Table Of Contents

Introduction

For the development of this project, I used Express Js - Sequelize - PostgreSQL - REST - JWT. It was done through 4 main steps that I will describe backwards; programmatically, things were done starting with the models, but for the sake of this report I would rather start with the Routers to follow the service-driven sequence diagram's logic.

Routers

Routers are the first contact the user gets with the backend, they are the ones that treat the HTTP requests and find the right controller to deal with those requests.

We specify the type of request with the REST verbs (get/ post/ put/ delete) along with the API path ("/getUsers") for example, then forward the request to the appropriate controller that will handle everything about it.

Controllers

The number of controllers matches the number of services for the controllers are what call the services on the behalf of the client.

The controllers contain a collection of functions that simply forward the requests to the right service, and send responses back. They take the request parameters and pass them to the service that might need them for the operations to be executed. Again, the controller knows nothing of the business logic is it calling and contains no logic itself.

The response the controller sends is in a JSON format and can be configured easily.

Services

There are 4 services as specified in the code and following from the function requirements of the project.

The user service is where all the functions related to users are implemented and then exported to be used elsewhere. In that service there are functions for creating and retrieving users with their association, searching for users using certain attributes, updating users' relevant information (there are more functions in the service but I chose not expose them to the client because they didn't add much to the project as it is)

I used JWT for user authentication in the Login() function. I create a token for a user when the authentication is successful then use that token when the user wants to call other services. In the project, token verification is only called once when trying to create new admins **- not because it is only needed there! -** it was just to make testing easier all the while demonstrating that the feature works as expected.

The pet/rating/request services also have their own specific functions and are exported to be used in their respective controllers.

Models

I created sequelize models that represented the different entities in my project, then defined them with the necessary attributes, which were later converted into tables with records and fields.

The attributes are given a type using sequelize's **DataTypes** library.

The associations are made by a sequelize method called **associate()** which takes the instance of the class that extends the Sequelize Model, and links it to another through "hasOne()" or "hasMany()" ,etc, associations.

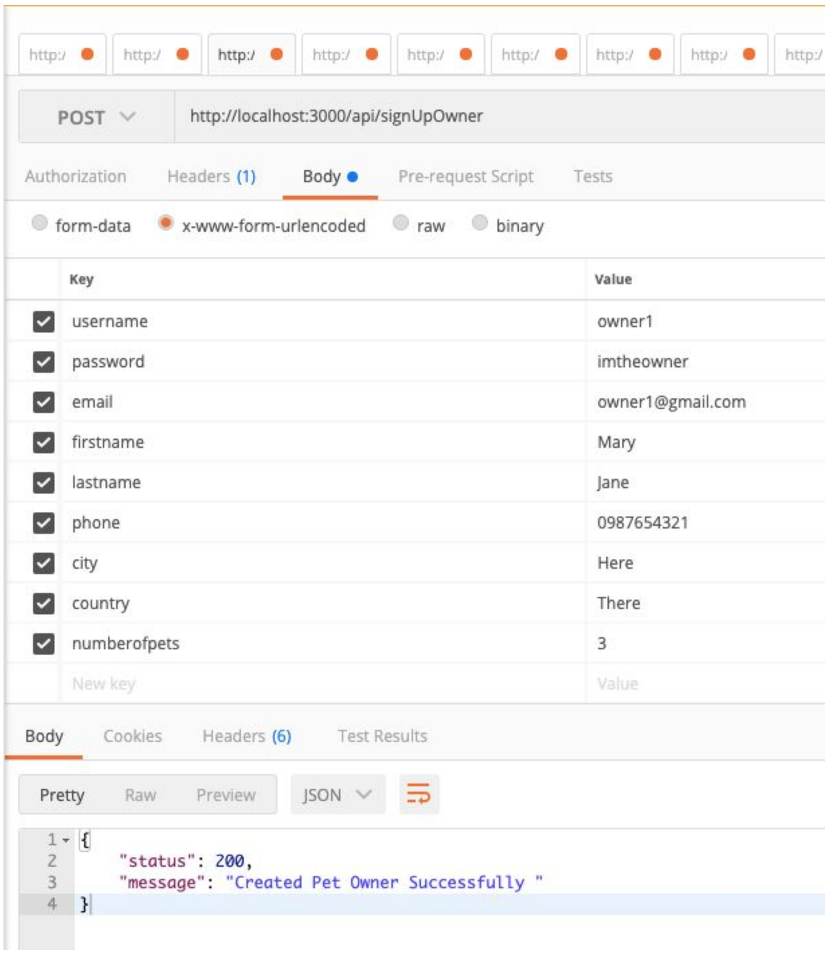
In the options of the associate() method, we can specify the name of the foreign key and any other configuration relevant to the two related models.

User Service Testing

To make the work easier, I tested all my services using **Postman** and all are working as expected to. Find below screenshots from my client requests and different scenarios that were interesting/relevant to handle.

Signing up as a pet Owner

When signing up as a pet owner/sitter: the service creates a user account first of all, then creates the pet sitter/owner record by association, as well as a rating record for the pet sitters only.



POST http://localhost:3000/api/signUpOwner

Authorization | Headers (1) | Body | Pre-request Script | Tests

form-data | [✓ x-www-form-urlencoded] | raw | binary

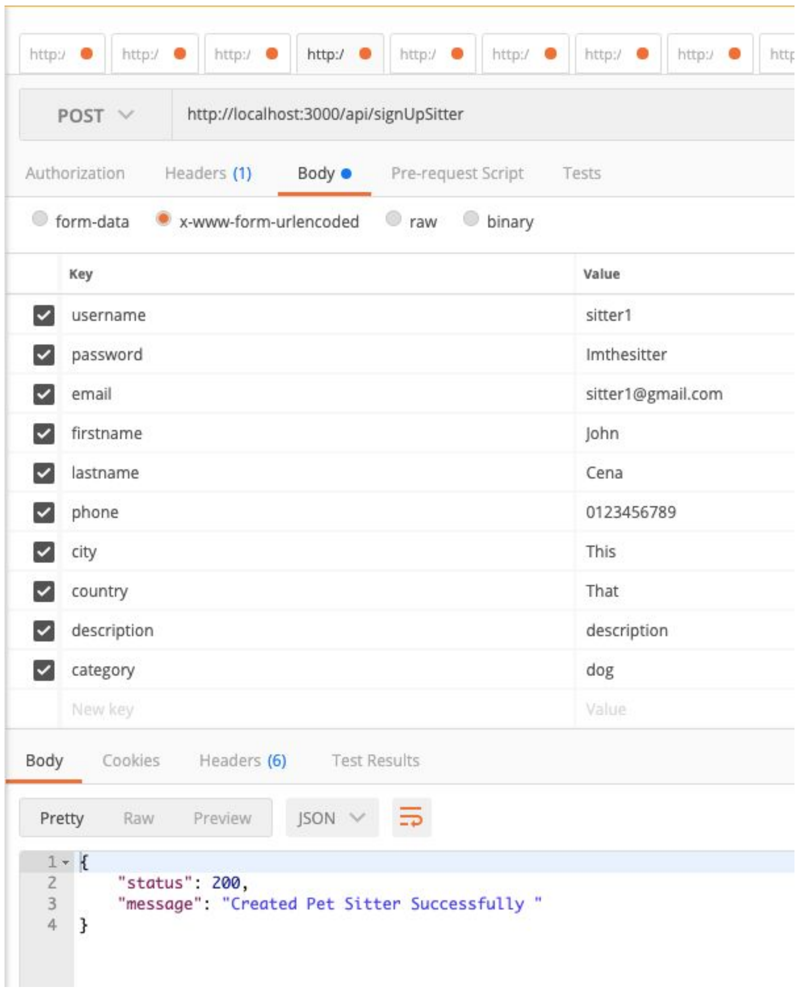
	Key	Value
✓	username	owner1
✓	password	imtheowner
✓	email	owner1@gmail.com
✓	firstname	Mary
✓	lastname	Jane
✓	phone	0987654321
✓	city	Here
✓	country	There
✓	numberofpets	3
	New key	Value

Body | Cookies | Headers (6) | Test Results

Pretty | Raw | Preview | JSON

```
{
    "status": 200,
    "message": "Created Pet Owner Successfully "
}
```

Signing up as a pet Owner



POST http://localhost:3000/api/signUpSitter

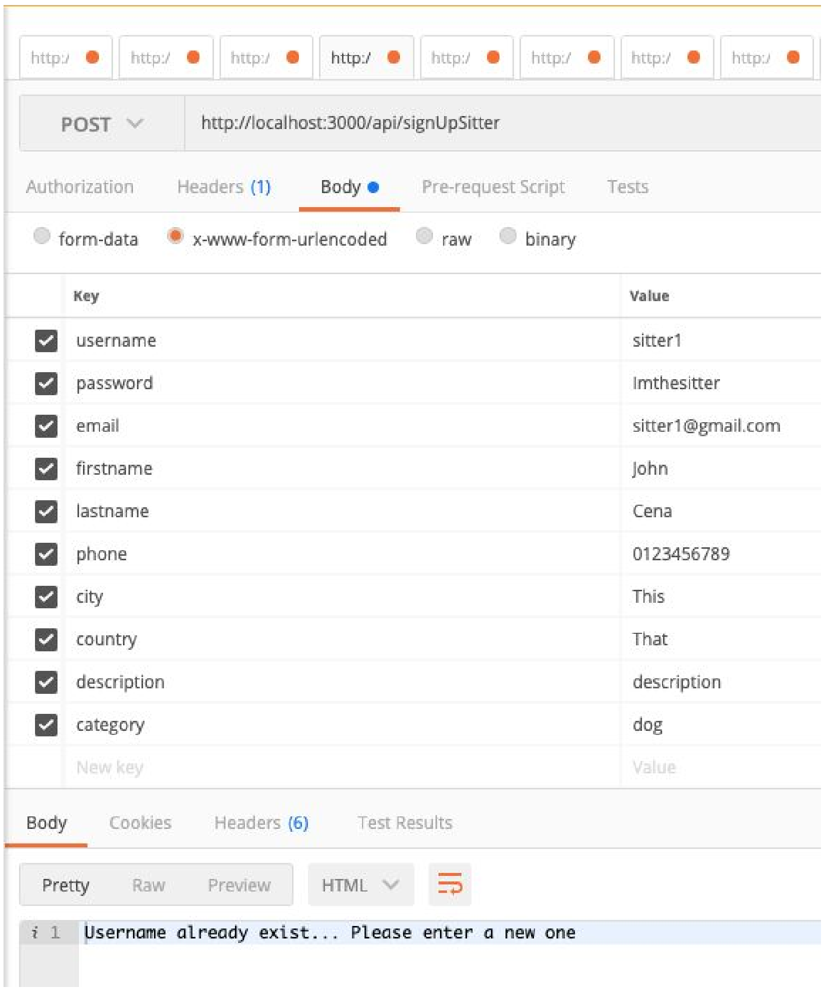
[form-data] [✓ x-www-form-urlencoded] [raw] [binary]

	Key	Value
✓	username	sitter1
✓	password	Imthesitter
✓	email	sitter1@gmail.com
✓	firstname	John
✓	lastname	Cena
✓	phone	0123456789
✓	city	This
✓	country	That
✓	description	description
✓	category	dog
	New key	Value

JSON

```
{
    "status": 200,
    "message": "Created Pet Sitter Successfully "
}
```

Trying to Sign up with a username that already exists in the database:



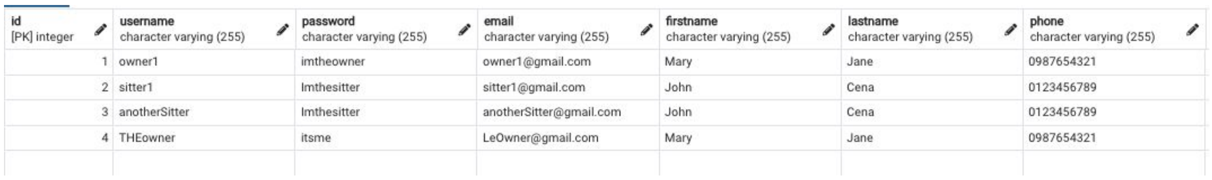
POST http://localhost:3000/api/signUpSitter

Body: x-www-form-urlencoded

Key	Value
username	sitter1
password	Imthesitter
email	sitter1@gmail.com
firstname	John
lastname	Cena
phone	0123456789
city	This
country	That
description	description
category	dog

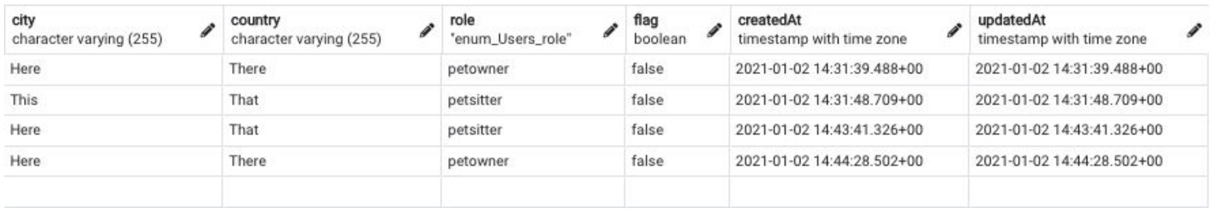
```
Username already exist... Please enter a new one
```

Database User Table



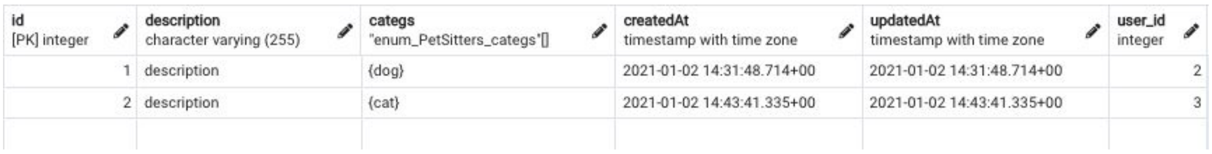
id [PK] integer	username character varying (255)	password character varying (255)	email character varying (255)	firstname character varying (255)	lastname character varying (255)	phone character varying (255)
1	owner1	imtheowner	owner1@gmail.com	Mary	Jane	0987654321
2	sitter1	Imthesitter	sitter1@gmail.com	John	Cena	0123456789
3	anotherSitter	Imthesitter	anotherSitter@gmail.com	John	Cena	0123456789
4	THEowner	itsme	LeOwner@gmail.com	Mary	Jane	0987654321

→ cont.



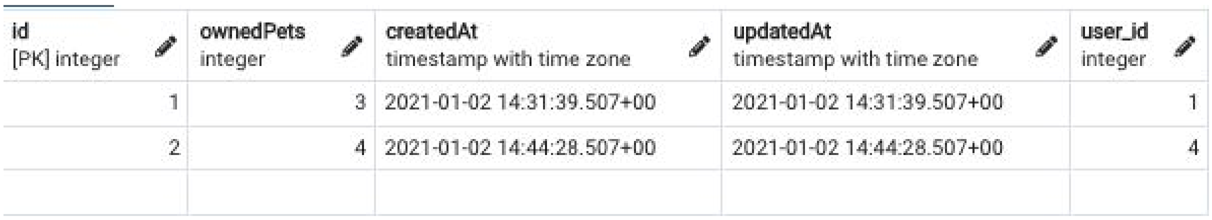
city character varying (255)	country character varying (255)	role "enum_Users_role"	flag boolean	createdAt timestamp with time zone	updatedAt timestamp with time zone
Here	There	petowner	false	2021-01-02 14:31:39.488+00	2021-01-02 14:31:39.488+00
This	That	petsitter	false	2021-01-02 14:31:48.709+00	2021-01-02 14:31:48.709+00
Here	That	petsitter	false	2021-01-02 14:43:41.326+00	2021-01-02 14:43:41.326+00
Here	There	petowner	false	2021-01-02 14:44:28.502+00	2021-01-02 14:44:28.502+00

Database PetSitter Table



id [PK] integer	description character varying (255)	categs "enum_PetSitters_categs"[]	createdAt timestamp with time zone	updatedAt timestamp with time zone	user_id integer
1	description	{dog}	2021-01-02 14:31:48.714+00	2021-01-02 14:31:48.714+00	2
2	description	{cat}	2021-01-02 14:43:41.335+00	2021-01-02 14:43:41.335+00	3

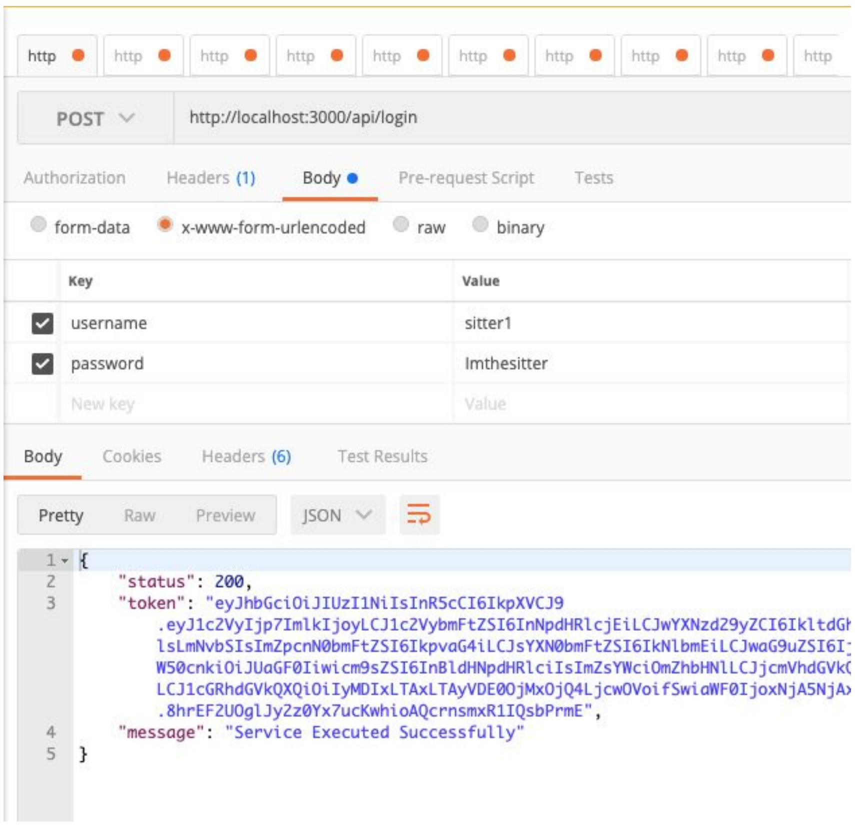
Database PetOwner Table



id [PK] integer	ownedPets integer	createdAt timestamp with time zone	updatedAt timestamp with time zone	user_id integer
1	3	2021-01-02 14:31:39.507+00	2021-01-02 14:31:39.507+00	1
2	4	2021-01-02 14:44:28.507+00	2021-01-02 14:44:28.507+00	4

User login

We can see the creation of a token when the authentication was successful.



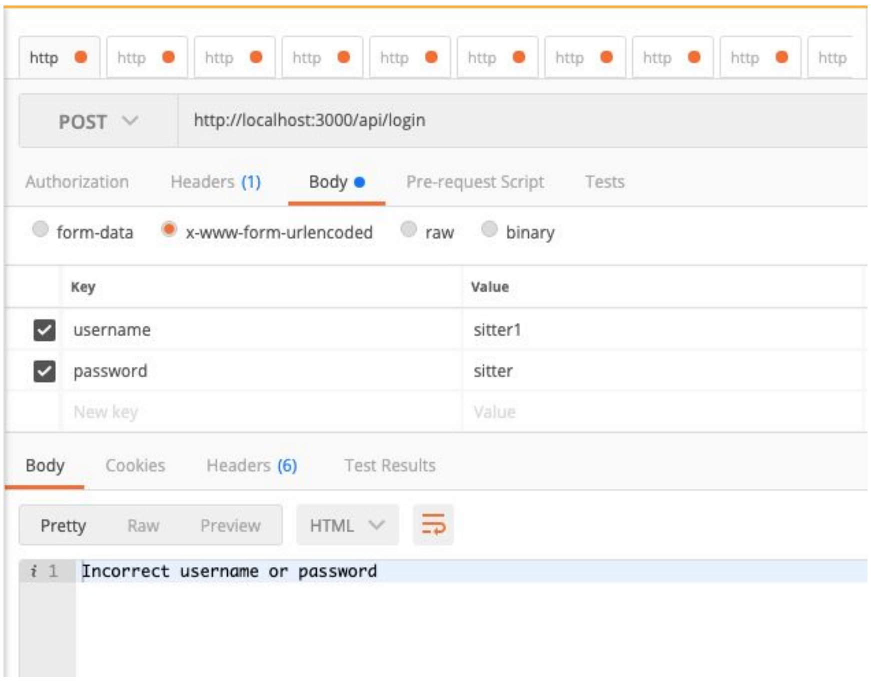
POST http://localhost:3000/api/login

Body: x-www-form-urlencoded

	Key	Value
✓	username	sitter1
✓	password	Imthesitter
	New key	Value

```
{
    "status": 200,
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9
        .eyJ1c2VyIjp7ImlkIjoyLCJ1c2VybmFtZSI6InNpdHRlcjEiLCJwYXNzd29yZCI6IkltdG
        lsLmNvbSIsImZpcnN0bmFtZSI6IkpvaG4iLCJsYXN0bmFtZSI6IkNlbmEiLCJwaG9uZSI6I
        W50cnkiOiJUaGF0Iiwicm9sZSI6InBldHNpdHRlciIsImZsYWciOmZhbHNlLCJjcmVhdGVk
        LCJ1cGRhdGVkQXQiOiIyMDIxLTAxLTAyVDE0OjMxOjQ4LjcwOVoifSwiaWF0IjoxNjA5NjA
        .8hrEF2UOglJy2z0Yx7ucKwhioAQcrnsmxR1IQsbPrmE",
    "message": "Service Executed Successfully"
}
```

Login with wrong username/password



POST http://localhost:3000/api/login

Body: x-www-form-urlencoded

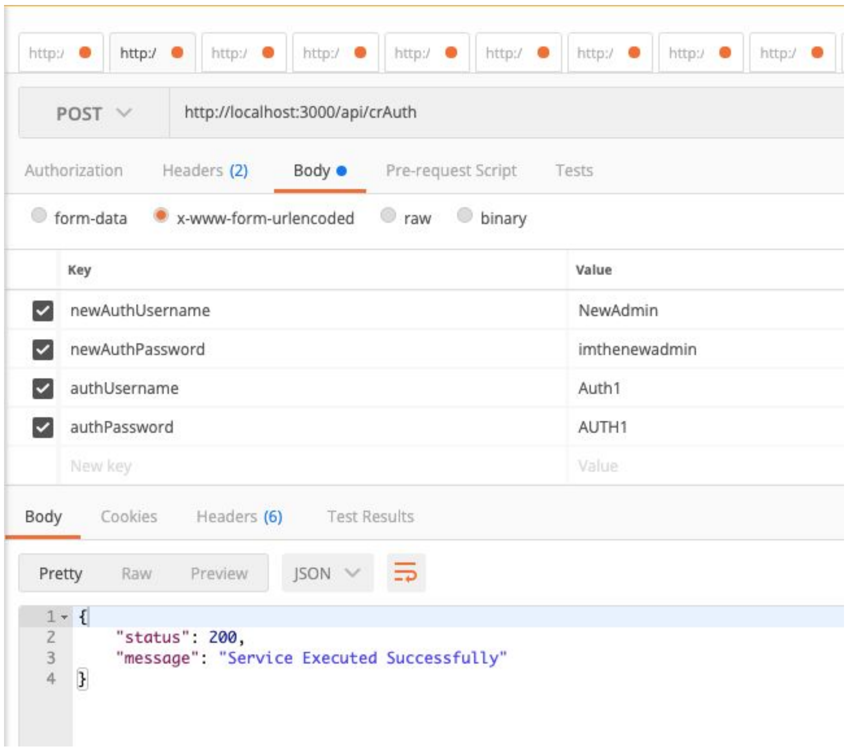
Key	Value
username	sitter1
password	sitter

HTML

```
Incorrect username or password
```

Creating Admin account

Here the service is authenticating the old admin + creating the new admin



POST http://localhost:3000/api/crAuth

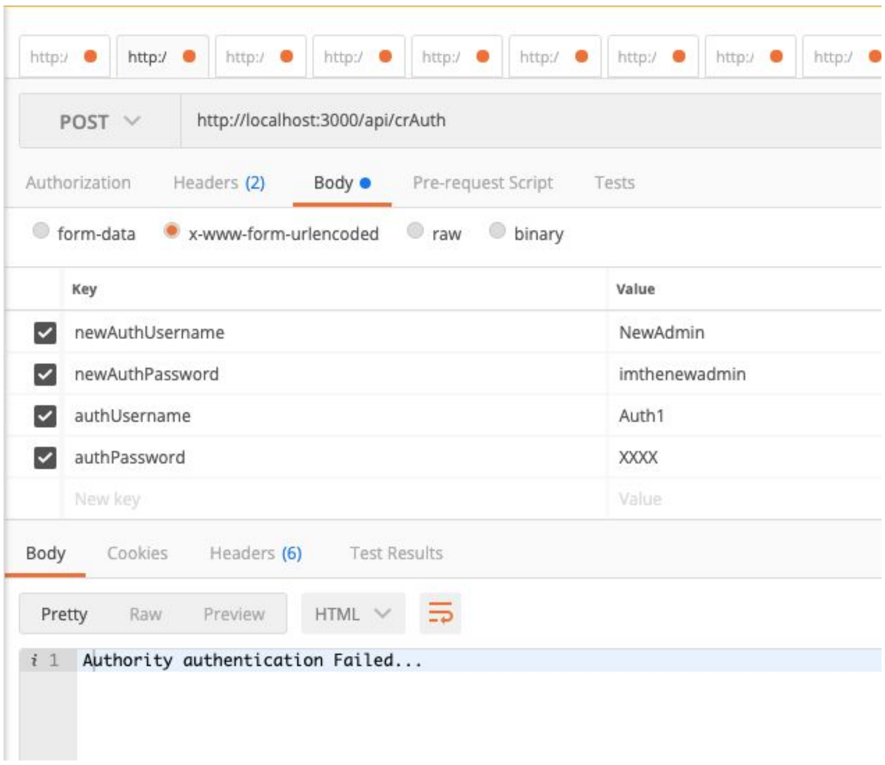
Body: x-www-form-urlencoded

Key	Value
newAuthUsername	NewAdmin
newAuthPassword	imthenewadmin
authUsername	Auth1
authPassword	AUTH1

JSON

```
{
    "status": 200,
    "message": "Service Executed Successfully"
}
```

Wrong admin password

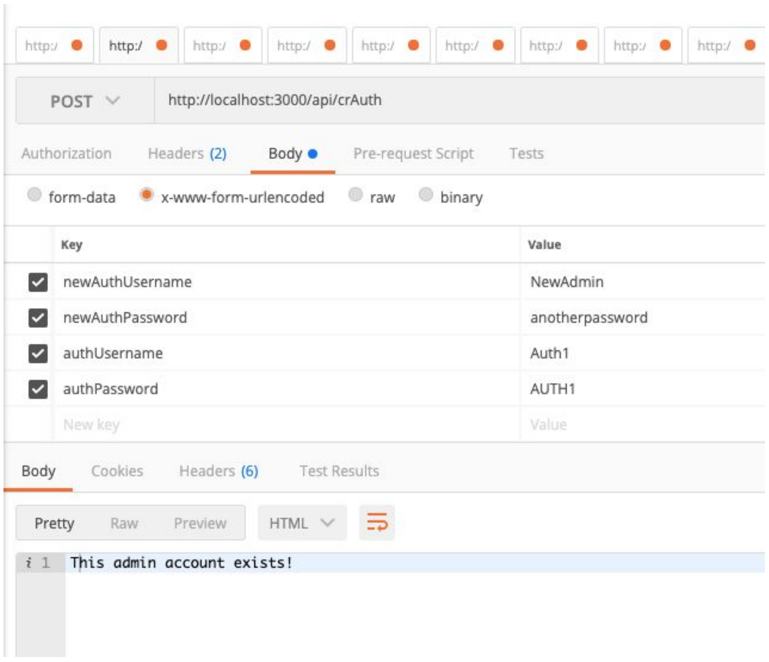


POST http://localhost:3000/api/crAuth

Key	Value
✓ newAuthUsername	NewAdmin
✓ newAuthPassword	imthenewadmin
✓ authUsername	Auth1
✓ authPassword	XXXX
New key	Value

```
Authority authentication Failed...
```

Creating an admin that already exists in the database

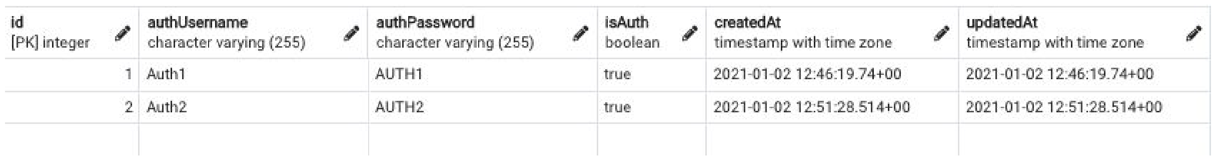


POST http://localhost:3000/api/crAuth

Key	Value
✓ newAuthUsername	NewAdmin
✓ newAuthPassword	anotherpassword
✓ authUsername	Auth1
✓ authPassword	AUTH1
New key	Value

```
This admin account exists!
```

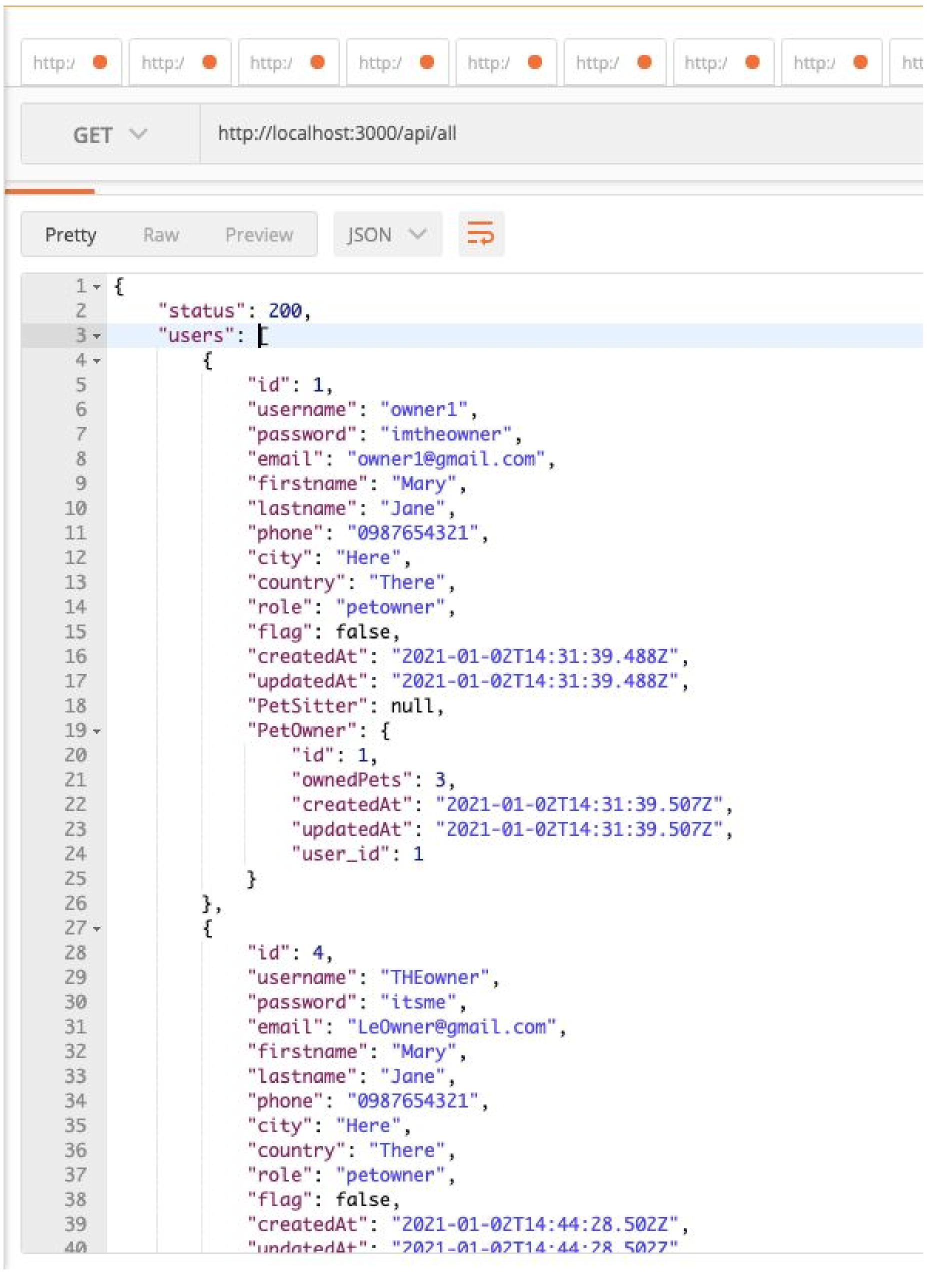
Database Authority Table



id [PK] integer	authUsername character varying (255)	authPassword character varying (255)	isAuth boolean	createdAt timestamp with time zone	updatedAt timestamp with time zone
1	Auth1	AUTH1	true	2021-01-02 12:46:19.74+00	2021-01-02 12:46:19.74+00
2	Auth2	AUTH2	true	2021-01-02 12:51:28.514+00	2021-01-02 12:51:28.514+00

Retrieving all the Users (both pet owner/sitter)

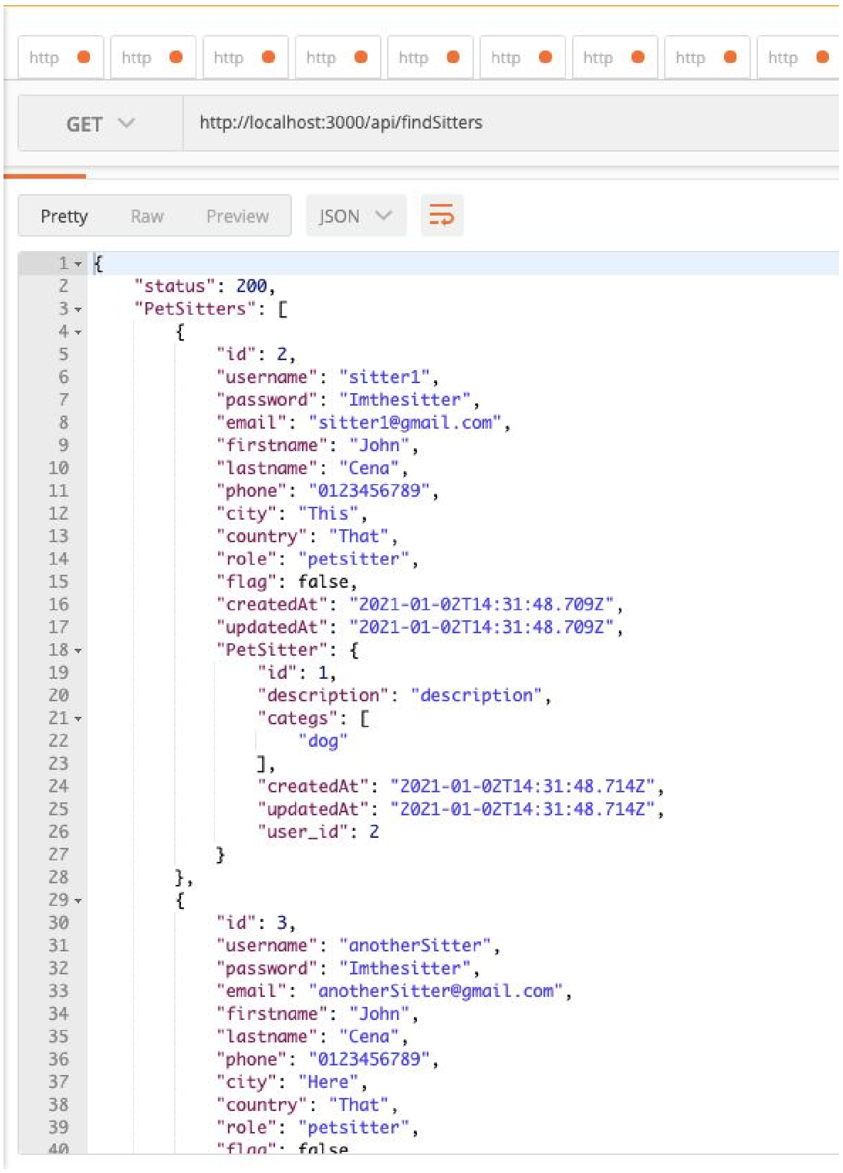
The user accounts are retrieved and their linked pet owner/sitter records are included as well.



http://localhost:3000/api/all

```
{
    "status": 200,
    "users": [
        {
            "id": 1,
            "username": "owner1",
            "password": "imtheowner",
            "email": "owner1@gmail.com",
            "firstname": "Mary",
            "lastname": "Jane",
            "phone": "0987654321",
            "city": "Here",
            "country": "There",
            "role": "petowner",
            "flag": false,
            "createdAt": "2021-01-02T14:31:39.488Z",
            "updatedAt": "2021-01-02T14:31:39.488Z",
            "PetSitter": null,
            "PetOwner": {
                "id": 1,
                "ownedPets": 3,
                "createdAt": "2021-01-02T14:31:39.507Z",
                "updatedAt": "2021-01-02T14:31:39.507Z",
                "user_id": 1
            }
        },
        {
            "id": 4,
            "username": "THEowner",
            "password": "itsme",
            "email": "LeOwner@gmail.com",
            "firstname": "Mary",
            "lastname": "Jane",
            "phone": "0987654321",
            "city": "Here",
            "country": "There",
            "role": "petowner",
            "flag": false,
            "createdAt": "2021-01-02T14:44:28.502Z",
            "updatedAt": "2021-01-02T14:44:28.502Z",
```

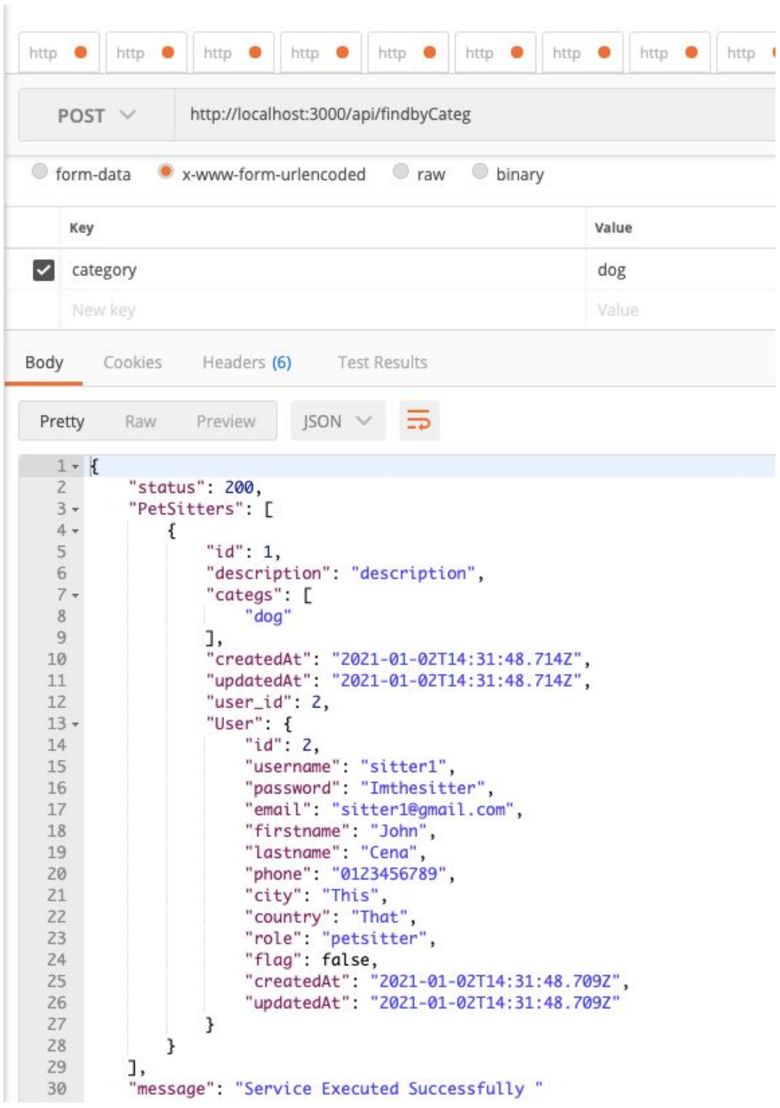
Retrieving only Pet sitters



GET http://localhost:3000/api/findSitters

```
{
    "status": 200,
    "PetSitters": [
        {
            "id": 2,
            "username": "sitter1",
            "password": "Imthesitter",
            "email": "sitter1@gmail.com",
            "firstname": "John",
            "lastname": "Cena",
            "phone": "0123456789",
            "city": "This",
            "country": "That",
            "role": "petsitter",
            "flag": false,
            "createdAt": "2021-01-02T14:31:48.709Z",
            "updatedAt": "2021-01-02T14:31:48.709Z",
            "PetSitter": {
                "id": 1,
                "description": "description",
                "categs": [
                    "dog"
                ],
                "createdAt": "2021-01-02T14:31:48.714Z",
                "updatedAt": "2021-01-02T14:31:48.714Z",
                "user_id": 2
            }
        },
        {
            "id": 3,
            "username": "anotherSitter",
            "password": "Imthesitter",
            "email": "anotherSitter@gmail.com",
            "firstname": "John",
            "lastname": "Cena",
            "phone": "0123456789",
            "city": "Here",
            "country": "That",
            "role": "petsitter",
            "flag": false
```

Searching pet sitters by pet category



```
POST http://localhost:3000/api/findbyCateg
```

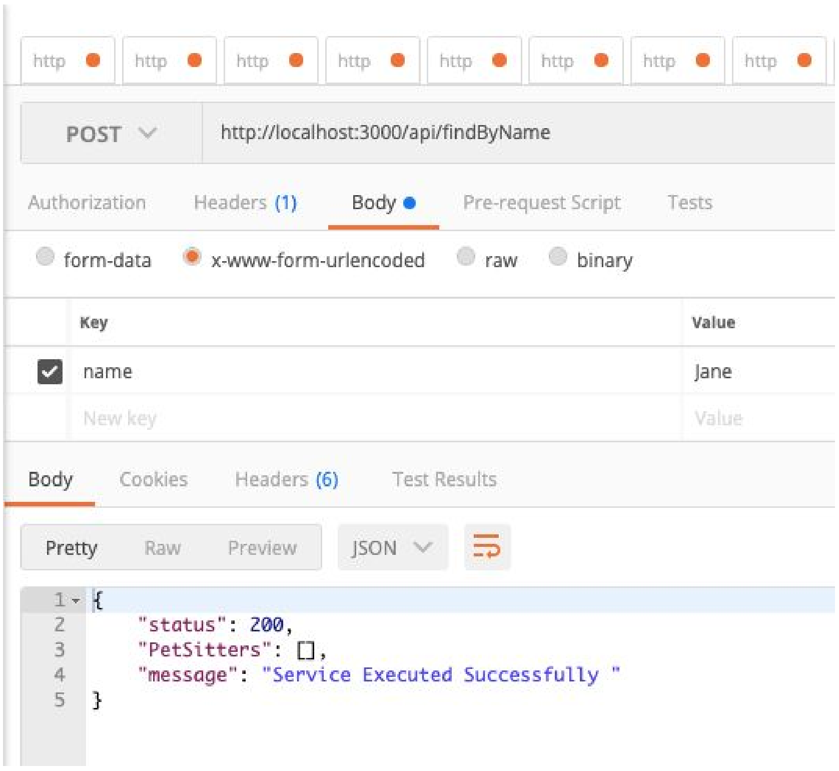
form-data [✓ x-www-form-urlencoded] raw binary

Key	Value
✓ category	dog
New key	Value

```json
{
    "status": 200,
    "PetSitters": [
        {
            "id": 1,
            "description": "description",
            "categs": [
                "dog"
            ],
            "createdAt": "2021-01-02T14:31:48.714Z",
            "updatedAt": "2021-01-02T14:31:48.714Z",
            "user_id": 2,
            "User": {
                "id": 2,
                "username": "sitter1",
                "password": "Imthesitter",
                "email": "sitter1@gmail.com",
                "firstname": "John",
                "lastname": "Cena",
                "phone": "0123456789",
                "city": "This",
                "country": "That",
                "role": "petsitter",
                "flag": false,
                "createdAt": "2021-01-02T14:31:48.709Z",
                "updatedAt": "2021-01-02T14:31:48.709Z"
            }
        }
    ],
    "message": "Service Executed Successfully "
```

Searching pet sitter by their name (part of it: either first or last)

Here we get an empty array because none of the pet sitters are called ‘Jane’



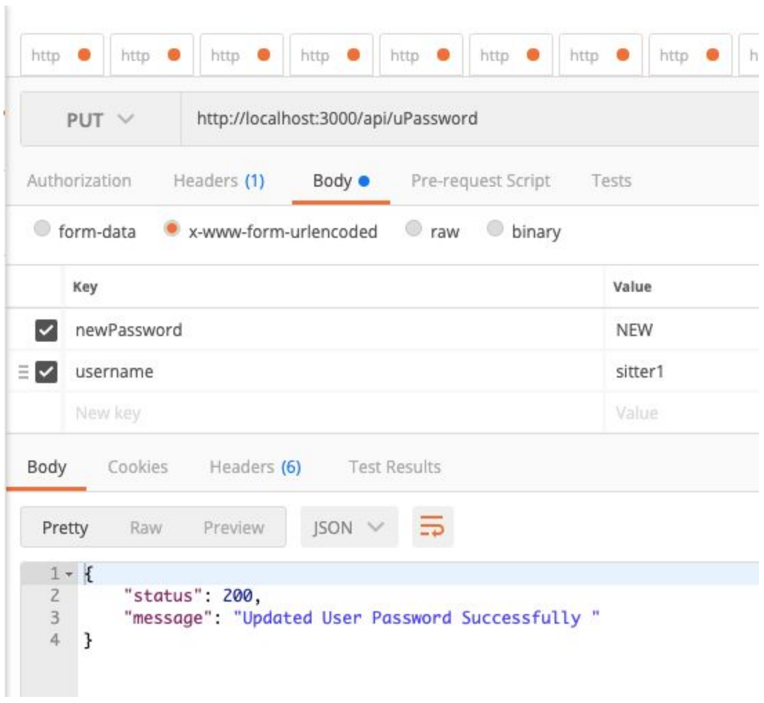
POST http://localhost:3000/api/findByName

Body: ○ form-data ● x-www-form-urlencoded ○ raw ○ binary

	Key	Value
✓	name	Jane
	New key	Value

```
{
    "status": 200,
    "PetSitters": [],
    "message": "Service Executed Successfully "
}
```

Update User Password

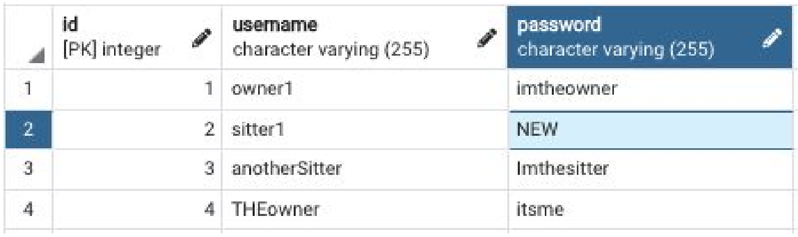


PUT http://localhost:3000/api/uPassword

Body: ○ form-data ● x-www-form-urlencoded ○ raw ○ binary

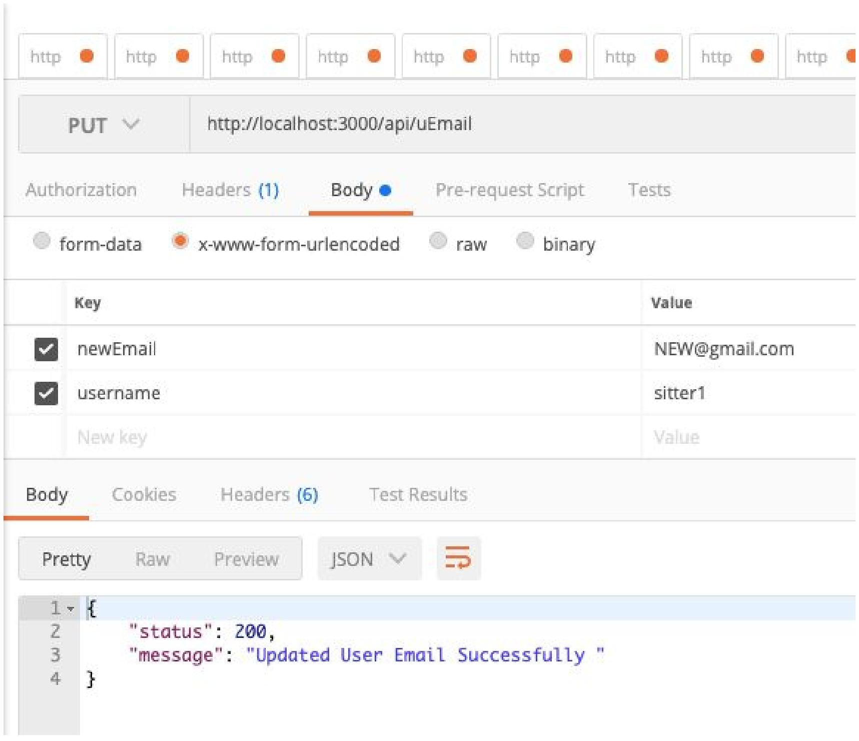
	Key	Value
✓	newPassword	NEW
✓	username	sitter1
	New key	Value

```
{
    "status": 200,
    "message": "Updated User Password Successfully "
}
```



	id [PK] integer	username character varying (255)	password character varying (255)
1	1	owner1	imtheowner
2	2	sitter1	NEW
3	3	anotherSitter	Imthesitter
4	4	THEowner	itsme

Update User Email

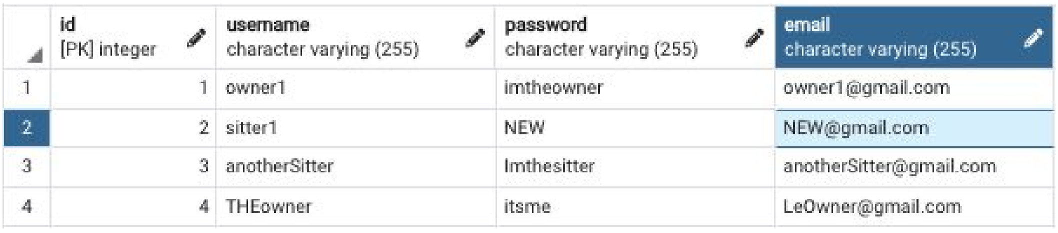


PUT http://localhost:3000/api/uEmail

Key	Value
✓ newEmail	NEW@gmail.com
✓ username	sitter1
New key	Value

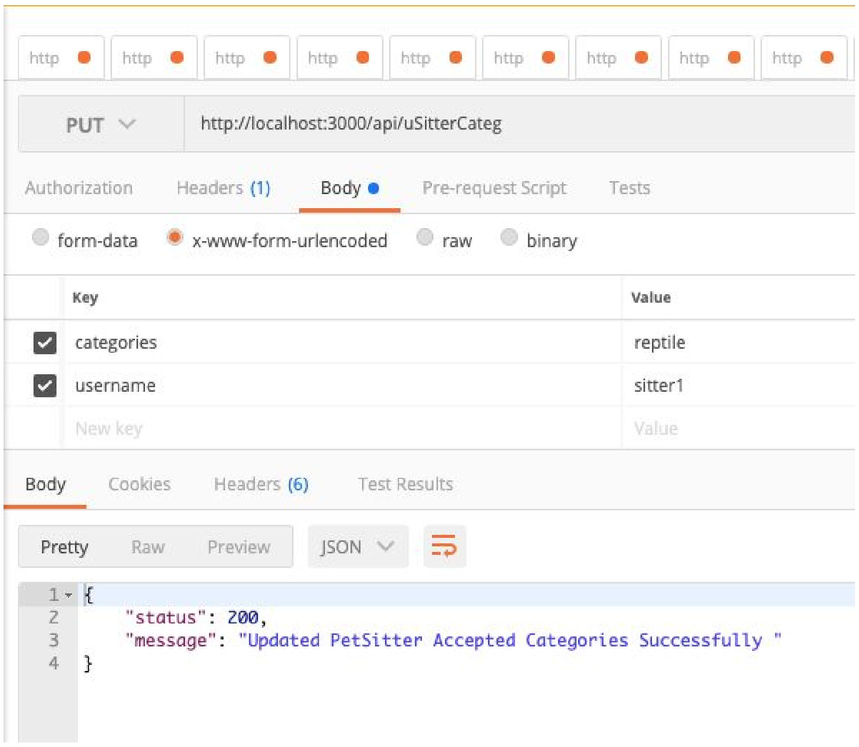
```
{
    "status": 200,
    "message": "Updated User Email Successfully "
}
```

l



	id [PK] integer	username character varying (255)	password character varying (255)	email character varying (255)
1	1	owner1	imtheowner	owner1@gmail.com
2	2	sitter1	NEW	NEW@gmail.com
3	3	anotherSitter	Imthesitter	anotherSitter@gmail.com
4	4	THEowner	itsme	LeOwner@gmail.com

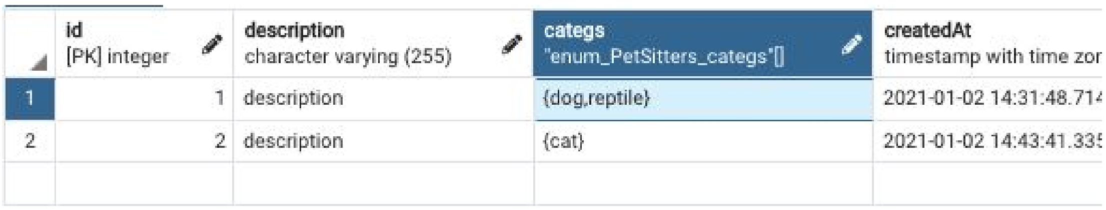
Update Pet Sitter Accepted pet Categories



PUT http://localhost:3000/api/uSitterCateg

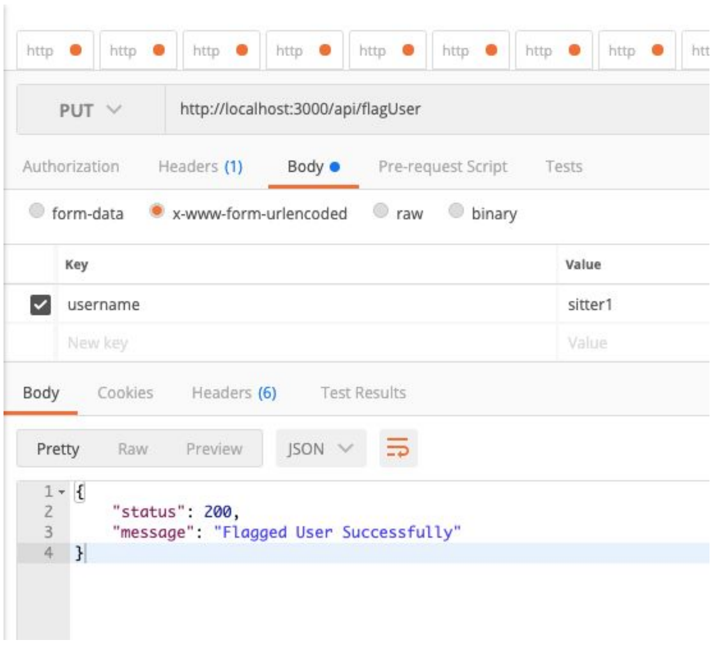
Key	Value
categories	reptile
username	sitter1
New key	Value

```
{
    "status": 200,
    "message": "Updated PetSitter Accepted Categories Successfully "
}
```



	id [PK] integer	description character varying (255)	categs "enum_PetSitters_categs"[]	createdAt timestamp with time zor
1	1	description	{dog,reptile}	2021-01-02 14:31:48.71
2	2	description	{cat}	2021-01-02 14:43:41.33

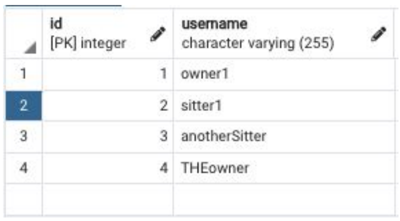
Flag a User



PUT http://localhost:3000/api/flagUser

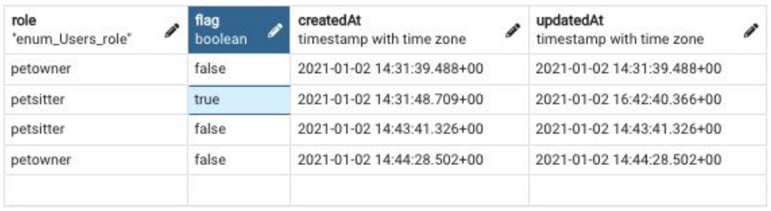
Key	Value
username	sitter1
New key	Value

```
{
    "status": 200,
    "message": "Flagged User Successfully"
}
```



	id [PK] integer	username character varying (255)
1	1	owner1
2	2	sitter1
3	3	anotherSitter
4	4	THEowner

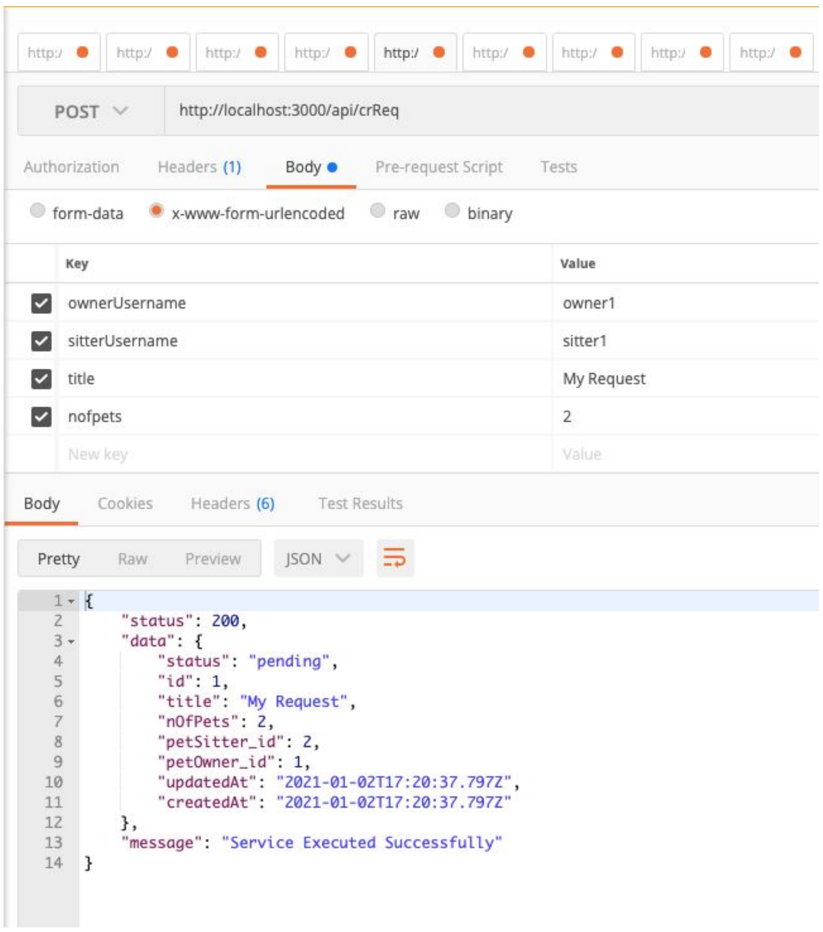
…



role "enum_Users_role"	flag boolean	createdAt timestamp with time zone	updatedAt timestamp with time zone
petowner	false	2021-01-02 14:31:39.488+00	2021-01-02 14:31:39.488+00
petsitter	true	2021-01-02 14:31:48.709+00	2021-01-02 16:42:40.366+00
petsitter	false	2021-01-02 14:43:41.326+00	2021-01-02 14:43:41.326+00
petowner	false	2021-01-02 14:44:28.502+00	2021-01-02 14:44:28.502+00

Request Service Testing

Making a Request

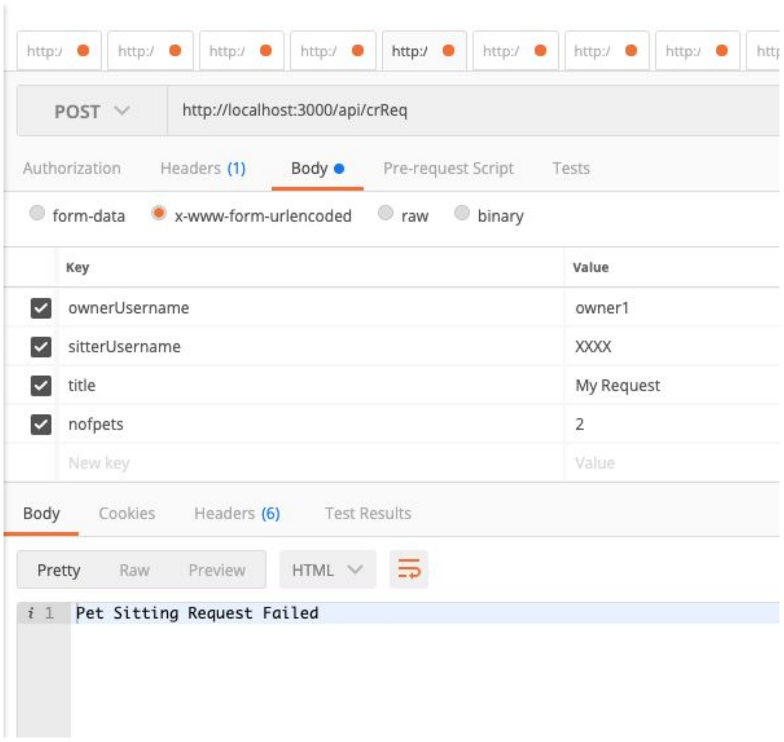


POST http://localhost:3000/api/crReq

Key	Value
ownerUsername	owner1
sitterUsername	sitter1
title	My Request
nofpets	2

```
{
    "status": 200,
    "data": {
        "status": "pending",
        "id": 1,
        "title": "My Request",
        "nOfPets": 2,
        "petSitter_id": 2,
        "petOwner_id": 1,
        "updatedAt": "2021-01-02T17:20:37.797Z",
        "createdAt": "2021-01-02T17:20:37.797Z"
    },
    "message": "Service Executed Successfully"
}
```

Making a request with wrong sender/receiver



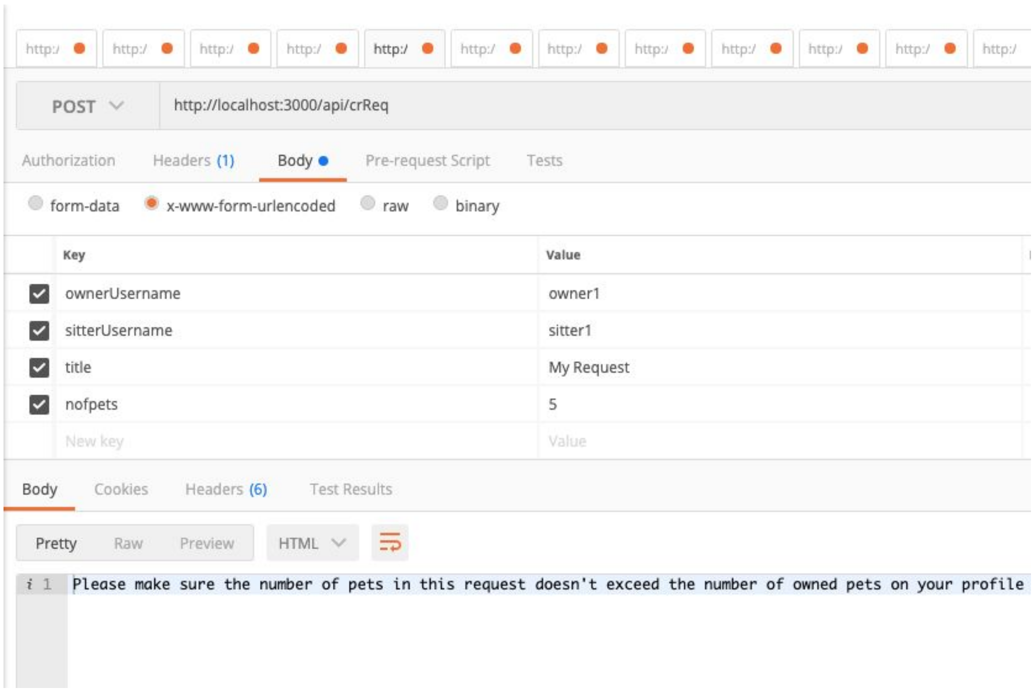
POST http://localhost:3000/api/crReq

Body: x-www-form-urlencoded

Key	Value
ownerUsername	owner1
sitterUsername	XXXX
title	My Request
nofpets	2

```
Pet Sitting Request Failed
```

Making a request with a number of pets that is more than specified in the owner's profile



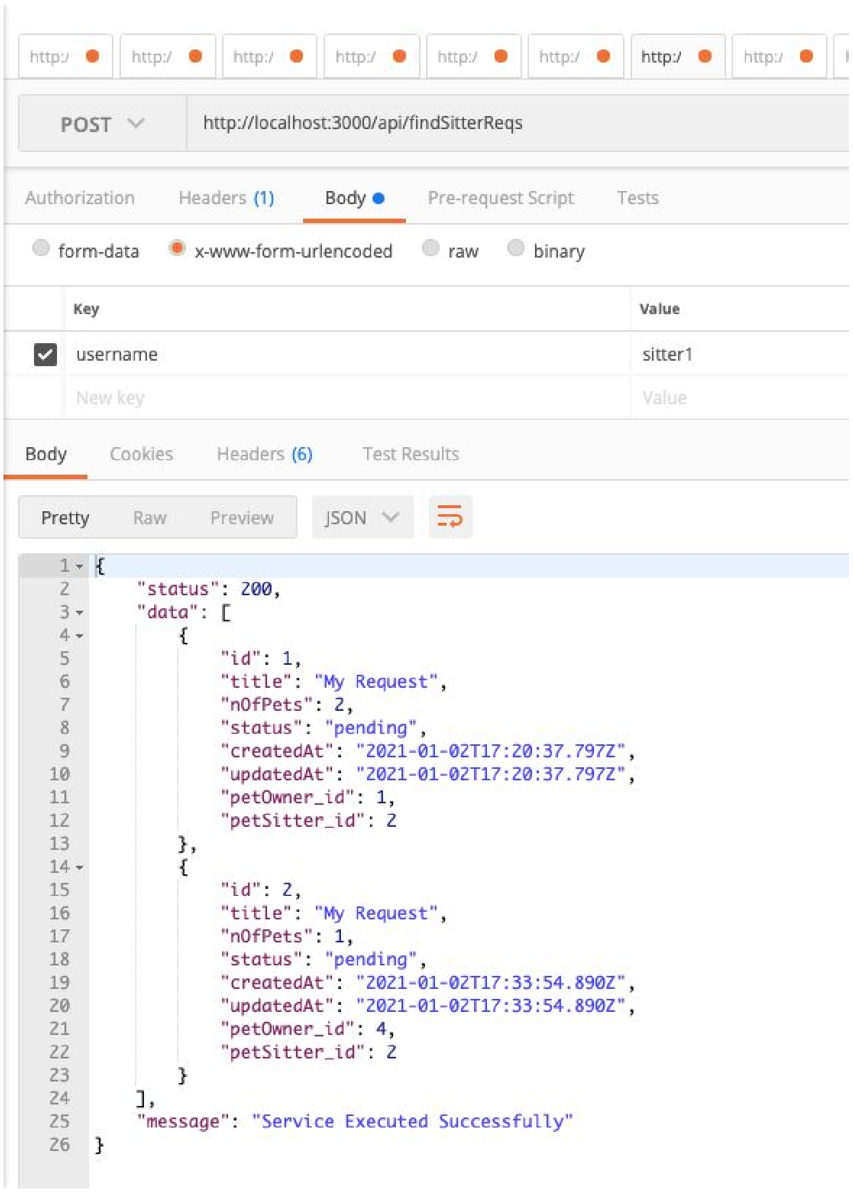
POST http://localhost:3000/api/crReq

Body: x-www-form-urlencoded

Key	Value
ownerUsername	owner1
sitterUsername	sitter1
title	My Request
nofpets	5

```
Please make sure the number of pets in this request doesn't exceed the number of owned pets on your profile
```

Pet sitter looking for all their received requests



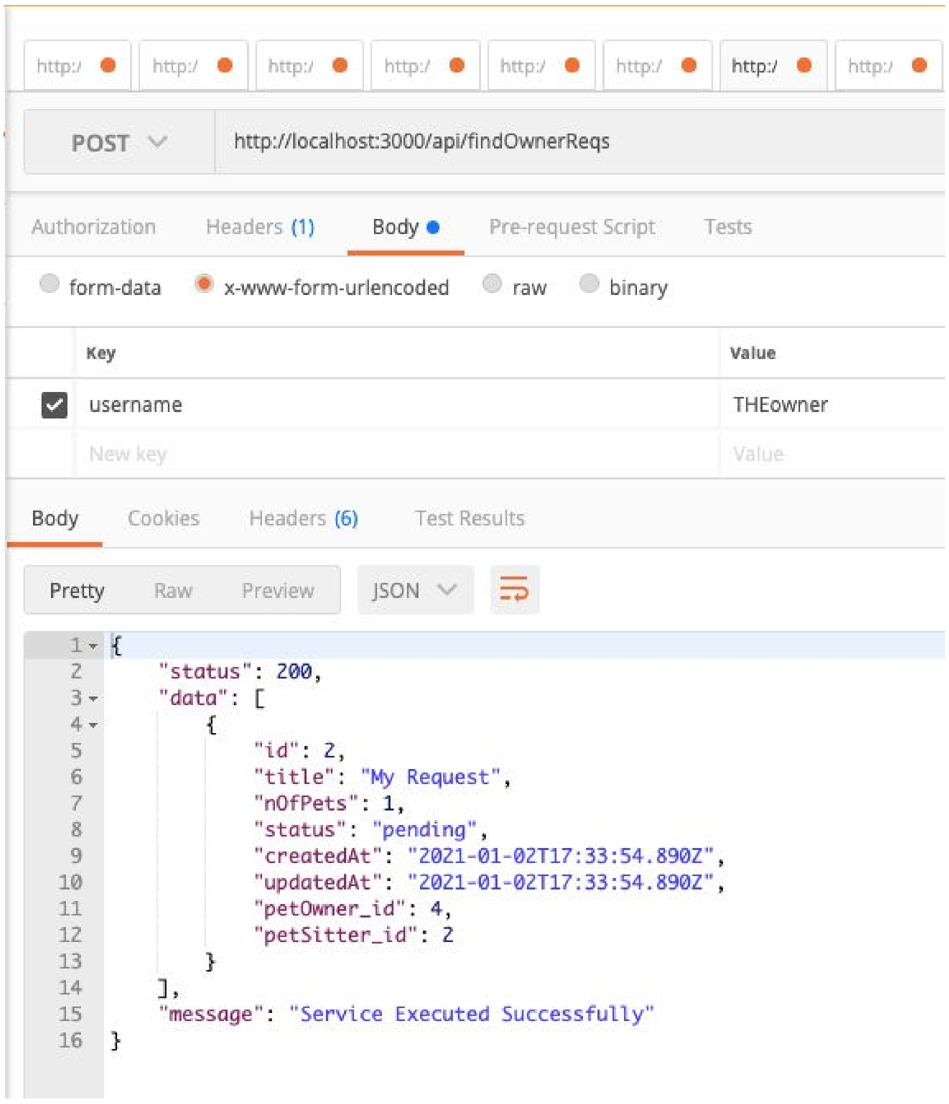
POST http://localhost:3000/api/findSitterReqs

Body: x-www-form-urlencoded

Key	Value
username	sitter1

```
{
    "status": 200,
    "data": [
        {
            "id": 1,
            "title": "My Request",
            "nOfPets": 2,
            "status": "pending",
            "createdAt": "2021-01-02T17:20:37.797Z",
            "updatedAt": "2021-01-02T17:20:37.797Z",
            "petOwner_id": 1,
            "petSitter_id": 2
        },
        {
            "id": 2,
            "title": "My Request",
            "nOfPets": 1,
            "status": "pending",
            "createdAt": "2021-01-02T17:33:54.890Z",
            "updatedAt": "2021-01-02T17:33:54.890Z",
            "petOwner_id": 4,
            "petSitter_id": 2
        }
    ],
    "message": "Service Executed Successfully"
}
```

Pet owner searching for all their sent requests



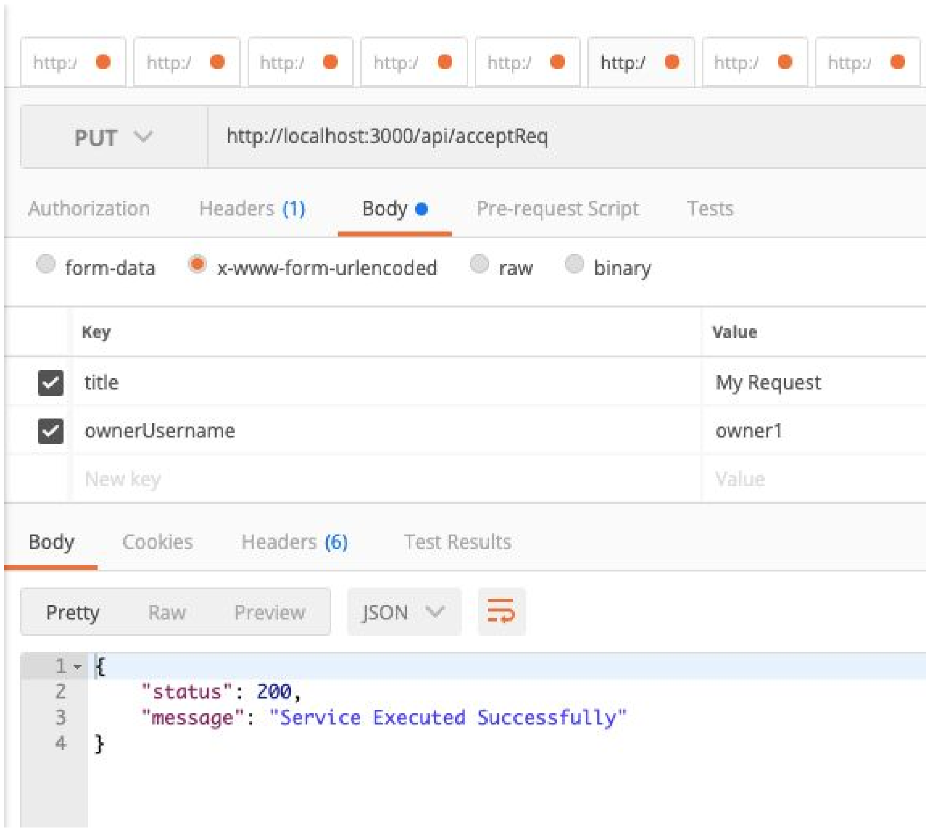
POST http://localhost:3000/api/findOwnerReqs

Body: x-www-form-urlencoded

Key	Value
username	THEowner

```
{
    "status": 200,
    "data": [
        {
            "id": 2,
            "title": "My Request",
            "nOfPets": 1,
            "status": "pending",
            "createdAt": "2021-01-02T17:33:54.890Z",
            "updatedAt": "2021-01-02T17:33:54.890Z",
            "petOwner_id": 4,
            "petSitter_id": 2
        }
    ],
    "message": "Service Executed Successfully"
}
```

Pet Sitter: Accepting a request

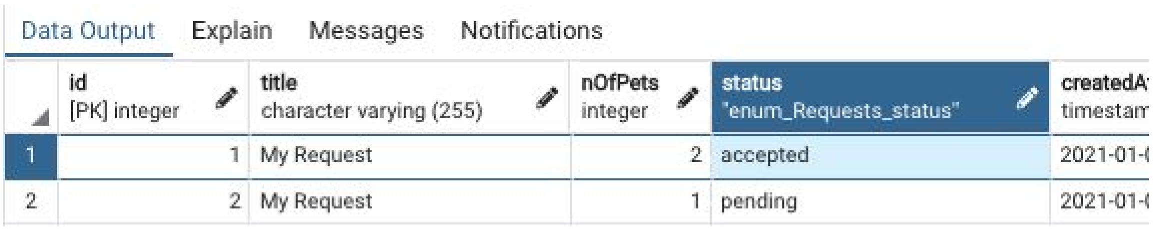


PUT http://localhost:3000/api/acceptReq

● x-www-form-urlencoded

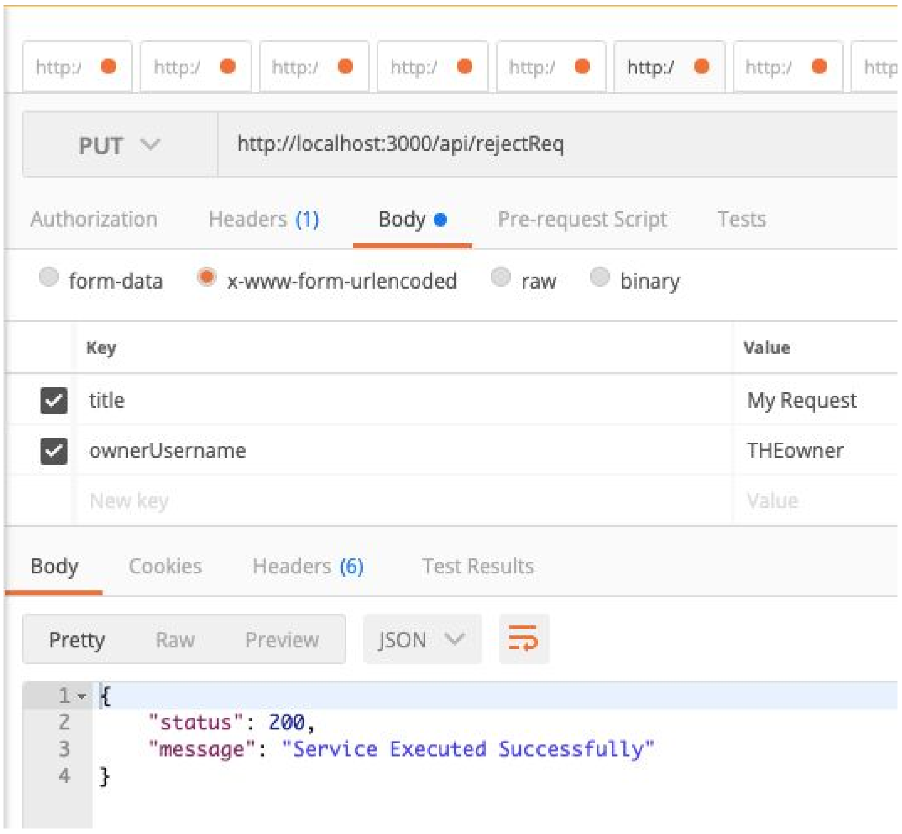
	Key	Value
✓	title	My Request
✓	ownerUsername	owner1
	New key	Value

```
{
    "status": 200,
    "message": "Service Executed Successfully"
}
```



	id [PK] integer	title character varying (255)	nOfPets integer	status "enum_Requests_status"	createdA timestam
1	1	My Request	2	accepted	2021-01-
2	2	My Request	1	pending	2021-01-

Pet Sitter: Reject a Request

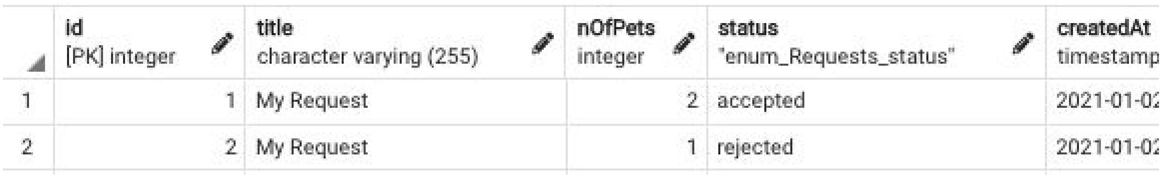


PUT http://localhost:3000/api/rejectReq

Body: x-www-form-urlencoded

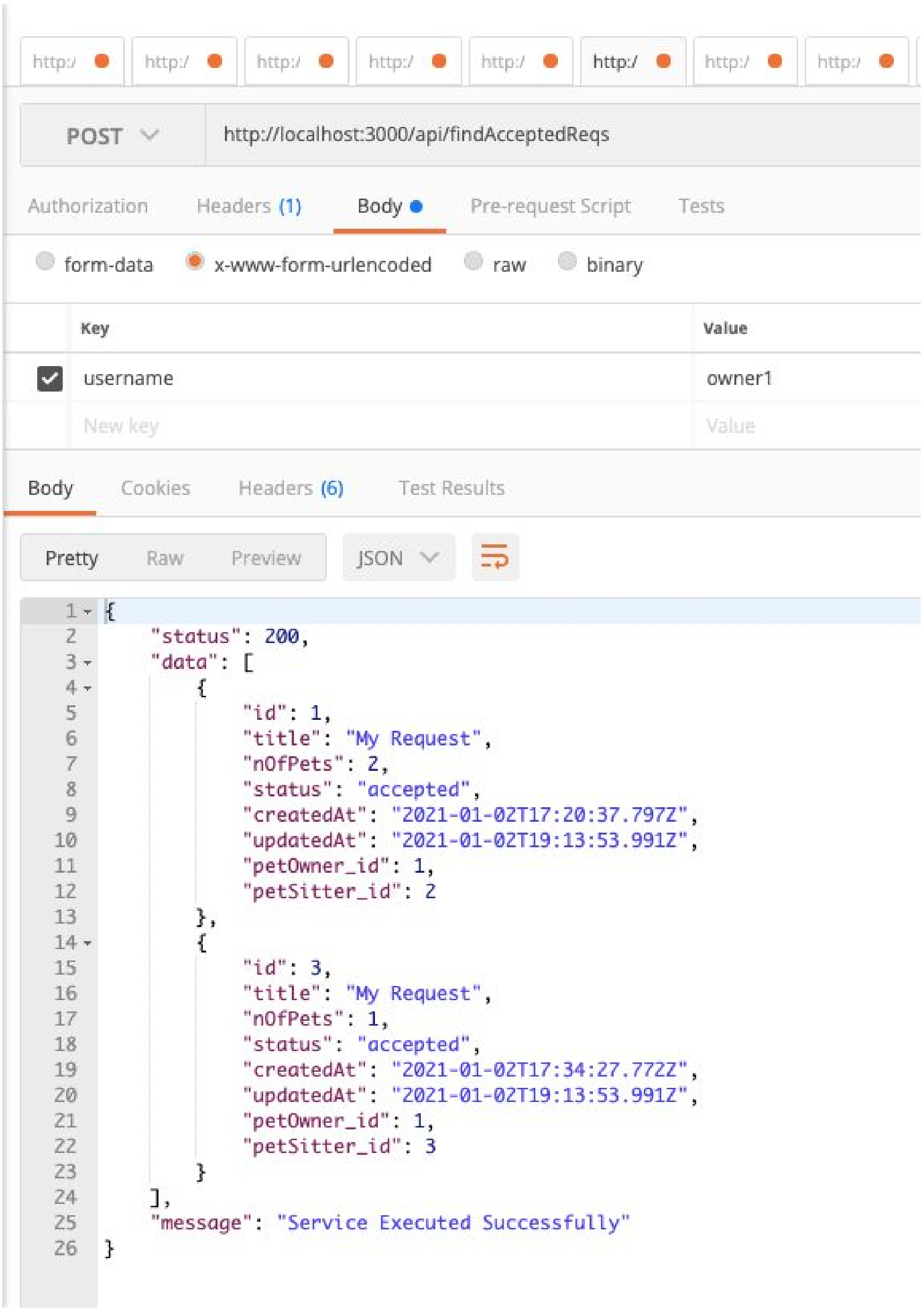
Key	Value
title	My Request
ownerUsername	THEowner

```
{
    "status": 200,
    "message": "Service Executed Successfully"
}
```



	id [PK] integer	title character varying (255)	nOfPets integer	status "enum_Requests_status"	createdAt timestamp
1	1	My Request	2	accepted	2021-01-0
2	2	My Request	1	rejected	2021-01-0

Find Accepted Requests



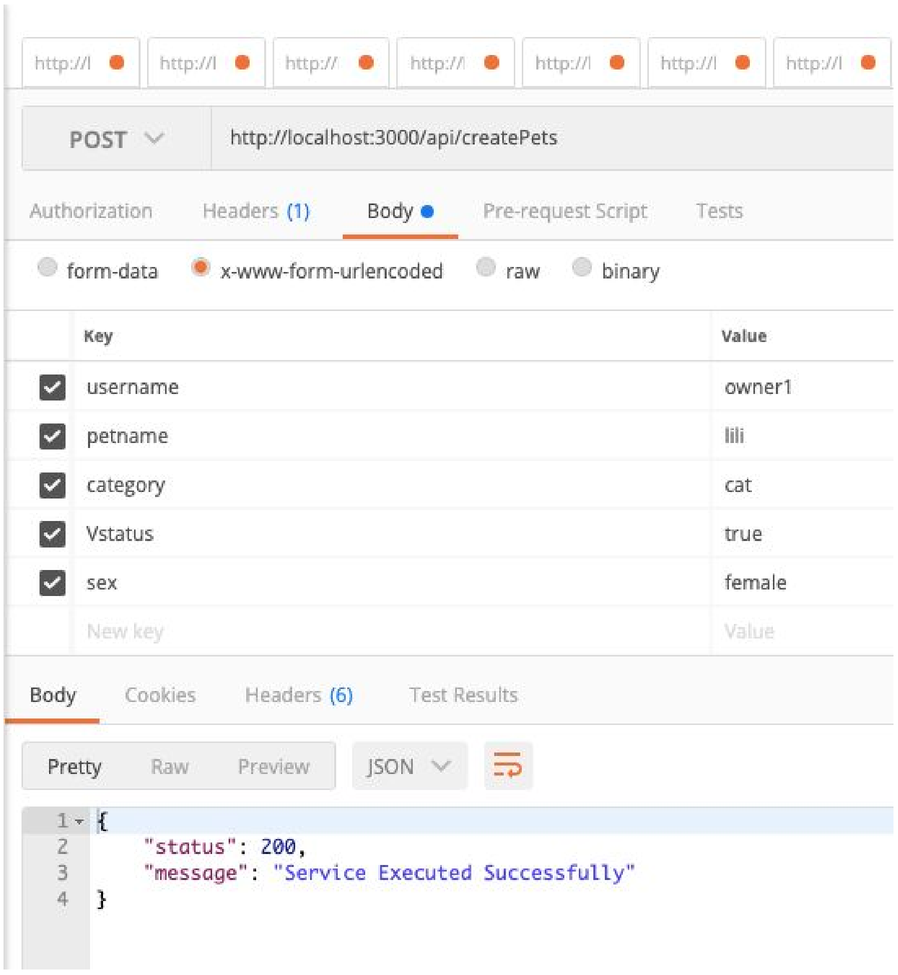
POST http://localhost:3000/api/findAcceptedReqs

Key	Value
username	owner1

```
{
    "status": 200,
    "data": [
        {
            "id": 1,
            "title": "My Request",
            "nOfPets": 2,
            "status": "accepted",
            "createdAt": "2021-01-02T17:20:37.797Z",
            "updatedAt": "2021-01-02T19:13:53.991Z",
            "petOwner_id": 1,
            "petSitter_id": 2
        },
        {
            "id": 3,
            "title": "My Request",
            "nOfPets": 1,
            "status": "accepted",
            "createdAt": "2021-01-02T17:34:27.772Z",
            "updatedAt": "2021-01-02T19:13:53.991Z",
            "petOwner_id": 1,
            "petSitter_id": 3
        }
    ],
    "message": "Service Executed Successfully"
}
```

Pet Service Testing

Create Pet

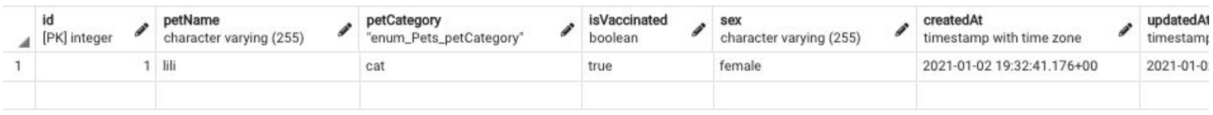


POST http://localhost:3000/api/createPets

form-data [✓ x-www-form-urlencoded] raw binary

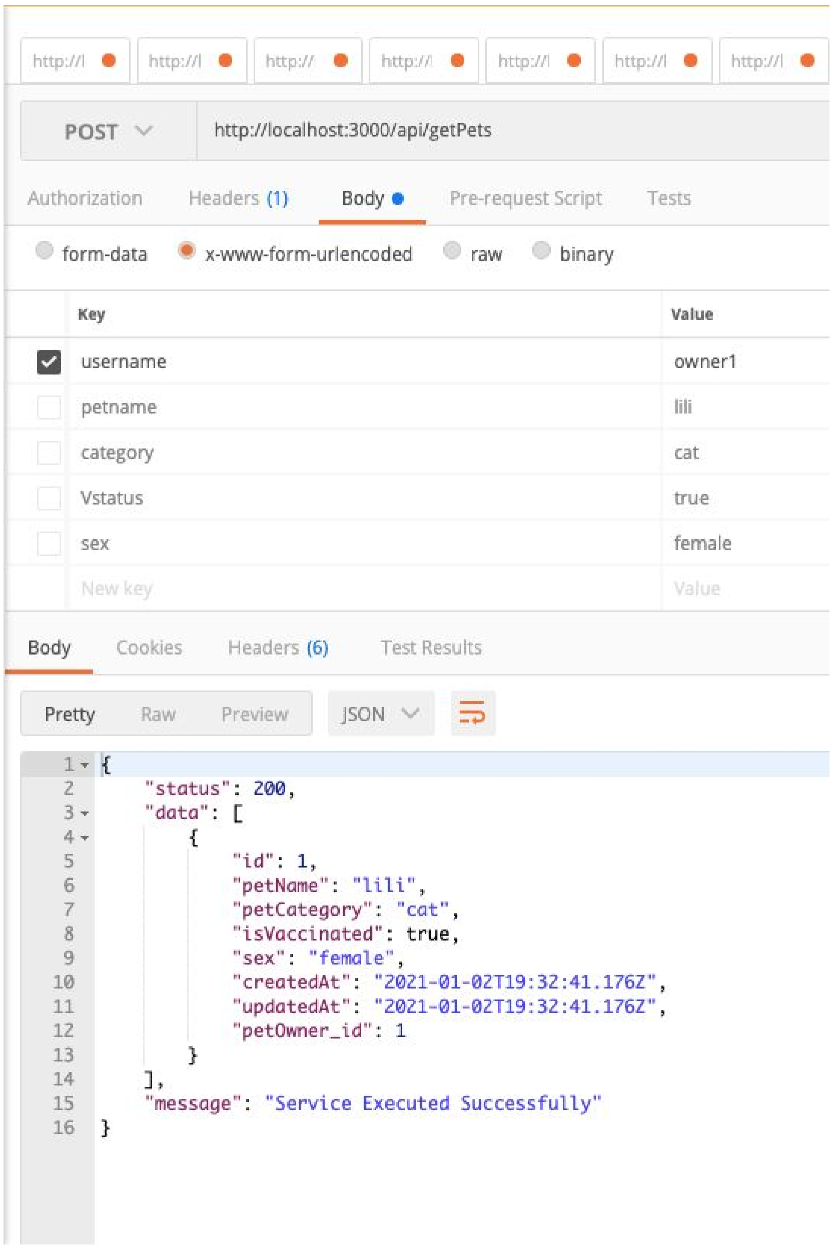
	Key	Value
✓	username	owner1
✓	petname	lili
✓	category	cat
✓	Vstatus	true
✓	sex	female
	New key	Value

```
{
    "status": 200,
    "message": "Service Executed Successfully"
}
```



	id [PK] integer	petName character varying (255)	petCategory "enum_Pets_petCategory"	isVaccinated boolean	sex character varying (255)	createdAt timestamp with time zone	updatedAt timestamp
1	1	lili	cat	true	female	2021-01-02 19:32:41.176+00	2021-01-0

Retrieve Pets

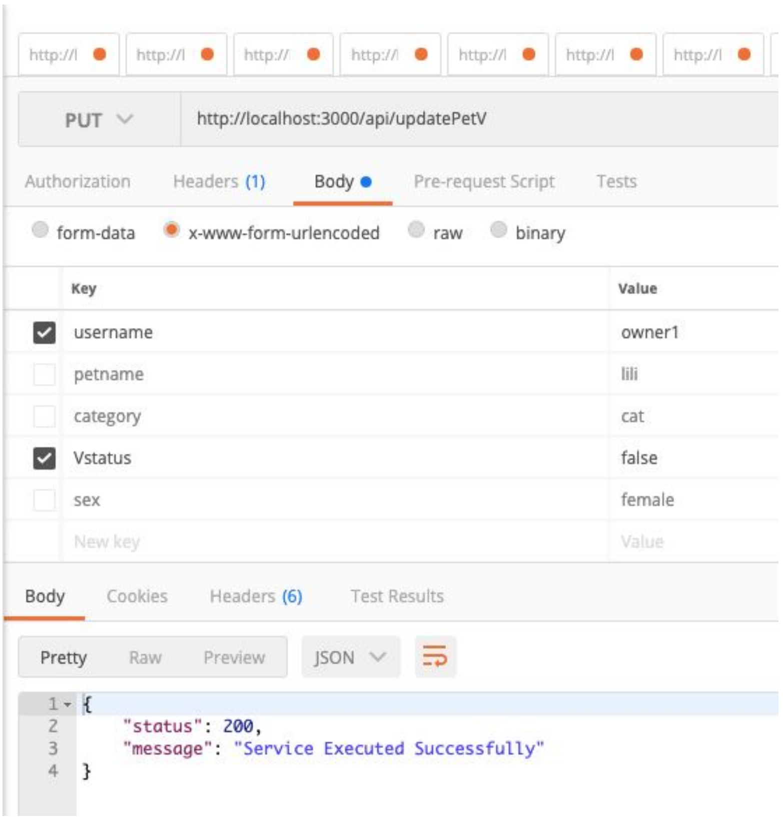

POST http://localhost:3000/api/getPets

Body: x-www-form-urlencoded

	Key	Value
✓	username	owner1
	petname	lili
	category	cat
	Vstatus	true
	sex	female

```
{
    "status": 200,
    "data": [
        {
            "id": 1,
            "petName": "lili",
            "petCategory": "cat",
            "isVaccinated": true,
            "sex": "female",
            "createdAt": "2021-01-02T19:32:41.176Z",
            "updatedAt": "2021-01-02T19:32:41.176Z",
            "petOwner_id": 1
        }
    ],
    "message": "Service Executed Successfully"
}
```

Update Vaccination Status

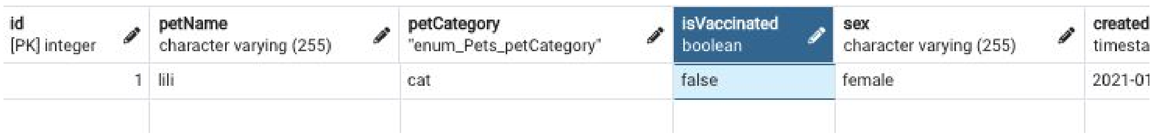


PUT http://localhost:3000/api/updatePetV

Body: form-data [✓ x-www-form-urlencoded] raw binary

	Key	Value
✓	username	owner1
	petname	lili
	category	cat
✓	Vstatus	false
	sex	female
	New key	Value

```
{
    "status": 200,
    "message": "Service Executed Successfully"
}
```

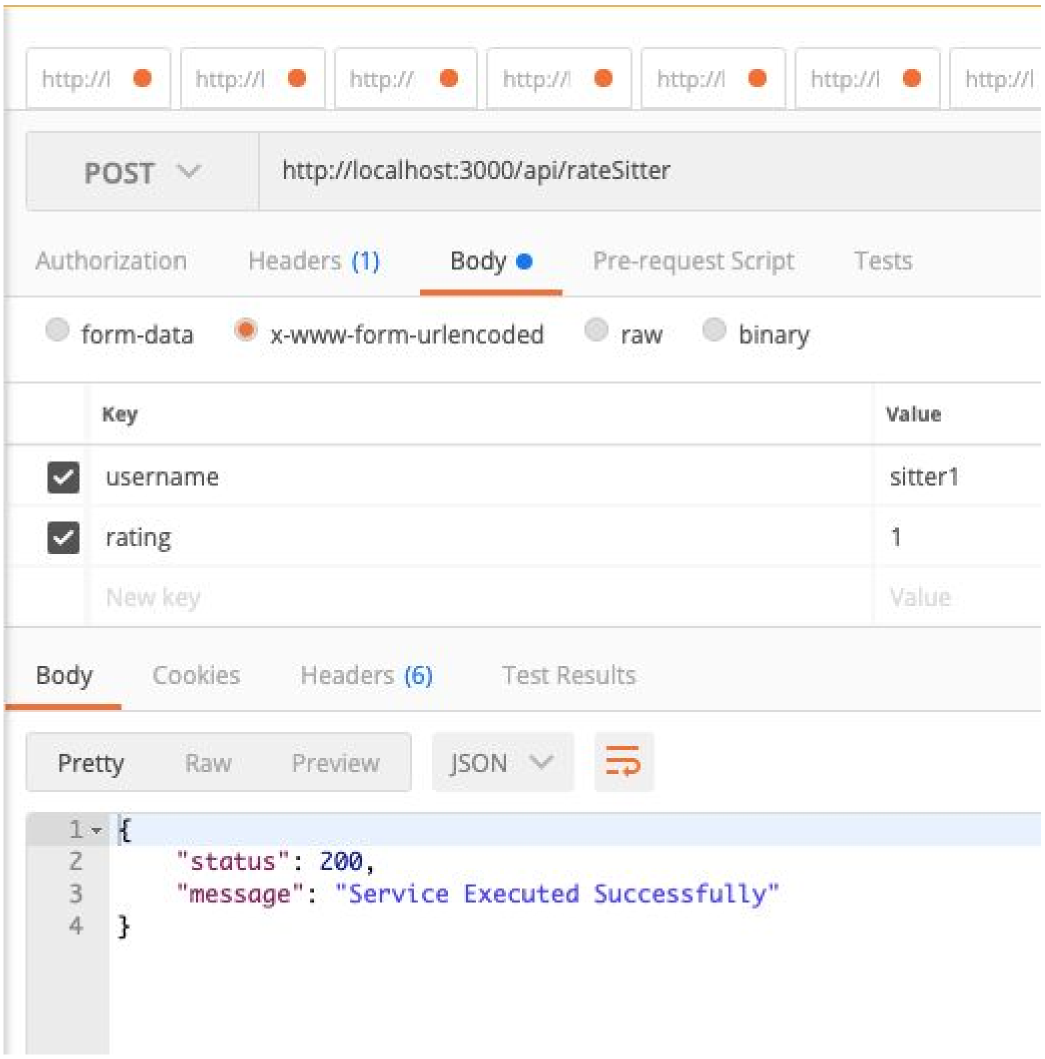


id [PK] integer	petName character varying (255)	petCategory "enum_Pets_petCategory"	isVaccinated boolean	sex character varying (255)	created timesta
1	lili	cat	false	female	2021-01

Rating Service Test

When a pet owner tries to rate a pet sitter, they choose a number from 1 - 5 which is appended to the array of ratings the sitter had collected, so that an average can be calculated.

In this example, the last rating the pet owner gave was a “1” and it got appended to the array, updating the average rating as well.

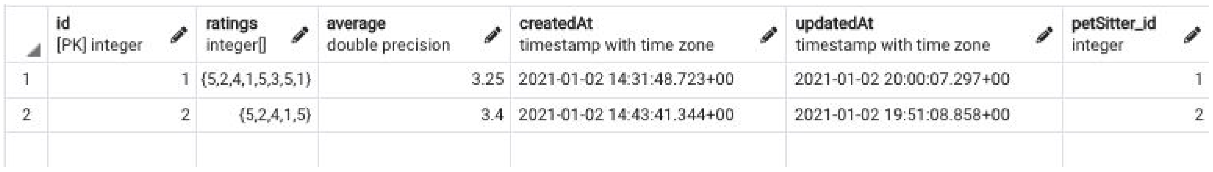


POST http://localhost:3000/api/rateSitter

Body: x-www-form-urlencoded

	Key	Value
✓	username	sitter1
✓	rating	1
	New key	Value

```
{
    "status": 200,
    "message": "Service Executed Successfully"
}
```



	id [PK] integer	ratings integer[]	average double precision	createdAt timestamp with time zone	updatedAt timestamp with time zone	petSitter_id integer
1	1	{5,2,4,1,5,3,5,1}	3.25	2021-01-02 14:31:48.723+00	2021-01-02 20:00:07.297+00	1
2	2	{5,2,4,1,5}	3.4	2021-01-02 14:43:41.344+00	2021-01-02 19:51:08.858+00	2