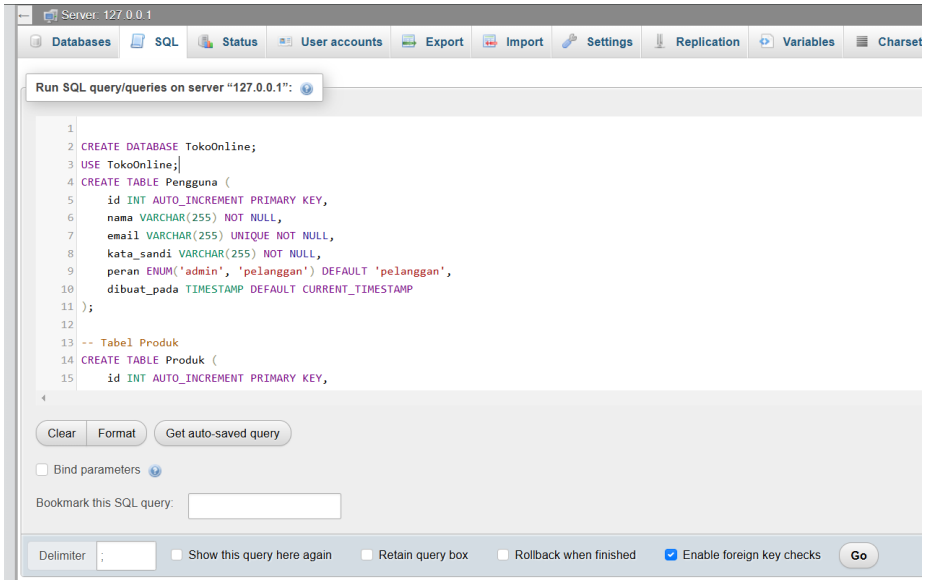
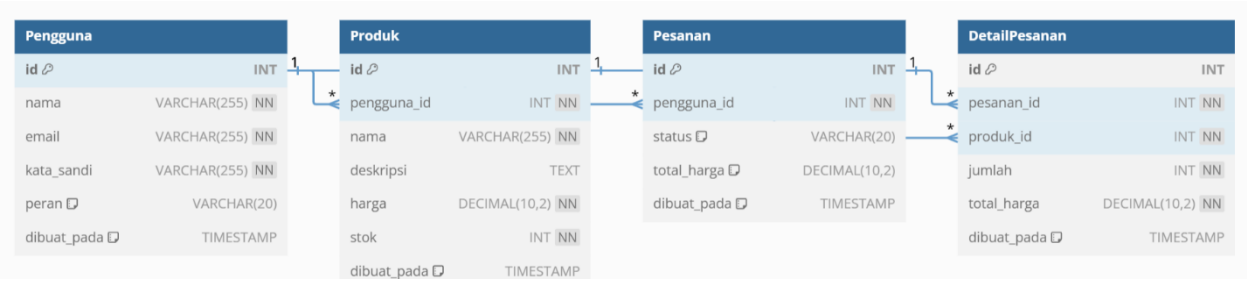
DOKUMENTASI HASIL PENGERJAAN

- Pembuatan Data Base



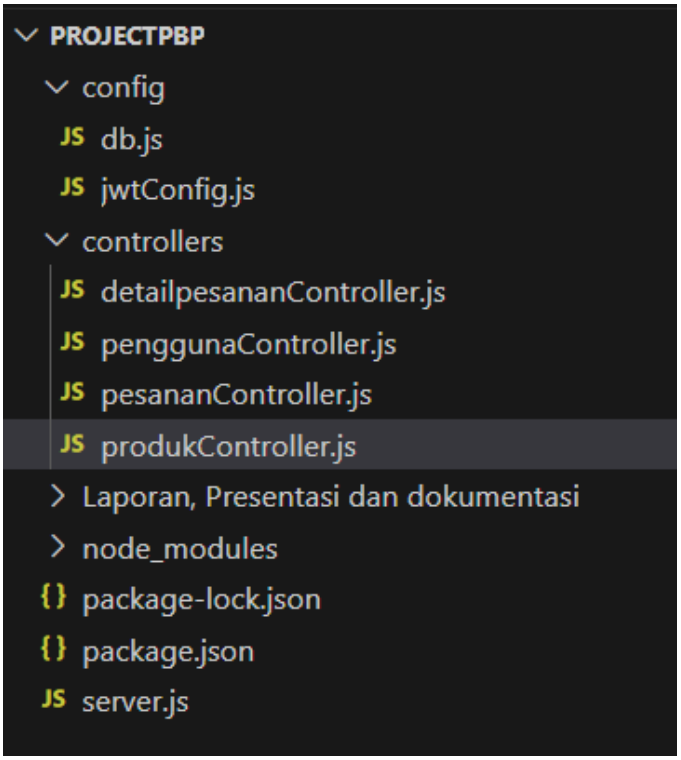
- Membuat ERD



- Instalsi Node Js

```
Microsoft Windows [Version 10.0.26100.2605]
(c) Microsoft Corporation. All rights reserved.

C:\Users\My Lenovo>mkdir tugaspbp

C:\Users\My Lenovo>cd tugaspbp

C:\Users\My Lenovo\tugaspbp>npm init -y
Wrote to C:\Users\My Lenovo\tugaspbp\package.json:

{
  "name": "tugaspbp",
  "version": "1.0.0",
  "description": "",
  "main": "index.js",
  "scripts": {
    "test": "echo \"Error: no test specified\" && exit 1"
  },
  "keywords": [],
  "author": "",
  "license": "ISC"
}



C:\Users\My Lenovo\tugaspbp>
```

- Struktur Folder



- Pembuatan Kode Program db.js , jwt.js, pengguna.js, produk.js, pesanan.js, detailpesanan.js dan server.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;
```

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

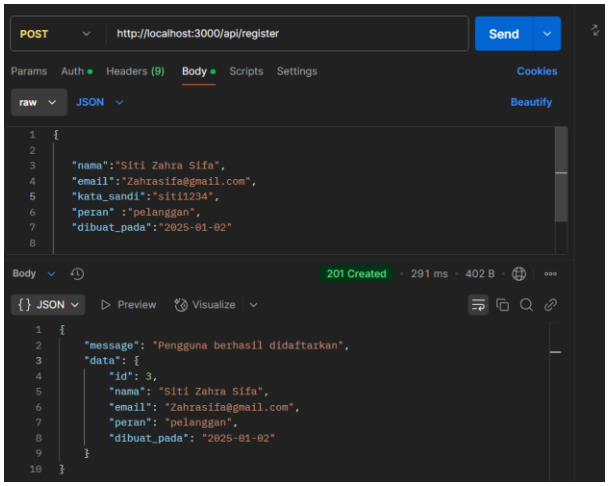
```
const express = require('express');
const router = express.Router();
const db = require('../config/db'); // Koneksi ke database
const bcrypt = require('bcrypt'); // Untuk hashing kata sandi
const jwt = require('jsonwebtoken'); // Untuk token JWT

// SECRET_KEY untuk JWT (ubah dengan nilai yang aman)
const SECRET_KEY = 'your_secret_key';

//1. POST /register - Registrasi pengguna
router.post('/register', async (req, res) => {
    const { nama, email, kata_sandi, peran, dibuat_pada } = req.body;

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

    try {
        // Cek apakah email sudah digunakan
        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!' });
        }

        // Hash kata sandi
        const hashedPassword = await bcrypt.hash(kata_sandi, 10);

        // Simpan pengguna ke database
        const result = await db.promise().query(
            'INSERT INTO pengguna (nama, email, kata_sandi, peran, dibuat_pada) VALUES (?, ?, ?, ?, ?)',
            [nama, email, hashedPassword, peran, dibuat_pada]
        );
```

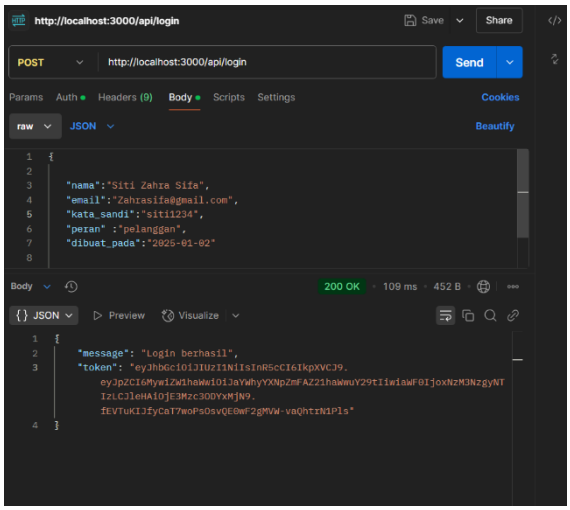
```
const express = require('express');
const router = express.Router();
const db = require('../config/db'); // Koneksi ke database

// 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) => {
```

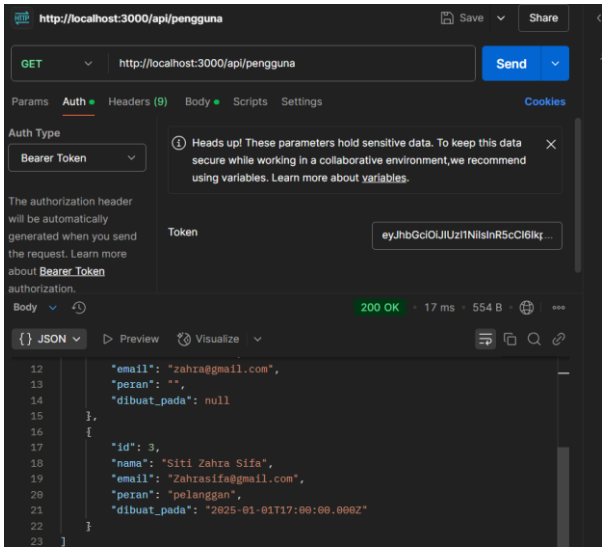
```
const express = require('express');
const router = express.Router();
const db = require('../config/db'); // Koneksi ke database

// 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' });
    }
});
```

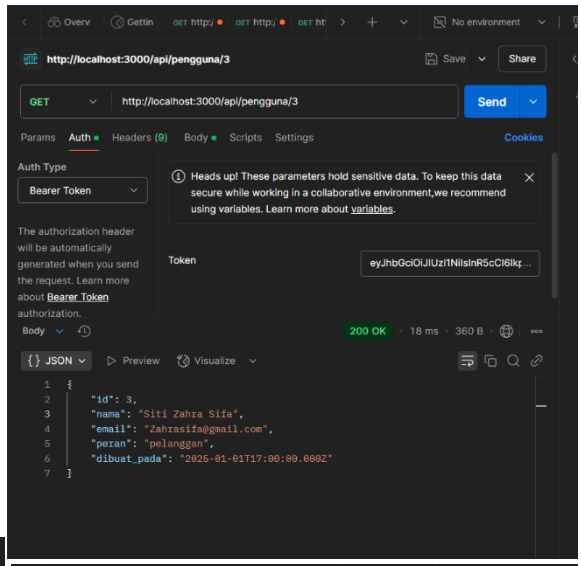
```
const express = require('express');
const router = express.Router();
const db = require('../config/db'); // Pastikan koneksi database benar

/**
 * 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]
        );

        res.status(201).json({
            message: 'Detail pesanan berhasil ditambahkan',
            data: {
                id: result.insertId,
                pesanan_id,
                produk_id,
                jumlah,
                dibuat_pada,
                total_harga
            }
        });
    } catch (error) {
        console.error('Error adding order detail:', error);
        res.status(500).json({ message: 'Terjadi kesalahan pada server' });
    }
});
```

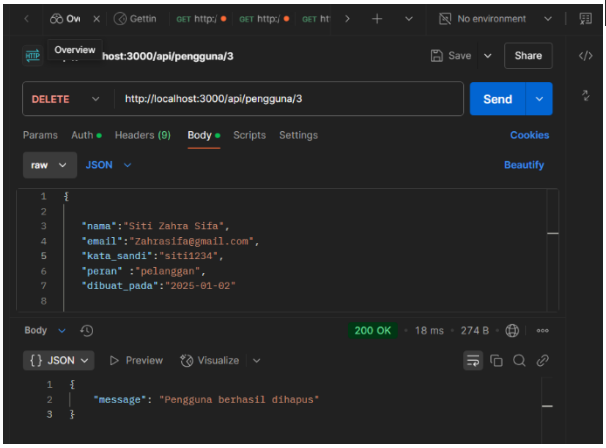
➢ Pengecekan Setiap Endpoint

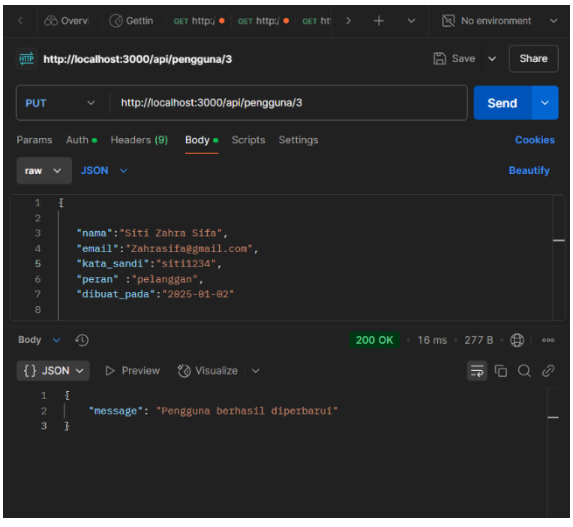


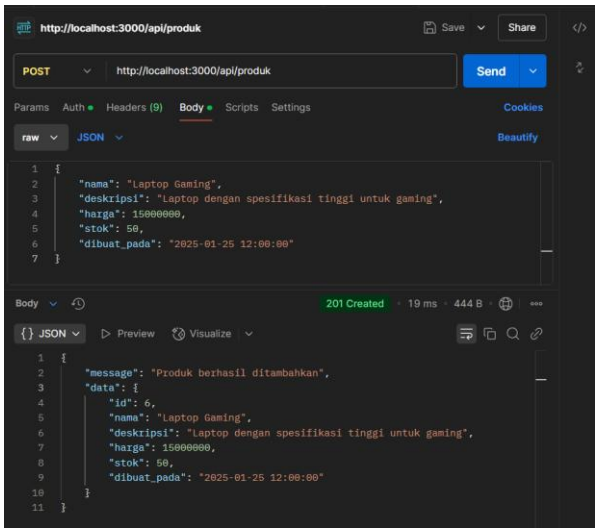


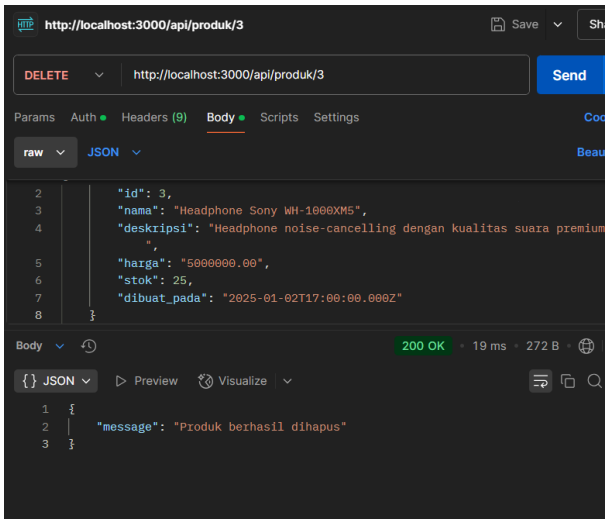


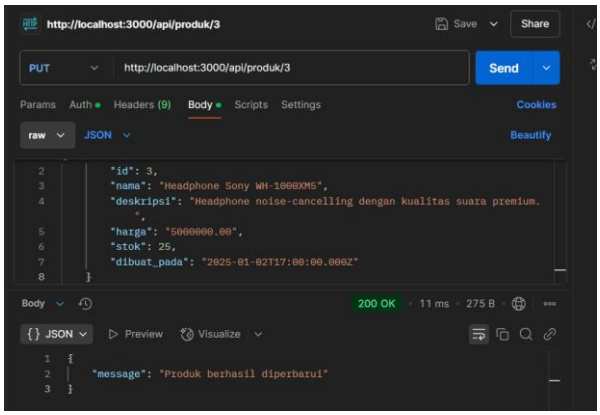


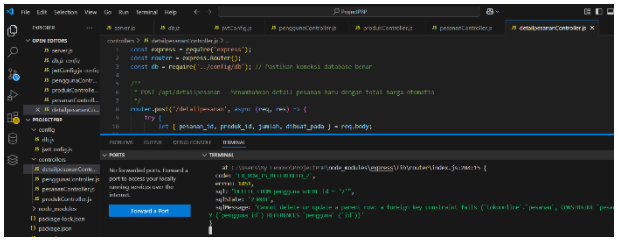


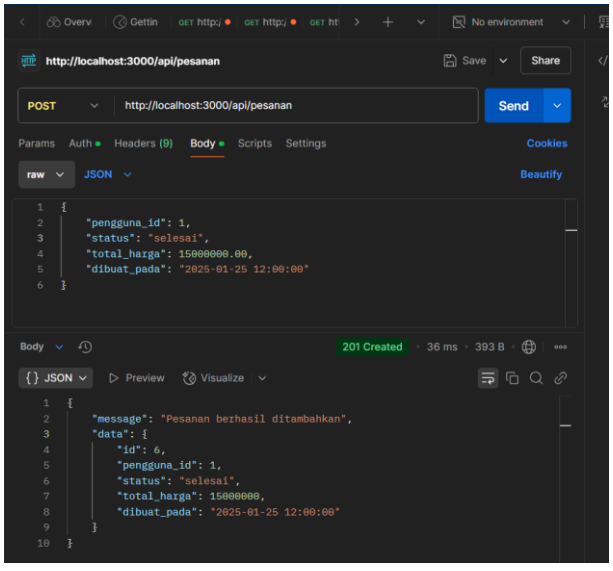


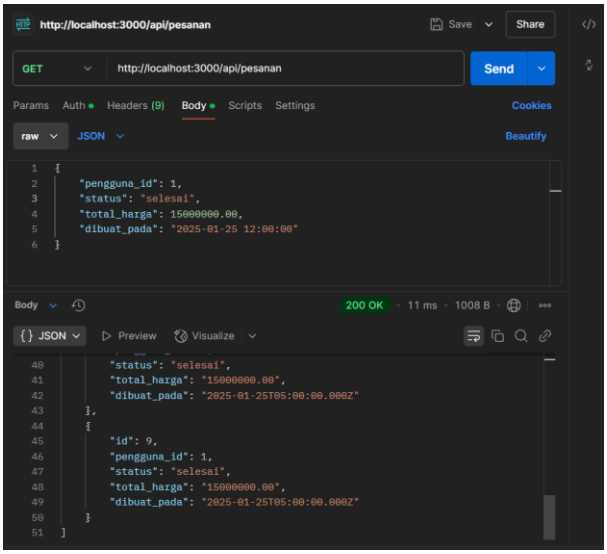

http://localhost:3000/api/produk
Save
Share
POST
http://localhost:3000/api/produk
Send
Params Auth Headers (9) Body Scripts Settings
Cookies
raw JSON
Beautify
{
"nama": "Laptop Gaming",
"deskripsi": "Laptop dengan spesifikasi tinggi untuk gaming",
"harga": 15000000,
"stok": 50,
"dibuat_pada": "2025-01-25 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"
}
}

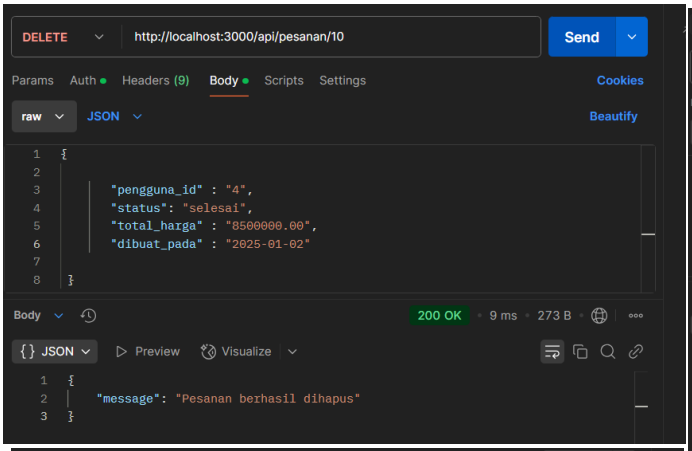

http://localhost:3000/api/produk/3
Save
DELETE
http://localhost:3000/api/produk/3
Send
Params Auth Headers (9) Body Scripts Settings
raw JSON
"id": 3,
"nama": "Headphone Sony WH-1000XM5",
"deskripsi": "Headphone noise-cancelling dengan kualitas suara premium
",
"harga": "5000000.00",
"stok": 25,
"dibuat_pada": "2025-01-02T17:00:00.000Z"
}
Body
200 OK
19 ms
272 B
JSON
Preview
Visualize
{
"message": "Produk berhasil dihapus"
}

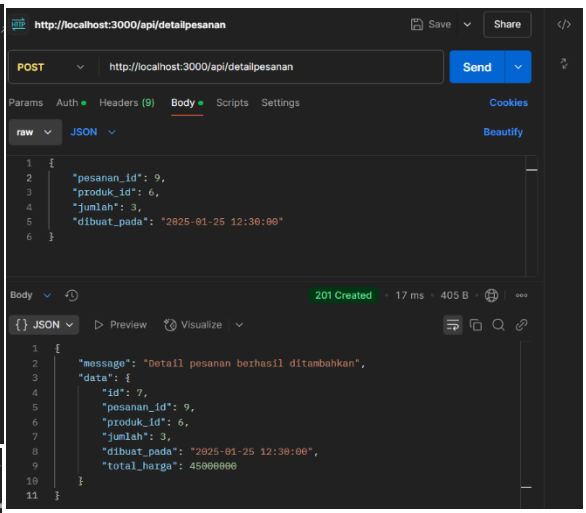

http://localhost:3000/api/produk/3
Save
Share
PUT
http://localhost:3000/api/produk/3
Send
Params Auth Headers (9) Body Scripts Settings
Cookies
raw JSON
Beautify
"id": 3,
"nama": "Headphone Sony WH-1000XM5",
"deskripsi": "Headphone noise-cancelling dengan kualitas suara premium.
",
"harga": "5000000.00",
"stok": 25,
"dibuat_pada": "2025-01-02T17:00:00.000Z"
}
Body
200 OK
11 ms
275 B
JSON
Preview
Visualize
{
"message": "Produk berhasil diperbarui"
}

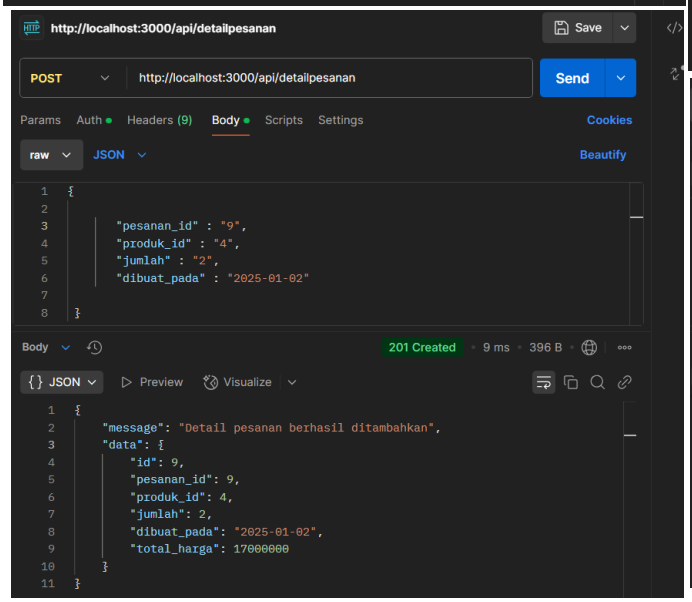


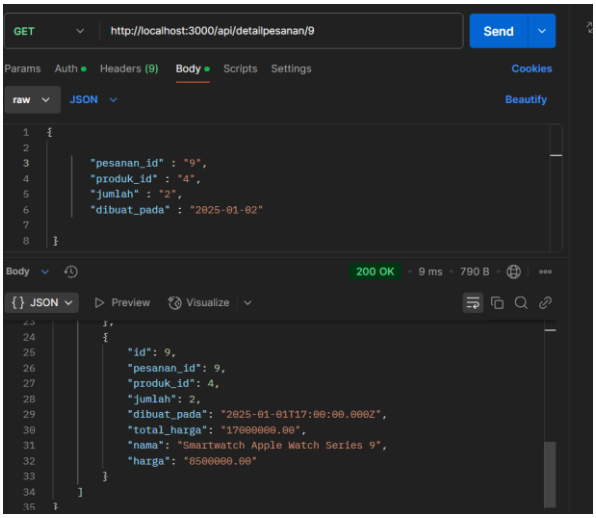

No environment
http://localhost:3000/api/pesanan
Save
Share
POST
http://localhost:3000/api/pesanan
Send
Params Auth Headers (9) Body Scripts Settings
Cookies
raw JSON
Beautify
{
"pengguna_id": 1,
"status": "selesai",
"total_harga": 15000000.00,
"dibuat_pada": "2025-01-25 12:00:00"
}
Body
201 Created
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"
}
}

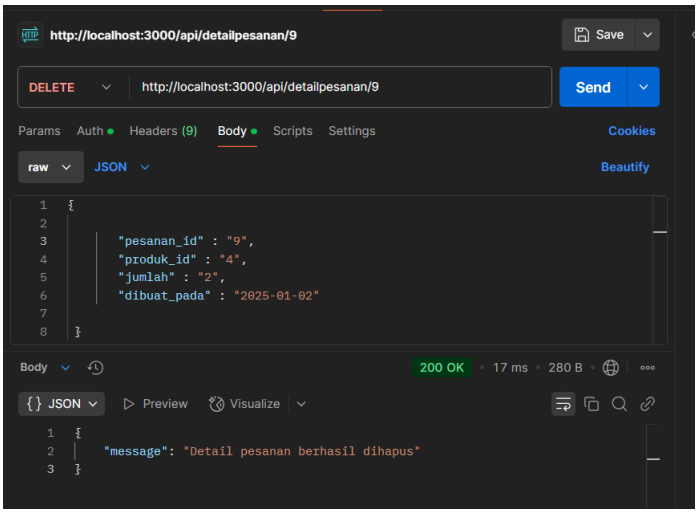

http://localhost:3000/api/pesanan
Save
Share
GET
http://localhost:3000/api/pesanan
Send
Params Auth Headers (9) Body Scripts Settings
Cookies
raw JSON
Beautify
{
"pengguna_id": 1,
"status": "selesai",
"total_harga": 15000000.00,
"dibuat_pada": "2025-01-25 12:00:00"
}
Body
200 OK
11 ms
1008 B
JSON
Preview
Visualize
"status": "selesai",
"total_harga": "15000000.00",
"dibuat_pada": "2025-01-25T05:00:00.000Z"
},
{
"id": 9,
"pengguna_id": 1,
"status": "selesai",
"total_harga": "15000000.00",
"dibuat_pada": "2025-01-25T05:00:00.000Z"
}
]











- Semua Endpoint telah dicoba dan hasilnya berjalan sesuai yang diharapkan, untuk kode program bisa dilihat pada github berikut SitiZahraSifa11/Project_PBP_Kelompok2_TI23F.