Design Decisions for the Task Management RESTful API

# Introduction:

Objective: To create a RESTful API that serves the purpose of a Task Management System. It provides the user with functionalities such as fetching a(all) task(s) stored in a system, adding new tasks, updating them, deleting a task and marking a task as “completed”.

# Structure of the “Task” Entity:

The “Task” entity is a Java class that has the following attributes:

|  |  |  |
| --- | --- | --- |
| Sr. No. | Attribute Name | Attribute Data Type |
| 1. | id | int |
| 2. | title | String |
| 3. | description | String |
| 4. | status | enum (pending, in\_progress, completed) |
| 5. | due\_date | LocalDate |
| 6. | created\_at | Timestamp |
| 7. | updated\_at | Timestamp |

# Backend Framework:

I decided to create the backend application by using the Java Spring framework. I have previously worked with this framework as part of my summer internship and this framework offered a wide range of functionalities that I needed to implemented the requirements as stated in the problem statement. The technologies that I have used are listed as follows:

1. H2 Database: This serves as the in-memory database for my application. As mentioned in the requirements, the use of an in-memory database was expected which resulted me into choosing the h2 database for the application.
2. Spring Starter Test, Junit Jupiter API, Mockito: These are used for writing the tests for the REST API which has been implemented. I have written the tests for “Getting all the Tasks”, “Adding a New Task” and “Deleting an Existing Task”.
3. Spring Security Web and Config: This was used to implement an authentication for the users and thereby authorizing access to the endpoints of the application. I’ve made use of the Basic Auth method in which the “username” and “password” are passed as part of the headers of the request made to the API. The user, after authentication, is assigned a “USER” role that facilitates authorisation to the API endpoints.

# Folder Structure:

📦

├─ .idea

│  ├─ .gitignore

│  ├─ compiler.xml

│  ├─ encodings.xml

│  ├─ jarRepositories.xml

│  ├─ misc.xml

│  ├─ modules.xml

│  ├─ restAPI.iml

│  ├─ uiDesigner.xml

│  └─ vcs.xml

├─ README.md

├─ Task REST API.postman\_collection.json

└─ restAPI

   ├─ .gitattributes

   ├─ .gitignore

   ├─ .mvn

   │  └─ wrapper

   │     └─ maven-wrapper.properties

   ├─ mvnw

   ├─ mvnw.cmd

   ├─ pom.xml

   └─ src

      ├─ main

      │  ├─ java

      │  │  └─ com

      │  │     └─ example

      │  │        └─ restAPI

      │  │           ├─ Configuration

      │  │           │  └─ SecurityConfig.java

      │  │           ├─ Controller

      │  │           │  └─ TaskController.java

      │  │           ├─ Entity

      │  │           │  └─ Task.java

      │  │           ├─ Enum

      │  │           │  └─ Status.java

      │  │           ├─ Repository

      │  │           │  └─ TaskRepository.java

      │  │           ├─ RestApiApplication.java

      │  │           └─ Service

      │  │              ├─ Implementation

      │  │              │  └─ TaskServiceImplementation.java

      │  │              └─ TaskService.java

      │  └─ resources

      │     └─ application.properties

      └─ test

         └─ java

            └─ com

               └─ example

                  └─ restAPI

                     ├─ RestApiApplicationTests.java

                     └─ TaskServiceImplementationTests.java

1. Configuration: This directory contains the SecurityConfig.java file that is responsible for handling the configuration to facilitate the authentication and authorisation to the API endpoints.
2. Controller: This directory contains the TaskController.java file which has the respective mappings to facilitate the user requests. It makes use of the defined services in the system to provide the functionality to the user as requested.
3. Entity: This directory stores the Task.java file which has the definition for the “Task” class. It contains the attributes along with the getter and setter methods for the same.
4. Enum: This directory contains the Status.java file which defines the three enum values that I have used namely, “pending”, “in-progress” and “completed”.
5. Repository: This directory consists the TaskRepository interface that extends the “JpaRepository”. This is done so that we can make use of the functionalities defined in the extended interface while dealing with the instances “Task” entity in our application.
6. Service: This directory consists of two main files, the first being the TaskService interface which has the abstract methods for this service. Secondly, we have the TaskServiceImplementation.java file that implements these methods by adding the business logic to facilitate the service requirements.

# API Documentation:

1. GET all Tasks: <http://localhost:8080/api/v1/tasks>
2. GET Task by id: [http://localhost:8080/api/v1/tasks/{id}](http://localhost:8080/api/v1/tasks/%7Bid%7D)
3. POST new Task: <http://localhost:8080/api/v1/tasks>  
   (Sample JSON Object: { "title": "Task 1", "description": "Description for the task", "due\_date": "2024-11-25" })
4. PUT Update existing Task: [http://localhost:8080/api/v1/tasks/{id}](http://localhost:8080/api/v1/tasks/%7Bid%7D) (Sample JSON Object: { "title": "Task 1.0", "description": "Description for the task 2.0", "due\_date": "2024-09-09" })
5. DELETE existing Task: [http://localhost:8080/api/v1/tasks/{id}](http://localhost:8080/api/v1/tasks/%7Bid%7D)
6. PATCH Mark Task as completed: [http://localhost:8080/api/v1/tasks/{id}](http://localhost:8080/api/v1/tasks/%7Bid%7D)