Ahmet Tuna Baykal

TeamSec Case Study Documentation

This project is an API service designed for users to perform various operations related to writers, articles, and comments. The system is structured using the Controller-Service-Repository pattern, ensuring a clear separation for managing different entities, including models for each object. This design offers several advantages, like ease of maintenance and better organization of code.

Core Features:

- **Creating Writers:** Users can create a Writer, which allows for subsequent CRUD operations on articles by using the Writer ID.
- **Article Management:** With a valid Writer ID, users can create, retrieve, update, and delete articles.
    - Articles can be fetched as a whole or filtered based on criteria such as "latest-published," "oldest-published," "latest-added," and "recent-added."
    - The "added date" refers to when the article was added to the site, whereas "published date" is the initial publish date of the article content, which writers can add manually.
- **Tagging and Interaction:**
    - Articles contain tags, which are a few keywords representing the content, allowing users to search based on their interests.
    - Users can like and rate articles by providing a Writer ID, furthering engagement with the content.
- **Comments:** Users willing to comment on articles can do so, adding to the interactive aspect of the service.

Data Management and Usage:

- The system utilizes in-memory collection for data storage instead of a traditional database.

- To use the API service, Writers should be created first via HTTP operations. Following this, articles can be created based on the Writers, and then comments can be added to the articles.
- When an object (Writer, Article, Comment) is deleted, its dependencies are automatically removed as well, maintaining data integrity.

Additional Tools:

- **Swagger UI:** Integrated into the system, making it easier to understand and use the service. It can be accessed via **{baseurl}/swagger**, where all APIs are presented clearly.
- **Dockerfile:** Provided for running the service in Docker, simplifying deployment and environment setup. *(Note: Details to be finalized.)*
- **README:** Includes a set of HTTP requests for testing the service in a sequential order, aiding users in quickly getting started with the API.

Running Dockerfile via terminal:

- **From the root of the project;**

```
docker build -t caseteamsec .
```

- **Then;**

```
docker run -d -p 8080:8080 --name teamseccasecon caseteamsec
```

teamseccasecon is the name of the container.

Application runs on 8080 port and the url's inside the ReadME.txt are prepared according to respectively.