LAPORAN

RANCANGAN REST API



TEMA: Sistem Peminjaman Lab Komputer

Dirancang oleh:

1. Perdy imam Samudra

2. M. Faisal Muzhaffar

3. Raja aditia

INFORMATICS ENGINEERING

NUSA PUTRA UNIVERSITY

Jl.Raya Cibolang kaler,kec.cisaat

Kab.Sukabumi,Jawa Barat 43152

**I. PENDAHULUAN**

Puji syukur kita panjatkan kehadirat Allah SWT milik yang senantiasa meridhoi apa yang hamba-Nya kerjakan dengan baik, sehingga kami dapat menyelesaikan tugas akhir ini, yaitu laporan berjudul *"Perancangan REST API pada Sistem Peminjaman Lab Komputer"* yang diberikan oleh dosen pengampu mata kuliah *Pemrograman Berbasis Platform*. Shalawat serta salam kami panjatkan kepada Nabi besar kita Muhammad SAW beserta para sahabat, keluarga, dan umatnya hingga akhir zaman.

Laporan ini kami buat dengan harapan dapat memberikan manfaat bagi para pembaca, khususnya sahabat-sahabat semua, dan juga bagi diri kami sendiri. Kami menyadari bahwa laporan ini masih memiliki kekurangan, baik dalam pembahasan maupun penulisan. Oleh karena itu, kami dengan tulus mengharapkan kritik dan saran yang membangun dari para pembaca demi penyempurnaan laporan ini di masa mendatang.

Dokumen ini bertujuan untuk merancang dan mengembangkan sebuah sistem REST API pada sistem peminjaman laboratorium komputer. Sistem ini diharapkan mampu memenuhi kebutuhan pengguna, meningkatkan efisiensi pengelolaan jadwal, dan memberikan pengalaman yang lebih baik bagi semua pihak yang terlibat dalam penggunaan laboratorium ini.

**A. LATAR BELAKANG**

Laboratorium komputer merupakan fasilitas penting dalam institusi pendidikan untuk mendukung pembelajaran berbasis teknologi. Namun, pengelolaan manual sering menghadapi kendala seperti kesulitan jadwal dan kurangnya transparansi. Untuk itu, sistem berbasis REST API diperlukan guna mengotomatisasi peminjaman, meningkatkan efisiensi, dan mempermudah aksesibilitas. Sistem ini diharapkan mampu memberikan solusi modern yang efektif bagi institusi pendidikan.

**B. TUJUAN**

Pengembangan REST API untuk sistem asuransi memerlukan perancangan database yang efisien dengan empat tabel utama: users, labs, schedules, dan borrowings . API ini dibuat untuk mendukung pengelolaan data asuransi, seperti manajemen pengguna, pengaturan polis, pengajuan klaim, serta proses pembayaran, dengan cara yang terstruktur dan efektif. Dengan pendekatan ini, sistem dapat memberikan layanan yang lebih terintegrasi dan mendukung kebutuhan operasional secara optimal.

**II. HASIL LAPORAN**

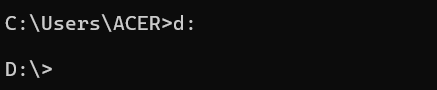
1. **Analisis kebutuhan**
2. Autentikasi dan Otorisasi Pengguna Login/logout dengan JWT.
3. CRUD Data Pengguna (Users)
4. CRUD Data Lab Komputer (Labs)
5. CRUD Data Jadwal (Schedules)
6. Data Peminjaman (Borrowings)
7. Performa: Implementasi caching menggunakan Node-Cache.
8. Format Data: JSON
9. Pengujian: Menggunakan Postman.
10. Relasi data antar table
11. Database



1. Endpoint

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| api/auth/register | POST | Mendaftar pengguna baru | - | {  “name” : “ fxilia”,  “email” : “fxilia22@gmail.com”,  “password” : “disinifxilia321”,  “role” : “User”  } | - |
| Api/auth/login | POST | Login pengguna | - | {  “email” : “fxilia22@gmail.com”,  “password” : “disinifxilia321”  } | - |
| Api/users | GET | Mendapatkan daftar semua pengguna | - | - | Authorization: Bearer <JWT> |
| Api/users/id | GET | Mengambil detail pengguna berdasarkan id | Id | - | Authorization: Bearer <JWT> |
| Api/users/id | DELETE | Menghapus pengguna | id | - | Authorization: Bearer <JWT> |
| Api/labs | GET | Mendapatkan daftar semua lab |  | - | Authorization: Bearer <JWT> |
| Api/labs/id | GET | Mengambil detail lab berdasarkan id | id | - | Authorization: Bearer <JWT> |
| Api/labs | POST | mendaftar lab | - | {  “name” : “Physisc Lab”,  “capacity”: 30,  “facilities”: “Computers, Projectirs, vga only”,  “status”: “2”  } | Authorization: Bearer <JWT> |
| Api/labs/id | PUT | Merubah lab | id | {  “name” : “Physisc Lab”,  “capacity”: 40,  “facilities”: “Computers, Projectirs, vga only”,  “status”: “2”  } | Authorization: Bearer <JWT> |
| Api/labs/id | DELETE | Mengapus lab | id | - | Authorization: Bearer <JWT> |
| Api/schadules | POST | Mendaftarkan jadwal | - | {  “lab\_id” : 2,  “start\_time”: “2025-01-16 10:00:00”,  “end\_time”: “2025-01-16 12:00:00”,  “is\_reserved”: false  } | Authorization: Bearer <JWT> |
| Api/schadules | GET | Mendapatkan daftar semua jadwal | - | - | Authorization: Bearer <JWT> |
| Api/schadules/id | GET | Mengambil detail jadwal berdasarkan id | id | - | Authorization: Bearer <JWT> |
| Api/schadules/id | PUT | Merubah jadwal | id | {  “lab\_id” : 3,  “start\_time”: “2025-01-16 10:00:00”,  “end\_time”: “2025-01-16 12:00:00”,  “is\_reserved”: false  } | Authorization: Bearer <JWT> |
| Api/schadules/id | DELETE | Menghapus jadwal | id |  | Authorization: Bearer <JWT> |
| Api/borrowings | POST | Mendaftarkan pemesanan | - | {  “user\_id” : 2,  “lab\_id”: 2,  “schedule\_id”: 5,  “reason”: “Final exam preparation”  “status”:”pending”  } | Authorization: Bearer <JWT> |
| Api/borrowings | GET | Mendapatkan daftar semua pemesanan | - | - | Authorization: Bearer <JWT> |
| Api/borrowings/id | GET | Mengambil detail pemesanan berdasarkan id | id | {  “user\_id” : 2,  “lab\_id”: 2,  “schedule\_id”: 5,  “reason”: “Final exam preparation”  “status”:”pending”  } | Authorization: Bearer <JWT> |
| Api/borrowings/id | PUT | Menambah pemesanan | id | {  “user\_id” : 1,  “lab\_id”: 2,  “schedule\_id”: 5,  “reason”: “Final exam preparation”  “status”:”pending”  } | Authorization: Bearer <JWT> |
| Api/borrowings/id | DELETE | Menghapus pemesanan | id | - | Authorization: Bearer <JWT> |

1. Langkah – Langkah Programan
2. Buka command prompt
3. Cari drive yang ingin anda gunakan, contoh nya seperti gambar di bawah ini:



saya menyimpan di hardisk D.

3) Buat folder “Project-lab” menggunakan sintaks nama-direktori mkdir seperti yang ditunjukkan di bawah ini:



4) Ketika sudah menentukan director,dan jika ingin memasuki direktori tersebut menggunakan sintax cd seperti gambar di bawah ini:



5) lalu menginstall proyek Node.js untuk membuat file package.json secara otomatis gunakan perintah :



6) dan jangan lupa untuk menginstall library berikut:

1. express

2. body-parser

3. cors

4. dotenv

5. mysql2

6. nodemon

7. axios

8. bcrypt

9. jsonwebtoken

7) Perancangan file dan folder yang saya buat sebagai berikut:

/PROJECT-LAB

├── /config

│ ├── db.js

│

├── /controllers

│ ├── authController.js

│ ├── BorrowingsController.js

│ ├── LabController.js

│ ├── publicApiController.js

│ ├── scheduleController.js

│ └── userController.js

│

├── /middleware

│ └── authMiddleware.js

│

├── /routes

│ ├── auth.js

│ ├── borrowings.js

│ ├── labs.js

│ ├── publicApi.js

│ ├── schadules.js

│ └── users.js

│

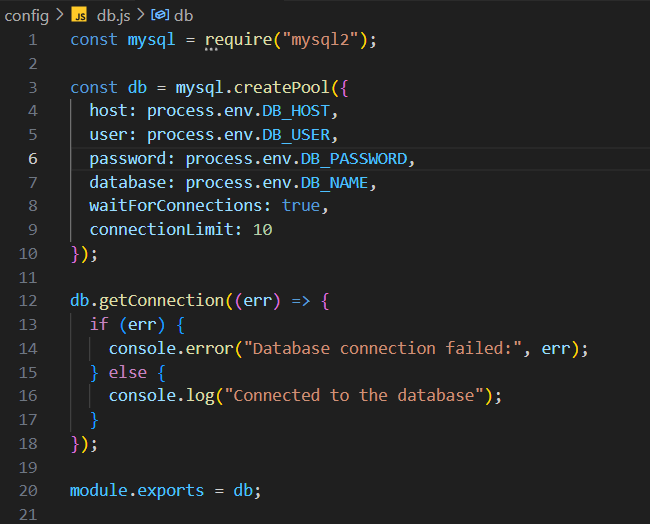
├── .env

│

└── index.js

Alasan kami tidak memakai model, hanya menggunakan router dan controller, bisa karena pendekatan yang kmi pilih adalah lebih sederhana dan langsung, tanpa menggunakan abstraksi tambahan yang seringkali diperlukan dalam arsitektur yang lebih kompleks.

8) Membuat folder config berisi file db.js fungsinya digunakan untuk mengelola koneksi database. File ini bertanggung jawab untuk mengatur pengaturan koneksi ke database, seperti MySQL, MongoDB, atau database lainnya yang digunakan dalam proyek.



Pada kode diatas membuat connection pool ke database MySQL menggunakan library mysql2 dengan konfigurasi seperti host, user, password, dan database dari file .env.

* mysql.createPool(): Membuat kumpulan koneksi untuk efisiensi.
* db.getConnection(): Menguji koneksi ke database (menampilkan pesan sukses atau error).
* module.exports = db: Mengekspor koneksi agar dapat digunakan di file lain.

Fungsinya adalah mengelola koneksi database secara efisien untuk aplikasi Anda.

9) Setelah itu membuat folder controller berisi 6 file yaitu authController.js, BorrowingsController.js, LabController.js, publicApiController.js, scheduleController.js, dan userController.js. Berikut tahapan kodingan nya :

1. userController.js

File userController.js menangani operasi terkait pengguna, yaitu:

1. **getUsers**: Mengambil daftar semua pengguna dengan memilih id, name, email, dan role dari tabel Users, dan mengirimkan data tersebut kepada pengguna.
2. **getUserById**: Mengambil data pengguna berdasarkan ID yang diberikan melalui parameter URL. Jika pengguna dengan ID tersebut tidak ditemukan, mengirimkan pesan "User not found".

File ini mengelola data pengguna dan mengirimkan respons yang sesuai, termasuk penanganan error jika terjadi masalah.

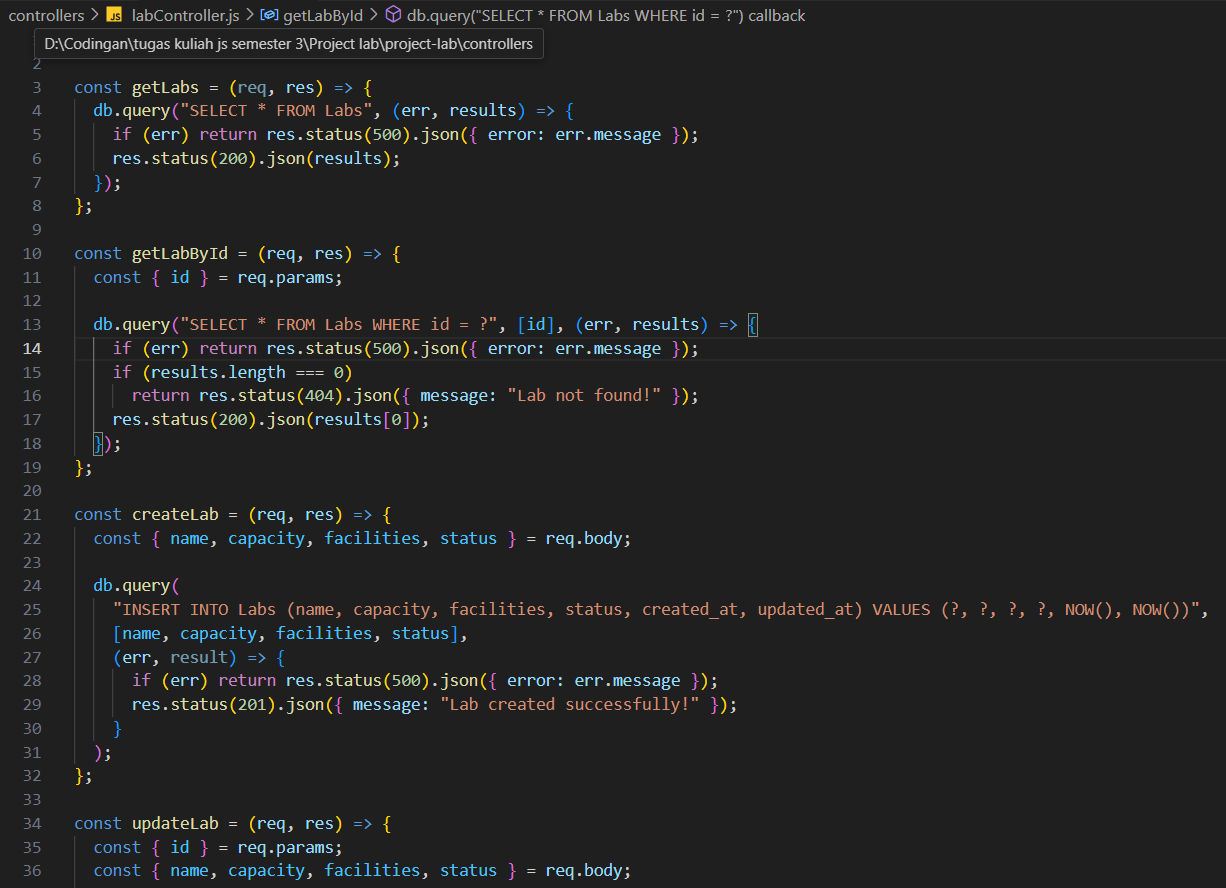


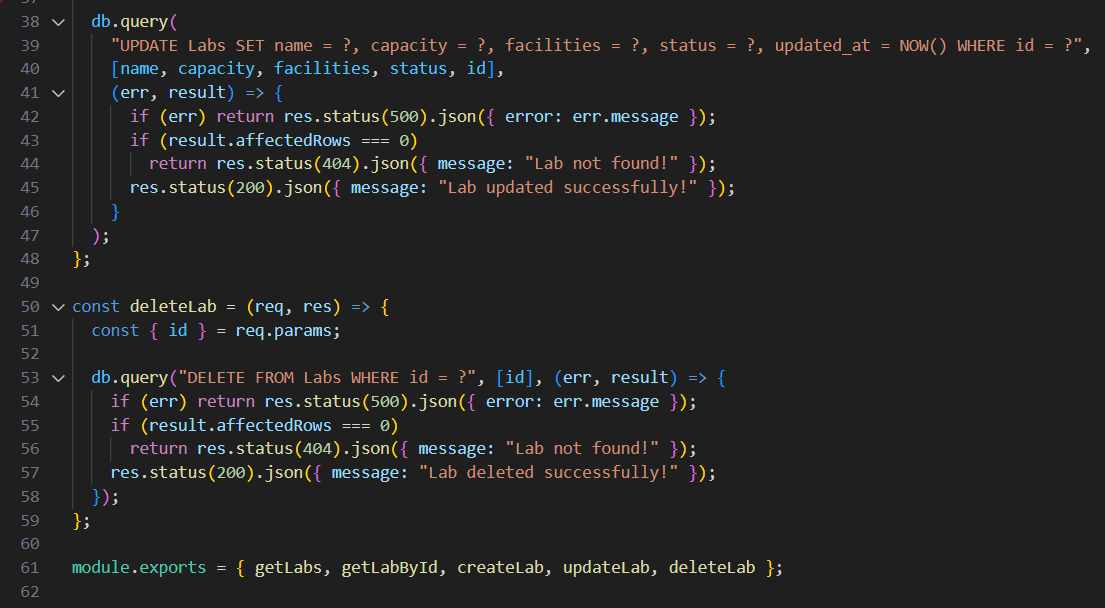
1. labController.js

File LabController.js menangani operasi terkait laboratorium dalam aplikasi. Berikut penjelasannya:

1. **getLabs**: Mengambil semua data laboratorium dari database.
2. **getLabById**: Mengambil data laboratorium berdasarkan ID yang diberikan.
3. **createLab**: Membuat data laboratorium baru dengan informasi seperti nama, kapasitas, fasilitas, dan status.
4. **updateLab**: Memperbarui data laboratorium berdasarkan ID.
5. **deleteLab**: Menghapus data laboratorium berdasarkan ID.

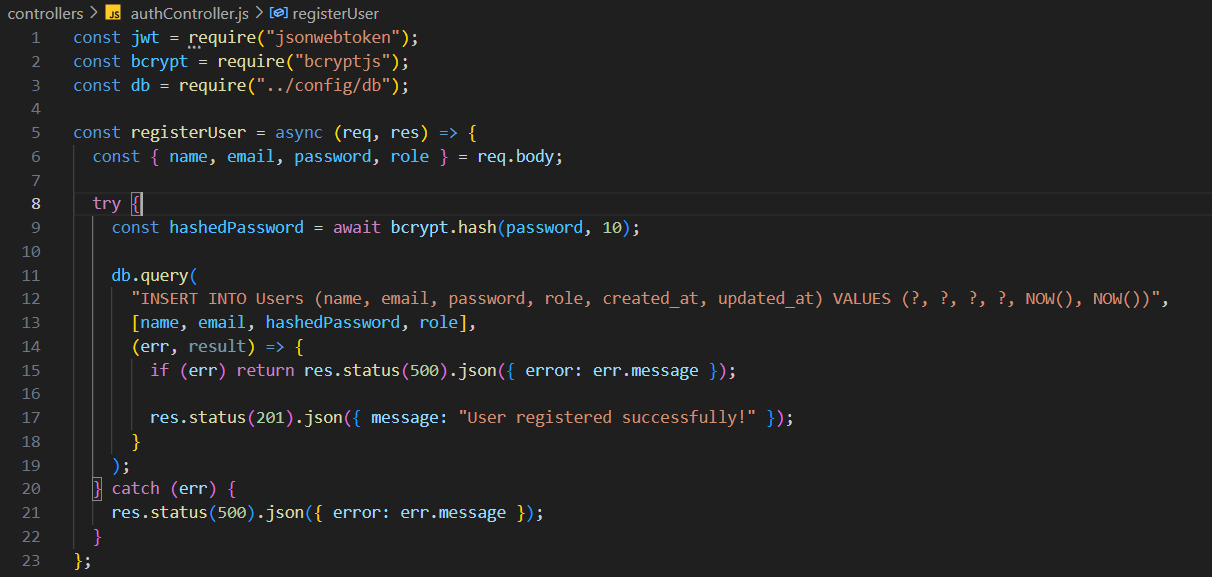
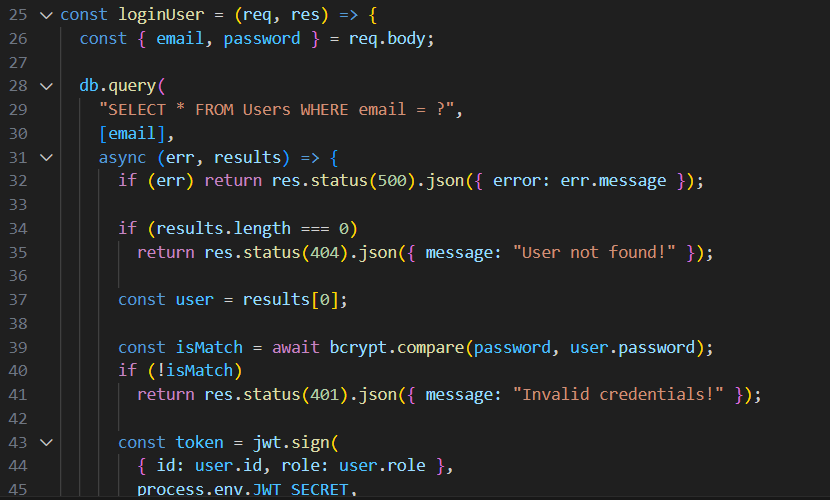
File ini berkomunikasi dengan database untuk mengelola data laboratorium dan memungkinkan pengguna untuk melakukan operasi CRUD (Create, Read, Update, Delete).





1. authController.js

File authController.js menangani proses autentikasi pengguna dalam aplikasi. Fungsi utamanya meliputi pendaftaran pengguna baru dengan enkripsi password menggunakan bcrypt, serta login pengguna yang valid dengan pengecekan password dan pembuatan token JWT. Keduanya berkomunikasi dengan database untuk menyimpan dan memverifikasi data pengguna.





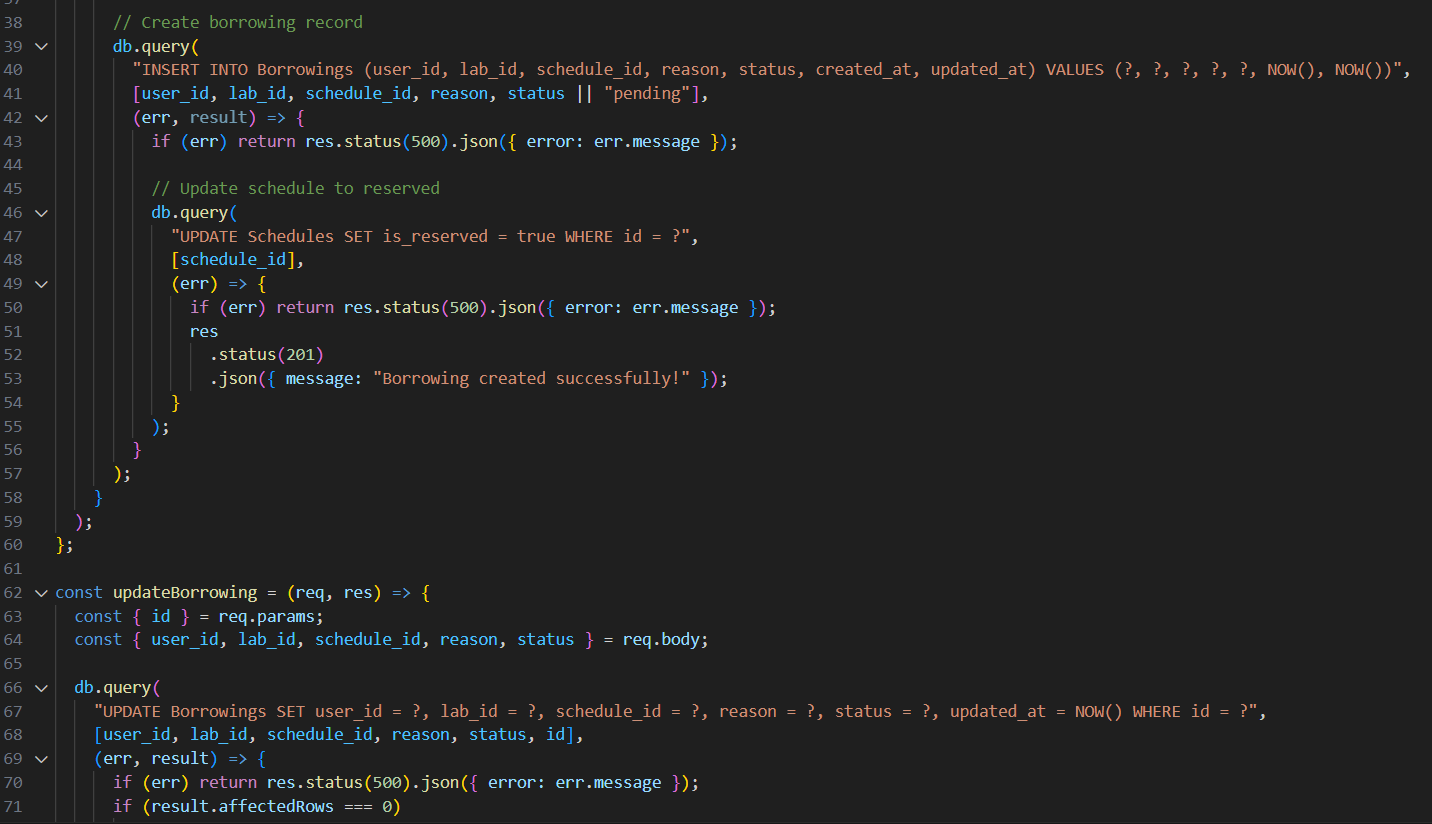
1. BorrowingsController.js

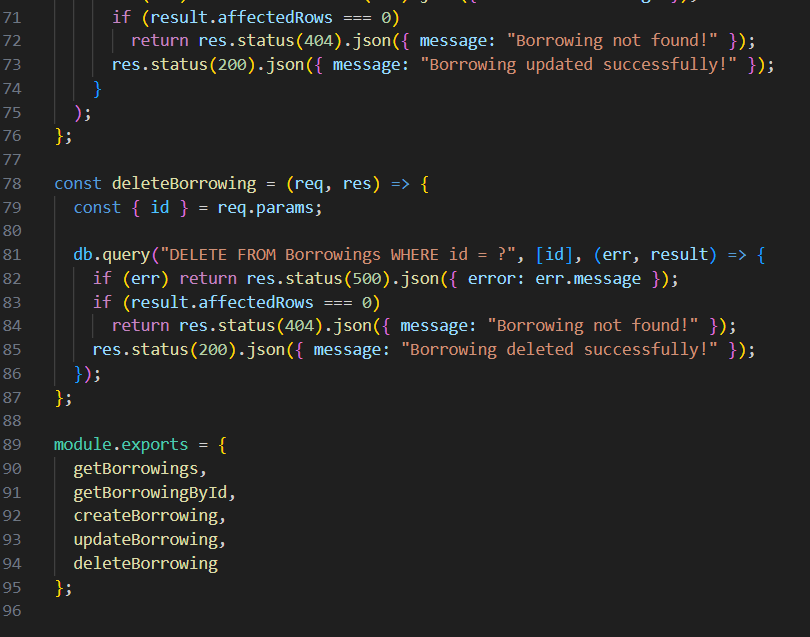
File BorrowingsController.js menangani operasi terkait peminjaman dalam aplikasi. Berikut penjelasan singkatnya:

1. **getBorrowings**: Mengambil semua data peminjaman dari database.
2. **getBorrowingById**: Mengambil data peminjaman berdasarkan ID yang diberikan.
3. **createBorrowing**: Membuat data peminjaman baru setelah memeriksa ketersediaan jadwal. Jika jadwal sudah terisi, peminjaman tidak dapat dilakukan.
4. **updateBorrowing**: Memperbarui data peminjaman berdasarkan ID.
5. **deleteBorrowing**: Menghapus data peminjaman berdasarkan ID.

File ini berinteraksi dengan database untuk mengelola data peminjaman dan memperbarui status jadwal yang terpengaruh oleh peminjaman.





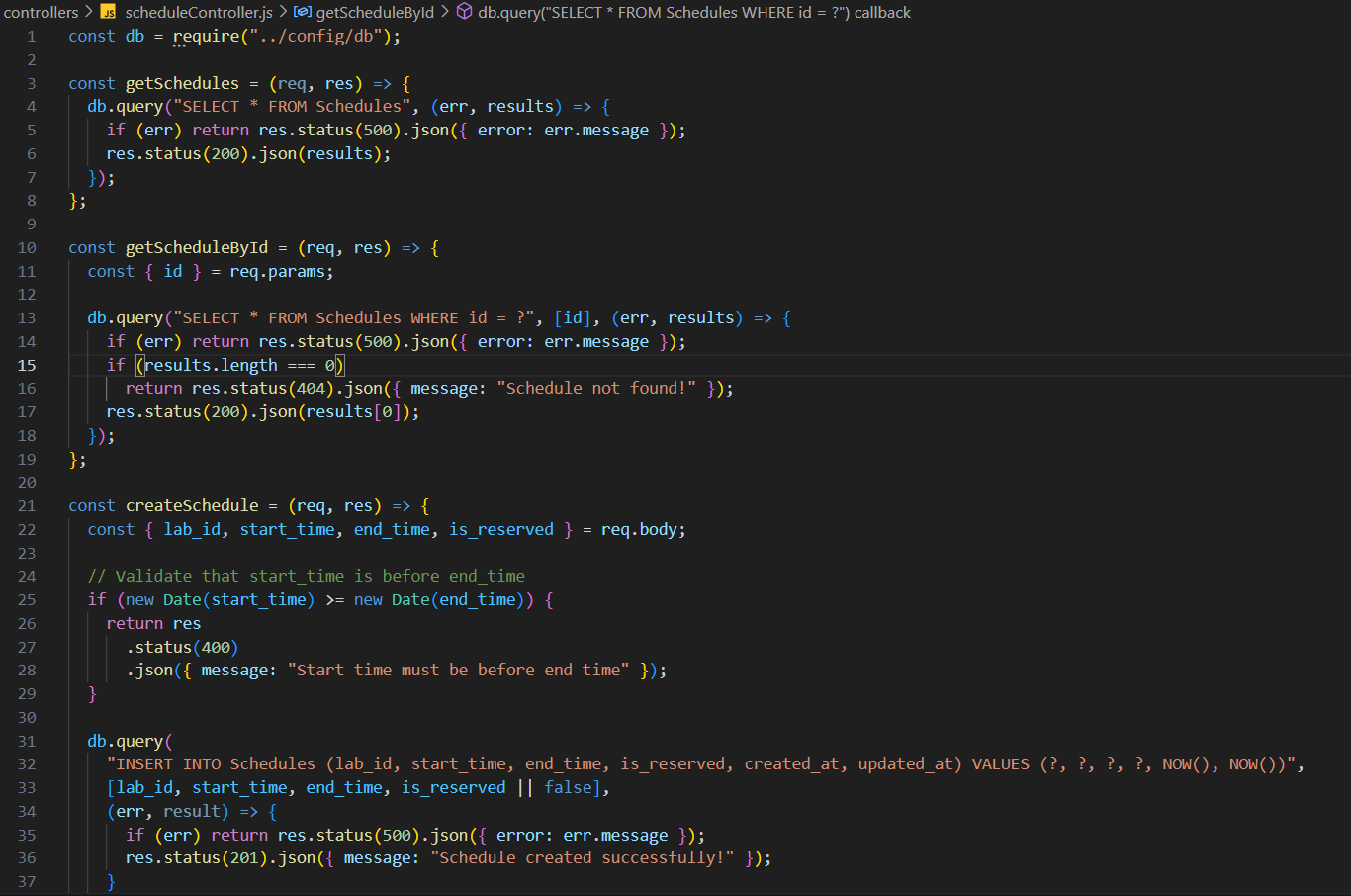
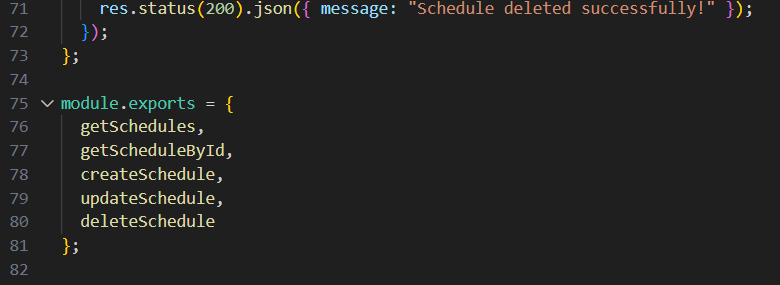


1. scheduleController.js

File scheduleController.js menangani operasi terkait jadwal dalam aplikasi, seperti mengambil, membuat, memperbarui, dan menghapus data jadwal. Berikut penjelasan untuk masing-masing fungsi:

1. **getSchedules**: Mengambil semua data jadwal dari database dan mengirimnya ke pengguna.
2. **getScheduleById**: Mengambil data jadwal berdasarkan ID yang diberikan, jika tidak ditemukan, akan mengirim pesan "Schedule not found".
3. **createSchedule**: Membuat jadwal baru dengan data yang diterima. Menyaring jika waktu mulai tidak lebih awal dari waktu selesai.
4. **updateSchedule**: Memperbarui jadwal yang sudah ada berdasarkan ID, dengan validasi waktu mulai dan selesai.
5. **deleteSchedule**: Menghapus jadwal berdasarkan ID yang diberikan. Jika jadwal tidak ditemukan, akan mengirim pesan "Schedule not found".

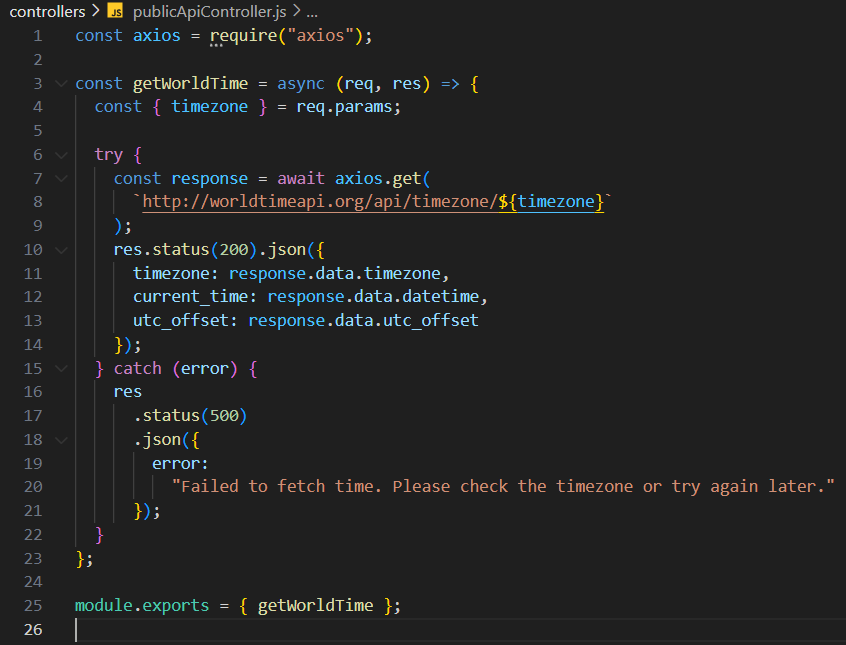
File ini mengelola jadwal dengan cara yang sederhana, termasuk validasi dan penanganan error jika ada masalah.

1. publicApiController.js

File publicApiController.js bertujuan untuk mengambil informasi waktu saat ini berdasarkan timezone yang diberikan melalui parameter URL. Fungsi getWorldTime menggunakan axios untuk mengirim permintaan HTTP ke API eksternal (WorldTime API) dan mengembalikan data seperti timezone, waktu saat ini, dan offset UTC.

Tetapi, implementasi ini berhenti karena ada bug dan error yang membuat kami kewalahan, dan menyebabkan aplikasi tidak dapat melanjutkan penggunaan API tersebut.



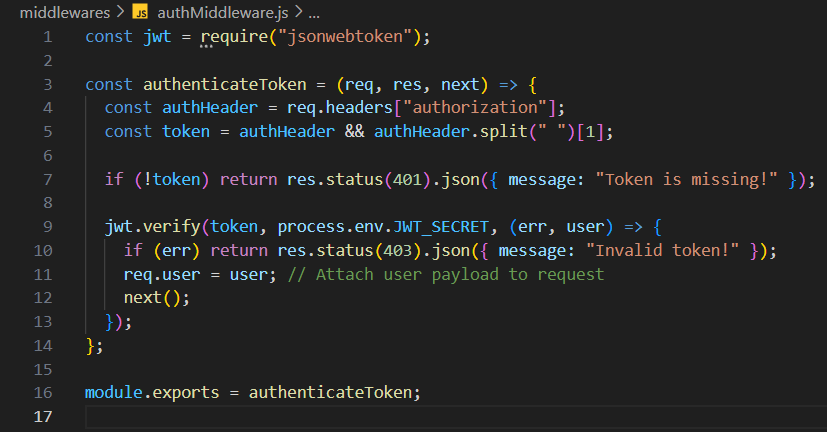
10) Setelah membuat folder controller.js, selanjutnya kita membuat folder middleware yang berisi file authMiddleware.js.Middleware adalah fungsi dalam aplikasi yang berada di antara permintaan (request) yang datang dan respons (response) yang dikirimkan. Middleware digunakan untuk memproses data, mengelola kontrol alur program, dan menambahkan fungsionalitas tambahan sebelum melanjutkan eksekusi ke fungsi berikutnya (seperti controller atau endpoint).

1. AuthMiddleware

File authMiddleware.js berisi middleware yang digunakan untuk memverifikasi dan mengautentikasi token JWT (JSON Web Token) dalam aplikasi. Middleware ini memastikan bahwa pengguna memiliki token yang valid sebelum mengakses rute tertentu.

Berikut penjelasannya:

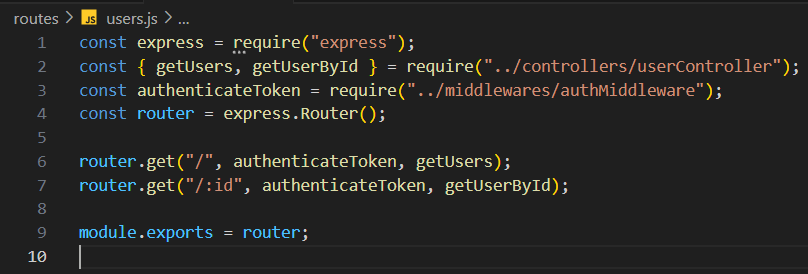
1. Token diambil dari header Authorization.
2. Jika token tidak ada, respons 401 dikirim.
3. Token yang ada diverifikasi dengan JWT\_SECRET.
4. Jika token valid, payload pengguna ditambahkan ke req.user dan melanjutkan ke middleware berikutnya.
5. Jika token invalid, respons 403 dikirim.



11) Selanjutnya membuat folder routes dengan 6 file yang tersambung dengan controller dan 6 file sebelumnya. Routes berfungsi untuk mengarahkan permintaan HTTP (seperti GET, POST, PUT, DELETE) ke controller atau fungsi yang sesuai. Folder ini memisahkan logika routing dari bagian lain aplikasi agar lebih terstruktur dan mudah dikelola.

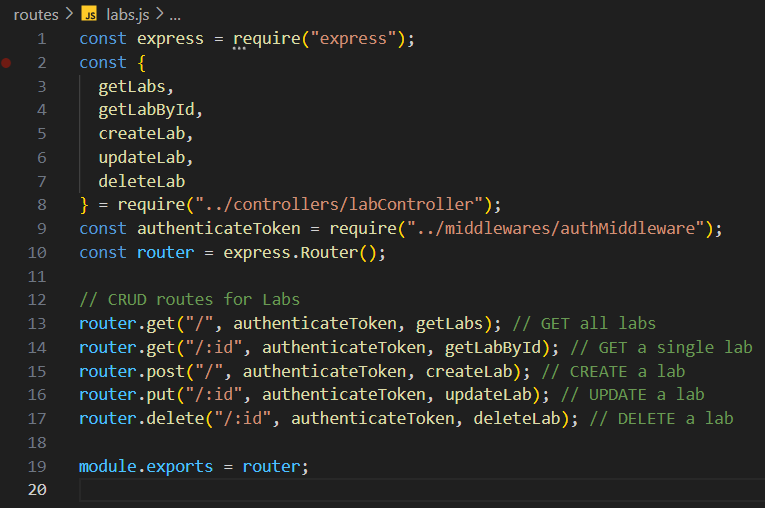
1. Users.js

File users.js ini bertanggung jawab untuk mendefinisikan rute-rute HTTP terkait pengguna, seperti mengambil semua pengguna dan mengambil pengguna berdasarkan ID, serta memastikan bahwa setiap permintaan dilindungi oleh autentikasi token menggunakan middleware authenticateToken sebelum diakses oleh controller.



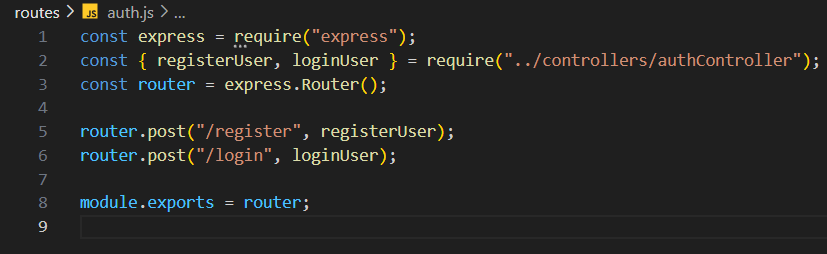
1. Labs.js

File labs.js ini mendefinisikan rute-rute HTTP untuk menangani operasi CRUD pada data lab, dengan melibatkan middleware authenticateToken untuk memverifikasi autentikasi pengguna. Rute-rute ini mengakses fungsi-fungsi yang ada pada labController.js untuk melakukan operasi pada database, seperti mendapatkan, membuat, memperbarui, dan menghapus lab.



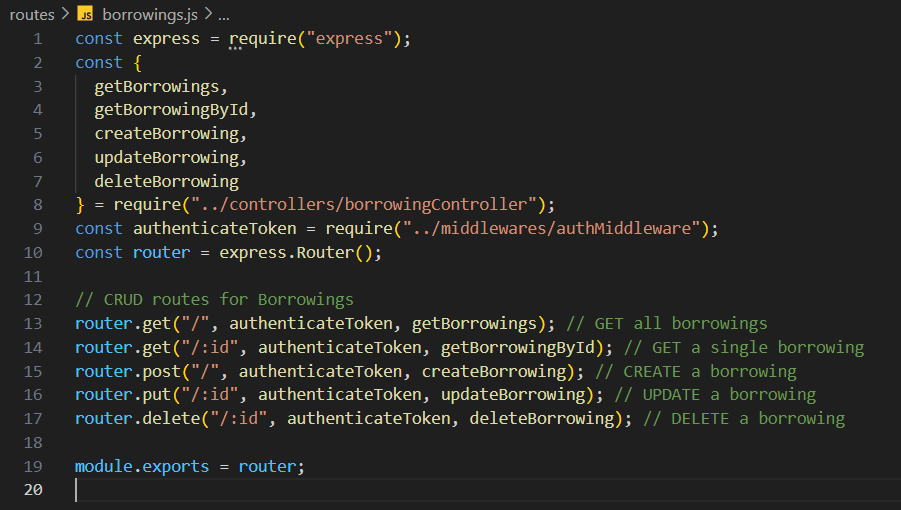
1. Auth.js

File auth.js mendefinisikan rute-rute untuk autentikasi pengguna, termasuk pendaftaran dan login. Fungsi-fungsi yang terkait dengan pendaftaran dan login di-handle oleh authController.js. Rute ini tidak memerlukan autentikasi pengguna, karena digunakan untuk registrasi dan login pertama kali.



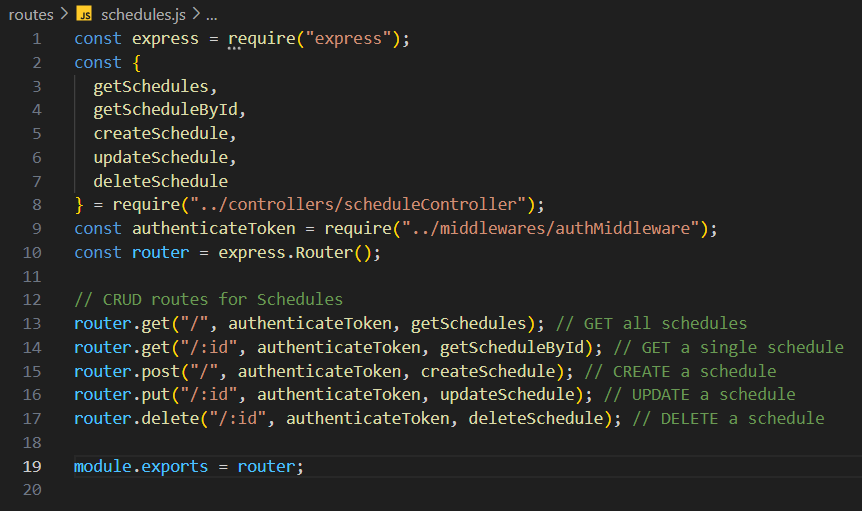
1. Borrowings

File borrowings.js mendefinisikan rute-rute untuk operasi CRUD terkait peminjaman. Setiap rute dilindungi dengan middleware authenticateToken untuk memastikan hanya pengguna yang terautentikasi yang dapat mengakses data peminjaman. Rute ini menghubungkan ke borrowingController.js yang berisi logika untuk menangani permintaan yang relevan.



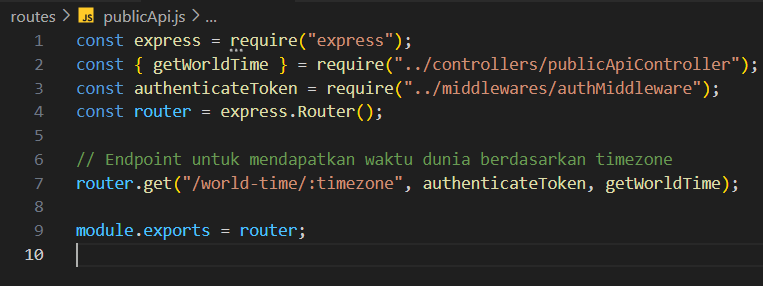
1. Schedules.js

File schedules.js mendefinisikan rute-rute yang menghubungkan ke operasi CRUD terkait jadwal dalam aplikasi. Setiap rute dilindungi dengan middleware authenticateToken yang memastikan hanya pengguna yang terautentikasi yang dapat mengaksesnya. Rute-rute ini kemudian mengarah ke fungsi-fungsi dalam scheduleController.js yang menangani logika untuk operasi-operasi tersebut.



1. publicApi.js

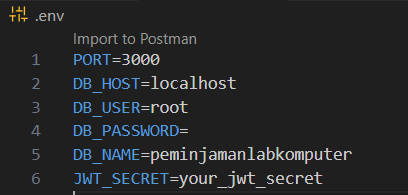
File publicApi.js mendefinisikan rute untuk mengakses waktu dunia berdasarkan timezone. Rute ini dilindungi oleh middleware authenticateToken, yang memastikan hanya pengguna yang terautentikasi yang dapat mengaksesnya. Rute tersebut kemudian memanggil fungsi getWorldTime dari controller publicApiController.js untuk mengambil data waktu dari API eksternal.



12) Setelah semua folder dan file di buat langkah selanjutnya membuat file .env untuk penyambungan database,dan index.js untuk mengatur gerbang utamanya.

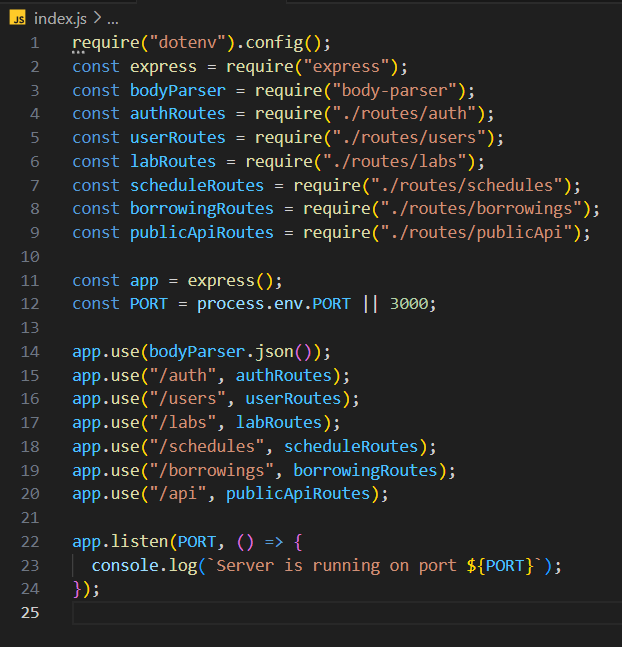
1. .env

File **.env** berfungsi untuk menyimpan variabel lingkungan (environment variables) yang digunakan dalam aplikasi, seperti konfigurasi port dan pengaturan database. File ini menjaga agar informasi sensitif tidak langsung tercatat dalam kode.

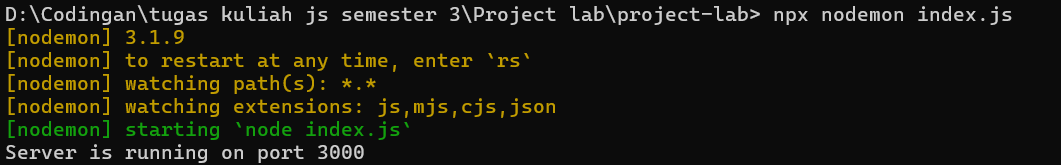


1. Index.js

index.js adalah untuk menginisialisasi dan menjalankan server Express serta mengatur rute-rute API yang akan digunakan oleh aplikasi. index.js menggunakan require("dotenv").config(); untuk memuat variabel lingkungan yang ada di dalam file .env ke dalam aplikasi, sehingga variabel tersebut bisa diakses dengan process.env.<variable\_name>.



13) Jalankan program dengan perintah **npx nodemon index.js/node index.js** pada terminal,untuk implementasi program kami menggunakan localhost dan Postman

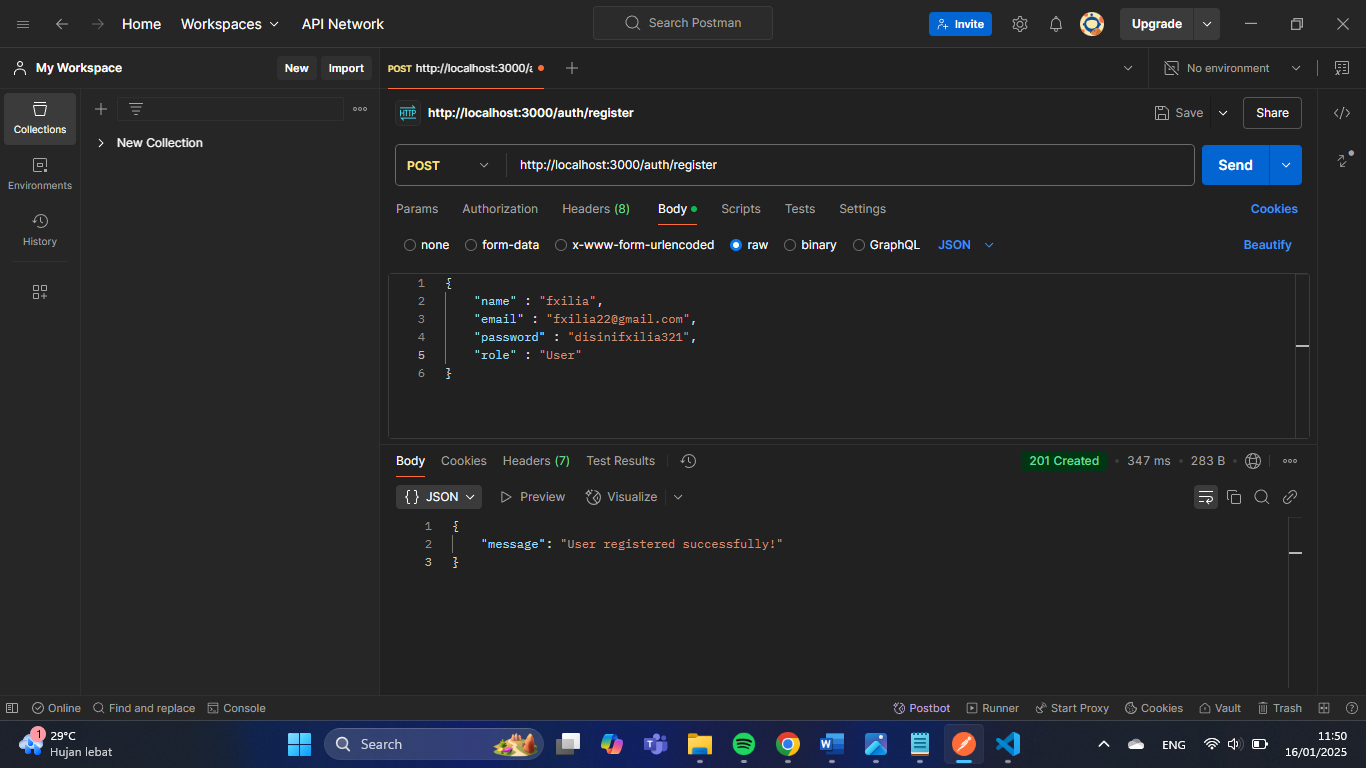


Apabila sudah seperti gambar di atas endpoint sudah bisa digunakan.

III. Implementasi Endpoint

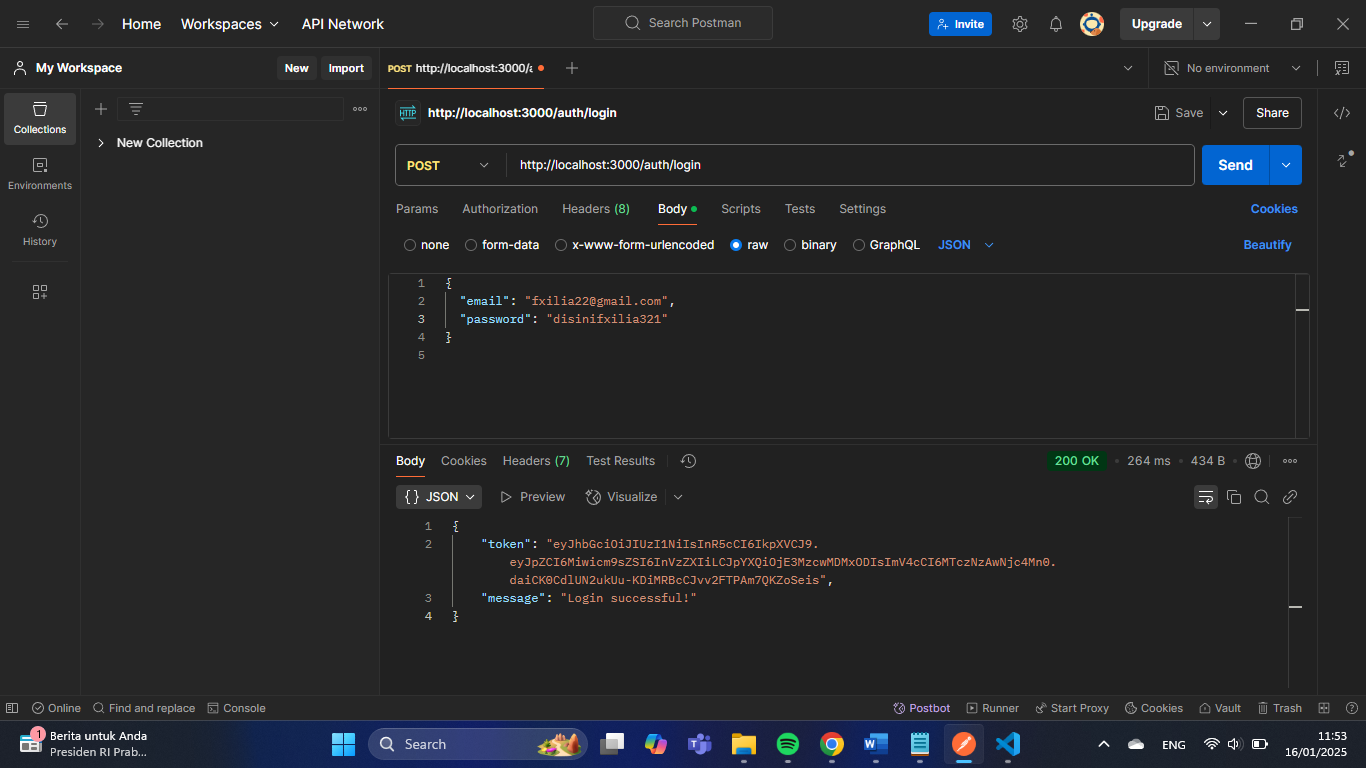
1.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| api/auth/register | POST | Mendaftar pengguna baru | - | {  “name” : “ fxilia”,  “email” : “fxilia22@gmail.com”,  “password” : “disinifxilia321”,  “role” : “User”  } | - |



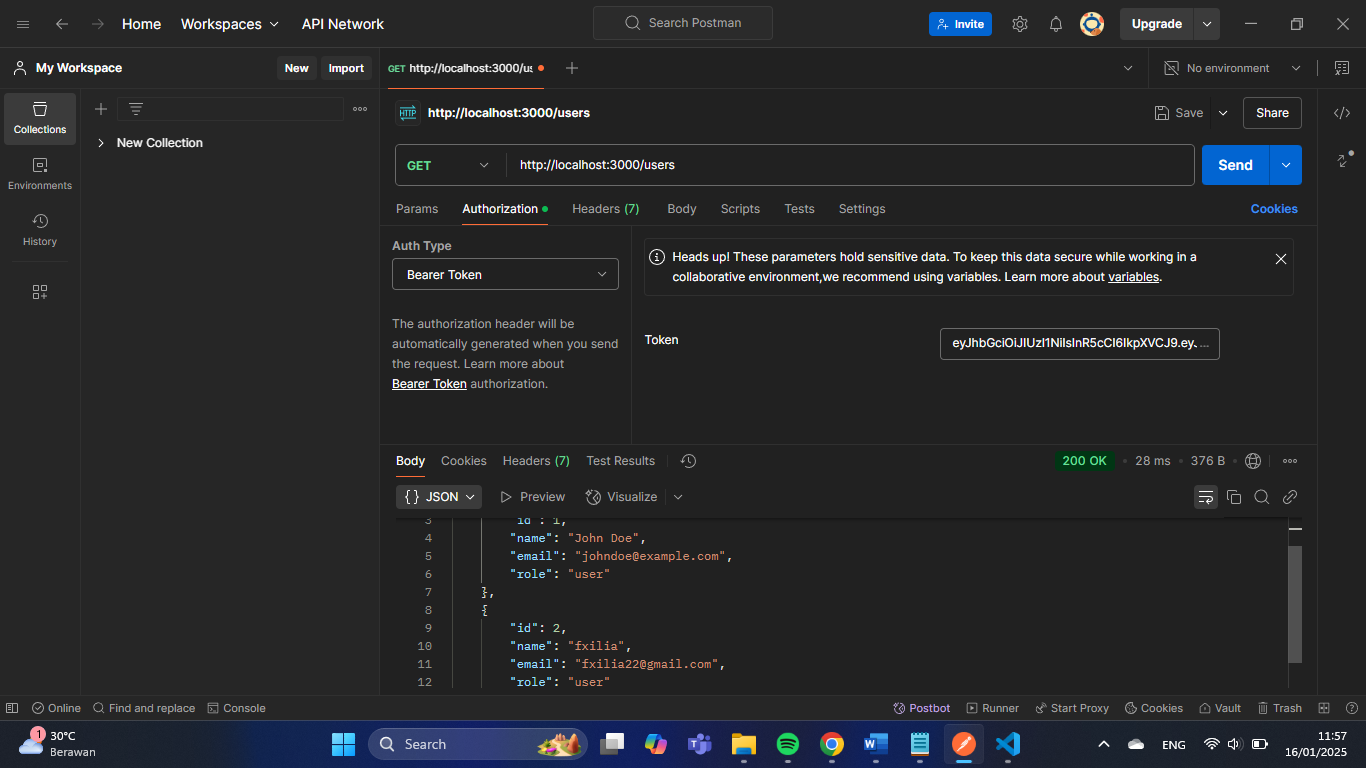
2.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/auth/login | POST | Login pengguna | - | {  “email” : “fxilia22@gmail.com”,  “password” : “disinifxilia321”  } | - |



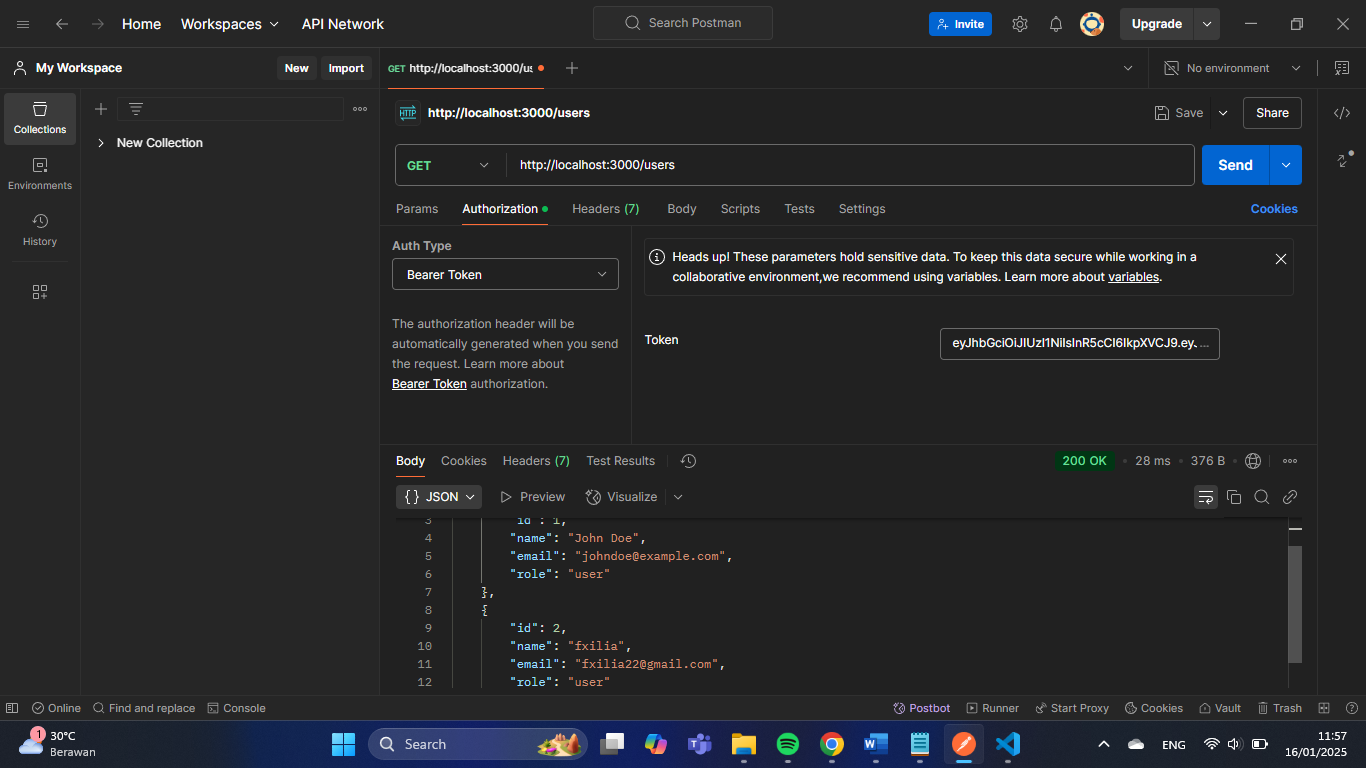
3.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/users | GET | Mendapatkan daftar semua pengguna | - | - | Authorization: Bearer <JWT> |



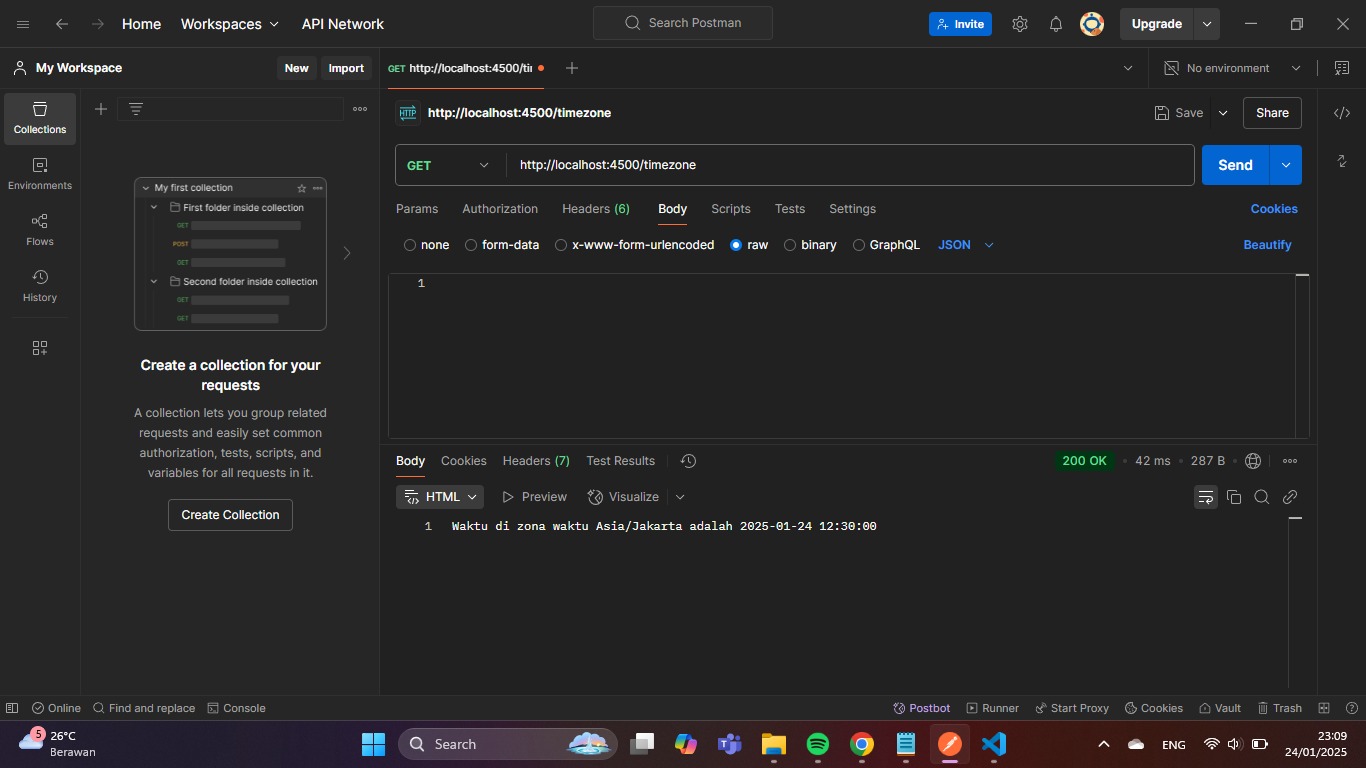
4.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/users/id | GET | Mengambil detail pengguna berdasarkan id | Id | - | Authorization: Bearer <JWT> |



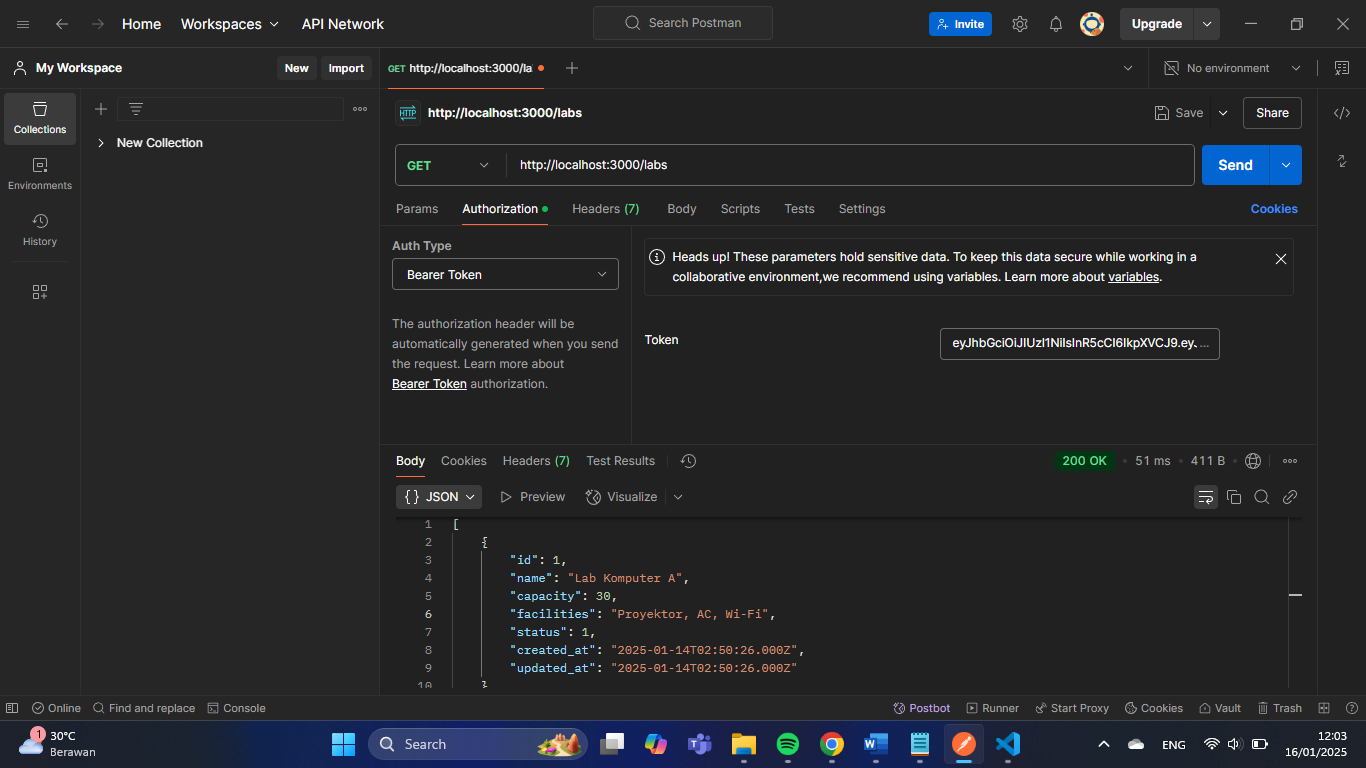
5.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/users/id | DELETE | Menghapus pengguna | id | - | Authorization: Bearer <JWT> |



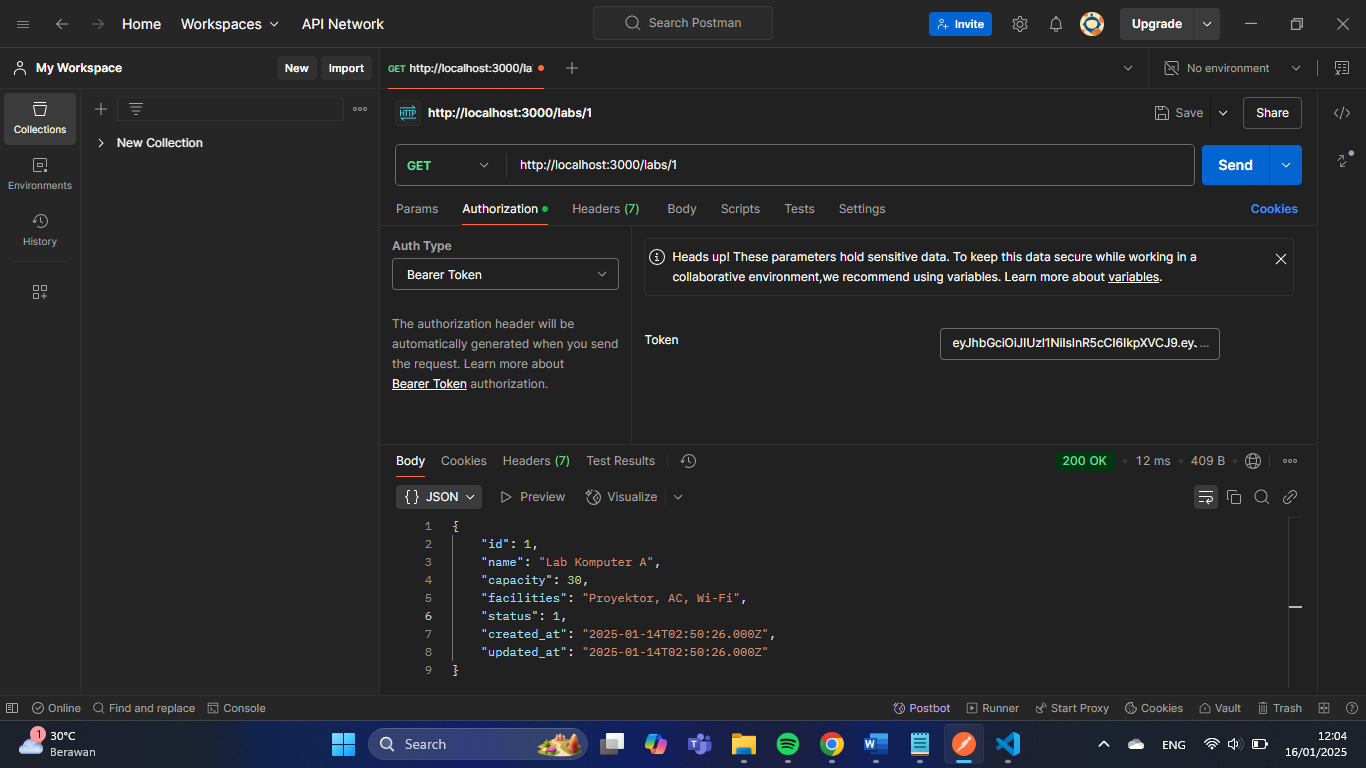
6.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/labs | GET | Mendapatkan daftar semua lab |  | - | Authorization: Bearer <JWT> |



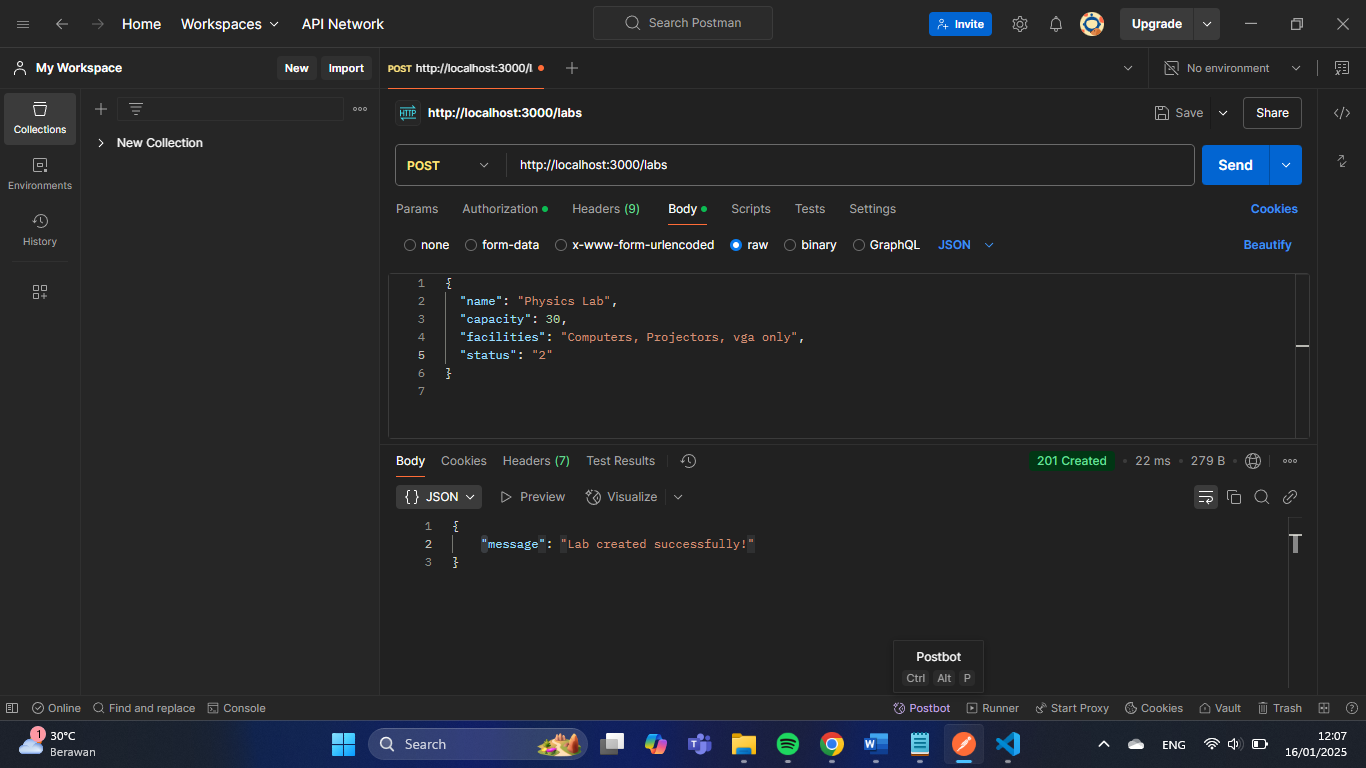
7.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/labs/id | GET | Mengambil detail lab berdasarkan id | id | - | Authorization: Bearer <JWT> |



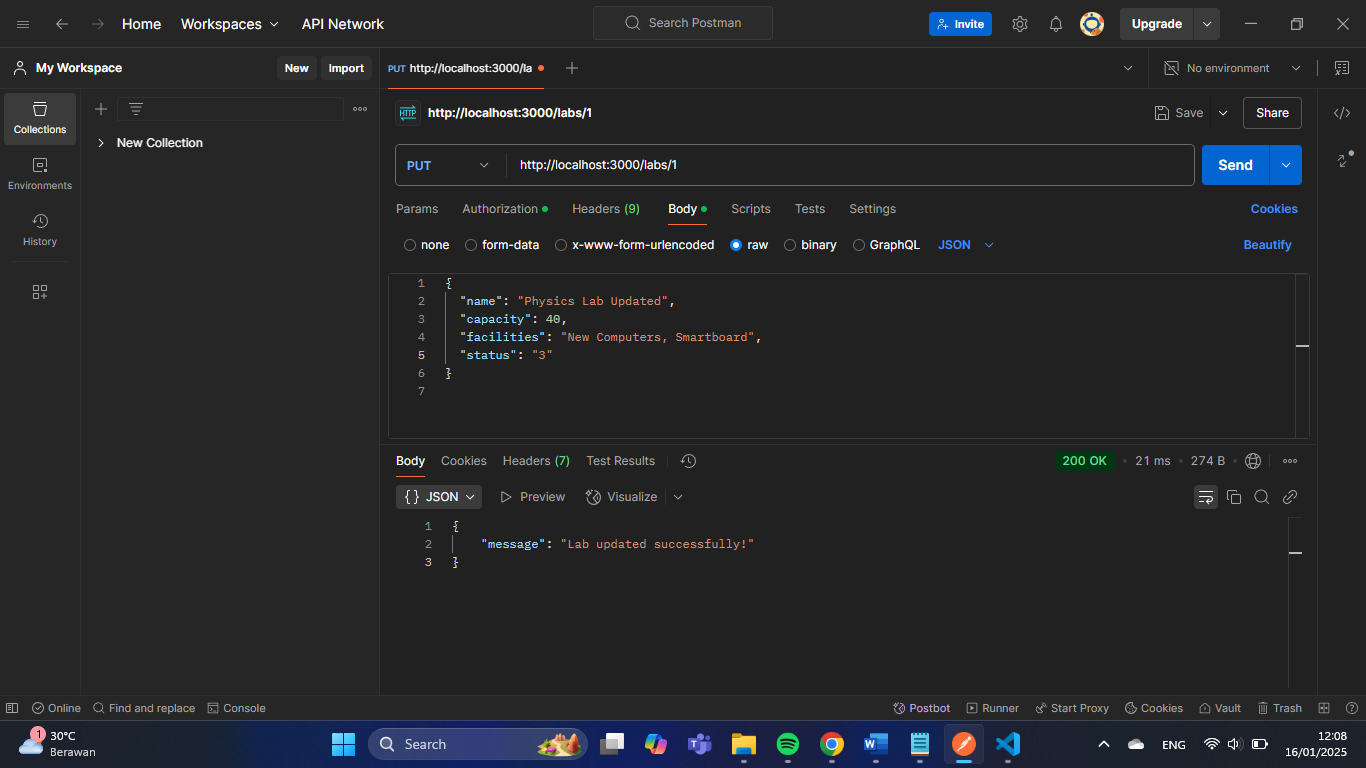
8.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/labs | POST | mendaftar lab | - | {  “name” : “Physisc Lab”,  “capacity”: 30,  “facilities”: “Computers, Projectirs, vga only”,  “status”: “2”  } | Authorization: Bearer <JWT> |



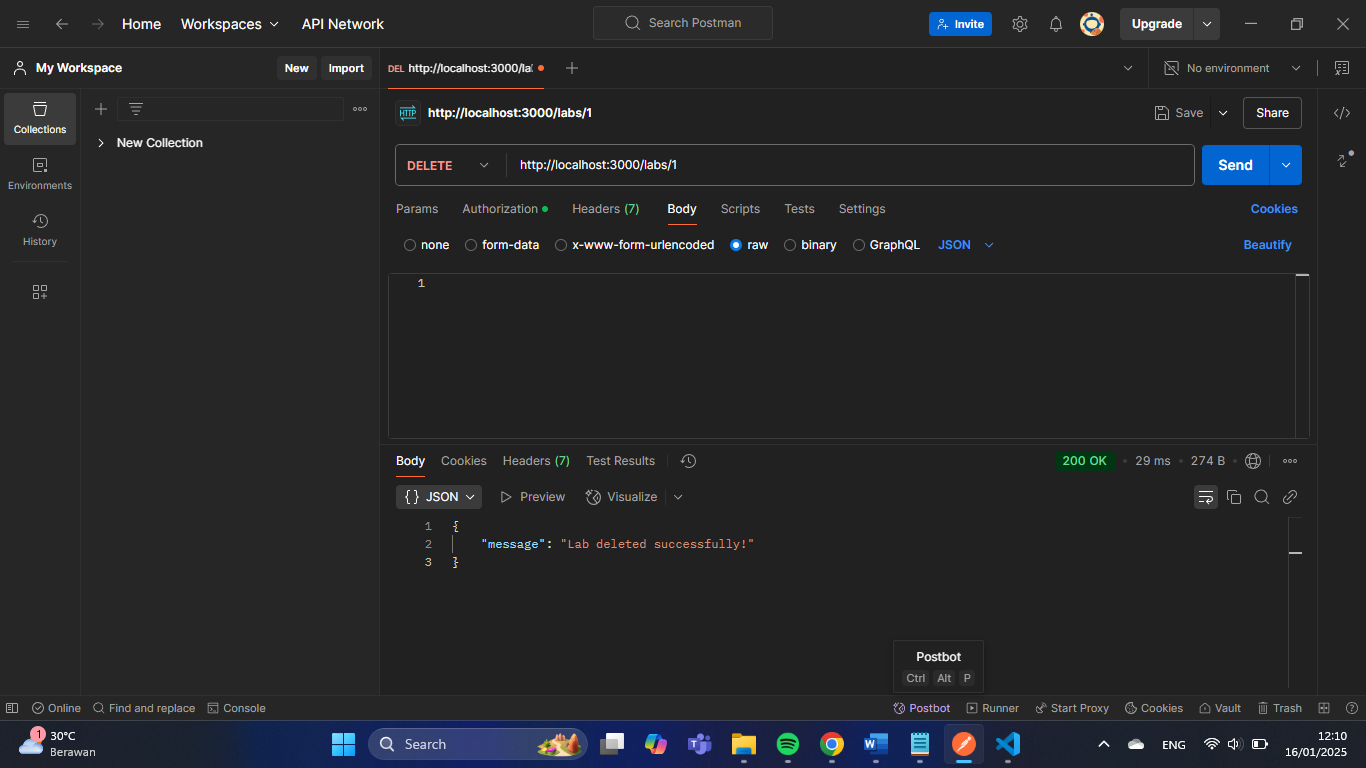
9.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/labs/id | PUT | Merubah lab | id | {  “name” : “Physisc Lab”,  “capacity”: 40,  “facilities”: “Computers, Projectirs, vga only”,  “status”: “2”  } | Authorization: Bearer <JWT> |



10.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/labs/id | DELETE | Mengapus lab | id | - | Authorization: Bearer <JWT> |



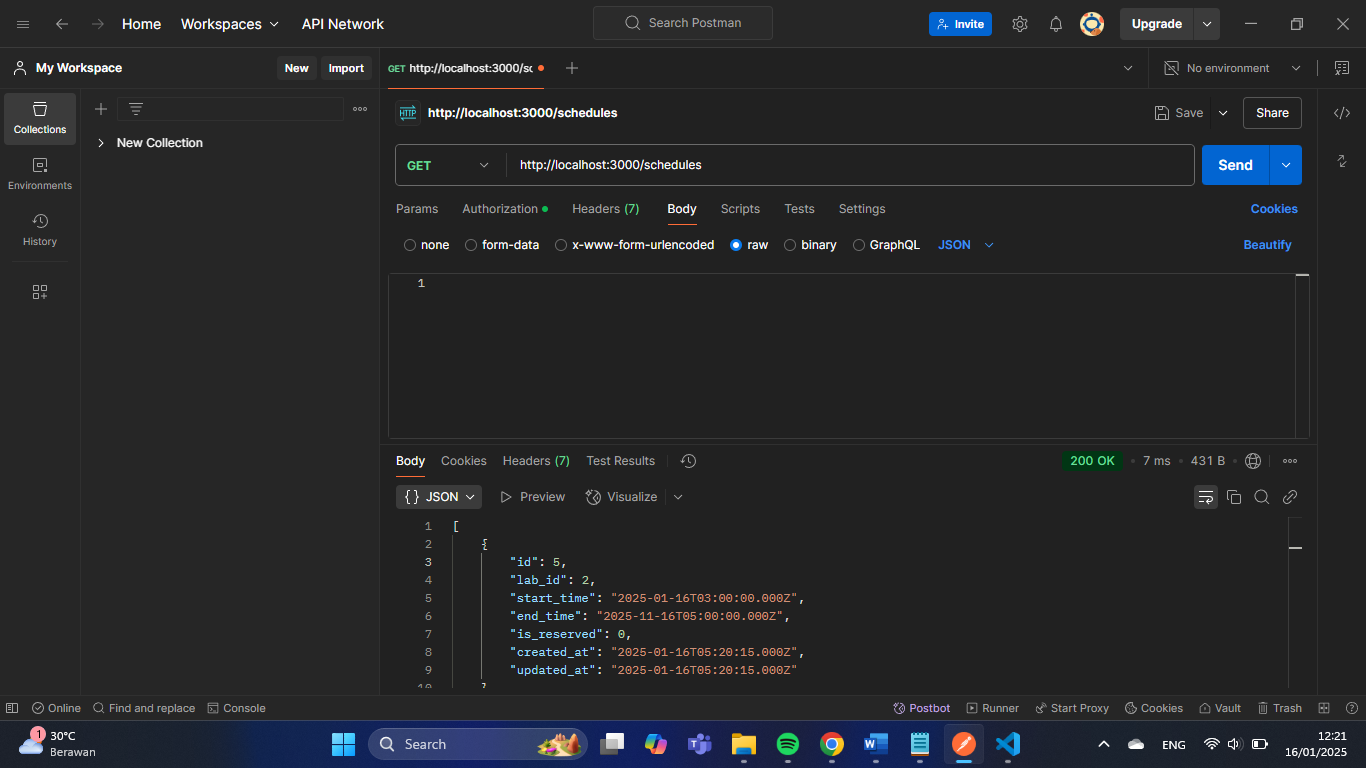
11.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/schadules | POST | Mendaftarkan jadwal | - | {  “lab\_id” : 2,  “start\_time”: “2025-01-16 10:00:00”,  “end\_time”: “2025-01-16 12:00:00”,  “is\_reserved”: false  } | Authorization: Bearer <JWT> |



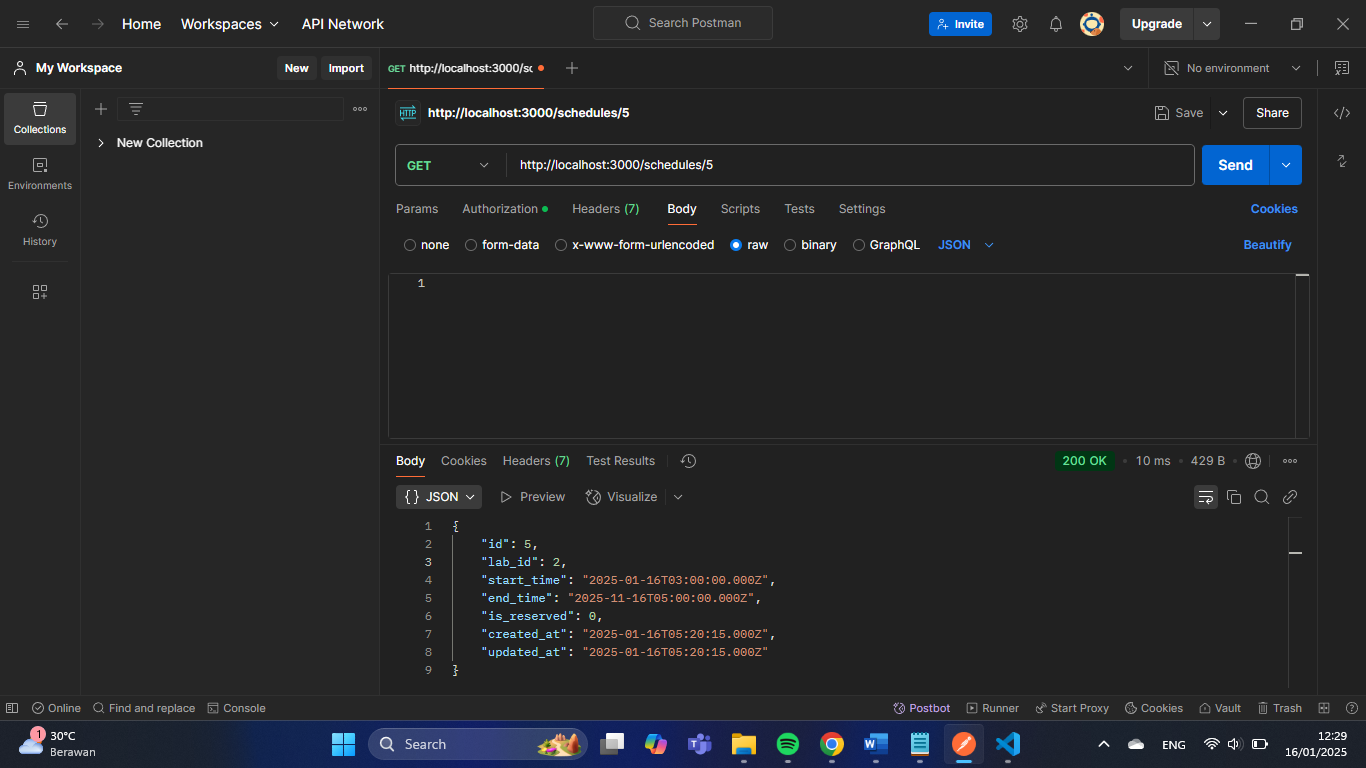
12.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/schadules | GET | Mendapatkan daftar semua jadwal | - | - | Authorization: Bearer <JWT> |



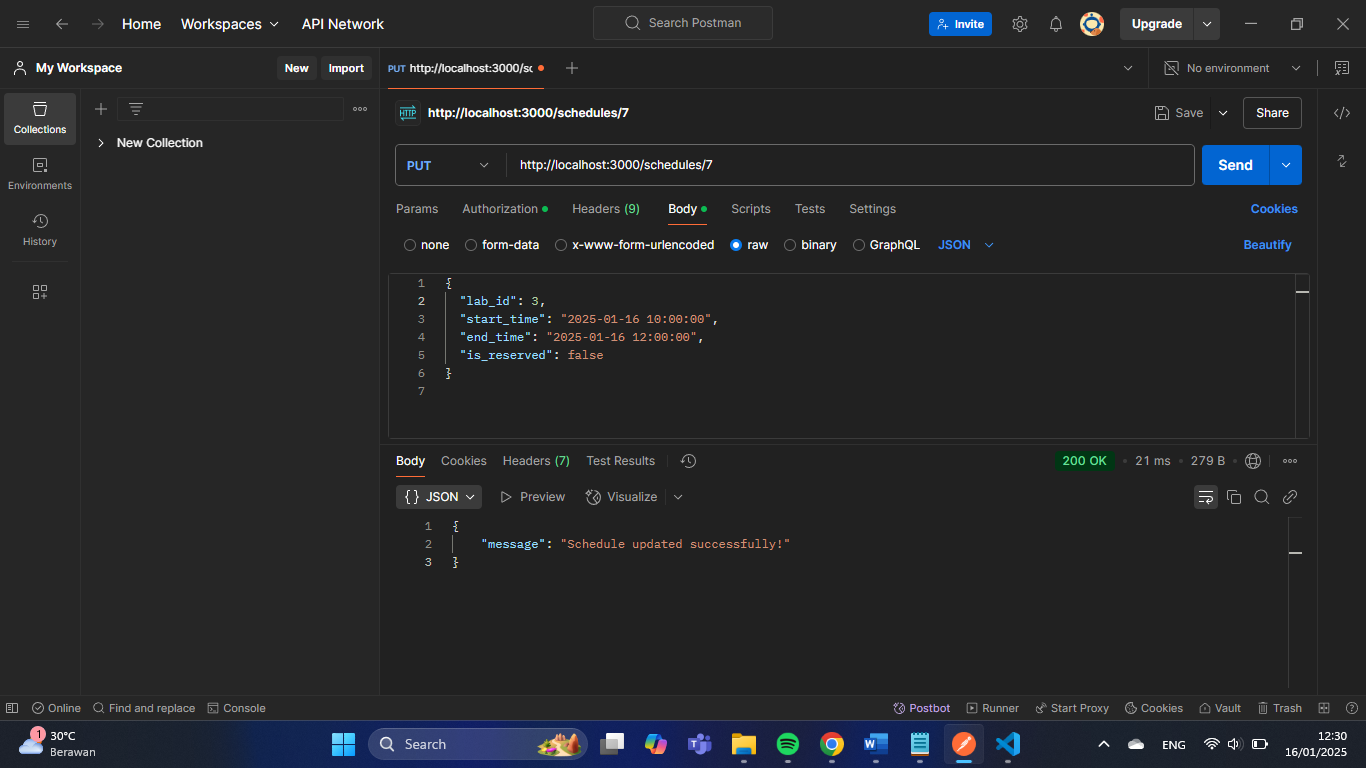
13.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/schadules/id | GET | Mengambil detail jadwal berdasarkan id | id | - | Authorization: Bearer <JWT> |



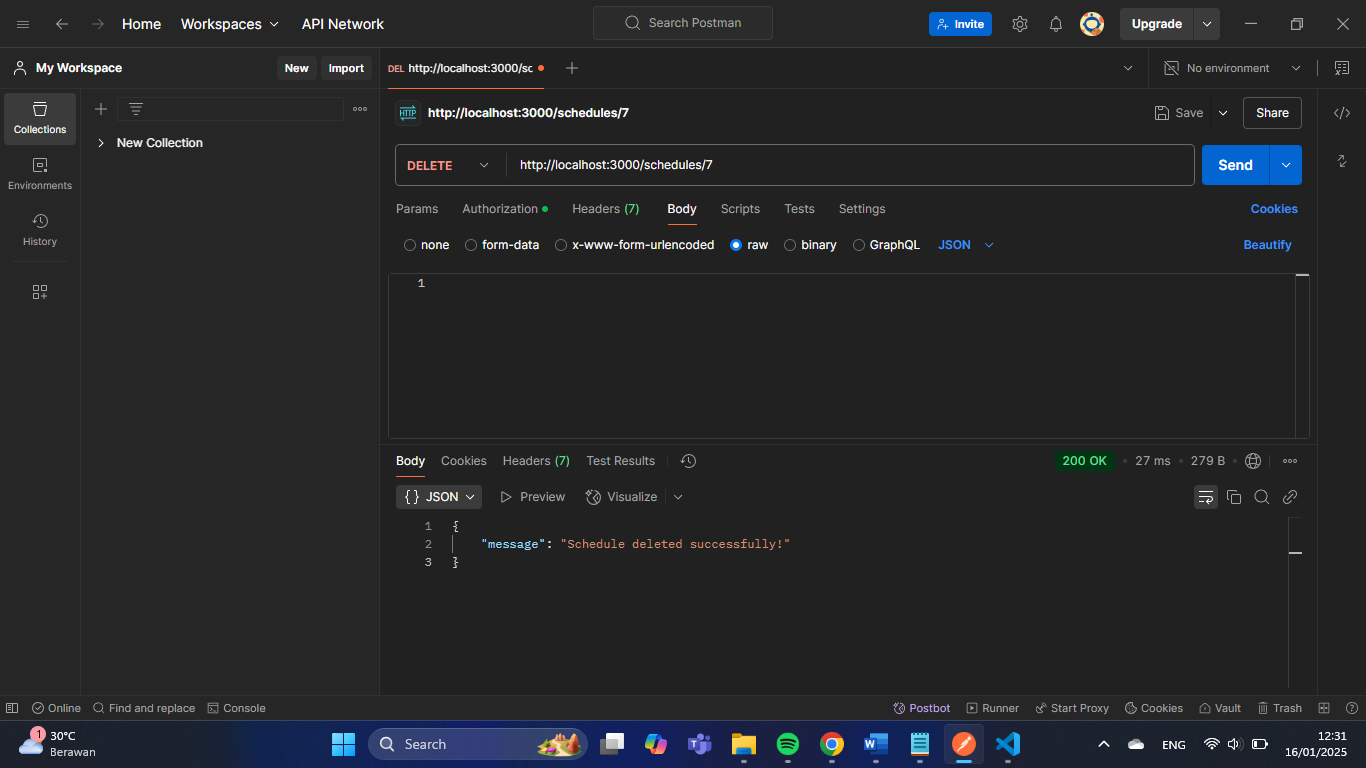
14.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/schadules/id | PUT | Merubah jadwal | id | {  “lab\_id” : 3,  “start\_time”: “2025-01-16 10:00:00”,  “end\_time”: “2025-01-16 12:00:00”,  “is\_reserved”: false  } | Authorization: Bearer <JWT> |



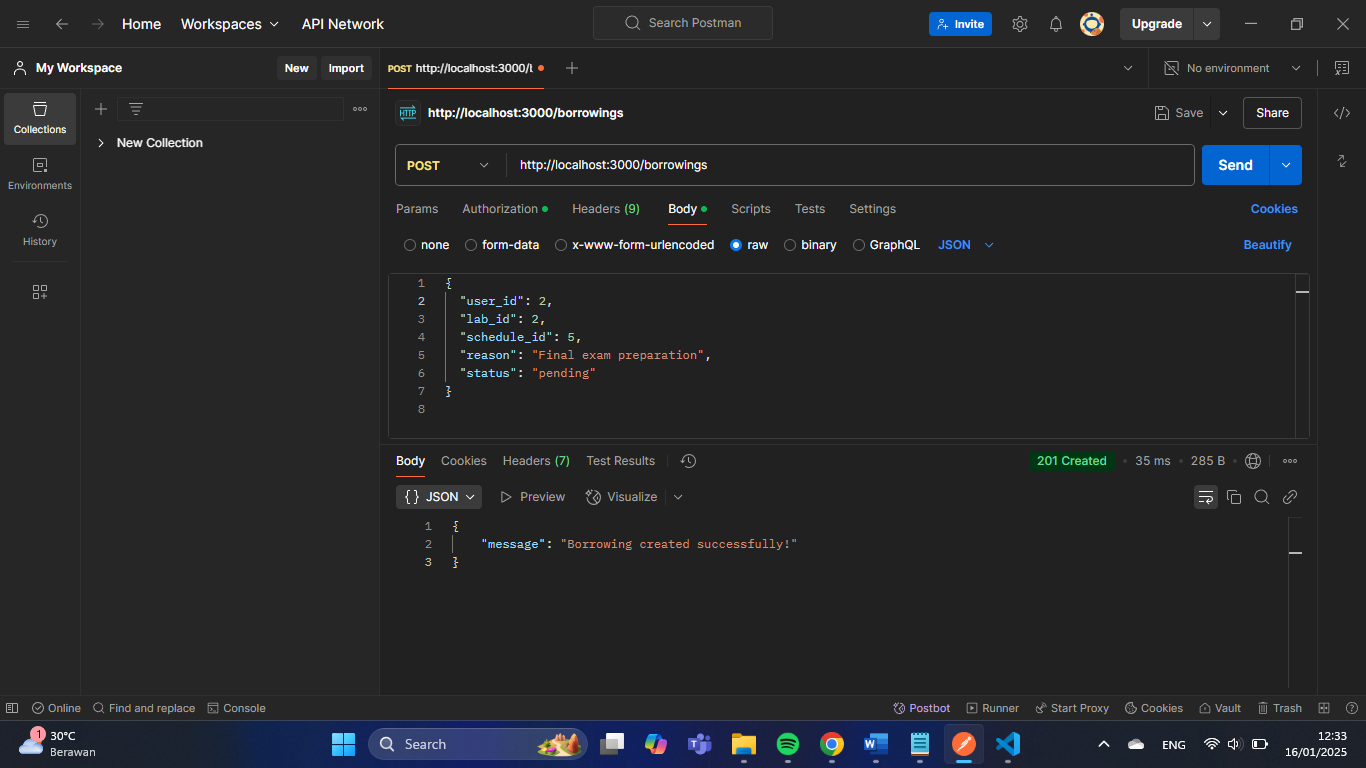
15.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/schadules/id | DELETE | Menghapus jadwal | id |  | Authorization: Bearer <JWT> |



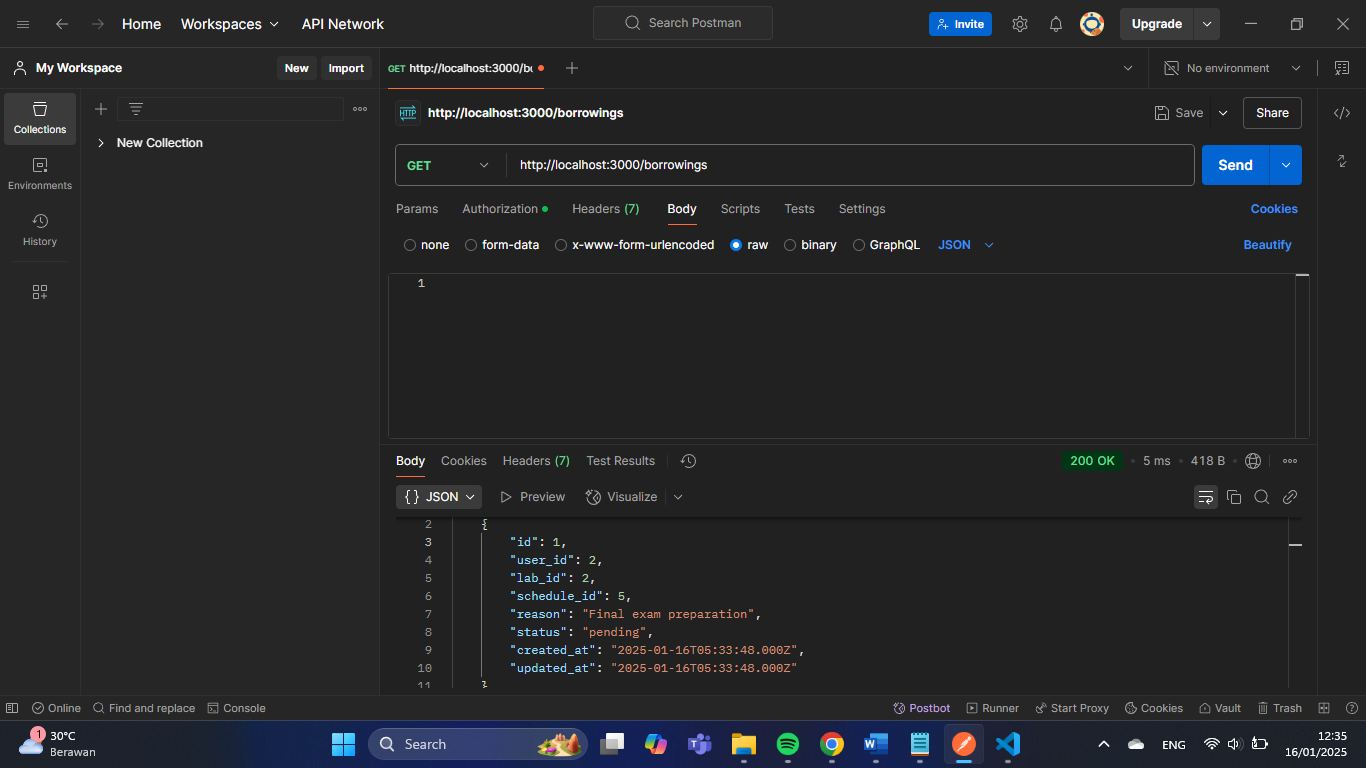
16.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/borrowings | POST | Mendaftarkan pemesanan | - | {  “user\_id” : 2,  “lab\_id”: 2,  “schedule\_id”: 5,  “reason”: “Final exam preparation”  “status”:”pending”  } | Authorization: Bearer <JWT> |



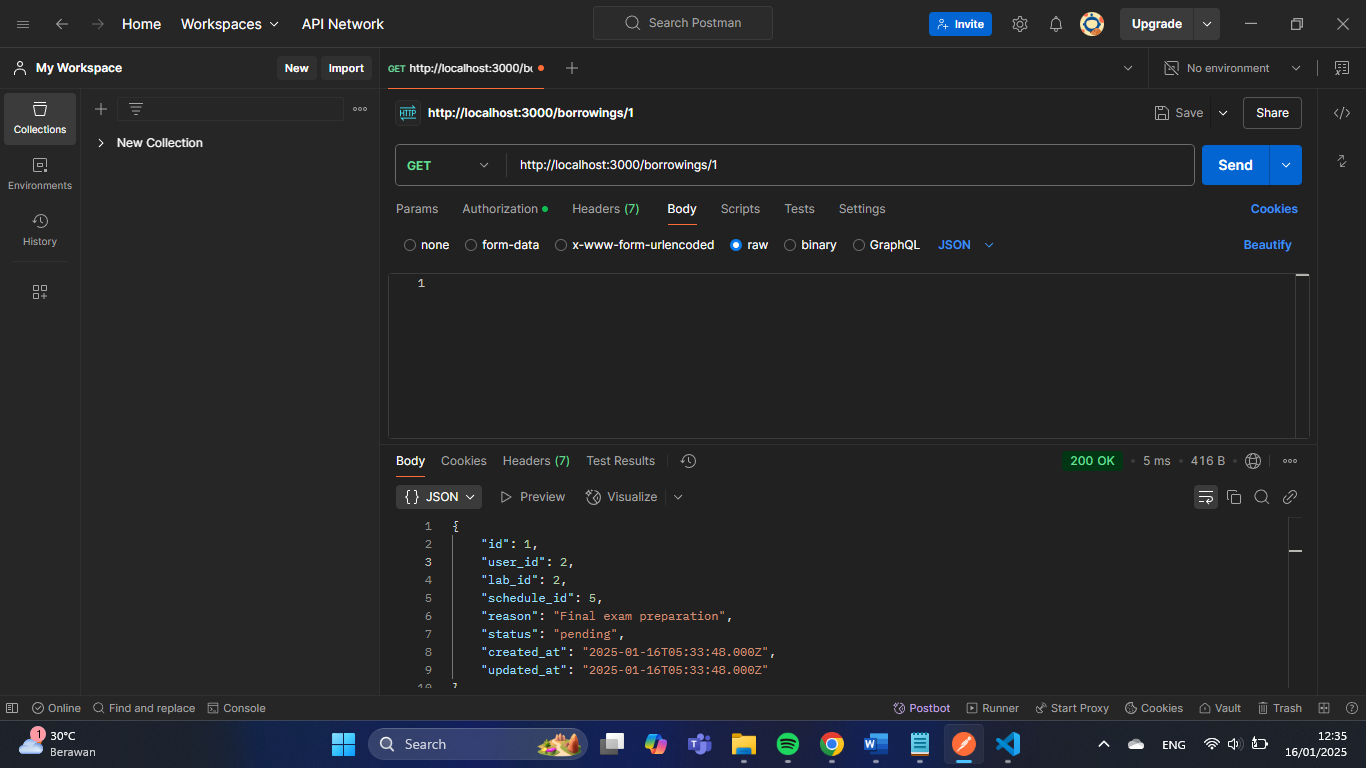
17.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/borrowings | GET | Mendapatkan daftar semua pemesanan | - | - | Authorization: Bearer <JWT> |



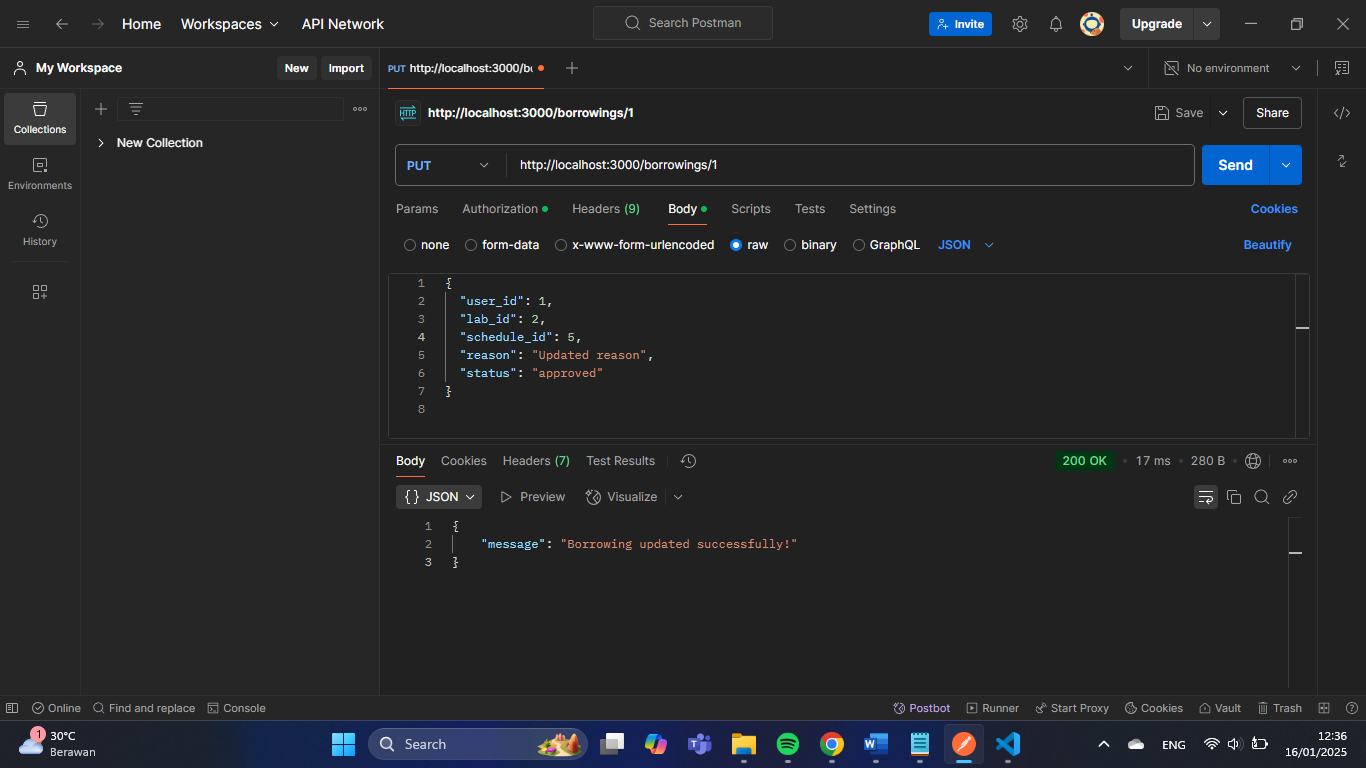
18.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/borrowings/id | GET | Mengambil detail pemesanan berdasarkan id | id | {  “user\_id” : 2,  “lab\_id”: 2,  “schedule\_id”: 5,  “reason”: “Final exam preparation”  “status”:”pending”  } | Authorization: Bearer <JWT> |



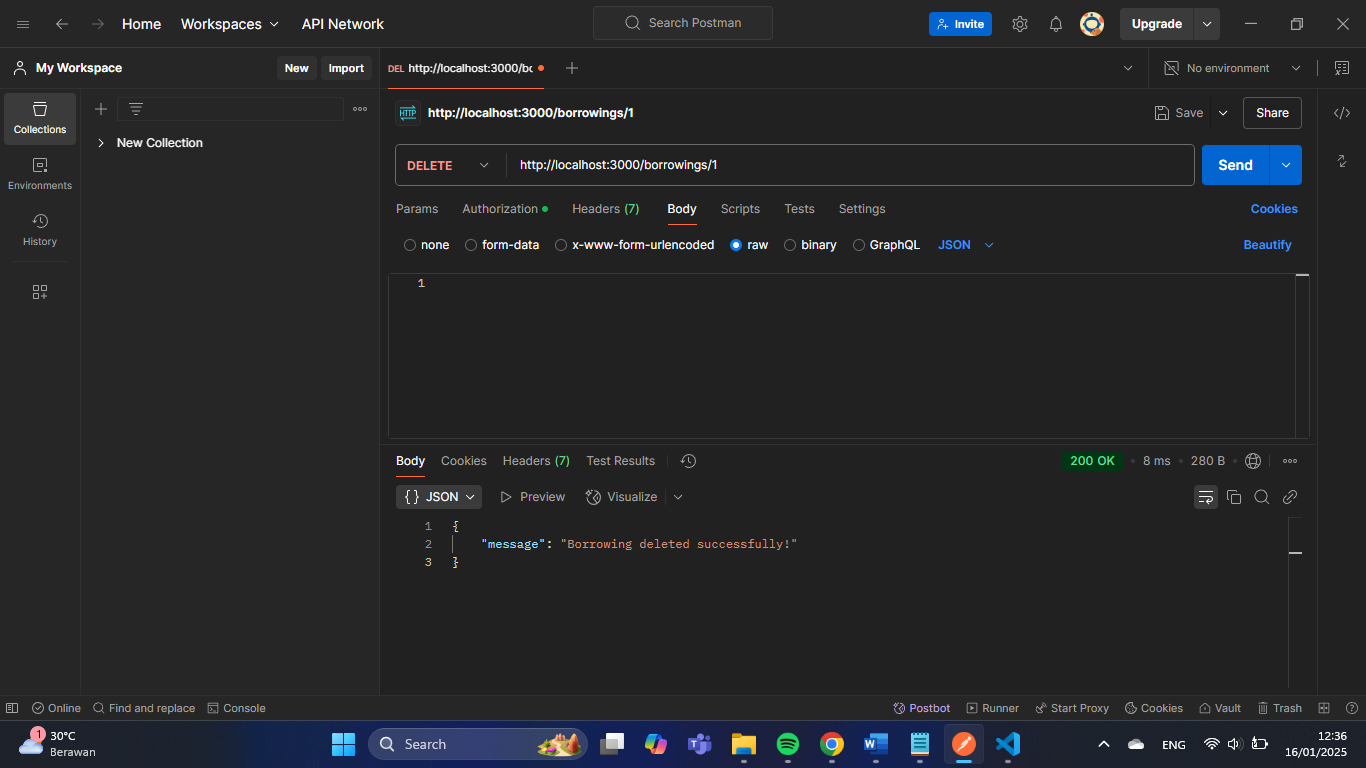
19.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/borrowings/id | PUT | Menambah pemesanan | id | {  “user\_id” : 1,  “lab\_id”: 2,  “schedule\_id”: 5,  “reason”: “Final exam preparation”  “status”:”pending”  } | Authorization: Bearer <JWT> |



20.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Endpoint | HTTP Method | Description | Parameter | Request Body | Request Header |
| Api/borrowings/id | DELETE | Menghapus pemesanan | id | - | Authorization: Bearer <JWT> |



selesai