

LAPORAN PRAKTIKUM
PEMROGRAMAN BERBASIS FRAMEWORK

Pertemuan Minggu-4

Modul 4 – Interaksi dengan API

(“*GET, fake API, DELETE, POST*”)



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

PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG


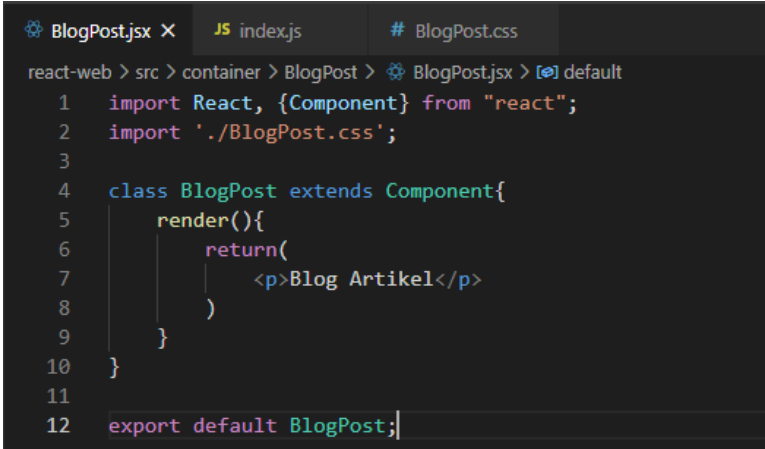
2021

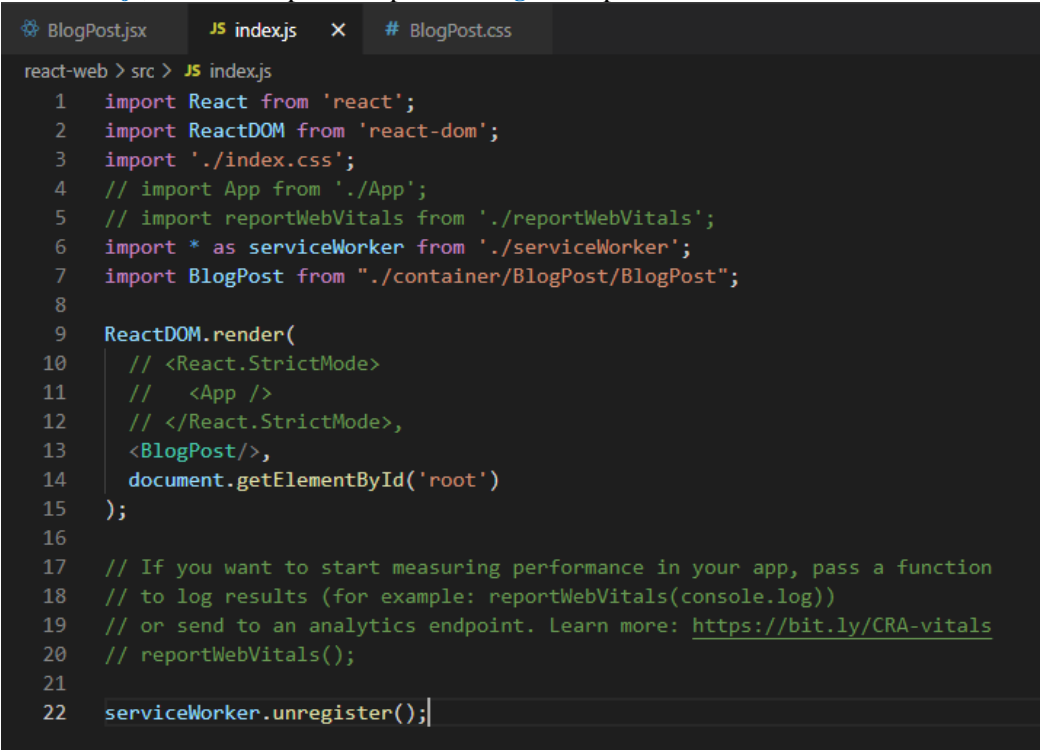
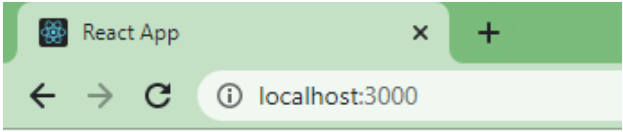
Praktikum 1

Interaksi dengan API menggunakan method GET

Langkah Praktikum

Dalam penggunaan API pada suatu website, maka data yang akan kita pakai adalah data dinamis dan memerlukan operasi logic pada ReactJS. Sehingga kita akan menggunakan *statefull component* ReactJS untuk membuatnya.

Langkah	Keterangan
1	Buka Project React pada pertemuan sebelumnya dan jalankan “ npm start ” menggunakan <i>cmd</i> dalam direktori tersebut.
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”, seperti pada Gambar 1.2.</p>  <p>Gambar 1.2. Pembuatan folder untuk statefull component</p>
4	<p>Buka file BlogPost.jsx dan ketikkan kode seperti Gambar 1.3</p>  <p>Gambar 1.3. Statefull component BlogPost</p>

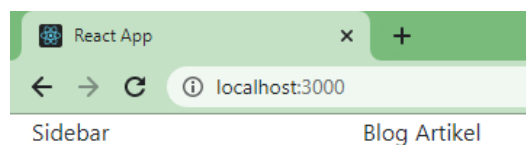
5	<p>Pada file <code>index.js</code>, lakukan import component <code>BlogPost</code> seperti Gambar 1.4.</p>  <pre> 1 import React from 'react'; 2 import ReactDOM from 'react-dom'; 3 import './index.css'; 4 // import App from './App'; 5 // import reportWebVitals from './reportWebVitals'; 6 import * as serviceWorker from './serviceWorker'; 7 import BlogPost from "../container/BlogPost/BlogPost"; 8 9 ReactDOM.render(10 // <React.StrictMode> 11 // <App /> 12 // </React.StrictMode>, 13 <BlogPost />, 14 document.getElementById('root') 15); 16 17 // If you want to start measuring performance in your app, pass a function 18 // to log results (for example: reportWebVitals(console.log)) 19 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals 20 // reportWebVitals(); 21 22 serviceWorker.unregister(); </pre> <p>Gambar 1.4. Import component BlogPost</p>
6	<p>Pada web browser akan tampil seperti Gambar 1.5.</p>  <p>Gambar 1.5. Tampilan Browser</p>
	<p>Tahapan selanjutnya adalah perbaikan tampilan sebuah website untuk mempercantik halaman website tersebut dengan menggunakan <code>Bootstrap</code> yang umum digunakan.</p>
7	<p>Import css <code>bootstrap.min.css</code> (css bootstrap yang sudah dikompresi) ke dalam <code>index.js</code> (seperti Gambar 1.6). Jika css tidak ditemukan, install lewat cmd dengan perintah “<code>npm install bootstrap</code>”</p>

	<div><pre>PS C:\Users\ASUS\ProgramFramework2021\minggu4\react-web> npm install bootstrap npm WARN tsutils@3.20.0 requires a peer of typescript@>=2.8.0 >= 3.2.0-dev >= 3.3 .0-dev >= 3.4.0-dev >= 3.5.0-dev >= 3.6.0-dev >= 3.6.0-beta >= 3.7.0-dev >= 3.7.0-beta but none is installed. You must install peer dependencies yourself. 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. Yo u must install peer dependencies yourself. npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@2.3.2 (node_modules\fsevents): npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@2.3.2: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"}) npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules\watchpack -chokidar2\node_modules\fsevents): npm WARN optional SKIPPING OPTIONAL DEPENDENCY: fsevents@1.2.13 (node_modules\webpack-dev-server\node_modules\fsevents): npm WARN notsup SKIPPING OPTIONAL DEPENDENCY: Unsupported platform for fsevents@1.2.13: wanted {"os":"darwin","arch":"any"} (current: {"os":"win32","arch":"x64"}) + bootstrap@4.6.0 added 1 package from 2 contributors and audited 1942 packages in 94.796s 128 packages are looking for funding run `npm fund` for details found 0 vulnerabilities</pre></div> <div><div>BlogPost.jsxJS index.jsindex.htmlBlogPost.css</div><pre>react-web > 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 App from './App'; 6 // import reportWebVitals from './reportWebVitals'; 7 import * as serviceWorker from './serviceWorker'; 8 import BlogPost from './container/BlogPost/BlogPost'; 9 10 ReactDOM.render(11 // <React.StrictMode> 12 // <App /> 13 // </React.StrictMode>, 14 <BlogPost />, 15 document.getElementById('content') 16); 17 18 // If you want to start measuring performance in your app, pass a function 19 // to log results (for example: reportWebVitals(console.log)) 20 // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals 21 // reportWebVitals(); 22 23 serviceWorker.unregister();</pre></div>
	Gambar 1.6. Import bootstrap css
8	Modifikasi file <code>index.html</code> pada folder " <code>public</code> " seperti Gambar 1.7. Cermati <i>code program</i> yang ada dalam gambar!.

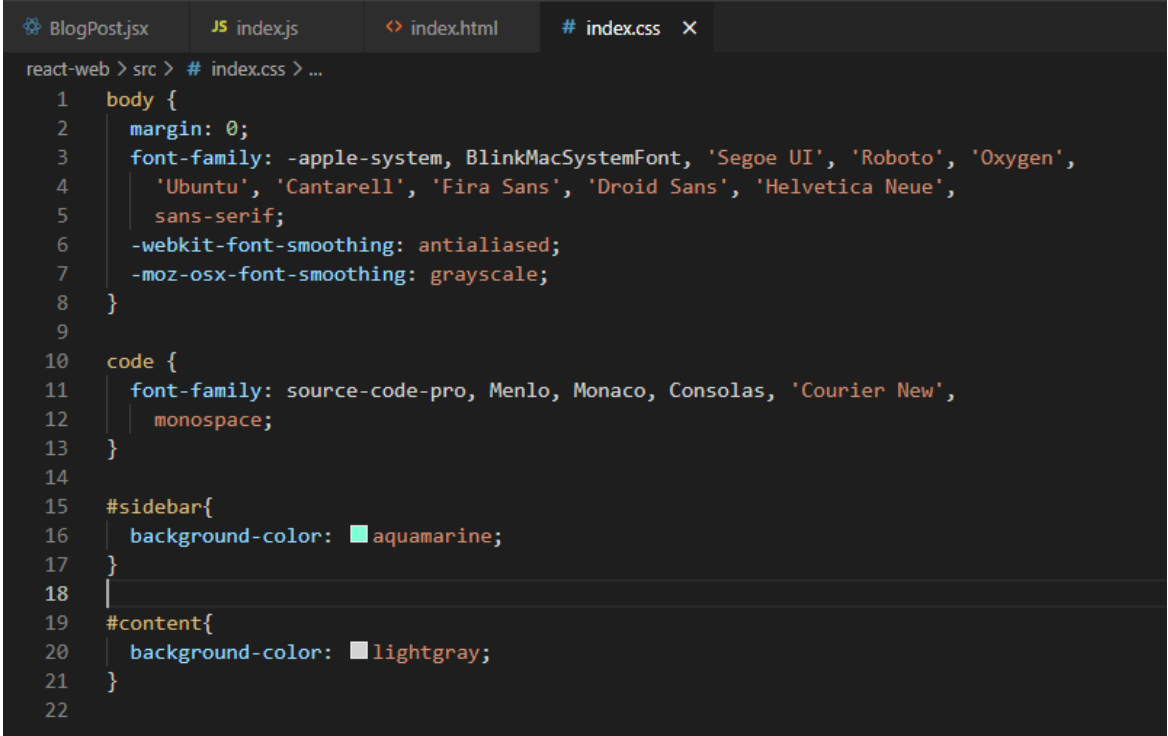

```
BlogPost.jsx  JS index.js  index.html X  # BlogPost.css
react-web > public > index.html > html > head > meta
1  <!DOCTYPE html>
2  <html lang="en">
3    <head>
4      <meta charset="utf-8" />
5      <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
6      <meta name="viewport" content="width=device-width, initial-scale=1" />
7      <meta name="theme-color" content="#000000" />
8      <meta
9        name="description"
10       content="Web site created using create-react-app"
11     />
12     <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
13     <!--
14       manifest.json provides metadata used when your web app is installed on a
15       user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manifest/
16     -->
17     <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
18     <!--
19       Notice the use of %PUBLIC_URL% in the tags above.
20       It will be replaced with the URL of the `public` folder during the build.
21       Only files inside the `public` folder can be referenced from the HTML.
22
23       Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
24       work correctly both with client-side routing and a non-root public URL.
25       Learn how to configure a non-root public URL by running `npm run build`.
26     -->
27     <title>React App</title>
28   </head>
29   <body>
30     <noscript>You need to enable JavaScript to run this app.</noscript>
31     <!-- <div id="root"></div> -->
32     <!--
33       This HTML file is a template.
34       If you open it directly in the browser, you will see an empty page.
35
36       You can add webfonts, meta tags, or analytics to this file.
37       The build step will place the bundled scripts into the <body> tag.
38
39       To begin the development, run `npm start` or `yarn start`.
40       To create a production bundle, use `npm run build` or `yarn build`.
41     -->
42     <div class="container-fluid">
43       <div class="row">
44         <div class="col-2" id="sidebar">Sidebar</div>
45         <div class="col-10" id="content">Blog Artikel</div>
46       </div>
47     </div>
48   </body>
49 </html>
50
```

Gambar 1.7. Modifikasi index.html

Amati tampilan yang ada pada browser (seperti Gambar 1.8)



Gambar 1.8. Tampilan hasil modifikasi

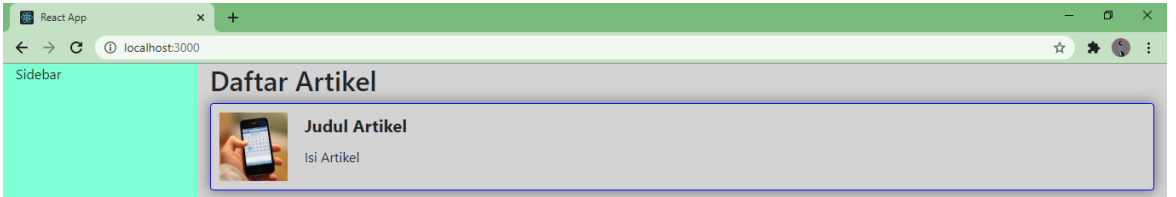
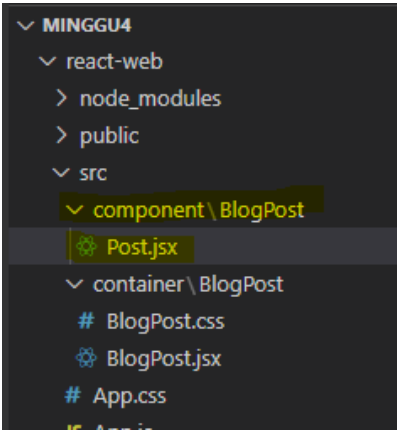
10	<p>Buka file index.css dan tambahkan code css seperti Gambar 1.9, untuk menambah sedikit style pada halaman web</p>  <pre> 1 body { 2 margin: 0; 3 font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen', 4 'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue', 5 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 </pre> <p>Gambar 1.9. Penambahan code css</p>
11	<p>Perhatikan kembali browser, dan lihat hasil tampilan seperti Gambar 1.10.</p> 
	<p>Kita ingin sebuah website memiliki tampilan seperti pada Gambar 1.1. Dengan minimal ada gambar artikel, judul, dan deskripsi artikel. Maka contoh <i>data dummy</i> yang akan kita pakai bisa menggunakan data dari http://placeimg.com contoh http://placeimg.com/120/120/any.</p> <p>Tahapan edit tampilan post artikel:</p>
12	<p>Ubah kode program untuk <i>statefull component</i> BlogPost.jsx menjadi seperti Gambar 1.11</p>

```
BlogPost.jsx X JS index.js < index.html # index.css
react-web > src > container > BlogPost > BlogPost.jsx > BlogPost > render
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;
```

Gambar 1.11. Edit kode program BlogPost

Tambahkan custom css ke **BlogPost.css** seperti Gambar 1.12

```
BlogPost.jsx # BlogPost.css X JS index.js < index.html # index.css
react-web > src > container > BlogPost > # BlogPost.css > .gambar-artikel
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%;
20   height: 100%;
21   object-fit: cover;
22 }
23
24 .konten-artikel {
25   flex: 1;
26 }
```

	<pre> 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 </pre> <p>Gambar 1.12. Edit kode program BlogPost.css</p>
14	<p>Perhatikan tampilan browser.</p> 
<p>Pemindahan dari <i>statefull component</i> ke <i>stateless component</i></p> <p>Pada component BlogPost (lihat Gambar 1.11), baris 9-17 merupakan daftar artikel yang bisa jadi dalam sebuah website berisi lebih dari 1 (satu) list artikel. Baris 9-17 dapat dipindah ke <i>stateless component</i> untuk dapat digunakan ulang (dipanggil kembali) karena fungsi dari bagian tersebut hanya mengembalikan deskripsi singkat artikel (bukan operasi logic).</p>	
15	<p>Buat folder BlogPost pada folder component (<i>stateless component</i>), lalu buat file Post.jsx</p> 
16	<p>Potong (<i>cut</i>) baris 9-17 pada <i>statefull component</i> BlogPost.jsx ke <i>stateless component</i> Post.jsx, dan modifikasi Post.jsx seperti Gambar 1.13.</p>


```
BlogPost.jsx  Post.jsx  # BlogPost.css  JS index.js  index.html  # index.css
react-web > src > component > BlogPost > Post.jsx > ...
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">Judul Artikel</div>
11         <p className="isi-artikel">Isi Artikel</p>
12       </div>
13     </div>
14   )
15 }
16
17 export default Post;
```

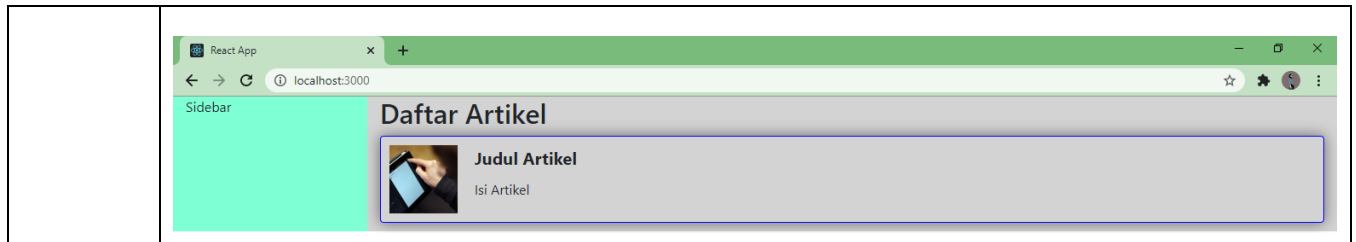
Gambar 1.13. Kode program Post.jsx

Untuk *statefull component* **BlogPost.jsx** pada baris 10, panggil *stateless component* **Post.jsx** seperti Gambar 1.14.

```
BlogPost.jsx  Post.jsx  # BlogPost.css  JS index.js
react-web > 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;
```

Gambar 1.14. Component BlogPost memanggil component Post

Perhatikan hasil tampilan browser, apa yang terjadi?



Muat Data Dinamis.

Bagaimana caranya untuk dapat membuat data dinamis (lebih dari 1 artikel) dimana data Judul dan Deskripsi pada artikel didapat dari API?

19

Pada *statefull component* **BlogPost.jsx**, tambahkan parameter yang ingin dilempar ke *stateless component* untuk ditampilkan. Kode program bisa dilihat pada Gambar 1.15.

```

BlogPost.jsx | Post.jsx | # BlogPost.css | JS index.js | index.html | # index.css
react-web > 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 judul="JTI Polinema" isi="Jurusan Teknologi Informasi - Politeknik Negeri Malang"/>
11             </div>
12         )
13     }
14 }
15
16 export default BlogPost;

```

Gambar 1.15. Penambahan parameter pada BlogPost

20

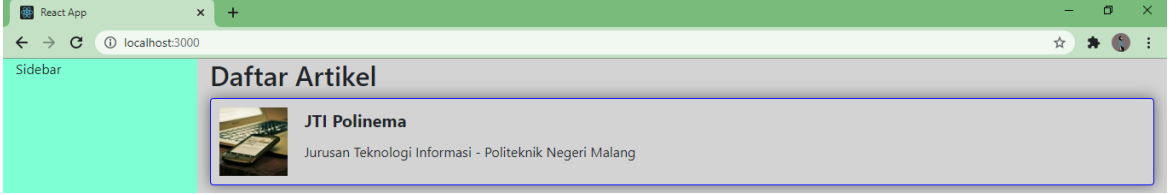
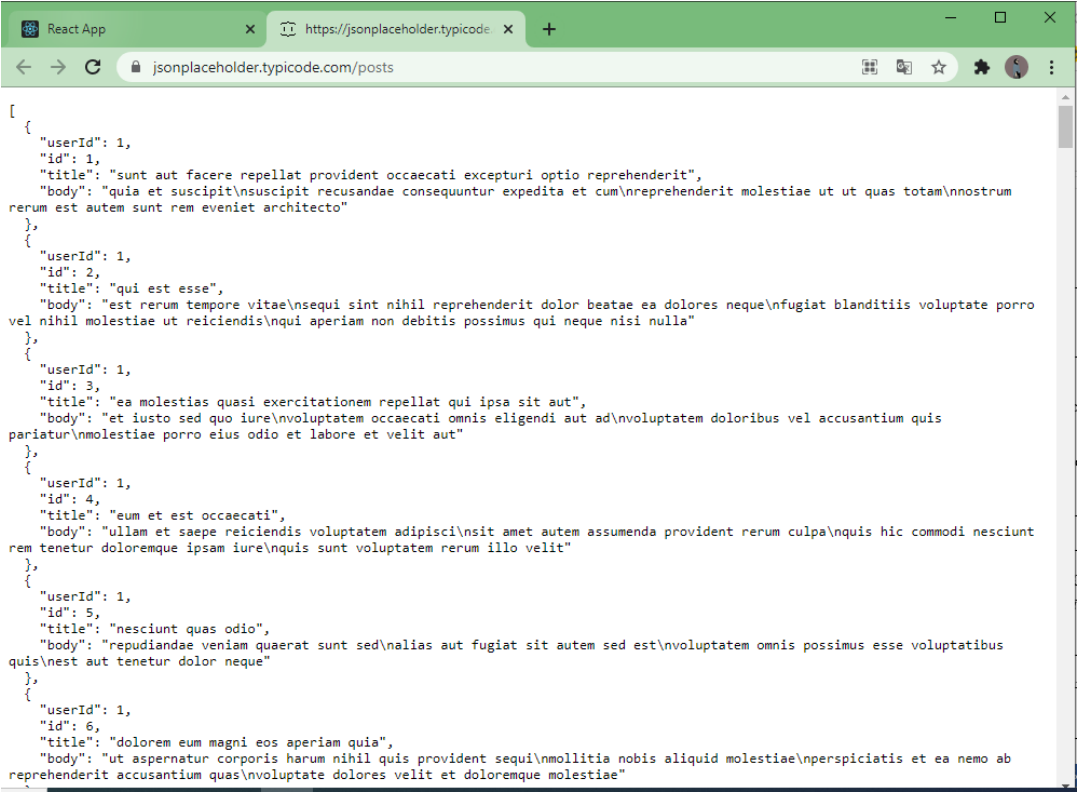
Setelah itu pada *stateless component* **Post.jsx** tangkap parameter yang dilempar oleh *statefull component* seperti pada Gambar 1.16 dan lihat pada browser apa yang terjadi!.

```

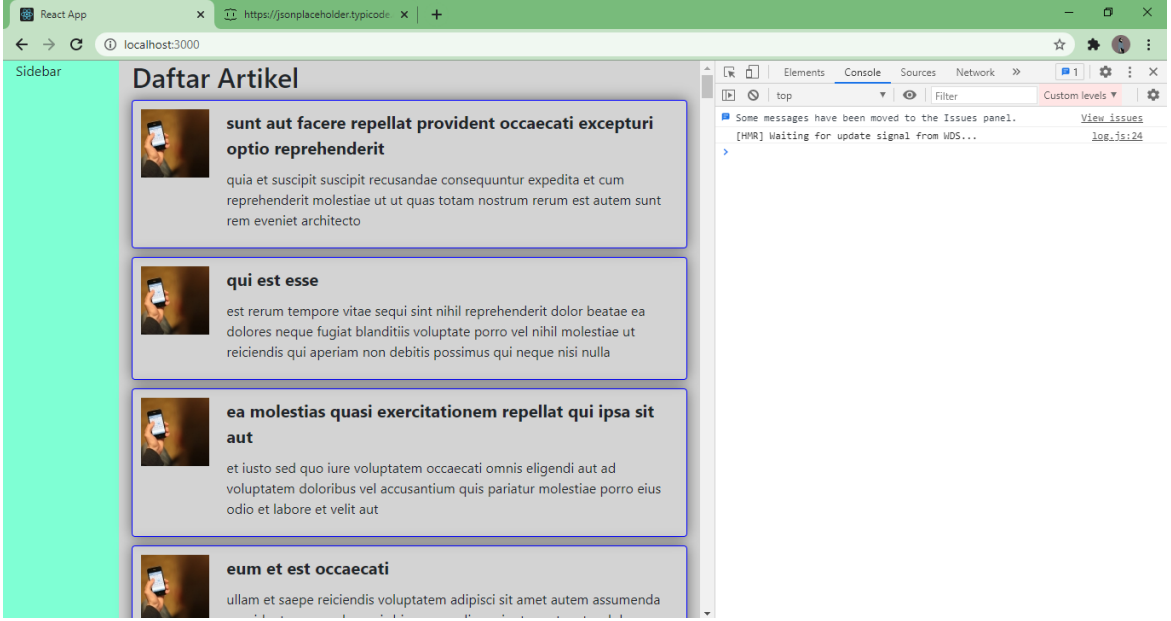
BlogPost.jsx | Post.jsx | # BlogPost.css | JS index.js | index.html | # index.css
react-web > src > component > BlogPost > Post.jsx > default
1  import React from "react";
2
3  const Post = (props) => {
4      return (
5          <div className="artikel">
6              <div className="gambar-artikel">
7                  
8              </div>
9              <div className="konten-artikel">
10                 <div 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;

```

Gambar 1.16. Kode program Post

21	<p>Simpan, dan amati apa yang terjadi pada browser kalian!.</p> 
	<p>Mengambil data Post/Artikel dari API.</p> <p>Bagaimana caranya untuk mendapatkan list artikel berdasarkan data json dari web API (contohnya: https://jsonplaceholder.typicode.com/posts) ?</p> <p>Kita gunakan <i>life cycle component</i> yaitu <code>componentDidMount()</code> dimana ketika komponen selesai di-<i>mount</i>-ing, program akan memanggil API.</p>
22	<p>Gunakan <code>state</code> untuk menyimpan data hasil <i>request</i> dari API</p>
23	<p>data API yang akan kita gunakan adalah data <i>dummy</i> dari https://jsonplaceholder.typicode.com/posts, dimana memiliki 4 element data yaitu <i>userid</i>, <i>id</i>, <i>title</i>, <i>body</i> (seperti pada Gambar 1.17)</p>  <pre>[{ "userId": 1, "id": 1, "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit", "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto" }, { "userId": 1, "id": 2, "title": "qui est esse", "body": "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\nqui aperiam non debitis possimus qui neque nisi nulla" }, { "userId": 1, "id": 3, "title": "ea molestias quasi exercitationem repellat qui ipsa sit aut", "body": "et iusto sed quo iure\nvoluptatem occaecati omnis eligendi aut ad\nvoluptatem doloribus vel accusantium quis pariatur\nmolestiae porro eius odio et labore et velit aut" }, { "userId": 1, "id": 4, "title": "eum et est occaecati", "body": "ullam et saepe reiciendis voluptatem adipisci\nsit amet autem assumenda provident rerum culpa\nquis hic commodi nesciunt rem tenetur doloremque ipsam iure\nquis sunt voluptatem rerum illo velit" }, { "userId": 1, "id": 5, "title": "nesciunt quas odio", "body": "repudiandae veniam quaerat sunt sed\nalias aut fugiat sit autem sed est\nvoluptatem omnis possimus esse voluptatibus quis\nnest aut tenetur dolor neque" }, { "userId": 1, "id": 6, "title": "dolorem eum magni eos aperiam quia", "body": "ut aspernatur corporis harum nihil quis provident sequi\nmollitia nobis aliquid molestiae\nperspiciatis et ea nemo ab reprehenderit accusantium quas\nvoluptate dolores velit et doloremque molestiae" }]</pre> <p>Gambar 1.17. Data response json dari web API</p>

24	<p>Edit pada <i>statefull component</i> BlogPost.jsx seperti pada Gambar 1.18 dan perhatikan dengan seksama akan penjelasan di beberapa baris kode program tersebut.</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 = { 7 listArtikel: [] // variable array yang digunakan untuk menyimpan data API 8 } 9 10 componentDidMount() { // komponen untuk mengecek ketika component telah di-mount-ing, maka panggil API 11 fetch('https://jsonplaceholder.typicode.com/posts') // alamat URL API yang ingin kita ambil datanya 12 .then(Response => Response.json()) // ubah response data dari URL API menjadi sebuah data json 13 .then(jsonHasilAmbilDariAPI => { // data json hasil ambil dari API kita masukkan ke dalam listArtikel pada state 14 this.setState({ 15 listArtikel: jsonHasilAmbilDariAPI 16 }) 17 }) 18 } 19 20 render(){ 21 return(22 <div class="post-artikel"> 23 <h2>Daftar Artikel</h2> 24 25 { 26 this.state.listArtikel.map(artikel => { // looping dan masukkan untuk setiap data yang ada di listArtikel ke variabel artikel 27 return <Post judul={artikel.title} isi={artikel.body}/> // mappingkan data json dari API sesuai dengan kategorinya 28 }) 29 } 30 </div> 31) 32 } 33 } 34 35 export default BlogPost; </pre> <p>Gambar 1.18. Penambahan componentDidMount pada <i>statefull component</i> BlogPost</p>
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>  <p>Gambar 1.19. Error pada browser</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. Lihat Gambar 1.20.</p> <pre> 20 render(){ 21 return(22 <div className="post-artikel"> 23 <h2>Daftar Artikel</h2> 24 { 25 this.state.listArtikel.map(artikel => { // looping dan masukkan untuk setiap data yang ada di litsArtikel ke variabel 26 return <Post key={artikel.id} judul={artikel.title} isi={artikel.body}/> // mappingkan data json dari API sesuai 27 }) 28 } 29 </div> 30) </pre> <p>Gambar 1.20. Penambahan key pada stateless component</p>
29	<p>Simpan dan lihat apa yang terjadi pada <i>console</i> browser (Gambar 1.21).</p>  <p>The screenshot shows a web browser with a React App running on localhost:3000. The page displays a list of articles under the heading "Daftar Artikel". The articles are listed in a sidebar and the main content area. The developer console is open, showing a message: "Some messages have been moved to the Issues panel. [HWB] Waiting for update signal from MDS... log_3:24".</p> <p>Gambar 1.21. Hasil akhir</p>

Pertanyaan Praktikum 1

- Pada langkah 8, sekarang coba kalian ganti class **container** dengan **container-fluid** atau sebaliknya pada file "**public/index.html**" dan lihat apa perbedaannya.
 - Tampilan seperti apa yang kalian temukan setelah mencoba mengganti nama class tersebut?
jawab : Saat class diganti menjadi container maka memiliki ukuran lebar container tidak memenuhi lebar layar (full width). Sedangkan class diganti menjadi container-fluid maka memiliki ukuran lebar container memenuhi lebar layar (full width).
 - Apa perbedaan dari **container** dan **container-fluid** ?

jawab: Container memiliki ukuran lebar tidak memenuhi lebar layer sedangkan container-fluid memiliki ukuran lebar memenuhi lebar layer.

- 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?

jawab: Install bootstrap dengan perintah “npm install bootstrap” dengan menginstall melalui terminal. Kemudian jika ingin mengimport suatu component menggunakan perintah syntax “import 'bootstrap/dist/css/bootstrap.min.css';”

Praktikum 2

Interaksi dengan API menggunakan *Fake API*

Saat kita mengakses API dengan method GET seperti Praktikum 1. Kita langsung menembak API dari server *jsonplaceholder* yaitu <https://jsonplaceholder.typicode.com/posts>. Data yang akan kita dapat sesuai dengan data yang disediakan oleh server tersebut.

Bagaimana jika kita ingin mendapatkan atau mengolah data API sendiri sehingga data yang akan kita pakai sesuai dengan yang kita inginkan? Solusinya bisa menggunakan *Fake API* yang kita install di local project ReactJS.

Install Fake API (JSON Server)

Fake API/JSON Server bisa kita dapatkan di halaman <https://github.com/typicode/json-server>. Tahapan install dan membuat data json sendiri

Langkah	Keterangan
1	Install pada direktori project reactjs kita dengan perintah npm <code>install -gjson-server</code>
2	Copy-kan file json <code>listArtikel.json</code> yang sudah ada pada direktori project reactjs kita.
3	Buka cmd baru pada direktori project, lalu ketik perintah <code>json-server --watch listArtikel.json --port 3001</code> .
4	Apabila pada cmd tampil seperti Gambar 2.1 , maka server <i>Fake API</i> local kita telah siap

```
PS C:\Users\ASUS\Documents\PemrogramanFramework2021\minggu4\react-web> npx json-server --watch listArtikel.json --port 3001
npx: installed 182 in 17.325s

\{^_^}/ 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...
```

Gambar 2.1 Tampilan response json-server

Kita cek *url resource* yang ada pada Fake API server ke browser apakah bisa diakses. Ketik url <http://localhost:3001/posts> pada browser.

```
localhost:3001/posts
https://jsonplaceholder.typicode.com/

[
  {
    "userId": 1,
    "id": 1,
    "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
    "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"
  },
  {
    "userId": 1,
    "id": 2,
    "title": "qui est esse",
    "body": "est rerum tempore vitae\nsequi sint nihil reprehenderit dolor beatae ea dolores neque\nfugiat blanditiis voluptate porro vel nihil molestiae ut reiciendis\nqui aperiam non debitis possimus qui neque nisi nulla"
  },
  {
    "userId": 1,
    "id": 3,
    "title": "ea molestias quasi exercitationem repellat qui ipsa sit aut",
    "body": "et iusto sed quo iure\nvoluptatem occaecati omnis eligendi aut ad\nvoluptatem doloribus vel accusantium quis pariatur\nmolestiae porro eius odio et labore et velit aut"
  },
  {
    "userId": 1,
    "id": 4,
    "title": "eum et est occaecati",
    "body": "ullam et saepe reiciendis voluptatem adipisci\nsit amet autem assumenda provident rerum culpa\nquis hic commodi nesciunt rem tenetur doloremque ipsam iure\nquis sunt voluptatem rerum illo velit"
  },
  {
    "userId": 1,
    "id": 5,
    "title": "nesciunt quas odio",
    "body": "repudiandae veniam quaerat sunt sed\nnalias aut fugiat sit autem sed est\nvoluptatem omnis possimus esse voluptatibus quis\nest aut tenetur dolor neque"
  },
  {
    "userId": 1,
    "id": 6,
    "title": "dolorem eum magni eos aperiam quia",
    "body": "ut aspernatur corporis harum nihil quis provident sequi\nmollitia nobis aliquid molestiae\nperspiciatis et ea nemo ab reprehenderit accusantium quas\nvoluptate dolores velit et doloremque molestiae"
  },
  {
    "userId": 1,
    "id": 7,
    "title": "consequuntur et aut officiis debitis aut accusantium amet neque quia",
    "body": "magnam voluptatem quam est\nvoluptas qui aut aut officiis debitis aut accusantium amet neque quia"
  }
]
```

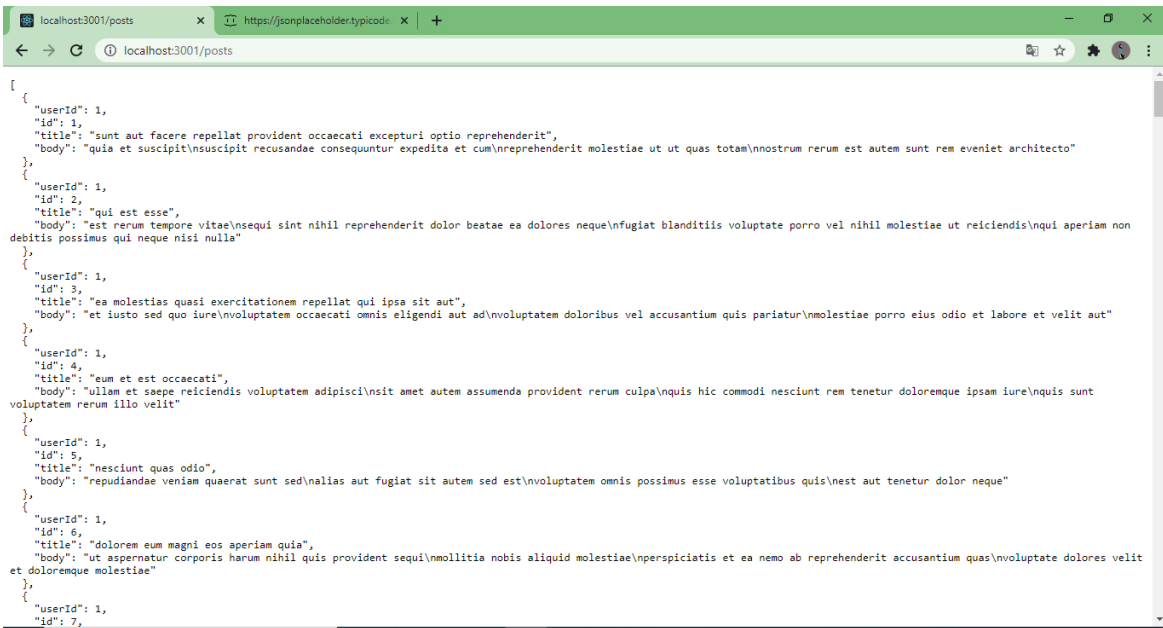
Gambar 2.2. Tampilan response json-server

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>

```
10   componentDidMount() { // komponen untuk mengecek keti
11     fetch('http://localhost:3001/posts') // alamat UP
12     .then(Response => Response.json()) // ubah res
13     .then(jsonHasilAmbiDariAPI => { // data jso
14       this.setState({
15         listArtikel: jsonHasilAmbiDariAPI
16       })
17     })
18   }
```

7

Simpan perubahan dan amati apa yang terjadi.



Pertanyaan Praktikum 2

- a. Kenapa *json-server* dijalankan pada port 3001? Kenapa tidak sama-sama dijalankan pada port 3000 seperti project react yang sudah kita buat?

jawab : Karena port 3000 sudah digunakan untuk pengoperasian method GET. Sedangkan untuk mengambil sebuah data pada local project react-web membutuhkan penggunaan port yang berbeda.

- b. Bagaimana jadinya kalau kita ganti *port json-server* menjadi 3000?

jawab : Kalau diganti ke port *json-server* menjadi 3000 bisa saja , hanya saja untuk mengambil data local yang menjadi server akan tetapi project sebelumnya harus dilakukan di terminate kemudian di jalankan pada port 3000.

Praktikum 3

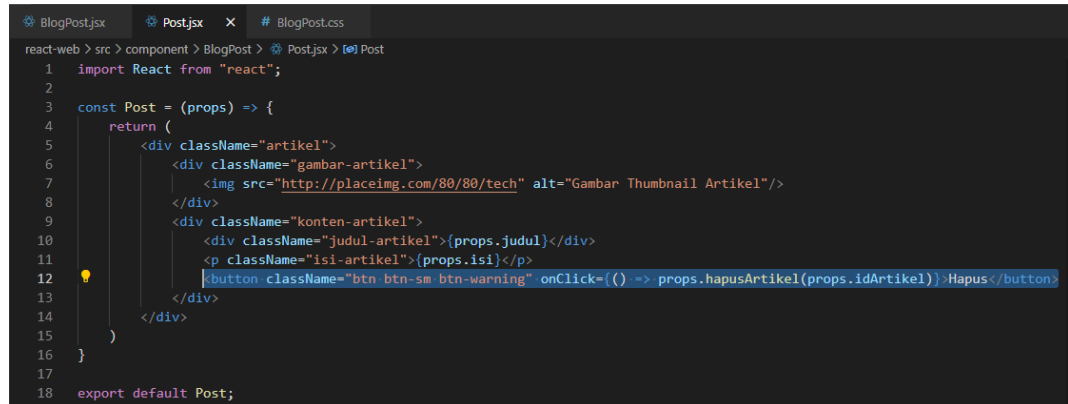
Interaksi dengan API menggunakan method DELETE

Method DELETE secara umum digunakan untuk melakukan proses hapus data. Saat kita ingin menghapus data, kita akan melakukan *request* ke server API dengan menggunakan *method* DELETE. Secara otomatis, server akan mengetahui bahwa *request* yang kita lakukan adalah untuk melakukan penghapusan data karena *request* kita menggunakan *method* DELETE.

Langkah Praktikum

Langkah	Keterangan
---------	------------

Buka *stateless component* **Post**. Tambahkan 1 baris kode program pada baris 10 seperti pada Gambar 3.1



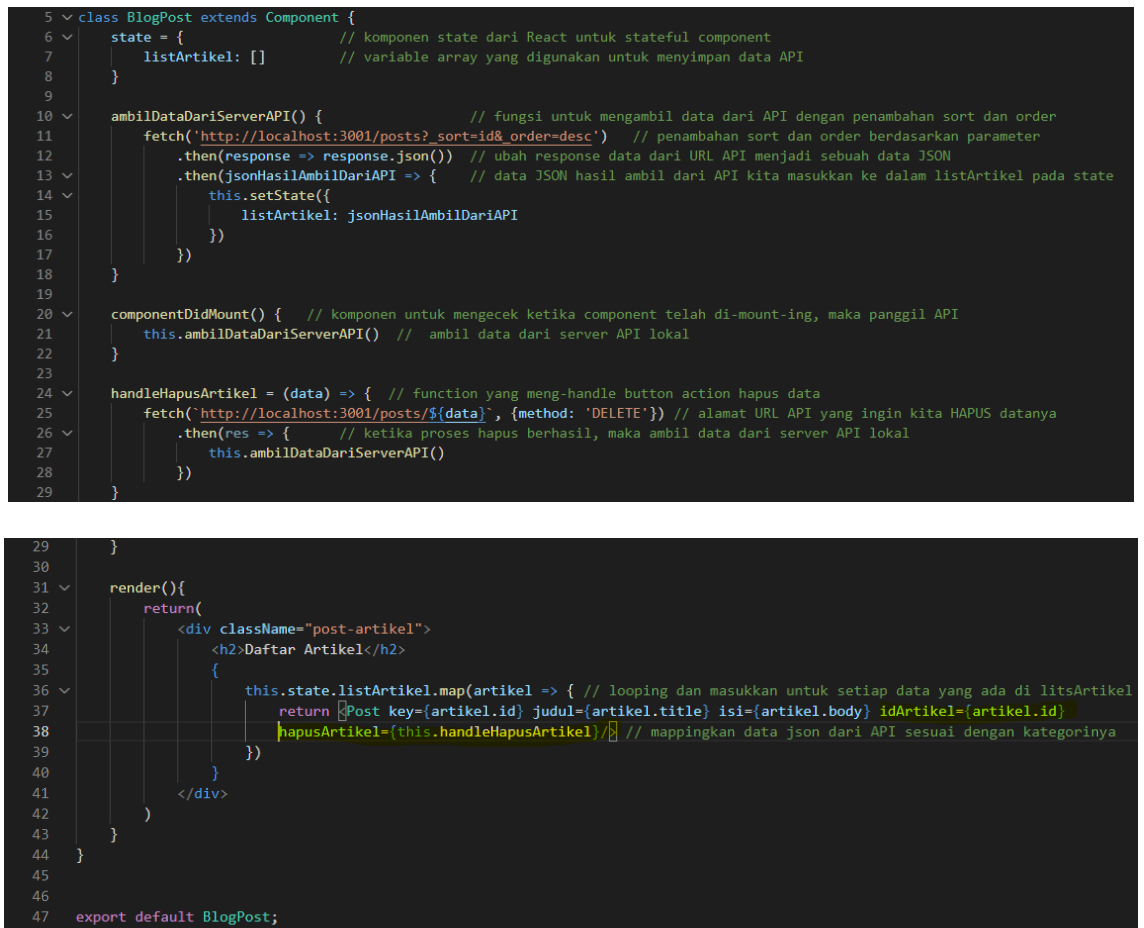
```

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         <button className="btn btn-sm btn-warning" onClick={() => props.hapusArtikel(props.idArtikel)}>Hapus</button>
13       </div>
14     </div>
15   )
16 }
17
18 export default Post;

```

Gambar 3.1 Tambah kode program Post.jsx

Kemudian pada *statefull component* **BlogPost**, modifikasi kode program sebelumnya sesuai dengan Gambar 3.2



```

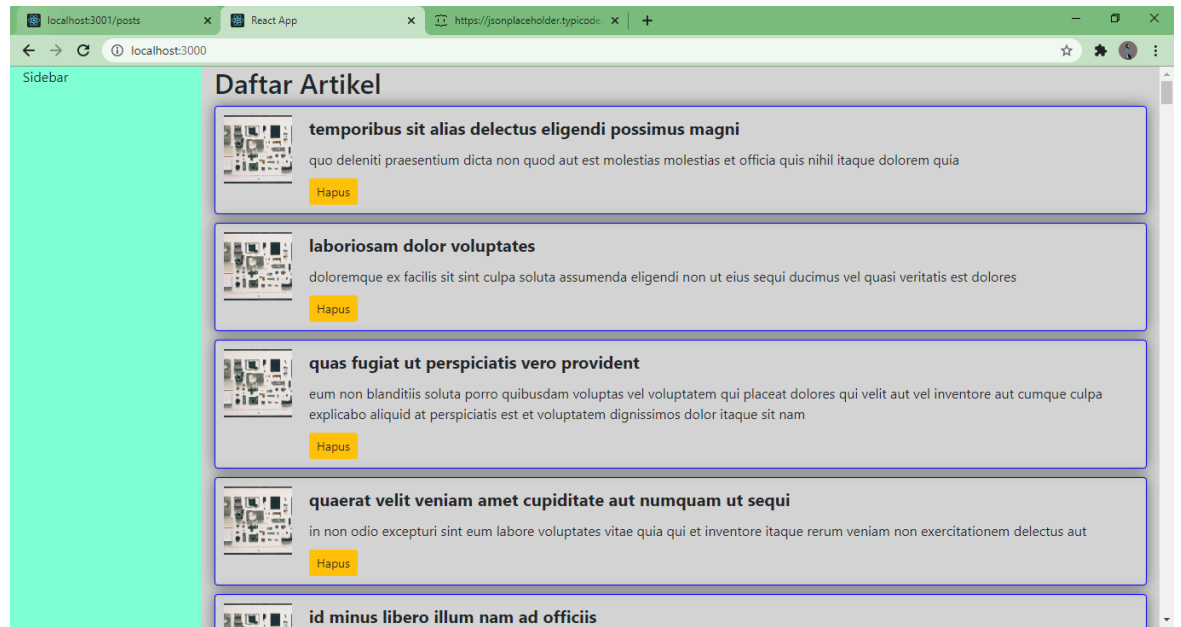
5  class BlogPost extends Component {
6    state = { // komponen state dari React untuk stateful component
7      listArtikel: [] // variable array yang digunakan untuk menyimpan data API
8    }
9
10   ambilDataDariServerAPI() { // fungsi untuk mengambil data dari API dengan penambahan sort dan order
11     fetch('http://localhost:3001/posts?sort=id&order=desc') // penambahan sort dan order berdasarkan parameter
12     .then(response => response.json()) // ubah response data dari URL API menjadi sebuah data JSON
13     .then(jsonHasilAmbilDariAPI => { // data JSON hasil ambil dari API kita masukkan ke dalam listArtikel pada state
14       this.setState({
15         listArtikel: jsonHasilAmbilDariAPI
16       })
17     })
18   }
19
20   componentDidMount() { // komponen untuk mengecek ketika component telah di-mount-ing, maka panggil API
21     this.ambilDataDariServerAPI() // ambil data dari server API lokal
22   }
23
24   handleHapusArtikel = (data) => { // function yang meng-handle button action hapus data
25     fetch('http://localhost:3001/posts/${data}', {method: 'DELETE'}) // alamat URL API yang ingin kita HAPUS datanya
26     .then(res => { // ketika proses hapus berhasil, maka ambil data dari server API lokal
27       this.ambilDataDariServerAPI()
28     })
29   }
30
31   render(){
32     return(
33       <div className="post-artikel">
34         <h2>Daftar Artikel</h2>
35         {
36           this.state.listArtikel.map(artikel => { // looping dan masukkan untuk setiap data yang ada di listArtikel
37             return <Post key={artikel.id} judul={artikel.title} isi={artikel.body} idArtikel={artikel.id}
38               hapusArtikel={this.handleHapusArtikel}/> // mappingkan data json dari API sesuai dengan kategorinya
39           })
40         }
41       </div>
42     )
43   }
44 }
45
46 export default BlogPost;

```

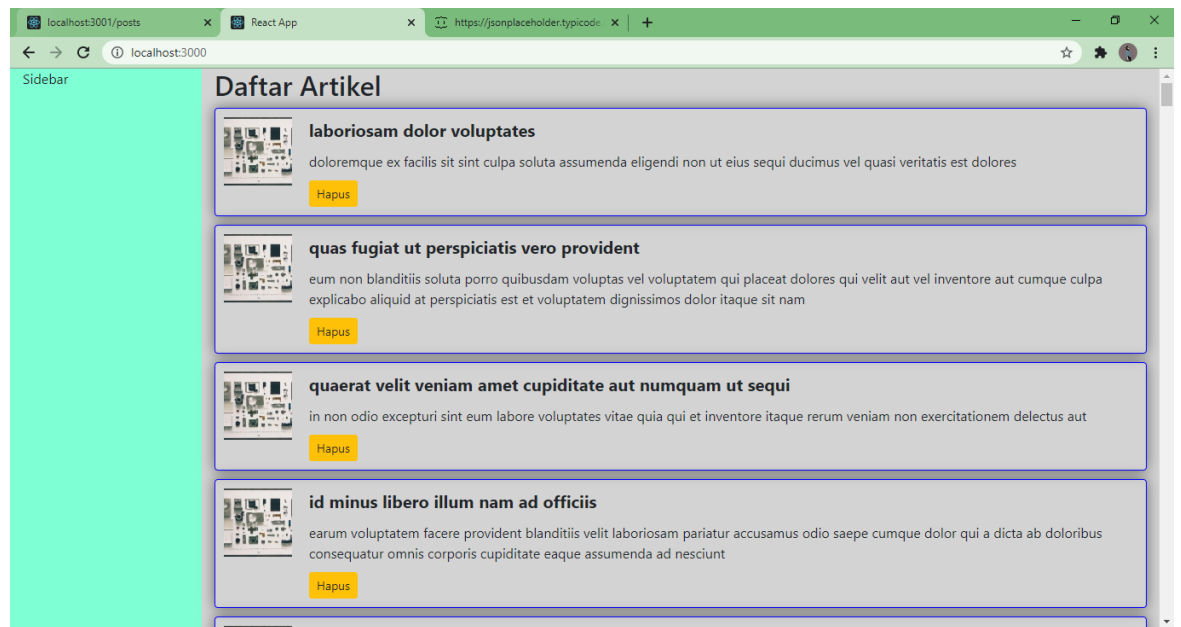
Gambar 3.2 Modifikasi kode program BlogPost

3

Klik tombol hapus pada list artikel di browser. Amati apa yang terjadi.



Setelah klik tombol Hapus pada list maka hasilnya akan menampilkan index yang tersisa



Pertanyaan Praktikum 3

a. Apa yang terjadi setelah kalian klik tombol hapus?

jawab : Maka data yang di klik akan hilang, sehingga halaman hanya menampilkan data yang tersisa.

b. Perhatikan file **listArtikel.json**, apa yang terjadi pada file tersebut? Kenapa demikian?

jawab : Data yang ada di file listArtikel.json ikut terhapus sesuai dengan yang di klik tombol pada id

yang bersangkutan

- c. Fungsi `handleHapusArtikel` itu untuk apa?

jawab: function yang meng-handle button action hapus data

- d. Jelaskan perbedaan fungsi `componentDidMount()` pada Gambar 1.18 dengan fungsi `componentDidMount()` pada Gambar 3.2 ?

jawab : fungsi `componentDidMount()` merupakan sebuah komponen untuk mengecek ketika component telah di-mount-ing. fungsi `componentDidMount()` pada Gambar 1.18 digunakan untuk mengambil data pada <https://jsonplaceholder.typicode.com/posts> sedangkan fungsi `componentDidMount()` pada Gambar 3.2 digunakan untuk mengambil data dari server API lokal.

Praktikum 4

Interaksi dengan API menggunakan method POST

Method POST sering digunakan dalam mengirimkan form *request* ke server. Dalam API method POST biasa digunakan untuk melakukan insert/tambah data pada server.

Langkah Praktikum

Langkah	Keterangan
1	<p>Buka <i>statefull component</i> <code>BlogPost</code>, dan modifikas pada fungsi <code>render()</code> untuk menampilkan <i>form input</i> artikel yang berisi judul dan isi berita. seperti pada Gambar 4.1</p> <pre>render() { return (<div className="post-artikel"> <div className="form pb-2 border-bottom"> <div className="form-group row"> <label htmlFor="title" className="col-sm-2 col-form-label">Judul</label> <div className="col-sm-10"> <input type="text" className="form-control" id="title" name="title" onChange={this.handleTambahArtikel}/> </div> </div> <div className="form-group row"> <label htmlFor="body" className="col-sm-2 col-form-label">Isi</label> <div className="col-sm-10"> <textarea className="form-control" id="body" name="body" rows="3" onChange={this.handleTambahArtikel}> </textarea> </div> </div> <button type="submit" className="btn btn-primary" onClick={this.handleTombolSimpan}>Simpan</button> </div> <h2>Daftar Artikel</h2> { this.state.listArtikel.map(artikel => { // looping dan masukkan untuk setiap data yang ada di listArtikel ke var return <Post key={artikel.id} judul={artikel.title} isi={artikel.body} idArtikel={artikel.id} hapusArtikel={this.handleHapusArtikel}/> // mappingkan data JSON dari API sesuai dengan kategorinya }) } </div>); }</pre>

	 <p>Gambar 4.1 modifikasi component BlogPost</p>
2	<p>Kemudian modifikasi BlogPost untuk bagian state dan request API dari server, seperti Gambar 4.2</p>  <p>Gambar 4.2 penambahan state pada BlogPost</p>
3	<p>Tambahkan untuk <i>handle</i> form tambah data artikel seperti Gambar 4.3</p>

```

react-web > src > container > BlogPost > BlogPost.jsx > BlogPost > handleTombolSimpan
21 |         listArtikel: jsonHasilAmbilDariAPI
22 |     })
23 | })
24 | }
25 |
26 | componentDidMount() { // komponen untuk mengecek ketika component telah di
27 |     this.ambilDataDariServerAPI() // ambil data dari server API lokal
28 | }
29 |
30 | handleHapusArtikel = (data) => { // function yang meng-handle button action
31 |     fetch(`http://localhost:3001/posts/${data}`, {method: 'DELETE'}) // alam
32 |     .then(res => { // ketika proses hapus berhasil, maka ambil data
33 |         this.ambilDataDariServerAPI()
34 |     })
35 | }
36 |
37 | handleTambahArtikel = (event) => { // fungsi untuk meng-handle form tam
38 |     let formInsertArtikel = {...this.state.insertArtikel}; // cloning data
39 |     let timestamp = new Date().getTime(); // digunakan untuk menyimpan
40 |     formInsertArtikel['id'] = timestamp;
41 |     formInsertArtikel[event.target.name] = event.target.value; // menyimpan
42 |     this.setState({
43 |         insertArtikel: formInsertArtikel
44 |     });
45 | }

```

Gambar 4.3 Handle tambah artikel

4

Langkah terakhir tambahkan fungsi untuk handle tombol simpan artikel, seperti pada Gambar 4.4

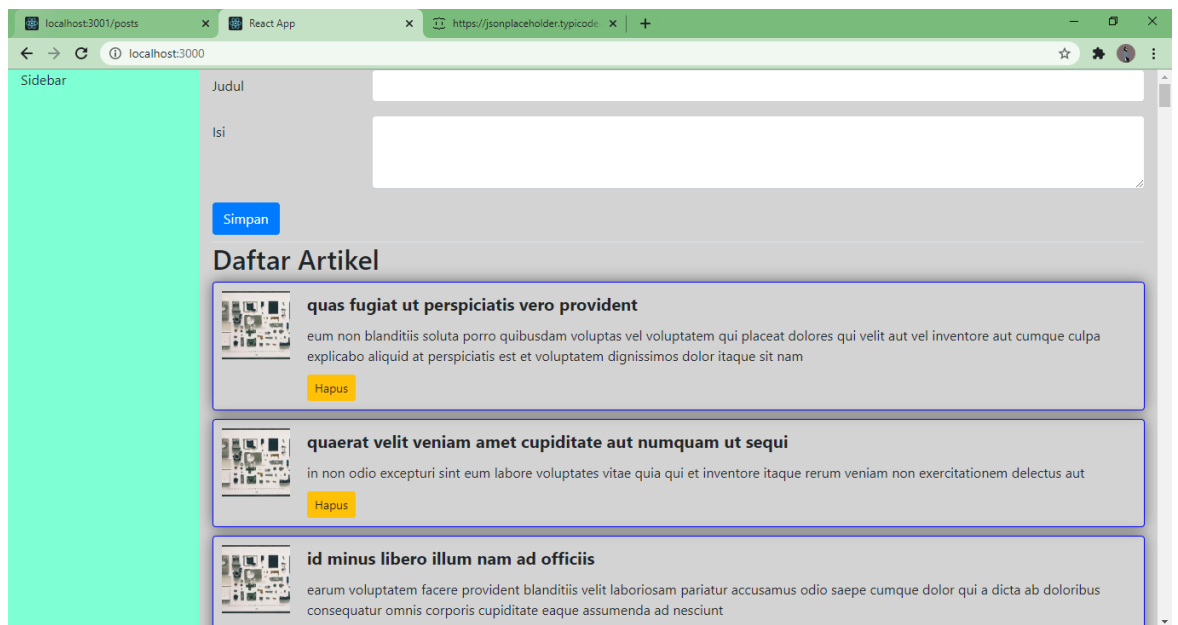
```

36
37   handleTambahArtikel = (event) => {      // fungsi untuk meng-handle form tambag data
38       let formInsertArtikel = {...this.state.insertArtikel}; // cloning data state ins
39       let timestamp = new Date().getTime();      // digunakan untuk menyimpan watu (se
40       formInsertArtikel['id'] = timestamp;
41       formInsertArtikel[event.target.name] = event.target.value; // menyimpan data onC
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 u
55       })
56       .then( (Response) => {
57         this.ambilDataDariServerAPI();      // reload/refresh data
58       });
59     }
60
61     render() {
62       return (

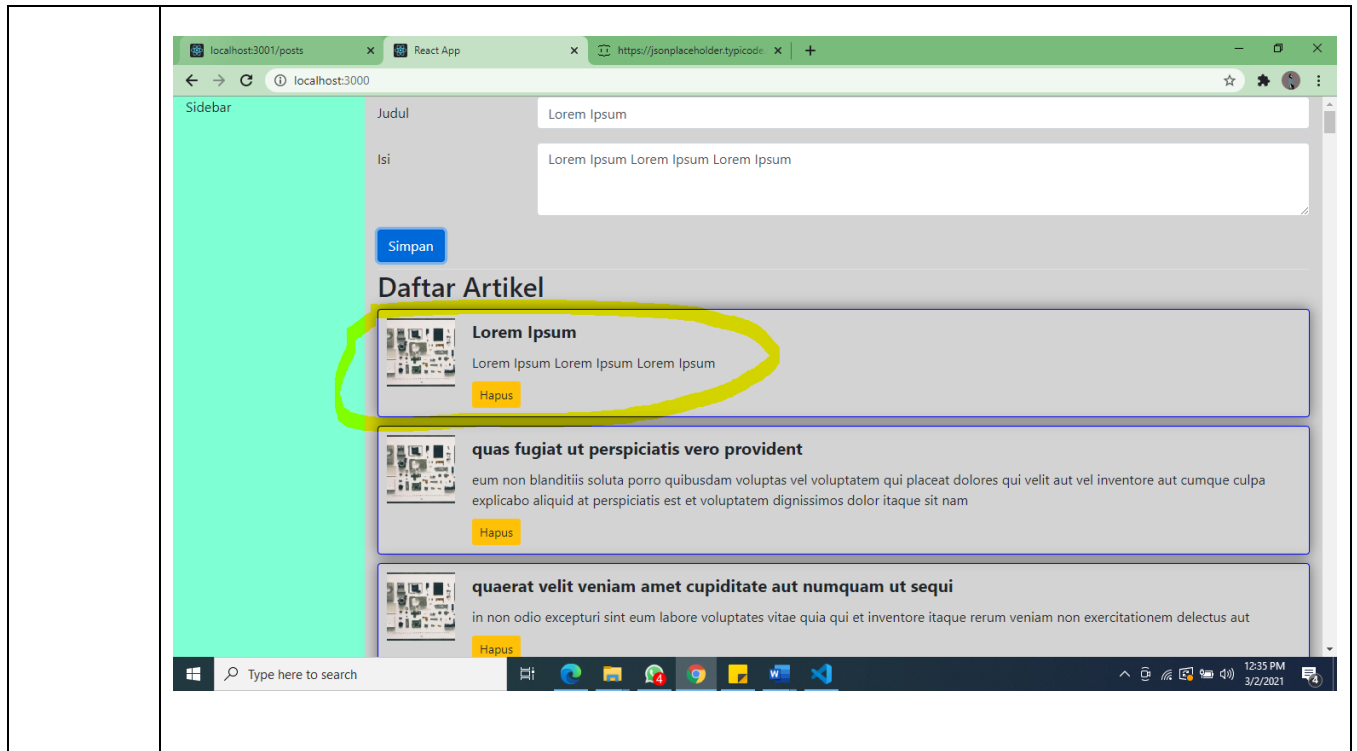
```

Gambar 4.4 Handle tombol simpan

Simpan, lakukan percobaan penambahan data, dan amati perubahannya.



Ketika melakukan input pada judul dan isi dan di klik tombol simpan maka data akan masuk pada Daftar Artikel seperti gambar berikut



Pertanyaan Praktikum

- a. Jelaskan apa yang terjadi pada file **listArtikel.json** sebelum dan setelah melakukan penambahan data?

jawab : Yang terjadi pada file **listArtikel.json** sebelum dilakukan penambahan, file tersebut dilakukan penambahan data terlebih dahulu kemudian dilakukan sorting dan order secara desc dan data diubah menjadi json lalu data json di insert ke file **listArtikel.json** pada suatu state. Dan setelah penambahan data yang terjadi tambahan isi dari file **listArtikel** bertambah itu bisa terjadi karena data tersebut ditampung sementara pada **insertSrtikel** lalu dilakukannya eksekusi fungsi **handleTambahArtikel** dan kemudian fungsi tersenut disimpan pada eksekusi **handleRombolSimpan**.

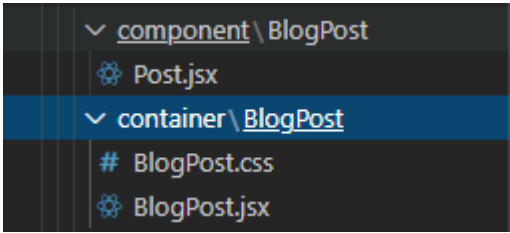
- b. Data yang ditampilkan di browser adalah data terbaru berada di posisi atas dan data lama berada di bawah, sedangkan pada file **listArtikel.json** data terbaru malah berada di bawah. Jelaskan mengapa demikian?

jawab : data yang ditampilkan di browser dengan data yang ditampilkan pada file **listArtikel.json** berbeda karena pada saat function **ambilDataDariServer()** itu dieksekusi dengan dilakukan order berdasarkan id nya secara desc sehingga data dengan id nya yang lebih besar atau terakhir dilakukannya input akan berbeda pada saat posisi paling atas diikuti dengan idnya yang lebih kecil berada di bawahnya.

Tugas Praktikum

Buatlah 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. Buatlah aplikasi yang menggunakan API dengan method GET, DELETE, dan POST.

Langkah Tugas

Langkah	Keterangan
1	Jika membuat project baru install bootstrap
2	Kemudian install npm <code>install -gjson-server</code>
3	<i>Copy</i> -kan file json listMahasiswa.json yang sudah ada pada direktori project reactjs kita.
4	Buka cmd baru pada direktori project, lalu ketik perintah <code>json-server --watch listArtikel.json --port 3001</code> .
5	<p>Buat folder dan isi nama file seperti pada gambar berikut</p> 
6	BlogPost.jsx
	<pre>minggu4 > react-web-tugas > src > container > BlogPost > BlogPost.jsx > ... 1 import { 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 stateful component 8 listMahasiswa: [], // variable array yang digunakan untuk menyimpan data API 9 insertMahasiswa: { // variable yang digunakan untuk menampung sementara data yang akan di insert 10 id: "", // kolom userId, id, title, dan body sama, mengikuti kolom yang ada pada listArtikel.json 11 nim: "", 12 nama: "", 13 alamat: "", 14 hp: "", 15 angkatan: "", 16 status: "" 17 } 18 } 19 20 ambilDataDariServerAPI() { // fungsi untuk mengambil data dari API dengan penambahan sort dan order 21 fetch('http://localhost:3001/mahasiswa? sort=id& order=desc') // penambahan sort dan order berdasarkan parameter 22 .then(response => response.json()) // ubah response data dari URL API menjadi sebuah data JSON 23 .then(jsonHasilAmbilDariAPI => { // data JSON hasil ambil dari API kita masukkan ke dalam listArtikel pada state 24 this.setState({ 25 listMahasiswa: jsonHasilAmbilDariAPI 26 }) 27 }) 28 } 29 }</pre>


```

28
29 ✓ componentDidMount() { // komponen untuk mengecek ketika component telah di-mount-ing, maka panggil API
30   this.ambilDataDariServerAPI() // ambil data dari server API lokal
31 }
32
33 ✓ handleHapusMahasiswa = (data) => { // function yang meng-handle button action hapus data
34   fetch('http://localhost:3001/mahasiswa/${data}', {method: 'DELETE'}) // alamat URL API yang ingin kita HAPUS datanya
35   .then(res => { // ketika proses hapus berhasil, maka ambil data dari server API lokal
36     this.ambilDataDariServerAPI()
37   })
38 }
39
40 ✓ handleTambahMahasiswa = (event) => { // fungsi untuk meng-handle form tambag data artikel
41   let formInsertMahasiswa = {...this.state.insertMahasiswa}; // cloning data state insertArtikel ke dalam variable formInsertArt
42   let timestamp = new Date().getTime(); // digunakan untuk menyimpan waktu (sebagai ID artikel)
43   formInsertMahasiswa['id'] = timestamp;
44   formInsertMahasiswa[event.target.name] = event.target.value; // menyimpan data onChange ke formInsertArtikel sesuai dengan tar
45   this.setState({
46     insertMahasiswa: formInsertMahasiswa
47   });
48 }
49
50 handleTombolSimpan = () => { // fungsi untuk meng-handle tombol simpan
51   fetch('http://localhost:3001/mahasiswa', {

```

```

52     method: 'post', // method POST untuk input/insert data
53     headers: {
54       'Accept': 'application/json',
55       'Content-Type': 'application/json'
56     },
57     body: JSON.stringify(this.state.insertMahasiswa) // kirimkan ke body request untuk data yang akan ditambahkan (insert)
58   })
59   .then((Response) => {
60     this.ambilDataDariServerAPI(); // reload/refresh data
61   });
62 }
63
64 ✓ render() {
65   return (
66     <div className="post-mahasiswa">
67       <br></br>
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.handleTambahMahasiswa}/>
73           </div>
74         </div>
75         <div className="form-group row">

```

```

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.handleTambahMahasiswa}/>
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.handleTambahMahasiswa}>
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.handleTambahMahasiswa}/>
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.handleTambahMahasi
97           </div>
98         </div>
99         <div className="form-group row">
100           <label htmlFor="status" className="col-sm-2 col-form-label">Status</label>

```

	<pre> 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.handleTambahMahasiswa} 103 </div> 104 </div> 105 <button type="submit" className="btn btn-primary" onClick={this.handleTombolSimpan}>Simpan</button> 106 </div> 107 <h2>Daftar Mahasiswa</h2> 108 { 109 this.state.listMahasiswa.map(mahasiswa => { // looping dan masukkan untuk setiap data yang ada di listArtikel ke \ 110 return <Post key={mahasiswa.id} nama={mahasiswa.nama} alamat={mahasiswa.alamat} 111 hp={mahasiswa.hp} angkatan={mahasiswa.angkatan} status={mahasiswa.status} 112 nimMahasiswa={mahasiswa.nim} idMahasiswa={mahasiswa.id} hapusMahasiswa={this.handleHapusMahasiswa}/> // mappir 113 }) 114 } 115 </div> 116) 117 } 118 } 119 export default BlogPost; </pre>
7	BlogPost.css
	<pre> minggu4 > react-web-tugas > src > container > BlogPost > # BlogPost.css > .mahasiswa 1 .mahasiswa { 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%; 20 height: 100%; 21 object-fit: cover; 22 } 23 24 .kategori { 25 flex: 0.15; 26 } 27 </pre>

	<pre> 28 √ .kategori div.kategori-nim { 29 font-size: 16px; 30 margin-bottom: 10px; 31 } 32 33 √ .kategori p.kategori-nama { 34 font-size: 16px; 35 margin-bottom: 10px; 36 } 37 38 √ .konten-mahasiswa p.nama-mahasiswa { 39 font-size: 16px; 40 margin-bottom: 10px; 41 padding-left: 0%; 42 padding-top: 10px; 43 } 44 </pre>
8	Post.jsx
	<pre> minggu4 > react-web-tugas > src > component > BlogPost > Post.jsx > [e] Post • 1 import React from "react"; 2 3 √ const Post = (props) => { 4 return (5 √ <div className="mahasiswa"> 6 √ <div className="gambar-artikel"> 7 8 </div> 9 √ <div className="kategori"> 10 <div className="kategori-nim">NIM</div> 11 <p className="kategori-nama">Nama</p> 12 <p className="kategori-alamat">Alamat</p> 13 <p className="kategori-hp">HP</p> 14 <p className="kategori-angkatan">Angkatan</p> 15 <p className="kategori-status">Status</p> 16 </div> 17 √ <div className="konten-mahasiswa"> 18 <div className="nim-mahasiswa">: {props.nimMahasiswa}</div> 19 <p className="nama-mahasiswa">: {props.nama}</p> 20 <p className="alamat-mahasiswa">: {props.alamat}</p> 21 <p className="hp-mahasiswa">: {props.hp}</p> 22 <p className="angkatan-mahasiswa">: {props.angkatan}</p> 23 <p className="status-mahasiswa">: {props.status}</p> 24 <button className="btn btn-sm btn-warning" onClick={props.hapusMahasiswa(props.idMahasiswa)}>Hapus</button> 25 </div> 26 </div> 27) 28 } 29 30 export default Post; 31 32 </pre>
9	Index.html

minggu4 > react-web-tugas > 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
9      name="description"
10     content="Web site created using create-react-app"
11   />
12   <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
13   <!--
14     manifest.json provides metadata used when your web app is installed on a
15     user's mobile device or desktop. See https://developers.google.com/web/fundamentals/web-app-manife
16   -->
17   <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
18   <!--
19     Notice the use of %PUBLIC_URL% in the tags above.
20     It will be replaced with the URL of the `public` folder during the build.
21     Only files inside the `public` folder can be referenced from the HTML.
22
23     Unlike "/favicon.ico" or "favicon.ico", "%PUBLIC_URL%/favicon.ico" will
24     work correctly both with client-side routing and a non-root public URL.
25     Learn how to configure a non-root public URL by running `npm run build`.
26   -->
```

```
27   <title>React App</title>
28 </head>
29 <body>
30   <noscript>You need to enable JavaScript to run this app.</noscript>
31   <!-- <div id="root"></div> -->
32   <!--
33     This HTML file is a template.
34     If you open it directly in the browser, you will see an empty page.
35
36     You can add webfonts, meta tags, or analytics to this file.
37     The build step will place the bundled scripts into the <body> tag.
38
39     To begin the development, run `npm start` or `yarn start`.
40     To create a production bundle, use `npm run build` or `yarn build`.
41   -->
42   <div class="container-fluid">
43     <div class="row">
44       <div class="col-2" id="sidebar">Sidebar</div>
45       <div class="col-10" id="content"></div>
46     </div>
47   </div>
48 </body>
49 </html>
50
```

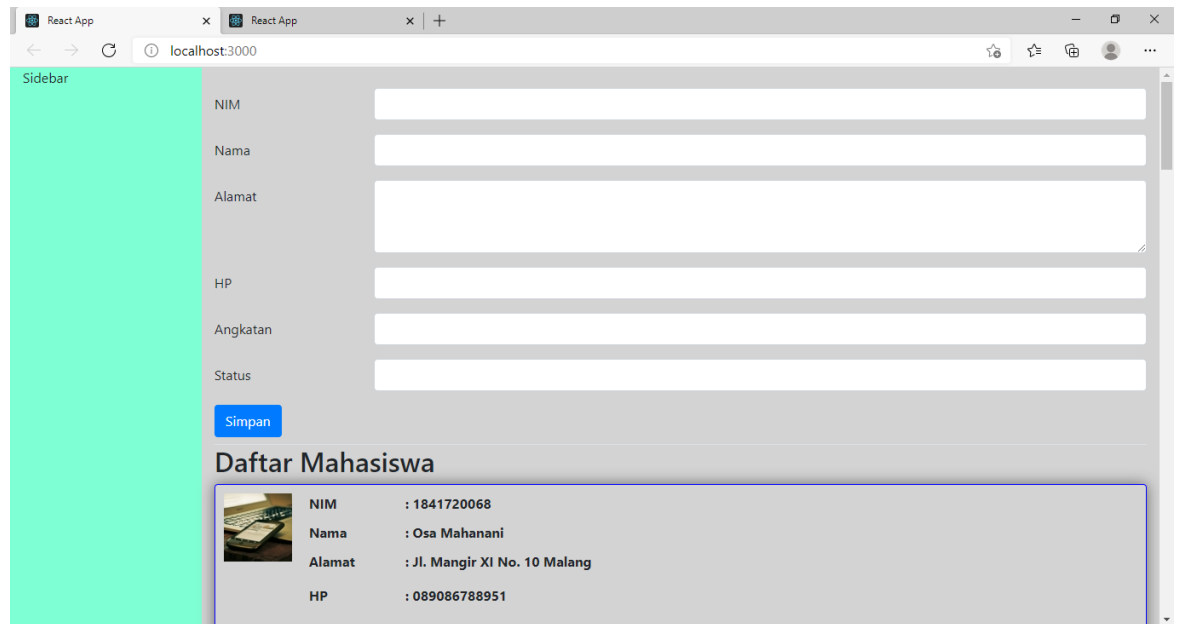
10

listMahasiswa.json

```
minggu4 > react-web-tugas > {} listMahasiswa.json > ...
1  {
2    "mahasiswa": [
3      {
4        "id": 1,
5        "nim": 1841720016,
6        "nama": "Dina Risky Alin Saputri",
7        "alamat": "Jl. S Supriadi III NO. 48 Malang",
8        "hp": "085649722109",
9        "angkatan": "2018",
10       "status": "aktif"
11     },
12     {
13       "id": 2,
14       "nim": 1841720018,
15       "nama": "Indriani Angelista",
16       "alamat": "Jl. Menur No. 9 Surabaya",
17       "hp": "082655387204",
18       "angkatan": "2019",
19       "status": "cuti"
20     },
21     {
22       "id": 3,
23       "nim": 1841720190,
24       "nama": "Agit Ari Irawan",
25       "alamat": "Jl. Masjid Jowar IX No. 10 Malang",
26       "hp": "081297840634",
27       "angkatan": "2018",
```

11

Hasil



React App x React App x +

localhost:3000

Sidebar

NIM

Nama

Alamat


HP



Angkatan

Status

Simpan

Daftar Mahasiswa

	NIM	: 1841720068
	Nama	: Osa Mahanani
	Alamat	: Jl. Mangir XI No. 10 Malang
	HP	: 089086788951

		<div> <div></div> <div> Angkatan : 2015 Status : lulus <div>Hapus</div> </div> </div>
		<div> <div>  <div> NIM : 1841720054 Nama : Burhanuddin Alamat : Jl. Adi Mulya X No. 3 Malang HP : 081523492211 Angkatan : 2019 Status : lulus <div>Hapus</div> </div> </div> </div>
		<div> <div>  <div> NIM : 1841720154 Nama : Kirana Putri </div> </div> </div>

Link Github : <https://github.com/dinariskyas/PemrogramanFramework2021/tree/master/minggu4>

Link Youtube :

- Pertemuan 1 : <https://youtu.be/1uWVJNRjvO8>
- Pertemuan 2 : <https://youtu.be/cjtAwWyaPJg>
- Pertemuan 3 : <https://youtu.be/7gwCfR19AVU>
- Pertemuan 4 : <https://youtu.be/Lbt9gPHqeYk>