Pada praktikum kali ini, saya membuat Restful API menggunakan nodeJs dengan memanfaatkan library cors,express, mysql2, dan sequelize. Topik dari project saya adalah users yang terdaftar pada platform e-commerce, alasan saya memilih topik ini karena banyak dari platform e-commerce yang memberikan kategori terhadap penggunanya berdasarkan member yang dimiliki dimana member tersebut bisa mempengaruhi penawaran yang diberikan oleh aplikasi terhadap member.

Topik : Member user aplikasi E-Commerce

Data user terdiri dari :

* Nama,
* Member,
* Gender

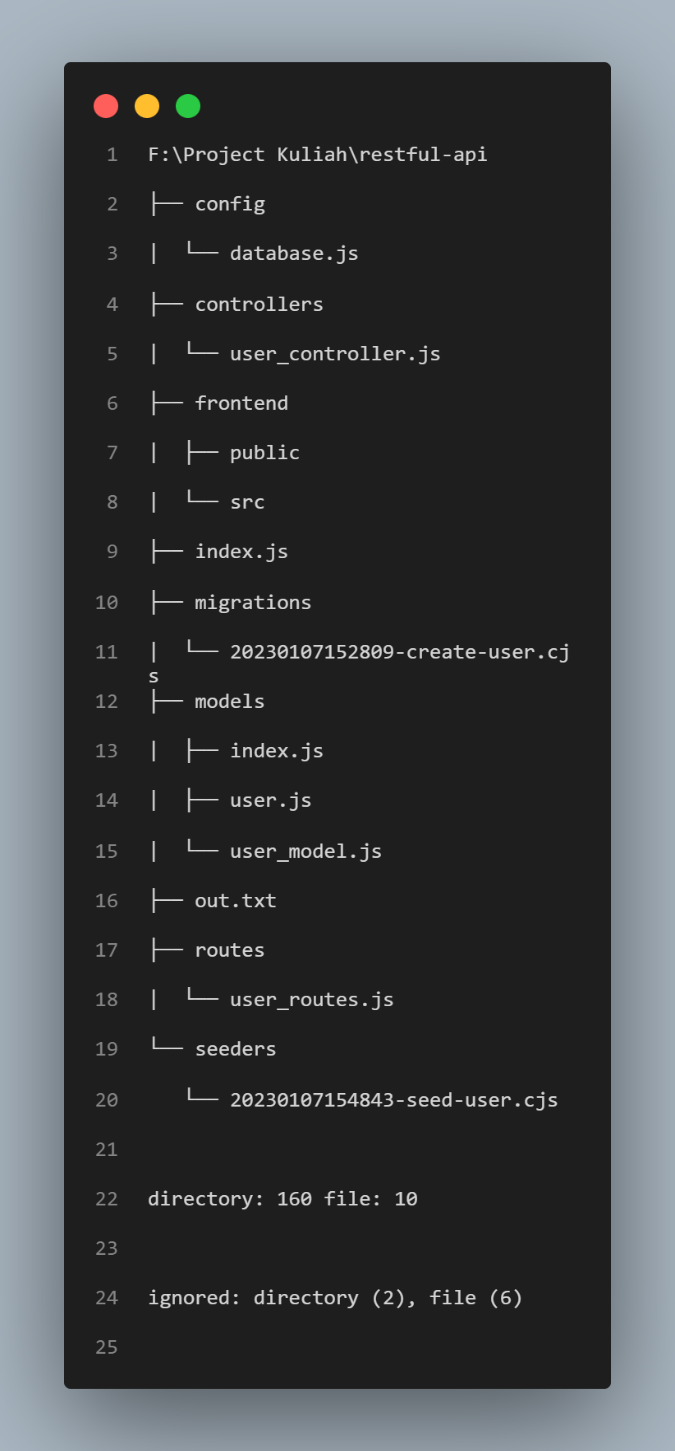
Database yang digunakan : MySql

Framework yang digunakan : Express.js

Package yang digunakan :

1. Backend : cors, express, sequelize, mysql2
2. Frontend : axios, bulma, react-router-dom

Skeleton Design :



Berikut adalah tahapan dalam praktikum yang saya lakukan.

1. **Restful API**

Pertama-tama saya membuat folder untuk menampung program saya. Setelah membuat folder kemudian saya melakukan perintah “npm init” pada terminal untuk membuat file package.json yang akan menyimpan informasi tentang aplikasi pada modul Node.js.

Setelah melakukan initialisasi pada folder tersebut saya kemudian menginstall package yang diperlukan untuk mengembangkan project ini dengan menggunakan perintah dibawah ini pada terminal.

“**npm I express mysql2 sequelize cors”**

Setelah perintah ini selesai dieksekusi maka dapat dilihat hasil install dari perintah tersebut pada file package.json sebagai berikut



**Index.js**

Hal yang pertama saya lakukan adalah menginisiasi server yang akan digunakan Ketika menjalankan project ini. Berikut adalah kode program yang saya gunakan untuk melakukan inisiasi server.



Import statement di baris pertama dan kedua mengimport package express dan cors yang akan digunakan oleh aplikasi. Package express digunakan untuk membuat aplikasi server, sedangkan package cors digunakan untuk mengaktifkan Cross-Origin Resource Sharing (CORS) pada aplikasi. CORS adalah mekanisme yang mengizinkan akses sumber daya dari domain yang berbeda.

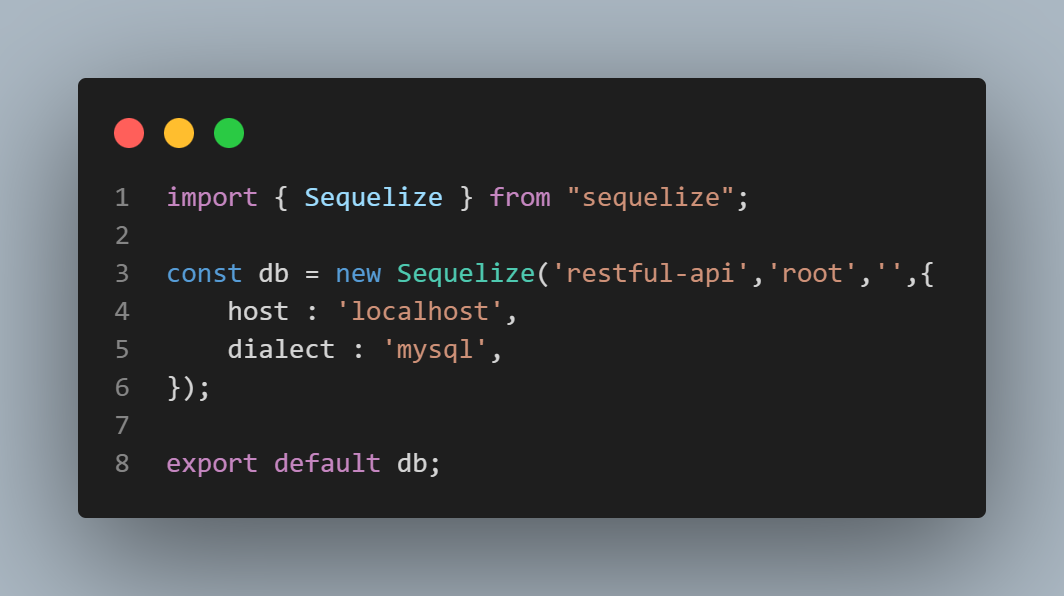
Import statement di baris ketiga mengimport modul UserRoute yang merupakan sebuah objek router yang menangani permintaan HTTP terkait dengan "user". Modul ini berisi definisi rute HTTP seperti GET, POST, PATCH, dan DELETE.

Di baris 5, objek aplikasi express dibuat dan disimpan dalam variabel app. Kemudian, pada baris 6-8, beberapa middleware diterapkan pada objek app. Middleware adalah sebuah fungsi yang dijalankan pada setiap permintaan HTTP sebelum diteruskan ke rute yang sesuai. app.use(cors()) mengaktifkan CORS pada aplikasi, app.use(express.json()) mengaktifkan permintaan HTTP yang berisi data JSON, dan app.use(UserRoute) mengaktifkan rute yang didefinisikan dalam modul UserRoute.

Di baris terakhir, aplikasi server akan mulai mendengarkan permintaan HTTP pada port 5000. Setiap kali ada permintaan HTTP yang masuk, aplikasi akan menjalankan middleware yang diterapkan dan mengarahkan permintaan ke rute yang sesuai.

**Config/database.js**

Setelah itu saya melakukan configurasi terhadap database yang akan digunakan dalam project ini. Berikut adalah kode program yang saya buat untuk melakukan konfigurasi database.



Import statement di baris pertama mengimport kelas Sequelize dari package Sequelize. Kemudian, di baris 3-6, terdapat inisialisasi objek Sequelize yang disimpan dalam variabel db. Objek ini akan menyimpan konfigurasi koneksi ke database yang akan digunakan.

Di baris 3, 'restful-api' adalah nama database yang akan digunakan, 'root' adalah nama pengguna yang akan digunakan untuk login ke database, dan '' adalah password untuk pengguna tersebut. Kemudian, di baris 4-5, terdapat objek yang menentukan host dan dialect yang akan digunakan. 'localhost' menandakan bahwa database tersebut dijalankan pada mesin yang sama dengan aplikasi, dan 'mysql' menandakan bahwa database yang digunakan adalah MySQL.

Di baris terakhir, objek db akan diekspor sebagai modul default yang dapat digunakan oleh aplikasi lain. Ini memungkinkan saya untuk mengakses objek Sequelize yang telah dikonfigurasi tersebut dari modul lain pada project ini.

**Model/user\_model.js**

Setelah itu saya membuat file model yang berfungsi untuk mengelola data yang disimpan dalam tabel tersebut, seperti membuat, membaca, memperbarui, atau menghapus data. Berikut adalah penjelasan kode program yang saya gunakan.



Import statement di baris pertama dan kedua mengimport kelas Sequelize dan objek database yang telah dikonfigurasi sebelumnya. Di baris 4, objek DataTypes dari kelas Sequelize disimpan dalam variabel DataTypes. Objek ini digunakan untuk menentukan tipe data yang akan disimpan dalam tabel.

Di baris 6-15, model User didefinisikan dengan memanggil fungsi define() dari objek database. Fungsi ini menerima nama tabel dan objek yang menentukan struktur tabel. Di dalam objek tersebut, setiap kolom ditentukan dengan menentukan nama kolom dan tipe datanya.

Di baris 17, model User akan diekspor sebagai modul default yang dapat digunakan oleh aplikasi lain.

Di baris 19-21, tabel "users" akan dibuat di database jika belum ada. Fungsi sync() akan memeriksa apakah tabel tersebut sudah ada di database, dan jika belum, tabel tersebut akan dibuat.

**Controller.js**

Setelah membuat mode, saya kemudian membuat file controller yang berfungsi untuk membuat, membaca, memperbarui, atau menghapus data pengguna. Berikut adalah penjelasan kode program yang saya gunakan.

Import statement di baris pertama mengimport model pengguna yang telah didefinisikan sebelumnya. Kemudian, di baris 3-24, terdapat beberapa fungsi yang akan diekspor sebagai modul yang dapat digunakan oleh aplikasi lain.



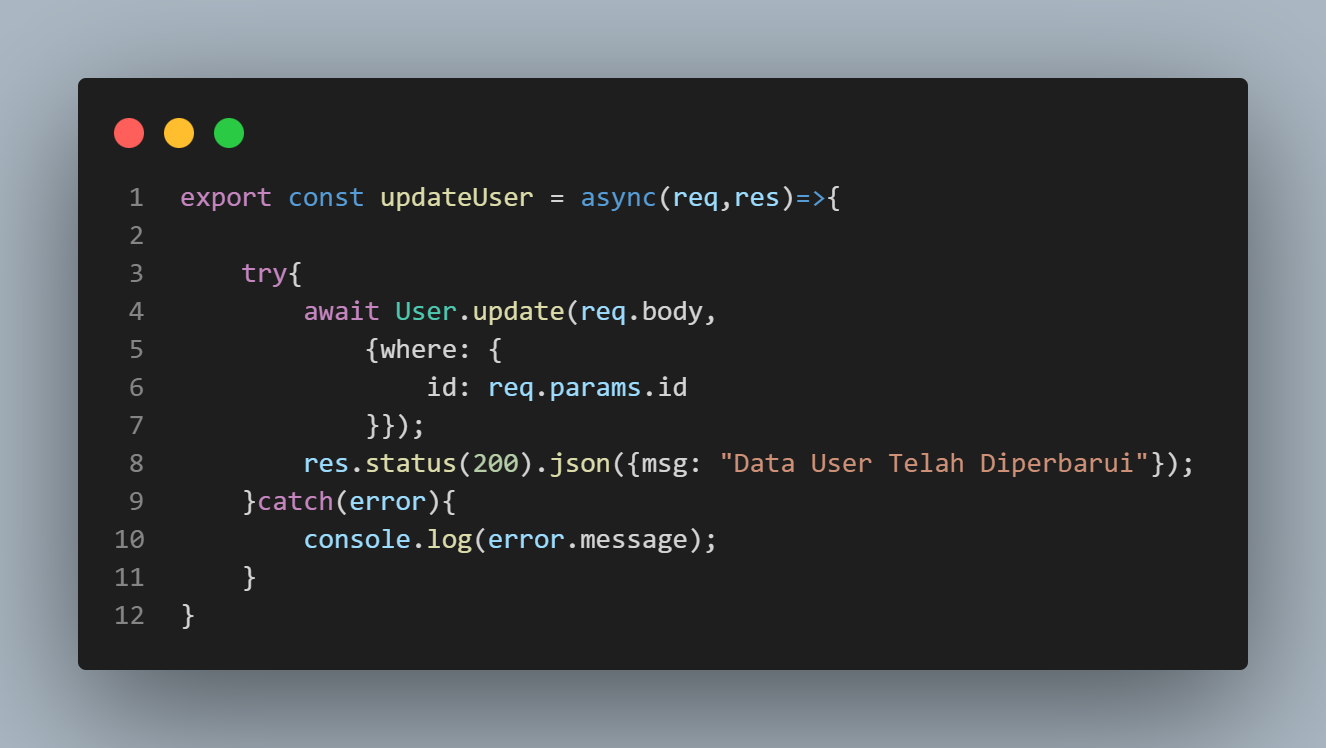
Fungsi pertama adalah getUser(), yang akan mengirim permintaan HTTP GET untuk mendapatkan semua data pengguna dari tabel "users". Fungsi ini akan mengirim jawaban HTTP dengan status 200 dan data pengguna yang ditemukan.



Fungsi kedua adalah getUserById(), yang akan mengirim permintaan HTTP GET untuk mendapatkan data pengguna berdasarkan ID pengguna. Fungsi ini akan mengirim jawaban HTTP dengan status 200 dan data pengguna yang ditemukan.



Fungsi ketiga adalah createUser(), yang akan mengirim permintaan HTTP POST untuk menambahkan data pengguna baru ke tabel "users". Fungsi ini akan mengirim jawaban HTTP dengan status 201 dan pesan yang menyatakan bahwa data pengguna telah berhasil ditambahkan.



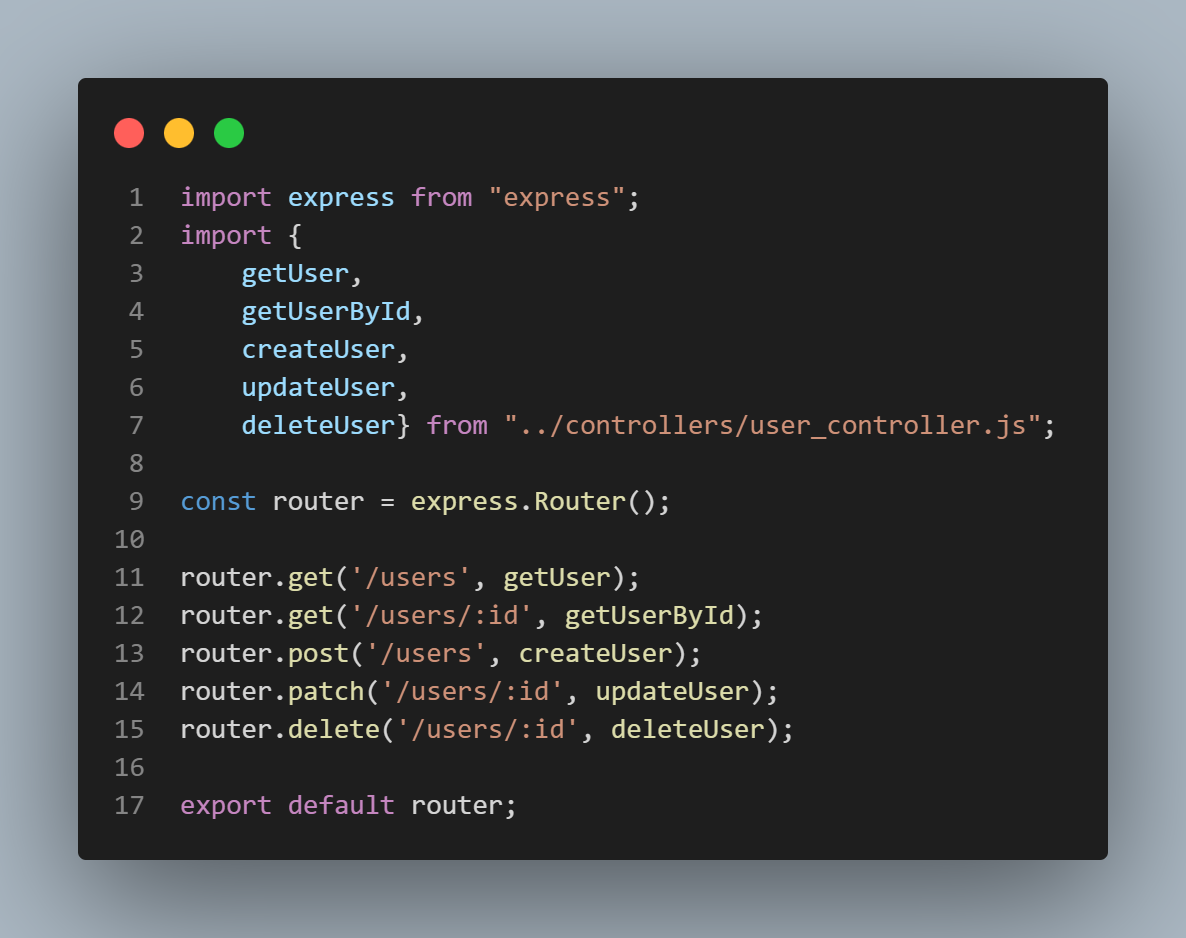
Fungsi keempat adalah updateUser(), yang akan mengirim permintaan HTTP PUT untuk memperbarui data pengguna yang sudah ada di tabel "users". Fungsi ini akan mengirim jawaban HTTP dengan status 200 dan pesan yang menyatakan bahwa data pengguna telah berhasil diperbarui.



Fungsi kelima adalah deleteUser(), yang akan mengirim permintaan HTTP DELETE untuk menghapus data pengguna dari tabel "users". Fungsi ini akan mengirim jawaban HTTP dengan status 200 dan pesan yang menyatakan bahwa data pengguna telah berhasil dihapus.

Semua fungsi di atas menggunakan fungsi async/await untuk menangani operasi asyncronous yang mungkin terjadi saat mengelola data di database. Fungsi async/await memungkinkan kode menjadi lebih mudah dibaca dan dipahami dibandingkan dengan menggunakan callback function atau Promise.

**Route.js**

****

Setealah membuat controller saya membuat file route yang akan mengarahkan permintaan HTTP ke fungsi yang sesuai di controller pengguna yang telah didefinisikan sebelumnya. Berikut adalah penjelasan dari program yang saya gunakan.

Import statement di baris pertama mengimport framework Express.js, sedangkan import statement di baris kedua mengimport fungsi dari controller pengguna. Kemudian, di baris 9, objek router dibuat dengan memanggil fungsi Router() dari framework Express.js.

Di baris 11-15, rute HTTP yang ditangani oleh objek router didefinisikan dengan memanggil fungsi yang sesuai dari objek router. Setiap fungsi akan menerima permintaan HTTP dan mengirim jawaban HTTP kepada client. Misalnya, pada baris 11, rute HTTP GET dengan alamat /users akan diteruskan ke fungsi getUser() dari controller pengguna. Fungsi ini akan menangani permintaan tersebut dan mengirim jawaban HTTP kepada client.

Di baris 17, objek router akan diekspor sebagai modul yang dapat digunakan oleh aplikasi lain.

**Middleware & Seeder**

Untuk memaksimalkan fungsi dari Restful Api yang saya buat, saya menambahkan middleware dan juga seeder. Dalam prosesnya saya harus menignstall package dan menginisiasi package tersebut pada project yang saya kerjakan. Dalam melakukan hal tersebut saya menggunakan perintah berikut pada terminal.

1. **npm install --save-dev sequelize-cli**

Perintah npm install --save-dev sequelize-cli adalah perintah yang digunakan untuk menginstal paket sequelize-cli sebagai dependensi proyek. sequelize-cli adalah paket npm yang menyediakan set perintah command line interface (CLI) yang dapat digunakan untuk mengelola database dari aplikasi Node.js yang menggunakan Sequelize.

Perintah npm install akan mengunduh paket sequelize-cli dari npm registry dan menginstalnya ke dalam proyek. Flag --save-dev akan menambahkan paket sequelize-cli ke dalam daftar dependensi development proyek, sehingga paket tersebut hanya akan diinstal saat mengembangkan aplikasi.

1. **npx sequelize-cli init**

Perintah npx sequelize-cli init adalah perintah yang digunakan untuk menginisialisasi struktur folder dan file yang diperlukan untuk mengelola database dari aplikasi Node.js menggunakan Sequelize. Perintah ini akan membuat folder baru bernama config yang berisi file konfigurasi database, folder baru bernama migrations yang berisi file migrasi database, folder baru bernama models yang berisi file model database, dan file .sequelizerc yang berisi konfigurasi tambahan untuk Sequelize.

Setelah perintah kedua dijalankan maka folder migrations, seeders,config, dan models akan otomatis dibuat dalam project yang saya punya yang akan saya sesuaikan dengan studi kasus yang saya pilih. Berikut adalah detail dari penyesuaian yang saya lakukan.

**20230107152809-create-user.cjs**



Kode di atas digunakan oleh sequelize-cli untuk membuat, memperbarui, atau menghapus tabel "users" dari database. Berikut adalah penjelasan dari program yang saya gunakan

Pada baris pertama, terdapat statement use strict yang akan mengaktifkan modus strict di dalam file ini. Modus strict akan membatasi beberapa aksi yang tidak aman dan membuat kode menjadi lebih tepat.

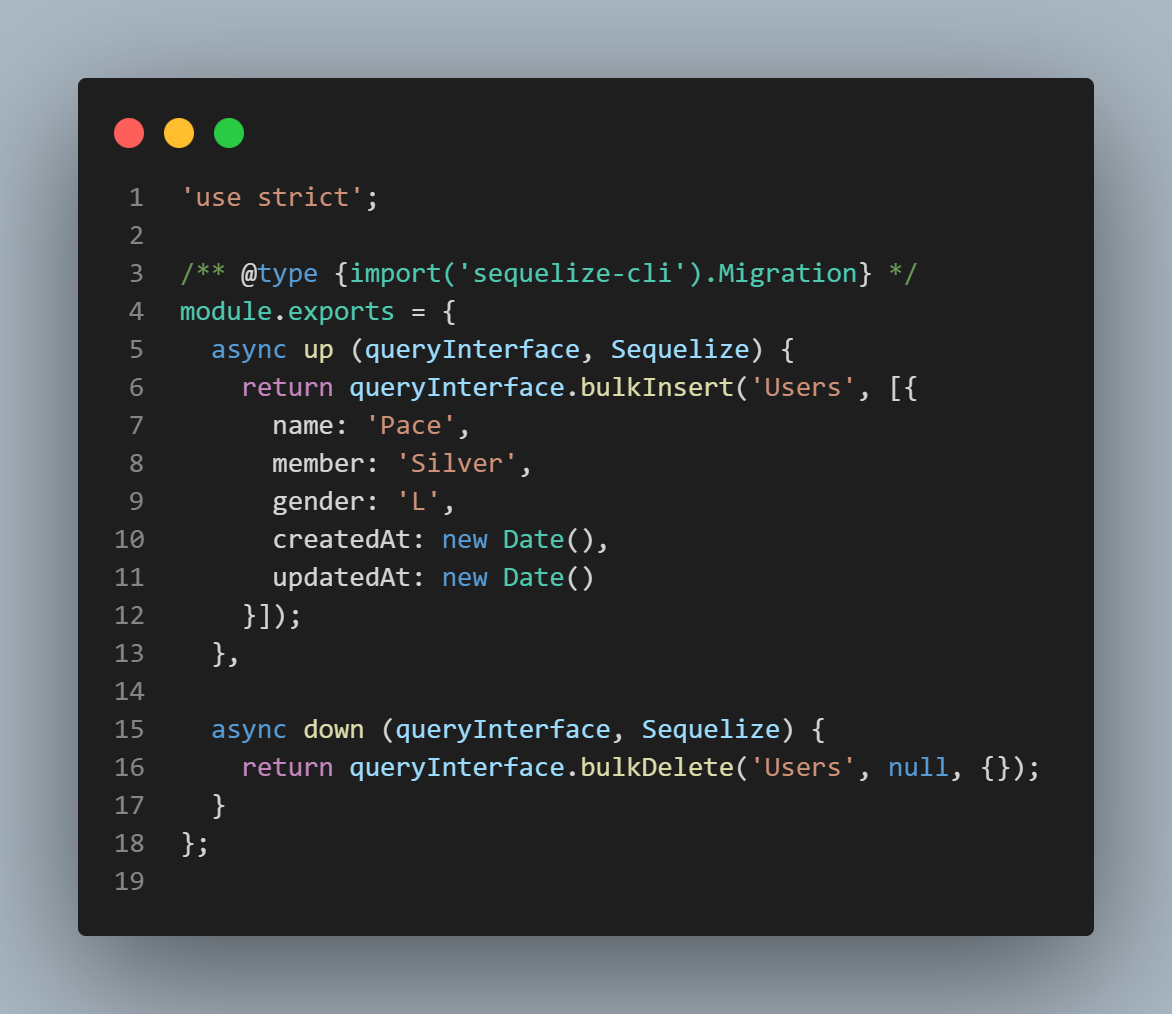
Di baris 2-3, terdapat sebuah block comment yang menyatakan bahwa file ini merupakan sebuah file migrasi database yang akan diimport oleh sequelize-cli.

Di baris 4-34, terdapat sebuah objek yang menyimpan dua fungsi, yaitu up() dan down(). Fungsi up() akan dijalankan saat file migrasi ini dijalankan menggunakan perintah sequelize-cli, sedangkan fungsi down() akan dijalankan saat file migrasi ini di-revert menggunakan perintah sequelize-cli.

Fungsi up() akan memanggil fungsi createTable() dari queryInterface yang telah diberikan oleh sequelize-cli untuk membuat tabel baru bernama "users" di database. Fungsi createTable() akan menerima sebuah objek yang menyatakan struktur tabel dan atribut yang diinginkan. Pada kode di atas, tabel "users" akan memiliki lima atribut, yaitu id, name, member, gender, dan createdAt. Atribut id akan ditetapkan sebagai primary key dan akan di-auto increment saat data baru ditambahkan ke dalam database.

Fungsi down() akan memanggil fungsi dropTable() dari queryInterface yang telah diberikan oleh sequelize-cli untuk menghapus tabel "users" dari database.

**20230107154843-seed-user.cjs**



Kode di atas digunakan oleh sequelize-cli untuk mengisikan data ke dalam tabel "users" di database. Berikut adalah penjelasan mengenai program yang saya gunakan.

Pada baris pertama, terdapat statement use strict yang akan mengaktifkan modus strict di dalam file ini. Modus strict akan membatasi beberapa aksi yang tidak aman dan membuat kode menjadi lebih tepat.

Di baris 2-3, terdapat sebuah block comment yang menyatakan bahwa file ini merupakan sebuah file seeder database yang akan diimport oleh sequelize-cli.

Di baris 4-18, terdapat sebuah objek yang menyimpan dua fungsi, yaitu up() dan down(). Fungsi up() akan dijalankan saat file seeder ini dijalankan menggunakan perintah sequelize-cli, sedangkan fungsi down() akan dijalankan saat file seeder ini di-revert menggunakan perintah sequelize-cli.

Fungsi up() akan memanggil fungsi bulkInsert() dari queryInterface yang telah diberikan oleh sequelize-cli untuk mengisikan data ke dalam tabel "users" di database. Fungsi bulkInsert() akan menerima tiga parameter, yaitu nama tabel, array data yang akan diisikan, dan opsi tambahan. Pada kode di atas, data yang akan diisikan adalah sebuah objek dengan nama "Fulan", member "Silver", gender "L", dan tanggal saat data diisikan.

Fungsi down() akan memanggil fungsi ‘bulkDelete()‘ yang akan menghapus semua data dari tabel "users" di database. Parameter pertama yang diberikan adalah nama tabel, parameter kedua adalah objek yang menyatakan kondisi penghapusan data, dan parameter ketiga adalah opsi tambahan. Pada kode di atas, semua data dari tabel "users" akan dihapus karena parameter kedua diisi dengan null.

Setelah menyesuaikan kedua file tersebut saya menuliskan perintah untuk menjalankan kode program tersebut yaitu

1. **npx sequelize-cli db:migrate** untuk membuat tabel "users" di database.
2. **npx sequelize-cli db:seed:all** untuk mengisikan data ke dalam tabel "users" di database.
3. **ReactJs**

Setelah membuat Restful API saya melanjutkan project saya dengan membuat tampilan frotnend agar proses CRUD dapat dilakukan dengan lebih mudah. Dalam membuat itu saya menggunakan framework ReactJs. Hal pertama yang saya lakukan adalah menginstall dan membuat folder yang akan menampung kode front end yang saya miliki. Saya menjalankan perintah “npx create-react-app frontend” yang akan membuat project ReactJs yang Bernama frontend pada folder project saya.

Setelah itu saya menginstall semua package yang saya perlukan dalam membuat tampilan aplikasi ini dengan menjalankan perintah “npm i react-route-dom axios bulma” yang akan menginstall package react-router-dom, axios, dan bulma untuk styling website.

Selanjutnya, saya melakukan modifikasi terhadap file yang ada pada foler src dengan menghapus semua file yang tidak diperlukan dan menyisakan file App.js dan index.js. Berikut adalah perubahan yang saya lakukan dalam membuat tampilan fontend dari website Restful API.

**App.js**



Kode di atas adalah sebuah contoh implementasi sebuah aplikasi React yang menggunakan pustaka react-router-dom untuk mengelola routing di dalam aplikasi. Kode ini menggunakan tiga komponen yang disediakan oleh pustaka react-router-dom, yaitu BrowserRouter, Routes, dan Route.

Pada baris pertama, terdapat statement import {BrowserRouter, Routes, Route} from "react-router-dom"; yang akan mengimport komponen BrowserRouter, Routes, dan Route dari pustaka react-router-dom.

Di baris 2-4, terdapat statement import UserList from "./components/UserList"; import AddUser from "./components/AddUser"; import EditUser from "./components/EditUser"; yang akan mengimport komponen UserList, AddUser, dan EditUser yang telah dibuat sebelumnya.

Baris 6-16 adalah sebuah fungsi bernama App yang akan mengembalikan sebuah elemen BrowserRouter. Elemen BrowserRouter akan memiliki sebuah elemen Routes sebagai anak elemennya. Elemen Routes akan memiliki tiga elemen Route, masing-masing menyatakan path yang akan dituju dan elemen yang akan ditampilkan saat path tersebut dituju.

Di baris 10, terdapat sebuah elemen Route dengan path "/" yang akan menampilkan komponen UserList saat path yang dituju adalah root ("/").

Di baris 11, terdapat sebuah elemen Route dengan path "add" yang akan menampilkan komponen AddUser saat path yang dituju adalah "/add".

Di baris 12, terdapat sebuah elemen Route dengan path "edit/:id" yang akan menampilkan komponen EditUser saat path yang dituju adalah "/edit/:id", di mana :id merupakan parameter yang akan diisi dengan nilai yang ditentukan.

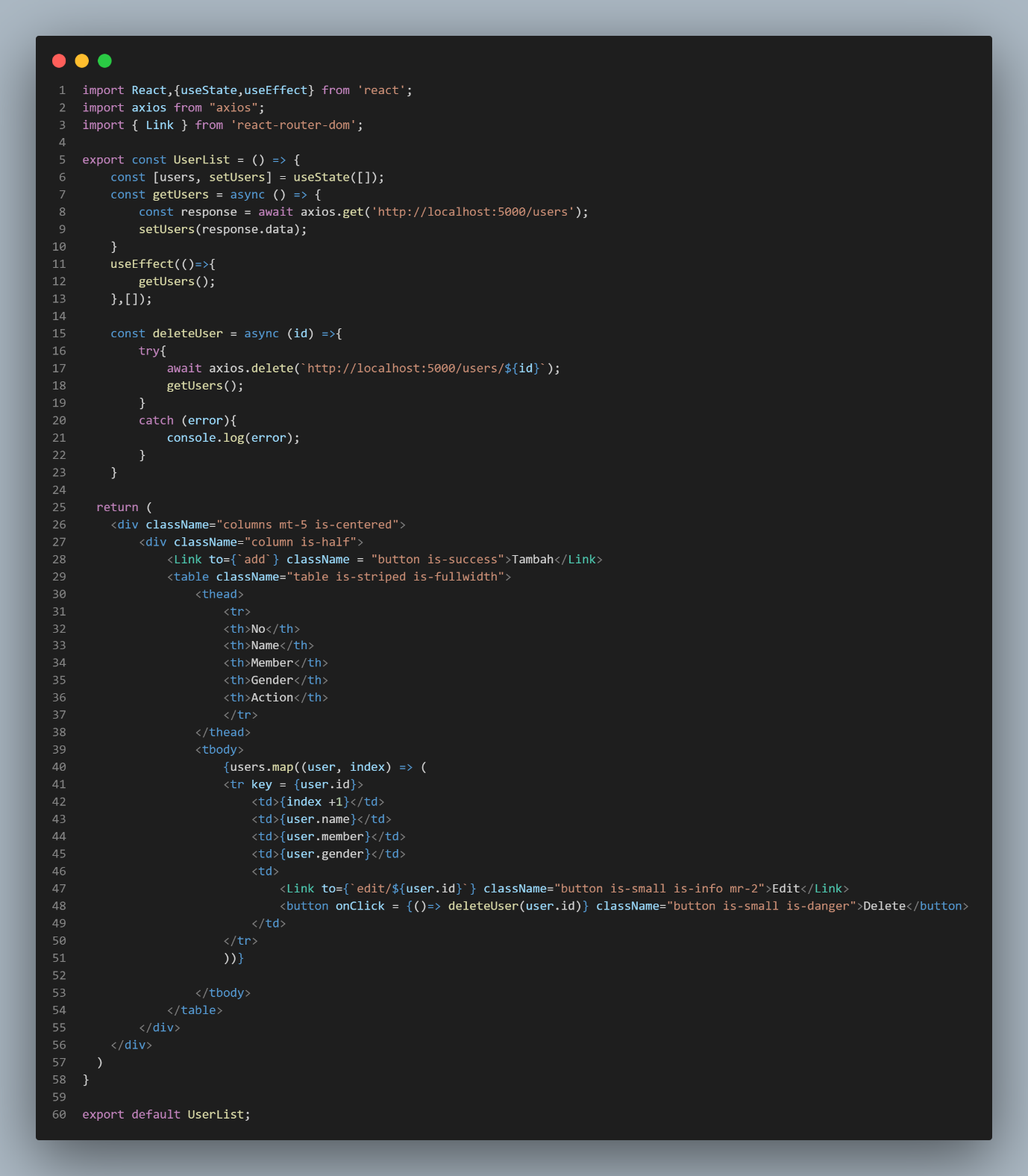
Path yang dituju dan elemen yang akan ditampilkan adalah sebagai berikut:

* Path "/" akan menampilkan komponen UserList
* Path "/add" akan menampilkan komponen AddUser
* Path "/edit/:id" akan menampilkan komponen EditUser

Jadi, saat pengguna mengakses path "/", maka akan ditampilkan daftar user menggunakan komponen UserList. Saat pengguna mengakses path "/add", maka akan ditampilkan form untuk menambah user menggunakan komponen AddUser. Dan saat pengguna mengakses path "/edit/:id", maka akan ditampilkan form untuk mengedit user menggunakan komponen EditUser, di mana :id akan diisi dengan nilai yang ditentukan.

Selanjutnya adalah membuat tampilan interface sesuai dengan fungsinya masing-masing. Tampilan yang pertama kali saya buat adalah tampilan UserList dimana ini akan menampilkan keseluruhan data users yang ada di dalam database.

**UserList.js**



Kode diatas merupakan menampilkan daftar user. Fungsi ini menggunakan react hooks yaitu useState dan useEffect. Pada baris pertama, useState digunakan untuk menyimpan daftar user yang diperoleh dari server ke dalam sebuah variabel bernama users. Kemudian, pada baris ke-2, terdapat sebuah fungsi bernama getUsers yang digunakan untuk mengambil daftar user dari server dengan menggunakan axios. Kemudian, pada baris ke-5, useEffect digunakan untuk menjalankan getUsers ketika component tersebut pertama kali di-render. Pada baris ke-7, terdapat sebuah fungsi Bernama deleteUser yang digunakan untuk menghapus user dari server dengan menggunakan axios.delete. Pada baris ke-18 sampai ke-43, terdapat sebuah table yang menampilkan daftar user yang telah di-render di dalam users. Setiap baris table tersebut terdapat 2 tombol yaitu tombol Edit yang mengarah ke halaman edit user dan tombol Delete yang menghapus user dari server.

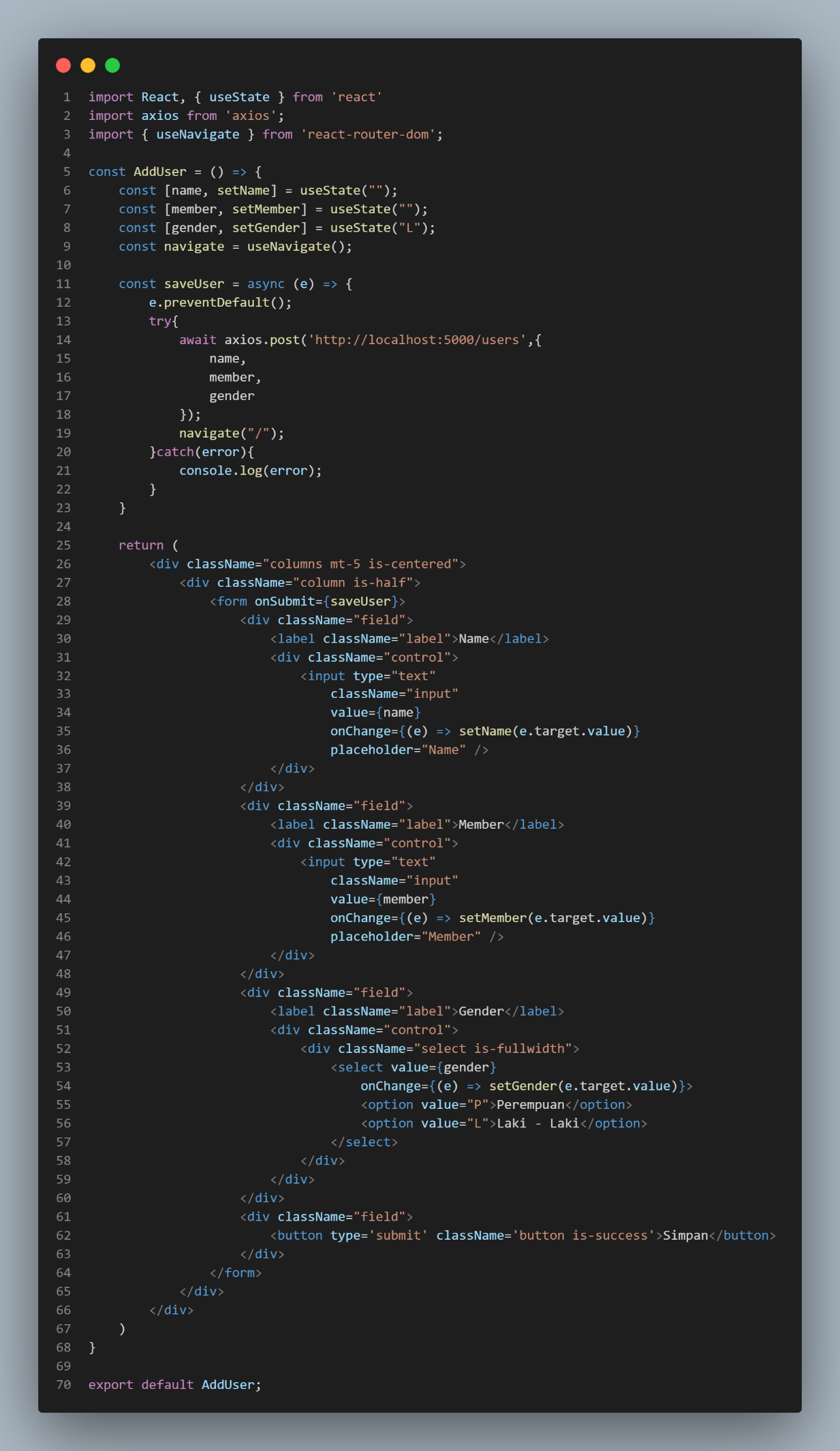
**EditUser.js**



Kode diatas berguna untuk menampilkan form edit user dan mengirimkan data yang diisi pada form ke API server. Pada bagian paling atas, komponen ini mengimport beberapa modul yang dibutuhkan, yaitu useState dan useEffect dari React serta axios dan useNavigate dan useParams dari react-router-dom. Kemudian, komponen ini mendeklarasikan beberapa state yaitu name, member, dan gender yang akan digunakan untuk menyimpan data yang diinput pada form. Kemudian, komponen ini juga mendeklarasikan navigate yang merupakan hook yang digunakan untuk mengarahkan ke halaman lain pada aplikasi, dan id yang merupakan parameter yang dikirim melalui URL.

Setelah itu, komponen ini menggunakan useEffect untuk melakukan pemanggilan getUserById yang akan mengambil data user dari server dengan menggunakan id yang terdapat pada URL saat ini. Kemudian, komponen ini juga mendeklarasikan fungsi updateUser yang akan dipanggil saat form disubmit. Fungsi ini akan mengirimkan data yang diinput pada form ke API server dengan menggunakan axios.patch dan setelah selesai, akan mengarahkan kembali ke halaman UserList menggunakan navigate(“/”). Fungsi getUserById akan dipanggil saat useEffect dijalankan pertama kali, dan akan mengisi data pada form dengan data yang didapat dari server. Pada akhirnya, komponen ini akan mengembalikan sebuah elemen div yang berisi form untuk mengubah data user.

**AddUser.js**



Kode diatas berfungsi untuk menambahkan data user baru ke dalam aplikasi. Pada kode tersebut terdapat beberapa elemen yaitu form untuk mengisi data user, input untuk mengisi nama user, input untuk mengisi member user, dan select untuk menentukan gender user. Ketika form di submit, maka akan memanggil fungsi saveUser yang akan mengirimkan data user baru ke endpoint '/users' menggunakan axios. Kemudian akan mengarahkan ke halaman UserList menggunakan navigate(“/”) dari react-router-dom. Jika terdapat error pada saat mengirim data, maka akan dicatat di console.

1. **Testing**

Pada praktikum Restful API ini saya akan melakukan testing terhadap API menggunakan aplikasi Postman untuk mengecek jalannya program yang sudah saya buat sebelumnya dan mengecek frontend dari project saya pada browser laptop saya. Berikut adalah dokumentasi testing yang telah saya lakukan .

1. Restful API

**getUser**

**[gambar dokumentasi]**

**getUserById**

**[gambar dokumentasi]**

**AddUser**

**[gambar dokumentasi]**

**EditUser**

**[gambar dokumentasi]**

**DeleteUser**

**[gambar dokumentasi]**

1. ReactJS

**getUser**

**[gambar dokumentasi]**

**getUserById**

**[gambar dokumentasi]**

**AddUser**

**[gambar dokumentasi]**

**EditUser**

**[gambar dokumentasi]**

**DeleteUser**

**[gambar dokumentasi]**