# DOKUMENTASI

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":

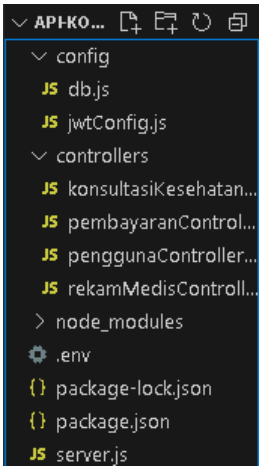
```
CREATE DATABASE konsultasi_kesehatan;

USE konsultasi_kesehatan;

-- Tabel Pengguna
CREATE TABLE pengguna (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nama VARCHAR(255) NOT NULL,
    email VARCHAR(255) UNIQUE NOT NULL,
    kata_sandi VARCHAR(255) NOT NULL,
    peran ENUM('pasien', 'dokter') DEFAULT 'pasien',
    dibuat_pada TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabel Jenis Konsultasi
```

Struktur Direktori Proyek




## Implementasi Rest API

1. **Konfigurasi Koneksi Database**

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

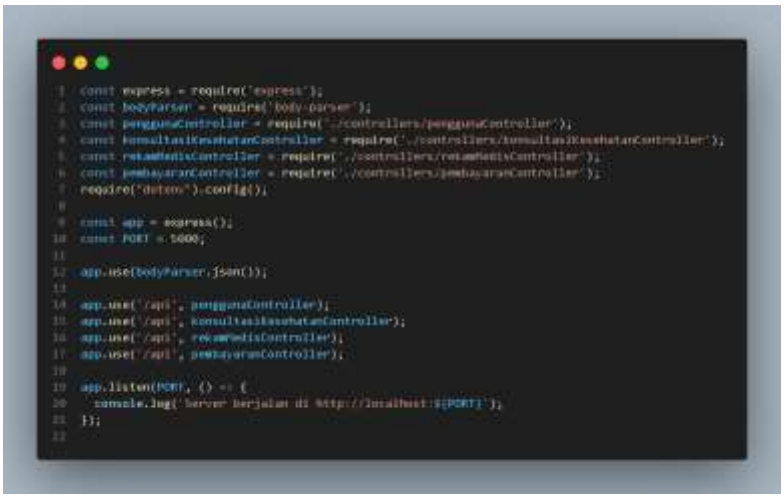
```
const mysql = require('mysql2');

const connection = mysql.createConnection({
  host: 'localhost',
  user: 'root',
  password: '',
  database: 'konsultasi_kesehatan'
});

connection.connect((err) => {
  if (err) {
    console.error('Kesalahan koneksi ke database: ', err);
    return;
  }
  console.log('Koneksi ke database berhasil!');
});

module.exports = connection;
```

2. Setup Server



```
const express = require('express');
const bodyParser = require('body-parser');
const penggunaController = require('./controllers/penggunaController');
const konsultasiKesehatanController = require('./controllers/konsultasiKesehatanController');
const rekamMedisController = require('./controllers/rekamMedisController');
const pembayaranController = require('./controllers/pembayaranController');
require('dotenv').config();

const app = express();
const PORT = 5000;

app.use(bodyParser.json());

app.use('/api', penggunaController);
app.use('/api', konsultasiKesehatanController);
app.use('/api', rekamMedisController);
app.use('/api', pembayaranController);

app.listen(PORT, () => {
  console.log(`Server berjalan di http://localhost:${PORT}`);
});
```

3. Membuat Route dan Operasi CRUD

```
const express = require('express');
const router = express.Router();
const db = require('../config/db');
const bcrypt = require('bcrypt');
const jwt = require('jsonwebtoken');


const SECRET_KEY = 'your_secret_key';

router.post('/register', async (req, res) => {
    const { nama, email, kata_sandi, peran } = req.body;

    if (!nama || !email || !kata_sandi) {
        return res.status(400).json({ message: 'Semua field harus diisi!' });
    }

    try {

        const [existingUser] = await db.promise().query('SELECT * FROM pengguna WHERE email = ?', [email]);
        if (existingUser.length > 0) {
            return res.status(400).json({ message: 'Email sudah digunakan!' });
        }


        const hashedPassword = await bcrypt.hash(kata_sandi, 10);

        await db.promise().query('INSERT INTO pengguna (nama, email, kata_sandi) VALUES (?, ?, ?)', [
            nama,
            email,
            hashedPassword,
        ]);

        res.status(201).json({ message: 'Pengguna berhasil didaftarkan' });
    } catch (error) {
        console.error('Error during registration:', error);
        res.status(500).json({ message: 'Internal Server Error' });
    }
});
```

Full code [https://github.com/nability/Project-UAS-PBP](https://github.com/nability/Project-UAS-PBP)

```
const express = require('express');
const router = express.Router();
const db = require('../config/db');

// POST /konsultasiKesehatan - Membuat janji konsultasi baru
router.post('/konsultasi', async (req, res) => {
    const { id_pengguna, id_dokter, jadwal_konsultasi } = req.body;

    if (!id_pengguna || !id_dokter || !jadwal_konsultasi) {
        return res.status(400).json({ message: 'Semua field harus diisi!' });
    }

    try {
        await db.promise().query(

            INSERT INTO janji_konsultasi (id_pengguna, id_dokter, jadwal_konsultasi)
            VALUES (?, ?, ?)
            `,
            [id_pengguna, id_dokter, jadwal_konsultasi]
        );
        res.status(201).json({ message: 'Janji konsultasi berhasil dibuat' });
    } catch (error) {
        console.error('Error creating appointment:', error);
        res.status(500).json({ message: 'Internal Server Error' });
    }
});
```

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

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

```
const express = require('express');
const router = express.Router();
const db = require('../config/db');

// POST /konsultasiKesehatan - Membuat janji konsultasi baru
router.post('/konsultasi', async (req, res) => {
    const { id_pengguna, id_dokter, jadwal_konsultasi } = req.body;

    if (!id_pengguna || !id_dokter || !jadwal_konsultasi) {
        return res.status(400).json({ message: 'Semua field harus diisi!' });
    }

    try {
        await db.promise().query(

            INSERT INTO janji_konsultasi (id_pengguna, id_dokter, jadwal_konsultasi)
            VALUES (?, ?, ?)
            `,
            [id_pengguna, id_dokter, jadwal_konsultasi]
        );
        res.status(201).json({ message: 'Janji konsultasi berhasil dibuat' });
    } catch (error) {
        console.error('Error creating appointment:', error);
        res.status(500).json({ message: 'Internal Server Error' });
    }
});
```

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

Routes dan operasi CRUD pada **pembayaranController.js**

```
const express = require('express');
const router = express.Router();
const db = require('../config/db');

// POST /konsultasiKesehatan - Membuat janji konsultasi baru
router.post('/konsultasi', async (req, res) => {
    const { id_pengguna, id_dokter, jadwal_konsultasi } = req.body;

    if (!id_pengguna || !id_dokter || !jadwal_konsultasi) {
        return res.status(400).json({ message: 'Semua field harus diisi!' });
    }

    try {
        await db.promise().query(

            INSERT INTO janji_konsultasi (id_pengguna, id_dokter, jadwal_konsultasi)
            VALUES (?, ?, ?)
            ,
            [id_pengguna, id_dokter, jadwal_konsultasi]
        );
        res.status(201).json({ message: 'Janji konsultasi berhasil dibuat' });
    } catch (error) {
        console.error('Error creating appointment:', error);
        res.status(500).json({ message: 'Internal Server Error' });
    }
});
```

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

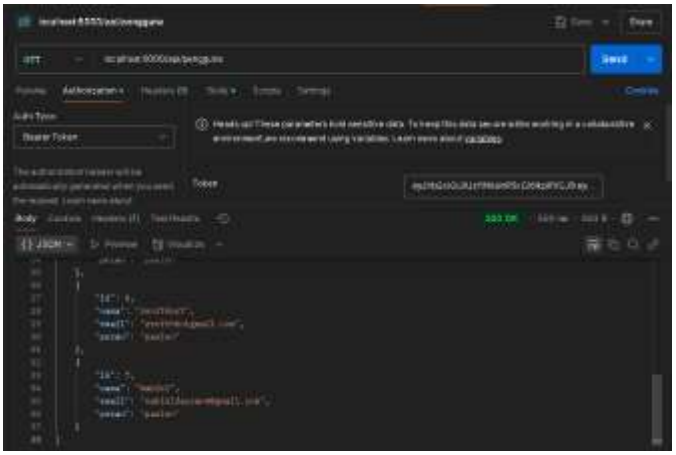
## Pengujian API

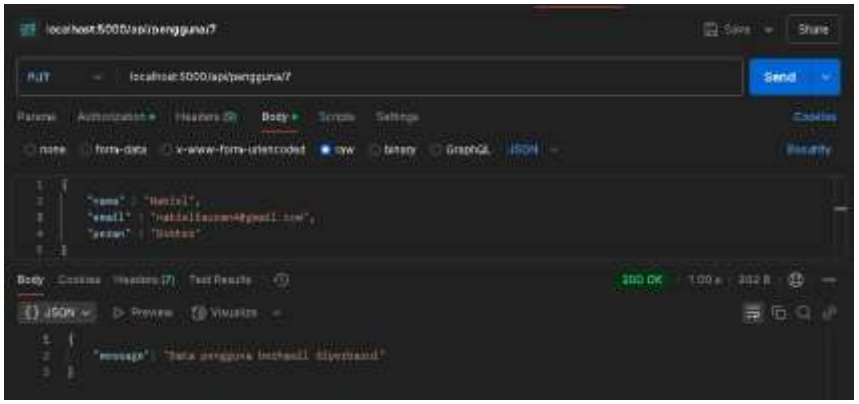
Pengujian digunakan menggunakan Postman

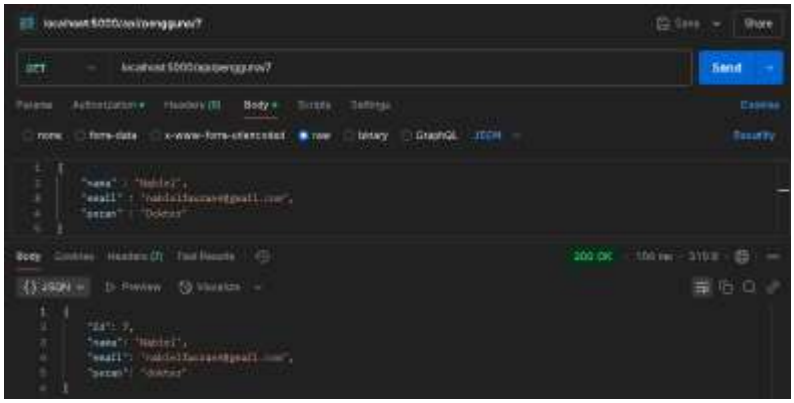
**Pengujian pada penggunaController.js dengan endpoint :**

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 :**

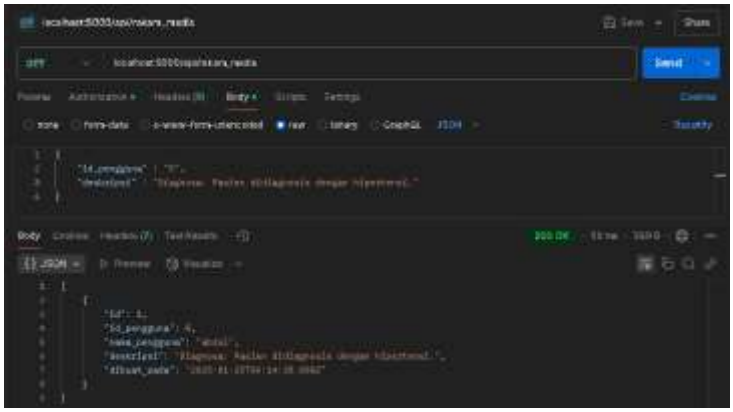
1. **GET /api/konsultasi**: Menampilkan semua jadwal untuk janji konsultasi
2. **GET/api/konsultasi/id :** Menampilkan jadwal untuk janji konsultasi berdasarkan id
3. **POST /api/konsultasi**: Mendaftarkan untuk jadwal janji konsultasi
4. **PUT/api/konsultasi/id:** Merubah data jadwal konsultasi
5. **DELETE/api/konsultasi/id:**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

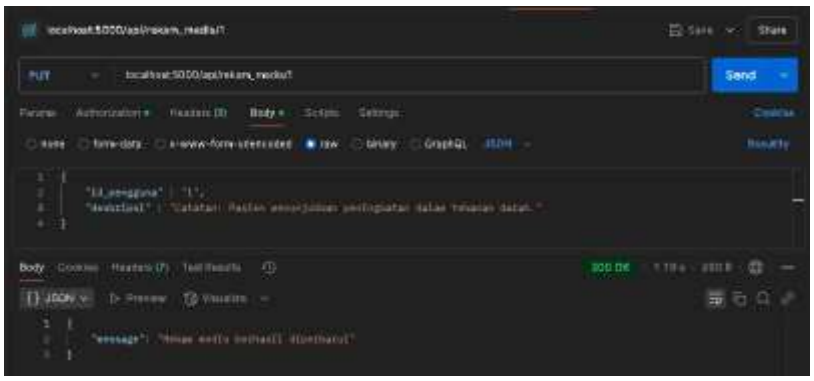
**Hasil Output :**

**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 :**

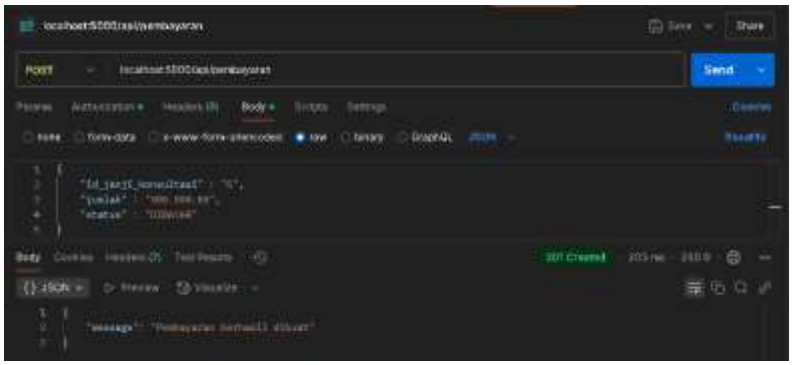




**Pengujian pada pembayaranController.js dengan endpoint :**

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.