Table of Contents

Project description

This project provides REST APIs for basic CRUD operations for Notes and provide below features

1. Notes creation with title , text and tags
2. Notes updation and deletion
3. Checking Notes statistics like number of unique words sorted in descending order of frequency
4. Notes listing with pagination ,sorting and filtration by tags

(API details are at the end)

Technology used

- Java 1.8 (Programming language)
- Spring boot (Application development framework)

- Maven - (Dependency and build management)
- MongoDB (NoSQL DB)
- Lombok - (Annotation based code generation)
- Swagger - (REST API documentation)
- ModelMapper (Objects Mapping)
- Spring boot devtools (Development ease)
- Junit (unit testing)
- Docker (deployment)

Design pattern

Strategy design pattern

I have designed validation functionality using strategy pattern where we have a Abstraction in class ValidationService and clients can change the validation strategy at runtime For example REST Apis are using RestInputValidationService strategy for validation Similarly Web Apps can use something like WebInputValidationService at runtime, we can configure it without affecting our existing code. So here we are changing validation strategy based on client's(caller's) requirement

Also here i have Composition as Validatable so any Entity can be validated using existing code just by implementing this interface

Exception handling

I am following Global Exception handling mechanism in class GlobalExceptionHandler It provides a centralized Exception and error handling mechanism and provides graceful failure and response handling.

API documentation

I am using Swagger to document REST APIs
It can be accessed at URL http://{HOST}:{PORT}/swagger-ui.html
Apart from this code level documentation is done in java comments

Database

Current i am using MongoDB to store notes as document

Logging

For logging i am using Slf4J and log configuration is kept in logback-spring.xml file
As per current configuration it generates logs folder under current directory
And rolling policy is daily or 10mb file size , rolling is done under archived folder

Testing

Unit testing

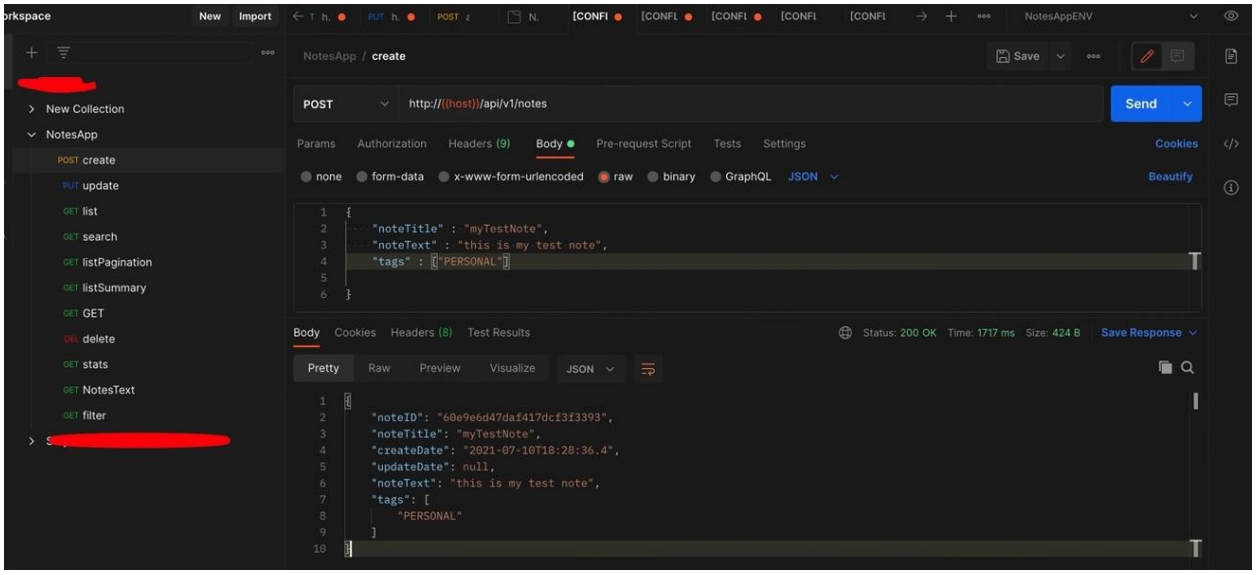
For unit testing my code i am using spring-boot-starter-test which inturn based on JUnit4
All my test cases are located under folder src/test/java

API testing

For API testing i am using Postman REST Client screen-shot below
And sample collection attached that can be downloaded and imported

Postman collection link below :

https://www.getpostman.com/collections/de654d524a6a50631297



API details below :

Below are requirement and respective APIs

- User should to be able to create notes that have Title, Created Date, Text and Tags that can be empty

POST : http://{{host}}/api/v1/notes

REQ :

```
{
  "noteTitle" : "myTestNote",
  "noteText" : "this is my test note",
  "tags" : ["PERSONAL"]
}
```

- The only allowed tags
 are “BUSINESS”, “PERSONAL” and “IMPORTANT” - User should be always able to update and delete notes

Update :
PUT : http://{{host}}/api/v1/notes/{noteID}

```
{
  "noteTitle" : "RahilTestingNote4Updated",
  "noteText" : "hello rahil",
```

```
  "tags" : ["BUSINESS","IMPORTANT"]
}
```

DELETE :
DELETE : http://{{host}}/api/v1/notes/{NoteID}

- The app should allow to obtain stats per each note that would calculate the number of unique words used in the note's text sorted descending.

GET : http://{{host}}/api/v1/notes/stats/{NoteID}

For instance, given a text "note is just a note" it should return a Map as follows {"note": 2, "is": 1, "just": 1, "a": 1}
- The app should not allow to create notes without title or text
- The app should allow listing the notes showing only their "Title" and "Created Date".
Getting the note text should be done in the separate page/request

Listing :
GET http://{{host}}/api/v1/notes/summary?page=0&size=10&filters=IMPORTANT

Get note text :
GET : http://{{host}}/api/v1/notes/text/{NoteID}

- While listing the notes the user should be able to filter by "Tags"

Listing :
GET http://{{host}}/api/v1/notes/summary?page=0&size=10&filters=IMPORTANT

- Notes should always be sorted in the way the user would see the newest first - Listing notes should support pagination as the user might have a lot of notes

Listing :
GET http://{{host}}/api/v1/notes/summary?page=0&size=10&filters=IMPORTANT