ANALISIS PERFORMANSI *SERVICE WORKER* DENGAN *LIBRARY REACT JS* STUDI KASUS : WEB PEMELIHARAAN MESIN PADA INDUSTRI KECIL MENENGAH (IKM)

TUGAS AKHIR

Disusun dalam rangka memenuhi salah satu persyaratan

Untuk menyelesaikan program Strata-1 Departemen Teknik Informatika

Fakultas Teknik Universitas Hasanuddin

Makassar

Disusun Oleh :

AL RIEFQY DASMITO

D421 14 502

DEPARTEMEN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS HASANUDDIN

MAKASSAR

2019


PDF
Optimization Software:
www.balesio.com

ABSTRAK

Teknologi *web* saat ini berkembang cukup pesat dengan hadirnya *web* modern. *Service worker* merupakan salah satu API (*Application Programming Interface*) *Javascript* yang memungkinkan pengembang dalam melakukan pemrograman *cache* dan melakukan *load asset data*, melakukan manajemen *push notification*, dan yang lainnya. Dengan kemampuan *service worker* menjadikan sebuah *website* dapat difungsikan walaupun koneksi jaringan yang lemah ataupun tidak ada koneksi internet sekalipun (*offline). Service worker* mampu mengatasi masalah *website* yang tidak harus selalu bergantung pada koneksi jaringan. Selain itu *service worker* mampu membuat waktu akses *web* lebih cepat karena fungsi *fetch event* yang berperan untuk mengakses *cache* terlebih dahulu kemudian melakukan *request* jaringan. Dengan memadukan teknologi *service worker* dan *React JS* sebuah sistem dapat dibuat dengan menggunakan pendekatan *offline first*. Sebagai studi kasus teknologi ini akan di manfaatkan untuk menunjang proses bisnis dari IKM (Industri Kecil Menengah) yang rata-rata berada pada sektor pedesaan yang minim terhadap jaringan internet. Proses analisis penelitian ini, akan membandingkan aplikasi *web* yang terintegrasi *service worker* menggunakan *javascript* dan *web*site konvensional dengan beberapa skenario pengujian. Analisis yang dilakukan menggunakan indikator waktu rata – rata akses pada jaringan 3G maupun 4G, proses *browser*, *throughput*, dan penggunaan CPU. Hasil penelitian menunjukkan bahwa *web* yang terintegrasi *service worker* memberikan nilai waktu proses *browser* yang lebih cepat dibandingkan dengan *web* tanpa *service worker* dan *web* konvensional. Untuk hasil pengujian *throughput web service worker* juga memberikan nilai hasil yang lebih besar dibandingkan dengan *web* tanpa *service worker* dan *web* konvensional. Dengan demikian, diharapkan *web service worker* dapat diimplementasikan bukan hanya pada IKM, tetapi juga pada berbagai sistem *web* yang memerlukan kehandalan jaringan yang lemah.

Kata Kunci: *web,service worker;offline;*ReactJS*;*

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh.

Segala puji dan syukur kami panjatkan ke hadirat Allah S.W.T Tuhan Yang Maha Esa yang dengan limpahan rahmat dan hidayah-Nya sehingga tugas akhir dengan judul "*Analisis Performansi Service worker dengan Library React JS, Studi Kasus : Web Pemeliharaan Mesin pada Industri Kecil Menengah (IKM)"* ini dapat diselesaikan sebagai salah satu syarat dalam menyelesaikan jenjang Strata-1 pada Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin.

Dalam penyusunan penelitian ini disajikan hasil penelitian terkait judul yang telah diangkat dan telah melalui proses pencarian dari berbagai sumber baik jurnal penelitian, prosiding pada seminar-seminar nasional/internasional, buku maupun dari situs-situs di internet.

Penulis menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, mulai dari masa perkuliahan sampai dengan masa penyusunan tugas akhir, sangatlah sulit untuk menyelesaikan tugas akhir ini. Oleh karena itu, pada kesempatan ini penulis menyampaikan ucapan terima kasih sedalam-dalamnya kepada:

1) Tuhan Yang Maha Esa atas semua berkat, karunia serta pertolongan-Nya yang tiada batas, yang telah diberikan kepada penulis disetiap langkah dalam pembuatan program hingga penulisan laporan skripsi ini.
2) Kedua orang tua penulis serta kedua saudara penulis, serta keluarga yang iasa memberikan kekuatan, inspirasi, motivasi, bimbingan moral, materi, ayaan dan kasih sayang yang tidak terbatas kepada penulis.

3) Bapak Dr.Eng Zulkifli Tahir, S.T.,M.Sc., selaku pembimbing 1 yang telah banyak memberi bimbingan, inspirasi, motivasi, dan masukan yang bermanfaat kepada penulis.
4) Bapak A.Ais Prayogi Alimuddin, S.T., M.Eng., selaku pembimbing II yang telah banyak memberi keyakinan, perhatian, bimbingan, motivasi, dan masukan yang bermanfaat kepada penulis.
5) Bapak Dr. Ir Zahir Zainuddin, M.Sc. dan Bapak Adnan, S.T., M.T., Ph.D., selaku dosen penguji yang telah memberikan saran sehingga laporan skripsi ini menjadi lebih baik.
6) Bapak Dr. Amil Ahmad Ilham, S.T., M.IT., selaku Ketua Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin atas bantuan dan bimbingannya selama masa perkuliahan penulis.
7) Bapak Dr.Eng. Muhammad Niswar, S.T., M.IT., selaku kepala LAB CCIE yang telah memberikan bimbingan selama masa pengerjaan tugas akhir penulis.
8) Bapak Robert dan Bapak Zainuddin serta segenap staf Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin yang telah membantu kelancaran penyelesaian tugas akhir penulis.
9) Segenap keluarga LAB CCIE Universitas Hasanuddin yang telah memberikan begitu banyak bantuan selama penelitian, pengambilan data dan diskusi *progress* penyusunan tugas akhir serta memberikan semangat di masa-masa sulit.
10) Para sahabat dekat penulis terutama Tiwi Nur Safitri,S.T, A. Khairil Fajri, Muh Ilii A. Suyuti, Gian Aron Angelo, Aryandi, S.T., Inka G. Mallisa, Fitriani Winda Astiyanti, Armiani Putri, Rizka Irianti, David Reinhart, Zulfahmi,

Fachrial yang telah memberikan doa, bantuan dan dukungan sejak masa awal perkuliahan.

11) Teman-teman Adyaksa Yakip, Muh Nur Alamsyah, Hermawan Safrin, Syarif Hidayatullah, Abdillah Satari Rahim, Rahmat Firman, Fadel Pratama, Muhammad Shadiq, Ahmad Setiadi.
12) Seluruh teman-teman RECTIFIER'14 atas semua bantuan dan semangat yang diberikan selama ini.
13) Serta seluruh pihak yang tidak sempat disebutkan satu persatu yang telah banyak meluangkan tenaga, waktu, dan pikiran selama penyusunan laporan tugas akhir ini.

Akhir kata, penulis berharap semoga Tuhan Yang Maha Esa berkenan membalas segala kebaikan dari semua pihak yang telah banyak membantu. Semoga tugas akhir ini dapat memberikan manfaat bagi pengembangan ilmu selanjutnya. Amin.

Wassalam

Gowa, 1 Maret 2019

Penulis

DAFTAR ISI

DAFTAR GAMBAR

DAFTAR TABEL

BAB I

PENDAHULUAN

1.1. Latar Belakang

Service worker merupakan salah satu APIs *Java script* yang memungkinkan pengembang dalam melakukan pemrograman *cache* dan melakukan *load asset data*, melakukan managemen *push notification*, dan yang lainnya. *Service worker* adalah sebuah *script* yang bekerja pada *browser background* tanpa interaksi dari pengguna. *Service worker* memiliki kehandalan dalam hal koneksi internet yang tidak stabil ataupun lambat sehingga membuat sistem yang ada tidak terlalu bergantung pada koneksi internet yang cepat. Selain itu disatu sisi *service worker* disebut sebagai *performance booster* karena menghemat jaringan dan menyediakan *user experience* yang lebih baik. *Service worker* diimplementasikan pada *Java script* yang berjalan pada *background website* utama dan berjalan secara paralel dengan *website* utama sehingga kerja dari *service worker* tidak terlalu terlihat secara langsung. *Service worker* pada dasarnya adalah sebuah *script* atau *Java script* file yang berjalan dibelakang *website* utama dan membantu dalam pengembangan aplikasi *web offline*.

Dengan kemampuan *service worker* menjadikan sebuah *website* dapat difungsikan walaupun koneksi jaringan yang lemah ataupun tidak ada koneksi internet sekalipun mampu mengatasi masalah *website* yang tidak harus selalu bergantung pada koneksi jaringan. Hal tersebut dapat dilakukan dengan kan pengembangan teknologi *offline first web application. Offline first* buah pendekatan dalam pengembangan perangkat lunak dimana *developer*

membangun fitur atau aplikasi yang ada akan bekerja dengan atau tanpa koneksi internet. Dengan menggunakan pendekatan ini data yang ada akan di simpan secara lokal dan secara berkala akan melakukan sinkronisasi ketika koneksi internet tersedia.

Telah disebutkan sebelumnya bahwa *service worker* adalah API *Java script* sehingga *service worker* didukung oleh berbagai *library Java script* dan *library* yang ada. Contohnya adalah *React JS* yang mendukung *service worker* dalam melakukan pengembangan aplikasi *website* dengan pendekatan *offline first*.

React JS adalah sebuah *Java script library* untuk membangun *user interface* yang didesain oleh *facebook* untuk membuat sebuah *website* yang cepat namun minim kode. *React JS* hanya menangani semua hal yang berkaitan dengan tampilan. Telah disebutkan bahwa *React JS* cepat dan efisien, hal ini disebabkan karena *React JS* menciptakan *Virtual DOM (Document Object Model)* dimana setiap perubahan yang terjadi pada setiap laman *website* hanya akan berubah pada DOM yang hanya mengalami perubahan saja sehingga tidak mengubah seluruh halaman *website*.

Dengan memadukan teknologi *service worker* dan *React JS* sebuah sistem dapat dibuat dengan menggunakan pendekatan *offline first* dan memiliki kemampuan yang cepat. Teknologi ini dapat dimanfaatkan untuk menunjang proes bisnis dari IKM yang ada, yang rata-rata berada pada sektor pedesaan yang minim terhadap jaringan internet. Dengan semakin baiknya proses pada industri kecil menengah di desa – desa yang ada di Indonesia nantinya akan mendorong an tenaga kerja secara merata dan meluas disegala sektor ekonomi yang

ada. IKM dituntut untuk mengikuti perkembangan teknologi yang ada agar dapat meningkatkan kinerja dan mampu bertahan dalam persaingan ekonomi terkini.

Sebuah sistem yang berbasis *online* adalah satu dari banyak cara yang dapat digunakan IKM untuk bertahan. Namun dalam menerapakan sebuah sistem berbasis *online* pada IKM yang mayoritas berada pada pedesaan memiliki kesulitan dalam meraih koneksi jaringan yang handal, sehingga dibutuhkan sebuah sistem yang mampu bekerja dengan kondisi jaringan yang lemah atau tidak ada sama sekali sehingga *service worker* merupakan salah satu teknologi yang dapat mengatasi masalah koneksi jaringan yang ada.

Berdasarkan uraian diatas penulis kemudian mengangkat sebuah penelitian dengan judul "**Analisis performansi *service worker* dengan *library React JS*. Studi kasus : *web* pemeliharaan mesin pada Industri Kecil Menengah (IKM)"** .

1.2. Rumusan Masalah

1. Bagaimana meningkatkan performansi *web* IKM sektor pedesaan yang mempunyai kemampuan internet terbatas?
2. Bagaimana membuat *website* pemeliharaan mesin industri kecil menengah (IKM) dengan menggunakan teknologi *service worker*?
3. Bagaimana mengintegrasikan *service worker* dengan teknologi *React JS*?

1.3. Tujuan Penelitian

1. Mengetahui cara mengatasi performa *website* dengan kondisi jaringan dan dwith yang lemah.

2. Mengetahui cara membuat *website* pemeliharaan mesin pada industri kecil menengah (IKM) dengan menggunakan teknologi *service worker*
3. Mengetahui bagaimana menggunakan teknologi *service worker* dengan teknologi *React JS library* dalam pembuatan *website*.

1.4. Manfaat Penelitian

1. Dapat mengetahui cara mengatasi performa *website* dalam kondisi jaringan yang lemah
2. Dapat membuat sebuah *website* pemeliharaan mesin industri kecil menengah (IKM) dengan menggunakan teknologi *service worker*.
3. Dapat mengetahui cara integrasi *service worker* dan teknologi React JS.

1.5. Batasan Masalah

1. Pembuatan *website* menggunakan *library React JS.*
2. *Website* yang dibangun hanya mampu memasukkan data kedalam basis data, memperbaharui data, menghapus data dengan menggunakan dua fitur utama yaitu melakukan pemasukan data mesin dan data pemeliharaan mesin.
3. Basis data yang digunakan adalah indexedDB sebagai *local storage* dan CouchDB sebagai *remote* basis data.
4. Analisis *service worker* pada *website* dengan menggunakan *library React JS* merupakan analisis yang lebih menekankan kepada bagaimana penggunaan aplikasi ketika diakses oleh banyak *request* dan kecepatan pengaksesan *website* yang dibuat.

tware yang digunakan untuk men

i performa *website* yang dibuat adalah apache Jmeter

1.6. Sistematika Penulisan

BAB I PENDAHULUAN : Bab ini berisi latar belakang masalah, rumusan masalah, tujuan penelitian, manfaat penelitian, batasan masalah, sistematika penulisan.

BAB II TINJAUAN PUSTAKA : Pada bab ini akan dijelaskan teori-teori yang menunjang percobaan yang dilakukan.

BAB III METODOLOGI PENELITIAN : Bab ini berisi analisis kebutuhan sistem, perancangan sistem, dan skenario pengujian.

BAB IV HASIL DAN PEMBAHASAN : Bab ini berisi hasil penelitian dan pembahasan penjabaran dari penelitian yang dilakukan.

BAB V PENUTUP : Bab ini berisi kesimpulan dari hasil penelitian dan saran

BAB II

TINJAUAN PUSTAKA

2.1 Pengertian *Website*

Website atau situs dapat diartikan sebagai kumpulan halaman-halaman yang digunakan untuk menampilkan informasi teks, gambar diam atau gerak, animasi, suara, dan atau gabungan dari semuanya baik yang bersifat statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait, yang masing-masing dihubungkan dengan jaringan- jaringan halaman. Hubungan antara satu halaman *web* dengan halaman *web* yang lainnya disebut *hyperlink*, sedangkan teks yang dijadikan media penghubung disebut *hypertext* (Batubara 2015) .

Dalam beberapa dekade, *website* telah menjadi lebih besar dan kompleks. Pada awalnya *website* digunakan hanya untuk mempermudah tukar menukar dan melakukan perbaruan informasi kepada sesama peneliti yang dilakukan oleh Sir Timothy John. namun dalam perkembangannya *website* dapat melakukan manajemen konten seperti video dan gambar (Butkiewicz 2011).

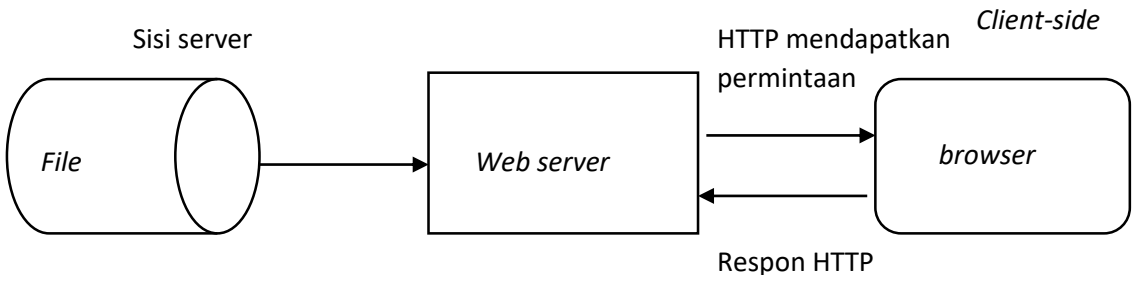
Secara teknis, *web* adalah sebuah sistem dimana informasi dalam bentuk teks, gambar, suara, dan lain-lain yang tersimpan dalam sebuah internet *web* server dipresentasikan dalam bentuk *hypertext*. Informasi di *web* dalam bentuk teks umumnya ditulis dalam format HTML. Informasi lainnya disajikan dalam bentuk grafis (dalam format GIF, JPG, PNG), suara (dalam format AU, WAV), dan objek ia lainnya (seperti MIDI, Shockwave, Quicktime Movie, 3D World). *Web* kses oleh perangkat lunak *web client* yang secara populer disebut sebagai *Browser* membaca halaman-halaman yang tersimpan dalam *webserver*

melalui protokol yang disebut HTTP (*Hypertext Transfer Protocol*). Sebagai dokumen *hypertext*, dokumen-dokumen di *web* dapat memiliki *link* dengan dokumen lain, baik yang tersimpan dalam *web* server yang sama maupun di *web* server lainnya. Link memudahkan para pengakses *web* berpindah dari satu halaman ke halaman lainnya, dan "berkelana" dari satu server ke server lain. Kegiatan penelusuran halaman *web* ini biasa diistilahkan sebagai *browsing*, ada juga yang menyebutnya sebagai *surfing* (berselancar). Seiring dengan semakin berkembangnya jaringan internet di seluruh dunia, maka jumlah situs *web* yang tersedia juga semakin meningkat. Hingga saat ini, jumlah halaman *web* yang bisa diakses melalui internet telah mencapai angka miliaran. Untuk memudahkan penelusuran halaman *web*, terutama untuk menemukan halaman yang memuat topik-topik yang spesifik, maka para pengakses *web* dapat menggunakan suatu mesin pencari (*search engine*). Penelusuran berdasarkan *search engine* dilakukan berdasarkan kata kunci (*keyword*) yang kemudian akan dicocokkan oleh *search engine* dengan basis data miliknya (Batubara 2015).

2.2 Modern *website*

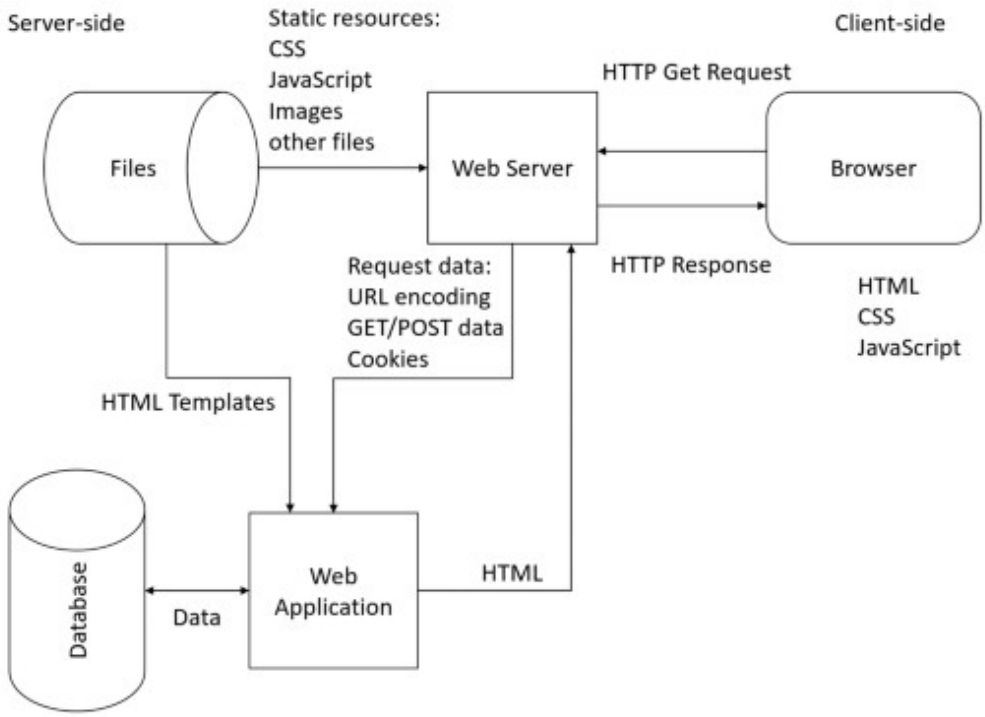
Saat ini, ada banyak cara dalam melakukan pengembangan dan pembuatan aplikasi *web*. Namun pengembangan *website* dimulai dengan memahami bagaimana arsitektur *web* yang akan digunakan, *web* statik atau dinamis dan juga penentuan pengembangan *tool* dan *service* yang digunakan.

Sebuah *website* statis terdiri atas beberapa halaman HTML, CSS yang secara bersamaan saling terhubung oleh *hyperlinks*. *Website* dinamis memiliki lokasi penyimpanan konten pada basis data dan ditampilkan berdasarkan permintaan penguna. Namun HTML dan CSS juga dapat digunakan dalam *website* dinamis ketika memiliki *java script* yang mengandung pemrograman *back-end*. *Website* modern dibangun menggunakan pemrograman *front-end* yang dieksekusi oleh *browser* dan berjalan pada sisi *client* yaitu HTML,CSS, dan pemrograman *back-end* yang dieksekusi pada sisi server yaitu *java script*, PHP, Python dan yang lainnya yang sering digunakan oleh pengembang *web*. Pemrograman *back-end* bekerja pada sisi belakang *website* yang tidak dilihat oleh pengguna dan bekerja berdampingan dalam melakukan akses basis data, sedangkan pemrograman *front-end* berhubungan dengan apa yang pengguna dapat lihat (Hannonen 2017).



Gambar 2.1 : Arsitektur *Website* statis (Hannonen 2017).

Pada *website* statis ketika pengguna ingin melakukan perpindahan halaman maka *browser* mengirimkan permintaan HTTP “GET” yang menunjukkan URL. Server kemudian mengirimkan dokumen permintaan dari sistem file menuju *browser* pengguna.

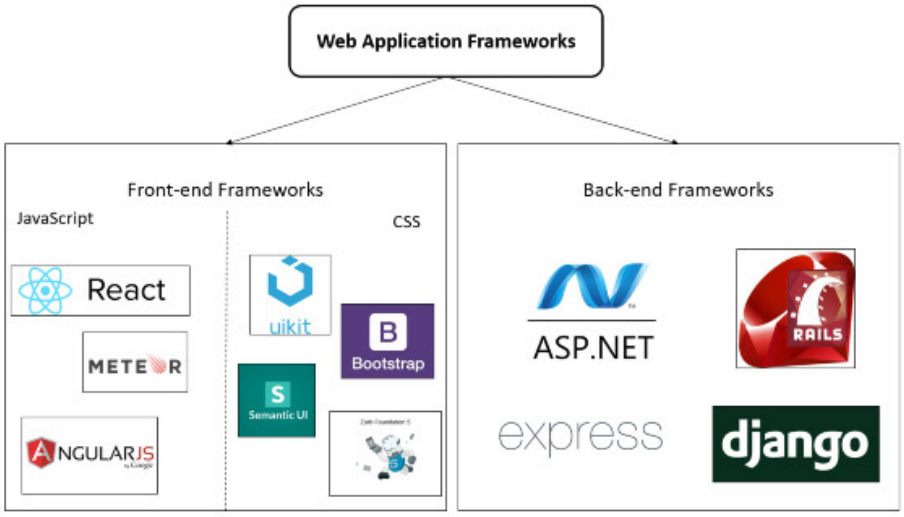


Gambar 2.2 : Arsitektur *web* dinamis (Hannonen 2017).

Dari kedua gambar yaitu gambar 2.1 dan gambar 2.2, masing masing arsitektur memiliki kekurangan dan kelebihan. *Website* statis memiliki kelebihan yaitu sederhana dan cepat dalam melakukan proses pengembangan. Namun kekurangannya adalah pengembang harus melakukan perubahan kode baik HTML CSS setiap konten mengalami perubahan. Berbeda dengan *website* dimana memiliki kemudahan dalam melakukan perubahan konten karena

bekerja dengan data yang dinamis. Namun dalam mengembangkan arsitektur *web* dinamis dibutuhkan lebih dari satu pengembang untuk memastikan *web* yang dibuat berjalan dengan semestinya dan memiliki desain yang menarik.

Seiring perkembangannya, hari ini pengembangan *web* memiliki cara pengembangan yang lebih mudah dan disukai oleh pengembang, yaitu dengan menggunakan *framework* dalam melakukan pengembangan *web*. *Framework* memudahkan para pengembang karena kemampuannya menyediakan manajemen *session*, penyimpanan data, dan melakukan *templating* sistem. *Framework* juga memudahkan dalam melakukan manajemen pengerjaan *web* yaitu dengan memisahkan bentuk pengerjaan dari sisi *front-end* dan *back-end.*



Gambar 2.3 : Contoh aplikasi *web Framework* (Hannonen 2017).

Framework sisi server adalah *Framework back-end* yang memudahkan ang dalam melakukan penulisan dan pemeliharaan aplikasi *web*. *rk* menyediakan pustaka yang sederhana yang sering digunakan dalam gun sebuah sistem *web*, termasuk URL *routing* , interaksi dengan basis

data, mendukung manajemen *session* maupun otorisasi pengguna (*user authorization*), manajemen masukan (HTML, JSON, XML), dan peningkatan keamanan terhadap serangan *web* (Hannonen 2017).

Sedangkan *front-end Framework* sangat membantu dalam melakukan desain proses dan masukan dari berbagai fitur yang ada, termasuk fitur kanvas, menu, dan model yang ada. *Framework front-end* dibagi menjadi dua yaitu *Java script Framework* (Angular,ReactJS,MeterJS) dan CSS *Framework* (Bootstrap, Foundation, SemanticUI, UIkit, MaterialUI) (Hannonen 2017).

2.3 Teknologi *Offline first app*

2.3.1 *Single-page Application*

Secara tradisional, *web* memiliki banyak tampilan halaman. Kebanyakan halaman *web* secara konsisten hanya sebuah halaman statis yang berisi teks dan gambar. Ini memungkinkan untuk melakukan koneksi antara halaman *web* melalui *hyperlink*. Ketika pengguna membuka halaman *web* yang lain melalui *hyperlink* maka *browser* akan melakukan *refresh* seluruh tampilan halaman *web* dengan konten yang baru dari halaman *web* yang baru. Kemudian muncul skrip sisi server seperti PHP yang memungkinkan untuk membuat halaman *web* yang dinamis berdasarkan dengan interaksi dari pengguna.

Kemudian paradigma baru dimulai sejak sekitar tahun 2005 ketika odel pemrograman *asynchronous Java script* dan XML (AJAX) menjadi aya tarik baru. AJAX memungkinkan halaman *web* untuk melakukan

permintaan HTTP ke *web* server, dan melakukan pembaharuan hanya pada bagian halaman *web* yang berubah. Revolusi lain dimulai pada sekitar tahun 2006 ketika dua kelompok yaitu *World Wide Web Consortium (W3C)* dan *Web Hypertext Application Technology Working Group (WHATWG)*, mulai bekerja bersama untuk sebuah standar *Hypertext Markup Language (HTML)* yang baru yaitu HTML 5. Standar HTML 5 yang baru menambahkan berbagai macam fitur kedalam sebuah perangkat *web* yang memungkinkan pengembang untuk membangun sebuah *web* yang lebih kompleks dan lebih kaya akan fitur (Vanhala 2017).

Evolusi *web* mengantarkan kepada teknologi baru dalam membangun sebuah aplikasi *web*. *Single Page Application* (SPA) adalah aplikasi *web* yang menggunakan halaman HTML yang tidak melakukan *refresh* pada saat digunakan. Sebagai gantinya SPA menggunakan *Java script* untuk setiap interaksi pengguna dan mengandalkan AJAX ketika ingin melakukan komunikasi dengan server. SPA memberikan pengalaman user yang lebih baik, waktu respon yang lebih cepat dari *web* tradisional sejak interaksi server terjadi secara *asynchronous* pada sisi belakang aplikasi, dan memerlukan data yang sedikit dalam melakukan transaksi (Vanhala 2017).

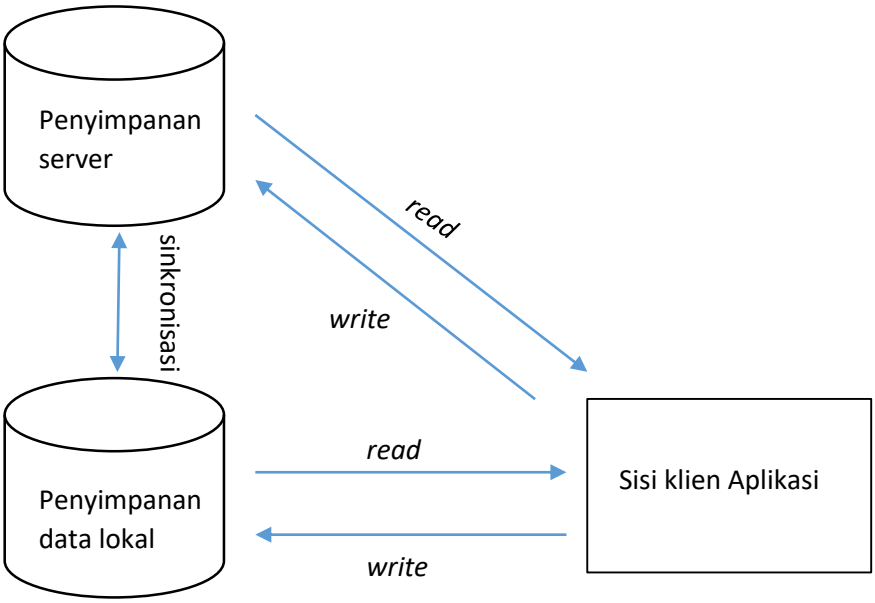
2.3.2 *Offline First*

Aplikasi *web* biasanya bekerja dengan kondisi yang memerlukan ngan internet. Oleh karena itu, ketika pengguna melakukan permintaan ke ver untuk melakukan interaksi ada proses yang harus di tempuh oleh

permintaan untuk bisa sampai ke server. Hal ini bisa menjadi masalah ketika proses yang ditempuh mengalami masalah jaringan sehingga membutuhkan waktu yang lama untuk server menerima respon dari pengguna.

Dalam membuat aplikasi *web offline* maka arsitektur yang dibuat harus menggunakan logika yang dapat bekerja walaupun koneksi internet tidak tersedia, tidak seperti aplikasi *web* yang standar. Jenis aplikasi ini menyimpan sumber daya yang penting dari aplikasi seperti *cache* dan secara terus menerus membawa informasi dari server ketika dilakukan permintaan dari *browser* (Tamire 2016).

Kemampuan aplikasi bekerja secara *offline* sangat berharga terutama ketika tidak tersedianya koneksi internet namun tetap memerlukan koneksi agar aplikasi dapat bekerja. Untuk membuat performa aplikasi *offline* bekerja dengan sukses maka perlu di lakukan pengecekan,pembaharuan dan membuat sumber daya menjadi aman yang tersimpan pada perangkat pengguna.



Gambar 2.4 : Elemen inti aplikasi *web offline* (Tamire 2016)

Sesuai dengan gambar 2.4, aplikasi *web offline* menggunakan penyimpanan lokal ketika jaringan internet tidak tersedia. Sebaliknya, ketika koneksi internet tersedia, data akan dapat dibaca dan di buat langsung dari server.

Offline first adalah paradigma desain terbaru dalam membuat sebuah aplikasi *web*. Teknologi ini digunakan pertama kali pada tahun 2012 oleh Lambert dalam artikelnya berjudul "*Offline First – A better HTML5 User Experience".* Dalam artikelnya, Lambert menjelaskan bahwa *offlline* adalah fitur yang penting dan harus dipertimbangkan pada awal proses pengembangan sistem. Lambert menjabarkan ada tiga panduan untuk melakukan pengembangan aplikasi *offline first* dari segi teknis. (1) semua gika aplikasi yang biasanya berada pada sisi server dipindahkan pada sisi ien. Server hanya digunakan sebagai media penyimpanan dan rkomunikasi dengan menggunakan *Java script Object Nation* (JSON)

antara klien dan server. (2) Membuat sebuah kode layer yang melakukan interaksi dengan sisi server API (*application programming interface*). (3) Membuat kode pemrograman yang melakukan permintaan data dari server dan melakukan *caching* di penyimpanan *browser* (Lambert 2012).

Feyerke memberikan teori yang hampir sama dengan Lambert dalam artikel nya. (1) Logika aplikasi berjalan pada *browser*. (2) Kedua, untuk memaksimalkan pengalaman *offline* pengguna, klien harus melakukan penyimpanan data pada *browser* dan mampu melakukan sinkronisasi dengan server (Feyerke 2013).

2.3.3 Web Caching

Ketika menggunakan aplikasi *web* atau melakukan pencarian informasi dari sebuah *web*, dua hal yang penting untuk diperhatikan yaitu kecepatan dan efisiensi. Setiap pengguna akan merasa puas ketika bekerja dengan kecepatan aplikasi yang tinggi. Apalagi ketika mendesain aplikasi *web*, dengan menerapkan teknologi *caching* akan mampu meningkatkan waktu akses dan menurunkan lalu lintas jaringan.

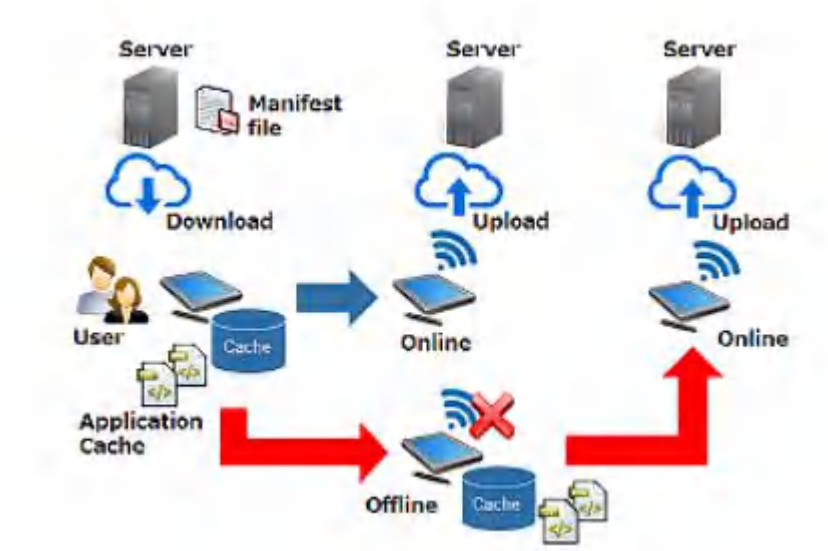
Akibatnya, *web cache* memungkinkan untuk meningkatkan kinerja dan skalabilitas aplikasi sehingga dengan menggunakan metode ini dapat mengoptimalkan kinerja aplikasi. Selain itu, *cache* mampu menyimpan Salinan halaman aplikasi *web* pada sistem penyimpanan lokal pengguna sehingga nanti dapat digunakan ketika jaringan internet tidak tersedia (Tamire 2016).

Web cache mampu melakukan penyimpanan *Cascading Style sheet* (CSS), *Java script*, gambar, HTML dan sumber daya yang lain pada *web browser* sehingga mampu menurunkan waktu akses data dari *web* server. Disisi lain ketika permintaan dikirm ke server, maka akan banyak sumber daya dikirim kembali kepada pengguna meminta halaman yang sama untuk kedua kalinya yang berarti akan membuat server menjadi sibuk dan membuat lalu lintas *web* menjadi tinggi.

Kebanyakan *web browser* akan melakukan *cache* pada halaman *web* ketika pertama kali diakses oleh pengguna. Namun melakukan *cache* halaman *web* memiliki beberapa keterbatasan. Yang pertama adalah pengembang *web* harus menentukan halaman *web* mana yang akan dilakukan *cache* dalam kondisi tidak adanya koneksi internet. Masalah kedua adalah *caching* halaman *web* otomatis tidak menjamin untuk melakukan *load cache* yang sudah disimpan ketika tidak tersedianya koneksi internet.

Hasilnya, untuk mengatasi masalah tersebut HTML5 memiliki API *offline cache*. Dengan menggunakan teknologi ini pembuat *web* dengan mudah mampu menentukan sumber daya apa yang akan dilakukan *cache* dan tersedia ketika aplikasi berjalan secara *offline*. Selain itu terdapat sebuah API *Java script* yang mampu melakukan managemen *cache* terhadap halaman *web*, sehingga pengembang *web* mampu mengatur dan melakukan anagemen *cache* (Tamire 2016).

Aplikasi HTML 5 *cache* adalah teknologi yang sangat penting untuk mendukung aplikasi berjalan secara *offline*. Hasilnya dengan menggunakan teknologi ini membuat *website* untuk bekerja dengan baik pada kasus tidak tersedianya koneksi .



Gambar 2.5 : Level tinggi Arsitektur HTML (Tamire 2016)

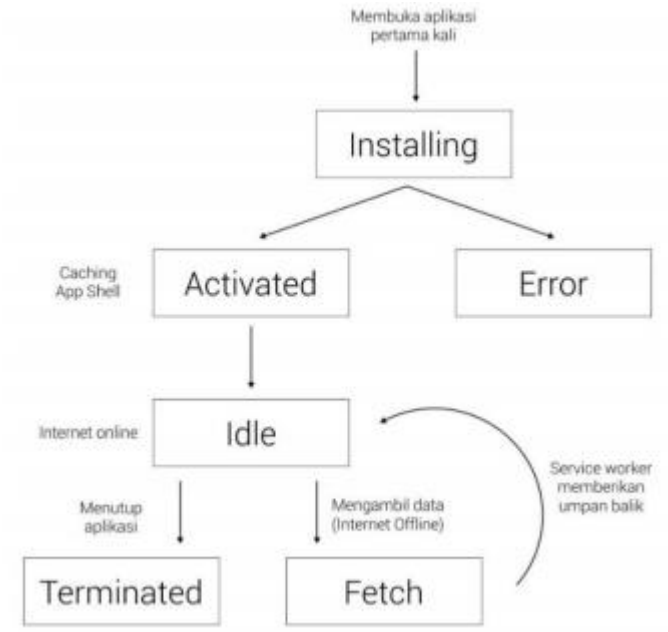
Sesuai dengan gambar 2.5, ketika permintaan dibuat untuk melakukan pada akses *single page*, maka kemudian sistem melakukan penguduhan setiap data yang berada pada daftar manifest aplikasi. Setelah semua manifest di simpan maka aplikasi akan bekerja dengan atau tanpa koneksi internet (Tamire 2016).

2.3.4 *Service worker*

Service worker adalah salah satu jenis dari *web* worker, yaitu script yang berjalan di belakang *browser* pengguna. *Service worker* pada dasarnya adalah berkas *Java script* yang berjalan pada *thread* yang berbeda dengan main *thread* menangani network *request*, *caching*, mengembalikan resource dari an bisa mengirimkan *push message*. Aset *web* dapat disimpan sebagai

cache lokal, sehingga dengan jaringan internet yang kurang memadai pun, pengguna tetap mendapat pengalaman yang baik. Aplikasi dapat tetap menjalankan halaman *web* yang sudah di-*cache* atau memberikan status koneksi tanpa *browser* menampilkan pesan kesalahan karena ketiadaan koneksi internet. Untuk memasang *service worker*, kita butuh melakukan registrasi dengan menggunakan *Java script* yang ada di halaman *web*. Setelah registrasi, *browser* akan memulai tahap pemasangan *service worker*. Setelah aktif, *service worker* akan menangani semua halaman di bawah scope dimana *service worker* di-install, lalu akan ada 2 kemungkinan *state*: terminated untuk menghemat memori, atau menangani operan data ketika ada permintaan dari halaman *web*.

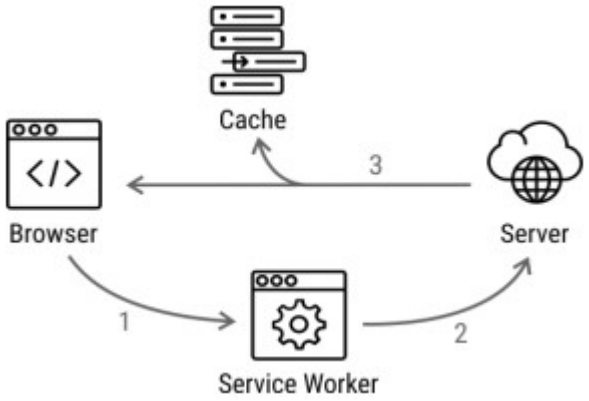
Service worker bisa menangani berbagai jenis *request*, tetapi yang bisa disimpan kedalam *cache* hanya semua *request* jenis GET. Pada *service worker*, bisa aplikasikan strategi *caching* sesuai keinginan kita, namun tidak ada satu strategi terbaik untuk *caching* konten dinamis, dan ada banyak situasi yang bisa mempengaruhi strategi *caching* (Adi, Akbar, and Khotimah 2017).



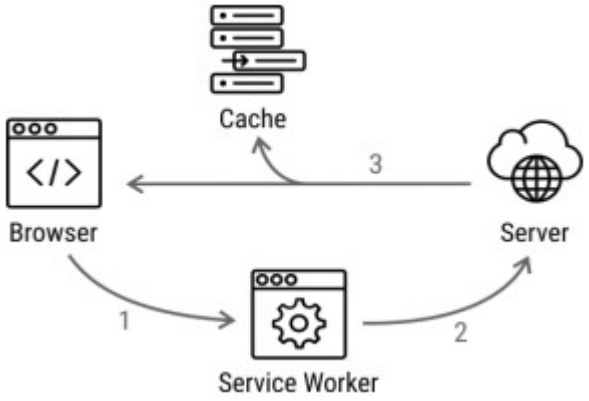
Gambar 2.6 : Daur hidup *service worker* (Adi et al. 2017)

Service worker bekerja sebagai pengatur *event fetch* dari *browser*, lalu *service worker* memutuskan apakah *request* akan diteruskan ke server atau ke *cache* berdasarkan kondisi jaringan *online* atau *offline*. Dari banyak jenis strategi *caching* pada *service worker* akan digunakan jenis *network first, cache fallback*. Pada strategi ini pertama-tama *service worker* mengecek apakah network memberikan respons, dan jika berhasil, kembalikan data sekarang ke halaman. Jika gagal, *service worker* mengembalikan data dari *cache*. Strategi ini dipakai ketika

membutuhkan data yang selalu baru seperti respons API, tapi butuh menampilkan sesuatu ketika network tidak bisa dicapai (Adi et al. 2017).



Gambar 2.6 : *Service worker* dalam kondisi *offline* (Adi et al. 2017).



Gambar 2.7 : *Service worker* dalam kondisi *onnline* (Adi et al. 2017)

Service worker akan meneruskan setiap *request* berjenis GET dari halaman *web* ke server dalam kondisi *online*, lalu menduplikasi respons server dan disimpan ke dalam *cache* di *browser*, lalu respons server diteruskan kembali ke halaman *web* seperti digambarkan pada Gambar 2. Setiap halaman yang pernah dikunjungi oleh pengguna secara otomatis akan tersedia dalam versi *offline*-nya, hanya saja tidak bisa mengirim data ke server. Ketika kondisi *offline* atau tidak ada alik dari server, maka akan dikembalikan halaman dari *cache*. Halaman

bisa ditampilkan apabila halaman tersebut sudah ada di dalam *cache* sebelumnya seperti digambarkan pada Gambar 2.7 (Adi et al. 2017).

Dengan menggunakan *service worker* maka berbagai jenis fitur dapat digunakan yaitu , melakukan *intretrcept request* dan *Responese* , melakukan *push notification,* dan *background sync. Intercept request dan response* bekerja melalui *service worker*, sehingga setiap *request* yang ada dapat dilakukan *intercept* dan membuat skenario pengaksesan *web*. Sedangkan fitur *push notification* dapat memunculkan notifikasi melalui *service worker* walaupun halaman *web* tertutup. Dan *background* sinkronisasi, ada atau tidaknya jaringan internet maka *service worker* tetap dapat melakukan permintaan ke server dan melakukan sinkronisasi secara berkala.

2.4 Bahasa Pemrograman

Bahasa pemrograman, atau sering diistilahkan juga dengan Bahasa komputer, adalah instruksi standar untuk memerintah komputer. Bahasa pemrograman ini merupakan suatu himpunan dari aturan sintaks dan semantik yang dipakai untuk mendefiniskan program komputer. Adapun Bahasa pemrograman yang digunakan dalam pengerjaan sistem ini adalah *java script* dengan menggunakan *Framework Node Js, React Js, dan Material UI*.

2.4.1. *Java script*

Java script adalah bahasa pemrograman yang popular. *Java script* adalah
pemrograman yang digunakan untuk HTML dan *web*, untuk Server, PC,
, tablet dan lebih banyak lagi. Kode pemrograman *Java script* dapat

disisipkan kedalam halaman HTML Pada awalnya, *Java script* mulai diperkenalkan di *browser* Netscape Navigator 2. Namun waktu itu namanya bukan *java script*, namun *LiveScript*. Mengingat pada waktu itu teknologi Java sedang panas-panasnya atau sedang tren, maka pihak Netscape memutuskan untuk mengganti namanya menjadi *Java script*, yang sepertinya nama tersebut lebih *marketible* dibandingkan *LiveScript*. Selanjutnya pihak Microsoft (rival Netscape) pun mulai ikut-ikutan memfasilitasi *web browser* buatannya, 'Internet Explorer', supaya bisa mendukung *Java script*. Namun mungkin karena gengsi, pihak Microsoft memberi nama bahasa yang lain, yaitu *Java script*. Mulai saat itu, Netscape dan Microsoft mulai berlomba-lomba mengembangkan bahasa tersebut dalam versi yang berlainan. Oleh sebab persaingan itulah terkadang suatu *Java script* mungkin bisa bekerja dengan baik di *browser* Netscape, tapi tidak demqikian halnya di IE, begitu pula sebaliknya (Permana 2016).

Pada awalnya, *Java script* mulai diperkenalkan di *browser* Netscape Navigator 2. Namun waktu itu namanya bukan *Java script*, namun *LiveScript*. Mengingat pada waktu itu teknologi Java sedang panas-panasnya atau sedang tren, maka pihak Netscape memutuskan untuk mengganti namanya menjadi *Java script*, yang sepertinya nama tersebut lebih marketible dibandingkan LiveScript. Selanjutnya pihak Microsoft (rival Netscape) pun mulai ikut-ikutan memfasilitasi *web browser* buatannya, 'Internet Explorer', supaya bisa mendukung *Java script*. Namun mungkin karena gengsi, pihak Microsoft eri nama bahasa yang lain, yaitu Jscript. Mulai saat itu, Netscape dan soft mulai berlomba-lomba mengembangkan bahasa tersebut dalam versi

yang berlainan. Oleh sebab persaingan itulah terkadang suatu *Java script* mungkin bisa bekerja dengan baik di *browser* Netscape, tapi tidak demikian halnya di IE, begitu pula sebaliknya (Permana 2016).

Ada dua jenis bagaimana *Java script* dibuat, pertama *Java script* ditulis dalam file yang terpisah dengan HTML, kedua *Java script* ditulis dalam HTML. *Java script* yang ditulis diluar HTML disebut Eksternal *Java script* dengan ektensi file .js. Dalam HTML, penulisan script diawali dengan. Script yang akan dijalankan harus diletakkan diantara <script> dan </script> tag <script> memiliki beberapa atribut, namun yang terpenting adalah atribut language dan type. Karena *Java script* bukna satu – satunya bahasa scripting, maka sangatlah perlu untuk memberitahukan kepada *browser* bahwa bahasa script yang digunakan adalah *Java script* dan selanjutnya *browser* akan menjalankan modul pendukung *Java script* untuk memprosesnya (Permana 2016).

2.4.2. **Node JS**

Node JS merupakan platform server yang dibangun menggunakan *Java script* dan berjalan di dalam interpreter Chrome *Java script* runtime. Dibuat untuk pengembangan perangkat lunak berbasis *web* dengan cepat, aplikasi jaringan yang scalable. Node.js menggunakan event-driven, model non-blocking I/O yang membuatnya menjadi ringan dan efisien. Sangat baik digunakan untuk aplikasi waktu-nyata yang digunakan diberbagai perangkat (Muhammad Agung ana 2014) .

ode.js adalah perangkat lunak yang didesain untuk mengembangkan si berbasis *web* dan ditulis dalam sintaks bahasa pemrograman *Java script*.

Bila selama ini *Java script* dikenal sebagai bahasa pemrograman yang berjalan di sisi *client* / *browser* saja, maka Node.js ada untuk melengkapi peran *Java script* sehingga bisa juga berlaku sebagai bahasa pemrograman yang berjalan di sisi server, seperti halnya PHP, Ruby, Perl, dan sebagainya. Node.js dapat berjalan di sistem operasi Windows, Mac OS X dan Linux tanpa perlu ada perubahan kode program. Node.js memiliki pustaka server HTTP sendiri sehingga memungkinkan untuk menjalankan server *web* tanpa menggunakan program server *web* seperti Apache atau Nginx.

Node JS dibuat dengan *engine* yang sama dengan *browser chrome* yang dikembangkan oleh google yang bersifat *open source* dimana Node JS memiliki keuntungan pada sifatnya yang *non blocking*. Sebagai contoh misalnya pada bahasa program biasa bila terdapat sebuah fungsi A yang berjalan, maka umumnya fungsi A harus selesai terlebih dahulu kemudian menjalankan fungsi B. tetapi berbeda dengan Node JS yang sifatnya parallel, atinya Node JS akan mengerjakan hal – hal yang sama dan ketika sudah selesai dengan fungsi yang ada maka dapat ditambahkan sebuah fungsi callback. *Callback* ini yang akan melihat apakah tugas A sudah selesai dikerjakan dan tidak akan menunggu A selesai tetapi juga akan sambil menjalankan tugas B.

Jadi pada program yang lain sebuah proses akan diselesaikan terlebih dahulu baru dapat ke proses berikutnya. Tapi pada Node JS berbeda karena dia berbasis *Java script* maka dia tidak akan menunggu proses A itu selesai tetapi proses akan dijalankan satu-satu lalu akan menjalankan proses berikutnya kita sudah tentukan. Dengan kata lain Node JS bisa mengatasi *multiple*

request secara bersamaan dan prosesnya itu tidak langsung diselesaikan tetapi menjalankan beberapa proses terlebih dahulu.

2.4.3. React js

ReactJS, juga di kenal dengan nama React atau ReactJS, adalah pustaka *java script* bersifat *open source* digunakan untuk membangun antar muka pengguna. ReactJS digunakan untuk mengatasi tampilan pada aplikasi berhalaman tunggal (*single page application)* dan pengembangan aplikasi mobile (Khuat 2018) .

React dikembangkan oleh facebook dengan berusaha untuk menyediakan kecepatan, kesederhanaan, dan skalabilitas. Beberapa fitur utama react adalah JSX, komponen *state*ful, dan Virtual Document Object Model.

2.4.3.1. JSX

JSX adalah ekstensi sintaks *Java script* yang terlihat mirip XML. JSX sintaks HTML atau XML yang digunakan pada React JS dan memperluas ECMAScript sehingga XML/HTML dapat bekerja pada kode *Java script* atau React. Pada dasarnya, JSX dapat diperlakukan seperti ekspresi pada *Java script*, artinya elemen JSX dapat disimpan dalam variabel,function,object, array, dan lain – lain seperti pada umumnya *Java script* (Anonym n.d.) .

2.4.3.2. *Stateful Components*

React memungkinkan pengguna untuk membagi antar muka pengguna menjadi independen, dan dapat digunakan berkali kali dengan melakukan pemanggilan komponen React. Komponen React meggunakan

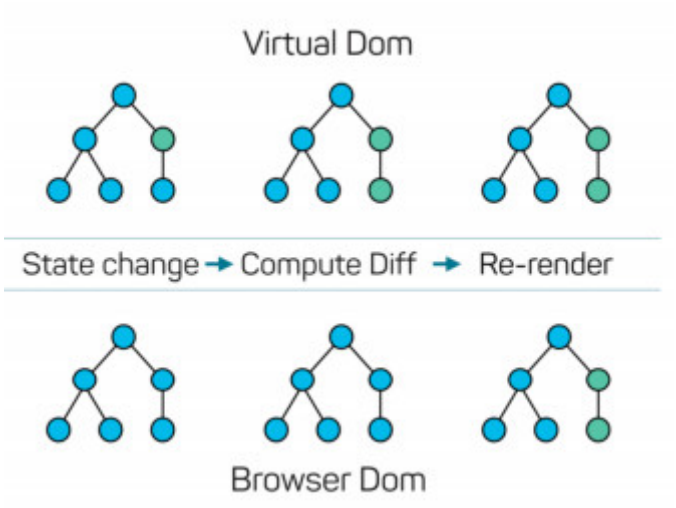
implementasi metode render yang mengambil dan menerima masukan dan mengembalikan hasil masukan menjadi sebuah tampilan antar muka. Setiap komponen memiliki beberapa metode *lifecycle* yang dapat mengganti untuk melakukan eksekusi kode pada waktu tertentu selama proses berlangsung. Pemanggilan method dapat dilakukan menggunakan API (*Aplication Programming Interface*) React.

State adalah objek *Java script* sederhana yang digunakan untuk merekam dan memberi reaksi sesuai perintah pengguna. Setiap kelas berbasis komponen mendefinisikan objek *state*nya sendiri. Kapanpun komponen *state* berubah, komponen, dan semua komponen turunan, secepat mungkin akan dilakukan *re-render*. *State* dapat menyimpan sebuah nilai dan dapat dikirim melalui antar komponen turunan dengan menggunakan *props*.

2.4.3.3. *Virtual Document Object Model*

HTML DOM (*Document Object Model)* awalnya digunakan pada halamam statis yang tidak bekerja pada halaman antarmuka yang bersifat dinamis. Ketika DOM mengalami pembaharuan, DOM akan melakukan pembaharuan setiap *node* tanpa melakukan *reload* halaman *web*. Ini sering digunakan pada *single page application* yang memiliki ratusan halaman yang dihasilkan secara dinamis melalui aksi pengguna. Pada halaman dinamis, HTML DOM harus melakukan pengecekan perubahan setiap *node* data pada interval reguler. Hal ini dapat mengurangi performansi

aplikasi. Sehinga virtual hadir sebagai solusi terhadap masalah tersebut (Khuat 2018).



Gambar 2.8 : Perbedaan Virtual DOM dan *Browser* DOM dalam beberapa tahap perubahan (Khuat 2018)

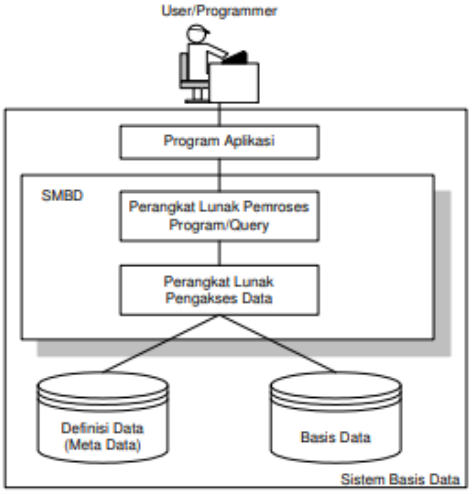
2.5 Basis data

Data merupakan fakta mengenai suatu objek seperti manusia, benda, peristiwa, konsep, keadaan dan sebagainya yang dapat dicatat dan mempunyai arti secara implisit. Data dapat dinyatakan dalam bentuk angka, karakter atau simbol, sehingga bila data dikumpulkan dan saling berhubungan maka dikenal dengan istilah basis data (basis data). Sedangkan menurut George Tsu-der Chou basis data merupakan kumpulan informasi bermanfaat yang diorganisasikan ke dalam aturan yang khusus. Informasi ini adalah data yang telah diorganisasikan ke dalam bentuk yang sesuai dengan kebutuhan seseorang . Menurut *Encyclopedia of er Science and Engineer*, para ilmuwan di bidang informasi menerima standar informasi yaitu data yang digunakan dalam pengambilan an. Definisi lain dari basis data menurut Fabbri dan Schwab adalah sistem

berkas terpadu yang dirancang terutama untuk meminimalkan duplikasi data. Menurut Ramez Elmasri mendefinisikan basis data lebih dibatasi pada arti implisit yang khusus, yaitu (Haidar Dzacko 2007):

a. Basis data merupakan penyajian suatu aspek dari dunia nyata (real world).
b. Basis data merupakan kumpulan data dari berbagai sumber yang secara logika mempunyai arti implisit. Sehingga data yang terkumpul secara acak dan tanpa mempunyai arti, tidak dapat disebut basis data.
c. Basis data perlu dirancang, dibangun dan data dikumpulkan untuk suatu tujuan. Basis data dapat digunakan oleh beberapa user dan beberapa aplikasi yang sesuai dengan kepentingan user.

Dari beberapa definisi-definisi tersebut, dapat dikatakan bahwa basis data mempunyai berbagai sumber data dalam pengumpulan data, bervariasi derajat interaksi kejadian dari dunia nyata, dirancang dan dibangun agar dapat digunakan oleh beberapa pengguna untuk berbagai kepentingan.



Gambar 2.9 : Konsep Sistem Basis data (Haidar Dzacko 2007)

C. J. Date menyatakan bahwa sistem basis data dapat dianggap sebagai tempat untuk sekumpulan berkas data yang terkomputerisasi dengan tujuan untuk memelihara informasi dan membuat informasi tersebut tersedia saat dibutuhkan (Haidar Dzacko 2007).

Basis data yang membutuhkan sebuah media untuk melakukan penyimpanan dan pengelolaan data yang ada yang diolah oleh program sql maupun nosql. Keduanya adalah media yang digunakan untuk melakukan pengolahan basis data yang digunakan sesuai dengan kebutuhan pengembang.

2.5.1 SQL Basis data

SQL atau *Structured Query Language* adalah sebuah bahasa yang digunakan untuk mengakses data dalam basis data relasional (Wikipedia,2018). SQL merupakan sebuah bahasa komputer yang mengikuti standar ANSI (American Nasional Standard Institute) yang digunakan dalam manajemen basis data relasional. Dengan SQL, kita dapat mengakses basis data, menjalankan query untuk mengambil data dari basis data, menambahkan data ke basis data, menghapus data di dalam basis data, dan mengubah data di dalam basis data. Saat ini hampir semua server basis data yang ada mendukung SQL untuk melakukan manajemen datanya.

Sejarah SQL dimulai dari artikel seorang peneliti dari IBM bernama EF Codd yang membahas tentang ide pembuatan basis data relasional pada bulan Juni 1970. Artikel ini juga membahas kemungkinan pembuatan bahasa standar untuk es data dalam basis data tersebut. Bahasa tersebut kemudian diberi nama (Structured English Query Language). Setelah terbitnya artikel tersebut,

IBM mengadakan proyek pembuatan basis data relasional berbasis bahasa SEQUEL. Akan tetapi, karena permasalahan hukum mengenai penamaan SEQUEL, IBM pun mengubahnya menjadi SQL. Implementasi basis data relasional dikenal dengan System/R. Di akhir tahun 1970-an, muncul perusahaan bernama Oracle yang membuat server basis data populer yang bernama sama dengan nama perusahaannya. Dengan naiknya kepopuleran Oracle, maka SQL juga ikut populer sehingga saat ini menjadi standar *de facto* bahasa dalam manajemen basis data (Haidar Dzacko 2007).

2.5.2 NoSQL Basis data

Istilah NoSQL diciptakan oleh Carlo Strozzi pada tahun 1998 dan mengacu pada basis data non-relasional, pada tahun 2009 Eric Evans memperkenalkan kembali istilah NoSQL. Baru-baru ini, istilah ini memiliki makna lain, yaitu "Not Only SQL", istilah yang lebih baik dari sebelumnya yang lebih dikenal dengan anti-relasional. NoSQL mengakomodasi tanda yang tidak terstruktur, kehadirannya bukan untuk menggantikan SQL namun kedua teknologi ini dapat saling berdapampingan. Perbedaan utama kedua basis data ini adalah SQL memiliki skema yang kaku sementara basis data NoSQL menawarkan desain yang fleksibel yang dapat diubah tanpa downtime atau gangguan layanan. NoSQL juga dirancang untuk menyimpan data yang didistribusikan untuk kebutuhan data dalam skala besar; misalnya Facebook memiliki 500 juta pengguna dan Twitter terakumulasi *terabyte* data, basis data NoSQL telah memiliki popularitas yang tinggi, sehingga ini diklaim lebih baik dari basis data SQL, Basis data NoSQL dimotivasi derhanaan, *horizontal scaling* dan kontrol yang lebih dalam kesediaan

data. NoSQL menjadi solusi dalam penanganan data dalam jumlah besar yang berkembang pesat saat ini. Data ini biasanya non-terstruktur, kompleks dan tidak cocok digunakan dalam model relasional. Contoh data yang bisa kita yang bisa kita rasakan adalah data yang berasal dari smartphone yang mencatat lokasi *broadcast* setiap saat, video dan kamera bahkan halaman halaman *website* yang berisi banyak informasi serta dokumen (No dan Junaidi 2016).

Teknologi NoSQL memiliki keunggulan sendiri dalam melakukan manajeman basis data sehingga penggunaan jenis basis data ini lebih diminati oleh pengembang. Ada banyak jenis basis data NoSQL yang sering digunakan oleh pengembang sesuai dengan kebutuhan, seperti IndexedDB, RXDB, PouchDB, dan CouchDB. keempat basis data tersebut memiliki keunggulan yaitu mampu menjadi basis data yang memungkinkan pengembang untuk membuat sebuah media penyimpanan *offline* bagi sistem yang bekerja, sehinga cocok digunakan pada sebuah aplikasi yang bekerja secara *online* maupun *offline*.

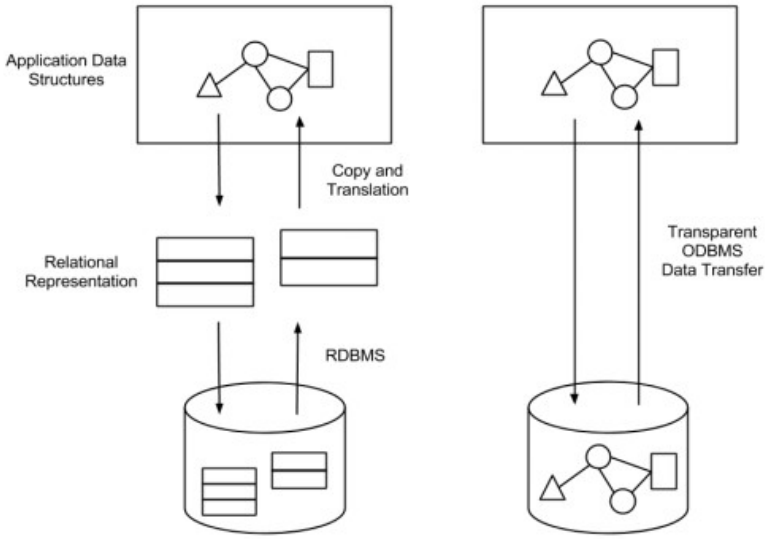
2.5.2.1 IndexedDB

IndexedDB adalah sebuah API untuk penyimpanan lokal sisi klien, yang berarti teknologi ini mampu meyimpan data terstruktur. Pada tahun 2009, Oracle menawarkan IndexedDB sebagai penyimpanan lokal baru pada *web browser* standar. Pada jenis basis data ini, data disimpan sebagai *key value pair* tapi *key* dapat diartikn sebagai property dari objek yang disimpan dalam nilai [illegible]zahhebi 2013).

Umumnya IndexedDB tidak memiliki batas kapasitas, tetapi setiap

browser memiliki aturan sendiri dalam kapasitas. Sebagai contoh, Firefox tidak menekankan pada kapasitas, namun ketika data lebih dari 50MB maka aka nada permintaan izin untuk melakukan penyimpanan data. Begitu pula dengan Google Chrome yang memungkinkan aplikasi untuk menggunkan 20% dari penyimpanan lokal, walaupun akan tidak mungkin untuk melakukan penyimpanan lebih (Mozahhebi 2013).

IndexedDB sebagai *object oriented* basis data NoSQL, yang membedakannya dengan kebanyakan basis data tradisional. Basis data NoSQL memiliki skema, dimana menyimpan dan mengembalikan data tidak menggunakan *structured query language* (*SQL).* Selain itu IndexedDB bukan merupakan basis data relasional (RDBMS), dimana tabel memiliki baris dan kolom. Sebenarnya, IndexedDB adalah basis data berbasi objek yang mengolah data dalam bentuk objek (Mozahhebi 2013).



Gambar 2.10 : Perbandingan *Relasional* basis data dan basis data *object* (Mozahhebi 2013)

IndexedDB API dapat digunakan melalui *Java script* dalam hal melakukan menyimpan,mengambil, menghapus, dan melakukan pembaharuan data. API ini menggunakan DOM *events* untuk menggunakan operasi basis data. Semua hirarki objek seperti JSON mendukung API ini.

2.5.2.2 RXDB

RXDB adalah *object store* yang memiliki kinerja yang cepat yang dibangun diatas SQLIite (Nylas,2016) . Rxdb digunakan sebagai media untuk melakukan komunikasi ke sisi server dari sebuah aplikasi *website*. Segala bentuk transaksi data dari user menuju basis data akan ditangani oleh Rxdb.

2.5.2.3 Couch DB

CouchDB merupakan salah satu basis data NoSQL berbasis dokumen yang masuk dalam pembinaan Apache Foundation. Bersama dengan Cassandra, Hadoop, Tomcat, Lucene, CouchDB tinggal serumah dalam dunia *open source.* CouchDB dibangun menggunakan bahasa pemrograman Erlang yang mengandalkan pada reliabilitas dan konkurensi. CouchDB juga menjadi salah satu basis dalam pengembangan IBM Cloudant, sebuah solusi basis data berbasis *cloud* yang ditawarkan IBM kepada *enterprise* (Ridwan Fajar,2016).

2.5.2.4 Pouch DB

PouchDB adalah basis data bersifat *open source* dibangun menggunakan *script* dan merupakan pengembangan dan model dari CouchDB. dengan ggunakan API ini kita mampu mebangun dan mengembangkan sebuah

aplikasi yang mampu bekerja secara *online* maupun *offline*. Teknologi ini menggunakan *Web*SQL dan IndexedDB untuk melakukan penyimpanan data.

Pouch DB bekerja ketika aplikasi berjalan *offline* , data akan disimpan secara lokal menggunakan *Web*SQL dan IndexedDB pada *browser*. Dan ketika aplikasi kembali *online*, data akan di sinkronisasi menggunakan CouchDB (Colanus, Drajana, and Selection 2017).