Laporan Praktikum
Mata Kuliah Pemrograman Web

"REST API & JSON Web Token (JWT)"

Dosen Pengampu :

Willdan Aprizal Arifin, S.Pd., M.Kom.

Disusun Oleh :

Angga Ardiansyah

2307300/A3

PROGRAM STUDI SISTEM INFORMASI KELAUTAN

UNIVERSITAS PENDIDIKAN INDONESIA

2024

I. PENDAHULUAN

Pada era digital yang semakin maju, komunikasi antarsistem komputer telah menjadi infrastruktur teknologi informasi. Sebuah lanskap digital berbagai aplikasi, platform, dan layanan web perlu berkomunikasi secara cepat, dan efisien. Hal ini, titik awal menuju pemahaman mendalam tentang REST API.

REST API (Representational State Transfer Application Programming Interface) merupakan sebuah filosofi komunikasi dalam dunia digital. REST API ada karena kebutuhan akan standar komunikasi yang universal, ringan, dan dapat dimengerti oleh berbagai sistem yang berbeda. REST API memungkinkan berbagai "bahasa" komputer untuk saling berkomunikasi dengan protokol yang disepakati bersama.

JSON Web Token (JWT) muncul sebagai solusi inovatif yang menjawab tantangan kompleksitas keamanan digital. JSON Web Token merupakan merancang mandiri untuk mentransfer informasi secara aman antara berbagai pihak menggunakan format objek JSON. Sistem autentikasi kompleks, pertukaran data antar mikroservis, hingga implementasi Single Sign-On (SSO), JSON Web Token membuktikan diri sebagai teknologi yang adaptif. Kemampuannya untuk membawa informasi secara mandiri, ditambah dengan mekanisme penandatanganan yang canggih membuat para pengembang menjadikannya pilihan utama.

II. ALAT DAN BAHAN

1. Laptop
2. Visual Code Studio
3. Node js
4. XAMPP
5. express
6. MySQL
7. JWT (JSON Web Token)
8. bcryptjs
9. Postman

III. LANGKAH-LANGKAH

REST API

1. Buka XAMPP, lalu nyalakan MySQL dan Apahe
2. Buka Vs Code, kemudian pilih terminal install lakukkan instalasi dengan ketik "npm init-y" "npm install express mysql body parser"
3. Buat database

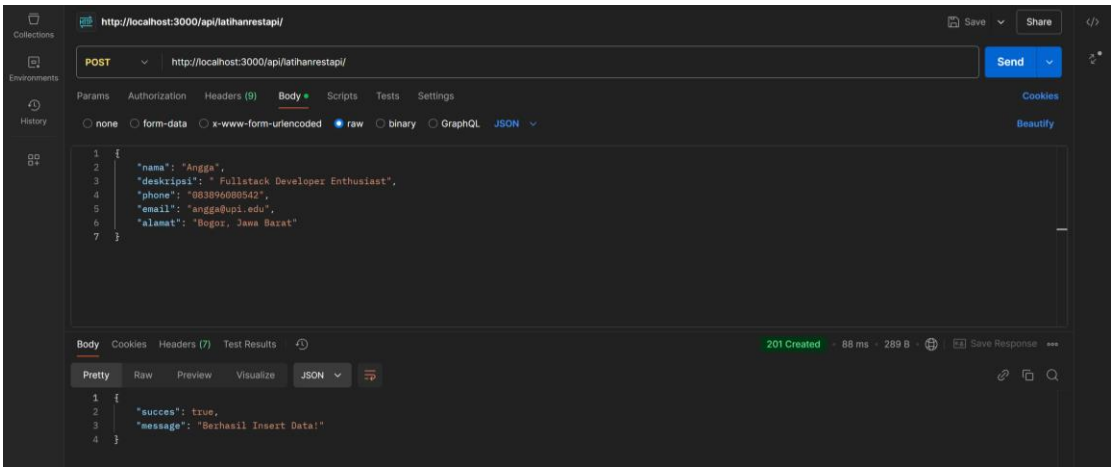
   CREATE DATABASE `latihanrestapi`; USE`latihanrestapi`; CREATE TABLE `latihanrestapi` (`id` int(11) NOT NULL PRIMARY KEY AUTO_INCREMENT, `nama` varchar(100) NOT NULL, `deskripsi` varchar(255) NOT NULL, `phone` varchar(20) NOT NULL, `email` varchar(50) NOT NULL, `alamat` varchar(100) NOT NULL);
4. Buat folder config, lalu buat file "db.js".
5. Buat file "app.js"
6. Buat kode create dalam file "app.js".

```
//create data
app.post('/api/latihanrestapi', (req, res) => {
    //buat variabel penampung data dan querry sql
    const data = {...req.body};
    const querySql = 'INSERT INTO latihanrestapi SET ?';

    // jalankan query
    koneksi.query(querySql, data, (err, rows, field) => {
        // error handling
        if (err) {
            return res.status(500).json({message: 'Gagal insert data!', error:err});
        }

        // jika request berhasil
        res.status(201).json ({ succes: true, message: 'Berhasil Insert Data!'});
    });
});
```

7. Buka postman, gunakan post untuk memastikan create sudah berjalan.

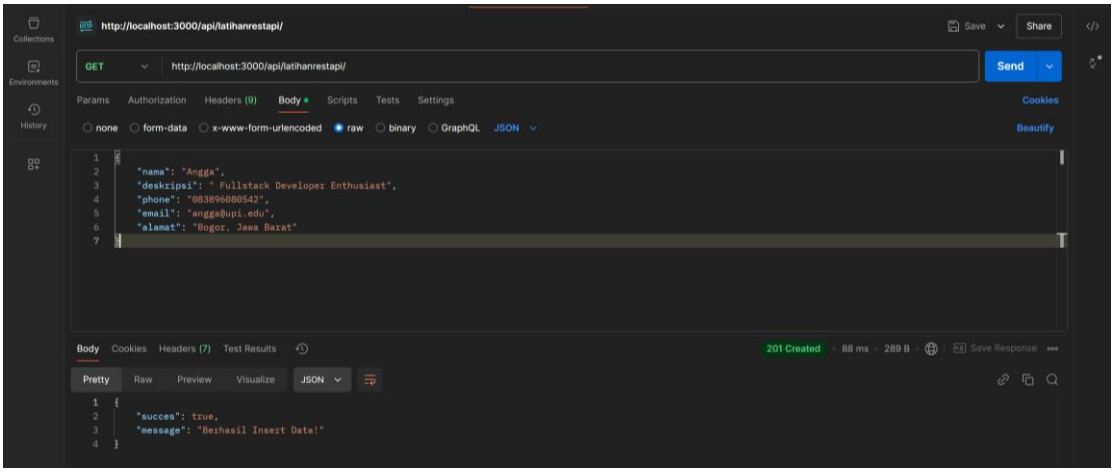


8. Buat kode read dalam file “app.js”.

```
// read data / get data
app.get('/api/latihanrestapi', (req, res) => {
    // buat query sql
 const querySql = 'SELECT * FROM latihanrestapi';
  // jalankan query
  koneksi.query(querySql, (err, rows, field)  => {
    // error handling
    if (err) {
        return res.status(500).json({ message: 'Ada kesalahan', error: err });
    }

    // jika request berhasil
    res.status(200).json({ success: true, data: rows });
  });
});
```

9. Buka postman, gunakan get untuk memastikan read sudah berjalan.



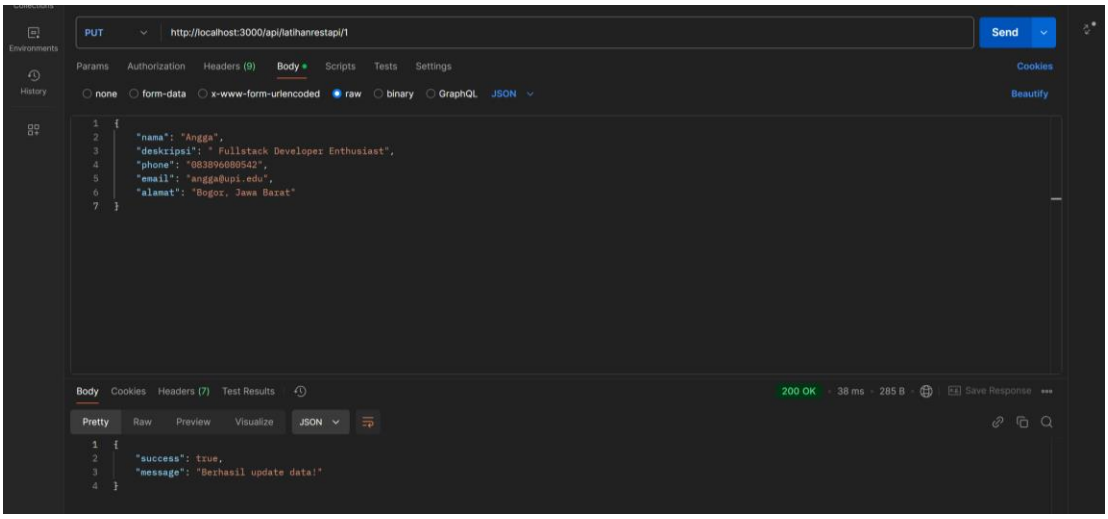
10. Buat kode update dalam file “app.js”.

```
 // update data
 app.put('/api/latihanrestapi/:id', (req, res) => {
```

```
    // buat variabel penampung data dan query sql
    const data = { ...req.body };
    const querySearch = 'SELECT * FROM latihanrestapi WHERE id = ?';
    const queryUpdate = 'UPDATE latihanrestapi SET ? WHERE id = ?';

     // jalankan query untuk melakukan pencarian data
     koneksi.query(querySearch, req.params.id, (err, rows, field) => {
        // error handling
        if (err) {
            return res.status(500).json({ message: 'Ada kesalahan', error: err });
        }
        // jika id yang dimasukkan sesuai dengan data yang ada di db
        if (rows.length) {
            // jalankan query update
            koneksi.query(queryUpdate, [data, req.params.id], (err, rows, field) => {
                // error handling
                if (err) {
                    return res.status(500).json({ message: 'Ada kesalahan', error: err });
                }
                // jika update berhasil
                res.status(200).json({ success: true, message: 'Berhasil update data!' });
            });
        } else {
            return res.status(404).json({ message: 'Data tidak ditemukan!', success: false });
        }
    });
});
```

11. Buka postman, gunakan put untuk memastikan update sudah berjalan.

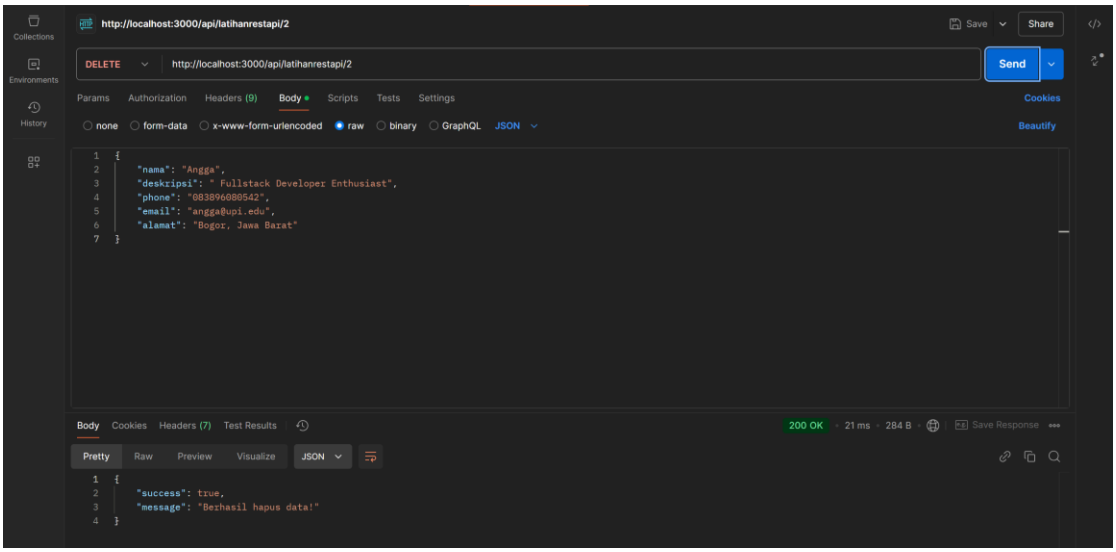


12. Buka kode delete dalam file “app.js”.

```
// delete data
app.delete('/api/latihanrestapi/:id', (req, res) => {
    // buat query sql untuk mencari data dan hapus
```

```
    const querySearch = 'SELECT * FROM latihanrestapi WHERE id = ?';
    const queryDelete = 'DELETE FROM latihanrestapi WHERE id = ?';
    // jalankan query untuk melakukan pencarian data
    koneksi.query(querySearch, req.params.id, (err, rows, field) => {
    // error handling
    if (err) {
    return res.status(500).json({ message: 'Ada kesalahan', error: err });
    }
  // jika id yang dimasukkan sesuai dengan data yang ada di db
  if (rows.length) {
    // jalankan query delete
    koneksi.query(queryDelete, req.params.id, (err, rows, field) => {
        // error handling
        if (err) {
            return res.status(500).json({ message: 'Ada kesalahan', error: err });
        }
        // jika delete berhasil
        res.status(200).json({ success: true, message: 'Berhasil hapus data!' });
    });
       } else {
    return res.status(404).json({ message: 'Data tidak ditemukan!', success: false });
       }
    });
});
```

13. Buka postman, gunakan delete untuk memastikan delete sudah berjalan.



API TOKEN

1. Pada terminal, ketik *npm install express body-parser mysql2 jsonwebtoken express-rate limit*.
2. Konfigurasi middleware dan server dengan menggunakan:

```
//set body parser
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({extended: false}));
```

3. Gunakan rate limit agar kestabilan sistem dapat terus berjalan dan melayani permintaan data.

```
const limiter = rateLimit({
        windowMs: 15 * 60 * 1000, // 15 menit
        max: 100 // batas 100 permintaan per IP
      });
```

4. Gunakan *access logger*

```
const accessLogger = (req, res, next) => {
  console.log(`Akses ke ${req.originalUrl} oleh ${req.ip}`);
  next();
};
```

5. *Middleware* dan *login*.

```
app.use(limiter);
app.use(accessLogger);
app.post("/api/login", (req, res) => {
    const { username, password } = req.body;

    // Validasi username dan password (ganti dengan logika autentikasi Anda)
    if (username === "admin" && password === "password") {
      const token = jwt.sign({ username }, secretKey, { expiresIn: "1h" });
      validTokens.add(token);
      return res.status(200).json({ token });
    }
    return res.status(401).json({ message: "Username atau password salah" });
  });
```

6. Validasi kembali token.

```
const authenticateToken = (req, res, next) => {
    const token = req.headers["authorization"]?.split(" ")[1];
    if (!token || revokedTokens.has(token)) return res.sendStatus(403);

    jwt.verify(token, secretKey, (err, user) => {
      if (err) return res.sendStatus(403);
      req.user = user;
      next();
    });
```

```
};
```

7. Akses data

```
app.get("/api/protected", authenticateToken, (req, res) => {
    res.status(200).json({ message: "Data yang dilindungi", user: req.user });
});
```

8. *Authentication* dan *revoke*(menghapus hak akses user MySQL)

```
//revoke
  app.post("/api/revoke", (req, res) => {
    const { token } = req.body;
    if (validTokens.has(token)) {
    validTokens.delete(token);
    revokedTokens.add(token);
    return res.status(200).json({ message: "Token berhasil direvoke" });
    }
    return res.status(400).json({ message: "Token tidak valid" });
    });
```

IV. KESIMPULAN

Berdasarkan praktikum yang telah dilaksanakan mengenai REST API, dapat disimpulkan bahwa mahasiswa mendapatkan pemahaman baru mengenai REST API. Melalui REST API mahasiswa dapat menghubungkan ide, sistem, dan inovasi. Selain itu, JSON Web Token (JWT) memberikan pemahaman tentang mekanisme autentikasi dan otorisasi modern dalam pengembangan aplikasi web. Melalui serangkaian eksplorasi dan implementasi, mahasiswa telah menerima kompleksitas dan keunggulan teknologi token berbasis JSON ini.