DEMO PROJECT

Objective

Create a standalone java application which allows users to manage their favourite recipes. It should allow adding, updating, removing, and fetching recipes. Additionally, users should be able to filter available recipes based on one or more of the following criteria:

1. Whether or not the dish is vegetarian
2. The number of servings
3. Specific ingredients (either include or exclude)
4. Text search within the instructions.

For example, the API should be able to handle the following search requests:

- All vegetarian recipes
- Recipes that can serve 4 persons and have “potatoes” as an ingredient.
- Recipes without “salmon” as an ingredient that has “oven” in the instructions.

Database Design

1. Uses mongo db to persist data.
2. Uses docker-compose to spin up a mongo database and configure a user to connect to database. URL and db name is provided by docker compose environment variable.
3. When running IT, TestAbnTechDemoNosqlApplication.java, is used to run mongo database with testcontainers.

Entity Recipe :

{

id	**string**: auto generated id by mongo db
name*	**string**: manadory field, cannot be null/empty
isVegetarian*	**Boolean:** mandatory boolean value true/false
numberOfServings	**integer**($int32)
instructions*	**string**: instruction text, max size of 1500
ingredients*	**[string]**: array of strings

}

Approach

The solution is designed with these features:

1. Recipe API(RecipeController) to create, update, delete, get recipe by id, get recipes by page, filter recipes (based on number of ingredients, include, and exclude ingredients, text search on instructions).
2. Uses Mongorepository to interact with ‘Recipe’ collection in mongodb database.
3. APIs are secured using JWT Tokens implement using spring-secuity.
4. Authenticate API to generate JWT Tokens (generated using dummy user and password, user management not in the scope of this demo).
5. Uses swagger-ui for easy API documentation.
6. Authenticate API to authenticate users.
7. Uses **testContainers** for testing. Requires *Docker daemon* to be running.
8. Uses **docker-compose** to pull mongo dB and connects to database.

Reference Documentation

1. RecipeController.java: the main controller to interact with recipes data.
    a. createRecipe:
        i. uses RecipeDTO for getting recipe data.
        ii. RecipeDTO uses Jakarta validation to ensure data validation.
        iii. Throws **DuplicateRecipeException** if recipe with same name is created again.
    b. updateRecipe:
        i. required Recipe ID for selecting recipe to update
        ii. uses RecipeDTO for getting recipe data.
        iii. Throws **RecipeNotFoundException** is recipe with given id is not found
    c. deleteRecipe:
        i. required Recipe ID for selecting recipe to delete.
        ii. Throws **RecipeNotFoundException** is recipe with given id is not found
    d. getRecipeById:
        i. required Recipe ID for selecting recipe to fetch
        ii. Throws **RecipeNotFoundException** is recipe with given id is not found
    e. getAllRecipes:
        i. get all recipes. Pagination is enabled
        ii. Provide params *pageNo(starts with 0), size*
    f. filterRecipes:
        i. get api to filter recipes.
        ii. Can use any combination of query params: isVegetarian, numberOfServings, ingredientsToInclude, ingredientsToExclude, instructionTextFilter.
        iii. Uses QueryDSL to generate criteria from given params for querying.
        iv. Throws **InvalidFilterException** if api is called without nay filter parameters.

2. AuthController.java: controller responsible for handling user authentication requests. This is sample service with fixed password (provided BCrypt Encoded string in application.properties: *security.dummy.password*)
    a. createAuthenticationToken:
        i. Accepts a AuthRequest(username, password) object to generate authentication token
        ii. Returns a TokenResponse(jwttoken) object.

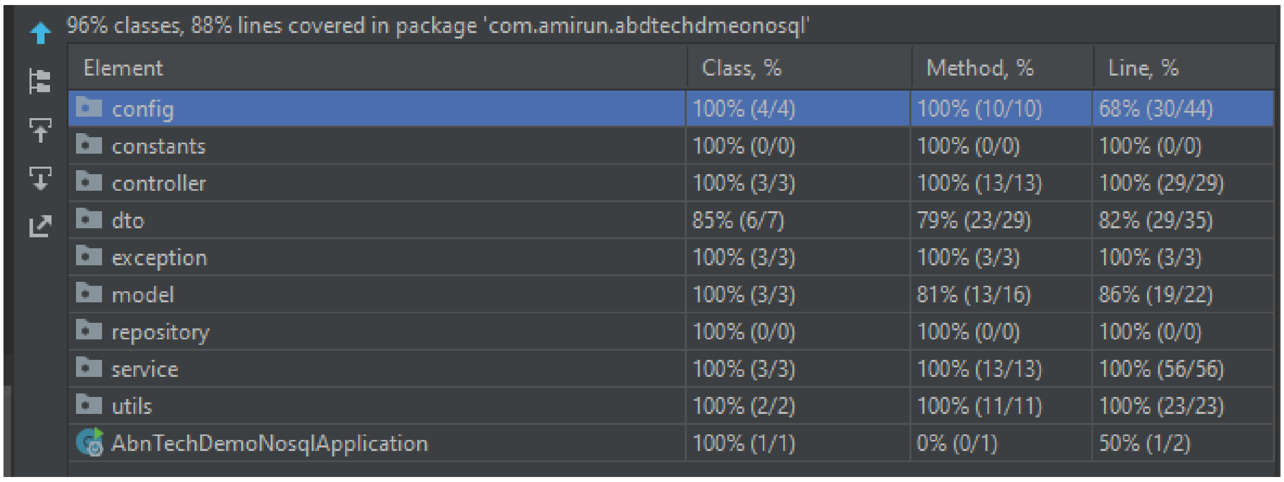
Application.properties

PROPERTY=VALUE	**Description**
logging.level.org.springframework.data.mongodb.core.MongoTemplate=info	Set this to debug to log all mongo queries.
spring.data.mongodb.auto-index-creation=true	Enables creation of indices based on Annotation in entity (Recipe) class

spring.docker.compose.enabled=false	Disables docker compose when starting application directly
spring.docker.compose.file=compose_for_test.yaml	Specifies docker compose file for testcontainers
jwt.secret=overridden-by-docker-env-OHN4u0ag4bWC7zcJPOIzQyZ6n3-bGHug7uTG0XlF9Gvi2p	Random string for encrypting jwt tokens. Usually overridden by external config sources(key vault, environment variables etc.).
security.dummy.password=$2a$12$bxAUJNO6HlNSh.diFJgmou.9NKGYXzGCgLo904fvSN512qGkUdqUS	Dummy BCrypt encoded password string: *pAssword* This is to enable a way to login into the application without user repository. Has been used in **JwtUserDetailsService** for creating dummy user.
jwt.validity.millis=1800000	30 mins(1800 seconds) validity of jwt tokens

Testing

Both UI and IT has been performed:



96% classes, 88% lines covered in package 'com.amirun.abdtechdmeonosql'

Element	Class, %	Method, %	Line, %
config	100% (4/4)	100% (10/10)	68% (30/44)
constants	100% (0/0)	100% (0/0)	100% (0/0)
controller	100% (3/3)	100% (13/13)	100% (29/29)
dto	85% (6/7)	79% (23/29)	82% (29/35)
exception	100% (3/3)	100% (3/3)	100% (3/3)
model	100% (3/3)	81% (13/16)	86% (19/22)
repository	100% (0/0)	100% (0/0)	100% (0/0)
service	100% (3/3)	100% (13/13)	100% (56/56)
utils	100% (2/2)	100% (11/11)	100% (23/23)
AbnTechDemoNosqlApplication	100% (1/1)	0% (0/1)	50% (1/2)

Unit Testing

UT has been performed for below classes. Covers the most important logic in the application. Both positive and negative scenarios have been tested.

1. RecipeControllerTest
2. RecipeServiceImplTest
3. JwtTokenUtilTest
4. RecipeUtilsTest

Integration Testing

IT has been performed for all API endpoints for both success and failure scenarios. We are using MockMVC and testcontainers for running full integration tests along with security.

1. AuthControllerIT
2. RecipeControllerIT