

MODUL 9
GLOBAL API

PEMROGRAMAN BERBASIS FRAMEWORK



Disusun oleh:
Livia Yurike Khuril Maula
D4 TI – 3F / 15
NIM : 1841720025

POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
PROGRAM STUDI D4 TEKNIK INFORMATIKA

APRIL 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

- **npm start**

```
Windows PowerShell
Copyright (C) Microsoft Corporation. All rights reserved.

Try the new cross-platform PowerShell https://aka.ms/pscore6

PS D:\Kuliah\SEMESTER 6\Pemrograman Berbasis Framework\Pertemuan 9\react-web> npm start

> react-web@0.1.0 start D:\Kuliah\SEMESTER 6\Pemrograman Berbasis Framework\Pertemuan 9\react-web
> react-scripts start
```

Hasil :



- **npx json-server --watch listArtikel.json --port 3001**

```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS  2: node

PS D:\Kuliah\SEMESTER 6\Pemrograman Berbasis Framework\Pertemuan 9\react-web> npx json-server --watch listArtikel.json --port 3001
npx: installed 182 in 42.51s

\{^_<^> hi!

Loading listArtikel.json
Done

Resources
http://localhost:3001/posts

Home
http://localhost:3001

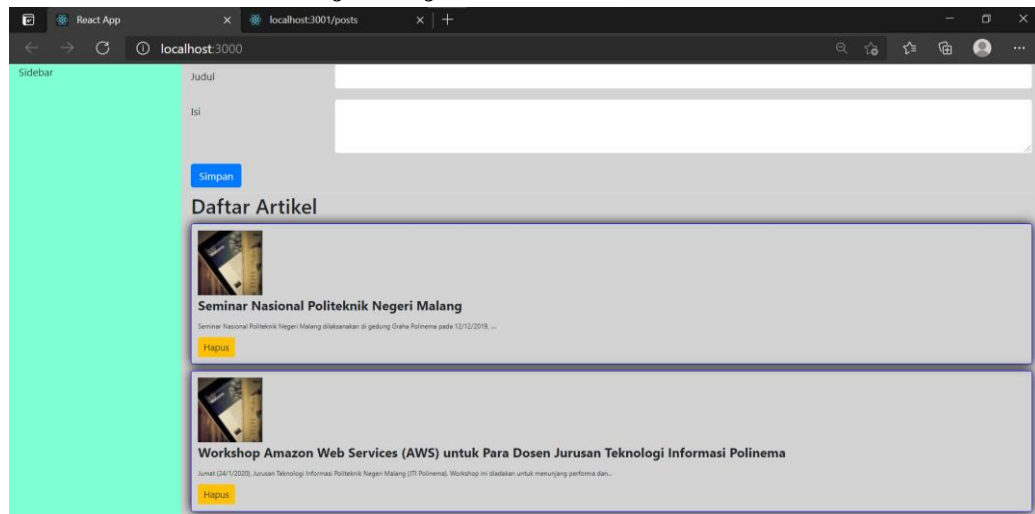
Type s + enter at any time to create a snapshot of the database
Watching
```

Hasil isi json :

```
localhost:3001/posts

[
  {
    "userId": 1,
    "id": 1,
    "title": "Workshop Amazon Web Services (AWS) untuk Para Dosen Jurusan Teknologi Informasi Polinema",
    "body": "Jumat (24/1/2020), Jurusan Teknologi Informasi Politeknik Negeri Malang (ITI Polinema). Workshop ini diadakan untuk menunjang performa dan..."
  },
  {
    "userId": 1,
    "id": 1614677184151,
    "title": "Seminar Nasional Politeknik Negeri Malang",
    "body": "Seminar Nasional Politeknik Negeri Malang dilaksanakan di gedung Graha Polinema pada 12/12/2019, ...."
  }
]
```

Hasil ketika listProduk.json dijalankan :



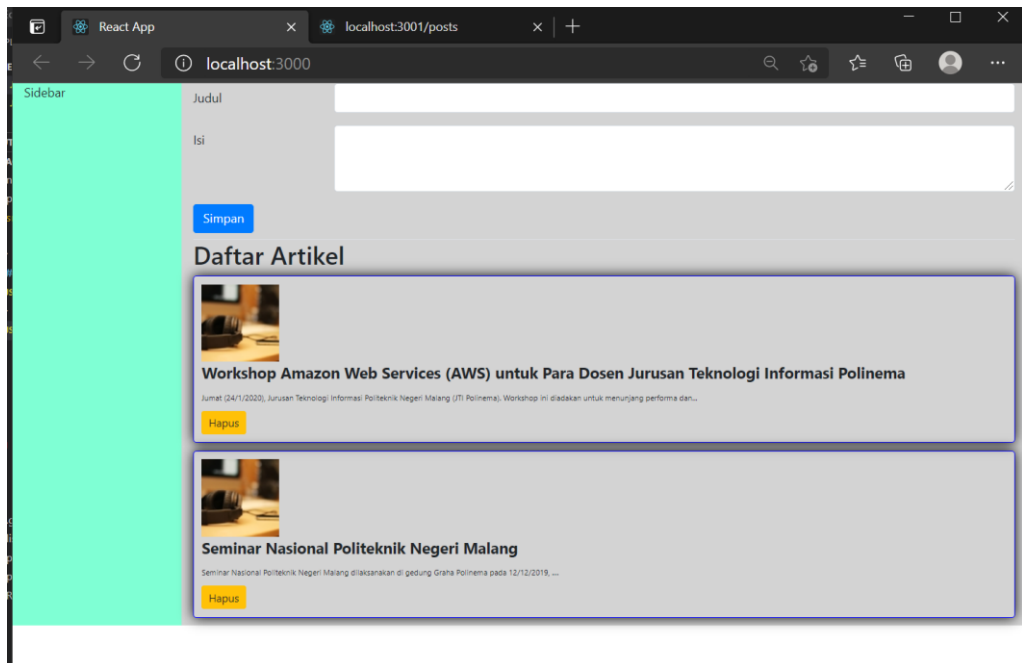
2.1 Langkah Praktikum

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

No	Keterangan
1.	<p>Buat folder baru bernama "Services" dalam folder src, kemudian buat file index.js</p> <div><pre>▼ REACT-WEB > node_modules > public ▼ src > component > container ▼ services JS index.js # App.css</pre></div>

2.	<p>Buka file BlogPost.jsx pada folder container (<i>statefull component</i>) dan akan tampil kode program</p>  <pre> 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 = { // komponen state dari React untuk statefull component 7 listArtikel: [], // variabel array yang digunakan untuk menyimpan data API 8 insertArtikel: { // kolom userId, id, title, dan body sama, mengikuti kolom yang ada pada listArtikel.json 9 userId: 1, 10 id: 1, 11 title: "", 12 body: "" 13 } 14 } 15 16 fetchDataDariServerAPI = () => { 17 fetch('http://localhost:3001/posts?_sort=id&_order=desc') //alamat URL API yang ingin kita ambil datanya 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 ketikan compnent telah di-mount-ing, maka panggil API 27 this.fetchDataDariServerAPI() // ambil data dari server API lokal 28 } </pre>
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>
4.	<p>Buka file index.js pada folder service, yang telah kita buat tadi dan tuliskan baris kode</p>  <pre> 1 const domainPath = "http://localhost:3001"; // simpan url domain server API pada variabel, sehingga bisa dinamis (diganti) 2 const GetAPI = (path) => { 3 // path digunakan untuk menunjuk alamat API mana yang akan di request 4 const promise = new Promise((resolve, reject) => { 5 fetch(`\${domainPath}/\${path}`) // alamat url domain + path untuk mengakses full alamat API yang di request 6 .then((response) => response.json()) // response dari server harus dijadikan json 7 .then(8 (result) => { 9 resolve(result); // jika success menerima response dari server maka resolve response ke user 10 }, 11 (err) => { 12 reject(err); // jika terjadi erroor dari server (server down, dll) 13 } 14); // maka kirim pesan error ke user melalui reject 15 }); 16 return promise; 17 }; 18 19 const getNewsBlog = () => GetAPI("posts?_sort=id&_order=desc"); 20 21 const API = { 22 // inisialisasi function-function yang akan disediakan global API. 23 getNewsBlog, 24 }; 25 26 export default API; 27 </pre>
5.	<p>Kembali ke file BlogPost.js. Ganti baris 17 sampai baris 23 menjadi seperti</p>  <pre> 20 //fungsi untuk mengambil data dari API dengan penambahan sort dan order 21 ambilDataDariServerAPI = () => { 22 API.getNewsBlog().then((result) => { 23 this.setState({ 24 listArtikel: result, 25 }); 26 }); 27 }; 28 </pre>

6. Simpan file **BlogPost.js** dan **index.js** tersebut, dan amati apa yang terjadi pada browser kalian



3.1 Peertanyaan Praktikum 1

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

Jawab :

- Fungsi **GetAPI** pada baris 2 di file **index.js** merupakan path yang digunakan untuk menunjuk alamat API mana yang akan di request
- Fungsi **getNewsBlog** pada baris 16 untuk melakukan sorting pada tampilan data json dari id paling terakhir berada pada urutan atas dan id pertama berada pada urutan terakhir

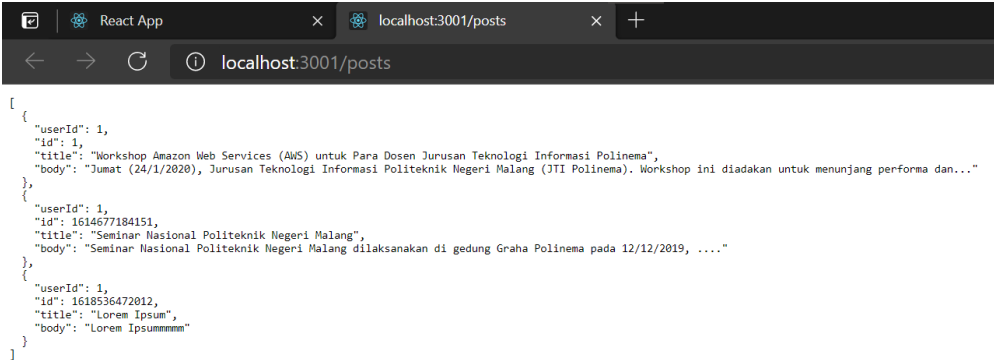
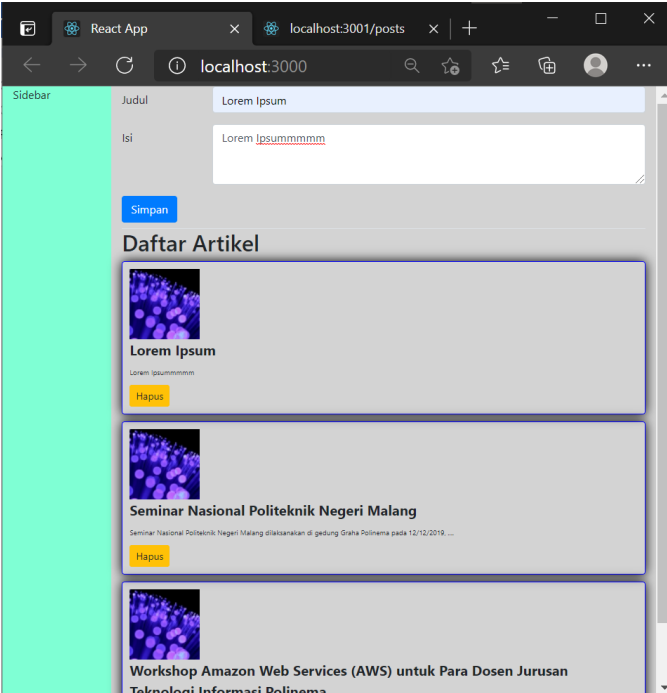
PRAKTIKUM 2

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.

No	Keterangan
1.	<p>Kita perhatikan pada fungsi handleTombolSimpan pada file BlogPost.js. Bagian inilah yang selanjutnya kita kelola agar bisa di-manage.</p> <pre>54 // fungsi untuk meng-handle tombol simpan 55 handleTombolSimpan = () => { 56 fetch("http://localhost:3001/posts", { 57 method: "post", //method POST untuk input/insert data 58 headers: { 59 Accept: "application/json", 60 "Content-Type": "application/json", 61 }, 62 // kirimkan ke body request untuk data artikel yang akan ditambahkan (insert) 63 body: JSON.stringify(this.state.insertArtikel), 64 }).then((Response) => { 65 this.ambilDataDariServerAPI(); // reload / refresh data 66 }); 67 }; 68</pre>
2.	<p>Buatlah fungsi untuk menampung action POST dari file BlogPost.js pada file services/index.js dan inisialisasi fungsi tersebut</p> <pre>19 const PostAPI = (path, data) => { 20 const promise = new Promise((resolve, reject) => { 21 fetch(`\${domainPath}/\${path}`, { 22 method: "post", // method POST untuk input/insert data 23 headers: { 24 'Accept': "application/json", 25 "Content-Type": "application/json" 26 }, 27 body: JSON.stringify(data), // kirimkan ke body request untuk data artikel yang akan ditambahkan (insert) 28 }) 29 .then((result) => { 30 resolve(result); // jika success menerima response dari server maka resolve response ke user 31 }, (err) => { 32 reject(err); // jika terjadi error dari server (server down, dll) 33 }) 34 }) 35 return promise; 36 } 37 38 const getNewsBlog = () => GetAPI(`posts?_sort=id&_order=desc`); 39 const postNewsBlog = (dataYgDiKirim) => PostAPI('posts', dataYgDiKirim); 40 41 const API = { 42 // inisialisasi function-function yang akan disediakan global API. 43 getNewsBlog, 44 postNewsBlog, 45 }; 46 47 export default API; 48</pre>

3.	<p>Selanjutnya pindah ke file BlogPost.js dan ganti isi dari fungsi handleTombolSimpan menjadi seperti</p> <pre> 53 // fungsi untuk meng-handle tombol simpan 54 handleTombolSimpan = () => { 55 API.postNewsBlog(this.state.insertArtikel) 56 .then((response) => { 57 this.ambilDataDariServerAPI(); //reload / refresh data 58 }); 59 }; 60 </pre>
4.	Selesai. Kode program pada BlogPost.js sedikit lebih simple dari seelumnya.
5.	<p>Silahkan kalian jalankan proses input artikel melalui browser dan amati apa yang terjadi</p> <ul style="list-style-type: none"> • Data json bertambah  <pre> [{ "userId": 1, "id": 1, "title": "Workshop Amazon Web Services (AWS) untuk Para Dosen Jurusan Teknologi Informasi Polinema", "body": "Jumat (24/1/2020), Jurusan Teknologi Informasi Politeknik Negeri Malang (JTI Polinema). Workshop ini diadakan untuk menunjang performa dan..." }, { "userId": 1, "id": 1614677184151, "title": "Seminar Nasional Politeknik Negeri Malang", "body": "Seminar Nasional Politeknik Negeri Malang dilaksanakan di gedung Graha Polinema pada 12/12/2019," }, { "userId": 1, "id": 1618536472012, "title": "Lorem Ipsum", "body": "Lorem Ipsummmmm" }] </pre> <p>Data yang baru ditambahkan akan berada pada urutan pertama, karena sudah di set descending</p> 

2.2 Pertanyaan Praktikum 2

1. Perhatikan file `index.js`, apa tujuan dibuatnya fungsi `PostAPI` dan fungsi `postNewsBlog`?
 - Fungsi `PostAPI` : digunakan untuk melakukan post data atau insert data ke server API `localhost:3001`
 - Fungsi `postNewsBlog` : untuk melakukan mapping dengan mengirimkan `dataYangDiKirim` berupa `idArtikel` ke `const postAPI`
2. Pada fungsi `postNewsBlog`, terdapat variable `dataYangDiKirim`. Apa tujuan dari dibuatnya variable tersebut?
 - `dataYangDiKirim` untuk menampung data sementara dari input user pada state `insertArtikel`

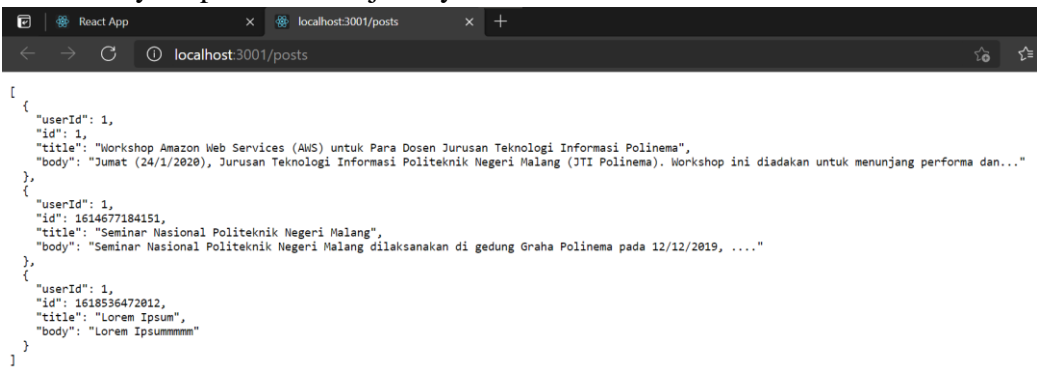
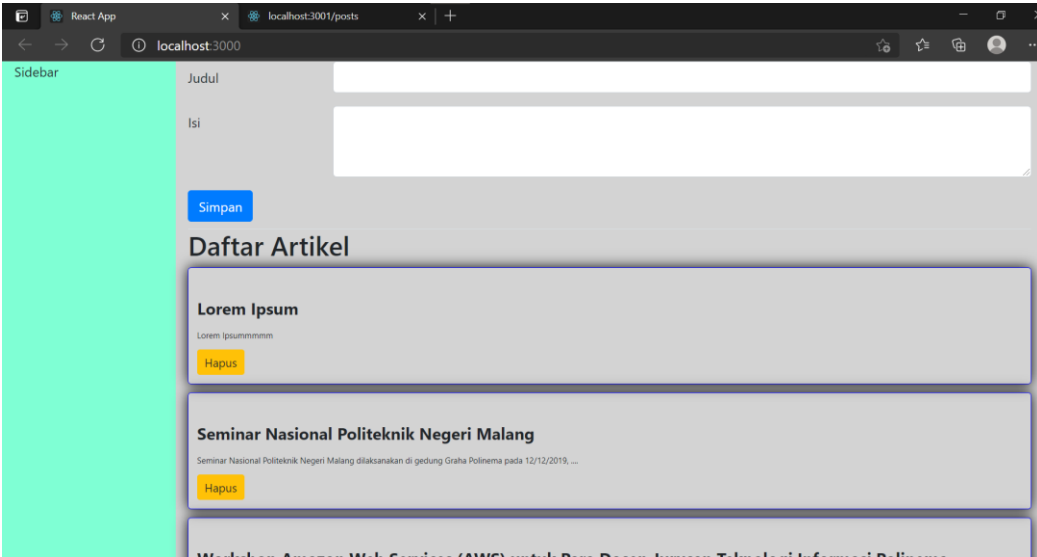
PRAKTIKUM 3

Global API service DELETE

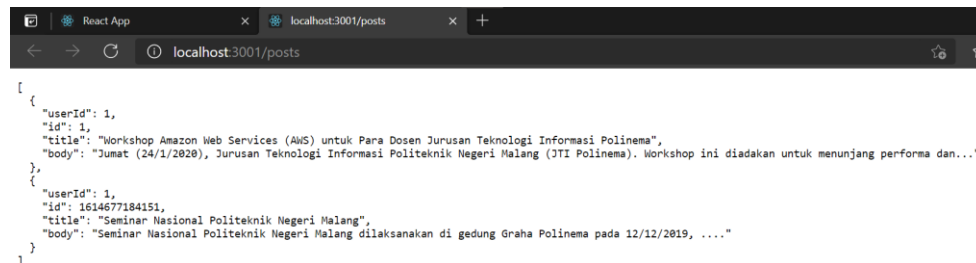
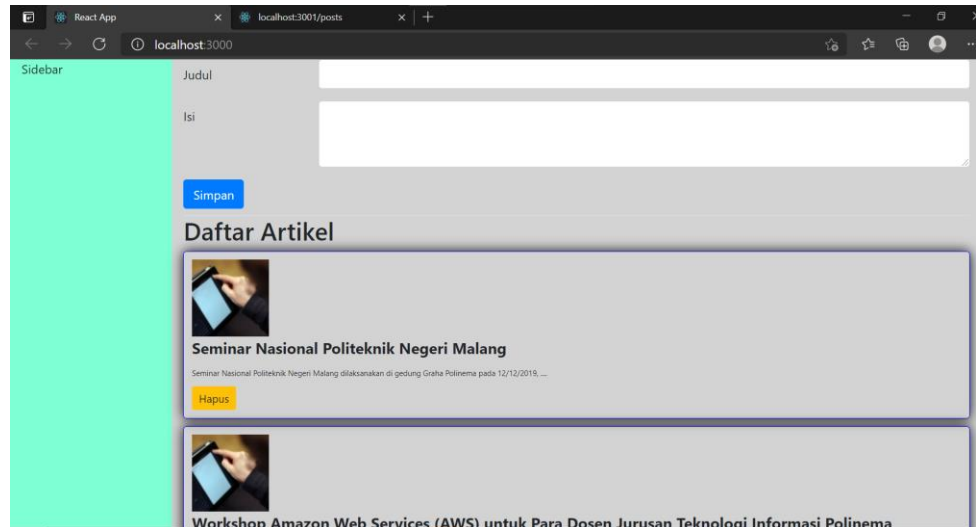
3.1 Langkah Praktikum 3

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

No	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>33 handleHapusArtikel = (data) => { 34 //fungsi yang handle button action hapus data 35 fetch(`http://localhost:3001/posts/\${data}`, { method: "DELETE" }) //alamat URL API yang ingin kita HAPUS datanya 36 .then((res) => { 37 //ketika proses hapus berhasil, maka ambil data dari server API lokal 38 this.ambilDataDariServerAPI(); 39 }); 40 }</pre>
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</p> <pre>40 const DeleteAPI = (path, data) => { 41 const promise = new Promise((resolve, reject) => { 42 fetch(`\${domainPath}/\${path}/\${data}`, {method: 'DELETE'}) // alamat URL API yang ingin kita HAPUS datanya 43 .then((result) => { 44 resolve(result); // jika success menerima response dari server maka resolve response ke user 45 }, (err) => { 46 reject(err); // jika terjadi error dari server (server down, dll) 47 }) 48 }) 49 } 50 51 const getNewsBlog = () => GetAPI(`posts?_sort=id&_order=desc`); 52 const postNewsBlog = (dataYgDiKirim) => PostAPI(`posts`, dataYgDiKirim); 53 const deleteNewsBlog = (dataYgDiHapus) => DeleteAPI(`posts`, dataYgDiHapus); 54 55 const API = { 56 // inisialisasi function-function yang akan disediakan global API 57 getNewsBlog, 58 postNewsBlog, 59 deleteNewsBlog 60 }; 61 62 export default API; 63</pre>

3.	<p>Selanjutnya pindah ke file BlogPost.jsx dan ganti isi dari fungsi handleHapusArtikel menjadi seperti</p> <pre> 33 handleHapusArtikel = (data) => { 34 //fungsi yang handle button action hapus data 35 API.deleteNewBlog(data).then(result => { 36 this.ambilDataDariServerAPI(); 37 }) 38 }; 39 </pre>
4.	<p>Selesai. Kode program pada BlogPost.jsx sedikit lebih simple dari seelumnya</p>
5.	<p>Silahkan kalian jalankan proses hapus artikel melalui browser dan amati apa yang terjadi</p> <ul style="list-style-type: none"> Awalnya seperti ini data jsonnya  <pre> [{ "userId": 1, "id": 1, "title": "Workshop Amazon Web Services (AWS) untuk Para Dosen Jurusan Teknologi Informasi Polinema", "body": "Jumat (24/1/2020), Jurusan Teknologi Informasi Politeknik Negeri Malang (JTI Polinema). Workshop ini diadakan untuk menunjang performa dan..." }, { "userId": 1, "id": 1614677184151, "title": "Seminar Nasional Politeknik Negeri Malang", "body": "Seminar Nasional Politeknik Negeri Malang dilaksanakan di gedung Graha Polinema pada 12/12/2019," }, { "userId": 1, "id": 1618536472012, "title": "Lorem Ipsum", "body": "Lorem Ipsummmmm" }] </pre> 

Lalu saya klik tombol hapus data akan hilang / terhapus



3.2 Pertanyaan Praktikum 3

1. Perhatikan file `index.js`, apa tujuan dibuatnya fungsi `DeleteAPI` dan fungsi `deleteNewsBlog`?
 - **DeleteAPI** : digunakan untuk melakukan method 'DELETE' pada server API localhost:3001
 - **deleteNewsBlog** : untuk melakukan mapping dengan mengirimkan `dataYgDihapus` berupa `idArtikel` ke `const DeleteAPI`
2. Pada fungsi `deleteNewsBlog`, terdapat variable `dataYangDiHapus`. Apa tujuan dari dibuatnya variable tersebut?
 - **dataYgDiHapus** : Untuk menampung data berupa `idArtikel` yang dihapus pada `BlogPost.jsx`

PRAKTIKUM 4

Manage Global API service

4.1 Manage Global API

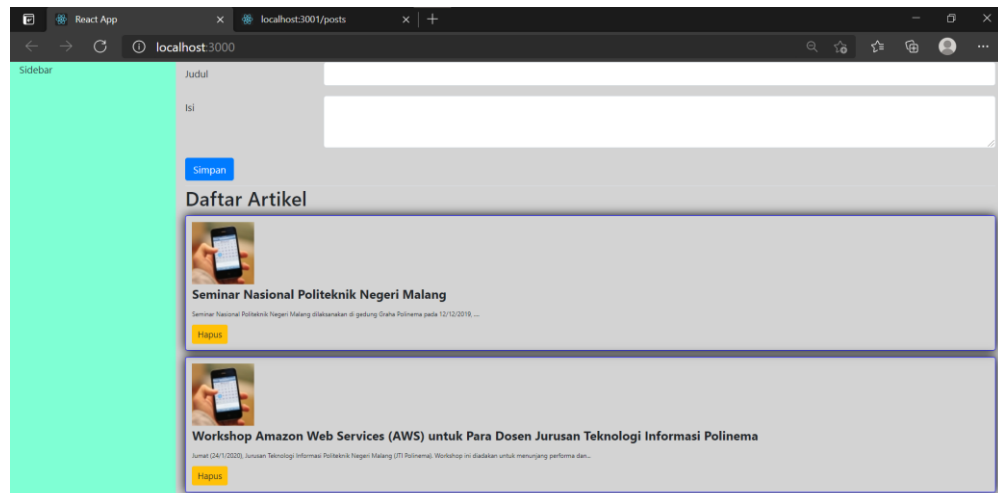
Pada global API yang telah kita lakukan pada praktikum 1, 2 dan 3. Kita mengetahui bahwa saat kita benar-benar melakukan coding untuk membuat API, kita akan dihadapkan pada banyak url API yang akan kita sediakan. Sehingga kita butuh manage lagi Global API untuk lebih teratur.

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.

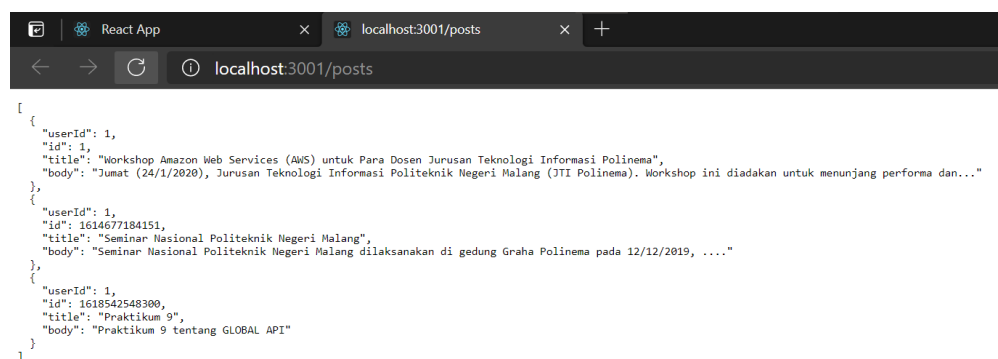
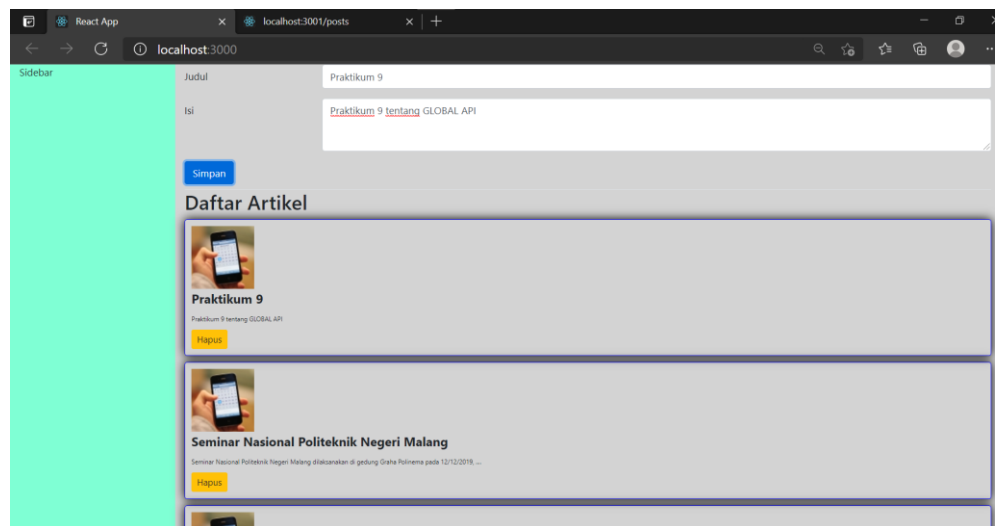
4.2 Langkah Praktikum 4

No	Keterangan
1.	<p>Buatlah file Config.js, Get.js, Post.js, dan Delete.js dalam folder services</p> 
2.	<p>Buka Config.js dan isikan kode seperti</p> 
3.	<p>Buka Get.js dan isikan kode seperti</p> 

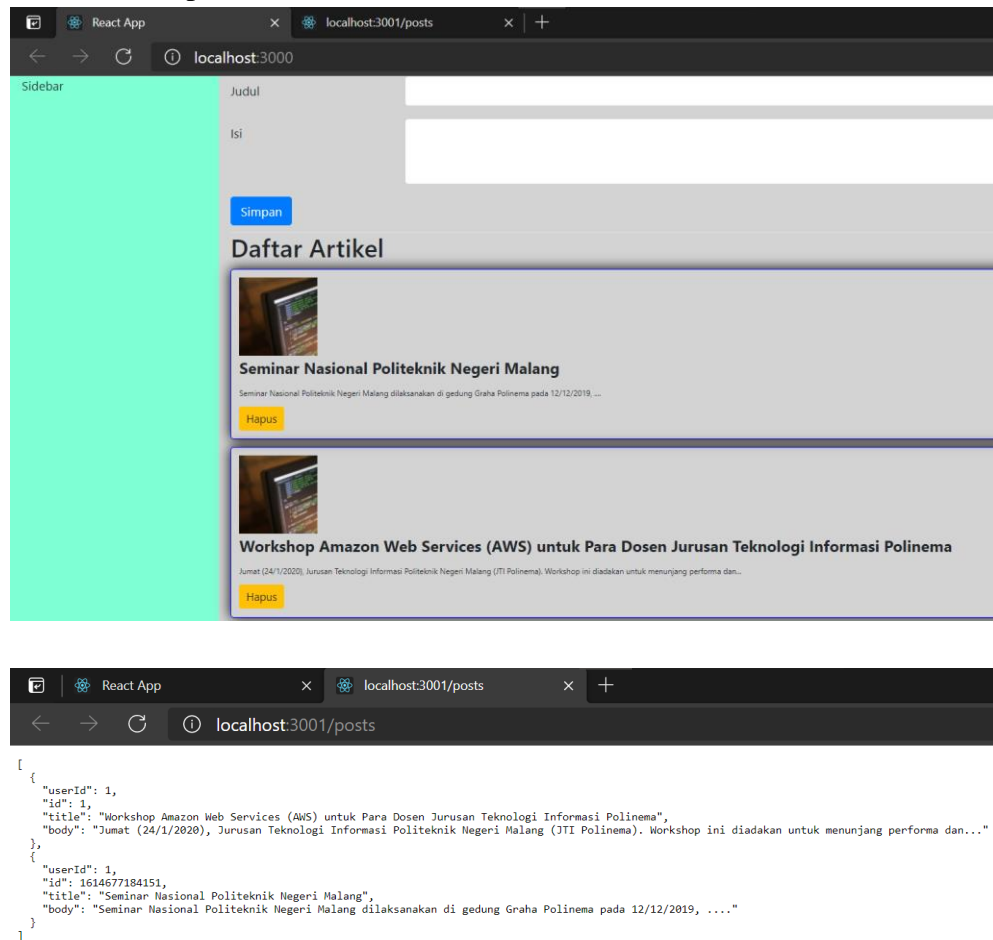
4.	<p>Buka Post.js dan isikan kode seperti</p> <pre> src > services > JS Postjs > [API] PostAPI > [API] promise > <function> 1 import {domainPath} from './Config'; 2 3 const PostAPI = (path) => { 4 const promise = new Promise((resolve, reject) => { 5 fetch(`\${domainPath}/\${path}`, { 6 method: 'post', 7 headers: { 8 'Accept': 'application/json', 9 'Content-Type': 'application/json' 10 }, 11 body: JSON.stringify(data) 12 }) 13 .then((result) => { 14 resolve(result); 15 }, (err) => { 16 reject(err); 17 }) 18 }) 19 return promise; 20 } 21 export default GetAPI; </pre>
5.	<p>Buka Delete.js dan isikan kode seperti</p> <pre> src > services > JS Deletejs > [API] DeleteAPI > [API] promise > <function> > then() callback 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((result) => { 7 resolve(result); 8 }, (err) => { 9 reject(err); 10 }) 11 }) 12 return promise; 13 } 14 export default DeleteAPI; </pre>
6.	<p>Pada index.js kita edit menjadi seperti</p> <pre> src > services > JS indexjs > [API] 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 8 // Daftar API - POST 9 const postNewsBlog = (dataYgDiKirim) => PostAPI(`posts`, dataYgDiKirim); 10 11 // Daftar API - DELETE 12 const deleteNewsBlog = (dataYgDiHapus) => DeleteAPI(`posts`, dataYgDiHapus); 13 14 const API = { // inisialisasi function-function yang akan disediakan global API 15 getNewsBlog, 16 postNewsBlog, 17 deleteNewsBlog 18 } 19 export default API; </pre>
7.	<p>Amati apa yang terjadi</p> <ul style="list-style-type: none"> Awal seperti ini 



Insert data



Jika data dihapus



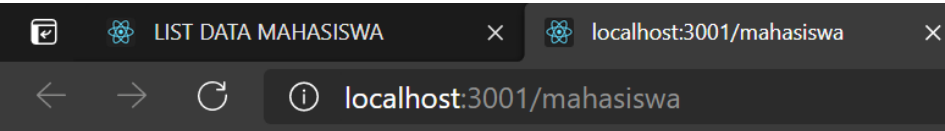
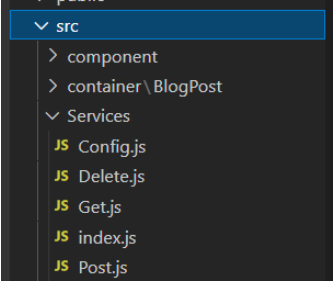
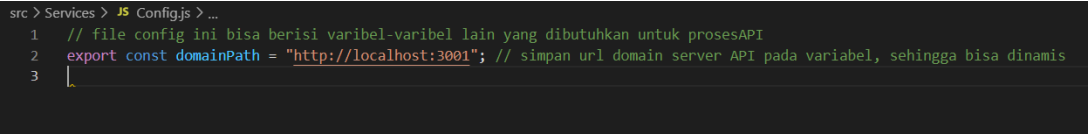
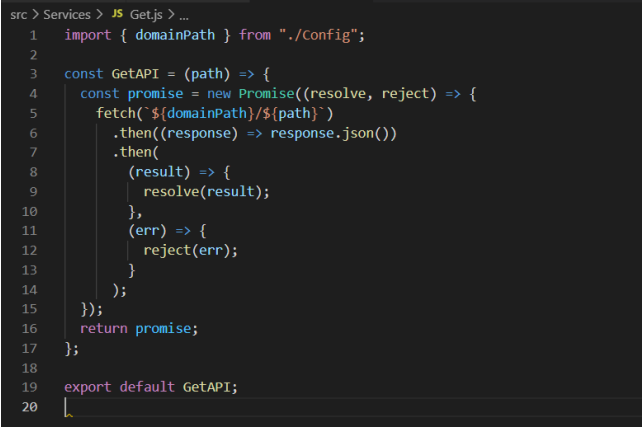
4.3 Pertanyaan Praktikum 4

1. Bagaimana caranya untuk menambahkan daftar API baru baik untuk method GET, POST, DELETE pada Global API?
 - **Dilakukan dengan cara memanggil const yang telah diimport dari file atau class lain dan melakukan passing parameter ketika dibutuhkan seperti pada 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.

No	Langkah
1.	<p>Run project tugas minggu 4, buka terminal dan ketikkan perintah npm start untuk menjalankan local server project, dan jendela terminal kedua kita isikan perintah json-server --watch listArtikel.json --port 3001 untuk menjalankan server fake API</p> <ul style="list-style-type: none"> npm start <div data-bbox="331 577 1136 1536" data-label="Image"> <p>The screenshot shows the VS Code interface. The terminal window displays the following text: Compiled successfully! You can now view tugaspertemuan4 in the browser. Local: http://localhost:3000 On Your Network: http://192.168.0.101:3000 Note that the development build is not optimized. To create a production build, use npm run build. Below the terminal, a web browser window is open at localhost:3000. It shows a form with input fields for NIM, Nama, Alamat, HP, angkatan, and status. A 'Simpan' button is at the bottom of the form. Below the form is a section titled 'Daftar Mahasiswa' which lists two students: ARDAN ANJUNG KUSUMA and OSA MAHANI SIHONO, each with their respective details and a 'Hapus' button.</p> </div> npx json-server --watch listMhs.json --port 3001 <div data-bbox="331 1630 1248 1877" data-label="Image"> <p>The screenshot shows a terminal window with the following text: PS D:\Kuliah\SEMESTER 6\Pemrograman Berbasis Framework\Pertemuan 9\tugaspertemuan9> npx json-server --watch listMhs.json --port 3001 npx: installed 182 in 28.455s \\(^_^)/ hi! Loading listMhs.json done Resources http://localhost:3001/mahasiswa Home http://localhost:3001</p> </div>

	 <pre> [{ "userId": 1, "id": 1, "NIM": "1841720025", "nama": "LIVIA YURIKE KHURIL MAULA", "alamat": "Jl. Piranha Atas No 264 Malang", "hp": "082228773286", "angkatan": 2018, "status": "AKTIF" }, { "userId": 2, "id": 2, "NIM": "1741720214", "nama": "IQBAL RAMADANI", "alamat": "Malang", "hp": "081802023523", "angkatan": 2017, "status": "CUTI" }, { "userId": 3, "id": 3, "NIM": "1641720017", "nama": "BARA PUTRI MAHARANI", "alamat": "SURABAYA", "hp": "085850336931", "angkatan": 2016, "status": "LULUS" }] </pre>
2.	<p>Buat folder baru bernama “Services” dalam folder src, kemudian buat file index.js, Config.js, Get.js, Post.js, dan Delete.js</p> 
3.	<p>Buka Config.js dan isikan kode seperti</p>  <pre> src > Services > JS Config.js > ... 1 // file config ini bisa berisi variabel-variabel lain yang dibutuhkan untuk prosesAPI 2 export const domainPath = "http://localhost:3001"; // simpan url domain server API pada variabel, sehingga bisa dinamis 3 </pre>
4.	<p>Buka Get.js dan isikan kode seperti</p>  <pre> src > Services > JS Get.js > ... 1 import { domainPath } from "./config"; 2 3 const GetAPI = (path) => { 4 const promise = new Promise((resolve, reject) => { 5 fetch(`\${domainPath}/\${path}`) 6 .then((response) => response.json()) 7 .then(8 (result) => { 9 resolve(result); 10 }, 11 (err) => { 12 reject(err); 13 } 14); 15 }); 16 return promise; 17 }; 18 19 export default GetAPI; 20 </pre>

5. Buka **Post.js** dan isikan kode seperti

```
src > Services > JS Post.js > [e] PostAPI > [e] promise > <function> > method
1  import { domainPath } from "./Config";
2
3  const PostAPI = (path, data) => {
4    const promise = new Promise((resolve, reject) => {
5      fetch(`${domainPath}/${path}`, {
6        method: "mahasiswa",
7        headers: {
8          Accept: "application/json",
9          "Content-Type": "application/json",
10        },
11        body: JSON.stringify(data),
12      }).then(
13        (result) => {
14          resolve(result);
15        },
16        (err) => {
17          reject(err);
18        }
19      );
20    });
21    return promise;
22  };
23
24  export default PostAPI;
25
```

6. Buka **Delete.js** dan isikan kode seperti

```
src > Services > JS Delete.js > ...
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" }).then(
6        (result) => {
7          resolve(result);
8        },
9        (err) => {
10         reject(err);
11       }
12     );
13   });
14   return promise;
15 };
16
17 export default DeleteAPI;
18
```

7. Pada index.js kita edit menjadi seperti

```
src > Services > JS index.js > [e] default
1  import GetAPI from "./Get";
2  import PostAPI from "./Post";
3  import DeleteAPI from "./Delete";
4
5  // Daftar API - GET
6  const getNewsBlog = () => GetAPI("mahasiswa?_sort=id&_order=desc");
7
8  // Daftar API - POST
9  const postNewsBlog = (dataVgDikirim) => PostAPI("mahasiswa", dataVgDikirim);
10
11 // Daftar API - DELETE
12 const deleteNewsBlog = (dataVgDiHapus) => DeleteAPI("mahasiswa", dataVgDiHapus);
13
14 const API = {
15   getNewsBlog,
16   postNewsBlog,
17   deleteNewsBlog,
18 };
19
20 export default API;
21
```

8.	<p>Buka file BlogPost.jsx pada folder container (<i>statefull component</i>), pada fungsi ambilDataDariServerAPI, ubah menjadi seperti ini</p> <pre> 20 ambilDataDariServerAPI = () => { 21 API.getNewsBlog().then((result) => { 22 this.setState({ 23 listMhs: result, 24 }); 25 }); 26 }; 27 </pre>
9.	<p>Pada fungsi handleTombolSimpan pada file BlogPost.js ganti isi dari fungsi handleTombolSimpan menjadi seperti</p> <pre> 50 handleTombolSimpan = () => { 51 API.postNewsBlog(this.state.insertMahasiswa).then((response) => { 52 this.ambilDataDariServerAPI(); 53 }); 54 }; 55 </pre>
10.	<p>pada fungsi handleHapusArtikel pada file BlogPost.jsx anti isi dari fungsi handleHapusArtikel menjadi seperti</p> <pre> 33 handleHapusMahasiswa = (data) => { 34 //fungsi yang handle button action hapus data 35 API.deleteNewsBlog(data).then((result) => { 36 this.ambilDataDariServerAPI(); 37 }); 38 }; 39 </pre>
11.	<p>Hasil nya seperti ini :</p> <ul style="list-style-type: none"> Awal seperti ini  <pre> [{ "userId": 1, "id": 1, "NIM": "1841728025", "nama": "LIVIA YURIKE KHURIL MAULA", "alamat": "Jl. Pirmaha Atas No 264 Malang", "hp": "082228773286", "angkatan": 2018, "status": "AKTIF" }, { "userId": 2, "id": 2, "NIM": "1741728214", "nama": "IQBAL RAMADANI", "alamat": "Malang", "hp": "081862823523", "angkatan": 2017, "status": "GUTI" }, { "userId": 3, "id": 3, "NIM": "1641728017", "nama": "RAHA PUTRI MAHARANI", "alamat": "SURABAYA", "hp": "085850336931", "angkatan": 2016, "status": "LULUS" }, { "userId": 4, "id": 4, "NIM": "1841728064", "nama": "USA MAHANI SIHOWO", "alamat": "BLITAH", "hp": "08966468527", "angkatan": 2018, "status": "AKTIF" }, { "userId": 5, "id": 5, "NIM": "1841728041", "nama": "ARDAN ANJUNG KUSUMA", "alamat": "BOJONEGORO", "hp": "085258967890", "angkatan": 2018, "status": "AKTIF" }] </pre>

LIST DATA MAHASISWA

localhost:3001/mahasiswa

localhost:3000

Sidebar

NIM

Nama

Alamat


HP

angkatan


status

Simpan


Daftar Mahasiswa



ARDAN ANJUNG KUSUMA
BOJONEGORO
085258967800
2018
AKTIF
Hapus



OSA MAHANI SIHONO
BLITAR
089665468527
2018
AKTIF
Hapus



RARA PUTRI MAHARANI
SURABAYA
085850336931
2016
LULUS
Hapus

Insert data

LIST DATA MAHASISWA

localhost:3001/mahasiswa

localhost:3000

Sidebar

NIM

Nama

Alamat


HP

angkatan


status

Simpan


Daftar Mahasiswa



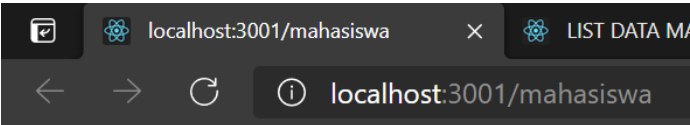
1841720897
Alifah
Malang
087888768987
2018
Aktif
Hapus



ARDAN ANJUNG KUSUMA
BOJONEGORO
085258967800
2018
AKTIF
Hapus

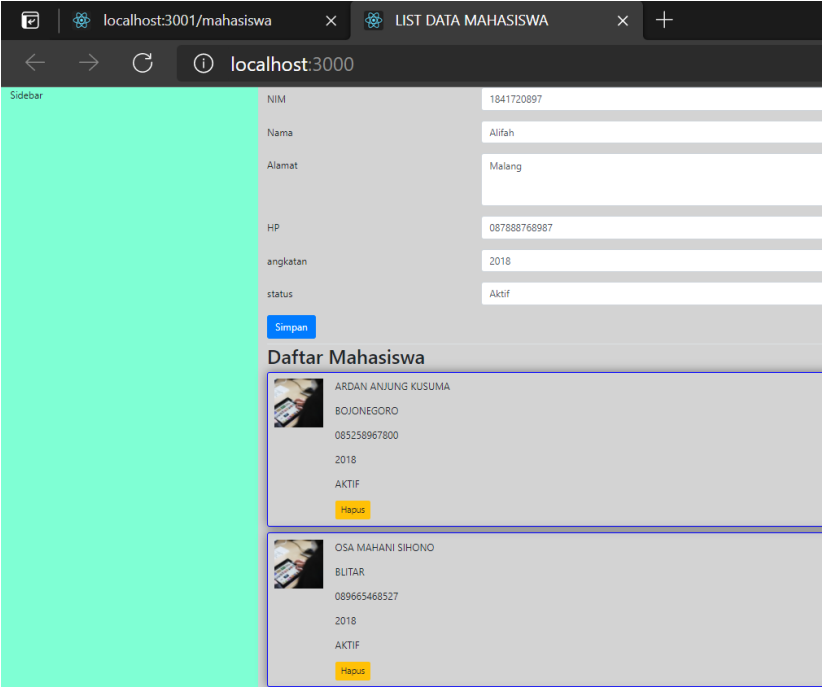


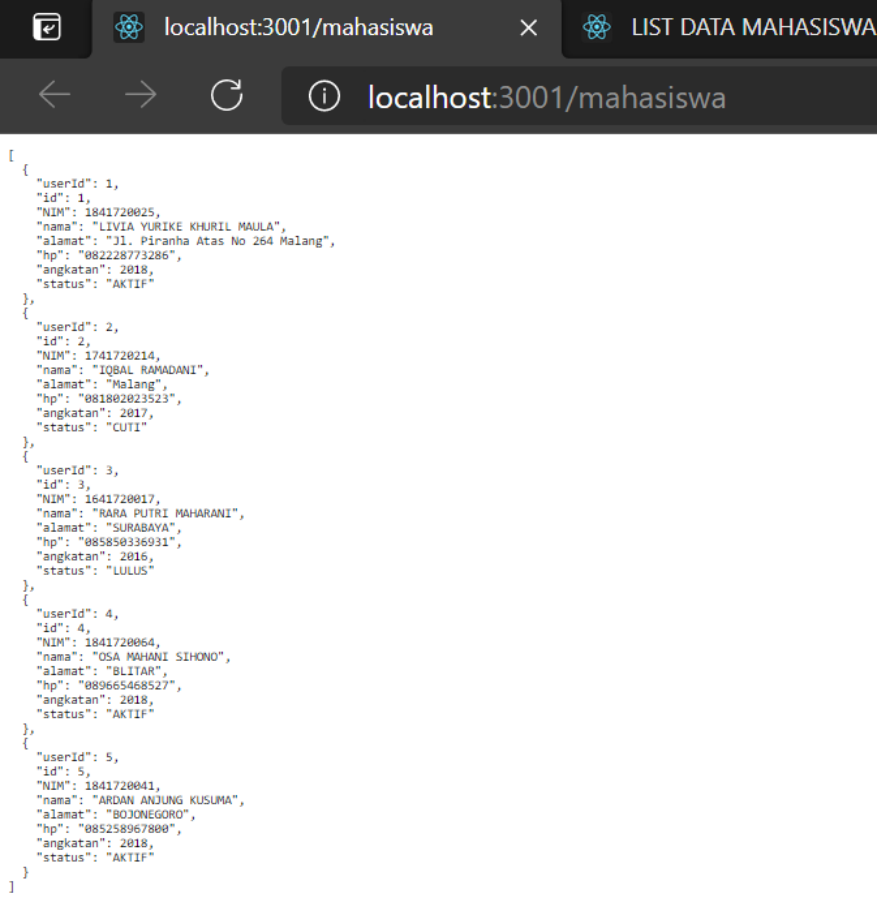
OSA MAHANI SIHONO
BLITAR
089665468527
2018
AKTIF
Hapus



```
[
  {
    "userId": 1,
    "id": 1,
    "NIM": 1841720025,
    "nama": "LIVIA YURIKE KHURIL MAULA",
    "alamat": "Jl. Piranha Atas No 264 Malang",
    "hp": "082228773286",
    "angkatan": 2018,
    "status": "AKTIF"
  },
  {
    "userId": 2,
    "id": 2,
    "NIM": 1741720214,
    "nama": "IQBAL RAMADANI",
    "alamat": "Malang",
    "hp": "081802023523",
    "angkatan": 2017,
    "status": "CUTI"
  },
  {
    "userId": 3,
    "id": 3,
    "NIM": 1641720017,
    "nama": "RARA PUTRI MAHARANI",
    "alamat": "SURABAYA",
    "hp": "085850336931",
    "angkatan": 2016,
    "status": "LULUS"
  },
  {
    "userId": 4,
    "id": 4,
    "NIM": 1841720064,
    "nama": "OSA MAHANI SIHONO",
    "alamat": "BLITAR",
    "hp": "089665468527",
    "angkatan": 2018,
    "status": "AKTIF"
  },
  {
    "userId": 5,
    "id": 5,
    "NIM": 1841720041,
    "nama": "ARDAN ANJUNG KUSUMA",
    "alamat": "BOJONEGORO",
    "hp": "085258967800",
    "angkatan": 2018,
    "status": "AKTIF"
  },
  {
    "userId": 1,
    "id": 1,
    "nim": "1841720897",
    "nama": "Alifah",
    "alamat": "Malang",
    "hp": "087888768987",
    "angkatan": "2018",
    "status": "Aktif"
  }
]
```

Jika data dihapus



	 <pre> [{ "userId": 1, "id": 1, "NIM": "1841720025", "nama": "LIVIA YURIKE KHURIL MAULA", "alamat": "Jl. Piranha Atas No 264 Malang", "hp": "082228773286", "angkatan": "2018", "status": "AKTIF" }, { "userId": 2, "id": 2, "NIM": "1741720214", "nama": "IQBAL RAMADANI", "alamat": "Malang", "hp": "081802023523", "angkatan": "2017", "status": "CUTI" }, { "userId": 3, "id": 3, "NIM": "1641720017", "nama": "RARA PUTRI MAHARANI", "alamat": "SURABAYA", "hp": "085850336931", "angkatan": "2016", "status": "LULUS" }, { "userId": 4, "id": 4, "NIM": "1841720064", "nama": "OSA MAHANI SIHONO", "alamat": "BLITAR", "hp": "089665468527", "angkatan": "2018", "status": "AKTIF" }, { "userId": 5, "id": 5, "NIM": "1841720041", "nama": "ARDAN ANJUNG KUSUMA", "alamat": "BOJONEGORO", "hp": "085258967800", "angkatan": "2018", "status": "AKTIF" }] </pre>
--	--

Link Github :

<https://github.com/liviayurike/Pemrograman-Berbasis-Framework/tree/master/Pertemuan%209>

Link Youtube :

<https://youtu.be/sONdVJdwFU0>