**LAPORAN TUGAS UTS BACKEND**



**Disusun Oleh**

**JANUARAYA KANANTA /3124101300**

**Program Studi D3 Manajamen Informatika  
STIKOM PGRI Banyuwangi  
Tahun 2025**

# Soal:

1. Buatlah sebuah database MySQL dengan ketentuan

a. Namadatabase: toko

b. Nama Tabel: supplier

c. Field: id Sup, namaSupplier, Perusahaan, barang, kuota, hari

2. Buatlah sebuah api untuk crud database diatas menggunakan express.js

3. Lakukan ujicoba dengan postman atau insomnia

4. Upload project dan database ke github.

5. Buatkan laporan.

6. Kirimkan link laporan ke elearning

### **B. Penjelasan Isi Folder dan File**

1. **Folder controller**  
   Berisi file toko\_controller.js yang berfungsi untuk menangani logika dari seluruh aksi CRUD (Create, Read, Update, Delete) pada data supplier.
   * Mengatur request dan response untuk:
     + Menampilkan semua data supplier
     + Menampilkan data supplier berdasarkan ID
     + Menambahkan supplier baru
     + Memperbarui data supplier
     + Menghapus data supplier
2. **Folder routes**  
   Berisi file toko\_routes.js yang berfungsi untuk mendefinisikan rute API dan menghubungkannya dengan fungsi-fungsi yang ada di controller.
   * Menangani endpoint seperti:
     + GET /api/supplier/
     + GET /api/supplier/:id
     + POST /api/supplier/
     + PUT /api/supplier/:id
     + DELETE /api/supplier/:id
3. **Folder models**  
   Berisi file supplier.js yang merupakan definisi model Sequelize untuk tabel supplier.
   * Menentukan struktur kolom data supplier yang akan disimpan ke dalam database MySQL.
4. **Folder config**  
   Berisi file db.js yang bertanggung jawab untuk mengatur koneksi ke database MySQL menggunakan Sequelize.
5. **Folder node\_modules**  
   Berisi seluruh paket/dependensi yang dibutuhkan oleh project Node.js seperti express, sequelize, mysql2, dll. Folder ini dibuat secara otomatis saat menjalankan npm install.
6. **File index.js**  
   Merupakan file utama (entry point) untuk menjalankan server Express.
   * Mengatur middleware seperti cors dan express.json()
   * Menyambungkan routing API supplier
   * Menjalankan server di port 3000
7. **File package.json**  
   Berisi metadata project seperti nama, versi, deskripsi, dan terutama **daftar dependensi** (package) yang digunakan oleh project, serta script untuk menjalankan server, seperti npm start.
8. **File package-lock.json**  
   Berfungsi untuk **mengunci versi** dari setiap dependensi agar instalasi project tetap **konsisten** di berbagai lingkungan, mencegah perbedaan versi paket antar developer atau server

# C. Code

1. Index.js

const express = require("express");

const bodyParser = require('body-parser');

const cors = require('cors');

const supplierRoutes = require('./routes/toko\_routes'); // Fix path dan nama

const app = express();

const port = 3000;

app.use(cors());

app.use(express.json()); // Menggunakan express.json() untuk parsing JSON body secara otomatis

app.use('/api/supplier', supplierRoutes); // Menggunakan route yang sudah dibuat

app.listen(port, () => {

console.log(`Server running at http://localhost:${port}`);

const { DataTypes } = require('sequelize');

const sequelize = require('../config/db.js'); // Pastikan path ke db.js benar

1. supplier.js

const Supplier = sequelize.define('supplier', {

id: {

type: DataTypes.INTEGER,

primaryKey: true,

autoIncrement: true,

},

namaSupplier: {

type: DataTypes.STRING,

allowNull: false,

},

Perusahaan: {

type: DataTypes.STRING,

allowNull: false,

},

Barang: {

type: DataTypes.STRING,

allowNull: false,

},

kuota: {

type: DataTypes.INTEGER,

allowNull: false,

},

hari: {

type: DataTypes.DATEONLY,

allowNull: false,

},

}, {

tableName: 'supplier',

timestamps: false, // Tidak menggunakan createdAt/updatedAt

});

module.exports = Supplier;

1. Routes.js

const express = require('express');

const router = express.Router();

const supplierController = require('../controller/toko\_controller'); // Pastikan path controller benar

// Routes untuk API Supplier

router.get('/', supplierController.get); // Mengambil semua supplier

router.get('/:id', supplierController.get); // Mengambil supplier berdasarkan ID

router.post('/', supplierController.post); // Membuat supplier baru

router.put('/:id', supplierController.put); // Update supplier berdasarkan ID

router.delete('/:id', supplierController.del); // Hapus supplier berdasarkan ID

module.exports = router;

1. Toko\_supplier.js

const Supplier = require('../models/supplier'); // Mengimpor model Supplier

// GET semua atau satu Supplier

const get = async (req, res) => {

try {

if (req.params.id) {

const supplier = await Supplier.findByPk(req.params.id);

if (supplier) {

res.json(supplier); // Menampilkan satu supplier berdasarkan ID

} else {

res.status(404).json({ message: 'Supplier not found' });

}

} else {

const suppliers = await Supplier.findAll();

res.json(suppliers); // Menampilkan semua supplier

}

} catch (error) {

res.status(500).json({ error: error.message });

}

};

// POST - buat Supplier baru

const post = async (req, res) => {

const { namaSupplier, Perusahaan, Barang, kuota, hari } = req.body;

// Validasi input sebelum menyimpan data ke DB

if (!namaSupplier || !Perusahaan || !Barang || !kuota || !hari) {

return res.status(400).json({ error: "Semua field harus diisi" });

}

try {

const supplier = await Supplier.create({ namaSupplier, Perusahaan, Barang, kuota, hari });

res.status(201).json(supplier); // Menampilkan supplier yang baru dibuat

} catch (error) {

res.status(400).json({ error: error.message });

}

};

// PUT - update Supplier berdasarkan ID

const put = async (req, res) => {

const { id } = req.params;

const { namaSupplier, Perusahaan, Barang, kuota, hari } = req.body;

try {

const updated = await Supplier.update(

{ namaSupplier, Perusahaan, Barang, kuota, hari },

{ where: { id } }

);

if (updated[0] > 0) {

res.json({ message: 'Supplier updated successfully' });

} else {

res.status(404).json({ message: 'Supplier not found' });

}

} catch (error) {

res.status(400).json({ error: error.message });

}

};

// DELETE - hapus Supplier berdasarkan ID

const del = async (req, res) => {

try {

const supplier = await Supplier.findByPk(req.params.id);

if (supplier) {

await supplier.destroy();

res.json({ message: 'Supplier deleted successfully' });

} else {

res.status(404).json({ message: 'Supplier not found' });

}

} catch (error) {

res.status(500).json({ error: error.message });

}

};

5.Db.js

module.exports = { get, post, put, del };

const { Sequelize } = require('sequelize');

const sequelize = new Sequelize('toko', 'root', '', {

host: 'localhost',

dialect: 'mysql',

port: 3306,

});

// Cek koneksi ke database

sequelize.authenticate()

.then(() => {

console.log('Connection has been established successfully.');

})

.catch((error) => {

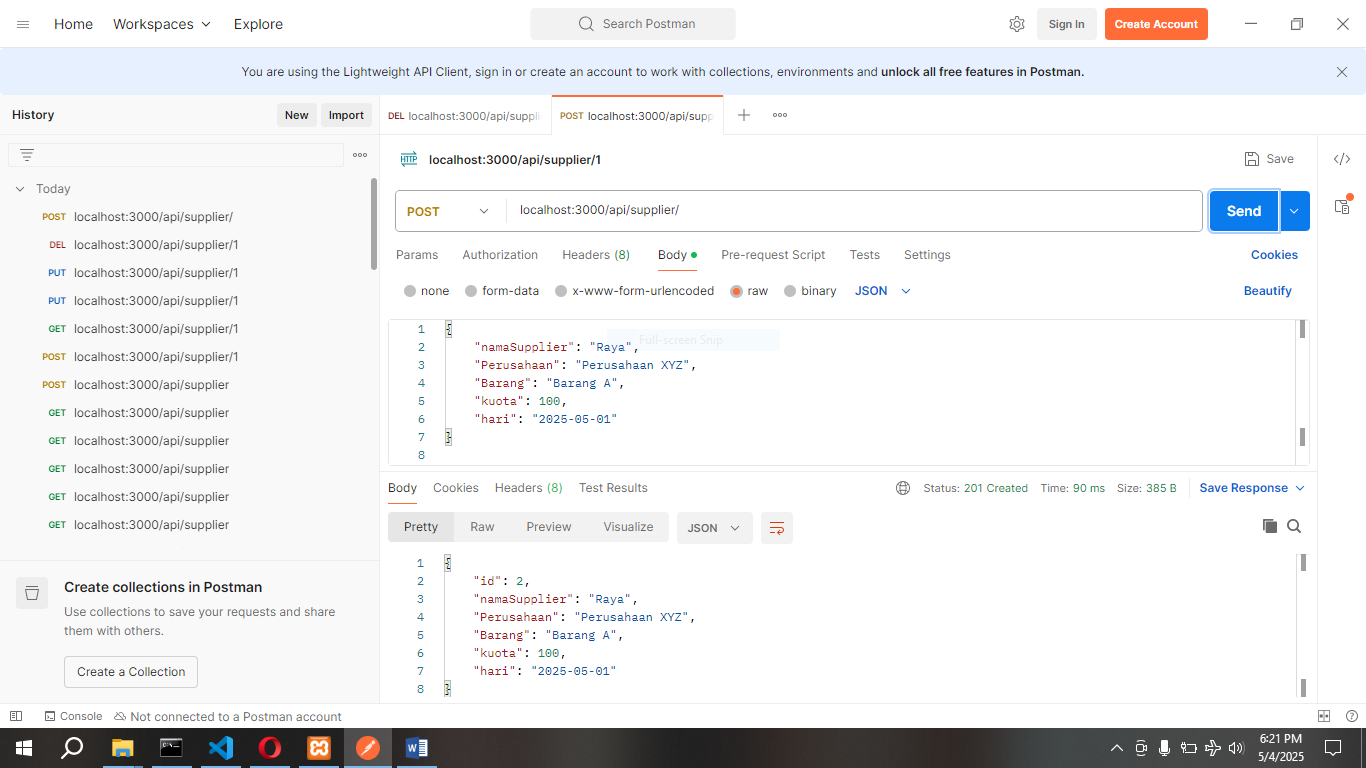
console.error('Unable to connect to the database:', error);

});

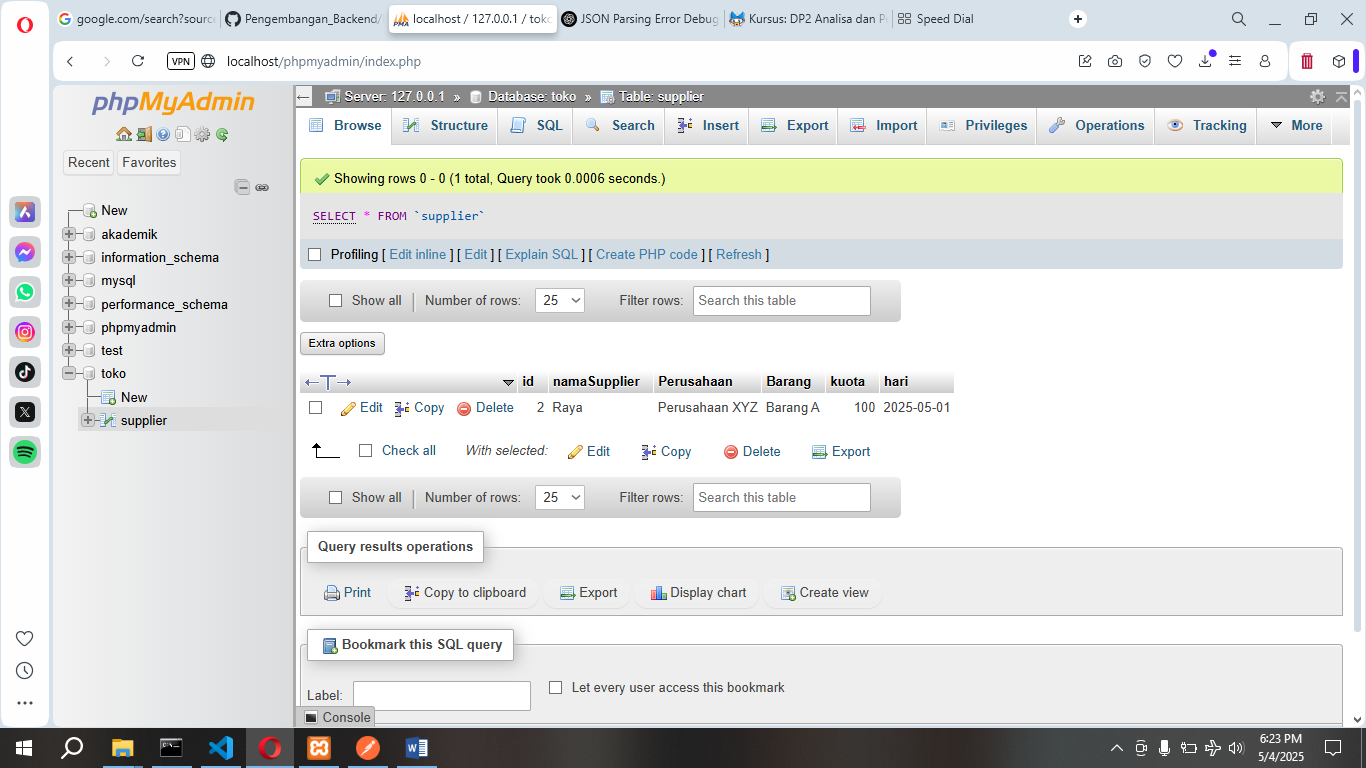
module.exports = sequelize;

# DATA BASE

1. Hasil Postman

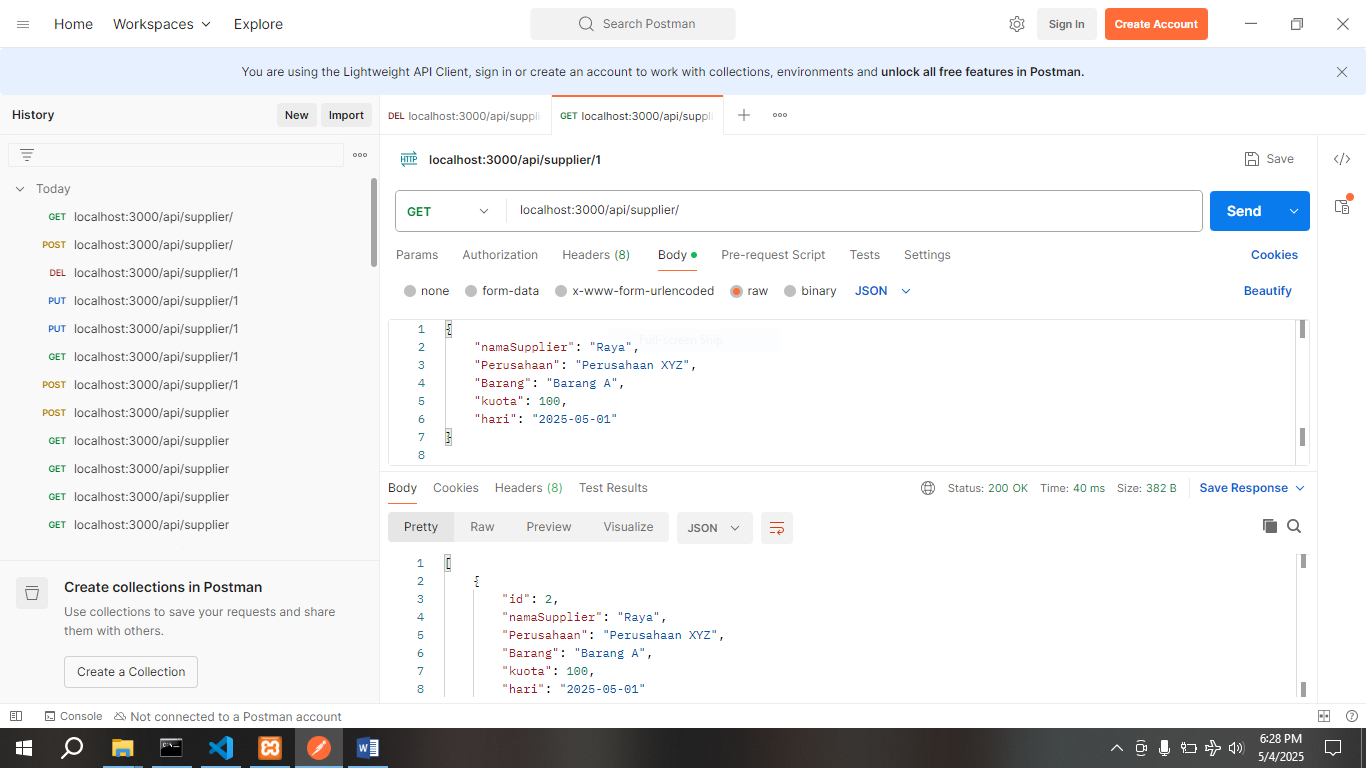


1. Database

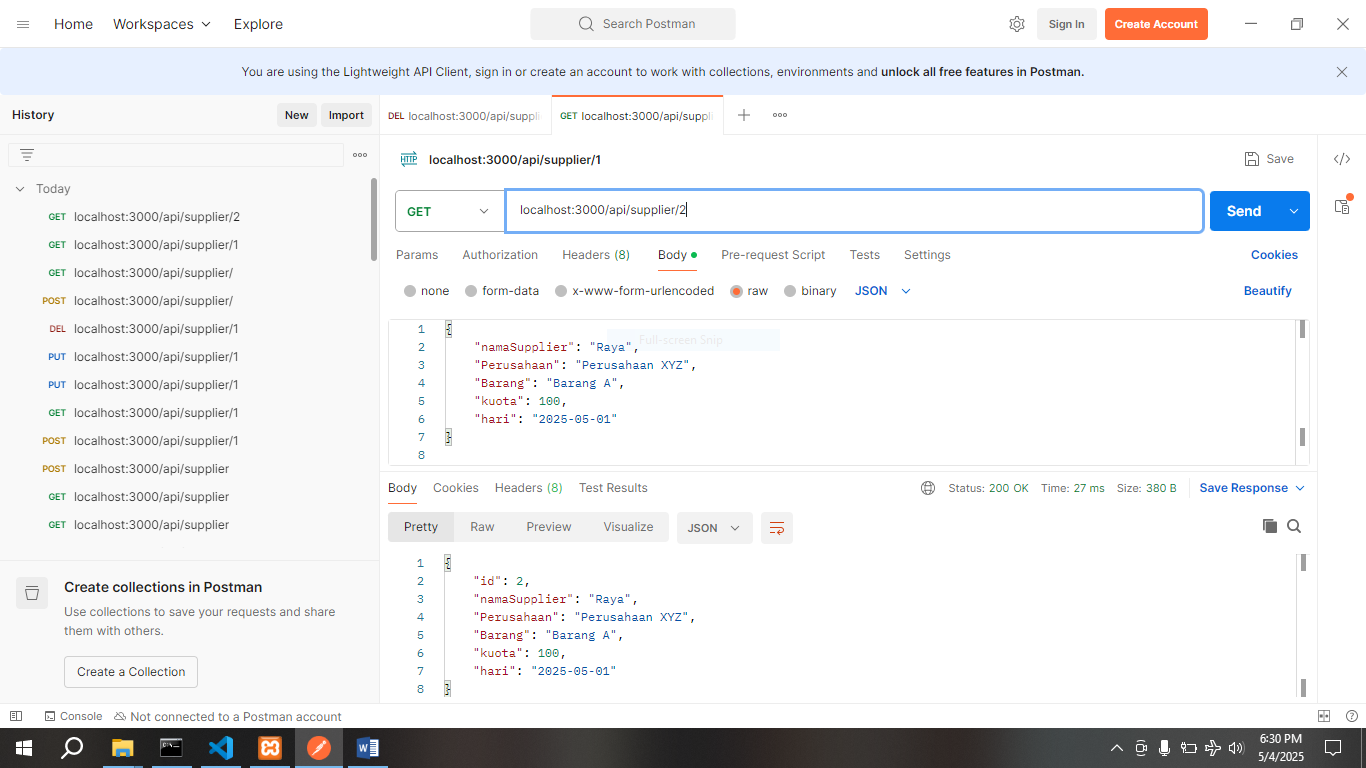


# HASIL TAMPILAN

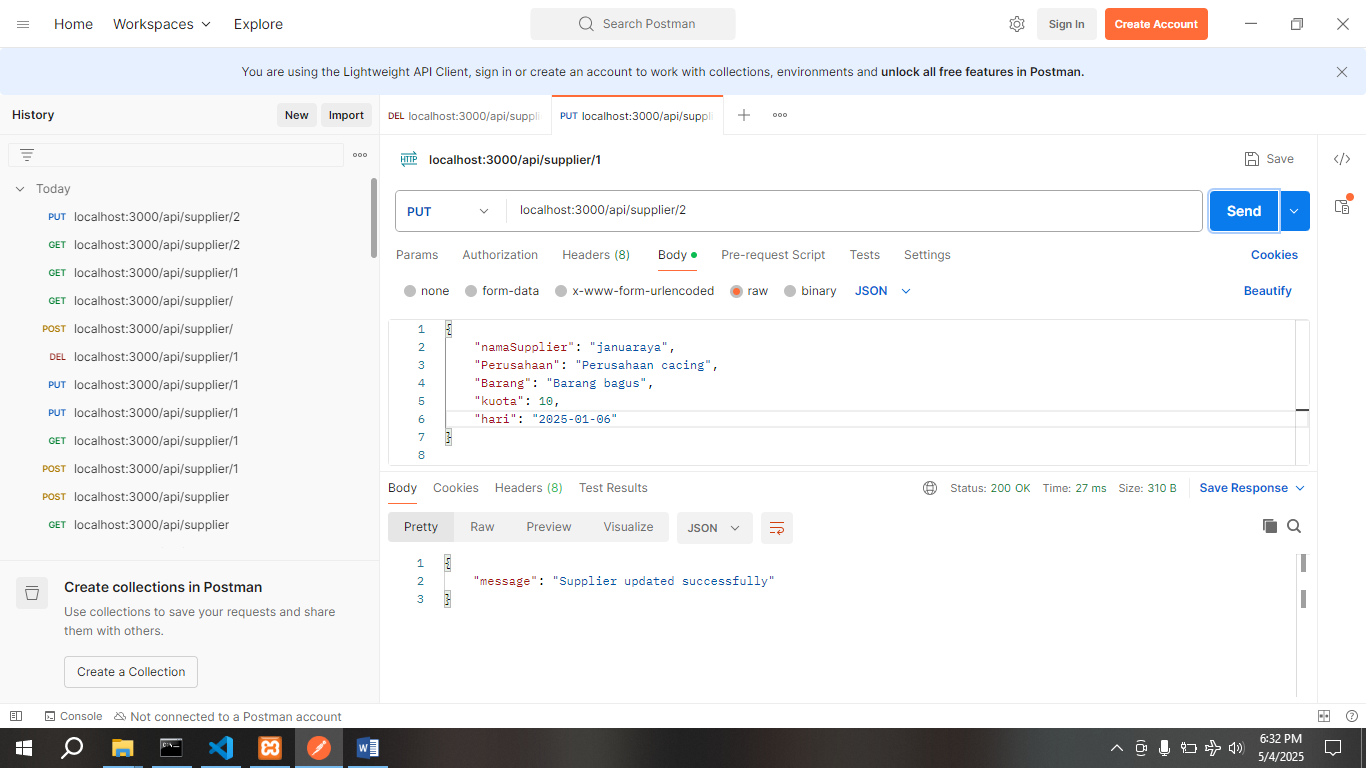
* + - 1. Get menampilkan data



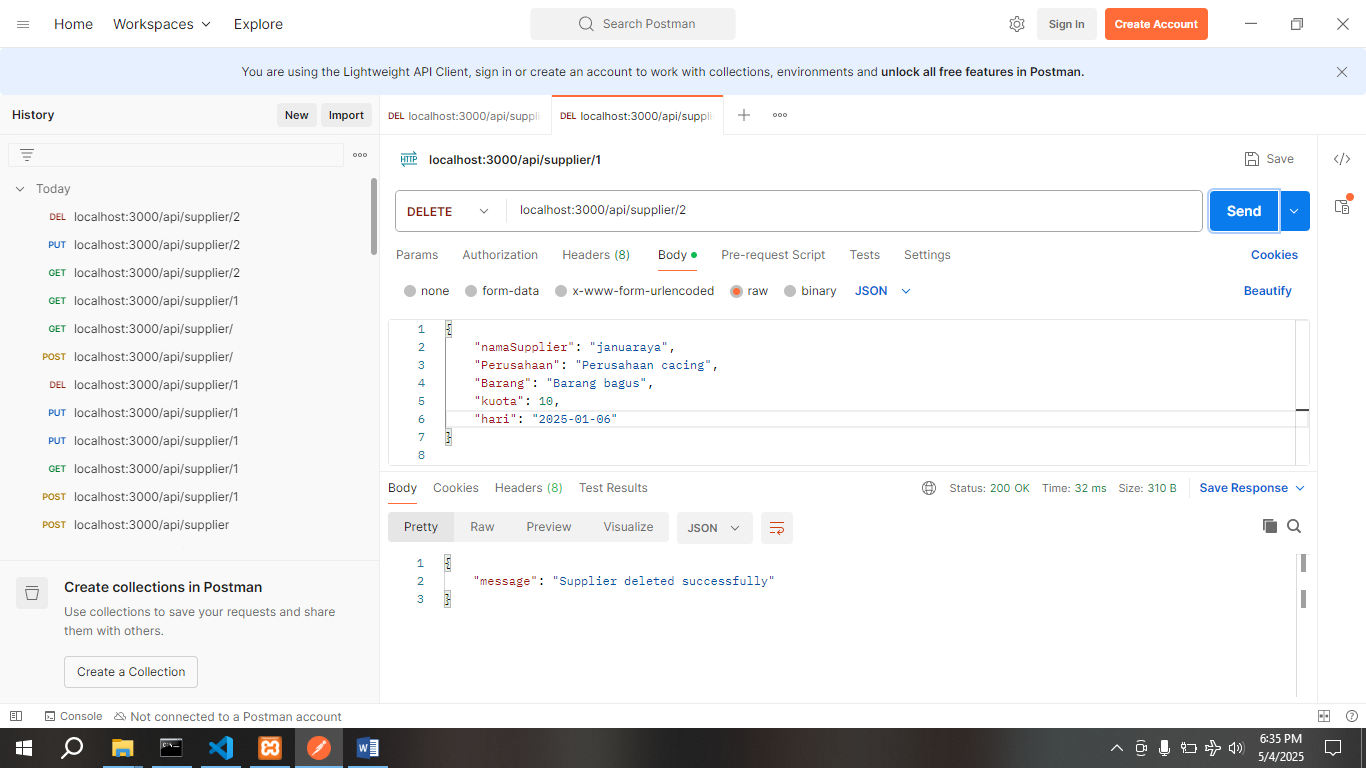
* + - 1. get berdasrkan ID



# tampilan put



* + - 1. tampilan delete



# Soal:

1. Buatlah sebuah database MySQL dengan ketentuan

a. Namadatabase: toko

b. Nama Tabel: supplier

c. Field: id Sup, namaSupplier, Perusahaan, barang, kuota, hari

2. Buatlah sebuah api untuk crud database diatas menggunakan express.js

3. Lakukan ujicoba dengan postman atau insomnia

4. Upload project dan database ke github.

5. Buatkan laporan.

6. Kirimkan link laporan ke elearning

### **B. Penjelasan Isi Folder dan File**

1. **Folder controller**  
   Berisi file toko\_controller.js yang berfungsi untuk menangani logika dari seluruh aksi CRUD (Create, Read, Update, Delete) pada data supplier.
   * Mengatur request dan response untuk:
     + Menampilkan semua data supplier
     + Menampilkan data supplier berdasarkan ID
     + Menambahkan supplier baru
     + Memperbarui data supplier
     + Menghapus data supplier
2. **Folder routes**  
   Berisi file toko\_routes.js yang berfungsi untuk mendefinisikan rute API dan menghubungkannya dengan fungsi-fungsi yang ada di controller.
   * Menangani endpoint seperti:
     + GET /api/supplier/
     + GET /api/supplier/:id
     + POST /api/supplier/
     + PUT /api/supplier/:id
     + DELETE /api/supplier/:id
3. **Folder models**  
   Berisi file supplier.js yang merupakan definisi model Sequelize untuk tabel supplier.
   * Menentukan struktur kolom data supplier yang akan disimpan ke dalam database MySQL.
4. **Folder config**  
   Berisi file db.js yang bertanggung jawab untuk mengatur koneksi ke database MySQL menggunakan Sequelize.
5. **Folder node\_modules**  
   Berisi seluruh paket/dependensi yang dibutuhkan oleh project Node.js seperti express, sequelize, mysql2, dll. Folder ini dibuat secara otomatis saat menjalankan npm install.
6. **File index.js**  
   Merupakan file utama (entry point) untuk menjalankan server Express.
   * Mengatur middleware seperti cors dan express.json()
   * Menyambungkan routing API supplier
   * Menjalankan server di port 3000
7. **File package.json**  
   Berisi metadata project seperti nama, versi, deskripsi, dan terutama **daftar dependensi** (package) yang digunakan oleh project, serta script untuk menjalankan server, seperti npm start.
8. **File package-lock.json**  
   Berfungsi untuk **mengunci versi** dari setiap dependensi agar instalasi project tetap **konsisten** di berbagai lingkungan, mencegah perbedaan versi paket antar developer atau server

# C. Code

1. Index.js

const express = require("express");

const bodyParser = require('body-parser');

const cors = require('cors');

const supplierRoutes = require('./routes/toko\_routes'); // Fix path dan nama

const app = express();

const port = 3000;

app.use(cors());

app.use(express.json()); // Menggunakan express.json() untuk parsing JSON body secara otomatis

app.use('/api/supplier', supplierRoutes); // Menggunakan route yang sudah dibuat

app.listen(port, () => {

console.log(`Server running at http://localhost:${port}`);

const { DataTypes } = require('sequelize');

const sequelize = require('../config/db.js'); // Pastikan path ke db.js benar

1. supplier.js

const Supplier = sequelize.define('supplier', {

id: {

type: DataTypes.INTEGER,

primaryKey: true,

autoIncrement: true,

},

namaSupplier: {

type: DataTypes.STRING,

allowNull: false,

},

Perusahaan: {

type: DataTypes.STRING,

allowNull: false,

},

Barang: {

type: DataTypes.STRING,

allowNull: false,

},

kuota: {

type: DataTypes.INTEGER,

allowNull: false,

},

hari: {

type: DataTypes.DATEONLY,

allowNull: false,

},

}, {

tableName: 'supplier',

timestamps: false, // Tidak menggunakan createdAt/updatedAt

});

module.exports = Supplier;

1. Routes.js

const express = require('express');

const router = express.Router();

const supplierController = require('../controller/toko\_controller'); // Pastikan path controller benar

// Routes untuk API Supplier

router.get('/', supplierController.get); // Mengambil semua supplier

router.get('/:id', supplierController.get); // Mengambil supplier berdasarkan ID

router.post('/', supplierController.post); // Membuat supplier baru

router.put('/:id', supplierController.put); // Update supplier berdasarkan ID

router.delete('/:id', supplierController.del); // Hapus supplier berdasarkan ID

module.exports = router;

1. Toko\_supplier.js

const Supplier = require('../models/supplier'); // Mengimpor model Supplier

// GET semua atau satu Supplier

const get = async (req, res) => {

try {

if (req.params.id) {

const supplier = await Supplier.findByPk(req.params.id);

if (supplier) {

res.json(supplier); // Menampilkan satu supplier berdasarkan ID

} else {

res.status(404).json({ message: 'Supplier not found' });

}

} else {

const suppliers = await Supplier.findAll();

res.json(suppliers); // Menampilkan semua supplier

}

} catch (error) {

res.status(500).json({ error: error.message });

}

};

// POST - buat Supplier baru

const post = async (req, res) => {

const { namaSupplier, Perusahaan, Barang, kuota, hari } = req.body;

// Validasi input sebelum menyimpan data ke DB

if (!namaSupplier || !Perusahaan || !Barang || !kuota || !hari) {

return res.status(400).json({ error: "Semua field harus diisi" });

}

try {

const supplier = await Supplier.create({ namaSupplier, Perusahaan, Barang, kuota, hari });

res.status(201).json(supplier); // Menampilkan supplier yang baru dibuat

} catch (error) {

res.status(400).json({ error: error.message });

}

};

// PUT - update Supplier berdasarkan ID

const put = async (req, res) => {

const { id } = req.params;

const { namaSupplier, Perusahaan, Barang, kuota, hari } = req.body;

try {

const updated = await Supplier.update(

{ namaSupplier, Perusahaan, Barang, kuota, hari },

{ where: { id } }

);

if (updated[0] > 0) {

res.json({ message: 'Supplier updated successfully' });

} else {

res.status(404).json({ message: 'Supplier not found' });

}

} catch (error) {

res.status(400).json({ error: error.message });

}

};

// DELETE - hapus Supplier berdasarkan ID

const del = async (req, res) => {

try {

const supplier = await Supplier.findByPk(req.params.id);

if (supplier) {

await supplier.destroy();

res.json({ message: 'Supplier deleted successfully' });

} else {

res.status(404).json({ message: 'Supplier not found' });

}

} catch (error) {

res.status(500).json({ error: error.message });

}

};

5.Db.js

module.exports = { get, post, put, del };

const { Sequelize } = require('sequelize');

const sequelize = new Sequelize('toko', 'root', '', {

host: 'localhost',

dialect: 'mysql',

port: 3306,

});

// Cek koneksi ke database

sequelize.authenticate()

.then(() => {

console.log('Connection has been established successfully.');

})

.catch((error) => {

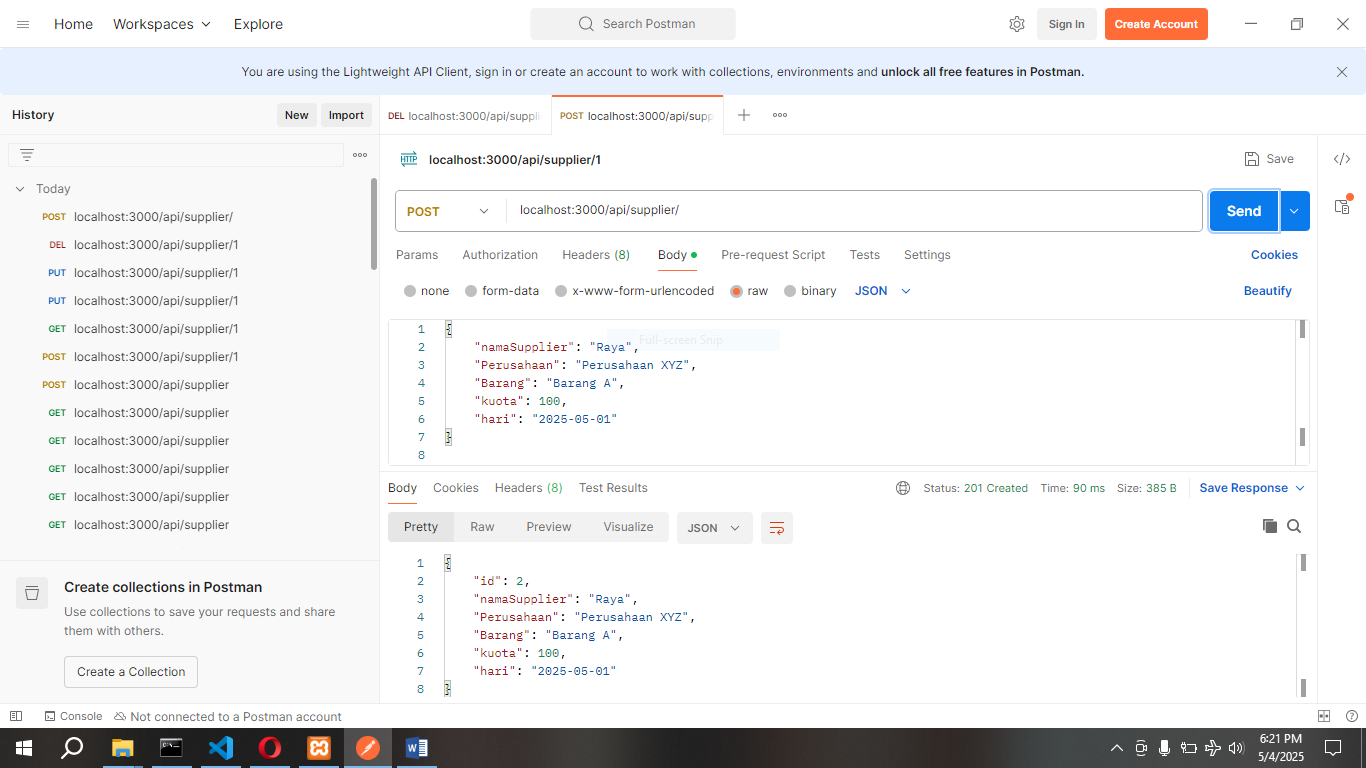
console.error('Unable to connect to the database:', error);

});

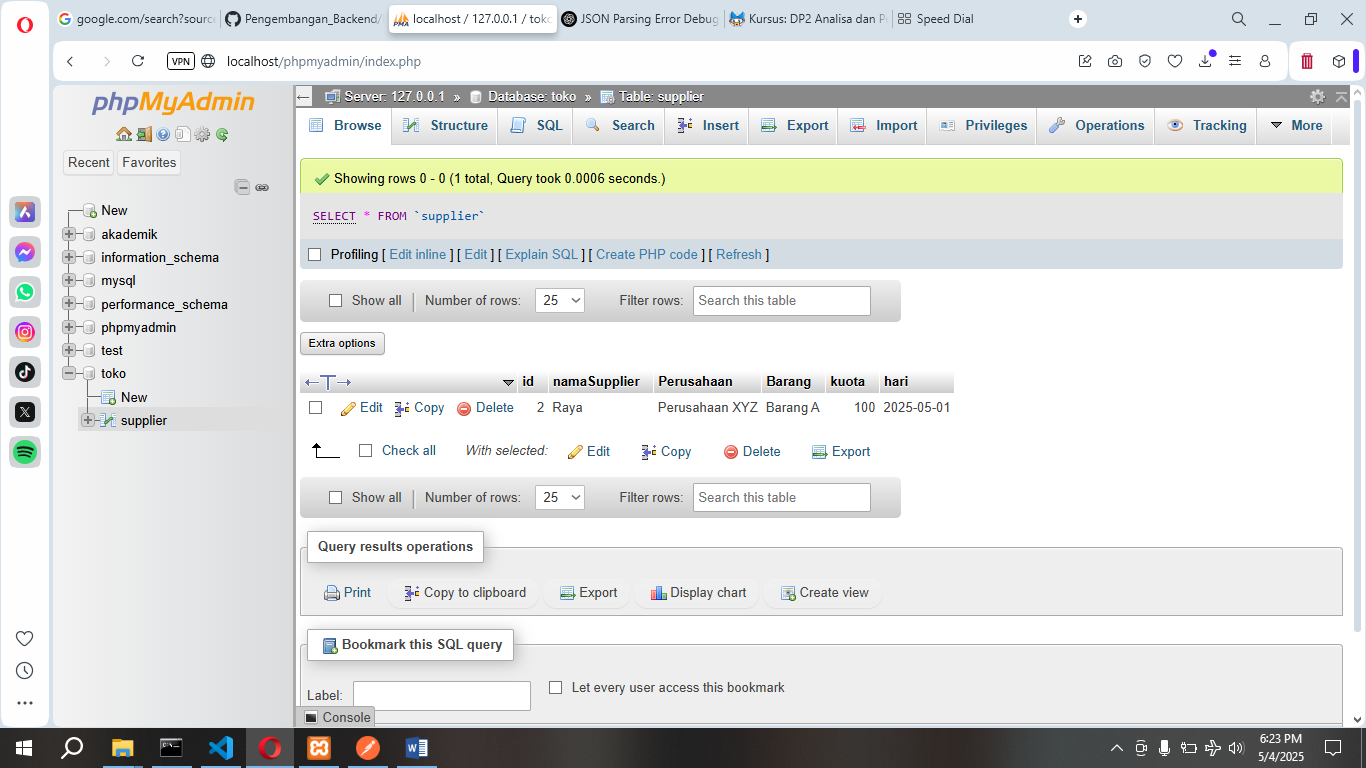
module.exports = sequelize;

# DATA BASE

1. Hasil Postman

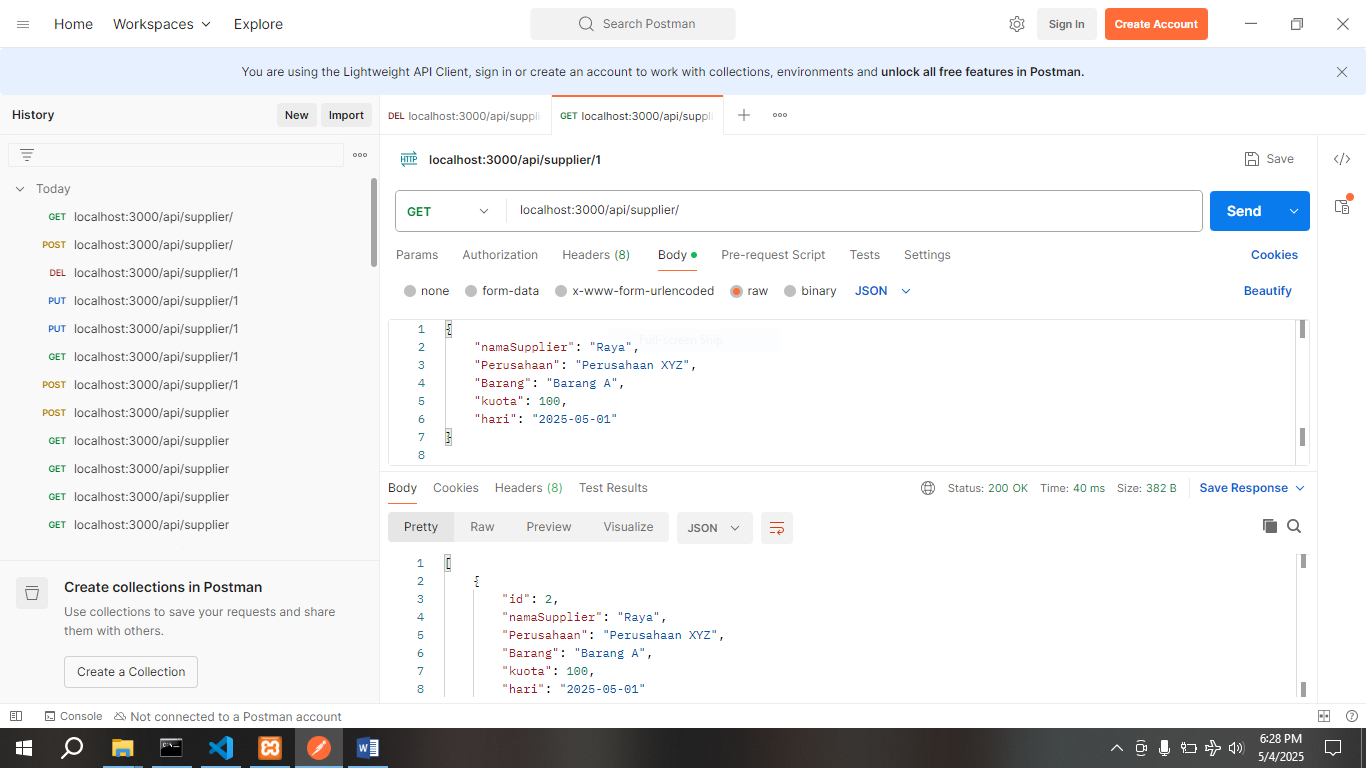


1. Database

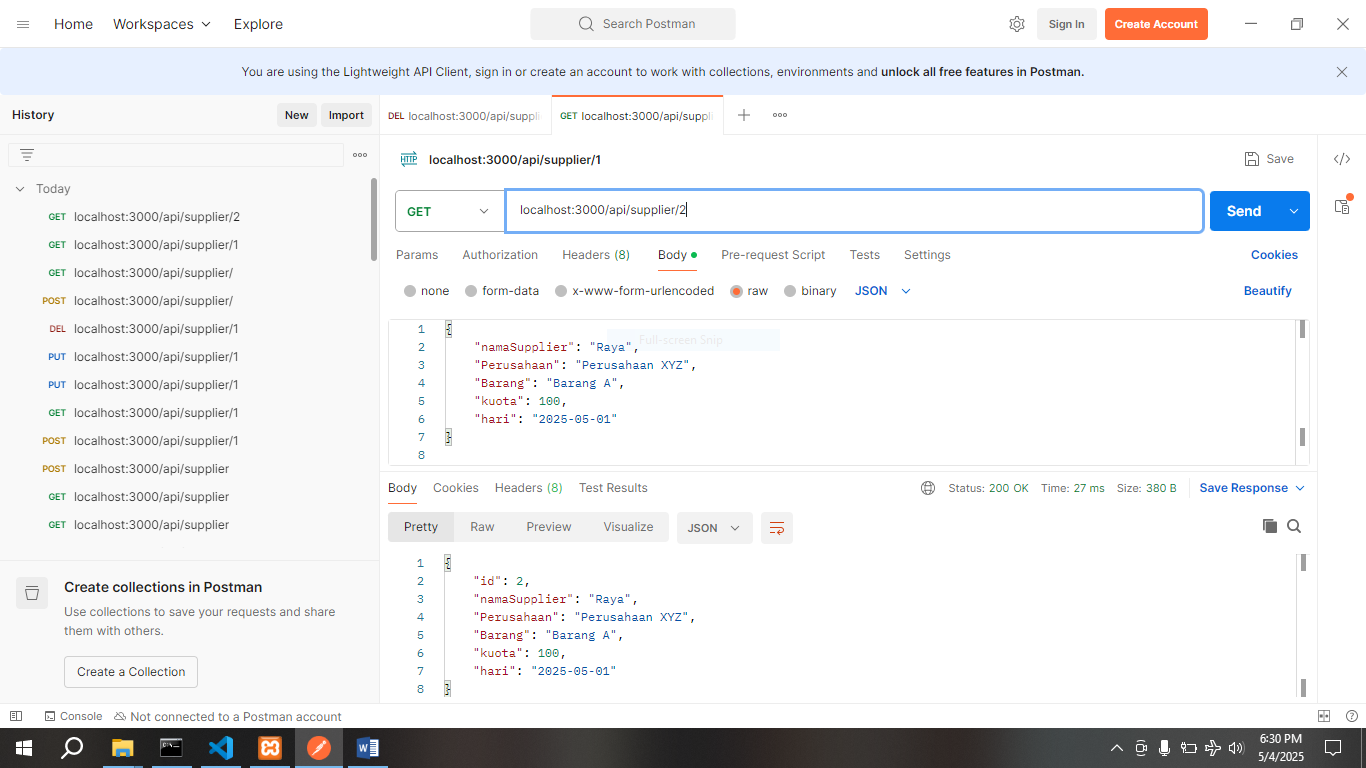


# HASIL TAMPILAN

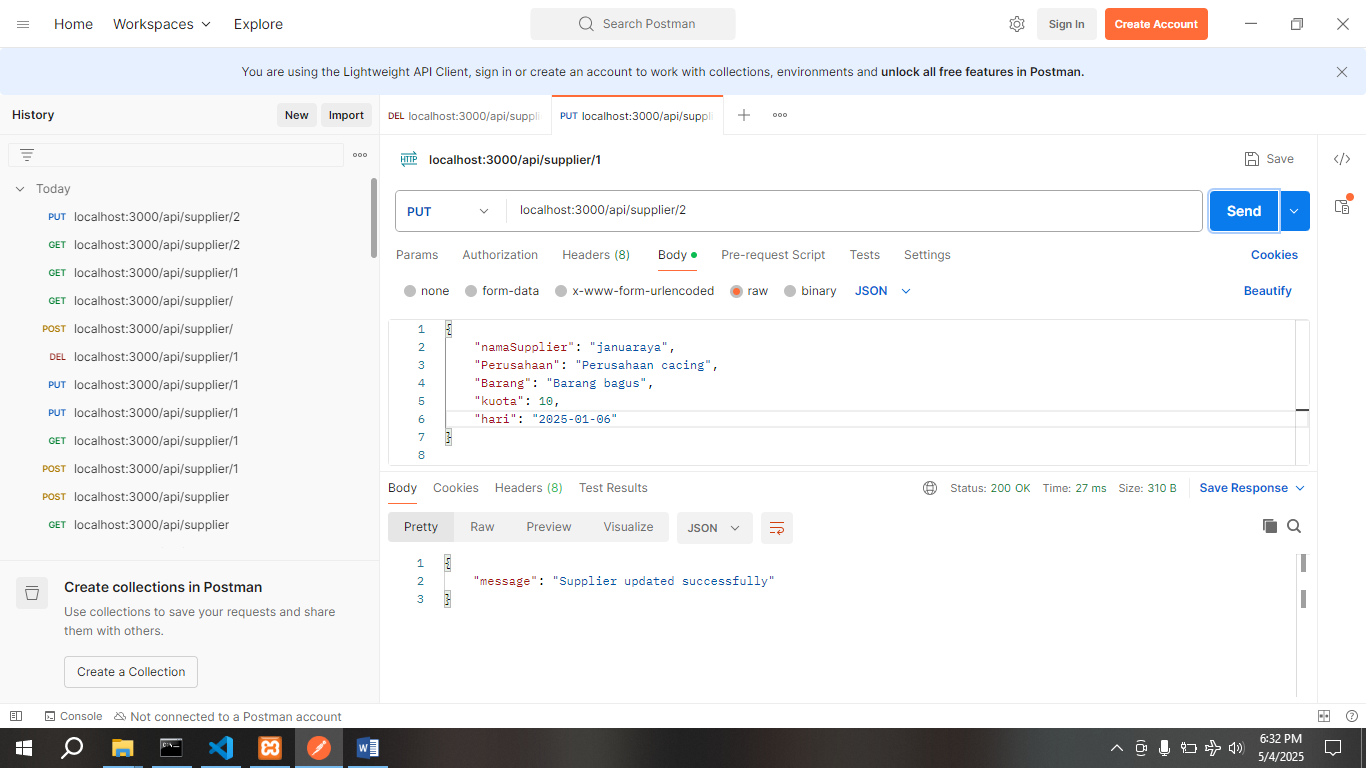
* + - 1. Get menampilkan data



* + - 1. get berdasrkan ID



# tampilan put



* + - 1. tampilan delete

