Известия высших учебных заведений. Материалы электронной техники. 2020. Т. 23, № 4. С. 297—303.
DOI: 10.17073/1609-3577-2020-4-297-303

УДК 621.315:004.925.8

Программные средства для индексирования и предметно–ориентированного поиска научных статей по кристаллографии

© *2020 г.* **Е. С. Гаврилов[1,2,§], Н. А. Зайцев[3]**

[1] ***Московский авиационный институт (национальный исследовательский университет),***
Волоколамское шоссе, д. 4, Москва, 125993, Россия

[2] ***Федеральный исследовательский центр «Информатика и управление» Российской академии наук,***
ул. Вавилова, д. 44, корп. 2, Москва, 119333, Россия

[3] ***ПАО «Сбербанк»,*** *ул. Вавилова, д. 19, Москва, 117312, Россия*

Аннотация. Представлены программные модули, реализующие индексацию научных статей, предметно–ориентированный поиск по ним, пользовательский интерфейс, а также программный интерфейс для сторонних потребителей. Приведено подробное описание процесса настройки полнотекстового поиска до уровня предметно–ориентированного.

Ключевые слова: полнотекстовый поиск, предметно–ориентированный поиск, инвертированный индекс, поисковая система Elasticsearch, язык Java, Фреймворк Spring, язык JavaScript, Фреймворк React

Введение

Одним из основных источников данных при подготовке вычислительных экспериментов являются научные статьи, из которых берутся свойства материалов, а также результаты экспериментов. В работе представлены программные средства, позволяющих исследователям–физикам в области материаловедения быстро находить тематические статьи по заданным предметно–ориентированным критериям поиска. Например, задав название материала, или химическую формулу, тип кристаллической решетки и метод расчета — получить список статей, где они упоминаются вместе или отдельно, упорядоченных по релевантности.

В статье решается задача поиска научных публикаций, необходимых для научных исследований в области вычислительного материаловедения. Для этого были разработаны программные модули для индексации научных статей и полнотекстового поиска по ним, настроен поисковый индекс для предметно–ориентированного поиска и реализован пользовательский интерфейс. Также был реализован программный интерфейс **(API)** для полученного поискового сервиса.

Сценарии использования

Пользователем системы является ученый–исследователь в области вычислительного материаловедения. Пользователь может производить поиск публикаций как по произвольному запросу, так и по предметно–ориентированному. С целью поиска входных данных для расчетов, результатов экспериментов, литературы для цитирования. Реализованный программный поисковый сервис подразумевает возможность интеграции с внешними информационными системами посредством предоставления своего API.

Полнотекстовой поиск

Автоматизированный поиск документов, при котором поиск ведется не только по именам документов, но и по их содержимому называется полнотекстовым поиском. В основе полнотекстового поиска лежит структура данных — инвертированный индекс.

Гаврилов Евгений Сергеевич[1,2,§] — старший преподаватель (1), научный сотрудник (2), e–mail: eugavrilov@gmail.com; **Зайцев Н. А.[3]** — разработчик, e–mail: JacksonGibsonESP@gmail.com

§ Автор для переписки

* Статья подготовлена по материалам доклада, представленного на II–й международной конференции «Математическое моделирование в материаловедении электронных компонентов», Москва, 19—21 октября 2020 г.

Добавление документа в индекс называется индексацией. Перед индексацией документа происходит его предобработка. Наиболее часто текст документа приводится к нижнему регистру, отбрасываются знаки пунктуации и текст бьется на слова (термы) по пробелам. Инвертированный индекс ставит в соответствие терм и номер документа в котором он встречается, и опционально позицию в нем. Что позволяет быстро находить вхождения терма в документы. В отличие от прямого индекса, где каждому документу ставится в соответствие набор термов, встречающихся в нем.

В качестве реализации инвертированного индекса используется поисковая система Elasticsearch [1]. Elasticsearch — это свободно распространяемая поисковая система, построенная на технологии Apache Lucene™ — свободной библиотеки для высокопроизводительного полнотекстового поиска. Elasticsearch написан на языке Java и использует Lucene внутри себя для индексации и поиска, но он стремится сделать полнотекстовый поиск простым и понятным для пользователя, скрывая сложности работы с Lucene за RESTful API.

Вывод результатов поиска (поисковая выдача) сортируется по релевантности (соответствию запросу пользователя).

Архитектура

На рис. 1 приведена диаграмма потоков данных, описывающая архитектуру реализованного поискового сервиса.

Первоначально на сервер помещаются все статьи, по которым необходимо производить полнотекстовый поиск. Затем, администратор запускает модуль индексации документов, написанный на языке Java и представляющий из себя консольное приложение. Это приложение считывает документы с диска и преобразовывает их в base64–представление. Далее приложение осуществляет отправку в поисковую систему Elasticsearch на индексацию, используя производительное Elasticsearch Java API.

Отдельно на сервере работает приложение, отвечающее за предоставление API реализованного поискового сервиса сторонним потребителям. В зависимости от поступающих к нему запросов производит либо запрос к Elasticsearch за поисковой выдачей, либо производит выгрузку потребителю исходного документа с диска.

Извне к поисковому сервису можно обратиться посредством предоставляемого API. Одним из потребителей данного программного интерфейса является пользовательский веб–интерфейс.

Модуль индексации документов. Для того, чтобы Elasticsearch мог проиндексировать наиболее часто встречающиеся форматы документов, такие как .pdf, .doc, .docx, .rtf, .htm, .html, .odt, был установлен плагин Ingest Attachment [2]. Для этого необходимо сконвертировать их в base64–формат.

Для добавления документа в индекс был настроен собственный пайплайн (*pipeline*). Пайплайн описывает порядок действий, предпринимаемый Elasticsearch при добавлении нового документа в индекс и состоит из обработчиков (*processors*), которые производят некоторые действия над документом или же описывают его конфигурацию.

Настройка обработчика с конфигурацией плагина Attachment, включала в себя указание имени

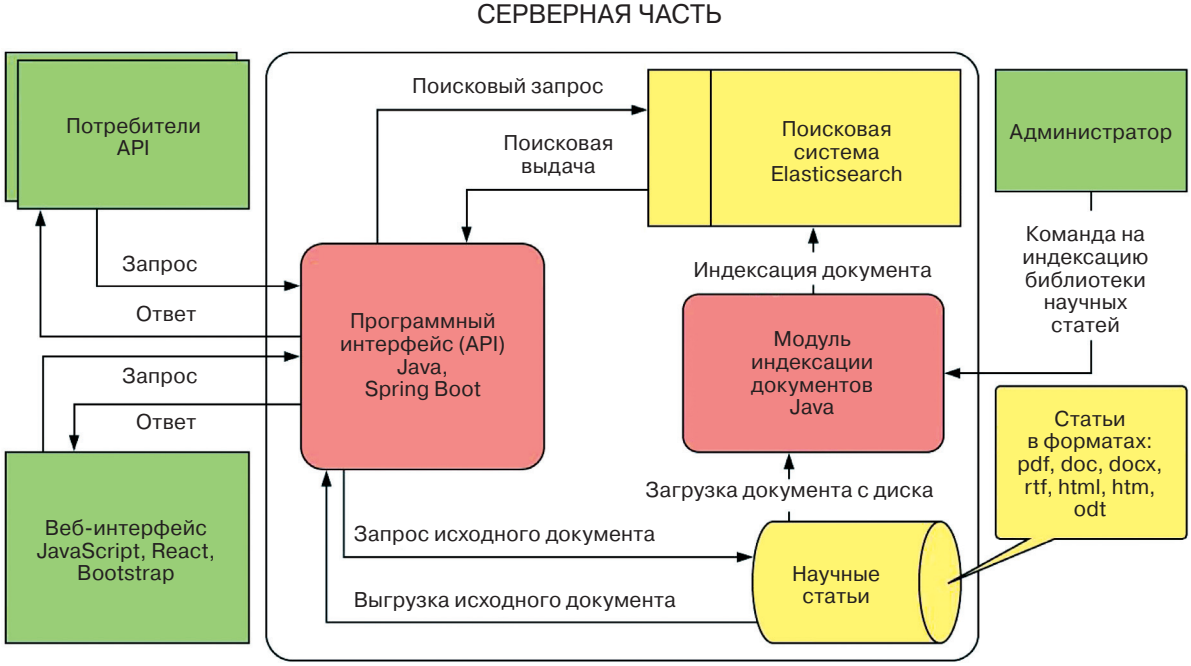

Рис. 1. Архитектура поискового сервиса
Fig. 1. Search service architecture

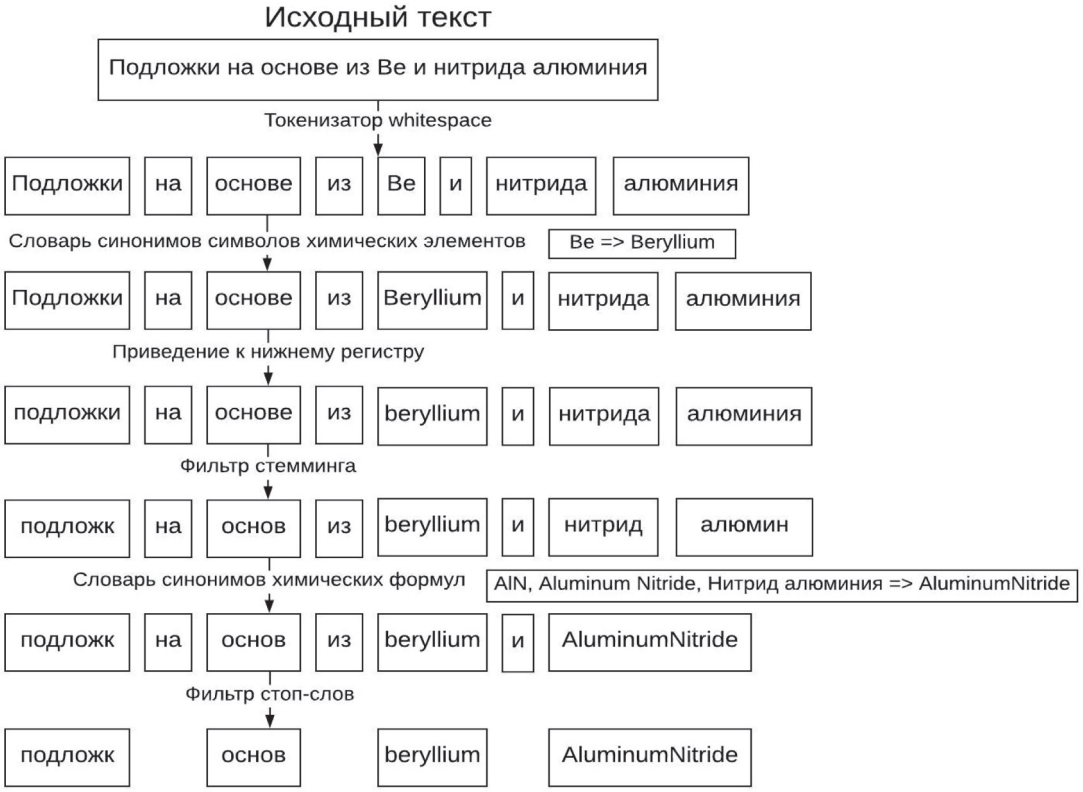

Рис. 2. Последовательность работы фильтров токенов
Fig. 2. Sequence of operation of token filters

поля, в котором находится base64–представление документа, а также снятие ограничений на количество символов в документе.

Помимо этого, для повышения производительности Elasticsearch, решено было уменьшить размер индекса, удалив поле data с base64–представлением исходного документа. Добавлен стандартный обработчик «remove» с именем поля, которое необходимо удалить из документа, после его добавления в индекс.

Таким образом поле «data» сначала обрабатывается плагином Ingest Attachment, чтобы извлечь из него текстовое представление документа, которое затем добавляется в индекс. После чего это поле удаляется. Таким образом объем индекса с 520 документами уменьшился с 2,4 Гб до 108 Мб, значительно повысив производительность.

Для анализа была взята готовая подборка статей по кристаллографии у представителя предметной области. Статьи проиндексировали приложением, получив набор из 594 документов, в основном на русском языке. Но среди них присутствует некоторое количество нерелевантных документов, состоящих исключительно из тэгов html–разметки. Большинство из этих файлов было удалено путем добавления условия на длину файлов — после индексации удаляются все документы не длиннее 1000 символов.

В Elasticsearch термы называются токенами. Был создан собственный анализатор токенов, с возможностью отсеивания стоп–слов, поиска без учета регистра, с использованием словаря синонимов, а также с учетом морфологии как английского, так и русского языков.

Данный анализатор «crystal» устроен следующим образом: используется токенайзер «whitespace», который разбивает текст на термы по пробелам. Затем указывается последовательность фильтров, которые могут модифицировать эти термы, — удалять их или добавлять новые. Причем очень важен порядок, в котором они указаны, т.к. они применяются к потоку термов последовательно, как при индексации, так и при поиске документа. Пример работы проиллюстрирован на рис. 2.

Первый фильтр — это словарь синонимов, который производит замену термов слева от символа «=>» на все термы, указанные справа от него. В этом словаре указаны символы химических элементов из периодической таблицы Менделеева. Каждая строчка ставит в соответствие символ химического элемента и его английское название, например, Be => Beryllium. Только затем применятся фильтр, приводящий термы к нижнему регистру. Это сделано затем, чтобы последующий за ними фильтр стоп–слов, не удалил приведенный к нижнему регистру терм be, который полностью совпадает с одним из стоп слов. Таким образом, сделав поисковый запрос по «Be» будут найдены все документы со вхождением данного слова с учетом регистра и его синонимов, а по запросу «be» отобразятся все документы в

индексе, что означает, что запрос был пустым, т. е. не содержал термов. Elasticsearch можно настроить так, чтобы он отображал пустую выдачу при пустом запросе, это поведение можно менять соответствующим параметром Java API.

Стоит сказать, что synonym–фильтр разбивает синонимы на токены при помощи токенайзера, объявленного в анализаторе, внутри которого он используется. Фильтрует же термы при помощи фильтров, объявленных до него в цепочке.

После этих двух фильтров, применяется третий — второй словарь синонимов, основной, который добавляет к русскоязычным написаниям химических элементов их англоязычные версии: Бериллий => Beryllium. Он содержит описание химических формул: AlN, Aluminum Nitride, Нитрид алюминия => AluminumNitride. Результирующий терм написан без пробела затем, чтобы избежать нерелевантного поиска, вызванного, по всей видимости, непредсказуемым циклическим слиянием правил замены термов. Также в этом словаре содержатся описания типов кристаллических систем и атомных радиусов: Кубическая, Cubic => Кубическая, Cubic, Ионный, Ionic => Ионный, Ionic.

Затем наступает очередь стоп–фильтров. Эти фильтры удаляют термы, содержащие предлоги, суффиксы, междометия, частицы и тому подобные часто встречающиеся термы, поиск по которым не будет повышать релевантность поиска. Используется два фильтра, один для русского языка, другой для английского. Эти фильтры поставляются вместе с Elasticsearch.

После наступает черед фильтров, реализующих стемминг (*stemming*) — это нахождение основы слова для заданного исходного слова. Основа необязательно совпадает с морфологическим корнем слова. Опять же, используется два стандартных фильтра для английского и русского языка. Пример работы стемминга: превращает слово «Подложка» в терм «подложк», что позволяет производить поиск, не зависящий от окончаний слова, входящего в запрос. Все слова «Подложка», «Подложкой», «Подложками» и т.д. будут порождать одинаковую поисковую выдачу. Теперь, Elasticsearch настроен и готов к работе.

Индексировать документы можно вручную, при помощи REST запросов к Elasticsearch, но это слишком неудобно, и поэтому была написана программа на языке Java с использованием Elasticsearch Java API. Это модуль был назван pdf_index.

При запуске программы происходит подключение к Elasticsearch, создание индекса «crystal» в случае, если его еще нет, иначе удаляется и создается заново, применяются настройки индекса, с указанием используемого анализатора токенов. Затем pdf_index считывает файлы из указанной пользователем директории, проверяет, что они относятся к одному из следующих форматов: .pdf, .doc, .docx, .rtf, .html, .htm или .odt. Файлы других форматов игнорируются. После проверки этого условия программа конвертирует файл в base64–представление, чтобы плагин Ingest Attachment смог распарсить[1] этот документ, и отправляет на индексацию в Elasticsearch. Затем эта строка помещается в поле «data» запроса. Также помимо этого поля, добавляется поле, содержащее дату добавления документа в индекс, имя документа и информация о пути, по которому он был найден на диске. Путь указывается для того, чтобы была возможность затем его выкачать с диска. Помимо прочего указывается имя пайплайна — «attachment», и задается id документа как md5 хеш–сумма от имени файла. Это необходимо для того, чтобы в индекс не добавлялись одинаковые документы, потому что если id не указать явно, то он будет сгенерирован случайным образом. Такой способ задания id был выбран также по тому, что его можно беспрепятственно использовать в составе url запроса к Elasticsearch.

Модуль, предоставляющий API. Чтобы изолировать фронтенд часть от особенностей работы с Elasticsearch, было создано приложение на Spring Boot [3] под названием crystal_proxy. Запуск данного приложения представляет из себя старт сервера приложений Apache Tomcat с загруженным в контейнер сервлетов скомпилированным Java приложением. Все это происходит автоматически и одной командой командной строки, что является одним из плюсов использования Spring Boot. Таким образом, приложение постоянно слушает заданный в конфигурационном файле порт, и принимает несколько видов REST запросов и обычных GET, делая исходя из них новые к Elasticsearch, получает ответ, и преобразует его в понятный для фронтенда вид. Это нельзя назвать проксированием в чистом виде, так как API crystal_proxy и Elasticsearch отличается. Далее следует описание API данного приложения:

- GET запрос по адресу /api/check должен вернуть следующий текст: «Hello! It's Crystal Proxy Application!». Этот запрос используется для проверки работоспособности приложения.
- GET запрос по адресу /api/getFileById с Id документа, переданным в атрибутах запроса. Произойдет загрузка документа на фронт.
- REST GET запрос по адресу /api/getContentById с Id документа, переданным в атрибутах запроса, вернет JSON с содержимым документа.
- REST GET запрос по адресу /api/ search с текстом запроса Query, типом химического элемента ChemicalElement, химической формулой ChemicalFormula, типом кристаллической решет-

[1] Парсинг — автоматизированный сбор и систематизация информации из открытых источников с помощью скриптов. Другое название этого процесса — веб-скрейпинг.

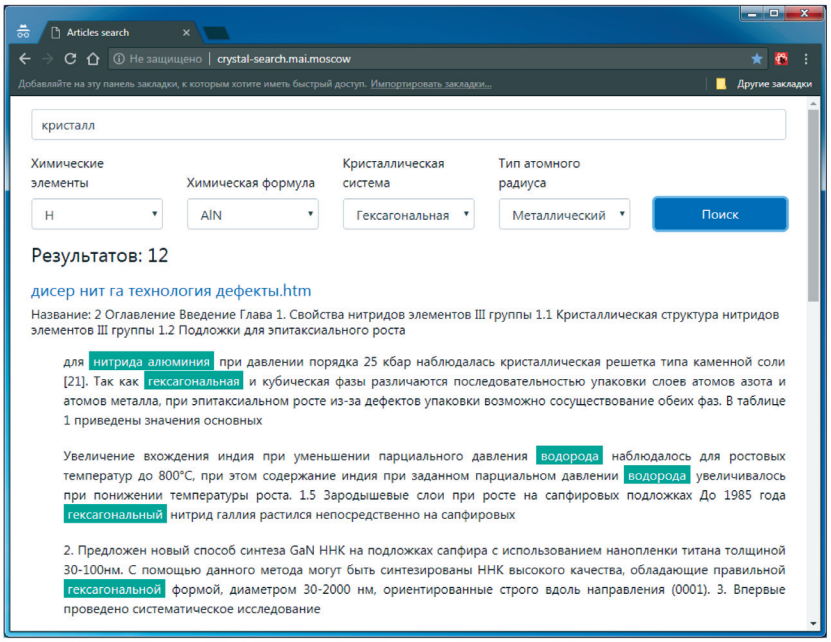

Рис. 3. Веб–интерфейс пользователя
Fig. 3. Web user interface

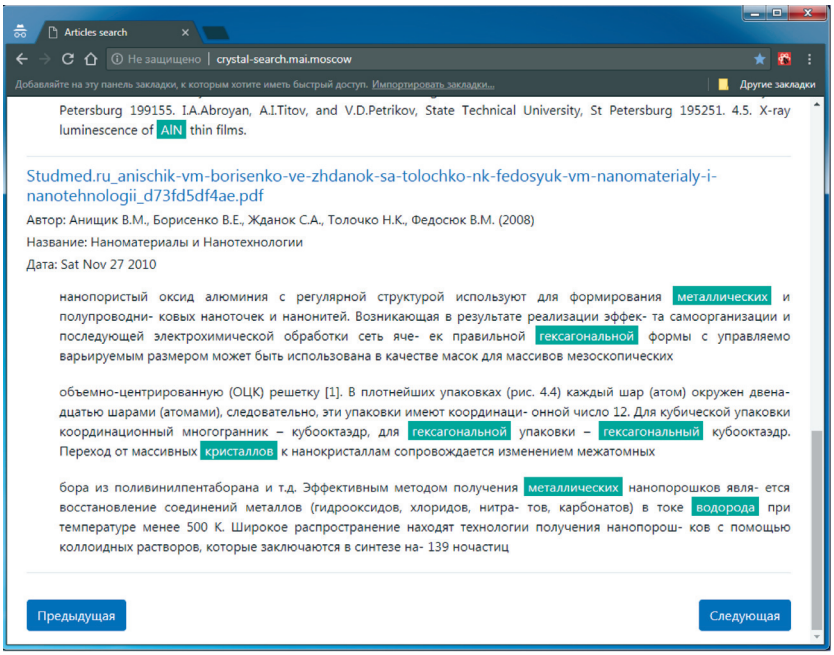

Рис. 4. Навигация веб–интерфейса
Fig. 4. Web interface navigation

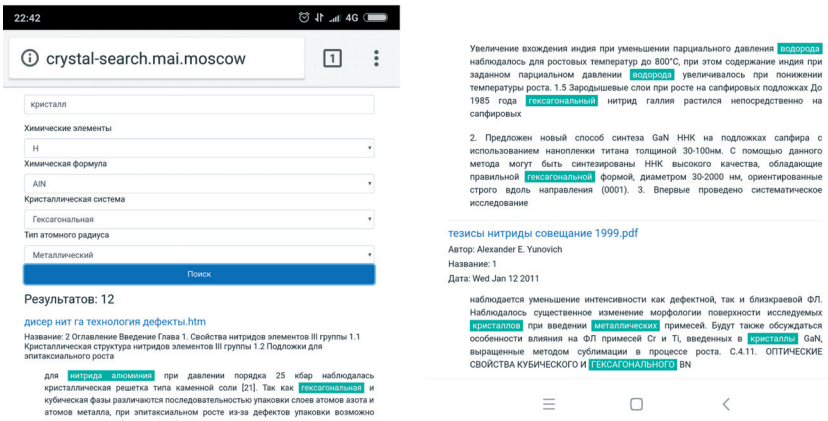

Рис. 5. Адаптивный дизайн интерфейса
Fig. 5. Responsive interface design

ки CrystalSystem, типом атомного радиуса RadiusType, пространственной группой SpaceGroup, номером документа с которого начинается выдача from (по умолчанию 0), размером выдачи size (по умолчанию 10), переданными в атрибутах запроса (ни один из параметров не является обязательным), вернет JSON с соответствующей поисковой выдачей.

- REST GET запрос по адресу / api/getChemicalElements возвращает JSON со списком химических элементов, которые фронтенд отобразит в соответствующем селекторе. Задается в файле на сервере.
- REST GET запрос по адресу / api/getChemicalFormulas возвращает JSON со списком химических формул, которые фронтенд отобразит в соответствующем селекторе. Задается в файле на сервере.
- REST GET запрос по адресу / api/getCrystalSystems возвращает JSON со списком кристаллических систем, которые фронтенд отобразит в соответствующем селекторе. Задается в файле на сервере.
- REST GET запрос по адресу /api/getRadiusTypes возвращает JSON со списком атомных радиусов, которые фронтенд отобразит в соответствующем селекторе. Задается в файле на сервере.

Есть возможность настройки адреса и порта Elasticsearch, имени индекса, количества фрагментов текста документа, отображаемого на фронтенде, с вхождениями поискового запроса, а также их длины. Можно настроить путь на сервере к файлам с данными для селекторов на фронтенде.

Поисковый запрос строится по двум полям документа: по тексту — «attachment.content» и имени — «filename». Таким образом повышается релевантность поиска, т.е. в поисковой выдаче первыми идут документы, у которых поисковый запрос имеет вхождение как в текст статьи, так и в ее название. Поисковый запрос строится таким образом, чтобы наибольшей релевантностью обладали

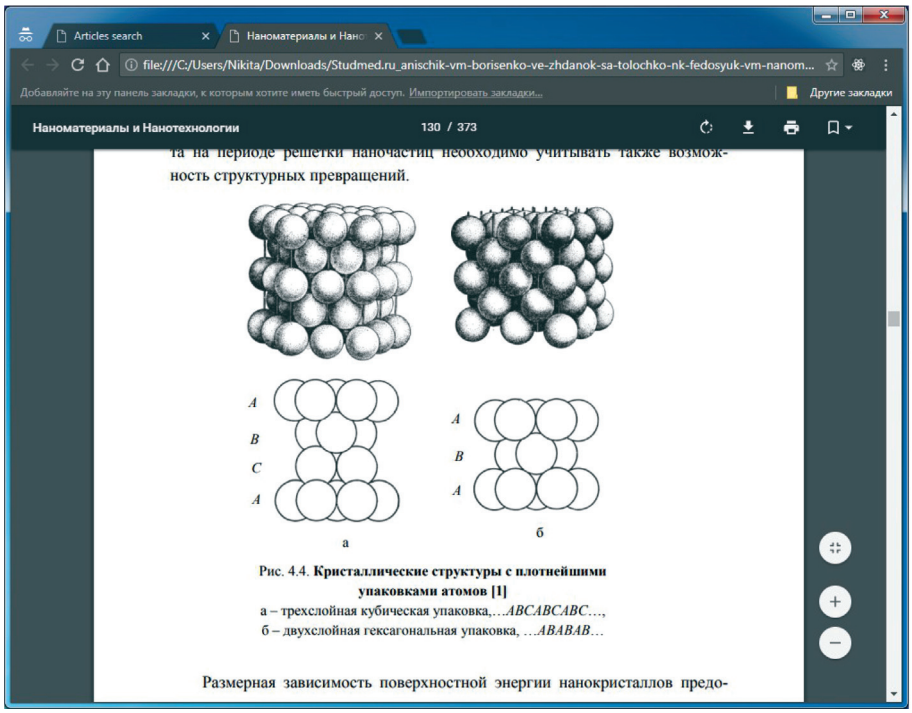

Рис. 6. Загрузка исходного документа
Fig. 6. Loading the original document

те документы, в которых все термы из поискового запроса входят в текст одновременно. В названии документа и его содержимом будут отмечаться вхождения термов поискового запроса.

Модуль пользовательского интерфейса. На рис. 3 изображено реализованное одностраничное React [4] приложение с использованием библиотеки компонентов Bootstrap 4.1.0 [5]. Приложение предоставляет строку поиска, а также четыре селектора с параметрами поиска: символ химического элемента, химическая формула, тип кристаллической решетки, тип атомного радиуса. На странице отображается количество найденных документов, затем следует не более трех документов, с подсветкой вхождения в текст поискового запроса, а также опциональной дополнительной информацией: имя автора документа, название документа, дата создания документа. Количество и длина этих фрагментов задаются на бекенде. Каждый документ предваряется своим именем, которое является ссылкой для загрузки документа. Это имя является именем файла на диске сервера, а название документа берется из служебного поля в документе. После списка документов идет поле навигации, отвечающее за переключение между страницами поисковой выдачи (рис. 4). Приложение осуществляет запросы к расположенному на бекенде приложению crystal_proxy.

Из рис. 3 можно видеть, как работает поиск, обратите внимание на стемминг — учитываются разные окончания слова. Выбраны данные во всех четырех селекторах. Списки селекторов заполняется по запросу на бекенд, то есть на бекенде хранятся все символы химических элементов, формул, кристаллических систем и атомных радиусов. Выводится количество найденных документов — 13.

Одно из преимуществ использования Bootstrap — это сетка (*grid system*) [5] которая позволяет задать ширину и положение каждого компонента. В основном это необходимо для корректного поведения веб–страницы при изменении ее размера, а также при отображении на разных устройствах, что можно проследить на рис. 5. Из рис. 5 видно, что компоненты не уходят за границы экрана, а перегруппируются, чтобы быть всегда на виду, что очень удобно на мобильных устройствах. Также используется класс–компонента *flex* [6], который позволяет задать положение компонента внутри ячейки, его работу можно проследить на рис. 3, где видно, что селекторы находятся на одной линии, иначе селектор с химическими формулами находился выше остальных.

Таким образом, у представителя предметной области есть понятный интерфейс для осуществления полнотекстового поиска по коллекции документов организации [7].

Заключение

В статье рассмотрена задача поиска научных публикаций, разработаны программные модули для индексации научных статей и полнотекстового поиска по ним, настроен поисковый индекс для предметно–ориентированного поиска и реализован пользовательский интерфейс. Реализован программный интерфейс для полученного поискового сервиса.

Библиографический список

1. Elasticsearch. URL: https://www.elastic.co/elasticsearch/ (дата обращения: 30.06.2020).
2. Ingest Attachment Processor Plugin. URL: https://www.elastic.co/guide/en/elasticsearch/plugins/7.17/ingest–attachment.html / (дата обращения: 30.06.2020).
3. Spring Boot. URL: https://spring.io/projects/spring–boot/ (дата обращения: 30.06.2020).
4. React. URL: https://legacy.reactjs.org/ (дата обращения: 30.06.2020).
5. Bootstrap 4.1.0. URL: https://bootstrap–4.ru/docs/4.1/getting–started/introduction/ (дата обращения: 30.06.2020).
6. Flexbox. URL: https://support.google.com/webdesigner/answer/10995586?hl=ru (дата обращения: 30.06.2020).
7. Зайцев Н. А. Инструментарий для сбора, индексирования и классификации статей по кристаллографии // Материалы XX Юбилейной Международной конференции по вычислительной механике и современным прикладным системам (ВМСППС'2017), Алушта, 24–31 мая 2017 г. Алушта: Издательство МАИ–Принт, 2017. С. 144—146.

Дополнительная информация

Для написания всех трех приложений использовалась система контроля версий Git с репозиторием на GitHub в среде разработки IntelliJ IDEA. Исходные коды доступны по следующим адресам:

https://github.com/JacksonGibsonESP/pdf_index
https://github.com/JacksonGibsonESP/crystal_proxy
https://github.com/JacksonGibsonESP/Elastic_Front

Все клиент–серверное приложение было целиком развернуто на удаленном сервере и доступно по адресу:

http://crystal-search.mai.moscow

Статья поступила в редакцию 14 января 2021 г.

Izvestiya vuzov. Materialy elektronnoi tekhniki = Materials of Electronics Engineering. 2020, vol. 23, no. 4, pp. 297—303. DOI: 10.17073/1609-3577-2020-4-297-303

Software tools for indexing and subject–oriented search of scientific articles on crystallography

E. S. Gavrilov[1,2,§], N. A. Zaitsev[3]

[1] ***Moscow Aviation Institute (National Research University),*** *4 Volokolamskoe Shosse, Moscow 125993, Russia*

[2] ***Federal Research Centre "Information and Control" of the Russian Academy of Sciences,*** *44–2 Vavilov Str., Moscow 119333, Russia*

[3] ***Sberbank,*** *19 Vavilov Str., Moscow 117312, Russia*

Abstract. Software modules are presented that implement the indexing of scientific articles, subject–oriented search on them, a user interface, as well as a software interface for third–party consumers. A detailed description of the process of setting up full–text search to the level of subject–oriented is given.

Keywords: full–text search, domain-specific search, inverted index, Elasticsearch search engine, Java language, Spring framework, JavaScript language, React framework

References

1. Elasticsearch. URL: https://www.elastic.co/elasticsearch/ (accessed: 30.06.2020).
2. Ingest Attachment Processor Plugin. URL: https://www.elastic.co/guide/en/elasticsearch/plugins/7.17/ingest–attachment.html / (accessed: 30.06.2020).
3. Spring Boot. URL: https://spring.io/projects/spring–boot/ (accessed: 30.06.2020).
4. React. URL: https://legacy.reactjs.org/ (accessed: 30.06.2020).
5. Bootstrap 4.1.0. (In Russ.). URL: https://bootstrap–4.ru/docs/4.1/getting–started/introduction/ (accessed: 30.06.2020).
6. Flexbox. (In Russ.). URL: https://support.google.com/webdesigner/answer/10995586?hl=ru (accessed: 30.06.2020).
7. Zaitsev N. A. Tools for collecting, indexing and classifying articles on crystallography. *Proc. XX Anniversary International Conference on Computational Mechanics and Modern Applied Systems, Alushta, May 24–31, 2017.* Alushta: Izdatel'stvo MAI–Print, 2017. Pp. 144—146. (In Russ.)

Information about authors:

Evgeny S. Gavrilov[1,2,§] — Senior Lecturer (1), Researcher (2), e–mail: eugavrilov@gmail.com; **N. A. Zaitsev[3]** — Developer, e–mail: JacksonGibsonESP@gmail.com

§ Corresponding author

Received January 14, 2021