Laporan Modul Praktik

Membuat REST API : Perancangan Sistem Toko Online menggunakan Node.js

Diajukan Untuk Memenuhi Salah Satu Tugas Project UAS Mata Kuliah Pemograman Berbasis Platfrom

Dosen Pengampu : Muhammad Ikhsan Thohir, M.Kom

Disusun Oleh Kelompok 2:

Anisa Cikal Virgifiani	20230040261
Rassya Ramadhani Priadi	20230040142
Siti Zahra Sifa	20230040212

PROGRAM STUDI S1 TEKNIK INFORMATIKA

FAKULTAS KOMPUTER DAN DESAIN

NUSA PUTRA UNIVERSITY

2024/2025

1. Pendahuluan

Toko online adalah platform digital yang memungkinkan pengguna untuk menjual dan membeli produk secara daring. Sistem ini mencakup beberapa fitur inti seperti manajemen pengguna, produk, pesanan, dan detail pesanan. Dalam laporan ini, akan dijelaskan perancangan sistem database beserta endpoint API untuk mendukung operasional toko online.

2. Kajian Teori

Dalam pengembangan sistem toko online berbasis REST API, laporan ini mendasarkan penerapannya pada beberapa teori dan konsep teknologi modern sebagai berikut:

1. Node.js dan Express.js

   Node.js adalah *runtime environment* berbasis JavaScript yang digunakan untuk pengembangan aplikasi *server-side*. Teknologi ini unggul dalam performa karena sifatnya yang *non-blocking* dan berbasis event-driven. Express.js, sebagai *framework* minimalis untuk Node.js, mempermudah pengelolaan *routing*, *middleware*, dan logika aplikasi, sehingga menjadi pilihan ideal untuk membangun REST API.

2. MySQL sebagai Database Relasional

   MySQL digunakan sebagai sistem manajemen basis data untuk mendukung kebutuhan penyimpanan dan pengelolaan data. Teori database relasional mengedepankan struktur data dalam bentuk tabel dengan relasi kunci asing (*foreign key*), seperti yang diterapkan dalam desain tabel Pengguna, Produk, Pesanan, dan Detail Pesanan.

3. REST API

   REST (Representational State Transfer) adalah arsitektur standar yang digunakan untuk membangun layanan web. REST API mendefinisikan serangkaian aturan untuk komunikasi antara klien dan server menggunakan metode HTTP, seperti GET, POST,

PUT, dan DELETE, yang diterapkan pada pengelolaan data dalam laporan ini.

4. Autentikasi dan Keamanan dengan JWT (JsonWebToken)

   JWT adalah metode yang digunakan untuk mengamankan akses terhadap sistem melalui token yang dienkripsi. Konsep ini mendukung validasi identitas pengguna dengan aman dan efisien tanpa perlu terus-menerus memeriksa basis data selama sesi berlangsung.

5. Pengujian API menggunakan Postman

   Postman adalah alat pengujian API yang membantu memvalidasi endpoint dan fungsi REST API. Pengujian ini didasarkan pada konsep validasi data, memastikan bahwa setiap respons dan logika berjalan sesuai dengan yang dirancang.

3. Landasan Teori dengan Implementasi

Kajian teori ini mendukung pendekatan sistematis dalam perancangan sistem toko online. Node.js dan Express.js digunakan untuk membangun server REST API yang cepat dan responsif. MySQL memastikan penyimpanan data yang terstruktur dan aman. Dengan menerapkan JWT, laporan ini memastikan keamanan autentikasi pengguna. Pengujian API melalui Postman menguatkan keandalan sistem yang dirancang. Melalui kombinasi teori dan praktik ini, sistem toko online yang dirancang menunjukkan penerapan teknologi berbasis platform yang relevan dan terkini.

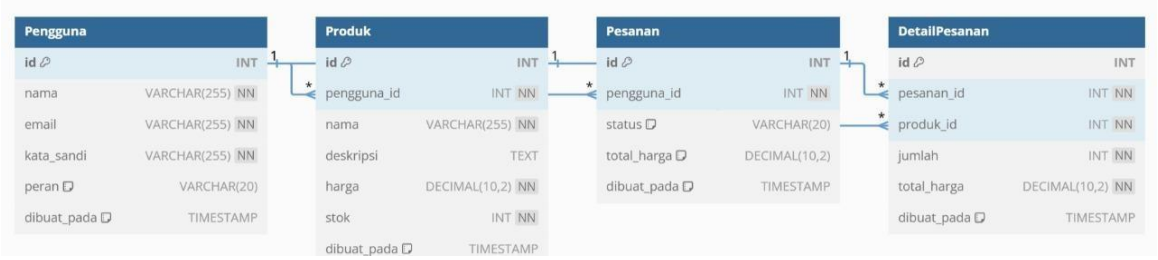
4. Analisis Kebutuhan

- **Alat dan Teknologi**
  - Backend: Node.js, Express.js
  - Database: MySQL
  - Pengujian Endpoint: Postman
  - Autentikasi: JWT untuk validasi pengguna

- **Fitur Utama**
  - Autentikasi Pengguna (Registrasi dan Login)
  - Manajemen Pengguna
  - Manajemen Produk
  - Manajemen Pesanan
  - Manajemen Detail Pesanan

5. Rancangan Database

Database dirancang dengan beberapa table utama: Pengguna, Produk, Pesanan, dan Detai pesanan. Berikut struktur database:



6. Skema Database Toko Online

```
-- Membuat database

CREATE DATABASE TokoOnline;


-- Menggunakan database

USE TokoOnline;


-- 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('admin', 'pelanggan') DEFAULT 'pelanggan',
```

```
    dibuat_pada TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabel Produk
CREATE TABLE Produk (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nama VARCHAR(255) NOT NULL,
    deskripsi TEXT,
    harga DECIMAL(10, 2) NOT NULL,
    stok INT NOT NULL,
    dibuat_pada TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);

-- Tabel Pesanan
CREATE TABLE Pesanan (
    id INT AUTO_INCREMENT PRIMARY KEY,
    pengguna_id INT NOT NULL,
    status ENUM('tertunda', 'selesai', 'dibatalkan') DEFAULT 'tertunda',
    total_harga DECIMAL(10, 2) DEFAULT 0.00,
    dibuat_pada TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (pengguna_id) REFERENCES Pengguna(id)
);

-- Tabel DetailPesanan
CREATE TABLE DetailPesanan (
    id INT AUTO_INCREMENT PRIMARY KEY,
    pesanan_id INT NOT NULL,
```

```
    produk_id INT NOT NULL,
    jumlah INT NOT NULL,
    dibuat_pada TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (pesanan_id) REFERENCES Pesanan(id),
    FOREIGN KEY (produk_id) REFERENCES Produk(id)
);
```

7. Endpoint yang dipakai

1. Autentikasi Pengguna

   POST /api/pengguna/register - Registrasi pengguna.

   POST /api/pengguna/login - Login pengguna.

2. Manajemen Pengguna

   GET /api/pengguna - Mendapatkan semua pengguna.

   GET /api/pengguna/:id - Mendapatkan detail pengguna berdasarkan ID.

   PUT /api/pengguna/:id - Memperbarui data pengguna.

   DELETE /api/pengguna/:id - Menghapus pengguna.

3. Manajemen Produk

   POST /api/produk - Menambahkan produk baru.

   GET /api/produk - Mendapatkan daftar semua produk.

   GET /api/produk/:id - Mendapatkan detail produk berdasarkan ID.

   PUT /api/produk/:id - Memperbarui data produk.

   DELETE /api/produk/:id - Menghapus produk.

4. Manajemen Pesanan

   POST /api/pesanan - Membuat pesanan baru.

   GET /api/pesanan - Mendapatkan semua pesanan.

   GET /api/pesanan/:id - Mendapatkan detail pesanan berdasarkan ID.

   PUT /api/pesanan/:id - Memperbarui status pesanan.

   DELETE /api/pesanan/:id - Menghapus pesanan.

5. Manajemen Detail Pesanan

POST /api/detailpesanan - Menambahkan item ke dalam pesanan.

GET /api/detailpesanan/:pesanan_id - Mendapatkan semua item dalam pesanan.

PUT /api/detailpesanan/:id - Memperbarui jumlah item dalam pesanan.

DELETE /api/detailpesanan/:id - Menghapus item dalam pesanan.

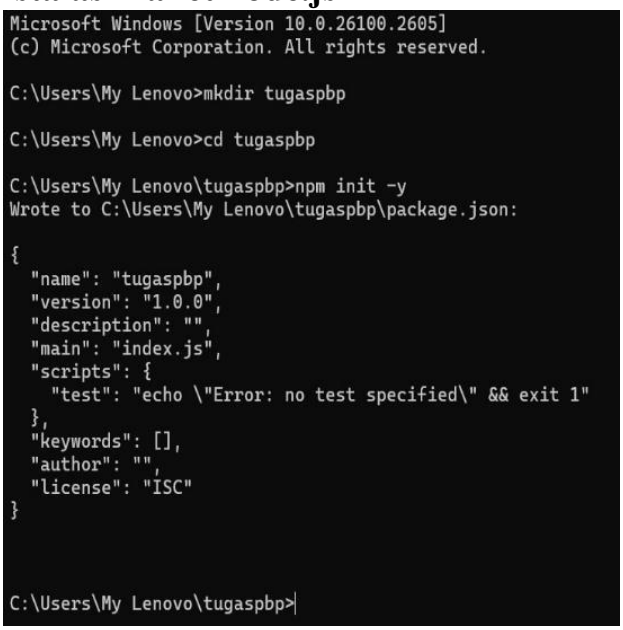
8. Data dan Analisis

Pada analisis kebutuhan, laporan merinci fitur inti, seperti autentikasi pengguna, manajemen produk, pesanan, dan detail pesanan. Struktur database dirancang sistematis untuk mendukung hubungan antar data menggunakan tabel utama seperti Pengguna, Produk, Pesanan, dan Detail Pesanan. Proses implementasi memanfaatkan metodologi pengujian API menggunakan Postman untuk menjamin keakuratan dan keandalan sistem.

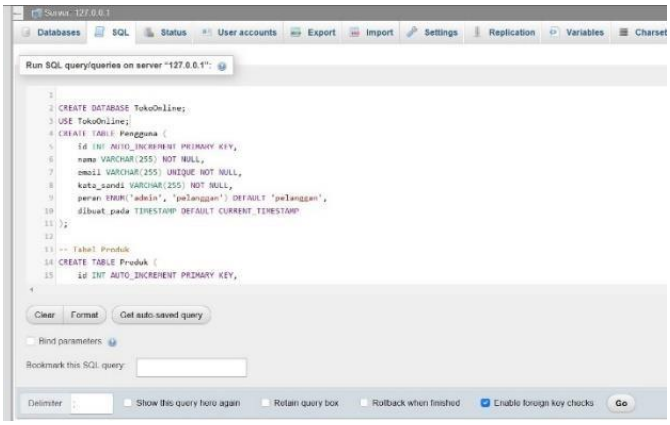
Pengujian yang dilakukan membuktikan bahwa setiap fitur, termasuk operasi CRUD (Create, Read, Update, Delete), bekerja dengan baik dalam lingkungan lokal. Dengan demikian, laporan ini tidak hanya mendokumentasikan tahap perancangan tetapi juga menyoroti keberhasilan implementasi sistem secara menyeluruh.

9. Implementasi

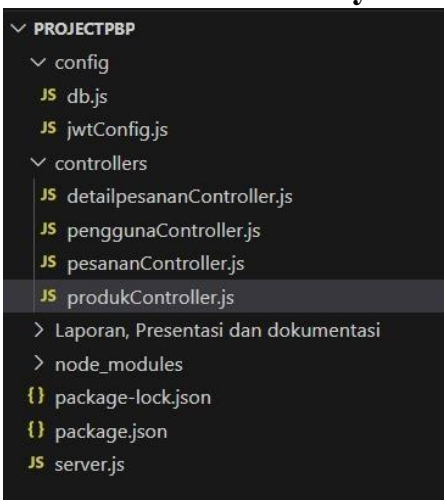
- ### Instalasi Paket node.js



- ## Membuat Database TokoOnline



- ## Struktur Direktori Proyek



- ## 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: 'tokoonline'
});

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

module.exports = connection;
```

- ## Setup Server

  Gunakan server.js untuk membuka local host

```
const express = require('express');
const penggunaController = require('./controllers/penggunaController');
const produkController = require('./controllers/produkController');
const pesananController = require('./controllers/pesananController');
const detailpesananController = require('./controllers/detailpesananController');
require("dotenv").config();

const app = express();


app.use(express.json());
app.use(express.urlencoded({ extended: true }));


app.use('/api', penggunaController);
app.use('/api', produkController);
app.use('/api', pesananController);
app.use('/api', detailpesananController);


app.use((err, req, res, next) => {
  console.error(err.stack);
  res.status(err.status || 500).json({
    message: err.message || 'Terjadi kesalahan pada server',
  });
});


const PORT = process.env.PORT || 3000;
app.listen(PORT, () => {
  console.log(`Server berjalan di http://localhost:${PORT}`);
});

```

- ## Mengatur kunci rahasia dan durasi validitas token JWT

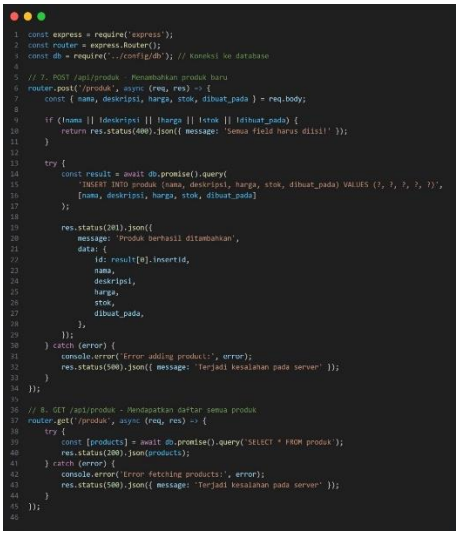
```
module.exports = {
    secret: 'your_jwt_secret_key', // Ganti dengan kunci rahasia Anda
    expiresIn: '1h',               // Token valid selama 1 jam
  };
```

- ## Membuat Route dan Operasi CRUD

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

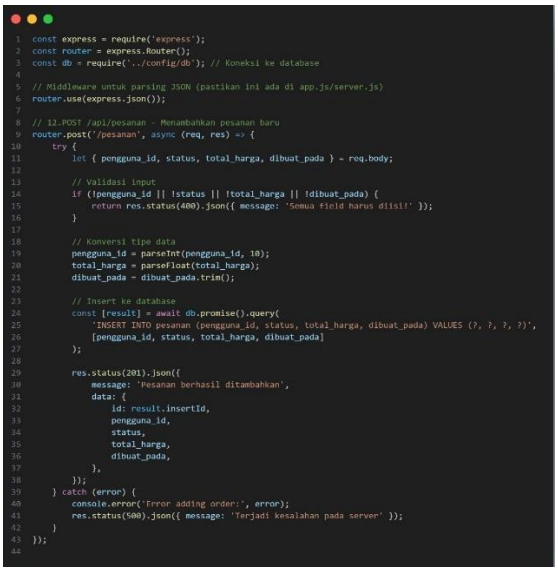
```
const express = require('express');
const router = express.Router();
const db = require('../config/db'); // Koneksi ke database

// 7. POST /api/produk - Menambahkan produk baru
router.post('/produk', async (req, res) => {
    const { nama, deskripsi, harga, stok, dibuat_pada } = req.body;

    if (!nama || !deskripsi || !harga || !stok || !dibuat_pada) {
        return res.status(400).json({ message: 'Semua field harus diisi!' });
    }

    try {
        const result = await db.promise().query(
            'INSERT INTO produk (nama, deskripsi, harga, stok, dibuat_pada) VALUES (?, ?, ?, ?, ?)',
            [nama, deskripsi, harga, stok, dibuat_pada]
        );

        res.status(201).json({
            message: 'Produk berhasil ditambahkan',
            data: {
                id: result[0].insertId,
                nama,
                deskripsi,
                harga,
                stok,
                dibuat_pada,
            },
        });
    } catch (error) {
        console.error('Error adding product:', error);
        res.status(500).json({ message: 'Terjadi kesalahan pada server' });
    }
});

// 8. GET /api/produk - Mendapatkan daftar semua produk
router.get('/produk', async (req, res) => {
    try {
        const [products] = await db.promise().query('SELECT * FROM produk');
        res.status(200).json(products);
    } catch (error) {
        console.error('Error fetching products:', error);
        res.status(500).json({ message: 'Terjadi kesalahan pada server' });
    }
});

```

Routes dan operasi CRUD pada produkController.js dan menggunakan validasi data JsonWebToken



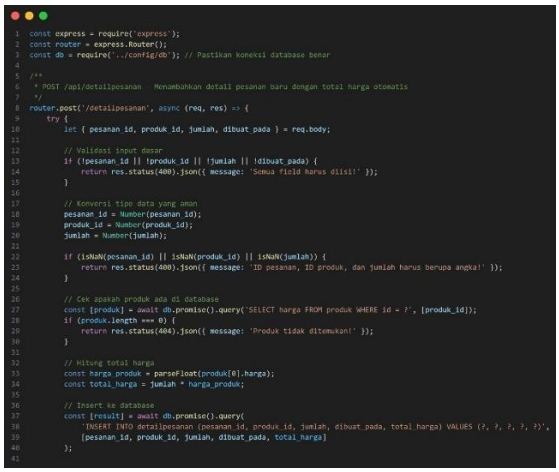
```
const express = require('express');
const router = express.Router();
const db = require('../config/db'); // Koneksi ke database

// 7. POST /api/produk - Menambahkan produk baru
router.post('/produk', async (req, res) => {
    const { nama, deskripsi, harga, stok, dibuat_pada } = req.body;

    if (!nama || !deskripsi || !harga || !stok || !dibuat_pada) {
        return res.status(400).json({ message: 'Semua field harus diisi!' });
    }

    try {
        const result = await db.promise().query(
            'INSERT INTO produk (nama, deskripsi, harga, stok, dibuat_pada) VALUES (?, ?, ?, ?, ?)',
            [nama, deskripsi, harga, stok, dibuat_pada]
        );

        res.status(201).json({
            message: 'Produk berhasil ditambahkan',
            data: {
                id: result[0].insertId,
                nama,
                deskripsi,
                harga,
                stok,
                dibuat_pada,
            },
        });
    } catch (error) {
        console.error('Error adding product:', error);
        res.status(500).json({ message: 'Terjadi kesalahan pada server' });
    }
});

// 8. GET /api/produk - Mendapatkan daftar semua produk
router.get('/produk', async (req, res) => {
    try {
        const [products] = await db.promise().query('SELECT * FROM produk');
        res.status(200).json(products);
    } catch (error) {
        console.error('Error fetching products:', error);
        res.status(500).json({ message: 'Terjadi kesalahan pada server' });
    }
});

```

Routes dan operasi CRUD pada pesananController.js dan menggunakan validasi data JsonWebToken



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

// Middleware untuk parsing JSON (pastikan ini ada di app.js/server.js)
router.use(express.json());

// 12.POST /api/pesanan - Menambahkan pesanan baru
router.post('/pesanan', async (req, res) => {
    try {
        let { pengguna_id, status, total_harga, dibuat_pada } = req.body;

        // Validasi input
        if (!pengguna_id || !status || !total_harga || !dibuat_pada) {
            return res.status(400).json({ message: 'Semua field harus diisi!' });
        }

        // Konversi tipe data
        pengguna_id = parseInt(pengguna_id, 10);
        total_harga = parseFloat(total_harga);
        dibuat_pada = dibuat_pada.trim();

        // Insert ke database
        const [result] = await db.promise().query(
            'INSERT INTO pesanan (pengguna_id, status, total_harga, dibuat_pada) VALUES (?, ?, ?, ?)',
            [pengguna_id, status, total_harga, dibuat_pada]
        );

        res.status(201).json({
            message: 'Pesanan berhasil ditambahkan',
            data: {
                id: result.insertId,
                pengguna_id,
                status,
                total_harga,
                dibuat_pada,
            },
        });
    } catch (error) {
        console.error('Error adding order:', error);
        res.status(500).json({ message: 'Terjadi kesalahan pada server' });
    }
});

```

Routes dan operasi CRUD pada detailpesananController.js dan menggunakan validasi data JsonWebToken

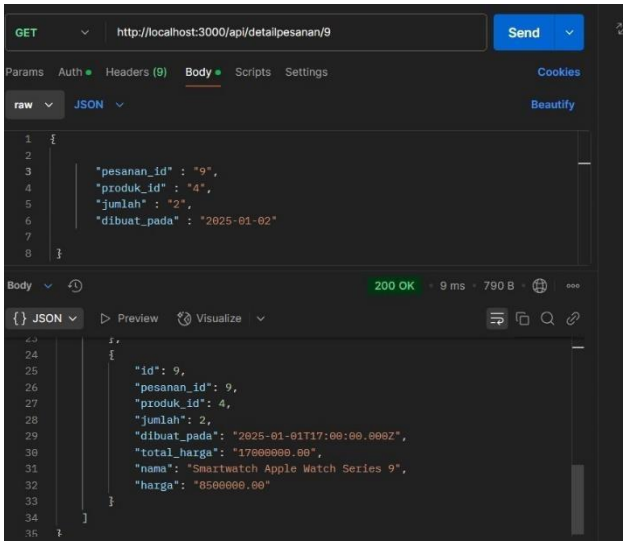


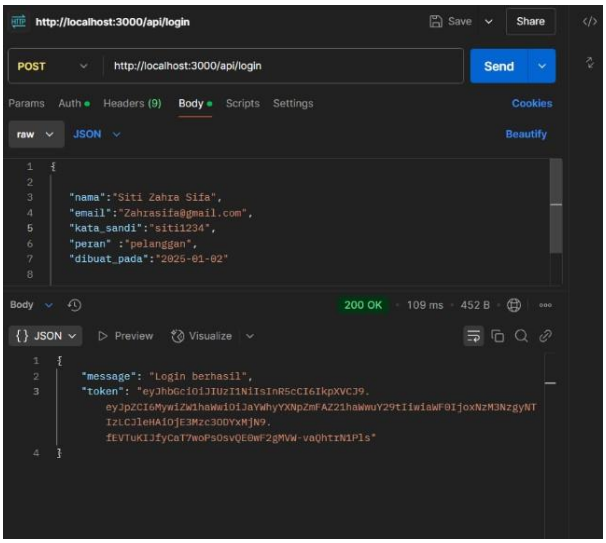
```
const express = require('express');
const router = express.Router();
const db = require('../config/db'); // Pastikan koneksi database benar

/**
 * POST /api/detailpesanan  Menambahkan detail pesanan baru dengan total harga otomatis
 */
router.post('/detailpesanan', async (req, res) => {
    try {
        let { pesanan_id, produk_id, jumlah, dibuat_pada } = req.body;

        // Validasi input dasar
        if (!pesanan_id || !produk_id || !jumlah || !dibuat_pada) {
            return res.status(400).json({ message: 'Semua field harus diisi!' });
        }

        // Konversi tipe data yang aman
        pesanan_id = Number(pesanan_id);
        produk_id = Number(produk_id);
        jumlah = Number(jumlah);

        if (isNaN(pesanan_id) || isNaN(produk_id) || isNaN(jumlah)) {
            return res.status(400).json({ message: 'ID pesanan, ID produk, dan jumlah harus berupa angka!' });
        }

        // Cek apakah produk ada di database
        const [produk] = await db.promise().query('SELECT harga FROM produk WHERE id = ?', [produk_id]);
        if (produk.length === 0) {
            return res.status(404).json({ message: 'Produk tidak ditemukan!' });
        }

        // Hitung total harga
        const harga_produk = parseFloat(produk[0].harga);
        const total_harga = jumlah * harga_produk;

        // Insert ke database
        const [result] = await db.promise().query(
            'INSERT INTO detailpesanan (pesanan_id, produk_id, jumlah, dibuat_pada, total_harga) VALUES (?, ?, ?, ?, ?)',
            [pesanan_id, produk_id, jumlah, dibuat_pada, total_harga]
        );
```

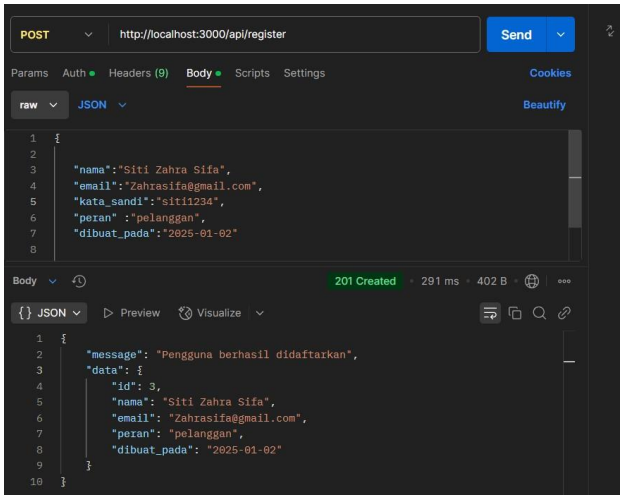
Full code: https://github.com/SitiZahraSifa11/Project_PBP_Kelompok2_TI23F

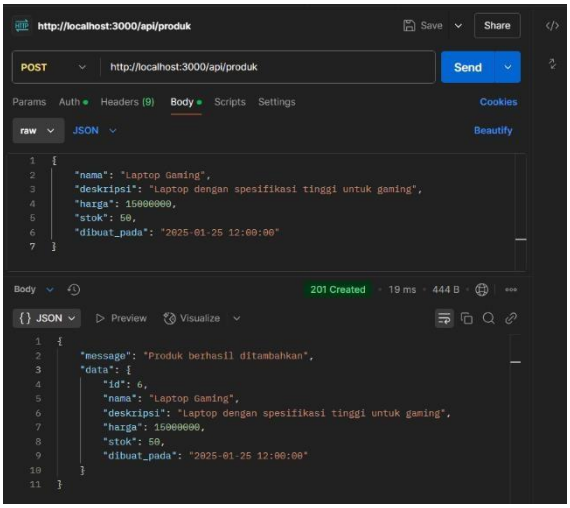
- ## Pengujian API

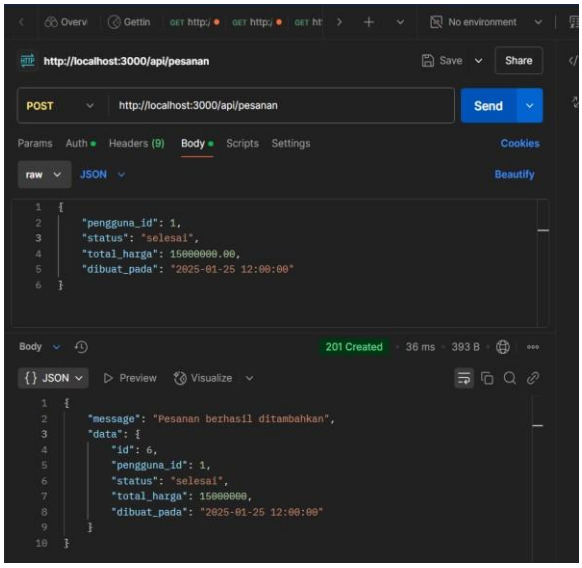
  Pengujian digunakan menggunakan Postman

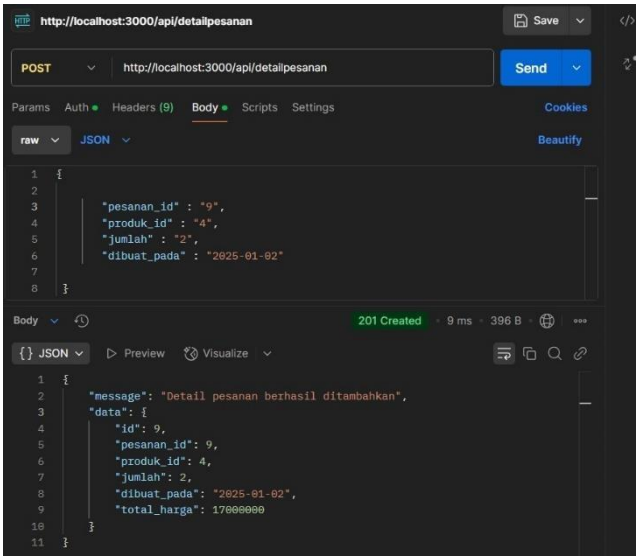


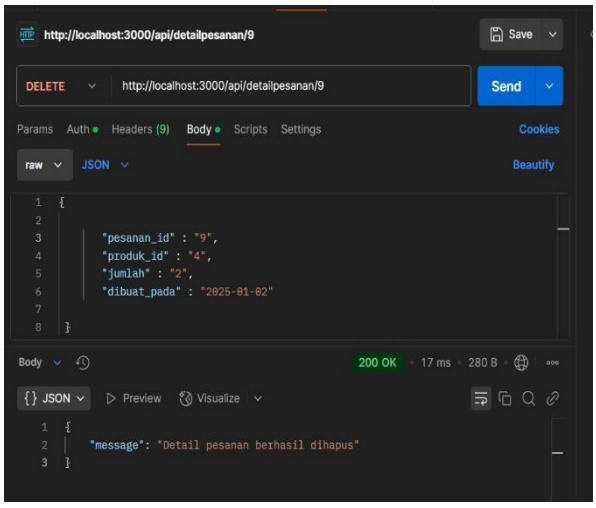


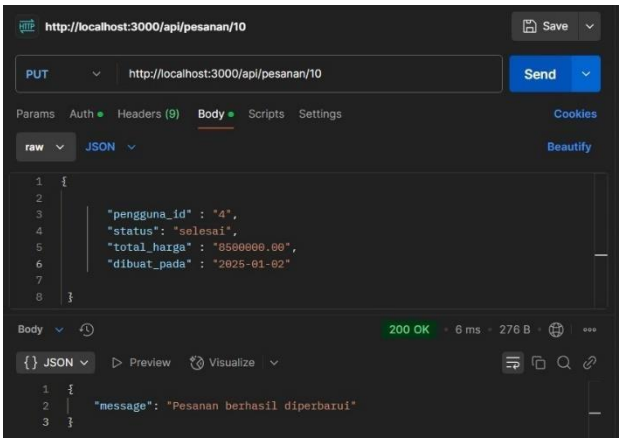

POST
http://localhost:3000/api/register
Send
Params Auth Headers (9) Body Scripts Settings
Cookies
raw JSON
Beautify
{
"nama":"Siti Zahra Sifa",
"email":"Zahrasifa@gmail.com",
"kata_sandi":"siti1234",
"peran" :"pelanggan",
"dibuat_pada":"2025-01-02"
}
Body
201 Created
291 ms
402 B
{ } JSON
Preview
Visualize
{
"message": "Pengguna berhasil didaftarkan",
"data": {
"id": 3,
"nama": "Siti Zahra Sifa",
"email": "Zahrasifa@gmail.com",
"peran": "pelanggan",
"dibuat_pada": "2025-01-02"
}
}


http://localhost:3000/api/produk
Save
Share
POST
http://localhost:3000/api/produk
Send
Params Auth Headers (9) Body Scripts Settings
Cookies
raw JSON
Beautify
{
"name": "Laptop Gaming",
"deskripsi": "Laptop dengan spesifikasi tinggi untuk gaming",
"harga": 15000000,
"stok": 50,
"dibuat_pada": "2025-01-25 12:00:00"
}
Body
201 Created
19 ms
444 B
{ } JSON
Preview
Visualize
{
"message": "Produk berhasil ditambahkan",
"data": {
"id": 6,
"nama": "Laptop Gaming",
"deskripsi": "Laptop dengan spesifikasi tinggi untuk gaming",
"harga": 15000000,
"stok": 50,
"dibuat_pada": "2025-01-25 12:00:00"
}
}


No environment
http://localhost:3000/api/pesanan
Save
Share
POST
http://localhost:3000/api/pesanan
Send
Params Auth Headers (9) Body Scripts Settings
Cookies
raw JSON
Beautify
{
"pengguna_id": 1,
"status": "selesai",
"total_harga": 15000000.00,
"dibuat_pada": "2025-01-25 12:00:00"
}
Body
201 Created
36 ms
393 B
{ } JSON
Preview
Visualize
{
"message": "Pesanan berhasil ditambahkan",
"data": {
"id": 6,
"pengguna_id": 1,
"status": "selesai",
"total_harga": 15000000,
"dibuat_pada": "2025-01-25 12:00:00"
}
}


http://localhost:3000/api/detailpesanan
Save
POST
http://localhost:3000/api/detailpesanan
Send
Params Auth Headers (9) Body Scripts Settings
Cookies
raw JSON
Beautify
{
"pesanan_id" : "9",
"produk_id" : "4",
"jumlah" : "2",
"dibuat_pada" : "2025-01-02"
}
Body
201 Created
9 ms
396 B
{ } JSON
Preview
Visualize
{
"message": "Detail pesanan berhasil ditambahkan",
"data": {
"id": 9,
"pesanan_id": 9,
"produk_id": 4,
"jumlah": 2,
"dibuat_pada": "2025-01-02",
"total_harga": 17000000
}
}


http://localhost:3000/api/detailpesanan/9
Save
DELETE
http://localhost:3000/api/detailpesanan/9
Send
Params Auth Headers (9) Body Scripts Settings
Cookies
raw JSON
Beautify
{
"pesanan_id" : "9",
"produk_id" : "4",
"jumlah" : "2",
"dibuat_pada" : "2025-01-02"
}
Body
200 OK
17 ms
280 B
{ } JSON
Preview
Visualize
{
"message": "Detail pesanan berhasil dihapus"
}


http://localhost:3000/api/pesanan/10
Save
PUT
http://localhost:3000/api/pesanan/10
Send
Params Auth Headers (9) Body Scripts Settings
Cookies
raw JSON
Beautify
{
"pengguna_id" : "4",
"status": "selesai",
"total_harga" : "8500000.00",
"dibuat_pada" : "2025-01-02"
}
Body
200 OK
6 ms
276 B
{ } JSON
Preview
Visualize
{
"message": "Pesanan berhasil diperbarui"
}

10.Kesimpulan

Laporan ini menggambarkan proses perancangan dan implementasi sistem toko online berbasis REST API dengan menggunakan teknologi modern seperti Node.js, Express.js, dan MySQL. Sistem ini dirancang untuk menyediakan fitur utama seperti autentikasi pengguna, manajemen produk, pengelolaan pesanan, serta detail pesanan. Setiap fitur didukung dengan validasi data yang aman menggunakan JsonWebToken (JWT).

Rancangan database yang sistematis memastikan integritas data melalui relasi antar tabel utama, seperti tabel pengguna, produk, pesanan, dan detail pesanan. Seluruh operasi CRUD diuji menggunakan Postman untuk menjamin keakuratan dan kestabilan API.

Proyek ini berhasil membangun sistem toko online yang andal dan efisien, yang tidak hanya mendukung kebutuhan operasional, tetapi juga memberikan pengalaman pengguna yang aman dan terstruktur. Laporan ini membuktikan bahwa teknologi yang digunakan dapat diimplementasikan secara efektif untuk menciptakan solusi digital berbasis platform.