LAPORAN PRAKTIKUM

KONSTRUKSI PERANGKAT LUNAK

MODUL 9

Rest API

Disusun Oleh :

Aji Prasetyo Nugroho / 2211104049

S1SE-06-2

Asisten Praktikum :

Muhammad Taufiq Hidayat

Dosen Pengampu :

Riyan Dwi Yulian Prakoso, S.Kom., M.Kom.

PROGRAM STUDI S1 SOFTWARE ENGINEERING

FAKULTAS INFORMATIKA

TELKOM UNIVERSITY PURWOKERTO

2025

A. LANDASAN TEORI

1. API (Application Programming Interface)

   API (Application Programming Interface) adalah sekumpulan definisi dan protokol yang memungkinkan perangkat lunak untuk saling berkomunikasi. API memungkinkan satu aplikasi untuk mengakses fitur atau data dari aplikasi lain, tanpa harus mengetahui bagaimana aplikasi tersebut bekerja secara internal. API digunakan untuk menghubungkan berbagai sistem secara efisien dan terstruktur.

2. REST (Representational State Transfer)

   REST adalah arsitektur perangkat lunak berbasis web yang menggunakan protokol HTTP untuk pertukaran data. REST diperkenalkan oleh Roy Fielding pada tahun 2000 dalam disertasinya dan kini menjadi standar dalam pengembangan web service. REST bekerja berdasarkan prinsip resource (sumber daya) yang diakses melalui URL dan dimanipulasi menggunakan metode HTTP seperti GET, POST, PUT, dan DELETE.

3. REST API

   REST API adalah implementasi API yang mengikuti prinsip REST. REST API menyediakan antarmuka yang memungkinkan klien (client) untuk berkomunikasi dengan server menggunakan HTTP. Data yang ditransfer biasanya menggunakan format JSON (JavaScript Object Notation) karena sifatnya yang ringan dan mudah dipahami.

   Karakteristik utama REST API antara lain:

   a. Stateless: Setiap permintaan dari klien ke server harus mengandung semua informasi yang dibutuhkan, karena server tidak menyimpan status sesi.
   b. Client-Server Architecture: Memisahkan tanggung jawab antara klien (pengguna) dan server (penyedia data).
   c. Cacheable: Respons dapat di-cache untuk meningkatkan performa.
   d. Uniform Interface: Memiliki antarmuka umum yang konsisten, seperti penggunaan HTTP method dan URL sebagai representasi resource.

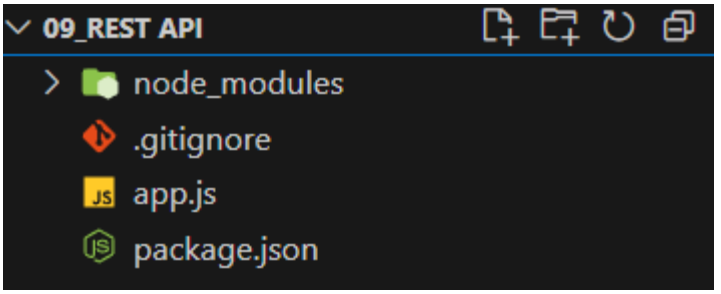
4. HTTP Method

   REST API umumnya menggunakan metode HTTP untuk mengatur operasi terhadap resource:

   a. GET: Mengambil data dari server.
   b. POST: Mengirim data baru ke server.
   c. PUT: Memperbarui data yang ada.
   d. DELETE: Menghapus data dari server.

B. GUIDED

1. Struktur Folder



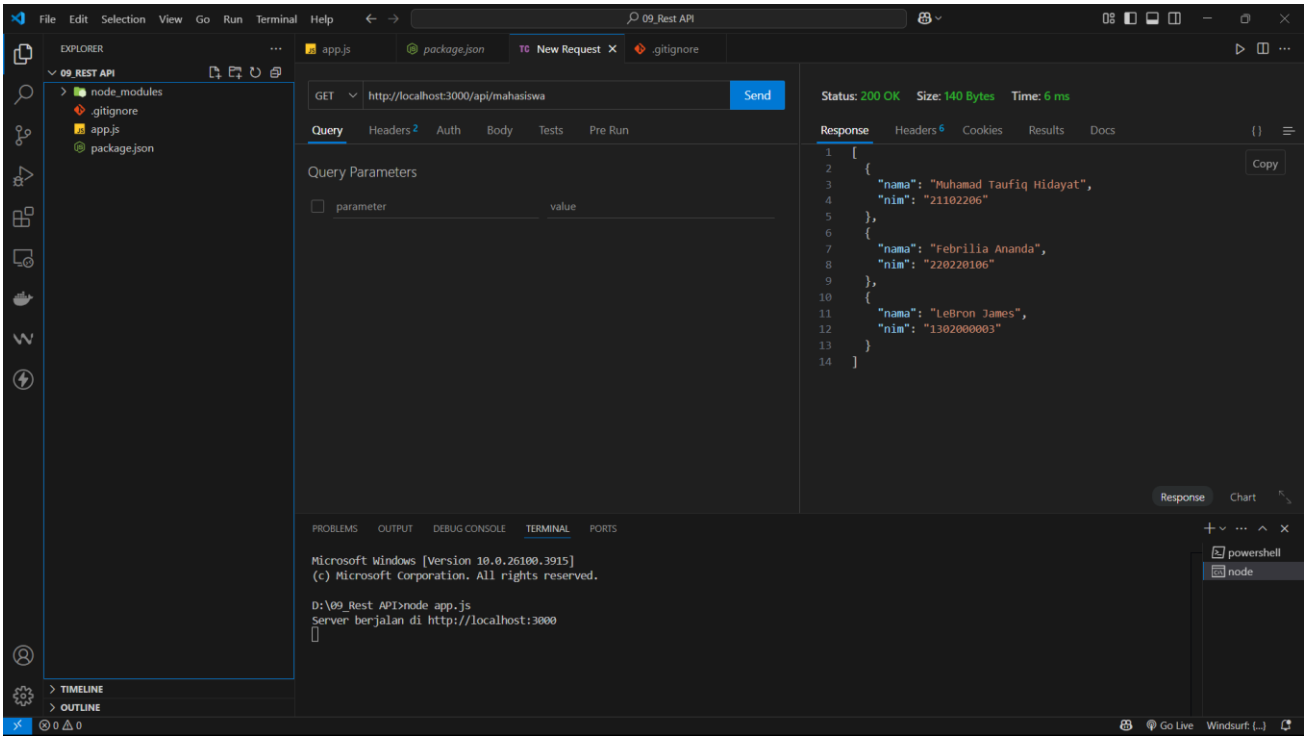
2. app.js

   Source Code

```
const express = require('express');
const app = express();
const port = 3000;

app.use(express.json());

// Simpan data mahasiswa di array static
let mahasiswa = [
  { nama: "Muhamad Taufiq Hidayat", nim: "21102206" },
  { nama: "Febrilia Ananda", nim: "220220106" },
  { nama: "LeBron James", nim: "1302000003" }
];

// GET semua mahasiswa
app.get('/api/mahasiswa', (req, res) => {
  res.json(mahasiswa);
});

// GET mahasiswa berdasarkan index
app.get('/api/mahasiswa/:index', (req, res) => {
  const index = parseInt(req.params.index);
  if (index >= 0 && index < mahasiswa.length) {
    res.json(mahasiswa[index]);
  } else {
    res.status(404).json({ message: 'Data tidak ditemukan' });
  }
});

// POST mahasiswa baru
app.post('/api/mahasiswa', (req, res) => {
  const { nama, nim } = req.body;
  mahasiswa.push({ nama, nim });
  res.status(201).json({ message: 'Data berhasil ditambahkan' });
});

// DELETE mahasiswa berdasarkan index
app.delete('/api/mahasiswa/:index', (req, res) => {
  const index = parseInt(req.params.index);
  if (index >= 0 && index < mahasiswa.length) {
    mahasiswa.splice(index, 1);
```

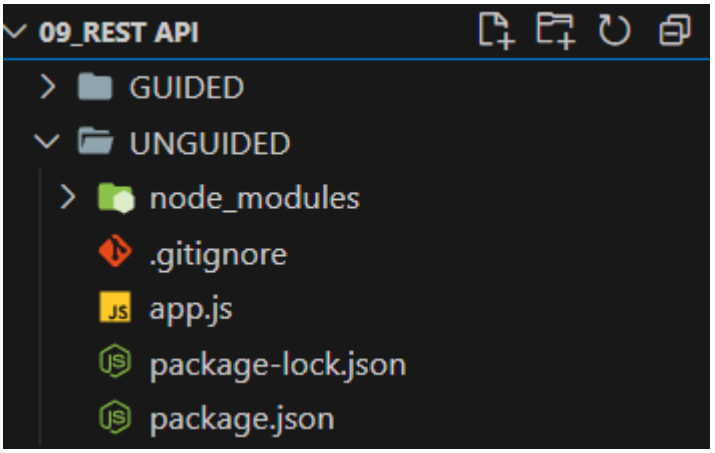
```
    res.json({ message: 'Data berhasil dihapus' });
  } else {
    res.status(404).json({ message: 'Data tidak ditemukan' });
  }
});

app.get('/', (req, res) => {
  res.send('REST API Mahasiswa - Gunakan endpoint /api/mahasiswa');
});

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

Output



C. UNGUIDED

1. Struktur Folder



2. Code Program
    1) app.js
        Source Code

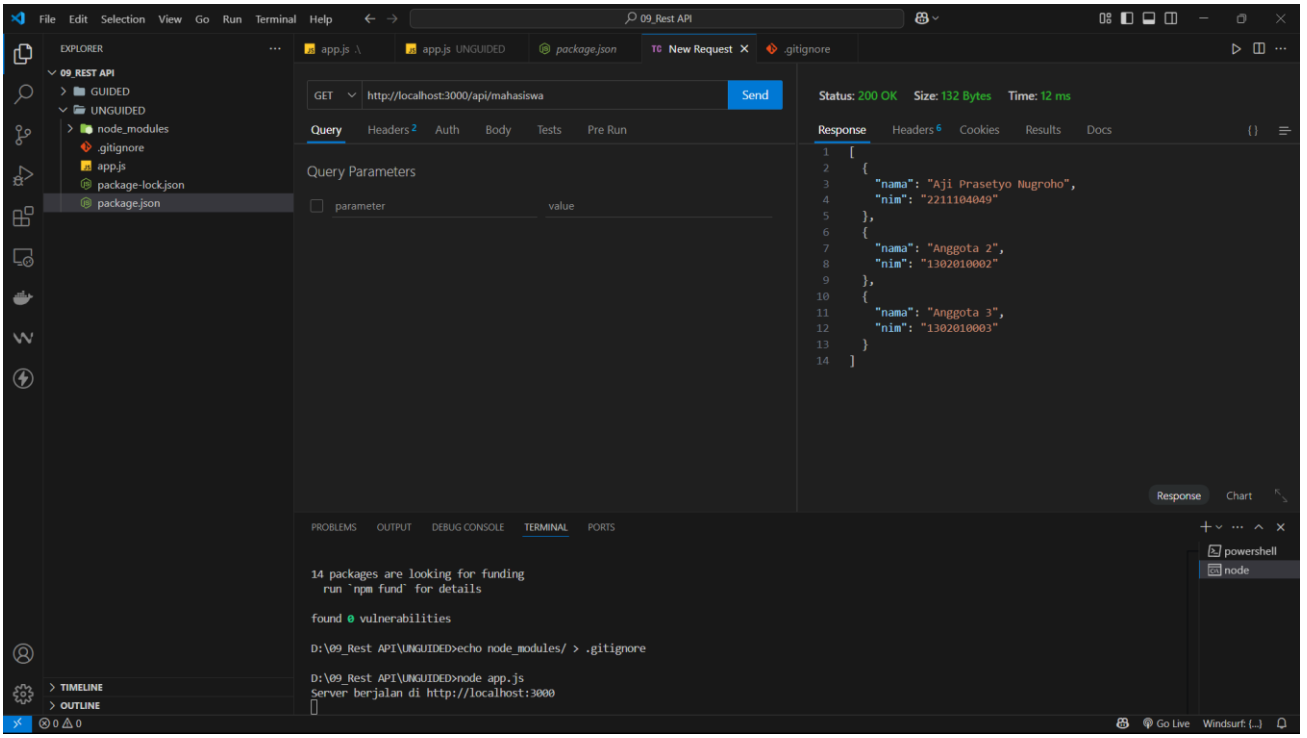
```
const express = require('express');
const app = express();
const PORT = 3000;

app.use(express.json());

let mahasiswa = [
  { nama: "Aji Prasetyo Nugroho", nim: "2211104049" },
  { nama: "Anggota 2", nim: "1302010002" },
  { nama: "Anggota 3", nim: "1302010003" }
];

// GET seluruh mahasiswa
app.get('/api/mahasiswa', (req, res) => {
  res.json(mahasiswa);
});

// GET mahasiswa berdasarkan index
app.get('/api/mahasiswa/:index', (req, res) => {
  const index = parseInt(req.params.index);
  if (mahasiswa[index]) {
    res.json(mahasiswa[index]);
  } else {
    res.status(404).json({ message: "Mahasiswa tidak ditemukan" });
  }
});

// POST mahasiswa baru
app.post('/api/mahasiswa', (req, res) => {
  const { nama, nim } = req.body;
  if (nama && nim) {
    mahasiswa.push({ nama, nim });
    res.status(201).json({ message: "Mahasiswa ditambahkan", data: { nama, nim } });
```

```
  } else {
    res.status(400).json({ message: "Data tidak lengkap" });
  }
});

// DELETE mahasiswa berdasarkan index
app.delete('/api/mahasiswa/:index', (req, res) => {
  const index = parseInt(req.params.index);
  if (mahasiswa[index]) {
    const deleted = mahasiswa.splice(index, 1);
    res.json({ message: "Mahasiswa dihapus", data: deleted[0] });
  } else {
    res.status(404).json({ message: "Mahasiswa tidak ditemukan" });
  }
});

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

3. Dokumentasi Uji Coba

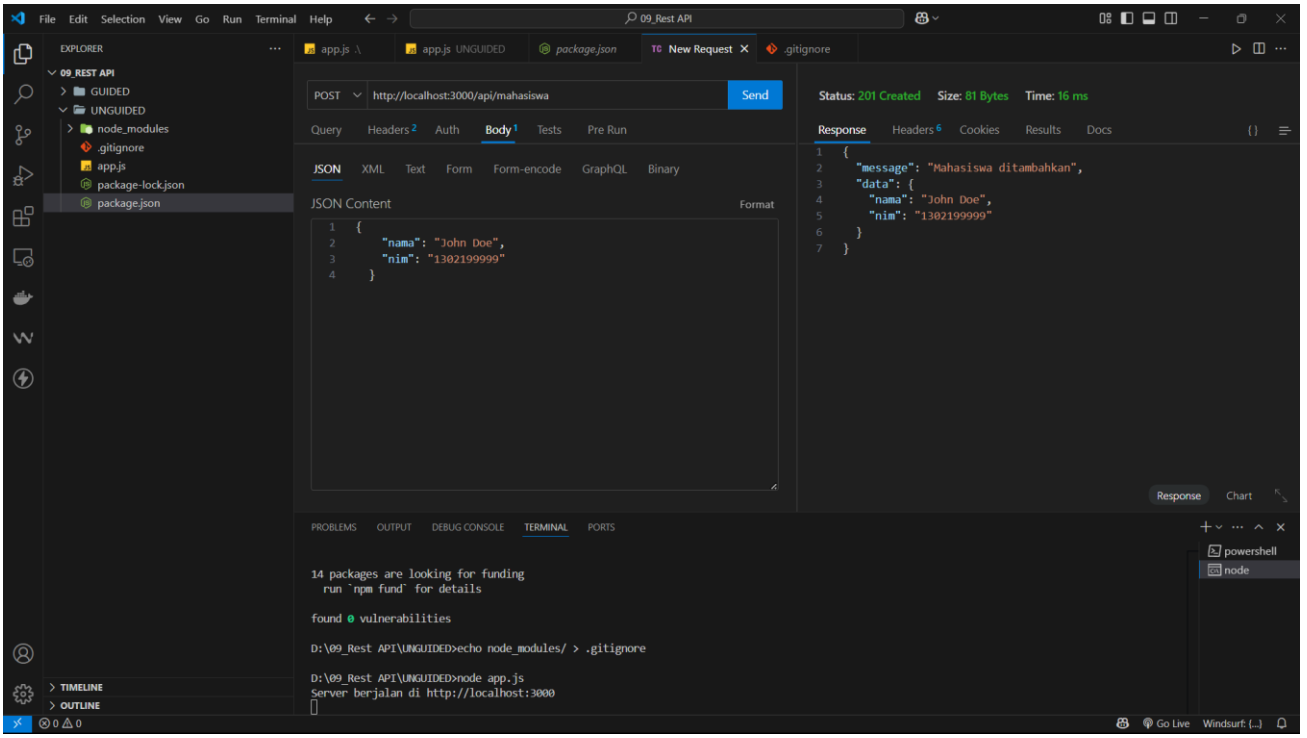
1.) Jalankan GET /api/mahasiswa saat program pertama kali dijalankan

Output



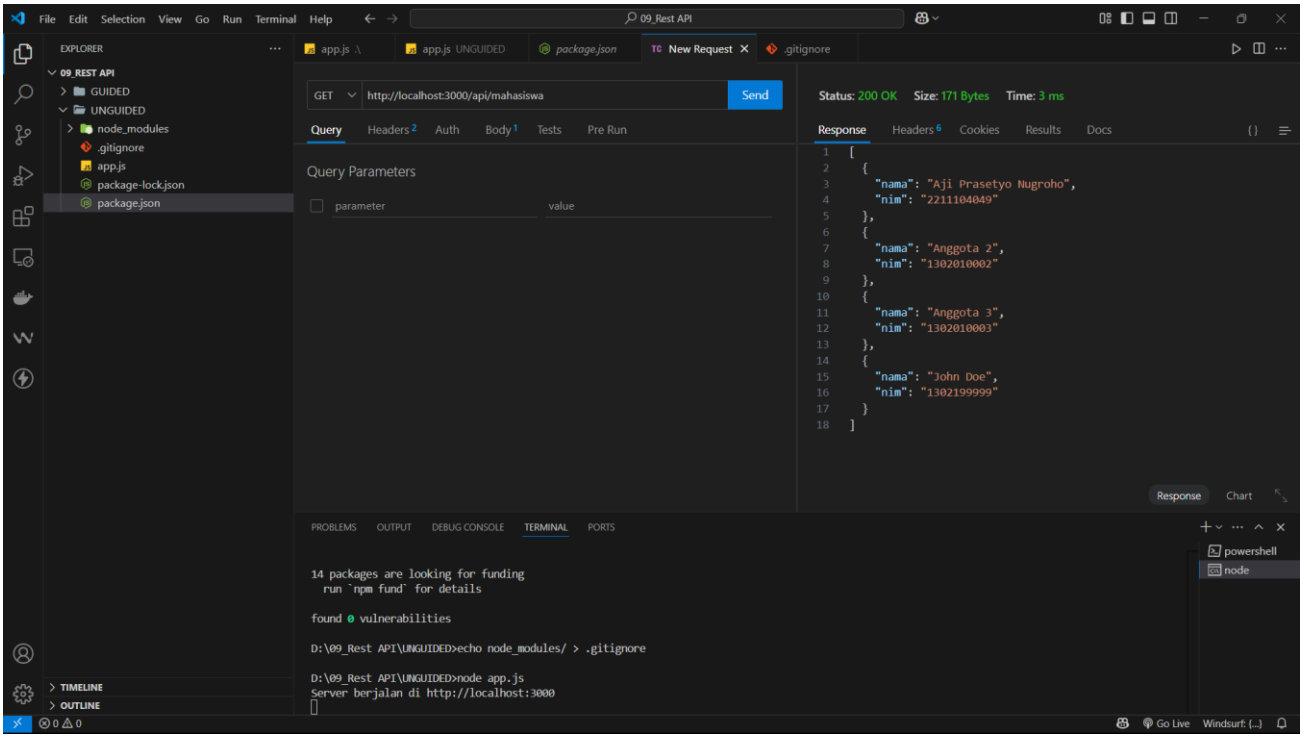
2.) Jalankan POST /api/mahasiswa untuk menambahkan: John Doe – 1302199999

Output



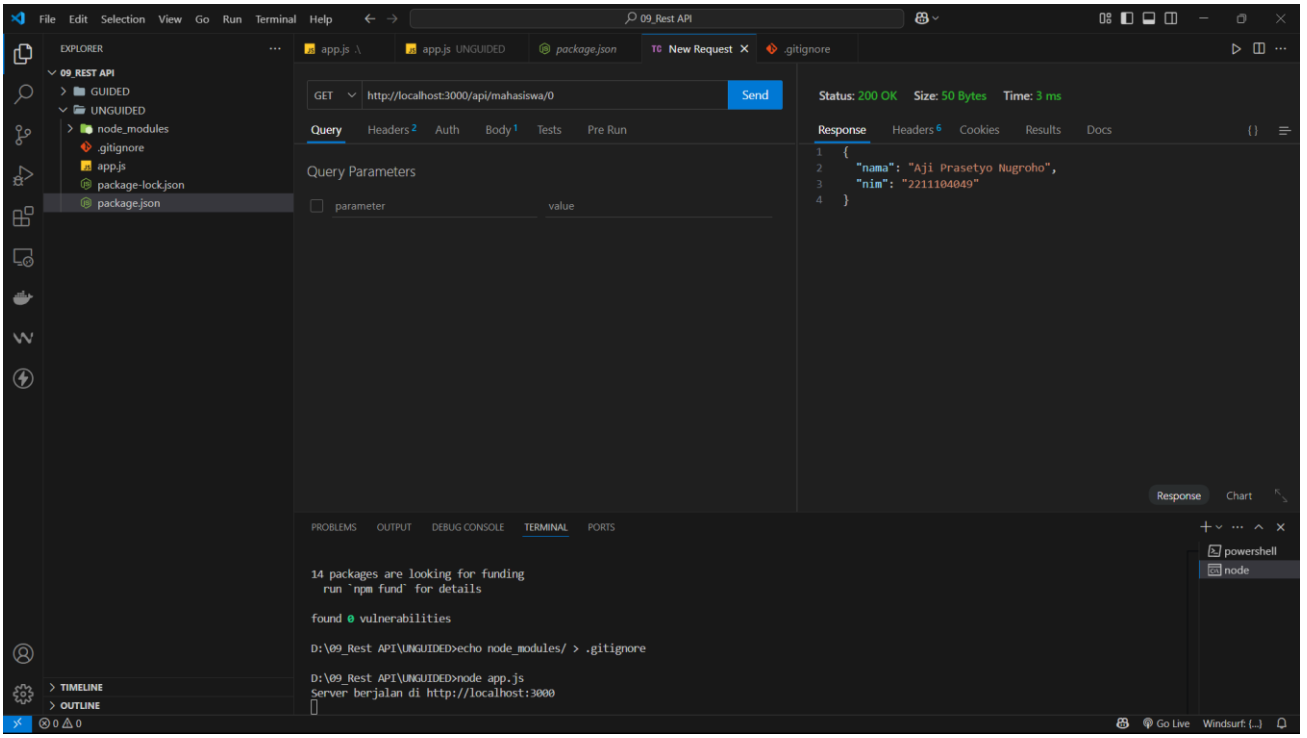
3.) Jalankan GET /api/mahasiswa lagi dan pastikan John Doe muncul

Output



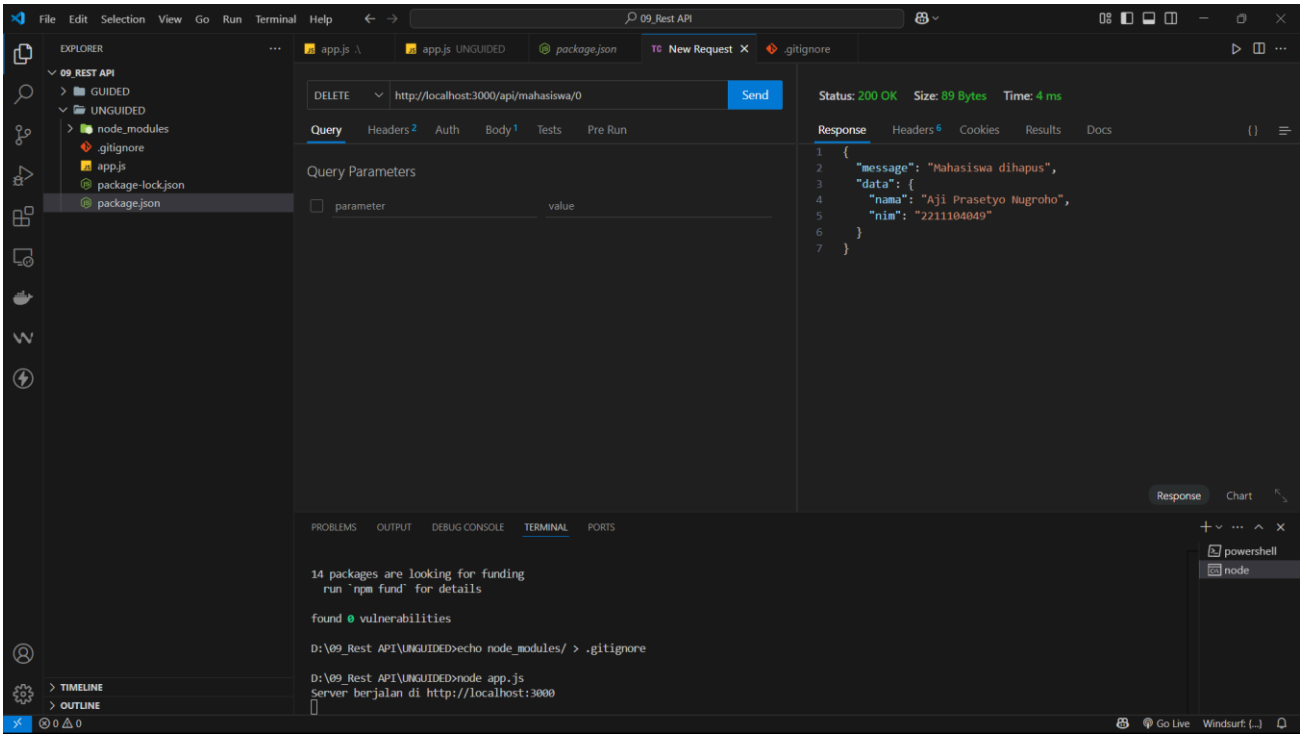
4.) Jalankan GET /api/mahasiswa/0 dan pastikan data pertama (nama Anda) muncul

Output



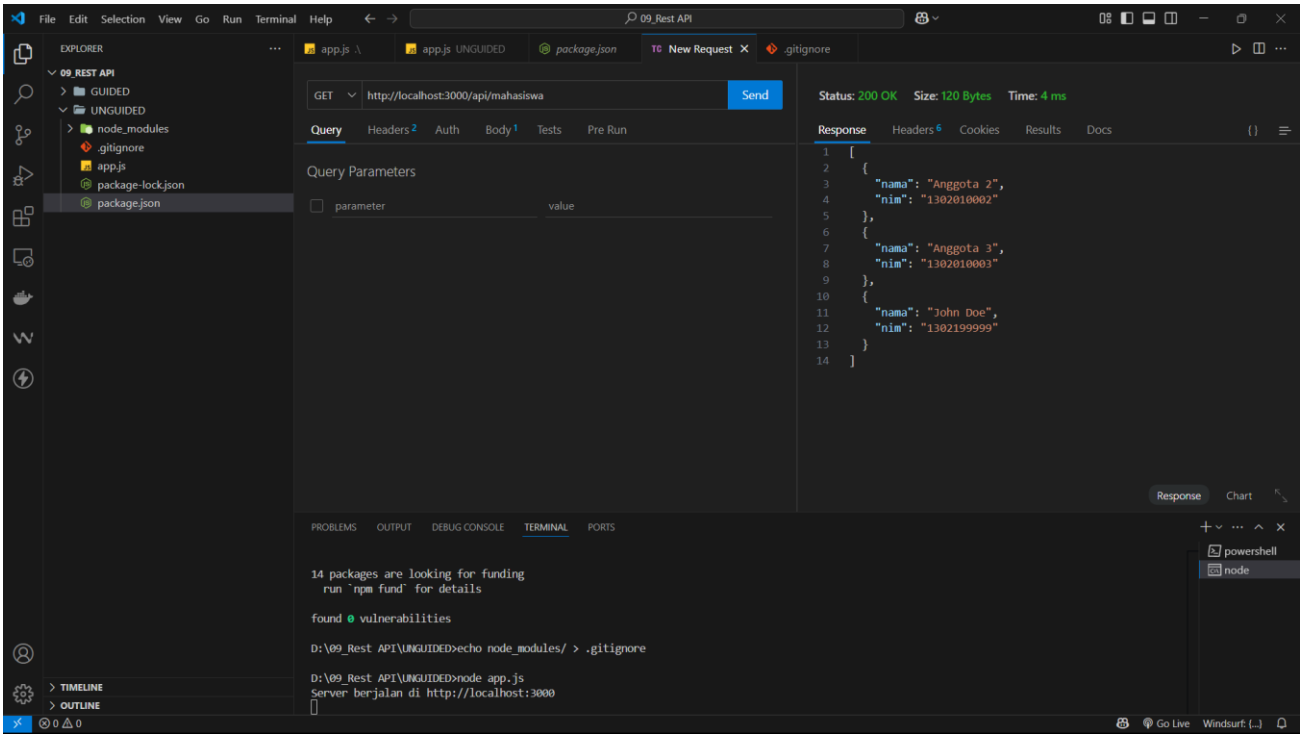
5.) Jalankan DELETE /api/mahasiswa/0 untuk menghapus data pertama

Output



6.) Jalankan GET /api/mahasiswa lagi dan pastikan data Anda sudah hilang

Output



D. PENJELASAN SINGKAT PROGRAM

Kode di atas merupakan implementasi sederhana REST API menggunakan Node.js dan Express. API ini mengelola data mahasiswa yang disimpan di dalam array JavaScript statis, tanpa menggunakan database. Di awal, data array mahasiswa sudah berisi tiga entri, termasuk nama pembuat tugas di urutan pertama. Aplikasi mendengarkan request pada port 3000, dengan endpoint dasar /api/mahasiswa, dan menggunakan middleware express.json() untuk memproses data JSON yang dikirim melalui body request, terutama untuk operasi POST.

Terdapat empat endpoint utama yang dibuat: GET /api/mahasiswa untuk mengambil seluruh data mahasiswa, GET /api/mahasiswa/:index untuk menampilkan data berdasarkan posisi di array, POST /api/mahasiswa untuk menambah data mahasiswa baru dari input JSON, dan DELETE /api/mahasiswa/:index untuk menghapus data berdasarkan indeks array. Setiap endpoint dilengkapi dengan validasi dasar, seperti pengecekan keberadaan indeks atau validasi input yang kosong, dan mengembalikan respons dalam format JSON yang menjelaskan hasil dari operasi tersebut. API ini dapat diuji menggunakan Postman sesuai skenario yang diminta pada tugas.