SKRIPSI

PERANCANGAN APLIKASI YANG HANDAL BENCANA DENGAN *PROGRESSIVE WEB APP* MENGGUNAKAN *REACT JS* DAN *COUCHDB* STUDI KASUS : INDUSTRI KECIL DAN MENENGAH

Disusun dan diajukan oleh :

CHARINA

D421 15 012

DEPARTEMEN TEKNIK INFORMATIKA

FAKULTAS TEKNIK

UNIVERSITAS HASANUDDIN

MAKASSAR

2021

LEMBAR PENGESAHAN SKRIPSI

PERANCANGAN APLIKASI YANG HANDAL BENCANA DENGAN *PROGRESSIVE WEB APP* MENGGUNAKAN *REACT JS* DAN *COUCHDB* STUDI KASUS : INDUSTRI KECIL DAN MENENGAH

Disusun dan diajukan oleh

CHARINA
D421 15 012

Telah dipertahankan di hadapan Panitia Ujian yang dibentuk dalam rangka Penyelesaian Studi Program Sarjana Program Studi Teknik Informatika Fakultas Teknik Universitas Hasanuddin pada tanggal 19 Februari 2021
dan dinyatakan telah memenuhi syarat kelulusan

Menyetujui,

Pembimbing Utama,	Pembimbing Pendamping
Dr. Eng. Zulkifli Tahir, S.T., M.Sc.	A. Ais Prayogi Alimuddin, S.T., M.Eng
Nip. 19840403 201012 1 004	Nip. 19830510 201404 1 001

Ketua Program Studi,

Dr. Amil Ahmad Ilham, S.T., M.IT
Nip. 19731010 199802 1 001

PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini:

Nama	: CHARINA
NIM	: D421 15 012
Program Studi	: TEKNIK INFORMATIKA
Jenjang	: S1

Menyatakan dengan ini bahwa karya tulisan saya berjudul

PERANCANGAN APLIKASI YANG HANDAL BENCANA DENGAN *PROGRESSIVE WEB APP* MENGGUNAKAN *REACT JS* DAN *COUCHDB* STUDI KASUS : INDUSTRI KECIL DAN MENENGAH

Adalah karya tulis saya sendiri dan bukan merupakan pengambilan alihan tulisan orang lain bahwa skripsi yang saya tulis ini benar-benar merupakan hasil karya saya sendiri.

Apabila dikemudian hari terbukti atau dapat dibuktikan bahwa sebagian atau keseluruhan skripsi ini hasil karya orang lain, maka saya bersedia menerima sanksi atas perbuatan tersebut.

Makassar, 23 Februari 2021

Yang Menyatakan

CHARINA

KATA PENGANTAR

Puji dan syukur penulis panjatkan kepada Tuhan Yang Maha Esa, karena berkat, rahmat dan karunia-Nya sehingga tugas akhir yang berjudul "*Perancangan Aplikasi yang Handal Bencana dengan Progressive Web App menggunakan React Js dan CouchDB Studi Kasus:Industri Kecil dan Menengah*" ini dapat diselesaikan sebagai salah satu syarat dalam menyelesaikan jenjang Strata-1 (S-1) pada Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin.

Penulis menyadari bahwa dalam penyusunan dan penulisan laporan tugas akhir ini tidak lepas dari bantuan, bimbingan serta dukungan dari berbagai pihak, dari masa perkuliahan sampai dengan masa penyusunan tugas akhir. Oleh karena itu, penulis dengan senang hati menyampaikan terima kasih kepada:

1. Kedua Orang tua penulis, Bapak Nessa Marulu dan Ibu Suryana dan Kakak Echi Nesthi serta Adik David Nestha yang selalu memberikan dukungan, doa, semangat dan kekuatan dalam menjalani perkuliahan ini terlebih pada saat mengerjakan Tugas Akhir;
2. Bapak Dr. Eng. Zulkifli Tahir, S.T., M.Sc., selaku pembimbing I dan Bapak A. Ais Prayogi Alimuddin, S.T., M.Eng., selaku pembimbing II yang selalu menyediakan waktu, tenaga, pikiran dan perhatian yang luar biasa untuk mengarahkan penulis dalam penyusunan Tugas Akhir;
3. Bapak Adnan, ST., M.T., Ph.D., dan Bapak Dr.Eng.Muhammad Niswar,ST.,M.I.T selaku dosen penguji yang telah memberikan saran sehingga laporan skripsi ini menjadi lebih baik;

4. Bapak Dr. Amil Ahmad Ilham, ST., M.IT., selaku Ketua Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin atas bimbingannya selama masa perkuliahan penulis;
5. Bapak Dr. Amil Ahmad Ilham, ST., M.IT.,Ph.D dan Bapak Dr. Eng. Zulkifli Tahir, S.T., M.Sc., selaku dosen pembimbing akademik yang telah memberikan bimbingan selama masa perkuliahan penulis;
6. Bapak Robert dan Bapak Zainuddin serta segenap Staf Departemen Teknik Informatika yang telah membantu penulis;
7. Kepada pemilik CV.Mycotopia, Bapak Akmal Iksan yang telah membantu penulis dalam pengambilan data;
8. Kepada sepupu penulis Hasmawati, S.Tr,Ak, dan nenek penulis I MELLE yang setia memberikan semangat, doa, dan dukungan dalam proses penyusunan Tugas Akhir ini;
9. Kepada teman-teman penulis Laura, Christine, Billa, dan Dewi, yang telah memberikan bantuan dalam proses penyelesaian Tugas Akhir ini;
10. Kepada Ardya Dwi Rahmasari, Dian Indani, dan Rizky Nadya Fatma yang selalu memberikan dukungan dan menghibur penulis dalam proses pengerjaan tugas akhir.
11. Kepada keluarga Sapu Lidi (Fuad Khairi Hamid, Jusmiati, Muhammad Arief Wicaksono, Laura Natalia Nainggolan, Fadel Rezky Ramadhan, Muhammad Zulfachril Asiari, Khusnul Khatima, Reka Regina, Ryan Rafli, Sabtian Juliana, Said Syamil Amas, dan Umniyah Nur Aprilyah**)**, yang telah memberikan

semangat dan menjadi teman penulis selama menjalani perkuliahan sampai dengan penyelesaian Tugas Akhir.

12. Teman-teman Lab UBICON, yang telah memberikan dukungan dan semangat;
13. Teman-teman HYPERV15OR FT-UH atas dukungan dan semangat yang telah diberikan selama ini.
14. Para responden yang bersedia meluangkan waktunya untuk berpartisipasi sebagai bagian penting dalam kesuksesan penelitian ini.
15. Serta seluruh pihak yang tidak sempat penulis sebutkan satu persatu, yang telah meluangkan waktu, tenaga dan pikiran selama penyusunan laporan Tugas Akhir ini.

Akhir kata, penulis berharap semoga Tuhan Yang Maha Kuasa berkenan membalas segala kebaikan dari semua pihak yang telah banyak membantu. Semoga Tugas Akhir ini dapat memberikan manfaat dan menambah wawasan pembaca khusus.

Makassar, Februari 2021

Penulis,

(Charina)

ABSTRAK

Letak geografis Indonesia yang terletak di *Ring of Fire* membuat Indonesia rentan terhadap bencana alam. Bencana alam yang terjadi mengakibatkan kerugian ekonomi, salah satunya di bidang industri kecil dan menengah (IKM). Sistem Informasi IKM tidak dapat diakses akibat jaringan yang tidak stabil. Untuk itu dibutuhkan sistem *website* yang dapat difungsikan walaupun koneksi jaringan lemah ataupun tidak ada koneksi internet sekalipun (*offline*). Dengan memadukan teknologi *service worker* dan React Js sebuah sistem dapat dibuat dengan menerapkan konsep *Progressive Web Apps*. Sebagai studi kasus teknologi ini akan dimanfaatkan untuk menunjang proses bisnis dari salah satu IKM yang beralamat di kota Makassar yakni CV.Mycotopia yang bergerak di bidang budidaya jamur tiram. Proses penelitian ini akan menerapkan teknologi sistem yang terintegrasi *service worker* di sisi client dengan menggunakan *raspberry pi* sebagai web server. Hasil penelitian menunjukkan bahwa web sistem informasi IKM dengan penerapan konsep *Progressive Web Apps* pada web server *Raspberry pi* dapat diakses secara *offline* serta dapat menyimpan, mengubah, atau menghapus dan membaca data dalam keadaan *offline*. Aplikasi ini dianalisis berdasarkan kuisioner dengan hasil untuk kategori *usefulness* 87% (sangat layak), *ease of use* 81% (sangat layak), *ease of learning* 79,666% (sangat layak), *System (Reliable & Connectivity Independent)* 81,5% (sangat layak), serta *Satisfaction* 74,666% (layak) yang menunjukkan bahwa pengguna puas dengan web sistem informasi IKM.

Kata Kunci : Industri Kecil dan Menengah, Web, *Progressive Web Apps, Service Worker, React Js*

DAFTAR ISI

DAFTAR TABEL

DAFTAR GAMBAR

BAB I

PENDAHULUAN

1.1. Latar Belakang

Di era 4.0, industri membutuhkan konektifitas serta interaksi melalui teknologi informasi dan komunikasi yang terintegrasi dan dapat dimanfaatkan di seluruh rantai nilai manufaktur guna mencapai efesiensi dan peningkatan kualitas produk. Dikutip dari laman ekonomi.bisnis.com, Menteri perindustrian Airlangga Hartanto mengatakan inovasi teknologi digital yang dihasilkan merupakan sebuah *tools* yang dapat menjawab kebutuhan industri. Transformasi digital menjadi sorotan utama karena jumlah IKM di Indonesia berjumlah lebih dari 4,4 juta Unit Usaha berdasarkan Badan Pusat Statistik (BPS). Jumlah tersebut setara dengan 99% dari seluruh unit usaha.

Namun, lokasi geografis Indonesia yang terletak di *Ring of Fire* membuat Indonesia rentan terhadap bencana alam yang mengakibatkan rusaknya berbagai infrastruktur yang ada. Seperti peristiwa gempa dan tsunami di Palu dan Donggala, yang mengakibatkan jaringan listrik terputus di wilayah tersebut. Akibatnya jaringan telekomunikasi mengalami *black out* atau putus. Selain itu, bencana alam juga mengakibatkan kerugian ekonomi yang cukup besar. Salah satunya di bidang industri. Mengingat banyaknya industri berbasis rumah tangga (industri skala kecil dan menengah), kerugian ekonomis disebabkan oleh rusak atau tidak adanya pasokan listrik di daerah tersebut menyebabkan beberapa komponen penunjang produksi maupun manajemen keuangan rusak atau tidak berfungsi, seperti perangkat telekomunikasi dan sistem informasi industri seperti website manajemen

produksi dan keuangan yang tidak bisa diakses, dimana website ini memiliki peranan penting dalam mengatur produksi, dan keuangan industri, keuntungan maupun manajemen bahan baku produksi. Ketika terjadi bencana, sistem informasi ini tidak dapat diakses karena jaringan tidak stabil/mati pada saat pasca terjadinya bencana. Sistem Informasi yang dimiliki Industri Kecil dan Menengah yang ada hanya bisa diakses ketika sedang *online*.

Sementara itu Industri Kecil dan Menengah rata-rata memiliki modal yang tidak besar dibandingkan industri besar, karena itu kita akademisi dan pemerintah perlu berinvestasi untuk mengembangkan dan membantu dalam mengurangi resiko industri pasca bencana termasuk dibidang teknologi informasi.

Penelitian ini akan menerapkan solusi web inovatif dengan teknologi handal bencana akan diterapkan menggunakan aplikasi web progressif (*Progressive Web App*) dengan arsitektur Komputasi Kabut (*Fog Computing*). Aplikasi web progressif memungkinkan sistem web bekerja secara cepat dan handal dengan kemampuan mengelola file-file dan data secara *offline* dengan teknologi *service worker* dibandingkan dengan *web* konvensional. Kemudian arsitektur komputasi kabut (*Fog Computing*) merupakan konsep yang diterapkan dengan meningkatkan komputasi pada level jaringan paling bawah, yaitu komponen yang paling dekat dengan sensor (contoh: router, switch). Dari hasil penelitian sebelumnya sistem mampu menangani 600 request secara bersamaan dengan waktu akses rata-rata 2448 namun performansi sistem akan menurun. (Muh. Zulfadli, 2019)

Dengan memadukan teknologi aplikasi web progressif (*Progressive Web App*) dan arsitektur komputasi kabut (*fog computing*) sebuah website yang dibuat memiliki kemampuan yang cepat dan dapat diakses ketika koneksi internet tidak stabil ataupun sedang *offline*. Teknologi ini dapat dimanfaatkan untuk menunjang proses sistem informasi yang ada pada industri kecil dan menengah pada keadaan konektifitas jaringan tidak handal yang diakibatkan oleh bencana alam. Solusi web inovatif diharapkan dapat terus aktif dan bekerja dengan baik pada setiap kondisi, seperti bandwith internet yang rendah dan konektifitas jaringan yang tidak stabil yang rentan terjadi pasca bencana.

Berdasarkan latar belakang di atas, penulis kemudian mengangkat sebuah penelitian dengan judul " **PERANCANGAN APLIKASI YANG HANDAL BENCANA DENGAN PROGRESSIVE WEB APP MENGGUNAKAN *REACT JS* DAN *COUCHDB* STUDI KASUS : INDUSTRI KECIL DAN MENENGAH** ".

1.2. Rumusan Masalah

1. Bagaimana membuat aplikasi berbasis website *Progressive Web App* untuk Industri Kecil dan Menengah yang dapat bekerja secara *offline* dan diterapkan pada arsitektur komputasi kabut ?
2. Bagaimana mengukur tingkat kepuasan pengguna terhadap website sistem informasi Industri Kecil dan Menengah ?

1.3. Tujuan Penelitian

1. Untuk membuat aplikasi berbasis *website Progressive Web App* untuk Industri Kecil dan Menengah yang dapat bekerja secara *offline* dan diterapkan pada arsitektur komputasi kabut.
2. Mengukur tingkat kepuasan pengguna terhadap *website* Sistem Informasi Industri Kecil dan Menengah.

1.4. Manfaat Penelitian

Manfaat yang diharapkan dalam penelitian ini adalah :

1. Bagi Masyarakat : Membantu para karyawan industri kecil dan menengah dalam meningkatkan efisiensi dalam menginput penjualan dan pengeluaran harian dengan aplikasi berbasis *website*.
2. Bagi Peneliti : diharapkan dapat menambah wawasan mengenai teknologi Aplikasi Web Progressif dan Arsitektur Komputasi Kabut.
3. Bagi Pendidikan : Mampu menjadi bahan referensi mengenai Aplikasi berbasis *website Progressive Web App*.

1.5. Batasan Masalah

1. Sistem difokuskan pada manajemen data divisi keuangan pada industri kecil dan menengah dan ditujukan untuk pihak internal Industri Kecil dan Menengah.
2. Pembuatan sistem ini menggunakan library React Js.
3. *Database* yang digunakan adalah CouchDB.
4. Menggunakan teknologi *Progressive Web Apps* (PWA) dan arsitektur komputasi kabut yaitu *Raspberry pi*.

5. Keberhasilan sistem divalidasi berupa pengujian konsep *Progressive Web Apps* (PWA) dengan *lighthouse,* pengujian fungsional, dan survey.

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 Bencana Alam

Berdasarkan Undang-undang nomor 24 tahun 2007 tentang penanggulangan bencana, didefinisikan bahwa bencana sebagai peristiwa atau rangkaian peristiwa yang mengancam dan mengganggu kehidupan dan penghidupan masyarakat yang disebabkan, baik oleh faktor alam dan/atau faktor non alam maupun faktor manusia sehingga mengakibatkan timbulnya korban jiwa manusia, kerusakan lingkungan, kerugian harta benda, dan dampak psikologis. Berdasarkan definisi tersebut, ada beberapa hal yang diperhatikan yakni bencana merupakan peristiwa yang mengakibatkan timbulnya korban jiwa manusia, kerusakan lingkungan, kerugian harta benda, dan dampak psikologis. Hal lain yakni bencana dapat ditimbulkan oleh faktor alam, faktor non alam, dan faktor manusia.

Dalam UU No. 24 Tahun 2007 dinyatakan ada tiga jenis bencana, yaitu meliputi bencana alam, bencana non alam, dan bencana sosial. Bencana alam adalah bencana yang diakibatkan oleh peristiwa atau serangkaian peristiwa yang disebabkan oleh alam antara lain berupa gempa bumi, tsunami, gunung meletus, banjir, kekeringan, angin topan, dan tanah longsor. Bencana non alam adalah bencana yang diakibatkan oleh peristiwa atau rangkaian peristiwa non alam yang antara lain berupa gagal teknologi, gagal modernisasi, epidemi, dan wabah penyakit. Bencana sosial adalah bencana yang diakibatkan oleh peristiwa atau serangkaian peristiwa yang diakibatkan oleh manusia yang meliputi konflik sosial antar kelompok atau antar komunitas masyarakat, dan teror.

Letak Geografis Indonesia yang berada di *Ring of Fire* membuat Indonesia rentan terhadap bencana alam . *Ring of Fire* atau deret sirkum pasifik merupakan jalur rangkaian gunung aktif di dunia. Pada tahun 2018, Indonesia dilanda tiga bencana langka yang memakan banyak korban jiwa . pertama, gempa bumi beruntun yang terjadi di Nusa Tenggara Barat. Kedua, Tsunami yang diikuti likuifaksi di Sulawesi Tengah dan ketiga, tsunami yang terjadi di selat Sunda akibat longsoran bawah laut dan erupsi gunung anak Krakatau. Badan Nasional Penanggulangan Bencana (BNPB) mencatat sepanjang 2018 telah terjadi 2.564 bencana. Jumlah tersebut lebih rendah dibandingkan 2017 yang mencapai 2.862 bencana. Namun dampak yang ditimbulkan bencana tahun 2018 lebih banyak menelan korban jiwa dari tahun sebelumnya. Bencana yang melanda sepanjang 2018 telah menyebabkan 3.349 orang meninggal dunia, 1.432 orang hilang. Sebanyak 21.064 orang mengalami luka-luka serta 10,2 juta orang harus mengungsi atau terdampak.

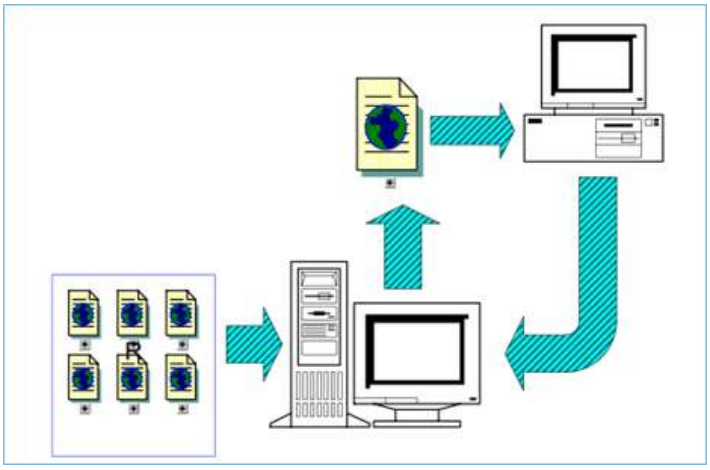
2.2 *Modern Website*

Website adalah halaman atau kumpulan beberapa halaman yang berisi tampilan berupa teks, gambar, animasi, audio, video, dan atau gabungan dari semuanya, baik yang bersifat statis maupun dinamis. Pada umumnya website terhubung pada sebuah alamat penunjuk yang spesifik yang dinamakan domain. Sebuah *website* biasanya bisa diakses secara umum. *Website* dapat diakses melalui *public Internet Protocol* (IP) dalam sebuah jaringan internet. Namun, adapula *website* yang dirancang untuk bisa diakses secara *private* dalam sebuah organisasi tertentu yang biasanya diakses secara *offline* melalui jaringan LAN.

Saat ini, ada berbagai cara dalam membuat dan mengembangkan sebuah aplikasi berbasis *website*. Namun setiap pengembangan web dimulai terlebih dahulu dengan memahami arsitektur web untuk menentukan lebih lanjut pengembangan tool dan service. Dari segi arsitektur, *website* dapat dibedakan menjadi *website* statis dan dinamis.

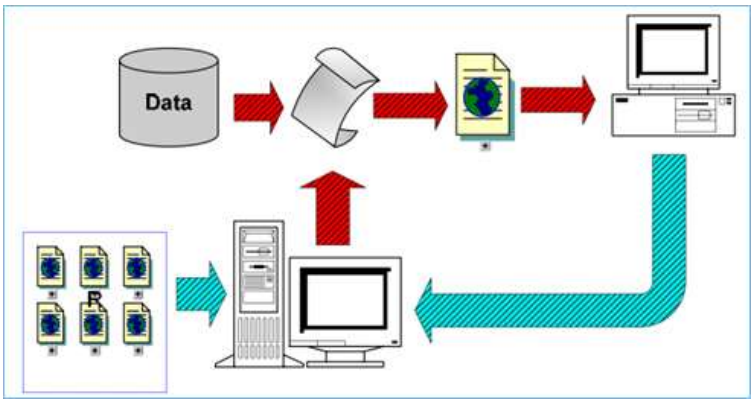
Web statis adalah web yang paling sederhana karena menampilkan informasi yang bersifat tetap (statis) karena pengguna tidak berinteraksi dengan *website* tersebut. Satu-satunya bentuk interaktivitas di web tersebut adalah *hyperlink*. Halaman web tidak memiliki database. Data dan informasi yang ada pada *website* statis tidak berubah-ubah kecuali diubah pada kode programnya. *Website* statis umumnya dikembangkan dengan menggunakan bahasa pemrograman *Hypertext Markup Language* (HTML) ataupun *Cascading Style Sheets* (CSS).

Web dinamis adalah web yang menampilkan informasi serta dapat berinteraksi dengan penggunanya. User bisa mengubah konten dari halaman tertentu dengan menggunakan form dalam browser. *Website* dinamis mempunyai arus informasi dua arah, yakni berasal dari user dan pengembang *website*. Jika pada static *website* kebanyakan diatur menggunakan HTML dan CSS, maka pada dynamic *website* dapat dikembangkan menggunakan bahasa pemrograman seperti PHP, Javascript, Python dan lain sebagainya. Situs web dinamis berisi konten yang mengakses dari basis data atau Sistem Manajemen Konten (CMS), sehingga ketika mengubah atau memperbaharui basis data, konten web akan berubah atau diperbaharui. Pada gambar 2.1 menampilkan arsitektur web statis.



Gambar 2.1 *Arsitektur Web Statis*

Dalam web statis, ketika web server menerima permintaan untuk halaman statis, server membaca permintaan, menemukan halaman, dan mengirimkannya ke browser pengguna. Gambar 2.2 menampilkan arsitektur *website* dinamis



Gambar 2.2 *Arsitektur Web Dinamis*

Dari kedua arsitektur tersebut, tentunya memiliki kelebihan dan kekurangannya masing-masing. Kelebihan yang dimiliki web statis ialah proses pengembangan yang dilakukan lebih cepat dan sederhana. Namun, pengembang harus melakukan perubahan kode baik HTML maupun CSS setiap konten mengalami perubahan. Berbeda dengan *website* dinamis yang memiliki kelebihan

yakni kemudahan dalam melakukan perubahan konten karena bekerja dengan data yang dinamis, namun memerlukan lebih dari satu pengembang dalam melakukan pengembangan web agar web yang dibuat lebih berjalan dengan fungsi yang semestinya dan memiliki desain yang menarik.

Dalam pengembangan *website* pada saat ini, para pengembang memiliki cara yang lebih mudah yakni menggunakan *framework. Framework* menyediakan manajemen *session*, penyimpanan data, dan melakukan *templating* sistem serta manajemen pengerjaan *web* yaitu dengan memisahkan pengerjaan menjadi dua sisi yaitu sisi *front-end* dan *back-end*. Dari sisi *front-end framework,* mempermudah dalam pengembangan *website* dalam proses desain dan masukan dari berbagai fitur yang akan dibuat pada *website*, termasuk fitur menu, dan model yang ada. *Framework front-end* dibagi menjadi dua yaitu Javascript *Framework* (Angular, ReactJs, MeterJS) dan CSS *Framework* (Boostrap,SemanticUI, UIkit, MaterialUI). Adapun dari sisi *back-end framework* mempermudah dalam melakukan penulisan dan pemeliharaan aplikasi web. Framework menyediakan pustaka sederhana yang digunakan dalam membangun 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.

2.3 *Progressive Web App*

Progressive Web App (PWA) adalah konsep pengembangan aplikasi berbasis web yang mencakup penerapan teknologi terbaru dari browser seperti *service worker* dan *app manifest*. Konsep PWA dapat memberikan pengalaman terbaik

dalam menggunakan suatu aplikasi web, aplikasi akan menjadi semakin powerfull. Aplikasi dapat dimuat dengan cepat, bahkan dalam kondisi internet yang kurang baik, bisa mengirim *push notification*, memiliki ikon aplikasi di *home screen*, dan bisa berjalan dalam mode layer penuh. PWA memiliki karakteristik utama dapat diandalkan (*reliable*), cepat (*fast*), dan menarik (*engaging*), memastikan pengguna mendapatkan pengalaman terbaik dalam menggunakan suatu aplikasi web walaupun dalam koneksi internet yang minim atau *offline* sekalipun. PWA sepenuhnya mengandalkan *browser* pengguna dan teknologi yang ada di dalamnya. Sampai saat studi ini ditulis, sudah 90,33% dari seluruh *browser* mendukung fitur *service worker*, seperti Mozilla Firefox, Microsoft Edge, Google Chrome, Safari, iOS Safari, Chrome for Android, UC Browser for Android, dan Samsung Internet.(Lauriensius Adi et all. 2017)

Menurut situs resmi google developer karakteristik dari progressive web app adalah :

- Progressive – dapat digunakan oleh semua pengguna, terlepas dari browser apa yang digunakan karena aplikasi dikembangkan secara progressive.
- Responsive – dapat digunakan pada semua perangkat mulai dari desktop, tablet, smartphone, dan lainnya.
- Connectivity Independent – memiliki *service workers* untuk dapat diakses secara *offline* atau pada kualitas jaringan internet yang rendah.
- App-like – terasa seperti aplikasi karena model aplikasi shell memisahkan fungsi dari konten aplikasi.

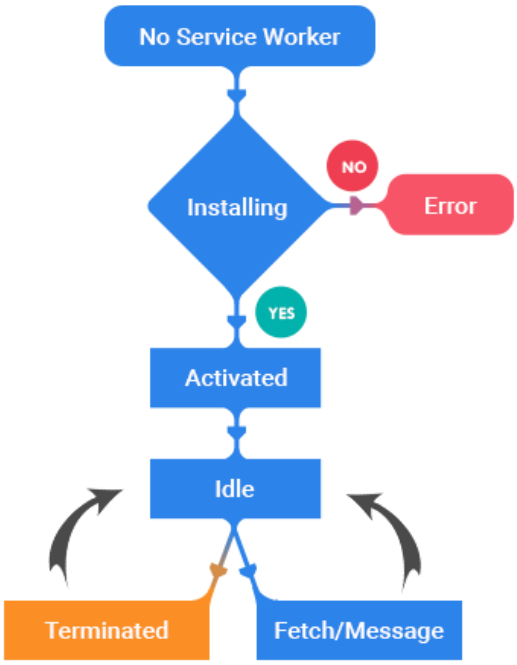
- Fresh – selalu up-to-date dikarenakan update proses dari *service worker*.
- Safe – dilayani via HTTPS untuk mencegah pengintaian dan untuk memastikan konten tidak rusak.
- Discoverable - diidentifikasi sebagai aplikasi berkat W3C *manifest* dan *service worker registration scope* sehingga mudah ditemukan oleh search engine
- Re-engageable – dapat dengan mudah di engage dengan fitur yang ada seperti push notification, dan lain-lain.
- Installable – memungkinkan user untuk menambahkan aplikasi ke home screen tanpa melalui app store.
- Linkable – dapat berbagi dengan mudah melalui URL dan tidak memerlukan instalasi.

2.3.1 *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 browser*, menangani *network request*, *caching*, mengembalikan *resource* dari *cache*, dan bisa mengirimkan *push message*. Asset *web* dapat disimpan sebagai *cache* lokal, sehingga dengan jaringan internet yang kurang memadai pun, pengguna tetap dapat mendapat pengalaman yang baik. Aplikasi dapat menjalankan halaman web yang sudah di-*cache* atau memberikan status koneksi

tanpa browser menampilkan tulisan *error* karena ketiadaan koneksi internet. (Muh. Zulfadli, 2019)

Pada gambar 2.3 dapat dilihat proses pemasangan *service worker*. Untuk memasang *service worker* pada laman, kita harus meregistrasinya dengan dengan menggunakan Javascript yang ada di halaman web. Setelah registrasi, *browser* akan memulai tahap *install service worker* di latar. Setelah aktif, *service worker* akan menangani semua halaman di bawah *scope* dimana *service worker* di-*install*, kemudian ada dua kemungkinan state: *terminated* untk menghemat memori, atau menangani *fetch* ketika ada *network request* dari halaman web.



Gambar 2.3 *Lifecycle Service Worker*

Service worker bisa menangani berbagai jenis *request*, tetapi yang disimpan ke dalam *cache* hanya *request* jenis GET. Pada *service worker*, pengguna bisa

mengaplikasikan strategi terbaik untuk *caching* konten dinamis, dan ada banyak yang mempengaruhi strategi *caching.*

2.3.2 Web Caching

Saat menggunakan aplikasi web atau menelusuri informasi dari web, ada dua hal yang sangat penting yang harus diperhatikan yakni kecepatan dan efisiensi. Pengguna akan puas jika dapat menggunakan aplikasi berkecepatan tinggi. Dalam merancang suatu aplikasi web, *caching* memungkinkan untuk meningkatkan akses informasi yang diperlukan karena mengurangi *network traffic.*

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 dikirim ke server, maka akan banyak sumber daya yang 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.

2.3.3 Web App Manifest

Web App Manifest adalah file JSON sederhana yang memberi tahu browser tentang aplikasi web pengguna dan bagaimana seharusnya aplikasi bekerja ketika "diinstal" pada perangkat seluler atau desktop pengguna. Untuk menampilkan permintaan "Tambahkan ke Layar Beranda" ketika aplikasi web dibuka maka diperlukan *manifest* oleh Chrome. *Web app manifest* mampu menyimpan *bookmark* situs ke layar beranda perangkat. Ketika sebuah situs dijalankan dengan

cara ini maka situs tersebut memiliki ikon dan nama yang unik sehingga pengguna dapat membedakannya dari situs lain. *Web app manifest* juga digunakan untuk membuat ikon di layar beranda, dan memuat *background* untuk aplikasi serta memuat *url* untuk menjalankan suatu aplikasi.

2.3.4 Application Shell

Application Shell adalah salah satu cara untuk membangun *Progressive Web App* yang andal dan dapat langsung dimuat pada layar pengguna, serupa yang dilihat pada tampilan aplikasi asli. Aplikasi "shell" ialah HTML, CSS, dan Javascript minimal hal yang diperlukan untuk menjalankan antarmuka pengguna dan bila di-*cache offline* bisa menjamin kinerja yang bagus, instan, dan bisa diandalkan pada saat pengguna membuka kembali laman tersebut. Dalam arsitektur *Application Shell* , shell disediakan oleh *service worker*. Hal ini di-*cache* oleh *service worker* dari sumbernya melalui permintaan API. Dengan adanya *App Shell*, konten dan UI aplikasi akan di-*cache* secara terpisah, sehingga meningkatkan persepsi pengguna tentang kinerja dan kegunaan aplikasi.

2.4 React JS

React JS adalah sebuah library yang dikembangkan oleh Facebook untuk memfasilitasi pembuatan komponen UI yang interaktif, stateful dan dapat digunakan kembali. React JS sangat baik untuk *rendering interface* pengguna yang kompleks dengan kinerja tinggi. Dasar di balik React adalah konsep Virtual DOM. React JS secara efektif menggunakan Virtual DOM yang dapat digunakan baik dari sisi klien atau sisi *server* dan berkomunikasi bolak-balik. (Kumar, 2016).

React dikembangkan untuk menyediakan kecepatan dan efisien, karena react hanya perlu me-*render resource* yang berhubungan dengan data yang diganti tanpa perlu me-*render seluruh* resource. React bersifat *Reusable* yang berarti dapat digunakan berulang kali sehingga sangat berguna untuk mempersingkat waktu dan mengurangi *resource* yang ada.

2.4.1. JSX

JSX adalah ekstensi *sintaks* Javascript yang terlihat mirip XML. JSX sintaks HTML atau XML yang digunakan pada React JS memperluas ECMA Script sehingga XML/HTML dapat bekerja pada kode-kode Javascript atau React. JSX bersifat opsional dan tidak harus digunakan pada React. Namun, JSX merupakan bantuan visual yang baik ketika bekerja dengan UI di dalam Javascript yang memungkinkan React untuk menampilkan pesan kesalahan dan peringatan yang lebih berguna. (Tung, 2018)

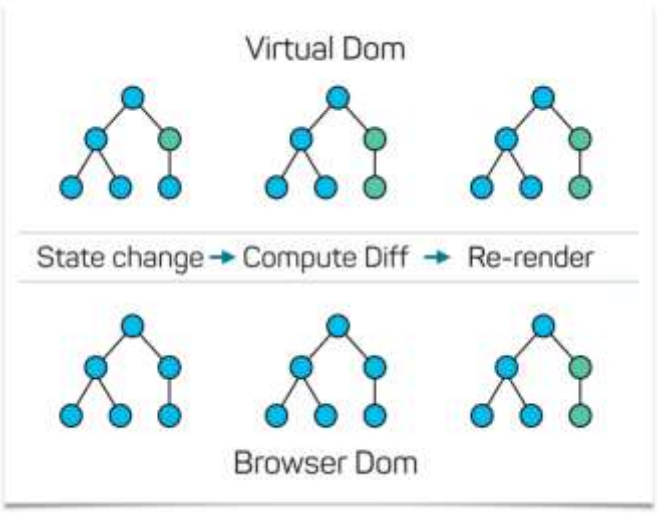
2.4.2. *Stateful Components*

React memungkinkan pengguna untuk membagi dua antarmuka menjadi independent, dan dapat digunakan berkali-kali dengan melakukan pemanggilan komponen react yang menggunakan implementasi metode *render* dengan 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 dengan menggunakan API React. (Muh. Zulfadli, 2019)

State adalah objek javascript sederhana yang digunakan untuk merekam dan memberi reaksi sesuai perintah pengguna. Setiap kelas berbasis komponen mendefinisikan objek statenya 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. *Virtual Document Object Model*

Dari situs resmi reactjs.org, Virtual DOM adalah sebuah konsep dalam pemrograman dimana representasi ideal atau "virtual" dari antarmuka pengguna disimpan dalam memori dan disinkronkan dengan "DOM" yang sebenarnya oleh library seperti ReactDOM. Proses ini disebut *reconciliation* yaitu menggunakan algoritma yang membandingkan dan membedakan perubahan untuk mengetahui elemen apa yang diperbaharui. Gambar 2.4 menunjukkan perbandingan Virtual DOM dan Browser "DOM" dalam berbagai tahap perubahan status.



Gambar 2.4 Perbandingan Virtual DOM dengan Browser DOM

2.5 Node JS

Node Js adalah perangkat lunak yang didesain untuk mengembangkan aplikasi berbasis web yang disebut juga sebagai *runtime environment.* Node js ditulis dalam bahasa Javascript dan berjalan dengan basis event (*event drive*) dan *asynchronous* I/O. Tidak sperti kebanyakan Javascript yang berjalan di sisi *client/browser* saja, node js dieksekusi sebagai aplikasi server yang didukung oleh V8 Engine buatan google dan beberapa modul bawaan lain yang terintegrasi. (Fajrin, 2017)

Node Js dapat menangani ribuan koneksi yang bersamaan dengan penggunaan *resource* minimum di setiap prosesnya sehingga dapat diandalkan untuk membuat aplikasi *real-time*. Penggunaan Javascript di sisi server dan juga di sisi client meminimalisir ketidakcocokan antar dua sisi pemrograman, terkait komunikasi data yang mana menggunakan struktur JSON yang sama di kedua sisi, validasi form yang sama yang dapat dijalankan di server dan client. Database NoSQL seperti MongoDB dan CouchDB mendukung langsung Javascript sehingga interfacing dengan database ini akan jauh lebih mudah.

2.6 NoSQL Database

Database NoSQL adalah database yang tidak memiliki perintah SQL dan konsep penyimpanannya semistruktural atau tidak struktural dan tidak harus memiliki relasi layaknya tabel-tabel SQL. Database NoSQL dibuat dengan tujuan khusus untuk model data spesifik dan memiliki skema fleksibel untuk membuat aplikasi modern. Database NoSQL menggunakan berbagai model data untuk mengakses dan mengelola data, seperti dokumen, grafik, dan nilai kunci, dalam memori, dan pencarian.

NoSQL memiliki karakteristik BASE (*Basically, Available, Soft State, and Eventual Consistency*) yang merupakan kebalikan dari ACID pada basis data SQL. Setelah transaksi yang konsisten , keadaan (state) yang didapat adalah keadaan sementara (*soft state*) bukan keadaan tetap (*solid state*). Fokus utama dari BASE adalah ketersediaan permanen. Basis data NoSQL tidak menggunakan model data rasional, dapat menyimpan data dalam ukuran besar, dan juga memperbolehkan data untuk disimpan di dalam record yang tidak mempunyai skema tertentu. Basis data NoSQL dibagi dalam beberapa jenis basis data: 1) Basis data *Key-Value* adalah kombinasi antara *key* dan *value* yang merupakan basis data inti dari semua basis data NoSQL. 2) Basis data *Document Store* adalah basis data yang menggunakan secord sebagai dokumen. Record menyimpan dokumen tidak terstruktur (*unstructured*) atau semi terstruktur (*semi structured documents*). Setiap dokumen terdiri dari satu set key dan value, hampir sama dengan basis data *Key-Value.* 3) Basis data berbentuk kolom adalah basis data yang berorientasi kolom. Terdiri dari 2 jenis basis data: a) Tempat penyimpan data dengan kolom lebar (*wide-column data store*) dan Basis data berorientasi kolom. 4) Basis data grafis terdiri dari *node, properties* (karakteristik), dan *edge.* (Suliyanti, 2019)

Basis data NoSQL memiliki kelebihan dapat memproses data terstruktur, semi terstruktur dan tidak terstruktur dalam jumlah besar dengan kecepatan tinggi, menggunakan sekma yang fleksibel dan basis data terdistribusi (*distributed database*). Namun basis data NoSQL juga memiliki kelemahan yaitu untuk memproses operasi yang kompleks membutuhkan yang lama dan tidak menemukan

dukungan untuk konsistensi. Selain itu basis data NoSQL tidak dapat memproses *subqueris, joins,* dan *grouping/aggregation.* (Suliyanti, 2019)

Teknologi NoSQL memiliki keunggulan sendiri dalam melakukan manajemen basis data sehingga penggunaan database ini lebih diminati leh pengembang. Ada beberapa jenis basis data NoSQl yang mampu untuk membuat sebuah media penyimpanan *offline* bagi sistem yang bekerja secara *online* maupun *offline*, yaitu seperti IndexedDB, RXDB, CouchDB, dan PouchDB.

2.6.1. IndexedDB

Indexed Database API (biasa disebut IndexedDB) adalah Javascript *Application Programming Interface* (API) yang disediakan oleh web browser untuk mengelola database NoSQL dari objek JSON untuk penyimpanan lokal sisi klien yang mampu menyimpan data terstruktur. IndexedDB dapat digunakan melalui Javascript dalam hal menyimpan, mengambil, menghapus, dan melakukan pembaharuan data.

IndexedDB termasuk dalam tipe *Key-Value Store* yang menyimpan objek dari struktur yang berbeda, semuanya diidentifikasi oleh sebuah *key* unik yang digunakan sebagai indeks untuk memfasilitasi pengambilan objek tertentu. IndexedDB merupakan database berorientasi objek. Mirip dengan database relasional dimana sebagian besar operasi database terkonsentrasi pada sekitar tabel, namun di IndexedDB konsentrasinya ialah di sekitar objek store. IndexedDB dibangun pada model basis data transaksional yang merupakan fitur kuat dari IndexedDB, selama berjalan pada browser sebagai database lokal. Sebagian besar

browser saat ini mendukung multi-tab, membuka *instance* lain dari database di tab lain dengan tidak adanya manajemen transaksi yang dapat menyebabkan aplikasi *crash*. (Al-Shaikh dan Azzam, 2017).

2.6.2. RxDB

RxDB (*Reactive Database*) adalah database NoSQL untuk aplikasi Javascript seperti situs web, Aplikasi Hybrid, dan NodeJs. Reactive berarti bukan hanya aktif pada query state sebelumnya, tetapi untuk semua perubahan state seperti hasil dari *query*, meskipun itu document field. RxDB dapat melakukan replikasi realtime dengan titik akhir yang sesuai dengan CouchDB dan juga dengan GraphQl *endpoints.*

RxDB adalah *object store* yang memiliki kinerja yang cepat dibangun diatas SQLlite (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.

2.6.3. CouchDB

CouchDB merupakan salah satu basis data NoSQL berbasis dokumen yang masuk dalam pembinaan Apache Foundation. Bersama dengan Cassandra, Hadoop, Tomcat, Lucene. CouchDB termasuk dalam dunia open source. CouchDB dibangun menggunakan bahasa pemrograman Erlang yang mengandalkan pada realibilitas 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)

CouchDB memiliki arsitektur database NoSQL yang berorientasi dokumen dan diimplementasikan dalam bahasa yang berorientasi konkurensi, menggunakan JSON untuk menyimpan data. Tidak seperti database relasional, database CouchDB tidak menyimpan data dan hubungan dalam tabel. Sebaliknya, setiap basis data adalah kumpulan dokumen independent. Setiap dokumen menyimpan data sendiri dan skema mandiri. CouchDB mengimplementasikan suatu bentuk control konkurensi multiversion sehingga tidak mengunci file basis data selama penulisan. Konflik diserahkan kepada aplikasi untuk diselesaikan. Menyelesaikan konflik umumnya melibatkan penggabungan data pertama ke dalam salah satu dokumen, kemudian menghapus yang basi.

Salah satu kelebihan dari CouchDB yakni memiliki arsitektur terdistribusi dengan replikasi. CouchDB dirancang dengan replikasi dua arah (atau sinkronisasi) dan operasi *offline*. Itu berarti beberapa replika dapat memiliki salinannya sendiri dari data yang sama, memodifikasinya, dan kemudian menyinkronkan perubahan itu di lain waktu. Karena hal itu, CouchDB dapat mereplikasi ke perangkat yang data *offline* dan menangani sinkronisasi data untuk user ketika perangkat kembali *online*.

2.6.4. Pouch DB

Pouch DB adalah basis data bersifat *open source* dibangun menggunakan javascript yang merupakan pengembangan dari CouchDB yang dirancang untuk berjalan dengan baik di browser. Pouch DB dibuat untuk membantu pengembang web membangun aplikasi yang berfungsi baik dalam keadaan *online* maupun *offline*. Pouch DB juga memakai protokol replikasi yang sama dengan CouchDB,

yang membuat data lokal dapat langsung tereplikasi satu arah, dua arah dengan data di server lain atau *cloud.*

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 disinkronkan menggunakan Couch DB (Colanus dkk, 2017).

2.7 Fog Computing

Fog Computing adalah paradigma dengan kemampuan terbatas seperti komputasi, penyimpanan, dan layanan jaringan secara terdistribusi antara berbagai *end device* komputasi awan klasik. Shanhe Yi et al telah menyatakan definisi umum komputasi kabut ialah "Fog Computing adalah arsitektur komputasi terdistribusi secara geografis dengan kumpulan sumber daya yang terdiri dari satu atau lebih perangkat heterogen yang terhubung dimana-mana (termasuk perangkat tepi) di tepi jaringan dan tidak secara eksklusif didukung mulus oleh layanan *cloud*, untuk secara kolaboratif memberikan perhitungan elastis, penyimpanan, dan komunikasi (dan banyak layanan dan tugas baru lainnya) di lingkungan yang terisolasi untuk sejumlah besar klien di dekatnya.

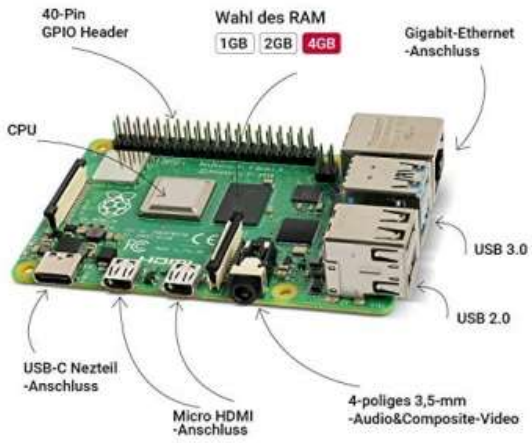
Fog Computing menyediakan *low latency* dan *location awareness*, memiliki distribusi geografis yang luas, mendukung mobilitas, terganggu karena sejumlah besar simpul. Tugas utama *fo*g (kabut) adalah mengirimkan data dan menempatkannya lebih dekat dengan pengguna yang diposisikan di lokasi yang berada di tepi jaringan. Istilah *edge* mengacu pada berbagai node yang terhubung

dengan *end user* yang disebut *edge computing*. Jika dilihat menurut arsitektur, kabut (fog) terletak di bawah cloud. Istilah *Fog Computing* diberikan oleh CISCO sebagai teknologi baru dimana perangkat seluler berinteraksi satu sama lain dan mendukung komunikasi data dalam *Internet of Things*. (Shenoy, 2013)

2.7.1. Raspberry Pi

Raspberry pi adalah komputer berukuran kecil yang mempunyai kinerja lebih rendah dari PC Dekstop yang memang didesain untuk melakukan pekerjaan yang lebih ringan. Raspberry pi ini mampu bekerja layaknya komputer pada umumnya dengan kemampuan untuk menjalankan sistem operasi linux dan aplikasi lainnya seperti *Libre Office, multimedia* (audio dan video), perambatan *web*, ataupun *programming.* Raspberry Pi dibuat di Inggris oleh *Raspberry Pi Foundation* pada awalnya Raspberry Pi ditunjukkan untuk modul pembelajaran ilmu komputer. *Raspbian* adalah sistem operasi bebas berbasis Debian dioptimalkan untuk perangkat keras *Raspberry pi*. (Permana, 2014)

Raspberry Pi merupakan perangkat *embedded system* dalam jenis *single board computer*. Raspberry Pi memiliki sistem pada *chip Broadcom bcm2835* dengan prosesor ARM1176JZF-S 700 MHz. Raspberry Pi dapat diinstal sistem operasi yang *support* dengan teknologi ARM seperti Raspbian OS, Arch Linux. Raspberry Pi bisa menjadi beberapa perangkat yang bisa digunakan untuk beberapa kegunaan antara lain sebagai file server, raspberry Pi bisa difungsikan sebagai *file server* dengan pengaturan yang tepat. Bisa memanfaatkan aneka *hard disk eksternal* (yang dihubungkan ke Raspberry Pi via port USB) untuk menjadi media penyimpanan data



Gambar 2.5 Board Raspberry Pi 4 Model B

Board Raspberry Pi sendiri sudah memiliki beberapa versi, mulai Raspberry Pi model B, Raspberry Pi model A+, Raspberry Pi 2 model B, Raspberry Pi 3 model B, Raspberry Pi 3 model B+, Raspberry Pi Zero, dan yang terbaru adalah Raspberry Pi 4 model B. Versi terbaru dari Raspberry Pi memiliki desain dan dimensi yang ralatif identik dengan versi pendahulunya, hanya saja Raspberry Pi 4 telah menggunakan prosesor baru yaitu Broadcom BCM2711B0. Prosesor ini menggunakan CPU 64-bit Quad-Core ARM Cortex-A72 dengan clockspeed 1.5 GHz.

2.8 Nginx

Nginx adalah server HTTP dan Proxy yang bersifat *open source.* Nginx secara resmi pertama kali diperkenalkan pada tahun 2004. Nginx menjalankan arsitektur yang *event-driven* dan asinkron. Hal ini menunjukkan bahwa thread yang sama atau serupa dikelola dibawah satu *worker process* terdiri atas unit yang lebih kecil dan disebut *worker connection.* Keseluruhan unit bertugas untuk menangani

request thread. *Worker connection* mengirimkan permintaan kepada *worker process*, yang juga dikirimkan ke *master process.* *Master process* kemudian menampilkan hasil dari permintaan atau *request* tersebut.

Maximun request merupakan batas atas permintaan simultan yang bisa dijalani oleh tiap-tiap aplikasi web server. Untuk halaman-halaman statis, nginx menduduki urutan pertama dengan *maximum request* sebesar 5300 dan dapat melayani permintaan-permintaan tersebut (*i.e. reply time*) rata-rata dalam 1.26 detik. sedangkan lightppd menduduki urutan kedua dengan *maximum request* sebesar 3000 tetapi dengan *reply time* rata-rata yang sangat cepat yaitu 0.18 detik. dan urutan terakhir diduduki oleh apache dengan *maximum request* terkecil yaitu 2900 dan *reply time* rata-rata yang relatif besar yaitu 2.85 detik. (Dawood, Qiana, & Muchallil, 2014)

Untuk halaman-halaman dinamis Nginx tetap memberikan *maximum request* terbesar yaitu 3000 dengan reply time 1,07 detik. Apache menduduki urutan kedua dengan *maximum request* sebesar 400 tetapi dengan reply time yang sangat besar yaitu 8,70 detik. sedangkan *lightppd* menduduki urutan terakhir dengan *maximum request* terkecil yaitu sebesar 200 dan dengan *reply time* yang relatif lambat yaitu 2,24 detik. (Dawood, Qiana, & Muchallil, 2014)

2.9 *Lighthouse*

Lighthouse adalah aplikasi *open source* yang dibangun oleh Google. *Lighthouse* dapat dijalankan sebagai Ekstensi Chrome, dari baris perintah, atau digunakan secara terprogram sebagai modul node. *Lighthouse* berfokus pada

kinerja dan kualitas PWA. Sebuah *url* diberikan pada *lighthouse* yang kemudian menjalankan serangkaian tes terhadap halaman, dan kemudian menghasilkan laporan tentang seberapa baik halaman tersebut.

Pengguna dapat menjalankan *Lighthouse* di Chrome DevTools, dari baris perintah, atau sebagai modul node. Dengan memberikan *url* untuk diaudit, *Ligthouse* akan menjalankan serangkaian audit terhadap halaman, dan menghasilkan serangkaian laporan tentang seberapa baik halaman itu. Dari sana, gunakan audit yang gagal sebagai indikator tentang cara meningkatkan halaman. Setiap audit memiliki dokumen rujukan yang menjelaskan mengapa audit itu penting, serta cara memperbaikinya.