

MODUL 4
INTERACTION WITH API

PEMROGRAMAN BERBASIS FRAMEWORK



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

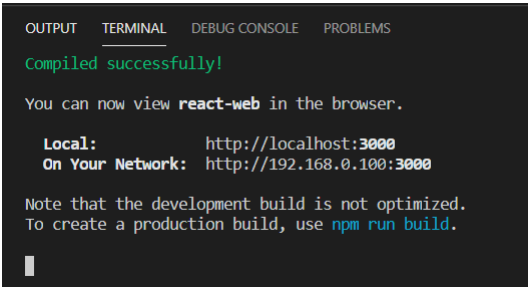
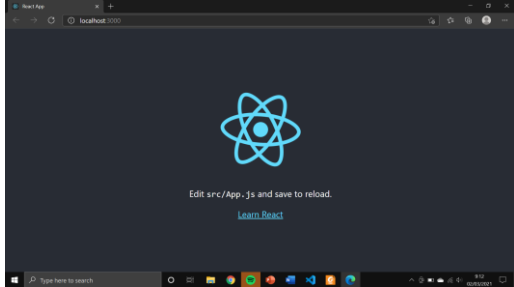
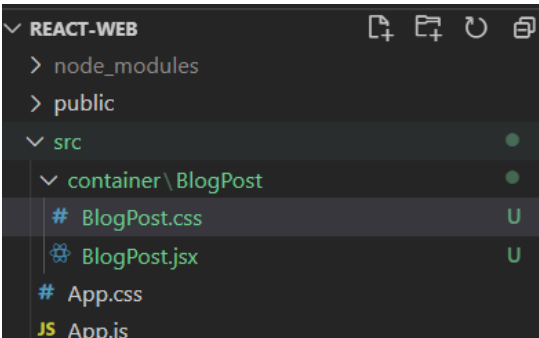
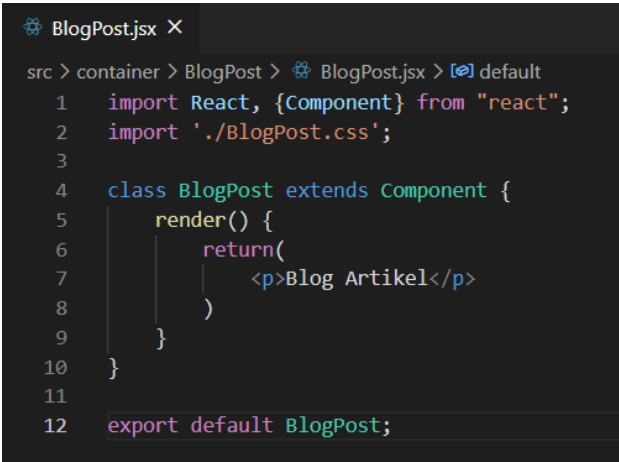
POLITEKNIK NEGERI MALANG
JURUSAN TEKNOLOGI INFORMASI
PROGRAM STUDI D4 TEKNIK INFORMATIKA

MARET 2021

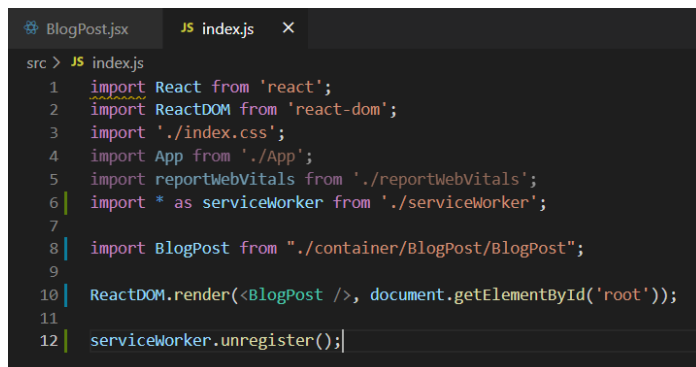
PRAKTIKUM 1

Interaksi dengan API menggunakan method GET

Langkah Praktikum

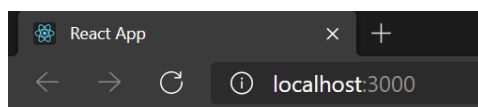
No	Keterangan
1	<p>Buka Project React pada pertemuan sebelumnya dan jalankan “npm start” menggunakan <i>cmd</i> dalam direktori tersebut.</p> <p>Hasil :</p> <div>   </div>
2	Buat folder baru bernama “BlogPost” pada folder container (<i>statefull component</i>).
3	<p>Buat file BlogPost.jsx dan BlogPost.css di dalam folder “BlogPost”</p> <div>  </div>
4	<p>Buka file BlogPost.jsx dan ketikkan kode</p> <div>  </div>

- 5 Pada file **index.js**, lakukan import component **BlogPost**



```
src > JS index.js
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import './index.css';
4 import App from './App';
5 import reportWebVitals from './reportWebVitals';
6 import * as serviceWorker from './serviceWorker';
7
8 import BlogPost from './container/BlogPost/BlogPost';
9
10 ReactDOM.render(<BlogPost />, document.getElementById('root'));
11
12 serviceWorker.unregister();
```

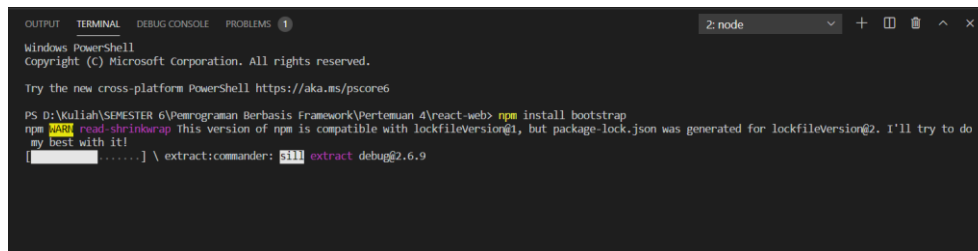
- 6 Pada web browser akan tampil seperti



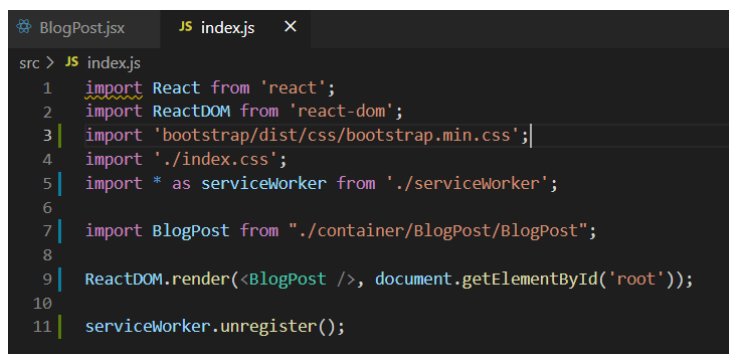
Blog Artikel

Tahapan selanjutnya adalah perbaikan tampilan sebuah website untuk mempercantik halaman website tersebut dengan menggunakan **Bootstrap** yang umum digunakan.

- 7 Import css **bootstrap.min.css** (css bootstrap yang sudah dikompresi) ke dalam **index.js** (seperti Gambar 1.6). Jika css tidak ditemukan, install lewat cmd dengan perintah “**npm install bootstrap**”



```
PS D:\Kuliah\SEMESTER 6\Program Berbasis Framework\Pertemuan 4\react-web> npm install bootstrap
npm WARN read-shrinkwrap This version of npm is compatible with lockfileVersion@1, but package-lock.json was generated for lockfileVersion@2. I'll try to do my best with it!
[.....] \ extract:commander: 5111 extract debug@2.6.9
```



```
src > JS index.js
1 import React from 'react';
2 import ReactDOM from 'react-dom';
3 import 'bootstrap/dist/css/bootstrap.min.css';
4 import './index.css';
5 import * as serviceWorker from './serviceWorker';
6
7 import BlogPost from './container/BlogPost/BlogPost';
8
9 ReactDOM.render(<BlogPost />, document.getElementById('root'));
10
11 serviceWorker.unregister();
```

8

Modifikasi file **index.html** pada folder "**public**" seperti Gambar 1.7. Cermati *code program* yang ada dalam gambar!.

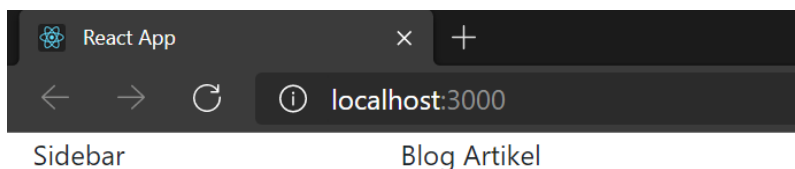
```

1 <!DOCTYPE html>
2 <html lang="en">
3 <head>
4   <meta charset="utf-8" />
5   <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
6   <meta name="viewport" content="width=device-width, initial-scale=1" />
7   <meta name="theme-color" content="#000000" />
8   <meta name="description" content="Web site created using create-react-app"/>
9   <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
10  <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
11  <title>React App</title>
12 </head>
13 <body>
14   <noscript>You need to enable JavaScript to run this app.</noscript>
15   <!--penerapan layout bisa dilihat di https://getbootstrap.com/docs/4.0/layout/overview -->
16   <!-- Containers adalah basic layout element di bootstrap dan DIPERLUKAN jika menggunakan default grid system bootstrap-->
17   <!-- ada 2 pilihan Containers
18    1. class container : untuk layout yang model box (tidak full-width layar browser)
19    2. class container-fluid : untuk layout model yang full-width layar browser
20    untuk lebih jelasnya, silahkan dicoba satu persatu.
21    Default kita saat ini, menggunakan yang full-width (container-fluid)-->
22   <div class="container-fluid">
23     <div class="row">
24       <div class="col-2" id="sidebar">Sidebar</div>
25       <div class="col-10" id="content"></div>
26     </div>
27   </div>
28 </body>
29 </html>
30

```

9

Amati tampilan yang ada pada browser



10

Buka file **index.css** dan tambahkan code css seperti Gambar 1.9, untuk menambah sedikit style pada halaman web

```

1 body {
2   margin: 0;
3   font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen",
4     "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",
5     sans-serif;
6   -webkit-font-smoothing: antialiased;
7   -moz-osx-font-smoothing: grayscale;
8 }
9
10 code {
11   font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier New",
12     monospace;
13 }
14
15 #sidebar {
16   background-color: #a9d0d9;
17 }
18
19 #content {
20   background-color: #f0f0f0;
21 }
22

```

11

Perhatikan kembali browser, dan lihat hasil tampilan seperti



12

Ubah kode program untuk *statefull component* **BlogPost.jsx**

```

BlogPost.jsx X JS index.js # index.css index.html
src > container > BlogPost > BlogPost.jsx > default
1  import React, {Component} from "react";
2  import './BlogPost.css';
3
4  class BlogPost extends Component {
5    render() {
6      return(
7        <div class="post-artikel">
8          <h2>Daftar Artikel</h2>
9          <div class="artikel">
10             <div class="gambar-artikel">
11               
12             </div>
13             <div class="konten-artikel">
14               <div class="judul-artikel">Judul Artikel</div>
15               <p class="isi-artikel">Isi Artikel</p>
16             </div>
17           </div>
18         </div>
19       )
20     }
21   }
22
23   export default BlogPost;

```

13

Tambahkan custom css ke **BlogPost.css**

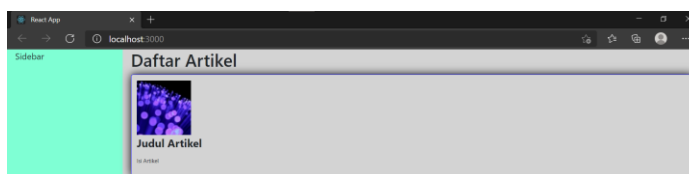
```

src > container > BlogPost > # BlogPost.css > ...
1  .artikel {
2    width: 100%;
3    padding: 10px;
4    border: 1px solid blue;
5    border-radius: 4px;
6    margin-bottom: 10px;
7    box-shadow: 0 0 16px rgba(0, 0, 0.5);
8    display: flex;
9  }
10
11  .gambar-partikel {
12    height: 80px;
13    width: 80px;
14    margin-right: 20px;
15    vertical-align: top;
16  }
17
18  .gambar-artikel img {
19    width: 100px;
20    height: 100px;
21    object-fit: cover;
22  }
23
24  .kontek-artikel {
25    flex: 1;
26  }
27
28  .konten-artikel div.judul-artikel {
29    font-size: 20px;
30    margin-right: 20px;
31    vertical-align: top;
32  }
33
34
35  .gambar-artikel {
36    width: 100%;
37    height: 100%;
38    object-fit: cover;
39  }
40
41  .konten-artikel {
42    flex: 1;
43  }
44
45  .konten-artikel div.judul-artikel {
46    font-size: 20px;
47    font-weight: bold;
48    margin-bottom: 10px;
49  }
50
51  .konten-artikel p.isi-artikel {
52    font-size: 10px;
53    margin-bottom: 10px;
54  }

```

14

Perhatikan tampilan browser



Pemindahan dari *statefull component* ke *stateless component*

15 Buat folder **BlogPost** pada folder **component** (*stateless component*), lalu buat file **Post.jsx**

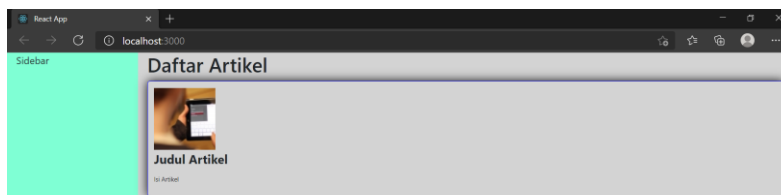
16 Potong (*cut*) baris 9-17 pada *statefull component* **BlogPost.jsx** ke *stateless component* **Post.jsx**, dan modifikasi **Post.jsx**

```
src > component > BlogPost > Post.jsx > default
1  import React from "react";
2
3  const Post = (props) => {
4    return(
5      <div className="artikel">
6        <div className="gambar-artikel">
7          
8        </div>
9        <div className="konten-artikel">
10         <div class="judul-artikel">Judul Artikel</div>
11         <p class="isi-artikel">Isi Artikel</p>
12       </div>
13     </div>
14   )
15 }
16
17 export default Post;
```

17 Untuk *statefull component* **BlogPost.jsx** pada baris 10, panggil *stateless component* **Post.jsx** seperti

```
src > container > BlogPost > BlogPost.jsx > BlogPost > render
1  import React, {Component} from "react";
2  import './BlogPost.css';
3  import Post from "../../component/BlogPost/Post";
4
5  class BlogPost extends Component {
6    render() {
7      return(
8        <div class="post-artikel">
9          <h2>Daftar Artikel</h2>
10         <Post />
11       </div>
12     )
13   }
14 }
15
16 export default BlogPost;
```

18 Perhatikan hasil tampilan browser, apa yang terjadi?



Muat Data Dinamis

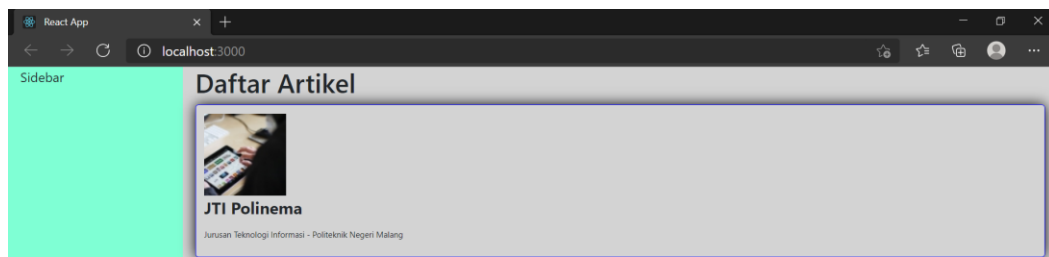
- 19 Pada *statefull component* **BlogPost.jsx**, tambahkan parameter yang ingin dilempar ke *stateless component* untuk ditampilkan

```
src > container > BlogPost > BlogPost.jsx > BlogPost > render
1  import React, {Component} from "react";
2  import './BlogPost.css';
3  import Post from "../../component/BlogPost/Post";
4
5  class BlogPost extends Component {
6    render() {
7      return(
8        <div className="post-artikel">
9          <h2>Daftar Artikel</h2>
10         <Post judul="JTI Polinema" isi="Jurusan Teknologi Informasi - Politeknik Negeri Malang"/>
11       </div>
12     )
13   }
14 }
15
16 export default BlogPost;
```

- 20 Setelah itu pada *stateless component* **Post.jsx** tangkap parameter yang dilempar oleh *statefull component* seperti pada Gambar dan lihat pada browser apa yang terjadi!.

```
src > component > BlogPost > Post.jsx > Post
1  import React from "react";
2
3  const Post = (props) => {
4    return(
5      <div className="artikel">
6        <div className="gambar-artikel">
7          
8        </div>
9        <div className="konten-artikel">
10         <div className="judul-artikel">{props.judul}</div>
11         <p className="isi-artikel">{props.isi}</p>
12       </div>
13     </div>
14   )
15 }
16
17 export default Post;
```

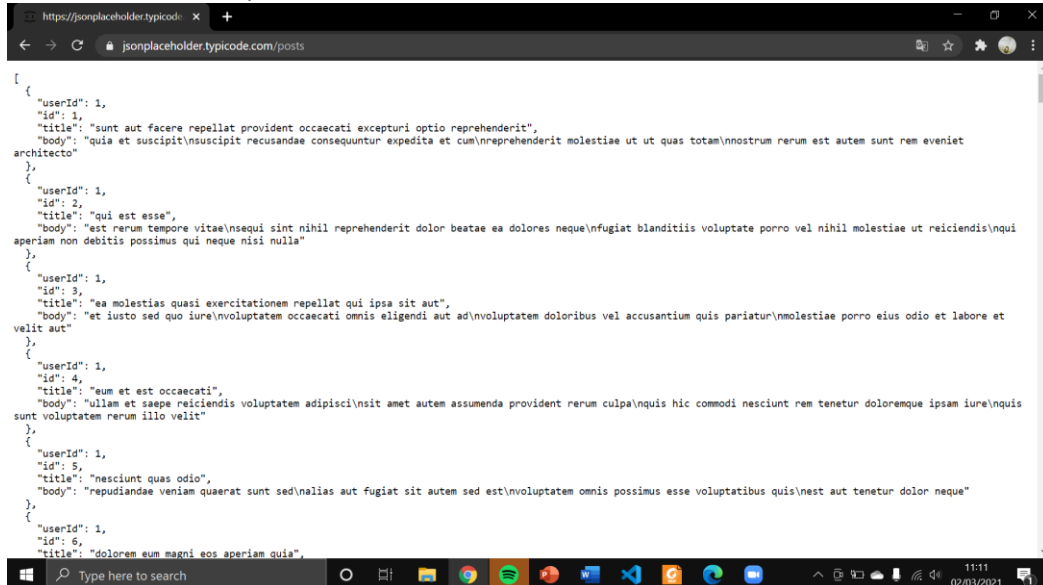
- 21 Simpan, dan amati apa yang terjadi pada browser kalian!.



Mengambil Data Post/Artikel dari API

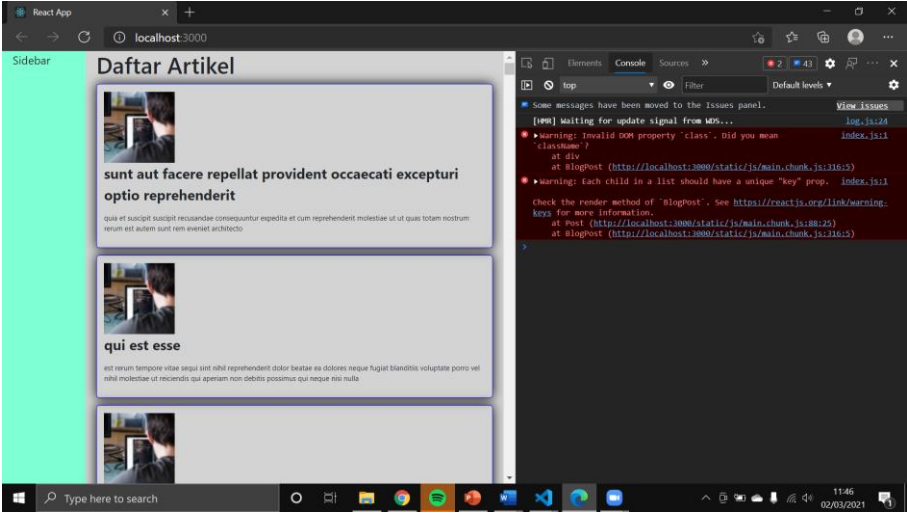


22 Gunakan **state** untuk menyimpan data hasil *request* dari API

23 data API yang akan kita gunakan adalah data *dummy* dari <https://jsonplaceholder.typicode.com/posts>, dimana memiliki 4 element data yaitu *userid*, *id*, *title*, *body*



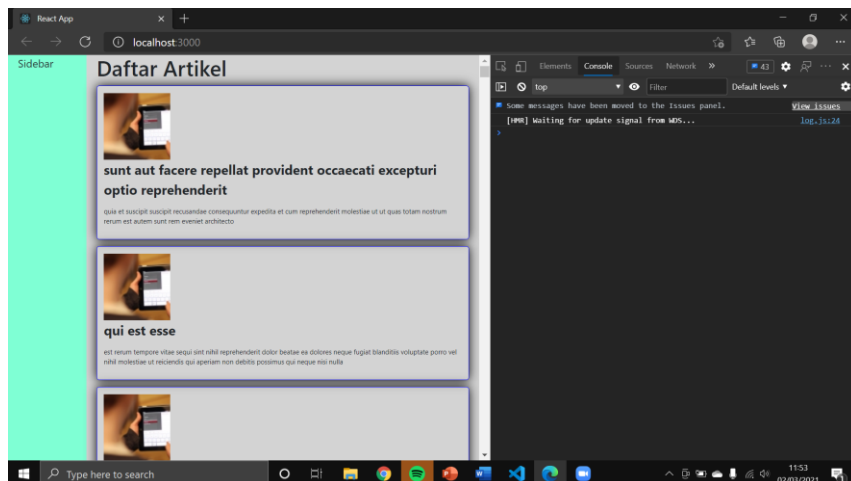
24 Edit pada *statefull component* **BlogPost.jsx** seperti pada Gambar 1.18 dan perhatikan dengan seksama akan penjelasan di beberapa baris kode program tersebut.

```
src > container > BlogPost > BlogPost.jsx > BlogPost > render > state.listArtikel.map() callback
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     // komponen state dari React untuk statefull component
8     listArtikel: [] // variabel yang digunakan untuk menyimpan data API
9   }
10
11   componentDidMount() { // komponen untuk mengecek ketika component telah di mount-ing, maka panggil API
12     fetch('https://jsonplaceholder.typicode.com/posts') // alamat URL API yang ingin kita ambil datanya
13     .then(response => response.json()) // ubah response data dari URL API menjadi sebuah data json
14     .then(jsonHasilAmbilDariAPI => { // data json hasil ambil dari API bisa kita masukkan ke dalam listArtikel pada state
15       this.setState({
16         listArtikel: jsonHasilAmbilDariAPI
17       })
18     })
19   }
20
21   render() {
22     return(
23       <div class="post-artikel">
24         <h2>Daftar Artikel</h2>
25         {
26           this.state.listArtikel.map(artikel => { // looping dan masukkan untuk setiap data yang ada di listArtikel ke variabel ar
27             return <Post judul={artikel.title} isi={artikel.body}/> // mappingkan data json dari API sesuai dengan kategori
28           })
29         }
30         <Post judul="JTI Polinema" isi="Jurusan Teknologi Informasi - Politeknik Negeri Malang"/>
31       </div>
32     )
33   }
34 }
35
36 export default BlogPost;
```


25	<p>Lihat hasilnya pada browser. Kemudian klik kanan pada browser pilih "inspect element" kemudian pilih tab "console". <i>Refresh</i> browser dan amati apa yang terjadi.</p>
26	<p>Jika terlihat seperti pada Gambar 1.19, maka terjadi kesalahan pada program yang kita buat</p> 
27	<p>Jika terjadi hal demikian, hal ini terjadi karena dalam react "class" dalam tag html harus ditulis menjadi "className". selain itu, pada <i>statefull component</i> yang dinamis, harus ada "UNIQUE KEY" pada tiap komponen yang diproses sehingga komponen perlu diberi UNIQUE KEY.</p> 
28	<p>UNIQUE KEY dapat diambil dari element yang ada pada data API yang sudah kita ambil (contoh saat ini adalah element id pada data API (userid, id, title, body) yang akan kita gunakan untuk UNIQUE KEY.</p> 

29

Simpan dan lihat apa yang terjadi pada *console* browser



Pertanyaan Praktikum 1

a. Pada langkah 8, sekarang coba kalian ganti class **container** dengan **container-fluid** atau sebaliknya pada file "**public/index.html**" dan lihat apa perbedaannya.

1. Tampilan seperti apa yang kalian temukan setelah mencoba mengganti nama class tersebut?

○ **Class container :**



Class container-fluid:



2. Apa perbedaan dari **container** dan **container-fluid** ?

- **Container** : layoutnya bermodel box, layoutnya tidak full-width layar browser
- **Container-fluid** : layoutnya bermodel full-width layer browser

b. Jika kita ingin meng-*import* suatu *component* contoh *component bootstrap*, akan tetapi *component* dalam tersebut belum terdapat pada module ReactJS. Apa yang akan dilakukan untuk dapat menggunakan component tersebut? Bagaimana caranya?

- **Jika component tersebut belum terdapat pada module ReactJS kita harus menginstall nya terlebih dahulu**

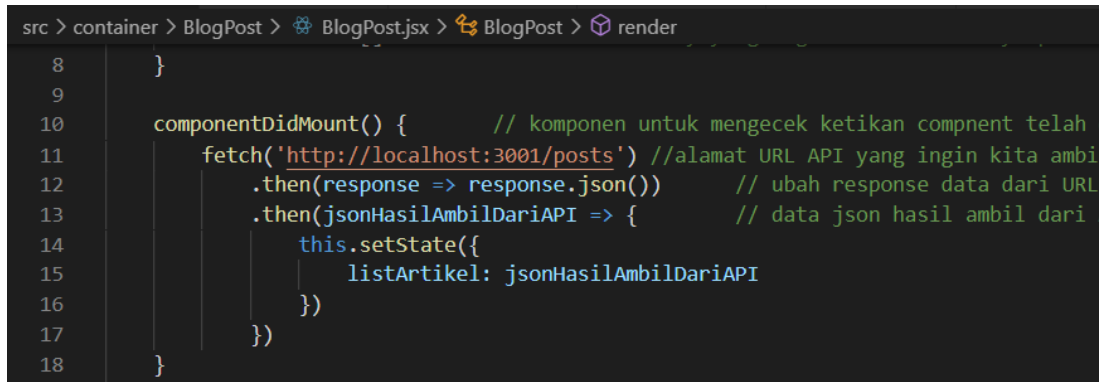
Praktikum 2

Interaksi dengan API menggunakan Fake API

No	Keterangan
Install Fake API (JSON Server)	
1	<p>Install pada direktori project reactjs kita dengan perintah npm install -g json-server</p> 
2	<p>Copy-kan file json listArtikel.json yang sudah ada pada direktori project reactjs kita</p> 
3	<p>Buka cmd baru pada direktori project, lalu ketik perintah json-server --watch listArtikel.json --port 3001.</p>
4	<p>Apabila pada cmd tampil seperti Gambar, maka server <i>Fake API</i> local kita telah siap</p> 
5	<p>Kita cek <i>url resource</i> yang ada pada Fake API server ke browser apakah bisa diakses. Ketik url http://localhost:3001/posts pada browser.</p> 

6

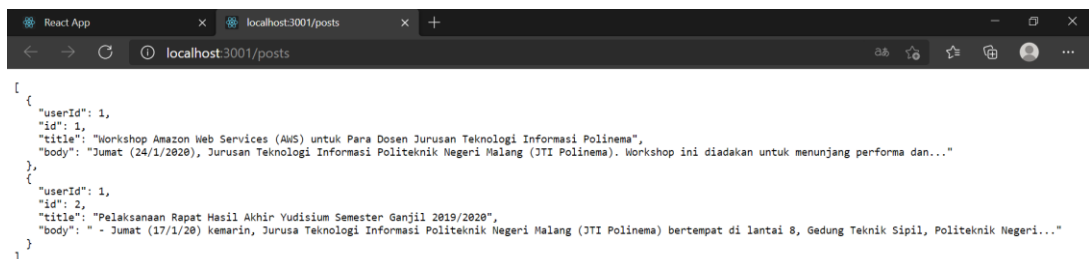
Untuk memastikan lagi, kita edit *statefull component* **BlogPost** (Gambar 1.18) pada baris 11. Kita ganti url API dari <https://jsonplaceholder.typicode.com/posts> menjadi <http://localhost:3001/posts>



```
src > container > BlogPost > BlogPost.jsx > BlogPost > render
8      }
9
10     componentDidMount() { // komponen untuk mengecek ketikan compnent telah d
11         fetch('http://localhost:3001/posts') //alamat URL API yang ingin kita ambil
12         .then(response => response.json()) // ubah response data dari URL
13         .then(jsonHasilAmbilDariAPI => { // data json hasil ambil dari A
14             this.setState({
15                 listArtikel: jsonHasilAmbilDariAPI
16             })
17         })
18     }
```

7

Simpan perubahan dan amati apa yang terjadi.




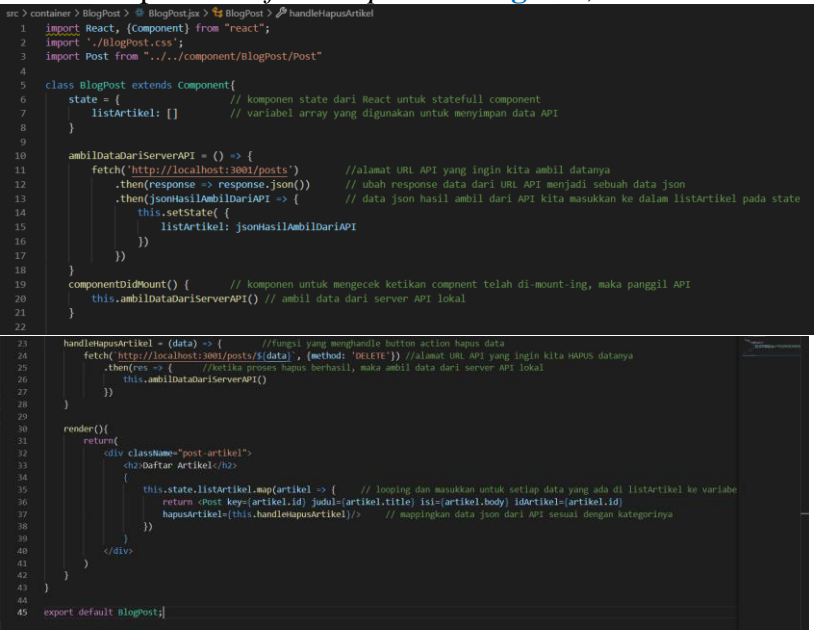
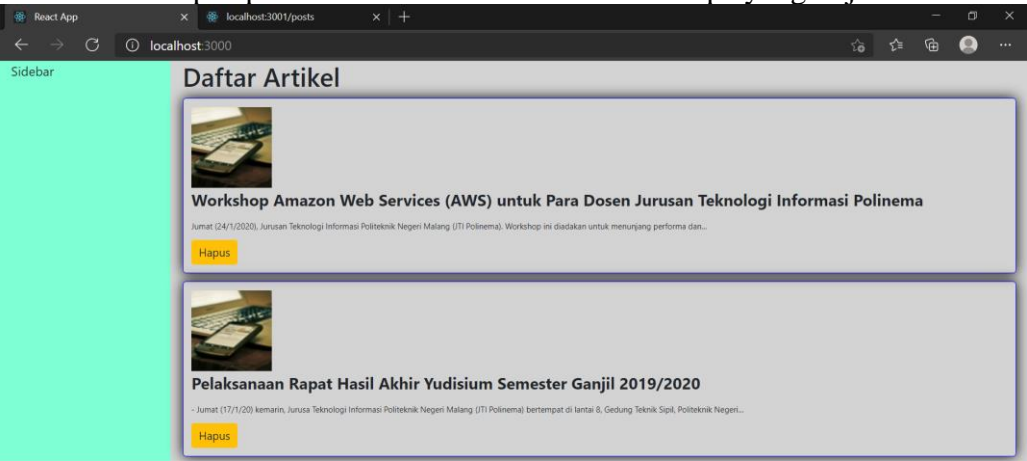
```
[
  {
    "userId": 1,
    "id": 1,
    "title": "Workshop Amazon Web Services (AWS) untuk Para Dosen Jurusan Teknologi Informasi Polinema",
    "body": "Jumat (24/1/2020), Jurusan Teknologi Informasi Politeknik Negeri Malang (JTI Polinema). Workshop ini diadakan untuk menunjang performa dan..."
  },
  {
    "userId": 1,
    "id": 2,
    "title": "Pelaksanaan Rapat Hasil Akhir Yudisium Semester Ganjil 2019/2020",
    "body": "- Jumat (17/1/20) kemarin, Jurusan Teknologi Informasi Politeknik Negeri Malang (JTI Polinema) bertempat di lantai 8, Gedung Teknik Sipil, Politeknik Negeri..."
  }
]
```

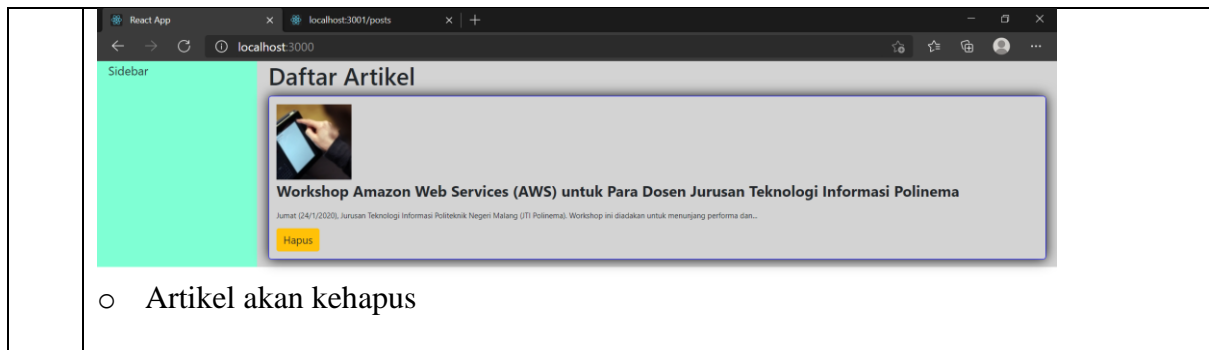
Pertanyaan Praktikum 2

- Kenapa *json-server* dijalankan pada port 3001? Kenapa tidak sama-sama dijalankan pada port 3000 seperti project react yang sudah kita buat?
 - Karena pada port 3000 telah digunakan untuk operasi GET pada <https://jsonplaceholder.typicode.com/posts>, sedangkan untuk mengambil data menggunakan port yang berbeda
- Bagaimana jadinya kalau kita ganti *port json-server* menjadi 3000?
 - Bisa jika port json-server diganti menjadi 3000 untuk mengambil data local yang menjadi server tetapi job yang sebelumnya harus dilakukan terminate kemudian run pada port 3000

Praktikum 3

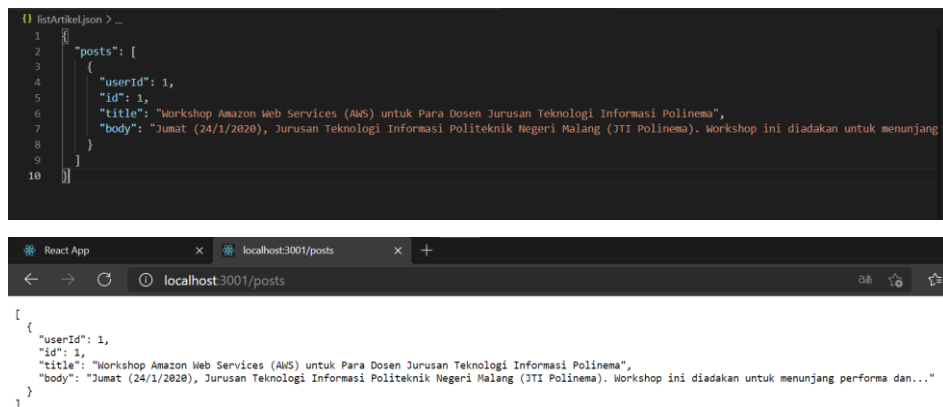
Interaksi dengan API menggunakan method DELETE

No	Keterangan
1	<p>Buka <i>stateless component</i> Post. Tambahkan 1 baris kode program pada baris 10</p> 
2	<p>Kemudian pada <i>statefull component</i> BlogPost, modifikasi kode program sebelumnya</p> 
3	<p>Klik tombol hapus pada list artikel di browser. Amati apa yang terjadi.</p> 



Pertanyaan Praktikum 3

- Apa yang terjadi setelah kalian klik tombol hapus?
 - Artikel akan terhapus
- Perhatikan file `listArtikel.json`, apa yang terjadi pada file tersebut? Kenapa demikian?
 - Data list artikel akan berkurang atau terhapus karena pada Langkah sebelumnya button hapus di klik



- Fungsi `handleHapusArtikel` itu untuk apa?
 - Fungsi `handleHapusArtikel` digunakan untuk menghandle button action hapus data json
- Jelaskan perbedaan fungsi `componentDidMount()` pada Gambar 1.18 dengan fungsi `componentDidMount()` pada Gambar 3.2 ?
 - `componentDidMount()` pada gambar 1.18 :
untuk mengecek component yang telah di-mount-ing, memanggil data API dari alamat URL API lalu mengubah respon data dari URL API menjadi sebuah data JSON. Data JSON hasil ambil dari API dimasukkan ke dalam `listArtikel` pada state
 - `componentDidMount()` pada gambar 3.2 :
untuk mengecek component yang telah di-mount-ing, ambil data dari server API lokal

PRAKTIKUM 4

Interaksi dengan API menggunakan method POST

No	Keterangan
1	<p>Buka <i>statefullcomponent</i> BlogPost, dan modifikasi pada fungsi render() untuk menampilkan <i>form input</i> artikel yang berisi judul dan isi berita</p> <pre>src > container > BlogPost > BlogPost.jsx > BlogPost > render 61 render(){ 62 return(63 <div className="post-artikel"> 64 <div className="form pb-2 border-bottom"> 65 <div className="form-group row"> 66 <label htmlFor="title" className="col-sm-2 col-form-label">Judul</label> 67 <div className="col-sm-10"> 68 <input type="text" className="form-control" id="title" name="title" onChange={this.handleTambahArtikel}/> 69 </div> 70 </div> 71 <div className="form-group row"> 72 <label htmlFor="body" className="col-sm-2 col-form-label">Isi</label> 73 <div className="col-sm-10"> 74 <textarea className="form-control" id="body" name="body" rows="3" onChange={this.handleTambahArtikel}></textarea> 75 </div> 76 </div> 77 <button type="submit" className="btn btn-primary" onClick={this.handleTambahSimpan}>Simpan</button> 78 </div> 79 <h2>Daftar Artikel</h2> 80 <div> 81 this.state.listArtikel.map(artikel => { // looping dan masukkan untuk setiap data yang ada di listArtikel ke variabel 82 return <Post key={artikel.id} judul={artikel.title} isi={artikel.body} idArtikel={artikel.id} 83 hapusArtikel={this.handleHapusArtikel}/> // mappingkan data json dari API sesuai dengan kategorinya 84 }) 85 </div> 86 </div> 87) 88 } 89 90</pre>
2	<p>Kemudian modifikasi BlogPost untuk bagian <i>state</i> dan <i>request</i> API dari server</p> <pre>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 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 ambilDataDariServerAPI = () => { 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 }</pre>

3

Tambahkan untuk *handle* form tambah data artikel

```
src > container > BlogPost > BlogPost.jsx > BlogPost > componentDidMount
25
26 // komponen untuk mengecek ketikan compnent telah di-mount-ing, maka panggil API
27 this.ambilDataDariServerAPI() // ambil data dari server API lokal
28
29
30 handleHapusArtikel = (data) => { //fungsi yang menghandle button action hapus data
31   fetch('http://localhost:3001/posts/${data}', {method: 'DELETE'}) //alamat URL API yang ingin kita HAPUS datanya
32   .then(res => { //ketika proses hapus berhasil, maka ambil data dari server API lokal
33     this.ambilDataDariServerAPI()
34   })
35 }
36
37 handleTambahArtikel = (event) => { // fungsi untuk meng-handle form tambah data artikel
38   let formInsertArtikel = {...this.state.insertArtikel}; // cloning data state insertArtikel ke dalam variabel formInsertArtikel
39   let timestamp = new Date().getTime(); // digunakan untuk menyimpan waktu (Sebagai ID artikel)
40   formInsertArtikel['id'] = timestamp;
41   formInsertArtikel[event.target.name] = event.target.value; //menyimpan data onchange ke formInsertArtikel sesuai dengan target yang
42   this.setState({
43     insertArtikel: formInsertArtikel
44   })
45 }
46
47 handleTombolSimpan = () => { // fungsi untuk meng-handle tombol simpan
48   fetch('http://localhost:3001/posts', {
49     method: 'post', //method POST untuk input/insert data
50     headers: {
51       'Accept': 'application/json',
52       'Content-Type': 'application/json'
53     },
54     body: JSON.stringify(this.state.insertArtikel) // kirimkan ke body request untuk data artikel yang akan ditambahkan (insert)
55   })
56   .then((Response) => {
57     this.ambilDataDariServerAPI(); // reload / refresh data
58   });
59 }
60
```

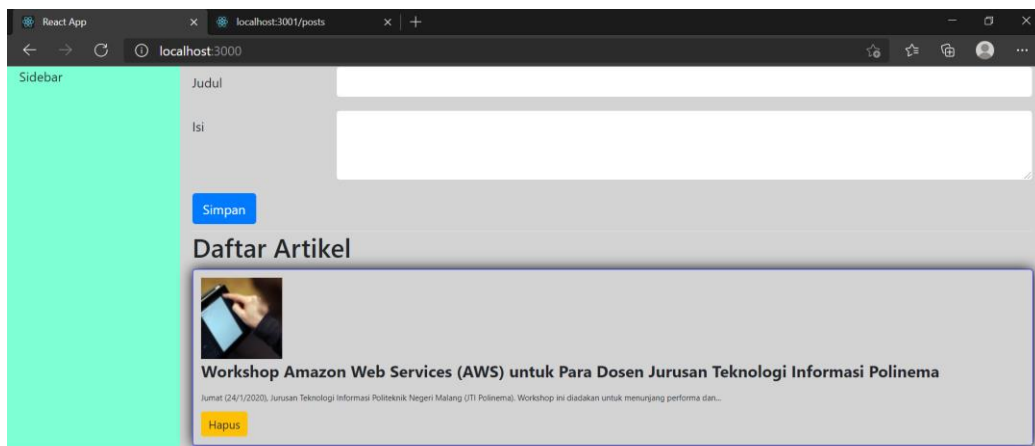
4

Langkah terakhir tambahkan fungsi untuk handle tombol simpan artikel

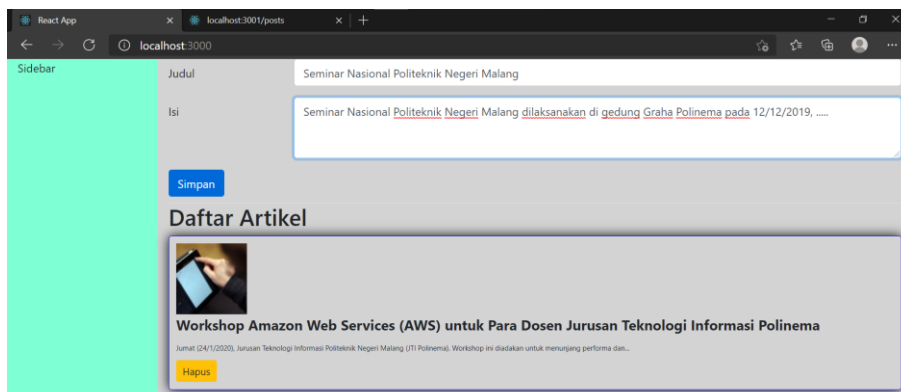
```
src > container > BlogPost > BlogPost.jsx > BlogPost > ambilDataDariServerAPI
46
47 handleTombolSimpan = () => { // fungsi untuk meng-handle tombol simpan
48   fetch('http://localhost:3001/posts', {
49     method: 'post', //method POST untuk input/insert data
50     headers: {
51       'Accept': 'application/json',
52       'Content-Type': 'application/json'
53     },
54     body: JSON.stringify(this.state.insertArtikel) // kirimkan ke body request untuk data artikel yang akan ditambahkan (insert)
55   })
56   .then((Response) => {
57     this.ambilDataDariServerAPI(); // reload / refresh data
58   });
59 }
60
```

5

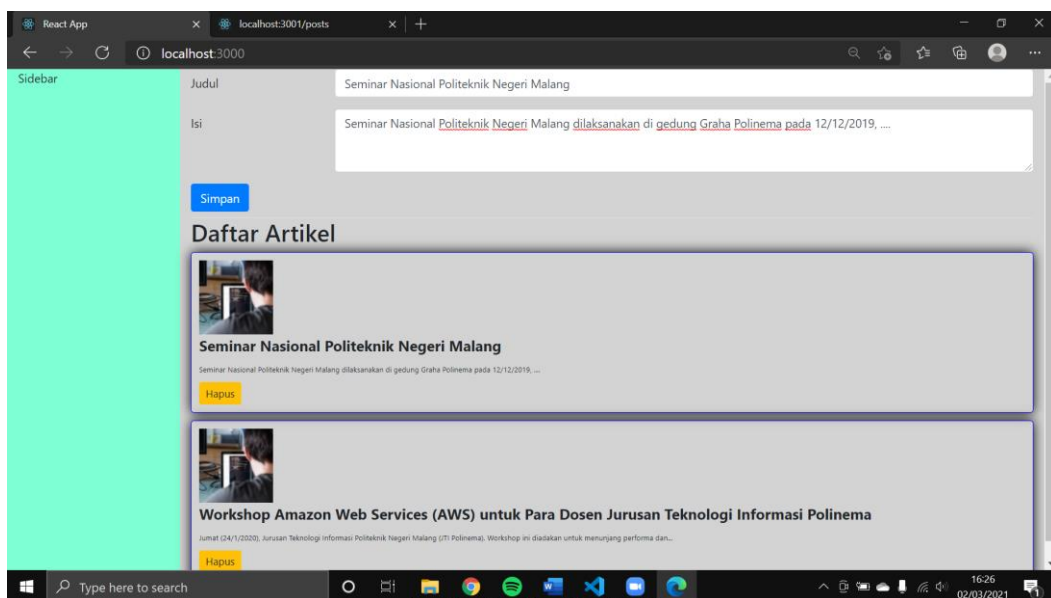
Simpan, lakukan percobaan penambahan data, dan amati perubahannya



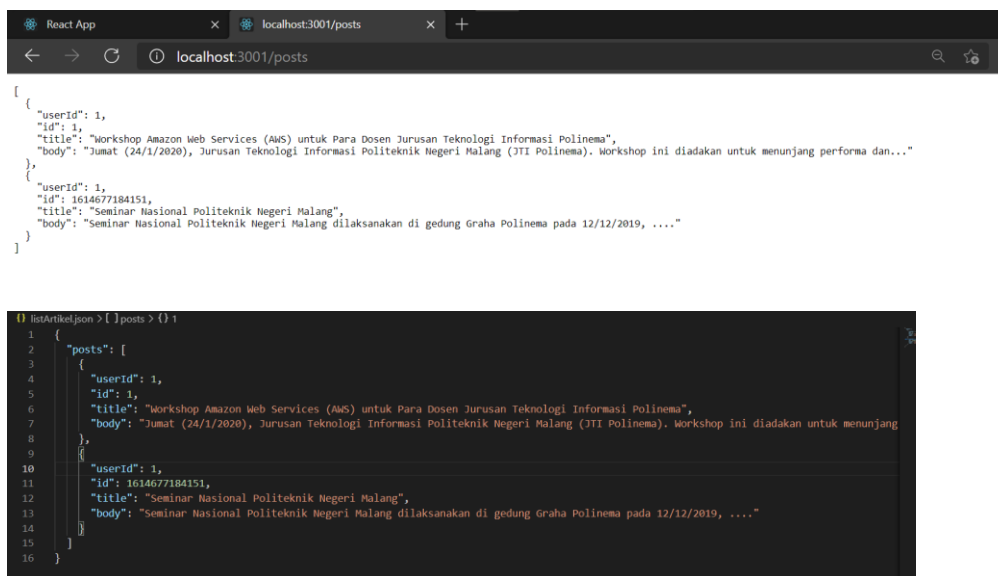
Penambahan Data :



Refresh data :



Pada json data bertambah :

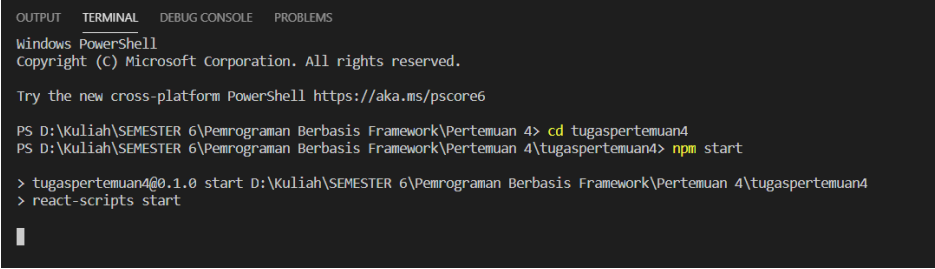
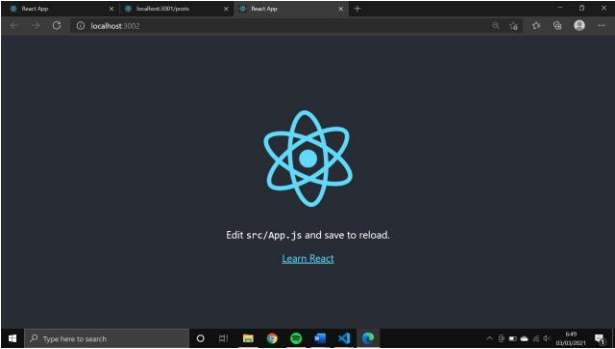


Pertanyaan Praktikum 4

- a. Jelaskan apa yang terjadi pada file **listArtikel.json** sebelum dan setelah melakukan penambahan data?
- **Sebelum penambahan data, data json tetap, setelah penambahan data, data list artikel dan data json nya bertambah**
- b. Data yang ditampilkan di browser adalah data terbaru berada di posisi atas dan data lama berada di bawah, sedangkan pada file **listArtikel.json** data terbaru malah berada di bawah. Jelaskan mengapa demikian?
- **Pada browser id disorting secara desc, sedangkan pada file listArtikel.json tampil secara asc**

TUGAS PRAKTIKUM

Membuat program menggunakan Fake API (JSON Server) tentang pendataan Mahasiswa aktif/cuti/lulus di Jurusan Teknologi Informasi. Atribut-atribut yang ada dari mahasiswa adalah NIM, nama, alamat, no hp, tahun Angkatan, dan status. Buat aplikasi yang menggunakan API dengan method GET, DELETE, dan POST.

No	Langkah
1	<p>Buat project react lalu jalankan dengan perintah “npm start” pada terminal vscode</p>  

2	Buat folder baru bernama “BlogPost” pada folder container (<i>statefull component</i>).
3	Buat file BlogPost.jsx dan BlogPost.css di dalam folder “BlogPost” , 
4	Buka BlogPost.jsx dan ketikan kode seperti ini : <pre data-bbox="284 757 1342 1877"> src > container > BlogPost > BlogPost.jsx > BlogPost > 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 listMhs:[], 8 insertArtikel: { 9 userId: 1, 10 id: 1, 11 nim: "", 12 nama: "", 13 alamat: "", 14 hp: "", 15 angkatan: "", 16 status: "" 17 } 18 } 19 20 ambilDataDariServerAPI = () => { 21 fetch('http://localhost:3001/posts?_sort=id&_order=desc') 22 .then(response => response.json()) 23 .then(jsonHasilAmbilDariAPI => { 24 this.setState({ 25 listMhs: jsonHasilAmbilDariAPI 26 }) 27 }) 28 } 29 30 componentDidMount() { 31 this.ambilDataDariServerAPI() 32 } 33 </pre>

```

34   handleHapusArtikel = (data) => {
35       fetch(`http://localhost:3001/posts/${data}`, {method: 'DELETE'})
36       .then(res => {
37           this.ambilDataDariServerAPI()
38       })
39   }
40
41   handleTambahArtikel = (event) =>{
42       let formInsertArtikel = {...this.state.insertArtikel};
43       let timestamp = new Date().getTime();
44       formInsertArtikel['id'] = timestamp;
45       formInsertArtikel[event.target.name] = event.target.value;
46       this.setState({
47           insertArtikel: formInsertArtikel
48       });
49   }
50
51   handleTombolSimpan = () => {
52       fetch('http://localhost:3001/posts',{
53           method: 'post',
54           headers: {
55               'Accept': 'applicarion/json',
56               'Content-Type': 'application/json'
57           },
58           body: JSON.stringify(this.state.insertArtikel)
59       })
60       .then((Response) => {
61           this.ambilDataDariServerAPI();
62       });
63   }
64

```

```

65   render(){
66       return(
67           <div className="post-artikel">
68               <div className="form pb-2 border-bottom">
69                   <div className="form-group row">
70                       <label htmlFor="nim" className="col-sm-2 col-form-label">NIM</label>
71                       <div className="col-sm-10">
72                           <input type="text" className="form-control" id="nim" name="nim" onChange={this.handleTambahArtikel}/>
73                       </div>
74                   </div>
75                   <div className="form-group row">
76                       <label htmlFor="nama" className="col-sm-2 col-form-label">Nama</label>
77                       <div className="col-sm-10">
78                           <input type="text" className="form-control" id="nama" name="nama" onChange={this.handleTambahArtikel}/>
79                       </div>
80                   </div>
81                   <div className="form-group row">
82                       <label htmlFor="alamat" className="col-sm-2 col-form-label">Alamat</label>
83                       <div className="col-sm-10">
84                           <textarea className="form-control" id="alamat" name="alamat" rows="3" onChange={this.handleTambahArtikel}></text.
85                       </div>
86                   </div>
87                   <div className="form-group row">
88                       <label htmlFor="hp" className="col-sm-2 col-form-label">HP</label>
89                       <div className="col-sm-10">
90                           <input type="text" className="form-control" id="hp" name="hp" onChange={this.handleTambahArtikel}/>
91                       </div>
92                   </div>
93                   <div className="form-group row">
94                       <label htmlFor="angkatan" className="col-sm-2 col-form-label">angkatan</label>
95                       <div className="col-sm-10">
96                           <input type="text" className="form-control" id="angkatan" name="angkatan" onChange={this.handleTambahArtikel}/>
97                       </div>
98                   </div>

```

	<pre> 99 <div className="form-group row"> 100 <label htmlFor="status" className="col-sm-2 col-form-label">status</label> 101 <div className="col-sm-10"> 102 <input type="text" className="form-control" id="status" name="status" onChange={this.handleTambahArtikel}/> 103 </div> 104 </div> 105 <button type="submit" className="btn btn-primary" onClick={this.handleTambahArtikel}>Simpan</button> 106 </div> 107 <h2>Daftar Mahasiswa</h2> 108 { 109 this.state.listMhs.map(artikel => { 110 return <Post key={artikel.id} nim={artikel.nim} nama={artikel.nama} alamat={artikel.alamat} hp={artikel.hp} angkatan={artikel.angkatan}/> 111 }) 112 } 113 </div> 114 } 115 } 116 } 117 118 export default BlogPost; </pre>
5	<p>Pada index.js ketikkan kode seperti ini :</p> <pre> src > JS index.js 1 import React from 'react'; 2 import ReactDOM from 'react-dom'; 3 import 'bootstrap/dist/css/bootstrap.min.css'; 4 import './index.css'; 5 import * as serviceWorker from './serviceWorker'; 6 import BlogPost from './container/BlogPost/BlogPost'; 7 // import App from './App'; 8 // import reportWebVitals from './reportWebVitals'; 9 10 ReactDOM.render(<BlogPost />, document.getElementById('content')); 11 12 serviceWorker.unregister(); </pre>
6	<p>Import css bootstrap.min.css (css bootstrap yang sudah dikompresi) ke dalam index.js. Jika css tidak ditemukan, install lewat cmd dengan perintah “npm install bootstrap”</p> <pre> PS D:\Kuliah\SEMESTER 6\Pemrograman Berbasis Framework\Pertemuan 4> npm install bootstrap npm WARN saveError ENOENT: no such file or directory, open 'D:\Kuliah\SEMESTER 6\Pemrograman Berbasis Framework\Pertemuan 4\package.json' npm notice created a lockfile as package-lock.json. You should commit this file. npm WARN enoent ENOENT: no such file or directory, open 'D:\Kuliah\SEMESTER 6\Pemrograman Berbasis Framework\Pertemuan 4\package.json' npm WARN bootstrap@4.6.0 requires a peer of jquery@1.9.1 - 3 but none is installed. You must install peer dependencies yourself. npm WARN bootstrap@4.6.0 requires a peer of popper.js@1.16.1 but none is installed. You must install peer dependencies yourself. npm WARN Pertemuan 4 No description. npm WARN Pertemuan 4 No repository field. npm WARN Pertemuan 4 No README data npm WARN Pertemuan 4 No license field. </pre>
7	<p>Modifikasi file index.html pada folder "public"</p> <pre> public > index.html > ... 1 <!DOCTYPE html> 2 <html lang="en"> 3 <head> 4 <meta charset="utf-8" /> 5 <link rel="icon" href="%PUBLIC_URL%/favicon.ico" /> 6 <meta name="viewport" content="width=device-width, initial-scale=1" /> 7 <meta name="theme-color" content="#000000" /> 8 <meta name="description" content="Web site created using create-react-app"/> 9 <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" /> 10 <link rel="manifest" href="%PUBLIC_URL%/manifest.json" /> 11 <title>LIST DATA MAHASISWA</title> 12 </head> 13 <body> 14 <noscript>You need to enable JavaScript to run this app.</noscript> 15 <!-- penerapan layout bisa dilihat di https://getbootstrap.com/docs/4.0/layout/overview --> 16 <!-- Containers adalah basic layout element di bootstrap dan DIPERLUKAN jika menggunakan default grid system bootstrap --> 17 <!-- ada 2 pilihan Containers 18 1. class container : untuk layout yang model box (tidak full-width layar browser) 19 2. class container-fluid : untuk layout model yang full-width layar browser 20 untuk lebih jelasnya, silahkan dicoba satu persatu. 21 Default kita saat ini, menggunakan yang full-width (container-fluid)--> 22 <div class="container-fluid"> 23 <div class="row"> 24 <div class="col-2" id="sidebar">Sidebar</div> 25 <div class="col-10" id="content"></div> 26 </div> 27 </div> 28 </body> 29 </html> 30 </pre>

8 Buka file **index.css** dan tambahkan code css

```
src > # index.css > #content
1  body {
2    margin: 0;
3    font-family: -apple-system, BlinkMacSystemFont, "Segoe UI", "Roboto", "Oxygen",
4      "Ubuntu", "Cantarell", "Fira Sans", "Droid Sans", "Helvetica Neue",
5      sans-serif;
6    -webkit-font-smoothing: antialiased;
7    -moz-osx-font-smoothing: grayscale;
8  }
9
10  code {
11    font-family: source-code-pro, Menlo, Monaco, Consolas, "Courier New",
12      monospace;
13  }
14
15  #sidebar {
16    background-color: aquamarine;
17  }
18
19  #content {
20    background-color: lightgray;
21  }
22
```

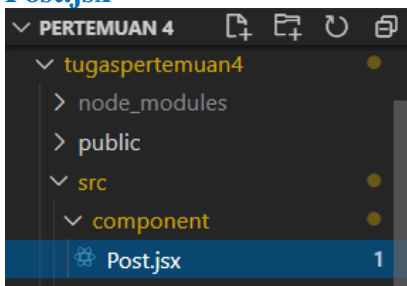
9 Tambahkan custom css ke **BlogPost.css**

```
src > container > BlogPost > # BlogPost.css > ...
1  .artikel {
2    width: 100%;
3    padding: 10px;
4    border: 1px solid blue;
5    border-radius: 4px;
6    margin-bottom: 10px;
7    box-shadow: 0 0 16px rgba(0, 0, 0, 0.5);
8    display: flex;
9  }
10
11  .gambar-artikel {
12    height: 80px;
13    width: 80px;
14    margin-right: 20px;
15    vertical-align: top;
16  }
17
18  .gambar-artikel img {
19    width: 100%;

```

```
src > container > BlogPost > # BlogPost.css > ...
19  width: 100%;
20  height: 100%;
21  object-fit: cover;
22  }
23
24  .konten-artikel {
25    flex: 1;
26  }
27
28  .konten-artikel div.judul-artikel {
29    font-size: 20px;
30    font-weight: bold;
31    margin-bottom: 10px;
32  }
33
34  .konten-artikel p.isi-artikel {
35    font-size: 16px;
36    margin-bottom: 10px;
37  }
38
```

10 Buat folder **BlogPost** pada folder **component** (*stateless component*), lalu buat file **Post.jsx**



11 Potong (*cut*) baris 9-17 pada *statefull component* **BlogPost.jsx** ke *stateless component* **Post.jsx**, dan modifikasi **Post.jsx**

	 <pre> 1 import React from "react"; 2 3 const Post = (props) => { 4 return(5 <div className="artikel"> 6 <div className="gambar-artikel"> 7 8 </div> 9 <div className="konten-artikel"> 10 <div className="judul">{props.judul}</div> 11 <p className="nama">{props.nama}</p> 12 <p className="alamat">{props.alamat}</p> 13 <p className="telfon">{props.hp}</p> 14 <p className="angkatan">{props.angkatan}</p> 15 <p className="status">{props.status}</p> 16 <button className="btn btn-sm btn-warning" onClick={() => props.hapusArtikel(props.idArtikel)}>Hapus</button> 17 </div> 18 </div> 19) 20 } 21 22 export default Post; </pre>
12	<p>Install pada direktori project reactjs kita dengan perintah npm install -g json-server</p>  <pre> PS D:\Kuliah\SEMESTER 6\Pemrograman Berbasis Framework\Pertemuan 4> npm install -g json-server [.....] loadIdealTree:loadAllDepsIntoIdealTree: sill install loadIdealTree </pre>
13	<p>Buat file json listDataMHS.json pada direktori project tugaspertemuan4</p>  <pre> 1 { 2 "mahasiswa": [3 { 4 "userId": 1, 5 "id": 1, 6 "NIM": "1841720164", 7 "nama": "LIVIA YURIKE KHURIL MAULA", 8 "alamat": "Jl. PIRANHA ATAS NO 264 MALANG", 9 "hp": "082228773286", 10 "angkatan": 2018, 11 "status": "AKTIF" 12 }, 13 { 14 "userId": 2, 15 "id": 2, 16 "NIM": "1741720214", 17 "nama": "IQBAL RAMADANI", 18 "alamat": "Malang", 19 "hp": "08123456789", 20 "angkatan": 2017, 21 "status": "CUTI" 22 }, 23 { 24 "userId": 3, 25 "id": 3, </pre>
14	<p>Buka cmd baru pada direktori project, lalu ketik perintah json-server --watch listArtikel.json --port 3001</p>  <pre> PS D:\Kuliah\SEMESTER 6\Pemrograman Berbasis Framework\Pertemuan 4> npx json-server -watch listDataMHS.json --port 3003 [.....] fetchMetadata: sill resolveWithNewModule ms@2.0.0 checking installable status </pre>
15	<p>Apabila pada terminal tampil seperti Gambar, maka server <i>Fake API</i> local kita telah siap</p>

```
OUTPUT  TERMINAL  DEBUG CONSOLE  PROBLEMS  2: node

PS D:\Kuliah\SEMESTER 6\Pemrograman Berbasis Framework\Pertemuan 4\tugaspertemuan4> npx json-server --watch listMhs.json --port 3001
npx: installed 182 in 19.512s

\{^_</pre>
</div>
<div data-bbox="125 269 857 312" data-label="Text">
<p>16</p>
<p>Kita cek <i>url resource</i> yang ada pada Fake API server ke browser apakah bisa diakses. Ketik url <a href="http://localhost:3001/posts">http://localhost:3001/posts</a> pada browser</p>
</div>
<div data-bbox="175 317 528 648" data-label="Code-Block">
<pre>LIST DATA MAHASISWA  localhost:3000/mahasiswa

localhost:3001mahasiswa

[
  {
    "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,
  }
]</pre>
</div>
<div data-bbox="175 681 231 698" data-label="Text">
<p>Hasil :</p>
</div>
<div data-bbox="175 704 717 851" data-label="Form">
<div>
<div>LIST DATA MAHASISWA</div>
<div>localhost:3000</div>
<div>
<div>Sidebar</div>
<div>
NIM
Nama
Alamat
HP
angkatan
status
</div>
<div>
<button>Simpan</button>
</div>
<div>Daftar Mahasiswa</div>
</div>
</div>
</div>
```


Link Youtube :

https://www.youtube.com/watch?v=_qKFG5NmJUA

Link Github :

<https://github.com/liviayurike/Pemrograman-Berbasis-Framework/tree/master/Pertemuan%204>