IMPLEMENTASI TEKNOLOGI *WEB REAL-TIME COMMUNICATION* (WEBRTC) DALAM PERANCANGAN KELAS VIRTUAL SEBAGAI SARANA PEMBELAJARAN INTERAKTIF DENGAN MEDIA SERVER KURENTO

TUGAS AKHIR

Disusun dalam rangka memenuhi salah satu persyaratan
Untuk menyelesaikan program Strata-1 Departemen Teknik Informatika
Fakultas Teknik Universitas Hasanuddin
Makassar

Disusun Oleh :

HERMAWAN SAFRIN

D42114314

DEPARTEMEN TEKNIK INFORMATIKA
FAKULTAS TEKNIK
UNIVERSITAS HASANUDDIN
MAKASSAR
2020

LEMBAR PENGESAHAN SKRIPSI

"Implementasi Teknologi *Web Real-Time Communication* (WebRTC) Dalam Perancangan Kelas Virtual Sebagai Sarana Pembelajaran Interaktif Dengan Media Server Kurento"

OLEH :

HERMAWAN SAFRIN

D421 14 314

Skripsi ini telah dipertahankan pada Ujian Akhir Sarjana tanggal 21 Agustus 2020. Diterima dan disahkan sebagai salah satu syarat memperoleh gelar Sarjana Teknik (S.T) pada Program Studi Strata-1 Teknik Informatika Fakultas Teknik Universitas Hasanuddin.

Makassar, 2 November 2020

Disetujui Oleh :

Pembimbing I,	Pembimbing II,
Dr.Eng. Muhammad Niswar, ST., M.IT	Dr. Amil Ahmad Ilham, S.T., M.IT
Nip. 19730922 199903 1 001	Nip. 19731010 199802 1 001

Diterima dan disahkan oleh:
Ketua Departemen S1 Teknik Informatika

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

PERNYATAAN KEASLIAN

Yang bertanda tangan di bawah ini :

Nama : HERMAWAN SAFRIN

NIM : D421 14 314

Program Studi : S1 Teknik Informatika

Menyatakan dengan sebenar-benarnya bahwa skripsi yang berjudul :

IMPLEMENTASI TEKNOLOGI WEB REAL-TIME COMMUNICATION (WEBRTC) DALAM PERANCANGAN KELAS VIRTUAL SEBAGAI SARANA PEMBELAJARAN INTERAKTIF DENGAN MEDIA SERVER KURENO

Adalah karya ilmiah saya sendiri dan sepanjang pengetahuan saya di dalam naskah skripsi ini tidak terdapat karya ilmiah yang pernah diajukan/ditulis/diterbitkan sebelumnya, kecuali yang secara tertulis dikutip dalam naskah ini dan disebutkan dalam sumber kutipan dan daftar pustaka.

Apabila dikemudian hari ternyata di dalam naskah skripsi ini dapat terdapat unsur-unsur jiplakan, saya bersedia menerima sanksi atas pebuatan tersebut dan diproses sesuai dengan peraturan perundang-undangan yang berlaku (UU No. 20 Tahun 2000, pasal 25 ayat 2 dan pasal 70)

Makassar, 11 November 2020

Yang membuat Pernyataan

HERMAWAN SAFRIN

ABSTRAK

E-learning merupakan sebuah proses pembelajaran yang berbasis elektronik dimana media yang digunakan salah satunya dengan teknologi jaringan komputer. Dalam *e-learning* dikenal istilah *synchronous learning*, yaitu interaksi yang berorientasi pada pembelajaran dan difasilitasi dengan instruksi-instruksi secara langsung (*realtime*) dan biasanya terjadwal. *Web real-time communication* (*WebRTC*) adalah satu dari sekian banyak teknologi yang dapat digunakan untuk melaksanakan *synchronous learning*. Aplikasi *WebRTC* ini cukup populer karena memberikan kemudahan bagi para penggunanya dalam melakukan pertukaran data baik data berupa video, gambar, suara, dan teks. *WebRTC* bekerja secara *client-to-client* tanpa adanya *plugin* tambahan pada *browser*. Namun pada pengaplikasiannya sering kali teknologi *WebRTC* memiliki keterbatasan pada jumlah *client* dalam satu waktu, apalagi ketika *client* menjalankan fitur *video conference*. Hal ini dikarenakan setiap *client* yang terhubung dalam satu komunikasi akan memakan *resource* berupa *bandwidth*, *random access memory* (RAM), dan *central processing unit* (CPU) yang cukup tinggi seiring dengan jumlah peningkatan *client* yang terhubung. Tujuan dari penelitian ini adalah menghasilkan sistem *syncronous learning* dengan teknologi *WebRTC* yang menggunakan *Kurento Media Server* (KMS), sehingga kendala pada aplikasi *WebRTC* dapat diatasi. Hasil dari penelitian ini adalah terciptanya sebuah sistem yang menerapkan teknologi *WebRTC* berupa *video conference* (baik media berupa *webcam* maupun *screen sharing*), *file sharing*, dan *chatting*. Selanjutnya ketika sistem telah dibangun, akan dilakukan pengukuran terkait beban kinerja KMS saat melakukan *video conference*. Skenario uji yang digunakan adalah melakukan penambahan *client* satu demi satu pada aplikasi *WebRTC*. *Client* yang bertindak sebagai *viewers* berjumlah sembilan *client*. Beban kinerja yang akan diukur adalah terkait RAM, CPU, dan *bandwidth*. Selanjutnya, hasil dari pengukuran beban kinerja KMS saat melakukan *broadcasting* media berupa *webcam* menunjukkan angka yang lebih kecil dibandingkan dengan *broadcasting* media berupa *screen*. Untuk *broadcasting* media berupa *webcam* menunjukkan konsumsi CPU sebesar 2,7% - 7,7%, konsumsi RAM sebesar 61.870,08 *kilobytes* – 94.867,456 *kilobytes*, dan konsumsi *bandwidth* sebesar 234-1.170 Kbps untuk *bit send* dan 158 – 225 Kbps untuk *bit received*. Sedangkan untuk *broadcasting* media berupa *screen* menunjukkan CPU sebesar sebesar 3,7% - 10,0%, konsumsi RAM sebesar 94.867,456 *kilobytes* – 127.864,832 *kilobytes*, dan konsumsi *bandwidth* sebesar 325 – 1.990 Kbps untuk *bit send* dan 181 – 262 Kbps untuk *bit received*.

Kata kunci : Kurento Media Server, *WebRTC, Synchronous Learning*, Socket.io , NodeJS

KATA PENGANTAR

Assalamu'alaikum Warahmatullahi Wabarakatuh.

Segala puji dan syukur Penulis panjatkan ke hadirat Allah S.W.T, Tuhan Yang Maha Esa yang dengan limpahan rahmat dan hidayah-Nya sehingga tugas akhir dengan judul "*Implementasi Teknologi Web Real-time Communication (WebRTC) dalam Perancangan Kelas Virtual Sebagai Sarana Pembalajaran Interaktif dengan Media Server Kurento"* ini dapat diselesaikan sebagai salah satu syarat dalam menyelesaikan jenjang Strata-1 pada Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin. Selanjutnya *shalawat* serta salam Penulis panjatkan pula kepada sang revolusioner sejati, Nabi Muhammad S.A.W sebagai sosok tauladan Penulis yang membawa zaman dari zaman jahiliyah ke zaman yang *In Shaa Allah* terang benderang ini, *aamiin*.

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 ataupun internasional, buku, dan dari berbagai situs-situs terpercaya yang ada di internet.

Penulis menyadari bahwa tanpa bantuan dan bimbingan dari berbagai pihak, mulai dari masa perkuliahan sampai dengan 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) Allah S.W.T karena atas semua berkat, karunia serta pertolongan-Nya yang tiada batas, yang telah diberikan kepada Penulis disetiap langkah dalam pembuatan tugas akhir ini hingga penulisan laporan skripsi ini;
2) Kedua orang tua Penulis, Bapak Pelda La Hasi dan Ibu Wa Bida yang selalu mendidik Penulis dan menjadi tempat untuk berkeluh kesah serta selalu memberikan dukungan, doa dan semangat kepada Penulis sejak kecil. Terima kasih untuk selalu ada bagi Penulis;
3) Keluarga besar Penulis, terkhusus Bapak Pelda Sudin bersera Ibu, Bapak Hastun,S.Pd., Bapak Rahmadin Gafar,ST., beserta Ibu, Bapak Safar Gaffar,ST., Muhammad Darul Alamsyah, SS., Bapak Hamid , Dwi Randi Murhum, dan Waode Nurmin yang telah memberikan nasihat serta dukungan kepada Penulis;
4) Bapak Dr.Eng. Muhammad Niswar, ST., M.IT., selaku pembimbing I dan Bapak Dr. Amil Ahmad Ilham, ST., M.IT., selaku pembimbing II sekaligus Ketua Departemen Teknik Informatika yang selalu menyediakan waktu, tenaga, pikiran dan perhatian yang luar biasa untuk mengarahkan Penulis dalam penyusunan tugas akhir ini;
5) Bapak Robert, Bapak Zainuddin dan Ibu Santi serta segenap staff Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin yang telah membantu kelancaran penyelesaian tugas akhir Penulis;
6) Kanda Ir. Iqra Aswad, ST., MT., Abdul Hannan,ST., Firmansyah, S.KOM yang telah membantu saya dalam penelitian dan proses kerja praktek serta memberikan inspirasi terkait dunia kerja;

7) Saudara seperjuangan Penulis Teknik 2014 yang telah menemani perjalanan penulis dalam dunia pengaderan sebagai anak teknik;
8) Keluarga besar Rectifier'14 yang juga menjadi rekan sekaligus tempat berbagi keluh dan kesah selama mengarungi dunia kampus sebagai anak teknik;
9) Teman-teman Adhyaksa *Boys* khususnya Muh. Nur Alamsyah, Yakip, Syarif Hidayatullah, Rahmat Firman, Muh. Ardiansyah, Ahmad Setiadi, Jamaluddin, Arya Jaka Putra, Muh. Yusuf Ramadhan, S.Ked., Siswono Nasir, Ulfah Rojiyyah, Al Riefqi Dasmito, ST., Fadel Pratama, ST., Abdillah Satari, ST., dan Gilbert HG yang selalu menemani Penulis dalam mengarungi segala rintangan dalam dunia perkuliahan sekaligus rumah untuk kembali pulang;
10) Teman-teman pengurus Himpunan Mahasiswa Informatika Fakultas Teknik Universitas Hasanuddin Periode 2017/2018 dan Ketua-ketua lembaga tinggi se-OKFT-UH Periode 2017/2018 yang telah menemani saya dalam perjalanan organisasi;
11) Teman-teman keluarga cendana, Widya Khairunissa, S.S., M.Hum., Nelly Triana, S.Hut., Winda Ayu Lestari, S.Pi., Dino Harjowiyono, S.Km., Muhammad Mudzar, S.Pd., dan *partner* Penulis Hassanuddin BL, S.Pi., terima kasih telah menjadi tempat untuk saling berbagi kisah selama di perantauan;
12) Teman-teman Kuliah Kerja Nyata gelombang sembilan puluh sembilan Kecamatan Labakkang, terkhusus James Leo Pasulle, S.Ip., Kanda Ryan

Alfari, ST., Aditya Djamhur, Muawwanah, S.Ip, Nur Afiah Mustafa, S.Ip, Syamsiani, S.TP, Usra Pabesak, SE., Fera Saskia, S.Pi, Muhammad Fadil, serta teman-teman yang lain yang tak bisa saya sebutkan satu per satu, terima kasih untuk kenangan empat puluh lima harinya yang menyenangkan;

13) Muh. Yurizal Mahendra, Firmansyah Bustam, S.T, Muhammad Al Ikhsan S, S.T, dan Fahri Antasari, S.T, *partner* seperjuangan yang sampai saat ini masih memberikan dukungan dan semangat kepada Penulis;
14) Saudari Rahmadani, yang telah dengan tabah mendengar segala keluh dan kesah dari penulis dalam proses penyusunan skripsi ini. Terimakasih untuk tetap tabah dan selalu ada;dan
15) 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 Allah SWT. berkenan membalas segala kebaikan dari semua pihak yang telah banyak membantu. Semoga Tugas Akhir ini dapat memberikan manfaat bagi pengembangan ilmu. Aamiin.

Wassalammualaikum wr.wb

Gowa, 21 Agustus 2020

Penulis

DAFTAR ISI

Halaman

DAFTAR GAMBAR

Nomor **halaman**

DAFTAR TABEL

Nomor **halaman**

BAB I
PENDAHULUAN

1.1. Latar Belakang

E-learning menurut Michael (2013) dapat didefinisikan sebagai sebuah proses pembelajaran yang disusun dengan tujuan menggunakan suatu sistem elektronik atau juga komputer sehingga mampu untuk mendukung suatu proses pembelajaran. *E-learning* adalah pembelajaran yang diperantarai teknologi menggunakan komputer baik dari jarak jauh atau dalam pengaturan tatap muka dalam ruang kelas (pembelajaran dengan bantuan komputer), hal ini merupakan pergeseran dari pendidikan tradisional atau pelatihan menuju pembelajaran yang berbasis ICT, fleksibel, individual, mandiri dan didasarkan pada kumpulan pelajar, guru, fasilitator, dan ahli.

Metode belajar dengan menggunakan sistem *e-learning* diharapkan dapat membantu para pengajar dalam mendistribusikan bahan ajar tanpa harus berada di kelas. Dengan pendekatan ini diharapkan mampu memaksimalkan waktu pembelajaran di kelas yang terbatas. Teknologi baru telah bermunculan dengan hadirnya inovasi-inovasi baru seperti dalam penggunanan teknologi *website* untuk *e-learning*. Salah satu teknologi yang dapat digunakan untuk mengembangkan *e-learning* atau kelas virtual yakni WebRTC (*Web Real-Time Communication*). WebRTC merupakan teknologi atau *platform* komunikasi *real-time* yang dapat dijalankan antar *browser website* tanpa menggunakan berbagai macam *plug-in*. WebRTC dapat melakukan komunikasi *real-time* seperti *teleconference* audio maupun video secara *peer-to-peer* dengan menggunakan *browser website* tanpa

adanya penginstalan *plug-in* ataupun membutuhkan perangkat lunak pihak ketiga dalam pengoperasianny. WebRTC dikembangkan dan didefinisikan untuk melakukan pekerjaan ini dengan menggunakan fungsionalitas dari HTML5. HTML5 berupaya menyediakan halaman *website* dengan fitur-fitur canggih dengan cara membuat perbaikan dalam bahasa *markup* dan menambahkan API (*Application Programming Interface*) JavaScript.

Dalam pengimplementasiannya, aplikasi-aplikasi pembelajaran jarak jauh yang dibangun dengan menggunakan WebRTC terkhusus aplikasi yang tidak menggunakan media server sering mengalami kendala-kendala seperti adanya keterbatasan jumlah *client* atau partisipan yang dikarenakan oleh kemampuan dari komputer *client* yang terbatas. Hal ini dikarenakan seluruh pekerjaan dilakukan secara langsung oleh komputer *client* mulai dari proses tukar menukar data antara *client* hingga proses *decoding* dan *encoding* video / audio. Karena keterbatasan dari aplikasi WebRTC sederhana yang mengalami kendala terhadap *resource* yang dibutuhkan oleh *client*, maka akan digunakan teknologi Kutento Media Server. Kurento akan melakukan penanganan dalam penampungan dan *broadcasting* media terhadap seluruh *client* yang terhubung dalam satu aliran data. Selain itu, penulis ingin mengetahui berapa *resource* yang digunakan dalam pembangunan *WebRTC* yang menggunakan media server Kurento.

Oleh karena itu, penulis mengangkat penelitian sesuai dengan judul "**Implementasi Teknologi *Web Real-Time Communication* (*WebRTC)* dalam Perancangan Kelas Virtual sebagai Sarana Pembelajaran Interaktif dengan Media Server Kurento**".

1.2. Rumusan Masalah

Berdasarkan latar belakang, maka rumusan masalah pada tugas akhir ini adalah:

1. Bagaimana penerapan teknologi WebRTC dalam perancangan kelas virtual ?
2. Bagaimana menerapkan WebRTC dengan bantuan media server Kurento ?
3. Bagaimana kinerja dari sebuah WebRTC yang menggunakan media server Kurento ?

1.3. Tujuan Penelitian

1. Untuk mengetahui bagaimana penerapan WebRTC dalam perancangan kelas virtual
2. Untuk mengetahui cara kerja WebRTC menggunakan media server Kurento
3. Untuk mengetahui kinerja dari sebuah WebRTC yang menggunakan media server Kurento

1.4 Batasan Masalah

1. Kelas virtual yang dibangun dibuat menggunakan PUG, Bootstrap, dan *framework Javascript* yaitu NodeJS;
2. *Website* yang dibangun berupa *Learning Management System;*
3. Fitur yang dimiliki pada Kelas Virtual ini adalah *chatting*, *file sharing*, *video conference*, dan *screen sharing*;
4. Bentuk dari kelas virtual yang dimaksud adalah *one-to-many*;

5. Kinerja yang akan diukur adalah penggunaan CPU, penggunaan RAM, dan penggunaan *bandwidth* terhadap jumlah *client* yang terkoneksi dalam sebuah kelas;
6. Perangkat yang digunakan dalam membuat siste ini adalah Laptop HP 14-AM015TX dengan memori RAM 8Gb dan *processor* Intel Core i5;
7. Tempat dilaksanakan tugas akhir ini adalah Laboratorium *Ubiquitous Computing and Networking Lab* Departemen Teknik Informatika Fakultas Teknik Universitas Hasanuddin;dan
8. *Browser* yang digunakan adalah *browser* yang memiliki jenis *chromium*.

1.5. Manfaat Penelitian

Manfaat dari tugas akhir ini adalah :

1. Bagi Peneliti

   Penelitian ini diharapkan mampu menambah pengetahuan peneliti dalam mengimplementasian WebRTC dalam pengembangan *learning management system*.

2. Bagi *Client*

   Penelitian ini diharapkan memberikan nilai tambah dalam pelaksanaan *e-learning* dalam lingkungan pendidikan.

3. Bagi Pendidikan

   Penelitian ini diharapkan mampu menjadi acuan bagi penelitian selanjutnya terutama dalam pengembangan sistem *e-learning*.

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 TINJAUN 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.

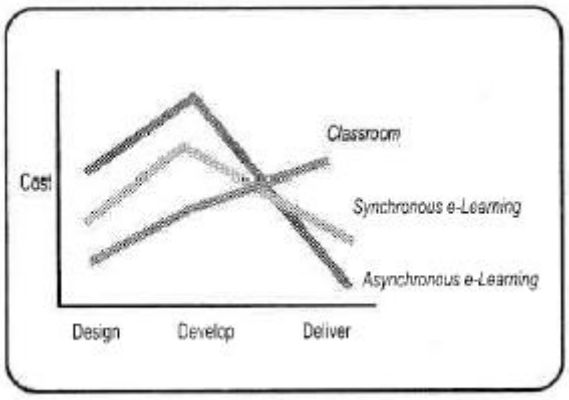
BAB V PENUTUP

Bab ini berisi kesimpulan dari hasil penelitian dan saran.

BAB II
TINJAUAN PUSTAKA

2.1 *Synchronous Learning*

Synchronous learning adalah didefinisikan sebagai interaksi yang berorientasi pada pembelajaran dan instruksi-instruksi srcara langsung (*real-time*) dan biasanya terjadwal. *Synchronous learning* berbeda dengan kuliah biasa, *demo*, atau penawaran suatu produk, dan aktivitas-aktivitas penyampaian informasi lainnya. *Synchronous e-learning* adalah *synchronous learning* yang dilaksanakan dengan memanfaatkan perangkat elektronik, khususnya komputer dan internet. *Synchronous e-learning* dapat dilaksanakan dengan berbagai macam strategi, salah satunya adalah dengan mengimplementasikan konsep *learning management system* berbasis *synchronous e-learning*, atau sering dikenal dengan sebutan *virtual classroom* (VC). Berikut di bawah ini adalah diagram perbandingan biaya desain (*design*), pengembangan (*develop*), dan penyampaian (*delivery*) pada beberapa metode pembelajaran :



Gambar 2.1 Diagram perbandingan beberapa metode pembelajaran

Pada gambar 2.1 menunjukkan bahwa kelas konvensional lebih murah untuk didesain dan dikembangkan, tapi mahal untuk diterapkan pada pembelajar yang sangat banyak dan tersebar karena membutuhkan perjalanan, fasilitas, dan infrastruktur. *Asynchronous e-learning* membutuhkan biaya yang mahal pada tahapan desain dan pengembangannya, namun memerlukan biaya yang sedikit bahkan gratis pada tahapan penyampaian materi. *Synchronous e-learning* memiliki posisi di tengah-tengah, yaitu di antara kelas konvensional dan *asynchronous e-learning* (Maesya, 2010)

2.2 Pengertian Website

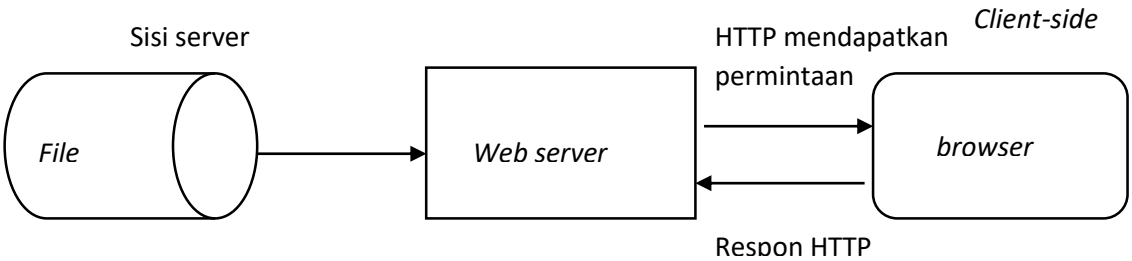
Website atau situs dapat diartikan sebagai kumpulan halaman-halaman yang digunakan untuk menampilkan infomasi teks, gambar diam atau gerak, animasi, suara, dan atau gabungan dari semuanya baik yang bersifar statis maupun dinamis yang membentuk satu rangkaian bangunan yang saling terkait, yang masing-masing 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 Tomothy Jon. 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 lainnya disajikan dalam bentuk grafis (dalam format GIF, JPG, PNG), suara (dalam format AU, WAV), da objek multimedia lainnya (seperti MIDI, Shockwave, Quicktime Movie, 3D World). *Web* dapat diakses oleh perangkat lunak *web client* yang secara pupuler disebyt sebagai *browser*. *Browser* membaca halaman-halaman yang tersimpan dalam *webserver* melalui protokol yang disebyt dengan 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).

2.3 *Website Modern*

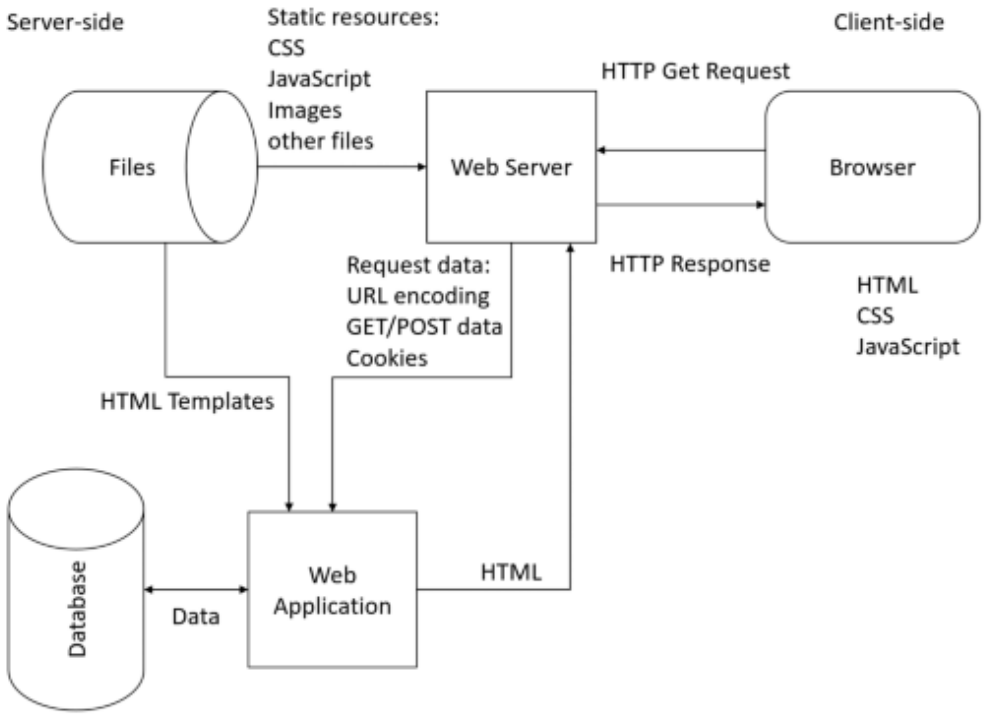
Saat ini, ada banyak cara dalam melakukan pengembangan dan pembuatan aplikasi *web*. Namun pengngembangan *website* dimulai dengan memahami bagaimana arsitektur *web* yang akan digunakan, *web* statik atau dinamis dan juga penentuan pengembangan *tool* dan *service* yang digunakan. Sebuah *website* statis 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 pengguna. Namun HTML dan CSS juga dapat digunakan dalam *website* dinamis ketika memiliki *javascript* yang mengandung pemrograman *back-end*. *Website* modern dibangun menggunakan pemrograman *front-end* yang dieksekusi oleh *browser* dan beralan pada sisi *client* yaitu HTML, CSS, dan pemrograman *back-end* yang dieksekusi pada sisi server yaitu *havascript*, PHP, Python dan yang lainya yang sering digunakan oleh pengembangan *web*. Pemrograman *front-end* berhubungan dengan apa yang pengguna dapat lihat (Hannonen 2017)



Gambar 2.2 Arsitektur *website* statis

Pada *website* statis ketika penguna ingin melakukan perpindahan halaman maka *browser* mengirimkan permintaan HTTP “GET” yang menunjukkan URL.

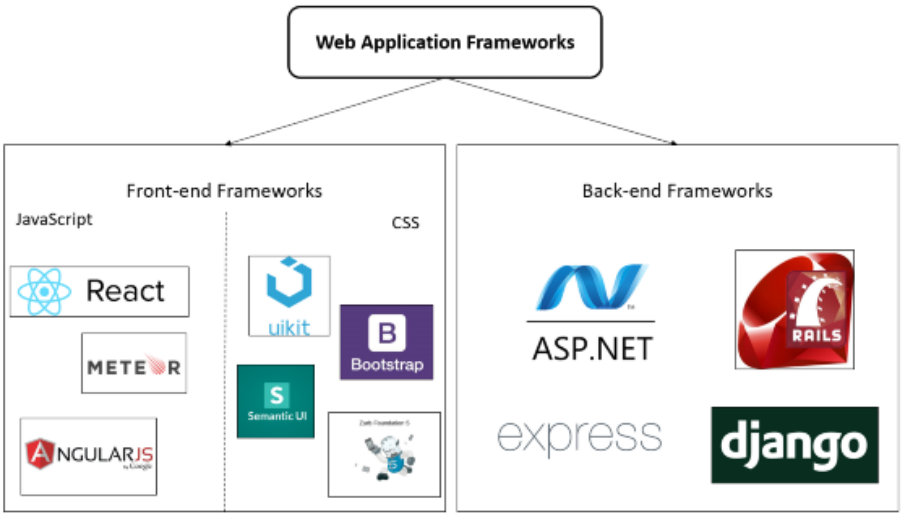
Server kemudian mengirimkan dokumen permintaan dari sistem file menuju *browser* pengguna.



Gambar 2.3 Arsitektur *web* dinamis

Dari kedua gambar yaitu gambar 2.2 dan gambar 2.3, 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 maupun CSS setiap konten mengalami perubahan. Berbeda dengan *website* dinamis dimana memiliki kemudahan dalam melakukan perubahan konten karena bekerja dengan data yang dinamis. Namun dalam mengebangkan 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 menggunaka *framework* dalam melakukan pengembangan *web*. *Framework* memudahkan para pengembang karena kemampuannya menyediakan manajemen pengerjaan *web* yaitu dengan memisahkan bentuk pengerjaan dari sisi *front-end* dan *back-end*.



Gambar 2.4 Contoh aplikasi *web framework*

Framework sisi server adalah *framework back-end* yang memudahakan pengembang dalam melakukan penulisan dan pemeliharaan aplikasi *web*. *Framework* menyediakan pustaka yang sederhana yang sering digunakan dalam membangun sebuah sistem *web*, termasuk URL *routing*, interaksi dengan basis data, mendukung menajemen *session* maupun otoriasi pengguna (*user authorizaion*), 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 *javascript framework* (Angular, ReactJS, MeterJS) dan CSS *framework* (Bootstrap, Foundation, SematicUI, UIkit, MaterialUI). (Hannonen 2017)

2.4 *Web Real-time Communication* (*WebRTC*)

WebRTC (*Web Realtime Communication*) merupakan *framework open source* yang mengizinkan komunikasi secara *real-time* antara penjelajah *web* tanpa mengunakan berbagai macam *plug-in*. Komunikasi secara *real-time* pada WebRTC dapat diakses memalui Javascript API. WebRTC mengizinkan penjelajah *web* dapat melakukan pertukaran data aplikasi dan juga melakukan performa *teleconferencing* suara/video secara *per-to-peer*, tanpa melakukan penginstalan *plug-in* atau perangkat lunak pihak ketiga. WebRTC dirilis Google sebagai proyek *open source* yang telah distandarisasi oleh IETF (*Internet Engineering Task Force*) dan W3C (*World Wide Web Consortium*). Google, Mozilla dan Opera mendukung WebRTC dan terlibat dalam proses pengembangan WebRTC (Virag, 2015).

WebRTC bertujuan untuk menciptakan suatu aplikasi yang *rich* dan *real-time* dengan kualitas tinggi. WebRTC dibangun untuk dapat dijalankan pada berbagai penjelajah *web* dan juga berbagai perangkat meliputi pada komputer pribadi, perangkat bergerak / *smartphone*, maupun *IoT device*, yang mana semua perangkat tersebut dapat saling berkomunikasi satu sama lain menggunakan protokol umum yang sudah ada. Dengan adanya teknologi ini, pengguna dapat secara langsung berkomunikasi dan bertukar data melalui penjelajah *web* tanpa penginstalan *plug-*

in dan lain sebagainya. Teknologi ini sangat mempermudah dan menguntungkan pengguna, yang dapat dijalankan secara langsung kapanpun dan dimanapun pengguna berada melalui beberapa peramban *web* dan juga beberapa *platform* yang berbeda (Rahadiyan Yuniar dkk, 2018)

Aplikasi komunikasi *real-time* berbasis *web* (WebRTC) yang cukup populer saat ini karena memberikan kemudahan bagi pengguna dalam melakukan pertukaran data berupa gambar, suara maupun video antar *client* secara *peer-to-peer* tanpa adanya *plug in* tambahan seperti *flash player* pada *browser*. Akan tetapi aplikasi WebRTC sendiri memiliki keterbatasan pada jumlah *client* yang dapat terhubung dalam satu waktu terkhusus pada aplikasi berupa *video conference*. Hal ini dikarenakan setiap *client* yang terhubung akan mengkonsumsi *bandwidth*, *Random Access Memory* (RAM), serta *processor* yang cukup tinggi seiring dengan peningkatan jumlah *client* (*peer*). Dalam pengimplementasiannya, aplikasi-aplikasi pembelajaran jarak jauh yang dibangun dengan menggunakan WebRTC terkhusus aplikasi yang tidak menggunakan media server sering mengalami kendala-kendala seperti keterbatasan jumlah *client* atau partisipan yang dikarenakan oleh kemampuan dari komputer *client* yang terbatas. Hal ini dikarenakan seluruh pekerjaan dilakukan secara langsung oleh komputer *client* mulai dari proses tukar menukar data antara *client* hingga proses *decoding* dan *encoding* video/audio (Iqra Aswad dkk 2017).

WebRTC mengunakan *PeerConnection* yang bertugas untuk menghubungkan dua aplikasi padakomputer yang berbeda untuk berkomunikasi mengunakan protokol *peer-to-peer*. Komunikasi antara *peer* dapat berupa video,

audio, atau data *arbitrary binary* (untuk *client* yang mendukung API RTCDataChannel). Untuk mengetahui bagaimana dua *peer* dapat terhubung, kedua *client* perlu menyediakan kofigurasi server ICE. Ini adalah server STUN atau TURN, dimana ini berperan untuk menyediakan ICE *Candidate* untuk setiap *client* kemudian ditransfer ke *remote peer*. Pengiriman ICE *Candidate* ini biasa disebut dengan proses *signaling*. Spesifikasi WebRTC mencakup API untuk berkomunikasi dengan Server ICE (*Internet Connectivity Establishment*), tetapi komponen *signaling* diperlukan agar dua *peer* dapat berbagi bagaimana mereka bisa saling terhubung. Setiap koneksi *peer* ditangani oleh objek RTCPeerConnection. Konstruktor untuk kelas ini menggunakan satu objek RTCConfiguration sebagai parameternya. Objek ini menentukan bagaimana koneksi *peer* diatur dan harus berisi informasi tentang server ICE untuk digunakan. Setelah RTCPeerConnection dibuat, kita perlu membuat *offer and answer* SDP (tergantung apakah kita *peer* yang pemanggil ataupun menerima). Setelah *offer and answer* SDP telah dibuat, maka akan dikirmkan ke *remote peer* melalui saluran data yang berbeda. Untuk memulai penyiapan koneksi *peer* dari sisi pemanggil, kita membuat objek RTCPeerConnection dan kemudian memanggil createOffer() untuk membuat objek RTCSesionDescription. Deskripsi sesi ini diatur sebagai deskripsi lokan menggunakan setLocalDescription() dan kemudian dikirim melalui saluran pensinyalan ke sisi penerima. Selain itu penerima juga akan disiapkan saluran *signaling* ketika jawaban untuk sesi deskripsi yang telah ditawarkan diterima. Di sisi penerima, penerima menunggu tawaran yang ditawarkan oleh pemanggil sebelum *instance* RTCPeerConnection dibuat. Setelah selesai, WebRTC akan

mengatur penawaran yang diterima menggunakan setRemoveDescription(). Selanjutnya WebRTC akan memangil createAnswer() untuk membuat jawaban atas tawaran yang diterima. Jawaban ini diatur sebagai deskripsi lokal menggunakan setLocalDescription() dan kemudian dikirim ke sisi panggilan melalui server *signaling*. Setelah kedua *peer* tersebut menetapkan *local* dan *remote session* deskripsinya, mereka mengetahui kemampuan dari sebuah *peer* yang akan dihubungkan. Ini tidak berarti bahwa koneksi antara *peer* sudah siap. Agar ini berhasil, WebRTC perlu mengumpulkan ICE *Candidates* di setiap *peer* dan transfer (melalui saluran *signaling*) ke *peer* lainnya https://webrtc.org/ (diakses pada 7 Agustus, 2020)

2.4.1 PeerConnection

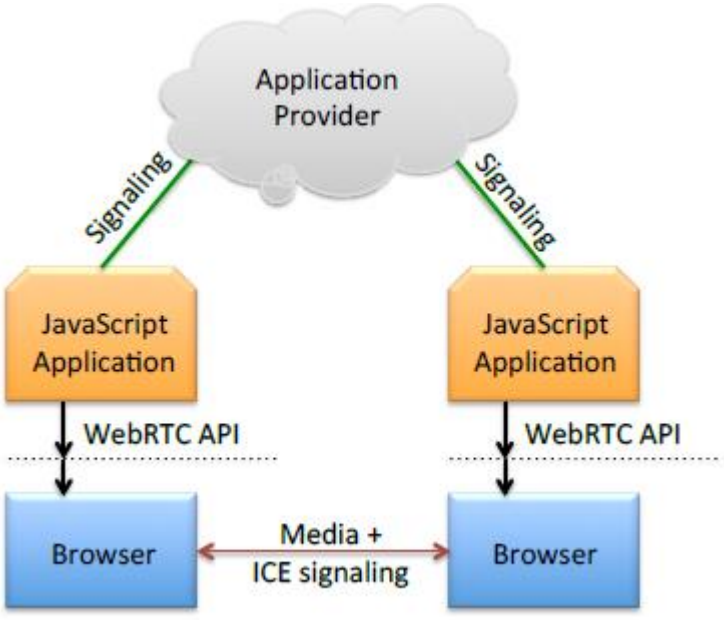
PeerConnection adalah sebuah komponen *website realtime communication* yang menangani komunikasi *streaming data* antara *peers* yang saling terhubung. Kemampuan komponen *PeerConnection* dilakukan melalui objek *javascript* yang diperuntukkan ke pengembang. Objek ini mengabstraksi sejumlah detail dan kompleksitas saat ini terkait dengan cara kerja dalam video dan audio termasuk dalam penyembunyian *packet loss*, pembatalan *echo*, adaptasi *bandwidth*, kontrol gain otomatis, kontrol ICE Candidate untuk gelombang *network address translate*, dan masih banyak lagi (Javier Lopez, 2013)

2.4.2 MediaStream

MediaStream adalah representasi media yang ada pada protokol *web realtime communication* dan dilengkapi dengan dua bagian, yaitu *local* dan *remote*. *Local mediastream* digunakan untuk menangani audio dan video yang ditangkap secara

lokal (contohnya melalui *browser* dalam koneksi lokal) dalam sebuah *webcam* ataupun mikrofon. Sebuah *mediastream* lokal dapat direnderdengan menggunakan tag <*video*> standar yang ada pada HTML5.

Dengan cara yang sama, *remote mediastream* juga dapat membawa saluran video dan audio dan dapat direnderpula mengunakan tag <*video*>. Namun dalam hal ini media yang ada bukan berasal dari kamera ataupun mikrofon lokal, akan tetapi dari *remote* (orang lain) yang ada di ujung komunikasi. Dari perspektif protokol, *stream* melintasi jaringan mengunakan format dan ICE *candidate* yang sudah dinegosiasikan. Untuk mengamankan jaringan komunikasi, SRTP digunakan menjadi DTLS yang salah satu diantaranya memunkinkan terjadi pertukaran kunci. (Javier Lopez 2013)



Gambar 2.5 Arsitektur WebRTC saat ini yang didasarkan pada pemisahan antara peroses pensinyalan dan media.

2.5 *Kurento Media Server*

Kurento Media Server (KMS) adalah sebuah teknologi berbasis *web realtime communication* yang menangani media tansmisi, pemrosesan, pemuatan, dan perekaman media serta mendukung berbagai protokol *streaming* pada jaringan (seperti HTTP, RTSP, dan WebRTC). Kurento Media Server dapat digunakan untuk membangun sebuah aplikasi berbasis *website* untuk melakukan *broadcast* media dari berbagai perangkat yang melayani dan mengkonsumsi media protokol secara otomatis antara media transkodifikasi dengan semua perangkat yang mendukung (Ngako, 2018).

Selain itu, Kurento adalah media server *website realtime communication* dan satu set API (*Application Programming Interface*) *client* yang memudahkan pengembangan aplikasi video canggih untuk www dan *platform smartphone*. Fitur-fitur Kurento Media Server termasuk komunikasi grup, *transcoding*, perekaman, pencampuran, penyiaran, dan *routing* aliran *audiovisual*.

Sebagai fitur lanjutan, Kurento Media Server juga menyediakan kemampuan pemrosesan media canggih yang melibatkan visi komputer, pengindeksan video, *augmented reality*, dan analisis ucapan. Arsitektur *modular* Kurento menyederhanakan pengintegrasian algoritma pemrosesan media pihak ketiga (misalnya pengenalan suara, analisis sentimen, pengenalan wajah, dan lain sebagainya), yang dapat digunakan secara transparan oleh pengembang aplikasi sebagai fitur bawaan dari Kurento.

Kurento menawarkan sebuah *framework* multimedia yang memudahkan untuk membangun sebuah aplikasi multimedia dengan fitur-fitur di bawah ini :

a. *Dynamic WebRTC Media Piplines*

Kurento mengijinkan kustomisasi *media pipeline* yang terkoneksi ke sambungan WebRTC seperti sebuah *web browser* dan aplikasi *mobile*. *Media pipeline* ini didasarkan pada elemen yang dapat digabungkan seperti pemain, perekam, mixer, dan lain-lain. Ini dapat digabungkan dan dicocokkan, diaktifkan, atau dinonaktifkan pada suatu waktu, bahkan ketika media yang ada sedang dialirkan.

b. *Client/Server Architecturei*

Aplikasi yang dikembangkan dengan Kurento mengikuti arsitektur dari *client*/*server*. Kurento adalah sebuah *server* dan menawarkan sebuah *interface websocket* yang mengimplementasikan protokol kurento, yang memungkinkan aplikasi *client* menentukan topologi aliran data.

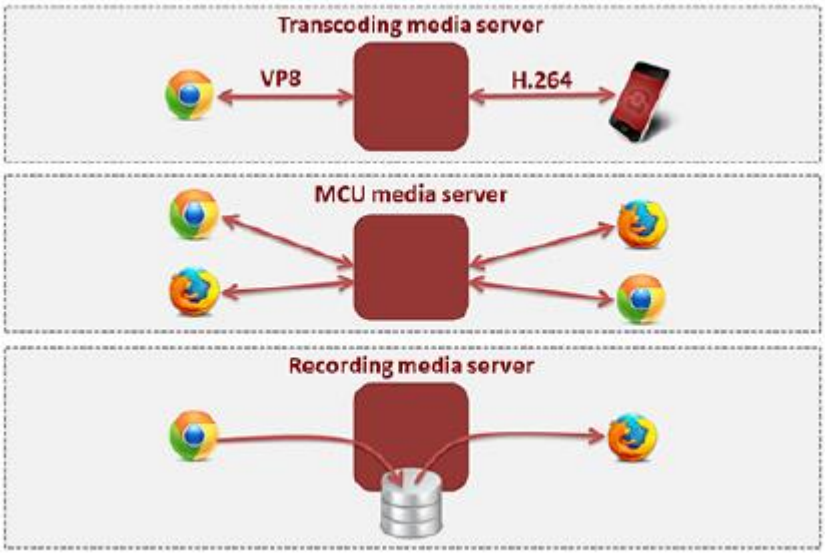
c. *Java and Javascript Client Applications*

Tipe penggunaan dari pengembangan Kurento terdiri dari tiga laipsan arsitektur, dimana *browser* pengguna berinteraksi dengan sebuah *server* kurento yang melalui aplikasi perantara antara pengguna. Ada beberapa *library* kurento yang resmi, yang mendukung penggunaan *java* dan *javascript* untuk aplikasi pengguna. Pengguna untuk bahasa lain dapat dengan mudah mengimplementasikan serta mengikuti protokol dari sebuah *websocket*.

d. *Third Party Modules*

Kurento memiliki arsitektur yang dapat dikembangkan berdasarkan *plugins*, yang memungkinkan pihak ketiga untuk mengimplementasikan modul yang dapat ditambahkan ke jalur *media pipelines*. Ini memungkinkan integrasi algoritma pemrosesan media ke aplikasi *webrtc* apa pun, seperti mengintegrasikan visi komputer, *augmented reality*, pengindeksan video, dan analisis ucapan. Yang diperlukan hanyalah membuat elemen kurento baru, dan menggunakannya di setiap saluran media yang sudah ada.

Komponen utama kurento adalah kurento media server itu sendiri, yang bertanggungjawab untuk melakukan *transmisi* media, pemrosesan, perekaman, dan pemutaran. Kurento dibangun di atas *library GStreamer* yang fantastis, dan menyediakan fitur-fitur berikut :

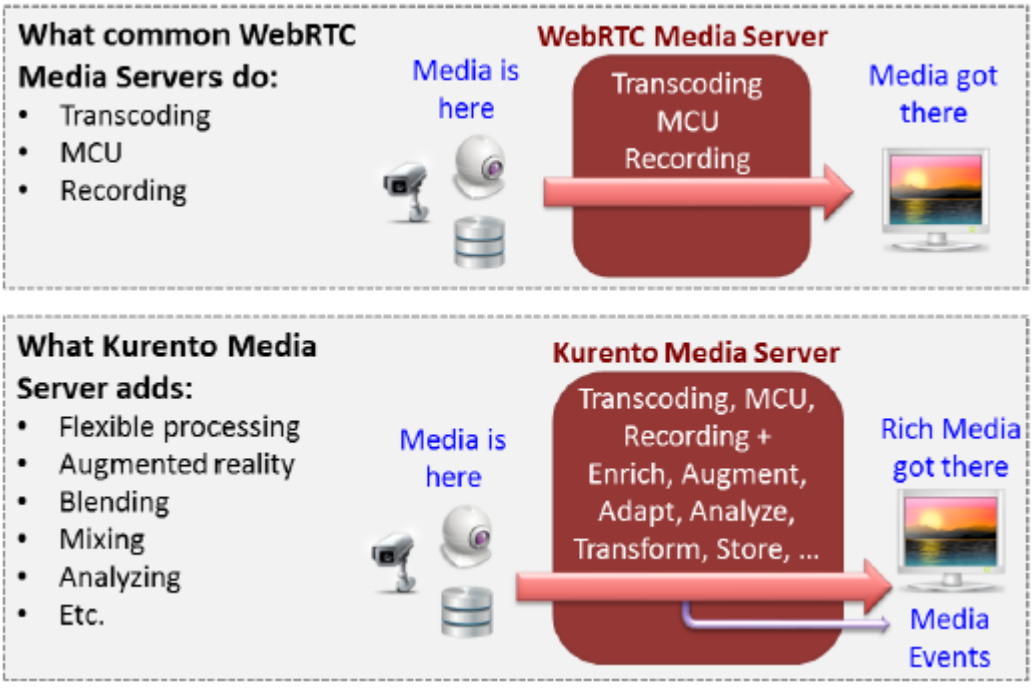


Gambar 2.6 Kemampuan umum Kurento

- Protokol jaringan *streaming*, termasuk *HTTP*, *RTP*, dan *WebRTC*

- Komunikasi grup (fungsionalitas MCU dan SFU) mendukung baik perpaduan pengiriman media maupun pengaturan jalur media
- Dukungan umum untuk filter yang menerapkan algoritma visi komputer dan *augmented reality*
- Media penyimpanan yang mendukung operasi penulisan untuk *WebM* dan *MP4* serta memainkan semua format yang didukung oleh *GStreamer*
- *Transcoding* media secara otomatis antara semua format yang didukung oleh *GStreamer* termasuk VP8, H.264, H.263, AMR, OPUS, Speex, G.711, dan masih banyak lagi

Kurento juga didesain berdasarkan prinsip-prinsip di bawah ini :



Gambar 2.7 Kemampuan Kurento Media Server

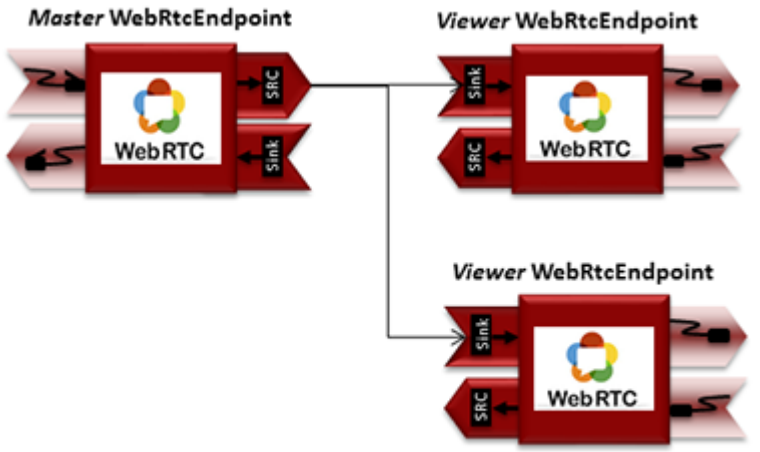
- Alur media dan *signaling*, *signaling* dan media adalah dua pesawat yang terpisah dan kurento dirancang sehingga aplikasi dapat menangani secara terpisah aspek-aspek pemrosesan multimedia
- Distribusi media dan layanan aplikasi, kurento media server dan aplikasi dapat dialokasikan, ditingkatkan atau didistribusikan di antara berbagai mesin yang berbeda. Satu aplikasi dapat meminta layanan lebih dari satu kurento media server, begitu juga sebaliknya yaitu satu kurento media server dapat hadir untuk sebuah permintaan lebih dari satu aplikasi
- Cocok untuk *cloud*, kurento cocok untuk diintegrasikan ke dalam lingkungan *cloud* untuk bertindak sebagai komponen PaaS (*Platform as a Service*)
- *Media pipelines*, menghubungkan elemen media melalui *media pipelines* adalah pendekatan intuitif untuk menantang kompleksitas pemrosesan multimedia
- Pengembangan aplikasi, pengembang tidak perlu memahami kompleksitas internal kurento media server, semua aplikasi dapat digunakan dalam teknologi atau *framwork* yang disukai oleh seorang *developer*, dari *client* ke *server*, serta dari *browser* ke layanan *cloud*
- Kemampuan komunikasi *end-to-end*, kurento menyediakan kemampuan komunikasi *end-to-end* sehingga pengembang tidak perlu berurusan dengan kerumitan pengangkutan, pengkodean/penguraian sandi dan *rendering* media pada perangkat *client*

- *Streaming* media yang dapat diproses sepenuhnya, kurento memungkinkan tidak hanya komunikasi interpersonal interaktif (misalnya *skype*, seperti dengan kemampuan panggilan/penerimaan pangglan), tetapi juga *human-to-machine* (misalnya permintaan video dengan akses *streaming* yang *realtime*) dan *machine-to-machine* (misalnya seperti remakan video jarak jauh, pertukaran data multisensor) komunikasi
- Pemrosesan media modular, modularisasi dicapai melalui elemen-elemen dan saluran-saluran media yang memungkinkan pendefinisian fungsi pemrosesan medai dari suatu aplikasi melalui bahasa "*graph-oriented*", dimana *developer* dapat membuat logika yang diinginkan dengan mengaitkan fungsi-fungsi yang sesuai
- Pemrosesan yang dapat diaudit, kurento mampu menghasilkan informasi yang kaya dan terperinci untuk pemantauan, penagihan, dan audit QoS
- Lapisan adaptasi media yang transparan, kurento menyediakan lapisan media transparan untuk membuat konvergensi di antara perangkat yang berbeda sehingga memiliki persyaratan yang berbeda dalam hal ukuran layar, konsumsi daya, laju transmisi, dan lain-lain

Tsahi Levent dalam tulisannya berjudul "*how many sessions can a kurento server hold ?"* pada tahun 2017 melakukan percobaan tentang berapa *client* yang dapat ditangani oleh Kurento Media Server. Ada tiga skenario yang dilakukan, salah satunya adalah melakukan *broadcast* media ke banyak *viewers*. Kurento diinstall pada mesin AWS t2.medium dengan c4.2xlarge *instance* (8vCPU, 15Gb RAM). Dari hasil penelitian bahwa untuk melakukan *live broadcast* kurento dapat

mengakomodasi 1 *client presenter* dan 80 – 150 *viewers.* Dalam sistem yang dibangun, Kurento di *install* pada server dengan spesifikasi CPU Intel(R) Xeon(R) Gold 6130 CPU 2 Core dan RAM 4 Gb. Jika dibandingkan maka akan di dapatkan perbandingan 4 (AWS t2.medium) : 1 (Server Penelitian) CPU dan 3(AWS t2.medium) : 1 (Server Penelitian) RAM dan akan mengakomodasi kurang lebih 20 – 40 *viewers* https://testrtc.com/sessions-kurento-server/ (diakses, 1 Agustus 2020).

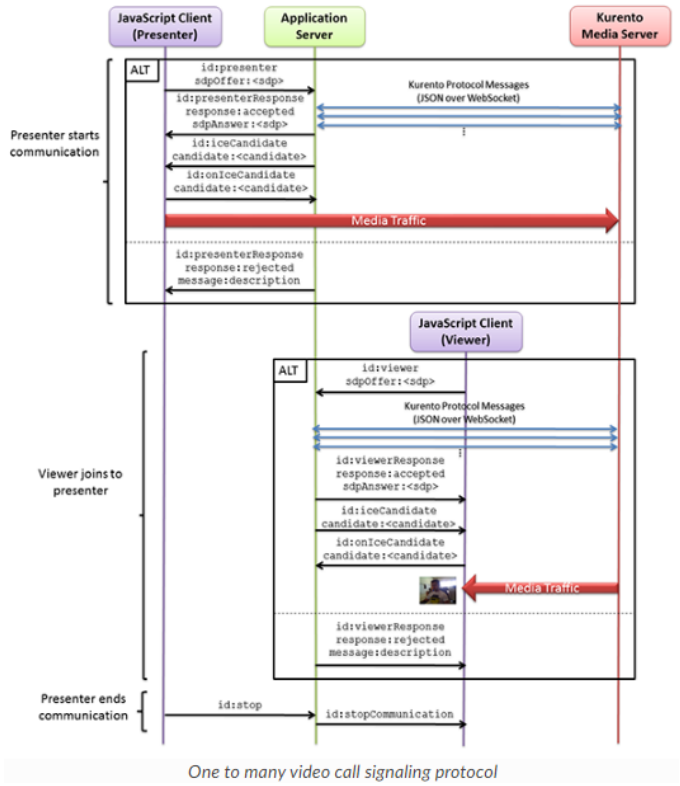
Untuk melakukan implementasi komunikasi *one-to-many* kita harus membuat *media pipeline* yang disusun oleh N+1 WebRtcEndpoints. *Peer* dari *Presenter* mengirimkan aliran datanya ke *viewers* lainnya. *Viewers* dikonfigurasikan dalam mode nanya untuk menerima. Proses implementasi dari *media pipeline* diilustrasikan dalam gambar di bawah ini :



Gambar 2.8 Implementasi *media pipeline* Kurento *one-to-many*

Ini adalah sebuah aplikasi berbasis *website*, dan oleh karena itu maka arsitektur *client* mengikuti *server*. Di sisi *client*, logika diterapkan dengan bahasa

javascript. Di sisi *server* Kurento menggunakan Kurento Javascript *Client* untuk mencapai Kurento Media Server. Secara keseluruhan, aritektur keseluruhan dari ini tersusun atas tiga tingkat. Untuk mengkomunikasikan entitas ini ada dua *websocket* yang digunakan. Yang pertama dibuat antara *browser client* dan *server* Node.JS untuk membawa proses *signaling*. Yang kedua digunakan untuk mengkomunikasikan Kurento Javascript *Client* yang dijalankan di Node.JS dan Kurento Media Server. Komunikasi ini diimplementasikan oleh protokol Kurento. *Client* dan aplikasi *server* berkomunikasi menggunakan sebuah protokol *signaling* yang berbasis JSON melewati WebSocket. Berikut di bawah ini adalah gambar dengan detail pesan antara *client* dan *server* :



Gambar 2.9 Protokol *one-to-many video call signalling* pada Kurento

Seperti yang dapat dilihat pada gambar, SDP dan ICE *candidates* perlu dipertukarkan antara *client* dan *server* untuk membuat koneksi WebRTC antara *client* dan *server* Kurento. Secara khusus, negoisasi SDP menghubungkan WebRtcPeer di *browser* dengan WebRtcEndpoint di *server*. https://doc-kurento.readthedocs.io/en/6.13.0/tutorials/node/tutorial-one2many.html (Diakses 08 Agustus 2020)

2.6 *Javascript*

Javascript adalah bahasa pemrogramanyang populer. *Javascript* adalah bahasa pemrograman yang digunakan untuk HTML dan *web*, untuk server, PC, laptop, tablet dan lebih banyak lagi. Kode pemrgoraman *Javascript* dapat disisipkan kedalam halaman HTML pada awalnya, *Javascript* mulai diperkenalkan di *browser* Netscape Navigator 2. Namun waktu itu namanya bukan *Javascript*, yaitu *LiveScript*. Mengingat pada waktu itu teknolgi Java sedang panas-panasnya atau sedang tren, maka pihak Netscape memutuskan untuk mengganti menjadi *Javascript*, yang sepertinya nama tersebut lebih *marketible* dibandingkan *Livesript*. Selanjutnya pihak Microsoft (rival Netscape) pun mulai ikut-ikutan memfasilitasi *web browser* buatannya, 'Internet Explorer', supaya bisa mendukung *javascript*. Namun mungkin karena gengsi, pihak Microsoft memberi nama bahasa yang lain, yaitu *Javascript.* Mulai saat itu, Netscape dan Microsoft mulai berlomba-lomba mengembangkan bahasa tersebut dalam versi yang berlainan. Oleh sebab persaingan itulah terkadang suatu *Javascript* mungkin bisa bekerja dengan baik di *browser* Netscape, tapi tidak demikian halnya di Internet Explorer, begitu pula sebaliknya (Permana 2016).

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

Ada dua jenis bagaimana *javascript* dibuat, pertama *Javascript* ditulis dalam file yang terpisah dengan HTML, kedua *Javascript* ditulis dalam HTML. *Javascript* yang ditulis diluar HTML disebut eksternal *Javascript* dengan ekstensi file .js. Dalam HTML, penulisan *script* diawali dengan sebuah tag *script*. Selanjutnya *script* yang dijalankan harus diletakkan diantara <script> dan </script> tag. <script> memiliki beberapa atribut, namun yang terpenting adalah atribut *language* dan *type*. Karnea *javascript* bukan satu-satunya bahasa *scripting*, maka sangatlah perlu untuk memberitahukan kepada *browser* bahwa bahasa *script* yang digunakan adalah *javascript* dan selanjutnya *browser* akan menjalankan modul *Javascript* untuk memprosesnya (Permana 2016).

2.7 Node JS

Node JS merupakan *platofrm* server yang dibangun menggunakan *javascript* dan berjalan di dalam *interpreter Chrome Javascript 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 baig digunakan untuk aplikasi waktu nyata yang digunakan diberbagai perangkat (Muhammad Agung Rizkyana, 2014).

Node JS adalah perangkat lunak yang didesain utnuk mengembangkan aplikasi berbasis *web* dan tertulis dalam *sintaks* bahasa pemrograman *javascript*. Bila selama in *javascript* dikenal sebagai bahasa pemrograman yang berjalan di sisi *client / browser* saja, maka Node JS ada untuk melengkapi persan *javascript* sehingga bisa juga berlaku sebagai bahasa pemrograman yang berjalan di sis server, sama seperti halnya PHP, Ruby, Perl, dan sebagainya. Node JS dapat berjalan di sistem opersai 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 Ngingx.

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 pemrograman 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* , artinya 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 *javascript* maka dia tidak akan menunggu proses A itu selesai tetapi semua proses akan dijalankan satu-satu lalu akan menjalankan proses berikutnya yang kita sudah tentukan. Dengan kata lain Node JS bisa mengatasi *multiple request* secara bersamaan dan prosesnya itu tidak langsung diselesaikan tetapi menjalankan beberapa proses telebih dahulu.

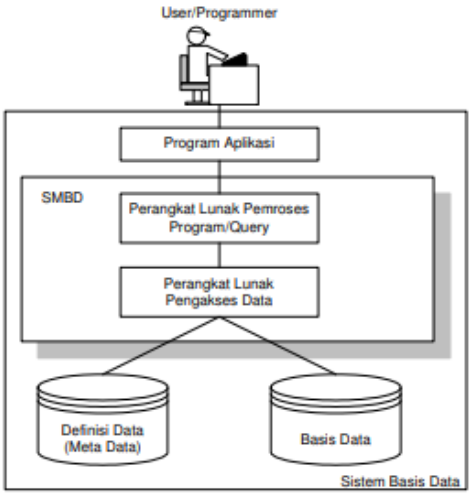
2.8 Basis Data

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 (*database*). 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 Computer Science and Engineer*, para ilmuwan di bidang informasi menerima definisi standar informasi yaitu data yang digunakan dalam pengambilan keputusan. 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 deifinisi-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 2.10 Konsep sistem basis data

C.J Date menyatakan bahwa sistem basis data 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 diolah oleh program *sql* maupun *mysql*. Keduanya adalah media yang digunakan untuk melakukan pengolahan basis data yang digunakan sesuai dengan kebutuhan pengembang.

2.8.1 NoSQL Basis Data

Istilah NoSQL diciptakn 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, yaiut "*Not Only SQL*", istilah yang lebih baik dari sebelumya yang lebih dikenal dengan anti relasional. NoSQL mengakomodasi tanda yang tidak terstruktur, kehadirannya bukan untuk menggantikan SQL namun kedua teknologi ini dapat saling berdampingan. Perbedaan utama kedua basis data ini dapat adlaah SQL memiliki skema yang kaku sementara basis data NoSQL menawarkan desain yang fleksibel yang dapat diubah tanpa *downtime* atau gangguan lainnya. NoSQL juga dirancang utnuk menyimpan data yang dapat didistribusikan untuk kebutuhan data dalam sekala besar; misalnya Facebook memiliki 500 juta pengguna dan Twitter terakumulasi *terabyte* data, basis data NoSQL telah memiliki popularitas yang tinggi, sehingga basis data ini diklaim lebih baik dari basis data SQL. Basis data NoSQL dimotivasi oleh kesederhanaan, *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 rasakan adlaah data yang berasal dari *smartphone* yang mencatat lokasi *broadcas* setiap saat, video dan kamera bahkan halaman *website* yang berisi banyak informasi serta dokumen (No dan Junaidi 2016)

Teknologi NoSQL memiliki keunggulan sendiri dalam melakukan manajemen 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. Kelima basis data tersebut memiliki keunggulan yaitu mampu menjadi basis data yang memungkinkan pengembang untuk membuat sebuah media penyimpanan *offline* bagi sistem yang bekerja, sehingga cocok digunakan pada sebuah aplikasi yang bekerja *online* maupun *offline*.

Sistem berbasis NoSQL biasanya digunakan dalam *database* yang sangat besar, yang sangat rentan terhadap masalah kinerja yang disebabkan oleh keterbatasan SQL dan model relasional dari *database*.

2.7.2.1 MongoDB

MongoDB adalah salah satu *software* NoSQL yang termasuk dalam kategori *Document Store / Document-Oriented Database*, yaitu data disimpan dalam bentuk dokumen. Suatu dokumen bisa diibaratkan seperti suatu *record* dalam basis data relasional dan isi dari masing-masing dokumen tersebut bisa berbeda-

beda dan ada pula yang sama. Hal ini berbeda dengan basis data relasional yang menetapkan keseragaman kolom serta tipe data dengan data yang NULL jika tidak terdapat data. MongoDB menyimpan data dalam bentuk dokumen dengan menggunakan format JSON (*javascript object notation*). Konse dasar yang harus dipahami dalam MongoDB sebagai *document-oriented database* adalah *documents* dan *collections*. Sama halnya dengan basis data relasional, MongoDB menyimpan data dalam suatu basis data. Di dalam basis data tersebut terdapat *collections* yang bisa diibaratkan seperti tabel dalam basis data relasional, *documents* adalah *records*.

Pembuatan *database* pada MongoDB sangat berbeda dengan pembuatan *database* di SQL. Dalam SQL Kita harus membuat *database* terlebih dahulu sebelum melakukan perintah '*use*' dengan menggunakan perintah *create database dbs_name*. Dalam MongoDB tidak perlu menggunakan perintah *create* Kita bisa langsung menggunakan perintah *use* meskipun Kita tahu bahwa *database* yang dimaksud belum ada dalam MongoDB, namun MongoDB mengijinkan Kita untuk melakukan hal itu. Tetapi saat keluar dari *database* tersebut, maka *database* akan hilang, hal ini dikarenakan Kita tidak membuat *collections* dan mengisikan *documents* dalam *collections* (Rohman 2016).

Seguin mengatakan bahwa ada enam konsep dasar yang perlu diketahui mengenai MongoDB, yaitu sebagai berikut :

a. MongoDB memiliki konsep yang sama dengan *database* pada umumnya seperti MySQL dan Windows SQL Server. MongoDB dapat memiliki nol atau lebih *database*

b. Sebuah *database* dapat memiliki nol atau lebih *collections* yang dapat disetarakan dengan *table* pada *database* umumnya
c. Sebuah *collections* terdiri dari nol atau lebih dokumen yang dapat disetarakan dengan basis pada *database* umumnya
d. Sebuah dokumen terdiri dari satu atau lebih *fields* yang dapat disertakan dengan kolom pada *database* umumnya
e. MongoDB memiliki indeks yang memiliki fungsi utama seperti indeks pada *database* umumnya
f. Data dari MongoDB akan dikembalikan dalam bentuk kursor

MongoDB bersifat *client-server*. Sisi *server* sebagai tempat proses data dan untuk penulisan instruksi pada sisi *client*. Pengaksesan *database* dilakukan dengan instruksi *use database_name* seperti *use learn*. Instruksi tersebut akan mengakses *database learn* yang sudah ada atau membuat baru jika belum ada ketika proses *insert* pertama. Instruksi yang akan dieksekusi pada *database* dieksekusi pada *collection* dituliskan dengan menggunakan objek db seperti db.help() atau db.stats(). Instruksi yag akan dieksekusi pada *collection* dituliskan dengan menggunakan objek db.*collection_name* seperti db.unicorns.help() atau db.unicorns.count().

Menurut Seguin, MongoDB tidak menggunakan *JOIN* seperti yang biasa dimiliki oleh *database* pada umumnya, tetapi dapat tetap dihubungkan secara *manual* seperti pengunaan *foreign key* di SQL. Langkah yang dapat dilakukan adalah dengan menyimpan _id dari satu dokumen dalam dokumen yang lain. Pilihan lainnya adalah dengan menggunakan DBRef yang sudah disediakan oleh

MongoDB. DBRef dapat digunakan untuk menghubungkan antara lebih dari satu dokumen dalam *collection* atau *database* yang berbeda. DBRef memiliki bebeapa *field* sebagai berikut :

a. $ref – *field* $ref akan berisi nama *collection* dimana dokumen yang akan dihubungkan berada
b. $id – *field* $id akan berisi nilai _id dari dokumen yang akan dihubungkan
c. $db – *field* $db bersifat *optional* dan akan berisi nama *database* dimana dokumen yang akan dihubungkan berada

Dalam MongoDB juga disediakan beberapa fitur tambahan lagi yang mendukung perfirma. Berikut ini akan dibahas beberapa hal yang akan sering digunakan pada *database* MongoDB. (Yovita Turnadi 2019)

a. Indexes

Index pada MongoDB berfungsi sama dengan indeks pada *database* relasional pada umumnya yang membantu performa *query* dan *sorting* data. Penambahan indeks dapat dilakukan dengan menggunakan instruksi *dropIndex*

Pembuatan *unique inde* dapat dilakukan dengan menambahkan *parameter* kedua. Pengurutan indeks secara *ascending* ditandai dengan angka satu dan *descending* ditandai dengan angka -1.

b. Explain

Pada MongoDB dapat dilakukan pengecekan apakah suatu *collection* dan *query* menggunakan indeks. Instruksi yang digunakan adalah

explain. Penggunaan instruksi db.unicorns.find() akan menampilkan kursor yang digunakan adalah *BasicCursor* yang menandakan tidak adanya penggunaan indeks. Penggunaan instruksi db.unicorns.find({name:'pilot'}).explain() akan menampilkan kursor yang digunakan adalah *BtreeCursor* adanya pengunaan indeks. (Yovita Turnadi 2019)

c. *Backup* dan *Restore*

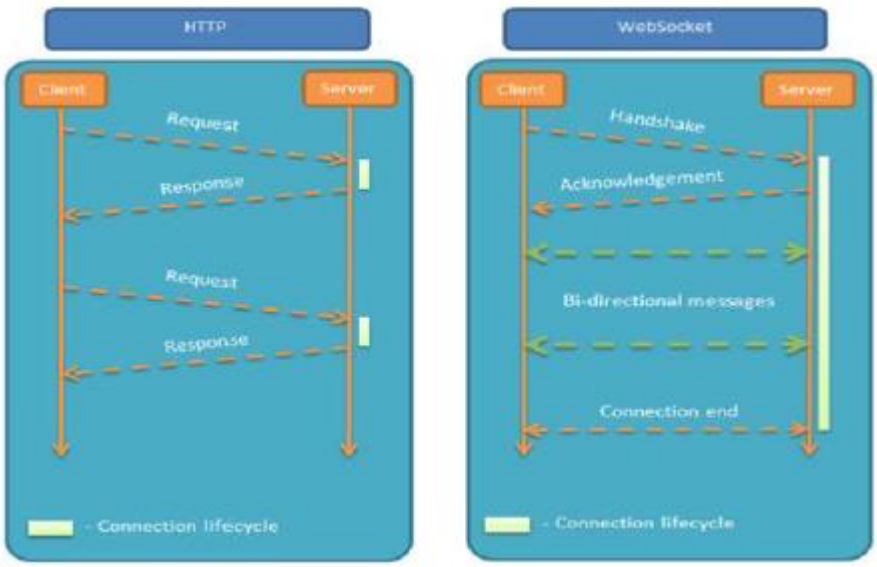
Backup dan *restore* pada MongoDB dapat dilakukan dengan menggunakan *fileexecuteable* yang sudah disediakan oleh MongoDB. *Backup* dilakukan dengan menjalankan *mongodump* dengan mengikuti pilihan *database, collection,* dan nama *filebackup*. Sedangkan untuk *restore* dengan menjalankan proses *mongorestore* dengan pilihan yang sama.

Selain *backup* dan *restore*, MongoDB juga memungkinkan untuk proses *import* dan *export fileexecuteable* yang digunakan adalah *mongoimport* dan *mongoexport* (Yovita Turnadi 2019)

2.9 WebSocket

Websocket adalah salah satu protokol yang berjalan pada *layer* aplikasi atau pada *layer* 7 di dalam *osi layer*. Websocket mendukung komunikasi *full duplex* dimana komunikasi terjadi dua arah antara *client* dan *server* berbeda dengan protokol yang kita kenal saat ini yaitu HTTP. HTTP adalah sebuah protokol yang berjalan pada *port* 80 dan bekerja dengan konsep *request* dan *response* atau yang biada disebut dengan *half duplex*. Dengan mengunakan mekanisme *single TCP connection* maka websocket dapat membangun satu buah jalur komunikasi antara

server dan *client*. Dimana websocket menggunakan metode *handshaking* untuk membuat sebuah koneksi yang terbuka antara *server* dan *client*. Proses yang pertama dilakukan adalah *client* mengirimkan *handshake* kepada server kemudian server membalas *handshake* tersebut lalu terciptalah sebuah *single TCP connection* antara *server* dan *client*. *Single TCP connection* tersebutlah yang mendukung komunikasi *bidirectional* antara *server* dan *client* sampai koneksi tersebut diputus oleh *client* ataupun *server* (Pimentel & Nickerson 2012)



Gambar 2.11 Perbandingan *life cycle* HTTP dan websocket

Websocket pada dasarnya menggunakan koneksi dua arah dalam satu *socket*. Websocket merupakan teknologi terbaru yang ada pada HTML5 untuk mendukung koneksi dua arah antara *client* dan *server* secara lebih cepat dan ringan dibandingakn dengan metode HTTP tradisional. Dalam Websocket permintaan HTTP menjadi permintaan untuk membuka koneksi Websocket dan kemudian

menggunakan kembali koneksi tersebut dari *client* ke *server* dan *server* ke *client*. (Wang, Salim & Moskovits 2013)

2.10 Socket.io

Socket.io adalah sebuah *library* yang dibuat menggunakan bahasa pemrograman Javascript dan merupakan salah satu dari Websocket API (*Application Promgramming Interface*). Tidak hanya dari sisi *client*, *library* ini mendukung node.js sebagai server dengan sangat baik. *Iibrary* ini memberikan performa yang sangat baik dalam *transfer* data secara *real-time* (Julisman, Agung. 2014)

Selain itu, socket.io adalah abstraksi dari *WebSocket* yang memungkinkan komunikasi *realtime* antara *browser* dan *server*. *Library javascript* yang dikembangkan oleh *LearnBoost* ini disediakan untuk mendukung pengembangan aplikasi *realtime* bagi halaman web. Karakter *event-driven* yang ada pada socket.io juga dimiliki oleh Node.JS, sehingga sangat cocok digunakan berdampingan dengan Node.JS (Gelens, 2014)