

PEMROGRAMAN BERBASIS FRAMEWORK

TUGAS IV



Dibuat oleh:

Nama: Lili Nur Indah Sari

NIM : 1841720037

**PROGRAM STUDI TEKNIK INFORMATIKA
JURUSAN TEKNOLOGI INFORMASI
POLITEKNIK NEGERI MALANG**

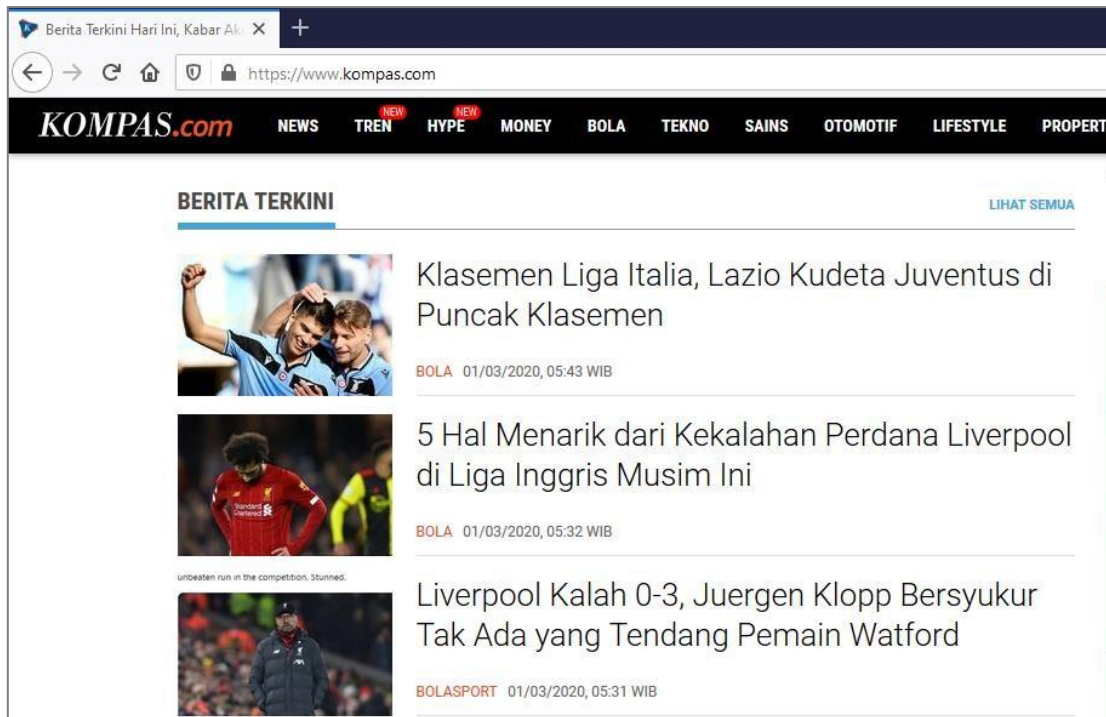
MARET 2021

Praktikum 1

Interaksi dengan API menggunakan method GET

1.1 Contoh Program

Contoh program yang akan kita buat adalah list artikel (*blog post*) pada suatu halaman website, seperti pada contoh Gambar 1.1.



Gambar 1.1. List berita

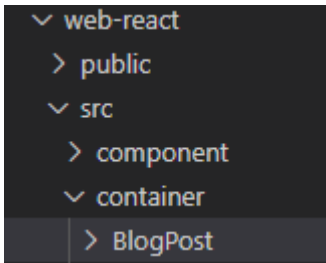
1.2 Data yang dipakai

Data yang akan kita pakai adalah data *dummy* atau *Fake Online REST API for Testing and Prototyping* dari halaman API <https://jsonplaceholder.typicode.com>.

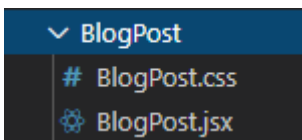
1.3 Langkah Praktikum

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.

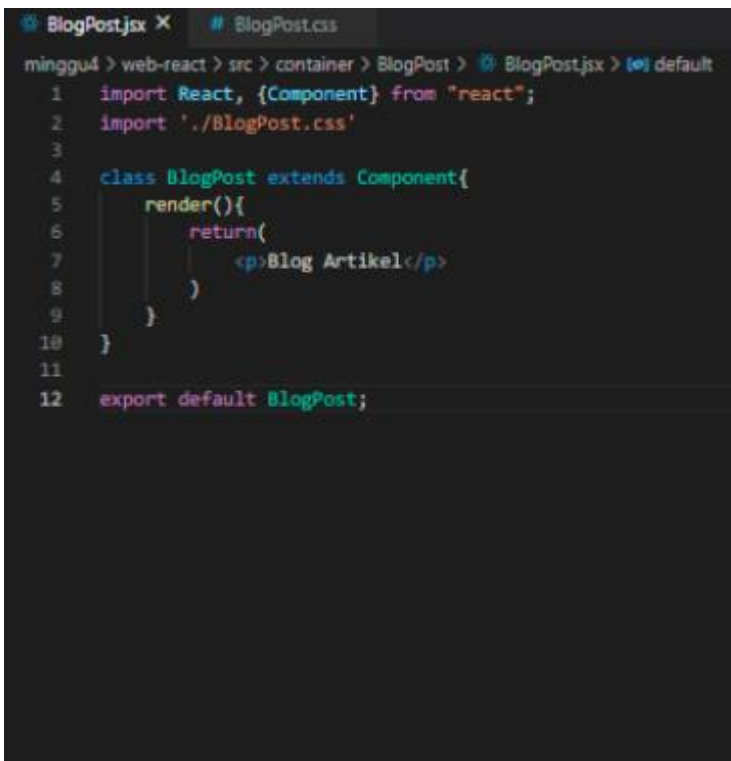
1. Buka Project React pada pertemuan sebelumnya dan jalankan “**npm start**” menggunakan *cmd* dalam direktori tersebut.
2. Buat folder baru bernama “**BlogPost**” pada folder **container** (*statefull component*).



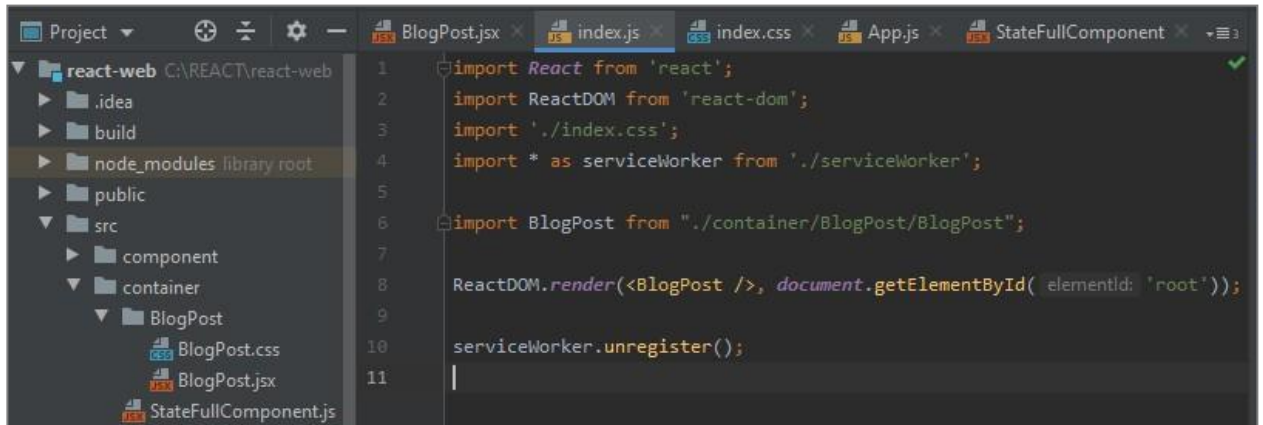
3. Buat file **BlogPost.jsx** dan **BlogPost.css** di dalam folder “**BlogPost**”, seperti pada Gambar 1.2.



4. Buka file **BlogPost.jsx** dan ketikkan kode seperti Gambar 1.3.

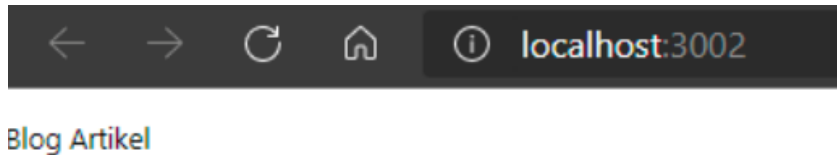


5. Pada file **index.js**, lakukan import component **BlogPost** seperti Gambar 1.4.



Gambar 1.4. Import component BlogPost

6. Pada web browser akan tampil seperti pada Gambar 1.5.



Gambar 1.5. Tampilan Browser

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"**

```
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 'bootstrap/dist/css/bootstrap.min.css'
7  //import BlogPost from './container/BlogPost/BlogPost'
8
9  import MahasiswaPost from './container/MahasiswaPost/MahasiswaPost'
10
11  ReactDOM.render(<MahasiswaPost/>, document.getElementById('root'));
12
13  // If you want to start measuring performance in your app, pass a function
14  // to log results (for example: reportWebVitals(console.log))
15  // or send to an analytics endpoint. Learn more: https://bit.ly/CRA-vitals
16  reportWebVitals();
17
```

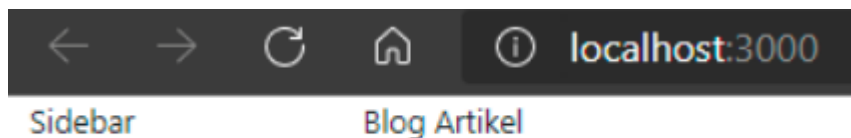
Gambar 1.6. Import bootstrap css

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
9       name="description"
10      content="Web site created using create-react-app"/>
11     <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
12     <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
13     <title>React App</title>
14   </head>
15   <body>
16     <noscript>You need to enable JavaScript to run this app.</noscript>
17     <!--penerapan layout bisa dilihat di https://getbootstrap.com/docs/4.0/layout/c
18     <!--Container adalah basic layout element di bootstrap dan DIPERLUKAN jika meng
19     default grid system bootstrap-->
20     <!--Ada 2 pilihan containera:
21       1. class container : untuk layout yang model bo (tidak full-width layar bro
22       2. class container-fluid : untuk layout model yang full-width layar browser
23       untuk lebih jelasnya, silahkan dicoba satu persatu.
24       default kita saat ini, menggunakan yang full-width (container-fluid)-->
25     <div class="container-fluid">
26       <div class="row">
27         <div class="col-2" id="sidebar">Sidebar</div>
28         <div class="col-10" id="content">
29           <div id="root"></div>
30         </div>
31       </div>
32     </div>
33   </body>
34 </html>
```

Gambar 1.7. Modifikasi index.html

9. Amati tampilan yang ada pada browser (seperti Gambar 1.8)



Gambar 1.8. Tampilan hasil modifikasi

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

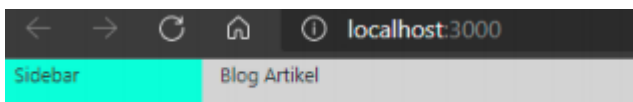
```

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: #002D4D;
17     text-decoration-color: white;
18     color: white;
19     font-size: 20px;
20 }
21
22 #content{
23     background-color: lightgray;
24 }

```

Gambar 1.9. Penambahan code css

11. Perhatikan kembali browser, dan lihat hasil tampilan seperti Gambar 1.10.



Gambar 1.10. Hasil penambahan css

Kita ingin sebuah website memiliki tampilan seperti pada Gambar 1.1. Dengan minimal ada gambar artikel, judul, dan deskripsi artikel. Maka contoh *data dummy* yang akan kita pakai bisa menggunakan data dari <http://placeimg.com> contoh <http://placeimg.com/120/120/any>.

Tahapan edit tampilan post artikel:

12. Ubah kode program untuk *statefull component* `BlogPost.jsx` menjadi seperti Gambar 1.11

```

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

13. Tambahkan custom css ke [BlogPost.css](#) seperti Gambar 1.12

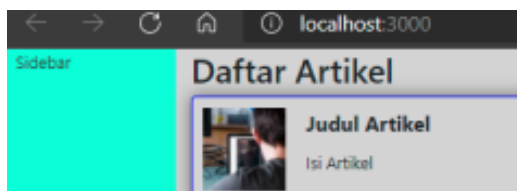

```

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  }
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;

```

Gambar 1.12. Edit kode program BlogPost.css

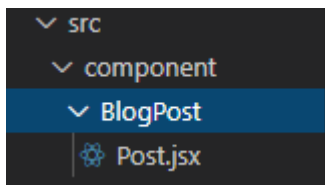
14. Perhatikan tampilan browser.



Pemindahan dari *statefull component* ke *stateless component*

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 *stateless component* untuk dapat digunakan ulang (dipanggil kembali) karena fungsi dari bagian tersebut hanya mengembalikan deskripsi singkat artikel (bukan operasi logic).

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*

```
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
18 export default Post;
```

Post.jsx, dan modifikasi **Post.jsx** seperti Gambar 1.13.

Gambar 1.13. Kode program Post.jsx

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

```
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;
```

seperti Gambar 1.14.

Gambar 1.14. Component BlogPost memanggil component Post

18. 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.

```
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
11           Malang"/>
12       </div>
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!.

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

Sidebar

Daftar Artikel



JTI Polinema

Jurusan Teknologi Informasi - Politeknik Negeri Malang

Mengambil data Post/Artikel dari API.

Bagaimana caranya untuk mendapatkan list artikel berdasarkan data json dari web API (contohnya: <https://jsonplaceholder.typicode.com/posts>) ?

Kita gunakan *life cycle component* yaitu `componentDidMount()` dimana ketika komponen selesai di-*mount*-ing, program akan memanggil 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* (seperti pada Gambar 1.17)

```
[
  {
    "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\nnihil aut fugiat sit autem sed est\nvoluptatem omnis possimus esse voluptatibus quis\nsit 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 ve et doloremque molestiae"
  }
]
```

Gambar 1.17. Data response json dari web API

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

```

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 class BlogPost extends Component{
17   state = {           // komponen state dari React untuk statefull component
18     listArtikel:[]    // variabel array yang digunakan untuk menyimpan data API
19   }
20
21   componentDidMount(){ // komponen untuk mengecek ketika component telah di-mount-ing, maka panggil API
22     fetch('https://jsonplaceholder.typicode.com/posts') // alamat URL API yang ingin kita ambil datanya
23     .then(Response => Response.json()) // ubah response data dari URL API menjadi sebuah data json
24     .then(jsonHasilAmbilDariAPI => { // data json hasil dari API kita masukkan ke dalam listArtikel pada state
25       this.setState({
26         listArtikel: jsonHasilAmbilDariAPI
27       })
28     })
29   }
30
31   render(){
32     return(
33       <div class="post-artikel">
34         <h2>Daftar Artikel</h2>
35         {
36           this.state.listArtikel.map(artikel => { // looping dan masukkan untuk setiap data yang ada di listArtikel ke variabel artikel
37             return <Post judul={artikel.title} isi={artikel.body}/> // mapping data json dari API sesuai dengan kategorinya
38           })
39         }
40       </div>
41     )
42   }
43 }

```

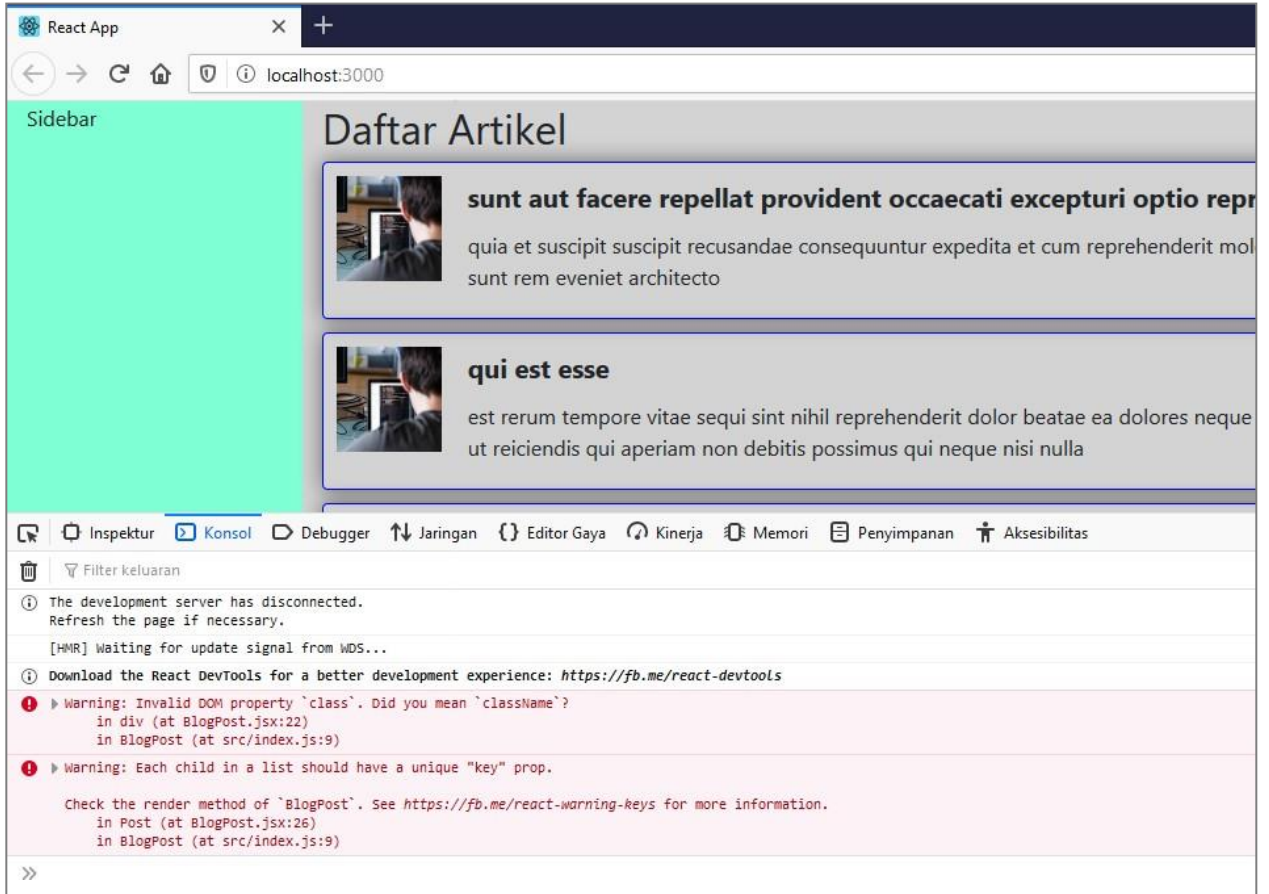
Gambar 1.18. Penambahan componentDidMount pada *statefull component* BlogPost

25. Lihat hasilnya pada browser. Kemudian [klik kanan](#) pada browser pilih "[inspect element](#)" kemudian pilih tab "[console](#)". *Refresh* browser dan amati apa yang terjadi.



Data artikel berubal dan pada console terdapat warning

26. Jika terlihat seperti pada Gambar 1.19, maka terjadi kesalahan pada program yang kita buat.



Gambar 1.19. Error pada browser

27. Jika terjadi hal demikian, hal ini terjadi karena dalam react "**class**" dalam tag html harus ditulis menjadi "**className**". selain itu, pada *statefull component* yang dinamis, harus ada "**UNIQUE KEY**" pada tiap komponen yang diproses sehingga komponen perlu diberi **UNIQUE KEY**.
28. **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.

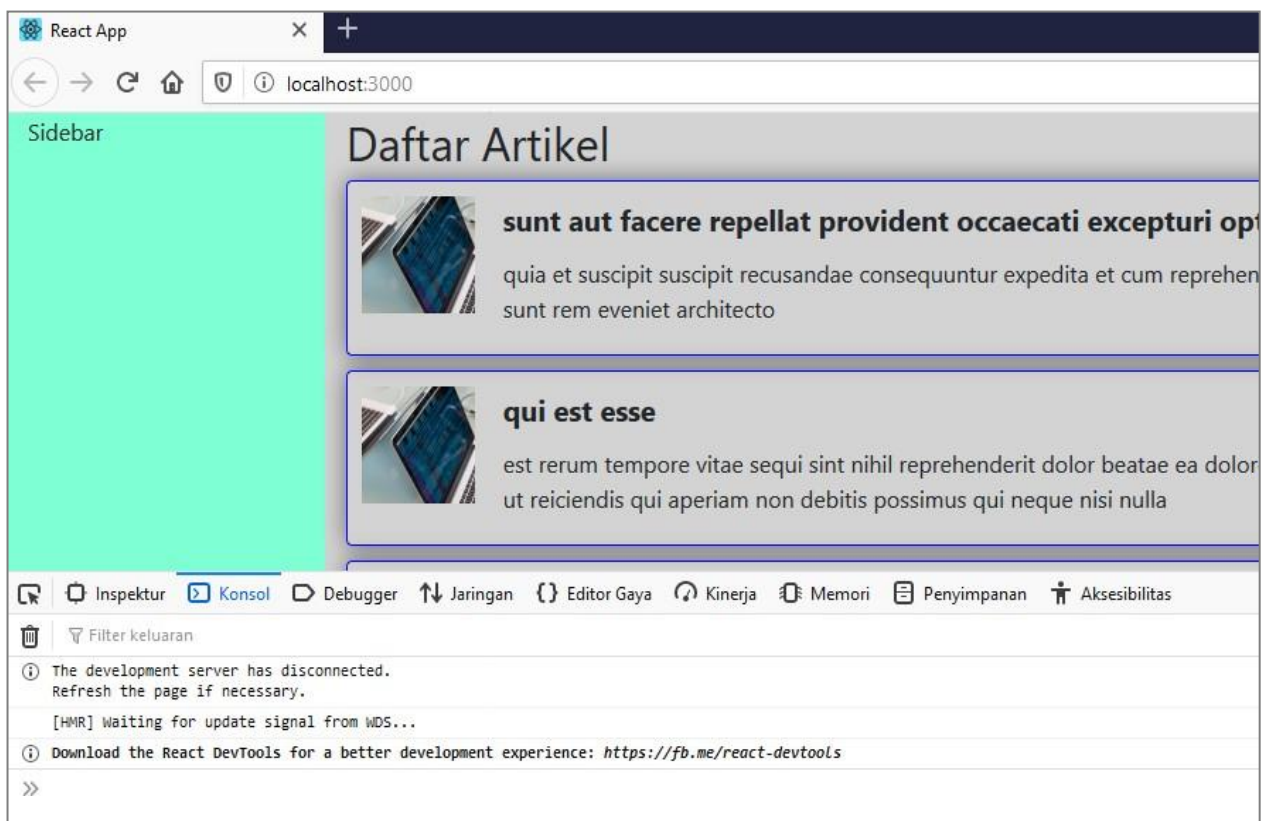
```

20  render() {
21    return(
22      <div className="post-artikel">
23        <h2>Daftar Artikel</h2>
24        {
25          this.state.listArtikel.map(artikel => { // looping dan masuk
26            return <Post key={artikel.id} judul={artikel.title} isi=
27          })
28        }
29      </div>
30    )
31  }

```

Gambar 1.20. Penambahan key pada stateless component

29. Simpan dan lihat apa yang terjadi pada *console* browser (Gambar 1.21).



Gambar 1.21. Hasil akhir

1.4 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?

- Container



- Container fluid



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

Container=ukuran dari layar tidak full di layar browser

Container-fluid= ukuran dari layar full ke layar 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

- Kita ketik di cmd npm install bootstrap

c. untuk dapat menggunakan component tersebut? Bagaimana caranya?

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.

2.1 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

1. Install pada direktori project reactjs kita dengan perintah npm `install -g json-server`

```
PS D:\framework\minggu4\web-react> npm install -g json-server
C:\Users\USER\AppData\Roaming\npm\json-server -> C:\Users\USER\AppData\Roaming\npm\node_modules\json-server\lib\cli\bin.js
+ json-server@0.16.3
added 182 packages from 80 contributors in 62.485s
PS D:\framework\minggu4\web-react> |
```

2. *Copy*-kan file json `listArtikel.json` yang sudah ada pada direktori project reactjs kita.
3. Buka cmd baru pada direktori project, lalu ketik perintah `json-server --watch listArtikel.json --port 3001`.
4. Apabila pada cmd tampil seperti Gambar 2.1, maka server *Fake API* local kita telah siap.

```
\(^_^)/ hi!

Loading listArtikel.json
Done

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

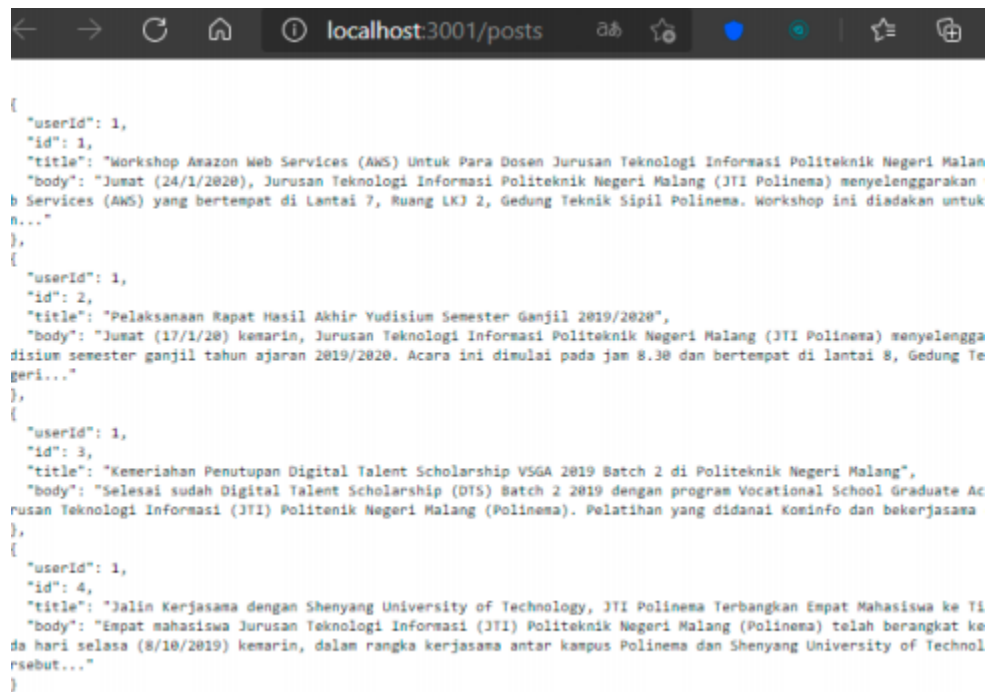
Home
http://localhost:3001

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

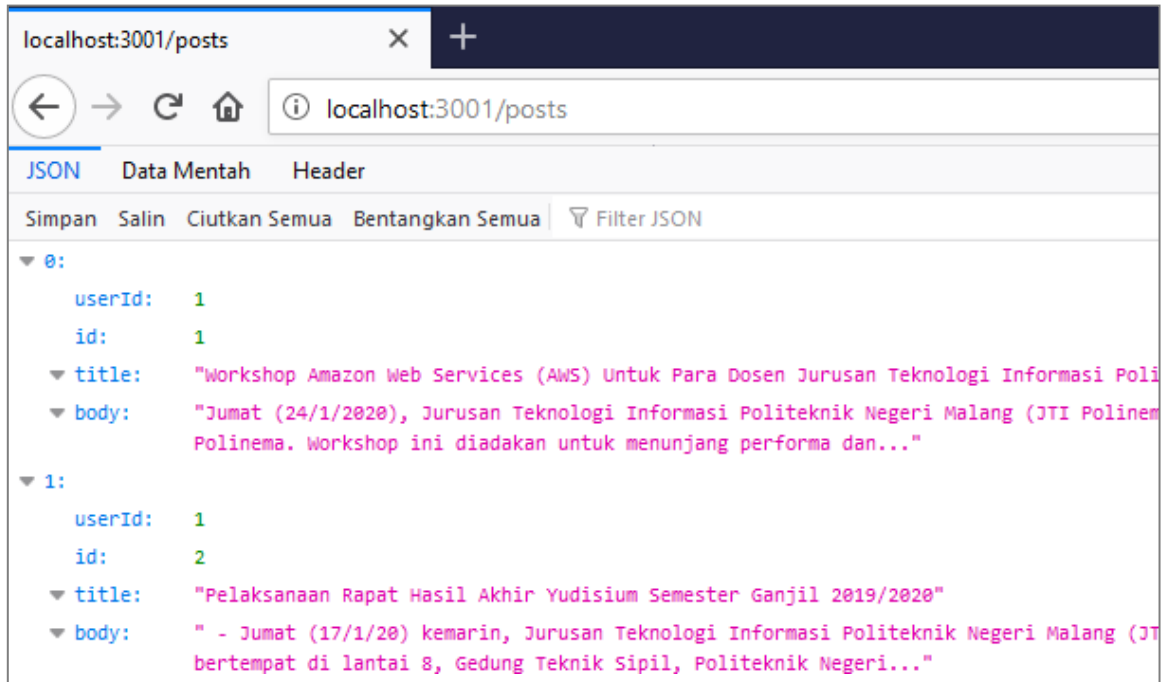
Gambar 2.1 Tampilan response json-server

5. Kita cek *url resource* yang ada pada Fake API server ke browser apakah bisa diakses. Ketik

url <http://localhost:3001/posts> pada browser.



```
{
  "userId": 1,
  "id": 1,
  "title": "Workshop Amazon Web Services (AWS) Untuk Para Dosen Jurusan Teknologi Informasi Politeknik Negeri Malang",
  "body": "Jumat (24/1/2020), Jurusan Teknologi Informasi Politeknik Negeri Malang (JTI Polinema) menyelenggarakan b Services (AWS) yang bertempat di Lantai 7, Ruang LKJ 2, Gedung Teknik Sipil Polinema. Workshop ini diadakan untuk n..."
},
{
  "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) menyelenggarakan disium semester ganjil tahun ajaran 2019/2020. Acara ini dimulai pada jam 8.30 dan bertempat di lantai 8, Gedung Te geri..."
},
{
  "userId": 1,
  "id": 3,
  "title": "Kemeriahan Penutupan Digital Talent Scholarship VSGA 2019 Batch 2 di Politeknik Negeri Malang",
  "body": "Selesai sudah Digital Talent Scholarship (DTS) Batch 2 2019 dengan program Vocational School Graduate Ac rusan Teknologi Informasi (JTI) Politeknik Negeri Malang (Polinema). Pelatihan yang didanai Kominfo dan bekerjasama ),
{
  "userId": 1,
  "id": 4,
  "title": "Jalin Kerjasama dengan Shenyang University of Technology, JTI Polinema Terbangkan Empat Mahasiswa ke TI",
  "body": "Empat mahasiswa Jurusan Teknologi Informasi (JTI) Politeknik Negeri Malang (Polinema) telah berangkat ke da hari Selasa (8/10/2019) kemarin, dalam rangka kerjasama antar kampus Polinema dan Shenyang University of Technol rsebut..."
}
```

Gambar 2.2. Tampilan response json-server

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>

```
componentDidMount(){ // komponen untuk mengecek ketika co
di-mount-ing, maka panggil API
  fetch('http://localhost:3001/posts') // alamat URL API y
  datanya
  .then(Response => Response.json()) // ubah response dat
  menjadi sebuah data json
  .then(jsonHasilAmbilDariAPI => { // data json hasil d
  ke dalam listArtikel pada state
    this.setState({
      listArtikel: jsonHasilAmbilDariAPI
    })
  })
})
```

7. Simpan perubahan dan amati apa yang terjadi.



2.2 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?
 - Karena port 3000 sudah dipakai dari project react
- b. Bagaimana jadinya kalau kita ganti *port json-server* menjadi 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.

3.1 Langkah Praktikum 3

1. 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)}>Hap
13       </div>
14     </div>
15   )
16 }
17
18 export default Post;
```

Gambar 3.1 Tambah kode program Post.jsx

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

```

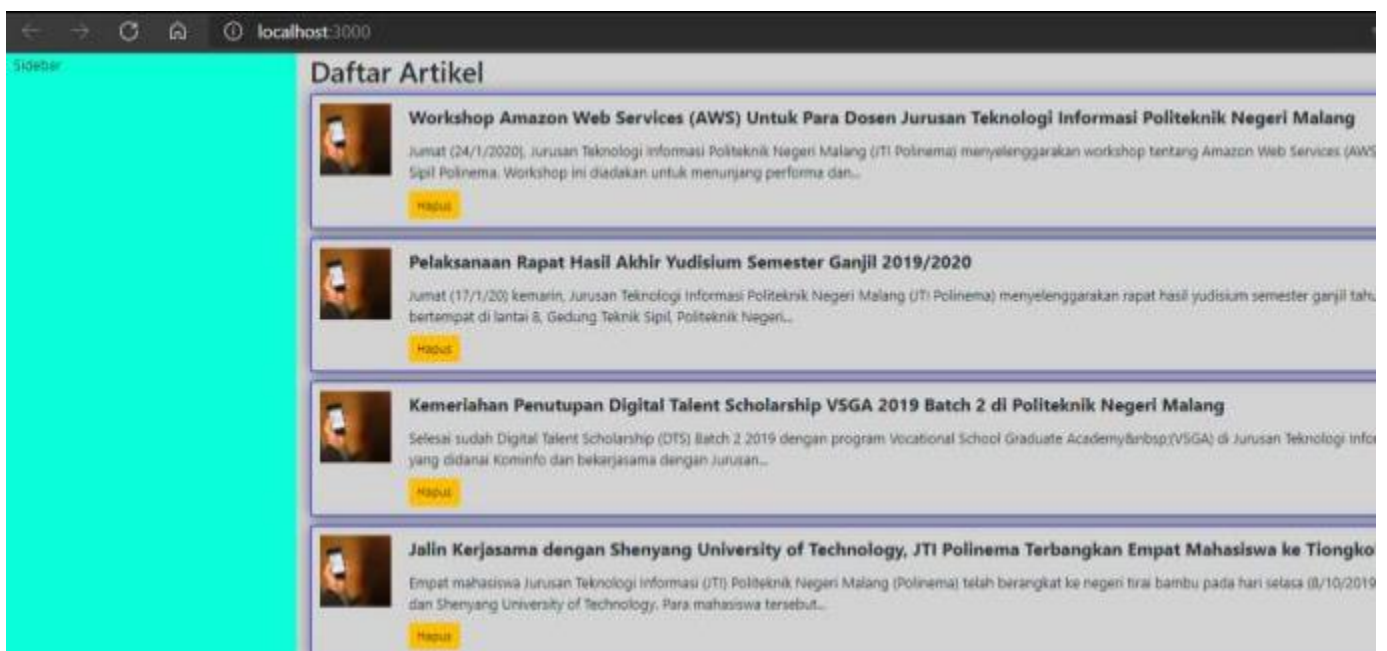
16 class BlogPost extends Component{
17   state = {} // komponen state dari React untuk statefull component
18   listArtikel:[], // variabel array yang digunakan untuk menyimpan data API
19   insertArtikel:{ // variabel yang digunakan untuk manampung sementara data yanag akan di insert
20     userId: 1, // kolom userId, id, title, dan body sama, mengikuti kolom yang ada pada listArtikel.json
21     id: 1,
22     title: "",
23     body: ""
24   }
25 }
26
27 ambilDataDariServerAPI = () => {
28   fetch('http://localhost:3001/posts') // alamat URL API yang ingin kita ambil datanya
29   .then(Response => Response.json()) // ubah response data dari URL API menjadi sebuah data json
30   .then(jsonHasilAmbilDariAPI => { // data json hasil dari API kita masukkan ke dalam listArtikel pada state
31     this.setState({
32       listArtikel: jsonHasilAmbilDariAPI
33     })
34   })
35 }
36
37 componentDidMount(){ // komponen untuk mengecek ketika component telah di-mount-ing, maka panggil API
38   this.ambilDataDariServerAPI() // ambil data dari server local
39 }
40
41 handleHapusArtikel = (data) => {
42   fetch(`http://localhost:3001/posts/${data}`, {method: 'DELETE'}) // alamat URL API yang ingin kita HAPUS datanya
43   .then(res =>{
44     this.ambilDataDariServerAPI() // ketika proses berhasil, maka ambil dari server API lokal
45   })
46 }
47

```

Gambar 3.2 Modifikasi kode program BlogPost

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

- Sebelum di hapus



- Sesudah dihapus



3.2 Pertanyaan Praktikum 3

1. Apa yang terjadi setelah kalian klik tombol hapus?
 - Artikel terhapus
2. Perhatikan file `listArtikel.json`, apa yang terjadi pada file tersebut? Kenapa demikian?
 - Isi json yg semula ada 4 menjadi 3 karena sudah di hapus pada tombol hapus
3. Fungsi `handleHapusArtikel` itu untuk apa?
 - Handle action button hapus data
4. Jelaskan perbedaan fungsi `componentDidMount()` pada Gambar 1.18 dengan fungsi `componentDidMount()` pada Gambar 3.2 ?
 - Pada gambar 1.8 digunakan mengambil data dari server, sedangkan pada 3.2 digunakan untuk memanggil fungsi ambildatadari server

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.

4.1 Langkah Praktikum 4

1. Buka *statefull component* `BlogPost`, dan modifikasi pada fungsi `render()` untuk menampilkan *form input* artikel yang berisi judul dan isi berita. seperti pada Gambar 4.1

```
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 name="body" id="body" cols="3" rows="3"  
            className="form-control" col="3" onChange={this.handleTambahArtikel}></div>  
        </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 set  
        data yang ada di listArtikel ke variabel artikel  
        return <Post key={artikel.id} judul={artikel.title} isi={artikel.body}  
          idArtikel = {artikel.id} hapusArtikel = {this.handleHapusArtikel}/> //  
        mapping data json dari API sesuai dengan kategorinya  
      })  
    }  
  )  
</div>
```

Gambar 4.1 modifikasi component BlogPost

2. Kemudian modifikasi `BlogPost` untuk bagian `state` dan *request* API dari server, seperti Gambar 4.2

```

class BlogPost extends Component{
  state = {                                // komponen state dari React untuk statefull
    listArtikel:[],                        // variabel array yang digunakan untuk meny
    insertArtikel:{                        // variabel yang digunakan untuk manampung se
      userId: 1,                          // kolom userId, id, title, dan body sama, me
      id: 1,
      title: "",
      body: ""
    }
  }

  ambilDataDariServerAPI = () => {
    fetch('http://localhost:3001/posts') // alamat URL API yang ingin
    .then(Response => Response.json()) // ubah response data dari UR
    .then(jsonHasilAmbilDariAPI => {      // data json hasil dari API
      this.setState({
        listArtikel: jsonHasilAmbilDariAPI
      })
    })
  }
}

```

Gambar 4.2 penambahan state pada BlogPost

3. Tambahkan untuk *handle* form tambah data artikel seperti Gambar 4.3

```

handleHapusArtikel = (data) => {
  fetch(`http://localhost:3001/posts/${data}`, {method: 'DELETE'}) // alamat URL API yan
  .then(res =>{
    this.ambilDataDariServerAPI() // ketika proses berhasil, maka ambil dari server A
  })
}

handleTambahArtikel = (event) => {      // fungsi untuk meng-handle form tambah data arti
  let formInsertArtikel = {...this.state.insertArtikel}; // clonning data state insert
  let timestamp = new Date().getTime(); //digunakan untuk menyimpan waktu (sebagai ID
  formInsertArtikel['id'] = timestamp;
  formInsertArtikel[event.target.name] = event.target.value; // menyimpan data onChange
  this.setState( {
    insertArtikel: formInsertArtikel
  });
}

```

Gambar 4.3 Handle tambah artikel

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

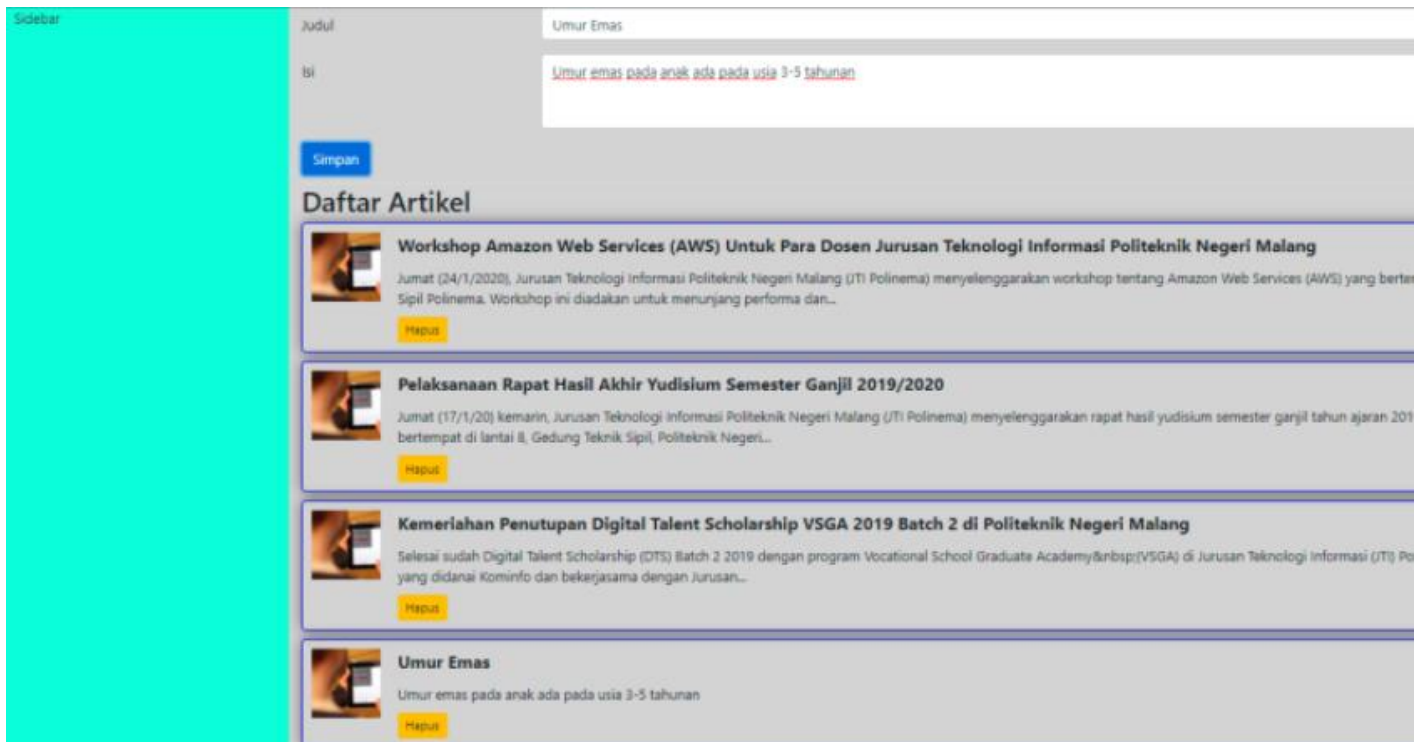
```

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

```

Gambar 4.4 Handle tombol simpan

5. Simpan, lakukan percobaan penambahan data, dan amati perubahannya.



4.2 Pertanyaan Praktikum 4

- a. Jelaskan apa yang terjadi pada file `listArtikel.json` sebelum dan setelah melakukan penambahan data?
 - Sebelum ada penambahan data, data pada `listArtikel` hanya 3. Setelah penambahan data, data bertambah menjadi 4

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?

- Data pada json disimpan dalam bentuk array, sehingga data diurutkan berdasarkan index dari terkecil ke terbesar

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.

Contoh data json yang digunakan.

```
{
  "mahasiswa": [
    {
      "NIM": 180823453,
      "nama": "Mahasiswa 1",
      "alamat": "Jl. Soekarno Hatta No. 9 Malang",
      "hp": "081802023249",
      "angkatan": 2018,
      "status": "aktif"
    },
    {
      "NIM": 140823453,
      "nama": "Mahasiswa X",
      "alamat": "Jl. Menur No. 9 Surabaya",
      "hp": "081802023523",
      "angkatan": 2014,
      "status": "lulus"
    }
  ]
}
```

ListMahasisiwa

```

1  {
2    "mahasiswa": [
3      {
4        "NIM": 1841720001,
5        "id": 1,
6        "nama": "Liiili",
7        "alamat": "Jl. Soekarno Hatta No. 29 Malang",
8        "hp": "081678900",
9        "angkatan": 2018,
0        "status": "Aktif"
1      },
2      {
3        "NIM": 174172002,
4        "id": 2,
5        "nama": "nanaa",
6        "alamat": "Jl. Menur No. 19 Surabaya",
7        "hp": "0814567890",
8        "angkatan": 2017,
9        "status": "Aktif"
0      },
1      {
2        "NIM": 1941720088,
3        "id": 3,
4        "nama": "Riyan",
5        "alamat": "Jl. Indah No.17 Malang",
6        "hp": "0890986777",
7        "angkatan": 2019,
8        "status": "Aktif"
9      }
0    ]
1  }

```

Mahasiswapost.jsx

```

import React, {Component} from "react";
import './MahasiswaPost.css'
import Post from '../component/MahasiswaPost/PostMahasiswa'

class MahasiswaPost extends Component{
  state = {
    listMahasiswa:[],
    insertMahasiswa:{
      NIM: "",
      id: 1,
      nama: "",

```

```

        alamat: "",
        hp: "",
        angkatan: "",
        status: ""
    }
}

ambilDataDariServerAPI = () => {
    fetch('http://localhost:3002/mahasiswa')
    .then(Response => Response.json())
    .then(jsonHasilAmbilDariAPI => {
        this.setState({
            listMahasiswa: jsonHasilAmbilDariAPI
        })
    })
}

componentDidMount(){
    this.ambilDataDariServerAPI()
}

handleHapusMahasiswa = (data) => {
    fetch(`http://localhost:3002/mahasiswa/${data}`, {method: 'DELETE'})
    .then(res =>{
        this.ambilDataDariServerAPI()
    })
}

handleTambahMahasiswa = (event) => {
    let formInsertMahasiswa = {...this.state.insertMahasiswa};
    let timestamp = new Date().getTime();
    formInsertMahasiswa['id'] = timestamp;
    formInsertMahasiswa[event.target.name] = event.target.value;
    this.setState( {
        insertMahasiswa: formInsertMahasiswa
    });
}

handleTombo1Simpan = () => {
    fetch('http://localhost:3002/mahasiswa', {
        method: 'post',
        headers: {
            'Accept' : 'application/json',
            'Content-Type': 'application/json'
        },

```

```

        body: JSON.stringify(this.state.insertMahasiswa)
    })
    .then( ( Response ) => {
        this.ambilDataDariServerAPI();
    });
}

render(){
    return(
        <div className="post-Mahasiswa">
            <h2>Tambah Data Mahasiswa</h2>
            <div className="form pb-2 border-bottom">
                <div className="form-group row">
                    <label htmlFor="title" className="col-sm-2 col-form-
label">NIM</label>
                    <div className="col-sm-10">
                        <input type="text" className="form-
control" id="NIM" name="NIM" onChange={this.handleTambahMahasiswa}/>
                    </div>
                </div>
                <div className="form-group row">
                    <label htmlFor="body" className="col-sm-2 col-form-
label">Nama</label>
                    <div className="col-sm-10">
                        <textarea id="nama" name="nama" cols="1" rows="1" classNa
me="form-control" col="3" onChange={this.handleTambahMahasiswa}></textarea>
                    </div>
                </div>
                <div className="form-group row">
                    <label htmlFor="body" className="col-sm-2 col-form-
label">Alamat</label>
                    <div className="col-sm-10">
                        <textarea id="alamat" name="alamat" cols="3" rows="3" cla
ssName="form-control" col="3" onChange={this.handleTambahMahasiswa}></textarea>
                    </div>
                </div>
                <div className="form-group row">
                    <label htmlFor="body" className="col-sm-2 col-form-
label">No HP</label>
                    <div className="col-sm-10">
                        <textarea id="hp" name="hp" cols="1" rows="1" className="
form-control" col="3" onChange={this.handleTambahMahasiswa}></textarea>
                    </div>
                </div>
                <div className="form-group row">

```

```

        <label htmlFor="body" className="col-sm-2 col-form-
label">Angkatan</label>
        <div className="col-sm-10">
            <textarea id="angkatan" name="angkatan" cols="1" rows="1"
            className="form-
control" col="3" onChange={this.handleTambahMahasiswa}></textarea>
        </div>
    </div>
    <div className="form-group row">
        <label htmlFor="body" className="col-sm-2 col-form-
label">Status Mahasiswa</label>
        <div className="col-sm-10">
            <textarea id="status" name="status" cols="1" rows="1" cla
ssName="form-control" col="3" onChange={this.handleTambahMahasiswa}></textarea>
        </div>
    </div>
    <button type="submit" className="btn btn-
primary" onClick={this.handleTombolSimpan} >Simpan</button>
</div>
<h2>Daftar Mahasiswa</h2>
{
    this.state.listMahasiswa.map(Mahasiswa => {
        return <Post key={Mahasiswa.id} NIM={Mahasiswa.NIM} nama=
{Mahasiswa.nama} alamat={Mahasiswa.alamat} hp={Mahasiswa.hp} angkatan={Mahasiswa.
angkatan} status={Mahasiswa.status} idMahasiswa={Mahasiswa.id} hapusMahasiswa = {
this.handleHapusMahasiswa}/>
    })
}
</div>
)
}
}
export default MahasiswaPost;

```

postmahasiswa.jsx

```

import React from "react"

const MahasiswaPost = (props) => {
    return(
        <div className="mahasiswa">
            <div className="gambar-polinema">
                
            </div>
        </div>
    )
}

```

```

        </div>
        <div className="konten-mahasiswa">
          <div className="nimnam-
mahasiswa">{props.NIM} - {props.nama}</div>
          <p className="isi-mahasiswa">-----
-----</p>
          <p className="isi-mahasiswa">{props.alamat}</p>
          <p className="isi-mahasiswa">{props.hp}</p>
          <p className="isi-mahasiswa">{props.angkatan}</p>
          <p className="isi-mahasiswa">{props.status}</p>
          <button className="btn btn-sm btn-
warning" onClick={() => props.hapusMahasiswa(props.idMahasiswa)}>Hapus</button>
        </div>
      </div>
    )
  }

export default MahasiswaPost;

```

link github: <https://github.com/lilipolinema/PBFReact.git>

~ ~ ~ Selamat Mengerjakan ~ ~ ~