

INTERAKSI DENGAN API – Pertemuan 4  
PEMOGRAMAN BERBASIS FRAMEWORK



Dosen Pengampu:

Arie Rahmat Syulistyo, S. Kom, M. Kom

Oleh:

Christian Daniel Prayogo

1941720181

TI 3F

Jurusan Teknologi Informasi

Prodi D4 Teknik Informatika

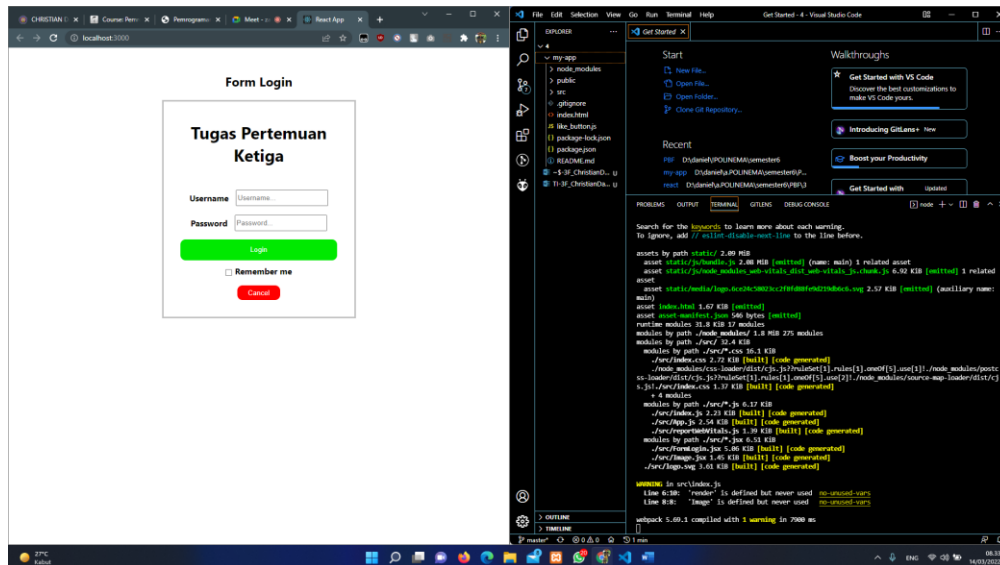
Politeknik Negeri Malang

2022

## Praktikum 1 Interaksi dengan API menggunakan method GET

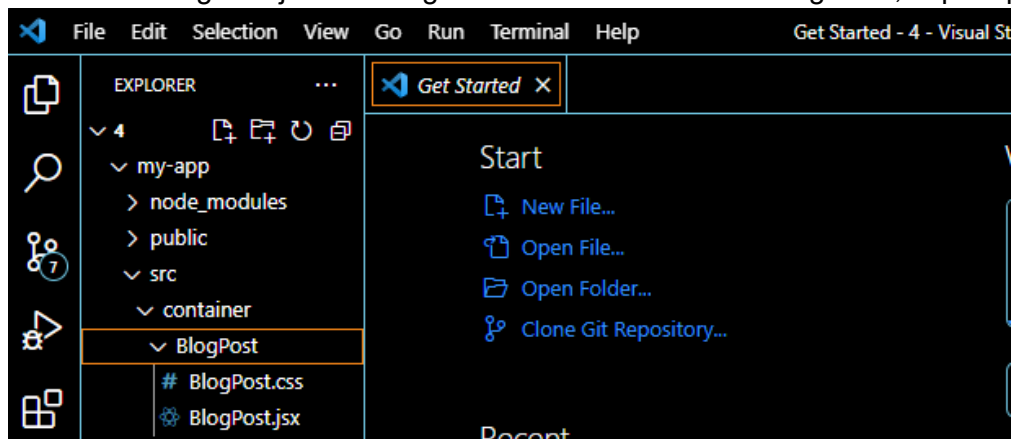
### 1.3 Langkah Praktikum

1. Buka Project React pada pertemuan sebelumnya dan jalankan “npm start” menggunakan cmd dalam direktori tersebut.

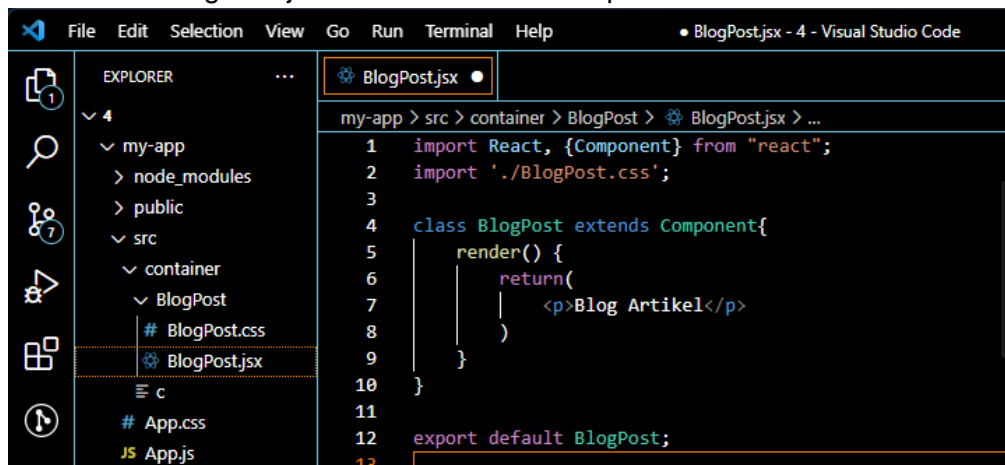


2. Buat folder baru bernama “BlogPost” pada folder container (statefull component).

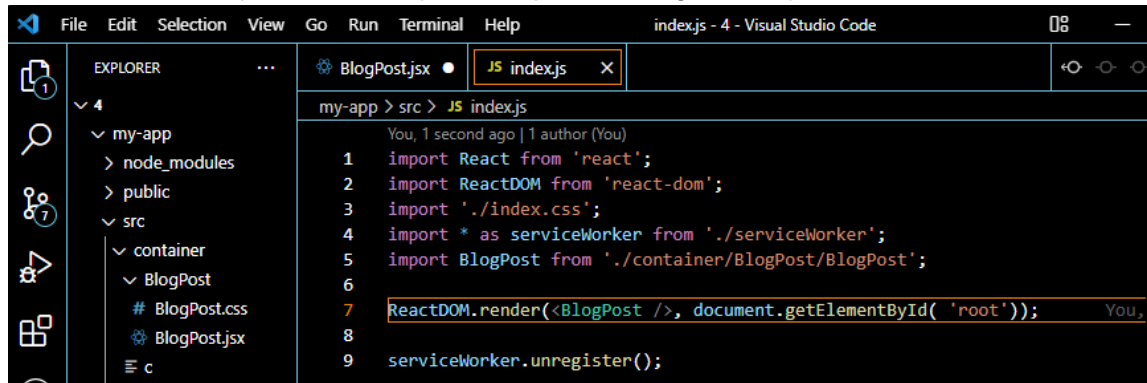
3. Buat file BlogPost.jsx dan BlogPost.css di dalam folder “BlogPost”, seperti pada Gambar 1.2.



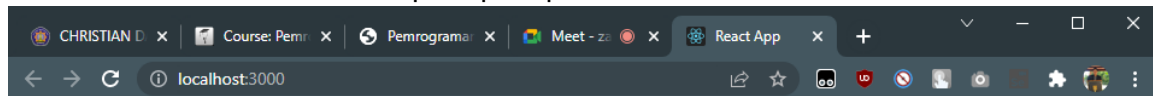
4. Buka file BlogPost.jsx dan ketikkan kode seperti Gambar 1.3.



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



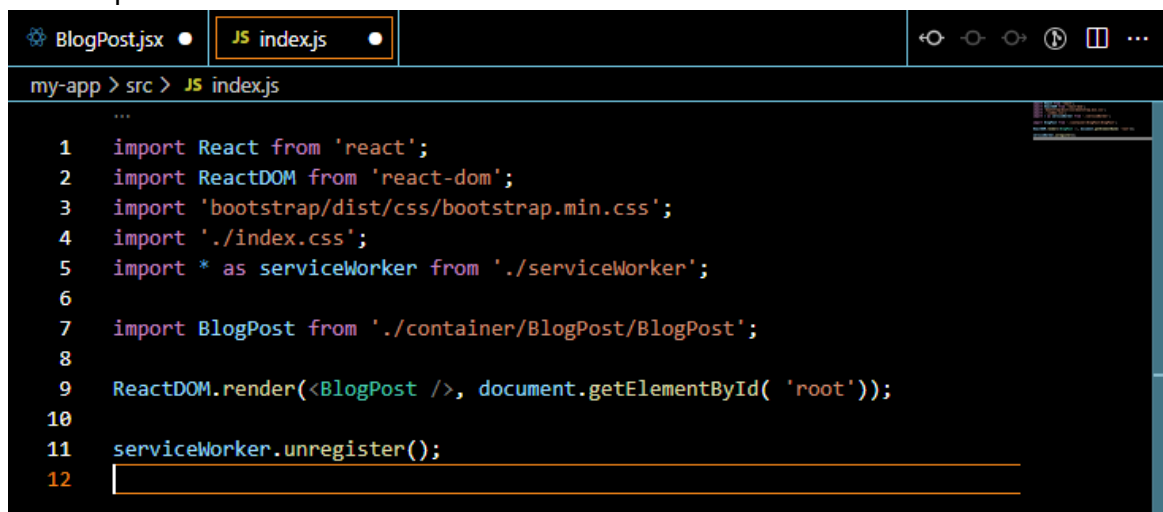
6. Pada web browser akan tampil seperti pada Gambar 1.5.



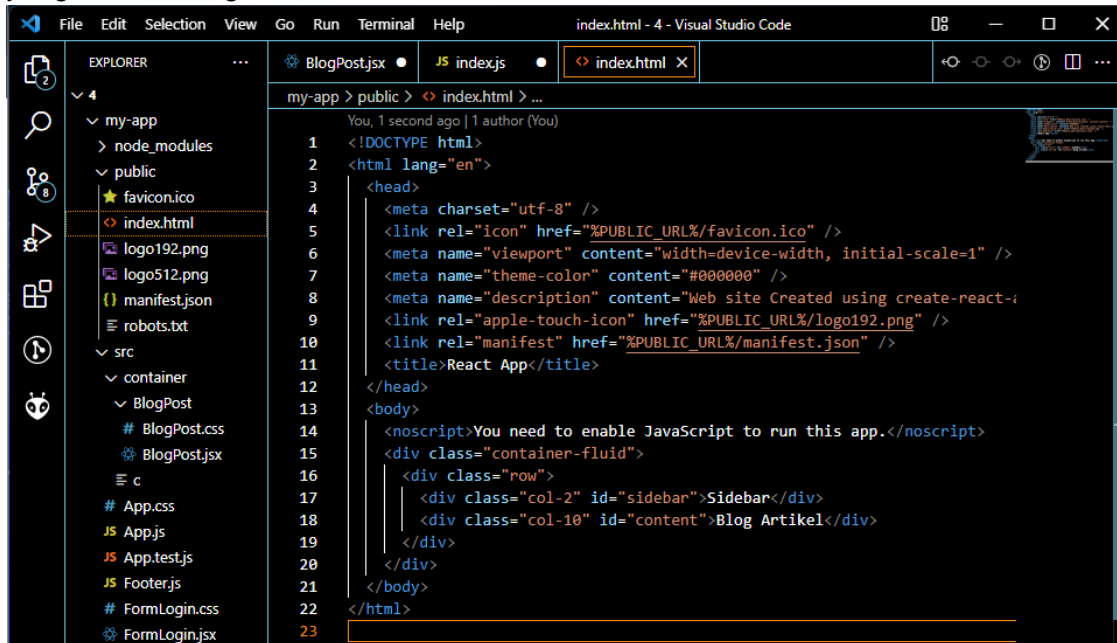
Blog Artikel

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

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

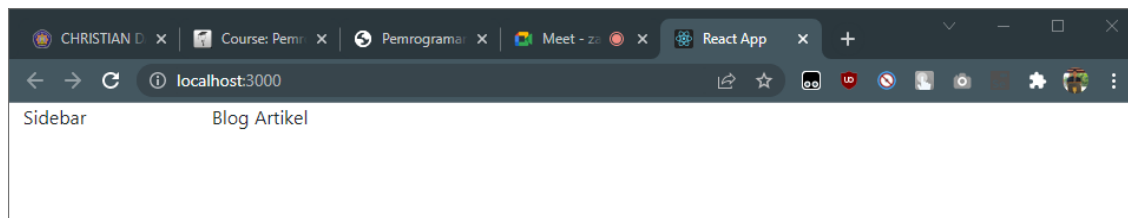


8. Modifikasi file index.html pada folder "public" seperti Gambar 1.7. Cermati code program yang ada dalam gambar!.

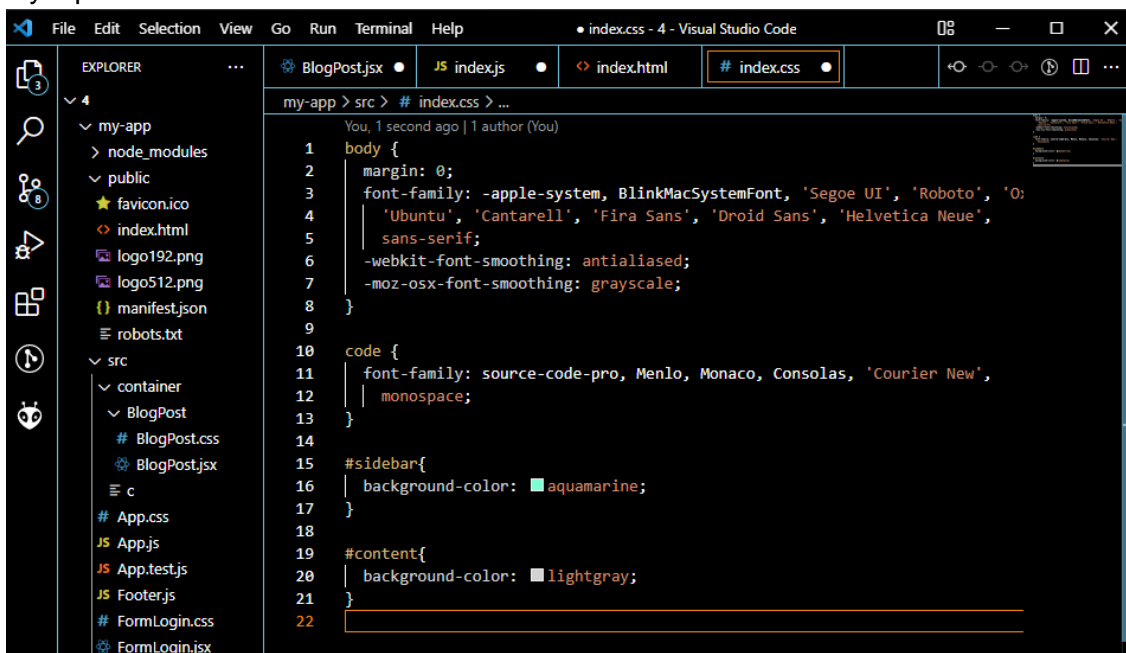


```
1 <!DOCTYPE html>
2 <html lang="en">
3   <head>
4     <meta charset="utf-8" />
5     <link rel="icon" href="%PUBLIC_URL%/favicon.ico" />
6     <meta name="viewport" content="width=device-width, initial-scale=1" />
7     <meta name="theme-color" content="#000000" />
8     <meta name="description" content="Web site Created using create-react-app" />
9     <link rel="apple-touch-icon" href="%PUBLIC_URL%/logo192.png" />
10    <link rel="manifest" href="%PUBLIC_URL%/manifest.json" />
11    <title>React App</title>
12  </head>
13  <body>
14    <noscript>You need to enable JavaScript to run this app.</noscript>
15    <div class="container-fluid">
16      <div class="row">
17        <div class="col-2" id="sidebar">Sidebar</div>
18        <div class="col-10" id="content">Blog Artikel</div>
19      </div>
20    </div>
21  </body>
22 </html>
```

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

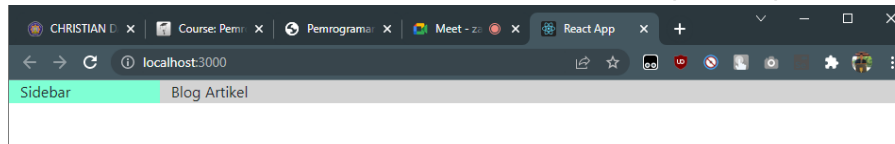


10. Buka file index.css dan tambahkan code css seperti Gambar 1.9, untuk menambah sedikit style pada halaman web



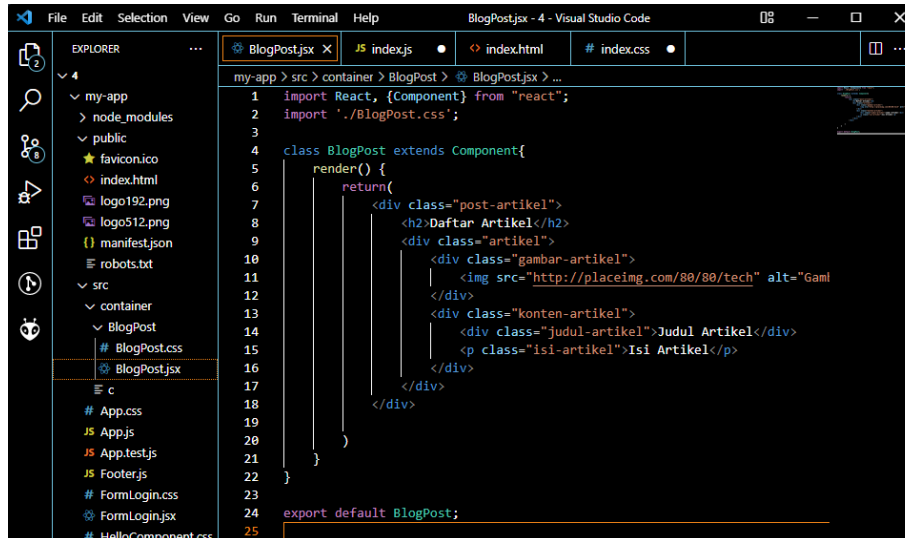
```
1 body {
2   margin: 0;
3   font-family: -apple-system, BlinkMacSystemFont, 'Segoe UI', 'Roboto', 'Oxygen',
4     'Ubuntu', 'Cantarell', 'Fira Sans', 'Droid Sans', 'Helvetica Neue',
5     sans-serif;
6   -webkit-font-smoothing: antialiased;
7   -moz-osx-font-smoothing: grayscale;
8 }
9
10 code {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

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



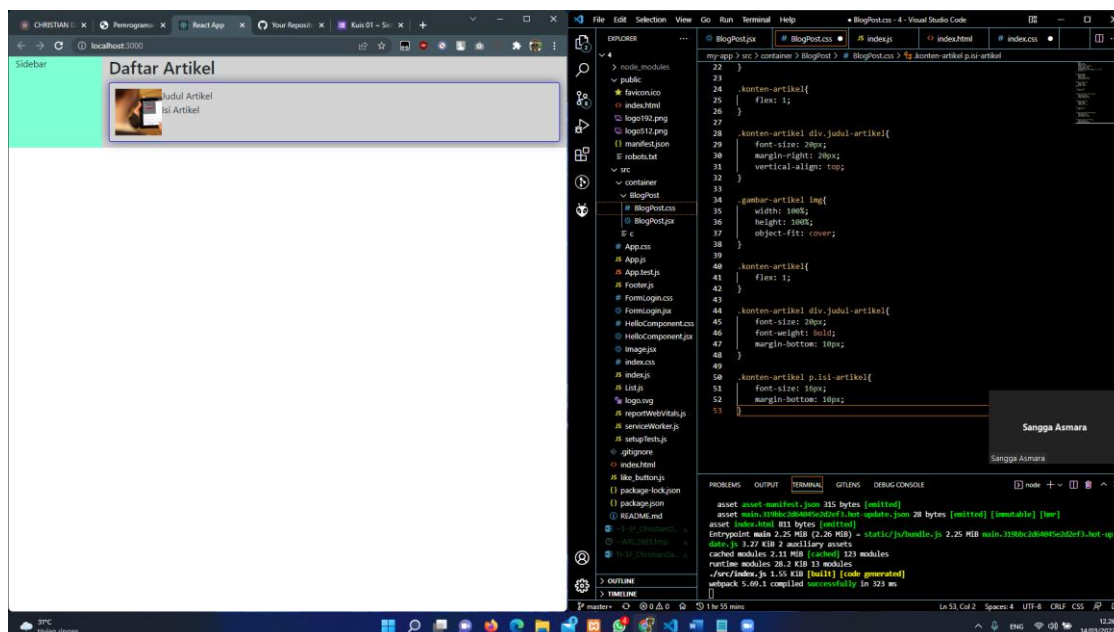
Kita ingin sebuah website memiliki tampilan seperti pada Gambar 1.1. Dengan minimal ada gambar artikel, judul, dan deskripsi artikel. Maka contoh data dummy yang akan kita pakai bisa menggunakan data dari <http://placeimg.com> contoh <http://placeimg.com/120/120/any>. Tahapan edit tampilan post artikel:

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



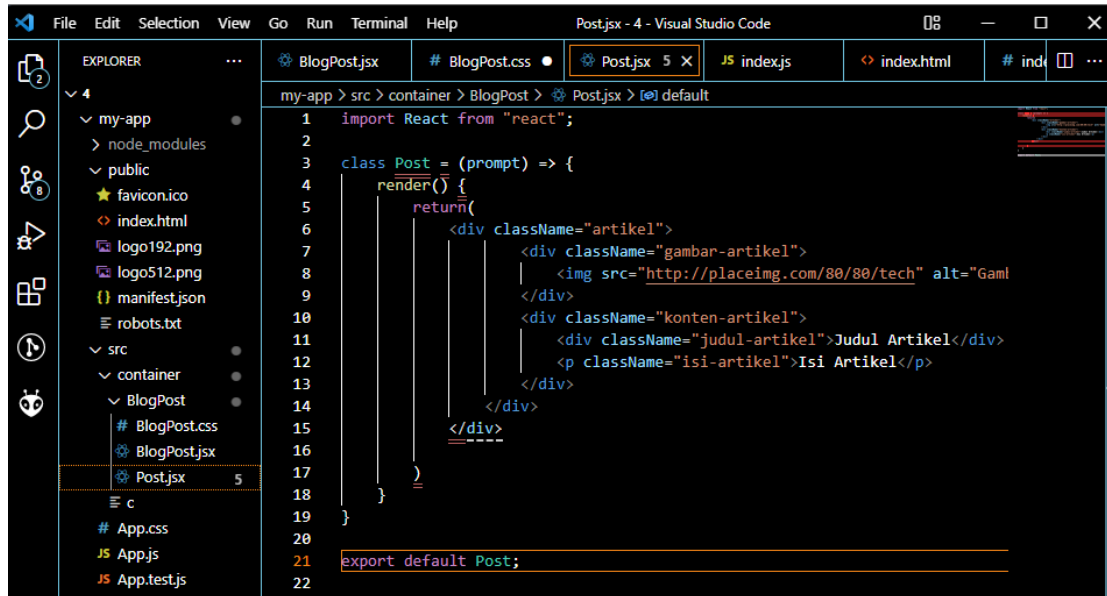
13. Tambahkan custom css ke BlogPost.css seperti Gambar 1.12

14. Perhatikan tampilan browser



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

16. Potong (cut) baris 9-17 pada statefull component BlogPost.jsx ke stateless component Post.jsx, dan modifikasi Post.jsx seperti Gambar 1.13.

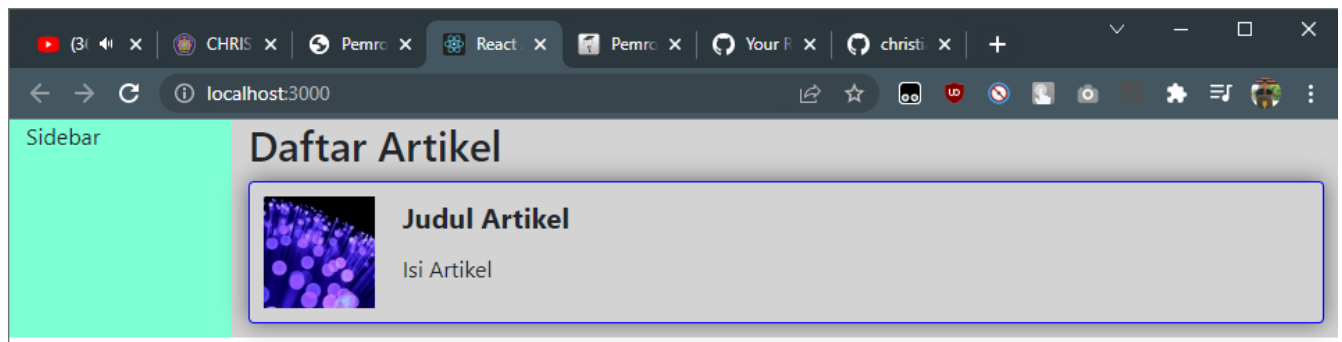


```
1 import React from "react";
2
3 class Post = (prompt) => {
4   render() {
5     return(
6       <div className="artikel">
7         <div className="gambar-artikel">
8           
9         </div>
10        <div className="konten-artikel">
11          <div className="judul-artikel">Judul Artikel</div>
12          <p className="isi-artikel">Isi Artikel</p>
13        </div>
14      </div>
15    )
16  }
17 }
18
19 export default Post;
```

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

seperti Gambar 1.14.

18. Perhatikan hasil tampilan browser, apa yang terjadi?



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

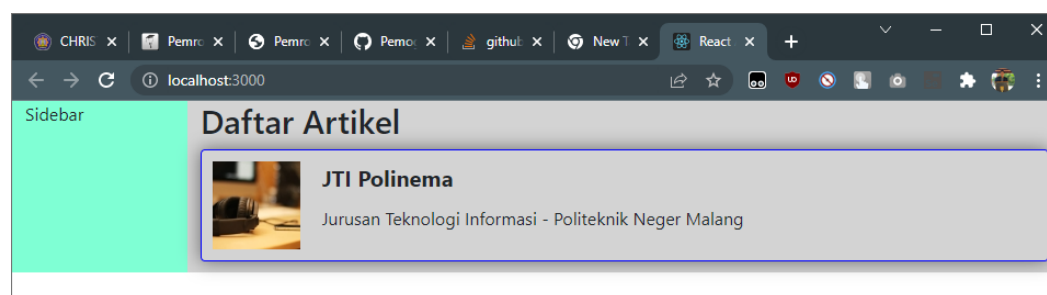
19. Pada statefull component BlogPost.jsx, tambahkan parameter yang ingin dilempar ke stateless component untuk ditampilkan. Kode program bisa dilihat pada Gambar 1.15

```
BlogPost.jsx | Post.jsx 5
4 > my-app > src > container > BlogPost > BlogPost.jsx > ...
1 import React, {Component} from "react";
2 import './BlogPost.css';
3 import Post from './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 - f
11        </div>
12      )
13    }
14  }
15
16
17 export default BlogPost;
18
```

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 2 X
4 > my-app > src > container > BlogPost > Post.jsx > ...
1 import React from "react";
2
3 const Post = (prompt) => {
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;
18
```

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



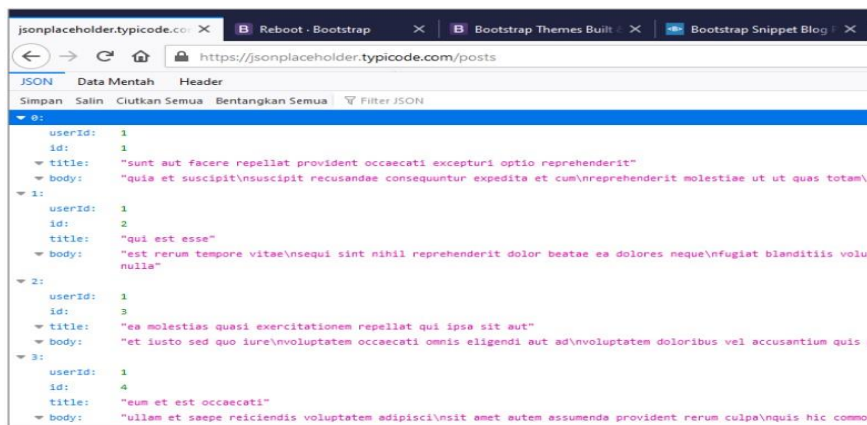
Mengambil data Post/Artikel dari API.

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 dimount-ing, program akan memanggil API.

22. Gunakan state untuk menyimpan data hasil request dari API

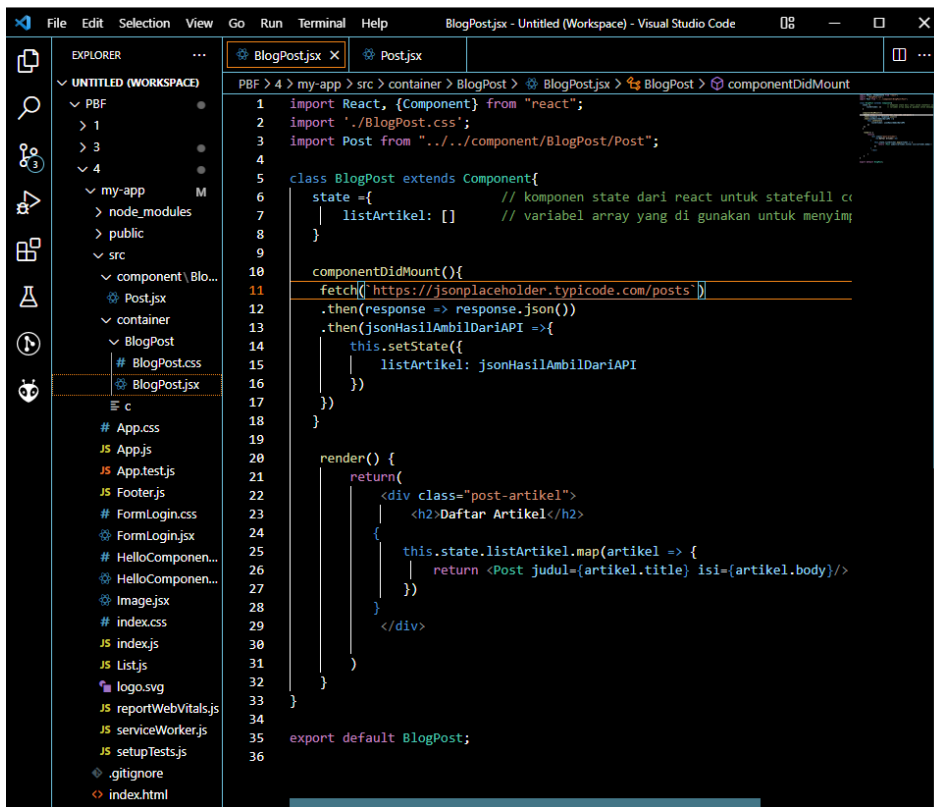
23. data API yang akan kita gunakan adalah data dummy dari <https://jsonplaceholder.typicode.com/posts>, dimana memiliki 4 element data yaitu `userid`, `id`, `title`, `body` (seperti pada Gambar 1.17)



Gambar 1.17. Data response json dari web API

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

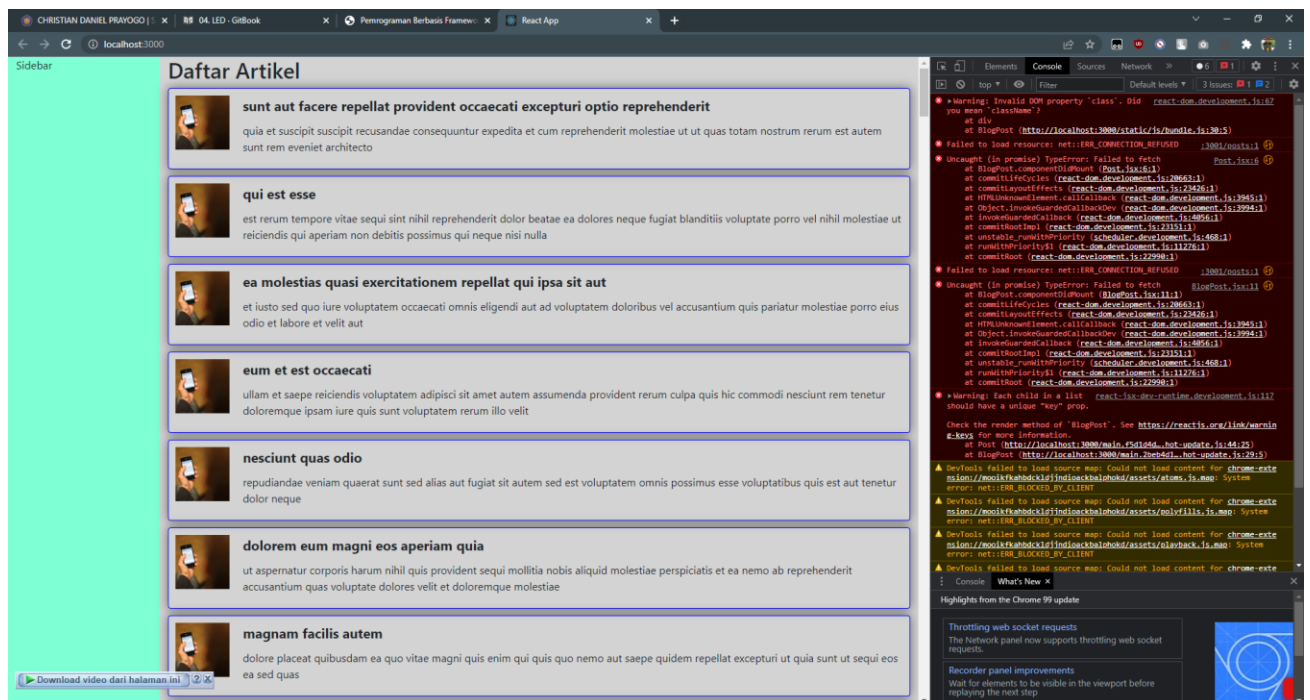




```
1 import React, {Component} from "react";
2 import './BlogPost.css';
3 import Post from "../../component/BlogPost/Post";
4
5 class BlogPost extends Component{
6   state = {
7     listArtikel: [] // variabel array yang di gunakan untuk menyimpan
8   }
9
10  componentDidMount(){
11    fetch('https://jsonplaceholder.typicode.com/posts:1')
12    .then(response => response.json())
13    .then(jsonHasilAmbilDariAPI =>{
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          this.state.listArtikel.map(artikel => {
26            return <Post judul={artikel.title} isi={artikel.body}/>
27          })
28        }
29      </div>
30    )
31  }
32 }
33
34 export default 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.

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

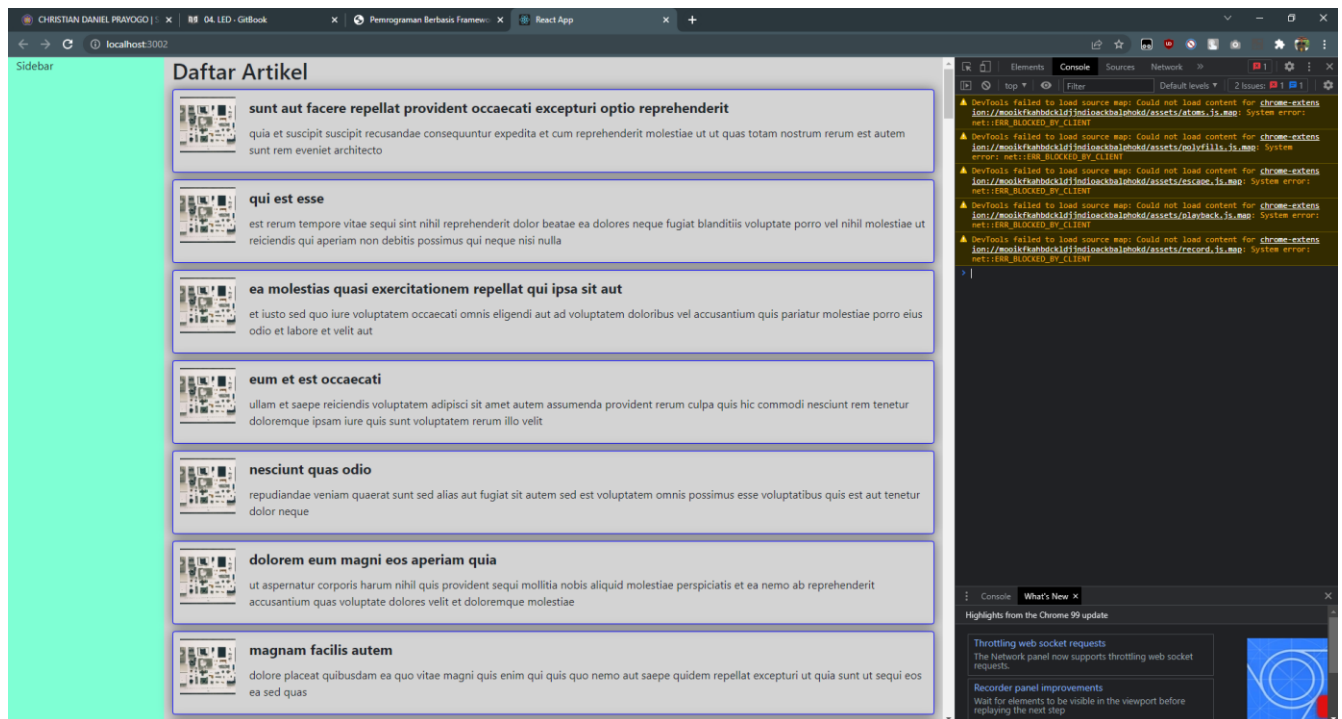


27. Jika terjadi hal demikian, hal ini terjadi karena dalam react "class" dalam tag html harus ditulis menjadi "className". selain itu, pada statefull component yang dinamis, harus ada "UNIQUE KEY" pada tiap komponen yang diproses sehingga komponen perlu diberi UNIQUE KEY.

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.

```
25 |   this.state.listArtikel.map(artikel => {
26 |     |   return <Post key={artikel.id} judul={artikel.title} isi={artikel
27 |     |   })
28 |   }
29 |   </div>
30 |
```

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



#### 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?

2. Apa perbedaan dari container dan container-fluid ?

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?

## Praktikum 2

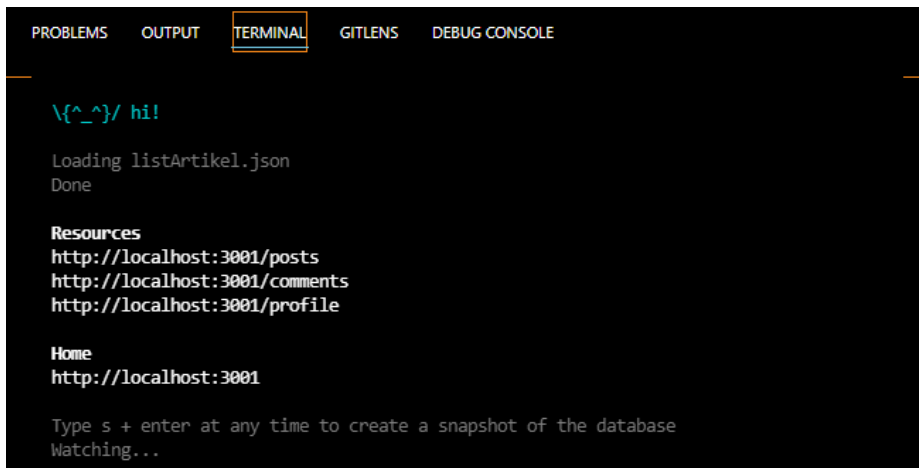
### Interaksi dengan API menggunakan Fake API

#### 2.1 Install Fake API (JSON Server)

Fake API/JSON Server bisa kita dapatkan di halaman <https://github.com/typicode/jsonserver>.

Tahapan install dan membuat data json sendiri

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



```
PROBLEMS OUTPUT TERMINAL GITLENS DEBUG CONSOLE

\{^_^}/ hi!

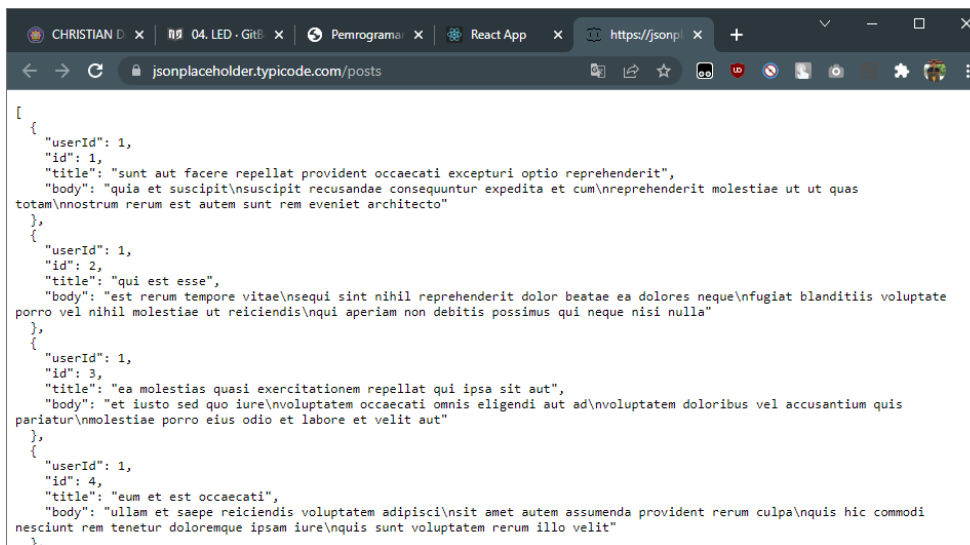
Loading listArtikel.json
Done

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...
```

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



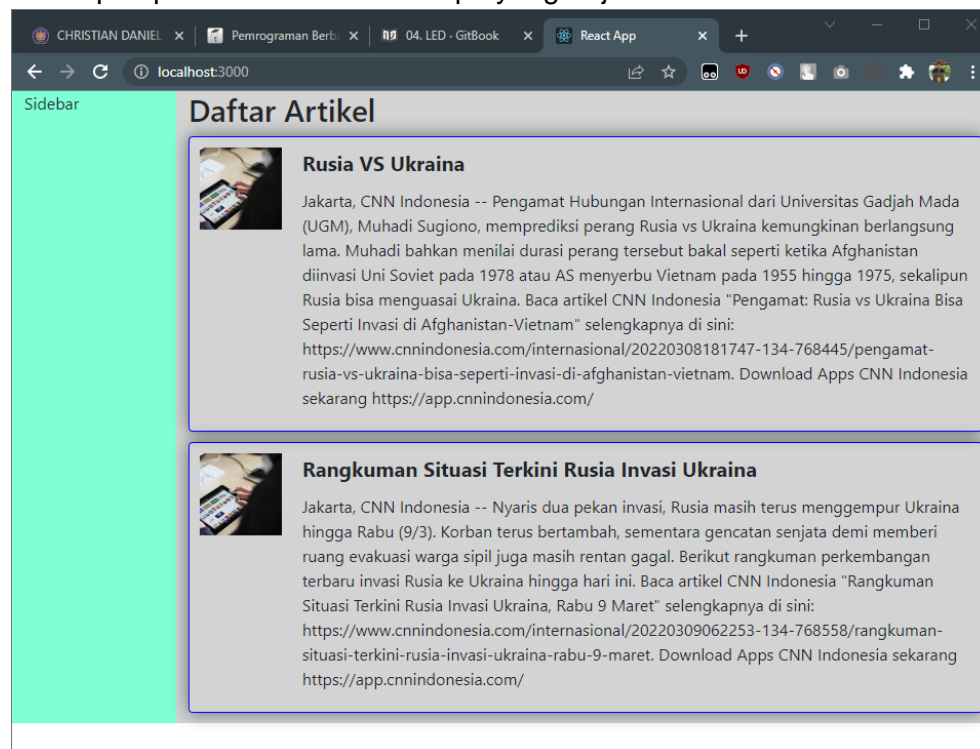
```
[
  {
    "userId": 1,
    "id": 1,
    "title": "sunt aut facere repellat provident occaecati excepturi optio reprehenderit",
    "body": "quia et suscipit\nsuscipit recusandae consequuntur expedita et cum\nreprehenderit molestiae ut ut quas totam\nnostrum rerum est autem sunt rem eveniet architecto"
  },
  {
    "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"
  }
]
```

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>

```
10   componentDidMount(){
11     fetch(`http://localhost:3001/posts`)
12     .then(response => response.json())
13     .then(jsonHasilAmbilDariAPI => {
14       this.setState({
15         listArtikel: jsonHasilAmbilDariAPI
16       })
17     })
18   }
```

7. Simpan perubahan dan amati apa yang terjadi.



## 2.2 Pertanyaan Praktikum 2

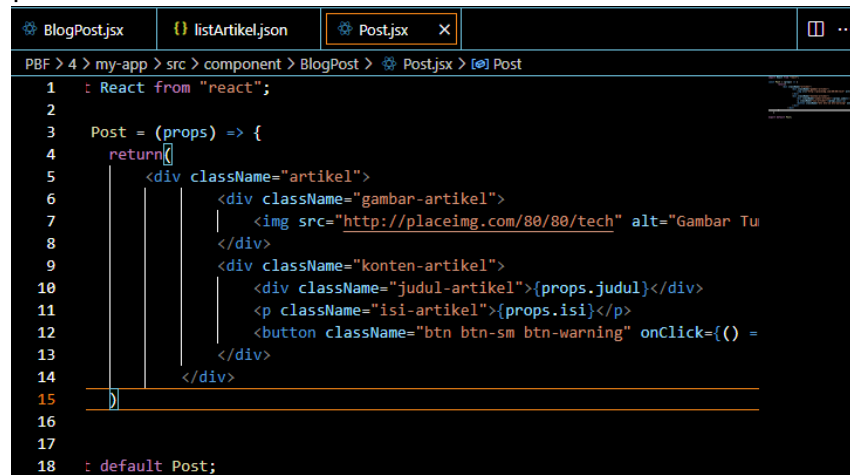
- Kenapa json-server dijalankan pada port 3001? Kenapa tidak sama-sama dijalankan pada port 3000 seperti project react yang sudah kita buat?
- Bagaimana jadinya kalau kita ganti port json-server menjadi 3000?

## Praktikum 3

### Interaksi dengan API menggunakan method DELETE

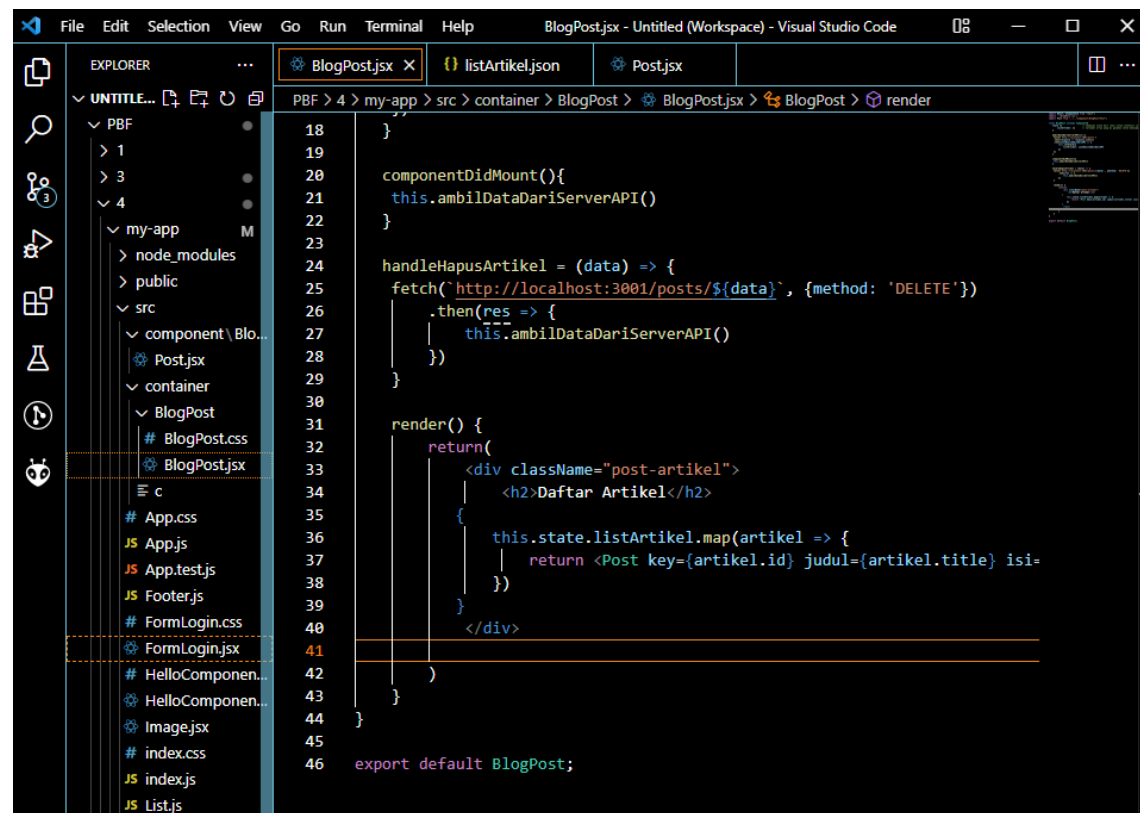
#### 3.1 Langkah Praktikum 3

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



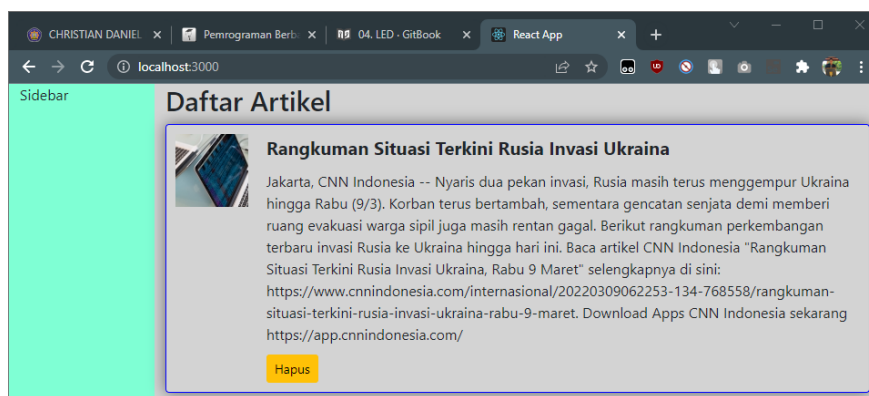
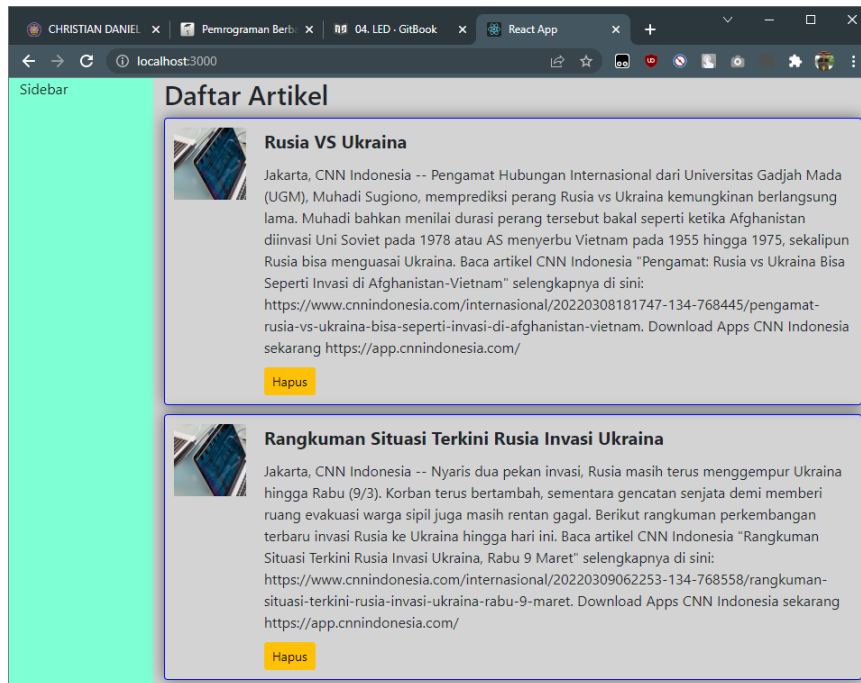
```
1  React from "react";
2
3  Post = (props) => {
4    return(
5      <div className="artikel">
6        <div className="gambar-artikel">
7          
9        <div className="konten-artikel">
10       <div className="judul-artikel">{props.judul}</div>
11       <p className="isi-artikel">{props.isi}</p>
12       <button className="btn btn-sm btn-warning" onClick={() =
13     </div>
14   )
15 }
16
17
18  default Post;
```

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



```
18  }
19
20  componentDidMount(){
21    this.ambilDataDariServerAPI()
22  }
23
24  handleHapusArtikel = (data) => {
25    fetch('http://localhost:3001/posts/${data}', {method: 'DELETE'})
26      .then(res => {
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 => {
37            return <Post key={artikel.id} judul={artikel.title} isi=
38          })
39        }
40      </div>
41    )
42  }
43
44  }
45
46  export default BlogPost;
```

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



### 3.2 Pertanyaan Praktikum 3

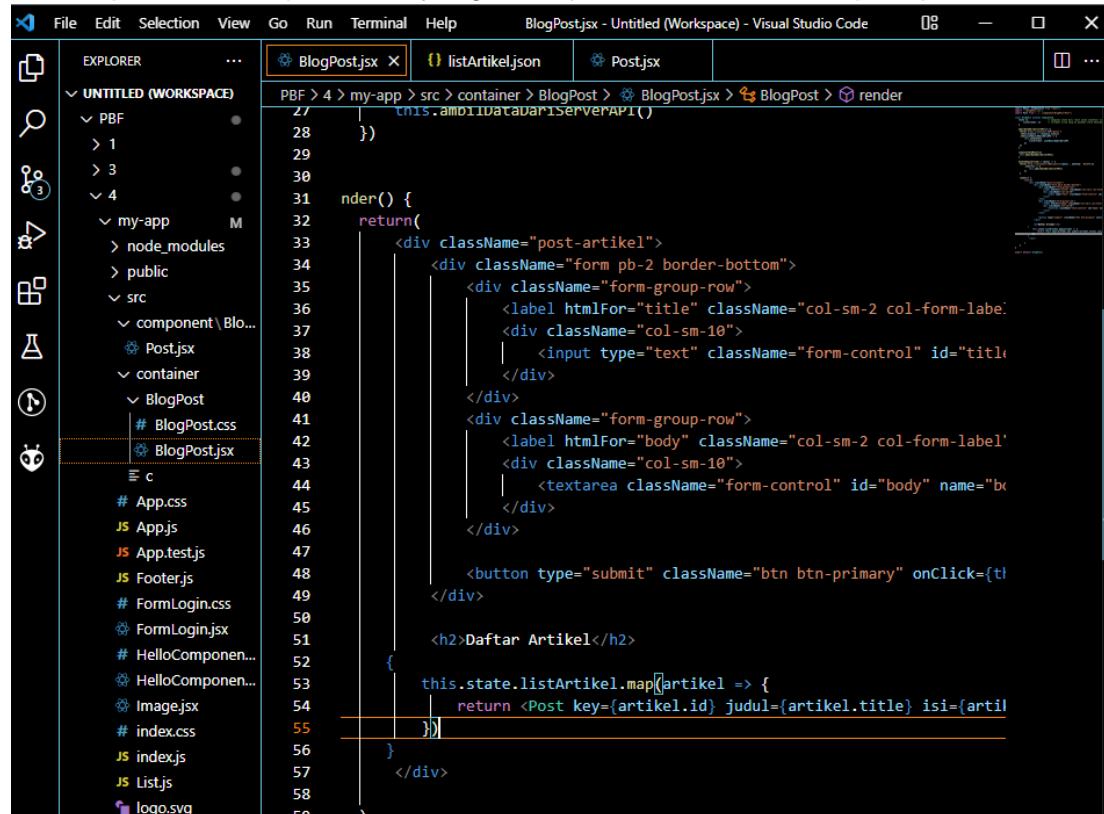
- a. Apa yang terjadi setelah kalian klik tombol hapus?
- b. Perhatikan file listArtikel.json, apa yang terjadi pada file tersebut? Kenapa demikian?
- c. Fungsi handleHapusArtikel itu untuk apa?
- d. Jelaskan perbedaan fungsi componentDidMount() pada Gambar 1.18 dengan fungsi componentDidMount() pada Gambar 3.2 ?

## Praktikum 4

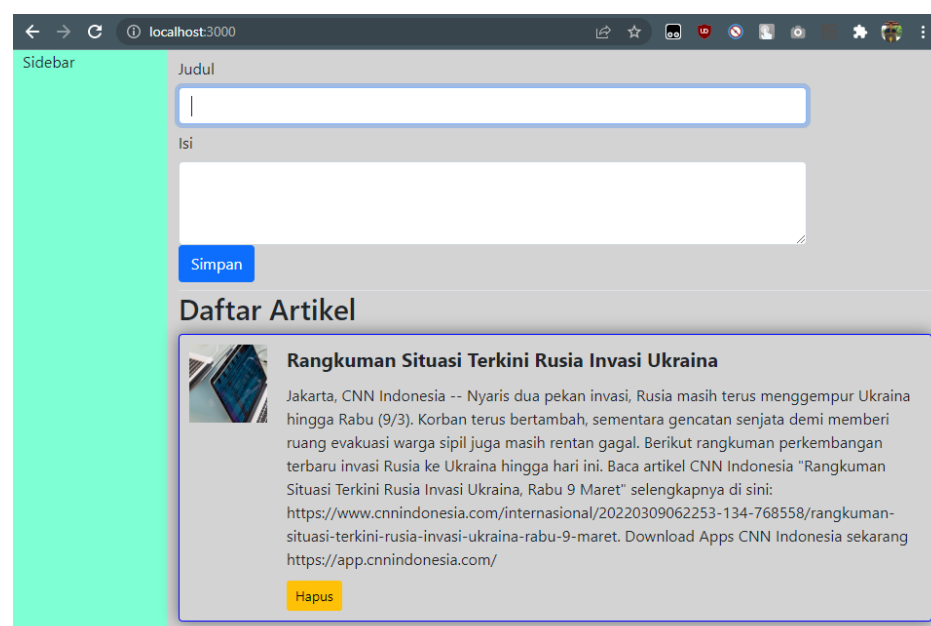
### Interaksi dengan API menggunakan method POST

#### 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



```
27 |   this.setState({articles: articles})
28 | }
29 |
30 |
31 | render() {
32 |   return(
33 |     <div className="post-artikel">
34 |       <div className="form pb-2 border-bottom">
35 |         <div className="form-group row">
36 |           <label htmlFor="title" className="col-sm-2 col-form-label">Judul</label>
37 |           <div className="col-sm-10">
38 |             <input type="text" className="form-control" id="title">
39 |           </div>
40 |         </div>
41 |         <div className="form-group row">
42 |           <label htmlFor="body" className="col-sm-2 col-form-label">Isi</label>
43 |           <div className="col-sm-10">
44 |             <textarea className="form-control" id="body" name="body">
45 |           </div>
46 |         </div>
47 |         <button type="submit" className="btn btn-primary" onClick={this.handleClick}>Simpan</button>
48 |       </div>
49 |       <h2>Daftar Artikel</h2>
50 |       <div>
51 |         {
52 |           this.state.listArtikel.map(artikel => {
53 |             return <Post key={artikel.id} judul={artikel.title} isi={artikel.body} />
54 |           })
55 |         }
56 |       </div>
57 |     </div>
58 |   )
59 | }
```

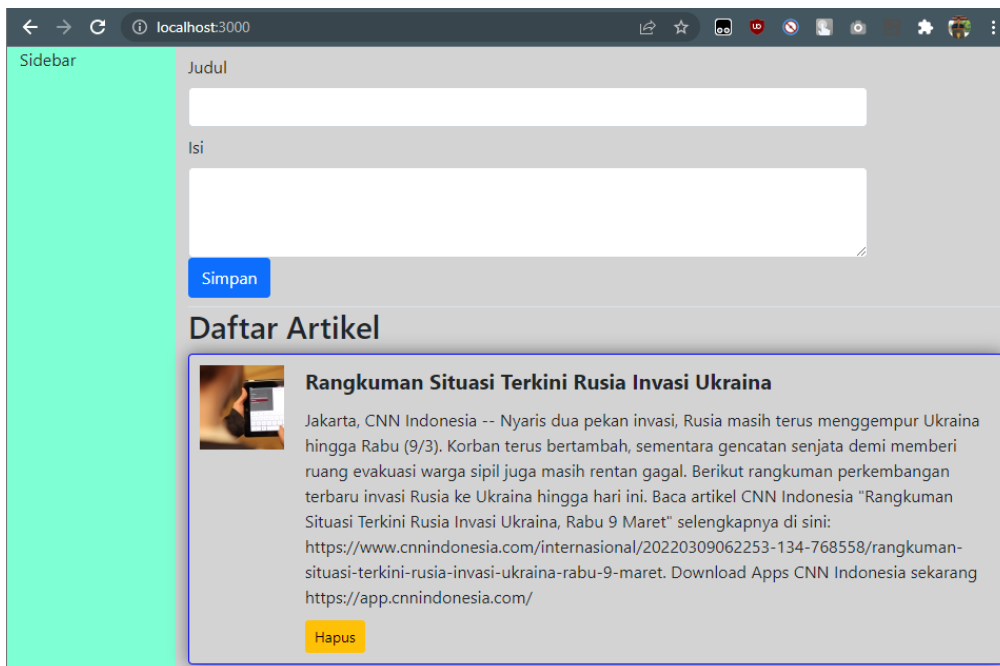




2. Kemudian modifikasi BlogPost untuk bagian state dan request API dari server, seperti

Gambar 4.2

```
6      state={                                // komponen state dari react untuk statefull c
7        listArtikel: [],
8        insertArtikel: {
9          userId: 1,
10         id: 1,
11         title: "",
12         body: ""
13       } // variabel array yang di gunakan untuk menyimpan data api
14     }
15
16     ambilDataDariServerAPI=()=>{
17       fetch('http://localhost:3001/posts?_sort=id&_order=desc')
18       .then(response => response.json())
19       .then(jsonHasilAmbilDariAPI => {
20         this.setState({
21           listArtikel: jsonHasilAmbilDariAPI
22         })
23       })
24     }
```



← → ↻ ⓘ localhost:3000


Sidebar

Judul

Isi

Simpan

### Daftar Artikel



#### Rangkuman Situasi Terkini Rusia Invasi Ukraina

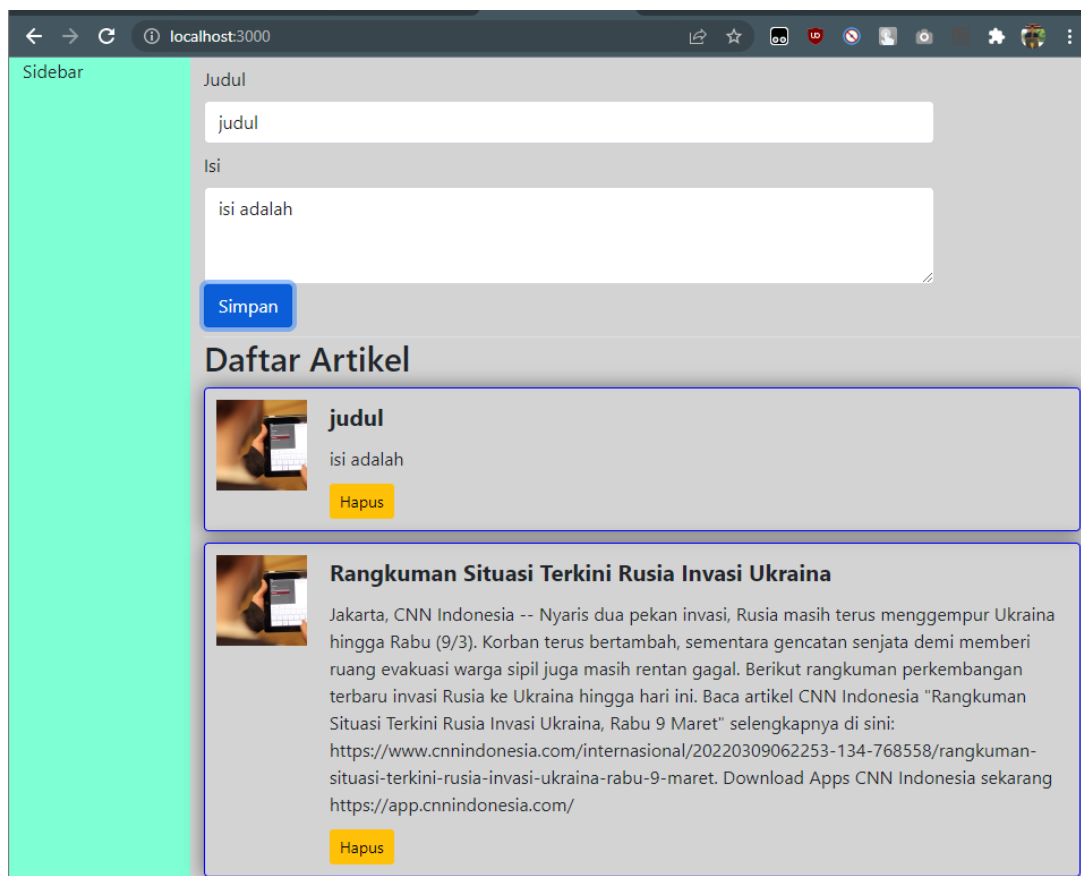
Jakarta, CNN Indonesia -- Nyaris dua pekan invasi, Rusia masih terus menggempur Ukraina hingga Rabu (9/3). Korban terus bertambah, sementara gencatan senjata demi memberi ruang evakuasi warga sipil juga masih rentan gagal. Berikut rangkuman perkembangan terbaru invasi Rusia ke Ukraina hingga hari ini. Baca artikel CNN Indonesia "Rangkuman Situasi Terkini Rusia Invasi Ukraina, Rabu 9 Maret" selengkapnya di sini: <https://www.cnnindonesia.com/internasional/20220309062253-134-768558/rangkuman-situasi-terkini-rusia-invasi-ukraina-rabu-9-maret>. Download Apps CNN Indonesia sekarang <https://app.cnnindonesia.com/>

Hapus

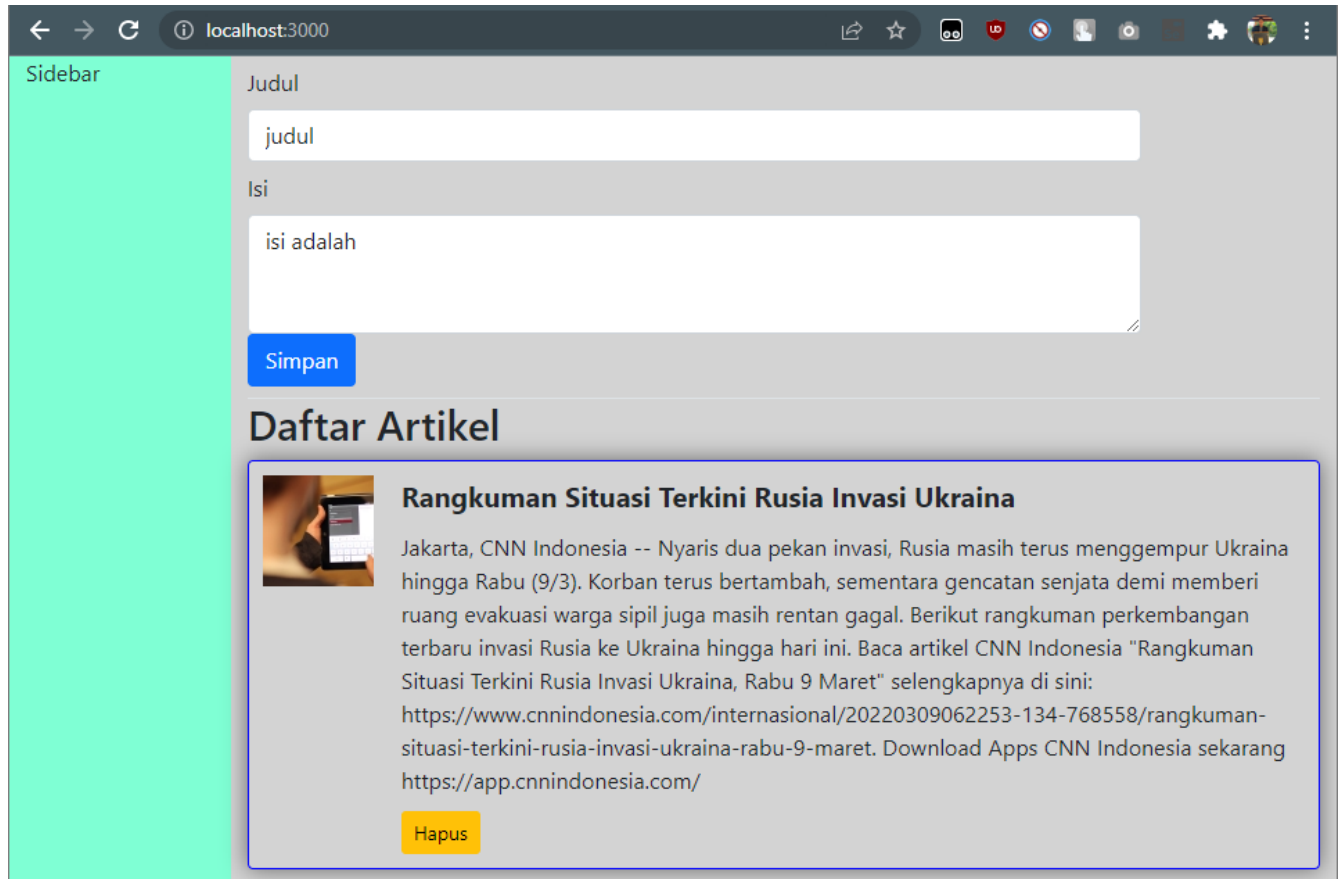
3. Tambahkan untuk handle form tambah data artikel seperti Gambar 4.3

```
30   handleTambahArtikel = (event) => {
31     let formInsertArtikel = { ...this.state.insertArtikel };
32     let timestamp = new Date().getTime();
33     formInsertArtikel['id'] = timestamp;
34     formInsertArtikel[event.target.name] = event.target.value;
35     this.setState({
36       insertArtikel: formInsertArtikel
37     })
38   }
39
40   handleTombolSimpan = () => {
41     fetch('http://localhost:3000/posts', {
42       method: 'POST',
43       headers: {
44         'Accept': 'application/json',
45         'Content-Type': 'application/json'
46       },
47       body: JSON.stringify(this.state.insertArtikel)
48     })
49     .then((response) => {
50       this.ambilDataDariServerAPI()
51     })
52   }
```

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



Jika sudah di hapus



#### 4.2 Pertanyaan Praktikum 4

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

<https://github.com/christiandaniel1505/Pemograman-Berbasis-Framework.git>