

TUGAS INDIVIDU

MODUL 8

GLOBAL API SERVICE

Disusun sebagai

MATA KULIAH : PEMROGRAMAN BERBASIS FRAMEWORK

Oleh :

Febri Toriqkhul Khoiri /1741720171

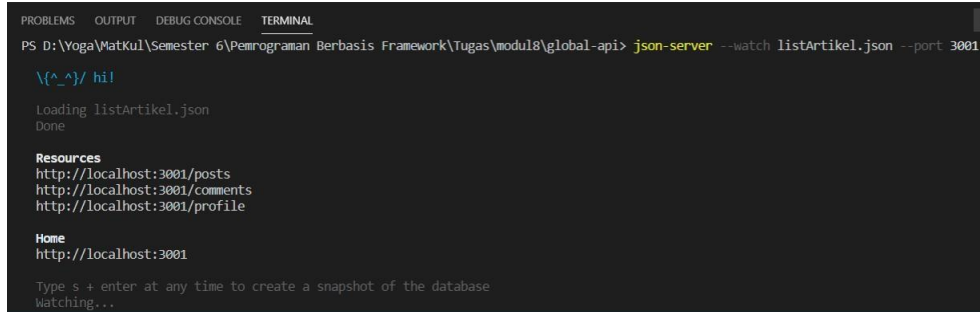
TI 3-A



**PROGRAM STUDI D-IV TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG
2020**

A. Praktikum 1

1. Run project dan server fake API pada modul sebelumnya.



```
PS D:\Voga\MatKul\Semester 6\Pemrograman Berbasis Framework\Tugas\modul8\global-api> json-server --watch listArtikel.json --port 3001

\(^_^)/ hi!

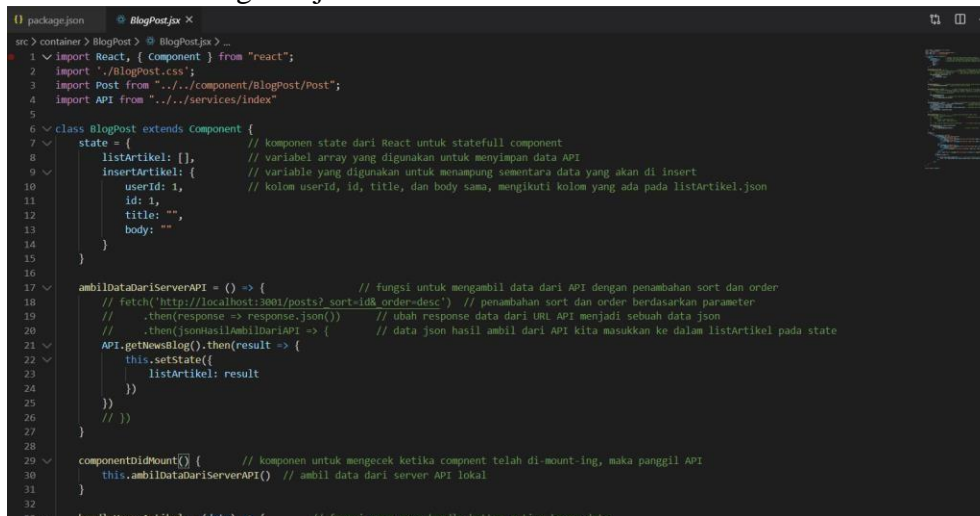
Loading listArtikel.json
Done

Resources
http://localhost:3001/posts
http://localhost:3001/comments
http://localhost:3001/profile

Home
http://localhost:3001

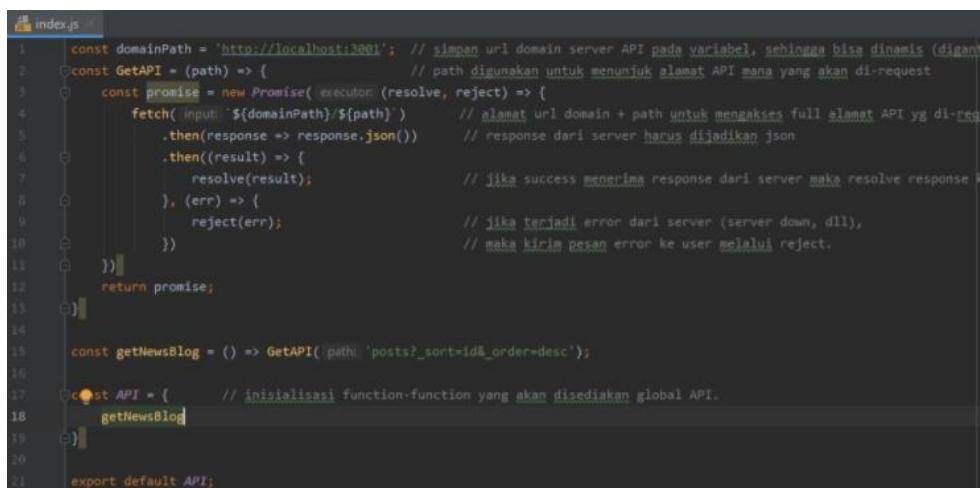
Type s + enter at any time to create a snapshot of the database
Watching...
```

2. Membuat index.js pada folder services.
3. Membuka file BlogPost.jsx.



```
package.json  BlogPost.jsx
src > container > BlogPost > BlogPost.jsx > ...
1 import React, { Component } from "react";
2 import './BlogPost.css';
3 import Post from '../services/Post';
4 import API from '../services/index'
5
6 class BlogPost extends Component {
7   state = { // komponen state dari React untuk statefull component
8     listArtikel: [], // variabel array yang digunakan untuk menyimpan data API
9     insertArtikel: { // variabel yang digunakan untuk menampung sementara data yang akan di insert
10       userId: 1, // kolom userId, id, title, dan body sama, mengikuti kolom yang ada pada listArtikel.json
11       id: 1,
12       title: "",
13       body: ""
14     }
15   }
16
17   ambilData dari server API = () => { // fungsi untuk mengambil data dari API dengan penambahan sort dan order
18     // fetch('http://localhost:3001/posts?_sort=id&_order=desc') // penambahan sort dan order berdasarkan parameter
19     // .then(response => response.json()) // ubah response data dari URL API menjadi sebuah data json
20     // .then(jsonHasilAmbil dari API => { // data json hasil ambil dari API kita masukkan ke dalam listArtikel pada state
21     API.getNewsBlog().then(result => {
22       this.setState({
23         listArtikel: result
24       })
25     })
26     // })
27   }
28
29   componentDidMount() { // komponen untuk mengecek ketika component telah di-mount-ing, maka panggil API
30     this.ambilData dari server API() // ambil data dari server API lokal
31   }
32
33   handlePostClick = (data) => { // fungsi untuk menambahkan data ke database
34     // ...
35   }
36 }
```

4. Membuat kode pada index.js.



```
index.js
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( (execution) (resolve, reject) => {
4     fetch( 'input: ${domainPath}/${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( path: '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;
```

5. Mengubah baris 17 pada BlogPost.jsx.

```
ambilDataDariServerAPI = () => { // fungsi untuk mengambil
  // fetch('http://localhost:3001/posts?_sort=id&_order=desc') // penam
  // .then(response => response.json()) // ubah response data d
  // .then(jsonHasilAmbilDariAPI => { // data json hasil ambi
  API.getNewsBlog().then(result => {
    this.setState({
      listArtikel: result
    })
  })
  // })
}
```

B. Praktikum 2

1. Buka fungsi handleTombolSimpan.

```
handleTombolSimpan = () => { // fungsi untuk meng-handle tombol simpan
  fetch('http://localhost:3001/posts', {
    method: 'post', // method POST untuk input/insert data
    headers: {
      'Accept': 'application/json',
      'Content-Type': 'application/json'
    },
    body: JSON.stringify(this.state.insertArtikel) // kirimkan ke body request untuk data artikel y
  })
  // API.postNewsBlog(this.state.insertArtikel)
  // .then((response) => {
  //   this.ambilDataDariServerAPI(); // reload / refresh data
  // });
}
```

2. Mengedit fungsi handleTombolSimpan dan membuat post.

```
handleTombolSimpan = () => { // fungsi untuk meng-handle tombol simpan
  // fetch('http://localhost:3001/posts', {
  //   method: 'post', // method POST untu
  //   headers: {
  //     'Accept': 'application/json',
  //     'Content-Type': 'application/json'
  //   },
  //   body: JSON.stringify(this.state.insertArtikel) // kirimkan ke body
  // })
  API.postNewsBlog(this.state.insertArtikel)
    .then((response) => {
      this.ambilDataDariServerAPI(); // reload / refresh da
    });
}
```

C. Praktikum 3

1. Buka fungsi handleHapusArtikel.

```
handleHapusArtikel = (data) => { // fungsi yang meng-handle button action hapus data
  fetch(`http://localhost:3001/posts/${data}`, { method: 'DELETE' }) // alamat URL API
  .then(res => { // ketika proses hapus berhasil, maka ambil data dari server API
    this.ambilDataDariServerAPI()
  })
  // API.deleteNewBlog(data)
  // .then(response => { // ketika proses hapus berhasil, maka ambil data dari
  //   this.ambilDataDariServerAPI()
  // })
}
```

2. Membuat fungsi untuk menampung delete pada index.js.

```
const DeleteAPI = (path, data) => {
  const promise = new Promise( (resolve, reject) => {
    fetch( input: `${domainPath}/${path}/${data}`, init: {method: 'DELETE'}) // alamat
    .then((result : Response) => {
      resolve(result); // jika success menerima response dari server
    }, (err) => {
      reject(err); // jika terjadi error dari server
    })
  })
}

const getNewsBlog = () => GetAPI( path: 'posts?sort=id&order=desc');
const postNewsBlog = (dataVgDiKirim) => PostAPI( path: 'posts', dataVgDiKirim);
const deleteNewsBlog = (dataVgDiHapus) => DeleteAPI( path: 'posts', dataVgDiHapus);

const API = { // inisialisasi function-function yang akan disediakan global API.
  getNewsBlog,
  postNewsBlog,
  deleteNewsBlog
}

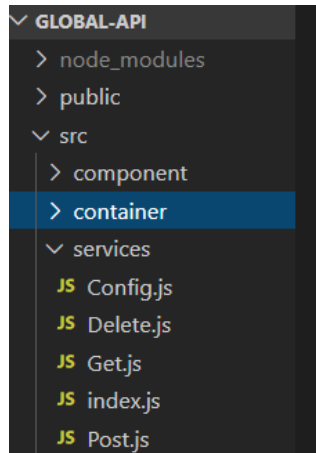
export default API;
```

3. Mengedit isi fungsi handleHapusArtikel.

```
handleHapusArtikel = (data) => { // fu
  // fetch(`http://localhost:3001/posts/${data}`
  // .then(res => { // ketika proses
  //   this.ambilDataDariServerAPI()
  // })
  API.deleteNewBlog(data)
  .then(response => { // ketika pr
    this.ambilDataDariServerAPI()
  })
}
```

D. Praktikum 4

1. Membuat Config.js, Delete.js, Get.js, Post.js pada folder services.



2. Mengisikan kode pada Config.js.

```
src > services > JS Config.js > ...  
1 | export const domainPath = "http://localhost:3001";
```

3. Mengisikan kode pada Delete.js

```
src > services > JS Delete.js > [⌕] default  
1 | import { domainPath } from './Config';  
2 |  
3 | const DeleteAPI = (path, data) => {  
4 |   const promise = new Promise((resolve, reject) => {  
5 |     fetch(`${domainPath}/${path}/${data}`, {  
6 |       method: "DELETE"  
7 |     })  
8 |     .then((result) => {  
9 |       resolve(result)  
10 |     }, err => {  
11 |       reject(err)  
12 |     })  
13 |   })  
14 |   return promise;  
15 | }  
16 |  
17 | export default DeleteAPI;
```

4. Mengisikan kode pada Get.js.

```
src > services > JS Get.js > [⌕] default
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((result) => {
8 |         resolve(result)
9 |       }, err => {
10 |        reject(err)
11 |      })
12 |   })
13 |   return promise;
14 | }
15 |
16 | export default getAPI;
```

5. Mengisikan kode pada Post.js.

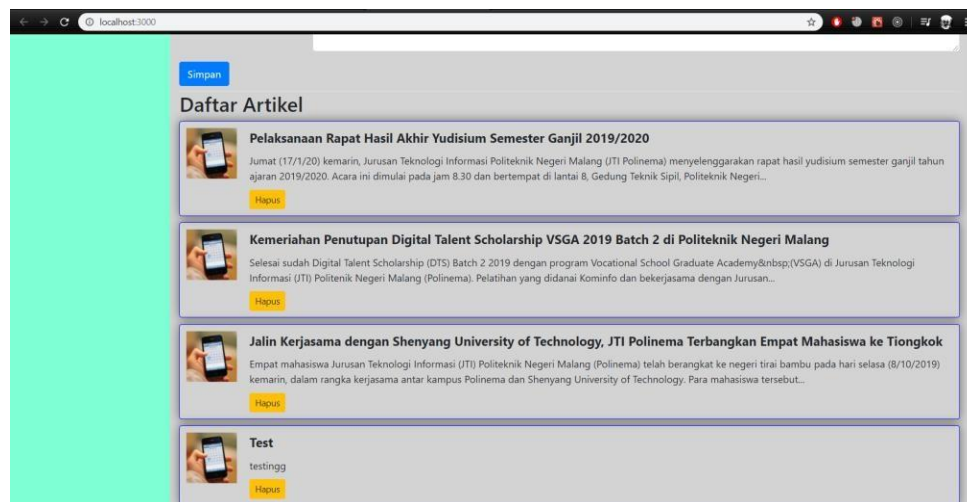
```
src > services > JS Post.js > [⌕] default
1 | import { domainPath } from './Config';
2 |
3 | const PostAPI = (path, data) => {
4 |   const promise = new Promise((resolve, reject) => {
5 |     fetch(`${domainPath}/${path}`, {
6 |       method: "post",
7 |       headers: {
8 |         'Accept': 'application/json',
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 |
22 | export default PostAPI;
```


6. Mengedit index.js.

```
src > services > JS index.js > [?] default
1  import React from 'react';
2  import getAPI from './Get';
3  import PostAPI from './Post';
4  import DeleteAPI from './Delete';
5
6  const getNewsBlog = () => getAPI("posts?_sort=id&_order=asc")
7  const postNewsBlog = (dataYangDikirim) => PostAPI("posts", dataYangDikirim)
8  const deleteNewBlog = (dataYgDihapus) => DeleteAPI("posts", dataYgDihapus)
9
10 const API = {
11   getNewsBlog,
12   postNewsBlog,
13   deleteNewBlog
14 }
15
16 export default API;
```

7. Hasil setelah program dijalankan.

- Setelah ditambahkan



- Setelah dihapus

