**LEMBAR JAWABAN**

**UJIAN AKHIR SEMESTER**

**SEMESTER GANJIL TAHUN AJARAN 2023/2024**

Mata Kuliah : Pemrograman PL/SQL

Kelas : IT 302

Prodi : Informatika PJJ S1

Nama Mahasiswa : Dasdo Yessa

NIM : 210401010176

Dosen : Abdul Azzam Ajhari, S.Kom., M.Kom

### **Dokumentasi Proyek dengan SDLC Waterfall**

1. **Identifikasi Kebutuhan**

**Kebutuhan Fungsional:**

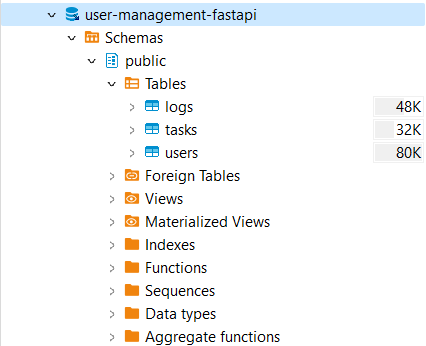
* + Mengelola informasi pengguna seperti username, email, nama lengkap, dan nomor telepon serta taks yang pengguna.
  + Menyediakan operasi CRUD (Create, Read, Update, Delete) untuk entitas pengguna dan task.
  + Melakukan autentikasi pengguna, menyimpan task pengguna, dan menghasilkan token akses.

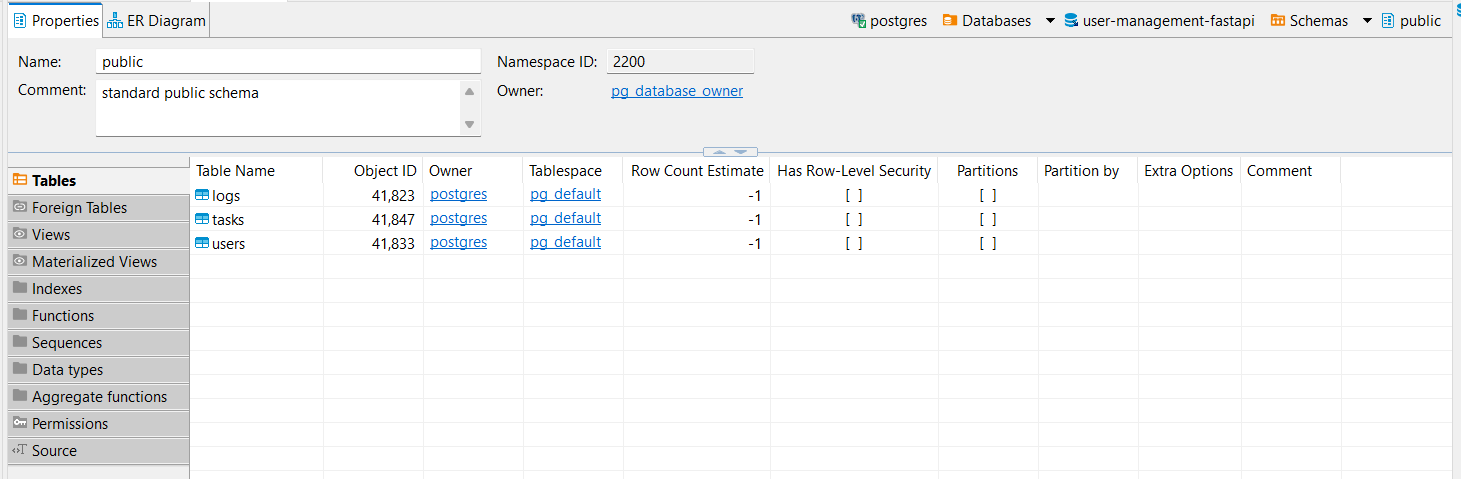
**Kebutuhan Non-Fungsional:**

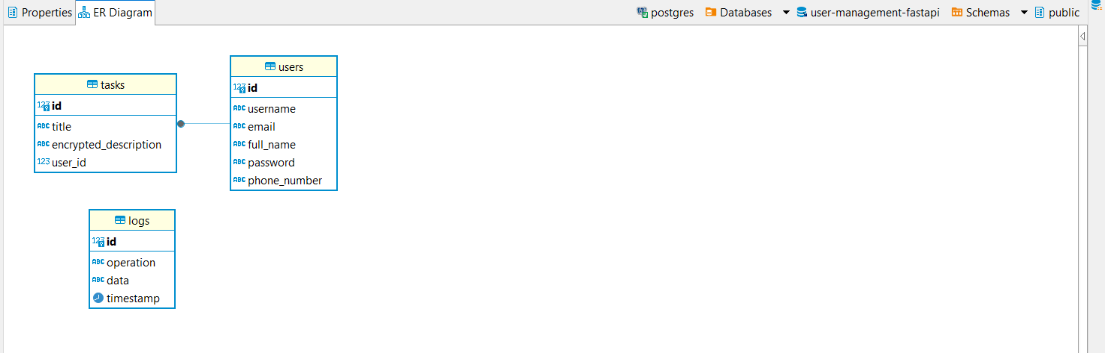
* + Keamanan: Mengimplementasikan enkripsi data untuk data sensitif seperti password.
  + Kinerja: Menangani permintaan dengan cepat menggunakan FastAPI.

1. **Desain**

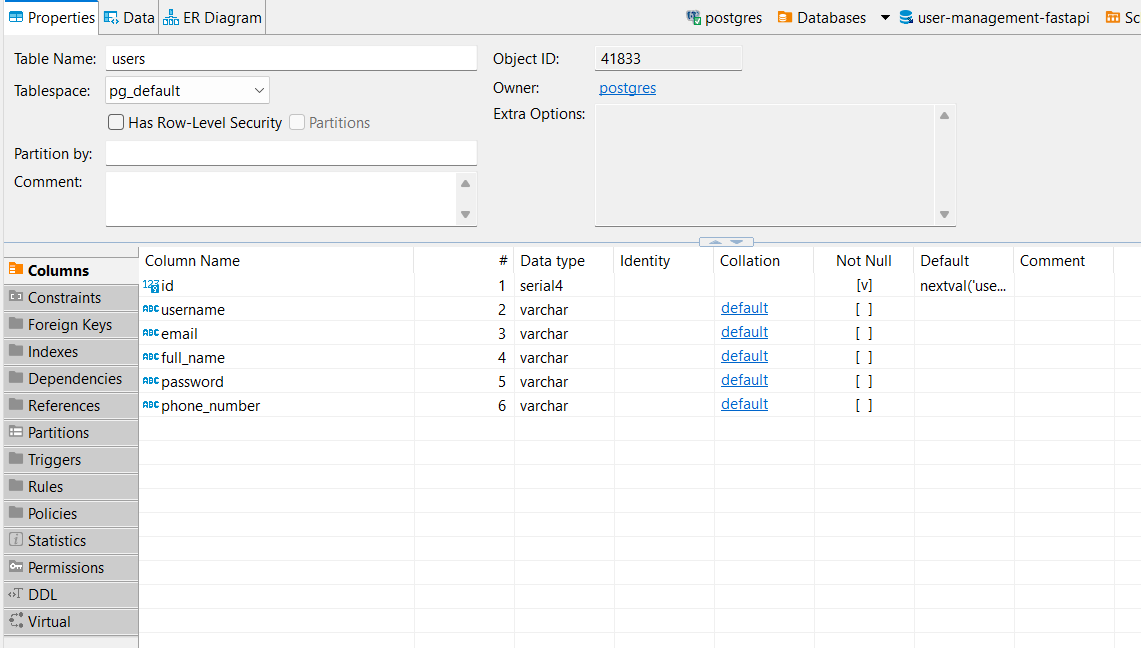
**Struktur Basis Data:**



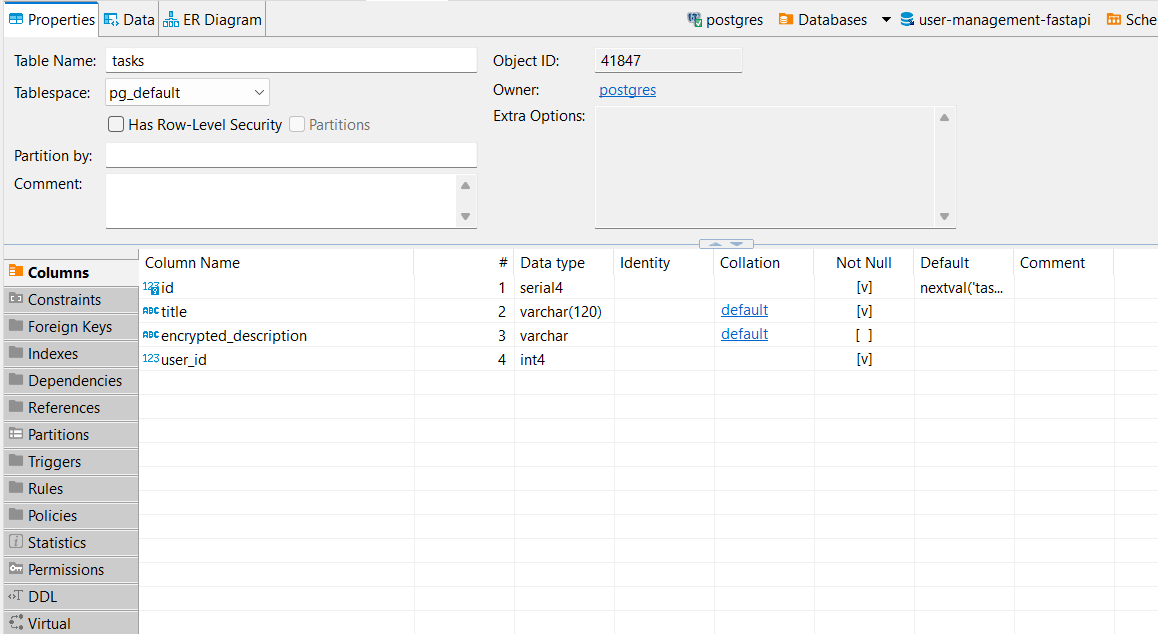




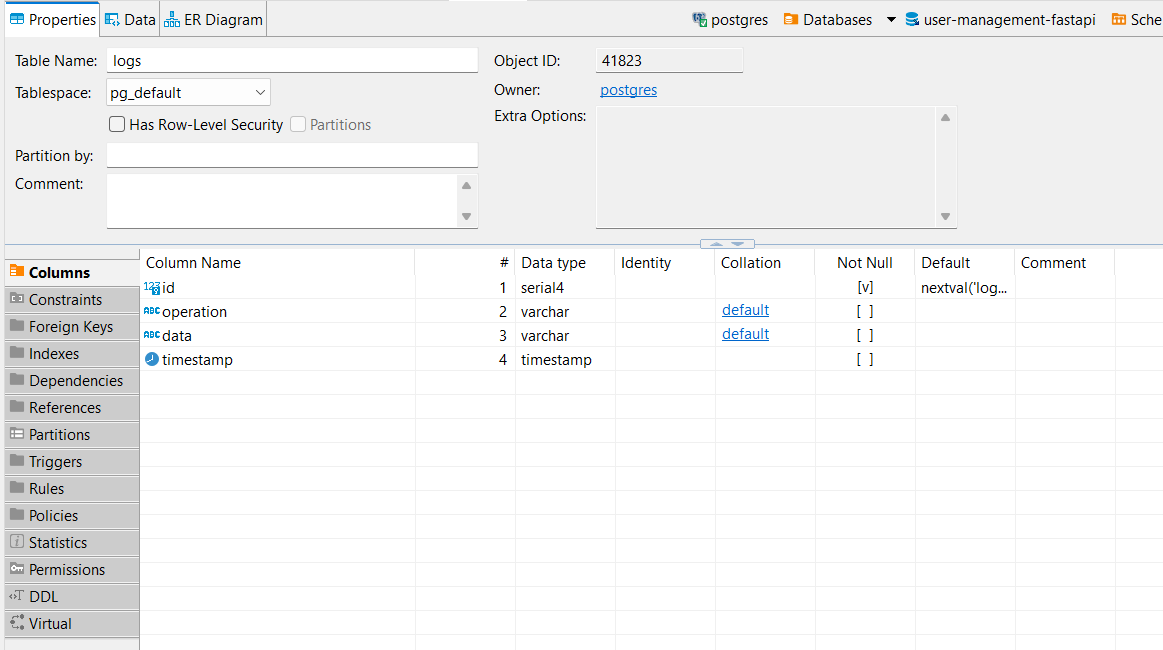
* + Tabel **users** untuk menyimpan informasi pengguna.



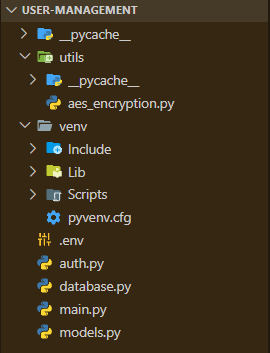
* + Tabel **tasks** untuk menyimpan informasi taks yang dibuat pengguna.



* + Tabel **logs** untuk mencatat operasi CRUD yang dilakukan pada entitas pengguna.



**Arsitektur Aplikasi:**



Berikut merupakan direktori folder dari project UAS ini. Terdiri dari empat file utama yaitu main, database, auth, dan model serta terdapat folder utils yang berisi file eas\_encryption. Project ini microservice ini dibagun menggunakan framework FastAPI yang diinstall dalam virtual env python.

* + Menggunakan FastAPI untuk membuat API RESTful.

FastAPI adalah kerangka kerja web modern untuk Python yang memungkinkan pembuatan API RESTful dengan cepat dan mudah. Dengan FastAPI, Anda dapat dengan cepat mendefinisikan endpoint-endpoint API dan mengatasi berbagai tugas web, seperti validasi data, dokumentasi otomatis, dan penanganan permintaan secara efisien.

* + Menggunakan SQLAlchemy untuk mengakses dan mengelola basis data.

FastAPI adalah kerangka kerja web modern untuk Python yang memungkinkan pembuatan API RESTful dengan cepat dan mudah. Dengan FastAPI, Anda dapat dengan cepat mendefinisikan endpoint-endpoint API dan mengatasi berbagai tugas web, seperti validasi data, dokumentasi otomatis, dan penanganan permintaan secara efisien.

* + Implementasi autentikasi menggunakan JWT (JSON Web Tokens).

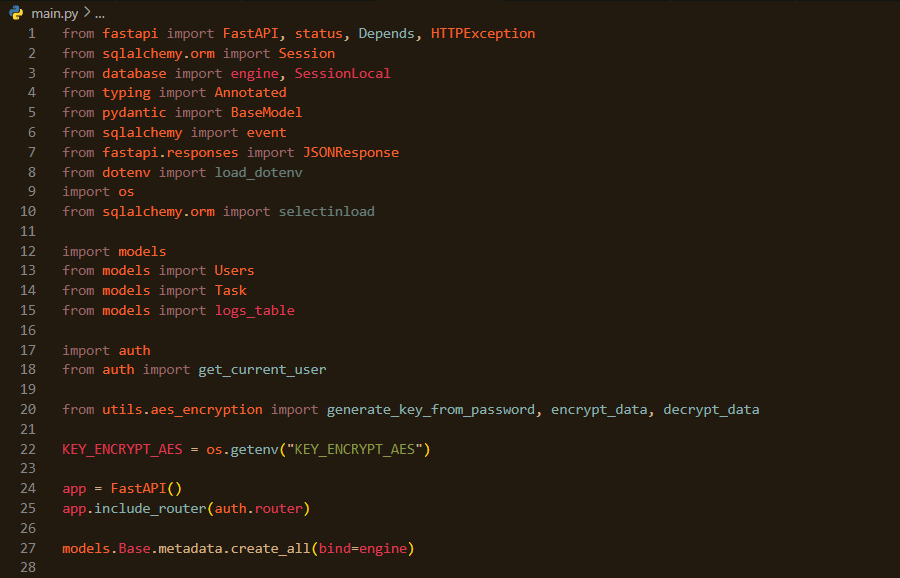
JWT adalah standar terbuka (RFC 7519) untuk mentransfer klaim antara dua pihak dalam bentuk objek JSON yang aman. Dalam konteks autentikasi, JWT sering digunakan untuk membuat token akses yang dapat digunakan oleh klien untuk mengotentikasi diri dan mengakses sumber daya yang dilindungi oleh server. Implementasi JWT dalam proyek ini melibatkan beberapa langkah, termasuk:

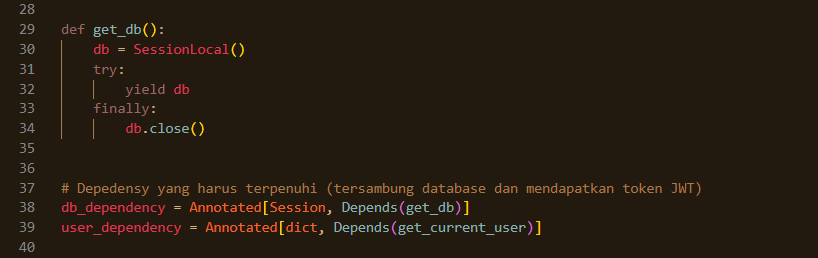
* Generasi token akses saat pengguna berhasil masuk.
* Validasi token akses yang diberikan oleh klien untuk mengakses sumber daya yang dilindungi.
* Menerapkan mekanisme penyandian dan penandatanganan untuk memastikan keamanan token.

1. **Implementasi**

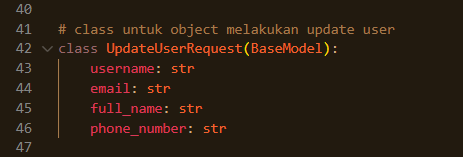
**File Main (main.py):**

* + Mendefinisikan endpoint-endpoint API untuk operasi read, update, dan delete users dan create, update, read, delete tasks.
  + Menggunakan SQLAlchemy event untuk mencatat log operasi create update delete.

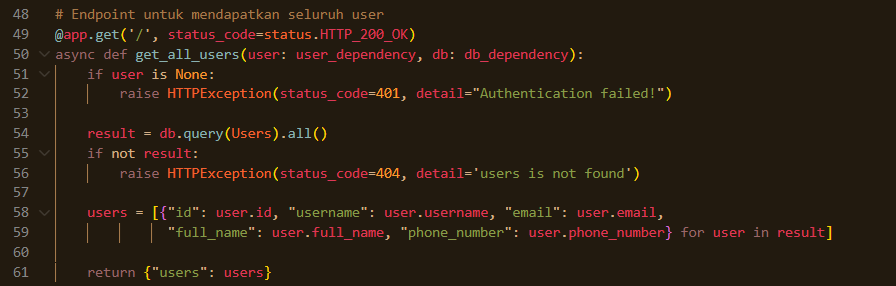




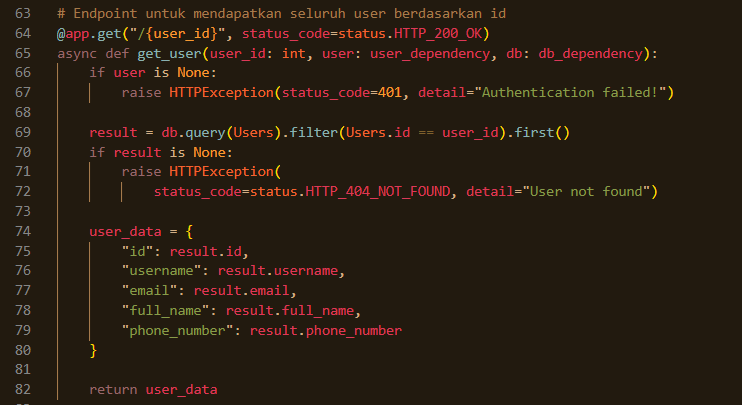
**Endpoint users**



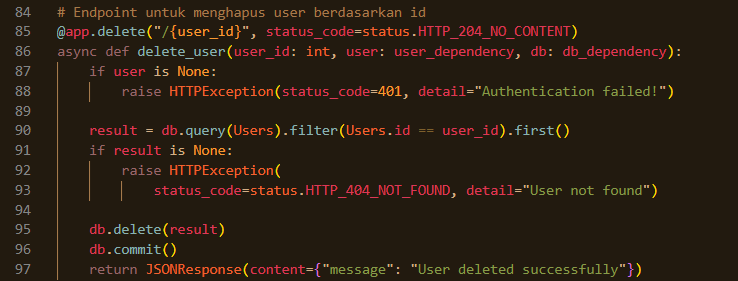
**Endpoint untuk mendapatkan seluruh user**



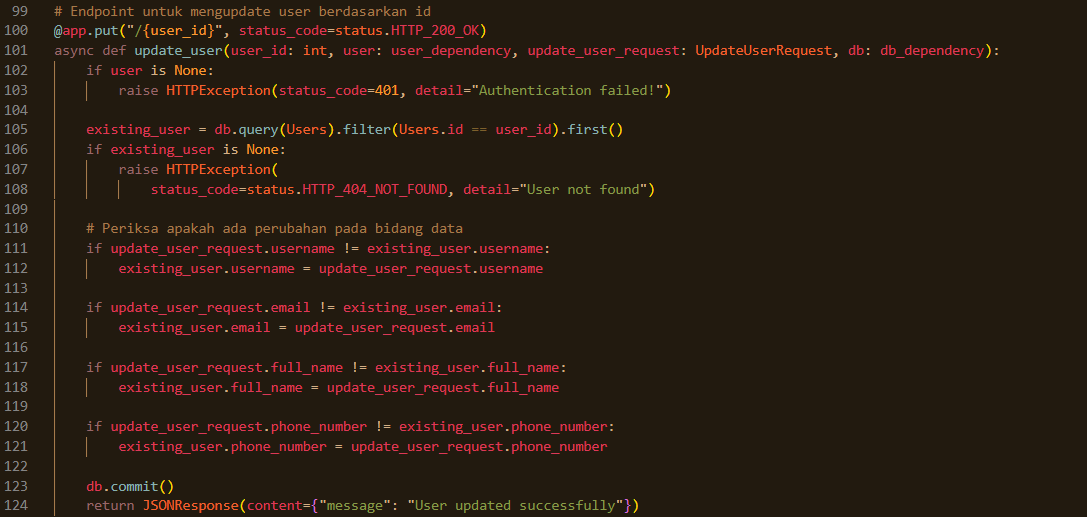
**Endpoint untuk mendapatkan seluruh user berdasarkan id**



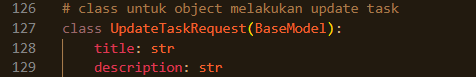
**Endpoint untuk menghapus user berdasarkan id**



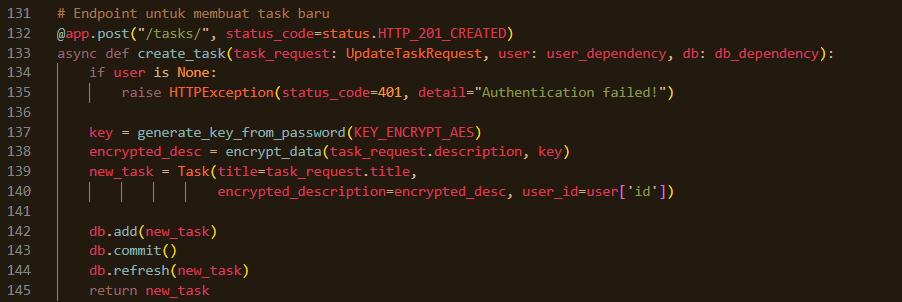
**Endpoint untuk mengupdate user berdasarkan id**



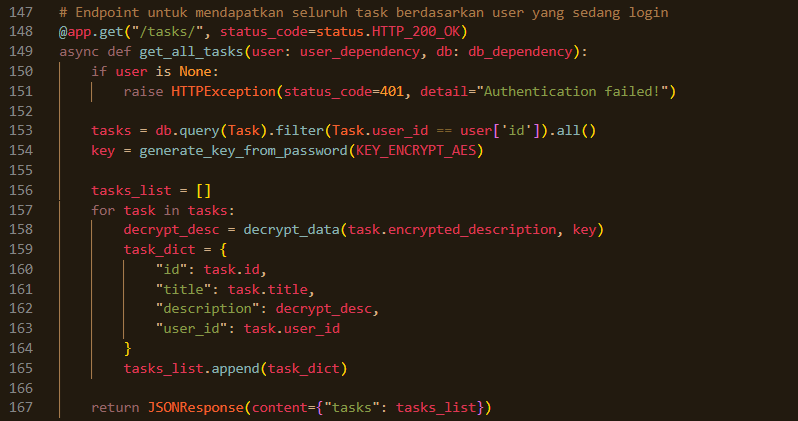
**Endpoint Tasks**



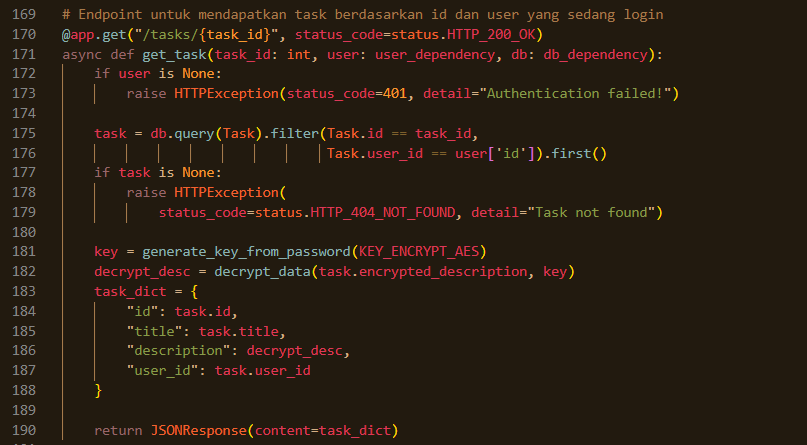
**Endpoint untuk membuat task baru user yang sedang login**



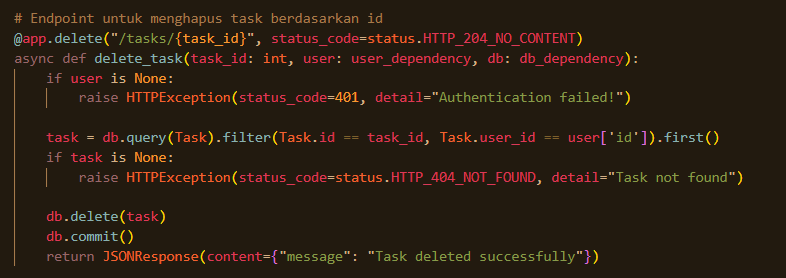
**Endpoint untuk mendapatkan seluruh task berdasarkan user yang sedang login**



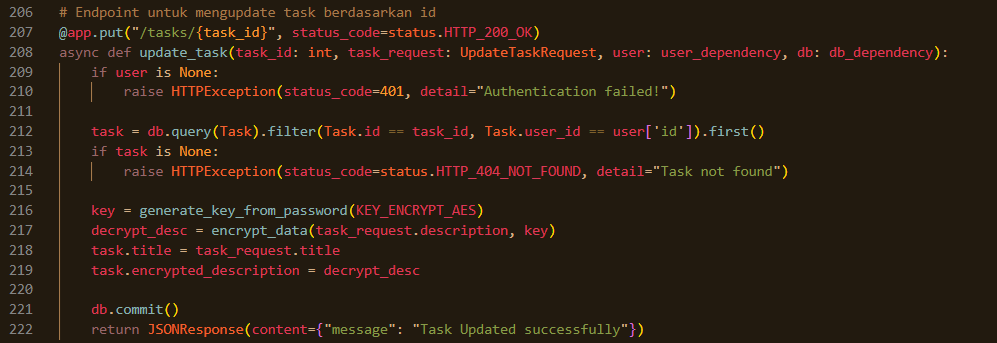
**Endpoint untuk mendapatkan task berdasarkan id dan user yang sedang login**



**Endpoint untuk menghapus task berdasarkan id**

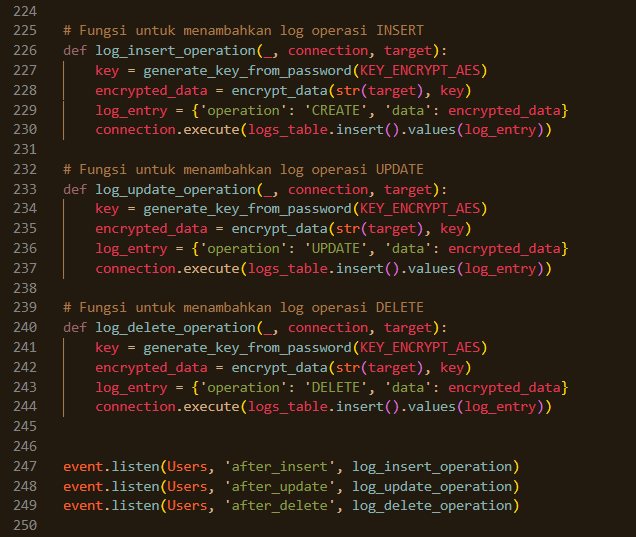


**Endpoint untuk mengupdate task berdasarkan id**



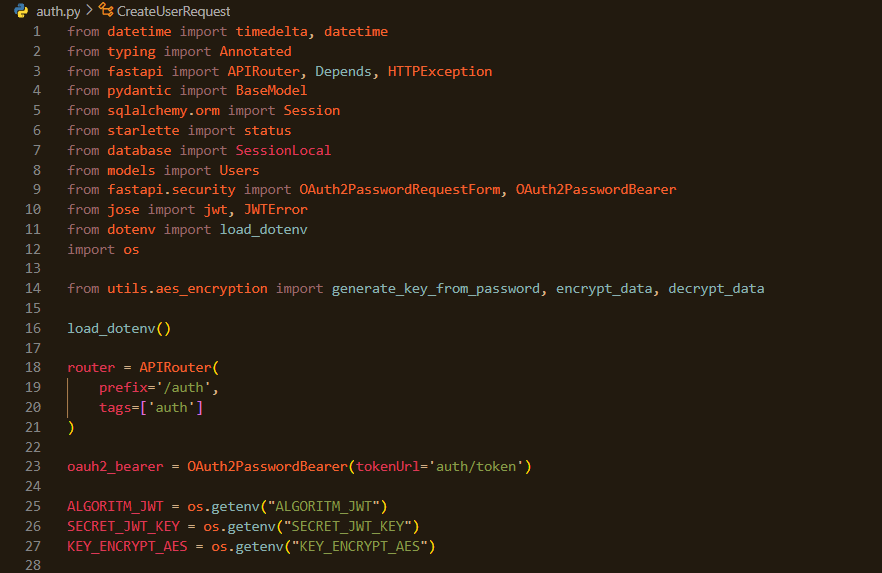
**LOG Function**

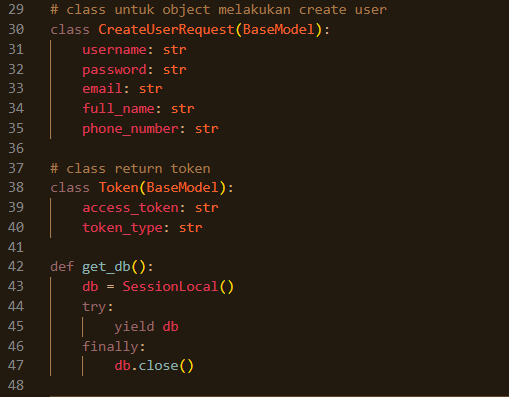
**Fungsi untuk menambahkan log operasi Update, Create, dan Delete**



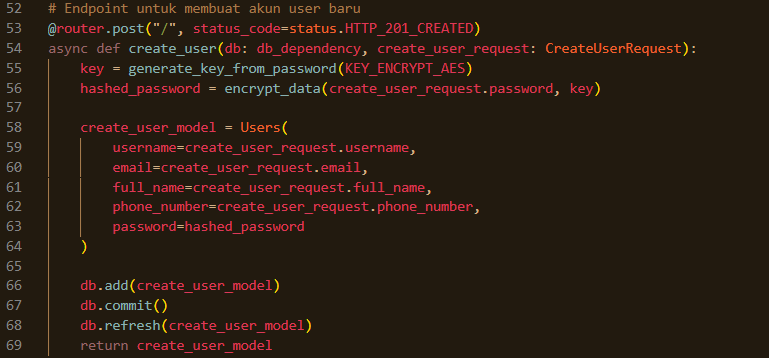
**File Autentikasi (auth.py):**

* + Mengelola proses autentikasi pengguna menggunakan JWT.
  + Mengenkripsi dan mendekripsi password pengguna sebelum disimpan ke basis data.

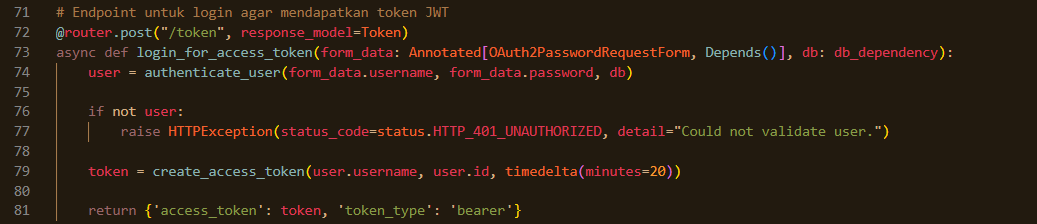




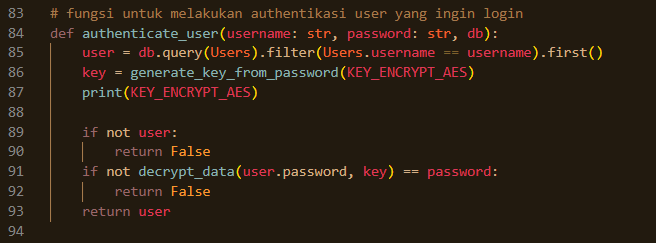
**Endpoint untuk membuat akun user baru**



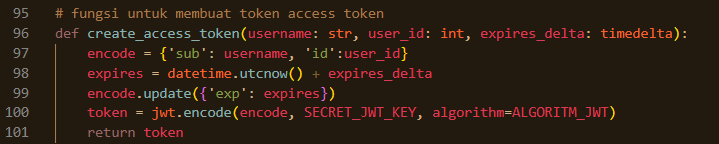
**Endpoint untuk login agar mendapatkan token JWT**



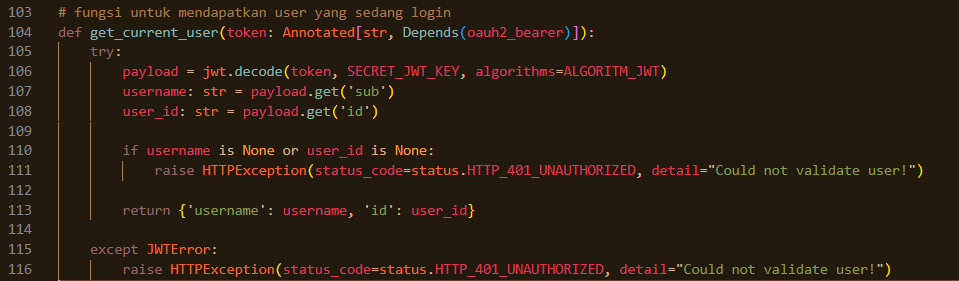
**Fungsi untuk melakukan authentikasi user yang ingin login**



**Fungsi untuk membuat token access token**

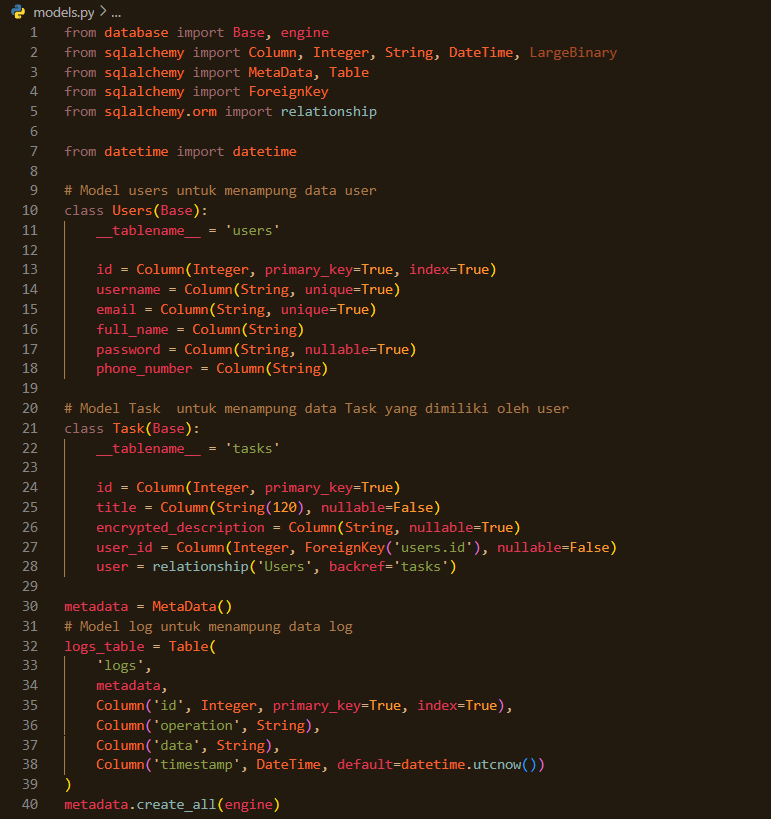


**Fungsi untuk mendapatkan user yang sadang login**



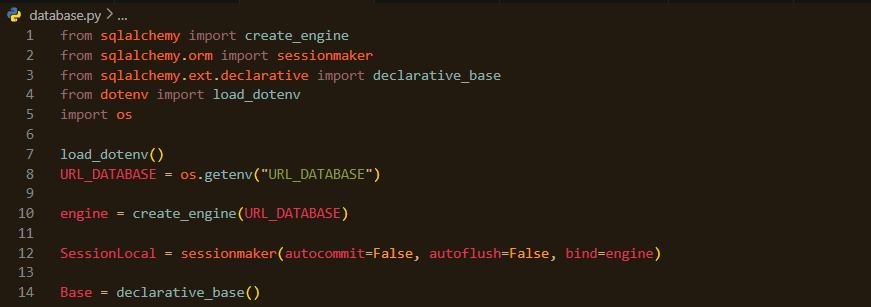
**File Model (models.py):**

* + Mendefinisikan struktur basis data menggunakan SQLAlchemy ORM.



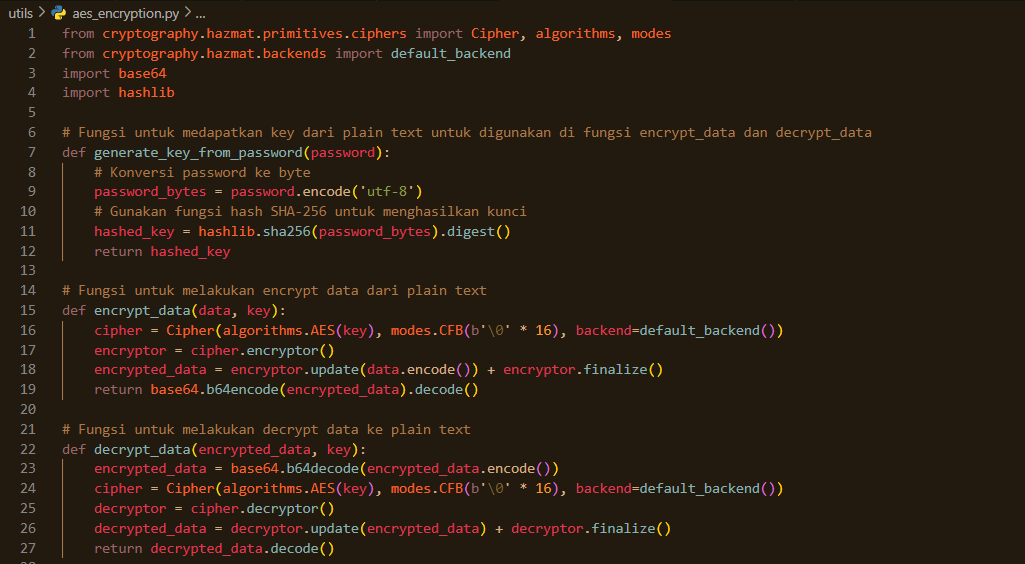
**File Koneksi Basis Data (database.py):**

* + Mengatur koneksi basis data menggunakan SQLAlchemy Engine dan Session.

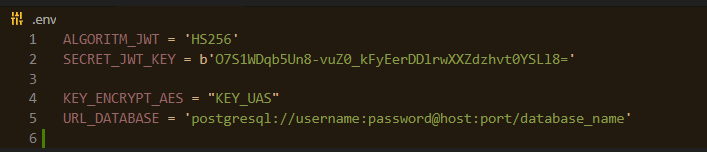


**File Enkripsi AES-128/256/512 (utils/aes\_encryption.py):**

* + Melakukan tugas enkripsi dekripsi plain text untuk **table users kolom password**, **table tasks kolom encrypted\_description,** dan **table logs kolom data**.



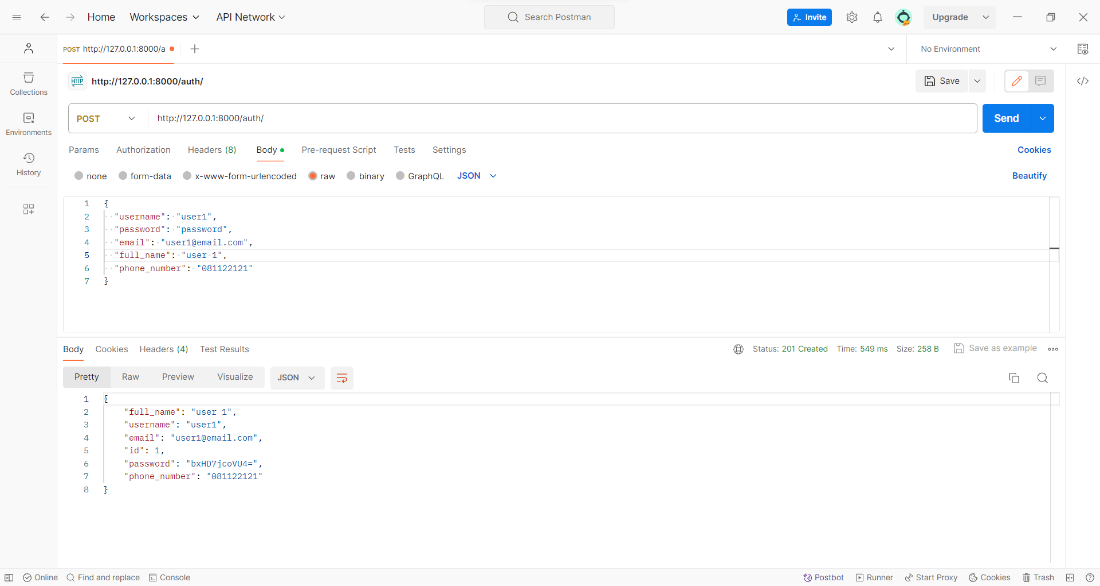
**File .env:**

****

1. **3Pengujian**

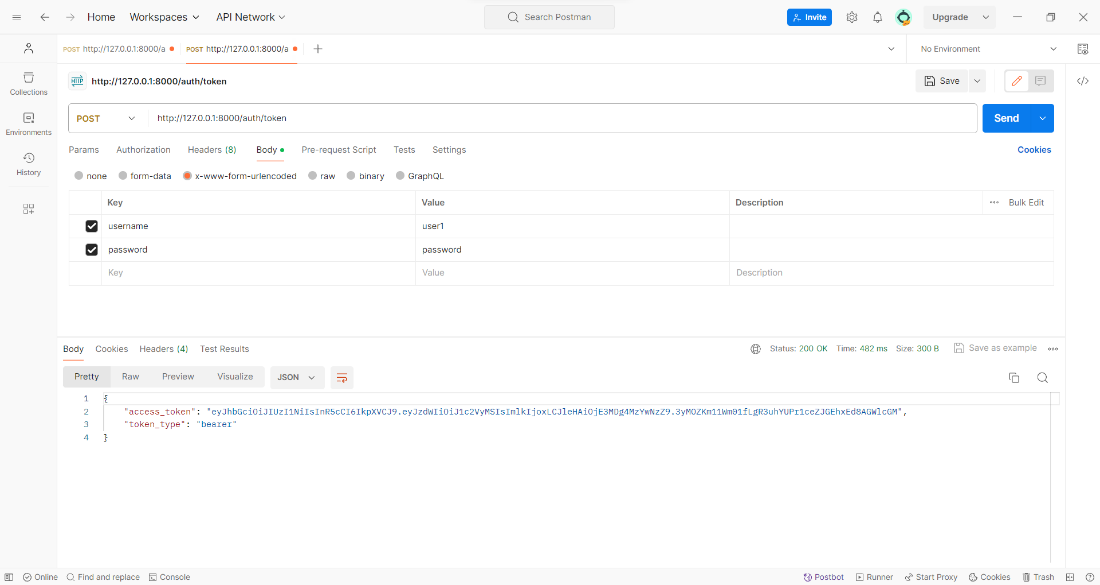
**Pengujian Menggunakan Postman**

**Create user:**



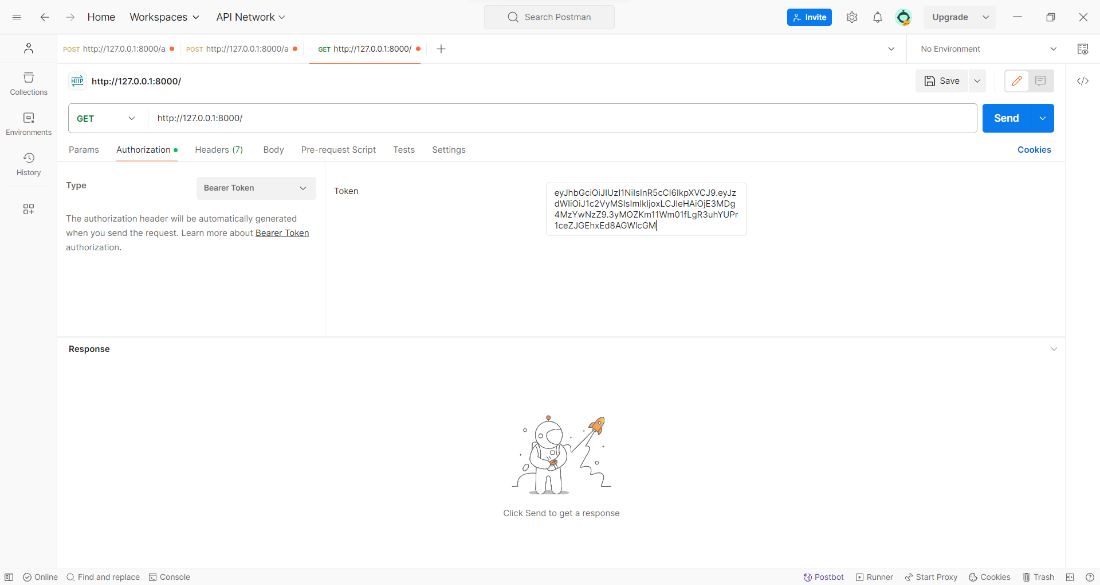
Lakukan pembuatan akun user seperti gambar diatas menggunakan aplikasi postman dengan hit endpoint “[http://127.0.0.1:8000/auth/”](http://127.0.0.1:8000/auth/) method post. Jika berhasil, response akan berupa status “201 Created” dan mendapatkan kembalian berupa data akun user yang dibuat.

**Login:**



Hit ke endpoint “[http://127.0.0.1:8000/auth/token”](http://127.0.0.1:8000/auth/token) method post, dengan cara memasukkan username dan password pada tab body seperti gambar diatas. Jika berhasil login maka akan mendapatkan kembalian berupa JSON “access token” dan ”token\_type” yang akan digunakan sebagai authentikasi ke method read, update, dan delete user.

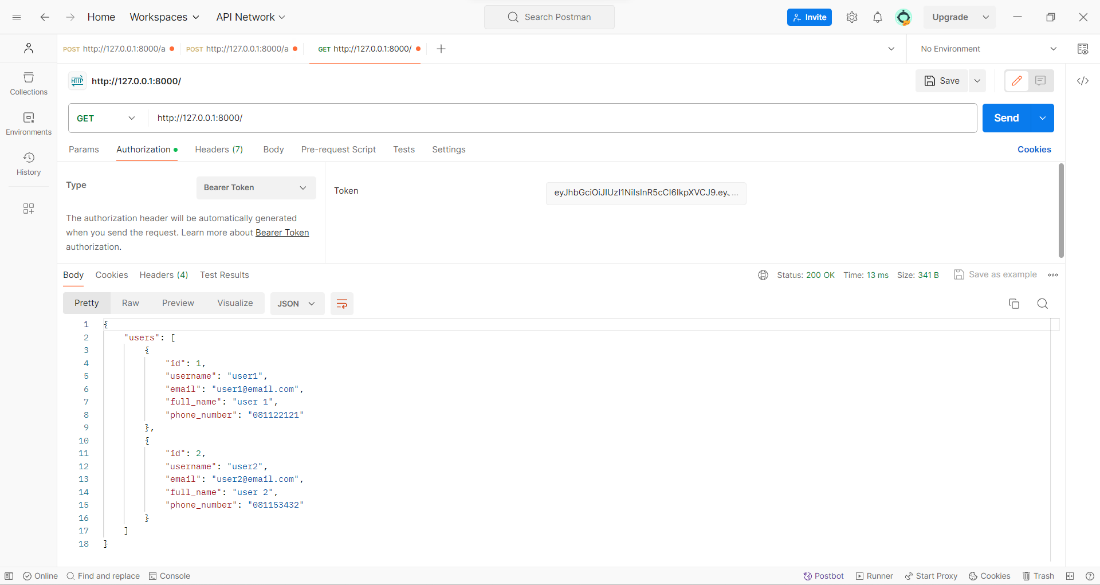
**Memasukkan Token Access:**



Setelah berhasil melakukan proses login dan mendatkan access token. Lanjutkan dengan memasukkan access token ke tab authorization lalu pilih Bearer Token sesuai dengan yang sudah kita siapkan. Masukan access token pada form agar dapat mengakses method read, update, dan delete table user maupun create, read, update, dan delete table task.

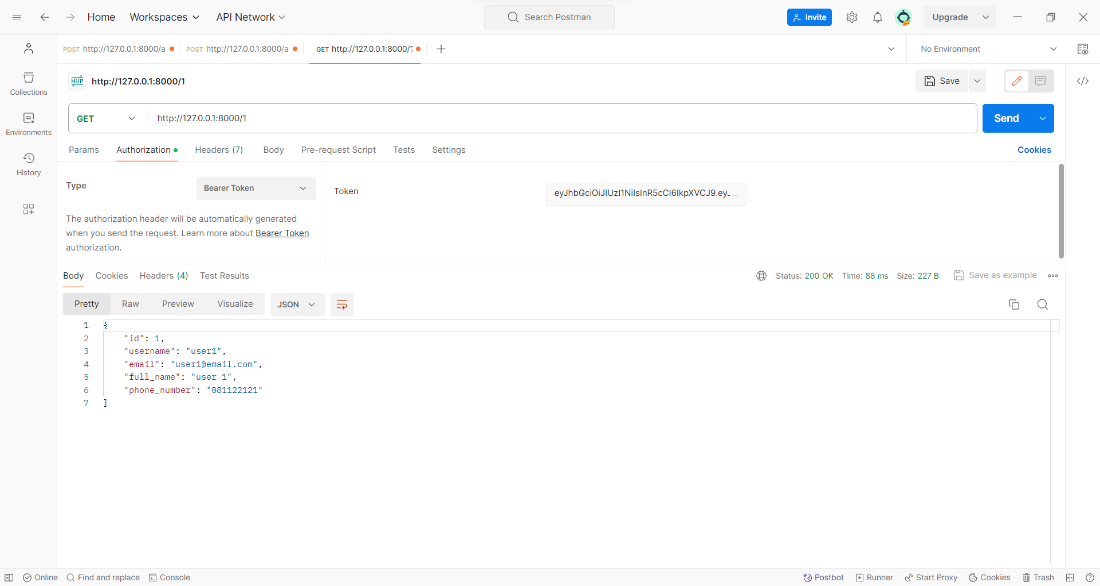
Jika kita tidak memasukkan access token pada tab authorization, maka akan diberikan kembalian json berupa “Not authenticated” dengan status 401 Unauthorized ketika hendak mengakses method read, update, dan delete user.

**View All User:**



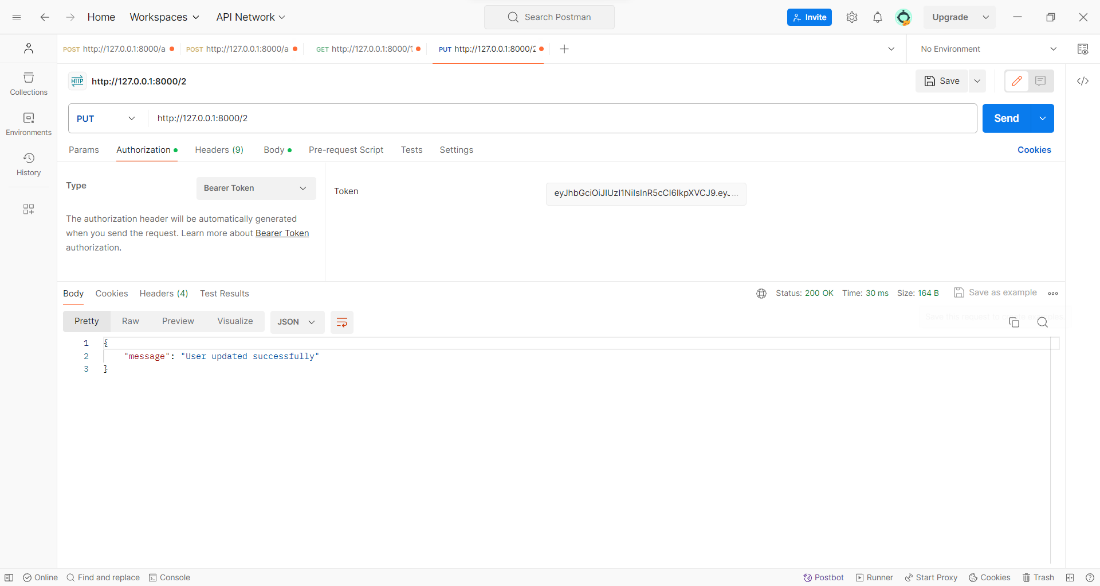
Hit endpoint <http://127.0.0.1:8000/> dengan method GET untuk mendapatkan kembalian berupa JSON seluruh akun user dengan status 200 OK.

**View User by Id:**

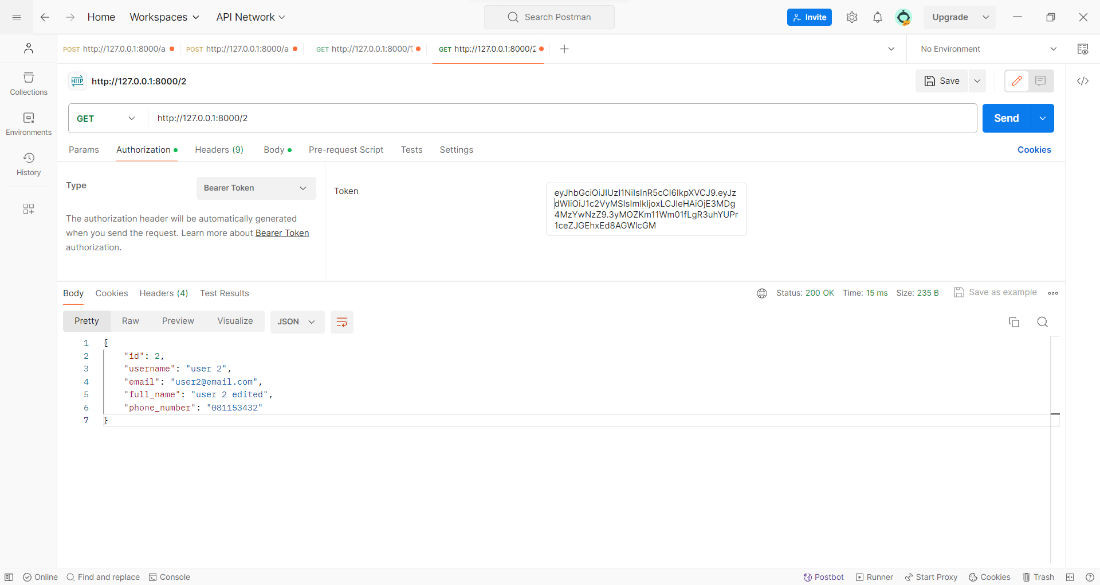


Hit endpoint [http://127.0.0.1:8000/{user\_id}](http://127.0.0.1:8000/%7buser_id%7d) dengan method GET untuk mendapatkan kembalian JSON akun user berdasarkan user id.

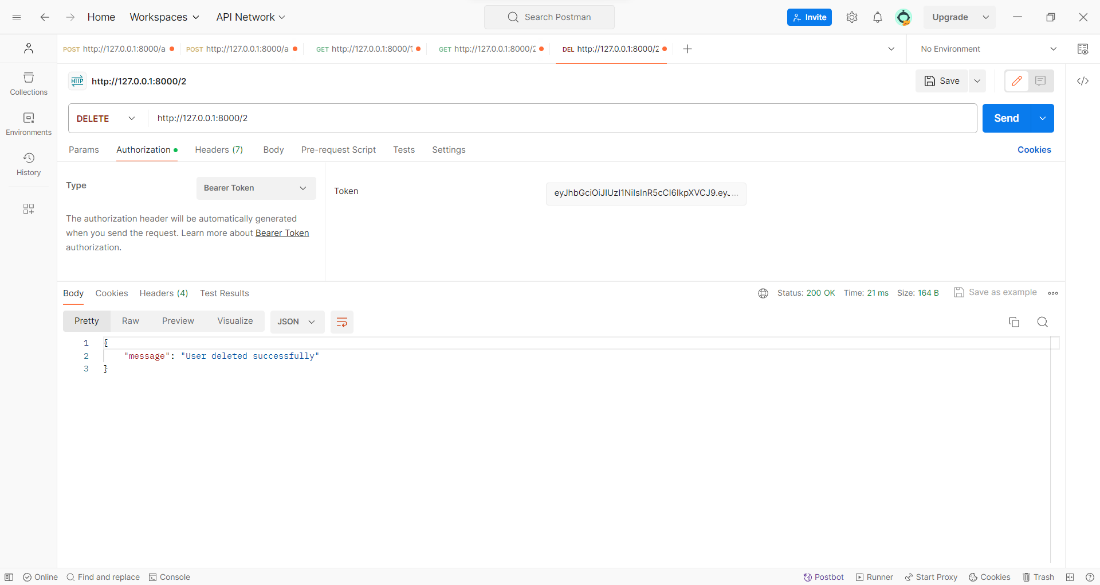
**Update User:**



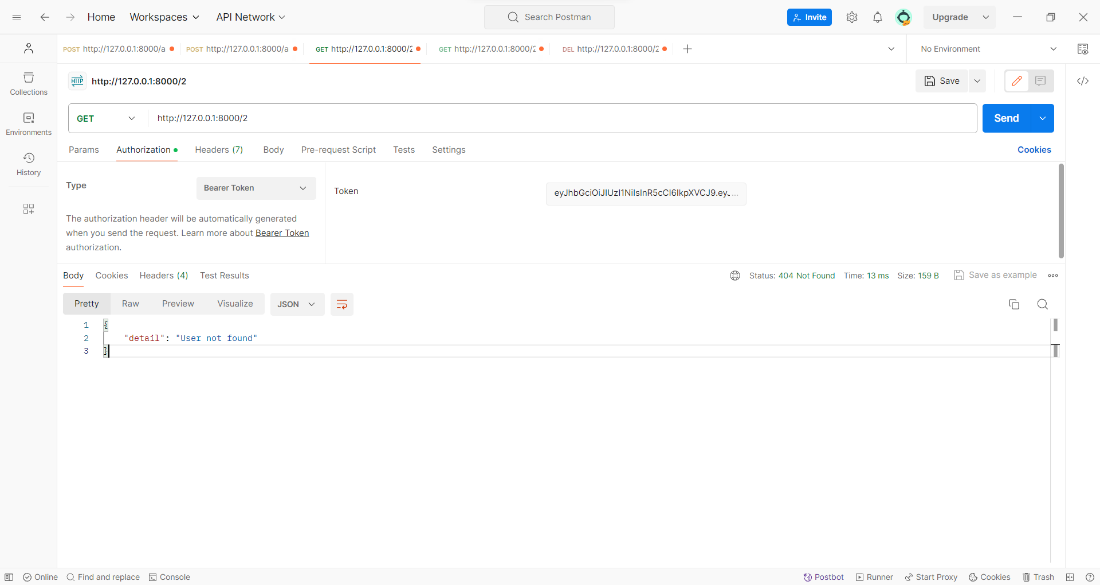
Hit endpoint [http://127.0.0.1:8000/{user\_id}](http://127.0.0.1:8000/%7buser_id%7d) dengan method PUT untuk melakukan update user. Ketika berhasil akan mengembalikan response berupa pesan “User updated successfully” dengan status 200 OK. Cek kembali apakah sudah terupdate dengan hit [http://127.0.0.1:8000/{user\_id}](http://127.0.0.1:8000/%7buser_id%7d) dengan method GET.



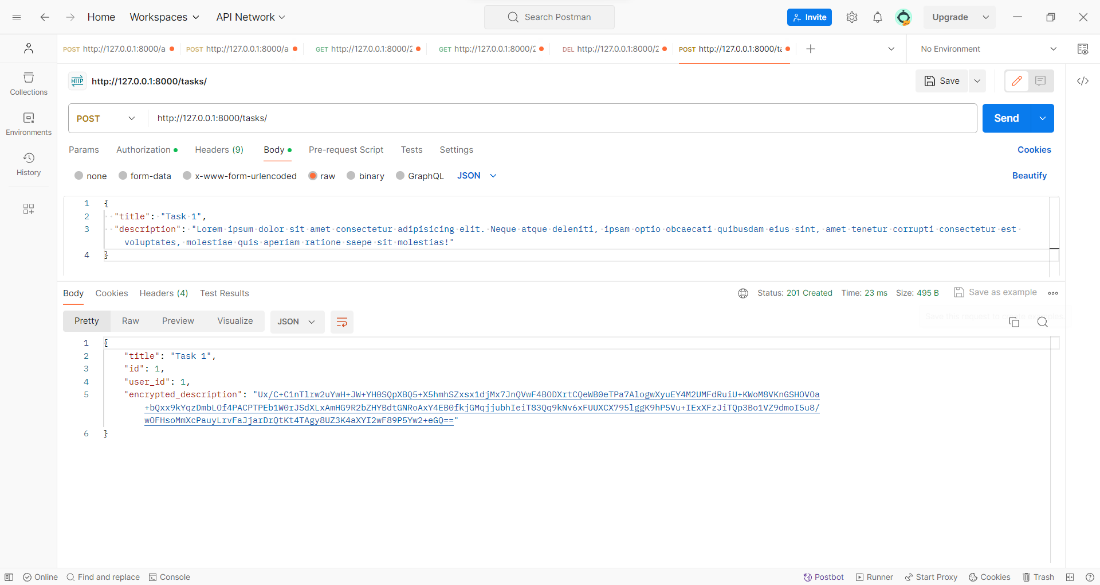
**Delete User:**



Hit endpoint [http://127.0.0.1:8000/{user\_id}](http://127.0.0.1:8000/%7buser_id%7d) dengan method DELETE untuk melakukan penghapusan user. Ketika berhasil akan mengembalikan response berupa pesan “User deleted successfully” dengan status 200 OK. Cek kembali apakah sudah terupdate dengan hit http://127.0.0.1:8000/{user\_id} dengan method GET.

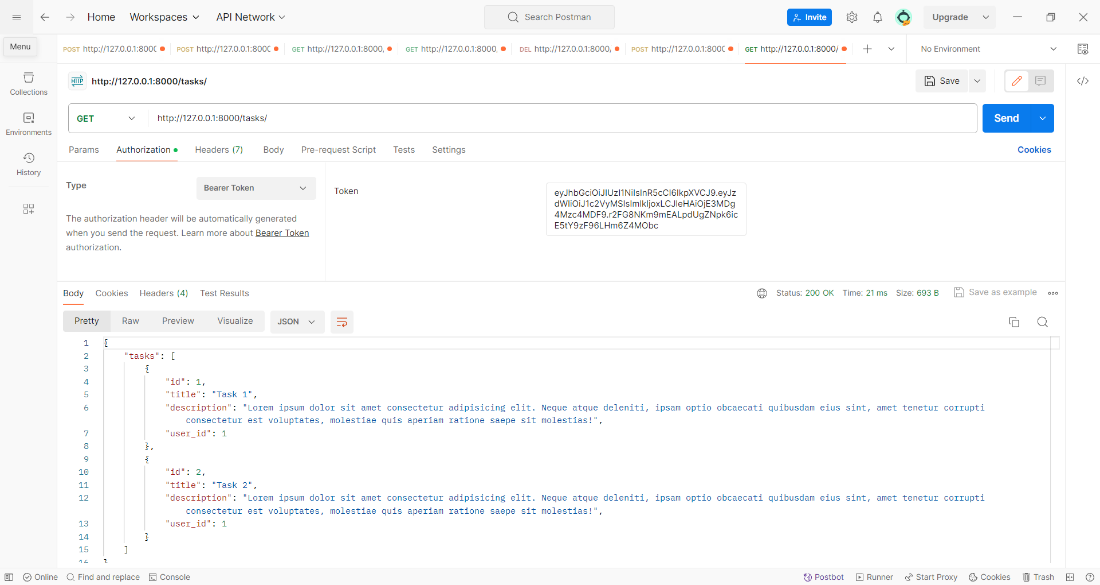


**Create Task:**



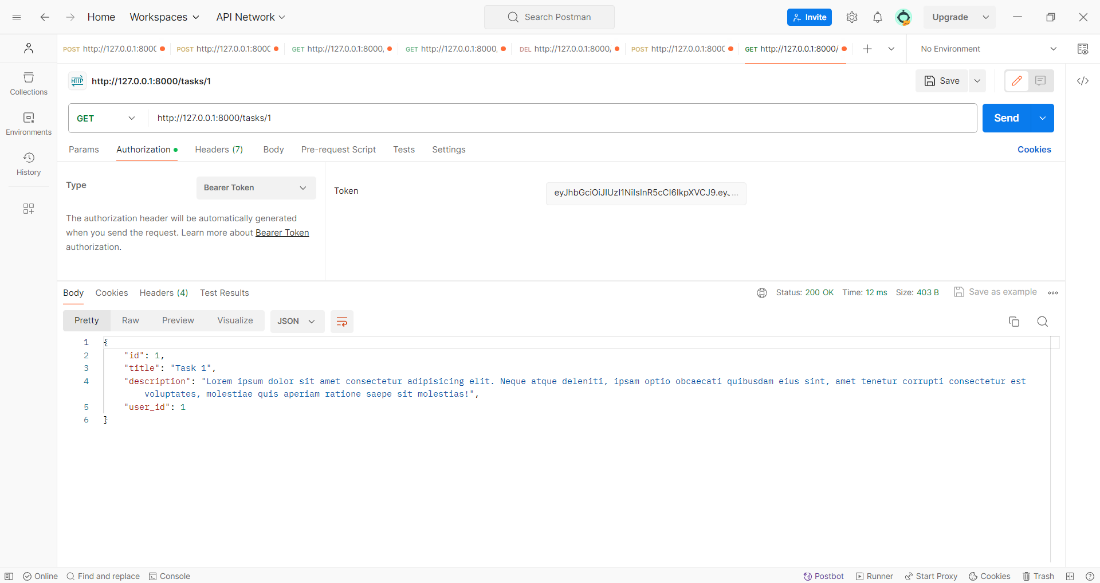
Untuk membuat task, user diharuskan membuat akun user dan lakukan login untuk mendapatkan token JWT agar bisa mengakses method create, read, update, dan delete table tasks. Hit endpoint “http://127.0.0.1:8000/ tasks/” method post. Jika berhasil, response akan berupa status “201 Created” dan mendapatkan kembalian berupa data task yang dibuat.

**View All Tasks:**



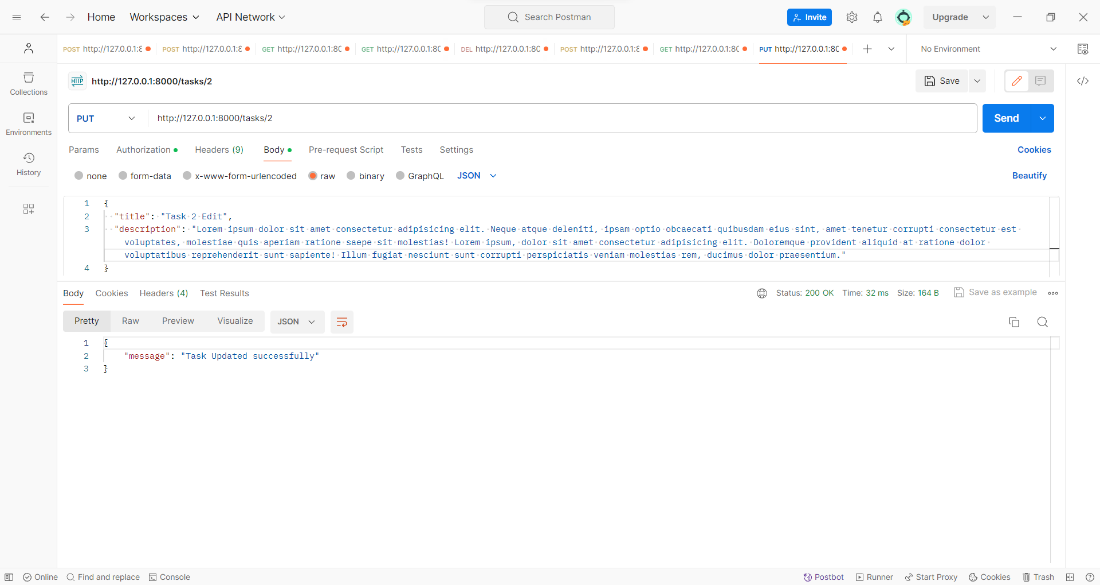
Hit endpoint http://127.0.0.1:8000/tasks/ dengan method GET untuk mendapatkan kembalian berupa JSON seluruh tasks dengan status 200 OK.

**View Tasks by Id:**

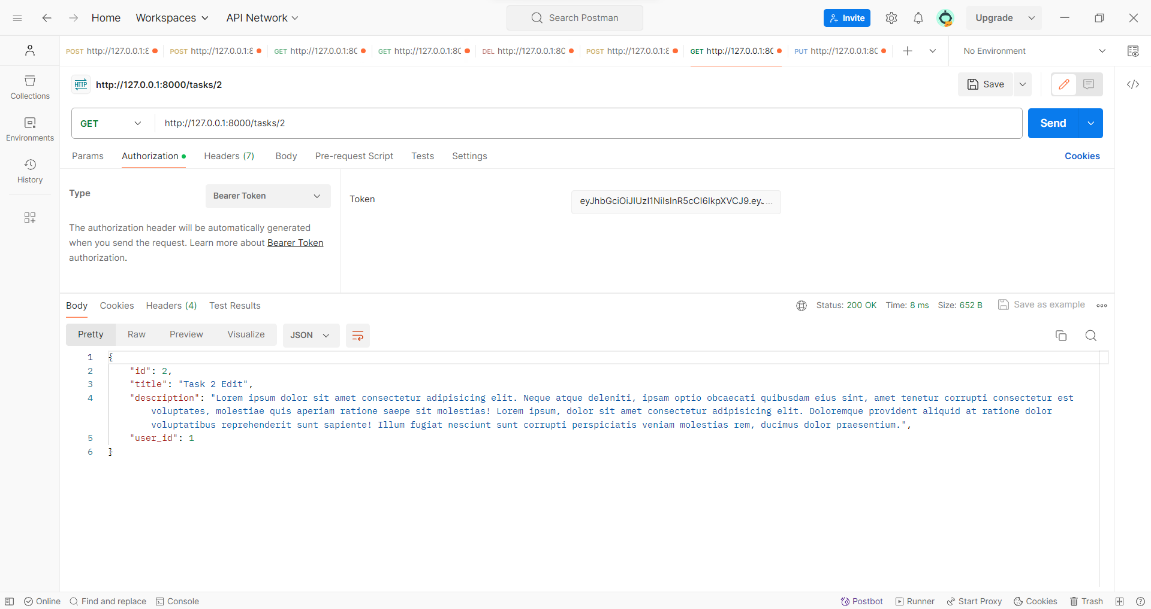


Hit endpoint [http://127.0.0.1:8000/tasks/{task\_id](http://127.0.0.1:8000/tasks/%7btask_id)} dengan method GET untuk mendapatkan kembalian JSON task berdasarkan task id.

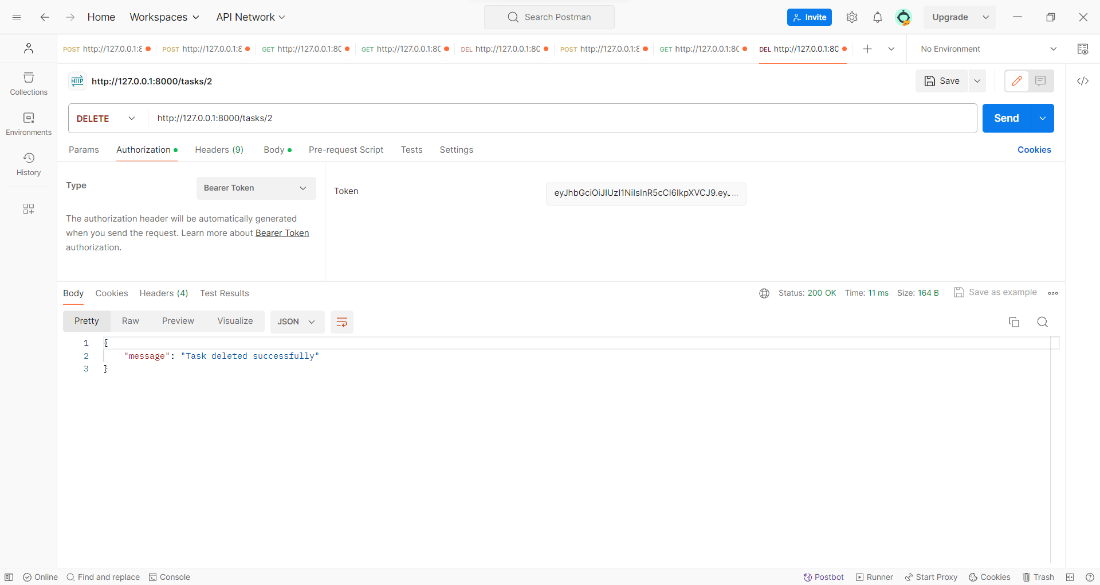
**Update Task:**



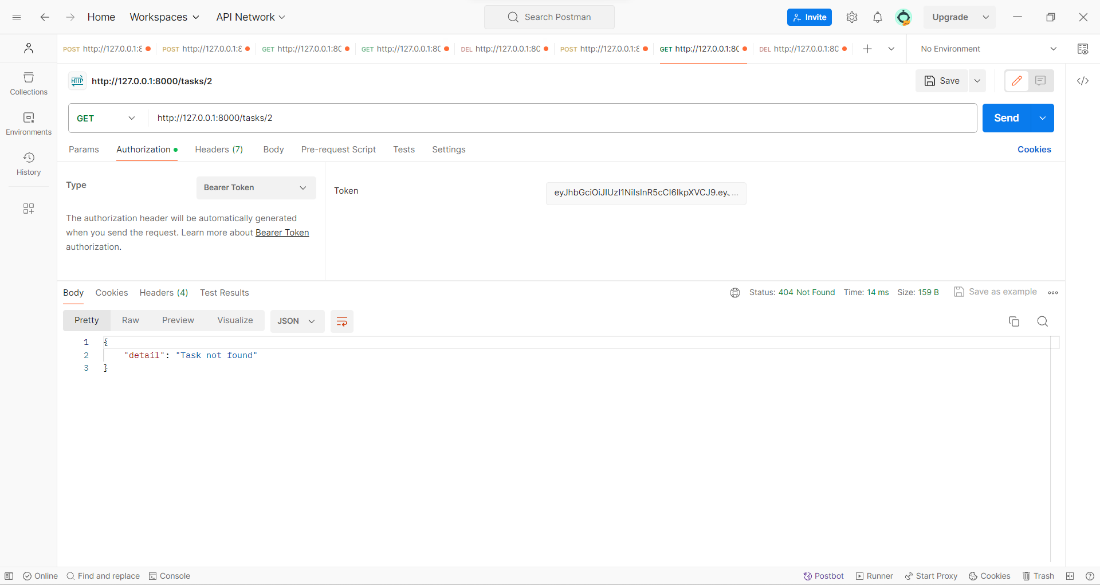
Hit endpoint http://127.0.0.1:8000/tasks/[{task\_id}](http://127.0.0.1:8000/%7buser_id%7d) dengan method PUT untuk melakukan update task. Ketika berhasil akan mengembalikan response berupa pesan “Task updated successfully” dengan status 200 OK. Cek kembali apakah sudah terupdate dengan hit http://127.0.0.1:8000/tasks/[{task\_id}](http://127.0.0.1:8000/%7buser_id%7d) dengan method GET.



**Delete User:**

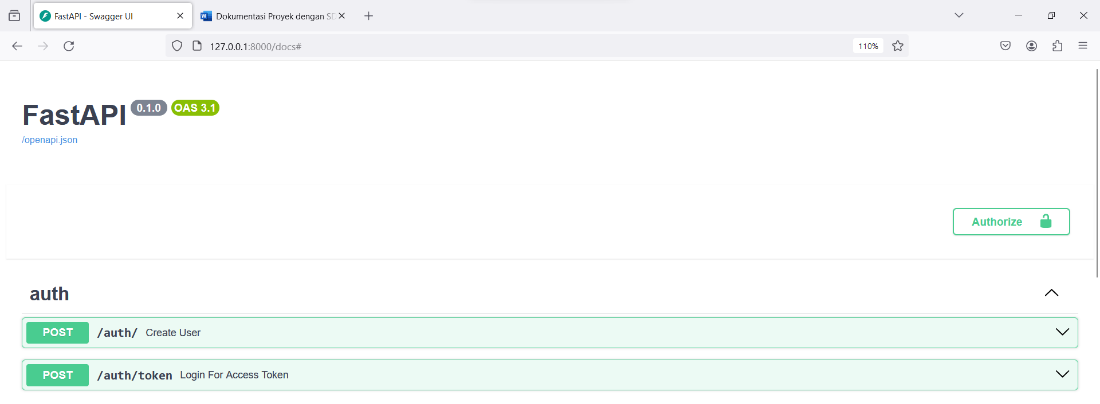


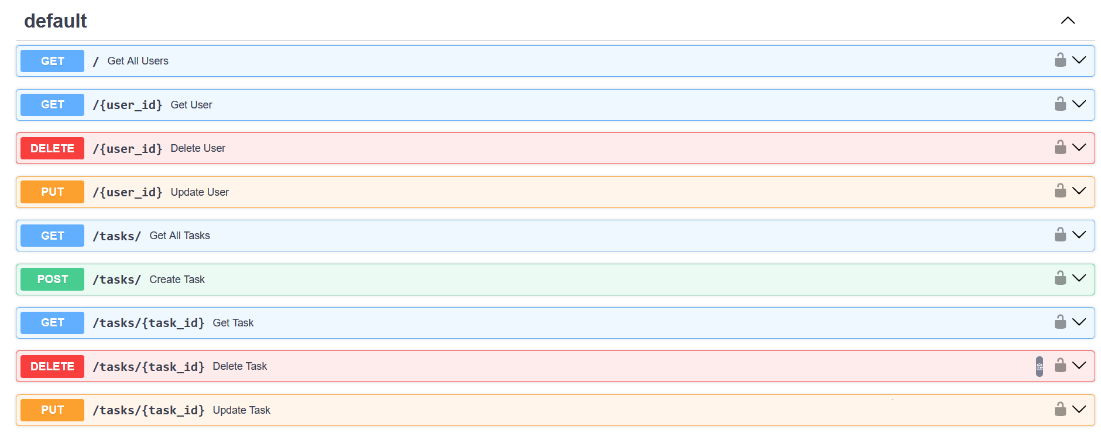
Hit endpoint http://127.0.0.1:8000/tasks/[{task\_id}](http://127.0.0.1:8000/%7buser_id%7d) dengan method DELETE untuk melakukan penghapusan task. Ketika berhasil akan mengembalikan response berupa pesan “Task deleted successfully” dengan status 200 OK. Cek kembali apakah sudah terupdate dengan hit http://127.0.0.1:8000/tasks/[{task\_id}](http://127.0.0.1:8000/%7buser_id%7d) dengan method GET.

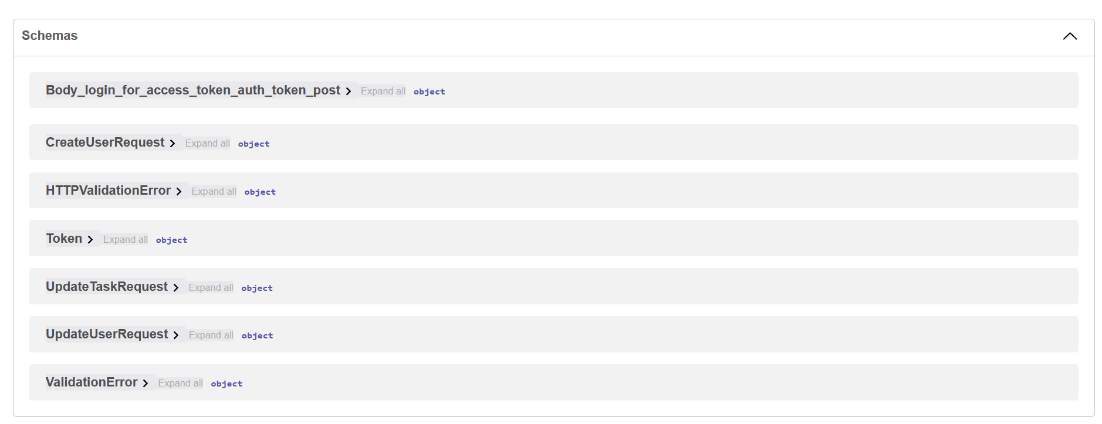


**Tampilan Swagger UI**

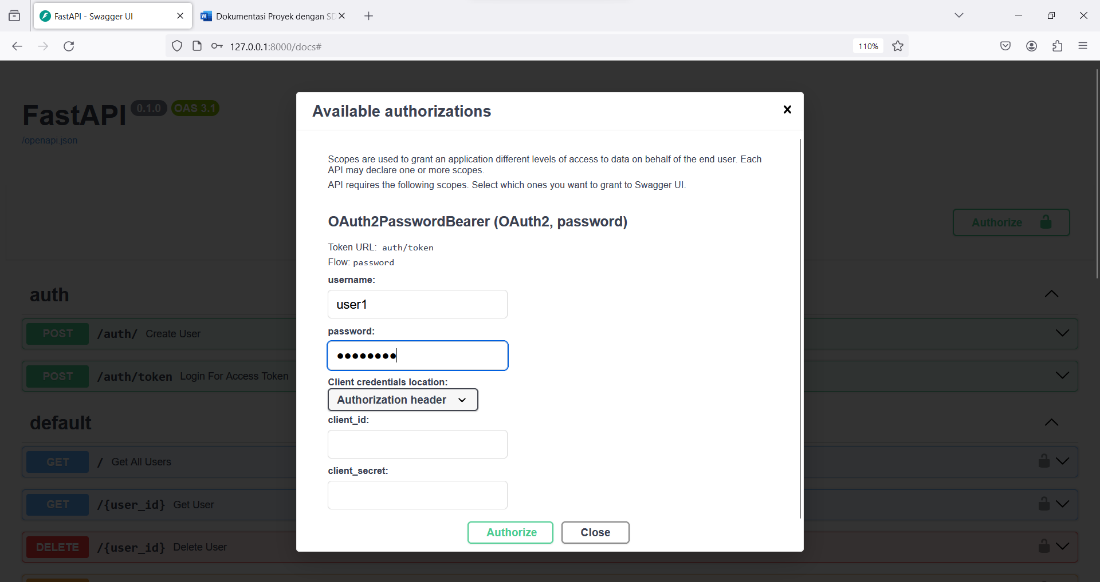
FastAPI menyediakan fitur Swagger UI untuk dokumentasi interaktif dan pengujian endpoint dengan mudah. Ini memungkinkan pengguna untuk menjelajahi dan mencoba endpoint API tanpa menulis kode tambahan atau menggunakan alat eksternal seperti Postman. Berikut tampilan dokumentasi dari endpoint yang ada pada fastAPI dengan mengakses localhost anda dengan URL “<http://127.0.0.1:8000/docs>#”.



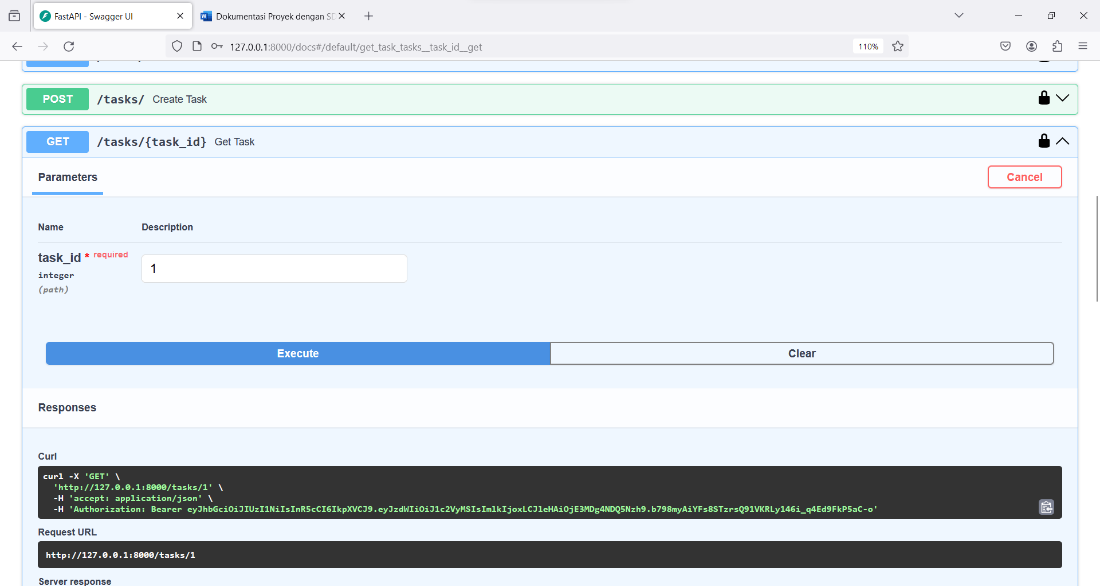




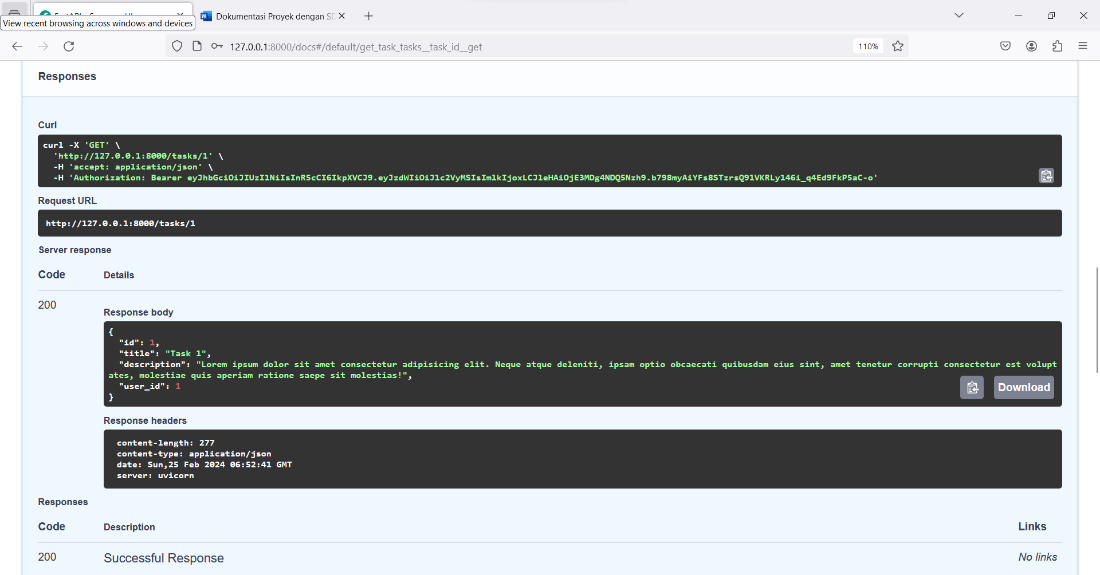
Untuk melakukan login, user cukup menekan tombol authrorize berwarna hijau dengan icon gembok, lalu akan mengeluarkan modal seperti dibawah ini.



Setelah mengisikan username dan password, dengan menekan tombol authorize maka user dapat mengakses seluruh endpoint default seperti update, read, dan delete table users dan create, update, read, dan delete table task. Sebagai contoh, saya mencoba untuk mendapatkan task berdasarkan id task.



Dengan mengklik tombol execute, maka akan menjalankan endpoint [http://127.0.0.1:8000/tasks/{task\_id](http://127.0.0.1:8000/tasks/%7btask_id)} dengan method GET dan mendapatkan response seperti dibawah ini.



1. **Maintenance:**

* **Pemantauan Kinerja**: Perlu memantau kinerja aplikasi secara teratur untuk memastikan bahwa waktu respons tetap cepat dan responsivitas aplikasi tetap baik. Gunakan alat pemantauan seperti Prometheus atau Grafana untuk melacak metrik kinerja secara real-time. Dengan memantau kinerja aplikasi, dapat dengan cepat mendeteksi masalah dan mengambil langkah-langkah perbaikan yang diperlukan untuk menjaga pengalaman pengguna yang optimal.
* **Pembaruan Reguler**: Lakukan pembaruan reguler pada aplikasi untuk memastikan bahwa selalu menggunakan versi perangkat lunak yang paling baru dan aman. Ini termasuk memperbarui dependensi, menanggapi kerentanan keamanan yang baru ditemukan, dan meningkatkan fungsionalitas sesuai dengan kebutuhan. Dengan melakukan pembaruan reguler, memastikan bahwa aplikasi tetap aman dan mendukung fungsionalitas terbaru.
* **Penanganan Bug**: Tanggapi dengan cepat dan perbaiki bug yang dilaporkan oleh pengguna. Setiap bug yang tidak ditangani dengan baik dapat mengganggu pengalaman pengguna dan bahkan menyebabkan kerugian finansial. Lakukan pengujian menyeluruh setelah memperbaiki bug untuk memastikan bahwa tidak ada efek samping yang tidak diinginkan dari perubahan yang dibuat.
* **Backup Data**: Lakukan backup data secara teratur untuk menghindari kehilangan data yang tidak terduga akibat kegagalan perangkat keras, serangan siber, atau kesalahan manusia. Simpan backup di lokasi yang aman dan pastikan bahwa proses pemulihan data berfungsi dengan baik jika diperlukan. Dengan melakukan backup data secara teratur, dapat memastikan bahwa dapat dengan cepat memulihkan data jika terjadi kegagalan.
* **Skalabilitas**: Pertimbangkan untuk meningkatkan skalabilitas aplikasi sesuai dengan pertumbuhan pengguna atau beban kerja yang meningkat. Periksa dan sesuaikan infrastruktur jika diperlukan untuk menangani beban yang lebih besar. Dengan meningkatkan skalabilitas aplikasi, dapat memastikan bahwa aplikasi tetap responsif dan dapat menangani lonjakan lalu lintas tanpa mengalami penurunan kinerja.

Link File GITHUB :

[dasdoyessa/UAS-Microservice-CRUD-API-PL-SQL-UNSIA: Jawaban UAS PL/SQL Dasdo Yessa 210401010176 (github.com)](https://github.com/dasdoyessa/UAS-Microservice-CRUD-API-PL-SQL-UNSIA/tree/main)

|  |  |  |
| --- | --- | --- |
| **Nilai** | **Tanda Tangan Dosen Pengampu / Tutor** | **Tanda Tangan**  **Mahasiswa** |
|  | **(……………………..)** | **Dasdo Yessa** |
| Diserahkan pada Tanggal: | | Tanggal Mengumpulkan:  18 Februari 2024 |