

Testovi za Task 5

Task	Test Scenario #	Scenario ID	Test Scenario Description	Test Cases
5	1	S1.1	Testovi za rutu GetAllUsers	1. Provjera ispravnosti formata odgovora 2. Provjera postojanja content tipa 3. Provjera da li je content-type u JSON formatu 4. Provjera da li response ima body 5. Provjera ispravnog statusnog koda
	2	S2.1	Testovi za rutu UpdateUserInfo – validni podaci	1. Provjera ispravnog tipa responsea 2. Provjera postojanja content tipa 3. Provjera da li je content-type u JSON formatu 4. Provjera da li response ima body 5. Provjera ispravnog statusnog koda
	3	S2.2	Testovi za rutu UpdateUserInfo – prazno obavezno polje za password	1. Provjera ispravnog statusnog koda 2. Provjera ispravnog tipa odgovora 3. Provjera postojanja content-type 4. Provjera da li je content-type u JSON formatu 5. Provjera da li response ima body

Testovi za rutu **GetAllUsers** :

1) Testiranje rute

Body :

GET ▼ https://localhost:3000/User/GetAllUsers

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

☐ none ☐ form-data ☐ x-www-form-urlencoded ☒ raw ☐ binary ☐ GraphQL **JSON** ▼

Response Body :

Body Cookies Headers (11) Test Results

Pretty Raw Preview Visualize JSON ▼

```
1  [
2    {
3      "id": 12,
4      "email": "user@example.com",
5      "name": "string",
6      "surname": "string",
7      "password": "novipass",
8      "deletedStatus": false
9    },
10   {
11     "id": 13,
12     "email": "hamzabegic@hamzabegic.com",
13     "name": "hamza",
14     "surname": "begic",
15     "password": "hamzabegic",
16     "deletedStatus": false
17   }
18 ]
```


Tests :

```
GET https://localhost:3000/User/GetAllUsers

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

1  const response = pm.response.json();
2  pm.test("The response has all properties with the expected type", () => {
3      for(var i = 0; i < response.length; i++){
4          pm.expect(response[i].id).to.be.an("number");
5          pm.expect(response[i].email).to.be.a("string");
6          pm.expect(response[i].name).to.be.a("string");
7          pm.expect(response[i].surname).to.be.an("string");
8          pm.expect(response[i].password).to.be.an("string");
9          pm.expect(response[i].deletedStatus).to.be.an("boolean");
10     }
11 });
12 pm.test("Content-Type is present", function () {
13     pm.response.to.have.header("Content-Type");
14 });
15
16 pm.test("Content-Type is JSON", function () {
17     pm.response.to.be.json;
18 });
19 pm.test("Response must have a body", function () {
20     pm.response.to.be.withBody;
21     pm.response.to.be.json;
22 });
23 pm.test("Status code is 200 and OK", () => {
24     pm.response.to.have.status(200);
25     pm.response.to.be.ok;
26 });
27
```

Rezultati testiranja :

Body Cookies Headers (11) Test Results (5/5)  Status: 200 OK

All	Passed	Skipped	Failed
PASS	The response has all properties with the expected type		
PASS	Content-Type is present		
PASS	Content-Type is JSON		
PASS	Response must have a body		
PASS	Status code is 200 and OK		

Testovi za rutu **UpdateUserInfo** :

2) Svi podaci validni

Body:

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

- Method:** PUT
- URL:** https://localhost:3000/User/UpdateUserInfo
- Body Type:** JSON
- Body Content:**

```
1 {
2   "id": 13,
3   "email": "hamzabegic@hamzabegic.com",
4   "name": "hamzaUPDATE",
5   "surname": "begic",
6   "password": "hamzabegic",
7   "deletedStatus": false
8 }
```

Response body:

The screenshot shows the response body of the PUT request in a REST client interface:

- Body Type:** JSON
- Body Content:**

```
1 {
2   "id": 13,
3   "email": "hamzabegic@hamzabegic.com",
4   "name": "hamzaUPDATE",
5   "surname": "begic",
6   "password": "hamzabegic",
7   "deletedStatus": false
8 }
```

Tests:

PUT


https://localhost:3000/User/UpdateUserInfo

ParamsAuthorizationHeaders (9)Body ●Pre-request ScriptTests ●Settings

```
1  const response = pm.response.json();
2  pm.test("The response has all properties with the expected type", () => {
3      for(var i = 0; i < response.length; i++){
4          pm.expect(response[i].id).to.be.an("number");
5          pm.expect(response[i].email).to.be.a("string");
6          pm.expect(response[i].name).to.be.a("string");
7          pm.expect(response[i].surname).to.be.an("string");
8          pm.expect(response[i].password).to.be.an("object");
9          pm.expect(response[i].deletedStatus).to.be.an("boolean");
10     }
11 });
12 pm.test("Content-Type is present", function () {
13     pm.response.to.have.header("Content-Type");
14 });
15
16 pm.test("Content-Type is JSON", function () {
17     pm.response.to.be.json;
18 });
19 pm.test("Response must have a body", function () {
20     pm.response.to.be.withBody;
21     pm.response.to.be.json;
22 });
23 pm.test("Status code is 200 and OK", () => {
24     pm.response.to.have.status(200);
25     pm.response.to.be.ok;
26 });
```

Test results:

BodyCookiesHeaders (11)Test Results (5/5)

 Status: 200 OK

AllPassedSkippedFailed

PASS

The response has all properties with the expected type

PASS

Content-Type is present

PASS

Content-Type is JSON

PASS


Response must have a body



PASS








Status code is 200 and OK

3) Prazno obavezno polje za password

Body:


PUT  https://localhost:3000/User/UpdateUserInfo



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

 none  form-data  x-www-form-urlencoded  **raw**  binary  GraphQL **JSON** 

```
1  {
2    "id": 12,
3    "email": "user@example.com",
4    "name": "string",
5    "surname": "string",
6    "password": "",
7    "deletedStatus": true
8  }
```

Tests:

PUT  https://localhost:3000/User/UpdateUserInfo

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

```
1  const response = pm.response.json();
2  pm.test("Status code is 400", () => {
3    pm.response.to.have.status(400);
4  });
5  pm.test("The response has all properties with the expected type", () => {
6    for(var i = 0; i < response.length; i++){
7      pm.expect(response[i].type).to.be.an("string");
8      pm.expect(response[i].title).to.be.a("string");
9      pm.expect(response[i].status).to.be.a("number");
10     pm.expect(response[i].traceId).to.be.an("string");
11     pm.expect(response[i].errors).to.be.an("object");
12     pm.expect(response[i].errors.password).to.be.an("string");
13   }
14 });
15
16 pm.test("Content-Type is present", function () {
17   pm.response.to.have.header("Content-Type");
18 });
```

```

16 pm.test("Content-Type is present", function () {
17 |     pm.response.to.have.header("Content-Type");
18 | });
19
20 pm.test("Content-Type is JSON", function () {
21 |     pm.response.to.be.json;
22 | });
23 pm.test("Response must have a body", function () {
24 |     pm.response.to.be.withBody;
25 |     pm.response.to.be.json;
26 | });

```

Response body:

Body Cookies Headers (11) Test Results (5/5)

Pretty Raw Preview Visualize JSON

```

1  {
2      "type": "https://tools.ietf.org/html/rfc7231#section-6.5.1",
3      "title": "One or more validation errors occurred.",
4      "status": 400,
5      "traceId": "00-e5ad0c2c2c3b95a41c4b3e96e8e271d9-7c1f3b4ba006d37e-00",
6      "errors": {
7          "Password": [
8              "The Password field is required."
9          ]
10     }
11 }

```

Test results:

Body Cookies Headers (11) Test Results (5/5) Status: 400 Bad Request

All Passed Skipped Failed

- PASS** Status code is 400
- PASS** The response has all properties with the expected type
- PASS** Content-Type is present
- PASS** Content-Type is JSON
- PASS** Response must have a body