Spring Boot Twitter Challenge

Description

As an Endava customer, we want to have an application like **twitter** to be able to use it internally in our company.

To start this project, we want to have basic functionalities, and depending on the needs, we will add more requirements in the future. As a first step, we want an **API** that exposes some endpoints so in the future the front-end team can use them.

Requirements

First set up:

- We already have some tweets that must be saved in the database.

User must be able to:

- Create account
- Login/Logout
- Add/Modify/Delete tweets
- Add/Remove like
- See timeline and be able to sort and set filter

Follow the next endpoint table as a guide to those requirements:

Important note: User ID must be sent in the header. If it doesn't include, it should return 401

HTTP Method	Path	Request Body	Description

?	**/signup**	<pre>{
		"first_name": "Pepito",
		"last_name": "Perez",
		"username": "pepito2022",
		"password": "hello123"
		}</pre>
			• Cannot add user if username already exists.
			• There are no password restrictions.
			• No need to encrypt the password.

?	**/login**	<pre>{
		"username": "pepito2022",
		"password": "hello123"
		}</pre>
			• Cannot login if user does not exist.
			• There are no password restrictions.
			• No need to encrypt the password.

?	**/logout**		• Show message when the user logout

POST	**/tweets**	<pre>{
		"text": "This is my first tweet",
		"image": "https://image.png"
		}</pre>
			• 280 characters limit
			• The text can be plain text
			• Save the date and time of tweet

GET
PATCH
DELETE
	/tweets/{tweet_id}	PATCH
		<pre>{
		"text": "This is my first tweet",
		"image": "https://image.png"
		}</pre>
			PATCH
			• 280 characters limit
			• The text can be plain text
			• Update the date and time of the tweet
			• Can update if you're the user tweeted.
			• User ID must be sent in the header. If it doesn't include, it should return 401
			DELETE
			• Show the message that notify the tweet was removed.
			• Can remove if you're the user tweeted.
			• User ID must be sent in the header. If it doesn't include, it should return 401
			GET

			• Show all information of the tweet • User ID must be sent in the header. If it doesn't include, it should return 401
POST	**/favorites/create?id={tweet_id}**		• The tweet with id {tweet_id}, must update the array favorite and favorite_count (see json sample) • Show the message that you put like
POST	**/favorites/destroy?id={tweet_id}**		• The tweet with id {tweet_id}, must update the array favorite and favorite_count (see json sample) • Show the message that you remove like
GET	**/tweets?page=?&limit=?&sort=?&user=?&search=?**		Show all tweet that saved in the data base. But this endpoint must be able to filter and sort from query param. Filter • Page: Optional field. The default value must be 1. • Limit: Optional field. The default value must be 100. • User: Optional field. If you have more than one user, separate them with a comma (,). • Search: Optional field. If you have more than one keyword, separate them with a comma (,). Sort • Newest: sort from new to old tweets. • Oldest: sort from old to new tweets.

			• Recommended: sort from tweet that has more likes to less likes. Example /tweet/all?page=2&limit=10&sort=newest&search=corona,pcr,hospital
POST	**/load/tweets**		Load initials tweets file and save to the database.

Tweet JSON sample:

```
{
    "created_at": "Thu Apr 06 15:24:15 +0000 2017",
    "last_modified_at": "Thu Apr 06 15:24:15 +0000 2017",
    "id_str": "850006245121695744",
    "text": "This is my first tweet",
    "image": "https://image.png",
    "favorite": [
        123,
        456,
        789
    ],
    "favorite_count": 3,
    "user": {
        "id": 123,
        "name": "Pepito Perez",
        "username": "pepito2022"
    }
}
```

User JSON sample:

```
{
    "id": 123,
    "name": "Pepito Perez",
    "username": "pepito2022",
    "favorites": [850006245121695744],
    "friends": [456,789]
}
```

Bonus

- Follow and unfollow friends
- See all followers
- See only favorites tweets
- Have unit test with more than 50% coverage
- Tweet data load efficiency

HTTP Method	Path	Request Body	Description
POST	**/followers/create?id={user_id}**		
POST	**/followers/destroy?id={user_id}**		• When you unfollow the friend, you can't put like in the user comment.
GET	**/followers**		
GET	**/favorites/?id={tweet_id}**		• Be able to see all tweet you set as like. • Be sure that when the original tweet was removed, this tweet must not seem here.

Rubric

Criteria	Description	Did	Comment
Create account	/signup		
Login	/login		
Logout	/logout		

Add new tweet	/tweets		
See specific tweet	/tweets/{tweet_id}		
Update specific tweet	/tweets/{tweet_id}		
Remove tweet	/tweets/{tweet_id}		
Add like in the specific tweet	/favorites/create? id={tweet_id}		
Remove like in the specific tweet	/favorites/destroy? id={tweet_id}		
Load initial tweets	/load/tweets		
Get timeline filtering by pagination	/tweets?page=		
Get timeline filtering by limit	/tweets?limit=		
Get timeline filtering by user(s)	/tweets?user=		
Get timeline filtering by keyword(s)	/tweets?search=		
Get timeline sorting	/tweets?sort=		
Get timeline filtering by all query param	/tweets?page=&limit=&user= &search=&search=		
Training concepts	Use all concepts learned in the trainings: concurrency, Java stream, exception handling		
Architecture	The application is using good practice of architecture.		

Clean code	The application is using good practice of clean code.		
Unit test	Coverage more than 30%		
Use of HTTP	Good practice of HTTP methods and HTTP status.		
Documentation	Use swagger		
Bonus	Follow friend		
	Unfollow friend		
	See all followers		
	See only favorite tweets		
	Have unit test with more than 50% coverage		
	Tweets data load efficiency		