

Введение

Вы почти закончили обучение и дошли до завершающей части курса, большая часть теории и практики осталась позади. Мы гордимся вашим трудолюбием и упорством! Теперь осталось совсем немного: выполнить и защитить итоговый дипломный проект.

Диплом — это тренировка знаний и дополнительный кейс в портфолио. Кейсы интересуют работодателей в первую очередь. Дипломный проект пригодится вам на собеседовании или станет дополнительным преимуществом при росте внутри компании.

Поэтому давайте соберёмся и сделаем это.



Будет интересно!



Тема вашего проекта

Реализация системы поиска по корпоративному порталу компании «АйТиБокс»



Перед вами подробное описание проекта, который станет отличным дополнением вашего портфолио. Здесь есть всё, что вам нужно, чтобы справиться с поставленной задачей.

- + [Описание задачи](#)
- + [ТЗ на дипломную работу](#)
 - + [Описание API](#)
 - [GET /api/search](#)
 - [POST /api/startIndexing](#)
 - + [Алгоритмы и принципы работы](#)
 - [Алгоритм индексации](#)
 - [Нормализация слов](#)
 - [Алгоритм выдачи результатов поиска](#)
 - + [Технические требования](#)
 - [Параметры конфигурации](#)
 - [Организация базы данных](#)
- + [Улучшения и дополнения к заданию](#)
 - + [Задача 1. Статус сервиса](#)
 - + [Задача 2. Стоп-слова](#)
 - + [Задача 3. Ранжирование слов 2.0](#)
 - + [Задание 4. Автотесты](#)
- + [Формат сдачи материалов и оценивание](#)
- + [Рекомендации по выполнению](#)
- + [Чек-лист](#)

👉 Последовательно изучите каждую часть.
Периодически обращайтесь к актуальным
для вас разделам в этом документе.



Описание задачи

Вы пришли в быстрорастущую технологический стартап «**АйТиБокс**». Когда-то один из основателей разработал специальный портал для сотрудников, который содержит базу знаний обо всех секретных разработках компании.

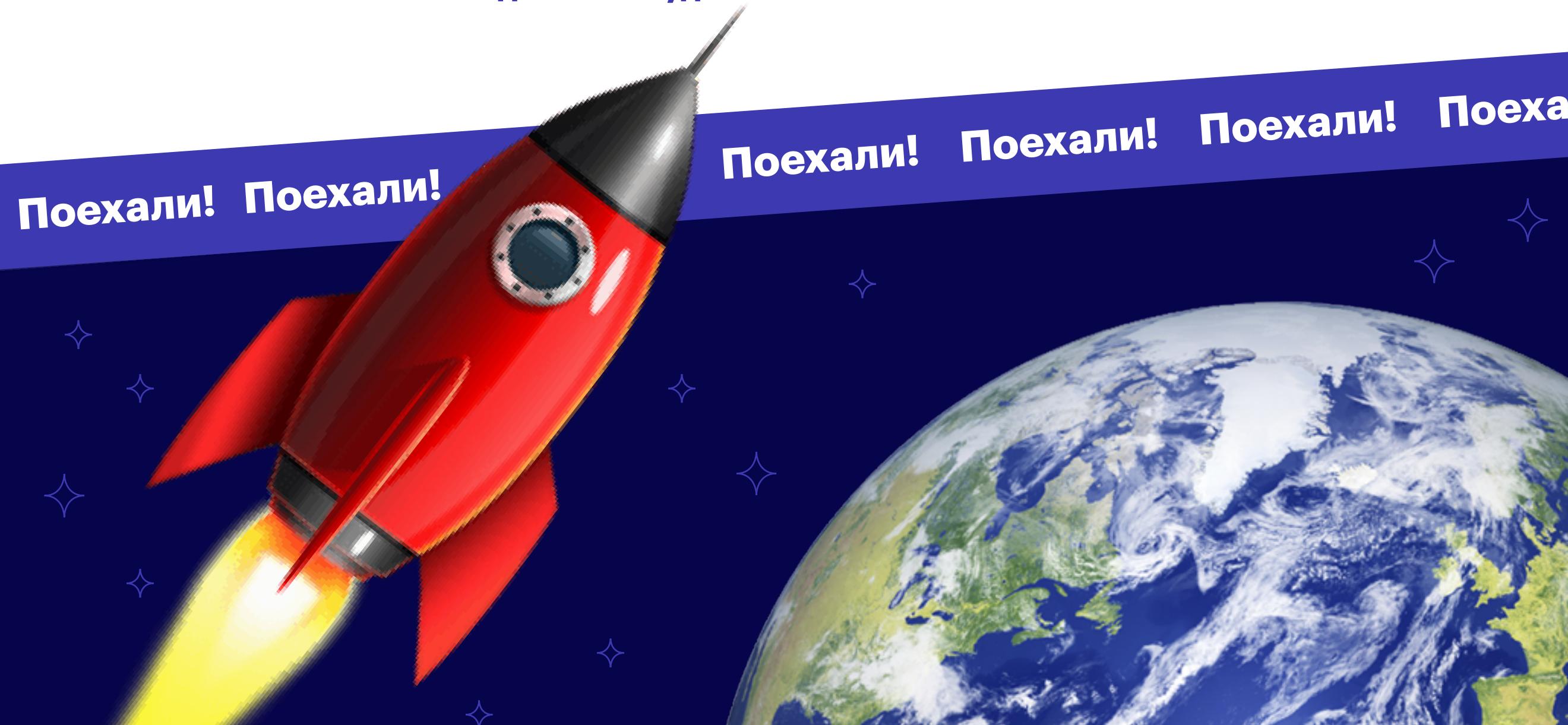
Портал отлично выполнял свою функцию, им с удовольствием пользовались все сотрудники, добавляли туда документы: от инструкции молодого бойца и инструкции на кофемашину до топологии корпоративной сети и описания секретных алгоритмов рекомендаций.

Но последнее время на портал начали очень много жаловаться. У него нашёлся один критический недостаток: **всё это время не было удобной системы поиска**. Когда документов было несколько десятков, искать было просто, но когда их стало несколько десятков тысяч, найти нужную информацию нелегко.

Технический директор решил поручить вам как подающему надежды новичку ответственную задачу — **разработать полнотекстовый поиск по базе знаний портала**. Тем более недавно инициативный верстальщик уже сверстал поисковую страницу, которую можно было бы прикрутить к порталу. Сделал бы кто бэкенд!

Так как в код портала уже очень давно никто не лез и в целом он оказался написан на очень экзотическом языке программирования, которым в то время увлекался основатель, то собирать информацию решили так же, как делает это Google или Яндекс — обходом и парсингом страничек, начиная с корневой.

Готовы спасти коллег и сделать им удобный поиск?



ТЗ на дипломную работу

Описание API

GET /api/search

Основным потребителем вашего API будет вот эта страница для поиска:

SEARCH

All sites ▾

слоны

SEARCH

Found 574 results

Первый сайт - Заголовок страницы, которую выводим
Фрагмент текста, в котором найдены совпадения, **выделенные жирным**, В формате HTML

Второй сайт - Заголовок страницы, которую выводим для примера
Фрагмент текста, в котором найдены еще совпадения, **выделенные жирным**, В формате HTML

SHOW MORE (572)

Скачать страницу поиска

Описание метода

Метод осуществляет поиск страниц по переданному поисковому запросу ([параметр query](#)).

Чтобы выводить результаты порционно, можно задать [параметры offset](#) (сдвиг от начала списка результатов) и [limit](#) (количество результатов, которое необходимо вывести).

В ответе выводится общее количество результатов ([count](#)), не зависящее от значений параметров [offset](#) и [limit](#), и массив [data](#) с результатами поиска. Каждый результат — это объект, содержащий свойства результата поиска (структура и описание каждого свойства даны ниже).

Если поисковый запрос не задан или ещё нет готового индекса (сайт, по которому ищем, или все сайты сразу не проиндексированы), метод должен вернуть соответствующую ошибку (как в примере ниже). Тексты ошибок должны быть понятными и отражать суть ошибок.

Параметры

- + **query** — поисковый запрос.
- + **limit** — сдвиг от 0 для постраничного вывода.
- + **offset** — количество результатов, которое необходимо вывести.

Формат ответа в случае успеха:



```
{  
    "result": true,  
    "count": 574,  
    "data": [  
        {  
            "uri": "/path/to/page/6784",  
            "title": "Заголовок страницы, которую выводим",  
            "snippet": "Фрагмент текста, в котором найдены  
совпадения, <b>выделенные  
жирным</b>, в формате HTML",  
            "relevance": 0.93362  
        },  
        ...  
    ]  
}
```

Формат ответа в случае ошибки:



```
{  
    "result": false,  
    "error": "Задан пустой поисковый запрос"  
}
```

POST /api/startIndexing

Но как новые страницы будут попадать в наш поиск?

Необходимо предусмотреть периодическое обновление индекса. Хорошо, что в компании есть свой планировщик задач и мы можем положиться на него. Он умеет в заданное время дёргать заданный метод API-сервиса. Вам остаётся только реализовать этот метод.

Описание метода

Метод запускает полную индексацию всех сайтов или полную переиндексацию, если они уже проиндексированы. Обратите внимание, что метод возвращает результат сразу, а индексация должна происходить в фоновом процессе. Если индексация или переиндексация уже запущена, метод возвращает соответствующее сообщение об ошибке.

Параметры

- + `queurls [опциональный]` — массив адресов страниц, которые нужно переиндексировать.

Формат ответа в случае успеха:

```
{  
  "result": true  
}
```



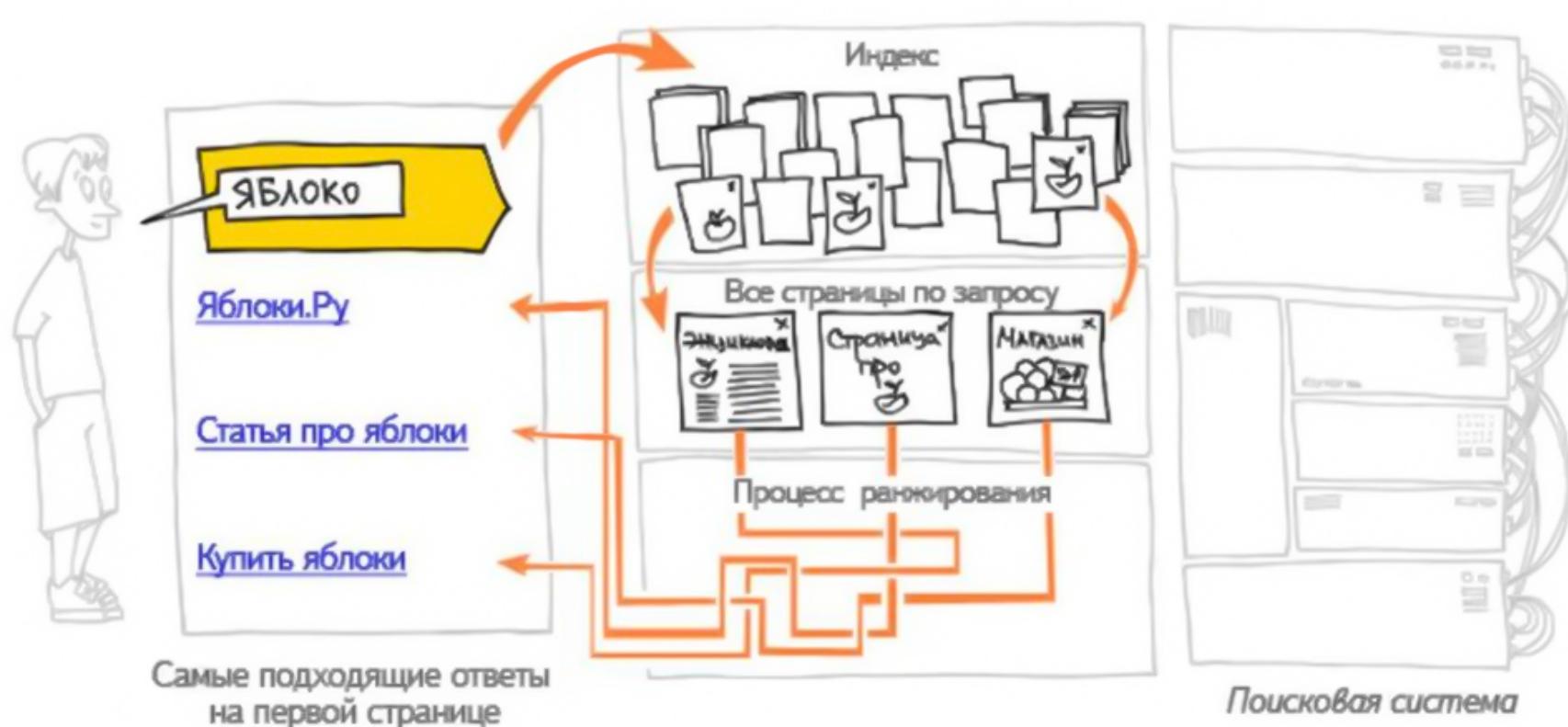
Формат ответа в случае успеха:

```
{  
  "result": false,  
  "error": "Индексация уже запущена"  
}
```



Алгоритмы и принципы работы

Как работает поиск



Алгоритм индексации

Индексация — это процесс формирования поискового индекса по некоторому объёму информации. Поисковый индекс — это специальным образом организованная база данных (в нашем случае — база данных PostgreSQL), позволяющая быстро и удобно осуществлять поиск по этой информации.

Поиск по поисковому индексу в любых поисковых системах, как правило, занимает очень короткое время (обычно доли секунды) по сравнению с обычным поиском перебором по всему массиву информации (вспомните разницу в скорости поиска перебором в простом массиве и бинарного поиска в отсортированном).

Удобство хранения информации в индексе достигается за счёт правильно организованной структуры хранения — базы данных. По обычному тексту или набору текстов вы не сможете искать информацию с учётом морфологии русского языка и одновременно оценивать релевантность результатов, если у вас не будет специально организованного поискового индекса.



**При запуске индексации поисковый движок должен выполнять
следующие операции:**

- 1.** Запускать в отдельных потоках индексацию каждого из разделов, перечисленных в конфигурационном файле.
- 2.** Получать HTML-код очередной страницы (сначала главной).
- 3.** Извлекать из него текстовые блоки информации. Это можно делать с помощью библиотеки [gumbo-parser](#).
- 4.** Извлекать из страницы все ссылки на страницы того же сайта и добавлять в очередь, проверив предварительно, не была ли какая-то из этих страниц уже проиндексирована через запрос к базе данных (таблица page).
- 5.** Полученные текстовые блоки разбивать на отдельные слова.
- 6.** Каждое слово преобразовывать в нормальную форму (см. ниже раздел «Нормализация слов»).
- 7.** Собирать все уникальные слова для данной страницы и считать их количества.
- 8.** Если слово отсутствует в базе, добавлять его в таблицу word со значением frequency, равным 1. Если присутствует, увеличивать число frequency на 1. Число frequency должно соответствовать количеству страниц, на которых слово встречается хотя бы один раз.
- 9.** Добавлять связку слова и страницы, на которой она встречается, в таблицу search_index со значением rank, равным количеству упоминаний на странице.
- 10.** Переходить снова к пункту 2, если на текущей странице остались непроиндексированные ссылки.
- 11.** Некоторые ссылки могут вести на несуществующие страницы (код ответа — 404) или страницы, содержащие ошибки (код ответа — 500). Такие страницы также необходимо добавлять в таблицу page, но не индексировать.

Нормализация слов

Нормализация слов — это процесс получения базовых форм слов.

Чаще всего в качестве нормализации используют либо лемматизацию, либо стемминг.

Лемматизация позволяет получать леммы слов — их исходные формы (для существительных, например, это слово в именительном падеже и единственном числе). То есть лемматизация позволяет учитывать морфологию слов.

Второй популярный способ — стемминг. Стемминг ничего не знает про морфологию, но с помощью эвристик позволяет получать общий «корень» слов.

Для нашей задачи подойдет стемминг как быстрый и достаточно надежный способ нормализации. Мы рекомендуем воспользоваться стеммером [snowball](#).

Алгоритм выдачи результатов поиска



Если индексирование портала завершено, по нему можно осуществлять поиск.

В этом случае пользователь вводит поисковый запрос, а поисковый движок выполняет следующие операции:

1. Разбивает поисковый запрос на отдельные нормализованные слова.
2. Формирует из этих слов список уникальных.
3. Сортирует слова в порядке увеличения частоты встречаемости (по возрастанию значения поля frequency) — от самых редких до самых частых.
4. По первому, самому редкому слову из списка находит все страницы, где слово встречается. Далее ищет соответствия следующего слова и этого списка страниц, и так по каждому следующему слову.
Список страниц на каждой итерации должен уменьшаться (или по крайней мере не увеличиваться).
5. Если в итоге не осталось ни одной страницы, выводит количество найденных страниц, равное 0.
6. Если страницы найдены, рассчитывает по каждой из них релевантность (и потом выводит её в поле relevance в ответе на запрос).

Для этого для каждой страницы рассчитывается абсолютная релевантность — сумма всех rank всех найденных на странице слов (из таблицы search_index), которая делится на максимальное значение этой абсолютной релевантности для всех найденных страниц.

Пример расчёта:

Страница	Rank слова «лошадь»	Rank слова «бегает»	Абсолютная релевантность	Относительная релевантность
1	4	3	7	0,7
2	1	2	3	0,3
3	5	5	10	1

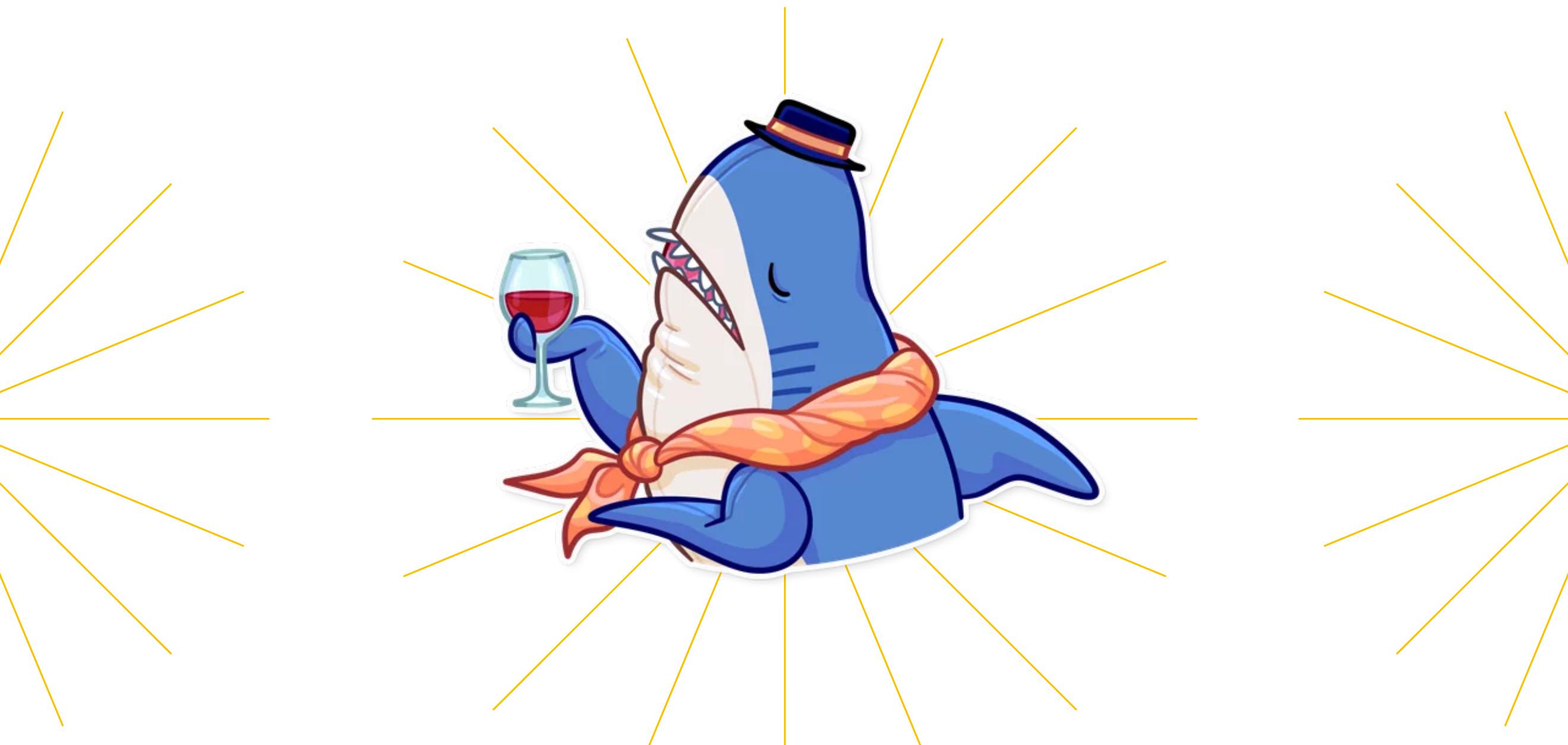
Пример расчёта абсолютной релевантности для первой страницы:

$$R_{abs} = 4 + 3 = 7$$

Относительную релевантность можно получить делением абсолютной для конкретной страницы на максимальную абсолютную релевантность среди всех страниц данной поисковой выдачи:

$$R_{rel} = 7 / 10 = 0,7$$

7. Сортирует страницы по убыванию релевантности (от большей к меньшей) и выдаёт пользователю в соответствии с форматом.



Технические требования

Параметры конфигурации

Параметры конфигурации приложения рекомендуется размещать в файле формата INI, а сам файл конфигурации должен размещаться за пределами приложения. Параметры конфигурации:

- + Данные доступа к базе данных: хост, логин, пароль, имя базы.
- + Стартовая страница портала, с которой начинается поиск.

Организация базы данных

page — проиндексированная страница

- + id INT NOT NULL.
- + path TEXT NOT NULL — адрес страницы от корня сайта (должен начинаться со слэша, например: /news/372189/).
- + code INT NOT NULL — код ответа, полученный при запросе страницы (например, 200, 404, 500 или другие).
- + content TEXT NOT NULL — контент страницы (HTML-код).

word — нормализованные слова, встречающиеся в текстах

- + id INT NOT NULL.
- + site_id INT NOT NULL — идентификатор сайта.
- + value TEXT NOT NULL — нормальная форма слова.
- + frequency INT NOT NULL — количество страниц, на которых слово встречается хотя бы один раз. Максимальное значение не может превышать общее количество слов на сайте.

search_index — поисковый индекс

- + id INT NOT NULL.
- + page_id INT NOT NULL — идентификатор страницы.
- + word_id INT NOT NULL — идентификатор слова.
- + rank FLOAT NOT NULL — ранг слова в данном поле этой страницы.

Улучшения и дополнения к заданию

Если вы уже выполнили задание по ТЗ, можете попробовать свои силы в дополнительных заданиях. Они расположены по сложности, от более простых к более сложным. Каждый пункт повышает качество системы и её привлекательность для пользователя. 💪😎



Сообщите преподавателю, за какие задания из этого раздела вы возьмёте дополнительно. Преподаватель уделит им особое внимание при проверке, а при возникновении трудностей будет готов поделиться советами и подсказками. За каждое выполненное задание вы получите дополнительные 0,5 балла, это позволит компенсировать недочёты в основной части проекта.

Задача 1. Статус сервиса

Ваш проект готов, все сотрудники с удовольствием начали им пользоваться и привыкли к нему. И вот в один прекрасный рабочий день всё сломалось! Админы получили по шапке и поняли, что у них нет никаких мониторингов вашего сервиса. К вам пришел главный админ Василий с идеей добавить ваш сервис на их главную панель управления.



Василий (главный админ)

Мы приглядим за твоим сервисом. Но нам нужен метод в API!

12:48

Вы договорились с ним, что для этого реализуете метод API такого вида:

GET /api/status

Метод должен возвращать статус и прочую служебную информацию о состоянии поисковых индексов и самого движка:

```
{  
  "status": "OK",  
  "pages_count": <количество страниц в индексе>,  
  "words_count": <количество слов в индексе>,  
  "index_size": <размер индекса, т.е. кол-во строк в search_index>  
}
```

Задача 2. Стоп-слова

Уже не первый раз вам жалуются, что поиск периодически подсовывает не те результаты, в основном на сложные запросы. Но в этот раз к вам пришла Наталья из отдела кадров и сказала, что на запрос «заявление на отпуск» показываются только какие-то спецификации на платы с заявлениями на патенты. И даже если запросить «на отпуск», то результат не лучше. А вот если оставить только «отпуск», то в десятке находится и заветное заявление.

Наталья (HR-менеджер)

Наши сотрудники не могут найти заявление на отпуск, не
отдыхают и выгорают прямо на рабочем месте! Помоги им найти
нужный документ.

12:48

И вот после продолжительной отладки вы находите причину: вся беда в предлоге «на»! 😱
Если этот предлог часто встречается на какой-то странице, то такая страница будет
очень релевантна запросу с этим предлогом.

Именно для этого были выделены стоп-слова. Это слова, которые не несут смысловой
нагрузки сами по себе, но из-за частого употребления могут сильно влиять на
ранжирование.

Список стоп-слов для русского языка

Доработайте алгоритм так, чтобы он перестал индексировать стоп-слова и учитывать
их при поиске.

Задача 3. Ранжирование слов 2.0

После того как поиск начал учитывать стоп-слова, результаты перестали быть случайными. И всё же периодически жалобы на качество поиска продолжались. С очень подробным описанием проблемы пришёл Владимир, один из маркетологов в компании.

Владимир (маркетолог)

На нашем портале мы храним описания наших рекламных компаний. Обычно мы их ищем по названию, которое включает категорию, период и регион. Например, «Имиджевая Новосибирск». Это название есть прямо в заголовках в описании. Проблема в том, что на портал также заливается куча табличных отчётов по рекламным кампаниям, которые не ссылаются на эту страничку. А в результатах поиска они показываются в первую очередь, потому что там название кампании встречается много раз. Можно ли что-то с этим сделать?



12:48

Конечно можно!

Надо всего лишь начать учитывать контекст тегов, где упоминается информация. Например, слова в названии страниц (тег title) или в заголовке (теги h1 – h6) слова значительно более значимы, чем в просто тексте (теги body, div, span, p) или в ячейке таблицы (тег td).

Давайте при ранжировании учитывать коэффициент тега.

Например, вот так:

Тег	Коэффициент
title	10
h1, h2, h3	5
h4, h5, h6	3
body, div, span, p и остальные	1
td	0,5

Тогда, если слово «Новосибирск» встречается один раз в title и два раза в заголовке h3, то его ранг будет считаться так:

$$R = 1 * 10 + 2 * 5 = 20$$

А если то же слово встречается 20 раз в ячейках таблицы td, то:

$$R = 20 * 0.5 = 10$$

Реализуйте этот алгоритм ранжирования! 😊

Задача 4. Автотесты

Поиск стал работать замечательно и каши не просил. И когда в команду вышел новый стажёр, вы решили передать ему проект на поддержку. Как раз прилетел небольшой запрос по улучшению поиска из бухгалтерии, и вы решили проверить стажёра на прочность. Он справился с задачей довольно быстро, но каково же было ваше разочарование, когда оказалось, что попутно он разломал половину проекта.



Иван (новый разработчик-стажёр)

Я всё сделал чётко по ТЗ, и моя фича работает. Не понимаю, что могло пойти не так...

12:48

После бессонной ночи починки новых багов вам пришло кристальное понимание, что спокойно передать проект вы сможете только после полного покрытия его тестами. Тогда вы будете уверены, что новая функциональность не сломает старую, если все тесты зелёные. ● ● ● ● ● ●

Напишите автоматические тесты, которые будут покрывать **не менее 80%** кода (хотя бы по количеству строк).

Формат сдачи материалов и оценивание

Результат проведённой работы опубликуйте на GitHub <https://github.com> и передайте проверяющим ссылку на проект.

Готовый проект мы будем проверять по следующим критериям:

- + В README описана команда, которой необходимо запускать проект после сборки.
- + Проект компилируется, собирается и запускается.
- + API возвращает ответ в том формате, который описан в задании.
- + В бизнес-логике нет ошибок.

Работа возвращается на доработку, если:

- + отсутствует или не работает базовая функциональность (индексация, поиск);
- + имеются грубые ошибки в организации или структуре проекта;
- + преподаватель не может запустить проект локально.

Оценка «5» выставляется, если проект реализует всю базовую функциональность, работает без ошибок. Если какая-то часть задания не реализована или реализована неправильно, **оценка снижается на 0,5** за каждый такой случай. При этом за каждое дополнительное задание оценка **повышается на 0,5**, но не может быть больше 5.

Работа возвращается на доработку, если итоговая оценка меньше 3, а также по желанию, если ваша оценка меньше 5 и вы хотите доработать..

В результате вас допускают к защите проекта. На ней нужно продемонстрировать результаты проделанной работы и ответить на вопросы комиссии.

Если итоговая оценка по результатам защиты меньше 3 баллов, работа считается невыполненной.

В этом случае есть ещё две попытки доработать проект по обратной связи и рекомендациям и вновь представить его комиссии.

Если кто-то не защитится с третьей попытки, вместо диплома он получит сертификат о том, что успешно прослушал курс. (Такого ещё ни разу не было 😊).

Рекомендации по выполнению

План работы над курсовой:

- 📝 Внимательно изучите задание. Убедитесь, что вам всё понятно. Это первый и очень важный шаг в начале работы над любым проектом. Если у вас возникают вопросы, фиксируйте их для себя и обязательно задавайте своему куратору дипломного проекта.
- ▣ Чтобы прийти к результату и выполнить проект целиком, рекомендуем разбить работу над ним на части.
- 📅 Составьте план работы по датам. Исходя из нашего опыта, продуктивнее выделять на дипломный проект по 2–3 часа несколько дней в неделю, чем делать этот же объём за один подход. Рекомендуем придерживаться этого графика и обязательно выделять время для отдыха.
- ☑ Отмечайте свой прогресс по мере выполнения плана. Это полезно по нескольким причинам: во-первых, вы будете держать ритм, во-вторых — контролировать ситуацию. И самое главное: каждый выполненный этап — это ваша маленькая победа: чем больше таких побед вы будете замечать, тем большее удовольствие от выполненного проекта получите в итоге.
- 👁️ Прежде чем отправлять каждую часть проекта на проверку преподавателю (а в реальной работе — тестировщику), всегда уделяйте время проверке за собой. Мы рекомендуем вам протестировать каждый метод отдельно, а также проверить весь проект целиком, когда вы его закончите. Идеально, если между завершением работы и проверкой за собой пройдёт некоторое время — это позволит вам отключиться и в результате посмотреть на работу свежим незамыленным взглядом.
- ☰ Используйте чек-лист, чтобы проверить, что вы реализовали все возможности.



Чек-лист

- Реализован метод /api/startIndexing.
- Индексация страниц происходит в фоновом режиме.
- Страницы при индексации обходятся рекурсивно.
- Одна и та же страница не индексируется два раза.
- HTML-теги не индексируются.
- Слова при индексации приводятся к нормальной форме при помощи стемминга.
- Для слов на странице рассчитывается их ранг по формуле.
- Реализован метод /api/search.
- Для каждой страницы в результате рассчитана её релевантность по формуле.
- Поиск возвращает результаты в порядке их релевантности.
- Метод поиска учитывает параметры limit и offset.
- Метод поиска возвращает общее количество результатов
- Найденные слова выделяются в сниппетах тегами .
- Параметры для подключения к базе данных хранятся в конфигурационном файле
- Если в базе данных нет таблиц, они создаются автоматически при старте приложения.
- Проект выложен на GitHub со всеми необходимыми файлами для сборки.
- В README описана команда, которой необходимо запускать приложение после сборки.

