

PetFinder - PetAdaptor

User Roles:

-adopter
-owner

Use Case:

- Adopter
 1. Basic search functions(pet.find: animal, breed, size, sex, location, age, ...)
 2. Front page with randomly selected pets(pet.getRandom)
 3. Shelter List --> pet lists by shelter
 4. Visualization with GoogleMap
- shelter/owner
 1. Login and edit owner profile
 2. View posted pet list
 3. Post a pet info
 4. Edit a pet info
 5. Delete a pet info
 6. Browse message list
 7. View detailed message
 8. Send message
- Admin
 1. Login admin workspace
 2. Browse pets list
 3. View detailed pet info
 4. Edit detailed pet info
 5. Delete detailed pet info

Database:

User

```
- _id: ObjectId;
- email: string
- password: string
- isOwner: boolean
- isAdopter: boolean
- lastName: string;
- firstName: string;
- gender: string;
- contact {
    phone: string;
    state: string;
    address1: string;
    address2: string;
    email: string;
    city: string;
    zip: string;
}
- sentmessage [
    {
        _id: string;
        messagecontent: string;
        userTo: string; (存userid)
        postTime: date;
    }
]
- receivemessage [
    {
        _id: string;
        messageContent: string;
        userFrom: string; (存userid)
        postTime: date;
    }
]
- bookmarkList: [
    {
        tag: string
        url: string
    }
]
```

```
- petList: [
  {
    _id: string
    name: string;
    type: string;
    primarybreed: string;
    age: string;
    sex: string;
    description: string;
    imagelink: Array<string>;
    postTime: date;
  }
]
```

Backend API

1. Login

```
- POST
- URL http://localhost/user/login
- body: { 'email': '###', 'password': '###', 'isowner' : boolean }
- response:
  {
    Userid: string,
    isowner: boolean
  }
```

Success:

{"status": true, "isowner": boolean, "_id": id, "email":String, "token": token String}

```
{
  "status": true,
  "isowner": true,
  "_id": "5ad50f7546c2a9597d48dd2e",
  "email": "a@a.com",
  "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXV
    MiLCJlbWFPbCI6InMuaG9lZmluZ08tY2hzaS5jb
    iOiI0YWQ1MGY2NTI2YTJhODU5N2Q0NWlkM2U1LC
    OjE4LjU5N1oiFV0sInJlY2VpdmVtZXNzYWdlIjpw
    jVhZDUwZjc1NDZjMmE5NTk3ZDQ4ZGQyZCI6Im5h
    9raW5nIGZvciBhbiBhY3RpdmUgZmFtaWw5LCBwci
    1OC8xLz9idXN0PTE0OTc0ODQ0Mzkmd21kdGg9NT
    Mzg0NDk3NTg0Mi8_YnVzdD0xNTA4OTYzNzkzJnd
    0"
```

Failed:

`{"status": false, "errmsg": errmsg String}`

```
{
  "status": false,
  "msg": "Wrong password"
}
```

2. Register

- POST
- URL <http://localhost/user/register>
- body: { 'email': '###', 'password': '###', 'isowner' : boolean }
- response:
{
 status: boolean,
 msg: String
}

Succeed:

`{"status": true, "msg": msg String}`

```
{
  "status": true,
  "msg": "Account registered"
}
```

Failed:

`{"status": false, "msg": msg String}`

```
{
  "status": false,
  "msg": "Email already exists"
}
```

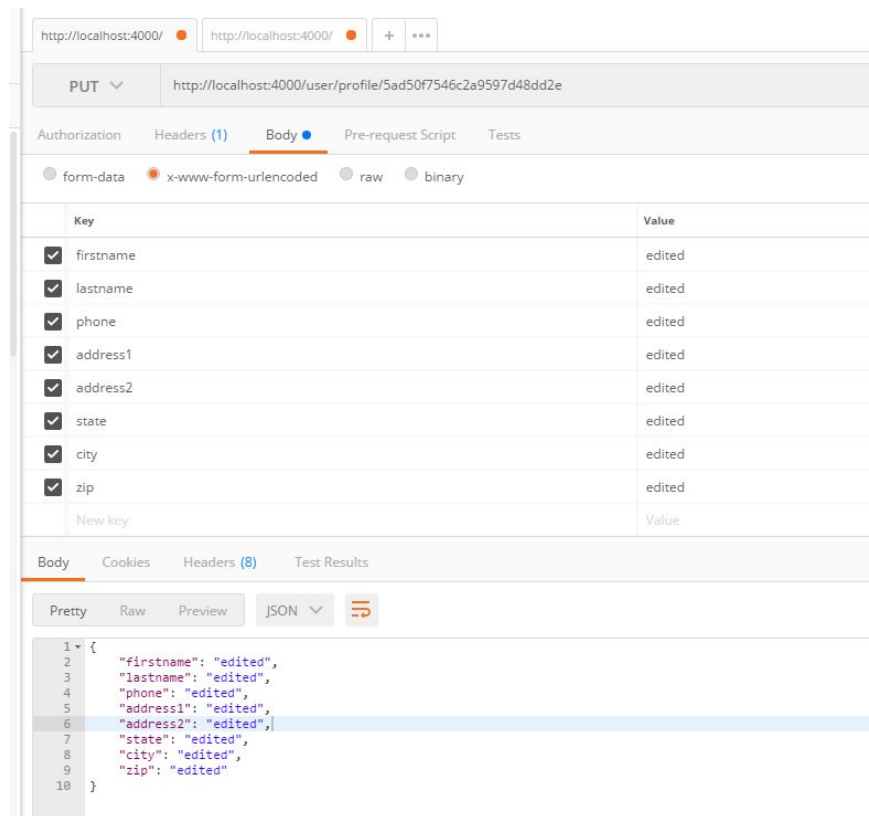
3. Edit User Profile

- PUT
- URL <http://localhost/user/profile/{userId}>
- Body: {
 Firstname: string
 Lastname: string,
 Phone: string,
 Address1: string,
 Address2: string,
 State: string,
 City: string,
 Zip: string
 }
- Response: {

```

Firstname: string
Lastname: string,
Phone: string,
Address1: string,
Address2: string,
State: string,
City: string,
Zip: string
}

```



4. View User Profile

- GET
- URL <http://localhost/user/profile/{userId}>
- Response: {

```

Firstname: string,
Lastname: string,
Phone: string,
Address1: string,
Address2: string,
State: string,
City: string,

```

Zip: string
}

```
1 {  
2   "firstname": "tom",  
3   "lastname": "jack",  
4   "phone": "32323",  
5   "address1": "Box 221",  
6   "address2": "506 Park",  
7   "state": "IA",  
8   "city": "Shenandoah",  
9   "zip": "51601"  
10 }
```

4.1 Get ownerID by petID

- GET
- URL <http://localhost/pet/{petid}/owner>
- Response: {
 _id: string (success: '###' / fail: 'Owner Not Found')
}

4.2 Get owner location by petID

- Get
- URL <http://localhost/pet/{petid}/ownerlocation>
- Response:{
 address1: string;
 address2: string;
 city: string;
 state: string;
 zip: string;
}

4.3 Get user email by userid

- GET
- URL <http://localhost/user/email/{userId}>
- Response: {
 email: '###' / 'not_existed_user'
}

5. Post Pet

- POST
- URL: <http://localhost/user/{userId}/pet>
- **body: { 'name': '##', 'type': '##', 'primarybreed': '###', 'age': '##', 'sex': '##', 'description': '##', 'image': '##' }**
- Response: {
 status: boolean

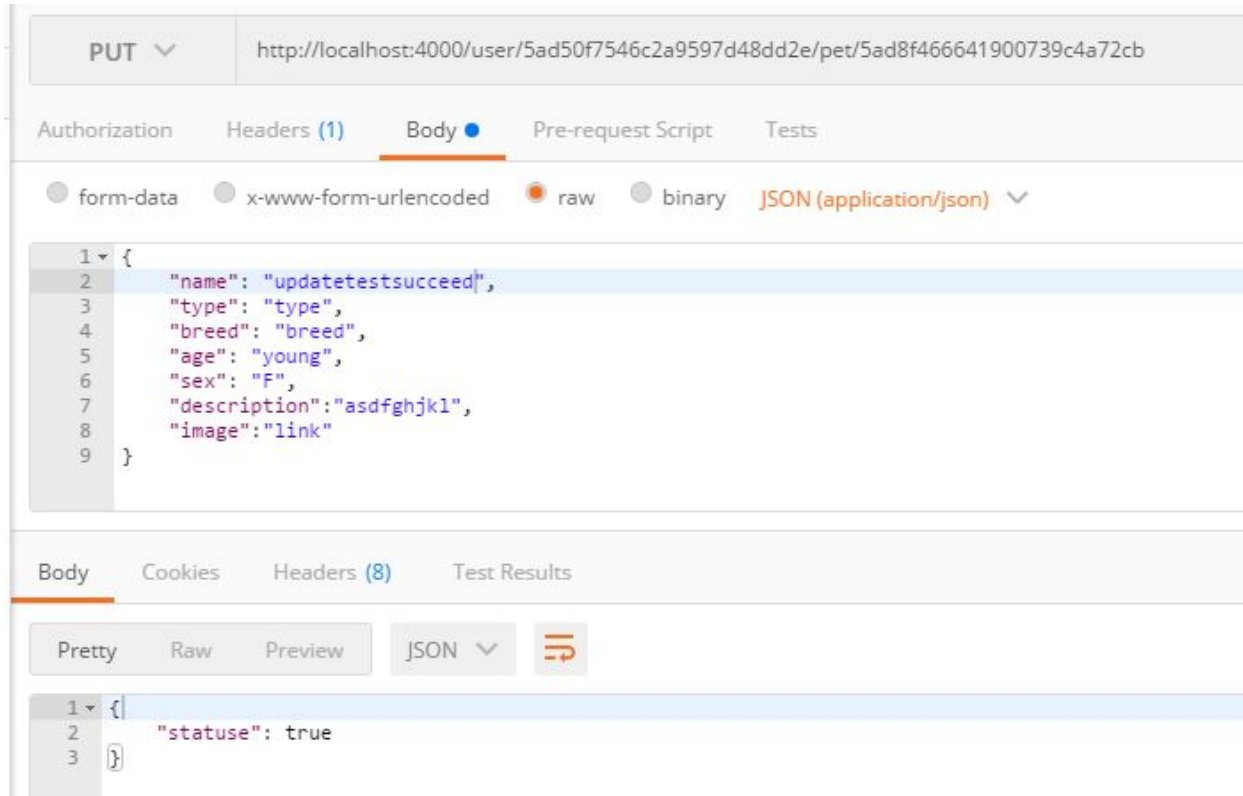
}

The screenshot shows a REST client interface with a POST request to `http://localhost:4000/user/5ad50f7546c2a9597d48dd2e/pet`. The request body is configured as `x-www-form-urlencoded`. The form data is as follows:

Key	Value
<input checked="" type="checkbox"/> name	newcat
<input checked="" type="checkbox"/> type	dog
<input checked="" type="checkbox"/> breed	breed
<input checked="" type="checkbox"/> age	OLD
<input checked="" type="checkbox"/> sex	M
<input checked="" type="checkbox"/> description	asdfghjk
<input checked="" type="checkbox"/> image	link
New key	Value

6. Edit Pet

- PUT
- URL: <http://localhost/user/{userId}/pet/{petid}>
- **body: { 'name': '##', 'type': '##', 'primarybreed': '###', 'age': '##', 'sex': '##', 'description': '##', 'image': '##' }**
- Response: {
 status: boolean
}



6. Get Pets by userid (owner)

- GET
- URL: <http://localhost/user/{userId}/pet>
- Response: [

```
{
  _id: string
  name: string;
  type: string;
  primarybreed: string;
  age: string;
  sex: string;
  description: string;
  imagelink: Array<string>;
  postTime: date;
}
```

]

GET <http://localhost:4000/user/5ad50f7546c2a9597d48dd2e/pet>

Authorization Headers (1) Body Pre-request Script Tests

TYPE

Inherit auth from parent

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

This request is not inheriting any auth

Body Cookies Headers (9) Test Results

Pretty Raw Preview JSON

```

1 [
2   {
3     "_id": "5ad50f7546c2a9597d48dd2d",
4     "name": "Sheila",
5     "type": "Dog",
6     "primarybreed": "Husky",
7     "sex": "F",
8     "age": "Adult",
9     "description": "Sheila is a fun-loving high energy girl. She is looking for an active family, preferably with a",
10    "imagelink": [
11      "http://photos.petfinder.com/photos/pets/38449758/1/?bust=1497484439&width=500&-x.jpg",
12      "http://photos.petfinder.com/photos/pets/38449758/1/?bust=1497484439&width=300&-pn.jpg",
13      "http://photos.petfinder.com/photos/pets/38449758/2/?bust=1508963793&width=500&-x.jpg",
14      "http://photos.petfinder.com/photos/pets/38449758/2/?bust=1508963793&width=300&-pn.jpg",
15      "http://photos.petfinder.com/photos/pets/38449758/3/?bust=1508963794&width=500&-x.jpg",
16      "http://photos.petfinder.com/photos/pets/38449758/3/?bust=1508963794&width=300&-pn.jpg"
17    ],
18    "postTime": "2018-04-15T01:03:18.597Z"
19  },
20  {
21    "_id": "5ad50f7546c2a9597d48dd2f",
22    "name": "Sampson",
23    "type": "Dog",
24    "primarybreed": "Golden Retriever",
25    "sex": "M",
26    "age": "Baby",
27    "description": "One of Kimmy's puppies\n\nSampson gets along well with others. He is a little shy around strang",
28    "imagelink": [
29      "http://photos.petfinder.com/photos/pets/29682223/1/?bust=1495847928&width=500&-x.jpg",
30      "http://photos.petfinder.com/photos/pets/29682223/1/?bust=1495847928&width=300&-pn.jpg"
31    ],
32    "postTime": "2018-03-15T01:03:18.597Z"
33  },
34  ]

```

7. Search Pet 参数要跟后台数据完全一样，大小写敏感

GET

URL:

http://localhost:4000/pets/search?type=Dog&primarybreed=Husky&age=baby&sex=M&location=boston&sort=recently_added (sort=longest_available)

- Response: [

```

{
  _id: string
  name: string;
  type: string;
  primarybreed: string;

```

```
age: string;
sex: string;
description: string;
imagelink: Array<string>;
postTime: date;
}
]
```

7.1 Search Pet by keyword(in description and petname)

- GET
- URL: <http://localhost:4000/pets/searchbykeyword?keyword=Pebbles>
- Response: [

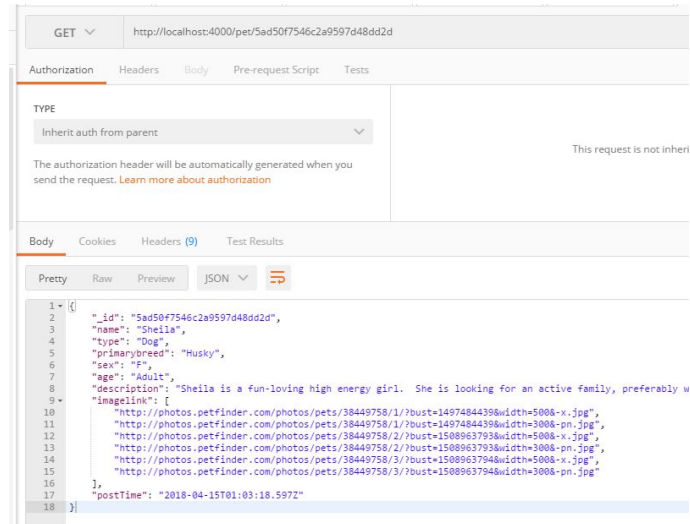
```
{
  _id: string
  name: string;
  type: string;
  primarybreed: string;
  age: string;
  sex: string;
  description: string;
  imagelink: Array<string>;
  postTime: date;
}
```

]

8. Get Certain Pet

- GET
- URL: <http://localhost:4000/pet/{petid}>
- Response:

```
{
  _id: string
  name: string;
  type: string;
  primarybreed: string;
  age: string;
  sex: string;
  description: string;
  imagelink: Array<string>;
  postTime: date;
}
```



10. Get Random Pets


- GET
- URL: <http://localhost:4000/pets/random>
- Response: [

```
{
  _id: string
  name: string;
  type: string;
  primarybreed: string;
  age: string;
  sex: string;
  description: string;
  imagelink: Array<string>;
  postTime: date;
}
]
```

9. Delete Pet


- DELETE
- URL: <http://localhost/user/{userId}/pet/{petid}>
- Response: {

```
status: boolean
}
```

DELETE  http://localhost:4000/user/5ad50f7546c2a9597d48dd2e/pet/5ad8f466641900739c4a72cb

Authorization Headers Body Pre-request Script Tests



TYPE

Inherit auth from parent 

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

This request

Body Cookies Headers (9) Test Results

Pretty Raw Preview JSON  

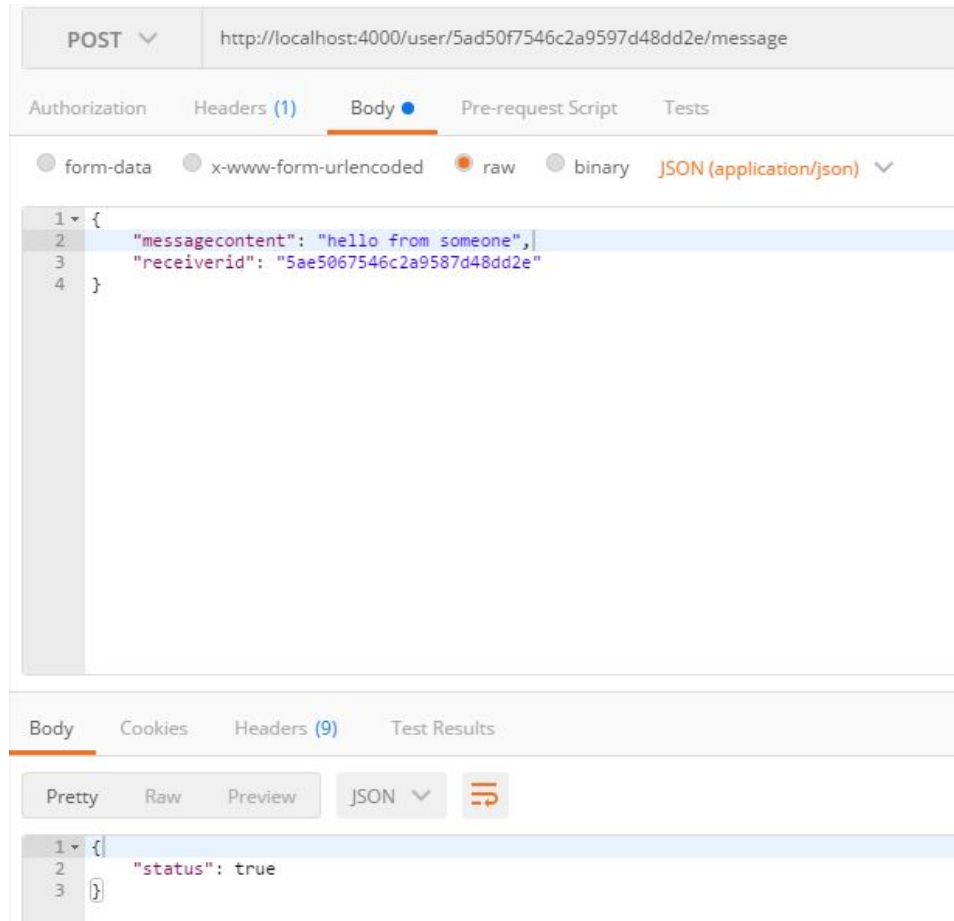
```
1 {
2   "status": true
3 }
```

10. Post Message

- POST
- URL: <http://localhost/user/{userId}/message>
- Body:

```
{
  messageContent: string;
  receiverid: string;
}
```
- Response:

```
{
  status: boolean
}
```



11. Get Message List(sentmessage)

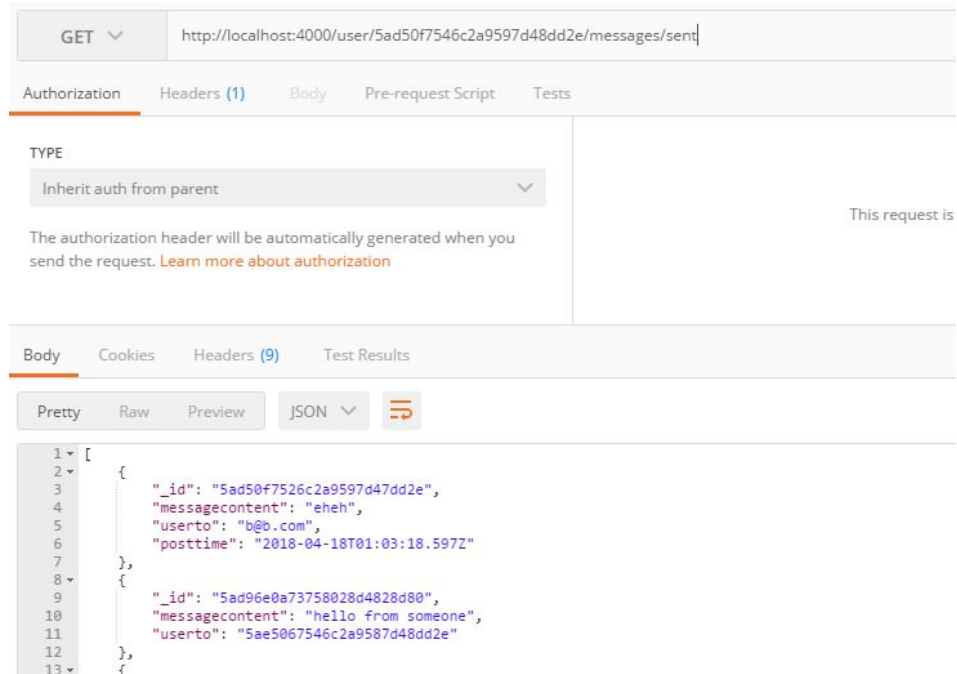
-GET

-URL <http://localhost/user/{userId}/messages/sent>

- Response:[

```
{  _id: string;  messageContent: string;  userid: string;  postTime: date;}
```

]



12. Get Message List(receivemessage)

-GET

-URL <http://localhost/user/{userId}/messages/received>

- Response:[

```
{
  messageId: string;
  messageContent: string;
  userId: string;
  postTime: date;
}
]
```

The screenshot shows a REST client interface. At the top, a dropdown menu is set to 'GET' and the URL is 'http://localhost:4000/user/5ad50f7546c2a9597d48dd2e/messages/received'. Below this, there are tabs for 'Authorization', 'Headers (1)', 'Body', 'Pre-request Script', and 'Tests'. The 'Authorization' tab is selected, showing a 'TYPE' dropdown set to 'Inherit auth from parent' and a note: 'The authorization header will be automatically generated when you send the request. [Learn more about authorization](#)'. Below the tabs, there are more tabs: 'Body', 'Cookies', 'Headers (9)', and 'Test Results'. The 'Body' tab is selected, showing a 'Pretty' button, a 'Raw' button, a 'Preview' button, a 'JSON' dropdown, and a refresh icon. The response body is displayed in a code editor with line numbers 1 through 12. It shows a JSON array with two objects. The first object has fields: '_id', 'messagecontent', 'userto', and 'posttime'. The second object has fields: '_id', 'messagecontent', and 'userto'.

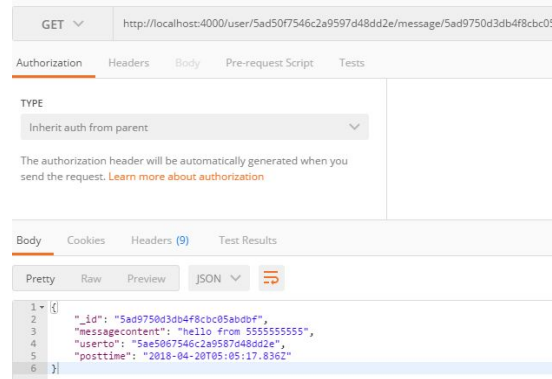
```
1 [
2   {
3     "_id": "5ad50f7526c2a9597d47dd2e",
4     "messagecontent": "eheh",
5     "userto": "b@b.com",
6     "posttime": "2018-04-18T01:03:18.597Z"
7   },
8   {
9     "_id": "5ad96e0a73758028d4828d80",
10    "messagecontent": "hello from someone",
11    "userto": "5ae5067546c2a9587d48dd2e"
12  }
```

13. Get Message by message id

-GET

-URL <http://localhost/user/{userId}/message/{messageid}>

- Response: {
 messageId: string;
 messageContent: string;
 userid: string;
 postTime: date;
}



14. Post bookmark (backend would check url for duplication, if yes, then delete it.)

-POST

-URL <http://localhost/user/{userId}/bookmark>

- Body:{

Tag:string,
url:string
}

- Response: {

status: boolean
}

15. Get bookmark list

-Get

-URL <http://localhost/user/{userId}/bookmarks>

- Response:[{

_id: string,
tag:string,
url:string
}]

16.Delete bookmark

-DELETE

-URL <http://localhost/user/{userId}/bookmark/{id}>

Reference

<https://www.petfinder.com/developers/api-docs>

