

Documentation

Project Structure

Description

With our project we have opted to use two primary apps, one for handling user functionality, and a second for handling any restaurant backend functionality. In the Accounts app our models essentially handle user accounts management and notifications that are sent to each user.

In the Restaurants app the models turn every aspect of the social network functionality, such as posts, following, liking, commenting into their own entities when needed and we also have a primary model for restaurants which references other models like Menu that represent the menu containing food item objects of a specific restaurant in Restify.

We use Generic APIViews to handle the creation, deletion, retrieval and updating of these objects on the backend. Token authentication is used for user accounts authentication for logging users in.

Apps:

- Restaurants
- Accounts

Models

Accounts App:

- UserProfile(AbstractUser)
 - **Fields:** phone_number, profile_picture
- Notifications
 - **Fields:** type, date, to_user, from_user, post, restaurant

Restaurants App:

- Post
 - **Fields:** title, owner, picture, topic, description, created
- Restaurant
 - **Fields:** owner, name, phone_number, address, postal_code, email, logo
- RestaurantImage
 - **Fields:** img, restaurant
- FoodItem
 - **Fields:** name, description, price, menu
- Menu
 - **Fields:** menu_name, restaurant
- Comment
 - **Fields:** title, text, author, rating, restaurant

Accounts

Endpoints:

POST	/accounts/register/	Checks validity of sign up form data and creates new user object
------	---------------------	--

Fields Required:

username

email

password

password2

Example Request & Response

The screenshot shows a REST client interface with a list of requests on the left and a detailed view of a selected POST request on the right. The selected request is to the endpoint `http://127.0.0.1:8000/accounts/register/`. The request body is in 'form-data' format with the following fields:

Key	Value	Description
<input checked="" type="checkbox"/> username	hermansid	
<input checked="" type="checkbox"/> email	herman@h.com	
<input checked="" type="checkbox"/> password	12345678	
<input checked="" type="checkbox"/> password2	12345678	

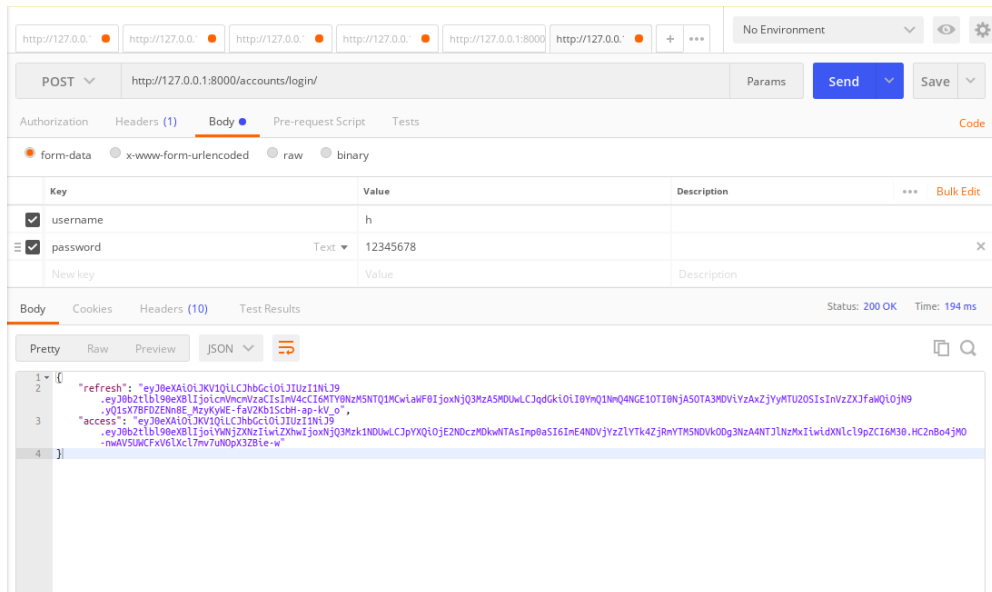
The response body is in 'Pretty' format and shows a successful JSON object:

```
{ "username": "hermansid", "email": "herman@h.com" }
```

POST	/accounts/login/	Obtains access token based on login credentials entered for authentication
------	------------------	--

No fields required, once Access token is given, copy the access token string and add to Headers section as a new key 'Authorization', the value of this key will be 'Bearer ' followed by the access token string.

Example Request & Response



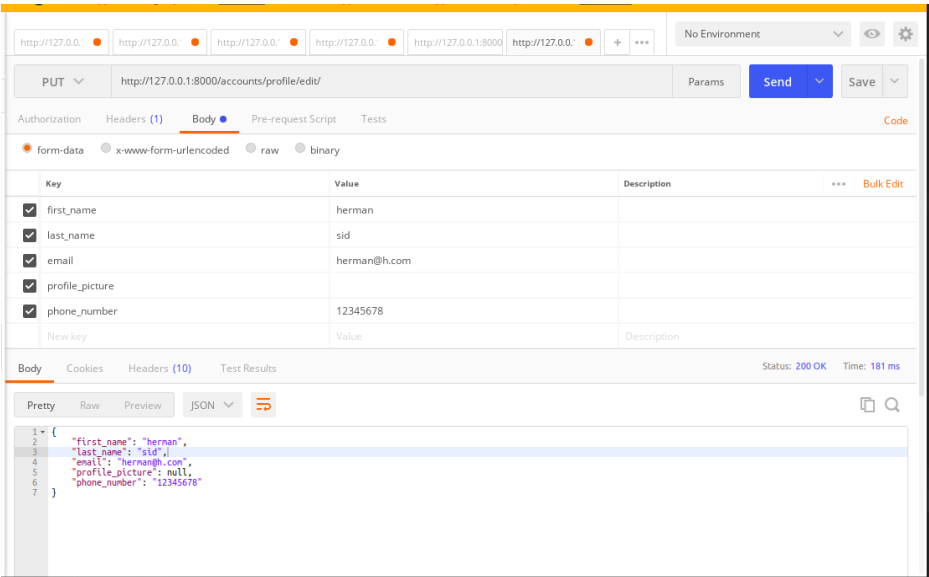
PUT	/accounts/profile/edit/	Makes changes to profile data of user after retrieving the user object associated with authenticated user
-----	-------------------------	---

Fields required:

first_name
 last_name
 email
 profile_picture
 phone_number

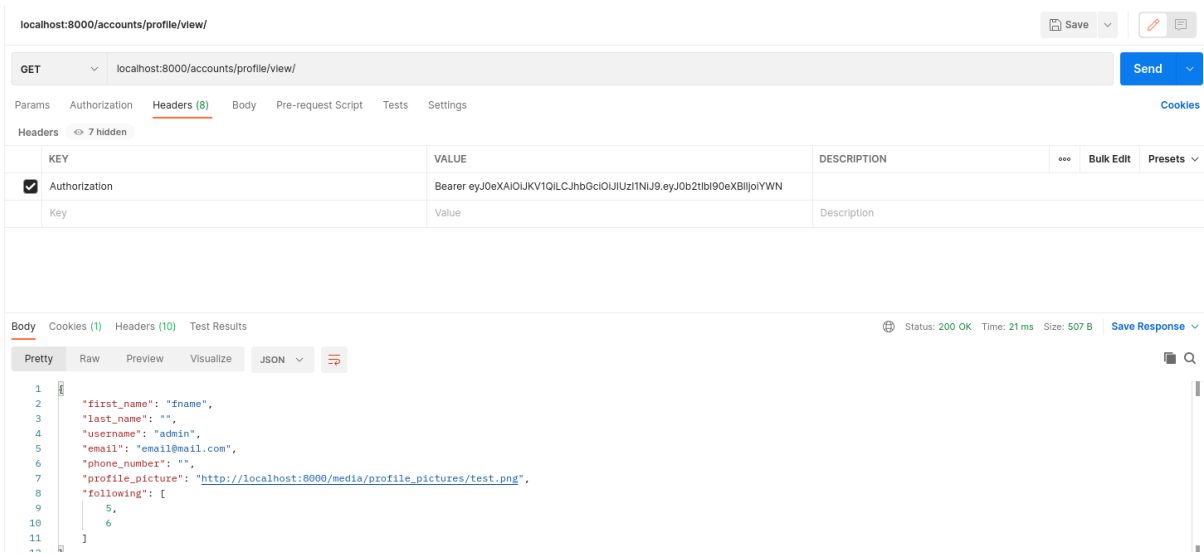
Of the above fields, it is only mandatory to fill email, rest are optional

Example Request & Response



GET	/accounts/profile/view/	Returns the information about the currently logged in user: first name, last name, username, email, phone number, profile picture, and a list of restaurants the user is following.
-----	-------------------------	---

Example Request & Response



GET	/accounts/profile/notifications/	Returns a list of the currently logged in user's notifications ordered with the newest notification first. A user gets a notification when another user likes, comments, or follows their restaurant or when a restaurant they follow makes a post.
-----	----------------------------------	---

Example Request & Response

GETlocalhost:8000/accounts/profile/notifications/

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettings

Headers

7 hidden

KEY	VALUE	DESCRIPTION
<input checked="" type="checkbox"/> Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2t1bi90eXBlljoiYWN	
Key	Value	Description

BodyCookies (1)Headers (10)Test Results

PrettyRawPreviewVisualizeJSON

```
1 {
2   "count": 7,
3   "next": "http://localhost:8000/accounts/profile/notifications/?page=2",
4   "previous": null,
5   "results": [
6     {
7       "type": "Comment",
8       "from_user": "admin",
9       "restaurant": "new_restraunt",
10      "date": "2022-03-15T04:45:59.338142Z"
11    },
12    {
13      "type": "Like",
14      "from_user": "admin",
15      "restaurant": "new_restraunt",
16      "date": "2022-03-15T04:14:11.254356Z"
17    },
18    {
19      "type": "Like",
20      "from_user": "admin",
21      "restaurant": "new_restraunt",
22      "date": "2022-03-15T01:35:32.314494Z"
23    }
24  ]
25 }
```

Status: 200 OKTime: 23 ms

GET	/accounts/profile/my-feed/	Returns a list of all the posts from restaurants that the user follows sorted by creation date with the newest first.
-----	----------------------------	---

Example Request & Response

localhost:8000/accounts/profile/my-feed/

GETlocalhost:8000/accounts/profile/my-feed/

Send

ParamsAuthorizationHeaders (8)BodyPre-request ScriptTestsSettings

Headers

KEY	VALUE	DESCRIPTION
Authorization	Bearer eyJ0eXAiOiJKV1QiLCJhbGciOiJIUzI1NiJ9.eyJ0b2t1bG90eXBhY2tYWN	
Key	Value	Description

BodyCookies (1)Headers (10)Test Results

PrettyRawPreviewVisualizeJSON

```
1 {
2   "count": 8,
3   "next": "http://localhost:8000/accounts/profile/my-feed/?page=2",
4   "previous": null,
5   "results": [
6     {
7       "id": 9,
8       "title": "New Post",
9       "picture": null,
10      "topic": "Food is good",
11      "description": "",
12      "created": "2022-03-14T20:30:12.034059Z",
13      "owner_name": "fname",
14      "num_likes": 2
15    },
16    {
17      "id": 8,
18      "title": "post 1234",
19      "picture": null,
20      "topic": "Food",
21      "description": "",
22      "created": "2022-03-14T19:50:43.697519Z",
23      "owner_name": "fname",
24      "num_likes": 0
25    }
26  ]
27 }
```

Status: 200 OKTime: 19 msSize: 115 KB

Save Response

Restaurant

POST	/restaurants/create/	<p>Creates a new restaurant with the owner set to the currently logged in user. If the user already has a restaurant, then it returns an error message.</p> <p>Fields:</p> <p><u>Name</u> - required</p> <p><u>Phone Number</u> - required</p> <p><u>Address</u> - required</p> <p><u>Postal code</u> - required</p> <p><u>Email</u> - required</p> <p><u>Logo</u> - required</p>
------	----------------------	---

Example Request & Response

localhost:8000/restaurants/create/

POSTlocalhost:8000/restaurants/create/

Send

ParamsAuthorizationHeaders (10)BodyPre-request ScriptTestsSettings

noneform-datax-www-form-urlencodedrawbinaryGraphQL

KEY	VALUE	DESCRIPTION
name	new_restaurant	
phone_number	4161231234	
address	123 Main street	
postal_code	ABSCD1	
email	email@gmail.com	
logo	test.png	

BodyCookies (1)Headers (10)Test Results

PrettyRawPreviewVisualizeJSON

```
1 {
2   "id": 0,
3   "owner_id": 4,
4   "name": "new_restaurant",
5   "phone_number": "4161231234",
6   "address": "123 Main street",
7   "postal_code": "ABSCD1",
8   "email": "email@gmail.com",
9   "logo": "http://localhost:8000/media/restaurant_pictures/test_TipPyts.png",
10  "num_likes": 0,
11  "num_followers": 0,
12  "images": []
13 }
```

Status: 201 CreatedTime: 80 msSize: 584 B

Save Response

GET	/restaurants/my-restaurant/	Returns the information about your restaurant. Returns 404 if the user does not have a restaurant.
-----	-----------------------------	---

Example Request & Response

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

- URL:** localhost:8000/restaurants/my-restaurant/
- Method:** GET
- Body:**

```

1  {
2    "id": 8,
3    "owner_id": 4,
4    "name": "new_restaurant",
5    "phone_number": "4161231234",
6    "address": "123 Main street",
7    "postal_code": "A8SC01",
8    "email": "email@gmail.com",
9    "logo": "http://localhost:8000/media/restaurant_pictures/test_TipRyts.png",
10   "num_likes": 0,
11   "num_followers": 0,
12   "images": []
13 }
```
- Status:** 200 OK, Time: 12 ms, Size: 584 B

GET & PUT/PATCH	/restaurants/my-restaurant/edit/	The GET request will return the information about the logged in users restaurant in the same way as above. A PUT/PATCH request will update the restaurant's information.
-----------------	----------------------------------	--

Example Request & Response

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

- URL:** localhost:8000/restaurants/my-restaurant/edit/
- Method:** PATCH
- Body:**

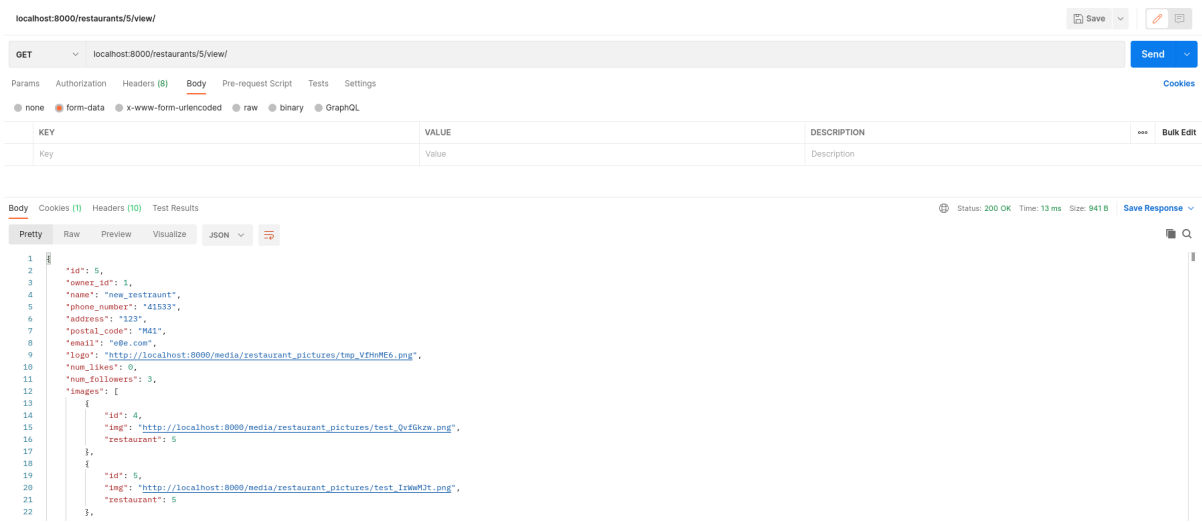
```

1  {
2    "id": 8,
3    "owner_id": 4,
4    "name": "changed",
5    "phone_number": "4161231234",
6    "address": "123 Main street",
7    "postal_code": "A8SC01",
8    "email": "email@gmail.com",
9    "logo": "http://localhost:8000/media/restaurant_pictures/test_TipRyts.png",
10   "num_likes": 0,
11   "num_followers": 0,
12   "images": []
13 }
```
- Status:** 200 OK, Time: 31 ms, Size: 589 B

Here the name attribute was changed to “changed”. The other attributes can be updated similarly.

GET	/restaurants/<int:rid>/view/	This will return the information about an arbitrary restaurant where <int:rid> is the id of the restaurant we want to view.
-----	------------------------------	---

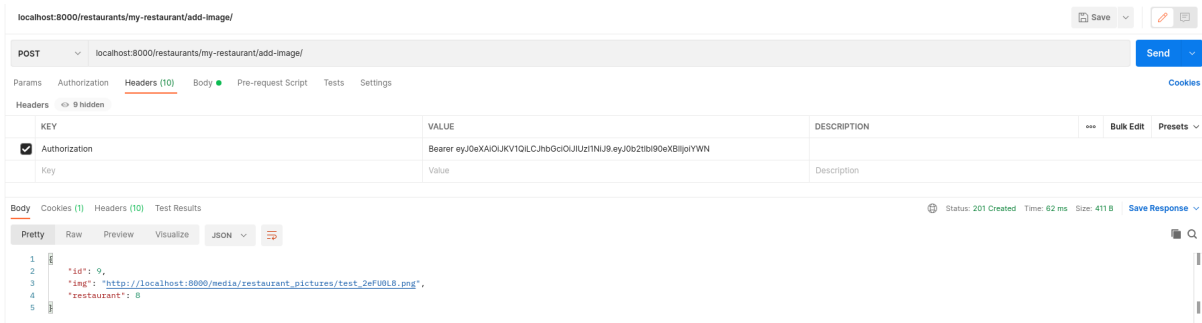
Example Request & Response



Here we are viewing the restaurant with id 5.

POST	/restaurants/my-restaurant/add-image/	<p>This will add an image to the restaurant that is owned by the currently logged in user. If the user does not own a restaurant, it will return an error message.</p> <p>Fields <u>Img</u> - required</p>
------	---------------------------------------	--

Example Request & Response



GET	/restaurants/<int:rid>/follow/	Follow the restaurant with id <i>rid</i> . Returns 404 if a restaurant with <i>rid</i> does not exist. Otherwise, it will add the restaurant to the list of the user's followed restaurants and return the current user object.
-----	--------------------------------	---

Example Request & Response

localhost:8000/restaurants/5/follow/

GET localhost:8000/restaurants/5/follow/ Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

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

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies (1) Headers (10) Test Results

Status: 200 OK Time: 27 ms Size: 455 B Save Response

```
1 {
2   "first_name": "",
3   "last_name": "",
4   "username": "newuser123",
5   "email": "email@gmail.com",
6   "phone_number": "",
7   "profile_picture": null,
8   "following": [
9     5
10  ]
11 }
```

The user is now following the restaurant with id=5.

GET	/restaurants/<int:rid>/unfollow/	Unfollow the restaurant with id <i>rid</i> . If the user is not following the restaurant, it will do nothing. Otherwise the restaurant is removed from the users following list and the user object is returned.
-----	----------------------------------	--

Example Request & Response

Overview POST localhost:8000/accou GET localhost:8000/restaur GET localhost:8000/restaur +

No Environment

localhost:8000/restaurants/5/unfollow/ Save

GET localhost:8000/restaurants/5/unfollow/ Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

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

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies (1) Headers (10) Test Results

Status: 200 OK Time: 15 ms Size: 454 B Save Response

```
1 {
2   "first_name": "",
3   "last_name": "",
4   "username": "newuser123",
5   "email": "email@gmail.com",
6   "phone_number": "",
7   "profile_picture": null,
8   "following": []
9 }
```

Unfollows the restaurant with id=5.

GET	/restaurants/<int:rid>/like/	Likes the restaurant with id <i>rid</i> for the currently logged in user. Returns the restaurant object with id=rid.
-----	------------------------------	--

Example Request & Response

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

- URL:** localhost:8000/restaurants/5/like/
- Method:** GET
- Body:**

```

1 {
2   "id": 5,
3   "owner_id": 1,
4   "name": "new_restaurant",
5   "phone_number": "415333",
6   "address": "123",
7   "postal_code": "M41",
8   "email": "ebe.com",
9   "logo": "http://localhost:8000/media/restaurant_pictures/tmp_VfHnME6.png",
10  "num_likes": 1,
11  "num_followers": 2,
12  "images": [
13    {
14      "id": 4,
15      "img": "http://localhost:8000/media/restaurant_pictures/test_QvFGkw.png",
16      "restaurant": 5
17    },
18    {
19      "id": 5,
20      "img": "http://localhost:8000/media/restaurant_pictures/test_IrWnM3t.png",
21      "restaurant": 5
22    }
23  ]
24 }
```
- Status:** 200 OK, Time: 21 ms, Size: 941 B

The logged in user likes the restaurant with id=5.

GET	/restaurants/<int:rid>/unlike/	Unlikes the restaurant with id <i>rid</i> for the logged in user. Returns the restaurant object with id=rid.
-----	--------------------------------	--

Example Request & Response

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

- URL:** localhost:8000/restaurants/5/unlike/
- Method:** GET
- Body:**

```

1 {
2   "id": 5,
3   "owner_id": 1,
4   "name": "new_restaurant",
5   "phone_number": "415333",
6   "address": "123",
7   "postal_code": "M41",
8   "email": "ebe.com",
9   "logo": "http://localhost:8000/media/restaurant_pictures/tmp_VfHnME6.png",
10  "num_likes": 0,
11  "num_followers": 2,
12  "images": [
13    {
14      "id": 4,
15      "img": "http://localhost:8000/media/restaurant_pictures/test_QvFGkw.png",
16      "restaurant": 5
17    },
18    {
19      "id": 5,
20      "img": "http://localhost:8000/media/restaurant_pictures/test_IrWnM3t.png",
21      "restaurant": 5
22    }
23  ]
24 }
```
- Status:** 200 OK, Time: 25 ms, Size: 941 B

Unlikes the restaurant with id=5.

Comments

POST	/restaurants/add_comment/	Adds a comment to a restaurant by authenticated user
------	---------------------------	--

Fields required:

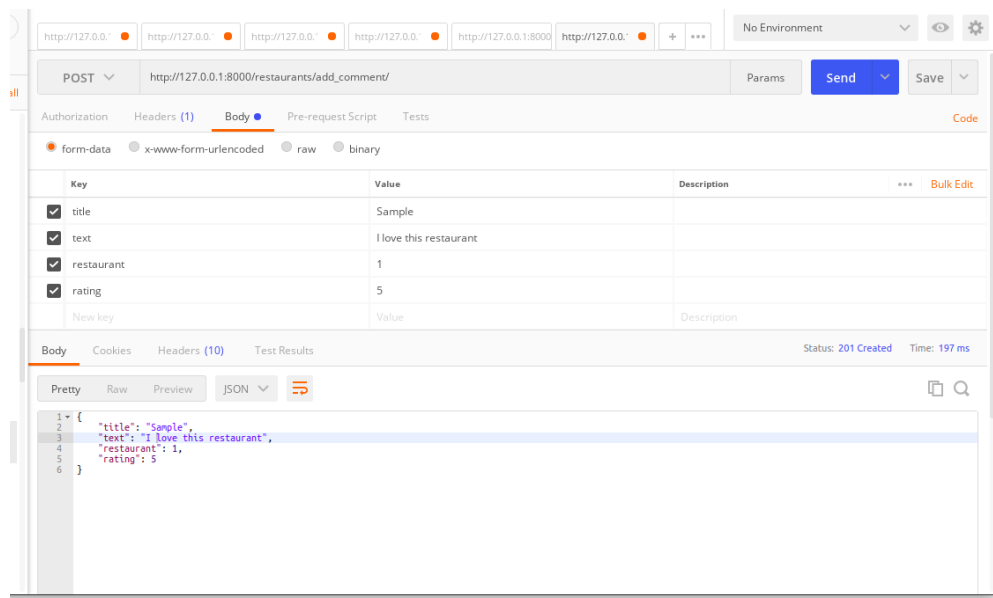
title: string

text: string

restaurant: Integer ID

rating: integer between 0 and 5

Example Request & Response



PUT	/restaurants/<str:title>/edit_comment	Retrieves comment from its title given in URL and makes changes to it if user is the author of comment
-----	---------------------------------------	--

Fields required:

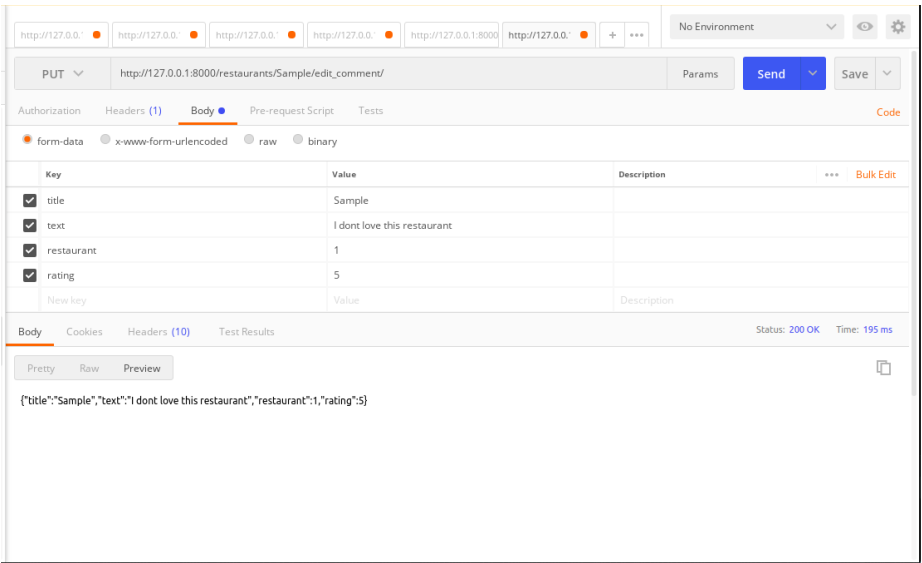
title: string

text: string

restaurant: Integer ID

rating: integer between 0 and 5

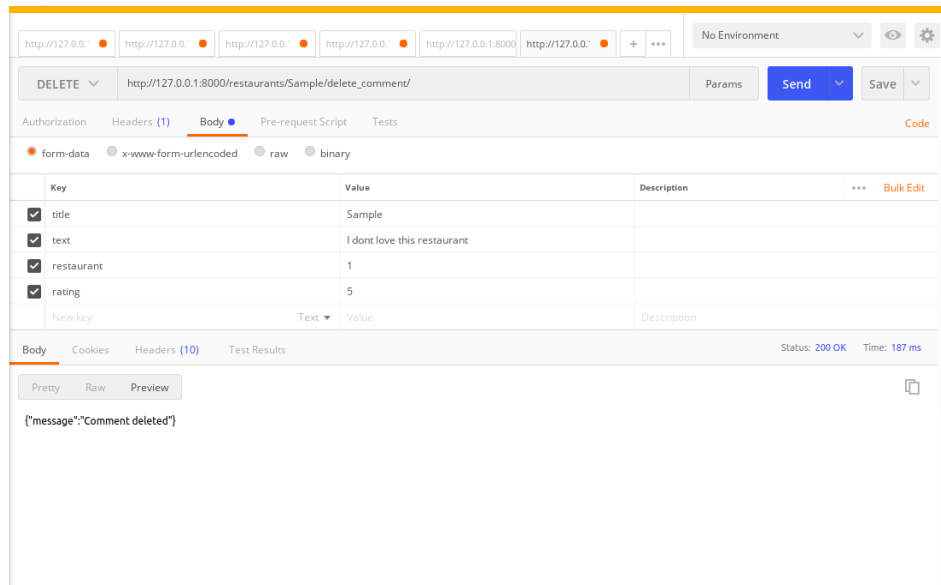
Example Request & Response



DELETE	/restaurants/<str:title>/delete_comment	Retrieves comment based on URL provided title and deletes it if user is the author of the comment
--------	---	---

No fields required

Example Request & Response



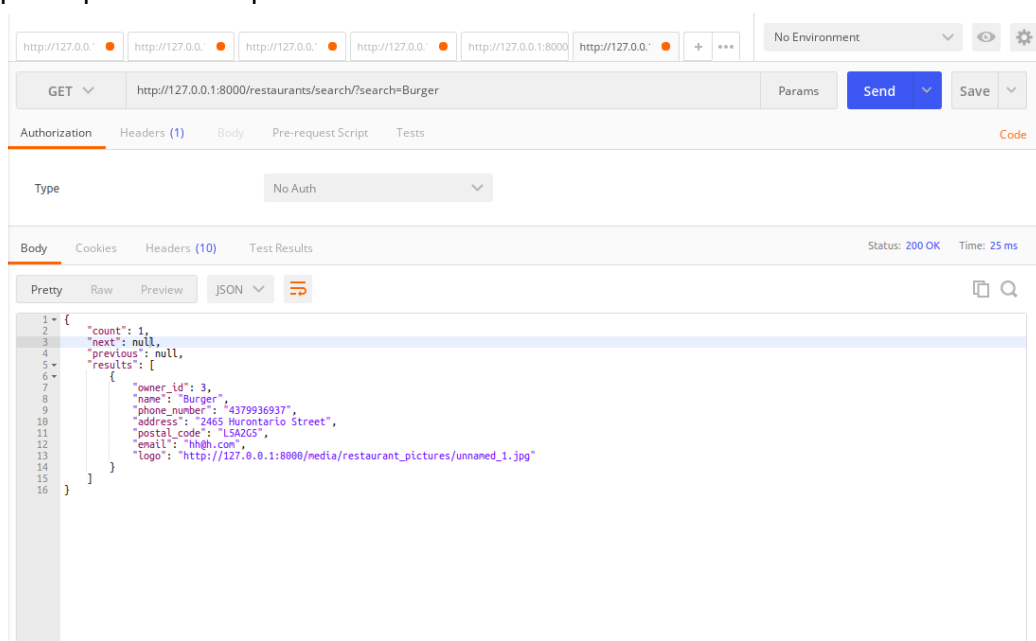
Restaurant Search

Endpoints:

GET	/restaurants/search/	Conducts a search for restaurant object(s) based on search input
-----	----------------------	--

For this no fields are required however to conduct the search, an argument ?search is required to be filled in the url, Thus if I wanted to search for keyword 'Burger' the url would be **/restaurants/search/?search=Burger**

Example request and response



Blog Posts

Endpoints:

POST	/restaurants/create-blog-post/	<p>Add a restaurant blog post. The author is the restaurant owner who is currently logged in.</p> <p>Request Body Parameters:</p> <p><i>title</i> : string (required) <i>picture</i> : file <i>topic</i>: string (required) <i>description</i> : string</p>
------	--------------------------------	---

POST

http://localhost:8000/restaurants/create-blog-post/

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
<input checked="" type="checkbox"/>	title	"hello world"			
<input checked="" type="checkbox"/>	topic	"first post"			
<input checked="" type="checkbox"/>	description	"This is a new post"			
<input checked="" type="checkbox"/>	picture	2014-02-18 21.35.14-1.jpg			
	Key	Value	Description		

body

Cookies

Headers (10)

Test Results

201 Created

46 ms

564 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

```
1 {
2   "id": 1,
3   "title": "\"hello world\"",
4   "picture": "http://localhost:8000/media/blog_pictures/2014-02-18_21.35.14-1.jpg",
5   "topic": "\"first post\"",
6   "description": "\"This is a new post\"",
7   "created": "2022-03-15T16:11:54.451970Z",
8   "owner_name": "",
9   "num_likes": 0
10 }
```

GET	/restaurants/<int:owner_id>/posts/	View blog posts created by a restaurant owner. <i>owner_id</i> is a restaurant owner's ID.
-----	------------------------------------	---

GET

http://localhost:8000/restaurants/2/posts/

Send

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettingsCookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

BodyCookiesHeaders (10)Test Results200 OK13 ms616 BSave Response

PrettyRawPreviewVisualizeJSON

```
1 {
2   "count": 1,
3   "next": null,
4   "previous": null,
5   "results": [
6     {
7       "id": 1,
8       "title": "\"hello world\"",
9       "picture": "http://localhost:8000/media/blog_pictures/2014-02-18_21.35.14-1.jpg",
10      "topic": "\"first post\"",
11      "description": "\"This is a new post\"",
12      "created": "2022-03-15T16:11:54.451970Z",
13      "owner_name": "",
14      "num_likes": 0
15    }
16  ]
17 }
```

DELETE	/restaurants/<int:pk>/delete-post/	Remove a restaurant blog post. <i>Pk</i> is the ID for the post
--------	------------------------------------	--

DELETE

http://localhost:8000/restaurants/1/delete-post/

Send

ParamsAuthorizationHeaders (7)BodyPre-request ScriptTestsSettingsCookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

	KEY	VALUE	DESCRIPTION	...	Bulk Edit
	Key	Value	Description		

BodyCookiesHeaders (10)Test Results200 OK15 ms351 BSave Response

PrettyRawPreviewVisualizeJSON

```
1 {
2   "message": "Post deleted successfully"
3 }
```

GET	/restaurants/post/<int:pid>/like/	Likes the post with id <i>pid</i> if it exists and returns the serialized post object. Otherwise returns 404.
-----	-----------------------------------	---

localhost:8000/restaurants/post/1/like/

GET localhost:8000/restaurants/post/1/like/ Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

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

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies (1) Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "title": "my first post!",
4   "picture": null,
5   "topic": "Food",
6   "description": "Desc",
7   "created": "2022-03-09T21:50:57.053148Z",
8   "owner_name": "fname",
9   "num_likes": 1
10 }
```

Status: 200 OK Time: 16 ms Size: 475 B Save Response

GET	/restaurants/post/<int:pid>/unlike/	Unlikes the post with id <i>pid</i> if it exists. Returns the serialized post object or 404 if it doesn't exist.
-----	-------------------------------------	--

Overview POST localhost:8000/accounts GET localhost:8000/restaur GET localhost:8000/restaur + ... No Environment

localhost:8000/restaurants/post/1/unlike/ Save

GET localhost:8000/restaurants/post/1/unlike/ Send

Params Authorization Headers (8) Body Pre-request Script Tests Settings

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

KEY	VALUE	DESCRIPTION
Key	Value	Description

Body Cookies (1) Headers (10) Test Results

Pretty Raw Preview Visualize JSON

```
1 {
2   "id": 1,
3   "title": "my first post!",
4   "picture": null,
5   "topic": "Food",
6   "description": "Desc",
7   "created": "2022-03-09T21:50:57.053148Z",
8   "owner_name": "fname",
9   "num_likes": 0
10 }
```

Status: 200 OK Time: 16 ms Size: 475 B Save Response

Menu

Endpoints

POST	/restaurants/create-menu/	Create a menu with food items. Request Body Parameters: - <i>menu_name</i> : string required - <i>foods</i> : a list of FoodItems required
------	---------------------------	---

POST

http://localhost:8000/restaurants/create-menu/

Send

Params

Authorization

Headers (9)

Body

Pre-request Script

Tests

Settings

Cookies

none

form-data

x-www-form-urlencoded

raw

binary

GraphQL

JSON

Beautify

1

2

3

4

5

6

7

```
{
  "menu_name": "fancy lunch",
  "foods": [
    { "name": "chickpea salad", "description": "gluten free option", "price": 19.0 },
    { "name": "salmon", "description": "seafood option", "price": 25.0 }
  ]
}
```

Body

Cookies

Headers (9)

Test Results

201 Created

143 ms

478 B

Save Response

Pretty

Raw

Preview

Visualize

JSON

1

2

3

4

5

6

7

8

9

10

11

12

13

14

```
{
  "id": 2,
  "menu_name": "fancy lunch",
  "restaurant": "cafe lux",
  "foods": [
    {
      "name": "chickpea salad",
      "description": "gluten free option",
      "price": 19.0
    },
    {
      "name": "salmon",
      "description": "seafood option",
      "price": 25.0
    }
  ]
}
```

GET	/restaurants/view-menu/<int:pk>	View menu. <i>PK</i> is the menu ID.
-----	---------------------------------	--------------------------------------

GET

http://localhost:8000/restaurants/view-menu/2/

Send

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

Cookies

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

This request does not have a body

Body Cookies Headers (10) Test Results

200 OK 12 ms 443 B Save Response

Pretty
Raw
Preview
Visualize
JSON

```

1  {
2    "id": 2,
3    "menu_name": "new menu",
4    "restaurant": "cafe lux",
5    "foods": [
6      {
7        "name": "waffles",
8        "description": "belgian style",
9        "price": 5.99
10     }
11   ]
12 }
```

DELETE	/restaurants/delete-menu/<int:pk>	Delete a menu by its ID (PK)
--------	-----------------------------------	------------------------------

DELETE

http://localhost:8000/restaurants/delete-menu/3/

Send

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

Cookies

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

Beautify

1

Body Cookies Headers (9) Test Results

200 OK 99 ms 309 B Save Response

Pretty
Raw
Preview
Visualize
JSON

```

1  {
2    "message": "Menu deleted successfully"
3 }
```