

CSC309 Restify Documentation

Maor Gornic, Michael Sheinman Orenstrakh, Richard Lu

March 16, 2022

Contents

- 1 Introduction**
- 2 Database Design and ERD Diagram**
- 3 Design Decisions**
- 4 How to Run**

1 Introduction

This document describes the implementation of the backend server of Restify. The backend server provides a RESTful API, implemented using Django. There are two Django apps: an accounts app and a restaurants app. We kept the API to these two apps since we noticed that all of the user stories are either related to the accounts or the restaurants. All of the data is stored using an SQLite database.

The project follows a standard Django structure, similar to how assignment 2 was handled. There are two apps corresponding to the restaurants and the accounts app and a main Resify project structure. There is a `urls.py` in each app. This file consists of all the path links referencing the different REST API endpoints. This url file references the corresponding REST view files.

2 Database Design and ERD Diagram

Our database design consists of a few key database entities, including Notification, User, Restaurant, Blog, Menu Items, and Restaurant images. These entities are connected through relationships. In the case of a many-to-many relationship, we further create a table for the relationships. Hence we have additional tables for a comment, restaurant like, blog post like, and followers.

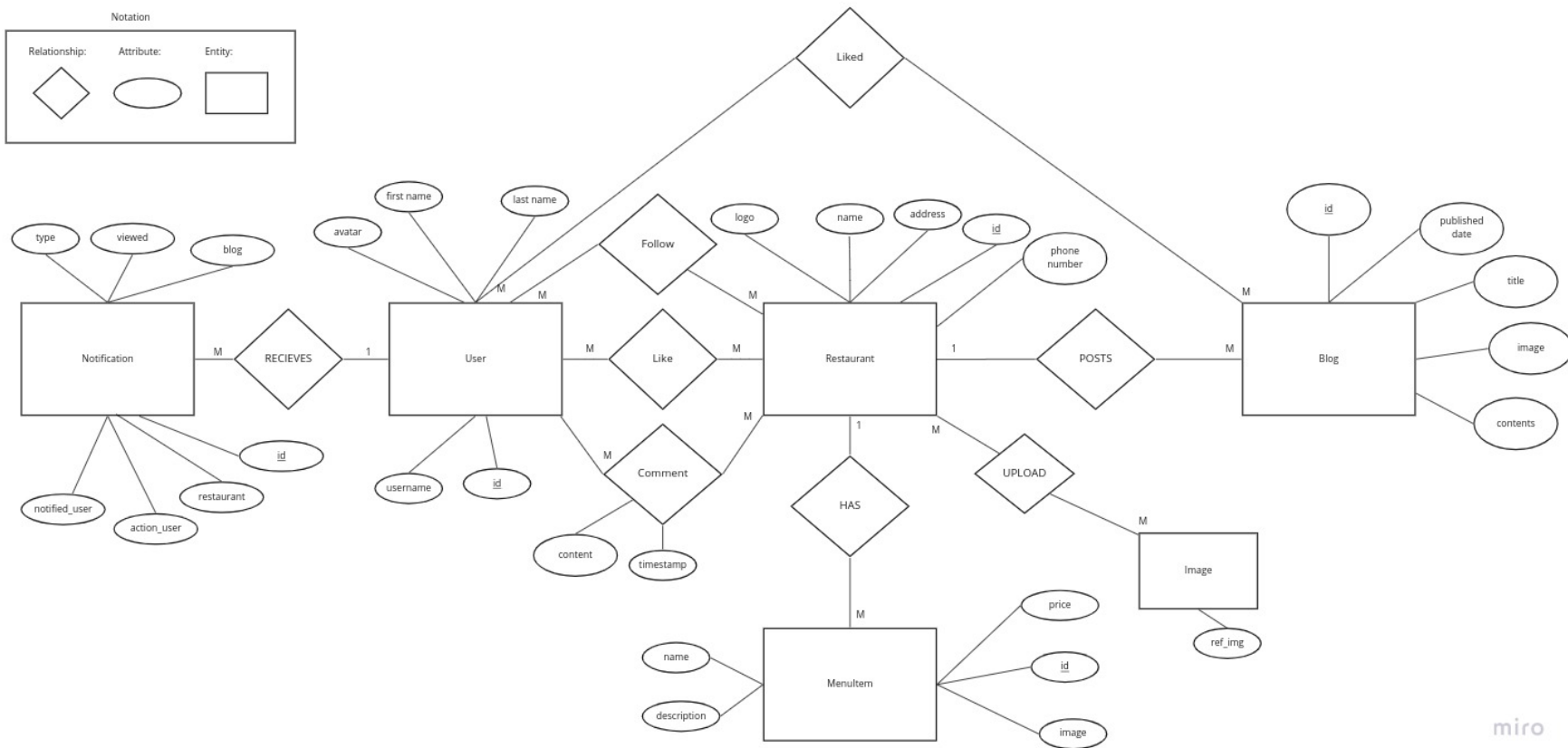
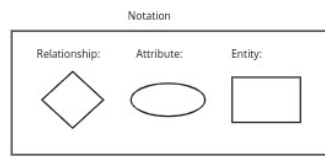
Notification

Our notification table consists of a significant number of attributes. These include:

1. Notified User: This is the user that will receive the notification
2. Type: The type of the notification. The type is an enum and has 6 types corresponding to a new blog post, a menu update, a follow, a restaurant like, a blog post like, and a comment on a restaurant. We plan to use each notification type to determine which of the following attributes need to be accessed.
3. Viewed: This is a boolean flag that indicates whether the user already viewed the notification. This is used by the frontend to determine how the notification should be displayed.
4. Blog, Restaurant and actor id: These are optional attributes that corresponds to specific notification types. These attributes will be used by the frontend to determine a notification message and a frontend redirect url for the notification.

Restaurant

The restaurant is the central entity as it is connected with the majority of tables. The restaurant stores contact information such as name, address, logo, postal code and phone number as attributes. As for more complex relationships like comment and like the restaurants uses separate tables (Like, Blog, Follow, etc). These tables are connected through foreign keys.



3 Design Decisions

Login and Logout

Login is implemented with JWT tokens. A login involves generating a new JWT token through the GET: `accounts/token` endpoint. Logout is implemented by invalidating the user's access token. Once a user has a token the user must send the token in all subsequent API requests as a Bearer token, otherwise the API endpoints will return a not authorized 401 status code.

Notifications

Our notification endpoints design allows users to create a notification, get notifications, and mark notifications as viewed. Each of these actions are handled separately.

- Creation: Our APIs do not provide a direct endpoint to create a notification. Instead, notifications are created as part of other APIs.
- Fetch: We provide an API endpoint `accounts/notifications/` for users to view their notifications. The endpoint is paginated.
- View: There is a separate API endpoint to mark a notification as viewed, located at `accounts/notifications/viewed/<notification-id>`. This notification updates the notification's status to viewed.

Restaurants

The restaurant app contains all endpoints relating to restaurants. These include endpoints to add menu items, add blog posts, and handle followers, likes, and comments. We keep all of these endpoints in one the restaurant app since all of these functionalities are bound to restaurants.

Comments

For the Comment model, we have decided to have the following attributes: a user that is referencing (foreign key) of Modifieduser object, an auto-generated and created timestamp of the comment creation time, a restaurant that is referencing (foreign key) of Restaurant object, and finally the content of the actual comments being sent. According to our ERD, the behaviour is correctly implemented regarding of the model. Correct usage are also introduced with endpoints.

Regarding of the endpoints, we have two endpoints:

- `restaurants/<int:restaurant_id>/comments/all/`, a GET request
- `restaurants/<int:restaurant_id>/comments/new/`, a POST request

Where after requesting with the `restaurant_id`, we can either send a GET request with endpoint 1 to get all comments made under a restaurant, and a POST request with endpoint 2 to make a comment under that restaurant specified as a user. We have designed this way so that it is easy to either get the comments or to make a comment under the Comment model.

Blog

For the Blog model, we have attributes:

`restaurants`, `title`, `banner`, `contents`, `publish_timestamp`, `likes`
Using this model we can correctly store the blog objects with the expected behaviour.

4 How to Run


To run the backend, you will need to type `source startup.sh`. This will create a virtual environment called `venv`, activate it, install all the required dependencies and lastly perform the migrations. Once this command is done executing, proceed to type `source run.sh` to actually run the server.

Restify API Documentation

[Base URL: 127.0.0.1:8000]

Schemes

HTTP

Authorize 

accounts



POST /accounts/api/token/

Parameters

Try it out

Name	Description
data (body)	<div>Example Value Model</div> <pre>{ "username": "string", "password": "string"}</pre> <div>Parameter content type</div> <div>application/json</div>

Responses		Response content type
		application/json
Code	Description	
201	<i>Login Successful</i>	
401	<i>Login Unauthorized</i>	

POST /accounts/register/		
Parameters		Try it out
Name	Description	
data (body)	<div>Example Value Model</div> <pre>{ "username": "string", "password": "string", "password2": "string", "email": "string", "first_name": "string", "last_name": "string" }</pre> <div>Parameter content type</div> <div>application/json</div>	
Responses		Response content type application/json

Code	Description
201	<i>User created</i>
400	<i>Bad Requests</i>

PATCH

/accounts/update/

Parameters

Try it out

Name	Description
data (body)	<div><div>Example Value</div><div>Model</div><pre>{ "username": "string", "first_name": "string", "last_name": "string", "email": "string", "avatar": "string", "phone_num": "string" }</pre><div>Parameter content type</div><div>application/json</div></div>

Responses

Response content type

application/json

Code	Description
------	-------------

Code	Description
200	<i>Updated successfully</i>
401	<i>Unauthorized user</i>

GET /accounts/view/	
Parameters	Try it out
No parameters	
Responses	Response content type application/json
Code	Description
200	<i>Viewed successfully</i>
401	<i>Unauthorized user</i>

GET /accounts/notifications/	
Parameters	Try it out
Name	Description

Name	Description
page integer (query)	A page number within the paginated result set.
Responses	Response content type <div>application/json</div>
Code	Description
200	<i>Fetch successful</i>
401	<i>User unauthorized</i>
404	<i>Invalid Page</i>

PATCH	/accounts/notifications/viewed/{notification_id}/
Parameters	<div>Try it out</div>
Name	Description
notification_id * required string (path)	

Name	Description
data (<i>body</i>)	<div>Example Value Model</div> <pre>{ "user": "string", "type": "string", "viewed": true, "blog": 0, "restaurant": 0, "actor_user": "string" }</pre> <div>Parameter content type</div> <div>application/json</div>

Responses	Response content type	application/json
-----------	-----------------------	------------------

Code	Description
200	<i>Notification viewed</i>
401	<i>User is not authorized</i>
403	<i>User does not own notification</i>
404	<i>Notification not found</i>

restaurants



GET

/restaurants/all/

Parameters

Try it out

Name	Description
page integer (query)	A page number within the paginated result set.

Responses

Response content type

application/json

Code	Description
200	Retrieved successfully

GET

/restaurants/blog/all/

Parameters

Try it out

Name	Description
------	-------------

Name	Description
page integer (query)	A page number within the paginated result set.
Responses	
Response content type	
application/json	
Code	Description
200	Retrieved successfully

GET	/restaurants/blog/doeslike/{blog_id}/
Parameters	
Try it out	
Name	Description
blog_id * required string (path)	
Responses	
Response content type	
application/json	
Code	Description

Code	Description
200	<i>Retrieved successfully</i>
401	<i>Unauthorized</i>
403	<i>No permission</i>
404	<i>Blog id not found</i>

GET /restaurants/blog/feed/					
Parameters	Try it out				
<table><tr><th>Name</th><th>Description</th></tr><tr><td>page integer (query)</td><td>A page number within the paginated result set.</td></tr></table>	Name	Description	page integer (query)	A page number within the paginated result set.	
Name	Description				
page integer (query)	A page number within the paginated result set.				
Responses	Response content type application/json				
<table><tr><th>Code</th><th>Description</th></tr><tr><td>200</td><td><i>Retrieved successfully</i></td></tr></table>	Code	Description	200	<i>Retrieved successfully</i>	
Code	Description				
200	<i>Retrieved successfully</i>				

Code	Description
401	<i>User unauthorized</i>
404	<i>Invalid Page</i>

GET /restaurants/blog/{blog_id}/							
Parameters	Try it out						
<table><tr><th>Name</th><th>Description</th></tr><tr><td>blog_id * required string (path)</td><td></td></tr></table>		Name	Description	blog_id * required string (path)			
Name	Description						
blog_id * required string (path)							
Responses	Response content type application/json						
<table><tr><th>Code</th><th>Description</th></tr><tr><td>200</td><td><i>Retrieved successfully</i></td></tr><tr><td>404</td><td><i>blog id not found</i></td></tr></table>		Code	Description	200	<i>Retrieved successfully</i>	404	<i>blog id not found</i>
Code	Description						
200	<i>Retrieved successfully</i>						
404	<i>blog id not found</i>						

PATCH /restaurants/blog/{blog_id}/like/	

Parameters

Try it out

Name	Description
blog_id * required string (path)	

Responses	Response content type	application/json
-----------	-----------------------	------------------

Code	Description
200	<i>Like successful</i>
401	<i>User unauthorized</i>
404	<i>blog id does not exist</i>
409	<i>User already liked this blog</i>

DELETE /restaurants/blog/{id}/delete/

Parameters	Try it out
------------	------------

Name	Description
------	-------------

Name	Description
id * required integer (path)	A unique integer value identifying this blog.
Responses	Response content type <div>application/json</div>
Code	Description
204	<i>Removed successfully</i>
401	<i>User unauthorized</i>
403	<i>No permission</i>
404	<i>Blog ID is not found</i>

GET	/restaurants/doesfollow/{restaurant_id}/
Parameters	<div>Try it out</div>
Name	Description
restaurant_id * required string (path)	

Responses

Response content type

application/json

Code	Description
200	<i>Retrieved successfully</i>
401	<i>User unauthorized</i>
404	<i>Not found</i>

GET /restaurants/doeslike/{restaurant_id}/

Parameters

Try it out

Name	Description
restaurant_id * required string (path)	

Responses

Response content type

application/json

Code	Description
------	-------------

Code	Description
200	<i>Retrieved successfully</i>
401	<i>User unauthorized</i>
404	<i>Not found</i>

GET

/restaurants/info/{restaurant_id}/

Parameters

Try it out

Name

Description

restaurant_id * required

string

(path)

Responses

Response content type

application/json

Code

Description

200

Retrieved successfully

404

restaurant id not found

POST

/restaurants/new/

Parameters

Try it out

Name	Description
data (body)	<div><div>Example Value</div><div>Model</div><pre>{ "name": "string", "address": "string", "email": "string", "phone_num": "string", "logo": "string", "postal_code": "string" }</pre><div>Parameter content type</div><div>application/json</div></div>

Responses

Response content type application/json

Code	Description
200	<i>Retrieved successfully</i>
401	<i>User unauthorized</i>
409	<i>Restaurant exists</i>

GET

/restaurants/owned/

Parameters

Try it out

No parameters

Responses

Response content type

application/json

Code	Description
200	Success
404	Restaurant with the given name was not found

GET

/restaurants/search/

Parameters

Try it out

Name	Description
page integer (query)	A page number within the paginated result set.
string string (query)	A query parameter to filter restaurants by. Options are (name, postal_code, food)

Responses

Response content type

application/json

Code	Description
200	<i>Success</i>
404	<i>page number not found</i>

POST /restaurants/{restaurant_id}/blog/new/

Parameters

Try it out

Name	Description
restaurant_id * required string (path)	Restaurant Integer ID to uniquely identify this restaurant.
data (body)	<div>The data field of the POST Request to create the restaurant blog.</div> <div>Example Value Model</div> <div><pre>{ "title": "string", "banner": "string", "contents": "string"}</pre></div> <div>Parameter content type</div> <div>application/json</div>

Responses

Response content type

application/json

Code

Description

201

Successfully added a blog under this restaurant

403

User authorization level is not enough

404

Restaurant not found

GET

/restaurants/{restaurant_id}/comments/all/

Parameters

Try it out

Name

Description

restaurant_id * required
string
(path)

Restaurant Integer ID to uniquely identify this restaurant.

page
integer
(query)

A page number within the paginated result set.

Responses

Response content type

application/json

Code

Description

Code	Description
200	<i>Successfully fetched all comments of ths restaurant</i>
404	<i>Restaurant not found</i>

POST

/restaurants/{restaurant_id}/comments/new/

Parameters

Try it out

Name	Description
restaurant_id * required string (path)	Restaurant Integer ID to uniquely identify this restaurant.
data (body)	<div>The data field of the POST Request to create the restaurant comment.</div> <div>Example Value Model</div> <div><pre>{ "contents": "string" }</pre></div> <div>Parameter content type</div> <div>application/json</div>

Responses

Response content type

application/json

Code	Description
------	-------------

Code	Description
201	<i>Successfully commented ths restaurant</i>
403	<i>User authorization level is not enough</i>
404	<i>Restaurant not found</i>

PATCH

/restaurants/{restaurant_id}/edit/

Parameters

Try it out

Name	Description
restaurant_id * required string (path)	Restaurant Integer ID to uniquely identify this restaurant.
data (body)	<div>The data field of the PATCH Request to modify the restaurant information.</div> <div>Example Value Model<pre>{ "name": "string", "address": "string", "email": "string", "phone_num": "string", "logo": "string", "postal_code": "string"}</pre></div> <div>Parameter content type<div>application/json</div></div>

Responses

Response content type

application/json

Code	Description
200	<i>Successfully updated the restaurant information</i>
403	<i>User authorization level is not enough</i>
404	<i>Restaurant not found</i>

PATCH /restaurants/{restaurant_id}/follow/

Parameters

Try it out

Name	Description
restaurant_id * required string (path)	Restaurant Integer ID to uniquely identify this restaurant to follow.

Responses

Response content type

application/json

Code	Description
200	<i>Successfully followed the restaurant</i>

Code	Description
403	<i>User authorization level is not enough</i>
404	<i>Restaurant not found</i>
409	<i>Already followed the restaurant</i>

GET

/restaurants/{restaurant_id}/followers/

Parameters

Try it out

Name	Description
restaurant_id * required string (path)	Restaurant Integer ID to uniquely identify this restaurant.

Responses

Response content type

application/json

Code	Description
200	<i>Successfully fetched all followers of the restaurant</i>
403	<i>User authorization level is not enough</i>

Code	Description
404	<i>Restaurant not found</i>

GET /restaurants/{restaurant_id}/images/							
Parameters	<div>Try it out</div>						
<table><tr><th>Name</th><th>Description</th></tr><tr><td>restaurant_id * required string (path)</td><td>Restaurant Integer ID to uniquely identify this restaurant.</td></tr><tr><td>page integer (query)</td><td>A page number within the paginated result set.</td></tr></table>	Name	Description	restaurant_id * required string (path)	Restaurant Integer ID to uniquely identify this restaurant.	page integer (query)	A page number within the paginated result set.	
Name	Description						
restaurant_id * required string (path)	Restaurant Integer ID to uniquely identify this restaurant.						
page integer (query)	A page number within the paginated result set.						
Responses	<div>Response content typeapplication/json</div>						
<table><tr><th>Code</th><th>Description</th></tr><tr><td>201</td><td><i>Successfully fetched all images of the restaurant</i></td></tr><tr><td>404</td><td><i>Restaurant not found</i></td></tr></table>	Code	Description	201	<i>Successfully fetched all images of the restaurant</i>	404	<i>Restaurant not found</i>	
Code	Description						
201	<i>Successfully fetched all images of the restaurant</i>						
404	<i>Restaurant not found</i>						

POST /restaurants/{restaurant_id}/images/upload/	
--	--

Parameters

Try it out

Name	Description
restaurant_id * required string (path)	Restaurant Integer ID to uniquely identify this restaurant.
data (body)	Post request to upload this image in the request to the restaurant. Example Value Model <pre>{ "ref_img": "string" }</pre> Parameter content type <div>application/json</div>

Responses

Response content type

application/json

Code	Description
201	<i>Successfully uploaded this image to the restaurant</i>
403	<i>User authorization level is not enough</i>
404	<i>Restaurant not found / Image does not exist</i>

DELETE

/restaurants/{restaurant_id}/images/{image_id}/remove/

Parameters

Try it out

Name	Description
restaurant_id * required string (path)	Restaurant Integer ID to uniquely identify this restaurant.
image_id * required string (path)	Integer ID to uniquely identify this image of the restaurant.

Responses

Response content type

application/json

Code	Description
204	<i>Successfully removed this image of the restaurant</i>
403	<i>User authorization level is not enough</i>
404	<i>Restaurant not found</i>

PATCH

/restaurants/{restaurant_id}/like/

Parameters

Try it out

Name	Description
restaurant_id * required string (path)	Restaurant Integer ID to uniquely identify this restaurant.
Responses	Response content type <div>application/json</div>
Code	Description
200	<i>Successfully liked this restaurant</i>
404	<i>Restaurant not found</i>
409	<i>User has already liked this restaurant</i>

GET	/restaurants/{restaurant_id}/menu/items/
Parameters	<div>Try it out</div>
Name	Description
restaurant_id * required string (path)	Restaurant Integer ID to uniquely identify this restaurant.
page integer (query)	A page number within the paginated result set.

Responses

Response content type

application/json

Code

Description

200 *Successfully fetched all the menu items*

404 *Restaurant not found*

POST

/restaurants/{restaurant_id}/menu/new/

Parameters

Try it out

Name

Description

restaurant_id * required
string
(path)

Restaurant Integer ID to uniquely identify this restaurant.

data
(body)

Below are the fields of the POST Request to create this menu item.

Example Value Model

```
{
  "name": "string",
  "description": "string",
  "price": 0,
  "picture": "string"
}
```

Parameter content type

application/json

Responses

Response content type

application/json

Code	Description
201	<i>Successfully created the menu item</i>
403	<i>User authorization level is not enough</i>
404	<i>Restaurant not found</i>

PATCH /restaurants/{restaurant_id}/menu/{id}/edit/

Parameters

Try it out

Name	Description
restaurant_id * required string (path)	Restaurant Integer ID to uniquely identify this restaurant.
id * required integer (path)	A unique integer value identifying this menu item.

Name	Description
data (<i>body</i>)	<p>Below are the fields of the PATCH Request to change this menu item.</p> <p>Example Value Model</p> <pre>{ "name": "string", "description": "string", "price": 0, "picture": "string" }</pre> <p>Parameter content type</p> <div>application/json</div>

Responses

Response content type

application/json

Code	Description
200	<i>Successfully updated the menu item</i>
403	<i>User authorization level is not enough</i>
404	<i>Restaurant not found</i>

DELETE

/restaurants/{restaurant_id}/menu/{id}/remove/

Parameters

Try it out

Name		Description
restaurant_id * required		
string (path)		Restaurant integer to locate the menu item, providing its ID.
id * required		
integer (path)		A unique integer value identifying this menu item.
Responses		Response content type <div>application/json</div>
Code	Description	
204	<i>Successfully removed the menu item of the restaurant</i>	
403	<i>User authorization level is not enough</i>	
404	<i>Restaurant not found</i>	

PATCH		/restaurants/{restaurant_id}/unfollow/
Parameters		<div>Try it out</div>
Name		Description
restaurant_id * required		
string (path)		Unfollow the restaurant providing the restaurant ID.

Responses

Response content type

application/json

Code

Description

200	<i>Successfully unfollow the restaurant</i>
403	<i>User authorization level is not enough</i>
404	<i>Restaurant not found</i>
409	<i>User does not followed the restaurant with this ID yet</i>

PATCH

/restaurants/{restaurant_id}/unlike/

Parameters

Try it out

Name

Description

restaurant_id * required string (path)	Unlike the restaurant providing the restaurant ID.
---	--

Responses

Response content type

application/json

Code	Description
200	<i>Successfully unliked the restaurant</i>
403	<i>User authorization level is not enough</i>
404	<i>Restaurant not found</i>
409	<i>User does not liked the restaurant with this ID yet</i>