

# **Laporan Modul Praktik**

**Membuat REST API : Konsultasi Kesehatan  
menggunakan Node.js**



**Mata kuliah Pemrograman Berbasis Platform  
Kelompok 4**

Nabiel Fauzan Ibrahim (20230040222)

Arif Ripandi (20230040082)

Erin Nurfajrina (20230040320)

Adam Baldan (20230040259)

**PROGRAM STUDI TEKNIK INFORMATIKA  
FAKULTAS TEKNIK KOMPUTER DAN DESAIN  
UNIVERSITAS NUSA PUTRA  
TAHUN 2025**

## Pendahuluan

Laporan ini menjelaskan langkah-langkah praktis dalam membangun REST API menggunakan Node.js dengan basis data konsultasi\_kesehatan. API ini akan mencakup operasi CRUD (Create, Read, Update, Delete) dan validasi data menggunakan JsonWebToken untuk tabel utama, seperti pengguna, janji\_konsultasi, rekam\_medis, dan pembayaran.

## Persyaratan

### Software dan Tools

- Node.js
- MySQL Server
- Postman untuk pengujian API
- Text Editor/IDE (VS Code disarankan)

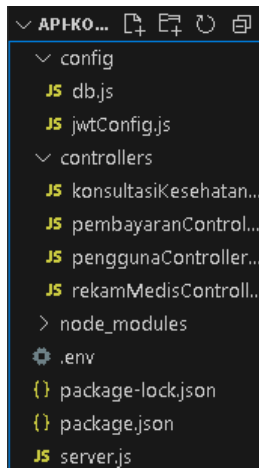
## Instalasi Paket node.js

```
C:\Users\ASUS\Documents>mkdir API-KONSULTASI-KESEHATAN-
C:\Users\ASUS\Documents>cd API-KONSULTASI-KESEHATAN-
C:\Users\ASUS\Documents\API-KONSULTASI-KESEHATAN->npm init -y
Wrote to C:\Users\ASUS\Documents\API-KONSULTASI-KESEHATAN-\package.json:
{
  "name": "api-konsultasi-kesehatan-",
  "version": "1.0.0",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC",
  "description": ""
}
```

## Membuat Database Konsultasi\_Kesehatan

```
Run SQL query/queries on server "127.0.0.1":
1 CREATE DATABASE konsultasi_kesehatan;
2
3 USE konsultasi_kesehatan;
4
5 -- Tabel Pengguna
6 CREATE TABLE pengguna (
7   id INT AUTO_INCREMENT PRIMARY KEY,
8   nama VARCHAR(255) NOT NULL,
9   email VARCHAR(255) UNIQUE NOT NULL,
10  kata_sandi VARCHAR(255) NOT NULL,
11  peran ENUM('pasien', 'dokter') DEFAULT 'pasien',
12  dibuat_pada TIMESTAMP DEFAULT CURRENT_TIMESTAMP
13 );
14
15 -- Tabel Janji Konsultasi
```

## Struktur Direktori Proyek



## Implementasi Rest API

### 1. Konfigurasi Koneksi Database

Membuat kode program untuk koneksi Database pada file Config/db.js



## 2. Setup Server

Gunakan **server.js** untuk membuka local host

```
1 const express = require('express');
2 const bodyParser = require('body-parser');
3 const penggunaController = require('./controllers/penggunaController');
4 const konsultasikanSahabatController = require('./controllers/konsultasikanSahabatController');
5 const rekomendasiController = require('./controllers/rekomendasiController');
6 const penyayaraController = require('./controllers/penyayaraController');
7 require('dotenv').config();
8
9 const app = express();
10 const PORT = 5000;
11
12 app.use(bodyParser.json());
13
14 app.use('/api', penggunaController);
15 app.use('/api', konsultasikanSahabatController);
16 app.use('/api', rekomendasiController);
17 app.use('/api', penyayaraController);
18
19 app.listen(PORT, () => {
20   console.log('Server berjalan di http://localhost:' + PORT);
21 });
22
```

## 3. Membuat Route dan Operasi CRUD

Routes dan operasi CRUD pada **penggunaController.js** dan menggunakan validasi data JsonWebToken

```
1 const express = require('express');
2 const router = express.Router();
3 const db = require('./config/db');
4 const bcrypt = require('bcrypt');
5 const jwt = require('jsonwebtoken');
6
7 const SECRET_KEY = 'your_secret_key';
8
9 router.post('/register', async (req, res) => {
10   const { nama, email, kata_sandi, peran } = req.body;
11
12   if (!nama || !email || !kata_sandi) {
13     return res.status(400).json({ message: 'Semua field harus diisi!' });
14   }
15
16   try {
17     const [existingUser] = await db.promise().query('SELECT * FROM pengguna WHERE email = ?', [email]);
18     if (existingUser.length > 0) {
19       return res.status(400).json({ message: 'Email sudah digunakan' });
20     }
21
22     const hashedPassword = await bcrypt.hash(kata_sandi, 10);
23
24     await db.promise().query('INSERT INTO pengguna (nama, email, kata_sandi, peran) VALUES (?, ?, ?, ?)', [
25       nama,
26       email,
27       hashedPassword,
28       peran
29     ]);
30
31     res.status(201).json({ message: 'Pengguna berhasil didaftarkan' });
32   } catch (error) {
33     console.error('Error during registration:', error);
34     res.status(500).json({ message: 'Internal Server Error' });
35   }
36 });
37
```

Full code <https://github.com/nability/Project-UAS-PBP>

## Routes dan operasi CRUD pada **konsultasiKesehatanController.js**

```
1. const express = require('express');
2. const router = express.Router();
3. const db = require('../config/db');
4.
5. // POST /konsultasiKesehatan - Membuat janji konsultasi baru
6. router.post('/konsultasi', async (req, res) => {
7.   const { id_pengguna, id_dokter, jadwal_konsultasi } = req.body;
8.
9.   if (!id_pengguna || !id_dokter || !jadwal_konsultasi) {
10.    return res.status(400).json({ message: 'Semua field harus diisi' });
11.  }
12.
13.  try {
14.    await db.promise().query(
15.      'INSERT INTO janji_konsultasi (id_pengguna, id_dokter, jadwal_konsultasi) VALUES (?, ?, ?)',
16.      [
17.        id_pengguna, id_dokter, jadwal_konsultasi
18.      ]
19.    );
20.    res.status(201).json({ message: 'Janji konsultasi berhasil dibuat' });
21.  } catch (error) {
22.    console.error('Error creating appointment:', error);
23.    res.status(500).json({ message: 'Internal Server Error' });
24.  }
25. });
```

Full code <https://github.com/nability/Project-UAS-PBP>

## Routes dan operasi CRUD pada **rekamMedisController.js**

```
1. const express = require('express');
2. const router = express.Router();
3. const db = require('../config/db');
4.
5. // POST /konsultasiKesehatan - Membuat janji konsultasi baru
6. router.post('/konsultasi', async (req, res) => {
7.   const { id_pengguna, id_dokter, jadwal_konsultasi } = req.body;
8.
9.   if (!id_pengguna || !id_dokter || !jadwal_konsultasi) {
10.    return res.status(400).json({ message: 'Semua field harus diisi' });
11.  }
12.
13.  try {
14.    await db.promise().query(
15.      'INSERT INTO janji_konsultasi (id_pengguna, id_dokter, jadwal_konsultasi) VALUES (?, ?, ?)',
16.      [
17.        id_pengguna, id_dokter, jadwal_konsultasi
18.      ]
19.    );
20.    res.status(201).json({ message: 'Janji konsultasi berhasil dibuat' });
21.  } catch (error) {
22.    console.error('Error creating appointment:', error);
23.    res.status(500).json({ message: 'Internal Server Error' });
24.  }
25. });
```

Full code <https://github.com/nability/Project-UAS-PBP>

## Routes dan operasi CRUD pada `pembayaranController.js`

```
1 const express = require('express');
2 const router = express.Router();
3 const db = require('../config/db');
4
5 // POST /konsultasiKesehatan - Membuat janji konsultasi baru
6 router.post('/konsultasi', async (req, res) => {
7   const { id_pengguna, id_dokter, jadwal_konsultasi } = req.body;
8
9   if (!id_pengguna || !id_dokter || !jadwal_konsultasi) {
10     return res.status(400).json({ message: 'Semua field harus diisi' });
11   }
12
13   try {
14     await db.promise().query(
15       'INSERT INTO janji_konsultasi (id_pengguna, id_dokter, jadwal_konsultasi) VALUES (?, ?, ?)',
16       [id_pengguna, id_dokter, jadwal_konsultasi]
17     );
18     res.status(201).json({ message: 'Janji konsultasi berhasil dibuat' });
19   } catch (error) {
20     console.error('Error creating appointment:', error);
21     res.status(500).json({ message: 'Internal Server Error' });
22   }
23 });
```

Full code <https://github.com/nability/Project-UAS-PBP>

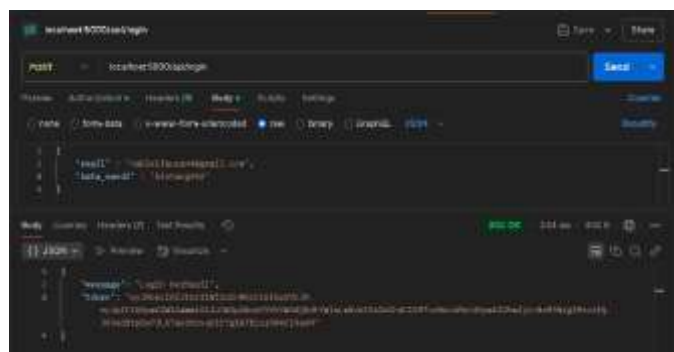
## Pengujian API

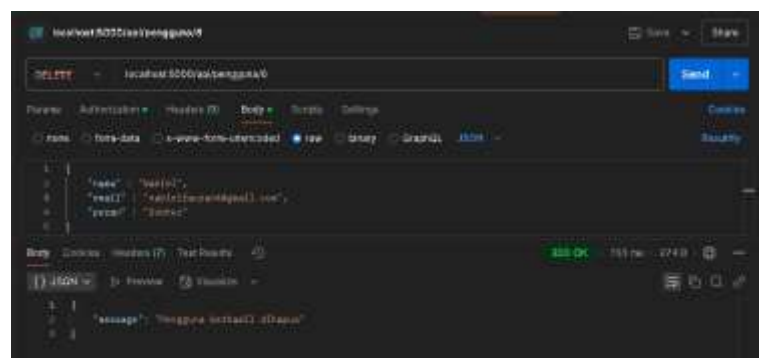
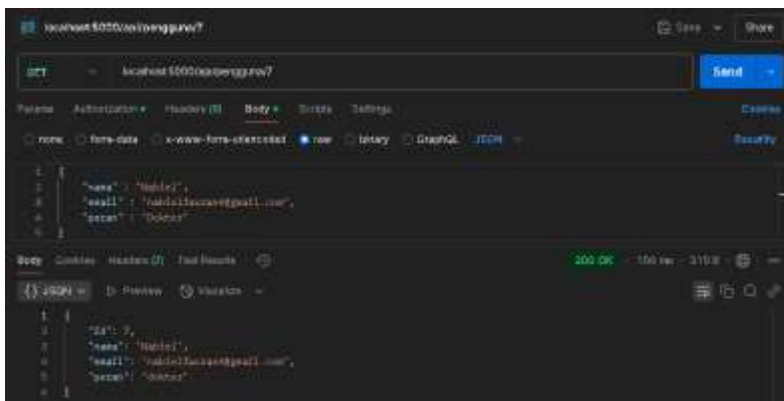
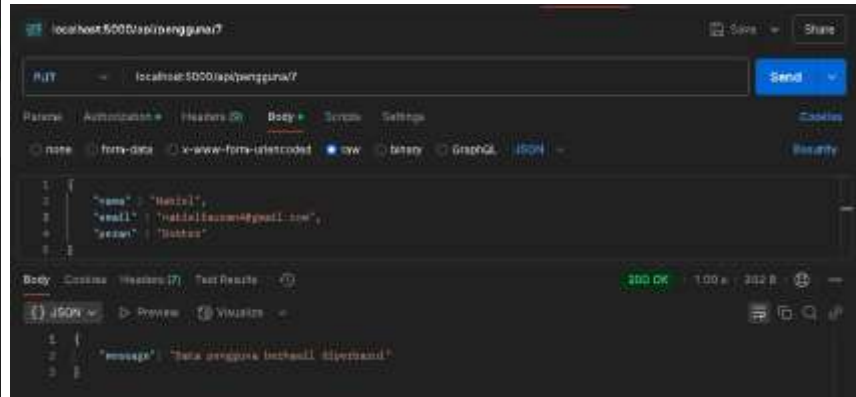
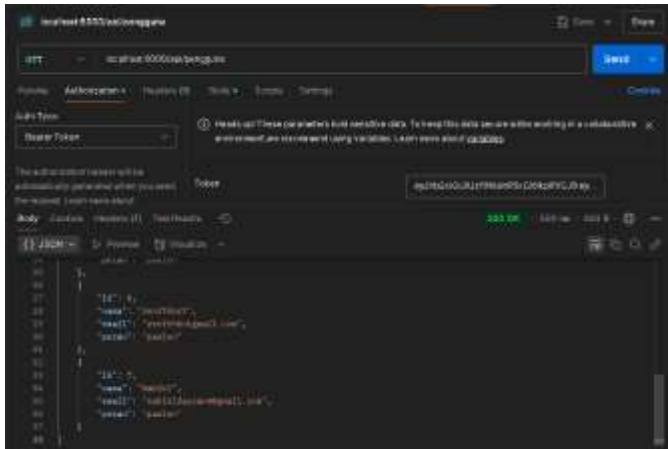
Pengujian digunakan menggunakan Postman

Pengujian pada `penggunaController.js` dengan endpoint :

1. **GET /api/pengguna**: Menampilkan semua pengguna yang telah login
2. **POST /api/register**: Mendaftarkan pengguna baru
3. **POST /api/login**: Menambahkan pengguna baru
4. **GET /api/pengguna/id** : Menampilkan pengguna berdasarkan id
5. **PUT /api/pengguna/id**: Merubah data pengguna
6. **DELETE /api/pengguna/id**: Menghapus pengguna

## Hasil Output

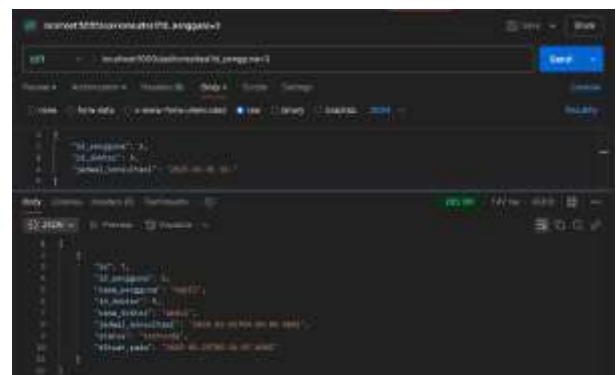




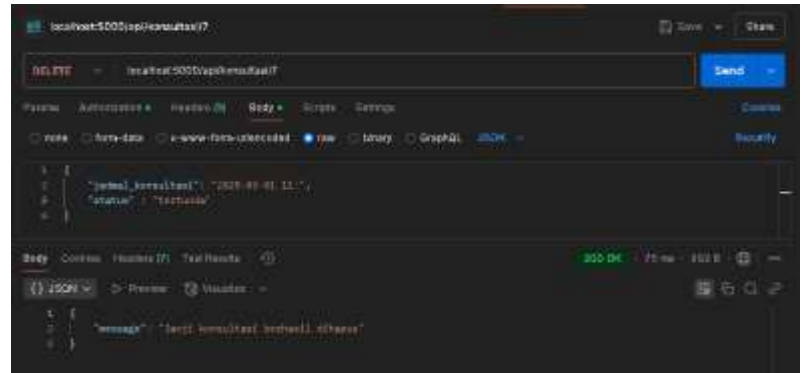
## Pengujian pada penggunaController.js dengan endpoint :

1. **GET /api/konsultasi:** Menampilkan semua jadwal untuk janji konsultasi
2. **GET/api/konsultasi/id :** Menampilkan jadwal untuk janji konsultasi berdasarkan id
3. **POST /api/konsultasi:** Mendaftarkan untuk jadwal janji konsultasi
4. **PUT/api/konsultasi/id:** Merubah data jadwal konsultasi
5. **DELETE/api/konsultasi/id:** Menghapus data konsultasi

## Hasil Output :



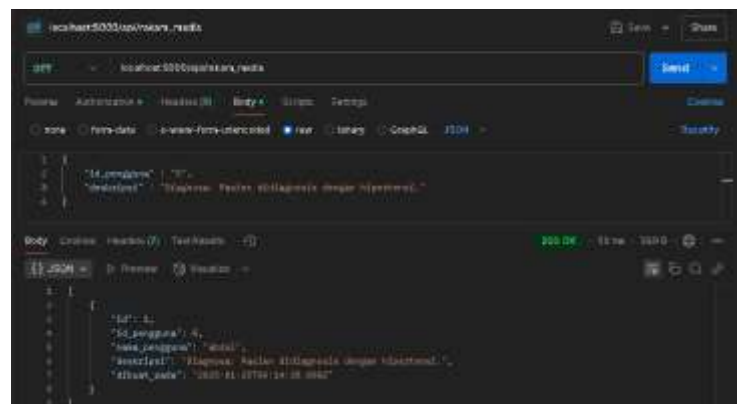
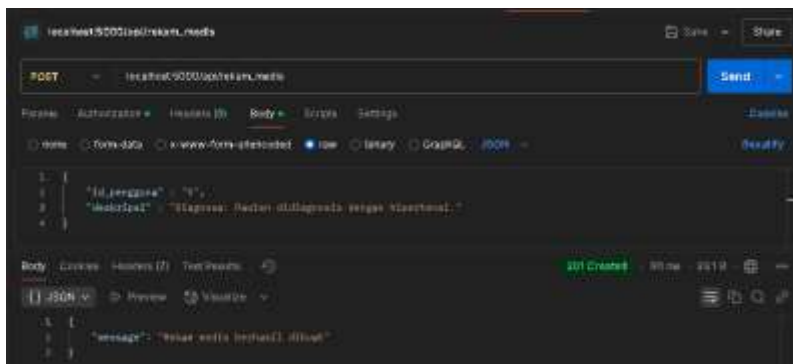




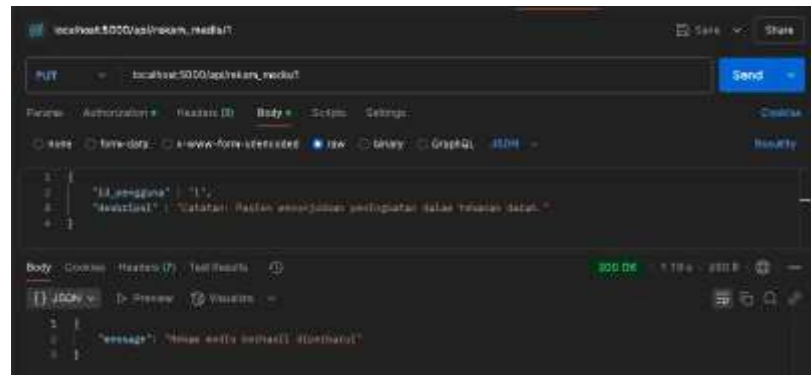
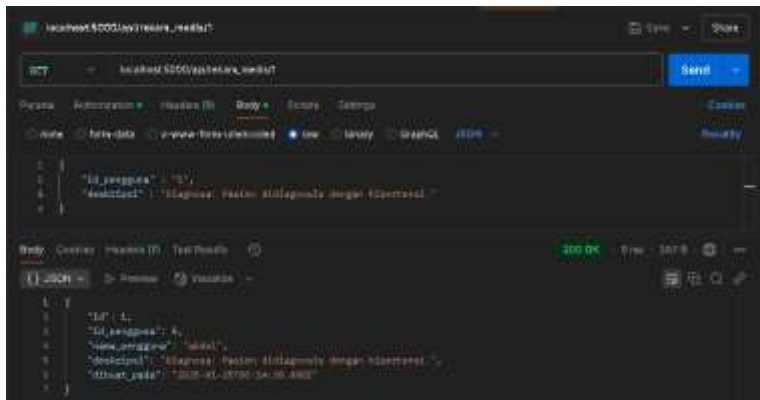
### Pengujian pada rekamMedisController.js dengan endpoint :

1. **GET /api/konsultasi:** Menampilkan semua hasil konsultasi
2. **GET /api/konsultasi/id :** Menampilkan hasil konsultasi berdasarkan id
3. **POST /api/konsultasi:** Membuat data rekam medis untuk hasil konsultasi
4. **PUT /api/konsultasi/id:** Merubah data hasil konsultasi
5. **DELETE /api/konsultasi/id:** Menghapus data hasil konsultasi

### Hasil Output :



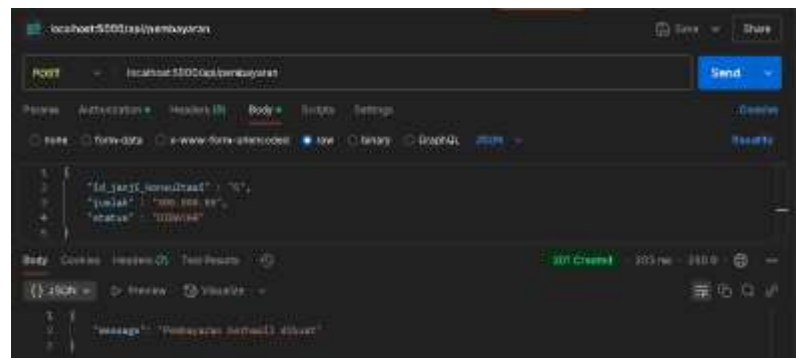
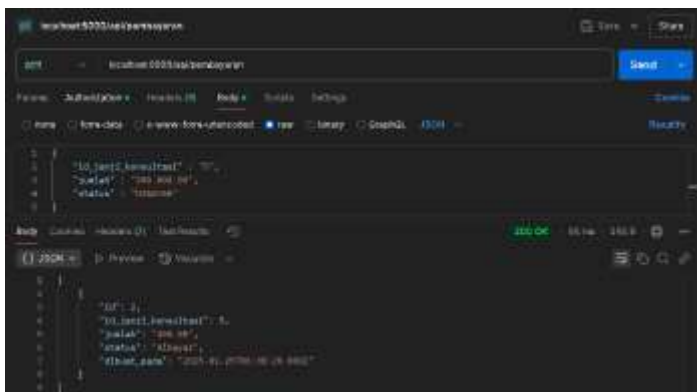


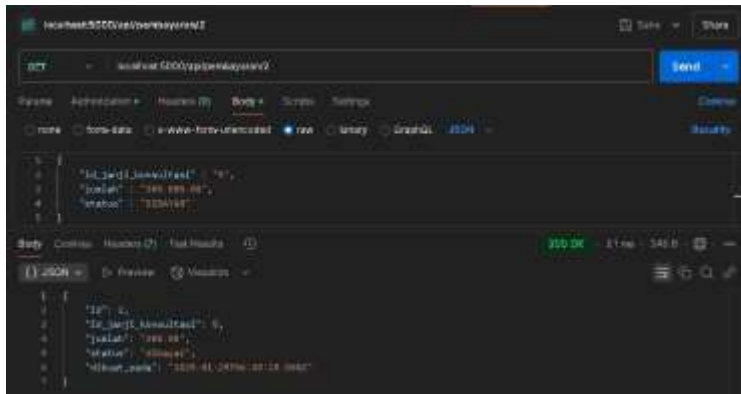


### Pengujian pada pembayaranController.js dengan endpoint :

1. **GET /api/konsultasi:** Menampilkan semua keterangan pembayaran
2. **GET/api/konsultasi/id :** Menampilkan keterangan pembayaran berdasarkan id
3. **POST /api/konsultasi:** Membuat data keterangan pembayaran
4. **PUT/api/konsultasi/id:** Merubah data keterangan pembayaran
5. **DELETE/api/konsultasi/id:** Menghapus data keterangan pembayaran

### Hasil Output :





## Kesimpulan

REST API yang dibuat memungkinkan pengelolaan data untuk aplikasi konsultasi kesehatan, dengan fitur pengguna dimulai dari daftar dan login, membuat jadwal janji untuk konsultasi, membuat rekam medis hasil konsultasi, dan juga membuat keterangan pembayaran.