

WORKING OF THE WHOLE PROJECT:

~ USER MANAGEMENT PAGE:

localhost/docs#/

User Management

0.0.1

0AS 3.1

[/openapi.json](#)

Application Overview:

This application is a robust user management system designed to facilitate the administration of user credentials and profiles in a secure and efficient manner. It leverages the capabilities of FastAPI to provide a high-performance, scalable API that adheres to the best practices of modern web service development.

Key Features:

- **User Authentication:** Implements OAuth2 with Password Flow to ensure secure access to the API. Users are required to authenticate using a JWT (JSON Web Token) which provides a secure and efficient means of user identification and authorization.
- **CRUD Operations:** Offers comprehensive endpoints for creating, reading, updating, and deleting user information. This includes management of user details such as email, passwords, and personal profiles.
- **Role-Based Access Control:** Enforces different access levels using a role-based mechanism that restricts certain operations to users with appropriate privileges. Supported roles include Admin, Manager, and regular Users, each with different permissions.
- **Email Integration:** Integrates with email services for account verification and notifications, enhancing the registration and password recovery processes.
- **HATEOAS (Hypermedia as the Engine of Application State):** Each response from the API includes hypermedia links to guide the client to other relevant endpoints based on the context of the current interaction, promoting discoverability and ease of navigation within the API.
- **Secure Password Handling:** Implements best practices for password security, including hashing and salting techniques, to ensure that user credentials are stored securely.
- **Error Handling:** Provides clear and informative error responses that help clients properly handle issues such as authentication failures, access violations, and data conflicts.

Security Features:

The application incorporates several security measures to protect data and ensure the integrity and confidentiality of user information:

- **Data Encryption:** Uses advanced encryption standards to secure sensitive data in transit and at rest.
- **Input Validation:** Employs rigorous validation checks to prevent SQL injection, XSS, and other common security threats.
- **Rate Limiting:** Protects against brute-force attacks by limiting the number of requests a user can make to the API within a given timeframe.

User Experience:

Designed with a focus on user experience, the API provides detailed documentation, descriptive error messages, and consistent interface patterns that make it intuitive and straightforward for developers to integrate with their applications.

This API is ideal for businesses and developers looking for a reliable and secure way to manage user authentication and authorization in their applications. It is particularly suited to environments where security and data privacy are paramount.

[API Support - Website](#)
[Send email to API Support](#)
[MIT](#)

← → ↺

localhost/docs#/

☆

k

Authorize

User Management Requires (Admin or Manager Roles)

^

GET

/users/{user_id}

Get User

PUT

/users/{user_id}

Update User

DELETE

/users/{user_id}

Delete User

POST

/users/

Create User

GET

/users/

List Users

Login and Registration

^

POST

/register/

Register

POST

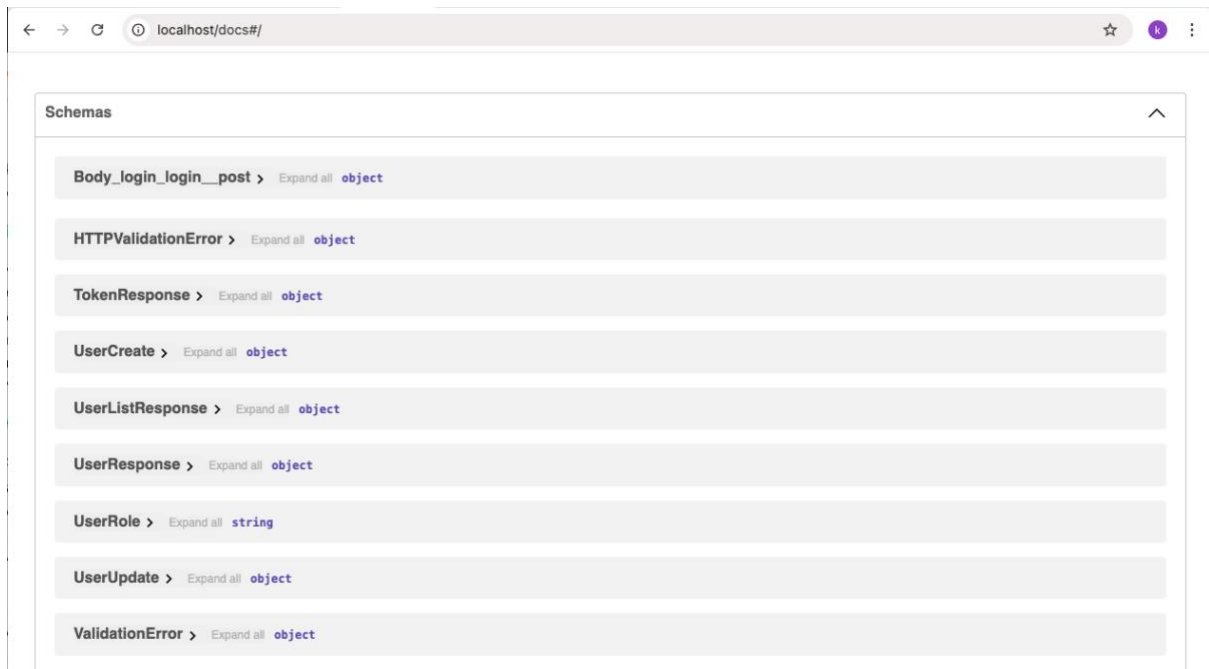
/login/

Login

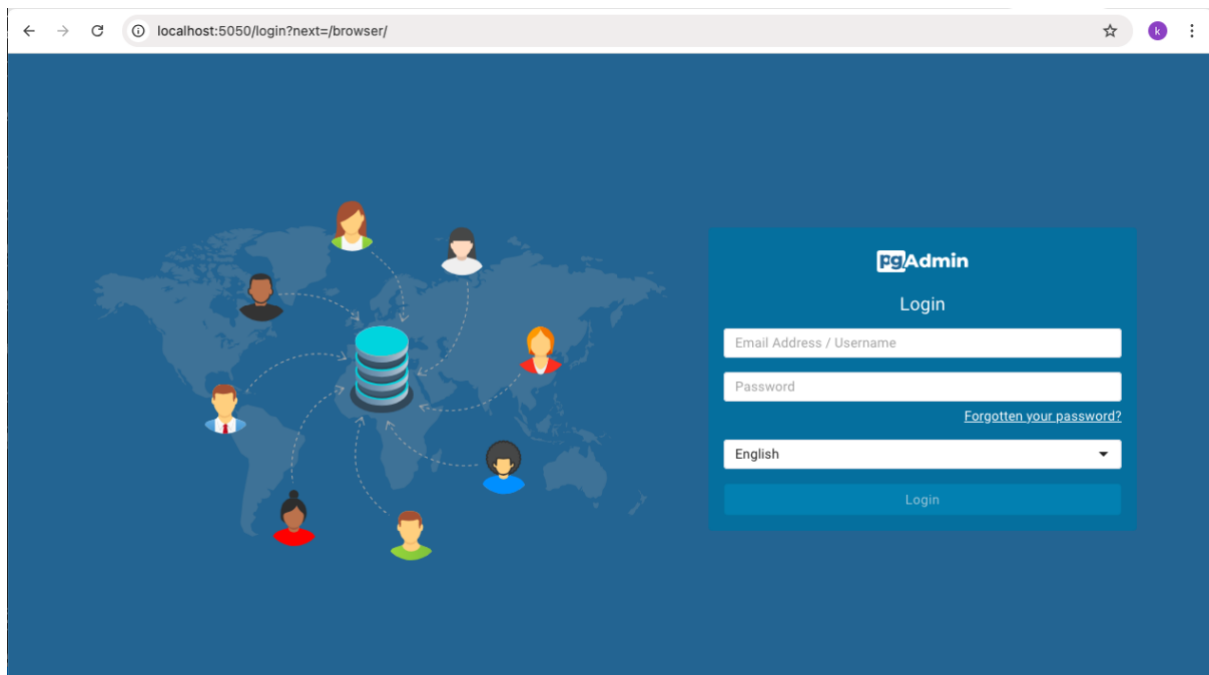
GET

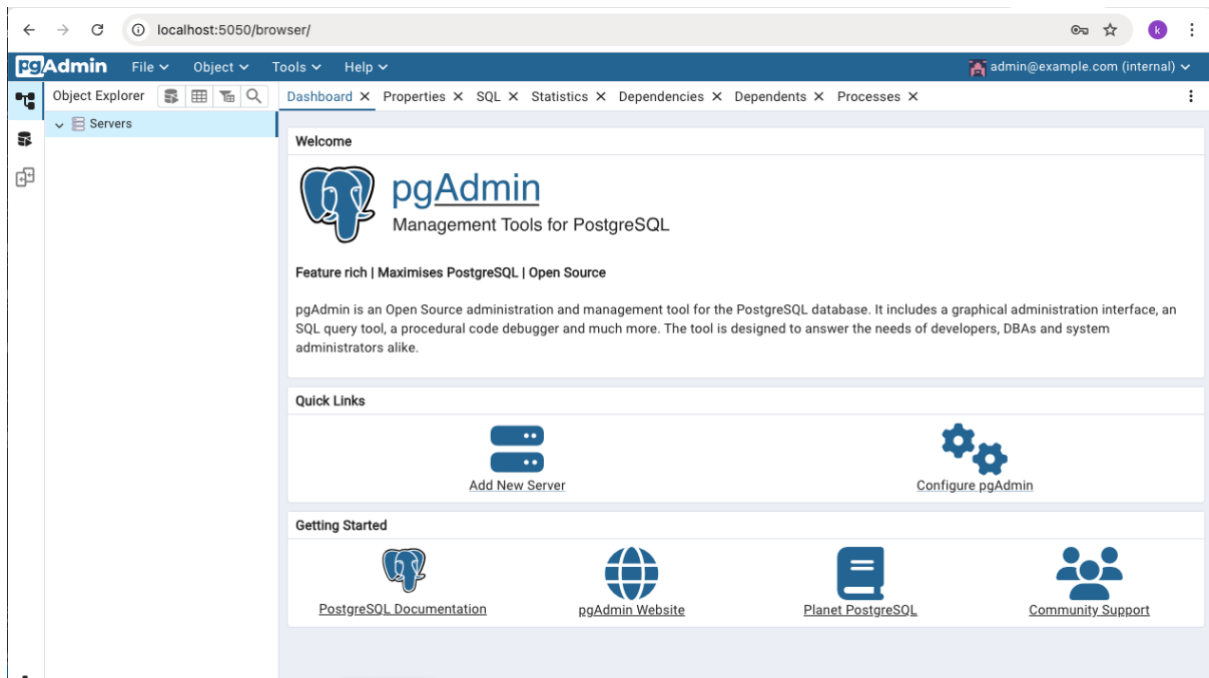
/verify-email/{user_id}/{token}

Verify Email

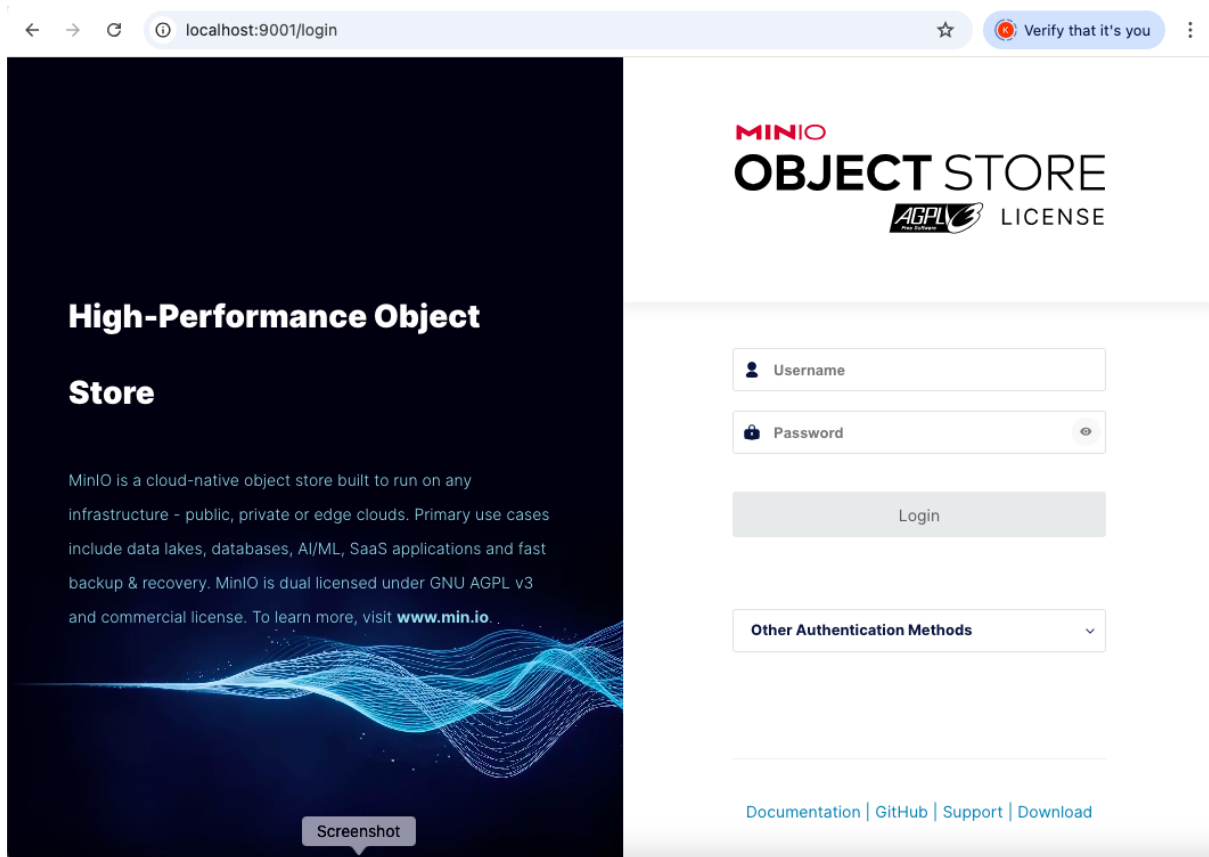


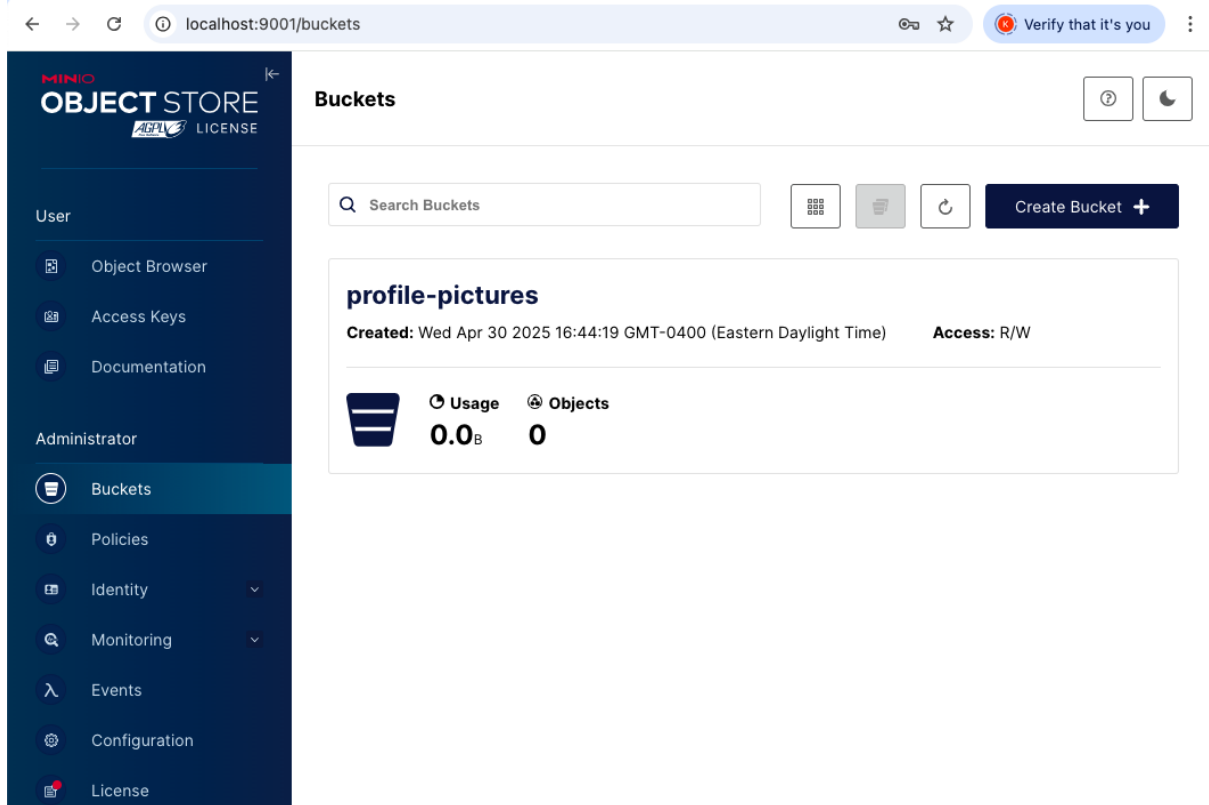
~ PGADMIN PAGE:





~ MINIO PAGE:





~ CREATION OF TABLES IN PGADMIN WITH THE COMMAND:

```
(venv) krishnasathvikaganni@krishnas-MacBook-Air FINAL_PROJECT_USER_MANAGEMENT % docker compose exec fastapi python -m alembic u
pgrade head
INFO [alembic.runtime.migration] Context impl PostgresqlImpl.
INFO [alembic.runtime.migration] Will assume transactional DDL.
INFO [alembic.runtime.migration] Running upgrade -> 25d814bc83ed, initial migration
(venv) krishnasathvikaganni@krishnas-MacBook-Air FINAL_PROJECT_USER_MANAGEMENT %
```

~ REGISTERING OF USER:

The screenshot displays a REST client interface for a POST request to `/register/`. The request body is a JSON object with user registration details. The response section shows a 200 status code with a successful JSON response containing user information and a unique ID.

Request Details:

- Method: POST
- URL: `/register/`
- Parameters: No parameters
- Request body (required): `application/json`
- Request Body (JSON):

```
{  "email": "john.doe@example.com",  "nickname": "john_doe_123",  "first_name": "John",  "last_name": "Doe",  "bio": "Experienced software developer specializing in web applications.",  "profile_picture_url": "https://example.com/profiles/john.jpg",  "linkedin_profile_url": "https://linkedin.com/in/johndoe",  "github_profile_url": "https://github.com/johndoe",  "role": "ANONYMOUS",  "password": "Secure*1234"}
```

Response Details:

- Status: 200 Successful Response
- Media type: `application/json`
- Example Value (JSON):

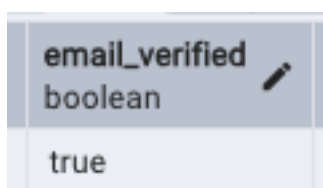
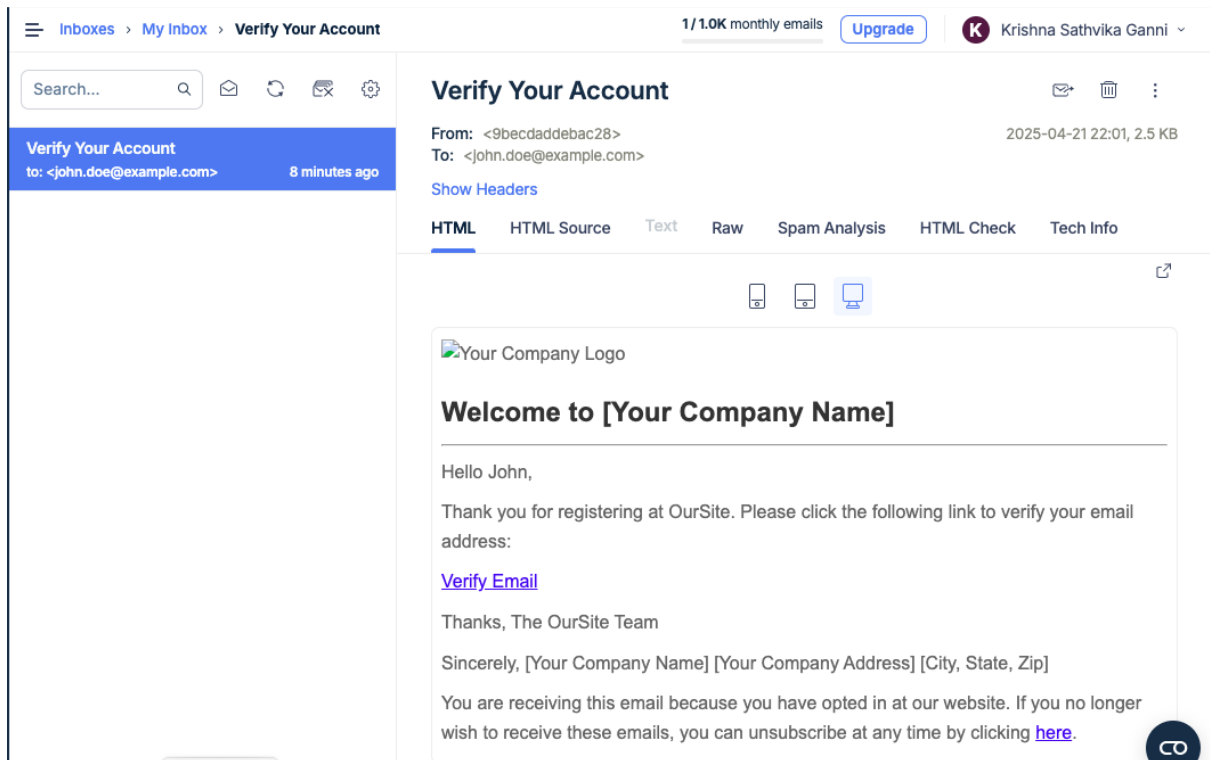
```
{  "email": "john.doe@example.com",  "nickname": "gentle_lion_252",  "first_name": "John",  "last_name": "Doe",  "bio": "Experienced software developer specializing in web applications.",  "profile_picture_url": "https://example.com/profiles/john.jpg",  "linkedin_profile_url": "https://linkedin.com/in/johndoe",  "github_profile_url": "https://github.com/johndoe",  "role": "ANONYMOUS",  "id": "33757d5c-2a53-4d28-a1e1-cf69329e1dcb",  "is_professional": false}
```

A 422 Validation Error response is also listed below the successful one.

~ REGISTERED USER STORED IN DATABASE:

	id [PK] uuid	nickname character varying (50)	email character varying (255)	first_name character varying (100)	last_name character varying (100)
1	7c64af1b-ce26-474c-885c-c6d0d4042fa8	john_doe_123	john.doe@example.com	John	Doe

~ VERIFICATION OF REGISTERED USER USING EMAIL:



~ LOGGING IN OF THE REGISTERED USER:

localhost/docs#/Login%20and%20Registration/register_register__post

POST /login/ Login

Parameters Cancel Reset

No parameters

Request body *required* application/x-www-form-urlencoded

grant_type
string | (string | null)
pattern: "password" ☐ Send empty value
password

username *required*
string john.doe@example.com

password *required*
string Secure!1234

scope
string ☐ Send empty value ☒ Send empty value
scope

client_id
string | (string | null) ☐ Send empty value
string

client_secret
string | (string | null) ☐ Send empty value
string

Execute

The screenshot shows a web browser window with the address bar displaying "localhost/docs#/Login%20and%20Registration/login_login__post". The page title is "Verify that it's you". Below the address bar, there's a section titled "Responses". Under "Curl", a curl command is shown:

```
curl -X 'POST' \
'http://localhost/login/' \
-H 'accept: application/json' \
-H 'Content-Type: application/x-www-form-urlencoded' \
-d 'email=john.doe@example.com&password=Secure*1234'
```

. Under "Request URL", the URL "http://localhost/login/" is displayed. Under "Server response", the status code "200" is shown. The "Response body" contains a JSON object:

```
{
  "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJqb2huLnRvZUBleGltcmVScSIsInJvbGUwLjBRElJTlIsImkiOiJoNTQwYTYtNDYtOGZlM0E0ZUSlTk4yZmZkZjY4ODMyZzQzIiwiaWF0IjoxNzQzMzE4MDIzfQ..xpyyt2G09tJEe22UCfSEXP030qN7ttBMQVJQMeb4",
  "token_type": "bearer"
}
```

. A "Download" button is next to the response body. Under "Response headers", several headers are listed:

```
access-control-allow-credentials: true
access-control-allow-origin: http://localhost
connection: keep-alive
content-length: 264
content-type: application/json
date: Sat, 03 May 2025 22:42:03 GMT
server: nginx/1.27.5
vary: Origin
```

~ AUTHORIZATION OF THE USER:

Available authorizations x

Scopes are used to grant an application different levels of access to data on behalf of the end user. Each API may declare one or more scopes.

API requires the following scopes. Select which ones you want to grant to Swagger UI.

OAuth2PasswordBearer (OAuth2, password)

Authorized

Token URL: /login
Flow: password
username: john.doe@example.com
password: *****
Client credentials location: basic
client_secret: *****

LogoutClose

~ PROFILE PICTURE UPLOAD OF THE REGISTERED USER:

POST /users/me/upload-profile-picture Upload Profile Picture Endpoint

Parameters

No parameters

Request body required

multipart/form-data

file required

string(\$binary)

Choose file

profile picture 1.jpg

Execute

← → ↻

localhost:9001/browser/profile-pictures/6f557612-1c2f-44a4-96c1-f093c7af6e49.jpg

☆

Verify that it's you

MINIO
OBJECT STORE
APACHE LICENSE

User

- Object Browser
- Access Keys
- Documentation

Administrator

- Buckets
- Policies
- Identity
- Monitoring
- Events
- Configuration
- License

← Object Browser

Start typing to filter objects in the bucket

profile-pictures

Created on: Wed, Apr 30 2025 23:07:12 (EDT) Access: PRIVATE 22.4 KiB - 1 Object

Rewind

Refresh

Upload

profile-pictures / 6f557612-1c2f-44a4-96c1-f093c7af6e49.jpg

Create new path

<input type="checkbox"/>	Name	Last Modified	Size
<input type="checkbox"/>	6f557612-1c2f-44a4-96c1-f093c7af6e4...	Thu, May 01 2025 23:28 (EDT)	22.4 KiB

Actions:

Download

Share

Preview

Legal hold

Retention

Tags

Inspect

Display Object Versions

Delete

Object Info

← → ↻

localhost:9001/browser/profile-pictures/6f557612-1c2f-44a4-96c1-f093c7af6e49.jpg

☆

Verify that it's you

MINIO
OBJECT STORE
APACHE LICENSE


User

- Object Browser
- Access Keys
- Documentation

Administrator

- Buckets
- Policies
- Identity
- Monitoring
- Events
- Configuration
- License

Preview - 6f557612-1c2f-44a4-96c1-f093c7af6e49.jpg



~ REGISTERING OF ANOTHER USER:

localhost/docs/#/User%20Management%20Requires%20(Admin%20or%20Manager%20Roles)/create_user_users__post

Verify that it's you

Parameters

No parameters

Request body required

application/json

```
{  "email": "smith.brown@example.com",  "nickname": "smith brown_12",  "first_name": "Smith",  "last_name": "Brown",  "bio": "Software Tester",  "profile_picture_url": "https://example.com/profiles/smith.jpg",  "linkedin_profile_url": "https://linkedin.com/in/smithbrown",  "github_profile_url": "https://github.com/smithbrown",  "role": "ANONYMOUS",  "password": "Nash#234"}
```

Execute

Responses

[illegible]

localhost/docs#/User%20Management%20Requires%20(Admin%20or%20Manager%20Roles)/create_user_users__post

Verify that it's you

Returns:

- UserResponse: The newly created user's information along with navigation links.

Parameters

No parameters

Request body required

application/json

```
{
  "email": "smith.brown@example.com",
  "nickname": "smith_brown_12",
  "first_name": "Smith",
  "last_name": "Brown",
  "bio": "Software Tester",
  "profile_picture_url": "https://example.com/profiles/smith.jpg",
  "linkedin_profile_url": "https://linkedin.com/in/smithbrown",
  "github_profile_url": "https://github.com/smithbrown",
  "role": "ANONYMOUS",
  "password": "Hash#234"
}
```

Execute

Clear

~ VERIFICATION OF ANOTHER USER:

Verify Your Account

From: <9becdaddebac28>
To: <smith.brown@example.com>

2025-05-03 13:54, 2.6 KB

Show Headers

HTML

HTML Source

Text

Raw

Spam Analysis

HTML Check

Tech Info

📧 Your Company Logo

Welcome to [Your Company Name]

Hello Smith,
Thank you for registering at OurSite. Please click the following link to verify your email address:
[Verify Email](#)
Thanks, The OurSite Team
Sincerely, [Your Company Name] [Your Company Address] [City, State, Zip]
You are receiving this email because you have opted in at our website. If you no longer wish to receive these emails, you can unsubscribe at any time by clicking [here](#).

← → ↻

localhost/verify-email/07643aec-775d-456d-96f3-c1a8e2cabece/3siblTn9Zt1YqdFNjVHng

☆

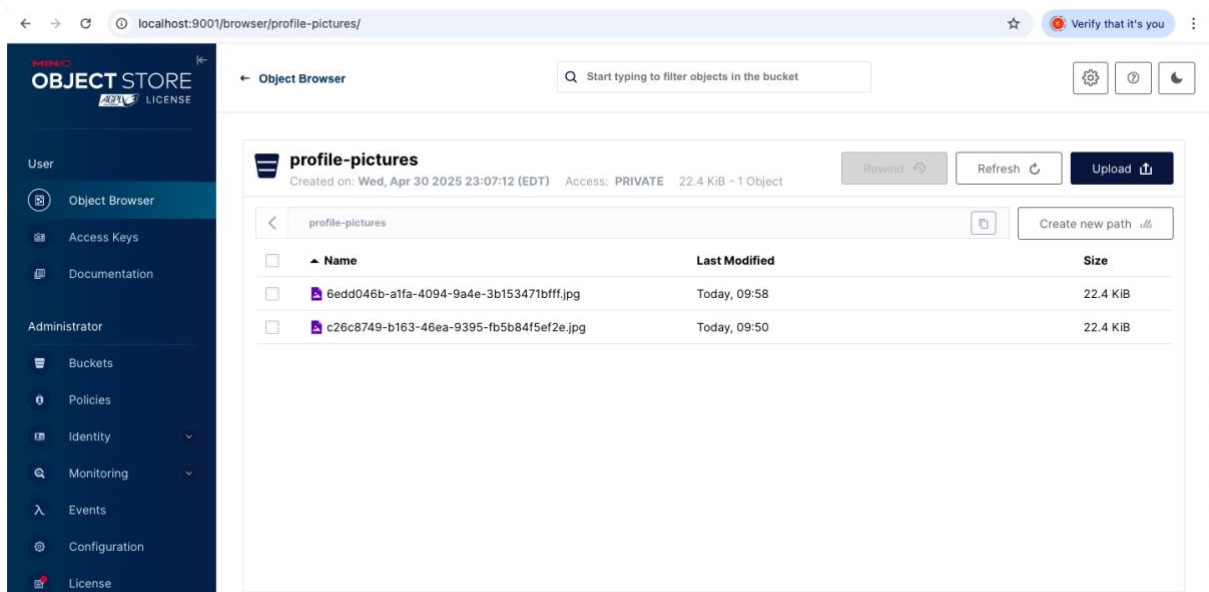
Pretty print

{"message": "Email verified successfully"}

~ REGISTERED USER STORED IN DATABASE:

	id [PK] uuid	nickname character varying (50)	email character varying (255)	first_name character varying (100)	last_name character varying (100)
1	07643aec-775d-456d-96f3-c1a8e2cabece	smith_brown_12	smith.brown@example.com	Smith	Brown
2	1badc113-0527-4d72-96d9-474266dcfa96	john_doe_123	john.doe@example.com	John	Doe

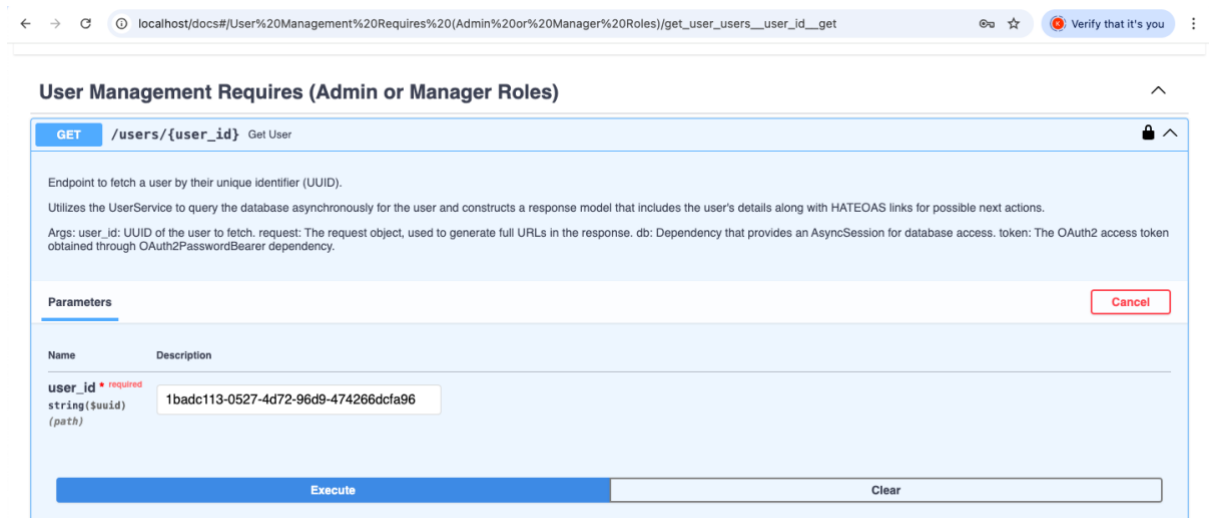
~ PROFILE PICTURE UPLOAD FOR ANOTHER USER:



The screenshot shows the MinIO Object Browser interface. The left sidebar contains navigation links for User, Administrator, and various system settings. The main area displays the 'profile-pictures' bucket, which was created on Wed, Apr 30 2025 23:07:12 (EDT) and is currently PRIVATE with 22.4 KiB of data and 1 object. A table lists the objects in the bucket:

Name	Last Modified	Size
6edd046b-a1fa-4094-9a4e-3b153471bfff.jpg	Today, 09:58	22.4 KiB
c26c8749-b163-46ea-9395-fb5b84f5ef2e.jpg	Today, 09:50	22.4 KiB

~ GETTING THE USER:

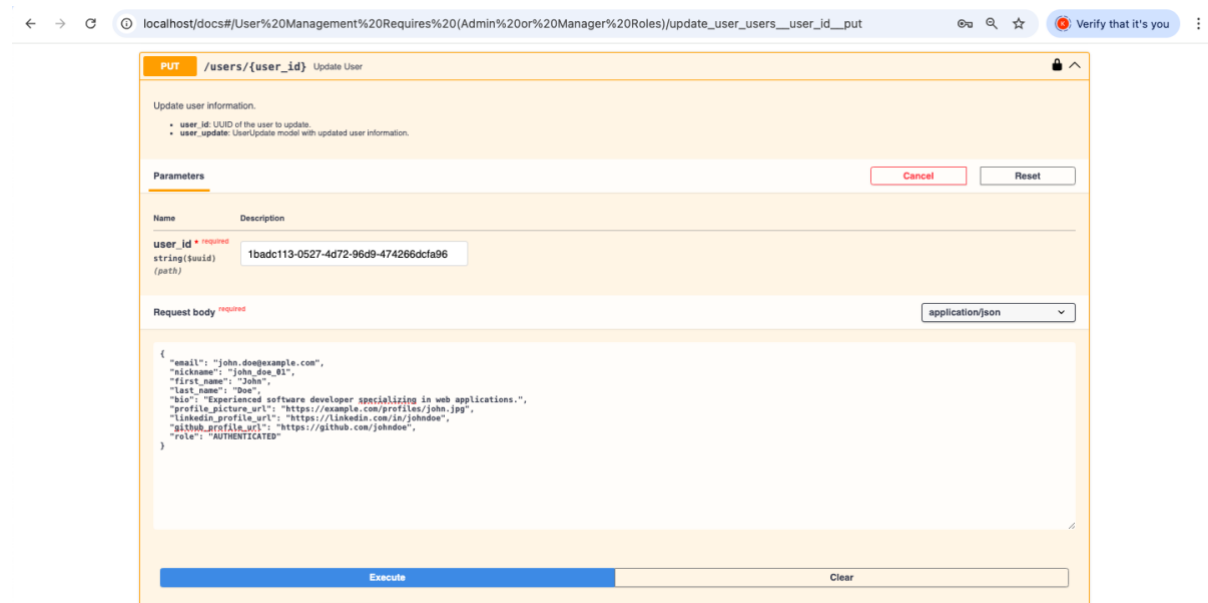


The screenshot shows the Swagger UI for the 'get_user_users__user_id__get' endpoint. The endpoint is a GET request to '/users/{user_id}' and is used to fetch a user by their unique identifier (UUID). The description states: 'Utilizes the UserService to query the database asynchronously for the user and constructs a response model that includes the user's details along with HATEOAS links for possible next actions.' The arguments are: 'user_id: UUID of the user to fetch. request: The request object, used to generate full URLs in the response. db: Dependency that provides an AsyncSession for database access. token: The OAuth2 access token obtained through OAuth2PasswordBearer dependency.'

Parameters

Name	Description
user_id required string(\$uuid) (path)	1badc113-0527-4d72-96d9-474266dcfa96

Buttons: Execute, Clear, Cancel





localhost/docs#/User%20Management%20Requires%20(Admin%20or%20Manager%20Roles)/delete_user_users__user_id_delete

Verify that it's you

Responses

Curl

```
curl -X 'DELETE' \
'http://localhost/users/07643aec-775d-456d-96f3-cla8e2cabece' \
-H 'accept: */*' \
-H 'Authorization: Bearer eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJzdWIiOiJqb2h1LnRvZGUtYmVudSIsInJvbGVuIjoiYmVWRIRUSUNBEVEiwiaWF0IjEwMTExMkYyZXMyOwNTI3LTRhNzItOTZkOS00ZnQy'<img alt="Copy icon" data-bbox="955 385 970 405"/>
```

Request URL

```
http://localhost/users/07643aec-775d-456d-96f3-cla8e2cabece
```

Server response

Code

Details

204

Response headers

```
access-control-allow-credentials: true
access-control-allow-origin: http://localhost
connection: keep-alive
date: Sat, 03 May 2025 14:12:48 GMT
server: nginx/1.27.5
vary: Origin
```

Responses

	id [PK] uuid	nickname character varying (50)	email character varying (255)	first_name character varying (100)	last_name character varying (100)	bio character varying (500)
1	1badc113-0527-4d72-96d9-474266dcfa96	john_doe_01	john.doe@example.com	John	Doe	Experienced software developer spe