

LAPORAN PRAKTIKUM
PEMROGRAMAN BERBASIS FRAMEWORK

Pertemuan Minggu-9

Modul 9 – Global API



NAMA : DINA RISKY ALIN SAPUTRI
NIM : 1841720016
KELAS : 3F

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG

2021

Praktikum 1

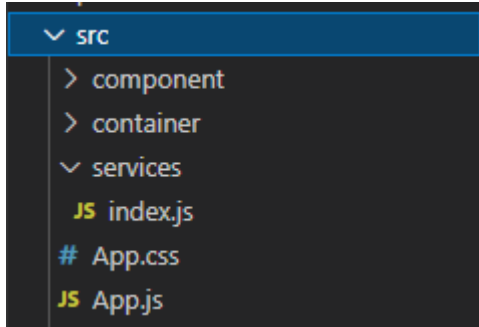
Global API service GET

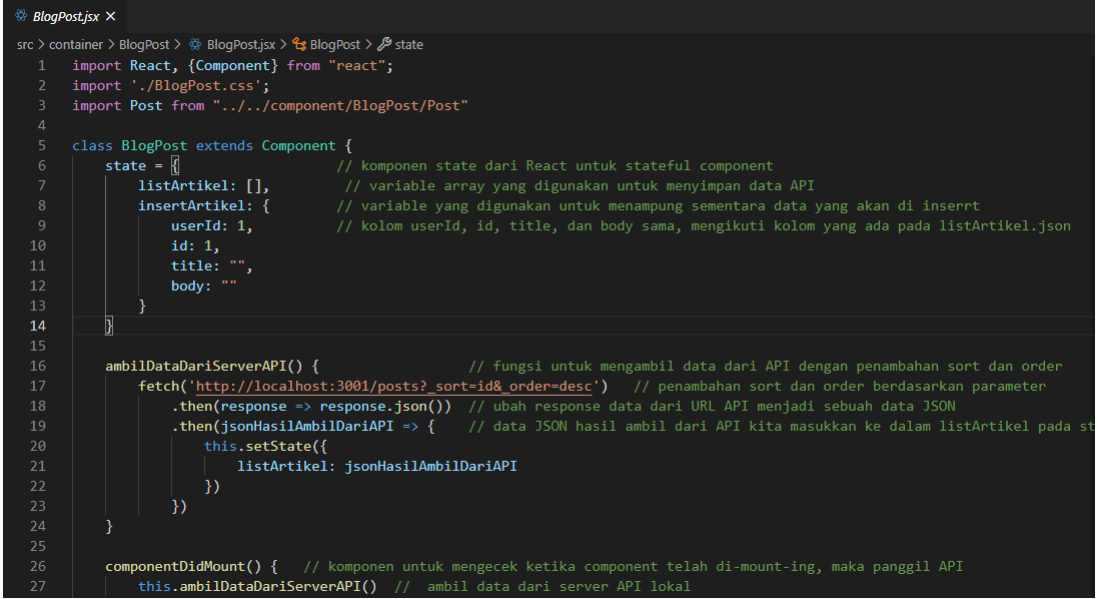
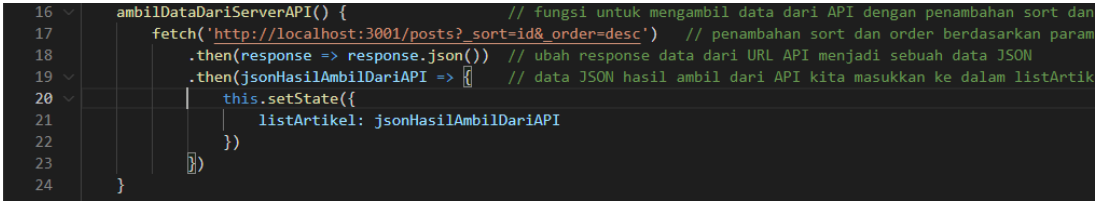
1.1 Run Project dan Server Fake API

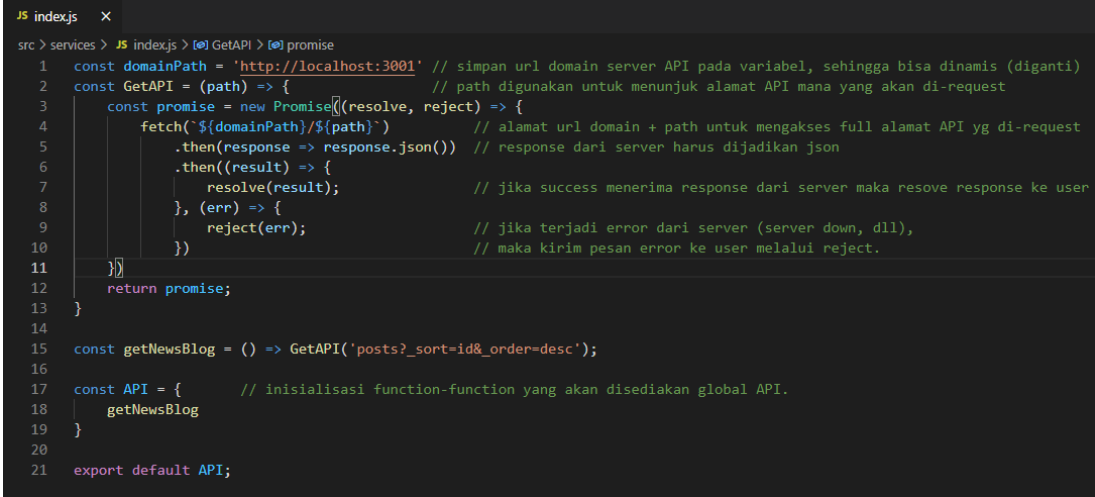
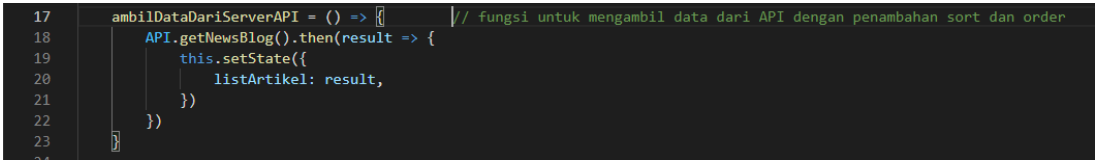
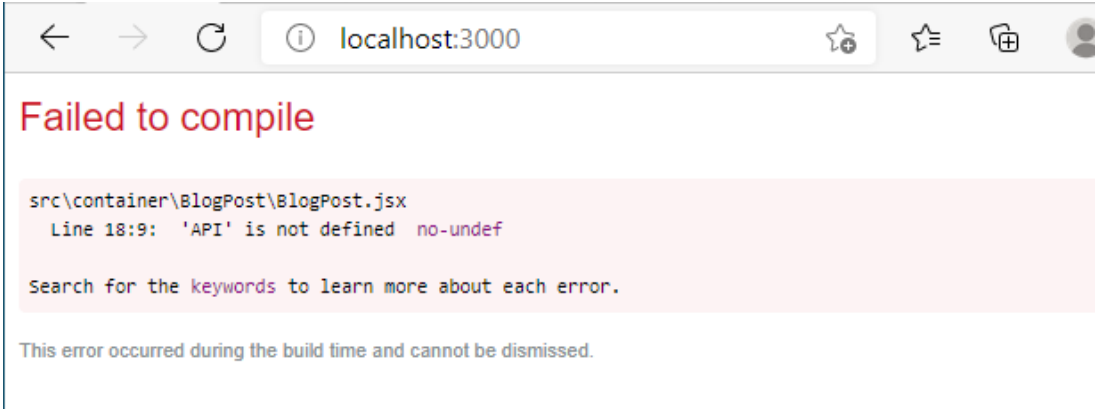
Sebelum memulai praktikum, kita mulai terlebih dahulu menyiapkan source hasil dari praktikum Modul 4 dan menjalankan server project dan server Fake API tersebut, yaitu buka 2 jendela command prompt (CMD) pada project yang sudah kita buat (pada Modul 4). Jendela pertama kita ketikkan perintah `npm start` untuk menjalankan local server project, dan jendela CMD kedua kita isikan perintah `json-server --watch listArtikel.json --port 3001` untuk menjalankan server fake API.

1.2 Langkah Praktikum

Dalam pembuatan Global Service API ini kita akan memerlukan tempat untuk mengumpulkan resource API yang ada, maka kita akan membuat tempat untuk menampung resource tersebut.

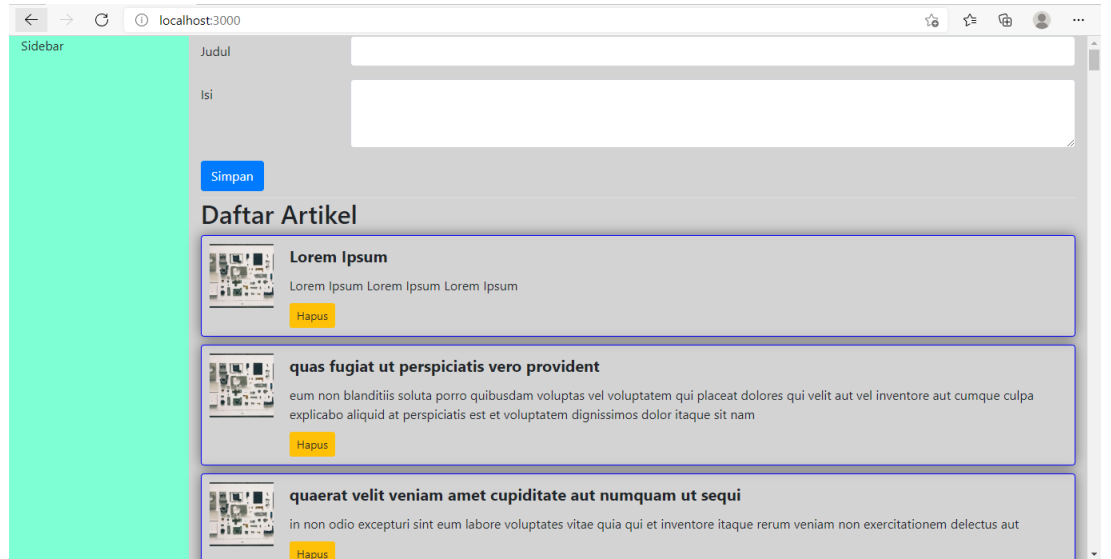
Langkah	Keterangan
1	<p>Buat folder baru bernama “Services” dalam folder src, kemudian buat file <code>index.js</code> seperti pada Gambar 1.1</p>  <p>Gambar 1.1. Pembuatan folder dan file untuk Global Service API</p>
2	<p>Buka file <code>BlogPost.jsx</code> pada folder container (statefull component) dan akan tampil kode program seperti Gambar 1.2.</p>

	 <pre> BlogPost.jsx src > container > BlogPost > BlogPost.jsx > state 1 import React, {Component} from "react"; 2 import './BlogPost.css'; 3 import Post from "../../component/BlogPost/Post" 4 5 class BlogPost extends Component { 6 state = { 7 listArtikel: [], // komponen state dari React untuk stateful component 8 insertArtikel: { // variable array yang digunakan untuk menyimpan data API 9 userId: 1, // variable yang digunakan untuk menampung sementara data yang akan di insert 10 id: 1, // kolom userId, id, title, dan body sama, mengikuti kolom yang ada pada listArtikel.json 11 title: "", 12 body: "" 13 } 14 } 15 16 ambilDataDariServerAPI() { // fungsi untuk mengambil data dari API dengan penambahan sort dan order 17 fetch('http://localhost:3001/posts? sort=id& order=desc') // penambahan sort dan order berdasarkan parameter 18 .then(response => response.json()) // ubah response data dari URL API menjadi sebuah data JSON 19 .then(jsonHasilAmbilDariAPI => { // data JSON hasil ambil dari API kita masukkan ke dalam listArtikel pada state 20 this.setState({ 21 listArtikel: jsonHasilAmbilDariAPI 22 }) 23 }) 24 } 25 26 componentDidMount() { // komponen untuk mengecek ketika component telah di-mount-ing, maka panggil API 27 this.ambilDataDariServerAPI() // ambil data dari server API lokal </pre> <p>Gambar 1.2. Isi kode program BlogPost</p>
3	<p>Pada fungsi ambilDataDariServerAPI (baris16) terdapat pemanggilan API GET untuk merequest data artikel. Proses dari fungsi inilah yang akan kita manage kedalam satu tempat yaitu index.js.</p>  <pre> 16 ambilDataDariServerAPI() { // fungsi untuk mengambil data dari API dengan penambahan sort dan 17 fetch('http://localhost:3001/posts? sort=id& order=desc') // penambahan sort dan order berdasarkan param 18 .then(response => response.json()) // ubah response data dari URL API menjadi sebuah data JSON 19 .then(jsonHasilAmbilDariAPI => { // data JSON hasil ambil dari API kita masukkan ke dalam listArtik 20 this.setState({ 21 listArtikel: jsonHasilAmbilDariAPI 22 }) 23 }) 24 } </pre>
4	<p>Buka file index.js pada folder service, yang telah kita buat tadi dan tuliskan baris kode seperti Gambar 1.3 berikut</p>

	 <pre> 1 const domainPath = 'http://localhost:3001' // simpan url domain server API pada variabel, sehingga bisa dinamis (diganti) 2 const GetAPI = (path) => { // path digunakan untuk menunjuk alamat API mana yang akan di-request 3 const promise = new Promise((resolve, reject) => { 4 fetch(`\${domainPath}/\${path}`) // alamat url domain + path untuk mengakses full alamat API yg di-request 5 .then(response => response.json()) // response dari server harus dijadikan json 6 .then((result) => { 7 resolve(result); // jika success menerima response dari server maka resolve response ke user 8 }, (err) => { 9 reject(err); // jika terjadi error dari server (server down, dll), 10 }) // maka kirim pesan error ke user melalui reject. 11 }) 12 return promise; 13 } 14 15 const getNewsBlog = () => GetAPI('posts?_sort=id&_order=desc'); 16 17 const API = { // inisialisasi function-function yang akan disediakan global API. 18 getNewsBlog 19 } 20 21 export default API; </pre> <p>Gambar 1.3. kode program index.js</p>
5	<p>Kembali ke file BlogPost.js. Ganti baris 17 sampai baris 23 menjadi seperti Gambar 1.4</p>  <pre> 17 ambilDataDariServerAPI = () => { // fungsi untuk mengambil data dari API dengan penambahan sort dan order 18 API.getNewsBlog().then(result => { 19 this.setState({ 20 listArtikel: result, 21 }) 22 }) 23 } </pre> <p>Gambar 1.4. edit kode program BlogPost</p>
6	<p>Simpan file BlogPost.js dan index.js tersebut, dan amati apa yang terjadi pada browser kalian.</p>  <p>Failed to compile</p> <pre> src\container\BlogPost\BlogPost.jsx Line 18:9: 'API' is not defined no-undef </pre> <p>Search for the keywords to learn more about each error.</p> <p>This error occurred during the build time and cannot be dismissed.</p> <p>Setelah dilakukan import API</p>

```
JS index.js BlogPost.jsx X
src > container > BlogPost > BlogPost.jsx > BlogPost > ambilDataDariServerAPI
1 import React, {Component} from "react";
2 import './BlogPost.css';
3 import Post from "../../component/BlogPost/Post";
4 import API from "../../services/index";
```

Ketika dijalankan maka hasilnya seperti gambar berikut :



1.3 Pertanyaan Praktikum 1

- Perhatikan file index.js, apa tujuan dibuatnya fungsi GetAPI pada baris 2 dan fungsi getNewsBlog pada baris 16?

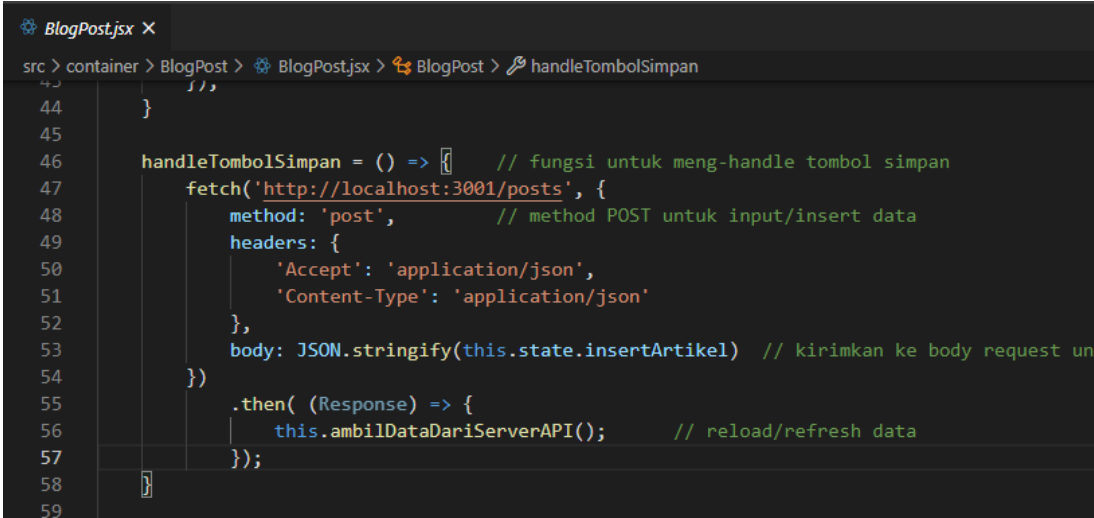
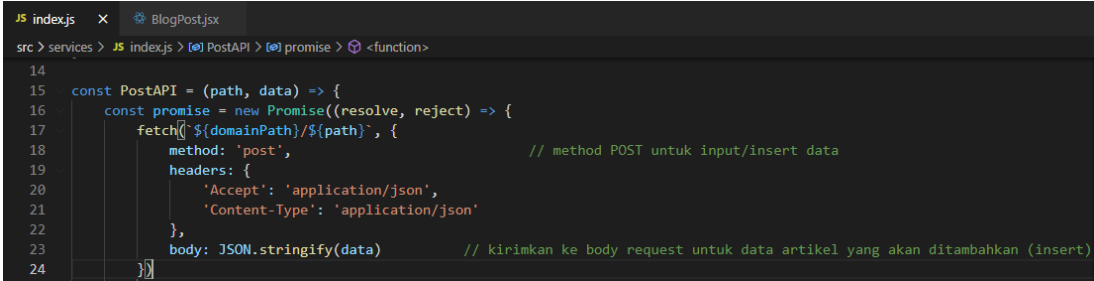
Jawab: Fungsi GetAPI berparameter path digunakan untuk mengakses full alamat API pada alamat url domain dan path yang nanti di request, yang selanjutnya akan di response dari server dalam keadaan harus dijadikan json. Fungsi getNewsBlog digunakan untuk memanggil domainPath yang nantinya akan dilakukan sorting berdasarkan id secara descending.

Praktikum 2

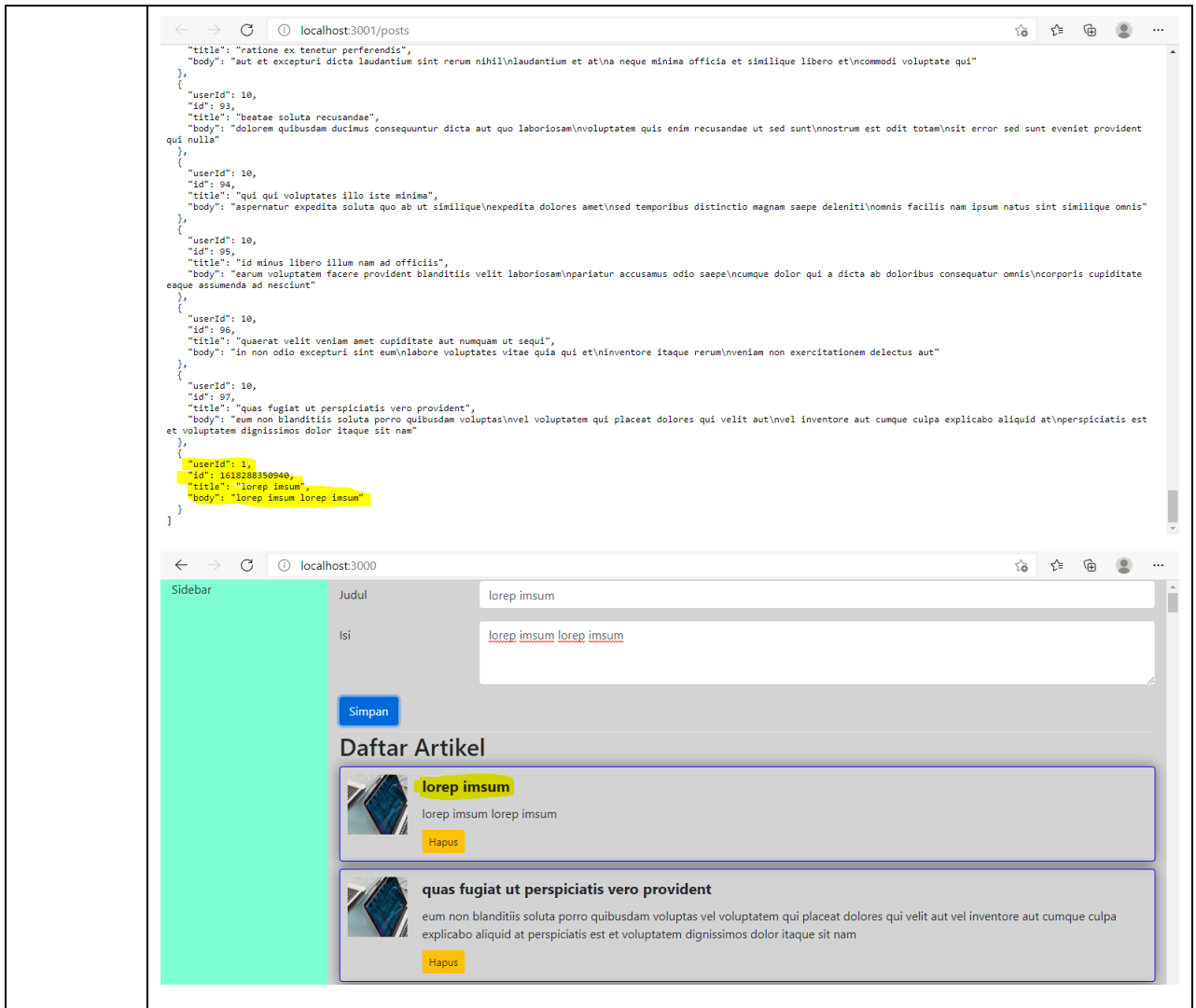
Global API service POST

2.1 Langkah Praktikum 2

Seperti pada langkah praktikum 1 dimana kita me-manage API GET untuk mendapatkan data dari server. Sekarang kita akan me-manage API POST untuk mengirimkan data kepada server API. Langkah-langkahnya adalah

Langkah	Keterangan
1	<p>Kita perhatikan pada fungsi handleTombolSimpan pada file BlogPost.js. Bagian inilah yang selanjutnya kita kelola agar bisa di-manage.</p>  <pre>BlogPost.jsx src > container > BlogPost > BlogPost.jsx > BlogPost > handleTombolSimpan 44 } 45 46 handleTombolSimpan = () => { // fungsi untuk meng-handle tombol simpan 47 fetch('http://localhost:3001/posts', { 48 method: 'post', // method POST untuk input/insert data 49 headers: { 50 'Accept': 'application/json', 51 'Content-Type': 'application/json' 52 }, 53 body: JSON.stringify(this.state.insertArtikel) // kirimkan ke body request un 54 }) 55 .then((Response) => { 56 this.ambilDataDariServerAPI(); // reload/refresh data 57 }); 58 59</pre> <p>Gambar 2.1. Kode program BlogPost fungsi handleTombolSimpan</p>
2	<p>Buatlah fungsi untuk menampung action POST dari file BlogPost.js pada file services/index.js dan inialisasi fungsi tersebut seperti pada Gambar 2.2</p>  <pre>JS index.js src > services > JS index.js > PostAPI > promise > <function> 14 15 const PostAPI = (path, data) => { 16 const promise = new Promise((resolve, reject) => { 17 fetch(`\${domainPath}/\${path}`, { 18 method: 'post', // method POST untuk input/insert data 19 headers: { 20 'Accept': 'application/json', 21 'Content-Type': 'application/json' 22 }, 23 body: JSON.stringify(data) // kirimkan ke body request untuk data artikel yang akan ditambahkan (insert) 24 }) 25 }) 26 }</pre>

	<pre> 25 .then((result) => { 26 resolve(result); 27 }, (err) => { 28 reject(err); 29 }) 30 }) 31 return promise; 32 } 33 34 const getNewsBlog = () => GetAPI('posts?_sort=id&_order=desc'); 35 const postNewsBlog = (dataYgDikirim) => PostAPI('posts', dataYgDikirim); 36 37 const API = { 38 getNewsBlog, 39 postNewsBlog 40 } 41 42 export default API; </pre> <p>Gambar 2.2. Kode program index.js untuk handle POST</p>
3	<p>Selanjutnya pindah ke file BlogPost.js dan ganti isi dari fungsi handleTombolSimpan menjadi seperti Gambar 2.3</p> <pre> 46 handleTombolSimpan = () => { // fungsi untuk meng-handle tombol simpan 47 API.postNewsBlog(this.state.insertArtikel) 48 .then((Response) => { 49 this.ambilDataDariServerAPI(); // reload/refresh data 50 }); 51 } </pre> <p>Gambar 2.3. Kode program pengganti handleTombolSimpan</p>
4	<p>Selesai. Kode program pada BlogPost.js sedikit lebih simple dari sebelumnya.</p>
5	<p>Silahkan kalian jalankan proses input artikel melalui browser dan amati apa yang terjadi.</p> <p>Data yang di inputkan berhasil ditambahkan pada file listArtikel.json dan bisa ditampilkan pada halaman web.</p>



2.2 Pertanyaan Praktikum 2

1. Perhatikan file index.js, apa tujuan dibuatnya fungsi PostAPI dan fungsi postNewsBlog?

Jawab: Fungsi PostAPI digunakan untuk melakukan penginputan data atau penambahan data pada domainPath yang nantinya akan dikirimkan ke body request untuk data artikel yang akan di tambahkan. Fungsi postNewsBlog digunakan untuk mengirim variabel dan data yang nantinya akan dipanggil domainPath.

2. Pada fungsi postNewsBlog, terdapat variable dataYangDiKirim. Apa tujuan dari dibuatnya variable tersebut?

Jawab: Variabel dataYgDiKirim pada postNewsBlog digunakan untuk menampung data sementara dari yang ditambahkan oleh user.


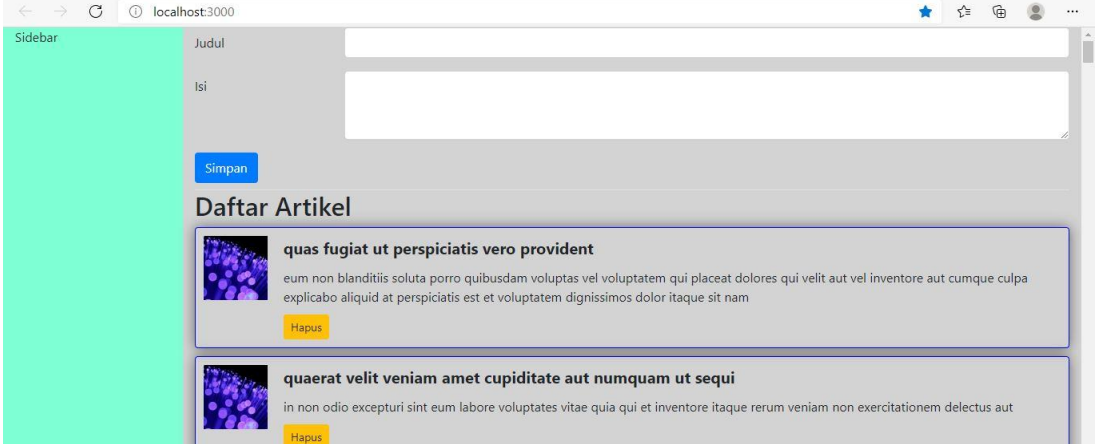
Praktikum 3

Global API service DELETE

3.1 Langkah Praktikum 3

Seperti pada langkah praktikum 2 dimana kita me-manage API GET dan POST untuk mendapatkan data dari server. Sekarang kita akan me-manage API DELETE untuk request hapus data pada server API. Langkah-langkahnya adalah

Langkah	Keterangan
1	<p>Kita perhatikan pada fungsi <code>handleHapusArtikel</code> pada file <code>BlogPost.jsx</code>. Bagian inilah yang selanjutnya kita kelola agar bisa di-manage.</p> <pre>29 handleHapusArtikel = (data) => { // function yang meng-handle button action hapus data 30 fetch(`http://localhost:3001/posts/\${data}`, {method: 'DELETE'}) // alamat URL API yang ingin kita HAPUS datanya 31 .then(res => { // ketika proses hapus berhasil, maka ambil data dari server API lokal 32 this.ambilDataDariServerAPI() 33 }) 34 }</pre> <p>Gambar 3.1. Kode program <code>BlogPost</code> fungsi <code>handleHapusArtikel</code></p>
2	<p>Buatlah fungsi untuk menampung action DELETE dari file <code>BlogPost.jsx</code> pada file <code>services/index.js</code> dan inisialisasi fungsi tersebut seperti pada Gambar 3.2</p> <pre>34 const DeleteAPI = (path, data) => { 35 const promise = new Promise((resolve, reject) => { 36 fetch(`\${domainPath}/\${path}/\${data}`, { method: 'DELETE' }) // alamat URL API yang ingin kita HAPUS datanya 37 .then((result) => { 38 resolve(result); // jika success menerima response dari server response ke user 39 }, (err) => { 40 reject(err); // jika terjadi error dari server (server down, dll), 41 }) 42 }) 43 } 44 45 const getNewsBlog = () => GetAPI('posts?_sort=id&_order=desc'); 46 const postNewsBlog = (dataYgDikirim) => PostAPI('posts', dataYgDikirim); 47 const deleteNewsBlog = (dataYgDiHapus) => DeleteAPI('posts', dataYgDiHapus); 48 49 const API = { // inisialisasi function-function yang akan disediakan global API. 50 getNewsBlog, 51 postNewsBlog, 52 deleteNewsBlog 53 } 54 55 export default API;</pre> <p>Gambar 3.2. Kode program <code>index.js</code> untuk handle DELETE</p>
3	<p>Selanjutnya pindah ke file <code>BlogPost.jsx</code> dan ganti isi dari fungsi <code>handleHapusArtikel</code> menjadi seperti Gambar 3.3</p>

	<pre> 29 handleHapusArtikel = (data) => { // function yang meng-handle button action hapus data 30 API.deleteNewsBlog(data); 31 this.ambilDataDariServerAPI() 32 } </pre> <p>Gambar 3.3. Kode program pengganti handleTombolSimpan</p>
4	Selesai. Kode program pada BlogPost.jsx sedikit lebih simple dari sebelumnya.
5	<p>Silahkan kalian jalankan proses hapus artikel melalui browser dan amati apa yang terjadi.</p>  <p>Ketika di klik hapus pada daftar artikel misalnya paling atas sendiri maka daftar artikel dengan judul “lorep imsum” akan terhapus.</p> 

3.2 Pertanyaan Praktikum 3

1. Perhatikan file index.js, apa tujuan dibuatnya fungsi DeleteAPI dan fungsi deleteNewsBlog?

Jawab: Fungsi DeleteAPI digunakan untuk melakukan delete atau penghapusan data pada domainPath. Fungsi deleteNewsBlog digunakan untuk melakukan mapping dengan mengirimkan variabel dataYgDihapus berdasarkan id dari penginputan data ke dalam const DeleteAPI.

2. Pada fungsi deleteNewsBlog, terdapat variable dataYangDiHapus. Apa tujuan dari dibuatnya variable tersebut?

Jawab: Fungsi variabel dataYgDiHapus digunakan untuk menmpung data berupa id dari artikel yang nantinya di hapus pada file BlogPost.jsx.

Praktikum 4

Manage Global API service

4.1 Manage Global API

Pada global API yang telah kita lakukan pada praktikum 1, 2 dan 3. Kita mengetahui bahwa saat kita benar-benar melakukan coding untuk membuat API, kita akan dihadapkan pada banyak url API yang akan kita sediakan. Sehingga kita butuh manage lagi Global API untuk lebih teratur.

Salah satu cara untuk me-manage Global API adalah dengan memisahkan API berdasarkan action-nya (GET, POST, DELETE). Sehingga, missal saat terjadi error pada DELETE API, kita cukup akan membuka Global API DELETE yang akan kita perbaiki.

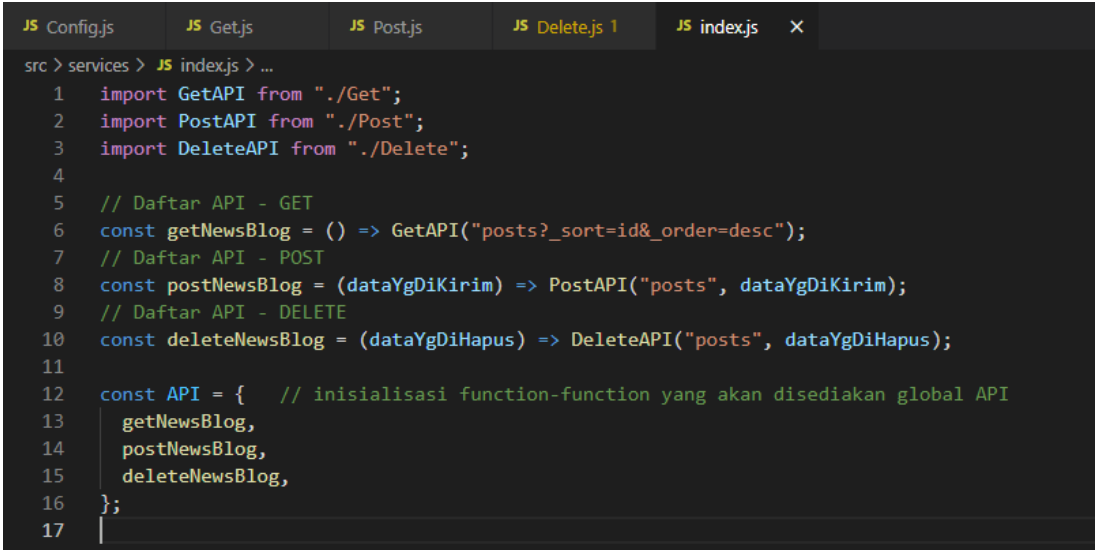
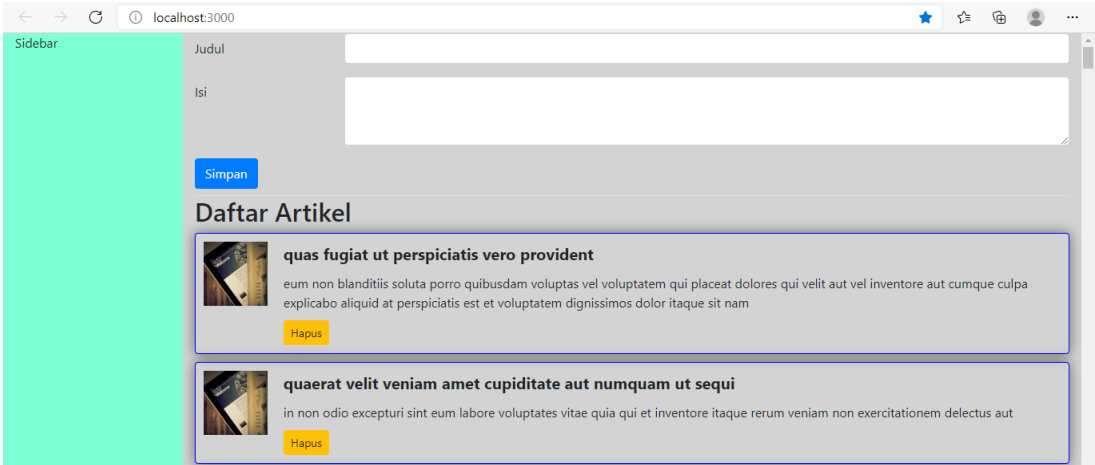
4.2 Langkah Praktikum 4

Langkah-langkah yang dilakukan pada praktikum 4 ini adalah

Langkah	Keterangan
1	Buatlah file Config.js, Get.js, Post.js, dan Delete.js dalam folder services seperti pada Gambar 4.1

	 <p>Gambar 4.1. Buat file</p>
2	<p>Buka Config.js dan isikan kode seperti Gambar 4.2</p>  <p>Gambar 4.2. Kode program config.js</p>
3	<p>Buka Get.js dan isikan kode seperti Gambar 4.3</p>  <p>Gambar 4.3. Kode program Get.js</p>
4	<p>Buka Post.js dan isikan kode seperti Gambar 4.4</p>

	 <pre> 1 import { domainPath } from "../Config"; 2 3 const PostAPI = (path, data) => { 4 const promise = new Promise((resolve, reject) => { 5 fetch(`\${domainPath}/\${path}`, { 6 method: "post", // method post untuk insert data 7 headers: { 8 Accept: "application/json", 9 "Content-Type": "application/json", 10 }, 11 body: JSON.stringify(data), // kirimkan ke body request untuk data artikel yang 12 }) // akan ditambahkan (insert) 13 }).then(14 (result) => { 15 resolve(result); // jika success menerima response dari server maka resolve response ke user 16 }, 17 (err) => { 18 reject(err); // jika terjadi error dari server (server down, dll) 19 } 20); 21 }); 22 return promise; 23 }; 24 25 export default PostAPI; </pre> <p>Gambar 4.4. Kode program Post.js</p>
5	<p>Buka Delete.js dan isikan kode seperti Gambar 4.5</p>  <pre> 1 import { domainPath } from "../Config"; 2 3 const DeleteAPI = (path, data) => { 4 const promise = new Promise((resolve, reject) => { 5 fetch(`\${domainPath}/\${path}/\${data}`, { method: "DELETE" }) // alamat URL API yang ingin kita hapus datanya 6 }).then(7 (result) => { 8 resolve(result); // jika success menerima response dari server maka resolve response ke user 9 }, 10 (err) => { 11 reject(err); // jika terjadi error dari server (server down, dll) 12 } 13); 14 }; 15 16 export default DeleteAPI; </pre> <p>Gambar 4.5. Kode program Delete.js</p>
6	<p>Pada index.js kita edit menjadi seperti Gambar 4.6</p>

	 <pre> src > services > JS index.js > ... 1 import GetAPI from "../Get"; 2 import PostAPI from "../Post"; 3 import DeleteAPI from "../Delete"; 4 5 // Daftar API - GET 6 const getNewsBlog = () => GetAPI("posts?_sort=id&_order=desc"); 7 // Daftar API - POST 8 const postNewsBlog = (dataYgDiKirim) => PostAPI("posts", dataYgDiKirim); 9 // Daftar API - DELETE 10 const deleteNewsBlog = (dataYgDiHapus) => DeleteAPI("posts", dataYgDiHapus); 11 12 const API = { // inisialisasi function-function yang akan disediakan global API 13 getNewsBlog, 14 postNewsBlog, 15 deleteNewsBlog, 16 }; 17 </pre> <p>Gambar 4.6. Kode program index.js</p>
7	<p>Amati apa yang terjadi</p> 

4.3 Pertanyaan Praktikum 4

1. Bagaimana caranya untuk menambahkan daftar API baru baik untuk method GET, POST, DELETE pada Global API?

Jawab: Dengan cara memanggil const yang telah diimport dari file atau kelas lain dan melakukannya passing parameter ketika dibutuhkan baik untuk method POST dan DELETE.

Tugas Praktikum

Ubahlah **Tugas pada Modul 4** yang sudah kalian buat dengan memanfaatkan Global API seperti praktikum yang kita lakukan pada Modul 8 ini.

Langkah	Keterangan
1	<p>Buat folder dengan nama services lalu buatlah file Config.js, Get.js, Post.js, dan Delete.js dalam folder services tersebut.</p> 
	<p>Config.js</p>  <pre>JS Config.js X react-web-tugas > src > services > JS Config.js > ... 1 // file config ini bisa berisi variabel-variabel lain yang dibutuhkan untuk proses API 2 export const domainPath = "http://localhost:3001"; // simpan url domain server API pada variabel, sehingga bisa dinamis (diganti)</pre>
	<p>Get.js</p>  <pre>JS Get.js X react-web-tugas > src > services > JS Get.js > [e] GetAPI > [e] promise > [e] <function> > [e] then() callback 1 import { domainPath } from "../Config"; 2 3 const GetAPI = (path) => { 4 //path digunakan untuk menunjuk alamat API yang akan di-request 5 const promise = new Promise((resolve, reject) => { 6 fetch(`\${domainPath}/\${path}`) // alamat url domain + path untuk mengakses full alamat API yang di-request 7 .then((response) => response.json()) // response dari server harus dijadikan json 8 .then((result) => { 9 resolve(result); // jika success menerima response dari server maka resolve response ke user 10 }); 11 (err) => { 12 reject(err); // jika terjadi error dari server (server down, dll) 13 } // maka kirim pesan error ke user melalui reject 14 }); 15 }; 16 return promise; 17 }; 18 19 export default GetAPI;</pre>

Post.js

```
JS Post.js X
react-web-tugas > src > services > JS Post.js > [e] PostAPI > [e] promise > <function> > <then() callback>
1  import { domainPath } from "../Config";
2
3  const PostAPI = (path, data) => {
4    const promise = new Promise((resolve, reject) => {
5      fetch(`${domainPath}/${path}`, {
6        method: "post", // method post untuk insert data
7        headers: {
8          Accept: "application/json",
9          "Content-Type": "application/json",
10         },
11        body: JSON.stringify(data), // kirimkan ke body request untuk data artikel yang
12      }) // akan ditambahkan (insert)
13    }).then(
14      (result) => {
15        resolve(result); // jika success menerima response dari server maka resolve response ke user
16      },
17      (err) => {
18        reject(err); // jika terjadi error dari server (server down, dll)
19      }
20    );
21  });
22  return promise;
23 };
24
25 export default PostAPI;
```

Delete.js

```
JS Delete.js X
react-web-tugas > src > services > JS Delete.js > [e] DeleteAPI > [e] promise > <function>
1  import { domainPath } from "../Config";
2
3  const DeleteAPI = (path, data) => {
4    const promise = new Promise((resolve, reject) => {
5      fetch(`${domainPath}/${path}/${data}`, { method: "DELETE" }) // alamat URL API yang ingin kita hapus datanya
6    }).then(
7      (result) => {
8        resolve(result); // jika success menerima response dari server maka resolve response ke user
9      },
10     (err) => {
11       reject(err); // jika terjadi error dari server (server down, dll)
12     }
13   );
14   });
15 };
16
17 export default DeleteAPI;
```

Buka file BlogPost.jsx kemudian ubah isi dari file tersebut seperti fungsi ambilDataSariServerAPI, handleHapusMahasiswa, handleTombolSimpan.

2

```
20  ambilDataDariServerAPI = () => { // fungsi untuk mengambil data dari API dengan penambahan sort dan order
21    API.getNewsMhs().then(result => {
22      this.setState({
23        listMahasiswa: result,
24      })
25    })
26  }
27
```



```

32  ✓   handleHapusMahasiswa = (data) => { // function yang meng-handle button action hapus data
33      API.deleteNewsMhs(data);
34      this.ambilDataDariServerAPI()
35  }
36

```

```

47  ✓   handleTombolSimpan = () => { // fungsi untuk meng-handle tombol simpan
48      API.postNewsMhs(this.state.insertMahasiswa)
49  ✓   .then( (Response) => {
50      this.ambilDataDariServerAPI(); // reload/refresh data
51      });
52  }
53

```

Hasil :

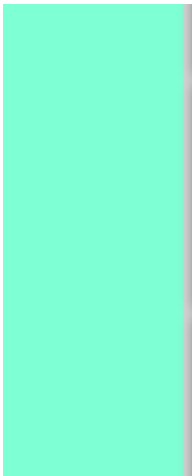


The screenshot shows a web application running on localhost:3000. The page has a sidebar on the left and a main content area. The main content area contains a form for adding a student and a list of students.

Form Fields:

- NIM
- Nama
- Alamat
- HP
- Angkatan
- Status
- Simpan** button

Daftar Mahasiswa

	<p>NIM : 1841720011</p> <p>Nama : Subhan Indra</p> <p>Alamat : Jalan Makassar No 8</p> <p>HP : 089789234567</p> <p>Angkatan : 2015</p> <p>Status : lulus</p> <p>Hapus</p>
	<p>NIM : 1841720068</p> <p>Nama : Osa Mahanani</p> <p>Alamat : Jl. Mangir XI No. 10 Malang</p> <p>HP : 089086788951</p> <p>Angkatan : 2015</p> <p>Status : lulus</p> <p>Hapus</p>
	<p>NIM : 1841720054</p> <p>Nama : Burhanuddin</p> <p>Alamat : Jl. Adi Mulya X No. 3 Malang</p> <p>HP : 081523492211</p> <p>Angkatan : 2019</p>

		<div>Status : lulus</div> <div>Hapus</div>
		<div></div> <div>NIM : 1841720154</div> <div>Nama : Kirana Putri</div> <div>Alamat : Jl. Anyer Nusa IV No. 1 Malang</div> <div>HP : 083424562159</div> <div>Angkatan : 2016</div> <div>Status : cuti</div> <div>Hapus</div>
		<div></div> <div>NIM : 1841720144</div> <div>Nama : Andini Kharisma Putri</div> <div>Alamat : Jl. Pondok Pelita VII No. 55 Jakarta</div> <div>HP : 081325249161</div> <div>Angkatan : 2016</div>

Link Github: <https://github.com/dinariskyas/PemrogramanFramework2021/tree/master/minggu9>

Link Youtube :

<https://youtu.be/rBOXHM-eDTc>