

# Laporan Praktikum WSE #10

Mata Kuliah : Web Service Engineering  
Dosen Pengampu : Muhayat, M.IT  
Praktikum : P10 – Simulasi API Key & OAuth 2.0  
Nama Mahasiswa : Husna Norgina  
NIM : 230104040056  
Kelas : TI23B  
Link Repo Github : <https://github.com/husna-norgina/p10-oauth2-api-key-230104040056>  
Tanggal Praktikum : 08-12-2025

## A. Tujuan Praktikum

1. Memahami konsep dan perbedaan antara API Key dan OAuth 2.0 (dengan fokus pada Authorization Code Grant atau Implicit Grant yang disederhanakan).
2. Mengimplementasikan Middleware dalam Express.js untuk validasi otentifikasi.
3. Mengelola dan memvalidasi API Keys sederhana di sisi server.
4. Mensimulasikan proses pemberian token (Token Grant) dan akses sumber daya terproteksi (Protected Resource) menggunakan JWT (JSON Web Tokens).
5. Menggunakan MongoDB Atlas untuk menyimpan data pengguna, client aplikasi, dan API Key

## B. Aktivitas Praktikum Tools

Kategori	Teknologi/Konsep	Keterangan
Backend	Node.js & Express.js	Server API utama.
Database	MongoDB Atlas	Menyimpan data pengguna, API Key, dan klien.
Autentikasi	JSON Web Tokens (JWT)	Untuk simulasi Access Token pada OAuth 2.0.
Alat	Postman / Insomnia	Untuk menguji endpoint API.
Prasyarat	Pemahaman dasar Node.js, Express.js, dan operasi CRUD MongoDB.	

## C. Struktur Project

```
p10-oauth2-api-key-230104040056/
├── controllers/
│   ├── authController.js      # [Langkah 4] Logika untuk Login dan token generation (JWT)
│   └── productController.js  # [Langkah 3 & 6] Handler GET Public dan CRUD Private
├── middleware/
│   ├── validateApiKey.js    # [Langkah 3] Middleware validasi API Key
│   └── validateToken.js     # [Langkah 5] Middleware verifikasi JWT (Bearer Token)
├── models/
│   ├── ApiKey.js            # [Langkah 2] Model penyimpanan API Keys
│   ├── Product.js           # [Langkah 2] Model data produk
│   └── User.js               # [Langkah 2] Model pengguna + bcrypt pre-save hook
├── routes/
│   ├── authRoutes.js        # [Langkah 4] Route POST /auth/token
│   └── productRoutes.js    # [Langkah 3 & 6] Route GET Public + CRUD Private
├── seeders/
│   └── seed.js                # [Langkah 2] Seeder awal (User, API Key, Product)
├── utils/
│   └── generateToken.js     # [Langkah 4] Util untuk membuat JWT
├── .env                      # [Langkah 1] MONGODB_URI, PORT, JWT_SECRET
├── package.json              # Metadata project + dependencies
└── server.js                 # [Langkah 1] Entry point server + koneksi DB
                            # Dokumentasi (opsional)
```

## D. Hasil & Bukti

Screenshot Hasil Uji Endpoint di Postman

- ✓ GET http://localhost:3000 → Menampilkan status server API (200 OK)

The screenshot shows the Postman interface with a collection named 'p10-oauth2-api-key-230104040056'. A specific test case, 'GET Server API', is selected. The request URL is set to 'http://localhost:3000'. The response status is '200 OK' with a response time of 7 ms and a body size of 317 B. The response body is a JSON object containing a message and a praktikum value.

1	{
2	"message": "API Server Berjalan!",
3	"praktikum": "P10: Simulasi API Key & OAuth 2.0"
4	}

- ✗ GET http://localhost:3000/api/v1/products/public → Key hilang (401 Unauthorized)

The screenshot shows the Postman interface with the same collection and test case. The request URL is now 'http://localhost:3000/api/v1/products/public'. The response status is '401 Unauthorized' with a response time of 5 ms and a body size of 322 B. The response body is a JSON object with a single message indicating that the API key was not found in the header.

1	{
2	"message": "Akses Ditolak: API Key tidak ditemukan di header \'x-api-key\'."
3	}

✗ GET http://localhost:3000/api/v1/products/public → Key palsu (401 Unauthorized)

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Collections', 'Environments', 'History', and 'Flows'. The main area displays a collection named 'p10-oauth2-api-key-230104040056' with various API endpoints listed under it. One endpoint, 'GET Key Palsu', is highlighted with a red border. The 'Headers' tab is selected, showing the following headers:

Key	Value	Description
User-Agent	PostmanRuntime/7.49.1	
Accept	/*	
Accept-Encoding	gzip, deflate, br	
Connection	keep-alive	
x-api-key	KEY_PALSU	

The 'Body' tab shows a JSON response with the message: "message": "Akses Ditolak: API Key tidak valid atau sudah dicabut." (Access Denied: API Key is invalid or has been revoked.). The status bar at the bottom indicates a 401 Unauthorized error.

✓ GET http://localhost:3000/api/v1/products/public → Key valid (200 OK)

This screenshot shows the same Postman interface as the previous one, but with a different API endpoint selected: 'GET Key Valid'. The 'Headers' tab is selected, showing the 'x-api-key' header set to 'PRACTICUM\_API\_KEY\_A\_1234567890'. The 'Body' tab displays a successful JSON response with the message: "message": "Daftar Produk berhasil diambil. Akses: API Key (Public App Client A)", and a data array containing two product objects. The first product is a laptop with ID 693635b14ff130ecbd4fc287, name 'Laptop Gaming Pro', price 15000000, stock 10, and descriptions. The second product is a monitor with ID 693635b14ff130ecbd4fc288, name 'Monitor 4K Ultra', and descriptions.

_id	name	price	stock	description	createdAt	updatedAt
693635b14ff130ecbd4fc287	Laptop Gaming Pro	15000000	10	Laptop performa tinggi.	2025-12-08T02:19:29.580Z	2025-12-08T02:19:29.580Z
693635b14ff130ecbd4fc288	Monitor 4K Ultra			Monitor dengan resolusi tinggi.	2025-12-08T02:19:29.580Z	2025-12-08T02:19:29.580Z

✗ POST http://localhost:3000/api/v1/auth/token → Gagal login admin (401 Unauthorized)

The screenshot shows the Postman interface with a collection named 'p10-oauth2-api-key-230104040056'. A POST request is made to 'http://localhost:3000/api/v1/auth/token' with the following body:

```
1 {
2   "username": "admin",
3   "password": "salah"
4 }
```

The response status is 401 Unauthorized, and the response body is:

```
1 {
2   "message": "otentikasi gagal: Username atau Password tidak valid."
3 }
```

✓ POST http://localhost:3000/api/v1/auth/token → Sukses login admin (200 OK)

The screenshot shows the Postman interface with the same collection. A POST request is made to 'http://localhost:3000/api/v1/auth/token' with the following body:

```
1 {
2   "username": "admin",
3   "password": "password123"
4 }
```

The response status is 200 OK, and the response body is a JSON token object:

```
1 {
2   "token_type": "Bearer",
3   "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZC16IjY5MzYzMWNlxGZMTMwZWN1ZDRmYzI4ZStsnJvbGU01JhZG1pbisImIhdCI6MTc2NTMzMDY2NCwiZXhwIjoxNzY1OTM1NDY0fQ.B7eMsxovv4BWPkRjWxYiMP_Flpz24uLlzR6S-p78jB",
4   "expires_in": "720",
5   "user": {
6     "id": "693635b14ff130ecbd4fc28e",
7     "username": "admin",
8     "role": "admin"
9   }
10 }
```

✗ POST http://localhost:3000/api/v1/auth/token → Gagal login user biasa (401 Unauthorized)

The screenshot shows the Postman interface with a collection named 'p10-oauth2-api-key-230104040056'. A POST request is made to `http://localhost:3000/api/v1/auth/token`. The request body is:

```
1 {
2   "username": "userbiasa",
3   "password": "salah"
4 }
```

The response status is 401 Unauthorized, with the message: "0tentikasi Gagal: Username atau Password tidak valid."

✓ POST http://localhost:3000/api/v1/auth/token → Sukses login user biasa (200 OK)

The screenshot shows the Postman interface with the same collection. A POST request is made to `http://localhost:3000/api/v1/auth/token`. The request body is:

```
1 {
2   "username": "userbiasa",
3   "password": "userpass"
4 }
```

The response status is 200 OK, with a detailed JSON response:

```
1 {
2   "token_type": "Bearer",
3   "access_token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJpZC16IjY5MzYzMjNlXNgZmMTeWn1ZDRmYzI5MCIsInJvbGUiOjI3ic2VyiwiwF0IjoxNzY1MzMmNzAyLCJleHAiOjE3NjU5MzU1MD39.oA_aYgBbR4LsVi6bqp2VaFnW7Jkh14zJzFvaijqZI",
4   "expires_in": "7d",
5   "user": [
6     {
7       "id": "693635b14ff130ecbd4fc290",
8       "username": "userbiasa",
9       "role": "user"
10    }
11 }
```

✗ POST http://localhost:3000/api/v1/products/private → Token hilang (403 Forbidden)

The screenshot shows the Postman application interface. On the left, there's a sidebar with 'Collections', 'Environments', 'History', 'Flows', and 'Files (BETA)'. The main area displays a collection named 'p10-oauth2-api-key-230104040056' with various API endpoints listed. One endpoint, 'POST Token Hilang', is selected. The request details show a POST method to 'http://localhost:3000/api/v1/products/private'. The 'Authorization' tab is selected, showing 'none' as the auth type. The 'Body' tab contains an empty JSON object. The response status is '403 Forbidden' with the message: "message": "Akses Ditolak: Tidak ada Token Bearez yang ditemukan.".

✗ POST http://localhost:3000/api/v1/products/private → Token palsu (403 Forbidden)

This screenshot is similar to the previous one but shows a different error. The 'Authorization' tab now has 'Bearer' selected as the auth type, and the 'Token' field contains a series of asterisks ('\*\*\*\*\*'). The response status is '403 Forbidden' with the message: "message": "Akses Ditolak: Token tidak valid atau kedaluwarsa.".

✓ POST http://localhost:3000/api/v1/products/private → Token valid (201 Created)

The screenshot shows the Postman interface with the following details:

- Collection:** p10-oauth2-api-key-230104040056
- Request:** POST http://localhost:3000/api/v1/products/private
- Authorization:** Bearer [REDACTED]
- Body:** JSON (empty)
- Response Status:** 201 Created
- Response Body:**

```
1 {
2   "message": "Produk berhasil dibuat oleh admin (ID: 693635b14ff130ecbd4fc28e)",
3   "data": {
4     "name": "Buku Algoritma",
5     "price": 100000,
6     "stock": 0,
7     "_id": "6938cfa1237d32a7dd996c82",
8     "createdAt": "2025-12-10T01:40:49.686Z",
9     "updatedAt": "2025-12-10T01:40:49.686Z",
10    "__v": 0
11  }
12 }
```

✗ POST http://localhost:3000/api/v1/products/private → Gagal create, user biasa (403 Forbidden)

The screenshot shows the Postman interface with the following details:

- Collection:** p10-oauth2-api-key-230104040056
- Request:** POST http://localhost:3000/api/v1/products/private
- Authorization:** Bearer [REDACTED]
- Body:** JSON (empty)
- Response Status:** 403 Forbidden
- Response Body:**

```
1 {
2   "message": "Akses ditolak! Anda (user) tidak memiliki izin (Admin diperlukan) untuk membuat produk."
3 }
```

✓ POST http://localhost:3000/api/v1/products/private → Sukses create, admin (201 Created)

The screenshot shows the Postman interface with a collection named 'p10-oauth2-api-key-230104040056'. A POST request titled 'Create Admin' is selected. The URL is set to `http://localhost:3000/api/v1/products/private`. The 'Body' tab shows a JSON payload:

```
1 {
2   "name": "Smartphone X Pro",
3   "price": 7500000,
4   "stock": 30,
5   "description": "Smartphone dengan performa tinggi dan kamera canggih."
6 }
```

The response status is '201 Created' with a response time of 65 ms and a body size of 571 B. The response body is:

```
1 {
2   "message": "Produk berhasil dibuat oleh admin (ID: 6938d069237d32a7dd996c88)",
3   "data": {
4     "name": "Smartphone X Pro",
5     "price": 7500000,
6     "stock": 30,
7     "description": "Smartphone dengan performa tinggi dan kamera canggih.",
8     "_id": "6938d069237d32a7dd996c88",
9     "createdAt": "2025-12-10T01:44:09.651Z",
10    "updatedAt": "2025-12-10T01:44:09.651Z",
11  }
12 }
```

✗ PUT http://localhost:3000/api/v1/products/private/6938d069237d32a7dd996c88 → Gagal update, user biasa (403 Forbidden)

The screenshot shows the Postman interface with the same collection. A PUT request titled 'Update User Biasa' is selected. The URL is set to `http://localhost:3000/api/v1/products/private/6938d069237d32a7dd996c88`. The 'Body' tab shows a JSON payload:

```
1 {
2   "price": 7000000,
3   "stock": 17
4 }
```

The response status is '403 Forbidden' with a response time of 7 ms and a body size of 345 B. The response body is:

```
1 {
2   "message": "Akses ditolak! Anda (user) tidak memiliki izin (Admin diperlukan) untuk mengedit produk."
3 }
```

- ✓ PUT http://localhost:3000/api/v1/products/private/6938d069237d32a7dd996c88 →  
Sukses update, admin (200 OK)

The screenshot shows the Postman interface with the following details:

- Collection:** Husna Norgina's Workspace
- Request:** PUT /products/private/6938d069237d32a7dd996c88
- Method:** PUT
- Body:** Raw JSON (selected)
 

```
{
        "price": 8000000,
        "stock": 25
      }
```
- Response Status:** 200 OK
- Response Body:**

```
{
  "message": "Produk berhasil diperbarui.",
  "data": {
    "_id": "6938d069237d32a7dd996c88",
    "name": "Smartphone X Pro",
    "price": 8000000,
    "stock": 25,
    "description": "Smartphone dengan performa tinggi dan kamera canggih.",
    "createdAt": "2025-12-10T01:44:09.651Z",
    "updatedAt": "2025-12-10T01:46:21.801Z"
  }
}
```

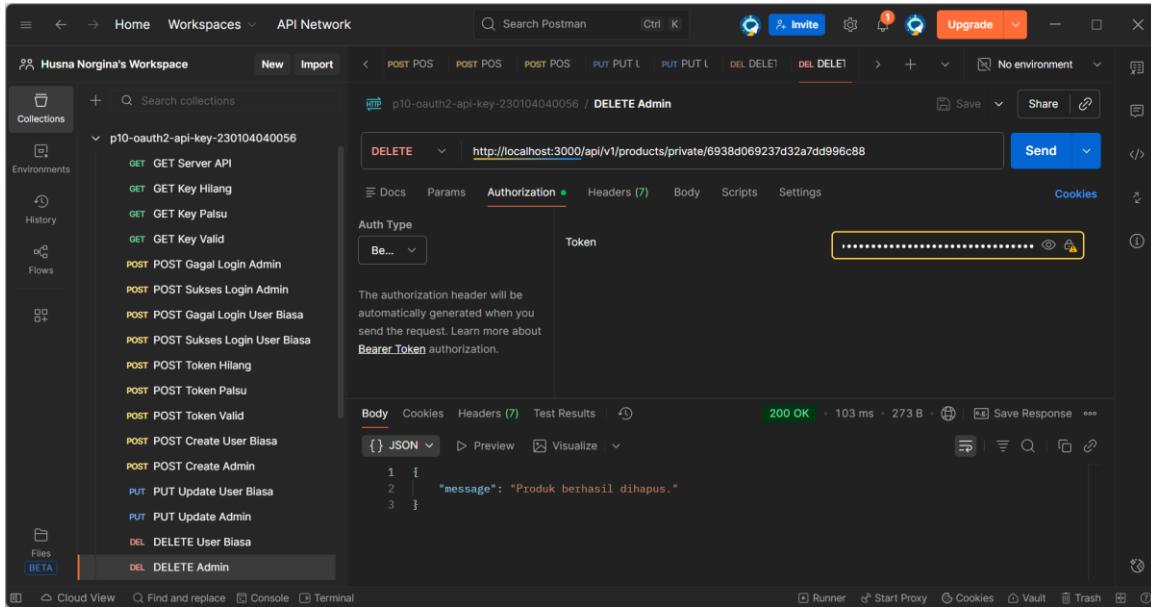
- ✗ DEL http://localhost:3000/api/v1/products/private/6938d069237d32a7dd996c88 →  
Gagal delete, user biasa (403 Forbidden)

The screenshot shows the Postman interface with the following details:

- Collection:** Husna Norgina's Workspace
- Request:** DELETE /products/private/6938d069237d32a7dd996c88
- Method:** DELETE
- Authorization:** Token (selected)
- Response Status:** 403 Forbidden
- Response Body:**

```
{
  "message": "Akses ditolak! Anda (user) tidak memiliki izin (Admin diperlukan) untuk menghapus produk."
}
```

- DEL http://localhost:3000/api/v1/products/private/6938d069237d32a7dd996c88 →  
Sukses delete, admin (200 OK)



## E. Analisis

Praktikum #10 berhasil menerapkan autentikasi dan otorisasi dengan API Key dan JWT secara benar, di mana middleware dapat memvalidasi API Key, memverifikasi token, dan membedakan hak akses antara user biasa dan admin. Pengujian Postman menunjukkan seluruh skenario berjalan sesuai harapan, request tanpa key atau token ditolak, token palsu terdeteksi invalid, dan admin dapat melakukan CRUD produk. Secara keseluruhan, sistem sudah aman, terstruktur, dan memenuhi tujuan praktikum, serta memberikan pemahaman yang lebih kuat tentang keamanan backend modern.

## F. Kesimpulan

Praktikum 10 berhasil membangun sistem autentikasi dan otorisasi yang aman menggunakan API Key dan JWT dengan arsitektur backend Node.js yang terstruktur melalui pemisahan controllers, middleware, models, routes, dan utils. Validasi API Key pada endpoint publik serta verifikasi JWT pada endpoint private berjalan efektif, didukung kontrol akses berbasis role yang membedakan hak admin dan user biasa. Seluruh skenario pengujian Postman seperti saat key atau token hilang, key atau token palsu, dan token yang valid menunjukkan hasil yang sesuai standar keamanan. Error handling memberikan respons yang konsisten dan mudah dipahami, sedangkan seeder membantu mempermudah proses penyiapan data awal. Secara keseluruhan, sistem yang dibangun stabil, aman, mengikuti best practices, dan memenuhi tujuan praktikum.

## G. Checklist Praktikum

- Validasi API Key berjalan pada endpoint publik
- Proses login menghasilkan JWT yang valid
- Middleware verifikasi token berfungsi untuk semua endpoint private
- Role admin dan user biasa terdeteksi dengan benar
- Akses CRUD hanya terbuka untuk admin sesuai aturan otorisasi
- Error handler mengembalikan respons JSON yang jelas dan konsisten
- Seed data (User, API Key, Product) berhasil dijalankan
- Semua skenario pengujian Postman berhasil sesuai ekspektasi
- Evidence Postman lengkap
- Dokumentasi README.md selesai
- Dokumentasi project tersusun rapi dalam struktur folder aplikasi