Web
Application

Technical Documentation

Prepared By:

Rizel Thomas (497836)

Class: DHIV1.So

Table of Contents

Introduction

Rizel's fun forum is a website that allows users to engage with one another by writing posts and exchanging knowledge about a specific topic. Rizel's fun forum was designed as it is a wonderful way to form social relationships and cultivate a feeling of community.

The application is like a basic forum that allows users to share various posts, ideas, and topics with other users.

There are 3 set of users in the application. Admin, Post creator and regular readers. Admin has access to the entire application and its features. Admin can add, modify, remove other users and their roles in the application. Admin can also create new categories that the posts beings to, which the creator selects while creating a post. Admin can create posts, review and either approve or reject other user's posts. Post creators has the access to view the posts from other users that are approved by the admin and create, edit, delete their own posts. Readers has only the ability to read the posts on the dashboard after login.

Class Diagram

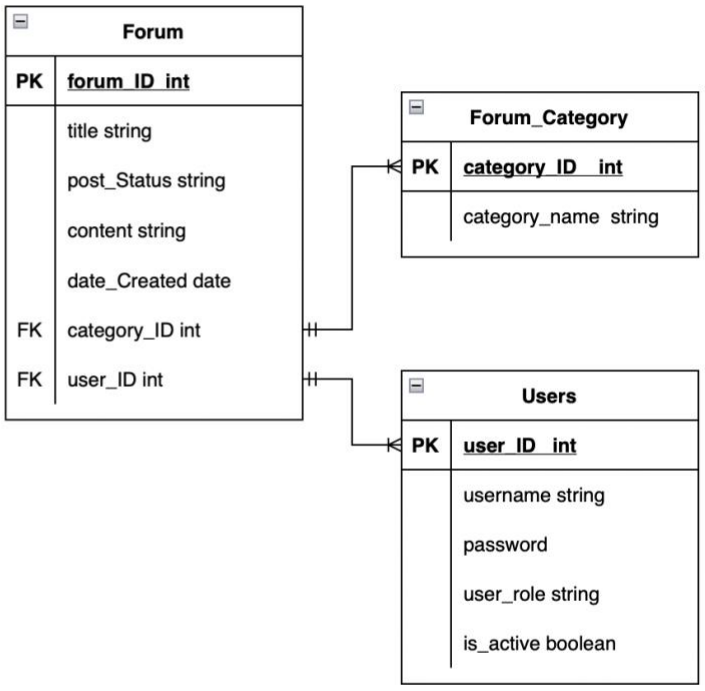

Figure 1:Class diagram for Rizel's fun forum

We have different forum categories and different users, each one with their own attributes. As well, we have different forums, each associated to a single category and a single user.
Each category can be associated to zero, one or many different forums, and also users have the same relationship with forums.
Every class has an ID attribute that identifies itself, so it is unique in all the instances. For example, many categories could share the same name, but not ID.
The attributes category_ID and user_ID of class Forum must correspond to the ID of valid instances of Category and User.

Rest API specification

1. GET requests

GET	/api/v1/users?username=value1&user_role=value2&is_active=value3		
Return list of all users that match the values of the given parameters			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Username	query	Filter Name of the users
	User_role	query	Filter role of the users
	Is_active	query	Filter status of the users
Responses:	**Code**	**Description / example if successful**	
	200	Successful list of users	

GET	/api/v1/users/user_ID		
Return info of user with given ID			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	User_ID*	path	ID of user
Responses:	**Code**	**Description / example if successful**	
	200	Information of user with given ID	
	404	User not found with given ID	

GET	/api/v1/categories?category_name		
Return list of all categories that match the value of the given parameter			
Parameters:	**Name**	**Type**	**Description**

*Add a * to the name of required parameters.*	Category_name	query	Filter name of the categories
Responses:	**Code**	**Description / example if successful**	
	200	Successful list of categories	

GET	**/api/v1/categories/category_ID**		
Return info of category with given ID			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Category_ID*	path	ID of category
Responses:	**Code**	**Description / example if successful**	
	200	Information of category with given ID	
	404	Category not found with given ID	

GET	**/api/v1/forums**		
Return list of all forums that match the values of the given parameters			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Title	path	Filter title of the forums
	Post_Status	path	Filter post status of the forums
	Content	path	Filter content of the forums
	Date_Created	path	Filter date of creation of the forums
	Category_ID	path	Filter category ID of the forums
	User_ID		Filter user ID of the forums
Responses:	**Code**	**Description / example if successful**	
	200	Successful list of forums	

GET	**/api/v1/forums/forum_ID**		
Return info of forum with given ID			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Forum_ID*	path	ID of forum
Responses:	**Code**	**Description / example if successful**	
	200	Information of forum with given ID	
	404	Forum not found with given ID	

2. POST requests

POST	**/ api/v1/forum**		
Create a new forum with the given parameters			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Title	body	Title of the new forum
	Post_Status	body	Post status of the new forum
	Date_Created	body	Creation date of the new forum
	Category_ID*	body	ID of the Category of the new forum
	User_ID*	body	ID of the user of the new forum
	Content		Content of the new forum
Responses:	**Code**	**Description / example if successful**	
	200	The forum was successfully created	
	404	The category ID or user ID do not belong to any found category or user	

POST	**/ api/v1/forum_category**		
Create a new forum category with the given parameter			
Parameters:	**Name**	**Type**	**Description**

*Add a * to the name of required parameters.*	Category_name	body	Name of the new category
Responses:	**Code**	**Description / example if successful**	
	200	The forum category was successfully created	

POST	/ api/v1/users		
Create a new user with the given parameters			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Username	body	Name of the new user
	User_role	body	Role of the new user
	Is_active	body	Active statusof the new user
	Content		Content of the new forum
Responses:	**Code**	**Description / example if successful**	
	200	The forum was successfully created	

3. PUT requests

PUT	/url/users/user_ID		
Update information of user with ID user_ID or create a new one if no existing			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Username	body	New name of the user
	User_role	body	New role of the user
	Is_active	body	New active status of the user
	User_ID*	path	ID of the updated user
Responses:	**Code**	**Description / example if successful**	
	200	User with given ID successfully updated or created	

PUT	/url/forum_categories/category_ID		
Update information of category with ID category_ID or create a new one if no existing			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Category_name	body	New name of the category
	Category_ID	path	ID of the updated category
Responses:	**Code**	**Description / example if successful**	
	200	Category with given ID successfully updated or created	

PUT	/url/forums/forum_ID		
Update information of forum with ID forum_ID or create a new one if no existing			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Title	body	New title of the forum
	Post_status	body	New post status of the forum
	Content	body	New content of the forum
	Date_Created	body	New creation date of the forum
	Category_ID	body	New category of the forum
	User_ID	body	New user ID of the forum
	Forum_ID	path	ID of the updated forum
Responses:	**Code**	**Description / example if successful**	
	200	Forum with given ID successfully updated or created	
	404	User of category with given user_ID or category_ID not found	

4. DELETE requests

DELETE	/url/users/user_ID		
Remove user with given ID and forums with this user			
Parameters:	**Name**	**Type**	**Description**

*Add a * to the name of required parameters.*	User_ID	path	ID of the user to be removed
Responses:	**Code**	**Description / example if successful**	
	200	User with given ID successfully removed	
	404	User with specified ID not found	

DELETE	**/url/forum_categories/category_ID**		
Remove forum category with given ID and forums with this user			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Category_ID	path	ID of the category to be removed
Responses:	**Code**	**Description / example if successful**	
	200	Category with given ID successfully removed	
	404	Category with given ID not found	

DELETE	**/url/forums/forum_ID**		
Remove forum with given ID			
Parameters:	**Name**	**Type**	**Description**
*Add a * to the name of required parameters.*	Forum_ID	path	ID of the forum to be removed
Responses:	**Code**	**Description / example if successful**	
	200	Forum with given ID successfully removed	
	404	Forum with given ID not found	

Sequence diagrams

1. POST new entity

Use case description:

An admin wants to create a new post with a new category and assign it to itself. A new category entity and a new post one will be created.

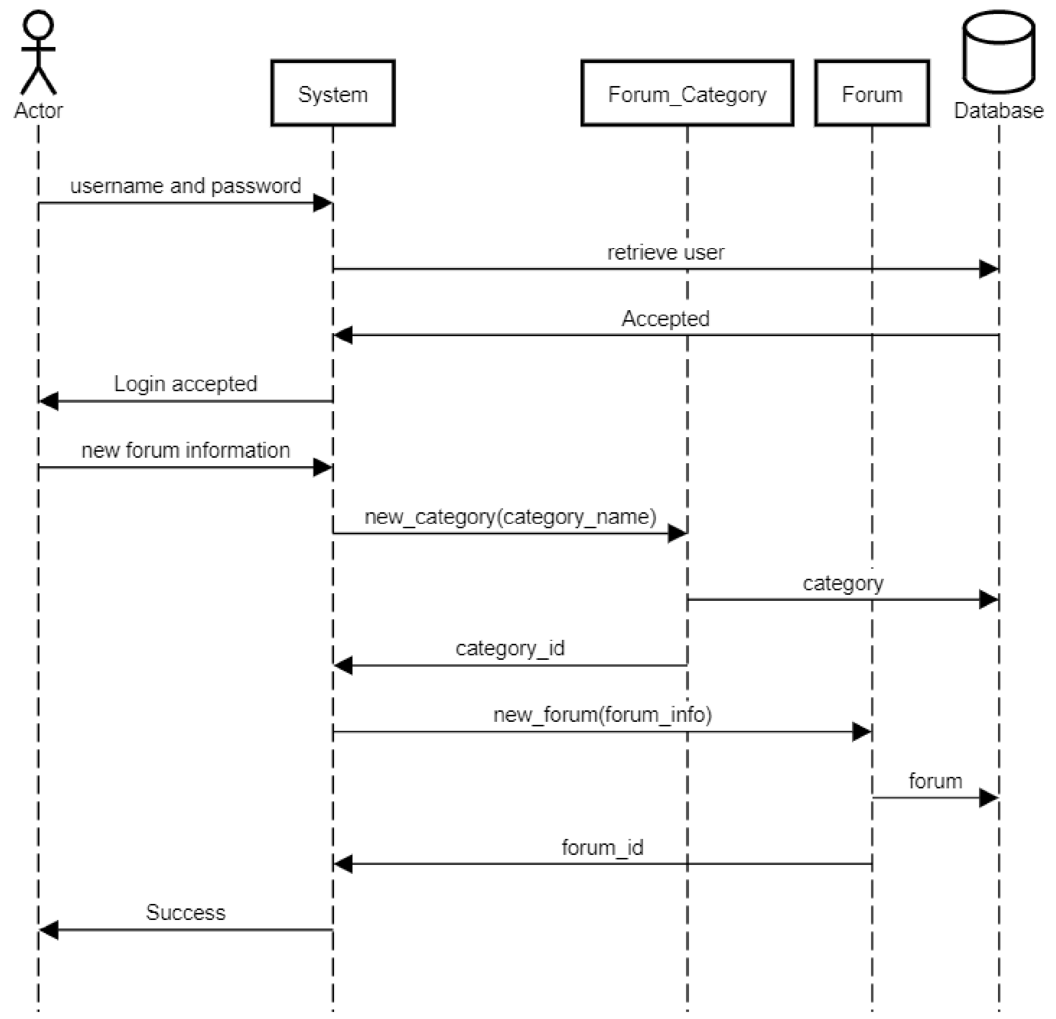

Figure 2: POST API Sequence diagram

2. Fetch sequence diagram

Use case info description:

Get the names of the users that participates in a forum category that meets the specified name.

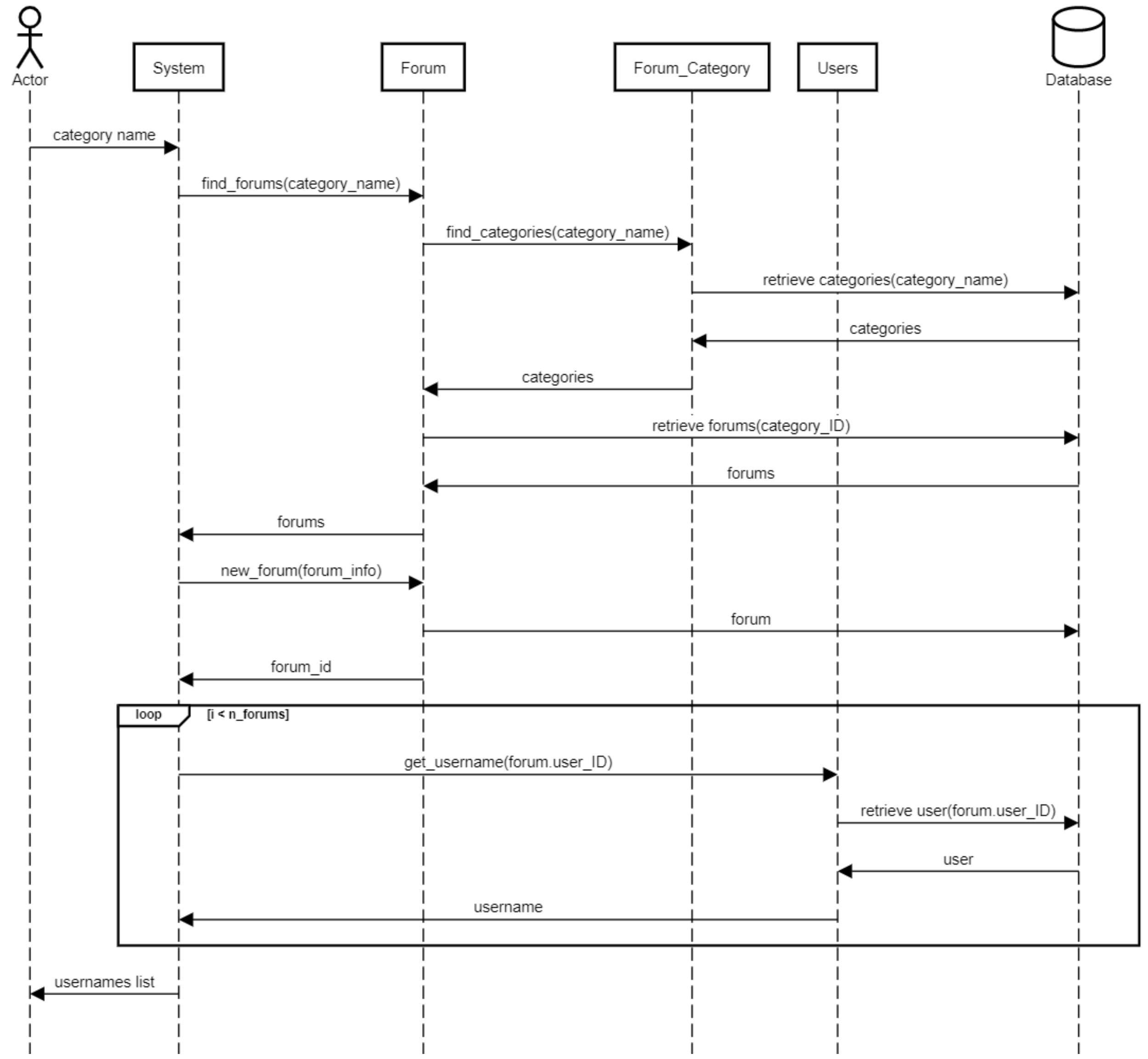
Actor
System
Forum
Forum_Category
Users
Database
category name
find_forums(category_name)
find_categories(category_name)
retrieve categories(category_name)
categories
categories
retrieve forums(category_ID)
forums
forums
new_forum(forum_info)
forum
forum_id
loop
[i < n_forums]
get_username(forum.user_ID)
retrieve user(forum.user_ID)
user
username
usernames list

Figure 3: FETCH API Sequence Diagram

Justification of choices

H2 database was used to connect to the spring boot backend. H2 is an embedded database and can run with less configuration compared to other relational databases. Hibernate with H2 server is 2.5 times quicker than Hibernate with MySQL server in this instance, according to a comparison of the normalized speeds of Hibernate with MySQL database server and Hibernate with H2 database server .

Test report

To make sure the system I created works the way it’s supposed to, I tested the system after each step to find errors early and minimize them. I also located a few willing participants and assigned each one of them a particular duty to do on the system, this helped me test multiple things on the system out.