*2. Describe P0 test cases in plain text about **CREATE**, **UPDATE** and **DELETE** operations.*
*To **CREATE** a resource you need to fill the next mandatory fields in the body of the request: title, body, userId.*
*To **UPDATE** a resource you need to fill the necessary field(s) in the body of the request.*
***UPDATE** and **DELETE** operations are possible by the id of the resource.*

Summary	Steps	Expected result
Operation for **creating** a new object	1. The user fills in the **title** and **body** fields 2. Sends a request to the server 3. The server accepts the request 4. The server gets the user's **userId** from the database 5. The server creates a POJO class and generates an **id** for the object 6. The server saves the object in the database 7. The server sends a response to the user that he **created** the object.	A new object was **created** with the specified **title** and **body** fields in the database

Operation for **updating** a new object	1. The user fills in the **id**, **title**, and **body** fields 2. Sends the request to the server. 3. The server accepts the request 4. The server retrieves the user's **userId** from the database. 5. The server gets the object from the database by **userId** and **id** 6. Modifies the object and adds it to the database. 7. The server sends a response to the user that he changed the object.	The object **updated** in the database
When trying to **update** an object, an error occurs that such an object is missing or does not belong to the user	1. The user fills in the **id**, **title**, and **body** fields 2. Sends the request to the server. 3. The server accepts the request 4. The server	An error that there is no such object in the database or does not belong to the user

	retrieves the user's **userId** from the database. 5. The server by **userId** and **id** makes a request to the database to get the object. 6. The server does not receive the object from the database. 7. The server sends a response with an error stating that such an object does not belong to the user or is not in the database.	
When trying to update an object, it turns out that there is no such object in the database, and a new object is added.	1. The user fills in the **id**, **title**, and **body** fields 2. Sends the request to the server. 3. The server accepts the request 4. The server retrieves the user's **userId** from the	A new object is **created** in the database with the specified fields

	database. 5. The server by **userId** and **id** makes a request to the database to get the object. 6. The server does not receive the object from the database. 7. The server adds a new object (meaning that there is no object with this **id** in the database) 8. And sends a response that the object has been changed.	
Operation for **deleting** an object	1. The user specifies the object **id** and sends a request to the server. 2. The server accepts the request 3. The server retrieves the user's **userId** from the database. 4. The server uses the **userId** and **id**	The object is being **deleted** from the database

	to make a request to the database to delete the object. 5. The server deletes the object 6. And sends a response that the object has been deleted.	
When you try to **delete** an object an error occurs that the object does not belong to the user or there is no such object in the database	1. The user specifies the object **id** and sends a request to the server. 2. The server accepts the request 3. The server retrieves the user's **userId** from the database. 4. The server uses the **userId** and **id** to make a request to the database to delete the object. 5. The database responds that there is no such object or it does not belong to the user 6. The server	Error when **deleting** an object

	sends an error message that there is no such object	

3. Create a bug report for the next issue:
The request GET /posts/0 returned 404 Not Found, but you expect an empty list with 200 OK

The request Get/posts/0

Expected outcome: An empty list with 200 OK

Actual outcome: Returned 404 Not Found

What's the problem: The client expects an empty sheet with the status 200 from us and the server sends 404 Not Found and this may be handled incorrectly by the clients