ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ АВТОНОМНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
НАЦИОНАЛЬНЫЙ ИССЛЕДОВАТЕЛЬСКИЙ УНИВЕРСИТЕТ
«ВЫСШАЯ ШКОЛА ЭКОНОМИКИ»

Факультет бизнеса и менеджмента

# РАЗРАБОТКА И ИССЛЕДОВАНИЕ ПОИСКОВОЙ СИСТЕМЫ, ПОДДЕРЖИВАЮЩЕЙ ОСНОВНЫЕ ПРИНЦИПЫ ФИЛЬТРАЦИИ И РАНЖИРОВАНИЯ ДОКУМЕНТОВ

Курсовая работа студента

Холодилина Максима Дмитриевича

2 курс, направление подготовки: 38.03.05 «Бизнес-информатика»

образовательная программа «Бизнес-информатика»

Научный руководитель

д-р наук, проф.

Ефремов Сергей Геннадьевич

Москва 2021

# СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Во введении к данной курсовой работе очень важно изначально обозначить смысл и первоначальную цель создания данного проекта. Формируя свои знания в области математики, программирования и баз данных, учась на направлении “бизнес-информатика”, мною была выдвинута идея создания непосредственно фундаментальной части программно-аппаратного обеспечения поисковой системы, с использованием ряда релевантных и одновременно актуальных решений, как в области объектно-ориентированного программирования (*далее ООП*), так информатики и математики. Данный много-интегрируемый проект может служить основой для разработки последующих IT-решений или продуктов, связанных с работой над поисковыми запросами как явления.

В своей курсовой работе я ставлю ряд последовательных задач, напрямую связанных с реализацией проекта по вышеуказанной теме, каждая из которых имеет свой конечный результат, а также процесс отслеживания задачи с помощью отдельного введения списка задач и их актуальности. В моем первоначальном исследовании, были сформулированы главные тезисы будущей работы, основанные на изученных мною материалах. В последующем проектировании создано абстрактное, а также системное понимание будущего решения. В разносторонней и финальной разработке создано уникальное решение с помощью сформированных знаний в различных научных сферах.

Структуру данной курсовой работы можно и стоит разделить на три основные части: исследование, в котором, подробно рассмотрена общая концепция и вариация понятия поисковой системы, и выделены главные свойства, теоретической части, в которой приведены математические, а также компьютерно-системные сведения и результаты исследований, далее используемых в работе. И практической части, в которой, реализовано проектирование, программирование и тестирование непосредственно проектного решения.

Ряд, изученных мною материалов и источников очень широк. В него входят научные статьи, описывающие системно-абстрактные понятия, материалы по математике, документация, по выбранному мною языку программирования “c++”, общие материалы по программированию, в частности *ООП*.

# 1. ИССЛЕДОВАНИЕ И АНАЛИЗ ТЕМАТИЧЕСКИХ МАТЕРИАЛОВ

## 1.1. Введение в исследование

В данном, довольно сжатом пункте исследовательской части своей курсовой работы, я бы хотел указать на первую, может не столь практическую, но очень важную цель – формирование знаний и глубокого понимания всех далее используемых: терминов, алгоритмов, методов решения различных задач и всей, на начальный момент работы, абстрактной модели поисковой системы, в ее различных проявлениях. Исследование и последующий анализ, с выявлением основных и более дифференцированных сущностей, механизмов и других особенностей такого явления, как – поисковая система, сформирует более точный и многогранный взгляд на реализацию моего будущего решения, а также даст более широкое понимания всего происходящего для читателя данной работы.

## 1.2. Статья “History of Search Engines: From 1945 to Google Today”

Представленная статья является отличным, как вводным, так и основательным материалом для последующих суждений и тем более анализа, название статьи уже говорит само за себя. Начиная с истории формирования поисковых систем, а тем более причин и предпосылок еще в послевоенный период (*1945–1960*), Аарон Уолл (*Aaron Wall*) дает ясное понимание того, что же такое поисковый движок, его основных принципов и механизмов работы. Автор приводит массу примеров поисковых систем, достаточно, отличающихся по своей природе и работе, но одновременно схожих по их основе.

В самом начале статьи автор упоминает Джерарда Солтона (*Gerard Salton*), а точнее его главную работу жизни “*SMART informational retrieval system*” – систему механического поиска и анализа текста, по сути своей, являющейся корневой основой большинства поисковых систем. Данная система основана на трех главных механизмах, которые в дальнейшем будут мною разобраны и применены в работе. Первый, и, пожалуй, самый основной механизм реализации релевантного поиска – “*TF-IDF*”, основанный на статистических измерениях внутри поисковых сущностей. Второй – дискриминация или исключение нерелевантных запросов или результатов

поиска, имеющий ряд механизмов, таких как дедупликация или валидация. И третий – механизмы обратной связи по релевантности, основанных скорее на человеческой осведомленности о ресурсе поиска и запросе, чем на математической алгоритмизации. В статье также был упомянута небезызвестная, первая полноценная система поиска “*Archie*”, созданная еще в начале девяностых годов для помощи студентам в поиске информации по имеющимся документам по их поисковому запросу. Благодаря именно этому поисковику, поддерживающим использование “*TCP*” протокола, были созданы все последующие веб-поисковики.

В статье рассматривается 3 главных компонента полноценной веб-поисковой системы: поискового робота, он же веб-паук, предназначенного для перебора страниц Интернета и добавления этих страниц в базу данных поискового сервера, индекса – каталога этих страниц, которые хранят информацию о странице, и программного обеспечения релевантного поиска, по-простому – алгоритмов поиска. Автор также говорит, что в большинстве поисковый систем выполняется следующие принципы работы: обработка запроса пользователя, дальнейшая работа по поиску релевантных документов, основанный на запросе, сбор всех релевантных документов и их ранжирование. Сразу стоит сказать, так как, моя будущая поисковая система, подразумевает работу с абстрактными данными, в ней не будет функции поиска по сети Интернет, однако, я сразу ставлю себе задачу имплементировать как можно больше основных компонентов более серьезных поисковых систем.

Ближе к концу статьи автор пишет о глобальной и более узкой оптимизации поискового движка (“*SEO – Search Engine Optimization”*), как о некой концепции улучшения поисковых возможностей подобных систем, благодаря определению контента страниц, как более нужного и понятного для человека, одновременно внедряя все свойства в поисковые возможности системы. Под этим подразумевается всевозможные рейтинги документа и другие более сложные особенности страниц, также основанных на абстрактных рейтингах. Я также учту эту идею, при создании своей системы.

(1)

## 1.3. Статья “The Anatomy of a Large-Scale Hypertextual Web Search Engine”

*“Когда мы заглядывали в интернет, мы не читали там гороскопы и не заходили на сайты знакомств. Нас интересовал поиск — та информация, которая по-настоящему влияет на жизнь людей.”*

*Сергей Михайлович Брин.*

Основываясь на теме курсовой работы, я не мог не воспользоваться какой-либо информацией от, непосредственно, создателей поисковых систем, и наткнулся на, не побоюсь этого слова, грааль из мира похожих статьей – публикацию Сергея Михайловича Брина и Лоуренса Эдварда Пейджа – создателей самого популярного и успешного поискового сервиса “Google”. Авторы статьи, пишут о работе и особенностях крупномасштабных гипертекстовых веб-поисковых систем, основываясь, на своем, тогда еще не очень популярном поисковом сервисе. Несмотря на то, что в статье, по большей части, рассматривается связь поискового движка и интернета, в работе также фигурирует ряд терминов и функциональных составляющих поисковой системы, упомянутых в анализе предыдущей статьи, что, в свою очередь, свидетельствует о важности их использования в дальнейшей работе.

Одна из главных особенностей данной поисковой системы, по словам авторов это диверсифицированное хранение данных о документах, в разных частях, названных бочками (“*barrels*”), позволяющих ускорить и оптимизировать поиск. Авторы пишут об индексации документов, конкретно, по переменной “*docID”,* являющейся главным индикатором документа в системе хранения, и о необходимости связи между такими сущностями как документ и страница (“*WebPage*”). Что касается фильтрации (“*parsing*”) поисковых запросов, авторы, как и автор прошлой статьи, говорят о большом количестве всевозможных ошибок, связанных с вводом текста пользователем, которые стоят отлавливать валидирующими методами на самых ранних этапах. Также важно заметить и использование, так называемой, частоты обратных документов (“*inverse document frequency / IDF*”), части, вышеупомянутой статистической методики ранжирования документов.

В статье было указано, что языком программирования, на котором написана большая функциональная часть поискового движка является “c++”. Имея опыт работы с данным языком, а главное – условный совет авторов о преимуществах данного языка, я решил остановиться именно на нем.

(2)

## 1.4. Выводы из исследования

Большинство поисковых систем используют модель булевого поиска (“*Boolean search*”), то есть, поддерживают обработку запросов, основанных на реляционной алгебре, связанных с такими логическими операторами как: “AND”, “OR”, “NOT”. Такие поисковые системы, в большинстве своем, используются при работе с базами данных, с помощью языка реляционных запросов “*SQL*”. Система, которая будет создана, как итоговый продукт данной курсовой работы, является альтернативой модели булевого поиска – модели поиска с ранжированием, позволяющей задавать запрос в произвольной форме. (3)

Изучив вышеуказанные материалы, я сформировал общее понимания того, как будет выглядеть моя модель поисковой системы, какие сущности, особенности и механики поиска она будет содержать. В последующих частях курсовой работы мною параллельно будет моделироваться и создаваться поисковый движок (*3. Практическая часть*), рассматриваться применяемые глобальные и, а также многопоточность поисковой системы (*2. Теоретическая часть*).

# 2. ТЕОРЕТИЧЕСКАЯ ЧАСТЬ

## 2.1. C++

Как уже было упомянуто, данный проект курсовой работы разработан на языке программирования “*C++*”, и было бы некорректно, не ввести и указать основные используемые концепции языка, которые были применены. Данная часть является вводной ко всей теоретической части.

“*C++*” является многофункциональным, высокоуровневым языком программирования, разработанным на основе “C”. Данный язык позволяет создавать различного рода высоконагружаемое программное обеспечение. Несмотря на то, что данный язык является высокоуровневым, в нем существует ряд механизмов, дающих возможность работы с памятью или многопоточностью (*2.4. Концепция многопоточности*).

В своей программной реализации поисковой системы мною были использованы исключительно модули стандартной библиотеки (*std*) языка, с целью создания более уникального и обособленного проекта. Используемые мною модули стандартной библиотеки, содержат объявление типа данных, таких как: вектор (*vector*), словарь (*map*), множество (*set*), строка (*string*), итераторы на строку (*string_view*), дек (*deque*), также модули для использования локальных алгоритмов (*algorithm, cmath*), выбрасывания исключений (*stdexcept*), потоков вывода (*iostream*), многопоточных вычислений (*execution, future, mutex*), а также модуля, содержащего работу с временными типами данных (*chrono*), для измерения работы программы.

(4)

## 2.2. Глобальные алгоритмы

### 2.2.1. TF-IDF

Используемый мною (*в таких функциях как FindTopDocuments*) релевантно сортирующий механизм TF-IDF основан на измерении двух статистических величин: частота термина (*term frequency*), и обратная частота документа (*inverse document frequency*).

Частота термина это – сколько раз одно слово встречается в документе. Обратная частота документа – это количество всех документов, поделённое на

количество тех документов, где встречается это слово (*повторы и пустые документы исключаются*). Соответственно релевантный вес ($w$) каждого $i$-того термина в каждом $j$-том документе вычисляется по следующей формуле, где $tf$ – частота появлений $i$ в $j$, $N$ – количество документов, $df$ – количество документов содержащих $i$:

$$w_{i,j} = tf_{i,j} * log\left(\frac{N}{df_i}\right) \quad (5)$$

### 2.2.2. Матчинг документов

Матчингом (*от англ. matching*) называется простое отношение всех совпавших слов запроса (*без повтора*) со словами из тех, что содержатся в документах. Отношение можно записать (*каждое i-тое слово пересекается с каждым i-словом j-того документа*) и изобразить (*рис.* 1) следующим образом:

$$w_i \cup dw_{i,j}$$

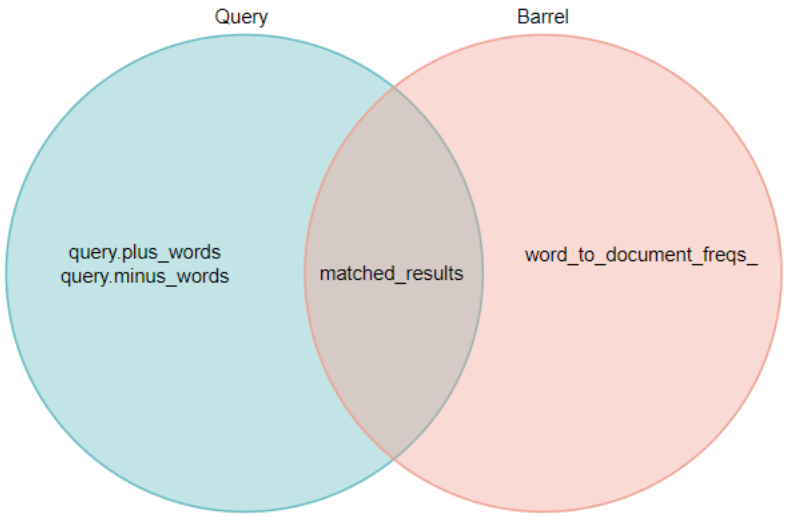


*рис. 1*

### 2.2.3 Фильтрация запроса

Учитывая тот факт, что большинство поисковых запросов вводится людьми, требуется применение не самого трудного алгоритма фильтрации

такого запроса с несколькими ступенями обработки. Для этого запрос разбивается двумя циклами на слова и на символы этого слова, сначала проверяется валидность слова, затем валидность на минус слово, далее валидированному слову дается один из трех типов: плюс слова (*plus_word*), минус слова (*plus_word*), стоп слова, исключаемого из запроса (*stop_word*). Алгоритм фильтрации запроса выглядит следующим образом (*рис. 2*).

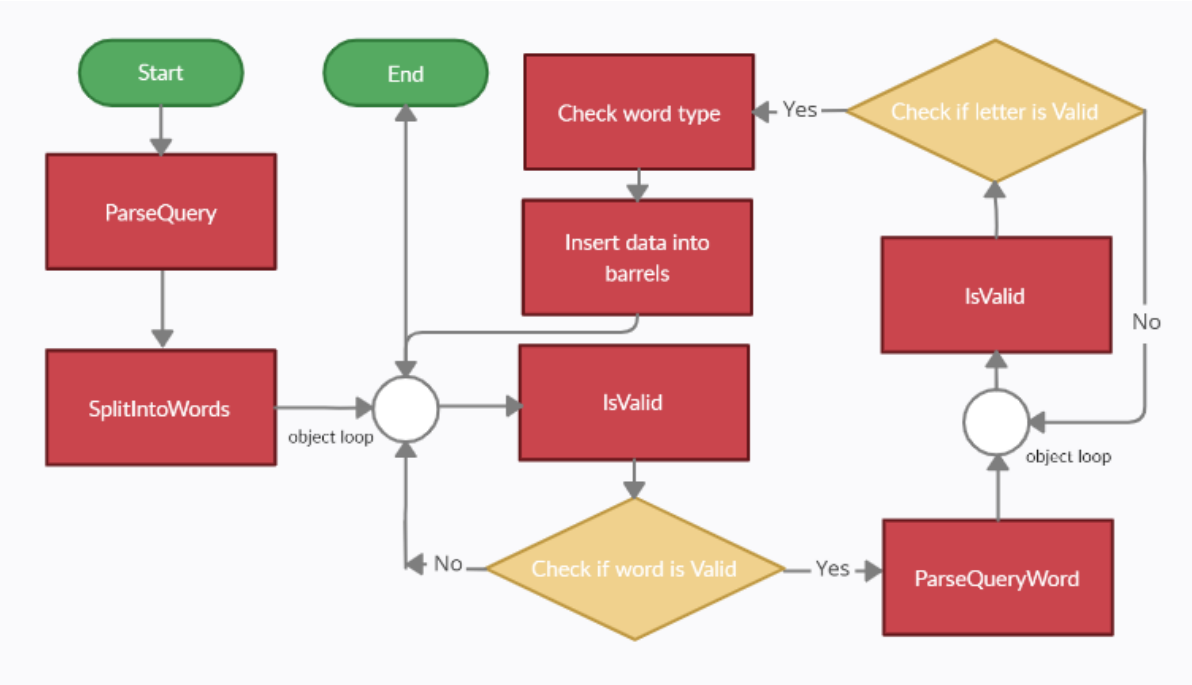


*рис. 2*

Красным отмечены функциональные блоки алгоритма, желтым подробности функционального блок, белым кругом обозначено циклическое взаимодействие с содержанием объекта, в нашем случае – запросом строкой. Зеленым – начало и конец алгоритмических действий.

## 2.3. Многопоточность поисковой системы

В программной реализации используется алгоритмы из модуля стандартной библиотеки (*algorithm*), такие как *sort*, сложность исполнения которого равна $O(N * log(N))$, где $N$ – расстояние от первого до последнего элемента. Также используемые алгоритмы поддерживающие многопоточность (*transform, for_each*), позволяют осуществить параллельные операции на уровне вычислений процессора, тем самым сократив сложность исполнения алгоритма до $N$.

(6, 7)

Помимо распараллеливания методов поисковой системы (*FindTopDocuments*, *MatchDocument*), также создан словарь (*concurrent_map*), поддерживающий параллельные операции в одноименных методах, и параллельная обработка запросов (*process_queries)*.

# 3. ПРАКТИЧЕСКАЯ ЧАСТЬ

## 3.1. Выявление всех сущностей и связей

### 3.1.1. Документ

Данная, упомянутая в изученных мною статьях, сущность, является основополагающим элементом поиска, которая будет подразумевать под собой, некий набор особенностей (*полей*), которые нужны, непосредственно, как – используемая информация. Во время поиска, соответственно, мы будем использовать большое количество таких документов (*массив*). Документ также должен быть в будущем связан с условной сущностью – страница (*об этом подробнее в разделе Paginator*). Сформировав видение того, что документ – это некий объект информации для будущего поиска, однако не содержащий самого текста (*он добавляется в систему и связывается с этой сущностью отдельным методом самой поисковой системы, все это делается как-раз для использования – бочек хранилища системы*), стоит сразу определить, что в него входит: индекс (*index*) – идентификатор документа в системе, релевантность (*relevance*) изначально равной нулю, и условного рейтинга (*взятый для использования концепции SEO*), подразумевающего под собой – оценки ресурса или важность.

Также стоит добавить где-то рядом статус этого будущего ресурса как статус документа (*Document Status*), для отметки этого ресурса в системе как актуального (*ACTUAL*), нерелевантного (*IRRELEVANT*), заблокированного (*BANNED*) или убранного (*REMOVED*). Делается это вне сущности документа, так как, статус документа определят система, а не он сам, соответственно, система должна иметь возможность добавлять в хранилища данных (*бочки*), один и тот же документ с разными статусами, в зависимости от того, зачем это требуется.

### 3.1.2. Поисковый запрос

Поисковый запрос, по сути своей, это произвольный набор символов, введенный пользователем. Изначально важно понимать, что данный запрос может содержать: не нужные символы, стоп-слова (*слова, которые не используется в механизмах построения релевантных запросов, например: предлоги*), а также минус-слова (*minus_words*), позволяющие исключить не

само слово, а целый документ из поиска. Данный запрос, будет приниматься методами поискового движка/сервера (*методом поиска всех документов, методом поиска самых релевантных документов, а также матчингом (matching) документов*), также фильтруя и, далее, распределяя запрос в объекты хранения.

### 3.1.3. Поисковый сервер

Данная часть всего проекта, является мозгом всей системы, формирующим и хранящим в дифференцированных формах данные различного рода о массиве из документов. Более того эта сущность позволяет выполнять ряд действий (*2.2. Глобальные алгоритмы*) с изначальной добавляемыми документами, а также со всей информацией, которая получена из них и распределена по бочкам хранилища. Помимо того, что данная система работает с документами и информацией для осуществления поиска по ним, она также должна уметь правильно работать с самими поисковыми запросами, а именно: применять способы фильтрации, парсинга (*parsing*), а также валидации (*validation*) (*2.2. Глобальные алгоритмы)*. Название данной сущности (*SearchServer*) ориентированно на будущее взаимодействие с сетью Интернет.

Одна из самых важны частей всего проекта – это создание оптимальных контейнеров хранения всей информации (*бочек*), которые будут, находится на самом поисковом сервере. С целью серьезной оптимизации поиска и ранжирования, мною были реализованы следующие контейнеры хранения информации о запросе и документах: множество стоп слов, которые задаются при создании объекта поискового сервера (*stop_words_*), словарь сопоставления конкретного слова со словарями из индекса документа и частоты (*word_to_document_freqs_*), словарь сопоставления индекса документа со словарями слов и частот (*document_to_word_freqs_*), словарь хранения всей ифнормации документа, кроме самих слов (*documents_*) и множества индексов всех документов (*document_ids_)*. Данная модель хранения, основанная, на материале одной из прочитанных мною статей, сильно поможет ускорить работу всей системы.

### 3.1.4. Условные страницы

Так как, создаваемая мною система, подразумевает возможную будущую интеграцию с сетью, в ней должны присутствовать базовые концепции, используемые, конкретно при поиске с помощью глобальной сети Интернет. Один, из основных, используемых интернетом, механизмов работы с информацией – это условные страницы (*pages*) и деление информации на такие страницы (*paging*). Говоря более простым языком, это механизм, создания более удобного формата хранения информации для человека.

В моей программной реализации это выполняют два отдельных класса. Первый - условная страница (*IteratorRange*), использующая механику итерирования (*перемещения*) между содержимым страницы, благодаря итераторам (*объектам доступа*) к содержимому контейнера хранения, в данном случае – условной страницы (*2.1. C++*). Второй – класс деления на страницы (*Paginator*), хранящий в себе набор таких условных страниц.

### 3.1.5. Дедупликатор

Дедупликатор – это простая оболочная над поисковым сервером функция, позволяющая найти и удалить документы, находящиеся на сервере в дифференцированных хранилищах, содержащие один и тот же набор слов/символов, с помощью внутренней функции самого поискового движка (*RemoveDocument*). Данный механизм, также направлен на улучшение качества поиска.

### 3.1.6. Очередь запросов

Так как поисковых запросов на сервер приходит много, а иногда – очень много, их физическая обработка занимает время, соответственно, для оптимизации поисковой системы, стоит имплементировать некий функциональный объект, осуществляющий формирование очереди из запросов. Делается это для того, чтобы разгрузить нагрузку на систему со стороны поступающих запросов. Запросы, ожидающие обработки будут находится в очереди, до того момента, как сервис-обработчик дойдет до них. Также данная функциональность, может быть применена для хранения только нужных запросов, например: системе нужно знать какие актуальные запросы были только за последние сутки, значит нужно хранить и время отправки запроса. Если обозначить, что каждый запрос приходит один раз в течение

минуты, то таких запросов в сутки будет 1440, соответственно, если придет 1441 запрос, то можно обнулить все прошлые, так как они уже не будут актуальны для следующего дня. Данную функцию будет выполнять класс очереди запросов (*RequestQueue*).

### 3.1.7. Верхнеуровневая модель системы

Данный пункт практической части работы, является суммированием нескольких предыдущих, и направлен на создание абстрактной верхнеуровневой модели всей поисковой системы, которая используется как шаблон для описания главных механизмов работы системы, а также как шаблон написания программной реализации проекта. Представленная модель (*рис. 3*), пока-что не подразумевает деления проекта на требуемые, для более организованной работы, файлы, однако в следующем пункте работы, это обязательно будет осуществлено.

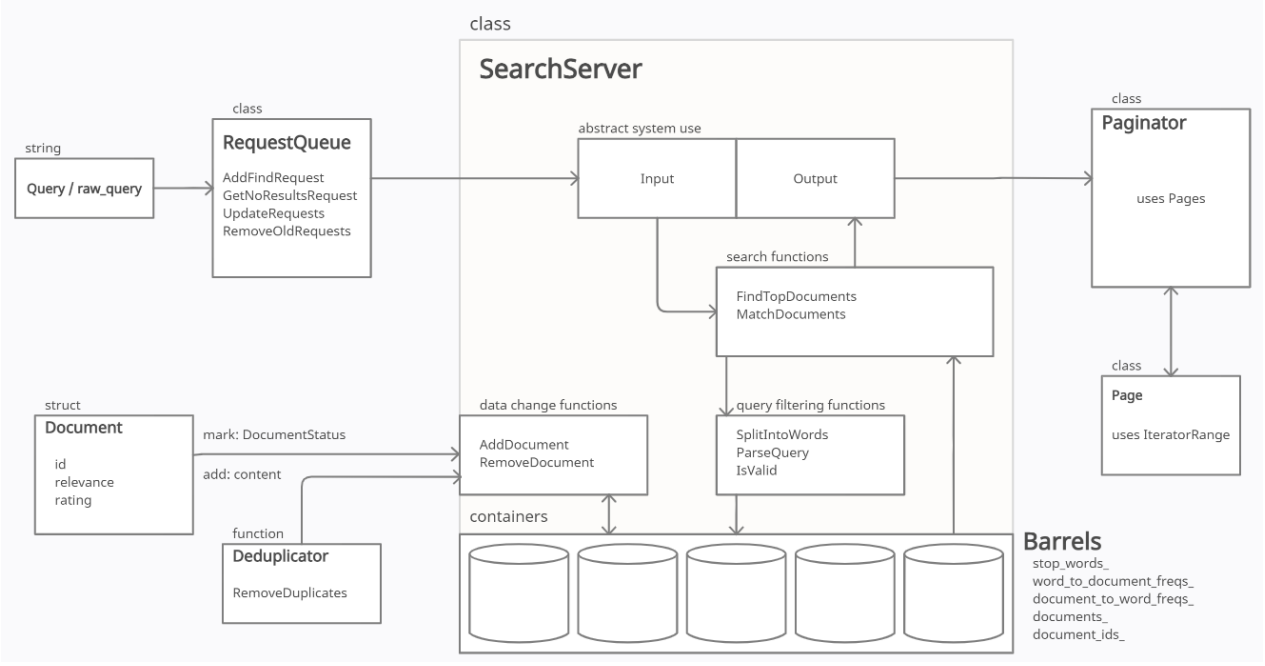


*рис. 3*

На данной модели, отображены все главные сущности и принципы работы поисковой системы: поисковой запрос (*Query*) в виде “*сырой*” строки сначала попадает в сервис обработчик запросов (*RequestQueue*), который выстраивает очередь из поступающих на поисковой сервер запросов. Этот

запрос в свою очередь используют такие поисковые механизмы (*функции*) как: поиск самых релевантных документов (*FindTopDocuments*) и матчинг документов по запросу (*MatchDocuments*). Эти главные поисковые механизмы, в свою очередь содержат методы фильтрации запроса, которые разделяют строку на слова, осуществляют парсинг и валидацию запроса, после чего, функция обращается к имеющимся в бочках данным и выдает результат в виде страниц, которые образует класс (*Paginator*).

Также стоит заметить ветку, использования системы, а именно, класса *SearchServer* не пользователем, а лицом или объектом, с большим правом доступа. Эта ветка содержит функции добавления (*AddDocument*) и удаления документа из системы (*RemoveDocument*), которую также использует функция дедупликации (*RemoveDuplicates*).

## 3.2. Создание файлового дерева

Деление программного кода на различные файлы, помогает работать с проектом, особенно крупным или постоянно масштабируемым, гораздо более организованно. Более того, это делает программную реализацию более гибкой и легкой в обслуживании и тестировании. Программная реализация моей проектной курсовой работы не является исключением; дерево (*рис. 4*).

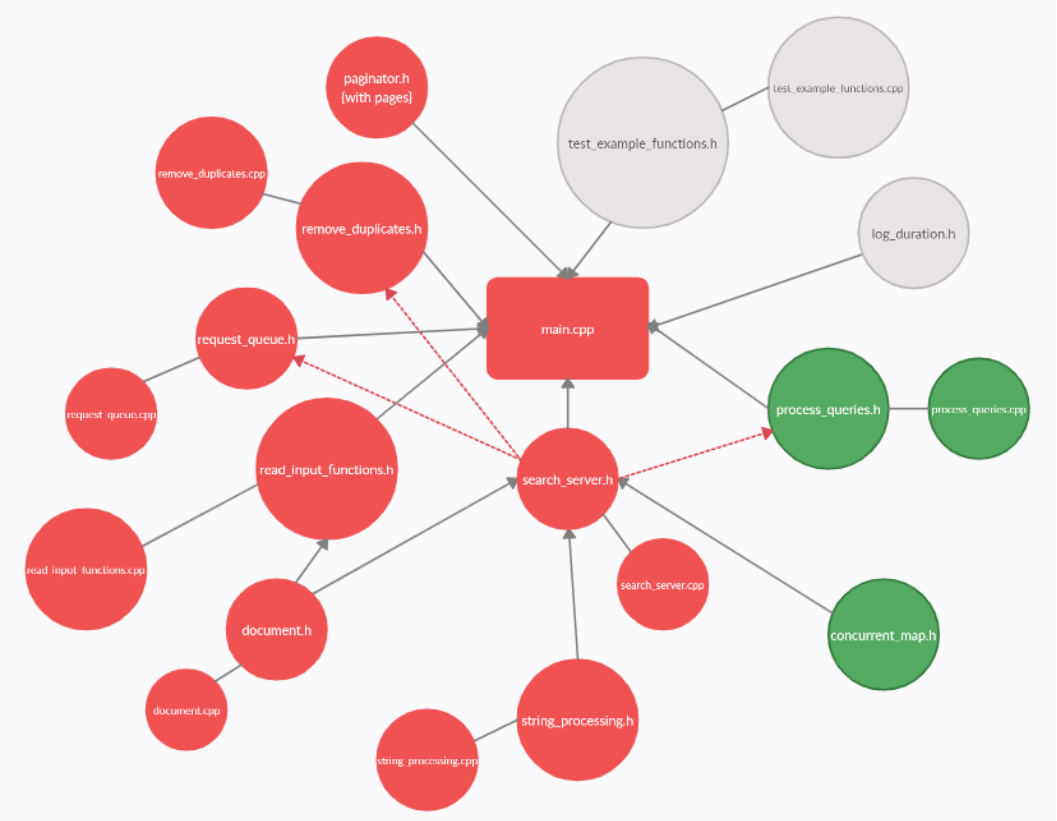


*рис.4*

Стоит сразу сказать, что файлы расширения *.h* (*от англ. "header"*) отличаются от формата *.cpp* тем, что в них лишь объявлены объекты и методы программы, в *.cpp* они же определены (*реализованы*), делается это для того же - улучшения организованности структуры программы.

Красным отмечены файлы, полученные на основании *рис. 1*, в них входят: поисковый сервер (*search_server*), фильтрация строк (*string_processing*), функции ввода для пользования более удобным вводом (*input_functions*), документ (*document*), очередь запросов (*request_queue*), дедупликатор (*remove_duplicates*), условные страницы (*paginator*). Зеленым отмечены файлы, добавленные после применения концепции многопоточности к программной реализации, с целью ее ускорения: многопоточный словарь (*concurrent_map*), многопоточная обработка поисковых запросов (*process_queries*). Серым отмечены добавленные файлы обертки и тестирования (*test_example_functions*) и класс тестирования (*log_duration*). Стоит также обратить внимания на три связи, выделенные красным пунктиром. Сделано это было с целью обозначения работы объектов, являющихся равно уровневых объектов системы, где есть связь использования одного объекта одним или более.

## 3.3. Тестирование системы

### 3.3.1. Юнит тестирование

С самого начала разработки поисковой системы мною применялись различные методы выявления ошибок, такие как: компиляция с помощью невстроенного компилятора *Clang++,* с использованием флагов-санитайзеров (*помогающих команд для поиска ошибок*). Однако, это не должно быть пределом по части безопасности используемой программы, поэтому мною был составлен ряд юнит-тестов (*unit-tests*) для главных механизмов системы еще на этапе одно файлового проекта.

Тесты включат в себя тестирование: исключение стоп-слов при поисковом запросе (*TestExcludeStopWordsFromAddedDocumentContent*), добавлении документов (*TestDocumentAddition*), исключения документов с минус-словами из поискового запроса (*TestExcludeDocumentWithMinusWordsFromSearch*), матчинга

(*TestDocumentMatching*), сортировки документов по релевантности (*TestSortDocumentsByRelevance*), вычисления среднего рейтинга (*TestCountDocumentRating*), тестирования поисковых механизмов (*TestSearchWithPredicate, TestSearchDocumentByStatus*), тестирование работы TF-IDF (*TestRelevanceCalculation*). На *рис. 5* можно увидеть результат проведенного юнит-тестирования.

```
TestExcludeStopWordsFromAddedDocumentContent OK
TestDocumentAddition OK
TestExcludeDocumentWithMinusWordsFromSearch OK
TestDocumentMatching OK
TestSortDocumentsByRelevance OK
TestCountDocumentRating OK
TestSearchWithPredicate OK
TestSearchDocumentByStatus OK
TestRelevanceCalculation OK
Search server testing finished
```

*рис. 5*

### 3.3.2. Измерение времени работы

Для измерения времени работы на разном количестве данных мною был реализован одноименный класс в отдельном файле *log_duration.h*, позволяющий провести временные замеры работы всевозможных частей программной реализации в миллисекундах. Данный файл также использует макросы (*функции-правила упрощающие интерфейс взаимодействия*) для более удобный работы с тестированием поисковой системы.

В данном пункте работы тестируется время работы самого главного механизма поисковой системы – поиску самых релевантных документов и их выдача. Было протестировано время работы данного алгоритма как без применения многопоточности (*seq*), так и с ее применением (*par*). Тестирование проводилось на разном количестве поисковых запросов (*queries*) и *10.000* документов (*documents*) с максимальным количеством слов внутри равном *70*. Для более качественного результата каждое время тестирование было взято как среднее из 3 тестов одного и того же юнита

тестирования. Результаты тестирования можно посмотреть на следующих изображениях (*рис. 6*):


*рис. 6*

Стоить заметить, что средняя разница в скорости выполнения работы на тестируемых данных составила *8–9%* (*ускорение худшего случая*), но также наблюдается процентный пропорциональный рост в выполнении скорости работы при увеличении количества тестируемых значений. Так как мой компьютер не обладает вычислительной мощностью среднестатистического сервера для подобных систем, вычисления происходят не самым быстрым образом. Именно поэтому тестирования на более больших значениях (*10000 поисковых запросов*) я осуществил один раз, чтобы увидеть это растущие различие в скорости работы, и оно составило *14%.*

# ЗАКЛЮЧЕНИЕ

Во время работы над данным курсовым проектом каждый раз я старался смотреть на задачи, которые передо мной вставали с разных сторон, а именно: на разные варианты имплементации решения какой-либо проблемы, возникшей на конкретном этапе работы. Мною был изучен ряд тематических материалов для формирования фундаментальных знаний в нужной сфере, после чего создана программная реализации, конкретно, – поисковой системы. Во время выполнения всей работы я использовал разные утилиты, помогавшие на всех этапах курсовой работы, будь то проектирование, теоретическая часть или тестирование системы.

Конечное проектное решение данной курсовой работы представляет из себя – многофункциональную интегрируемую (*с дальнейшим обновлением*) шаблонную, не булевую поисковую систему, поддерживающую основные принципы фильтрации и ранжирования документов.

# СПИСОК ИСПОЛЬЗОВАННОЙ ЛИТЕРАТУРЫ


1 “History of Search Engines: From 1945 to Google Today” [Электронный ресурс]. – 2017. – URL: http://www.searchenginehistory.com/ (дата обращения 06.04.2021).

2 “The Anatomy of a Large-Scale Hypertextual Web Search Engine” [Электронный ресурс]. – 2000. – URL: http://ilpubs.stanford.edu:8090/361/1/1998-8.pdf (дата обращения 08.04.2021).

3 “Поисковая Система” [Электронный ресурс]. – 2019. – URL: https://bigenc.ru/technology_and_technique/text/3151090 (дата обращения 08.04.2021).

4 “The C++ programming language fourth edition c++11” [Электронный ресурс]. – 2013. – URL: http://index-of.co.uk/Programming/The%20C++%20Programming.Language.4th.Edition.Jun.2013%5BA4%5D.pdf (дата обращения 01.04.2021).

5 “On the provenance of TF-IDF” [Электронный ресурс]. – 2021. – URL: http://sauparna.sdf.org/Search/tf_idf (дата обращения 02.05.2021).

6 “Big O Notation” [Электронный ресурс]. – 2021. – URL: https://en.wikipedia.org/wiki/Big_O_notation (дата обращения 23.04.2021).

7 Документация по языку программирования C++ [Электронный ресурс]. – 2021. – URL: https://en.cppreference.com/w/ (дата обращения 23.04.2021).

# ПРИЛОЖЕНИЯ

**Приложение 1**

Ссылка на код программною реализации проекта

https://github.com/BigBoyMato/SearchServer (доступ по приглашению)