Tugas Sistem Basis Data

Membuat REST API dengan Node.js melalui URL

Kelompok 12:

* Jonathan Frederick Kosasih (2306225981)
* Bonifasius Raditya Pandu Hendrianto (2306242350)
* Wesley Frederick Oh (23062027630)

Link github ke files (jika diperlukan):

<https://github.com/sleepingpolice-afk/kelompoksbd>

Requirements:

* **npm install pg dotenv**
* **npm install cors**

Cara menghidupkan server:

ketik **node index.js** dalam root folder.

Tools yang digunakan:

* Postman: Untuk testing REST API dalam URL.
* Visual Studio Code: Code Editor untuk JavaScript.
* NeonTech Serverless Postgres: Untuk menunjukkan interaksi antara database non-local dan API.
* Postgresql
* Node.js

**DOKUMENTASI**

Query yang digunakan:

CREATE TYPE categories AS ENUM ('Makanan', 'Minuman', 'Snack', 'Perlengkapan Rumah', 'Perlengkapan Dapur', 'Perlengkapan Kamar Mandi', 'Perlengkapan Tidur', 'Perlengkapan Sekolah', 'Perlengkapan Kantor', 'Perlengkapan Elektronik', 'Perlengkapan Olahraga', 'Perlengkapan Hewan', 'Perlengkapan Bayi', 'Perlengkapan Anak', 'Perlengkapan Dewasa', 'Perlengkapan Lansia', 'Perlengkapan Daur Ulang', 'Perlengkapan Kesehatan', 'Perlengkapan Fashion', 'Perlengkapan Kecantikan', 'Perlengkapan Hobi', 'Perlengkapan Lainnya');

CREATE TYPE status\_pesanan AS ENUM ('Menunggu Pembayaran', 'Diproses', 'Dikirim', 'Selesai', 'Dibatalkan');

CREATE TYPE status\_pembayaran AS ENUM('Menunggu Konfirmasi', 'Diterima', 'Ditolak');

CREATE TYPE methods AS ENUM('Transfer Bank', 'Kartu Kredit', 'E-wallet');

CREATE TABLE produk (

    id\_produk SERIAL PRIMARY KEY,

    nama\_produk VARCHAR(255) NOT NULL,

    kategori categories DEFAULT 'Perlengkapan Lainnya',

    harga BIGINT CHECK(harga >= 0),

    stok INT CHECK(stok >= 0),

    deskripsi TEXT

);

CREATE TABLE users (

    id\_user SERIAL PRIMARY KEY,

    nama VARCHAR(255),

    email VARCHAR(255) UNIQUE,

    password VARCHAR(255),

    alamat TEXT,

    no\_telepon VARCHAR(15)

);

CREATE TABLE pesanan (

    id\_pesanan SERIAL PRIMARY KEY,

    id\_user INT,

    total\_harga BIGINT CHECK(total\_harga >= 0),

    tanggal\_pesanan TIMESTAMP DEFAULT CURRENT\_TIMESTAMP,

    status status\_pesanan DEFAULT 'Menunggu Pembayaran',

    FOREIGN KEY (id\_user) REFERENCES users(id\_user) ON DELETE CASCADE

);

CREATE TABLE rincian\_pesanan (

    id\_rincian SERIAL PRIMARY KEY,

    id\_pesanan INT,

    id\_produk INT,

    jumlah INT CHECK(jumlah >= 0),

    harga BIGINT CHECK(harga >= 0),

    FOREIGN KEY (id\_pesanan) REFERENCES pesanan(id\_pesanan) ON DELETE CASCADE,

    FOREIGN KEY (id\_produk) REFERENCES produk(id\_produk) ON DELETE CASCADE

);

A screenshot of a computer

AI-generated content may be incorrect.

**Penjelasan**

Jadi dalam database transaksi toko online yang kami buat, terdapat 4 buah table, yaitu:

* produk

Table produk digunakan untuk menyimpan informasi tentang produk yang terdapat di toko, mulai dari nama produk, kategorinya, harga, stok, dan deskripsi produk.

* users

Table users, seperti namanya yaitu menyimpan informasi atas user yang teregister dalam database ini. Table ini memiliki beberapa kolom, yaitu nama, email, password, alamat, dan nomor telepon user.

* pesanan

Untuk table pesanan, table ini menyimpan informasi atas pesanan yang dibuat oleh seorang user. Oleh karena itu, table ini memiliki relation dengan table users, serta memiliki kolom seperti total harga, tanggal pesanan, dan status pesanan tersebut apakah sudah dibayar atau belum.

* rincian\_pesanan

Table ini berperan sebagai junction table antara table produk dan table pesanan. Alasan perlu dibuatnya junction table ini adalah karena adanya hubungan many-to-many antara pesanan dan produk, yang artinya sebuah produk bisa dipesan oleh beberapa pesanan sekaligus, dan sebuah pesanan juga bisa memesan beberapa produk sekaligus. Jadi, dibuatlah table rincian\_pesanan yang menyimpan informasi atas pesanan dan produk yang dipesan sehingga table ini memiliki beberapa kolom seperti jumlah, harga, dan beberapa foreign key seperti id\_pesanan dan id\_produk untuk menghubungkan table ini dengan table pesanan dan table produk.

**Struktur Backend**

Struktur penempatan file dan folder adalah seperti berikut:

/SBD\_Kelompok12\_REST API.zip  
├── src/

|  |  |
| --- | --- |
| │  │  │ | ├── controller.js ├── route.js ├── connect.js |

├── .env  
├── SBD\_Kelompok12\_REST API.pdf  
├── query.sql  
├── index.js  
├── package.json  
├── package.lock.json

Keterangan:

* **controller.js**: Adalah file yang berisi seluruh function API yang digunakan untuk memasukkan query postgresql, serta mereturn hasilnya, apakah error ataupun berhasil.
* **route.js**: Adalah file yang menspesifikasikan URL yang digunakan untuk menggunakan function yang bersangkutan dengan URL tersebut. Misalnya route /product/get akan memanggil function getProduct di controller dan kemudian mereturn seluruh isi table produk, ataupun pesanan/update yang memanggil function updatePesanan dan menggunakan query UPDATE untuk memperbarui sebuah pesanan di table pesanan.
* **connect.js**: Adalah file yang menyambungkan seluruh file backend ke database online (NeonTech) dengan menggunakan metode Connection String.
* **.env**: Adalah file yang menyimpan Connection String yang digunakan pada connect.js, serta port yang bersangkutan.
* **index.js**: Adalah file kunci yang menggabungkan kinerja keseluruhan backend. File ini memanggil dependency dan middleware seperti express.js, body parser, cors, pg, dan dotenv. File ini juga melakukan setup pada routes.js sehingga route bisa digunakan saat mengetik URL di postman.
* **package-lock.json dan package.json**: package.json adalah file bawaan dari npm installer, yang menjadi configuration file untuk node.js. Sedangkan package-lock.json adalah file yang menyimpan lokasi beserta banyak infomasi lainnya untuk semua dependency (biasanya di node\_modules) agar nantinya dependency bisa digunakan dalam backend.
* **query.sql**: Adalah file yang berisi seluruh query sql yang digunakan dalam tugas ini.
* **SBD\_Kelompok12\_REST API.pdf**: File dokumentasi ini.

**TESTING API**

Untuk mengetes API, kita di sini menggunakan konsep CRUD, yaitu Create, Read, Update, dan Delete, yang dalam nodejs, kita melihat ini sebagai method POST, GET, PUT, dan DELETE.

Sebenarnya ada banyak sekali function-function CRUD yang digunakan dalam REST API ini. Namun untuk menjaga dokumentasi tetap sederhana, kita akan menunjukkan beberapa contoh saja yang merepresentasikan masing-masing CRUD.

* Create

Contoh : Membuat entry baru di table pesanan

Query SQL : INSERT INTO pesanan (id\_user, total\_harga, status)

VALUES (id, harga, status)

Method : POST

URL : localhost:3000/store/pesanan/add

Contoh output

A screenshot of a computer

AI-generated content may be incorrect.

* Read

Contoh : Membaca seluruh isi table pesanan.

Query SQL : SELECT \* FROM produk

Method : GET

URL : localhost:3000/store/pesanan/get

Contoh output

A screenshot of a computer program

AI-generated content may be incorrect.

* Update

Contoh : Mengupdate salah satu entry produk

Query SQL : UPDATE produk SET nama\_produk = ‘namaproduk’, kategori = ‘kategori’, harga = harga, stok = stok

Method : PUT

URL : localhost:3000/store/product/update/

Contoh output

A screenshot of a computer program

AI-generated content may be incorrect.

Bukti update:

Cek kolom ketiga dimana entry itu diupdate menjadi yang baru.

A black screen with white text

AI-generated content may be incorrect.

* Delete

Contoh : Menghapus salah satu produk

Query SQL : DELETE FROM produk WHERE id\_produk = id

Method : DELETE

URL : localhost:3000/store/product/delete/:id

Contoh output:

A screenshot of a computer program

AI-generated content may be incorrect.

Bukti terhapus:

Perhatikan bahwa produk id 1, yaitu sabun (yang diupdate di bagian Update) telah terhapus.

A black screen with white text

AI-generated content may be incorrect.