StackOverflow Application

Assignment 1 Documentation

Motogna Alexandru

Architecture Diagram:

A picture containing text, electronics

Description automatically generated

* The client requests from the server and receives responses.
* The controller receives the requests and calls the service layer to execute them.
* The repository layer/database accesses the data using dependency injection, which is modelled by the model layer.

Class Diagram:

Diagram

Description automatically generated

(the dependencies from the Model layer have been omitted to keep the diagram more clean).

Database Diagram:

Diagram

Description automatically generated

End points(localhost:8081):

1. User Controller(/users):

* Get all users (/getAll, GET)
* Get user by id (/get, GET, Request parameter id – the user’s id)
* Delete user (/delete, DELETE, Request parameter id – the user’s id)
* Create user (/create, POST, Request body – the user’s object as JSON)
* Update user (/update, PUT, Request body – the user’s object as JSON)

1. Question Controller(/questions):

* Get all questions (/getAll, GET)
* Get question by id (/get, GET, Request parameter id – the question’s id)
* Get all from author by id (/getAllFromUser, GET, Request parameter authorid – the author’s id)
* Delete question (/delete, DELETE, Request parameter id – the question’s id)
* Create question (/create, POST, Request body – the question’s object as JSON, Request Param authorid - the author’s id)
* Update question (/update, PUT, Request body – the question’s object as JSON)
* Add tag to question (/addTag, POST, Request Param qid - the question’s id, Request Param tagid - the tag’s id)
* Remove tag from question (/removeTag, DELETE, Request Param qid - the question’s id, Request Param tagid - the tag’s id)
* Get all votes for a question (/votes, GET, Request parameter id – the question’s id)
* Vote question (upvote or downvote) (/upvote, POST, Request Param qid - the question’s id, Request Param userid - the user’s id who is voting, Request Param upvote – true if upvote or false if downvote)

1. Answer Controller (/answers):

* Get all answers (/getAll, GET)
* Get answer by id (/get, GET, Request parameter id – the answers id)
* Get all from author by id (/getAllFromQuestion, GET, Request parameter qid – the question’s id)
* Delete answer (/delete, DELETE, Request parameter id – the answer’s id)
* Create answer (/create, POST, Request body – the answer’s object as JSON, Request Param authorid - the author’s id)
* Update answer (/update, PUT, Request body – the answer’s object as JSON)
* Get all votes for an answer (/votes, GET, Request parameter id – the answer’s id)
* Vote answer (upvote or downvote) (/upvote, POST, Request Param aid - the answer’s id, Request Param userid - the user’s id who is voting, Request Param upvote – true if upvote or false if downvote)

1. Tag Controller (/tags):

* Get all tags (/getAll, GET)
* Get tag by id (/get, GET, Request parameter id – the tag’s id)
* Create tag (/create, POST, Request body – the tag’s object as JSON)
* Delete tag (/delete, DELETE, Request parameter id – the tag’s id)

Bibliography:

<https://www.baeldung.com/spring-boot-vue-js>

<https://www.javatpoint.com/spring-boot-architecture>

<https://app.diagrams.net/>