FAKULTAS ILMU KOMPUTER
UNIVERSITAS INDONESIA

Tutorial 10 – ReactJs 2
Arsitektur dan Pemrograman Aplikasi Perusahaan
Semester Ganjil 2018/2019

Melakukan Request ke Web Service pada ReactJs App

Versi: 1.0, 03 Desember 2018

Requirements

Sebelum memulai tutorial ini pastikan komputer Anda sudah terinstall **NodeJs**, dan **VSCode** atau IDE atau text editor lainnya sesuai preferensi Anda. Anda juga harus **mengunduh berkas ReactJs *app* yang telah disediakan di SCeLE** sebagai *base application* untuk tutorial kali ini. Setelah mengunduh, jalankan perintah "**npm i**" pada berkas tersebut.

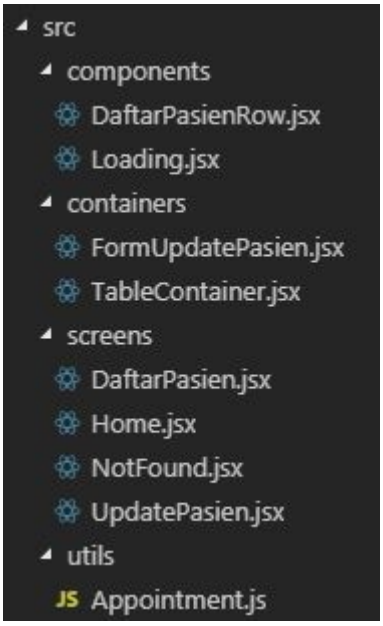
Request ke Web Service

Ide untuk memisahkan antara *back-end* dengan *front-end* suatu sistem telah Anda realisasikan dengan mempelajari ReactJs. Akan tetapi, dalam implementasinya Anda masih menggunakan data statik yang Anda *declare* pada *code* Reactjs. Untuk itu, pada tutorial kali ini Anda akan mempelajari bagaimana melakukan *request* ke web service.

Pada tutorial ini Anda akan menggunakan package whatwg-fetch untuk melakukan *request* ke web service. Selain itu, Anda juga akan menggunakan API SI-Appointment sebagai *sample* web service.

Struktur App

Pada ReactJs apps, idealnya suatu component hanya akan melakukan tugas yang spesifik, misalnya untuk berhubungan dengan *web service*, atau untuk melakukan render JSX.



Pada berkas template yang telah disediakan, terdapat struktur sebagai berikut:

- Pada folder "screens" akan berisi halaman apa saja yang akan ada pada website ini. Pada tutorial kali ini, *component* yang berada pada folder ini akan bertugas untuk melakukan komunikasi data dengan web service.
- Pada folder "containers" akan berisi *wrapper* untuk beberapa *components* lainnya.
- Pada folder "components" akan berisi *component* yang biasanya dapat di *reuse*.
- Pada folder "utils" akan berisi fungsi-fungsi global yang akan sering digunakan, seperti untuk *parsing* data, maupun untuk melakukan *request* ke web service.

Struktur tersebut tidaklah baku, terdapat banyak praktik dalam implementasi ReactJs, sehingga sangat mungkin anda akan menemukan struktur yang lain saat anda mengerjakan ReactJs app lainnya. Tetapi pada umumnya akan mengacu pada *pattern* "Composition", anda dapat mempelajari hal tersebut lebih lanjut disini.

Tutorial

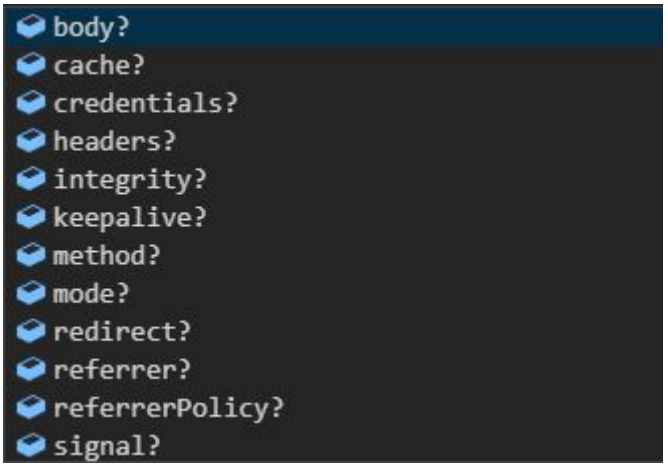
Untuk melakukan *request* ke SI-Appointment, pada Appointment.js terdapat *variable* Appointment yang berisi beberapa *function*, yakni **getAllPasien()**, **getDetailPasien(idPasien)**, dan **updateStatusPasien(requestBody)**. Anda akan mengimplementasikan masing-masing *function* tersebut untuk melakukan komunikasi data dengan SI-Appointment.

Melakukan Request getAllPasien

Untuk melakukan *request* getAllPasien, anda dapat melakukan implementasi pada fungsi getAllPasien() pada Appointment.js seperti berikut:

```
getAllPasien() {
    return fetch(`${cors}${baseUrl}/1/getAllPasien`, {
        method: 'GET',
    })
    .then(response => {
        return response.json()
    })
    .then(jsonResponse => {
        return jsonResponse
    })
},
```

Method fetch() digunakan dengan parameter pertama berisi **url** dengan *wrapper* **grave accent bukan apostrophe**, dan parameter kedua berisi **object** dengan informasi seperti *header*, *credentials*, *request method*, maupun beberapa parameter lainnya.

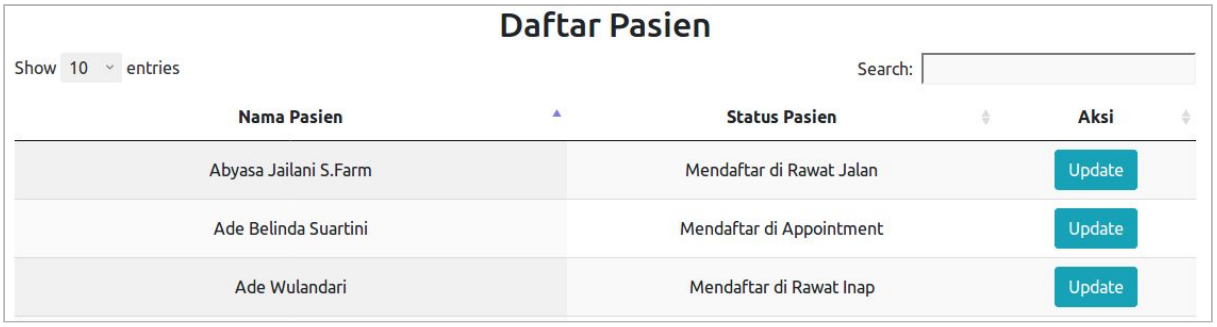


Setelah pemanggilan *method* fetch(), terdapat *syntax* then() yang biasanya digunakan untuk selanjutnya memanipulasi *response* dari *request* tersebut sebelum pada akhirnya dikembalikan sebagai *output* dari fungsi getAllPasien().

Selanjutnya Anda dapat melakukan pemanggilan *function* getAllPasien() pada DaftarPasien.jsx, sehingga Anda dapat menampilkan daftar pasien sesuai dengan data SI-Appointment.

```
Appointment.getAllPasien().then(response => {
    this.setState({
        loading: false,
        listPasien: response.result
    })
})
```

Tempatkan pemanggilan fungsi tersebut pada constructor() atau pada *lifecycle* componentDidMount(). Dengan demikian ketika anda mengakses http://localhost:3000/all-pasien akan menampilkan data seperti berikut:



Melakukan Request [getDetailPasien]

Pada tabel daftar pasien, terdapat *button* yang jika diklik maka akan berpindah *page* ke http://localhost:3000/update-pasien/<idPasien>. Pada *page* tersebut akan ditampilkan *form* untuk melakukan *update* pasien, sehingga Anda perlu melakukan pemanggilan fungsi getDetailPasien(idPasien) untuk ditampilkan data pasien yang dimaksud pada *form* tersebut.

Pada Appointment.js, tambahkan implementasi fungsi getDetailPasien(idPasien) seperti berikut:

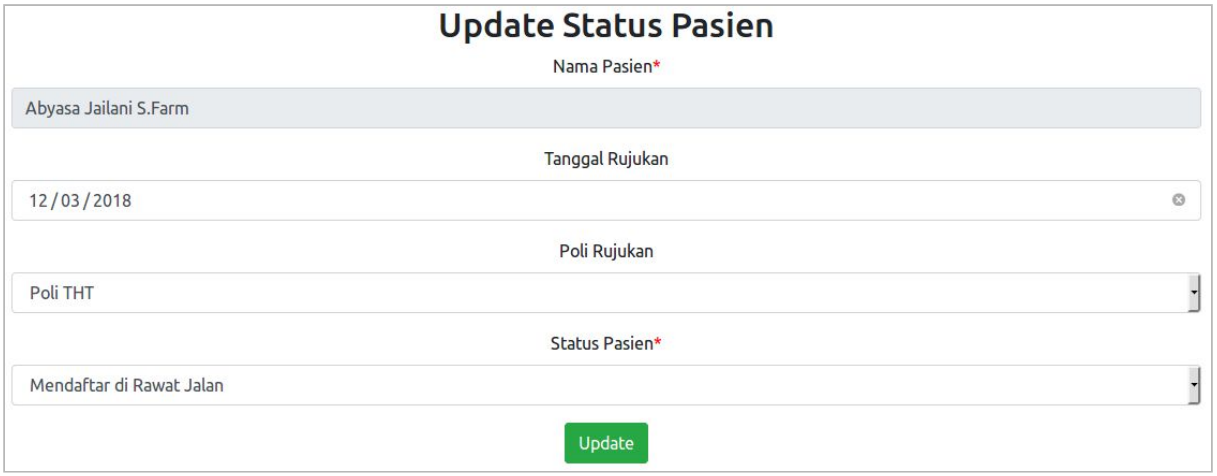
```
return fetch(`${cors}${baseUrl}/getPasien/${idPasien}`, {
    method: 'GET',
})
.then(response => {
    return response.json()
})
.then(jsonResponse => {
    return jsonResponse
})
```

Seperti yang telah dijelaskan sebelumnya, pengaksesan web service ini sangat mirip dengan getAllPasien(), namun dengan tambahan parameter idPasien yang digunakan untuk *url* akses.

Selanjutnya Anda dapat melakukan pemanggilan fungsi getDetailPasien(idPasien) pada UpdatePasien.jsx dengan cara seperti berikut:

```
Appointment.getDetailPasien(this.props.match.params.id).then(response => {
    if (response.status === 200) {
        this.setState({
            loading: false,
            pasien: response.result
        })
    } else {
        alert('Data tidak ditemukan')
        this.props.history.push('/all-pasien')
    }
})
```

Tempatkan pemanggilan fungsi tersebut pada constructor() atau pada *lifecycle* componentDidMount(). Dengan demikian ketika anda mengakses http://localhost:3000/update-pasien/3799 akan menampilkan data seperti berikut:



Melakukan Request [updateStatusPasien(requestBody)]

Untuk dapat melakukan update status pasien di SI-Appointment, Anda perlu melakukan *request* POST. *Request* tersebut dapat Anda lakukan dengan menambahkan implementasi fungsi updateStatusPasien(requestBody) pada Appointment.js seperti berikut:

```
return fetch(`${cors}${baseUrl}/1/updatePasien`, {
    method: 'POST',
    headers: {
        'Content-Type': 'application/json'
    },
    body: JSON.stringify(requestBody)
})
.then(response => {
    return response.json()
})
.then(jsonResponse => {
    return jsonResponse
})
```

Pada parameter kedua *method* fetch() terdapat tambahan informasi headers dan body. *Property* header diisi sesuai dengan dokumentasi SI-Appointment yang membutuhkan informasi 'Content-Type': 'application/json'. Sedangkan pada *property* body, diisi dengan data terbaru pasien setelah di update pada *form*.

Jika anda coba, *button* “Update” pada *page* update pasien belum fungsional. Maka dari itu anda perlu menambahkan implementasi pada fungsi handleFormSubmit(e) di UpdatePasien.jsx. Anda dapat menambahkan implementasi seperti berikut:

```
this.setState({
    loading: true
})

const data = new FormData(e.target)
const dataJson = {}

data.forEach((val, key) => {
    if (val !== "") {
        let name = key.split('.');
        if (name.length > 1) {
            let last = name.pop()
            name.reduce((prev, next) => {
                return prev[next] = prev[next] || {};
            }, dataJson)[last] = val
        } else {
            dataJson[key] = val
        }
    }
})

Appointment.updateStatusPasien(dataJson).then(response => {
    if (response.status === 200) {
        this.setState({
            loading: false,
            pasien: response.result
        })
        alert(`Sukses update pasien ${this.state.pasien.nama}`)
    } else {
        this.setState({
            loading: false
        })
        alert(`Gagal update pasien ${this.state.pasien.nama}`)
    }
})
```

Dengan demikian, jika Anda melakukan klik pada *button* “Update”, maka akan dilakukan request POST dan mengubah data-data pasien di SI-Appointment. Jika Anda masih belum yakin bahwa aksi tersebut berhasil, anda dapat melakukan pengecekan dengan mengakses langsung http://si-appointment.herokuapp.com/api/getPasien/<idPasien> sesuai dengan pasien yang anda update.

Pertanyaan

1. Pada setiap fungsi di Appointment.js, parameter pertama pasti memiliki *prefix* “${cors}” yang merefer pada *variable* cors. Mengapa hal tersebut perlu dilakukan? Apa yang terjadi jika *prefix* tersebut dihilangkan?
2. Pada fungsi handleFormSubmit(e) di UpdatePasien.jsx terdapat klausa:

   ```
   let last = name.pop()
   name.reduce((prev, next) => {
       return prev[next] = prev[next] || {};
   }, dataJson)[last] = val
   ```

   Apa yang dilakukan pada code tersebut? Jika klausa tersebut diganti dengan “dataJson[key] = val”, apa yang terjadi dengan isi *variable* dataJson?

   ***ganti seluruh code pada screenshot**

Latihan

1. Tambahkan menu pada *navbar* untuk menampilkan daftar seluruh staf Farmasi, dan implementasikan *page*nya hingga menampilkan data seperti pada *page* daftar pasien.
2. Pada daftar pasien, tambahkan *button* untuk menambahkan hasil lab pasien, yang jika diklik maka akan menampilkan *form* untuk menginput hasil lab pasien. Implementasikan *form* hingga berhasil melakukan request POST ke SI-Appointment.

Deliverables

Deliverables untuk tutorial kali ini adalah:

1. **Berkas Project**
   Buat sebuah project baru pada *organization* github.com/Apap-2018 dengan format nama tutorial10_NPM, contoh tutorial10_1601234567. Push project Anda ke repository GitHub tersebut. **Jangan salah push ke repository tutorial sebelumnya**.
2. **File Write-up (Khusus Kelas C)**
   Buat file *write-up* dengan format .txt atau .pdf yang berisi **jawaban dari pertanyaan**, dan **cara Anda mengimplementasikan soal latihan**. *Push* file tersebut pada *repository* pada deliverable [1].

Deadline

05 Desember 2018, 23:59:59

Penalti

Penalti:

- Keterlambatan
  Penalti keterlambatan sebesar -10 poin akan ditambahkan setiap 1 jam keterlambatan