CREATE:

Overview

The create endpoint for our project creates a new user with no exercise volume and default values for non-required parameters.

Endpoint URI and Parameters

The URI for interacting with the **create** endpoint:

```
.../create?<Parameter>=<Value>
```

There are seven required parameters and six optional parameters for this endpoint.

Parameter	Description	Example
uid	(Required) The email for the newly created account	localhost:8080/create?uid=value
pass	(Required) The password for the newly created account	localhost:8080/create?pass=value
name	(Required) The name of the user connected to the account	localhost:8080/create?name=value
age	(Required) The age of the user in years	localhost:8080/create?age=value
weight	(Required) The weight of the user in pounds	localhost:8080/create?weight=value
height	(Required) The height of the user in inches	localhost:8080/create?height=value
sex	(Required) The biological sex of the user Values: Male, Female, Other	localhost:8080/create?sex=value
chest	The volume of exercises the chest has trained Default Value: 0	localhost:8080/create?chest=value
back	The volume of exercises the	localhost:8080/create?back=value

	back has trained Default Value: 0	
arms		localhost:8080/create?arms=value
	The volume of exercises the arms has trained Default Value: 0	
legs		localhost:8080/create?legs=value
	The volume of exercises the legs has trained Default Value: 0	
abs		localhost:8080/create?abs=value
	The volume of exercises the abs has trained Default Value: 0	
exercises		localhost:8080/create?exercises=[value]
	The list of exercises that the user has completed Default Value: [] (empty_array)	

Responses

The counter API returns all response data as a JSON object. The full details for the response format are shown below:

Key	Value Type	Description
result	string	The type of operation status: one of "created" or "error"
name	string	The name of the newly created counter

READ:

Overview

The read endpoint for our project reads the user data

Endpoint URI and Parameters

The URI for interacting with the **create** endpoint:

```
.../read?<Parameter>
```

There are two required parameters and eleven optional parameters for this endpoint.

Parameter	Description	Example
uid	(Required)The email for the created account	localhost:8080/read?uid
pass	(Required)The password for the created account	localhost:8080/read?pass
name	The name of the user connected to the account	localhost:8080/read?name
age	The age of the user in years	localhost:8080/read?age
weight	The weight of the user in pounds	localhost:8080/read?weight
height	The height of the user in inches	localhost:8080/read?height
sex	The biological sex of the user	localhost:8080/read?sex
chest	The volume of exercises the chest has trained	localhost:8080/read?chest
back	The volume of exercises the back has trained	localhost:8080/read?back
arms	The volume of exercises the arms has trained	localhost:8080/read?arms
legs	The volume of exercises the legs has trained	localhost:8080/read?legs
abs	The volume of exercises the abs has trained	localhost:8080/read?abs
exercises	The list of exercises that the user has completed Default Value: [] (empty_array)	localhost:8080/read?exercises

Responses

The counter API returns all response data as a JSON object. The full details for the response format are shown below:

Key	Value Type	Description
result	string	The type of operation status: one of "retrieved" or "error"
value	string	The response retrieved from the database

UPDATE:

Overview

The update endpoint for our project updates the object with user data

Endpoint URI and Parameters

The URI for interacting with the **update** endpoint:

```
.../update?<Parameter>
```

There are two required parameters and eleven optional parameters for this endpoint.

Parameter	Description	Example
uid	(Required)The email for the created account	localhost:8080/update?uid=value
pass	(Required)The password for the created account	localhost:8080/update?pass=value
name	The name of the user connected to the account	localhost:8080/update?name=value
age	The age of the user in years	localhost:8080/update?age=value
weight	The weight of the user in pounds	localhost:8080/update?weight=value
height	The height of the user in inches	localhost:8080/update?height=value

sex	The biological sex of the user	localhost:8080/update?sex=value
chest	The volume of exercises the chest has trained	localhost:8080/update?chest=value
back	The volume of exercises the back has trained	localhost:8080/update?back=value
arms	The volume of exercises the arms has trained	localhost:8080/update?arms=value
legs	The volume of exercises the legs has trained	localhost:8080/update?legs=value
abs	The volume of exercises the abs has trained	localhost:8080/update?abs=value
exercises	The list of exercises that the user has completed	localhost:8080/update?exercises=value

Responses

The counter API returns all response data as a JSON object. The full details for the response format are shown below:

Key	Value Type	Description
result	string	The type of operation status: one of "updated" or "error"
value	string	The updated part of the profile

DELETE:

Overview

The delete endpoint for our project delete the object

Endpoint URI and Parameters

The URI for interacting with the **update** endpoint:

```
.../delete?<Parameter>
```

There are two required parameters.

Parameter	Description	Example
uid	(Required)The email for the created account	localhost:8080/update?uid=value
pass	(Required)The password for the created account	localhost:8080/update?pass=value

Responses

The counter API returns all response data as a JSON object. The full details for the response format are shown below:

Key	Value Type	Description
result	string	The type of operation status: one of "deleted" or "error"

Heroku App URL: https://polar-gorge-31936.herokuapp.com/

Group Participation:
Zach: Set up most the Herokuapp and did a lot of the initial ts skeleton work and most of the routing for the client
Lucas: Cleaned up a lot of the data interaction, server side ts and a lot of the conversion from the skeleton code
Nick: Debugged a good amount of the ts, worked on the js for the pages, and wrote up the documentation