

COMP229 ASSIGNMENT 2: NODEJS ENDPOINTS
R MARCO AURELIO DOMINGUEZ SANTANA

Contents

1	Contacts endpoints.....	3
1.1	Get contact by id.....	3
1.2	Get all contacts.....	4
1.3	Add new contact.....	4
1.4	Update contact by id.....	5
1.5	Remove contact by id	7
1.6	Remove all contacts	8
2	Projects endpoints.....	9
2.1	Get all projects.....	9
2.2	Get projects by Id	10
2.3	Add new projects.....	11
2.4	Update project by id.....	12
2.5	Remove project by id	13
2.6	Remove all projects	14
3	Services endpoints	15
3.1	Get all services.....	15
3.2	Add new service	16
3.3	Get service by id.....	17
3.4	Update service by id	18
3.5	Remove service by id	19
3.6	Remove all services.....	20
4	Users endpoints	21
4.1	Get all users	22
4.2	Get User By Id	22
4.3	Add new user	23
4.4	Update user by id	24
4.5	Remove user by id	26
4.6	Remove all users.....	27

1 Contacts endpoints

1.1 Get contact by id

The screenshot shows two separate instances of the Postman application interface, each displaying a GET request to retrieve a contact by its ID.

Top Instance:

- URL: `http://localhost:3000/api/contacts/68f41e519e7c3db5706e91f9`
- Method: GET
- Body: Raw JSON response (shown below)

```
1 {  
2   "_id": "68f41e519e7c3db5706e91f9",  
3   "firstName": "first name contact 1",  
4   "lastName": "last name contact 1",  
5   "email": "email-contact-1@example.com",  
6   "__v": 0  
7 }
```

Bottom Instance:

- URL: `http://localhost:3000/api/contacts/68f41e8e9e7c3db5706e91fb`
- Method: GET
- Body: Raw JSON response (shown below)

```
1 {  
2   "_id": "68f41e8e9e7c3db5706e91fb",  
3   "firstName": "first second name contact 2",  
4   "lastName": "last second name contact 2",  
5   "email": "email-contact-2@example.com",  
6   "__v": 0  
7 }
```

Both instances show a status of 200 OK with response times around 35-40 ms and a body size of 416-430 B. The Postman toolbar at the bottom includes icons for Runner, Start Proxy, Cookies, Vault, and Trash, along with system status indicators like battery level and network connection.

1.2 Get all contacts

The screenshot shows the Postman interface with a GET request to `http://localhost:3000/api/contacts/`. The response status is 200 OK, and the response body is a JSON array containing two contact objects:

```
[{"id": "68141e8e9e7c3db5786e91fb", "firstName": "first name contact 1", "lastName": "last name contact 1", "email": "email-contact-1@example.com", "__v": 0}, {"id": "68141e8e9e7c3db5786e91fb", "firstName": "first second name contact 2", "lastName": "last second name contact 2", "email": "email-contact-2@example.com", "__v": 0}]
```

1.3 Add new contact

The screenshot shows the Postman interface with a POST request to `http://localhost:3000/api/contacts/`. The response status is 200 OK, and the response body is a JSON object indicating success:

```
{"success": true, "message": "Contact created successfully."}
```

The screenshot shows the Postman application interface. A POST request is made to `http://localhost:3000/api/contacts/`. The request body is a JSON object:

```
1 {  
2   "firstName": "first second name contact 2",  
3   "lastName": "last second name contact 2",  
4   "email": "email-contact-2@example.com"  
5 }
```

The response status is 200 OK, with a response time of 40 ms and a size of 325 B. The response body is:

```
1 {  
2   "success": true,  
3   "message": "Contact created successfully."  
4 }
```

1.4 Update contact by id

The screenshot shows the Postman application interface. A PUT request is made to `http://localhost:3000/api/contacts/68f41e8e9e7c3db5706e91fb`. The request body is a JSON object:

```
1 {  
2   "firstName": "first second name contact 2 UPDATED",  
3   "lastName": "last second name contact 2 UPDATED",  
4   "email": "email-contact-2@example.com UPDATED"  
5 }
```

The response status is 200 OK, with a response time of 50 ms and a size of 325 B. The response body is:

```
1 {  
2   "success": true,  
3   "message": "Contact updated successfully."  
4 }
```

The screenshot shows the Postman application interface. In the top navigation bar, 'Home' and 'Workspaces' are selected. A search bar at the top right contains the text 'Search Postman'. On the left sidebar, there are sections for 'Collections', 'Environments', 'Flows', and 'History'. The main workspace shows a request to 'http://localhost:3000/api/contacts/68f41e519e7c3db5706e91fb'. The method is set to 'PUT' and the URL is 'http://localhost:3000/api/contacts/68f41e519e7c3db5706e91fb'. The 'Body' tab is active, showing a raw JSON payload:

```
1  {
2   "firstName": "first name contact 1 UPDATED",
3   "lastName": "last name contact 1 UPDATED",
4   "email": "email-contact-1@example.com UPDATED"
5 }
```

Below the body, the 'Test Results' section shows a successful response with status code 200 OK, duration 41 ms, and 325 B transferred. The response body is:

```
1 {
2   "success": true,
3   "message": "Contact updated successfully."
4 }
```

The bottom of the screen shows the Windows taskbar with various pinned icons.

RESULT (UPDATED CONTACTS)

The screenshot shows the Postman application interface again. The top navigation bar has 'Workspaces' selected. The main workspace shows a request to 'http://localhost:3000/api/contacts'. The method is set to 'GET' and the URL is 'http://localhost:3000/api/contacts'. The 'Body' tab is active, showing a raw JSON payload:

```
1  {
2   "firstName": "first name contact 1 UPDATED",
3   "lastName": "last name contact 1 UPDATED",
4   "email": "email-contact-1@example.com UPDATED"
5 }
```

Below the body, the 'Test Results' section shows a successful response with status code 200 OK, duration 37 ms, and 630 B transferred. The response body is:

```
1 [
2   {
3     "_id": "68f41e519e7c3db5706e91fb",
4     "firstName": "first name contact 1 UPDATED",
5     "lastName": "last name contact 1 UPDATED",
6     "email": "email-contact-1@example.com UPDATED",
7     "__v": 0
8   },
9   {
10    "_id": "68f41e8e9e7c3db5706e91fb",
11    "firstName": "first second name contact 2 UPDATED",
12    "lastName": "last second name contact 2 UPDATED",
13    "email": "email-contact-2@example.com UPDATED",
14    "__v": 0
15  }
16 ]
```

The bottom of the screen shows the Windows taskbar with various pinned icons.

1.5 Remove contact by id

The screenshot shows the Postman application interface. A DELETE request is being made to `http://localhost:3000/api/contacts/68f41e519e7c3db5706e91f9`. The request body contains the following JSON:

```
1 {
2   "firstName": "first name contact 1 UPDATED",
3   "lastName": "last name contact 1 UPDATED",
4   "email": "email-contact-1@example.com UPDATED"
5 }
```

The response status is 200 OK, with a response time of 38 ms and a size of 325 B. The response body is:

```
{ "success": true, "message": "Contact deleted successfully." }
```

The bottom status bar shows the system temperature at 16°C and the date/time as 10/18/2025 7:15 PM.

RESULT (FIRST CONTACT REMOVED)

The screenshot shows the Postman application interface. A GET request is being made to `http://localhost:3000/api/contacts`. The request body contains the same JSON as the previous screenshot:

```
1 {
2   "firstName": "first name contact 1 UPDATED",
3   "lastName": "last name contact 1 UPDATED",
4   "email": "email-contact-1@example.com UPDATED"
5 }
```

The response status is 200 OK, with a response time of 36 ms and a size of 456 B. The response body is:

```
[ {
2   "_id": "68f41e8e9e7c3db5786e91fb",
3   "firstName": "first second name contact 2 UPDATED",
4   "lastName": "last second name contact 2 UPDATED",
5   "email": "email-contact-2@example.com UPDATED",
6   "__v": 6
7 } ]
```

The bottom status bar shows the system temperature at 16°C and the date/time as 10/18/2025 7:16 PM.

1.6 Remove all contacts

The screenshot shows the Postman application interface. In the top navigation bar, 'Home' and 'Workspaces' are selected. The main workspace shows a collection named 'Collections' containing a single item 'http://localhost:3000/api/contacts'. A 'DELETE' request is selected with the URL 'http://localhost:3000/api/contacts'. The 'Body' tab is active, showing the raw JSON response:

```
1 Ctrl+Alt+P to Ask AI
```

The 'Test Results' tab shows a successful response with status code 200 OK, duration 40 ms, and 335 B transferred. The JSON response body is:

```
1 {  
2   "success": true,  
3   "message": "All contacts were deleted successfully."  
4 }
```

The bottom status bar indicates 'Heavy rain Tomorrow' and the system date and time as 7:37 PM on 10/18/2025.

RESULTS (ALL CONTACTS DELETED)

The screenshot shows the Postman application interface. In the top navigation bar, 'Home' and 'Workspaces' are selected. The main workspace shows a collection named 'Collections' containing a single item 'http://localhost:3000/api/contacts'. A 'GET' request is selected with the URL 'http://localhost:3000/api/contacts'. The 'Body' tab is active, showing the raw JSON response body:

```
1 Ctrl+Alt+P to Ask AI
```

The 'Test Results' tab shows a successful response with status code 200 OK, duration 38 ms, and 267 B transferred. The JSON response body is:

```
1 []
```

The bottom status bar indicates 'Heavy rain Tomorrow' and the system date and time as 7:37 PM on 10/18/2025.

2 Projects endpoints

2.1 Get all projects

The screenshot shows the Postman application interface. At the top, there's a navigation bar with 'Home', 'Workspaces', 'API Network', and various search and settings icons. Below the bar, the main workspace shows a collection named 'http://localhost:3000/api/projects'. A 'GET' request is selected with the URL 'http://localhost:3000/api/projects'. The 'Body' tab is active, showing a raw JSON payload:

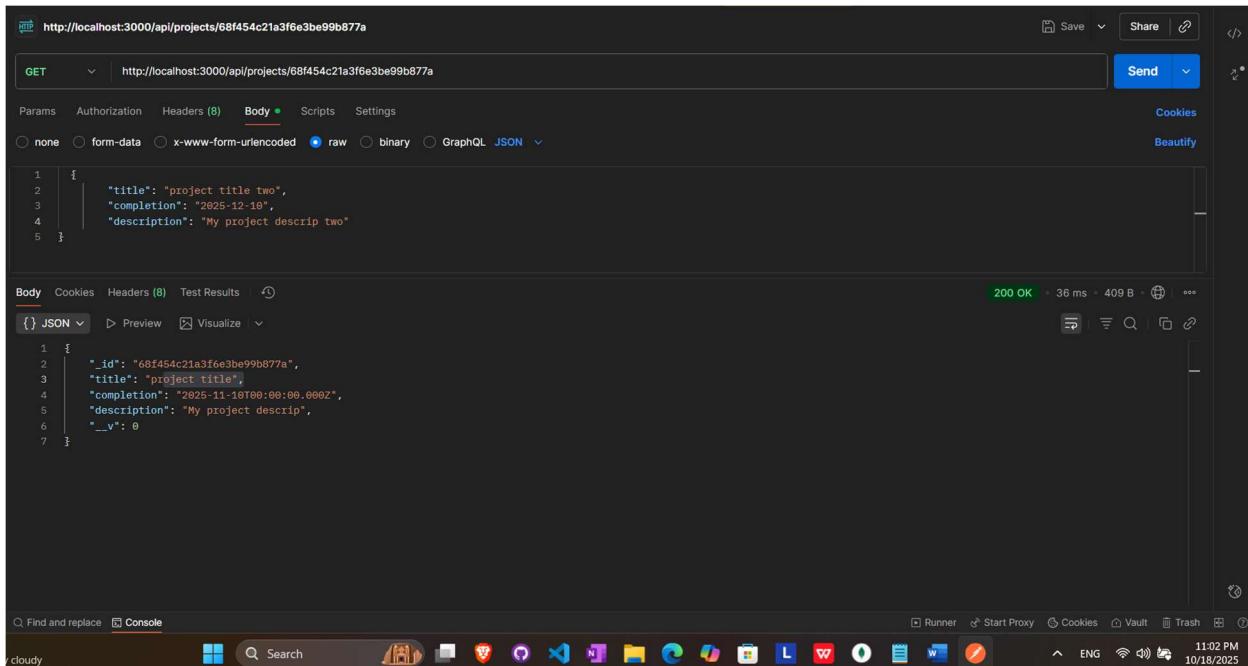
```
1  [
2    {
3      "title": "Project title TWO",
4      "completion": "2025-12-10T00:00:00Z",
5      "description": "My project descrip TWO"
6    }
7  ]
```

Below the request, the 'Test Results' section displays the response from a '200 OK' request. The response body contains two project documents:

```
1  [
2    {
3      "_id": "68f4252eaa0cb5cc73b3die2",
4      "title": "Project title",
5      "completion": "2025-11-18T08:00:00Z",
6      "description": "My project descrip",
7      "__v": 0
8    },
9    {
10      "_id": "68f4253faaf0cb5cc73b3die4",
11      "title": "Project title TWO",
12      "completion": "2025-12-18T08:00:00Z",
13      "description": "My project descrip TWO",
14      "__v": 0
15    }
]
```

At the bottom of the interface, there's a toolbar with various icons and a status bar showing the weather (16°C, Partly cloudy), system icons, and the date/time (10/18/2025, 7:39 PM).

2.2 Get projects by Id



HTTP <http://localhost:3000/api/projects/68f454c21a3f6e3be99b877a>

GET <http://localhost:3000/api/projects/68f454c21a3f6e3be99b877a>

Send

Params Authorization Headers (8) Body Scripts Settings Cookies Beautify

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

```
1 {  
2   "title": "project title two",  
3   "completion": "2025-12-10T00:00:00.000Z",  
4   "description": "My project descrip two"  
5 }
```

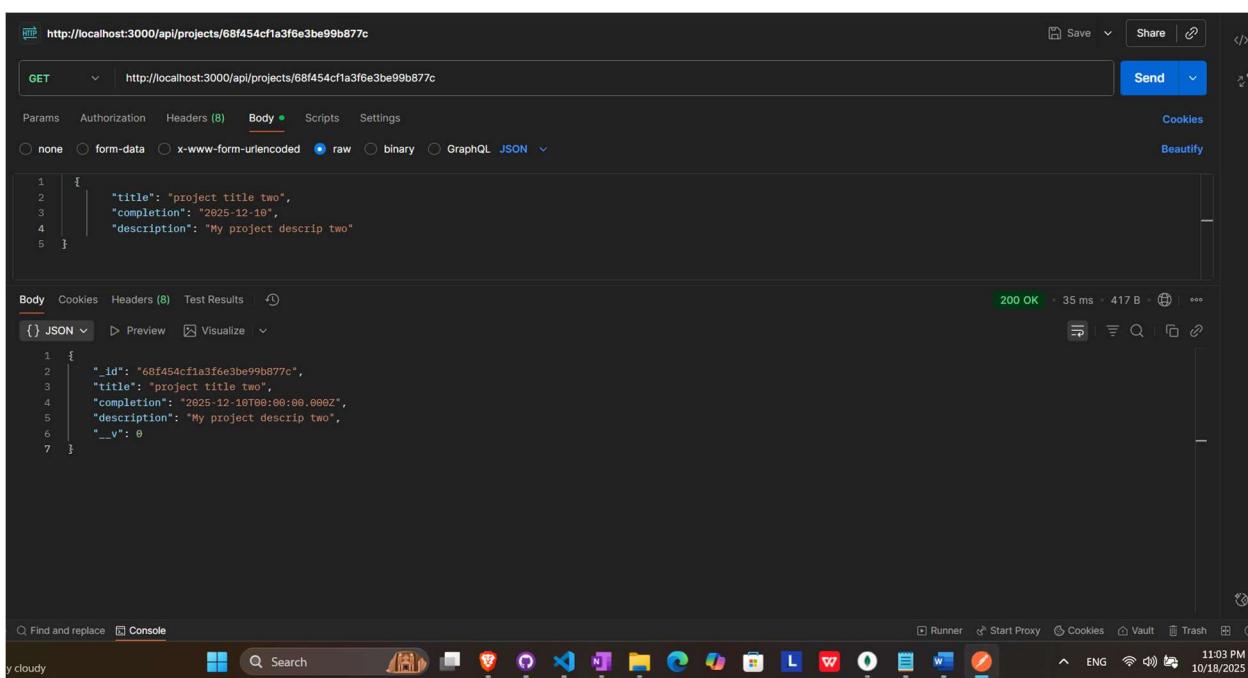
Body Cookies Headers (8) Test Results 41ms

{ } JSON Preview Visualize

200 OK 36 ms 409 B

Find and replace Console

Runner Start Proxy Cookies Vault Trash ENG 11:02 PM 10/18/2025



HTTP <http://localhost:3000/api/projects/68f454cf1a3f6e3be99b877c>

GET <http://localhost:3000/api/projects/68f454cf1a3f6e3be99b877c>

Send

Params Authorization Headers (8) Body Scripts Settings Cookies Beautify

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

```
1 {  
2   "title": "project title two",  
3   "completion": "2025-12-10T00:00:00.000Z",  
4   "description": "My project descrip two"  
5 }
```

Body Cookies Headers (8) Test Results 41ms

{ } JSON Preview Visualize

200 OK 35 ms 417 B

Find and replace Console

Runner Start Proxy Cookies Vault Trash ENG 11:03 PM 10/18/2025

2.3 Add new projects

The screenshot shows the Postman application interface. A POST request is made to `http://localhost:3000/api/projects`. The request body contains the following JSON:

```
1 {  
2   "title": "Project title",  
3   "completion": "2025-11-18",  
4   "description": "My project descrip"  
5 }
```

The response status is 200 OK, with a response time of 57 ms and a size of 325 B. The response body is:

```
1 {  
2   "success": true,  
3   "message": "Project created successfully."  
4 }
```

The screenshot shows the Postman application interface. A POST request is made to `http://localhost:3000/api/projects`. The request body contains the following JSON:

```
1 {  
2   "title": "Project title TWO",  
3   "completion": "2025-12-18",  
4   "description": "My project descrip TWO"  
5 }
```

The response status is 200 OK, with a response time of 39 ms and a size of 325 B. The response body is:

```
1 {  
2   "success": true,  
3   "message": "Project created successfully."  
4 }
```

2.4 Update project by id

The screenshot shows two separate instances of the Postman application interface, each displaying a successful API call to update a project.

Top Instance:

- URL: `http://localhost:3000/api/projects/68f4252eaaf0cb5c73b3d1e2`
- Method: `PUT`
- Body content (raw JSON):

```
1 {  
2   "title": "Project title UPDATED",  
3   "completion": "2025-12-10",  
4   "description": "My project descrip UPDATED"  
5 }
```
- Response status: `200 OK`
- Response body (JSON):

```
1 {  
2   "success": true,  
3   "message": "Project updated successfully."  
4 }
```

Bottom Instance:

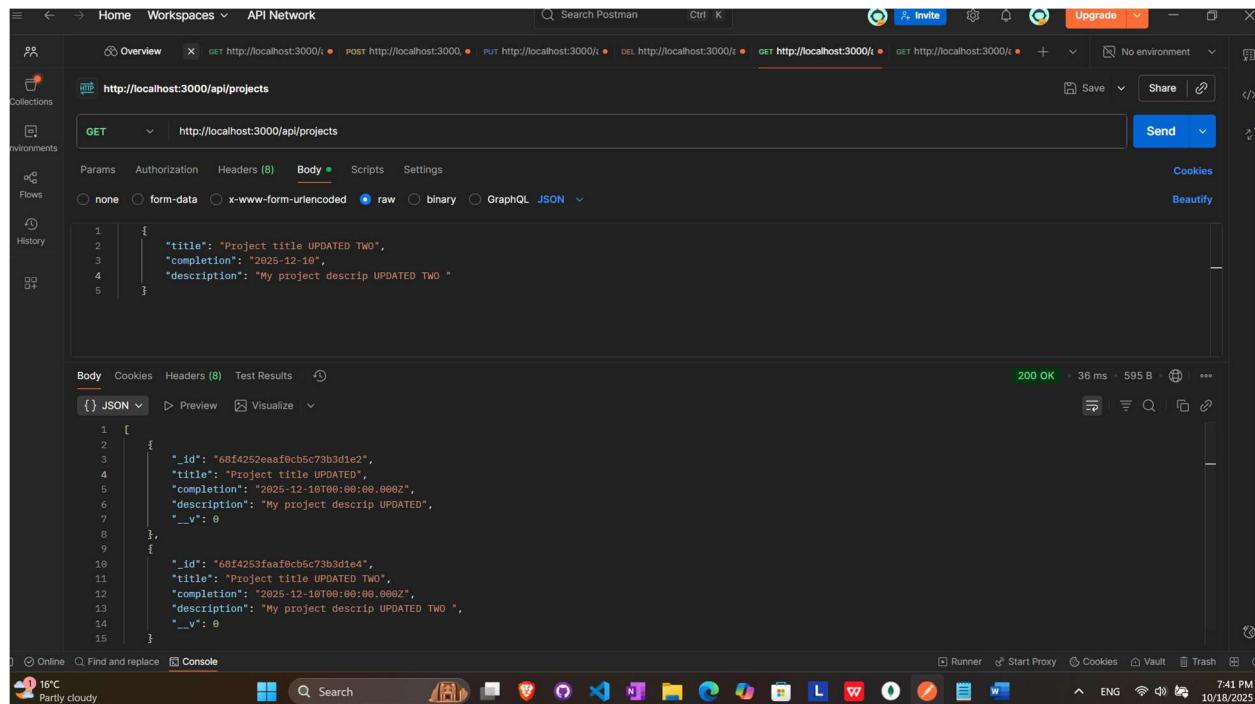
- URL: `http://localhost:3000/api/projects/68f4253faaf0cb5c73b3d1e4`
- Method: `PUT`
- Body content (raw JSON):

```
1 {  
2   "title": "Project title UPDATED TWO",  
3   "completion": "2025-12-10",  
4   "description": "My project descrip UPDATED TWO"  
5 }
```
- Response status: `200 OK`
- Response body (JSON):

```
1 {  
2   "success": true,  
3   "message": "Project updated successfully."  
4 }
```

Both instances show the Postman interface with various tools and environment settings visible at the bottom.

RESULT (ALL PROJECTS UPDATED)



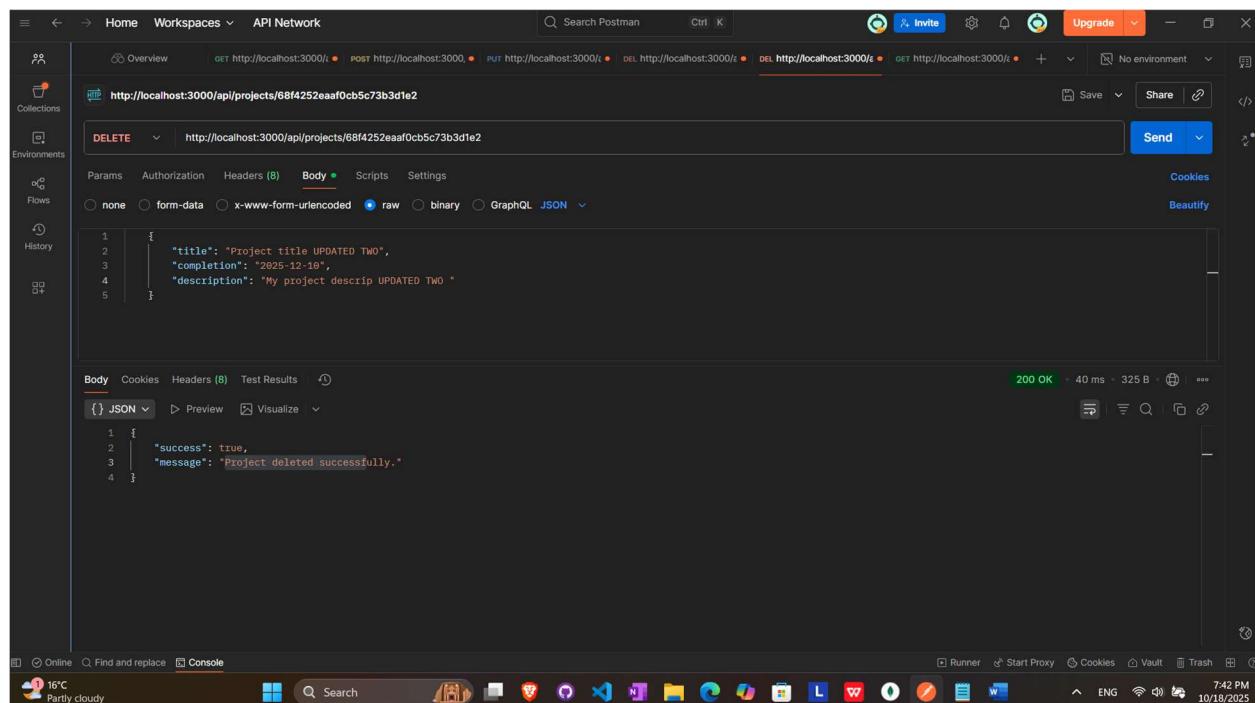
HTTP Request URL: <http://localhost:3000/api/projects>

Method: GET

Response Body:

```
1  [
2    {
3      "_id": "68f4252eaaf0cb5c73b3d1e2",
4      "title": "Project title UPDATED TWO",
5      "completion": "2025-12-10T00:00:00Z",
6      "description": "My project descrip UPDATED TWO"
7    }
8  ],
9  [
10   {
11     "_id": "68f4253faaf0cb5c73b3d1e4",
12     "title": "Project title UPDATED TWO",
13     "completion": "2025-12-10T00:00:00Z",
14     "description": "My project descrip UPDATED TWO"
15   }
]
```

2.5 Remove project by id



HTTP Request URL: <http://localhost:3000/api/projects/68f4252eaaf0cb5c73b3d1e2>

Method: DELETE

Response Body:

```
1  {
2    "success": true,
3    "message": "Project deleted successfully."
4 }
```

RESULT (ONE PROJECT MISSING)

Postman screenshot showing a successful GET request to `http://localhost:3000/api/projects`. The response body is a JSON object representing a project:

```
1  {
2   "title": "Project title UPDATED TWO",
3   "completion": "2025-12-10T00:00:00Z",
4   "description": "My project descrip UPDATED TWO"
5 }
```

The response status is 200 OK with a 37 ms duration and 436 B size.

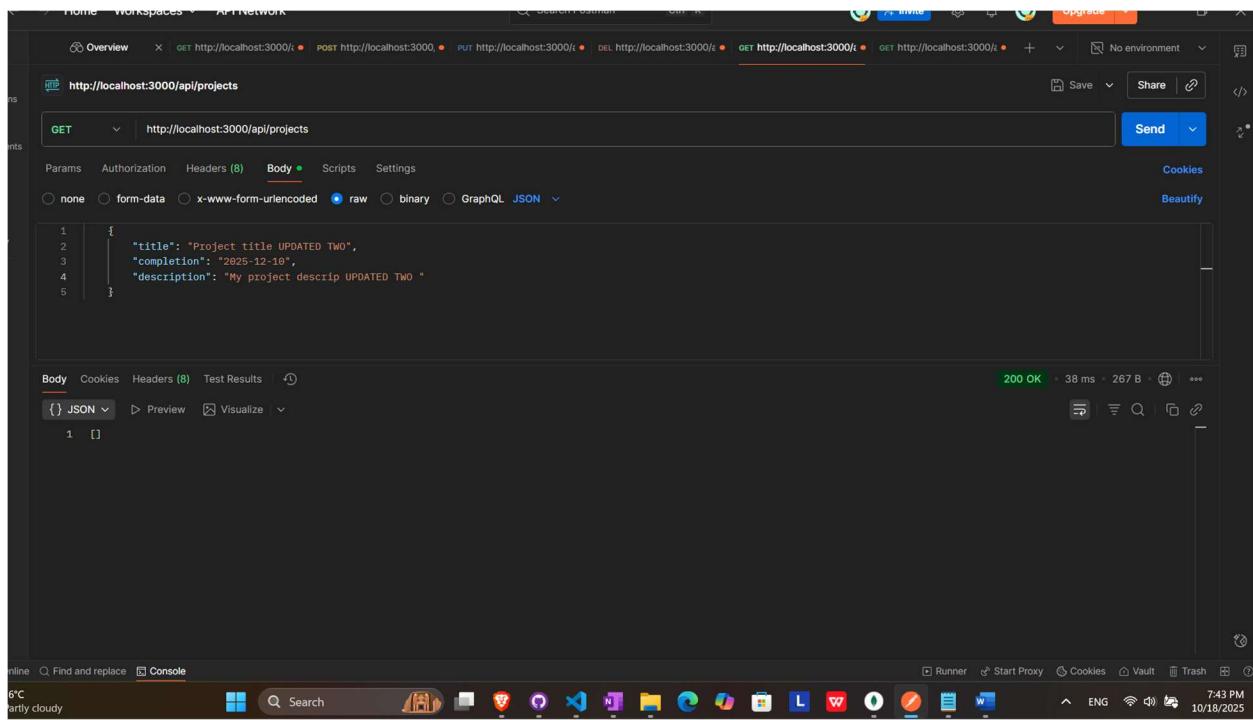
2.6 Remove all projects

Postman screenshot showing a successful DELETE request to `http://localhost:3000/api/projects`. The response body is a JSON object indicating success:

```
1  {
2   "success": true,
3   "message": "All projects were deleted successfully."
4 }
```

The response status is 200 OK with a 38 ms duration and 335 B size.

RESULT (ALL PROJECTS DELETED)



```
1  {
2   "title": "Project title UPDATED TWO",
3   "completion": "2025-12-10",
4   "description": "My project descrip UPDATED TWO"
5 }
```

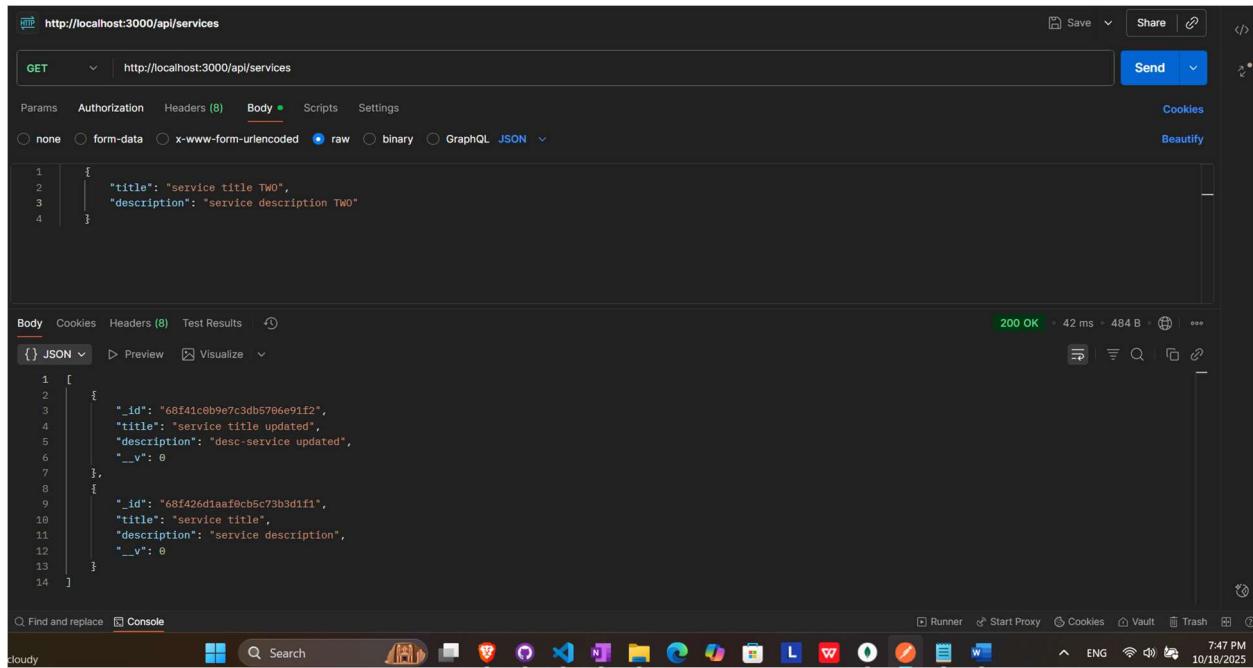
Body Cookies Headers (8) Test Results ⚡

{ JSON Preview Visualize

200 OK 38 ms 267 B

3 Services endpoints

3.1 Get all services



```
1 [
2   {
3     "_id": "68f41c0b9e7c3db5706e91f2",
4     "title": "service title updated",
5     "description": "desc-service updated",
6     "__v": 0
7   },
8   {
9     "_id": "68f426d1aaaf0cb5c73b3d1f1",
10    "title": "service title",
11    "description": "service description",
12    "__v": 0
13 }
14 ]
```

Body Cookies Headers (8) Test Results ⚡

{ JSON Preview Visualize

200 OK 42 ms 484 B

3.2 Add new service

The screenshot shows the Postman application interface. At the top, there's a header bar with tabs for Overview, GET, POST, PUT, DEL, and POST. Below the header, the URL is set to <http://localhost:3000/api/services>. The method is set to POST. The Body tab is selected, showing a raw JSON payload:

```
1 {  
2   "title": "service title",  
3   "description": "service description"  
4 }
```

Below the body, the Test Results section shows a successful response:

```
200 OK • 53 ms • 325 B • 7:46 PM  
{"success": true, "message": "Service created successfully."}
```

At the bottom, the status bar indicates it's 7:46 PM on 10/18/2025.

This screenshot shows another instance of the Postman application. The URL is again <http://localhost:3000/api/services> and the method is POST. The Body tab is selected with the same JSON payload:

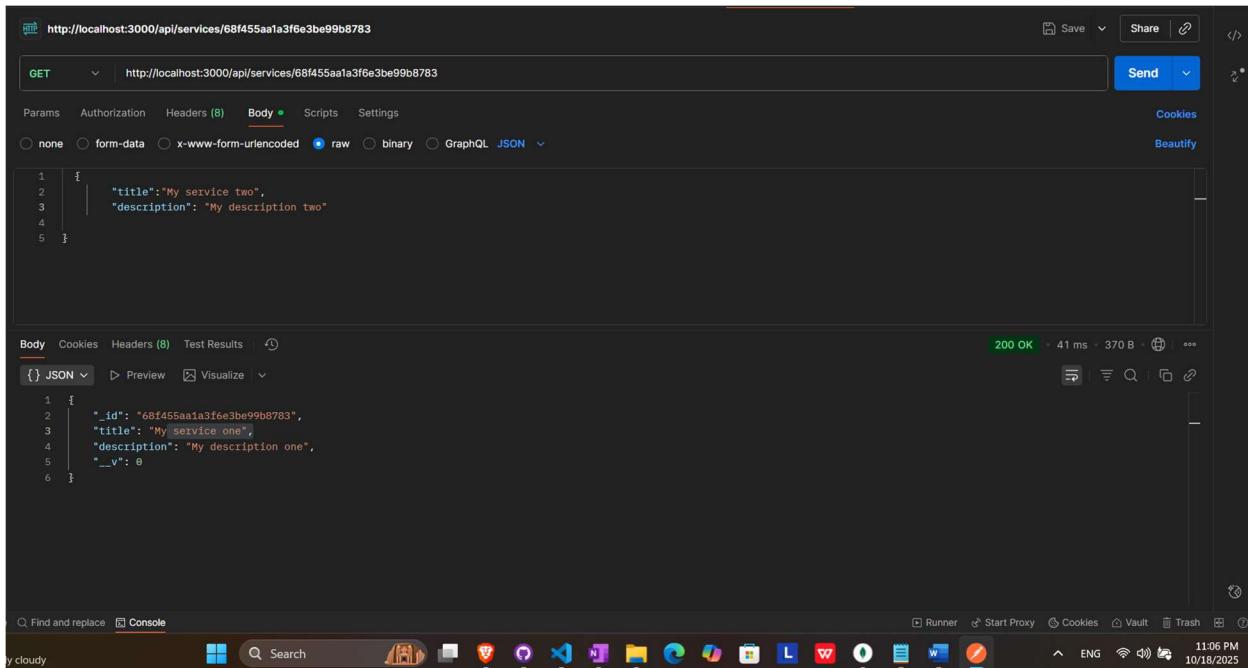
```
1 {  
2   "title": "service title TWO",  
3   "description": "service description TWO"  
4 }
```

The Test Results section shows a successful response:

```
200 OK • 53 ms • 325 B • 7:46 PM  
{"success": true, "message": "Service created successfully."}
```

The status bar at the bottom shows the same timestamp and date as the first screenshot.

3.3 Get service by id



HTTP <http://localhost:3000/api/services/68f455aa1a3fe3be99b8783>

GET <http://localhost:3000/api/services/68f455aa1a3fe3be99b8783>

Params Authorization Headers (8) Body Scripts Settings Cookies Beautify

Body

```
1 {
2   "title": "My service two",
3   "description": "My description two"
4 }
5 }
```

Headers (8) Test Results

200 OK 41 ms 370 B

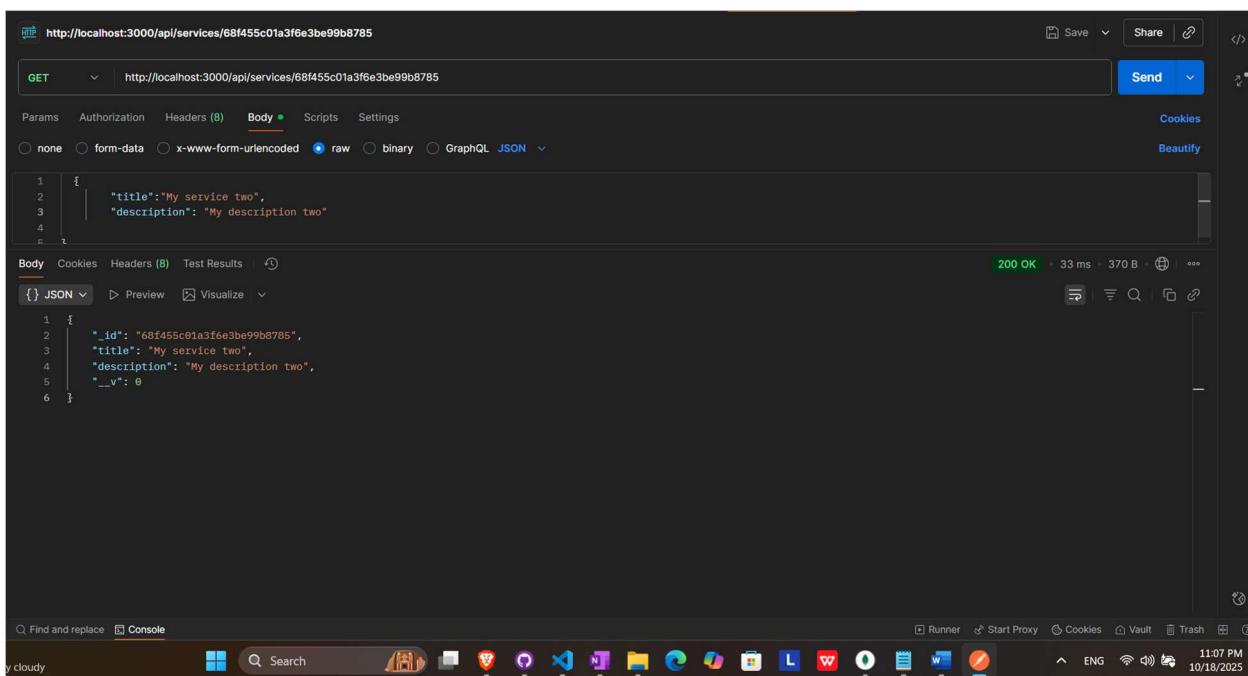
Body Cookies Headers (8) Test Results

{ } JSON Preview Visualize

```
1 {
2   "_id": "68f455aa1a3fe3be99b8783",
3   "title": "My service one",
4   "description": "My description one",
5   "__v": 0
6 }
```

Find and replace Console

Runner Start Proxy Cookies Vault Trash ENG 11:06 PM 10/18/2025



HTTP <http://localhost:3000/api/services/68f455c01a3fe3be99b8785>

GET <http://localhost:3000/api/services/68f455c01a3fe3be99b8785>

Params Authorization Headers (8) Body Scripts Settings Cookies Beautify

Body

```
1 {
2   "title": "My service two",
3   "description": "My description two"
4 }
5 }
```

Headers (8) Test Results

200 OK 33 ms 370 B

Body Cookies Headers (8) Test Results

{ } JSON Preview Visualize

```
1 {
2   "_id": "68f455c01a3fe3be99b8785",
3   "title": "My service two",
4   "description": "My description two",
5   "__v": 0
6 }
```

Find and replace Console

Runner Start Proxy Cookies Vault Trash ENG 11:07 PM 10/18/2025

3.4 Update service by id

The screenshot shows two separate instances of the Postman application interface, each displaying a successful PUT request to update a service record.

Request 1: A PUT request to `http://localhost:3000/api/services/68f426d1aaef0cb5c73b3d1f1`. The request body is:

```
1 | {  
2 |     "title": "service title TWO UPDATED",  
3 |     "description": "service description TWO UPDATED"  
4 | }
```

The response is a 200 OK status with a response body:

```
1 | {  
2 |     "success": true,  
3 |     "message": "Service updated successfully."  
4 | }
```

Request 2: A PUT request to `http://localhost:3000/api/services/68f41c0b9e7c3db5706e91f2`. The request body is:

```
1 | {  
2 |     "title": "service title ONE UPDATED",  
3 |     "description": "service description ONE UPDATED"  
4 | }
```

The response is a 200 OK status with a response body:

```
1 | {  
2 |     "success": true,  
3 |     "message": "Service updated successfully."  
4 | }
```

The Postman interface includes various tabs like Body, Cookies, Headers, and Test Results, and a bottom toolbar with icons for different tools and settings.

RESULTS (SERVICES UPDATED)

The screenshot shows the Postman interface with a successful GET request to `http://localhost:3000/api/services`. The response status is `200 OK` with a response time of 33 ms and a body size of 523 B. The response body is a JSON array containing two service objects:

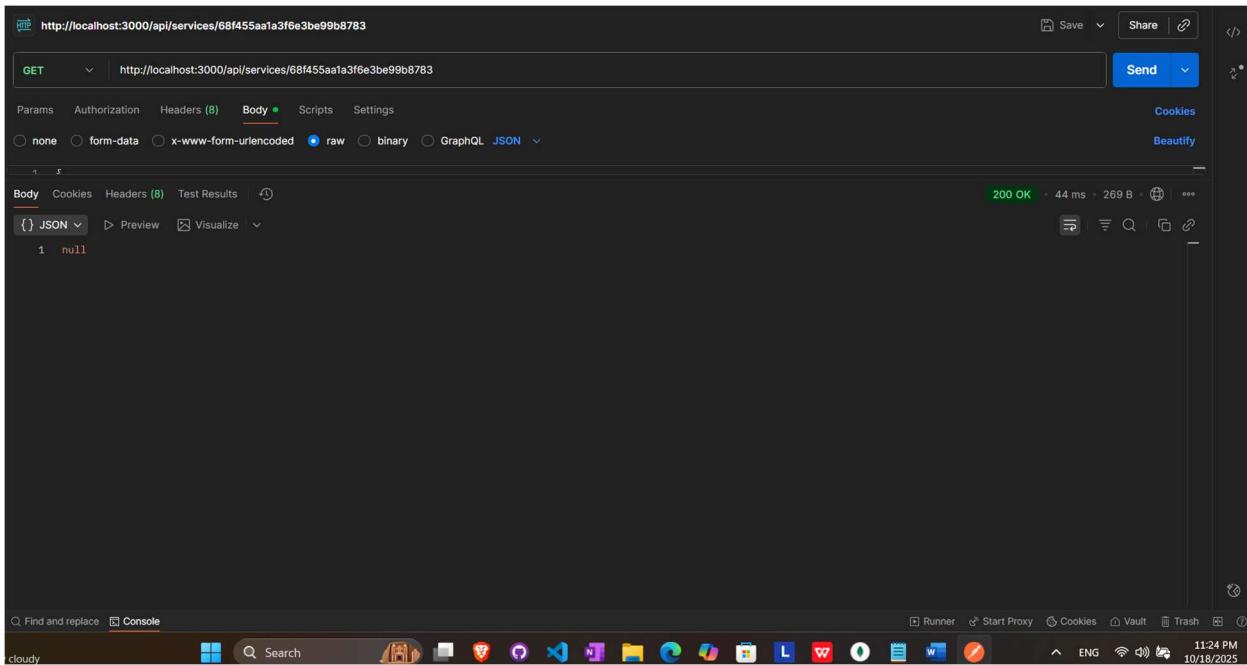
```
[{"_id": "68f41c0b9e7c3db5706e91f2", "title": "service title ONE UPDATED", "description": "service description ONE UPDATED", "__v": 0}, {"_id": "68f426d1aa0cb5c73b3d1f1", "title": "service title TWO UPDATED", "description": "service description TWO UPDATED", "__v": 0}]
```

3.5 Remove service by id

The screenshot shows the Postman interface with a successful DELETE request to `http://localhost:3000/api/services/68f455aa1a3f6e3be99b8783`. The response status is `200 OK` with a response time of 40 ms and a body size of 325 B. The response body is a JSON object:

```
{"success": true, "message": "Service deleted successfully."}
```

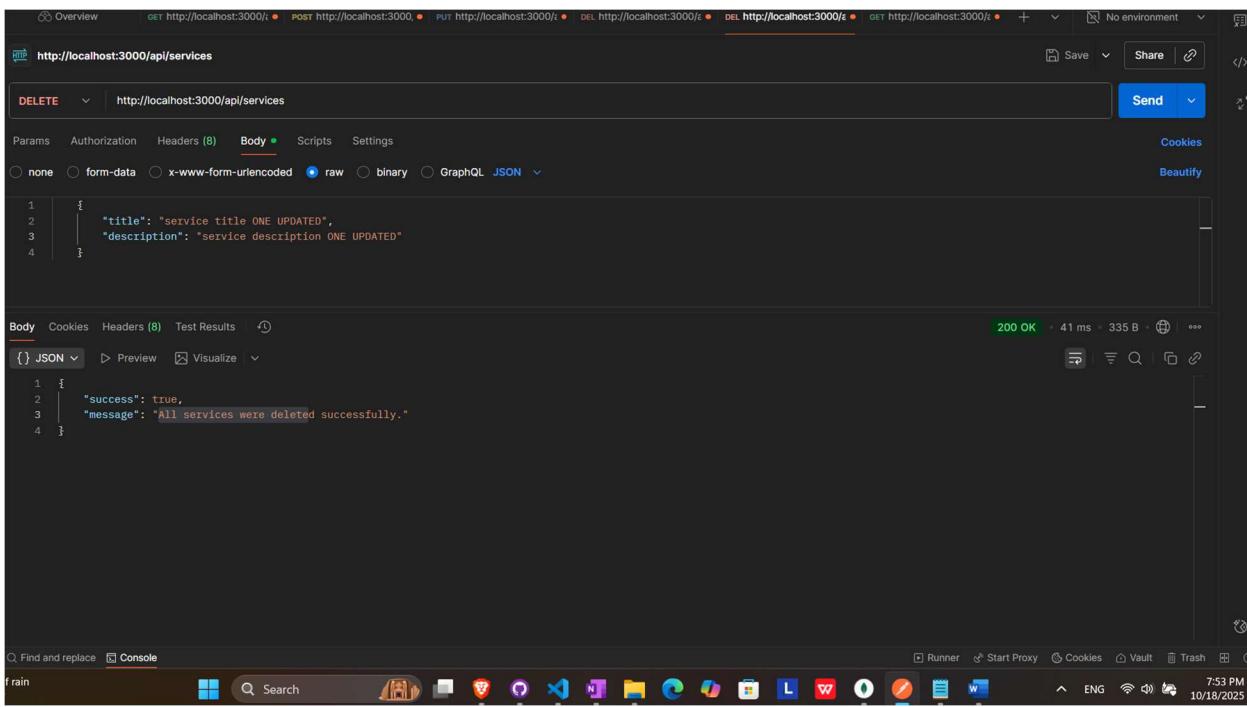
RESULT (SERVICE DELETED)



The screenshot shows a Postman interface with the following details:

- Method: GET
- URL: <http://localhost:3000/api/services/68f455aa1a3f6e3be99b8783>
- Body tab selected
- Response status: 200 OK
- Response time: 44 ms
- Response size: 269 B
- Response content: null

3.6 Remove all services



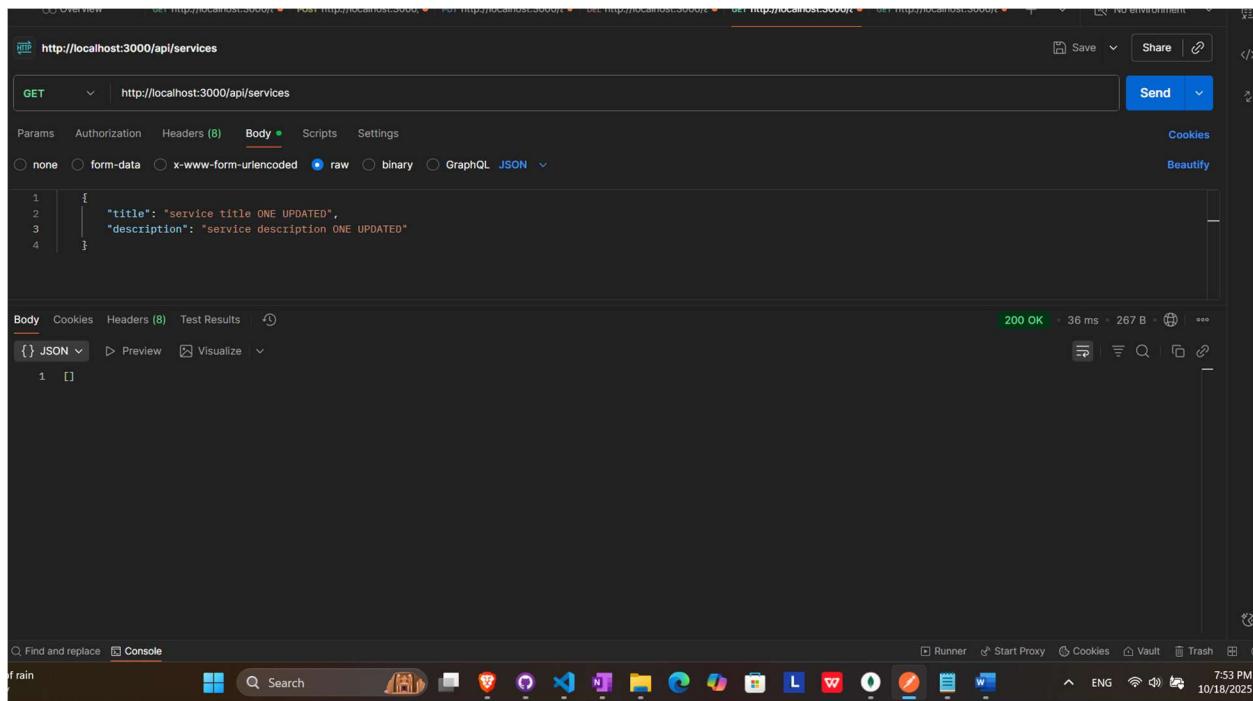
The screenshot shows a Postman interface with the following details:

- Method: DELETE
- URL: <http://localhost:3000/api/services>
- Body tab selected
- Request body (raw JSON):

```
1 | {  
2 |   "title": "service title ONE UPDATED",  
3 |   "description": "service description ONE UPDATED"  
4 | }
```
- Response status: 200 OK
- Response time: 41 ms
- Response size: 335 B
- Response content:

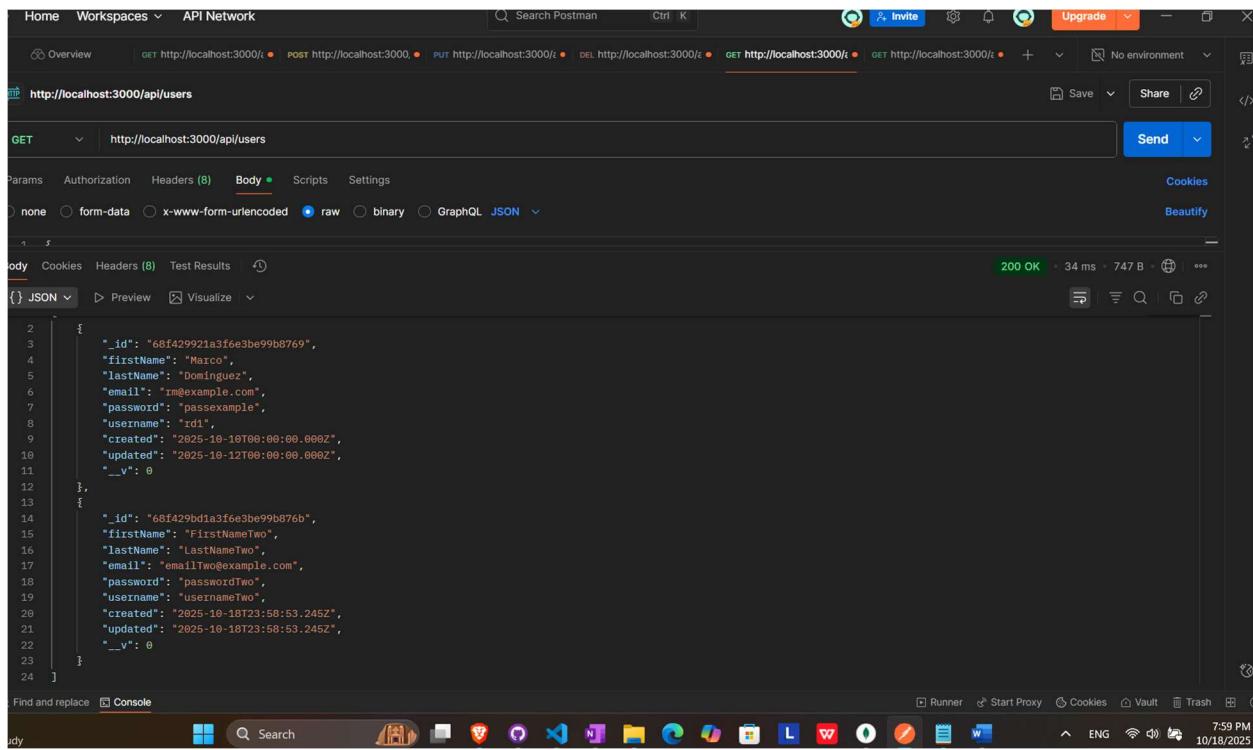
```
1 | {  
2 |   "success": true,  
3 |   "message": "All services were deleted successfully."  
4 | }
```

RESULT (ALL SERVICES DELETED)



4 Users endpoints

4.1 Get all users

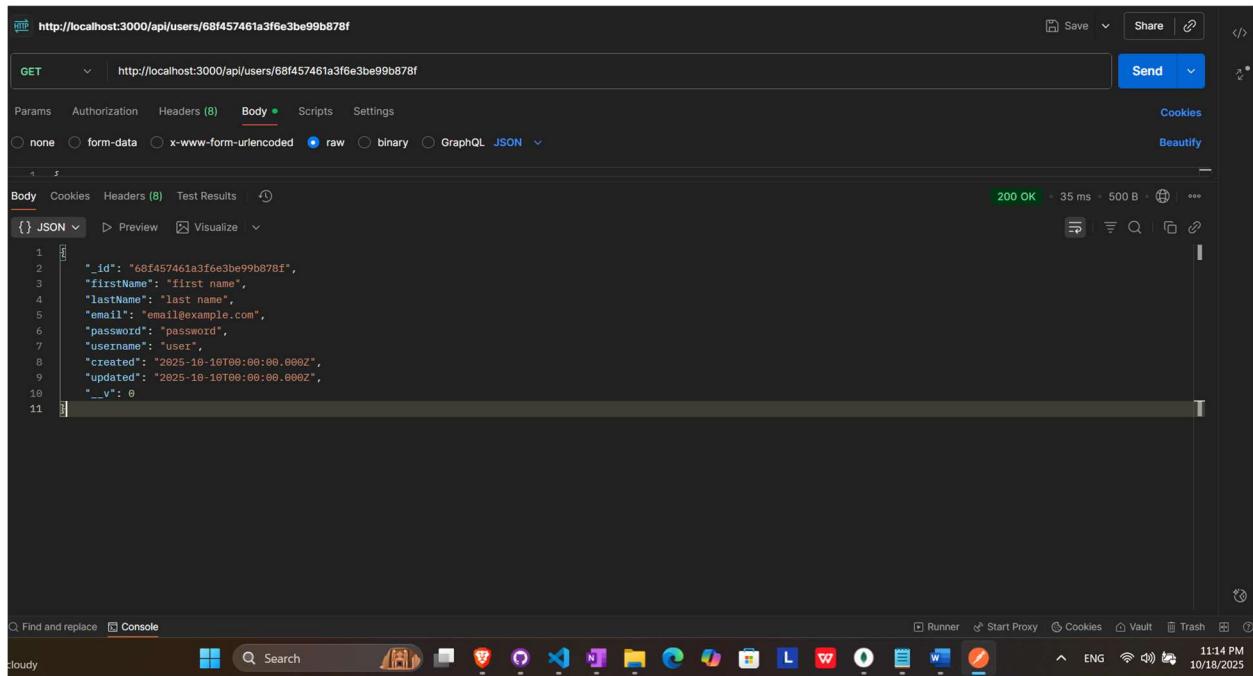


The screenshot shows the Postman application interface. At the top, there are tabs for Home, Workspaces, and API Network. A search bar says 'Search Postman' and a keyboard shortcut 'Ctrl K' is shown. Below the tabs, there's a toolbar with icons for Overview, Invite, Upgrade, and other functions. The main area shows a request URL 'http://localhost:3000/api/users' with a 'GET' method selected. The 'Body' tab is active, showing raw JSON data for two users:

```
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
{
  "_id": "68f429921a3f6e3be99b8769",
  "firstName": "Marco",
  "lastName": "Dominguez",
  "email": "im@example.com",
  "password": "passexample",
  "username": "rd1",
  "created": "2025-10-10T00:00:00.000Z",
  "updated": "2025-10-12T00:00:00.000Z",
  "__v": 0
},
{
  "_id": "68f429bd1a3f6e3be99b876b",
  "firstName": "FirstNameTwo",
  "lastName": "LastNameTwo",
  "email": "emailTwo@example.com",
  "password": "passwordTwo",
  "username": "usernameTwo",
  "created": "2025-10-18T23:58:53.245Z",
  "updated": "2025-10-18T23:58:53.245Z",
  "__v": 0
}
```

The response status is '200 OK' with a timestamp of '34 ms' and a size of '747 B'. Below the JSON preview, there are tabs for Body, Cookies, Headers, Test Results, and a dropdown for JSON, Preview, and Visualize. The bottom of the screen shows the Windows taskbar with various pinned icons like File Explorer, Microsoft Edge, and others.

4.2 Get User By Id



The screenshot shows the Postman application interface. At the top, there are tabs for Home, Workspaces, and API Network. A search bar says 'Search Postman' and a keyboard shortcut 'Ctrl K' is shown. Below the tabs, there's a toolbar with icons for Overview, Invite, Upgrade, and other functions. The main area shows a request URL 'http://localhost:3000/api/users/68f457461a3f6e3be99b878f' with a 'GET' method selected. The 'Body' tab is active, showing raw JSON data for one user:

```
1
2
3
4
5
6
7
8
9
10
11
{
  "_id": "68f457461a3f6e3be99b878f",
  "firstName": "first name",
  "lastName": "last name",
  "email": "email@example.com",
  "password": "password",
  "username": "user",
  "created": "2025-10-10T00:00:00.000Z",
  "updated": "2025-10-10T00:00:00.000Z",
  "__v": 0
}
```

The response status is '200 OK' with a timestamp of '35 ms' and a size of '500 B'. Below the JSON preview, there are tabs for Body, Cookies, Headers, Test Results, and a dropdown for JSON, Preview, and Visualize. The bottom of the screen shows the Windows taskbar with various pinned icons like File Explorer, Microsoft Edge, and others.

http://localhost:3000/api/users/68f457681a3fe3be99b8792

GET http://localhost:3000/api/users/68f457681a3fe3be99b8792

Params Authorization Headers (B) Body Scripts Settings

Body Cookies Headers (B) Test Results

200 OK 34 ms 519 B

Body JSON Preview Visualize

```
1 2 3 4 5 6 7 8 9 10 11
{
  "_id": "68f457681a3fe3be99b8792",
  "firstName": "second name",
  "lastName": "third name",
  "email": "email@email@mail.com",
  "password": "password_two",
  "username": "user_two",
  "created": "2025-10-10T00:00:00.000Z",
  "updated": "2025-10-10T00:00:00.000Z",
  "__v": 0
}
```

4.3 Add new user

The screenshot shows the Postman application interface. At the top, there's a navigation bar with tabs for Overview, GET, POST, PUT, and other HTTP methods. The URL is set to `http://localhost:3000/api/users`. Below the URL, a dropdown menu shows the selected method is `POST`, and the body type is `raw` with `JSON` selected. The `Body` tab contains a JSON object representing a user:

```
1 {  
2   "firstName": "Marco",  
3   "lastName": "Dominguez",  
4   "email": "rm@example.com",  
5   "password": "passexample",  
6   "username": "rd1",  
7   "created": "2025-10-10",  
8   "updated": "2025-10-12"  
9 }
```

Below the body, the `Body` tab is active, showing the raw JSON response from the server:

```
200 OK 67 ms 322 B  
{} JSON  
1 {  
2   "success": true,  
3   "message": "User created successfully."  
4 }
```

At the bottom, there are search and replace fields, a console tab, and system status icons.

The screenshot shows the Postman interface with a successful POST request to `http://localhost:3000/api/users`. The request body contains the following JSON:

```
1 {  
2   "firstName": "FirstNameTwo",  
3   "lastName": "LastNameTwo",  
4   "email": "emailTwo@example.com",  
5   "password": "passwordTwo",  
6   "username": "usernameTwo"  
7 }  
8 }
```

The response is a 200 OK status with a response time of 40 ms and a size of 322 B. The response body is:

```
1 {  
2   "success": true,  
3   "message": "User created successfully."  
4 }
```

4.4 Update user by id

The screenshot shows the Postman interface with a successful PUT request to `http://localhost:3000/api/users/68f429921a3f6e3be99b8769`. The request body contains the following JSON:

```
1 {  
2   "firstName": "Marco-UPDATED",  
3   "lastName": "Dominguez-UPDATED",  
4   "email": "xm_UPDATED@example.com",  
5   "password": "passexample-UPDATED",  
6   "username": "rdi-UPDATED"  
7 }  
8 }
```

The response is a 200 OK status with a response time of 54 ms and a size of 322 B. The response body is:

```
1 {  
2   "success": true,  
3   "message": "User updated successfully."  
4 }
```

The screenshot shows the Postman interface with a successful PUT request to `http://localhost:3000/api/users/68f429bd1a3f6e3be99b876b`. The request body contains updated user information:

```

1 {
2   "firstName": "FirstNameTwo-UPDATED",
3   "lastName": "LastNameTwo-UPDATED",
4   "email": "emailTwo-UPDATED@example.com",
5   "password": "passwordTwo-UPDATED",
6   "username": "usernameTwo-UPDATED"
7 }

```

The response status is 200 OK, with a message: "User updated successfully."

RESULT (ALL USERS UPDATED)

The screenshot shows the Postman interface with a successful GET request to `http://localhost:3000/api/users`. The response body contains two updated user documents:

```

1 [
2   {
3     "_id": "68f429921a3f6e3be99b8769",
4     "firstName": "Maico-UPDATED",
5     "lastName": "dominguez-UPDATED",
6     "email": "im_UPDATED@example.com",
7     "password": "passexample-UPDATED",
8     "username": "rdi-UPDATED",
9     "created": "2025-10-18T09:00:00.000Z",
10    "updated": "2025-10-19T09:00:48.765Z",
11    "__v": 0
12  },
13  {
14    "_id": "68f429bd1a3f6e3be99b876b",
15    "firstName": "FirstNameTwo-UPDATED",
16    "lastName": "LastNameTwo-UPDATED",
17    "email": "emailTwo-UPDATED@example.com",
18    "password": "passwordTwo-UPDATED",
19    "username": "usernameTwo-UPDATED",
20    "created": "2025-10-18T23:58:53.245Z",
21    "updated": "2025-10-19T09:02:41.918Z",
22    "__v": 0
23  }
]

```

4.5 Remove user by id

The screenshot shows the Postman interface with a successful DELETE request. The URL is `http://localhost:3000/api/users/68f429921a3f6e3be99b8769`. The response status is 200 OK, and the response body is:

```
{ "success": true, "message": "User deleted successfully." }
```

RESULT (USER DELETED)

The screenshot shows the Postman interface with a successful GET request. The URL is `http://localhost:3000/api/users/68f429921a3f6e3be99b8769`. The response status is 200 OK, and the response body is:

```
null
```

4.6 Remove all users

The screenshot shows a Postman request to `http://localhost:3000/api/users` using the `DELETE` method. The response status is `200 OK` with a response time of 37 ms and a body size of 332 B. The response JSON is:

```
1  {
2   "success": true,
3   "message": "All users were deleted successfully."
4 }
```

The Postman interface includes tabs for Body, Cookies, Headers (8), Test Results, and a preview of the JSON response. The bottom of the screen shows a taskbar with various application icons.

RESULTS (ALL USERS DELETED)

The screenshot shows a Postman request to `http://localhost:3000/api/users` using the `GET` method. The response status is `200 OK` with a response time of 36 ms and a body size of 267 B. The response JSON is:

```
1  []
```

The Postman interface includes tabs for Body, Cookies, Headers (8), Test Results, and a preview of the JSON response. The bottom of the screen shows a taskbar with various application icons.