**Challenge Chapter 4**

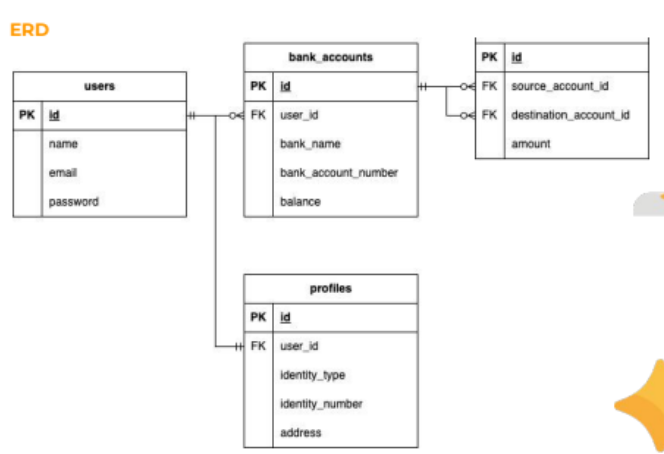
* **Study Kasus:**

Pada challenge kali ini, kamu akan mengintegrasikan Express.js dan Prisma.js ke Dalam "Basic-Banking-System".

* **Ketentuan:**

1. Inisialisasi proyek Express.js dengan menggunakan perintah npm init -y
2. Instal Express.js dan Prisma.js dengan menjalankan perintah npm install express prisma
3. Implementasikan server Express.js dengan beberapa endpoint yang memanfaatkan Prisma.js untuk berinteraksi dengan basis data PostgreSQL yang telah Anda buat pada Challenge 3
4. Contoh endpoint: /accounts untuk mengambil daftar akun, /deposit untuk melakukan deposit, /withdraw untuk melakukan penarikan, dan lainnya
5. Buatlah pull request dari branch feature ke branch main di repositori GitHub

* **ERD**



**bank\_accounts\_transactions**

* **Relations:**

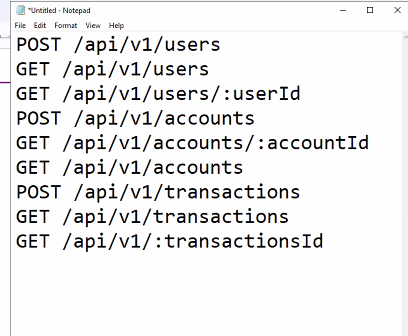
Dari ERD sebelumnya, relasi yang dibentuk seperti berikut:

1. Setiap User dapat memiliki banyak Akun (One-to-Many antara **Users** dan **Bank Accounts**).
2. Setiap Akun hanya dimiliki oleh satu User (Many-to-One antara **Bank Accounts** dan **Users**).
3. Setiap **User**s hanya memiliki satu **Profiles** (One-to-One antara **Users** dan **Profiles**)
4. Setiap **Profile**s hanya dimiliki oleh satu **Users** (One-to-One antara **Profiles** dan **Users**)
5. Setiap Akun dapat memiliki banyak Transaksi (Many-to-Many antara **Bank Accounts** dan

**Bank Accounts** melalui table penampung **Transactions**).

* **Langkah-Langkah:**

1. Setup Express JS.
2. etup Prisma JS.
3. Definisikan Prisma Model berdasarkan ERD yang sudah disediakan.
4. Definisikan relasi model di dalam Prisma Model berdasarkan ERD yang sudah disediakan.
5. Tambahkan beberapa endpoint berikut:



* **POST /api/v1/users** = menambahkan user baru beserta dengan

profilnya.

* **GET /api/v1/users** = menampilkan daftar users.
* **GET /api/v1/users/:userId** = menampilkan detail informasi user

(tampilkan juga profilnya).

* **POST /api/v1/accounts** = menambahkan akun baru ke user yang s

udah didaftarkan.

* **GET /api/v1/accounts** = menampilkan daftar akun.
* **GET /api/v1/accounts/:accountsId** = menampilkan detail akun.
* **POST /api/v1/transactions** = mengirimkan uang dari 1 akun ke akun

lain (tentukan request body nya).

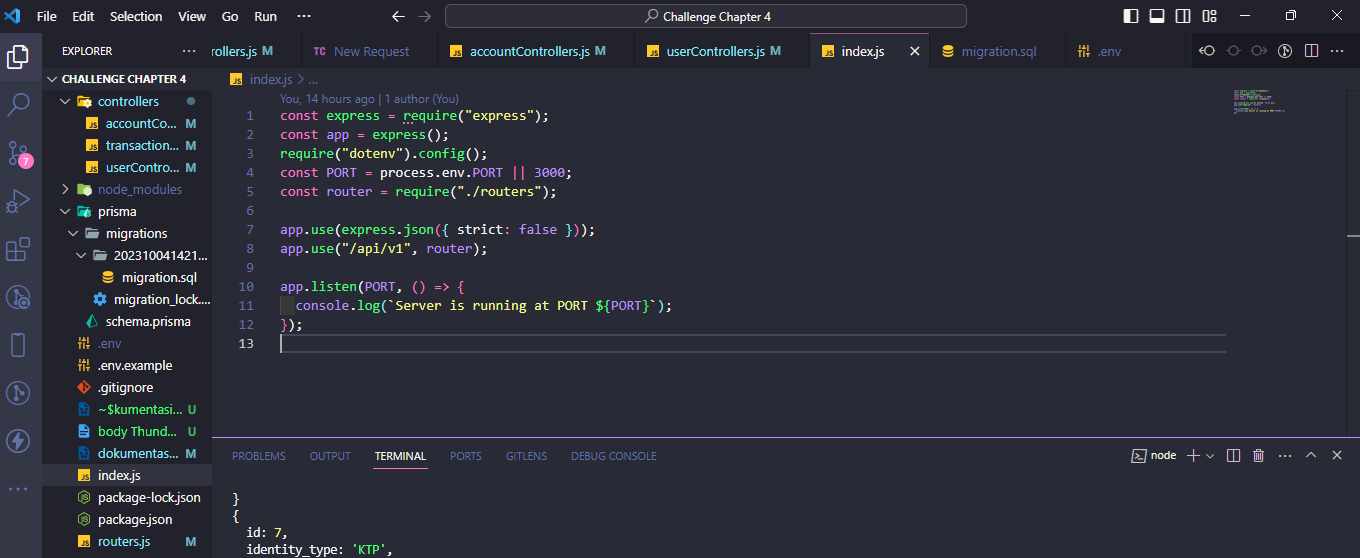
* **GET /api/v1/transactions** = menampilkan daftar transaksi.
* **GET /api/v1/transactions/:transactionId** = menampilkan detail transaksi

(tampilkan juga pengirim dan penerimanya).

1. Tambahkan Update dan Delete pada **users**, **accounts**, dan **transactions.**
2. Buatlah pull request dari branch feature ke branch main di repositori GitHub

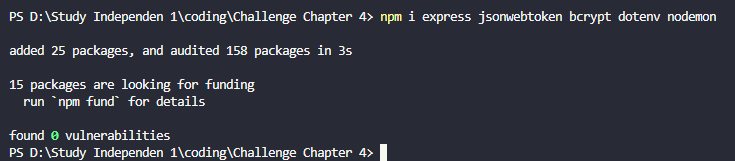


Index.js



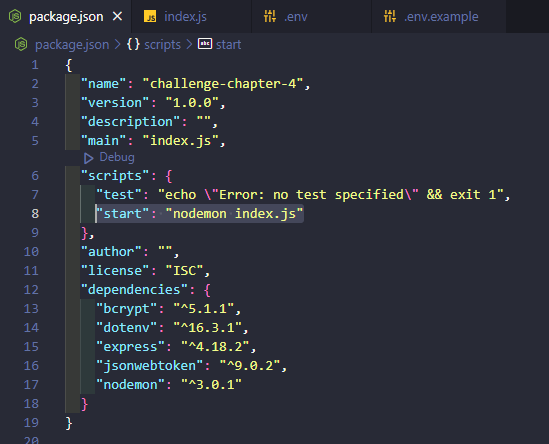
**npm i express jsonwebtoken bcrypt dotenv**

nodemon = agar merefresh sendiri jika merubah kode

****

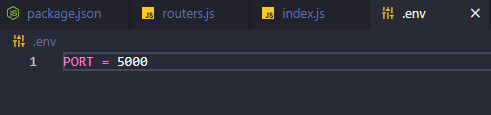
Menambahkan **“start”: “nodemon index.js”**

Atau **“start”: “nodemon (nama\_file.js)”**

****

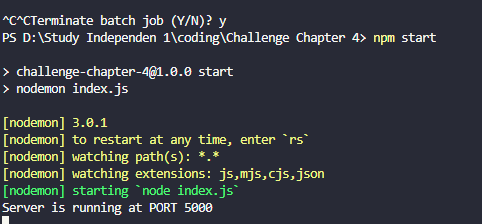
Membuat file **.env**  dan **.env.example**

Di dalam file .env => **PORT = 5000**

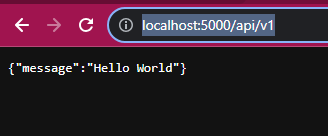
****

**Ctrl + C = Untuk nge-stop**

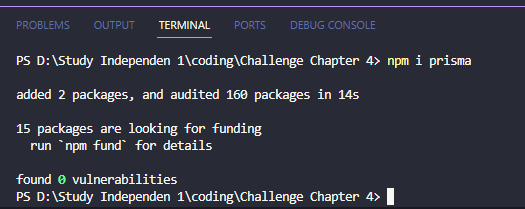
**npm start = untuk memulai**

****

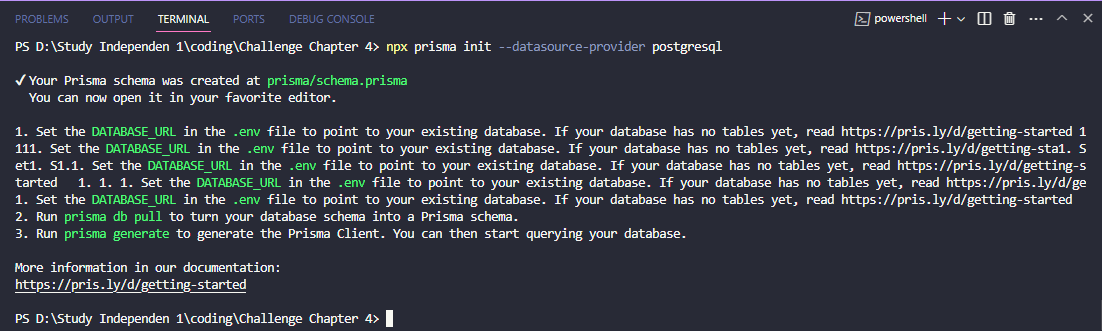
[**http://localhost:5000/api/v1**](http://localhost:5000/api/v1)

****

**npm i prisma = untuk install prisma**

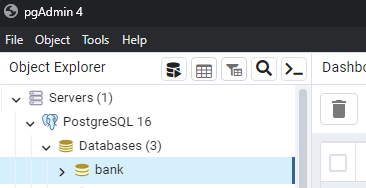
****

**npx prisma init --datasource-provider postgresql**

****

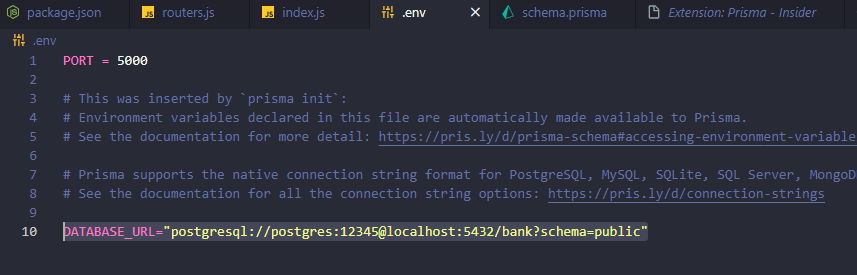
Membuat database = **bank**

Dengan klik kanan pada Database dan Create.

****

Akan muncul di file **.env** yaitu **Database\_URL** **=**

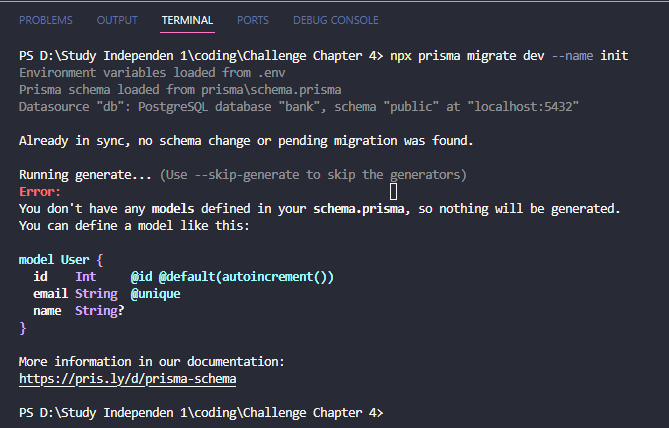
Ubah password menjadi **12345** dan nama database menjadi **bank**

****

**npx prisma migrate dev --name init**

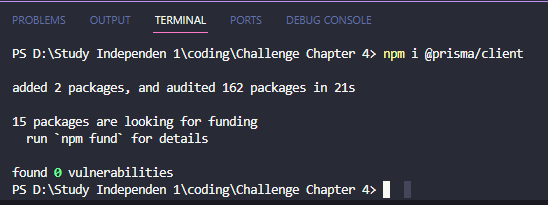
jika terdapat Error: You don’t have any….

Maka eksekusi benar.

****

**npm i @prisma/client**

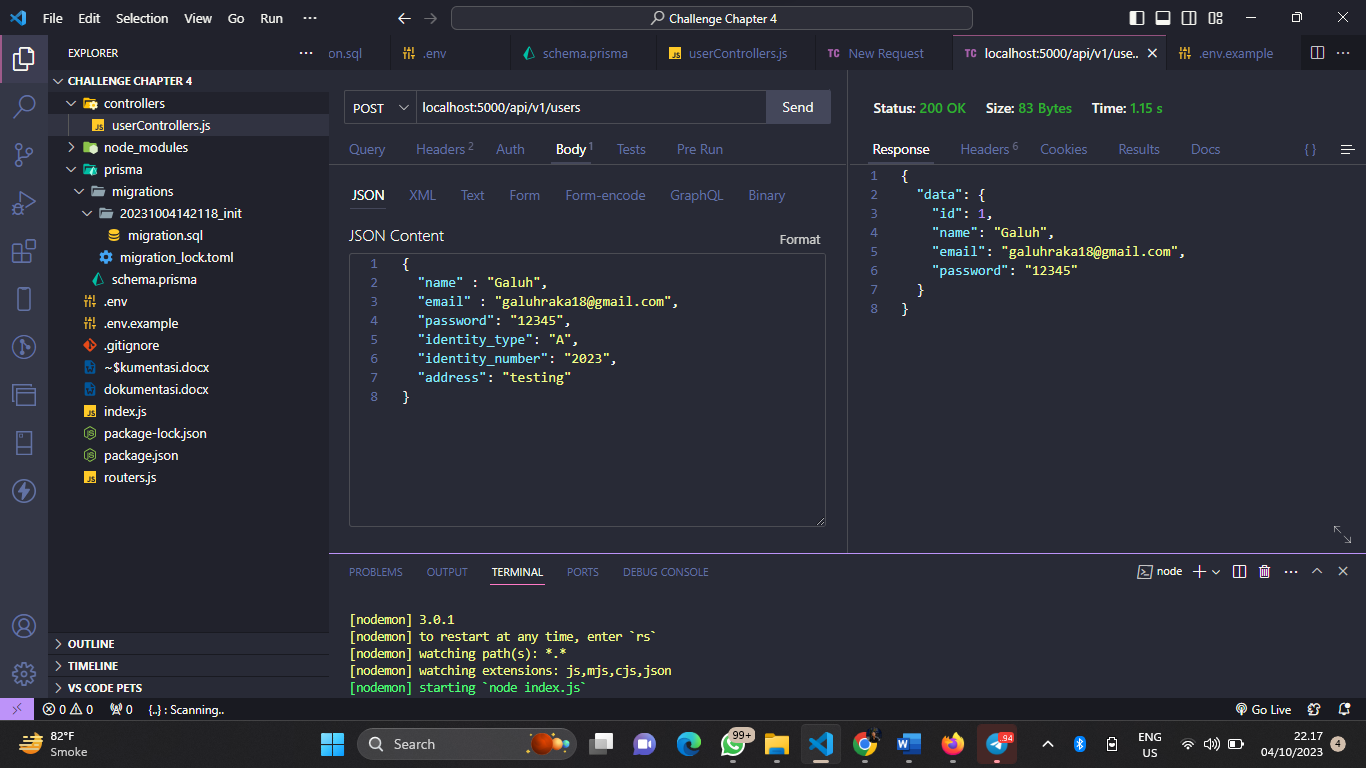
**Untuk koneksikan database client**

****

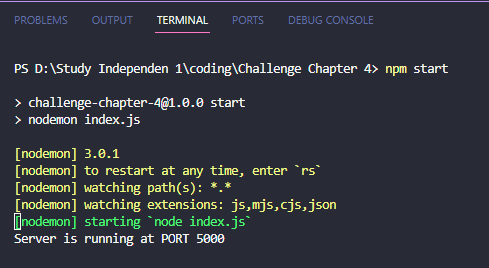
Ketik query pada **body**

Jalankan npm **start**

Klik **send**

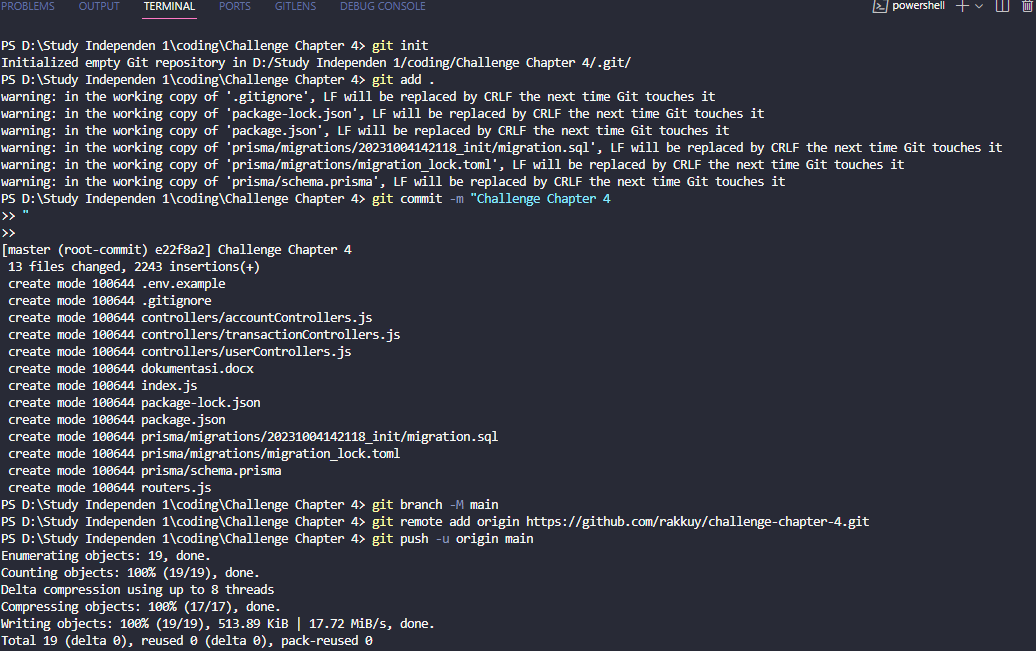
****

**Npm start =>** untuk memulai node.js

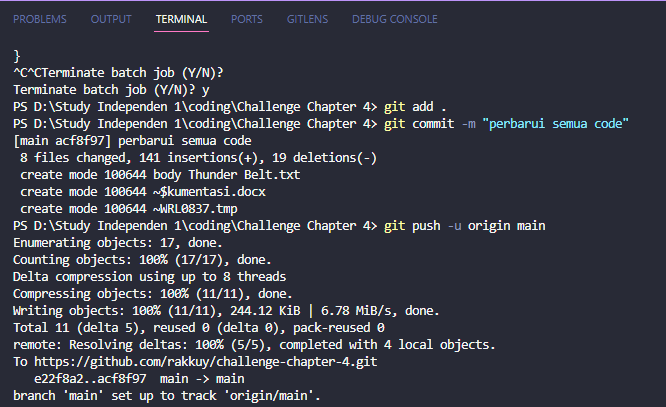
****

**Ctrl + C =>** Untuk close node.js

**Branc ke repository pertama**

****

**Branc ke repository kedua**

****