

# **Java Spring Boot Backend Code API Description and Sample Postman Requests and Responses**

**Writer:** E.Asankarayigit

# AUTHENTICATION CONTROLLER LAYER METHODS TEST

## RESULT

### User Registration

---

```
@PostMapping("/register")
public ResponseEntity<UserDTO> registerUser(@RequestBody
RegisterRequest registerRequest) {
    try {
        UserDTO createdUser = userService.registerUser(
            registerRequest.getName(),
            registerRequest.getSurname(),
            registerRequest.getEmail(),
            registerRequest.getUsername(),
            registerRequest.getPassword()
        );
        return
ResponseEntity.status(HttpStatus.CREATED).body(createdUser);
    } catch (RuntimeException e) {
        return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body(null);
    }
}
```

---

### Java Spring Boot Code

URL: <http://localhost:8080/api/auth/register>

Method Type: POST

#### Sample JSON File 1:

```
{
    "name": "Ali",
    "surname": "Veli",
    "email": "ali.veli@example.com",
    "username": "aliveli",
    "password": "sifre123"
}
```

#### Sample JSON File 2:

```
{
    "username": "johndoe",
    "email": "john.doe@example.com",
    "password": "password123",
    "name": "John",
}
```

```
"surname": "Doe"
}
```

DB Screen after both files are sent:

	owner_id	email	name	password	surname	username
1	1	john.doe@example.com	John	\$2a\$10\$bKZYvD/YjqAuHLegBCPIRuWjQKshbd2r0sNTbPMDMux2hXZgCY3q	Doe	johndoe
2	2	ali.veli@example.com	Ali	\$2a\$10\$yASgfiTjDxKsBS2ltGxsOoaOxjvFgPi41Kra75UCOoTOj9pM8eSC	Veli	aliveli

## Login Operation

```
@PostMapping("/login")
public ResponseEntity<LoginResponse> loginUser(@RequestBody
LoginUserDto loginUserDto) {
    try {
        String token = authService.loginUser(loginUserDto.getEmail(),
loginUserDto.getPassword());
        LoginResponse loginResponse = new LoginResponse("Bearer " +
token, jwtUtil.getEXPIRATION_TIME());
        return ResponseEntity.ok(loginResponse);
    } catch (Exception e) {
        return
ResponseEntity.status(HttpStatus.UNAUTHORIZED).body(null);
    }
}
```

## Java Spring Boot Code

URL: <http://localhost:8080/api/auth/login>

Method Type: POST

Sample JSON File 1:

```
{
    "email": "aliveli",
    "password": "sifre123"
}
```

Sample Response:

```
{
    "token": "Bearer
eyJhbGciOiJIUzI1NiJ9.eyJzdWUiOiJhbGJlZWxpliwiWF0ljoXNzQzNzQ0NzU4LlJleHAI
OjE3NDM3ODA3NTth9.-feFcTg7RoXq22j0aJKnnT-lfKbJU-qezwXPBBYJeN0",
    "expiresIn": 36000000
}
```



## Update e-mail

```
@PutMapping("/update-email/{username}")
public ResponseEntity<UserDTO> updateEmail(@PathVariable String
username, @RequestBody String newEmail) {
    try {
        UserDTO updatedUser = userService.updateEmail(username,
newEmail);
        return ResponseEntity.ok(updatedUser);
    } catch (RuntimeException e) {
        return
ResponseEntity.status(HttpStatus.NOT_FOUND).body(null);
    }
}
```

### Java Spring Boot Code

URL: <http://localhost:8080/api/users/update-email/{username}>  
(Example: <http://localhost:8080/api/users/update-email/aliveli> )

Method Type: PUT

Token Type: Bearer (Example Token:

"eyJhbGciOiJIUzI1NiJ9.eyJzdWl0OiJhbGl2ZWxpliWF0IjoxNzQzNzY5NjE1LCJleHAiOiJlE3NDM4MDU2MTV9.VeUTbMhbi96BYjz\_7RbLNpID1x1\_xuOsyjog1xl3ryc")

Sample JSON File 1:

[aliveli@gmail.com](mailto:aliveli@gmail.com)

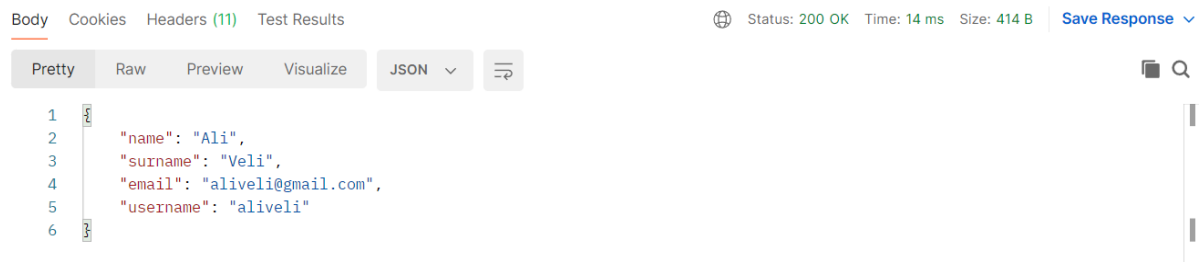
Sample JSON File 2:

[aliveli@hotmail.com](mailto:aliveli@hotmail.com)

### During Sending Process in Postman:

The screenshot shows the Postman interface for a PUT request. The URL bar contains "http://localhost:8080/api/users/update-email/aliveli". The "Authorization" tab is selected, showing "Bearer To..." with a dropdown arrow. The "Token" field contains the token "eyJhbGciOiJIUzI1NiJ9.eyJzdWl0OiJhbGl2ZWxpliWF0IjoxNzQzNzY5NjE1LCJleHAiOiJlE3NDM4MDU2MTV9.VeUTbMhbi96BYjz\_7RbLNpID1x1\_xuOsyjog1xl3ryc". Below the token field, there is a note: "The authorization header will be automatically generated when you send the request. Learn more about authorization". The "Send" button is visible in the top right corner.

After request is sent, email data field belongs to user will be updated:



## Update Password

```

@PutMapping("/update-password/{username}")
public ResponseEntity<String> updatePassword(@PathVariable String
username,
                                           @RequestParam String
currentPassword,
                                           @RequestParam String
newPassword) {
    try {
        String result = userService.updatePassword(username,
currentPassword, newPassword);
        return ResponseEntity.ok(result);
    } catch (RuntimeException e) {
        return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body(e.getMessage());
    }
}

```

## Java Spring Boot Code

URL: <http://localhost:8080/api/users/update-password/{username}>(RequestParam)

(Example: <http://localhost:8080/api/users/update-password/aliveli>  
?currentPassword=sifre123&newPassword=123456789)

Method Type: PUT

Token Type: Bearer (Example Token:

"eyJhbGciOiJIUzI1NiJ9.eyJzdWUiOiJhbG12ZWxpIiwiaWF0IjoxNzQzNzY5NjE1LCJleHAiOiJENzNDMDU2MTV9.VeUTbMhbi96BYjz\_7RbLNpID1x1\_xuOsyjog1xl3ryc")

### During Sending Process in Postman (RequestParam (Key & Value)):

PUT

http://localhost:8080/api/users/update-password/aliveli?currentPassword=sifre123&newPassword=123456789

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	currentPassword	sifre123	
<input checked="" type="checkbox"/>	newPassword	123456789	
	Key	Value	

**After request is sent, password data field belongs to user will be updated if and only if current password is provided correctly:**

A screenshot of a web browser's developer console. The top bar shows 'Body', 'Cookies', 'Headers (11)', and 'Test Results'. On the right, it displays 'Status: 200 OK', 'Time: 202 ms', 'Size: 363 B', and a 'Save Response' button. Below this, there are tabs for 'Pretty', 'Raw', 'Preview', and 'Visualize', with 'Pretty' selected. To the right of these tabs is a 'Text' dropdown menu and a refresh icon. The console log shows a single entry: '1 Password updated successfully'.

## Update Username

```
@PostMapping("/update-username/{oldUsername}")
public ResponseEntity<UserDTO> updateUsername(@PathVariable String
oldUsername, @RequestBody String newUsername) {
    try {
        UserDTO updatedUser = userService.updateUsername(oldUsername,
newUsername);
        return ResponseEntity.ok(updatedUser);
    } catch (RuntimeException e) {
        return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body(null);
    }
}
```

## Java Spring Boot Code

URL: <http://localhost:8080/api/users/update-username/{username}>  
(Example: <http://localhost:8080/api/users/update-username/123ahaliveli> )

**Method Type:** PUT

**Token Type:** Bearer (*Example Token:*

“eyJhbGciOiJIUzI1NiJ9.eyJzdWUiOiJhbGl2ZWxpIiwiaWF0IjoxNzQzNzY5NjE1LCJleHAiOiJlE3NDM4MDU2MTV9.VeUTbMhbi96BYjz\_7RbLNpId1x1\_xuOsyjog1xl3ryc”)

## During Sending Process in Postman (RequestParam (Key & Value)):

PUT ▼ http://localhost:8080/api/users/update-username/123ahaliveli Send ▼

Params Authorization ● Headers (9) Body ● Pre-request Script Tests Settings Cookies

Type Bearer To... ▼ Token eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJhMjNhG...

The authorization header will be automatically generated when you send the request.  
[Learn more about authorization](#)

Params Authorization ● Headers (9) Body ● Pre-request Script Tests Settings

● none ● form-data ● x-www-form-urlencoded ● raw ● binary JSON ▼

1 veliali

## DB Before Sending Request:

Grid	owner_id	email	name	password	surname	username
1	2	john.doe@example.com	John	\$2a\$10\$02/ot.pWNR0oKHxniyIWvurOYQpW05kbW0hf7onEva/Hr.khOxl86	Doe	johndoe
2	1	aliveli@gmail.com	Ali	\$2a\$10\$wQjMFJH9DaOnzECReqDP1ubudkxDaFO9zfZdpOJAjg9pTUs4jwFbO	Veli	123ahaliveli

## Response After Sending Request:

Body Cookies Headers (11) Test Results 🌐 Status: 200 OK Time: 191 ms Size: 468 B Save Response ▼

Pretty Raw Preview Visualize Text ↵

1 eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ2ZWxpYWxpIiwiaWF0IjoxNzQzODc5Nzc5LCJleHAiOiJlE3NDM5MTU3Nz19.EXQGYxz1aWVeMstCGxDpn80tfcwN7H7x

New token will be created in db screen,

## DB After Sending Request:

Grid	owner_id	email	name	password	surname	username
1	2	john.doe@example.com	John	\$2a\$10\$02/ot.pWNR0oKHxniyIWvurOYQpW05kbW0hf7onEva/Hr.khOxl86	Doe	johndoe
2	1	aliveli@gmail.com	Ali	\$2a\$10\$wQjMFJH9DaOnzECReqDP1ubudkxDaFO9zfZdpOJAjg9pTUs4jwFbO	Veli	veliali



## Update Name & Surname

```
@PostMapping("/update-name/{username}")
public ResponseEntity<UserDTO> updateNameAndSurname(@PathVariable
String username,
                                                    @RequestBody
UserDTO updatedInfo) {
    try {
        UserDTO updatedUser = userService.updateNameAndSurname(
            username,
            updatedInfo.getName(),
            updatedInfo.getSurname()
        );
        return ResponseEntity.ok(updatedUser);
    } catch (RuntimeException e) {
        return
        ResponseEntity.status(HttpStatus.NOT_FOUND).body(null);
    }
}
```

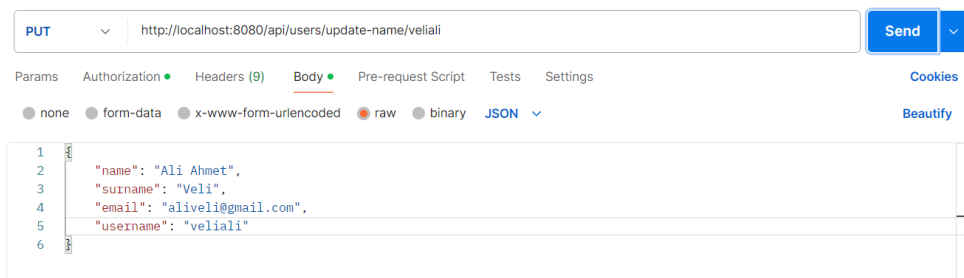
### Java Spring Boot Code

URL: <http://localhost:8080/api/users/update-name/{username}>

Method Type: PUT

Token is again necessary, but new one must be used if username is updated.

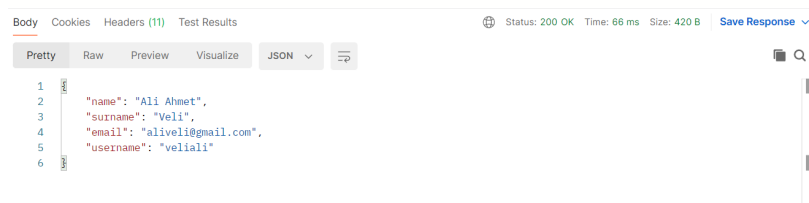
During Sending Process in Postman (RequestParam (Key & Value)):



DB Before Sending Request:

users						
	owner_id	email	name	password	surname	username
1	2	john.doe@example.com	John	\$2a\$10\$02/ot.pWNR0oKHxniyWvurOYQpWO5kbW0hf7onEva/Hr.khOxI86	Doe	johndoe
2	1	aliveli@gmail.com	Ali	\$2a\$10\$wQjMFJH9DaOnzECReqDP1ubudkoDaF09zfZdpOIAJg9pTUs4jwFbO	Veli	veliali

Response After Sending Request:



**DB After Sending Request:**

users Enter a SQL expression to filter results (use Ctrl+Space)							
	owner_id	email	name	password	surname	username	
1	2	john.doe@example.com	John	\$2a\$10\$02/ot.pWNR0oKHxniyWvurOYQpW05kbW0hf7onEva/Hr.khOxl86	Doe	johndoe	
2	1	aliveli@gmail.com	Ali Ahmet	\$2a\$10\$wQjMFJH9DaOnzECReqDP1ubudKxDaFO9zfZdpOJAjg9pTUs4JwFbO	Veli	veliali	

**Get User Tags**

```

@GetMapping("/getTags")
public ResponseEntity<?> getUserTags(@RequestHeader("Authorization")
String authorizationHeader) {
    try {
        // Extract token from "Bearer <token>"
        if (authorizationHeader == null ||
!authorizationHeader.startsWith("Bearer ")) {
            return ResponseEntity.status(HttpStatus.UNAUTHORIZED)
                .body("Authorization header missing or invalid");
        }
        String token = authorizationHeader.replace("Bearer ", ""); //
Remove "Bearer " prefix

        // Call UserService to get tags
        List<TagDTO> tags = userService.getUserTags(token);
        return ResponseEntity.ok(tags);

    } catch (UsernameNotFoundException e) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
            .body("User not found: " + e.getMessage());
    } catch (JwtException e) {
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED)
            .body("Invalid or expired token: " + e.getMessage());
    } catch (Exception e) {
        return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
            .body("An error occurred: " + e.getMessage());
    }
}

```

**Java Spring Boot Code**

URL: <http://localhost:8080/api/users/getTags>

Method Type: GET

## SCREEN SENDING PROGRESS:

GET ⌵ http://localhost:8080/api/users/getTags Send ⌵

Params Authorization **Headers (7)** Body Pre-request Script Tests Settings Cookies

Headers Hide auto-generated headers

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	Postman-Token ①	<calculated when request is sent>	
<input checked="" type="checkbox"/>	Host ①	<calculated when request is sent>	
<input checked="" type="checkbox"/>	User-Agent ①	PostmanRuntime/7.43.3	
<input checked="" type="checkbox"/>	Accept ①	*/*	
<input checked="" type="checkbox"/>	Accept-Encoding ①	gzip, deflate, br	
<input checked="" type="checkbox"/>	Connection ①	keep-alive	
<input checked="" type="checkbox"/>	Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ2ZWxpYWxpIiwiaWF0Ijox...	

## Response:

Body **Cookies** Headers (11) Test Results Status: 200 OK Time: 88 ms Size: 606 B Save Response ⌵

Pretty Raw Preview Visualize JSON ⌵ 🔍

```

1  {
2    {
3      "tagId": 3,
4      "user": {
5        "name": "Ali Ahmet",
6        "surname": "Veli",
7        "email": "aliveli@gmail.com",
8        "username": "veliali"
9      },
10     "todos": [],
11     "tagName": "aybu"
12   },
13   {
14     "tagId": 2,
15     "user": {
16       "name": "Ali Ahmet",
17       "surname": "Veli",
18       "email": "aliveli@gmail.com",
19       "username": "veliali"
20     },
21     "todos": [],
22     "tagName": "project"

```

# TAG CONTROLLER LAYER METHODS TEST RESULT

## Create a Tag

```
@PostMapping("/create")
public ResponseEntity<?> createTag(@RequestParam String tagName,
@RequestHeader("Authorization") String token) {
    try {
        TagDTO createdTag = tagService.createTag(tagName, token);
        return
        ResponseEntity.status(HttpStatus.CREATED).body(createdTag);
    } catch (Exception e) {
        return
        ResponseEntity.status(HttpStatus.BAD_REQUEST).body("Error occurred
        during Tag creation");
    }
}
```

### Java Spring Boot Code

URL: (Example: <http://localhost:8080/api/tags/create?tagName=cloudcomputing> )

Method Type: POST

POST
http://localhost:8080/api/tags/create?tagName=cloudcomputing
Send

Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies

Query Params

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	tagName	cloudcomputing	
	Key	Value	

### Response:

Body
Cookies
Headers (11)
Test Results

Status: 201 Created
Time: 148 ms
Size: 484 B
Save Response

Pretty
Raw
Preview
Visualize
JSON

```

1  {
2    "tagId": 1,
3    "user": {
4      "name": "Ali Ahmet",
5      "surname": "Veli",
6      "email": "aliveli@gmail.com",
7      "username": "veliali"
8    },
9    "toDos": null,
10   "tagName": "cloudcomputing"
11 }
```

DB:

tags Enter a SQL expression to filter results (use Ctrl+Space)

	123 tag_id	123 user_id	A-Z tag_name
1	1	1	cloudcomputing

## Delete a Tag

```
@DeleteMapping("/delete/{tagId}")
public ResponseEntity<?> deleteTag(@PathVariable Long tagId,
@RequestHeader("Authorization") String token) {
    try {
        String result = tagService.deleteTag(tagId, token);
        return ResponseEntity.status(HttpStatus.OK).body(result);
    } catch (Exception e) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Tag
not found or unauthorized");
    }
}
```

## Java Spring Boot Code

Example URL: <http://localhost:8080/api/tags/delete/1>

Method Type: DELETE

Before sending request:

DELETE http://localhost:8080/api/tags/delete/1

Params	Authorization	Headers (7)	Body	Pre-request Script	Tests	Settings	Cookies
<input checked="" type="checkbox"/>	Accept					*/*	
<input checked="" type="checkbox"/>	Accept-Encoding					gzip, deflate, br	
<input checked="" type="checkbox"/>	Connection					keep-alive	
<input checked="" type="checkbox"/>	Authorization					Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWwiOiJ2ZWxpYWxpIiwiaWF0Ijox...	
	Key					Value	

Response:

Body Cookies Headers (11) Test Results Status: 200 OK Time: 148 ms Size: 359 B

Pretty Raw Preview Visualize Text

1 Tag deleted successfully.

DB:

123 tag_id	123 user_id	A-Z tag_name	

## Update Tag Name

```
@PostMapping("/update/{tagId}")
public ResponseEntity<?> updateTagName(@PathVariable Long tagId,
    @RequestParam String newTagName, @RequestHeader("Authorization")
    String token) {
    try {
        TagDTO updatedTag = tagService.updateTagName(tagId,
newTagName, token);
        return ResponseEntity.status(HttpStatus.OK).body(updatedTag);
    } catch (Exception e) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body("Tag
not found or unauthorized");
    }
}
```

## Java Spring Boot Code

Example URL: <http://localhost:8080/api/tags/update/2?newTagName=project>

Method Type: PUT

Token has to be supplied.

Before sending request:

PUT

http://localhost:8080/api/tags/update/2?newTagName=project

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests




Settings


Cookies

Query Params

Key	Value	Bulk Edit
<input checked="" type="checkbox"/> newTagName	project	
Key	Value	



**Response:**

Body Cookies Headers (11) Test Results  Status: 200 OK Time: 130 ms Size: 470 B [Save Response](#)  

Pretty Raw Preview Visualize JSON 

```
1 {
2   "tagId": 2,
3   "user": {
4     "name": "Ali Ahmet",
5     "surname": "Veli",
6     "email": "aliveli@gmail.com",
7     "username": "veliali"
8   },
9   "todos": [],
10  "tagName": "project"
11 }
```

**DB After Request:**

tags   Enter a SQL expression to filter results (use Ctrl+Space)				
Grid		123 tag_id	123 user_id	A-Z tag_name
1	3	1	aybu	
2	2	1	project	
Text				

## TO DO CONTROLLER LAYER METHODS TEST RESULT

### Create To Do

```
@PostMapping("/create")
public ResponseEntity<?> createToDo(@RequestBody
ToDoCreationRequestWrapper newToDo, @RequestHeader("Authorization")
String token) {
    try {
        ToDo createdToDo = toDoService.createToDo(
            newToDo.getTitle(),
            newToDo.getDescription(),
            newToDo.getStartTime(),
            newToDo.getEndTime(),
            token
        );
        return
ResponseEntity.status(HttpStatus.CREATED).body(createdToDo);
    } catch (RuntimeException e) {
        return
ResponseEntity.status(HttpStatus.BAD_REQUEST).body(e.getMessage());
// Spesifik hata mesaji
    } catch (Exception e) {
        return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR).body("Unexpected error: " + e.getMessage());
    }
}
```

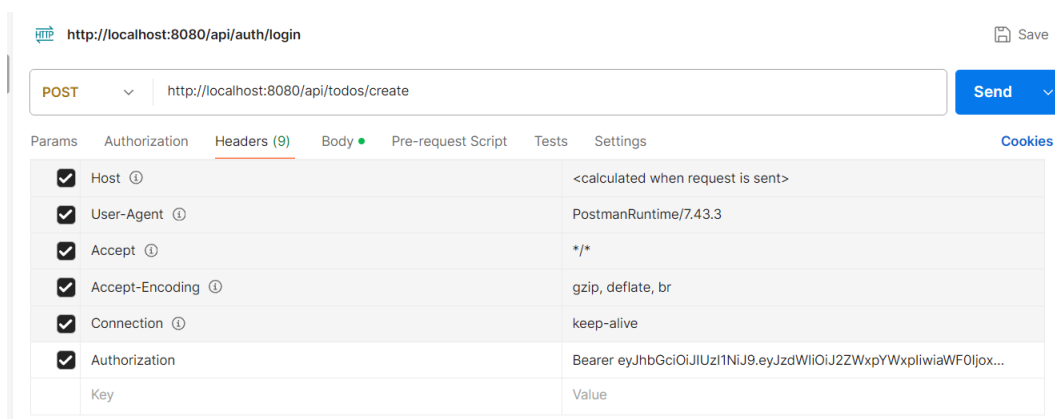
### Java Spring Boot Code

URL: <http://localhost:8080/api/todos/create>

Method Type: POST

Token must be supplied in the headers part.

Request Screen:





JSON:

```
{
  "title": "Toplantı Planla 2 ",
  "description": "Ekip ile 2 haftalık toplantı düzenle",
  "startTime": "2025-04-15",
  "endTime": "2025-04-15"
}
```

Response Json:

```
{
  "todold": 2,
  "title": "Toplantı Planla 2 ",
  "description": "Ekip ile 2 haftalık toplantı düzenle",
  "startDate": "2025-04-15",
  "expectedEndDate": "2025-04-15",
  "owner": {
    "name": "Ali Ahmet",
    "surname": "Veli",
    "email": "aliveli@gmail.com",
    "username": "veliali"
  }
}
```

DB screen after request is sent:

to_do	expected_end_time	starting_date	tag_priority	to_do_id	user_id	to_do_title	to_do_detailed_description
1	2025-04-15	2025-04-15	[NULL]	1	1	Toplantı Planla	Ekip ile haftalık toplantı düzenle
2	2025-04-15	2025-04-15	[NULL]	2	1	Toplantı Planla 2	Ekip ile 2 haftalık toplantı düzenle

**Delete To Do**

```
@DeleteMapping("/delete/{todoId}")
public ResponseEntity<?> deleteToDo(@PathVariable Long todoId,
@RequestHeader("Authorization") String token) {
    try {
        String result = toDoService.deleteToDo(todoId, token);
        return ResponseEntity.status(HttpStatus.OK).body(result);
    } catch (Exception e) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body("ToDo
not found or unauthorized");
    }
}
```

Java Spring Boot Code

URL: <http://localhost:8080/api/todos/delete/{toDoid}>  
 (EXAMPLE URL: <http://localhost:8080/api/todos/delete/2> )

Method Type: DELETE

Token must be supplied in the headers part.

The screenshot shows a REST client interface with the following details:

- URL: <http://localhost:8080/api/auth/login>
- Method: DELETE
- URL: <http://localhost:8080/api/todos/delete/2>
- Headers (7):
  - Accept: \*/\*
  - Accept-Encoding: gzip, deflate, br
  - Connection: keep-alive
  - Authorization: Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWwiOiJ2ZWxpYWxpIiwiaWF0Ijox...
  - Key: Value

HAVING NO JSON!

Response:

The screenshot shows the response of a DELETE request with the following details:

- Status: 200 OK
- Time: 314 ms
- Size: 360 B
- Response: 1 Todo deleted successfully.

DB screen after request is sent:

Grid	to_do	expected_end_time	starting_date	tag_priority	to_do_id	user_id	to_do_title	to_do_detailed_description
1		2025-04-15	2025-04-15	[NULL]	1	1	Toplantı Planla	Ekip ile haftalık toplantı düzenle

Update Priority

```
@PostMapping("/update-priority/{toDoId}")
public ResponseEntity<?> updatePriority(@PathVariable Long toDoId,
    @RequestParam Priority newPriority, @RequestHeader("Authorization")
    String token) {
    try {
        ToDoDTO updatedToDo = toDoService.updatePriority(toDoId,
            newPriority, token);
        return
    } catch (Exception e) {
```

```

        return ResponseEntity.status(HttpStatus.NOT_FOUND).body("ToDo
not found or unauthorized");
    }
}

```

## Java Spring Boot Code

Example URL: <http://localhost:8080/api/todos/update-priority/1?newPriority=P1>  
(Priority can be P1,P2,P3,P4 based on the Eisenhower Matrix)

Method Type: PUT

DB Screen Before Sending Request:

	expected_end_time	starting_date	tag_priority	to_do_id	user_id	to_do_title	to_do_detailed_description
1	2025-04-15	2025-04-15	[NULL]	1	1	Toplantı Planla	Ekip ile haftalık toplantı düzenle

Response:

Body Cookies Headers (11) Test Results Status: 200 OK Time: 174 ms Size: 578 B Save Response

Pretty Raw Preview Visualize JSON

```

1 {
2   "todoId": 1,
3   "title": "Toplantı Planla",
4   "description": "Ekip ile haftalık toplantı düzenle",
5   "startDate": "2025-04-15",
6   "expectedEndDate": "2025-04-15",
7   "owner": {
8     "name": "Ali Ahmet",
9     "surname": "Veli",
10    "email": "aliveli@gmail.com",
11    "username": "veliali"
12  }
13 }

```

DB Screen After Sending Request:

	expected_end_time	starting_date	tag_priority	to_do_id	user_id	to_do_title	to_do_detailed_description
1	2025-04-15	2025-04-15	0	1	1	Toplantı Planla	Ekip ile haftalık toplantı düzenle

## Update To Do Dates (Start Date and Expected End Date)

```

@PutMapping("/update-dates/{todoId}")
public ResponseEntity<?> updateDates(@PathVariable Long todoId,
                                     @RequestParam LocalDate
newStartDate,
                                     @RequestParam LocalDate
newExpectedEndDate,
                                     @RequestHeader("Authorization")
String token) {
    try {

```

```

        ToDoDTO updatedToDo = todoService.updateDates(todoId,
newStartDate, newExpectedEndDate, token);
        return
    ResponseEntity.status(HttpStatus.OK).body(updatedToDo);
    } catch (Exception e) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body("ToDo
not found or unauthorized");
    }
}

```

## Java Spring Boot Code

Example URL:

<http://localhost:8080/api/todos/update-dates/1?newStartDate=2025-04-17&newExpectedEndDate=2025-05-19>

Method Type: PUT

DB Screen Before Sending Request:

Grid	to_do	expected_end_time	starting_date	tag_priority	to_do_id	user_id	to_do_title	to_do_detailed_description
1		2025-04-15	2025-04-15	0	1	1	Toplantı Planla	Ekip ile haftalık toplantı düzenle

SCREEN SENDING PROGRESS:

PUT
http://localhost:8080/api/todos/update-dates/1?newStartDate=2025-04-17&newExpectedEndDate=2025-05-19
Send

Params
Authorization
Headers (8)
Body
Pre-request Script
Tests
Settings
Cookies

Query Params

Key	Value	Bulk Edit
<input checked="" type="checkbox"/> newStartDate	2025-04-17	
<input checked="" type="checkbox"/> newExpectedEndDate	2025-05-19	

RESPONSE:

Body
Cookies
Headers (11)
Test Results
Status: 200 OK Time: 70 ms Size: 578 B Save Response

Pretty
Raw
Preview
Visualize
JSON

```

1  {
2    "todoId": 1,
3    "title": "Toplantı Planla",
4    "description": "Ekip ile haftalık toplantı düzenle",
5    "startDate": "2025-04-17",
6    "expectedEndDate": "2025-05-19",
7    "owner": {
8      "name": "Ali Ahmet",
9      "surname": "Veli",
10     "email": "aliveli@gmail.com",
11     "username": "veliali"
12   }
13 }

```

DB Screen After Sending Request:

Grid	to_do	expected_end_time	starting_date	tag_priority	to_do_id	user_id	to_do_title	to_do_detailed_description
1		2025-05-19	2025-04-17	0	1	1	Toplantı Planla	Ekip ile haftalık toplantı düzenle

## Update To Do Title

```
@PostMapping("/update-title/{todoId}")
public ResponseEntity<?> updateTitle(@PathVariable Long todoId,
@RequestParam String newTitle, @RequestHeader("Authorization")
String token) {
    try {
        ToDoDTO updatedToDo = toDoService.updateTitle(todoId,
newTitle, token);
        return
ResponseEntity.status(HttpStatus.OK).body(updatedToDo);
    } catch (Exception e) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body("ToDo
not found or unauthorized");
    }
}
```

## Java Spring Boot Code

Example URL: <http://localhost:8080/api/todos/update-title/1?newTitle=Advisor ile gorusme>

Method Type: PUT

Request Screen:

PUT

http://localhost:8080/api/todos/update-title/1?newTitle=Advisor ile gorusme

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

	Key	Value	Bulk Edit
<input checked="" type="checkbox"/>	newTitle	Advisor ile gorusme	

Response:

Body

Cookies

Headers (11)

Test Results

Status: 200 OK

Time: 103 ms

Size: 581 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```

1  {
2    "todoId": 1,
3    "title": "Advisor ile gorusme",
4    "description": "Ekip ile haftalık toplantı düzenle",
5    "startDate": "2025-04-17",
6    "expectedEndDate": "2025-05-19",
7    "owner": {
8      "name": "Ali Ahmet",
9      "surname": "Velili",
10     "email": "aliveli@gmail.com",
11     "username": "veliali"
12   }
13 }
```

**DB Screen After Request:**

	expected_end_time	starting_date	tag_priority	to_do_id	user_id	to_do_title	to_do_detailed_description
1	2025-05-19	2025-04-17	0	1	1	Advisor ile gorusme	Ekip ile haftalık toplantı düzenle

**Update Description**

```

@PutMapping("/update-description/{todoId}")
public ResponseEntity<?> updateDescription(@PathVariable Long
todoId, @RequestParam String newDescription,
@RequestHeader("Authorization") String token) {
    try {
        ToDoDTO updatedToDo = todoService.updateDescription(todoId,
newDescription, token);
        return
ResponseEntity.status(HttpStatus.OK).body(updatedToDo);
    } catch (Exception e) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body("ToDo
not found or unauthorized");
    }
}

```

**Java Spring Boot Code****Example URL:**

<http://localhost:8080/api/todos/update-description/1?newDescription=Advisor ile gorusmede gelecege yönelik tavsiyeler alınacak>

**Method Type: PUT****Request Screen:**

PUT

http://localhost:8080/api/todos/update-description/1?newDescription=Advisor ile gorusmede gelecege yönelik tavsiyeler alınacak

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

Key	Value	Bulk Edit
<input checked="" type="checkbox"/> newDescription	Advisor ile gorusmede gelecege yönelik tavsiyeler alınacak	
Key	Value	

**Response:**

Body

Cookies

Headers (11)

Test Results

Status: 200 OK Time: 78 ms Size: 605 B Save Response

Pretty

Raw

Preview

Visualize

JSON

```

1  {
2    "todoId": 1,
3    "title": "Advisor ile gorusme",
4    "description": "Advisor ile gorusmede gelecege yönelik tavsiyeler alınacak",
5    "startDate": "2025-04-17",
6    "expectedEndDate": "2025-05-19",
7    "owner": {
8      "name": "Ali Ahmet",
9      "surname": "Veli",
10     "email": "aliveli@gmail.com",
11     "username": "veliali"
12   }
13 }

```

## DB Screen After Sending Request:

to_do	expected_end_time	starting_date	tag_priority	to_do_id	user_id	to_do_title	to_do_detailed_description
1	2025-05-19	2025-04-17	0	1	1	Advisor ile gorusme	Advisor ile gorusmede gelecege yönelik tavsiyeler alınacak

## Assign Tag to To Do

```
@PostMapping("/add-tag/{todoId}")
public ResponseEntity<?> addTag(@PathVariable Long todoId,
@RequestParam Long tagId, @RequestHeader("Authorization") String
token) {
    try {
        ToDoDTO updatedToDo = todoService.addTag(todoId, tagId,
token);
        return
ResponseEntity.status(HttpStatus.OK).body(updatedToDo);
    } catch (Exception e) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body("ToDo
or Tag not found, or unauthorized");
    }
}
```

## Java Spring Boot Code

Example URL: <http://localhost:8080/api/todos/add-tag/1?tagId=3>

Method Type: PUT

Sending process:

PUT

http://localhost:8080/api/todos/add-tag/1?tagId=3

Send

Params

Authorization

Headers (8)

Body

Pre-request Script

Tests

Settings

Cookies

Query Params

Key	Value	Bulk Edit
<input checked="" type="checkbox"/> tagId	3	
Key	Value	

Response:

Body

Cookies

Headers (11)

Test Results

Status: 200 OK Time: 118 ms Size: 605 B Save Response

Pretty

Raw

Preview

Visualize

JSON

```

1  {
2    "todoId": 1,
3    "title": "Advisor ile gorusme",
4    "description": "Advisor ile gorusmede gelecege yönelik tavsiyeler alınacak",
5    "startDate": "2025-04-17",
6    "expectedEndDate": "2025-05-19",
7    "owner": {
8      "name": "Ali Ahmet",
9      "surname": "Veli",
10     "email": "aliveli@gmail.com",
11     "username": "veliali"
12   }
13 }
```

**DB Screen After Request:**

to_do_tags <small>Enter a SQL expression to filter res</small>			
Grid	123 tag_id	123 to_do_id	
1	3	1	
Text			

**Delete Tag From To Do**

```

@DeleteMapping("/delete-tag/{todoId}")
public ResponseEntity<?> deleteTag(@PathVariable Long todoId,
@RequestParam Long tagId, @RequestHeader("Authorization") String
token) {
    try {
        ToDoDTO updatedToDo = toDoService.deleteTag(todoId, tagId,
token);
        return
ResponseEntity.status(HttpStatus.OK).body(updatedToDo);
    } catch (Exception e) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND).body("ToDo
or Tag not found, or unauthorized");
    }
}

```

**Java Spring Boot Code**

Example URL: <http://localhost:8080/api/todos/delete-tag/1?tagId=3>

Method Type: DELETE

**Sending process:**

DELETE	http://localhost:8080/api/todos/delete-tag/1?tagId=3	Send									
<div>Params • Authorization Headers (7) Body Pre-request Script Tests Settings Cookies</div> <div>Query Params</div> <table border="1"> <thead> <tr> <th>Key</th> <th>Value</th> <th>Bulk Edit</th> </tr> </thead> <tbody> <tr> <td><input checked="" type="checkbox"/> tagId</td> <td>3</td> <td></td> </tr> <tr> <td>Key</td> <td>Value</td> <td></td> </tr> </tbody> </table>			Key	Value	Bulk Edit	<input checked="" type="checkbox"/> tagId	3		Key	Value	
Key	Value	Bulk Edit									
<input checked="" type="checkbox"/> tagId	3										
Key	Value										



**Response:**

Body Cookies Headers (11) Test Results Status: 200 OK Time: 42 ms Size: 605 B Save Response

Pretty Raw Preview Visualize JSON

```

1  {
2    "todoId": 1,
3    "title": "Advisor ile gorusme",
4    "description": "Advisor ile gorusmede gelecege yönelik tavsiyeler alınacak",
5    "startDate": "2025-04-17",
6    "expectedEndDate": "2025-05-19",
7    "owner": {
8      "name": "Ali Ahmet",
9      "surname": "Veli",
10     "email": "aliveli@gmail.com",
11     "username": "veliali"
12   }
13 }

```

**DB Screen After Request:**

to_do_tags	tag_id	to_do_id
	123	123

Tag belongs to a particular to-do list item is deleted.

**GET TO DO FROM USER**

```

@GetMapping("/getTodos")
public ResponseEntity<?>
getUserTodos (@RequestHeader("Authorization") String
authorizationHeader) {
    try {
        // Extract token from "Bearer <token>"
        if (authorizationHeader == null ||
!authorizationHeader.startsWith("Bearer ")) {
            return ResponseEntity.status(HttpStatus.UNAUTHORIZED)
                .body("Authorization header missing or invalid");
        }
        String token = authorizationHeader.replace("Bearer ", ""); //
Remove "Bearer " prefix

        // Call UserService to get todos
        List<ToDoDTO> todos = toDoService.getUserTodos(token);
        return ResponseEntity.ok(todos);
    }
}

```

```

    } catch (UsernameNotFoundException e) {
        return ResponseEntity.status(HttpStatus.NOT_FOUND)
            .body("User not found: " + e.getMessage());
    } catch (JwtException e) {
        return ResponseEntity.status(HttpStatus.UNAUTHORIZED)
            .body("Invalid or expired token: " + e.getMessage());
    } catch (Exception e) {
        return
ResponseEntity.status(HttpStatus.INTERNAL_SERVER_ERROR)
            .body("An error occurred: " + e.getMessage());
    }
}

```

## Java Spring Boot Code

URL: <http://localhost:8080/api/todos/getTodos>

Method Type: GET

Sending process:

GET	Send
http://localhost:8080/api/todos/getTodos	
Params	Authorization
Headers (7)	Body
Pre-request Script	Tests
Settings	Cookies
Accept	*/*
Accept-Encoding	gzip, deflate, br
Connection	keep-alive
Authorization	Bearer eyJhbGciOiJIUzI1NiJ9.eyJzdWIiOiJ2ZWxpYWxpIiwiaWF0Ijox...
Key	Value

Response:

Body Cookies Headers (11) Test Results

Status: 200 OK Time: 114 ms Size: 607 B Save Response

Pretty Raw Preview Visualize JSON

```

1  [
2  {
3      "todoId": 1,
4      "title": "Advisor ile gorusme",
5      "description": "Advisor ile gorusmede gelecege yönelik tavsiyeler alınacak",
6      "startDate": "2025-04-17",
7      "expectedEndDate": "2025-05-19",
8      "owner": {
9          "name": "Ali Ahmet",
10         "surname": "Veli",
11         "email": "aliveli@gmail.com",
12         "username": "veliali"
13     }
14 }
15 ]

```