BAB III

TINJAUAN PUSTAKA

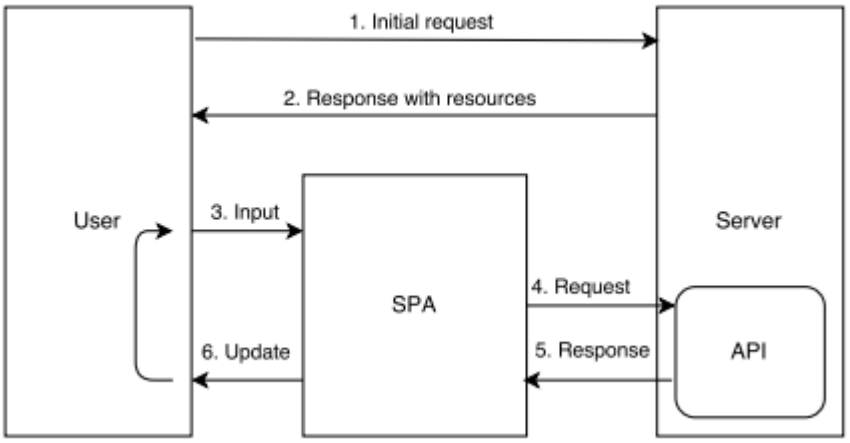
3.1 *Single Page Application*

Perkembangan dunia teknologi saat ini tidak akan pernah terlepas dari penggunaan website. Website atau aplikasi web telah menjadi wadah bagi seluruh orang di seluruh dunia untuk melakukan aktivitas secara daring dengan basis visual dan audio, yaitu tulisan, foto, hingga video. Ketika berbicara tentang aplikasi web, tentunya tidak dapat terlepas dari HTML (*HyperText Markup Language*) yang telah menjadi standar utama dalam pembuatan aplikasi web yang tiap harinya dapat kita akses melalui *web browser*.

HTML yang akrab kita lihat bahwa penggunaanya selalu didampingi dengan penggunaan CSS (*Cascading Style Sheets*) untuk memberikan tampilan yang ramah dan nyaman untuk dilihat oleh user ternyata terus berkembang pesat hingga sekarang. Salah satu perkembangannya adalah dengan kehadirannya *Single Page Application* atau dapat disebut aplikasi dengan satu halaman. [1] *Single Page Application* (SPA) adalah aplikasi web yang direpresentasikan melalui satu halaman HTML saja dimana memberikan performa lebih responsif yang hampir menyerupai aplikasi *desktop* ataupun aplikasi *native*. SPA menerima seluruh kode aplikasi HTML, CSS, ataupun Javascript pada *initial load* ataupun menerima secara dinamis untuk memberikan respon kepada user tanpa mengharuskan halaman diperbaharui terlebih dahulu.

[2] Saat ini, SPA telah menjadi salah satu metode terbaik dalam pembuatan aplikasi web dan telah digunakan oleh banyak orang, mulai dari aplikasi web personal, korporat, hingga *e-commerce*. Tidak sedikit dari beberapa perusahaan telah mengganti arsitektur website mereka dengan SPA dan meninggalkan aplikasi web konvensional yang berbasis pada banyak halaman web dan saling terhubung. Sebagai contoh penggunaan SPA dalam *e-commerce* yang sangat menguntungkan karena dapat memotong banyak waktu untuk *reload* halaman sehingga interupsi yang ada dapat diminimalisir untuk mendongkrak *shopping experience* yang ada.

Dalam implementasinya, SPA didukung dengan API (*Application Programming Interface*) untuk berkomunikasi dengan server. Setiap *request* maupun *response* yang dikirim atau dari server akan dikelola oleh SPA untuk menciptakan interaksi yang cepat (tanpa reload) dan dinamis sehingga user yang menggunakan aplikasi tersebut akan terasa nyaman (baik dalam segi *user experience*). Untuk mengimplementasikan ini, setiap API akan ditangani oleh AJAX yang mengirim dan menerima data secara *asynchronously* sehingga tidak mengganggu tampilan untuk user.



Gambar 3.1 Komunikasi antara user dengan server melalui SPA

SPA telah memiliki beberapa *framework* populer yang sering kali digunakan untuk membangun aplikasi web besar dan kompleks, seperti React, Angular JS, dan Angular 2.

3.2 React JS

React sendiri adalah *framework* SPA besutan Meta (ex. Facebook Inc.) yang di rilis pada tahun 2013. *Framework* ini menggunakan *Virtual DOM* dimana *DOM* baru dapat dibuat dengan Javascript yang membuat model pemrograman menjadi lebih sederhana. *Data bindings* pada React menggunakan *diff algorithm* yang dapat membuat aplikasi melakukan *full re-render* ketika ada sesuatu yang berubah. Ketika ada perubahan pada Virtual DOM maka React akan melakukan perbandingan terhadap DOM pada setiap level dan melakukan *re-render*.

Ketika kita berbicara tempat penyimpanan sementara maka kita dapat menggunakan *state* dimana setiap kali *state* berubah maka React akan melakukan *re-render* untuk memperbaharui data yang ada di dalam *state* tanpa harus melakukan pembaharuan halaman terlebih dahulu. Pada React sendiri terdapat beberapa konsep pemahaman yang sering digunakan, seperti *class, state, props, context, ref,* dan lainnya.

Namun secara keseluruhan komponen menjadi bagian terpenting dalam penggunaan React karena aplikasi React sendiri tersusun dari berbagai komponen yang menjadi satu sehingga dapat digunakan. Dengan kehadirannya komponen ini juga menjadikan kode React menjadi *reusable* atau setiap komponen dapat digunakan kembali apabila membutuhkan fungsi yang sama yang ada pada komponen tersebut.

Pada *update* React 16.8, React secara resmi menambahkan *Hooks* yakni fitur yang memungkinkan kita menggunakan *state* tanpa harus membuat *class* melainkan cukup membuat *function* saja. Penambahan *Hooks* pada React didasari pada beberapa masalah yang mungkin selama ini terjadi pada React tanpa *Hooks*, yaitu:

1. Sulitnya menggunakan kembali logika *stateful* antar komponen dimana kita harus melakukan *restructure* dan memahami *pattern* mengenai *render props* dan *higher-order components* untuk menggunakan suatu *state* antar komponen.
2. Komponen yang kompleks membuat sulit untuk dipahami. Terkadang komponen yang kompleks akan sulit untuk me-*manage stateful logic* dan *side effect*. Hal ini memicu timbulnya *bugs* dan hasil yang tidak konsisten.
3. Penggunaan *classes* dapat membingungkan bagi orang maupun mesin. *Classes* menghambat dalam pembelajaran React karena pengguna harus memahami kapan harus menggunakan *classes* maupun *function* yang justru dapat menimbulkan kebingungan.

Dengan adanya kehadiran *Hooks,* penggunaan React menjadi lebih mudah dan sederhana. Lewat *Hooks* kita bisa menggunakan beberapa fungsi yang sangat

membantu dan jauh lebih sederhana dari sebelumnya, seperti *useState* dan *useEffect*.

[3] Dari segi performa berdasarkan suatu penelitian yang dilakukan oleh Eric Molin dari *KTH Royal Institute of Technology* dengan membandingkan React, Angular JS, dan Angular 2 diperoleh hasil rata-rata yang baik untuk React mulai dari pengujian performa *loading time, editing time*, hingga *memory allocation* untuk 10-5000 *items*. React selalu menempati posisi rata-rata atau kedua jika dibandingkan dengan Angular JS maupun Angular 2 dimana penggunaan Angular sendiri masih tergolong sulit dan kompleks.

3.3 Redux

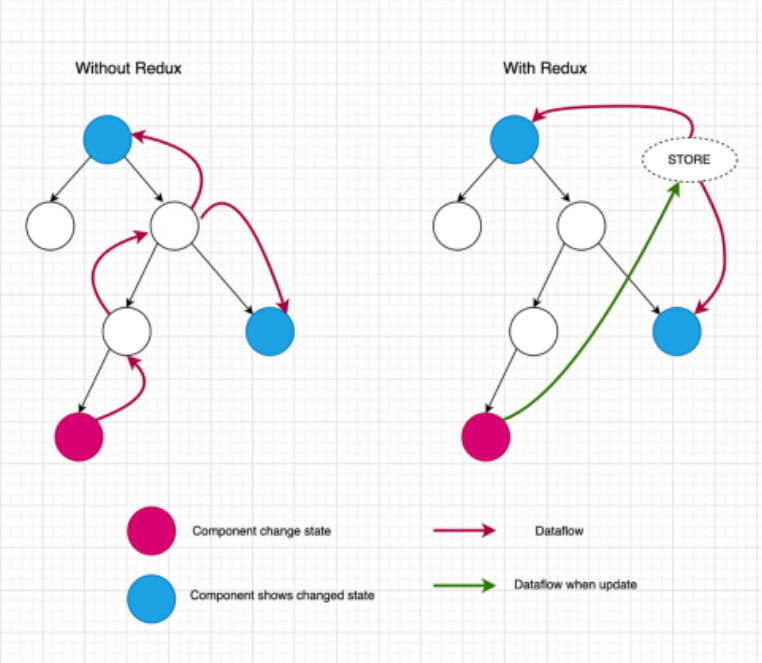
Penggunaan React untuk membangun sebuah aplikasi web yang besar dan cukup kompleks akan memerlukan sebuah *library* tambahan untuk menunjang performa dan mengurangi kompleksitas. Salah satu hal yang perlu diperhatikan adalah *state management* dimana apabila kita memerlukan *state* yang bersifat global agar dapat dipanggil pada beberapa komponen secara bebas maka kita perlu menggunakan Redux. Redux adalah *global state management* dimana *state* akan disimpan dalam suatu penampungan global tanpa terikat dengan sebuah komponen yang ada.

[4] Redux adalah *open-source library* yang dibuat oleh Dan Abramov and Andrew Clark pada tahun 2015. *Library* ini ditujukan untuk memudahkan manipulasi data dan *data-flow*. Terdapat beberapa keunggulan dalam penggunaan Redux, yaitu Redux memiliki *Store* yang menjadi wadah untuk menyimpan *state* secara global, *data outcome* lebih terprediksi sehingga memudahkan *developer* untuk melakukan pengaturan dan pemeliharaan kode, dan dengan menggunakan *developer tools* kita dapat melakukan monitor *state* secara *real-time*.

Konsep dasar dari Redux sendiri tetap mempertahankan *one-way data bindings* dari React dimana setiap *data flow* selalu dalam satu arah. Terdapat beberapa bagian dari Redux, yaitu:

1. *Store* adalah tempat penyimpanan global dari seluruh *state* yang ada dan akan melakukan *subscribe* ke *view* yakni komponen dari React.

2. *Actions* adalah proses yang nantinya akan dieksekusi terhadap *state* dengan *view* melakukan *dispatch actions* dan mengirimkan informasi *content type* serta *payload*.
3. *Reducer* adalah layaknya manufaktur yang menerima kondisi *state* saat ini serta melakukan return *state* yang sudah diperbaharui.



Gambar 3.2 Perbandingan *state management* dengan dan tanpa Redux

3.4 Axios

Saat ini penggunaan *Application Programming Interface* (API) sering kali digunakan dalam pembuatan aplikasi baik itu *mobile* maupun aplikasi web. API sendiri memberikan kemudahan dalam menjembatani *frontend* dengan *backend* yang kemudian terhubung dengan *database*. Terdapat metode bawaan untuk melakukan panggilan kepada API, yakni menggunakan Fetch. Namun, Fetch sendiri memiliki beberapa kekurangan yang membuat proses menangani *request* dan *response* dari API kurang maksimal. Maka dari itu muncullah *library* Axios yang saat ini penggunaannya sudah mendominasi menggantikan Fetch.

[5] Axios memiliki beberapa kelebihan, yaitu dapat melakukan *error handling* yang lebih baik dibandingkan Fetch karena Axios memiliki fitur *canceling request* sehingga kode yang diperlukan jauh lebih sederhana

dibandingkan Fetch yang harus menggunakan .then() beberapa kali. Selain itu, *response* dari API call yang dilakukan oleh Axios sudah dalam format JSON sehingga tidak perlu lagi melakukan *formatting* pada *response* yang diterima.

Dalam penanganan *request* dan *response*, Axios juga memiliki fitur yang bernama *interceptors*. Fitur ini digunakan untuk melakukan interupsi terhadap *request* maupun *response* yang dipanggil sehingga di dalam interupsi tersebut dapat disisipkan beberapa fungsi atau proses terlebih dahulu sebelum nantinya data dikirim maupun diterima.

[6] Axios *request interceptors* dapat digunakan untuk menyisipkan *access token* pada *authorization header* dari sebuah *request*. Hal ini bertujuan untuk memberikan fitur *authorization* dari setiap *request* sehingga API tidak dapat di *hit* oleh pihak yang tidak memiliki akses tertentu. Sedangkan, Axios *response interceptors* dapat digunakan untuk mengatur fitur *refresh token* agar dapat memperpanjang *session* dari sebuah *access token*.

3.5 Docker

Perkembangan teknologi saat ini tentunya telah mendorong munculnya berbagai jenis program dan aplikasi. Hal ini membuat proses distribusi sebuah aplikasi menjadi suatu hal yang perlu diperhatikan. Maka dari itu, munculah beberapa teknologi yang dapat memudahkan proses pendistribusian aplikasi, salah satunya *Docker*.

[7] *Docker* mengemas *file* yang ada dan yang diperlukan untuk menjalankan suatu aplikasi dalam sebuah wadah yang disebut kontainer. Kontainer ini nanti nya akan berjalan dalam satu OS yang sama yang terdapat pada *Docker Engine*. Penggunaan *Docker* ini menjadi lebih diminati dibandingkan *deployment* dengan *Virtual Machine*. Disamping itu perbedaan yang paling terasa adalah kemudahan yang dimiliki oleh *Docker* dalam melakukan proses *scaling* sebuah aplikasi. [8] *Docker* diinstal pada *server* kemudian *Docker* ini akan mengatur proses pembuatan, memulai, hingga menghentikan setiap kontainer yang ada.