Database Table Details:-

create database blog

use blog

select \* from blog\_post

select \* from category

select \* from comment

select \* from liked\_posts

select \* from disliked\_posts

select \* from post\_categories

select \* from user

select \* from user\_blocked\_users

select \* from user\_following

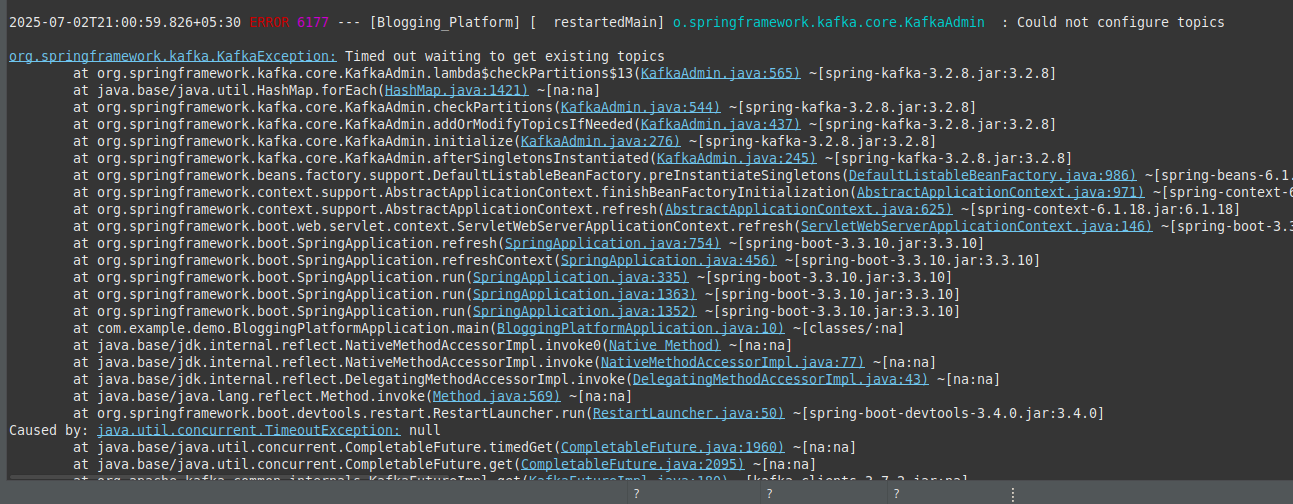
select \* from user\_roles

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

* Public APIs - All GET APIs and Login API
* There you will see two projects - i) Blogging\_Platform ii)AdminTool

And for their communication I have used Kafka.

So for any reason if we **don’t have Kafka running** still you **can run the** **Blogging\_Platform**. You will get some errors only,



But you can ignore them and wait for the Started BloggingPlatformApplication message .It will work and run all the APIs only. You will not be able to use Kafka Controller APIs.

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# REST API

Swagger URL to Access REST APIs

<http://localhost:8080/swagger-ui/index.html>

User Controller

* This is the starting point of the application.First need to create a user.
* Permitted roles - ADMIN,EDITOR,USER (USER < EDITOR < ADMIN)
* Currently supporting one role per participant
* If you give any other role it gives “Invalid Role” error with 400 Bad Request
* ADMIN has all permissions but he/she can not delete another ADMIN

API - /users/register (POST)

Sample request body

| {  "username": "sreetama",  "password": "1234",  "email": "sree@gmail.com",  "bio": "exploring my life",  "roles": [  "ADMIN"  ]  } |
| --- |

Sample response

| {  "id": 1,  "username": "sreetama",  "followers": 0,  "following": 0,  "bio": "exploring my life",  "totalPosts": 0,  "roles": [  "ROLE\_ADMIN"  ],  "email": "sree@gmail.com",  "blogPosts": [],  "comments": []  } |
| --- |

2)You can also update your details later on

API - /users (PUT)

Example : changing the bio of the user

Request Body -

| {  "id": 1,  "username": "sreetama",  "password": "1234",  "email": "sree@gmail.com",  "bio": "changing my life",  "roles": [  "ADMIN"  ]  } |
| --- |

Response:-

| {  "id": 1,  "username": "sreetama",  "followers": 0,  "following": 0,  "bio": "changing my life",  "totalPosts": 0,  "roles": [  "ROLE\_ADMIN"  ],  "email": "sree@gmail.com",  "blogPosts": [],  "comments": []  } |
| --- |

3)Get by user id

API - /users/{id} [GET]

| {  "id": 1,  "username": "sreetama",  "followers": 0,  "following": 0,  "bio": "changing my life",  "totalPosts": 0,  "roles": [  "ROLE\_ADMIN"  ],  "email": "sree@gmail.com",  "blogPosts": [],  "comments": []  } |
| --- |

4) Also you can delete any use, below are the rules -

* User can not delete itself
* Editor can delete itself only
* Admin can delete itself and any other editor or user

API - /users/{id} [DELETE]

| curl -X 'DELETE' \  'http://localhost:8080/users/1' \  -H 'accept: \*/\*' \  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTE0NTE2NiwiZXhwIjoxNzUxMTQ4NzY2fQ.3LGGBsUAfRQpSTV0yPVUNk1YdOXLBHZhi46TggCX3Fw' |
| --- |

Response :-

| {  "id": 1,  "username": "sreetama",  "followers": 0,  "following": 0,  "bio": "changing my life",  "totalPosts": 0,  "roles": [  "ROLE\_ADMIN"  ],  "email": "sree@gmail.com",  "blogPosts": [],  "comments": []  } |
| --- |

5)We can fetch blog posts of a particular user by title of the blogpost

API - /users/{userId}/posts (GET)

BlogPost Controller

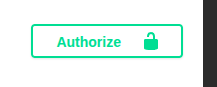
Now let's create a blog post!

1)Login (POST /login)

| {  "id": 1,  "token": "eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTA5MDg1MiwiZXhwIjoxNzUxMDk0NDUyfQ.wXpfHozcOelnetwOC7Z6pp0-8Ex-OV7YUi3GjJe1dJ8",  "role": "[ROLE\_ADMIN]"  } |
| --- |

The token will be valid for 1hr.

2)Use the jwt token here -



Click on this and paste the jwt token

3)Use - POST /blogs

* ROLE should be - ADMIN or EDITOR

Sample Request Payload -

| {  "title": "A Day of my life",  "content": "Eating,Sleeping,Watching movies and...it goes on",  "categories": [  {  "name": "#life"  }  ]  } |
| --- |

Response :-

| {  "id": 2,  "title": "A Day of my life",  "content": "Eating,Sleeping,Watching movies and...it goes on",  "author": {  "username": "sreetama",  "id": 1,  "email": "sree@gmail.com"  },  "categories": [  {  "id": 1,  "name": "#life"  }  ],  "comments": [],  "createAt": "2025-06-28T21:06:15.005933666",  "updateAt": "2025-06-28T21:06:15.005950071",  "likes": 0,  "dislikes": 0  } |
| --- |

4) You can update the blogpost

* Before update - "title": "A Day of my life"
* Want to update - "title": "A day of my life"

Here you have to provide the id of the blog.

API - Use - PUT /blogs

Request Payload

| {  "id":2,  "title": "A day of my life",  "content": "Eating,Sleeping,Watching movies and...it goes on",  "categories": [  {  "name": "#life"  }  ]  } |
| --- |

Response-

| {  "id": 2,  "title": "A day of my life",  "content": "Eating,Sleeping,Watching movies and...it goes on",  "author": {  "username": "sreetama",  "id": 1,  "email": "sree@gmail.com"  },  "categories": [  {  "id": 1,  "name": "#life"  }  ],  "comments": [],  "createAt": "2025-06-28T21:06:15.005934",  "updateAt": "2025-06-28T21:08:34.289629065",  "likes": 0,  "dislikes": 0  } |
| --- |

**5)To fetch any blogpost details we can use below 2 APIs -**

1)Get by user id - it will show you all the blog posts of any participant.

In user controller

API - users/{userId}/posts

2)Get by title and user id - it will show you blog posts of that user with filtering by title of the blog post

API - blogs/title/{title}/user/{userId}

6)Can delete a blog post by id

Use - DELETE /blogs/{id}

| curl -X 'DELETE' \  'http://localhost:8080/blogs/2' \  -H 'accept: \*/\*' \  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTE0NTE2NiwiZXhwIjoxNzUxMTQ4NzY2fQ.3LGGBsUAfRQpSTV0yPVUNk1YdOXLBHZhi46TggCX3Fw' |
| --- |

Response-

| {  "id": 2,  "title": "A day of my life",  "content": "Eating,Sleeping,Watching movies and...it goes on",  "author": {  "username": "sreetama",  "id": 1,  "email": "sree@gmail.com"  },  "categories": [  {  "id": 1,  "name": "#life"  }  ],  "comments": [],  "createAt": "2025-06-28T21:06:15.005934",  "updateAt": "2025-06-28T21:08:34.289629",  "likes": 0,  "dislikes": 0  } |
| --- |

If you delete a blogpost then all the comments linked with the blogpost will also got **deleted**.

Category Controller

Now comes to category,

* So while creating a blogpost if you have noticed we are giving some categories and yes one blogpost can be linked to any number of categories minimum 1 is mandatory.
* If any particular category is not present while creating the blogpost that category will also be created at the sametime

But we have one separate category controller as well you can do the below operation-

1)Get all the category

API - /categories [GET]

2)Create any category later on you can use this while creating your blog post

API - /categories [POST]

3)Can see blog posts those are linked with this category

| curl -X 'GET' \  'http://localhost:8080/categories/name?name=life' \  -H 'accept: \*/\*' |
| --- |

Response :-

| [  {  "id": 2,  "title": "A day of my life",  "content": "Eating,Sleeping,Watching movies and...it goes on",  "author": {  "username": "sreetama",  "id": 1,  "email": "sree@gmail.com"  },  "categories": [  {  "id": 1,  "name": "#life"  }  ],  "comments": [],  "createAt": "2025-06-28T21:06:15.005934",  "updateAt": "2025-06-28T21:08:34.289629",  "likes": 0,  "dislikes": 0  }  ] |
| --- |

4) You can also delete a category

API - /categories/{id} [DELETE]

Note:-

Only any **ADMIN** can delete a category and only if that category is **not linked** with any blogposts

| curl -X 'DELETE' \  'http://localhost:8080/categories/1' \  -H 'accept: \*/\*' \  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTE0NTE2NiwiZXhwIjoxNzUxMTQ4NzY2fQ.3LGGBsUAfRQpSTV0yPVUNk1YdOXLBHZhi46TggCX3Fw' |
| --- |

Response -

| {  "id": 1,  "name": "#life",  "blogPosts": []  } |
| --- |

Comment Controller

Now let’s add some comments to the blogpost

1. Add comment to blogs

API - /comments [ POST ]

* Yes, the creator of that blogpost can add comments.
* Any other user [ROLE - USER,EDITOR] can also add comments to each other’s post

Curl of the request:-

| curl -X 'POST' \  'http://localhost:8080/comments?blogPostId=2' \  -H 'accept: \*/\*' \  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTEyNDkxMywiZXhwIjoxNzUxMTI4NTEzfQ.3YfFYpYK1V4JsGnH8WDvpr18ILWZuDE2lg2CCrJr7Es' \  -H 'Content-Type: application/json' \  -d '{    "content": "Same here!"  }' |
| --- |

Response-

| {  "id": 2,  "title": "A day of my life",  "content": "Eating,Sleeping,Watching movies and...it goes on",  "author": {  "username": "sreetama",  "id": 1,  "email": "sree@gmail.com"  },  "categories": [  {  "id": 1,  "name": "#life"  }  ],  "comments": [  {  "content": "Nice Content!",  "id": 1,  "createAt": "2025-06-28T21:33:44.518696"  },  {  "content": "Same here!",  "id": 2,  "createAt": "2025-06-28T21:38:56.328994394"  }  ],  "createAt": "2025-06-28T21:06:15.005934",  "updateAt": "2025-06-28T21:08:34.289629",  "likes": 0,  "dislikes": 0  } |
| --- |

2)To fetch any particular comment details by it’s id use the below API

API - /comments/{id} [GET]

3)You can also update a comment need to provide the **id** of the comment,

And obviously everyone can update only their own comments and updated comment will be identified with “**(edited**)”

API - /comments/ [PUT]

| curl -X 'PUT' \  'http://localhost:8080/comments?blogPostId=2' \  -H 'accept: \*/\*' \  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTEyNDkxMywiZXhwIjoxNzUxMTI4NTEzfQ.3YfFYpYK1V4JsGnH8WDvpr18ILWZuDE2lg2CCrJr7Es' \  -H 'Content-Type: application/json' \  -d '{  "id":2,  "content": "Nice Content!"  }' |
| --- |

Response -

| {  "id": 2,  "title": "A day of my life",  "content": "Eating,Sleeping,Watching movies and...it goes on",  "author": {  "username": "sreetama",  "id": 1,  "email": "sree@gmail.com"  },  "categories": [  {  "id": 1,  "name": "#life"  }  ],  "comments": [  {  "content": "Nice Content!",  "id": 1,  "createAt": "2025-06-28T21:33:44.518696"  },  {  "content": "**Nice Content!(edited)**",  **"id": 2,**  "createAt": "2025-06-28T21:38:56.328994"  }  ],  "createAt": "2025-06-28T21:06:15.005934",  "updateAt": "2025-06-28T21:08:34.289629",  "likes": 0,  "dislikes": 0  } |
| --- |

4)You can even delete your comment -

API - /comments /{cId}/blogposts/{bpId} [DELETE]

| curl -X 'DELETE' \  'http://localhost:8080/comments/2/blogposts/2' \  -H 'accept: \*/\*' \  -H 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTEzNTk5MiwiZXhwIjoxNzUxMTM5NTkyfQ.4iwK7HEBBMt45-rFOo-pPhrnsbVj7lOFmeFIbyO8Qpg' |
| --- |

Response-

| {  "content": "Nice Content!(edited)",  "id": 2,  "createAt": "2025-06-28T21:38:56.328994"  } |
| --- |

# GraphQL API

URL - <http://localhost:8080/graphql>

APIs -

**1)Pagination - apply pagination to list of blogposts**

query

| curl --location 'http://localhost:8080/graphql' \  --header 'Content-Type: application/json' \  --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTEyOTMyMywiZXhwIjoxNzUxMTMyOTIzfQ.9OM2IeIyKUmtQYsOjDHarpJnfFRmQ7Nnzz5zLK-ZAGI' \  --data '{"query":"query Pagination{\n getPosts(page: 0,size:2){\n content\n }\n}\n","variables":{}}' |
| --- |

Response -

| {  "data": {  "getPosts": [  {  "content": "Eating,Sleeping,Watching movies and...it goes on"  }  ]  }  } |
| --- |

**2)Search - search by a keyword**

query-

| curl --location 'http://localhost:8080/graphql' \  --header 'Content-Type: application/json' \  --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTEyOTMyMywiZXhwIjoxNzUxMTMyOTIzfQ.9OM2IeIyKUmtQYsOjDHarpJnfFRmQ7Nnzz5zLK-ZAGI' \  --data '{"query":"query Search{\n searchPosts(keyword:\"day\"){\n title,\n content\n \n }\n}\n","variables":{}}' |
| --- |

Response-

| {  "data": {  "searchPosts": [  {  "title": "A day of my life",  "content": "Eating,Sleeping,Watching movies and...it goes on"  }  ]  }  } |
| --- |

**3)GetPinnedPostsOfTheUser - to get all the pinned posts of a user**

query -

| curl --location 'http://localhost:8080/graphql' \  --header 'Content-Type: application/json' \  --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTEyOTMyMywiZXhwIjoxNzUxMTMyOTIzfQ.9OM2IeIyKUmtQYsOjDHarpJnfFRmQ7Nnzz5zLK-ZAGI' \  --data '{"query":" query GetPinnedPostsOfTheUser {\n getPinnedPostsOfTheUser(uId:1){\n pinnedDate,\n blogPostResponse {\n content\n }\n }\n}\n\n","variables":{}}' |
| --- |

Response-

| {  "data": {  "getPinnedPostsOfTheUser": [  {  "pinnedDate": "2025-06-28T22:24:46.113719",  "blogPostResponse": {  "content": "Eating,Sleeping,Watching movies and...it goes on"  }  }  ]  }  } |
| --- |

**4)pinnedPost - pin a post**

query-

| curl --location 'http://localhost:8080/graphql' \  --header 'Content-Type: application/json' \  --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTEyOTMyMywiZXhwIjoxNzUxMTMyOTIzfQ.9OM2IeIyKUmtQYsOjDHarpJnfFRmQ7Nnzz5zLK-ZAGI' \  --data '{"query":"\nmutation PinAPost {\n pinnedPost(uId:1 , bpId: 2 ){\n pinnedDate\n blogPostResponse {\n content\n }\n }\n}\n","variables":{}}' |
| --- |

Response-

| {  "data": {  "pinnedPost": {  "pinnedDate": "2025-06-28T22:24:46.113718725",  "blogPostResponse": {  "content": "Eating,Sleeping,Watching movies and...it goes on"  }  }  }  } |
| --- |

**5)getFollowers - to get all the followers of a user**

query-

| curl --location 'http://localhost:8080/graphql' \  --header 'Content-Type: application/json' \  --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTEyOTMyMywiZXhwIjoxNzUxMTMyOTIzfQ.9OM2IeIyKUmtQYsOjDHarpJnfFRmQ7Nnzz5zLK-ZAGI' \  --data '{"query":"query GetFollowers{\n getFollowers(uId:1){\n username\n }\n}","variables":{}}' |
| --- |

Response-

| {  "data": {  "getFollowers": []  }  } |
| --- |

**6)getFollowings - to get all the followings of a user**

query:

| curl --location 'http://localhost:8080/graphql' \  --header 'Content-Type: application/json' \  --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTEyOTMyMywiZXhwIjoxNzUxMTMyOTIzfQ.9OM2IeIyKUmtQYsOjDHarpJnfFRmQ7Nnzz5zLK-ZAGI' \  --data '{"query":"query GetFollowings{\n getFollowings(uId:1){\n id\n username\n }\n}","variables":{}}' |
| --- |

Response -

| {  "data": {  "getFollowings": [  {  "id": "2",  "username": "sabitri"  }  ]  }  } |
| --- |

**7)TrendingPosts - getting top 10 posts based on maximum likes**

query -

| curl --location 'http://localhost:8080/graphql' \  --header 'Content-Type: application/json' \  --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTEyOTMyMywiZXhwIjoxNzUxMTMyOTIzfQ.9OM2IeIyKUmtQYsOjDHarpJnfFRmQ7Nnzz5zLK-ZAGI' \  --data '{"query":"query TrendingPosts{\n trendingPosts{\n content,\n createAt\n }\n}","variables":{}}' |
| --- |

Response-

| {  "data": {  "trendingPosts": [  {  "content": "Eating,Sleeping,Watching movies and...it goes on",  "createAt": "2025-06-28T21:06:15.005934"  }  ]  }  } |
| --- |

**8)userLikedPost - to get the posts those are liked by a user**

query:

| curl --location 'http://localhost:8080/graphql' \  --header 'Content-Type: application/json' \  --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTEyOTMyMywiZXhwIjoxNzUxMTMyOTIzfQ.9OM2IeIyKUmtQYsOjDHarpJnfFRmQ7Nnzz5zLK-ZAGI' \  --data '{"query":"query UserLikedPost{\n userLikedPost(uId:1){\n content,\n createAt\n }\n}","variables":{}}' |
| --- |

Response:

| {  "data": {  "userLikedPost": [  {  "content": "Eating,Sleeping,Watching movies and...it goes on",  "createAt": "2025-06-28T21:06:15.005934"  }  ]  }  } |
| --- |

**9)setReaction - set a reaction to post (i.e.like(true) or dislike(false))**

query:

| curl --location 'http://localhost:8080/graphql' \  --header 'Content-Type: application/json' \  --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTEyOTMyMywiZXhwIjoxNzUxMTMyOTIzfQ.9OM2IeIyKUmtQYsOjDHarpJnfFRmQ7Nnzz5zLK-ZAGI' \  --data '{"query":"mutation SetReaction {\n setReaction(\n request : {\n bpId: 2,\n uId: 1,\n reaction: true\n }\n ) {\n content\n createAt\n }\n}\n","variables":{}}' |
| --- |

Response-

| {  "data": {  "setReaction": {  "content": "Eating,Sleeping,Watching movies and...it goes on",  "createAt": "2025-06-28T21:06:15.005934"  }  }  } |
| --- |

**10)followOrUnfollowAuthor - to follow or unfollow a user**

If a user “Ram” is following another user “Raju” and now if Ram hit the API we will assume

Ram wants to unfollow Raju.

query-

| curl --location 'http://localhost:8080/graphql' \  --header 'Content-Type: application/json' \  --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTEyOTMyMywiZXhwIjoxNzUxMTMyOTIzfQ.9OM2IeIyKUmtQYsOjDHarpJnfFRmQ7Nnzz5zLK-ZAGI' \  --data '{"query":"mutation FollowOrUnFollowAuthor{\n followOrUnFollowAuthor(follower: 1, followee:2){\n username\n }\n}","variables":{}}' |
| --- |

Response -

| {  "data": {  "followOrUnFollowAuthor": [  {  "username": "sabitri"  },  {  "username": "sreetama"  }  ]  }  } |
| --- |

-> Sreetama is following Sabitri

**11)blockUser - block a user**

If a user (Ram) was following another user (Raju) and now for any reason Ram wants to block Raju so now he(i.e.Raju) will add in the block users list and ALSO from the following list Raju will be deleted.

And from Raju’s follower’s list one user (Ram) will get removed

query-

| curl --location 'http://localhost:8080/graphql' \  --header 'Content-Type: application/json' \  --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTEyOTMyMywiZXhwIjoxNzUxMTMyOTIzfQ.9OM2IeIyKUmtQYsOjDHarpJnfFRmQ7Nnzz5zLK-ZAGI' \  --data '{"query":"mutation BlockUser{\n blockUser (blockerId: 1, blockedUserId:2){\n username\n }\n}","variables":{}}' |
| --- |

Response -

| {  "data": {  "blockUser": [  {  "username": "sreetama"  },  {  "username": "sabitri"  }  ]  }  } |
| --- |

Now if you see the followings for sreetama, you will get empty list

| curl --location 'http://localhost:8080/graphql' \  --header 'Content-Type: application/json' \  --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTEyOTMyMywiZXhwIjoxNzUxMTMyOTIzfQ.9OM2IeIyKUmtQYsOjDHarpJnfFRmQ7Nnzz5zLK-ZAGI' \  --data '{"query":"query GetFollowings{\n getFollowings(uId:1){\n id\n username\n }\n}","variables":{}}' |
| --- |
| {  "data": {  "getFollowings": []  }  } |

**12)blockedUsers - to see list of all the blocked users**

query-

| curl --location 'http://localhost:8080/graphql' \  --header 'Content-Type: application/json' \  --header 'Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJzcmVldGFtYSIsInJvbGVzIjpbIkFETUlOIl0sImlhdCI6MTc1MTEyOTMyMywiZXhwIjoxNzUxMTMyOTIzfQ.9OM2IeIyKUmtQYsOjDHarpJnfFRmQ7Nnzz5zLK-ZAGI' \  --data '{"query":"query BlockedUsers{\n blockedUsers (uId:1){\n id\n username\n }\n}","variables":{}}' |
| --- |

Response -

| {  "data": {  "blockedUsers": [  {  "id": "2",  "username": "sabitri"  }  ]  }  } |
| --- |

# Kafka

Kafka Controller

AdminTool application is involved with BlogPost application. To run this endpoint you need to run **zookeeper**, **kafka** and finally the **AdminTool application** otherwise you may face some unexpected errors.

API - /admintool [GET]

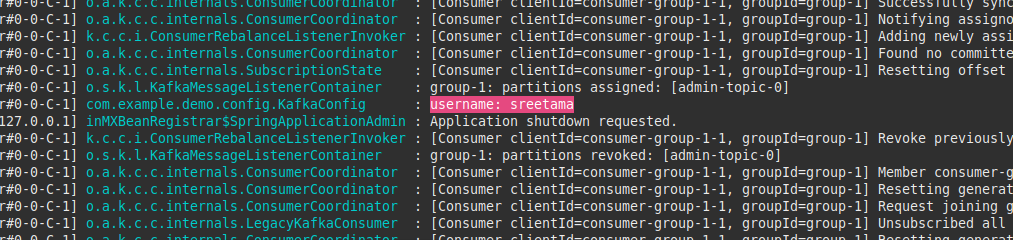
| curl -X 'GET' \  'http://localhost:8080/admintool' \  -H 'accept: \*/\*' |
| --- |

This is a special kind of controller.It is like a monitoring service.

After completing authorization if you hit this API it will send the authenticated user name in the console.

Refer the screenshot for the name-

**username: sreetama**



From the API you will see in the response with 200 status code -

| Sending the username, check the console |
| --- |

If you have not authenticated and hit the API the below message will show-

| no user currently logged in... |
| --- |

Point to be noted - To run this controller your zookeeper and kafka should be running.

======================================================================

* **I have also added the Junit test cases for all the APIs**

**Thank you**