственно с базой данных, что крупные сервисы не могут предоставить, ввиду обеспечения безопасности и сохранности своих данных.

Таким образом актуальность данной задачи выражается в необходимости непосредственного доступа к базе данных системы и работы с ней и интерфейса,

отвечающего требованиям и спроектированного для определенного количества людей.

### **Вывод**

В итоге, были оформлены требования к разрабатываемой системе, выявлена необходимость и важность индексации в цифровую эпоху. Продемонстрированы системы, которые ставят своей задачей не только индексацию, но и некоторый анализ информации. Доказана актуальность работы и задач, которые не могут быть выполнены на других платформах.



## **Глава 2. Модель данных и инструментарий для системы ввода и учета библиографической информации**

### **2.1. Модель данных хранения информации**

Для сохранения информации будем использовать базу данных (БД). В начале объясним, что собой представляет БД. Итак, это специально разработанное хранилище для различных типов данных. Каждая БД, имеет определённую модель (реляционная, документно-ориентированная), которая обеспечивает наиболее удобный доступ к данным. Системы управления базами данных (СУБД) - специальные приложения (или библиотеки) для управления базами данных различных размеров и форм. [2]

СУБД должна обеспечивать реляционную модель работы с данными. Сама модель подразумевает определенный тип связи между сущностями из разных таблиц. Чтобы хранить и работать с данными, такой тип СУБД должен иметь определенную структуру (таблицы). В таблицах каждый отдельный столбец может содержать данные различного типа. Каждая отдельная запись состоит из множества атрибутов (столбцов) и имеет уникальный ключ, хранящейся в той же таблице – все эти данные взаимосвязаны между собой, как описано в реляционной модели. [3]

Удачная разработка базы данных обеспечивает простоту ее поддержки. Данные следует сохранять в таблицах, причем каждая таблица должна содержать информацию одного типа, например, сведения об издании. Тогда достаточно будет обновить конкретные данные, такие как год выпуска издания, только в одном месте, чтобы обновленная информация отображалась во всей базе данных.

#### **2.1.1. Варианты использования системы пользователем**

Для разработки архитектуры системы будем использовать CASE-инструмент для проектирования и конструирования программного обеспечения Enterprise Architect.

Пользователь в системе будет иметь возможность просматривать весь список внесенных в базу данных публикаций. В частности, получать информацию о конкретной публикации или при помощи соответствующих фильтров искать необходимую. Пользователь также будет иметь возможность работать с данными публикаций, а именно добавлять новую публикацию в систему или изменять информацию о существующей. Диаграмма, демонстрирующая эти возможности, представлена на рисунке 2.1.1:

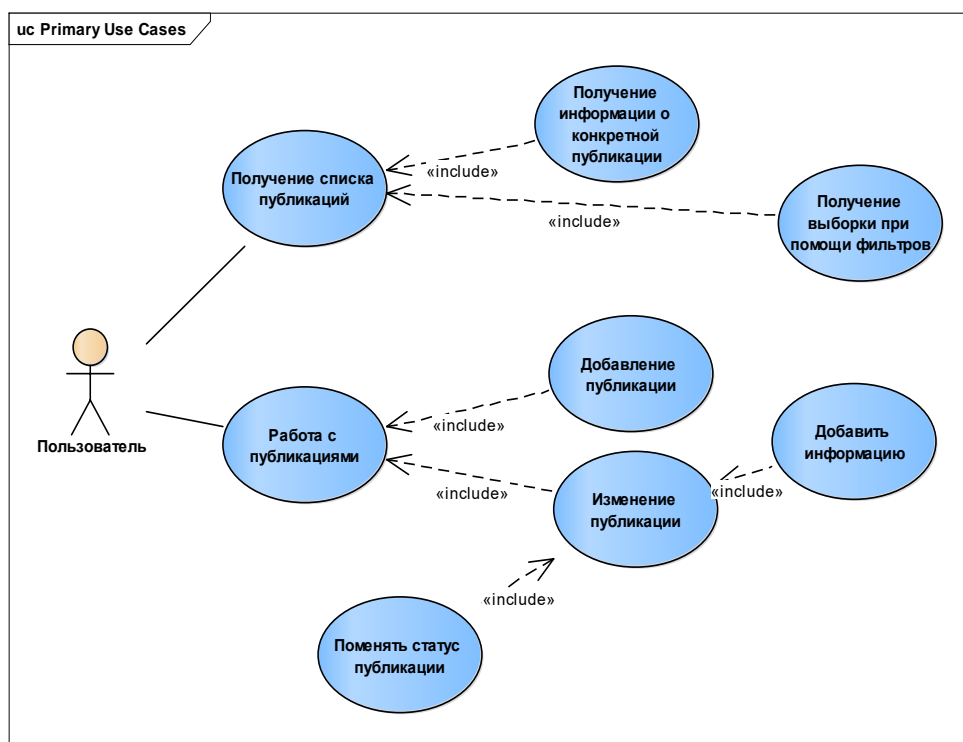


Рисунок 2.1.1 – Диаграмма прецедентов для пользователя

### 2.1.2. Разработка модели классов

Сохранение информации о внесенных публикациях это задача базы данных.

Правильно спроектированная база данных позволяет при помощи разнообразных запросов отображать необходимую информацию так, чтобы составление запросов не вызывало больших сложностей. В начале создадим диаграмму классов, содержащую концепцию будущей модели данных и позволяющая понять с чем будет работать в дальнейшем интерфейс этой системы. Разработанная диаграмма представлена на рисунке 2.1.2.



- Comment – комментарии, которые может вписать пользователь.

Сущность *Conference* позволяет хранить все данные о конференциях.

Содержит в себе следующие атрибуты:

- StartDate – дата начала конференции;
- EndDate – дата окончания конференции;
- ShortName – сокращенное название конференции;
- ConferenceName – полное название конференции;
- Comment – комментарии, которые может вписать пользователь.

Сущность *Book* предназначена для хранения информации об издании, которое содержит публикацию. Содержит в себе следующие атрибуты:

- Name – название издания. Не заполняется для обычных журнальных публикаций;
- Editors – редактор описываемого издания;
- BookYear – год печати издания;
- ISBN – международный стандартный книжный номер;
- BookVol – том, в который входит издание (при наличии ссылки на серию);
- BookIss – номер выпуска издания в томе (при наличии ссылки на серию).

Сущность *ConferenceLocation* позволяет хранить список мест проведения конференций. Содержит в себе следующие атрибуты:

- Location – название места проведения конференции.

Сущность *Series* позволяет хранить данные о серии изданий. Содержит в себе следующие атрибуты:

- SeriesName – название серии, в которую входит издание;
- ShortName – сокращенное название серии;
- ISSN – международный стандартный серийный номер.

Сущность *IndexSystem* хранит список названий систем индексирования для изданий или их серий. Содержит в себе следующие атрибуты:

- IndexName – название системы.

Сущность *Rate* хранит данные о рейтингах серий книг. Содержит в себе следующие атрибуты:

- RateYear – год вычисления и присвоения индекса;
- JCR – значение индекса научного издания;
- SJR – значение индекса научного издания.

Сущность *Publisher* позволяет хранить данные об издателях. Содержит в себе следующие атрибуты:

- PublisherName – название издателя;
- PublisherCity – город издателя.

Сущность *AuthorName* позволяет хранить полную информацию об авторе публикации. Содержит в себе следующие атрибуты:

- LastName – фамилия автора;
- FirstName – имя автора;
- OtherName – отчество автора (при наличии).

Сущность *StatusDictionary* хранит список статусов, присваиваемые публикации. Содержит в себе следующие атрибуты:

- Description – описание статуса труда.

Сущность *Affiliation* хранит список всех мест работы всех авторов. Содержит в себе следующие атрибуты:

- AffiliationName – название места работы.

Сущность *Acknowledgement* позволяет хранить список благодарностей, упоминаемых в публикации. Содержит в себе следующие атрибуты:

- AcknowName – краткий внутренний идентификатор;
- Comment – полное описание записи.

Сущность *Language* позволяет хранить список языков для публикации или гражданства автора. Содержит в себе следующие атрибуты:

- Code – международный код языка;
- FullNameLang – полное название языка;

- ShortNameLang – краткое название языка.

Все остальные сущности являются связующими для описанных выше.

## **2.2. Обзор используемых программных продуктов**

Для разработки подобной системы необходим инструментарий, который обеспечит не только удобство работы, но и предоставит множество возможностей для воплощения поставленных задач и идей. Выбор обширен и для воплощения задач подходят многие решения в сфере программных продуктов. Поэтому критерии выбора будут простыми:

- Предлагаемый инструмент должен предоставлять возможность интеграции с другими инструментами для удобства разработки.
- Быть свободным программным обеспечением для минимизации затрат.
- Обеспечивать приемлемый уровень производительности системы.
- Иметь большое сообщество для оперативного решения возникающих сложностей.
- Предпочтительно иметь опыт работы с выбранным продуктом для меньшего времени адаптации к новому инструментарию. [4]

В начале выберем систему управления базами данных(СУБД). При создании таблицы каждый столбец должен иметь заранее заявленный тип (например: строка, целочисленное значение и т.д.). Все СУБД умеют обращаться с различными типами данных, которые не всегда взаимозаменяемы. При работе с СУБД всегда приходится сталкиваться с подобными ограничениями.

PostgreSQL представляет собой самую профессиональную из наиболее популярных СУБД. Она свободно распространяемая и наиболее полно отвечает эталонам SQL. PostgreSQL или же Postgres стремятся всецело использовать ANSI/ISO SQL стандарты своевременно с выходом свежих версий. От иных СУБД PostgreSQL выделяется поддержкой необходимого объектно-ориентированного и/или реляционного подхода к базам данных. К примеру, полная поддержка достоверных транзакций, т.е. атомарность, очередность,

изоляция, прочность. Благодаря мощным технологиям PostgreSQL довольно производительна. Параллельность достигнута не за счет блокировки операций чтения, а благодаря реализации управления многовариантным параллелизмом (MVCC), что также обеспечивает соответствие стандартам ACID. Хотя PostgreSQL и не может похвастаться большой известностью в отличие от MySQL, существует довольно большое количество приложений, облегчающих работу с PostgreSQL, несмотря на всю мощь функционала. На данный момент не составит проблем установить эту СУБД, используя стандартные менеджеры пакетов операционных систем.

#### Преимущества:

- Открытое ПО соответствующее эталону SQL. PostgreSQL - бесплатное ПО с открытым исходным кодом.
- Является очень мощной системой.
- Большое сообщество - существует довольно большое сообщество, в котором вы легко найдёте ответы на собственные вопросы.
- Большое количество дополнений - несмотря на огромное количество встроенных функций, существует очень много дополнений, позволяющих разрабатывать данные для этой СУБД и управлять ими.
- Расширения - существует возможность расширения функционала за счет сохранения собственных процедур.
- Объектность - PostgreSQL это не только реляционная СУБД, но также и объектно-ориентированная с поддержкой наследования и много прочего.

#### Недостатки:

- Популярность - по своей природе известностью эта СУБД похвастаться не может, хотя и присутствует довольно большое сообщество.
- Хостинг - в силу выше перечисленных факторов довольно сложно найти хостинг с поддержкой этой СУБД.

Приняв во внимание все вышеуказанные характеристики, использование PostgreSQL как СУБД будет оправданно как наиболее подходящей ввиду строгого синтаксиса, исключающего множество ошибок, производительности, большого сообщества и приобретенного ранее опыта по работе с этой СУБД.

В качестве веб-сервера для интерфейса будем использовать Apache Tomcat. Это контейнер сервлетов, который может использоваться как сервер. Разрабатывается Apache Software Foundation, реализует спецификацию сервлетов и спецификацию JavaServer Pages (JSP) и JavaServer Faces (JSF). Также является отдельным веб-сервером, который обеспечивает работу как динамической части сайта, так и статических файлов (изображения и т.д.).

Для связи всех инструментов и непосредственно написания кода и алгоритмов системы будем использовать IntelliJ IDEA. Это интегрированная среда разработки программного обеспечения на многих языках программирования, в частности Java, JavaScript, Python, разработанная компанией JetBrains. Дизайн среды ориентирован на продуктивность работы программистов, позволяя сконцентрироваться на функциональных задачах, в то время как IntelliJ IDEA берёт на себя выполнение рутинных операций.

Начиная с версии 9.0, среда доступна в двух исполнениях: Community Edition и Ultimate Edition. Community Edition является полностью свободной версией, доступной под лицензией Apache 2.0, в ней реализована полная поддержка Java SE, Groovy, Scala, а также интеграция с наиболее популярными системами управления версиями. [5]



## 2.3 Дополнительно используемые библиотеки

При разработке были использованы и подключены в работу несколько библиотек, расширяющих возможности взаимодействия модулей и демонстрации в самом интерфейсе.

**jQuery** - это библиотека, которая облегчает и ускоряет написание JavaScript кода.

Принцип jQuery *"write less, do more"* (пиши меньше, делай больше) отражает ее главную особенность и предназначение. Что умеет jQuery:

- Обращаться к любому элементу DOM (объектной модели документа) и не только обращаться, но и манипулировать ими.
- Обрабатывать события.
- Легко оперировать различными визуальными эффектами.
- Работать с AJAX (технология, позволяющая общаться с сервером без перезагрузки страницы).

Имеет огромное количество JavaScript плагинов, предназначенных для создания элементов пользовательских интерфейсов. Они позволяют расширить ее возможности еще больше.

Данная библиотека работает со всеми браузерами (*IE 6.0+*, *FF 2.0+*, *Safari 3.0+*, *Opera 9.0+*, *Chrome*). Это означает, что эта библиотека поддерживает кроссбраузерную совместимость JavaScript кода для перечисленных браузеров.

**JSON** - простой, основанный на использовании текста, способ хранить и передавать структурированные данные. При помощи простого синтаксиса возможно хранить все, что угодно, начиная от одного числа до строк, массивов и объектов, в простом тексте. Также можно связывать между собой массивы и объекты, формируя сложные структуры данных.

После создания строки JSON формата, ее возможно легко отправить другому приложению или в другое место сети, так как она представляет собой простой текст.

JSON имеет следующие достоинства:

- Компактность.

- Предложения легко читаются и составляются как человеком, так и компьютером.
- Легко преобразовать в структуру данных поддерживаемые большинством языков программирования (числа, строки, логические переменные, массивы и так далее)
- Многие языки программирования имеют функции и библиотеки для чтения и создания структур JSON.

Название JSON расшифровывается как JavaScript Object Notation (представление объектов JavaScript). Как и представляет имя, он основан на сериализации объектов (похоже на создание ассоциативных массивов в других языках), массивов, чисел, строк логических значений и значения null. Наиболее частое распространенное использование JSON - пересылка данных от сервера к браузеру. Как правило данные JSON доставляются с помощью технологии AJAX, которая позволяет обмениваться данными браузеру и серверу без необходимости перезагружать страницу. [6]

## **Вывод**

В итоге были выбраны несколько программных продуктов для инструментария разработки: PostgreSQL для базы данных, Apache Tomcat для взаимодействия клиента и сервера, IntelliJ IDEA для интеграции всех элементов. Представлены модели данных, поддерживающих работу с информацией в системе.

## Глава 3. Реализация

### 3.1 Сценарии использования системы

Для предоставления и описания результатов работы системы в начале опишем сценарии и шаги пользователя, которые он может предпринять.

Для просмотра всех публикаций пользователю достаточно зайти на главную страницу интерфейса, она предоставит список с основной информацией о всех доступных публикациях, как показано на рисунке 3.1.

<b>HOME    ADD ARTICLE    ABOUT</b>					
Title article	Name Book	Series	Book Year	Status	
Crowd-Based Decision Support in Business Processes: Platform Architecture and Case-Based Analysis	Business Information Systems	Lecture Notes in Business Information Processing	2016	Rejected	<a href="#">Edit</a>
Hybrid Automated Line Workstations Interaction Scenario for Optical Devices Assembly	Proceeding of the 18th Conference of Open Innovations Associations FRUCT	Conference of Open Innovation Association, FRUCT	2016	Published	<a href="#">Edit</a>
Service-Based Socio-Cyberphysical Network Modeling for Guided Self-Organization (TBU)		International Journal of Information Systems and Project Management	2016	Cancelled	<a href="#">Edit</a>
Recommending Tourist Locations Based on Data from Photo Sharing Service: Method and Algorithm	Proceeding of the 18th Conference of Open Innovations Associations FRUCT	Conference of Open Innovation Association, FRUCT	2016	Published	<a href="#">Edit</a>
Многоуровневая самоорганизация ресурсов киберфизической системы: контекстно-ориентированный подход и реализация		Искусственный интеллект и принятие решений	2015	Published	<a href="#">Edit</a>
Smartphone-Based	Proceeding of the 18th Conference of	Conference of	2016	Published	<a href="#">Edit</a>

Рисунок 3.1 – Главная страница

Для редактирования публикации пользователю необходимо зайти на главную страницу интерфейса и нажать кнопку “Edit” рядом с соответствующей записью, которую необходимо изменить. Откроется новая страница с имеющейся информацией о выбранной публикации. Пользователь редактирует данные и подтверждает изменения, отправляя их на сервер, кнопкой “Submit”. Эта форма продемонстрирована на рисунке 3.2.

**HOME**   **ADD ARTICLE**   **ABOUT**

**Title:**

Hybrid Automated Line Workstations Interaction Scenario fo

**Book:**

Proceeding of the 18th Conference of Open Innovations Assc **New**

92 99

**Language:**

English ▼

**DOI:**

**Author:**

Архипов + **New Author** **New Affiliation**

Кашевник

Киприянов

Рисунок 3.2 – Страница с формой редактирования записи

Для добавления нового научного труда в базу пользователю необходимо выбрать из верхнего меню вкладку “Add article”. Произойдет перенаправление на соответствующую страницу с формами для заполнения данных о новой публикации. Пользователь заполняет элементы, следуя подсказкам. После этого необходимо подтвердить отправку данных на сервер кнопкой “Submit”. В двух последних сценариях возможно появления ошибок при заполнении. Если пользователь введет некорректные данные или данные, которые необходимо добавить в систему перед отправкой, система обратит внимание на это, заблокировав возможность отправки и укажет где произошла ошибка. Эта форма продемонстрирована на рисунке 3.3.



**HOME   ADD ARTICLE   ABOUT**

**Title:**

**Book:**  
 **New**

Page start  Page end

**Language:**

**DOI:**

**Author:**  
 **+ New Author New Affiliation**

**Acknowledgment:**  
 **+ New**

Рисунок 3.3 – Страница с формой добавления записи

Если пользователю при вводе фамилии автора система не предложила подсказки, значит этот автор отсутствует в базе данных. Необходимо внести данные о нем. Для этого вначале нужно нажать на кнопку “New” на вкладке “Add article” рядом с полем авторов. После этого откроется новое окно, демонстрируемое на рисунке 3.4. Пользователь заполнит информацию о новом авторе. После этого необходимо подтвердить отправку данных на сервер кнопкой “Submit”.

New Author - Google Chrome

localhost:8080/views/newauthor.jsp

**Last Name:**

**First Name:**

**Other Name:**

**Language:**

**Submit !**

Рисунок 3.4 – Окно для добавления нового автора

Если пользователю при вводе кода благодарности или участия система не предложила подсказки, значит этот код отсутствует в базе данных. Необходимо внести данные о нем. Для этого вначале нужно нажать на кнопку “New” на вкладке “Add article” рядом с полем кодов благодарностей. После этого откроется новое окно, демонстрируемое на рисунке 3.5. Пользователь заполнит информацию о новом авторе. После этого необходимо подтвердить отправку данных на сервер кнопкой “Submit”.

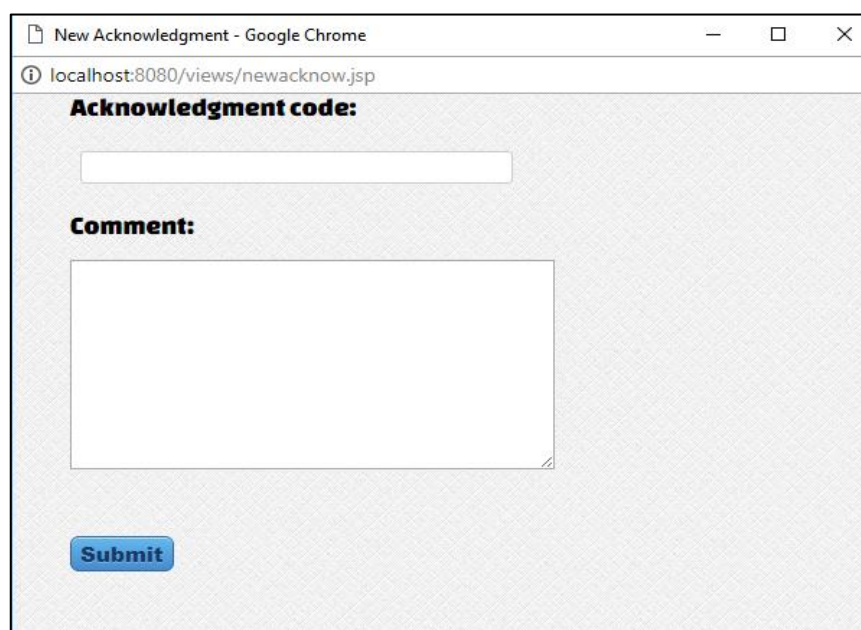


Рисунок 3.5 – Окно для добавления новой благодарности

Если пользователю при заполнении поля “Book” на вкладке “Add article” система не предложила подсказки, значит это издание отсутствует в базе данных. Для внесения нового экземпляра необходимо нажать на кнопку “New” рядом с полем названия. После этого откроется новое окно, представленное на рисунке 3.6. Пользователь заполнит информацию о новом издании. Затем необходимо подтвердить отправку данных на сервер кнопкой “Submit”.

New Book - Google Chrome  
localhost:8080/views/newbook.jsp

**Name:**

\*Не заполнять для обычных журнальных публикаций!

**Editors:**

**Book Year:**

**ISBN:**

**Conference:**

New

Если относится к периодическому изданию

Рисунок 3.6 – Окно для добавления нового издания

Если пользователю при заполнении поля “Conference” в окне “New Book” система не предложила подсказки, значит это издание отсутствует в базе данных. Для внесения нового экземпляра необходимо нажать на кнопку “New” рядом с полем. После этого откроется новое окно, представленное на рисунке 3.7. Пользователь заполнит информацию о новой конференции. Затем необходимо подтвердить отправку данных на сервер кнопкой “Submit”.

New Conference - Google Chrome  
localhost:8080/views/newconference.jsp

**Conference name:**

Date start  Date end

**Short name:**

**Location:**

New

**Comment:**

Рисунок 3.7 – Окно для добавления новой конференции

Если пользователю при заполнении поля “Location” в окне “New Conference” система не предложила подсказки, значит это место отсутствует в базе данных. Для внесения нового экземпляра необходимо нажать на кнопку “New” рядом с полем. После этого откроется новое окно, представленное на рисунке 3.8. Пользователь заполнит информацию о новом месте. Затем необходимо подтвердить отправку данных на сервер кнопкой “Submit”.

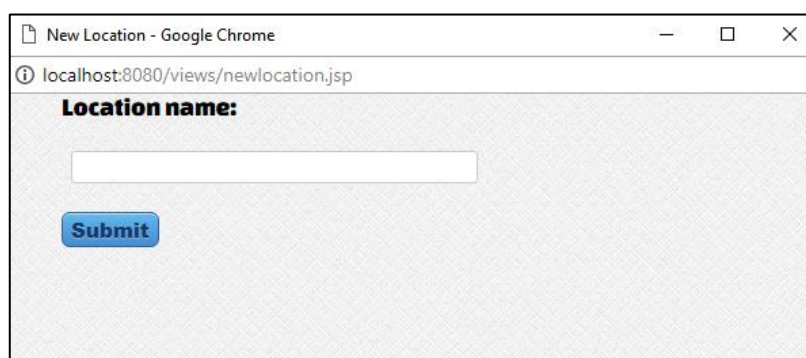
The screenshot shows a web browser window titled "New Location - Google Chrome". The address bar displays "localhost:8080/views/newlocation.jsp". The main content area has a label "Location name:" in bold. Below the label is a single-line text input field. Underneath the input field is a blue button with the text "Submit" in white.

Рисунок 3.8 – Окно для добавления нового места конференций

Если пользователю при заполнении поля “Publisher” в окне “New Book” система не предложила подсказки, значит конкретный издатель отсутствует в базе данных. Для внесения нового издателя необходимо нажать на кнопку “New” рядом с полем. После этого откроется новое окно, представленное на рисунке 3.9. Пользователь заполнит информацию о новом издателе. Затем необходимо подтвердить отправку данных на сервер кнопкой “Submit”.

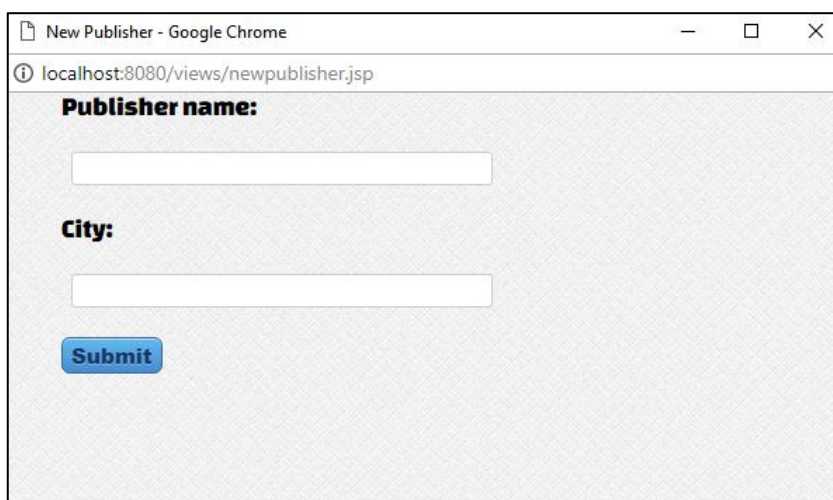
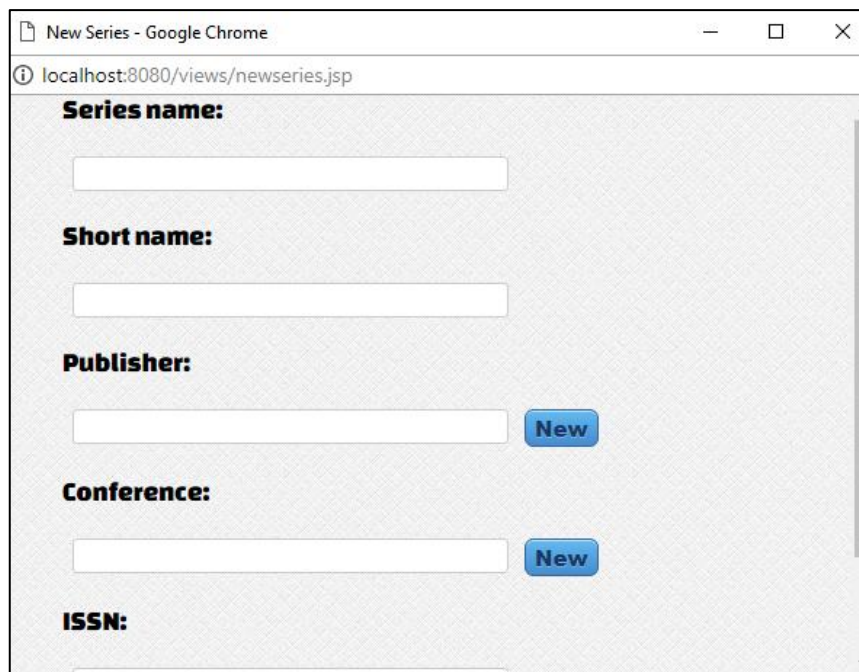
The screenshot shows a web browser window titled "New Publisher - Google Chrome". The address bar displays "localhost:8080/views/newpublisher.jsp". The main content area has a label "Publisher name:" in bold, followed by a single-line text input field. Below this is a label "City:" in bold, followed by another single-line text input field. At the bottom of the form is a blue button with the text "Submit" in white.

Рисунок 3.9 – Окно для добавления нового издателя



Если пользователю при заполнении поля “Series” в окне “New Book” система не предложила подсказки, значит искомая серия изданий отсутствует в базе данных. Для внесения информации о новой серии необходимо нажать на кнопку “New” рядом с полем. После этого откроется новое окно, представленное на рисунке 3.10. Пользователь заполнит данными элементы. Затем необходимо подтвердить отправку данных на сервер кнопкой “Submit”.



The image shows a web browser window with the title "New Series - Google Chrome". The address bar displays "localhost:8080/views/newseries.jsp". The main content area contains a form with the following elements:

- Series name:** A text input field.
- Short name:** A text input field.
- Publisher:** A text input field with a blue button labeled "New" to its right.
- Conference:** A text input field with a blue button labeled "New" to its right.
- ISSN:** A text input field.

Рисунок 3.10 – Окно для добавления нового издателя

Как видно, из этого окна также можно получить доступ к окнам нового издателя или новой конференции.

Если пользователю захочет внести новую систему индексирования, то в окне “New Series” необходимо нажать на кнопку “New” рядом с полем “Index System”. После этого откроется новое окно, представленное на рисунке 3.11. Пользователь заполнит данными элементы. Затем необходимо подтвердить отправку данных на сервер кнопкой “Submit”.

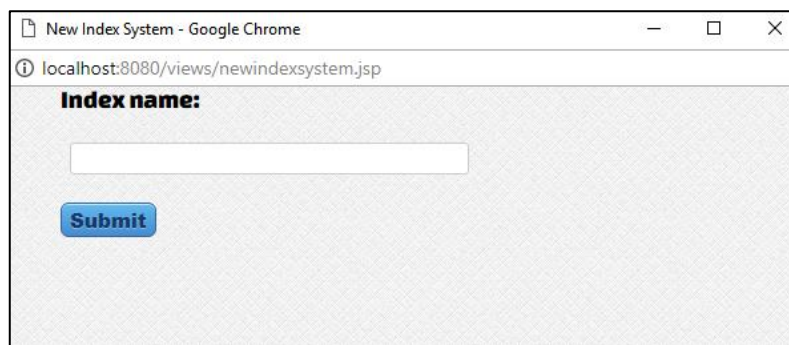


Рисунок 3.10 – Окно для добавления новой системы индексации

### 3.2 Используемые формы

В интерфейсе присутствует несколько форм, представляющие области для заполнения информации перед отправкой в базу данных.

1) Форма отправки данных новой или отредактированной записи о публикации. Содержит в себе элементы, заключенные в блоки `<div>`, такие как:

- Название публикации;
- Книга, связанная с этой публикацией;
- Язык добавляемого труда;
- Индекс DOI;
- Авторы;
- Места работы авторов;
- Благодарности, принадлежности к чему-либо;
- Ответственный за статью человек;
- Статус статьи.

Также присутствуют кнопки добавления новых строк ввода, к примеру, если необходимо добавить ещё одного известного системе автора, и кнопки на открытие нового окна для внесения новой информации в систему, к примеру, внести данные о новом авторе.

По завершению заполнения формы происходит подтверждение кнопкой “Submit”, данные собираются со всей формы и отправляются на сервер.

2) Форма для занесения параметров нового автора. Содержит в себе элементы, оформленные аналогично предыдущей форме:

- Фамилия автора;
- Имя автора;
- Отчество автора (при наличии);
- Гражданство автора.

По завершению заполнения формы происходит подтверждение кнопкой “Submit”, данные собираются со всей формы и отправляются на сервер.

3) Форма для занесения информации о новой благодарности. Содержит в себе элементы, оформленные аналогично предыдущей форме:

- Код благодарности;
- Комментарии, описание благодарности.

По завершению заполнения формы происходит подтверждение кнопкой “Submit”, данные собираются со всей формы и отправляются на сервер.

4) Форма для занесения информации о новом издании. Содержит в себе элементы, оформленные аналогично предыдущей форме:

- Название издания;
- Редактор издания;
- Год издания;
- Международный стандартный книжный код;
- Название конференции;
- Название серии (если издание периодическое);
- Номер тома (если издание периодическое);
- Номер выпуска (если издание периодическое);
- Издатель (если издание не периодическое).

По завершению заполнения формы происходит подтверждение кнопкой “Submit”, данные собираются со всей формы и отправляются на сервер.

5) Форма для занесения информации о новой конференции. Содержит в себе элементы, оформленные аналогично предыдущей форме:

- Название конференции;
- Даты начала и окончания;
- Короткое название;
- Место проведения;
- Комментарии.

По завершению заполнения формы происходит подтверждение кнопкой “Submit”, данные собираются со всей формы и отправляются на сервер.

6) Форма для занесения информации о новой серии изданий. Содержит в себе элементы, оформленные аналогично предыдущей форме:

- Название серии;
- Короткое название;
- Издатель серии;
- Название конференции;
- Международный стандартный серийный номер;
- Система индексации, проиндексировавшая эту серию;
- Рейтинг по двум оценкам на один год.

По завершению заполнения формы происходит подтверждение кнопкой “Submit”, данные собираются со всей формы и отправляются на сервер.

### **3.3 Сценарии выполняемые системой**

Сценарии системы содержатся в качестве сервлетов в Apache Tomcat на языке JavaScript. Сервлет принимает запрос пользователя вместе с данными о нем из браузера в виде объекта, реализующего интерфейс `javax.servlet.http.HttpServletRequest`, сгенерированный процессором сервлета. Сервлеты расширяющий интерфейс `javax.servlet.GenericServlet` или реализующий интерфейс `javax.servlet.Servlet`, получают объект `ServletRequest`. Однако, JSP-страницы каждый раз принимают объект класса `HttpServletRequest` с названием `request`, потому что JSP-страницы используется исключительно в

контексте HTTP. Список сервлетов, реализующих необходимые функциональные возможности:

- AutoPublication – соединяется с базой данных, посылает SQL запрос, в ответе для интерфейса посылает результаты запроса, в частности, информацию о всех публикациях, оформленные в HTML теги таблицы, в виде текста, что позволяет встроиться в код странички. Необходим для главной страницы.
- SearchBook – соединяется с базой данных, посылает SQL запрос с встроенными данными, пришедшими от интерфейса, в ответ высылает JSON строку с названиями всех книг, схожих с полученным параметром.
- AuthorList – аналогично с предыдущим. Результатами является JSON строка с фамилиями авторов, начинающихся на полученный параметр.
- AffiliationList – аналогично с предыдущим. Результатами является JSON строка с кодами благодарностей(причастностей), схожих с полученным параметром.
- AcknowList – аналогично с предыдущим. Результатами является JSON строка с местами работы, схожих с полученным параметром.
- NewAuthor – принимаются параметры о новом авторе, происходит соединение с базой данных. Составляется SQL запрос для внесения данных в таблицу. Результатом будет сообщение об успешности или неудачи запроса.

Код основных сценариев представлен в приложении А.

### **3.4 Удобство работы**

Ранней версией этой системы была попытка оформить интерфейс в базе данных Microsoft Office Access. Плюсом этой версии была простота и необходимость только одного инструмента. Недостатков было заметно больше. Во-первых, недостаток удобства при заполнении данных, особенно при

добавлении авторов и мест их работы. Во-вторых, неочевидность определенных действий, как связи мест работ с авторами при заполнении данных полей. В-третьих, главное функциональное неудобство было в необходимости обновления всей формы для ввода при добавлении новой, связанной с главной формой, информации в базу данных о книге, авторе или прочем. Что приводило к необходимости разработки новой системы, которая обеспечит комфортность в работе и добавит необходимые особенности функционирования.

### **Вывод**

Было рассмотрено, что система реализует все необходимые сценарии. Формы, присутствующие в интерфейсе обеспечивают необходимое удобство. Сценарии сервера, в том числе, поддерживают необходимую работоспособность и связь базы данных с интерфейсом.

## **Глава 4. Технико-экономическое обоснование**

### **4.1 Этапы технико-экономического обоснования**

Целью технико-экономического обоснования является определение экономической целесообразности реализации проекта.

Этапами ТЭО являются:

1. Составить подробный план, график выполнения работ;
2. Оценить затраты на заработные платы и социальные отчисления участникам исследования;
3. Оценить затраты, связанные с приобретением необходимого сырья и материалов;
4. Оценить затраты, связанные с услугами, оказанные сторонними организациями;
5. Оценить затраты, связанные с содержанием и эксплуатацией оборудования;
6. Определить величину амортизационных отчислений;
7. Оценить накладные расходы

Целью проекта является разработка web-интерфейса для базы данных библиографического содержания. Разработка проводится на ноутбуке инженера при помощи программного обеспечения, находящегося в свободном доступе. Вся деятельность сосредоточена в написании кода web-приложения для запросов к БД и не предполагает иных видов выражения итога, кроме как демонстрация работоспособности интерфейса.

## 4.2 Трудоемкость выполнения работ

Расчет расходов на выполнение работ начинается с вычисления длительности каждого этапа, которые необходимо завершить. Для определения продолжительности работы будем руководствоваться либо по факту, либо расчетным путем с помощью вычисляемых оценок по формуле:

$$t_j^0 = \frac{3t_{\min} + 2t_{\max}}{5} (1),$$

где  $t_j^0$  - ожидаемая длительность j-й работы;  $t_{\min}$  и  $t_{\max}$  - наименьшая и наибольшая по мнению эксперта длительность работы. Данные расчетов представлены в таблице 1.

Таблица 7.1 – Перечень выполняемых работ и их длительность

№ этапа	Наименование работ	Длительность, чел.-дни.t		
		$t_{\min}$	$t_{\max}$	$t_0$
1	Разработка и составление ТЗ.	2	7	4
	Работа с источниками.	1	3	1,8
2	Разработка методов решения задачи.  Выбор языка программирования.	1	3	1,8
5	Создание БД	2	6	3,6
6	Модификация БД под разрабатываемое приложение	1	5	2,6
7	Разработка интерфейса web-приложения для библиографического интерфейса.	7	14	9,8



8	Разработка сервера web-приложения.	4	7	5,2
9	Тестирование и отладка программного обеспечения.	2	7	4
10	Составление технической документации.	1	5	2,6
11	Оформление ВКР.	4	10	6,4
12	Оформление пояснительной записки	1	4	2,2
Итого:		26	71	44

Таким образом, общее расчетное время выполнения НИР составляет 44 чел. – дней.

Для вычисления ставки заработной платы за единицу времени, в частности, за человеко-дни, исходя из месячной заработной платы соответствующего исполнителя, необходимо разделить заработную плату (оклад) за месяц на количество рабочих дней в месяце (21 рабочий день) или разделить заработную плату (оклад) за месяц на количество рабочих часов в месяце (21 рабочий день x 8 часов = 168 часов), если принята другая мера времени. Данные расчетов представлены в таблице 2.

Таблица 7.2 – Распределение времени по исполнителям

№ этапа	Наименование работ	Исполнитель	Длительность, $t_0$ , чел.-дни.t	Ставка, руб./ед.t
1	Разработка и составление ТЗ.	Научный руководитель	4	952

	Работа с источниками.	Инженер	1,8	1428,5
2	Разработка методов решения задачи.  Выбор языка программирования.	Инженер	1,8	1428,5
5	Создание БД	Научный руководитель	3,6	952
6	Модификация БД под разрабатываемое приложение	Инженер	2,6	1428,5
7	Разработка интерфейса web-приложения для библиографического интерфейса.	Инженер	9,8	1428,5
8	Разработка сервера web-приложения.	Инженер	5,2	1428,5
9	Тестирование и отладка программного обеспечения.	Инженер	4	1428,5
10	Составление технической документации.	Инженер	2,6	1428,5

11	Оформление ВКР.	Инженер	6,4	1428,5
12	Оформление пояснительной записки	Инженер	2,2	1428,5
Всего времени:		Инженер	36,4	1428,5
		Научный руководитель	7,6	952

Будем полагать, что зарплата инженера 30 000 р., а научного сотрудника

20 000 р.. Их ставки соответственно  $\frac{30000}{21\text{день}} = 1428,5$  р/чел.-дней и  $\frac{20000}{21\text{день}} = 952$  р/чел.-дней.

На основе полученных расчетов необходимо определить расходы на заработную плату исполнителей и отчислений на страховые взносы на обязательное социальное, пенсионное и медицинское страхование. Вычислим по формуле:

$$Z_{оснзн} = \sum_{i=1}^k T_i * C_i, (2)$$

где  $Z_{оснзн}$  - расходы на основную заработную плату исполнителей (руб.);  $k$  – количество исполнителей;  $T_i$  - время, затраченное  $i$ -м исполнителем на проведение исследования (дни или часы);  $C_i$  - ставка  $i$ -го исполнителя (руб./день или руб./час).

$$Z_{оснзн} = (4 + 3,6) * 952 + (1,8 + 1,8 + 2,6 + 9,8 + 5,2 + 4 + 2,6 + 6,4 + 2,2) * 1428,5 = 59232,6 \text{ р.}$$

Расходы на дополнительную заработную плату исполнителей

$$Z_{допзн} = Z_{оснзн} \frac{H_{доп}}{100}, (3)$$

где  $Z_{допзн}$  - расходы на дополнительную заработную плату исполнителей (руб.);  $Z_{оснзн}$  - расходы на основную заработную плату исполнителей (руб.);  $H_{доп}$  -

норматив дополнительной заработной платы (%). При выполнении расчетов норматив дополнительной заработной платы принимаем равным 14%.

$$З_{допзн} = 59232,6 * 0,14 = 8292,6 \text{ р.}$$

Отчисления на страховые взносы на обязательное социальное, пенсионное и медицинское страхование с основной и дополнительной заработной платы исполнителей рассчитаем по формуле:

$$З_{соц} = (З_{оснзн} + З_{допзн}) \frac{Н_{соц}}{100}, (4)$$

где  $З_{соц}$  - отчисления на социальные нужды с заработной платы (руб.);  $Н_{соц}$  - норматив отчислений на страховые взносы на обязательное социальное, пенсионное и медицинское страхование (%).

$$З_{соц} = (59232,6 + 8292,6) * 0,3 = 20257,5 \text{ р.}$$

Вычисление стоимости разрабатываемого программного обеспечения будем проводить по следующим статьям:

- сырье и материалы;
- затраты по работам, выполняемыми сторонними организациями;
- амортизационные отчисления;
- накладные расходы;
- спецоборудование.

Далее приведем подробные вычисления, приведенных статей расходов, в определенном выше порядке.

### 4.3 Сырье и материалы

В данном расчете на статью “Сырье и материалы” относятся расходы на основные и вспомогательные материалы, покупные полуфабрикаты и комплектующие изделия, использованные при выполнении разработки. Оценка потребности в ресурсах устанавливается в натуральных и стоимостных показателях. Затраты по данной статье приведены в таблице 3.

Таблица 7.3 – Затраты по статье “материалы”

Материалы	Единица измерения	Количество	Цена, руб.	Сумма, руб.
Бумага писчая	Пачка	1	202-00	202-00
Картридж для принтера	шт.	1	1100-00	1100-00
Фотобарабан для принтера	шт.	1	792-00	792-00
Компакт диск CD-R	шт.	2	25-00	50-00
Флэш-накопитель	шт.	1	262-00	262-00
Итого				2 406-00
Транспортные расходы, 10%				240-00
Всего:				2 646-00

При проведении расчетов норму транспортно-заготовительных расходов примем равной 10%.

Проведем расчет нормативным методом для данной статьи расходов. Производится по следующей формуле:

$$З_{\text{м}} = \sum_{i=1}^L G_i C_i (1 + \frac{H_{\text{м.з.}}}{100}), (5)$$

где  $З_m$  – затраты на сырье и материалы (руб.); L – индекс вида сырья или материала;  $G_i$  – норма расхода l-того материала на единицу продукции (ед.);  $Ц_i$  – цена приобретения единицы l-го материала (руб./ед.);  $H_{м.з}$  – норма транспортно-заготовительных расходов (%).

$$З_m = (202 + 1100 + 792 + 50 + 262) * 1.1 = 2646,6 \text{ р.}$$

Затрат на работы, выполняемые сторонними организациями нет.

#### 4.4 Амортизационные отчисления

Основные средства, использованные при разработке, были: ноутбук, принтер. Амортизационные отчисления по основному средству i за год определяются как:

$$A_i = Ц_{н.н.i} \frac{H_{ai}}{100}, (6)$$

где  $A_i$  – амортизационные отчисления за год по i-му основному средству (руб.);  $Ц_{н.н.i}$  – первоначальная стоимость i-го основного средства (руб.);  $H_{ai}$  – годовая норма амортизации i-го основного средства (%).

Таблица 7.4 – Список основных средств

Основные средства	Первоначальная стоимость, руб.	Норма амортизации, %
Ноутбук HP Pavilion m6	30 000-00	16
Принтер HP LaserJet 1100	2 500-00	10

$$A_1 = 30000 * 0,16 = 4800 \text{ р.}$$

$$A_2 = 2500 * 0,1 = 250 \text{ р.}$$

Для определения отчислений для ВКР определим время использования основных средств. Величина амортизационных отчислений по i-му основному

средству, используемому студентом при работе над ВКР, определяется по формуле:

$$A_{iBKP} = A_i \frac{T_{iBKP}}{12}, (7)$$

где  $A_{iBKP}$  - амортизационные отчисления по  $i$ -му основному средству, используемому студентом в работе над ВКР (руб.);  $A_i$  - амортизационные отчисления за год по  $i$ -му основному средству (руб.);  $T_{iBKP}$  - время, в течение которого студент использует  $i$ -ое основное средство (мес.).

$$A_{1BKP} = 4800 * \frac{2}{12} = 800 \text{ р.} \quad A_{2BKP} = 250 * \frac{1}{12} = 21 \text{ р.}$$

$$A_{\Sigma BKP} = 800 + 21 = 821 \text{ р.}$$

#### 4.5 Накладные расходы

В эту статью включаются расходы на управление и хозяйственное обслуживание проектной команды. [9]

$$C_{HP} = \frac{C_{30} \cdot H_{HP}}{100}, (8)$$

где  $H_{HP}$  - норматив накладных расходов, составляет 40%,  $C_{30}$  - расходы на оплату труда составляют 67525,2 р.

$$C_{HP} = \frac{67525,2 \cdot 40}{100} = 27010 \text{ р.}$$

Дополнительное спецоборудование не требуется.

#### 4.6 Совокупные расходы на ВКР

Оформим смету, основанную на затратах, по статьям расходов, приведенных выше, представив в таблице 5.

Таблица 7.5 – Смета затрат

№	Статья затрат	Сумма, руб.
1	Расходы на оплату труда	67525-20
2	Отчисления на социальные нужды	20257-50
3	Сырье и материалы	2646-00
4	Затраты по работам, выполняемым сторонними организациями	0-00
5	Амортизационные отчисления	821-00
6	Накладные расходы	27010-00
7	Спецоборудование	0-00
Итого себестоимость разработки		118259-70

Итоговые затраты составляют 118259-70 руб. Подсчитаем сколько процентов по каждой статье расходов входят в этот итог.

- Статья расходов “Расходы на оплату труда” составляет 57%
- Статья расходов “Отчисления на социальные нужды” составляет 17,1%
- Статья расходов “Сырье и материалы” составляет 2,2%
- Статья расходов “Затраты по работам, выполняемым сторонними организациями” составляет 0%
- Статья расходов “Амортизационные отчисления” составляет 0,7%
- Статья расходов “Накладные расходы” составляет 22,8%
- Статья расходов “Спецоборудование” составляет 0%

Самая затратная статья — это расходы на заработную плату персоналу, участвовавшему в выполнении работы.



## **Вывод**

В настоящей главе рассмотрены вопросы организационно-экономического обоснования процесса разработки автоматизированной системы для построения и визуализации модели глаза человека. Приведена оценка единовременных затрат в процессе разработки. Полная величина затрат составила 188153 рублей, что определяет себестоимость полученного программного продукта. Большая часть от этой суммы приходится на расходы по оплате труда (53,2%) и на накладные расходы (28,6%). Оставшаяся часть суммы затрат приходится на отчисления на социальные нужды (16%), амортизационные отчисления (1,7%) и материальные затраты (0,5%).

## **Заключение**

В процессе выполнения ВКР были достигнуты следующие результаты:

- Разработан удобный интерфейс для занесения информации о научной публикации.
- Создана специально подготовленная для публикаций база данных.
- Предоставлена возможность по редактированию базы данных через разработанный интерфейс системы ввода и учета.
- Присутствуют возможности интеграции и улучшения системы для добавления новых алгоритмов фильтрации, а также анализа информации.

Программный продукт позволяет комфортно заполнять и редактировать базу данных. В целом, поставленные в работе задачи были выполнены.

## Список литературы

1. О проекте [Электронный ресурс] // eLibrary. URL: [https://elibrary.ru/elibrary\\_about.asp](https://elibrary.ru/elibrary_about.asp)
2. Советов Б.Я., Цехановский В.В., Чертовской В.Д. Базы данных: теория и практика. Учебник М.: «Юрайт», 2014. – 464 с. Гриф УМО по университетскому политехническому образованию.
3. Голицына О.Л. Базы данных: Учеб. пособие для студ. сред. проф. Образования/ О.Л. Голицына, Н.В. Максимов, И.И. Попов. – М.: ФОРУМ: ИНФРА-М, 2004
4. Д. Рейсиг. Инструменты отладки и тестирования // JavaScript. Профессиональные приёмы программирования - Pro JavaScript™ Techniques / Перевод Н. Вильчинский. – СПб.: Питер, 2008. –(Библиотека программиста)– С. 76.
5. С. Давыдов, А. Ефимов. IntelliJ IDEA. Профессиональное программирование на Java. — СПб.: БХВ, 2005. — 800 с.
6. JSON: основы использования [Электронный ресурс] // Ruseller.com URL: <http://ruseller.com/lessons.php?rub=28&id=1212>
7. Zakas N. 1. What is JavaScript? // Professional JavaScript for Web Developers. – 2nd ed. – USA, Canada: Wiley Publishing, Inc., 2009.
8. Энциклопедия языков программирования [Электронный ресурс] // Progopedia alpha. URL: <http://progopedia.ru/language/c-plus-plus/>
9. Алексеева О.Г. Методические указания по экономическому обоснованию выпускных квалификационных работ бакалавров: Метод. указания, СПб.: Изд-во СПбГЭТУ “ЛЭТИ”, 2013. – 17 с.

## Приложение А

### Листинг основных сценариев, исполняемых системой

```
/* Сервлет AutoPublication */

import java.io.IOException;

import java.sql.*;

import java.util.Properties;

import javax.servlet.ServletConfig;

import javax.servlet.ServletContext;

import javax.servlet.ServletException;

import javax.servlet.http.HttpServlet;

import javax.servlet.http.HttpServletRequest;

import javax.servlet.http.HttpServletResponse;


public class AutoPublication extends HttpServlet {

    private ServletContext context;

    private Connection connection;

    @Override

    public void init(ServletConfig config) throws ServletException {

        context = config.getServletContext();

    }

    @Override

    public void doGet(HttpServletRequest request, HttpServletResponse response)

        throws IOException, ServletException {

        try{

            Class.forName("org.postgresql.Driver");
```

```

Properties prop=new Properties();

prop.setProperty("user","postgres");

prop.setProperty("password","postgrespass");

prop.setProperty("characterEncoding","UTF8");

connection =
DriverManager.getConnection("jdbc:postgresql://localhost:5432/dbstate",prop);

} catch (SQLException e ){

    System.out.println(e.getMessage());

} catch ( ClassNotFoundException e) {

    e.printStackTrace();

}

StringBuilder sb = new StringBuilder();

String action = request.getParameter("val");

String selectTableSQL = "SELECT publication.id, Publication.Titile, Book.Name,
Series.SeriesName , Book.BookYear, statusdictionary.description " +

    "FROM ((Publication INNER JOIN StatusDictionary ON StatusDictionary.id =
Publication.Status) " +

    " INNER JOIN Book ON Book.id = Publication.Book) LEFT JOIN Series ON Series.id
= Book.Series " +

    //"WHERE Status = 2 " + //"and Book.bookyear =" + action +

    " ORDER BY 1";

Statement statement = null;

try {

    statement = connection.createStatement();

    // выбираем данные с БД

    ResultSet rs = statement.executeQuery(selectTableSQL);

```

```

sb.append("<tr>");

sb.append("<th>Title article</th>");

sb.append("<th>Name Book</th>");

sb.append("<th>Series</th>");

sb.append("<th>Book Year</th>");

sb.append("<th>Status</th>");

sb.append("</tr>");

while (rs.next()) {

    String Title = rs.getString("Titile");

    String stat = rs.getString("description");

    String name = rs.getString("Name");

    if(name == null)

        name = "";

    String series = rs.getString("SeriesName");

    if(null == series)

        series = "";

    String year = rs.getString("BookYear");

    sb.append("<tr>");

    sb.append("<td>" + Title + "</td>");

    sb.append("<td>" + name + "</td>");

    sb.append("<td>" + series + "</td>");

    sb.append("<td>" + year + "</td>");

    sb.append("<td>" + stat + "</td>");

    sb.append("<td> <button type=\"button\" value=\"" + rs.getString("id") + "\" "
name=\"Entity[Art" + rs.getString("id") + "]\"\" +

```

```

        " id=\"Entity_art"+rs.getString("id")+\" \"
onclick=\"editarticle(\"+rs.getString("id")+\")\" >Edit</button></td>");

        sb.append("</tr>");

    }

}

catch (SQLException e){

    System.out.println(e.getMessage());

}

if (connection != null) {

    try {

        connection.close();

    } catch (SQLException ex) {

        System.out.println(ex.getMessage());

    }

}

request.setCharacterEncoding("CP1251");

response.setContentType("text/html; charset=windows-1251");

response.setHeader("Cache-Control", "no-cache");

response.getWriter().write("<table>" + sb.toString() + "</table>");

}

protected void doPost(HttpServletRequest request, HttpServletResponse response)

    throws ServletException, IOException {

}

}

/* Сервлет NewPublication */

```

```

public class NewPublicat extends HttpServlet {

    private Connection connection;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)

        throws ServletException, IOException {

    try {

        Class.forName("org.postgresql.Driver");

        Properties prop = new Properties();

        prop.setProperty("user", "postgres");

        prop.setProperty("password", "postgrespass");

        prop.setProperty("characterEncoding", "UTF8");

        connection = DriverManager.getConnection("jdbc:postgresql://localhost:5432/dbstate",
prop);

    } catch (SQLException e) {

        System.out.println(e.getMessage());

    } catch (ClassNotFoundException e) {

        e.printStackTrace();

    }

    StringBuilder sb = new StringBuilder();

    BufferedReader reader = request.getReader();

    try {

        String line;

        while ((line = reader.readLine()) != null) {

            sb.append(line).append("\n");

        }

    }

```



```

    } finally {

        reader.close();

    }

    JSONObject jsond = new JSONObject(sb.toString());

    System.out.println(jsond.get("title"));

    String insertpubl = "INSERT INTO publication (titile, book, decisiondate, status, lang,
startpage, endpage, path, responsible, comment, doi)" +

        " VALUES ("'+jsond.get("title")
+''',findbook("'+jsond.get("book")+''"),'''+jsond.get("date")+'''," +
""'+jsond.getInt("status")+''",'''+jsond.getInt("lang")+''",'''+jsond.getInt("pageStart")+''",'''+jsond.getI
nt("pageEnd")+'''," +
""'+jsond.get("Pathfile")+''',findresp("'+jsond.get("responsible")+''"),'''+jsond.get("commentpubl")+''
',''+jsond.get("doi")+''")";

    Statement statement = null;

    try {

        statement = connection.createStatement();

        boolean rs = statement.execute(insertpubl);

        System.out.println(jsond.getJSONArray("authors").length());

        for(int i =0;i < jsond.getJSONArray("authors").length();i++)

        {   String auth = jsond.getJSONArray("authors").getString(i);

            String insertauth = "INSERT INTO authorpublication (publication, author)" +

                " VALUES (findidpubl("'+jsond.get("title") +'''),findresp("'+auth+'''))";

            boolean rs2 = statement.execute(insertauth);

            for(int j=0;j < jsond.getJSONArray("jobs").getJSONArray(i).length(); j++){

                //System.out.println(i+" "+j+":

"+jsond.getJSONArray("jobs").getJSONArray(i).get(j).equals(null));

                if(!jsond.getJSONArray("jobs").getJSONArray(i).get(j).equals(null)) {

```

```

        Integer aff =jsond.getJSONArray("jobs").getJSONArray(i).getInt(j);

        System.out.println(aff);

        String insertaff = "INSERT INTO authorpublicationaffiliation (authorpublication,
affiliation)" +

                " VALUES (findidauthorpubl(findidpubl(""+jsond.get("title")
+""),findresp(""+auth+"")),"" + aff + "")";

        boolean rs3 = statement.execute(insertaff);

        }    }

    }

    for(int i =0;i < jsond.getJSONArray("acknow").length();i++)

    {

        String acknow = jsond.getJSONArray("acknow").getString(i);

        String insertackn = "INSERT INTO publicationacknowledgement (publication,
acknowledgement)" +

                " VALUES (findidpubl(""+jsond.get("title") +""),findidacknow(""+acknow+""))";

        boolean rs4 = statement.execute(insertackn);

        }    }

    catch (SQLException e){

        System.out.println(e.getMessage()+"1");

    }

    if (connection != null) {

        try {

            connection.close();

        } catch (SQLException ex) {

            System.out.println(ex.getMessage());

        }

    }

```

```

    }

    response.setContentType("text/html; charset=windows-1251");

    request.setCharacterEncoding("CP1251");

    response.setHeader("Cache-Control", "no-cache");

    response.getWriter().write("ok");

}

@Override

public void doGet (HttpServletRequest request, HttpServletResponse response)

    throws IOException, ServletException {

}}

/*Сервлет AuthorList */

public class AuthorList extends HttpServlet{

    private Connection connection;

    protected void doPost(HttpServletRequest request, HttpServletResponse response)

        throws ServletException, IOException {

    }

    @Override

    public void doGet(HttpServletRequest request, HttpServletResponse response)

        throws IOException, ServletException {

        try{

            Class.forName("org.postgresql.Driver");

            Properties prop=new Properties();

            prop.setProperty("user","postgres");

            prop.setProperty("password","postgrespass");

            prop.setProperty("characterEncoding","UTF8");

```

```

        connection =
DriverManager.getConnection("jdbc:postgresql://localhost:5432/dbstate",prop);

    }catch (SQLException e ){

        System.out.println(e.getMessage());

    } catch ( ClassNotFoundException e) {

        e.printStackTrace();

    }

    String action = request.getParameter("term");

    String selectTableSQL = "SELECT author.authorname FROM author WHERE
lower(authorname) LIKE lower('%" + action+"%') ";

    System.out.println("Data from ajax call Author " + action);

    JSONArray arrayObj = new JSONArray();

    Statement statement = null;

    try {

        statement = connection.createStatement();

        ResultSet rs = statement.executeQuery(selectTableSQL);

        while (rs.next()) {

            String name = rs.getString("authorname");

            arrayObj.put(name);

        }

    }

    catch (SQLException e){

        System.out.println(e.getMessage());

    }

    if (connection != null) {

        try {

```

```
        connection.close();

    } catch (SQLException ex) {

        System.out.println(ex.getMessage());

    }

}

response.setContentType("text/html; charset=windows-1251");

request.setCharacterEncoding("CP1251");

response.setHeader("Cache-Control", "no-cache");

response.getWriter().write(arrayObj.toString());

}}
```