Document Search Service

A secure, multi-tenant document indexing and search service built with **Spring Boot**, **PostgreSQL**, **Elasticsearch**, and **Redis**.
The service supports **JWT-based authentication**, **role-based access control**, **tenant isolation**, and **high-performance search**.

Features

- JWT authentication with role-based authorization (USER, ADMIN)
- Multi-tenant architecture with strict data isolation
- Document CRUD operations
- Full-text search powered by Elasticsearch
- Redis caching for faster search responses
- Tenant-level rate limiting
- Structured logging and observability-ready design

Tech Stack

Layer	**Technology**
Backend	Java 17, Spring Boot
Security	Spring Security, JWT
Database	PostgreSQL
Search Engine	Elasticsearch
Cache	Redis
Build Tool	Maven
Containerization	Docker

Architecture Overview

Client

|

| JWT

▼

Spring Boot Application

├── Security & RBAC

├── Tenant Context

├── Rate Limiting

├── Cache Layer

|

├── PostgreSQL (Source of Truth)

├── Elasticsearch (Search Index)

└── Redis (Caching)

API Endpoints

Authentication

POST /auth/token

Document APIs

POST /documents

GET /documents/{id}

DELETE /documents/{id} (ADMIN only)

Search API

GET /search?q={query}&tenant={tenantId}

Sample Requests

Get JWT Token

POST /auth/token

Content-Type: application/json

```
{
  "username": "user1",
```

```
  "password": "password"
}
```

Create Document

```
POST /documents
Authorization: Bearer <JWT>
Content-Type: application/json

{
  "title": "Sample Document",
  "content": "This is a searchable document",
  "tags": ["spring", "search"]
}
```

Search Documents

```
GET /search?q=spring&tenant=tenant1
Authorization: Bearer <JWT>
```

Multi-Tenancy Model

- Tenant ID is embedded in JWT claims
- Extracted and enforced using TenantContext
- All database and search queries are tenant-scoped
- Cross-tenant access is explicitly blocked

Caching Strategy

- Redis used as a read-through cache
- Cached search results with configurable TTL
- Cache invalidation on document create/delete
- Jackson JavaTimeModule enabled for date serialization

Security Model

- Stateless JWT authentication
- Role-based access control at:
    - Route level
    - Method level (@PreAuthorize)
- Enforced tenant isolation
- HTTPS-ready configuration

Consistency Guarantees

- Strong consistency in PostgreSQL
- Eventual consistency between PostgreSQL and Elasticsearch
- Cache TTL ensures freshness while improving performance

Running Locally

Prerequisites

- Java 17+
- Docker & Docker Compose
- Maven

Start Dependencies

docker-compose up -d

Run Application

mvn spring-boot:run

Production Readiness (Design Considerations)

- Horizontal scaling with stateless services
- Elasticsearch index sharding
- Redis cluster support
- Retry and circuit breaker readiness

- Centralized logging & metrics integration
- Zero-downtime deployments

Project Structure

src/main/java

├── config

├── controller

├── security

├── service

├── repository

├── model

└── component

Future Enhancements

- Async indexing with Kafka
- Advanced relevance tuning
- Pagination & sorting support
- Kubernetes deployment manifests
- OpenAPI (Swagger) documentation

Author

Aakanksha Saxena
Senior Java Backend Engineer