9. Сервис поиска документов. Сервис частых поисковых запросов.  
[habr.com/ru/post/114997/](https://habr.com/ru/post/114997/)  
npgsql.org/efcore/mapping/full-text-search.html  
postgrespro.ru/docs/postgresql/9.6/pgtrgm  
github.com/JakeBayer/FuzzySharp

Тип разрабатываемого сервиса для поиска документов – Scope. Он должен использовать неточный поиск для поиска документа по названию. Сразу стоит заметить, что сам алгоритм поиска (нечёткого поиска) не будет написан собственноручно, будут использоваться встроенные функции SQL, конкретно для PostgreSQL – pg\_trgm.

Также будет разработана служба частых поисковых запросов документов (FrequentDocumentSearchRequestService), её тип, очевидно, Singleton. Идея та же, что и с оптимизацией «последних документов». Как реализовать эту службу? Введём поле FrequentRequests данной службы, которое в виде IEnumerable<KeyValuePair<int Times, Scope<SearchService>>> типа (это – пример, конкретная реализация несет специально написанный класс) будет хранить запрос и ответ в виде экземпляра службы, которая подготовила ответ на заданный запрос и переменную Times, в условных единицах показывающую частоту запроса. Также введем поле RecentRequests, которое будет хранить все запросы за последние N1 минут, но не более N2 штук в виде IEnumerable<Scope<SearchService>>. Либо по триггеру времени, либо по триггеру количества будет вызываться процедура обработки. Во-первых, она группирует повторяющиеся запросы, считая их количество. Естественно, группировка должна происходить с использованием «нечеткого сравнения», ведь «апельсины купить» и «Купить Апельсины» - это один и тот же запрос. В итоге получает запросы и количество данных запросов. Далее, из переменной Times каждого члена FrequentRequsts мы вычитаем единицу, что позволит вытеснять запросы, которые **ранее** были популярны. Далее, если член RecentRequests уже присутствует в FrequentRequests, то добавляем к переменной Times в RecentRequests количество раз встречи данного члена. Если же член отсутствует – просто добавляем его в коллекцию. Далее сортируем по переменной Times, отсекаем до определенного количества.

Выше была описана теоретическая идея, конкретную же реализацию сервиса на момент написания статьи уместить в неё невозможно (более 200 строк).