Laporan Praktikum WSE #3

Mata Kuliah : Web Service Engineering
Dosen Pengampu : Muhayat, M.IT
Praktikum : P3 – **Web Service Architecture**
Nama Mahasiswa : Zaidannur Muzanni
NIM : 230104040225
Kelas : TI23A
Tanggal Praktikum : 06-10-2025

A. Tujuan Praktikum

1. Memahami perbedaan arsitektur Client–Server, SOA, dan Microservices.
2. Membangun layanan sederhana yang sesuai setiap arsitektur.
3. Menguji komunikasi antar komponen dengan tools (Postman/Browser).
4. Menganalisis kelebihan & kekurangan tiap arsitektur melalui studi kasus.

B. Alat & Bahan

1. Laptop / PC
2. Software: Node.js (Express)
3. IDE: VS Code
4. Testing : Postman, Browser.
5. Tool Diagram Arsitektur: miro.com
6. Koneksi internet aktif.

C. Langkah Kerja

1. Persiapan (15 menit)

- Memahami konsep arsitektur Web Service.
- Membuat skenario praktikum.

2. Eksperimen Client–Server (25 menit)

- Membuat server sederhana (Node.js/Python) dengan endpoint GET /hello.
- Uji dengan browser/Postman untuk memahami pola dasar request–response.
- Mencatat dan menyimpan screenshot hasil pengujian

3. Simulasi SOA (30 menit)

- Membuat 2 layanan terpisah: Authentication Service & Data Service.
- Uji integrasi keduanya → Data Service hanya bisa diakses jika melewati Authentication.
- Mencatat dan menyimpan screenshot hasil pengujian

4. Microservices Mini Project (50 menit)

- Membuat aplikasi mini menjadi 4 service (Authentication, Product, Order, Notification).
- Uji akses setiap service melalui API terpisah.
- Mencatat dan menyimpan screenshot hasil pengujian

5. Membuat Laporan Praktikum

- Judul & Tujuan
- Langkah Kerja & Kode Program
- Hasil Uji (screenshot Postman/Browser)
- Diagram Arsitektur
- Tabel Perbandingan • Analisis & Kesimpulan.

D. Hasil & Pembahasan

Port Terpakai:

No	Service	Port	Status
1	Client-Server (CS)	3001	OK
2	Auth Service (SOA)	4000	OK
3	Data Service (SOA)	5000	OK
4	API Gateway (MS)	3001	OK
5	Auth Service (MS)	4001	OK
6	Product Service (MS)	5001	OK
7	Order Service (MS)	5002	OK
8	Notification Service (MS)	5003	OK

Hasil Uji

1. ARSITEKTUR 1 – **CLIENT–SERVER**

1.1 Diagram Singkat

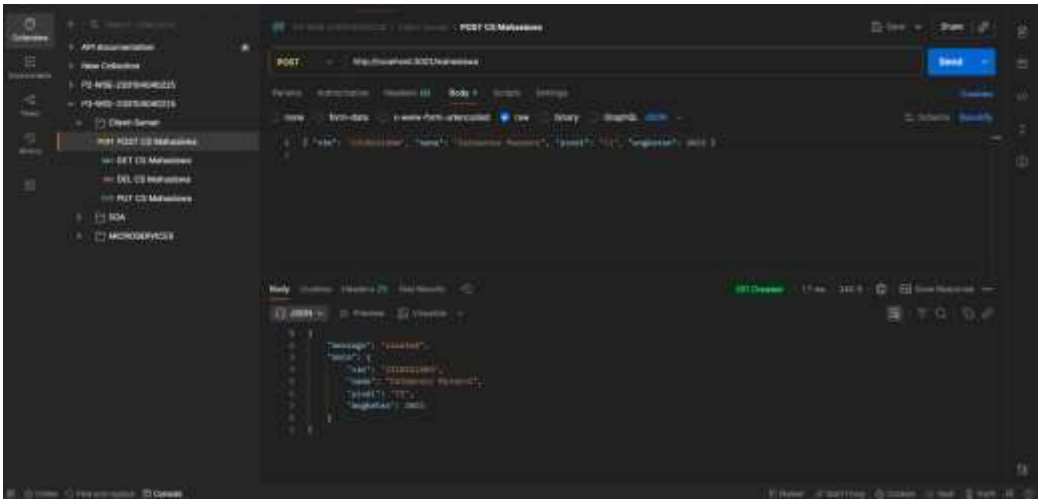
```
flowchart LR
  C[Client (Postman/Browser)] -->|HTTP| S[Server (Express)]
  S -->|Response| C
```

1.2 Skenario Uji & Bukti

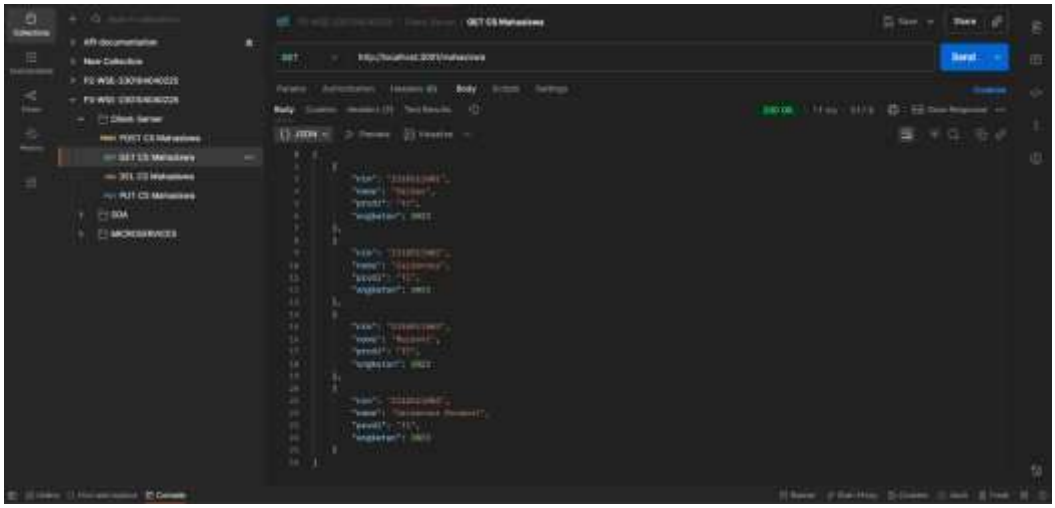
ID	Perintah / Endpoint	Input	Status	Message

CS1	POST http://localhost:3001/mahasiswa	{ "nim": "2310511004", "nama": "Zaidannur Muzanni", "prodi": "TI", "angkatan": 2023 }	201 Created	"Mahasiswa dibuat"
CS2	GET http://localhost:3001/mahasiswa	—	200 OK	Daftar Mahasiswa
CS3	DELETE http://localhost:3001/mahasiswa/2310511001	—	200 OK	"Mahasiswa dihapus"
CS4	http://localhost:3001/mahasiswa/2310511002	{ "prodi": "Teknologi Informasi", "angkatan": 2024 }	200 OK	"Mahasiswa diupdate"

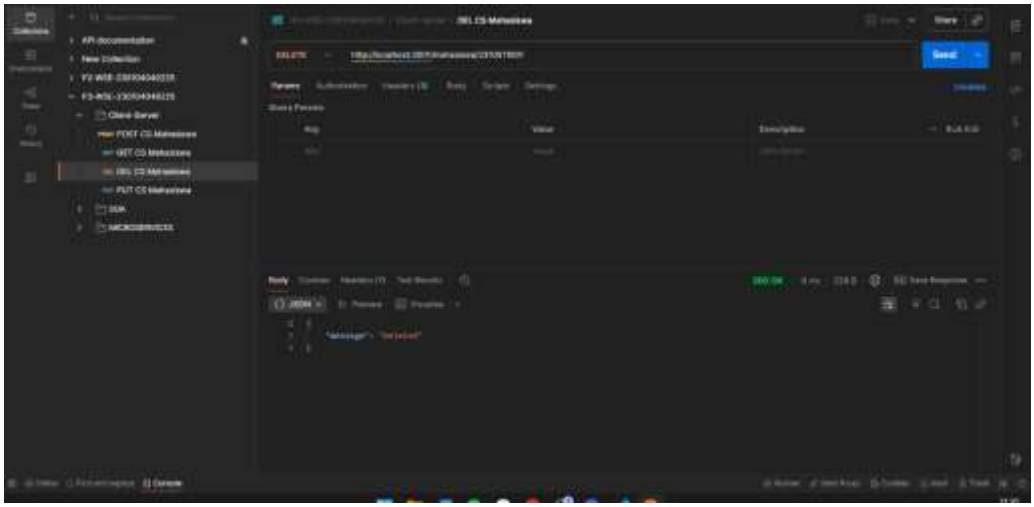
Screenshot CS1 POST http://localhost:3001/mahasiswa



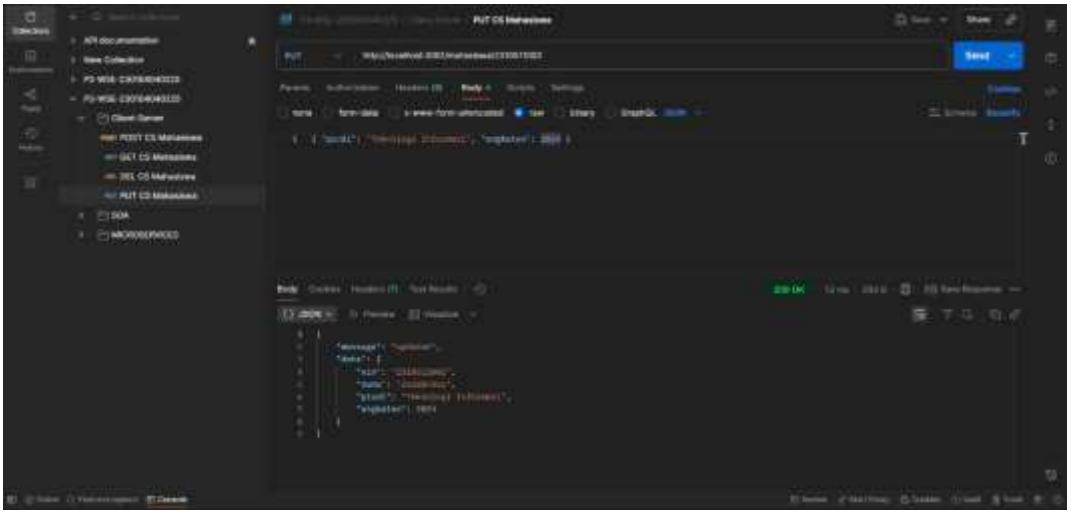
Screenshot CS2 – GET http://localhost:3001/mahasiswa



Screenshot CS3 : DELETE http://localhost:3001/mahasiswa/2310511001



Screenshot CS4 : PUT http://localhost:3001/mahasiswa/2310511002



2. ARSITEKTUR 2 – **SERVICE ORIENTED ARCHITECTURE (SOA)**

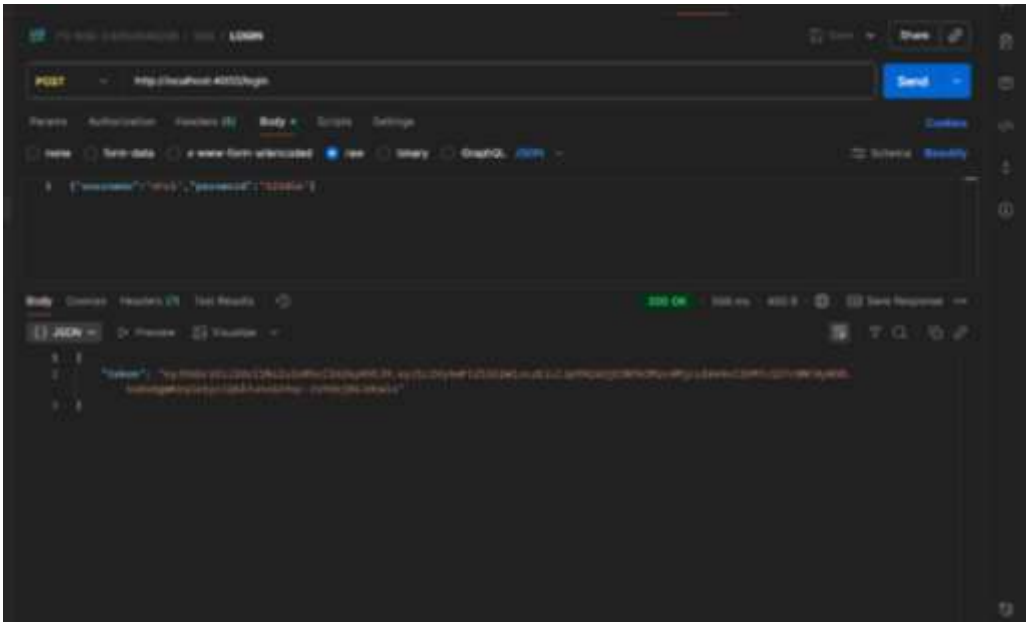
2.1 Diagram Singkat

```
flowchart LR
  Client -->|POST /login| Auth[(Auth Service :4000)]
  Client -->|GET /data \n Authorization: Bearer <JWT>| Data[(Data Service :5000)]
  Auth -->|JWT| Client
  Data -->|Data (200)| Client
```

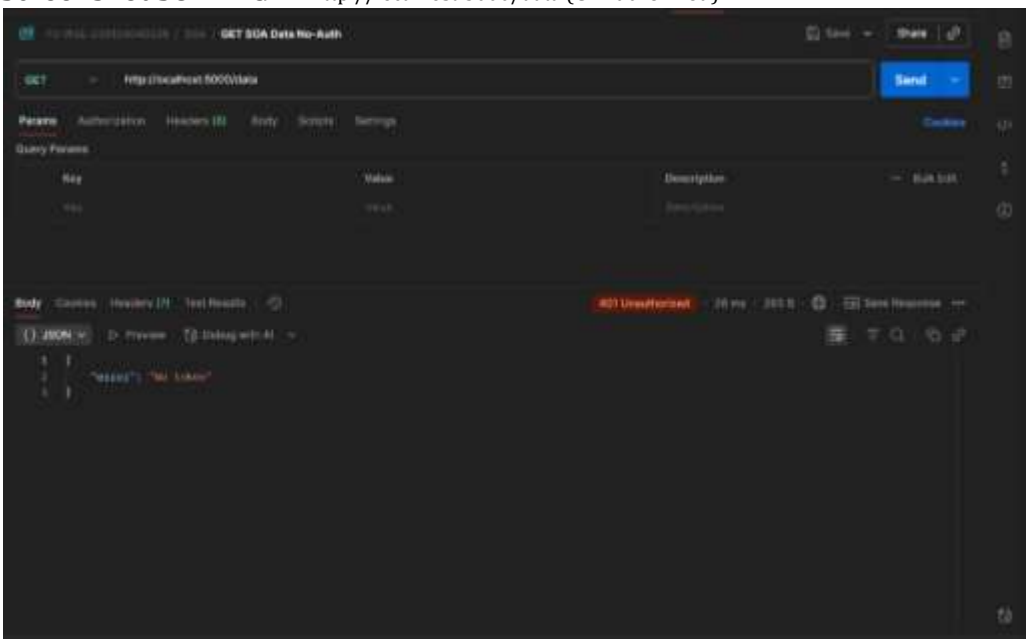
2.2 Skenario Uji & Bukti

ID	Endpoint	Header/Body	Ekspektasi	Status
SOA1	POST http://localhost:4000/login	{"username":"mhs1","password":"123456" }	200 + token	✔
SOA2	GET http://localhost:5000/data	(tanpa Authorization)	401 Anauthorized	✔
SOA3	GET http://localhost:5000/data	Authorization: Bearer <token>	200 + data milik user	✔
SOA4	POST http://localhost:5000/data	Header JSON + body {"name":"Data Rahasia C"} --> Ambil Token di POST Login	201 + item baru	✔

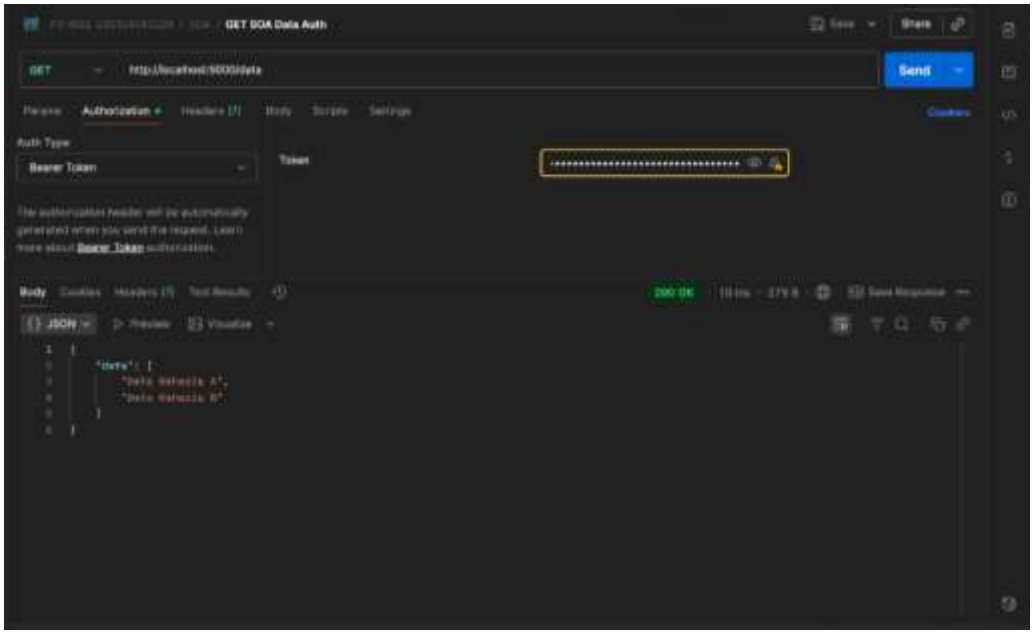
Screenshot SOA1 - POST http://localhost:4000/login



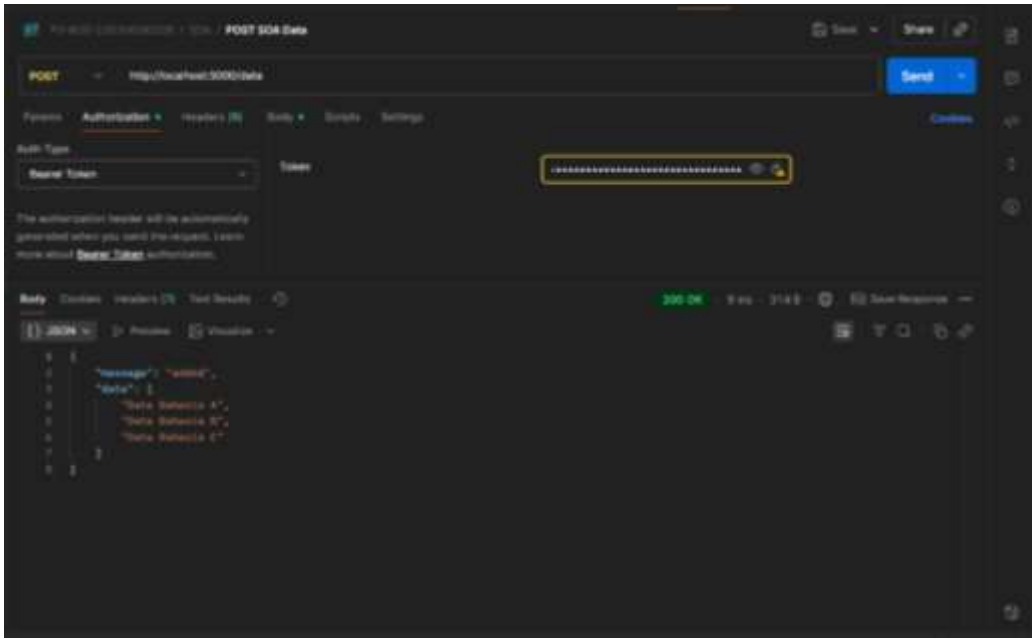
Screenshot SOA2 - GET http://localhost:5000/data (**UnAuthorized**)



Screenshot SOA3 - GET http://localhost: 5000/data (**Authorized**)



Screenshot SOA4 - POST http://localhost: 5000/data



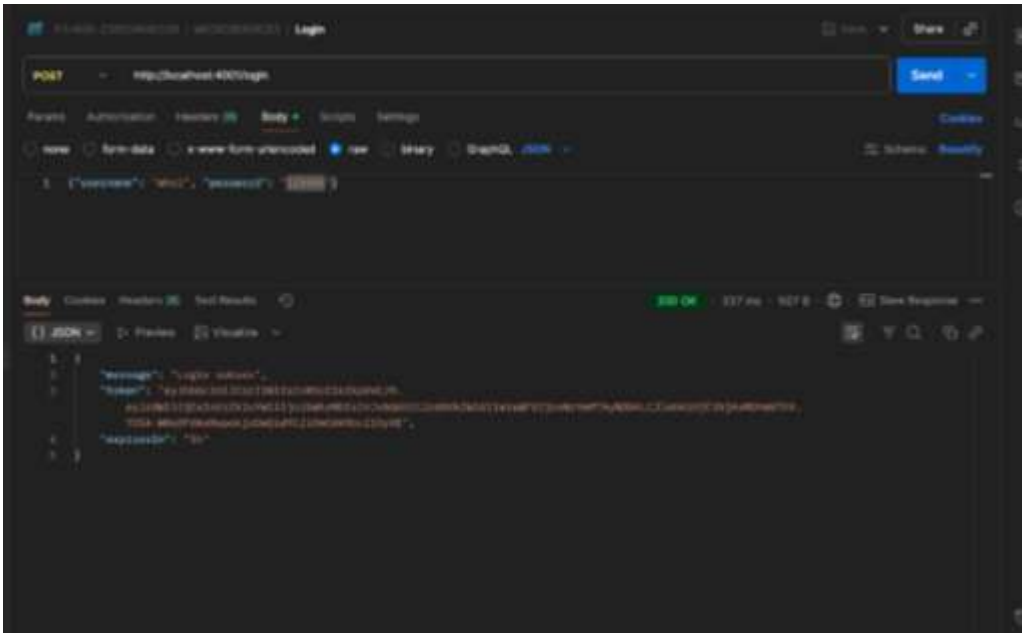
3. ARSITEKTUR 3 – **MICROSERVICE (MS) -** (Gateway + Auth + Product + Order + Notification)

3.1 Diagram Singkat

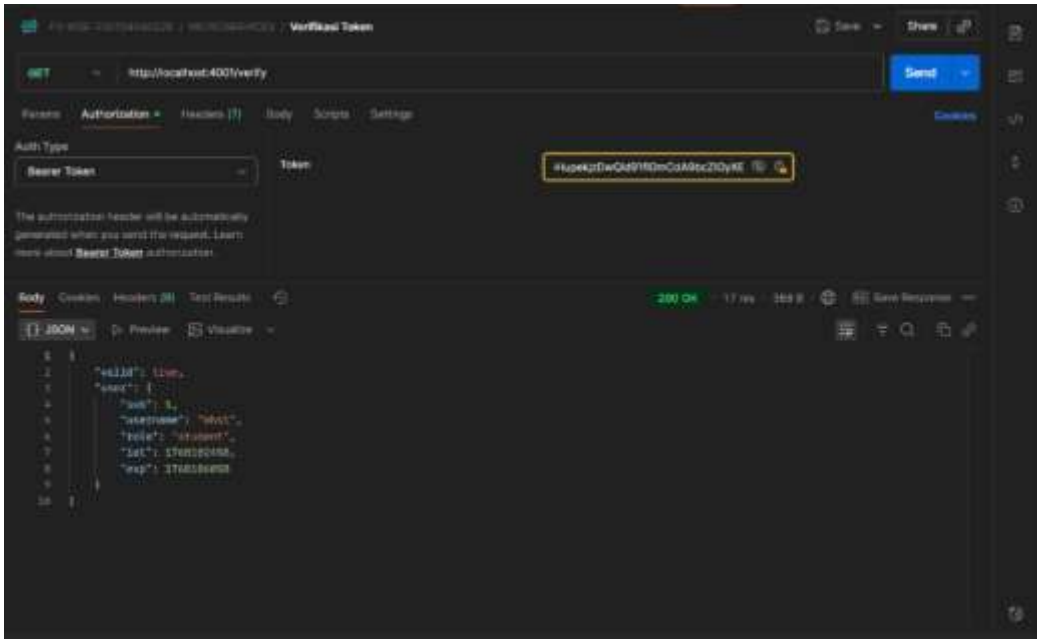
```
flowchart LR
  Client -.Bearer JWT .-> GW[API Gateway :3001]
  GW -->|/api/products| PS[Product :5001]
  GW -->|/api/orders| OS[Order :5002]
  GW -->|/api/notifications| NS[Notification :5003]
  OS -->|GET product/:id| PS
  OS -->|POST /notify| NS
```

NS -->|/notifications/my| GW --> Client

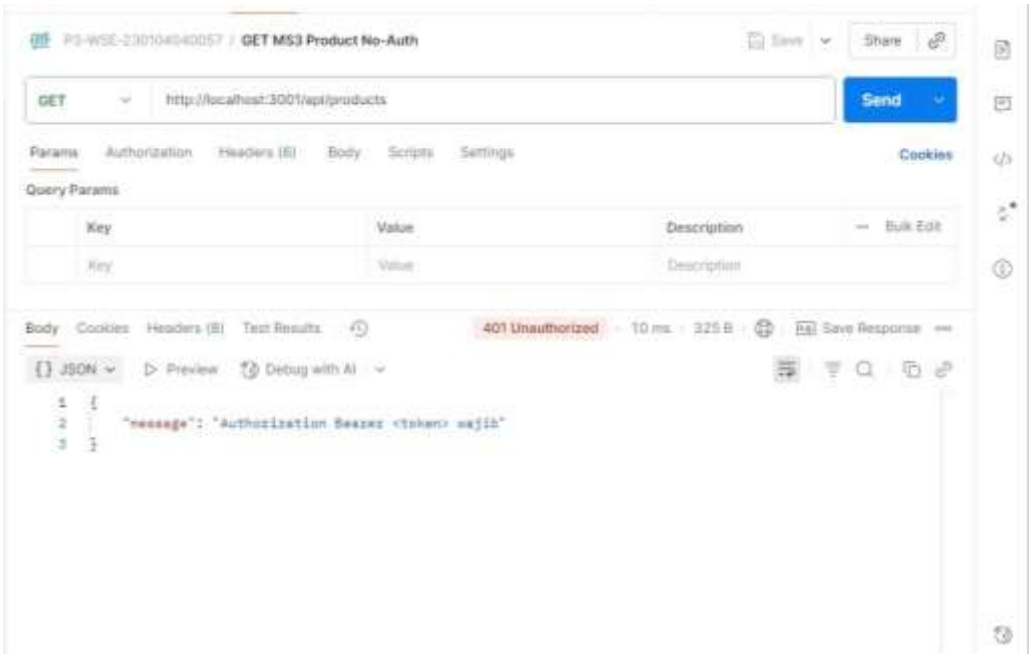
```

## 3.2 Skenario Uji & Bukti

| ID | Tujuan | Method / Endpoint | Header/Body | Ekspektasi | Catatan |
|---|---|---|---|---|---|
| MS1 | Ambil token (Login) via Gateway (jika proxy /auth sudah ada) | POST http://localhost:3001/auth/login | Headers: Content-Type: application/json Body: { "username":"mhs1","password":"123456" } | 200 → { "message":"Login sukses", "token":"<JWT>", "expiresIn":"1h" } (token dari AuthService) | Login via satu pintu (gateway) |
| MS2 | Verifikasi token via Gateway (opsional, jika diproxy) | GET http://localhost:3001/auth/verify | Headers: Content-Type: application/json Body Authorization: Bearer <JWT> • Token di ambil dari MS1 | 200 → { "valid": true, "decoded": { "sub":..., "username":"mhs1", "role":"student", "exp":... } } | Cek cepat masa berlaku token. |
| MS3 | Akses resource terproteksi tanpa token → harus ditolak | GET http://localhost:3001/api/products | (Tanpa header Authorization) | 401 → { "message":"Authorization Bearer <token> wajib" } | Menguji gatekeeper gateway. |
| MS4 | Akses resource terproteksi dengan token → diizinkan | GET http://localhost:3001/api/products | Headers: Authorization: Bearer <JWT> | 200 → daftar produk (sesuai implementasi Product-Service). | Verifikasi alur Client → Gateway → Product. |
| MS5 | Buat data di service terproteksi (contoh: tambah produk) | POST http://localhost:3001/api/products | Headers: Authorization: Bearer <JWT>, Content-Type: application/json Body: { "name":"Pensil 2B","price":5000 } | 201 Created → detail produk baru / konfirmasi sukses (dari ProductService). | Uji metode POST + propagate header user. |
| MS6 | Buat order (uji lintas service) | POST http://localhost:3001/api/orders | Headers: Authorization: Bearer <JWT>, Content-Type: application/json Body: { "productId":1,"qty":2 } | 201 Created → detail order (mis. total, status). Bisa memicu notifikasi async bila diimplementasi. | Uji orkestrasi lintas domain. |
| MS7 | Akses notif terproteksi (cek header | GET http://localhost:3000/api/notifications | Headers: Authorization: Bearer <JWT> | 200 OK { "data": [], "total": 0 | Menguji forwarding header user dari Gateway |
|  | dari gateway diteruskan) |  |  | } | ke Notification Service (Axios + onProxyReq) |

**Screenshot MS1 - POST** http://localhost:3001/auth/login
```



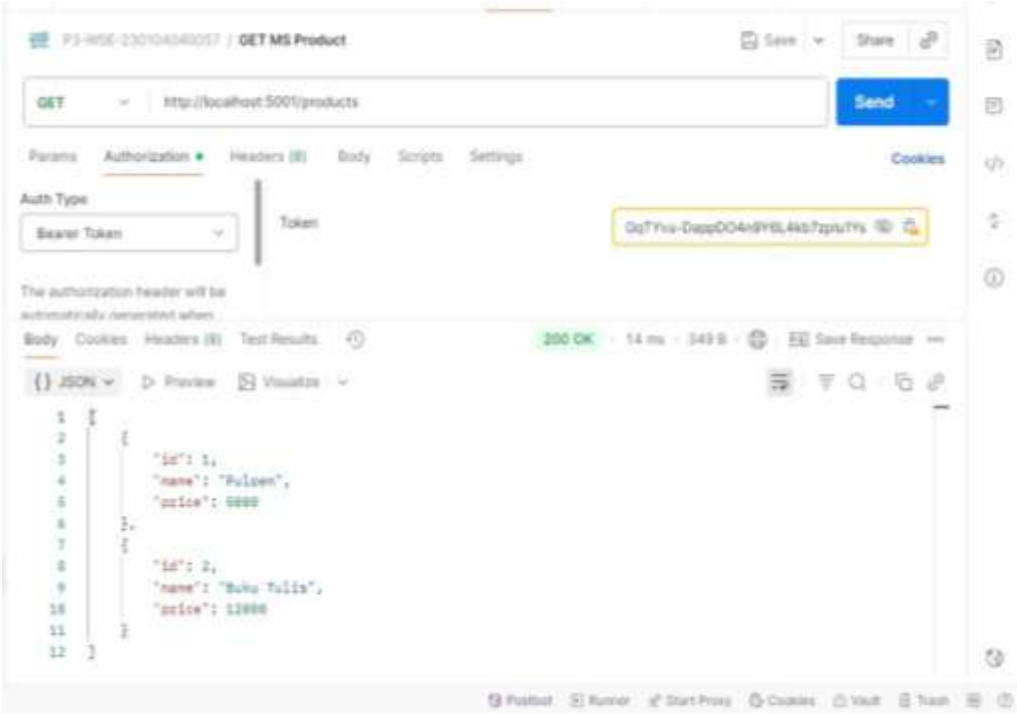
Screenshot MS2 - GET http://localhost:3001/auth/verify



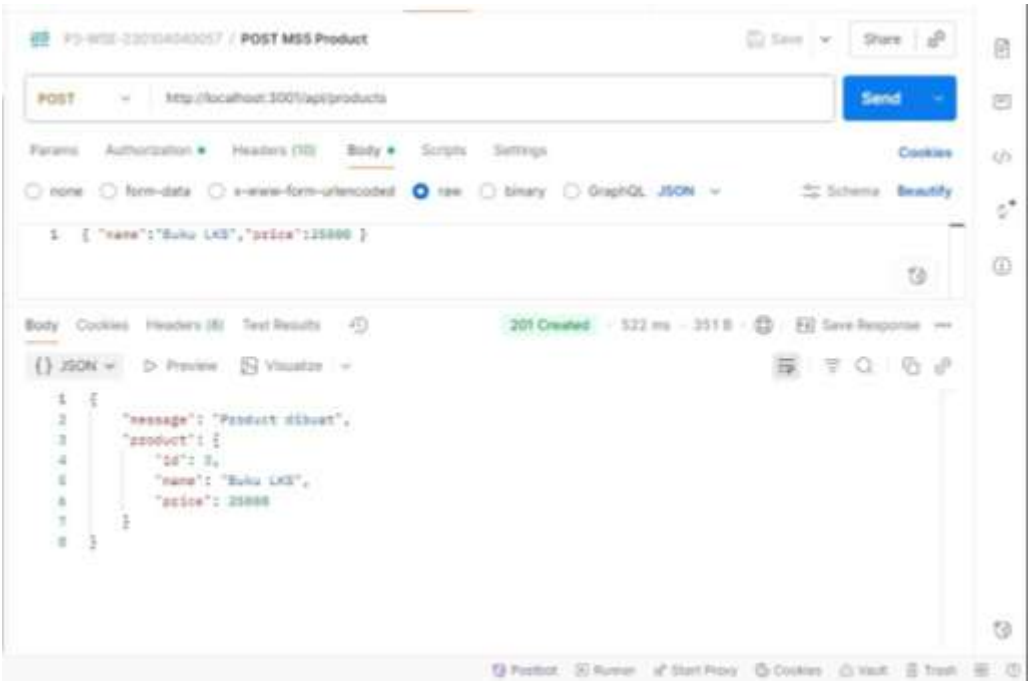
Screenshot MS3 - GET http://localhost:3001/api/products (Tanpa Token)



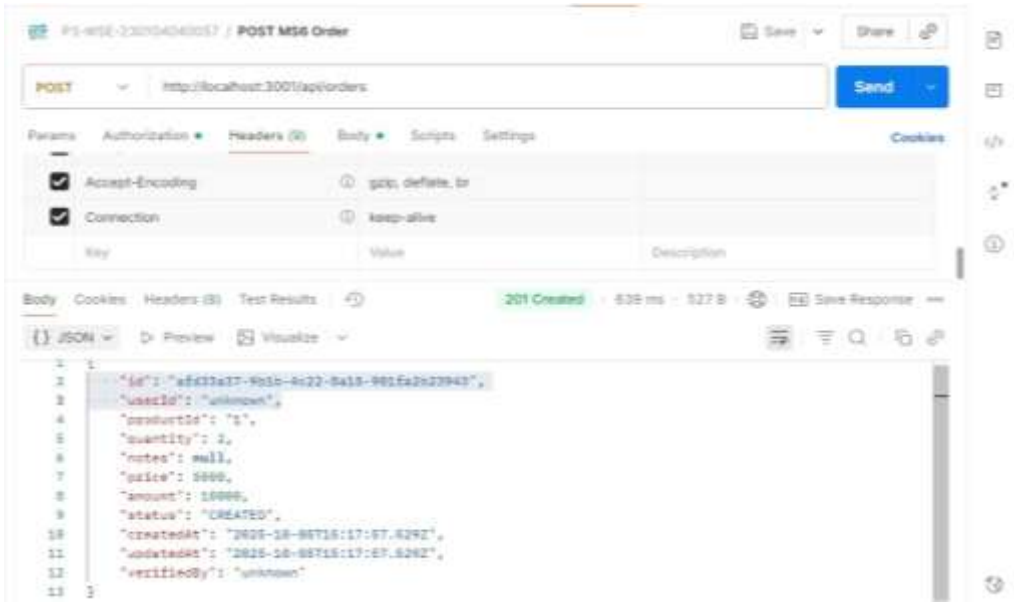
Screenshot MS4 - GET http://localhost:3001/api/products (dengan Token)



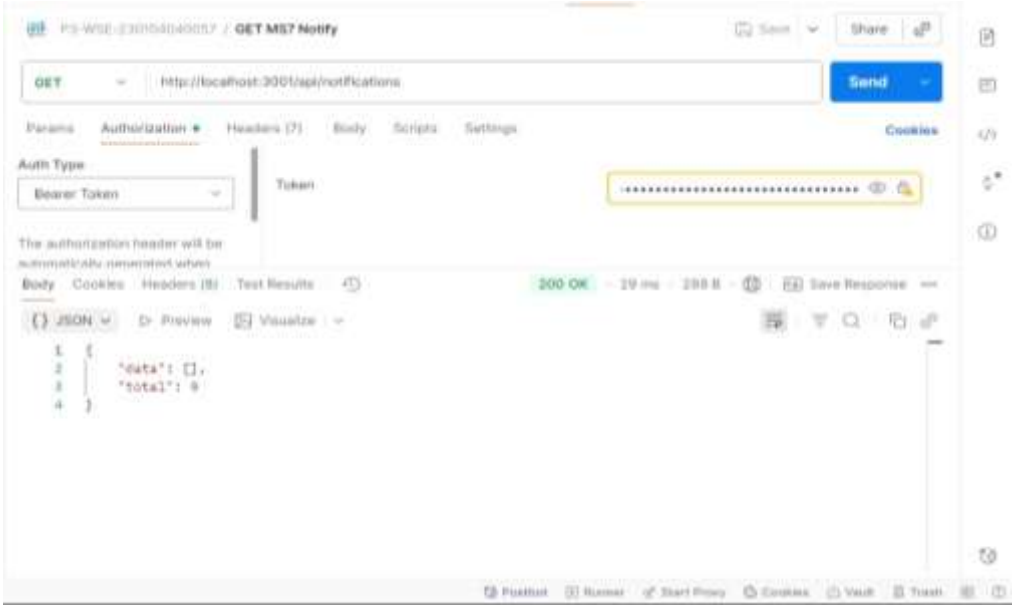
Screenshot MS5 - POST http://localhost:3001/api/products (dengan Token + Tambah Product)



Screenshot MS6 - POST http://localhost:3001/api/orders



Screenshot MS7 - GET http://localhost:3001/api/notifications



4. KESIMPULAN UMUM & REKOMENDASI

a) Client–Server: Arsitektur Client–Server menunjukkan konsep dasar komunikasi satu arah antara *client* dan *server* melalui HTTP request–response. Implementasi sederhana menggunakan Express dapat berjalan dengan mudah dan cocok untuk aplikasi kecil. Namun, model ini memiliki keterbatasan dalam skalabilitas dan integrasi, karena seluruh logika bisnis dan data dikelola oleh satu server tunggal.

b) SOA: SOA membagi aplikasi menjadi beberapa layanan yang saling terhubung melalui protokol standar (HTTP, JSON, JWT). Dengan memisahkan Authentication Service dan Data Service, sistem menjadi lebih modular dan mudah dikembangkan. Kelebihannya adalah keamanan dan fleksibilitas lebih baik, namun kompleksitas meningkat karena setiap service perlu dikelola dan diatur komunikasi antarservicenya.

c) Microservices: Arsitektur microservices lebih lanjut memecah sistem menjadi layanan-layanan kecil yang independen (Auth, Product, Order, Notification) dan

dihubungkan melalui API Gateway. Pendekatan ini memberikan skalabilitas tinggi, kemudahan deployment, dan pengelolaan fitur secara terpisah. Kekurangannya adalah membutuhkan konfigurasi lebih rumit (misalnya proxy, komunikasi antar service, dan sinkronisasi data).

d) Rekomendasi: Untuk sistem berskala kecil–menengah, SOA sudah cukup efektif digunakan. Namun, untuk sistem besar dengan banyak modul dan kebutuhan *scalability* tinggi (misalnya e-commerce atau platform digital), arsitektur microservices lebih direkomendasikan. Penggunaan API Gateway sangat membantu dalam manajemen keamanan dan routing antar service.