Индексирование поиска.  
<https://www.youtube.com/watch?v=WJLx1462roU>  
<https://www.npgsql.org/efcore/mapping/full-text-search.html?tabs=pg12%2Cv5>

После попытки реализовать поиск банальным сравнением с помощью функции pg\_trgm, которая сравнивает две токенизированные строки (запрос с каждой записью в БД) стало понятно, что это не вариант – даже при поиске по заголовку документа в таблице из 10 000 записей поиск занимал более **30** секунд. Оно и понятно, pg\_trgm при каждом запросе токенизирует заголовок каждого документа из таблицы, что невероятно медленно. Подготовить токенизированные значения заранее – это и есть индексация. При разработке через EntityFramework мы должны указать автогенерирующуюся колонку. Следующим образом:

modelBuilder.Entity<Website.Models.DocumentModel.DbDocument>().

HasGeneratedTsVectorColumn(p => p.TitleTsVector,"english", p => new { p.Title })

.HasIndex(p => p.TitleTsVector)

.HasMethod("GIN");

Таким образом, при добавлении документа в БД, SQL будет автоматически токенизировать его заголовок. Метод токенизиации и поиска – GIN, оптимизированный для частого извлечения (в отличии от GIST, оптимизированного для частой записи).

Рассмотрим теперь строку запроса:

this.Response = this.DbContextScopeFactory.CreateScope().ServiceProvider

.GetRequiredService<WebsiteContext>().DbDocuments

.OrderByDescending(d => d.TitleTsVector

.Rank(EF.Functions.WebSearchToTsQuery(UserRequest)))

.Take(20).Include("Author").ToList();

Последовательность выполнения простая – создаать экзепляр (Scope) DbContext, запросить источник данных DbDocuments, применить выражение сортировки по совпадению (Rank), выбрать первые 20 записей, загрузить связанную сущность (Author).