

Laporan Praktikum WSE #8

Mata Kuliah : Web Service Engineering
Dosen Pengampu : Muhayat, M.IT
Praktikum : P8 – **Secure & Observable RESTful CRUD API (JWT + Hardening + Observability)**
Nama Mahasiswa : Husna Norgina
NIM : 230104040056
Kelas : TI23B
Link Repo Github : <https://github.com/husna-norgina/P8-SecureArticles-230104040056>
Tanggal Praktikum : 24-11-2025

A. Tujuan Praktikum

1. Merancang API CRUD sesuai 7 RESTful Principles secara konsisten (resource, method, status code, HATEOAS ringan, stateless, caching, layered system).
2. Mengimplementasikan JWT Authentication (register/login/refresh/logout) dengan keamanan standar industri.
3. Menerapkan role-based authorization (admin vs user).
4. Melakukan hardening API: validation, rate limit, security headers, CORS, error hygiene, env secrets, dsb.
5. Membangun observability: structured logging + correlation-id + health/metrics endpoint.
6. Menyusun dokumentasi OpenAPI yang akurat, test-ready, dan bisa dipakai integrasi lintas layanan.

B. Aktivitas Praktikum (Workflow Advance)

Gunakan pola kerja modern:

1. Design-First (OpenAPI 3.1 + Spectral Linting)
2. Mock-First (Prism/Postman Mock Server)
3. Test-First (RED with Jest/Supertest + auth tests)
4. Implementasi (GREEN)
5. Hardening (Security + Reliability)
6. Observability (Logs + Metrics + Healthcheck)
7. CI (GitHub Actions: lint → test → build)
8. Demo & Dokumentasi Final

C. Tools yang Dipakai

Wajib:

- Node.js 18+ / 20 LTS
- Express.js
- MongoDB / PostgreSQL (pilih salah satu)
- JWT library (mis. jsonwebtoken)
- bcrypt
- Joi/Zod/Express-validator (validation)
- Jest + Supertest
- OpenAPI + Spectral
- Postman/Insomnia

- GitHub Actions

Observability/Hardening (advance layer):

- pino / winston (structured logger)
- express-rate-limit
- helmet
- cors
- morgan diganti structured log
- uuid / nanoid (correlation-id)
- swagger-ui-express (docs)
- optional: prom-client (metrics)

D. Struktur Project yang Direkomendasikan (Advance)

```

src/
  app.js
  server.js

  config/
    env.js
    db.js

  routes/
    auth.routes.js
    articles.routes.js
    system.routes.js

  controllers/
    auth.controller.js
    articles.controller.js
    system.controller.js

  services/
    auth.service.js
    articles.service.js

  repositories/
    articles.repo.js
    users.repo.js

  middlewares/
    auth.middleware.js
    role.middleware.js
    validate.middleware.js
    correlationId.middleware.js
    rateLimit.middleware.js
    error.middleware.js
    notFound.middleware.js

  utils/
    jwt.js
    logger.js
    response.js

  docs/
    openapi.yaml

tests/
  auth.test.js
  articles.test.js

```

E. Tabel Endpoint Project Praktikum #8

AUTH ENDPOINTS

Method	Endpoint	Auth	Deskripsi
POST	/api/auth/register	✗ Public	Register user baru (role: user/admin)
POST	/api/auth/login	✗ Public	Login → dpt accessToken+refreshToken
POST	/api/auth/refresh	✗ Public	Minta accessToken baru via refreshToken
POST	/api/auth/logout	✓ Access Token	Logout & invalidate refreshToken
GET	/api/auth/me	✓ Access Token	Ambil profil user dari JWT

ARTICLES ENDPOINTS (CRUD + RBAC)

Method	Endpoint	Auth	Role	Deskripsi
GET	/api/articles	✗ Public	public	List all articles + pagination + search
POST	/api/articles	✓ Access Token	user/admin	Create article (author otomatis dari JWT)
PUT	/api/articles/:id	✓ Access Token	owner/admin	Update article
DELETE	/api/articles/:id	✓ Access Token	admin	Hapus article

SYSTEM / OBSERVABILITY

Method	Endpoint	Auth	Deskripsi
GET	/health	✗ Public	Cek status server
GET	/docs	✗ Public	Swagger UI (OpenAPI Documentation)

F. Hasil & Bukti

Screenshot Hasil Uji Endpoint di Postman

AUTH ENDPOINTS

- ✓ POST http://localhost:3000/api/auth/register → Register user baru (201 Created)

The screenshot shows the Postman interface with the following details:

- Collection:** Husna Norgina's Workspace
- Request:** POST /api/auth/register
- Body (raw JSON):**

```
1 {
2   "name": "Husna Norgina",
3   "email": "hsn.nrgina@gmail.com",
4   "password": "123456"
5 }
```

- Response Status:** 201 Created
- Response Body (JSON):**

```
1 {
2   "success": true,
3   "message": "Registered",
4   "data": {
5     "id": "692ab46fb6157d58233c79cb",
6     "name": "Husna Norgina",
7     "email": "hsn.nrgina@gmail.com",
8     "role": "user"
9   }
10 }
```

- ✓ POST http://localhost:3000/api/auth/login → Login user (200 OK)

The screenshot shows the Postman interface with the following details:

- Collection:** Husna Norgina's Workspace
- Request:** POST /api/auth/login
- Body (raw JSON):**

```
1 {
2   "email": "hsn.nrgina@gmail.com",
3   "password": "123456"
4 }
```

- Response Status:** 200 OK
- Response Body (JSON):**

```
1 {
2   "success": true,
3   "message": "Logged in",
4   "data": {
5     "accessToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0ZXciOiYjYmMfInDZmYjYxNtdkNTgyMzNjNzIjYiIsInJvbGU0iJ1c2VyiwiZW1haWwiOjJoc24ubnJnaW5hQGdtYWlsLnNvbSIiMlhcdI6MTc2NDQwNjQwMCw1ZXhwIjoxNzY0NDA3MzAwfQ.85JTRNLZ5szt26-08x6ep4P2EfnLP0HRVKhKzNaUk",
6     "refreshToken": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ0ZXciOiYjYmMfInDZmYjYxNtdkNTgyMzNjNzIjYiIsInJvbGU0iJ1c2VyiwiZW1haWwiOjJoc24ubnJnaW5hQGdtYWlsLnNvbSIiMlhcdI6MTc2NDQwNjQwMCw1ZXhwIjoxNzY1MDExMjAwfQ.3oLelXnbFwZ2tgChFj_q01ix42YXp-L9dnQu18ZdpS",
7   }
8 }
```

✓ POST http://localhost:3000/api/auth/refresh → Refresh access token (200 OK)

The screenshot shows the Postman interface with the following details:

- Collection:** Husna Norgina's Workspace
- Request:** POST /api/auth/refresh
- Body:** raw JSON (copied from the previous screenshot)
- Response Status:** 200 OK
- Response Body:** A JSON object indicating success, message, and data (refreshed token).

✓ POST http://localhost:3000/api/auth/logout → Logout & invalidate token (200 OK)

The screenshot shows the Postman interface with the following details:

- Collection:** Husna Norgina's Workspace
- Request:** POST /api/auth/logout
- Authorization:** Bearer [REDACTED]
- Response Status:** 200 OK
- Response Body:** A JSON object indicating success, message, and data (null).

✓ GET http://localhost:3000/api/auth/me → Get profile user (200 OK)

The screenshot shows the Postman interface with a collection named "P8-SecureArticles-230104040056". A specific endpoint, "GET auth-me", is selected. The request URL is set to "http://localhost:3000/api/auth/me". The "Authorization" tab is active, showing a "Token" field with a redacted value. The response status is 200 OK, and the response body is a JSON object:

```
1 {  
2   "success": true,  
3   "message": "OK",  
4   "data": {  
5     "_id": "692a040fb6157d58233c79cb",  
6     "name": "Husna Norgina",  
7     "email": "hsn.nrgina@gmail.com",  
8     "role": "user",  
9     "createdAt": "2025-11-29T08:53:03.823Z",  
10    "updatedAt": "2025-11-29T08:54:09.149Z",  
11    "_v": 0  
12  }  
13 }
```

ARTICLES ENDPOINTS (CRUD + RBAC)

✓ GET http://localhost:3000/api/articles → List artikel (200 OK)

The screenshot shows the Postman interface with the same collection. A specific endpoint, "GET articles", is selected. The request URL is set to "http://localhost:3000/api/articles". The "Params" tab is active, showing a table for "Query Params" with one row: "Key" and "Value". The response status is 200 OK, and the response body is a JSON object:

```
1 {  
2   "success": true,  
3   "message": "OK",  
4   "data": {  
5     "page": 1,  
6     "limit": 10,  
7     "total": 0,  
8     "results": []  
9   }  
10 }
```

✓ POST http://localhost:3000/api/articles → Created article (201 Created)

The screenshot shows the Postman interface with the following details:

- Collection:** Husna Norgina's Workspace
- Request:** POST articles
- URL:** http://localhost:3000/api/articles
- Body (raw JSON):**

```
1 {
2   "title": "Artikel Pertama",
3   "content": "Belajar MongoDB Atlas untuk Praktikum 8",
4   "tags": ["wse", "articles"],
5   "status": "published"
6 }
```
- Response Status:** 201 Created
- Response Body (JSON):**

```
1 {
2   "success": true,
3   "message": "Created",
4   "data": {
5     "id": "692ab4d9b6157d58233c79d6",
6     "title": "Artikel Pertama",
7     "slug": "artikel-pertama",
8     "content": "Belajar MongoDB Atlas untuk Praktikum 8",
9     "tags": [
10       "wse",
11     ]
12   }
13 }
```

✓ POST http://localhost:3000/api/auth/register → Register admin baru (201 Created)

The screenshot shows the Postman interface with the following details:

- Collection:** Husna Norgina's Workspace
- Request:** POST register-admin
- URL:** http://localhost:3000/api/auth/register
- Body (raw JSON):**

```
1 {
2   "name": "Admin1",
3   "email": "admin@gmail.com",
4   "password": "123456",
5   "role": "admin"
6 }
```
- Response Status:** 201 Created
- Response Body (JSON):**

```
1 {
2   "success": true,
3   "message": "Registered",
4   "data": {
5     "id": "692ab541b6157d58233c79df",
6     "name": "Admin1",
7     "email": "admin@gmail.com",
8     "role": "admin"
9   }
10 }
```

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

The screenshot shows the Postman interface with the following details:

- Collection:** Husna Norgina's Workspace
- Request:** POST /api/auth/login
- Body (raw JSON):**

```
1 {
2   "email": "admin@gmail.com",
3   "password": "123456"
4 }
```
- Response Status:** 200 OK
- Response Body (JSON):**

```
1 {
2   "success": true,
3   "message": "Logged in",
4   "data": {
5     "accessToken": "eyJhbGciOiJIUzI1NiJzInR5cT6IkpxVCJ9.
6       eyJpZC16IjYSMmF1NTQyJyNtdkNTgyMzNjNzKzIisInJvbGUjO1JhZG1pbIIsImVtYWIisIjoiYWtaW5AZ21haWW
7       uY29tIiwiWF0ijoxNzY0ND2NjAyLCJ1eHA10jE3NjQ0MDc1M039.
8       Fk5K5b3RGr4P1NbUXMx-nuxvZavkb0PM16M1JX2M",
9     "refreshToken": "eyJhbGciOiJIUzI1NiIinRSccI6IkpxVCJ9.
10      eyJpZC16IjYSMmF1NTQyJyNtdkNTgyMzNjNzKzIisInJvbGUjO1JhZG1pbIIsImVtYWIisIjoiYWtaW5AZ21haWW
11      uY29tIiwiWF0ijoxNzY0ND2NjAyLCJ1eHA10jE3NjUwMTE0MDj.
12      Sce2sM7KYrbpF0aFby0ejAW0NRTq4FmXkmfIh3H_MGU",
13   }
14 }
```

✓ PUT http://localhost:3000/api/articles → Update article (200 OK)

The screenshot shows the Postman interface with the following details:

- Collection:** Husna Norgina's Workspace
- Request:** PUT /articles/:id
- URL Parameters:** /articles/692ab4d9b6157d58233c79d6
- Body (raw JSON):**

```
1 {
2   "title": "Artikel Pertama (Revisi)",
3   "content": "Konten diperbarui oleh pemilik."
4 }
```
- Response Status:** 200 OK
- Response Body (JSON):**

```
1 {
2   "success": true,
3   "message": "updated",
4   "data": {
5     "id": "692ab4d9b6157d58233c79d6",
6     "title": "Artikel Pertama (Revisi)",
7     "slug": "artikel-pertama-revisi",
8     "content": "Konten diperbarui oleh pemilik.",
9     "tags": [
10       "wse",
11       "articles"
12     ],
13   }
14 }
```

✓ DELETE http://localhost:3000/api/articles → Hapus article (204 No Content)

The screenshot shows the Postman interface with the following details:

- Collection:** Husna Norgina's Workspace
- Request:** DELETE articles-id
- URL:** http://localhost:3000/api/articles/692ab4d9b6157d58233c79d6
- Authorization:** Bearer [REDACTED]
- Response Status:** 204 No Content
- Body:** 1

⚙️ SYSTEM / OBSERVABILITY

✓ GET http://localhost:3000/health → Cek status server (200 OK)

The screenshot shows the Postman interface with the following details:

- Collection:** Husna Norgina's Workspace
- Request:** GET health
- URL:** http://localhost:3000/health
- Headers:** Authorization, Headers (24)
- Response Status:** 200 OK
- Body:** JSON response showing success: true, message: "Service healthy", data: {status: "UP", uptime: 3176.7832079, timestamp: "2025-11-29T08:58:24.348Z"}

GET http://localhost:3000/docs → Swagger UI (200 OK)

The screenshot shows the Postman interface with a collection named "Husna Norgina's Workspace". A specific request for "GET docs" at "http://localhost:3000/docs" is selected. The response details show a 200 OK status with a response time of 11 ms and a size of 4.06 KB. The response body is displayed as an HTML snippet:

```
1 <!-- HTML for static distribution bundle build -->
2 <!DOCTYPE html>
3 <html lang="en">
4 <head>
5   <meta charset="UTF-8">
6   <title>Swagger UI</title>
7   <link rel="stylesheet" type="text/css" href="/swagger-ui.css" />
8   <link rel="icon" type="image/png" href="/favicon-32x32.png" sizes="32x32" /><link rel="icon" type="image/png" href="/favicon-16x16.png" sizes="16x16" />
9   <style>
10
11
```

The screenshot shows a web browser window titled "Swagger UI" with the URL "localhost:3000/docs". The page displays the "WSE P8 - Secure Articles API" documentation. The main content area shows the API structure with sections for "Auth", "Articles", and "System". The sidebar on the left lists various schema definitions:

- User > Request or object
- RegisterRequest > Request or object
- LoginRequest > Request or object
- LoginResponse > Request or object
- RefreshRequest > Request or object
- Article > Request or object
- ArticleCreate > Request or object
- ArticleUpdate > Request or object
- ArticleResponse > Request or object
- ArticleListResponse > Request or object

G. Analisis

API Articles pada Praktikum #8 telah memenuhi prinsip RESTful dengan struktur modular (routes, controllers, services, repositories) dan status code konsisten (200, 201, 204, 400, 401, 403, 404, 500). Autentikasi dan otorisasi berbasis JWT berjalan benar, termasuk refresh token dan RBAC untuk membatasi akses create, update, delete artikel. Middleware keamanan seperti Helmet, CORS, rate limiter, input validation membuat API tahan serangan, observability diperkuat dengan Pino logger, correlation-id, /health, dan dokumentasi OpenAPI/Swagger memudahkan pengujian serta siap production.

H. Kesimpulan

Praktikum 8 berhasil membangun Secure & Observable RESTful API tingkat lanjut untuk resource Articles dengan menerapkan JWT Authentication (access & refresh token), role-based authorization (admin dan user), serta hardening menyeluruh seperti validation, rate limit, security headers Helmet, CORS whitelist, sanitasi input, dan pengelolaan secret melalui .env. Observability ditingkatkan dengan structured logging menggunakan Pino, middleware correlation-id, healthcheck endpoint, serta dokumentasi OpenAPI yang konsisten. Seluruh operasi CRUD Articles berjalan sesuai 7 RESTful Principles, dilengkapi pagination, filter, sort, dan search. Endpoint Auth (register, login, refresh, logout, me) berfungsi dengan baik, integrasi RBAC berjalan aman, dan struktur project tetap modular, production-ready, serta sesuai standar industri.

I. Checklist Praktikum

- JWT Authentication
- Refresh Token + revoke
- RBAC (user/admin)
- CRUD Articles
- Helmet + CORS whitelist
- Rate limiting
- Request logging (Pino)
- Correlation ID
- Health check
- Dokumentasi OpenAPI
- Evidence Postman lengkap
- Dokumentasi README.md selesai