Laporan Praktikum

Mata Kuliah Pemprograman Beorientasi Objek

"Rest Api"

Dosen Pengampu :
Willdan Aprizal Arifin, S.Pd., M.Kom.

Disusun Oleh :

Gizza Auralia Pradnya. G

(2309292)

PROGRAM STUDI SISTEM INFORMASI KELAUTANUNIVERSITAS PENDIDIKAN INDONESIA

2024

PENDAHULUAN

Dalam era digital yang semakin terhubung, aplikasi dan layanan online menjadi bagian tak terpisahkan dari kehidupan kita. Di balik layar interaksi kita dengan berbagai aplikasi ini, terdapat suatu mekanisme yang memungkinkan mereka berkomunikasi dan bertukar data secara efisien. Salah satu mekanisme tersebut adalah REST API (Representational State Transfer Application Programming Interface). REST API adalah suatu standar arsitektur perangkat lunak yang memungkinkan aplikasi yang berbeda untuk berkomunikasi dan berinteraksi satu sama lain melalui internet.

Bayangkan REST API sebagai sebuah menu di restoran. Ketika Anda memesan makanan, Anda memberikan pesanan Anda kepada pelayan (klien), dan pelayan akan menyampaikan pesanan Anda ke dapur (server). Dapur kemudian akan menyiapkan makanan sesuai pesanan Anda dan memberikannya kembali kepada pelayan untuk disampaikan kepada Anda. Dalam konteks REST API, klien (misalnya, aplikasi mobile) mengirimkan permintaan ke server (misalnya, database), dan server akan merespon dengan data yang diminta. Proses ini terjadi secara real-time dan memungkinkan aplikasi untuk mendapatkan data yang dibutuhkan secara dinamis.

REST API telah menjadi salah satu teknologi inti dalam pengembangan perangkat lunak modern. Dengan menggunakan REST API, pengembang dapat membangun aplikasi yang lebih terintegrasi, skalabel, dan mudah dikembangkan. Selain itu, REST API juga memungkinkan perusahaan untuk membuka data dan layanan mereka kepada pihak ketiga melalui API marketplace, sehingga menciptakan peluang bisnis baru dan memperluas ekosistem digital.

ALAT DAN BAHAN

1. Laptop
2. VSCode
3. Mouse
4. Internet
5. Postman

Rest Api

1. (Config)Database.js

```
const mysql = require('mysql');
//buat configurasi ke koneksi
const koneksi=mysql.createConnection({
    host:'localhost',
    user:'root',
    password:'',
    database:'latihanrestapi'
});
//koneksi database
koneksi.connect((err)=>{
    if(err)throw err;
    console.log('MYSQL Connected...');
});
module.exports=koneksi
```

2. App.js

```
const express = require("express");
const bodyParser = require("body-parser");
const koneksi = require("./config/Database.js");
const app = express();
const PORT = process.env.PORT || 3000;

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

// Insert data
app.post("/api/latihanrestapi", (req, res) => {
  const data = { ...req.body };
  const querySql = "INSERT INTO latihanrestapi SET ?";

  koneksi.query(querySql, data, (err, rows) => {
    if (err) {
      return res
        .status(500)
        .json({ message: "GAGAL Insert data!", error: err });
    }
    res.status(201).json({ success: true, message: "Berhasil insert data" });
  });
});

// Read data / get data
app.get("/api/latihanrestapi", (req, res) => {
  const querySql = "SELECT * FROM latihanrestapi";
```

```
  koneksi.query(querySql, (err, rows) => {
    if (err) {
      return res.status(500).json({ message: "Ada kesalahan", error: err });
    }
    res.status(200).json({ success: true, data: rows });
  });
});

// Update data
app.put("/api/latihanrestapi/:id", (req, res) => {
  const data = { ...req.body };
  const querySearch = "SELECT * FROM latihanrestapi WHERE id = ?";
  const queryUpdate = "UPDATE latihanrestapi SET ? WHERE id = ?";

  koneksi.query(querySearch, req.params.id, (err, rows) => {
    if (err) {
      return res.status(500).json({ message: "Ada kesalahan", error: err });
    }

    // Pastikan data ditemukan
    if (rows.length) {
      koneksi.query(queryUpdate, [data, req.params.id], (err) => {
        if (err) {
          return res.status(500).json({ message: "Ada kesalahan", error: err });
        }
        res
          .status(200)
          .json({ success: true, message: "Berhasil update data!" });
      });
    } else {
      return res
        .status(404)
        .json({ message: "Data tidak ditemukan!", success: false });
    }
  });
});

//delete data
app.delete("/api/latihanrestapi/:id", (req, res) => {
  //buat query sql untuk mencari data dan hapus
  const querySeacrh = "SELECT * FROM latihanrestapi WHERE id = ? ";
  const queryDelete = "DELETE FROM latihanrestapi WHERE id = ?";

  // jalankan query untuk melakukan pencarian data
  koneksi.query(querySeacrh, 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({ succes: true, message: "Berhasil hapus data!" });
      });
    } else {
      return res
        .status(404)
        .json({ message: "Data tidak ditemukan!", succes: false });
    }
  });
});

// Jalankan server
app.listen(PORT, () => console.log(`Server running at port: ${PORT}`));
```

Rest Api Json Web Token

Database.js

```
const mysql = require('mysql');
//buat configurasi ke koneksi
const koneksi=mysql.createConnection({
    host:'localhost',
    user:'root',
    password:'',
    database:'latihanrestapi'
});
//koneksi database
koneksi.connect((err)=>{
    if(err)throw err;
    console.log('MYSQL Connected...');
});
module.exports=koneksi
```

app.js

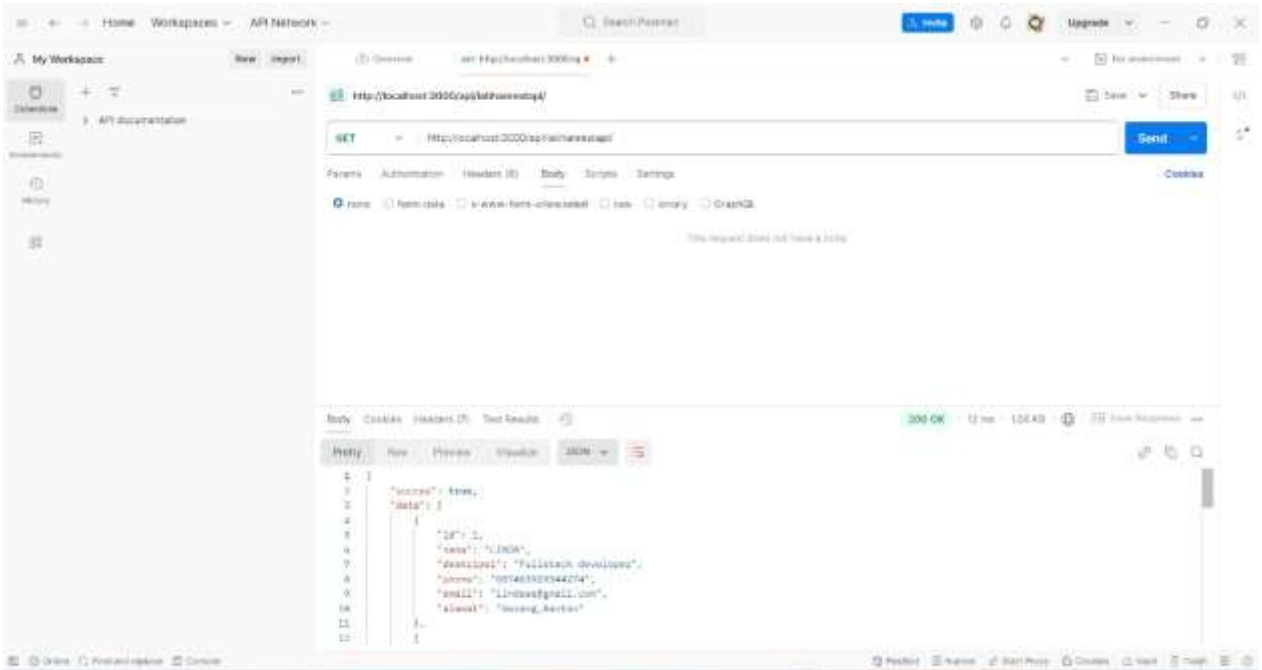
```
const express = require("express");
const bodyParser = require("body-parser");
const koneksi = require("./config/Database.js");
const app = express();
const PORT = process.env.PORT || 3000;
const jwt = require("jsonwebtoken");
const rateLimit = require("express-rate-limit");
const secretKey = "your_secret_key"; // Ganti dengan kunci rahasia yang lebih
aman
const validTokens = new Set(); // Menyimpan token yang valid
const revokedTokens = new Set(); // Menyimpan token yang tidak berlaku
```

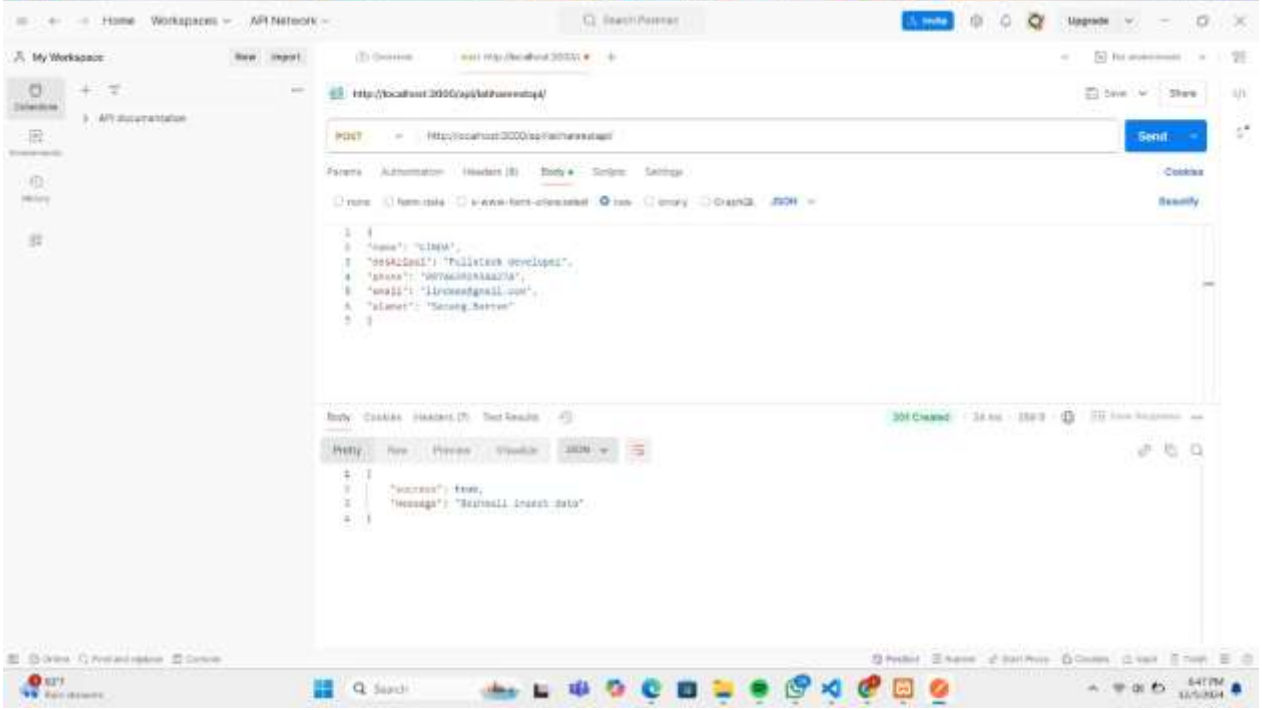
```
// Set body parser
app.use(bodyParser.json());
app.use(bodyParser.urlencoded({ extended: false }));

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

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

// Gunakan middleware
app.use(limiter);
app.use(accessLogger);

// Endpoint untuk login dan mendapatkan token
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" });
});

// Middleware untuk memvalidasi 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();
  });
};

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

// Endpoint untuk merevoke token
app.post("/api/revoke", (req, res) => {
```

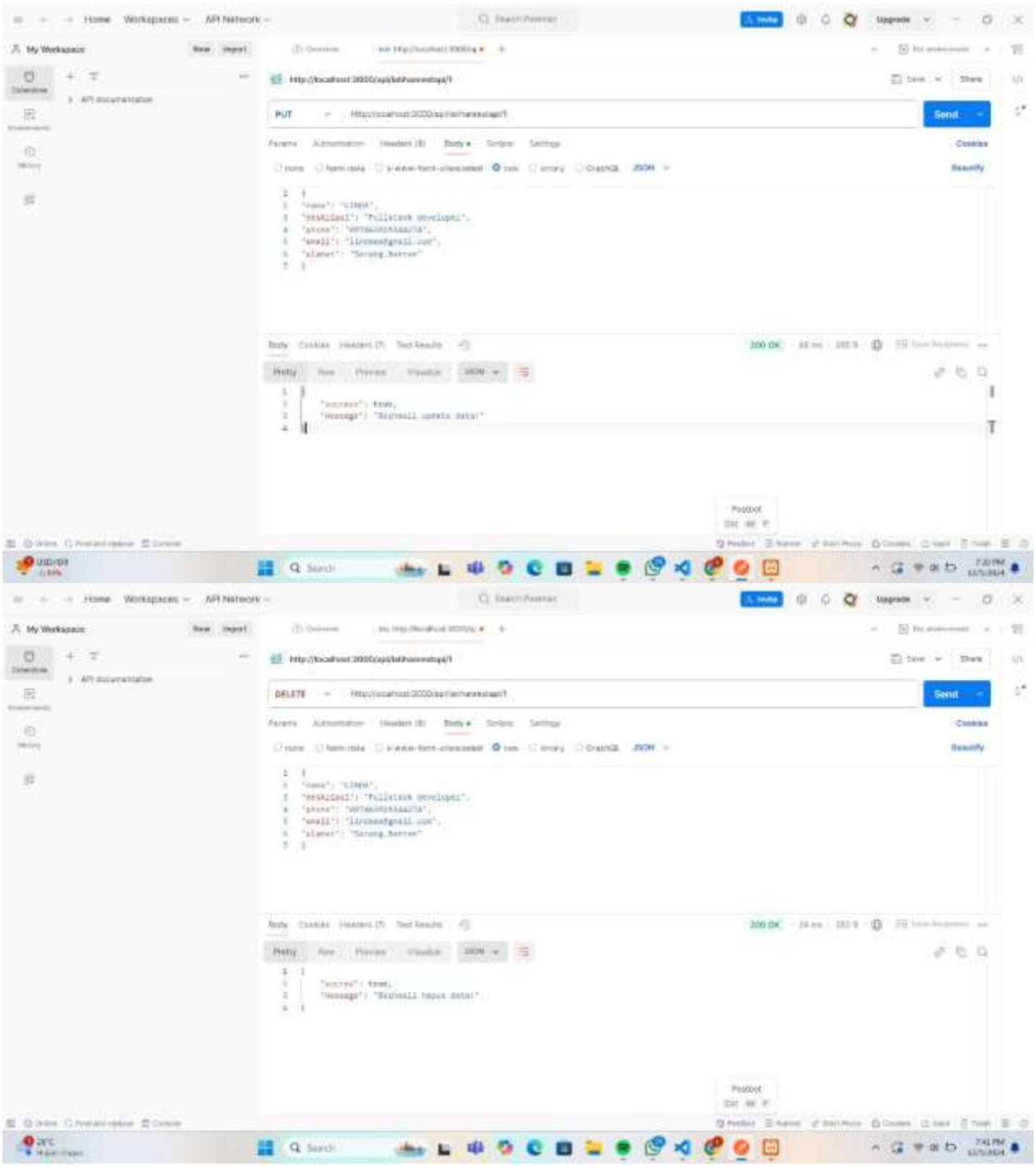
```
  const { token } = req.body;
  if (validTokens.has(token)) {
    validTokens.delete(token);
    revokedTokens.add(token);
    return res.status(200).json({ message: "Token berhasil direvoke" });
  }
  return res.status(400).json({ message: "Token tidak valid" });
});

// Insert data
app.post("/api/latihanrestapi", (req, res) => {
  const data = { ...req.body };
  const querySql = "INSERT INTO latihanrestapi SET ?";

  koneksi.query(querySql, data, (err) => {
    if (err) {
      return res
        .status(500)
        .json({ message: "GAGAL Insert data!", error: err });
    }
    res.status(201).json({ success: true, message: "Berhasil insert data" });
  });
});

// Read data / get data
app.get("/api/latihanrestapi", (req, res) => {
  const querySql = "SELECT * FROM latihanrestapi";

  koneksi.query(querySql, (err, rows) => {
    if (err) {
      return res.status(500).json({ message: "Ada kesalahan", error: err });
    }
    res.status(200).json({ success: true, data: rows });
  });
});

// Update data
app.put("/api/latihanrestapi/:id", (req, res) => {
  const data = { ...req.body };
  const querySearch = "SELECT * FROM latihanrestapi WHERE id = ?";
  const queryUpdate = "UPDATE latihanrestapi SET ? WHERE id = ?";

  koneksi.query(querySearch, req.params.id, (err, rows) => {
    if (err) {
      return res.status(500).json({ message: "Ada kesalahan", error: err });
    }

    // Pastikan data ditemukan
    if (rows.length) {
      koneksi.query(queryUpdate, [data, req.params.id], (err) => {
        if (err) {
          return res.status(500).json({ message: "Ada kesalahan", error: err });
```

```
        }
        res
          .status(200)
          .json({ success: true, message: "Berhasil update data!" });
      });
    } else {
      return res
        .status(404)
        .json({ message: "Data tidak ditemukan!", success: false });
    }
  });
});

// Delete data
app.delete("/api/latihanrestapi/:id", (req, res) => {
  const querySearch = "SELECT * FROM latihanrestapi WHERE id = ?";
  const queryDelete = "DELETE FROM latihanrestapi WHERE id = ?";

  koneksi.query(querySearch, req.params.id, (err, rows) => {
    if (err) {
      return res.status(500).json({ message: "Ada kesalahan", error: err });
    }
    if (rows.length) {
      koneksi.query(queryDelete, req.params.id, (err) => {
        if (err) {
          return res.status(500).json({ message: "Ada kesalahan", error: err });
        }
        res.status(200).json({ success: true, message: "Berhasil hapus data!" });
      });
    } else {
      return res
        .status(404)
        .json({ message: "Data tidak ditemukan!", success: false });
    }
  });
});

// Jalankan server
app.listen(PORT, () => console.log(`Server running at port: ${PORT}`));
```

Postman







KESIMPULAN

REST API telah menjadi tulang punggung dalam pengembangan aplikasi modern. Dengan prinsip-prinsip arsitektur yang sederhana namun efektif, REST API memungkinkan aplikasi yang berbeda untuk berkomunikasi dan bertukar data secara efisien dan fleksibel. Kemampuannya untuk menggunakan metode HTTP standar dan format data yang umum seperti JSON membuatnya menjadi pilihan populer di kalangan pengembang.