

Laporan Praktikum WSE #3

Mata Kuliah : Web Service Engineering
Dosen Pengampu : Muhayat, M.IT
Praktikum : P3 - **Web Service Architecture**
Nama Mahasiswa : Febi Novia Putri
NIM : 230104040055
Kelas : TI23B
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	Body/raw/json: { "nim": "2310511004", "nama": "Siti Alayda Salsabila", "prodi": "Teknologi Informasi", "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	PUT http://localhost:3001/mahasiswa/2310511002	Body/raw/json: { "prodi": "Sistem Informasi", "angkatan": 2024 }	200 OK	"Mahasiswa diupdate"

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

The screenshot shows the Postman interface with a collection named 'New Collection / POST CS Mahasiswa'. A POST request is selected with the URL `http://localhost:3001/mahasiswa`. The request body is set to raw JSON:

```

1 {
2   "nim": "230104040055",
3   "name": "Febi Novia Putri",
4   "prodi": "Teknologi Informasi",
5   "angkatan": 2023
6 }

```

The response status is `201 Created` with a response time of 105 ms and a size of 373 B. The response body is also a JSON object:

```

1 {
2   "message": "Mahasiswa dibuat",
3   "data": [
4     {
5       "nim": "230104040055",
6       "name": "Febi Novia Putri",
7       "prodi": "Teknologi Informasi",
8       "angkatan": 2023
9     }
10 ]

```

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

The screenshot shows the Postman interface with a collection named 'New Collection / GET CS Mahasiswa'. A GET request is selected with the URL `http://localhost:3001/mahasiswa`. The response status is `200 OK` with a response time of 11 ms and a size of 517 B. The response body is a JSON array containing two objects:

```

1 [
2   {
3     "nim": "2310511000",
4     "name": "Siti Aulia",
5     "prodi": "Teknik Informatika",
6     "angkatan": 2023
7   },
8   {
9     "nim": "230104040055",
10    "name": "Febi Novia Putri",
11    "prodi": "Teknologi Informasi",
12    "angkatan": 2023
13 },
14   {
15     "nim": "2310511001",
16     "name": "Siti Alayda Salsabila",
17     "prodi": "Teknologi Informasi",
18     "angkatan": 2023
19   }
20 ]

```

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

The screenshot shows the Postman interface with a collection named "New Collection". A DELETE request is made to <http://localhost:3001/mahasiswa/2310511001>. The request body is a JSON object:

```
1  {
2     "nim": "2310511001",
3     "nama": "Siti Alayda Salsabila",
4     "prodi": "Teknologi Informasi",
5     "angkatan": 2023
6 }
```

The response status is 200 OK, and the response body is:

```
1  {
2     "message": "Mahasiswa dihapus",
3     "nim": "2310511001"
4 }
```

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

The screenshot shows the Postman interface with a collection named "New Collection". A PUT request is made to <http://localhost:3001/mahasiswa/2310511002>. The request body is a JSON object:

```
1  {
2     "prodi": "Sistem Informasi",
3     "angkatan": 2024
4 }
```

The response status is 200 OK, and the response body is:

```
1  {
2     "message": "Mahasiswa diupdate",
3     "data": {
4         "nim": "230104040055",
5         "nama": "Febi Novia Putri",
6         "prodi": "Sistem Informasi",
7         "angkatan": 2024
8     }
9 }
```

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

The screenshot shows a Postman collection named "New Collection / POST SOA Login". A POST request is made to "http://localhost:4000/login". The request body is a JSON object with "username": "mhs1" and "password": "123456". The response status is 200 OK, and the response body contains a success message and a JWT token.

```
1 {  
2   "username": "mhs1",  
3   "password": "123456"  
4 }
```

```
1 {  
2   "message": "Login sukses",  
3   "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1c2VybmFtZSI6ImloczEILCjpyXQ10]E3NjAyNzU2MTcsImV4cCI6MTc2MDI3NzQxN30.BqIae7l4FBsMBCR39kgNnG1Fqc0RM8zIwEna2seguh4"  
4 }
```

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

New Collection / GET SOA Data No-Auth

GET http://localhost:5000/data

Params Authorization Headers (6) Body Scripts Settings

Query Params

| Key | Value | Description |
|-----|-------|-------------|
| Key | Value | Description |

Body Cookies Headers (8) Test Results

401 Unauthorized 11 ms 345 B Save Response

```
{
  "message": "Header Authorization harus dalam format Bearer <token>"
}
```

Runner Start Proxy Cookies Vault Trash

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

New Collection / GET SOA Data Auth

GET http://localhost:5000/data

Params Authorization Headers (7) Body Scripts Settings

Auth Type Bearer Token

Token

The authorization header will be automatically generated when you send the request. Learn more about [Bearer Token](#) authorization.

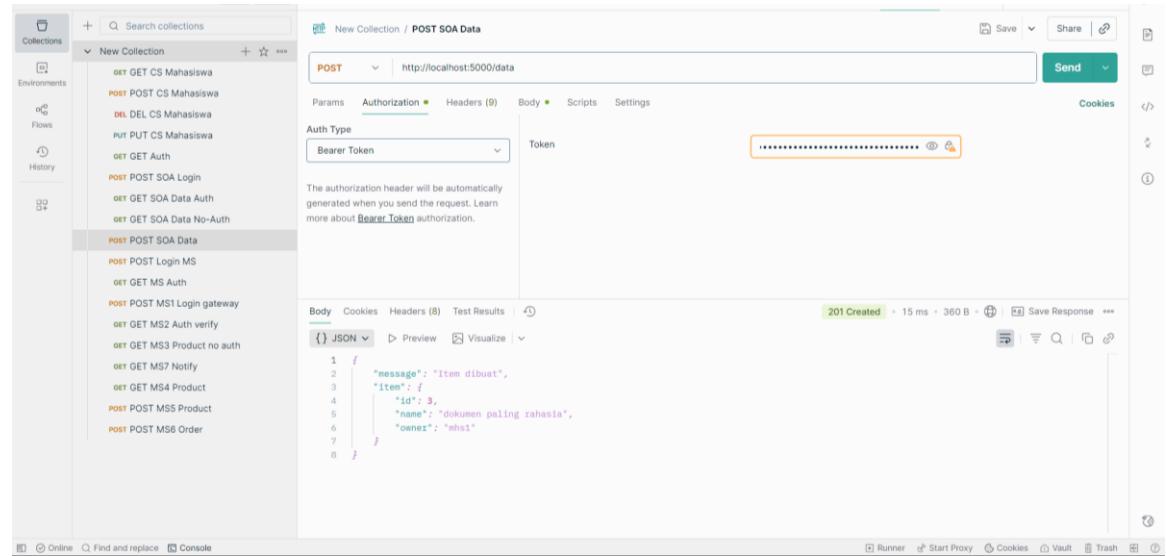
Body Cookies Headers (8) Test Results

200 OK 48 ms 384 B Save Response

```
{
  "message": "Halo mhs1, berikut data yang dapat kamu akses",
  "data": [
    {
      "id": 1,
      "name": "Data Rahasia A",
      "owner": "mhs1"
    }
  ]
}
```

Runner Start Proxy Cookies Vault Trash

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 Auth-Service) | 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 | GET
http://localhost:3001/api/products | (Tanpa header Authorization) | 401 → {
"message": "Authorization Bearer <token> wajib" } | Menguji gatekeeper gateway. |

| | | | | | |
|-----|--|--|--|--|--|
| | → harus ditolak | | | | |
| 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 Product-Service). | 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 dari gateway diteruskan) | GET
http://localhost:3000/api/notifications | Headers:
Authorization: Bearer <JWT> | 200 OK
{ "data": [], "total": 0 } | Menguji forwarding header user dari Gateway ke Notification Service (Axios + onProxyReq) |

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

The screenshot shows the Postman interface with a collection named "P3_WSE_230104040055". A new collection named "New Collection" is expanded, showing various API endpoints. The "POST MS1 Login gateway" endpoint is selected. The request method is set to POST, and the URL is http://localhost:4001/login. The "Body" tab is selected, showing an empty JSON object. The "Headers" tab shows the following headers:

```

    Headers (8)
    Authorization: Bearer <JWT>
    Content-Type: application/json
  
```

The response tab shows a 200 OK status with the following JSON content:

```

    {
      "message": "Login success",
      "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.",
      "expiresIn": "1h"
    }
  
```

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

The screenshot shows the Postman interface with a collection named "P3_WSE_230104040055". A new collection named "New Collection" is being created. The "Auth Type" dropdown is set to "Bearer Token" and contains a placeholder token. The response body is a JSON object:

```

1  {
2    "valid": true,
3    "user": {
4      "sub": 1,
5      "username": "nhs1",
6      "role": "student",
7      "iat": 1768276458,
8      "exp": 1768280658
9    }
10   }

```

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

The screenshot shows the Postman interface with the same collection. A new collection named "New Collection" is being created. The "Authorization" tab is selected, showing an empty "Key" field. The response status is "401 Unauthorized" and the message is "Authorization Bearer <token> wasjib".

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

New Collection / GET MS4 Product

GET http://localhost:5001/products

Auth Type: Bearer Token

```

1 [
2   {
3     "id": 1,
4     "name": "Pulpem",
5     "price": 5000
6   },
7   {
8     "id": 2,
9     "name": "Buku Tulis",
10    "price": 12000
11  }
12 ]

```

200 OK | 51 ms | 349 B | Save Response

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

New Collection / POST MS5 Product

POST http://localhost:3001/api/products

Body: raw

```

1 {
2   "id": 3,
3   "name": "Pensil",
4   "price": 5000
5 }

```

201 Created | 41 ms | 348 B | Save Response

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

POST MS6 Order

```

1 {
2   "id": "62259841-b6d0-40c4-821a-71b659887476",
3   "userId": "unknown",
4   "productId": "1",
5   "quantity": 1,
6   "notes": null,
7   "price": 5000,
8   "amount": 10000,
9   "status": "CREATED",
10  "createdAt": "2025-10-12T13:42:58.478Z",
11  "updatedAt": "2025-10-12T13:42:58.478Z",
12  "verifiedBy": "unknown"
13 }

```

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

GET MS7 Notify

```

1 {
2   "data": [],
3   "total": 0
4 }

```

4. KESIMPULAN UMUM & REKOMENDASI

- Client-Server: [ringkas temuan utama]
- SOA: [ringkas temuan utama]
- Microservices: [ringkas temuan utama]
- Rekomendasi: [Jika Ada]