BAB IV
HASIL DAN PEMBAHASAN

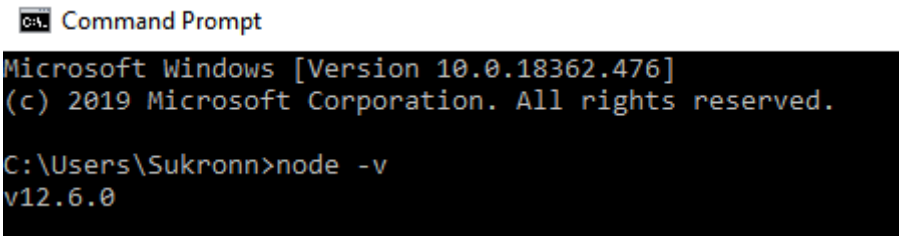
Perancangan Tampilan dibuat dengan menggunakan *framework javascript Vue.js* untuk tampilannya dan menghubungkan data dengan menggunakan *API* yang disediakan tim *backend developer* MedUp melalui postman yang bisa dilihat oleh peneliti. Tampilan *website user* dibagi menjadi dua tampilan yaitu tampilan *website* MedUp yang digunakan oleh *user* pasien dengan tampilan *Dashboard* MedUp yang digunakan oleh *admin* klinik dan *admin* MedUp.

4.1 Persiapan Implementasi

Vue.js adalah *framework javascript* untuk membangun interface sebuah *web* agar menjadi lebih interaktif dengan *user*. Sebelum melakukan instalasi *Vue.js*, ada beberapa hal yang harus dipersiapkan dan diperhatikan yaitu:

a. Node.js

Node.js adalah perangkat lunak yang bersifat *non-blocking* untuk mengembangkan suatu aplikasi berbasis *web* yang ditulis dalam bahasa pemrograman *javascript. Node.js* hadir sebagai pelengkap *javascript* yang berperan sebagai bahasa pemrorgraman yang berjalan di sisi *client* atau *server*, dengan Node.js *javascript* bisa berjalan di sisi *server* juga (Lutfi, 2017). Cara *Install* Node.js yaitu *download installer* (.msi) untuk Node.js pada *https://nodejs.org/en/download/*, lalu jalankan *installer* dan ikuti petunjuk sampai selesai. Adapun cara mengecek Node.js sudah ter*install* atau belum dengan mengecek versi node melalui *command prompt* atau terminal dengan mengetikan *"node-v"*. Berikut contoh gambar pengecekan versi node:



Gambar 4.1 Cek Versi Node.js

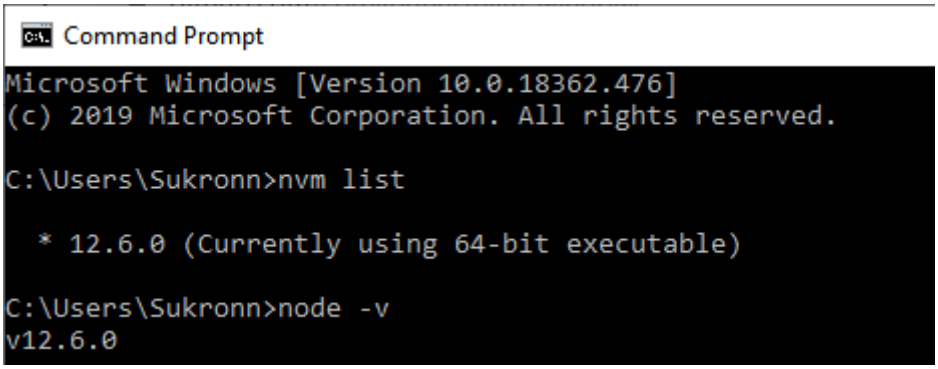
b. NPM (Node Packaged Manager)

NPM atau Node Package Manager adalah repositori online untuk *publishing* proyek Node.js *open-source* dan merupakan utilitas command-line untuk berinteraksi

dengan repositori yang membantu dalam instalasi paket, manajemen versi, dan manajemen dependensi (Node.js, 2011).

c. *NVM (Node Version Manager)*

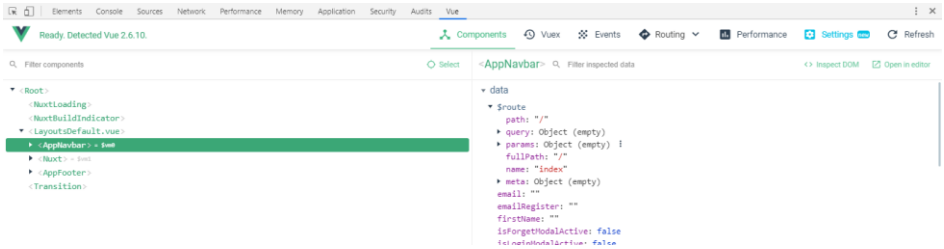
NVM adalah manajer untuk vers–versi node, karena Node.js perkembangan nya yang termasuk cepat dan untuk menghindari beberapa paket dalam Node.js versi lama tidak bisa lagi digunakan pada versi baru, maka dibutuhkan NVM sebagai solusi untuk menyimpan beberapa versi dari Node.js. Cara *install* NVM dengan mendownload *installer* atau *nvm-setup* pada *https://github.com/coreybutler/nvm-windows/releases*, lalu jalankan *file installer* yang sudah di *download* dan ikuti petunjuk sampai selesai. Adapun cara mengecek *NVM* sudah ter*install* atau belum di dalam PC dengan membuka terminal atau *command prompt* lalu ketik "*nvm list*" yang berarti mengecek versi-versi node yang ada pada *NVM*. Berikut contoh gambar pengecekan versi node:



Gambar 4.2 Cek *NVM list*

d. *Vue.js Devtools*

Vue.js Devtools adalah ekstensi *Vue.js* yang di pasang pada *browser*, *Vue.js devtools* mempermudah pengembang dalam menginspeksi *web* yang dikembangkan. Tampilan *Vue.js Devtools* terdapat pada gambar Gambar 4.3

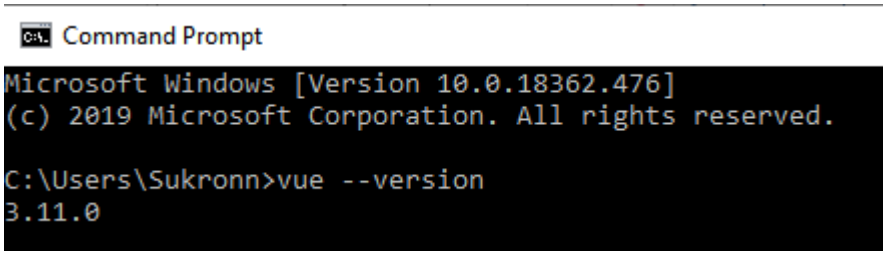


Gambar 4.3 Tampilan *Vue.js Devtools*

Setelah selesai mempersiapkan hal – hal yang dibutuhkan untuk meng*install Vue.js*, maka kita bisa langsung meng*install Vue.js*. ada beberapa cara untuk meng*install Vue.js* js seperti

yang dijelaskan pada *https://v1.Vue.jsjs.org/guide/installation.html*. Akan tetapi, yang akan dijelaskan pada penelitian kali ini adalah cara *install* dengan cara *CLI* atau *Command Line Interface.*

Vue.js menyediakan *CLI* resmi yang berisi paket untuk alur kerja *front-end* modern. Cara meng*install Vue.js* menggunakan CLI dengan cara membuka *command prompt* atau terminal yang sudah ada Node.JS dan *NPM* didalamnya lalu ketik "*npm install –g @Vue.js-cli*", "*npm install*" berarti klien baris perintah dari *npm* untuk meng*install* suatu *framework* atau *library*, "-g" yang berarti *global* yaitu meng*install* suatu *framework* atau *library* secara menyeluruh, dan "*Vue.js*-cli" yang berarti CLI resmi dari *Vue.js* untuk meng*installVue.js*. Lalu tunggu beberapa menit sampai *install Vue.js* selesai, dan cek apakah *Vue.js* sudah terpasang pada cli dengan mengetik "*Vue.js* –version". Berikut contoh gambar pengecekan versi *Vue.js*:



Gambar 4.4 Cek Versi *Vue.js*

Setelah melakukan *install*, hal selanjutnya yang harus diketahui adalah cara untuk membuat projek atau produk dengan menggunakan *Vue.js*. Cara membuat projek atau produk menggunakan *Vue.js* dengan mengetikkan "*Vue.js create* nama_projek" pada terminal yang sudah ter*install Vue.js* dan ikuti proses instalasi sampai selesai.

4.2 Hasil Pengembangan Tampilan

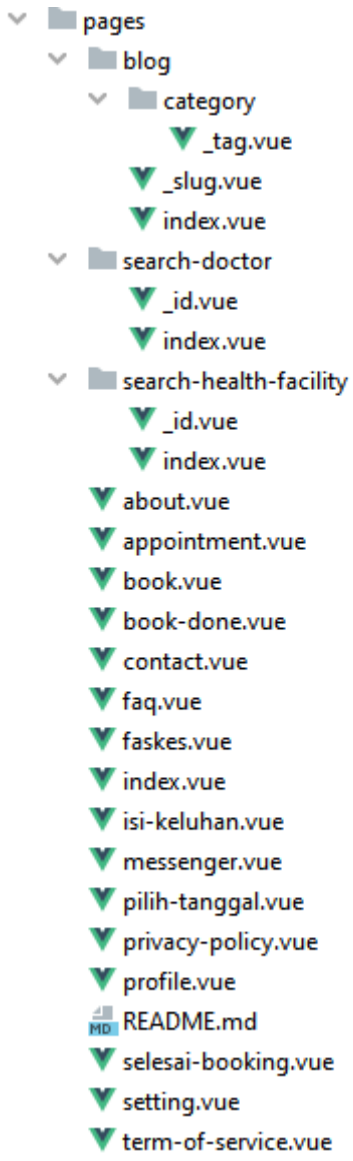
Pada hasil pengembangan tampilan, hasil yang di dapat dibagi menjadi dua jenis tampilan *website*, yaitu:

a. Tampilan *website* MedUp yang digunakan oleh *user* untuk menjalankan *use case* yang sudah dianalisis.
b. Tampilan *Dashboard* MedUp yang digunakan oleh *admin* MedUp atau *admin* klinik untuk menjalankan *use case* yang sudah dianalisis.

Masing – masing *use case* memiliki tampilan *file pages* dan *file Component.* Hasil dan pembahasan yang ditampilkan pada laporan penelitian ini adalah Tampilan *website* MedUp saja, dan untuk Tampilan *Dashboard* MedUp tidak ditampilkan tapi tetap disediakan oleh peneliti. Untuk lebih detailnya akan dijelaskan kembali sebagai berikut:

Pada pengembangan tampilan *website* MedUp digunakan untuk *user* pasien dalam menjalankan *use case* yang ada pada *user* seperti mencari dokter, mencari fasilitas kesehatan, *booking*, lihat antrian, menambah data pengguna, dan melihat artikel.

Sebelumnya, akan ditampilkan struktur *file* dan *component* code yang sudah dibuat pada tampilan *website* MedUp dan fokus pembahasan peneliti hanya pada setiap *use case* yang sudah dianalisis sebelumnya. Struktur *file* dapat dilihat pada Gambar 4.5 dan komponen *file* dapat dilihat pada Gambar Gambar 4.6



Gambar 4.5 Struktur *File Website* MedUp

Pada Gambar 4.5 merupakan struktur *file* dari tampilan *website* MedUp, *use case diagram* pada strutktur *file* terdiri ini terdiri dari enam *use case* yaitu mencari dokter, mencari fasilitas kesehatan, *booking*, lihat antrian, dan lihat artikel. Untuk *use case booking* dan lihat antrian

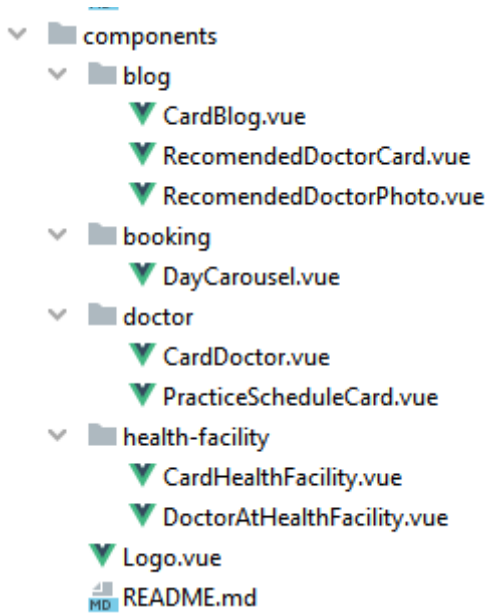
diharuskan untuk *login* terlebih dahulu kecuali *use case* mencari rumah sakit, mencari dokter, dan lihat artikel. Adapun pendefisian setiap struktur *file* dapat dilihat pada Tabel 4.1

Tabel 4.1 Struktur *File Website* MedUp

No	*Use case*	Alamat *File*	Deskripsi
1	Mencari Fasilitas Kesehatan	*Search-health-facility/ index.vue*	Folder *search-health facility* ditujukan untuk tampilan fasilitas kesehatan. Pada *file index.vue* berisi tampilan untuk *list–list* fasilitas kesehatan berdasarkan pencarian.
	Mencari Fasilitas Kesehatan	*Search-health-facility/ _id.vue*	Folder *search-health facility* ditujukan untuk tampilan Fasilitas Kesehatan. Pada *file _id.vue* berisi tampilan untuk profil dari setiap Fasilitas Kesehatan
2	Mencari Dokter	*search-doctor/index.vue*	Folder *search-doctor facility* ditujukan untuk tampilan dokter. Pada *file* index.*Vue.js* berisi tampilan untuk *list–list* dokter berdasarkan pencarian.
		search-doctor/_id.vue	Folder *search-doctor facility* ditujukan untuk tampilan dokter. Pada *file _id.vue* berisi tampilan untuk setiap profil dokter.
3	Melihat Artikel	*Blog/index.vue*	Folder *blog* ditujukan untuk tampilan artikel. Pada *file index.vue* ditujukan untuk tampilan *list - list* artikel.
		Blog/_slug.vue	Folder *blog* ditujukan untuk tampilan artikel. Pada *file _slug.vue* ditujukan untuk tampilan isi dari setiap artikel dalam *list*.
		Blog/category/_tag.vue	Folder *category* yang ada didalam *blog* ditujukan untuk tampilan *tag* artikel. Pada *file _tag.vue* ditujukan untuk tampilan *list–list* berdasarkan tag yg sama.
4	Menambahkan Data Pengguna	Profil.*vue*	Profil.*vue* menampilkan data profil pasien
5	*Booking*	*Book.vue*	*File book.vue* ditujukan untuk tampilan buat perjanjian ketika pasien sudah memilih rumah sakit dan dokter.
		Book-done.vue	*File book-done.vue* ditujukan untuk tampilan perjanjian tercatat ketika pasien sudah melakukan *booking*.
6	Melihat Antrian	*Appointment.vue*	*File appointmen.vue* menampilkan *list* daftar perjanjian dan *list*

			riwayat perjanjian yang dilakukan oleh pasien

Selain terdapat struktur *file website* MedUp, peneliti juga membuat struktur tampilan *component* yang terhubung ke dalam struktur *file*, untuk lebih detailnya dapat dilihat pada Gambar 4.6



Gambar 4.6 Struktur *Component Website* MedUp

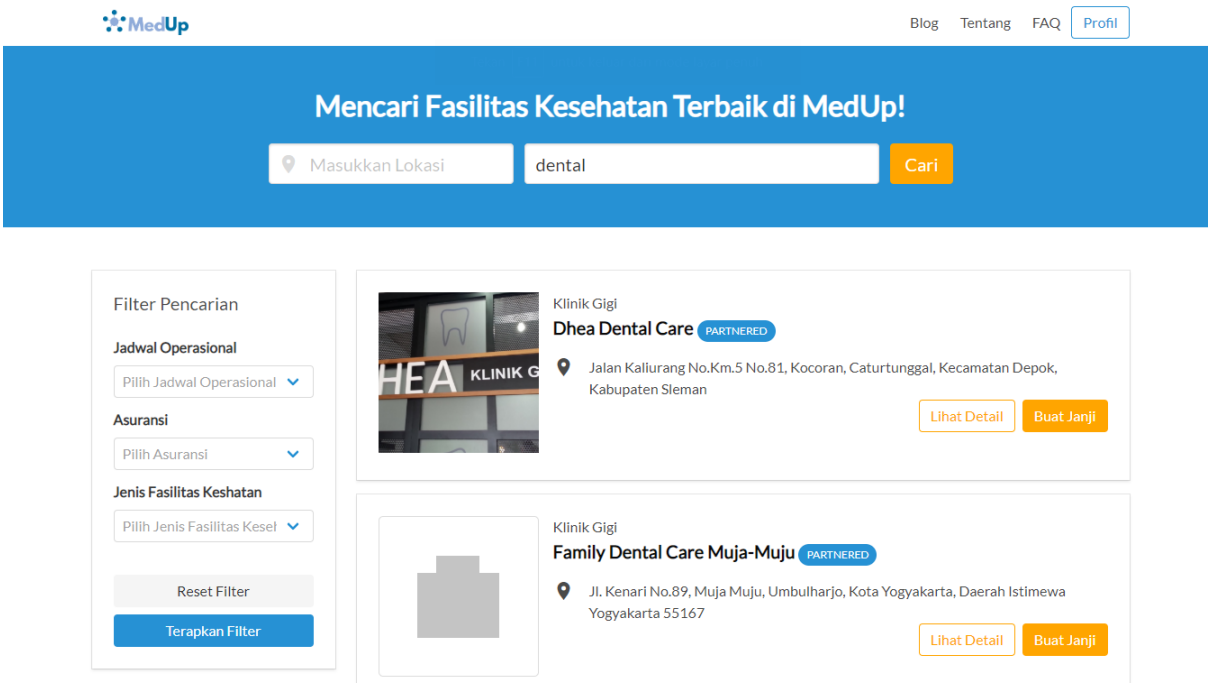
4.2.1 *Use case* Mencari Fasilitas Kesehatan

Halaman Mencari Fasilitas Kesehatan adalah halaman untuk mencari fasilitas kesehatan berdasarkan nama dan lokasi lalu diarahkan kepada halaman *list* berdasarkan hasil pencarian yang disesuaikan dengan *use case* dan *diagram acitivity* agar alur mudah dibaca dan dimengerti. Untuk masuk ke halaman ini tidak harus *login* terlebih dahulu jika tujuannya hanya mencari informasi fasilitas kesehatan saja. Berikut *file component* yang terlibat dalam pengembangan tampilan mencari fasilitas kesehatan:

Tabel 4.2 *File* Komponen Mencari Fasilitas Kesehatan

Component Mencari Fasilitas Kesehatan			
No	*Use case*	*File* Implementasi	Deskripsi
1.	Mencari Fasilitas Kesehatan	*health-facility/ CardHealthFacility.vue*	Folder *health-facility* ditujukan untuk *component* tampilan fasilitas kesehatan. Pada *file CardHealthFacility.vue* ditujukan untuk tampilan *card* pada tampilan *list–list* fasilitas kesehatan.
2	Melihat Profil Fasilitas Kesehatan	*Health-facility/ DoctorAtHealthFacility.vue*	Folder *health-facility* ditujukan untuk *component* tampilan fasilitas kesehatan. Pada *file DoctorAtHealthFacility.vue* ditujukan untuk tampilan *tab* dokter pada tampilan profil fasilitas kesehatan.

Pada Tabel 4.2 dijelaskan *file* yang digunakan dalam tampilan mencari fasilitas kesehatan. *File index.vue* yang terdapat pada folder *search-health-facility.* Tampilan yang dibuat Pada Gambar 4.7 adalah tampilan untuk bagian *header*, *side filter search*, dan *template card* fasilitas kesehatan. *Component file CardHealthFacility.vue* yang terdapat pada folder *health-facility* adalah untuk tampilan untuk bagian dari setiap *card* yang dipanggil oleh *file index.vue* dan dimasukan ke dalam bagian *template card* fasilitas kesehatan. Hasil tampilan halaman mencari fasilitas kesehatan terlihat pada Gambar 4.7

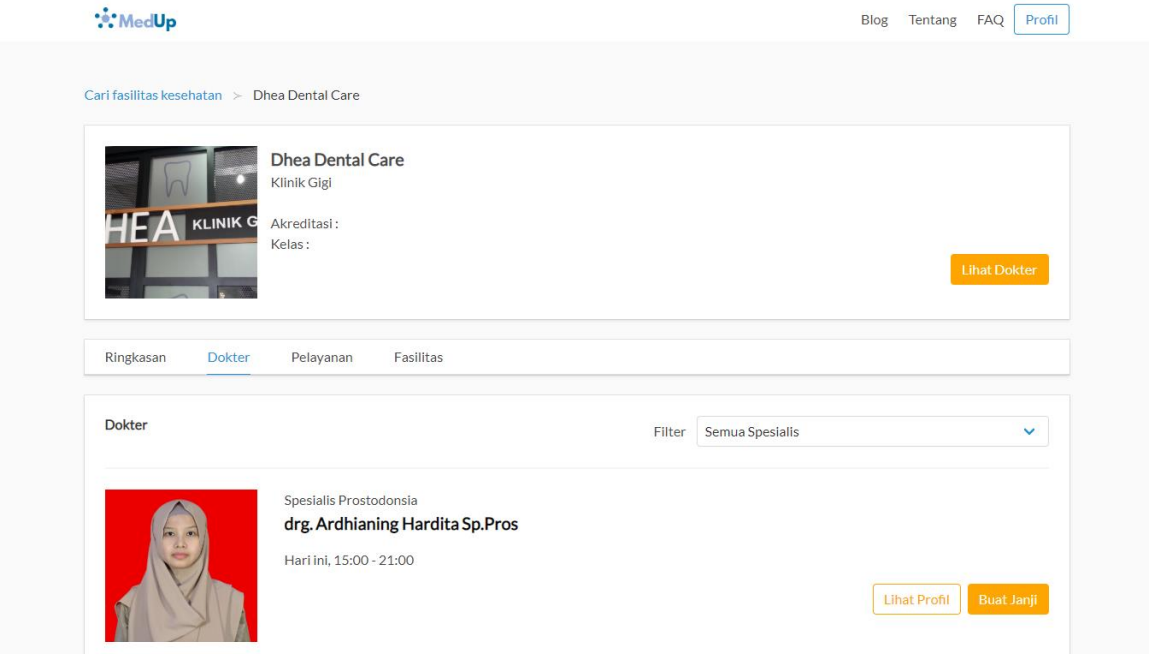


Gambar 4.7 Halaman Mencari Fasilitas Kesehatan

Profil Fasilitas Kesehatan

Halaman Profil Fasilitas Kesehatan adalah halaman yang berisi informasi – informasi tentang fasilitas kesehatan terkait seperti informasi umum atau ringkasan, fasilitas, dokter, dan pelayanan. Halaman profil fasilitas kesehatan juga adalah alur lanjutan dari halaman mencari fasilitas yang disesuaikan dengan *use case* dan *diagram acitivity* agar alur mudah dibaca dan dimengerti.

Pada Tabel 4.2 dijelaskan *file* yang digunakan dalam tampilan profil fasilitas kesehatan. *file _id.vue* yang terdapat pada folder *search-health-facility*. Tampilan yang dibuat Pada Gambar 4.8 adalah tampilan yang berisi informasi informasi yang dibutuhkan pasien seperti nama, foto, peta, dokter, fasilitas, dan asuransi yang ada pada profil fasilitas kesehatan. *Component file Doctor*At*HealthFacility.vue* yang terdapat pada folder *health-facility* adalah untuk tampilan bagian setiap *card* dokter yang dipanggil oleh *file _id.vue* dan dimasukan ke dalam bagian *tab doctor* di dalam profil fasilitas kesehatan. Hasil tampilan halaman profil fasilitas kesehatan terlihat pada Gambar 4.8



Gambar 4.8 Profil Fasilitas Kesehatan

4.2.2 *Use case* Mencari Dokter

Halaman Mencari Dokter adalah halaman untuk mencari dokter berdasarkan nama atau spseialis lalu diarahkan kepada halaman *list* hasil pencarian yang disesuaikan dengan *use case* dan *diagram acitivity* agar alur mudah dibaca dan dimengerti. Untuk masuk ke halaman ini tidak harus *login* terlebih dahulu jika tujuannya hanya mencari informasi fasilitas dokter saja. Berikut *file* dan *component* yang terlibat dalam pengembangan tampilan mencari fasilitas kesehatan:

Tabel 4.3 *File* dan Komponen Mencari Dokter

Component Mencari Fasilitas Kesehatan			
No	*Use case*	*File* Implementasi	deskripsi
1.	Mencari Dokter	*Doctor*/*CardDoctor.vue*	Folder *doctor* ditujukan untuk *component* tampilan dokter. Pada *file CardDoctor.vue* ditujukan untuk tampilan *card* pada tampilan *list – list* dokter.
2.	Mencari Dokter	*Doctor*/PracticeSchedule*Card.vue*	Folder *doctor* ditujukan untuk *component* tampilan dokter. Pada *file* PracticeSchedule*Card.vue* ditujukan untuk tampilan *card* Jadwal Praktik pada tampilan profil dokter.

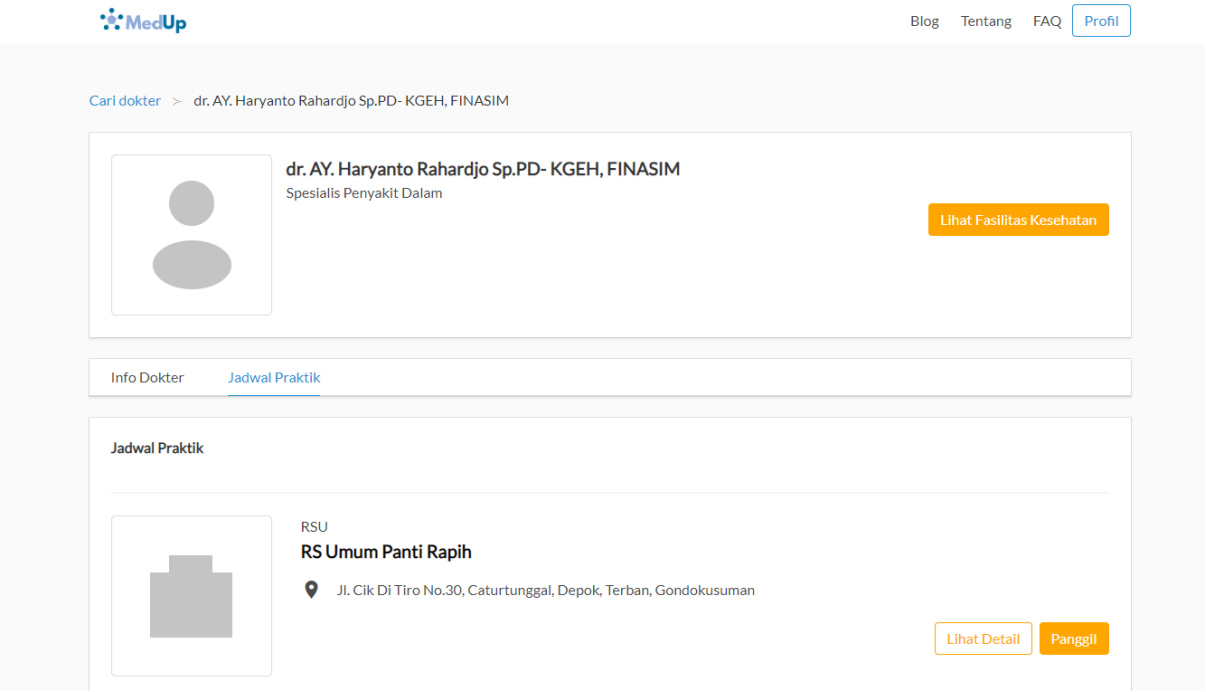
Pada Tabel 4.3 dijelaskan *file* dan *component* yang digunakan dalam tampilan mencari dokter. *file index.vue* yang terdapat pada folder *search-doctor*. Tampilan yang dibuat Pada Gambar 4.9 adalah tampilan untuk bagian *header*, *side filter search*, dan *template card* dokter. Dan *component file CardDoctor.vue* yang terdapat pada folder *doctor* adalah untuk tampilan bagian setiap *card* yang dipanggil oleh *file index.vue* dan dimasukan ke dalam bagian *template card* dokter. Hasil tampilan halaman mencari dokter terlihat pada Gambar 4.9

Gambar 4.9 Halaman Mencari Dokter

Halaman Profil Dokter

Halaman Profil Dokter adalah halaman yang berisi informasi – informasi tentang fasilitas kesehatan terkait seperti informasi umum atau ringkasan dan jadwal praktik. Halaman profil dokter juga adalah alur lanjutan dari halaman mencari dokter yang disesuaikan dengan *use case* dan *diagram acitivity* agar alur mudah dibaca dan dimengerti.

Pada Tabel 4.3 dijelaskan *file* dan *component* yang digunakan dalam tampilan profil dokter. *file _id.vue* yang terdapat pada folder *search-doctor*. Tampilan yang dibuat Pada Gambar 4.10 adalah tampilan yang berisi informasi-informasi yang dibutuhkan pasien seperti nama, spesialis, foto, jadwal praktik yang ada pada profil fasilitas kesehatan. Dan *component file* PracticeSchedule*Card.vue* yang terdapat pada folder *doctor* adalah untuk tampilan bagian setiap *card* jadwal dokter yang dipanggil oleh *file _id.vue* dan dimasukan ke dalam bagian *tab* jadwal praktik di dalam profil dokter. Hasil tampilan halaman profil dokter terlihat pada Gambar 4.10



Gambar 4.10 Profil Dokter

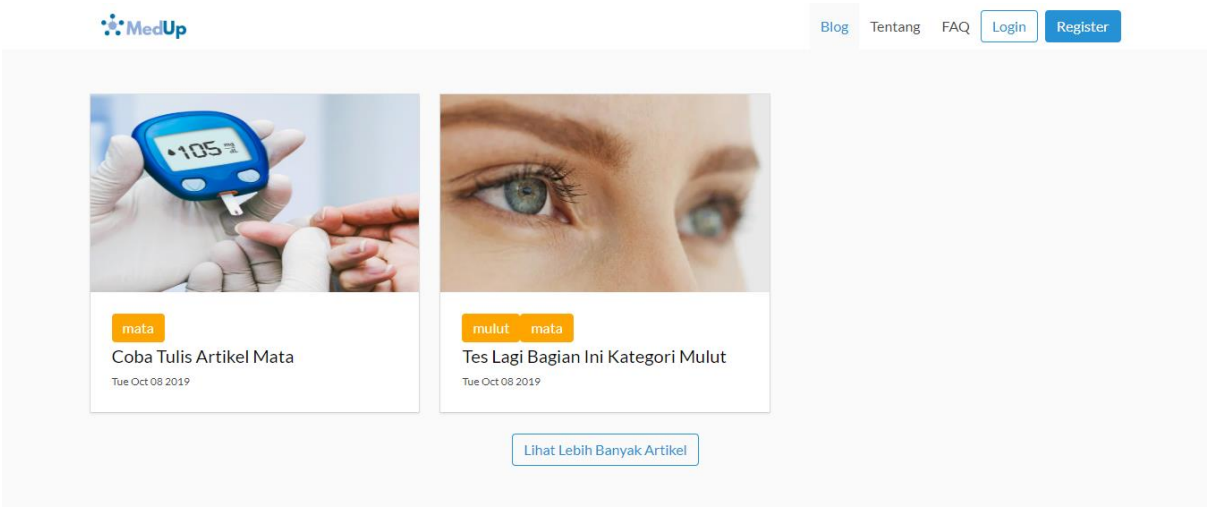
4.2.3 *Use case* Melihat Artikel

Halaman Melihat Artikel adalah halaman *blog* yang berisi artikel–artikel mengenai informasi kesehatan yang disesuaikan dengan *use case* dan *diagram acitivity* agar alur mudah dibaca dan dimengerti. Untuk masuk ke halaman ini tidak harus *login* terlebih dahulu jika tujuannya hanya mencari informasi kesehatan saja. Berikut *component file* yang terlibat dalam pengembangan tampilan melihat artikel:

Tabel 4.4 *File* dan Komponen Melihat Artikel

Component Mencari Fasilitas Kesehatan			
No	*Use case*	*File* Implementasi	deskripsi
1.	Melihat Artikel	*Blog/CardBlog.vue*	Folder *blog* ditujukan untuk *component* tampilan artikel. Pada *file CardBlog.vue* ditujukan untuk tampilan *card* pada tampilan *list – list* artikel.
2.	Melihat Artikel	*Blog/ RecommendedDoctorCard.vue*	Folder *blog* ditujukan untuk *component* tampilan artikel. Pada *file RecommendedDoctorCard.vue* ditujukan untuk tampilan *card* dokter pada tampilan isi artikel.
3.	Melihat Artikel	*Blog/ RecommendedDoctorPhoto.vue*	Folder *blog* ditujukan untuk *component* tampilan artikel. *RecommendedDoctorPhoto.vue* ditujukan untuk tampilan foto didalam *card* dokter pada tampilan isi artikel.

Pada Tabel 4.1 Dijelaskan *file component* yang digunakan dalam tampilan lihat artikel. *file index.vue* yang terdapat pada folder *blog.* Tampilan yang dibuat pada Gambar 4.11 adalah tampilan untuk menampilkan *card–card* artikel yang dibuat. *Component file CardBlog.vue* yang terdapat pada folder *blog* adalah untuk tampilan bagian setiap *card* yang dipanggil oleh *file* index.*vue* dan dimasukan ke dalam bagian *template card* artikel. Hasil tampilan halaman mencari dokter terlihat pada Gambar 4.11



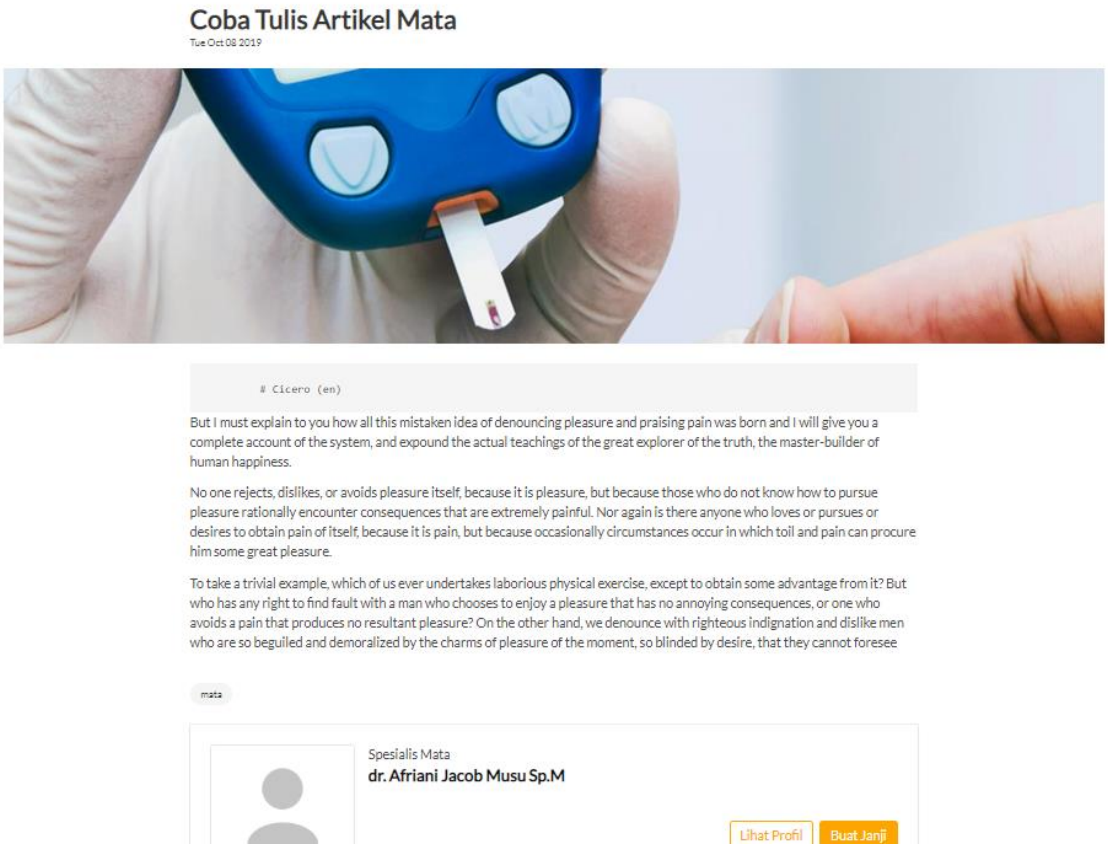
Gambar 4.11 Halaman Lihat Artikel

Halaman Detail Artikel

Halaman Detail Artikel adalah halaman yang berisi informasi–informasi dari masing *card* yang ditampilkan sebelumnya pada halaman *blog*, pada setiap artikel

juga akan ditampilkan rekomendasi dokter berdasarkan kategori artikelnya. Halaman detail artikel juga adalah alur lanjutan dari halaman memelihat artikel yang disesuaikan dengan *use case* dan *diagram acitivity* agar alur mudah dibaca dan dimengerti.

Pada Tabel 4.1 dijelaskan *file component* yang digunakan dalam tampilan detail artikel. *file _slug.vue* yang terdapat pada folder *blog*. Tampilan yang dibuat Pada Gambar 4.12 adalah tampilan yang untuk artikel mulai dari judul, tanggal, isi, sampai rekomendasi dokter. Dan *component file RecommendedDoctorCard.vue* yang terdapat pada folder *blog* adalah untuk tampilan *card* rekomendasi dokter yang ada pada isi setiap artikel yang dipanggil oleh *_slug.vue* dan *file RecommendedDoctorPhoto.vue* adalah tampilan untuk pemanggilan foto dokter di setiap *card* berdasarkan *id* dokternya yang dimasukan ke dalam *card* dokter. Hasil tampilan halaman profil dokter terlihat pada Gambar 4.12

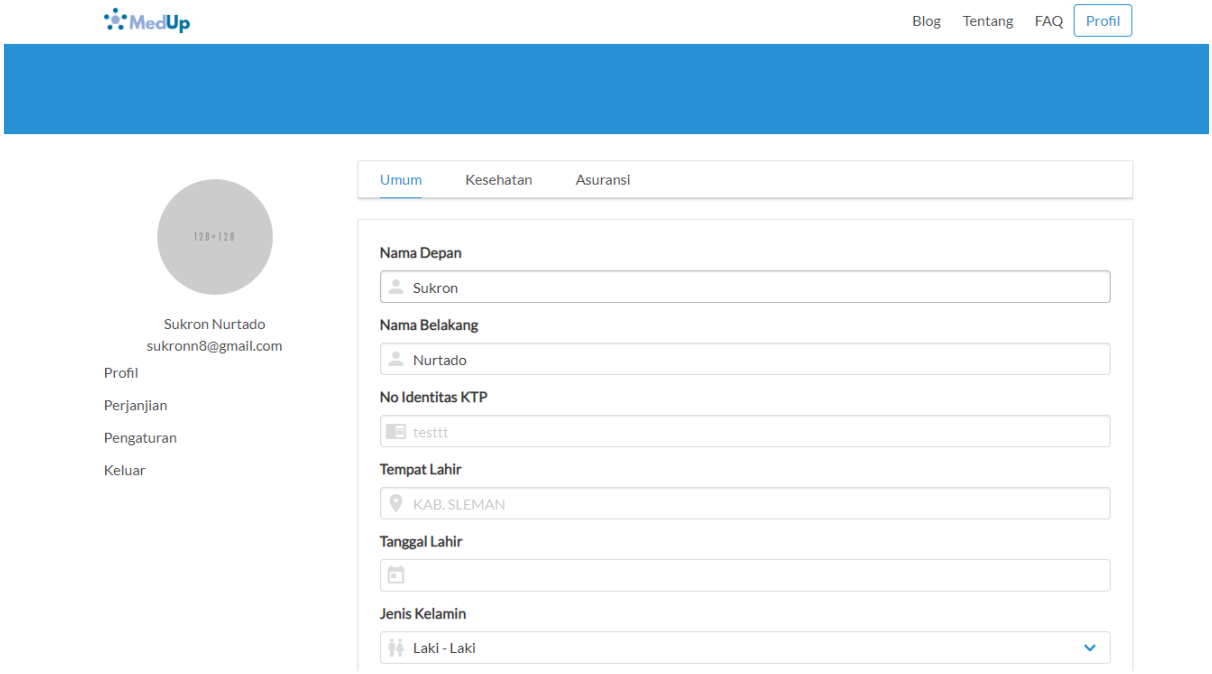


Gambar 4.12 Detail Artikel

4.2.4 *Use case* Menambahkan Data Pengguna

Halaman Menambahkan Data Pengguna adalah halaman untuk pasien ketika selesai *register* atau baru *login* untuk mengisi data dirinya sebelum melakukan *booking* secara *online* yang disesuaikan dengan *use case* dan *diagram acitivity* agar alur mudah dibaca dan dimengerti. Untuk masuk ke halaman diharuskan *login* terlebih dahulu. Pada halaman ini tidak menggunakan *file component* dikarenakan data yang dipersiapkan bisa langsung dimasukan ke dalam halaman.

Pada Tabel 4.1 dijelaskan *file* yang digunakan dalam tampilan menambahkan data pengguna. *File* profil.*Vue.js* adalah tampilan berisi *form–form* pengisian data diri seperti data umum, kesehatan, dan asuransi. Hasil tampilan halaman menambahkan data pengguna terlihat pada Gambar 4.13



Gambar 4.13 Halaman Profil *User*

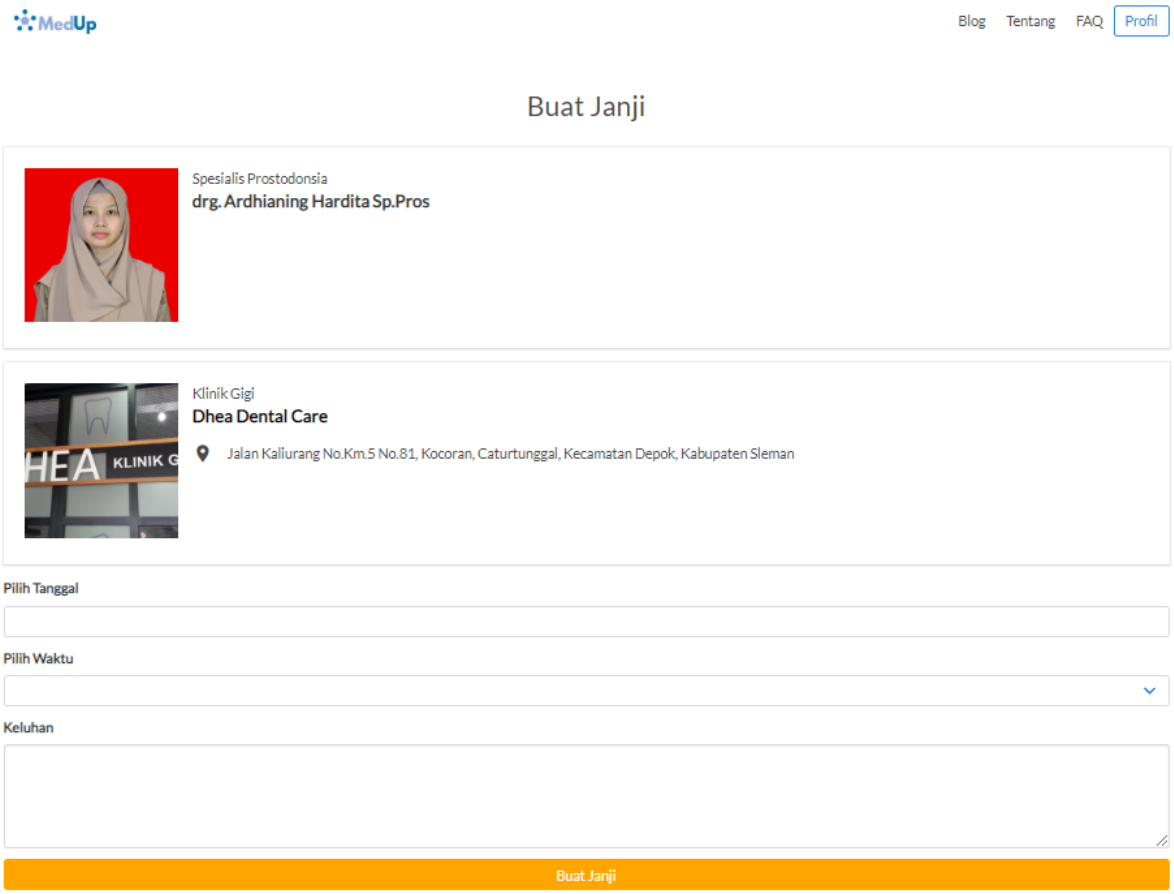
4.2.5 *Use case Booking*

Halaman *Booking* adalah halaman untuk pasien bisa melakukan *booking* secara *online* untuk mendapatkan antrian yang disesuaikan dengan *use case* dan *diagram acitivity* agar alur mudah dibaca dan dimengerti. Untuk masuk ke halaman diharuskan *login* terlebih dahulu. Berikut *file component* yang terlibat dalam pengembangan tampilan *booking*:

Tabel 4.5 *File* dan Komponen Mencari Dokter

Component Mencari Fasilitas Kesehatan			
No	*Use case*	*File* Implementasi	deskripsi
1.	*Booking*	*Book*/*CardDoctorBook.vue*	Folder book ditujukan untuk *component* tampilan *booking*. *CardDoctorBook.vue* ditujukan untuk tampilan *card* dokter pada tampilan *booking*.
2.	*Booking*	*Book*/*CardHealthFacilityBook.vue*	Folder *book* ditujukan untuk *component* tampilan *booking*. *CardHealthFacilityBook.vue* ditujukan untuk tampilan *card* fasilitas kesehatan pada tampilan *booking*.

Pada Tabel 4.5 dijelaskan *file* dan *component* yang digunakan dalam tampilan *booking*. *File CardDoctorBook.vue* adalah tampilan untuk *form* memilih waktu *booking* dan pengisian keluhan pasien ketika pasien sudah memilih rumah sakit dan dokter. *Component file CardDoctor*Book.*Vue.js* yang terdapat pada folder *book* adalah untuk tampilan bagian *card* dokter yang dipanggil oleh *file CardDoctorBook.vue* berdasarkan *id* dokter dan *CardHealthFacilityBook.vue* adalah untuk tampilan *card* fasilitas kesehatannya yang dipanggil juga oleh *CardHealthFacilityBook.vue* untuk ditampilkan berdasarkan *id* faskes yang dipilih. Hasil tampilan halaman *Booking* pada Gambar 4.14

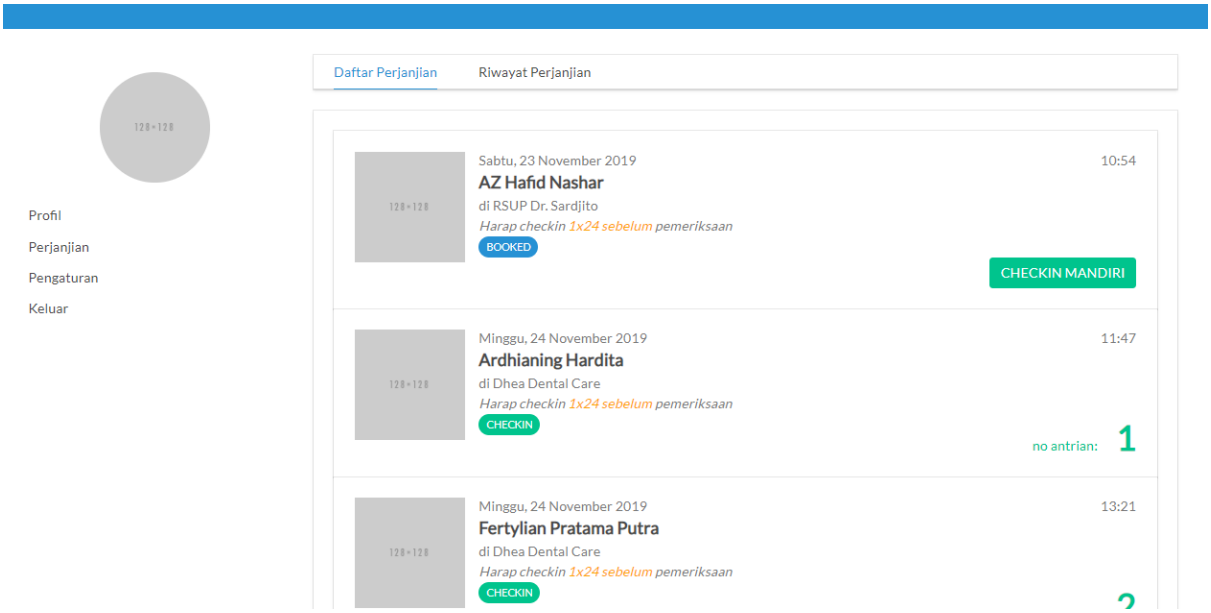


Gambar 4.14 Halaman *Booking Online*

a. Halaman Melihat Antrian

Halaman melihat antrian adalah halaman untuk melihat riwayat perjanjian dengan status yang berbeda–beda seperti status "*BOOKED*" yang berarti pasien baru selesai *booking* dan belum mendapatkan nomor antrian, status "*CHECKIN*" yang berarti pasien sudah mendapatkan nomor antriannya dan bisa datang 30 menit sebelum waktu yang dipilihnya, dan yang terakhir adalah status "ONGOING" yang berarti pasien sedang di cek oleh dokter yang disesuaikan dengan *use case* dan *diagram acitivity* agar alur mudah dibaca dan dimengerti.

Pada Tabel 4.1 dijelaskan *file* yang digunakan dalam tampilan lihat antrian. *file* appointment.*vue* yang adalah tampilan untuk *tab* daftar penjanjian dan riwayat perjanjian yang berisi *list card booking* pasien. Hasil tampilan halaman mencari dokter terlihat pada Gambar 4.15



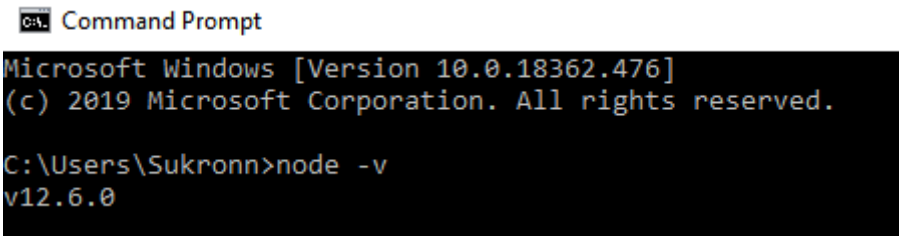
Gambar 4.15 Halaman Lihat Antrian

4.3 Dokumentasi *Angular*8

Strategi Pengembangan dilakukan untuk mengevaluasi pengembangan tampilan khususnya pada *framework* yang digunakan agar menjadi pertimbangan untuk pengembangan selanjutnya menjadi lebih baik lagi. *Angular*8 adalah *framework javascript* versi terbaru dari *Angular* yaitu *framework* typescript yang digunakan untuk membuat aplikasi *web* yang dinamis. Sebelum melakukan instalasi *Angular*, hal–hal yang harus dipersiapkan dan diperhatikan kurang lebih sama seperti pada *Vue.js* sebelumnya, yaitu:

a. Node.js

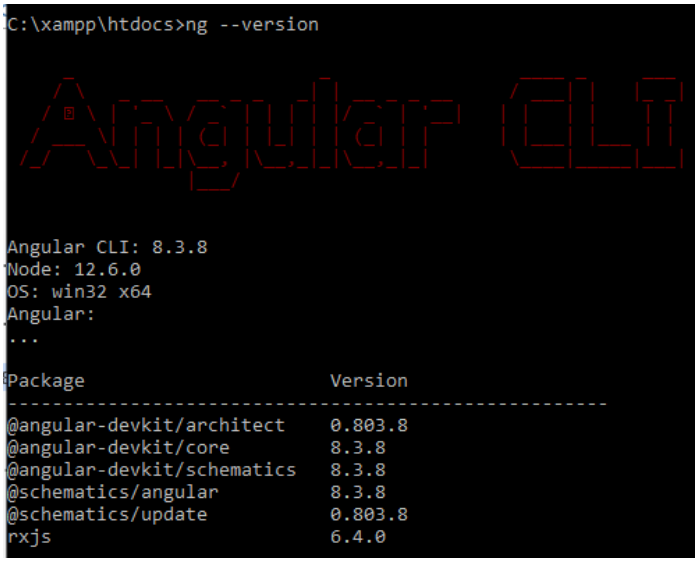
Node.js adalah perangkat lunak yang bersifat *non-blocking* untuk mengembangkan suatu aplikasi berbasis *web* yang ditulis dalam bahasa pemrograman *javascript. Node.js* hadir sebagai pelengkap *javascript* yang berperan sebagai bahasa pemrorgraman yang berjalan di sisi *client* atau *server*, dengan Node.js *javascript* bisa berjalan di sisi *server* juga. (Lutfi. F., 2017). Cara *Install* Node.js yaitu download *installer* (.msi) untuk Node.js pada *https://nodejs.org/en/download/*, lalu jalankan *installer* dan ikuti petunjuk sampai selesai. Adapun cara mengecek Node.js sudah ter*install* atau belum dengan mengecek versi node melalui *command prompt* atau terminal dengan mengetikan *"node-v"*. Berikut contoh gambar pengecekan versi node:



Gambar 4.16 Cek Versi Node.js

b. *Angular* CLI

Angular CLI adalah *Command Line Interface tool* yang digunakan untuk mengisialisasi, membuat atau mengembangkan, dan memelihara aplikasi *Angular*, cara penggunaan melalui terminal atau *command prompt* seperti *npm* yang digunakan oleh *Vue.js* dan React.js. Cara *install Angular* CLI yaitu dengan mengetikan "npm *install* –g *@Angular*/cli" dengan catatan terminal atau *command prompt* sudah memiliki node.js. Adapun cara mengecek *Angular* CLI sudah ter*install* atau belum dengan mengetikan "ng –version" yang berarti melihat versi *Angular* CLI. Berikut contoh gambar pengecekan versi *Angular* CLI:



Gambar 4.17 Cek Versi *Angular* CLI

Hal selanjutnya yang harus diketahui adalah cara untuk membuat projek atau produk dengan menggunakan *Angular*. Cara membuat projek atau produk menggunakan *Angular* dengan mengetikkan "ng new nama_project" pada terminal yang sudah ter*install Angular* dan ikuti proses instalasi sampai selesai.

4.4 Pengujian

Pengujian pada penelitian ini terfokus pada penggunaan *framework javascript Vue.js.* peneliti membuat dua duplikat *website* dengan menggunakan *framework* yang berbeda-beda yaitu *Vue.js* dan *Angular*8 untuk dapat diuji. pengujian didasarkan pada dua perbandingan yaitu dengan membandingkan data–data yang ada pada internet dan membandingkan duplikat *website* yang sudah dibuat.

4.4.1 Metode Black Box

a. Pengujian *Use case* Mencari Fasilitas Kesehatan

Pengujian ini dilakukan oleh pasien untuk mengetahui apakah fungsionalitas yang terdapat pada halaman mencari fasilitas kesehatan dapat berjalan dengan baik atau belum. Pengujian ini meliputi menguji ketika pasien belum melakukan *input* nama fasilitas kesehatan, menguji ketika pasien memasukkan *input* nama yang benar, dan ketika pasien memasukkan *input* nama yang salah. Adapun pengujian di halaman mencari fasilitas kesehatan dapat dilihat pada Tabel 4.6

Tabel 4.6 Pengujian Mencari Fasilitas Kesehatan

Kasus dan Hasil Uji *Admin* (Data Normal)			
Skenario	Yang diharapkan	Hasil	Kesimpulan
Tidak memberi *input*an pada kolom pencarian nama fasilitas kesehatan.	Muncul *list* semua nama fasilitas kesehatan	Muncul *list* semua nama fasilitas kesehatan	Berhasil
Memberi *input*an "Dental" pada kolom pencarian nama fasilitas kesehatan	Muncul *list* dengan mengandung nama "Dental" berdasarkan kolom pencarian	Muncul *list* dengan mengandung nama "Dental" berdasarkan kolom pencarian	Berhasil
Kasus dan Hasil Uji *Admin* (Data Salah)			
Skenario	Yang Diharapkan	Hasil	Kesimpulan
Memberi *input*an "1234" pada kolom pencarian fasilitas kesehatan	Muncul halaman dengan pesan" Mohon maaf	Muncul halaman dengan pesan "Mohon maaf	Berhasil

	pencarian tidak ditemukan“	pencarian tidak ditemukan“	

b. Pengujian *Use case* Mencari Dokter

Pengujian ini dilakukan oleh pasien untuk mengetahui apakah fungsionalitas yang terdapat pada halaman mencari dokter dapat berjalan dengan baik atau belum. Pengujian ini meliputi menguji ketika pasien belum melakukan *input* nama dokter, menguji ketika pasien memasukkan *input* nama yang benar, dan ketika pasien memasukkan *input* nama yang salah. Adapun pengujian di halaman mencari dokter dapat dilihat pada Tabel 4.7.

Tabel 4.7 Pengujian *Login Admin*

Kasus dan Hasil Uji *Admin* (Data Normal)			
Skenario	Yang diharapkan	Hasil	Kesimpulan
Tidak memberi *input*an pada kolom pencarian nama dokter.	Muncul *list* semua nama dokter	Muncul *list* semua dokter	Berhasil
Memberi *input*an “Fenti” pada kolom pencarian nama dokter	Muncul *list* dengan mengandung nama “Fenti” berdasarkan kolom pencarian	Muncul *list* dengan mengandung nama “Fenti” berdasarkan kolom pencarian	Berhasil
Kasus dan Hasil Uji *Admin* (Data Salah)			
Skenario	Yang Diharapkan	Hasil	Kesimpulan
Memberi *input*an “1234” pada kolom pencarian dokter	Muncul halaman dengan pesan“ Mohon maaf pencarian tidak ditemukan“	Muncul halaman dengan pesan“ Mohon maaf pencarian tidak ditemukan“	Berhasil

c. Pengujian *Use case* Menambahkan Data Pengguna

Pengujian ini dilakukan oleh pasien untuk mengetahui apakah fungsionalitas yang terdapat pada halaman menambahkan data pengguna dapat berjalan dengan baik atau belum. Pengujian ini meliputi menguji ketika pasien Meng*input*kan data yang salah, dan meng*input*kan data yang benar. Adapun pengujian di halaman menambahkan data pengguna dapat dilihat pada Tabel 4.8.

Tabel 4.8 Pengujian Menambahkan Data Pengguna

Kasus dan Hasil Uji Pasien			
Skenario	Yang diharapkan	Hasil	Kesimpulan
Memberi *input*an yang tidak sesuai.	Menampilkan pesan "*input*an tidak sesuai"	Menampilkan pesan" *input*an tidak sesuai"	Berhasil
Memberi *input*an yang sesuai	Menampilkan pesan "data berhasil disimpan"	Menampilkan pesan "data berhasil disimpan"	Berhasil

d. Pengujian *Use case Booking*

Pengujian ini dilakukan oleh pasien untuk mengetahui apakah fungsionalitas yang terdapat di halaman *booking* dapat berjalan dengan baik atau belum. Pengujian ini meliputi menguji ketika pasien meng*input*kan jadwal dan keluhan. Adapun pengujian *booking* dapat dilihat pada Tabel 4.9.

Tabel 4.9 Pengujian Halaman Utama

Kasus dan Hasil Uji *User* dan *Admin*			
Skenario	Yang diharapkan	Hasil	Kesimpulan
Baru memilih dokter atau rumah sakit	Diarahkan ke halaman rumah sakit atau dokter (dengan situasi belum terpilih)	Diarahkan ke halaman rumah sakit atau dokter (dengan situasi belum terpilih)	Berhasil
Memilih tanggal yang salah	Tidak muncul waktu praktik dokter	Tidak muncul waktu praktik dokter	Berhasil

Belum meng*input* isi keluhan	Muncul pesan "mohon isi keluhan anda"	"mohon isi keluhan anda"	Berhasil

e. Pengujian *Use case* Melihat Antrian

Pengujian ini dilakukan oleh pasien untuk mengetahui apakah fungsionalitas yang terdapat di halaman antrian dapat berjalan dengan baik atau belum. Pengujian ini meliputi menguji ketika pasien masuk ke sub menu perjanjian yang ada pada halaman profil, menguji ubah status dari *BOOKED* ke *CHECKIN*, dan membatalkan perjanjian. Adapun pengujian dapat dilihat pada Tabel 4.10.

Tabel 4.10 Pengujian Melihat Antrian

Kasus dan Hasil Uji Pasien			
Skenario	Yang diharapkan	Hasil	Kesimpulan
Masuk ke *sub menu* perjanjian	Menampilkan perjanjian - perjanjian dengan status "*BOOKED"*, "*CHECKIN"*, dan "ONGOING" yang ada pada *tab* Daftar Perjanjian	Menampilkan perjanjian - perjanjian dengan status "*BOOKED"*, "*CHECKIN"*, dan "ONGOING" yang ada pada *tab* Daftar Perjanjian	Berhasil
Menekan tombol *CHECKIN* MANDIRI pada perjanjanjian dengan status "*BOOKED"*	Mengubah status perjanjian menjadi "*CHECKIN"* dan mendapat nomor antrian	Mengubah status perjanjian menjadi "*CHECKIN"* dan mendapat nomor antrian	Berhasil
Menekan tombol batalkan perjanjian	Menghapus perjanjian dari *tab*	Menghapus perjanjian dari *tab*	Berhasil

	Daftar Perjanjian dan memindahkan nya ke *tab* Riwayat Perjanjian dengan status "*ABORTED*"	Daftar Perjanjian dan memindahkan nya ke *tab* Riwayat Perjanjian dengan status "*ABORTED*"	

f. Pengujian *Use case* Melihat Artikel

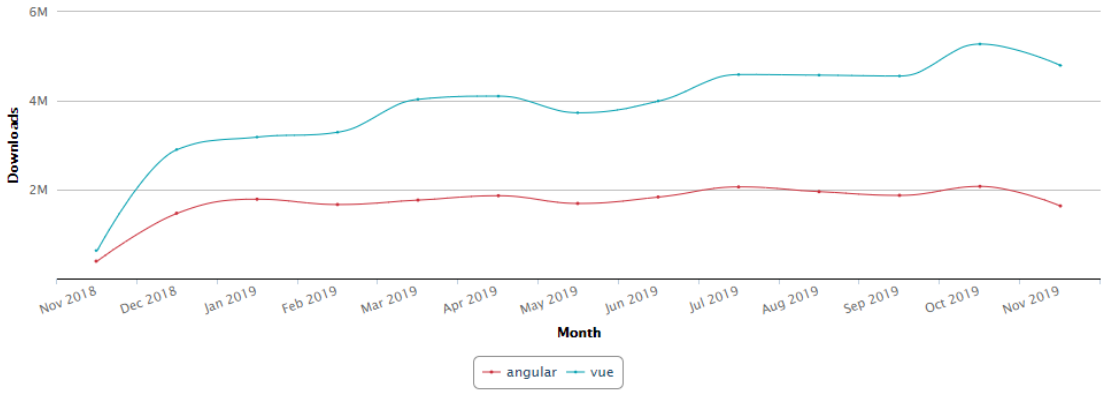
Pengujian ini dilakukan oleh pasien untuk mengetahui apakah fungsionalitas yang terdapat pada halaman *blog* artikel dapat berjalan dengan baik atau belum. Pengujian ini meliputi membuka halaman artikel. Adapun pengujian di halaman menambahkan data pengguna dapat dilihat pada Tabel 4.11.

Tabel 4.11 Pengujian Menambahkan Data Pengguna

Kasus dan Hasil Uji Pasien			
Skenario	Yang diharapkan	Hasil	Kesimpulan
Menekan atau klik salah satu *card* artikel	Membuka halaman isi artikel dari *card* tersebut	Membuka halaman isi artikel dari *card* tersebut	Berhasil

4.4.2 *Download Results*

Pada pembahasan *download result*, peneliti menggunakan data dari *npm–stat* yaitu halaman yang menampilkan grafik paket apapun yang ada pada *npm*, peneliti akan menampilkan *statistics* data *download* untuk paket *framework Vue.js* dan *Angular* berdasarkan tanggal 26 November 2018 sampai tanggal 26 November 2019 (Npm-stat, 2018).



Gambar 4.18 Grafik *Download Vue.js* dan *Angular* Per Bulan

Pada *Vue.js*, total *download npm* untuk paket *Vue.js* adalah 49.657.490 (empat puluh sembilan juta enam ratus lima puluh tujuh ribu empat ratus sembilan pulun) *download* pengguna. Pada *Angular*, total *download npm* untuk paket *Angular* adalah 22.045.437 (dua puluh dua juta empat puluh lima ribu empat ratus tiga puluh tujuh) *download* pengguna.

Dapat disimpulkan bahwa pengguna lebih banyak total mengunduh *Vue.js* dari pada *Angular* sebagai *framework* untuk projek atau produk menurut data *npm-stat* selama satu tahun terakhir, total *Vue.js* js mencapai dua kali lebih banyak daripada *Angular* (Npm-stat, 2018).

4.4.3 *Most Popularity*

Pada pembahasan *most popularity*, peneliti akan mengevaluasi popularitas diantara *framework Vue.js* dan *Angular*. Peneliti menggunakan data dari *stateofjs* 2018 yaitu *website* yang menampilkan hasil survei yang dilakukan selama kurang lebih 2 bulan di tahun 2018. Pada survei yang dilakukan, terbagi menjadi lima jenis untuk kategorinya mulai dari belum pernah mendengar, pernah mendengar tetapi belum tertarik, pernah mendengar dan sedang mempelajari, pernah menggunakan tetapi tidak ingin menggunakannya kembali, dan pernah menggunakan dan akan menggunakannya kembali. Koresponden yang telah mengisi survei kurang lebih 20.000 (dua puluh ribu) koresponden. Adapun tabel Tabel 4.12 mengenai popularitas *framework* menurut *stateofjs* tahun 2018 (Stateofjs, 2018):

Tabel 4.12 Popularitas *Framework* Tahun 2018

Vue.js	*Angular*	Kategori
263 koresponden	0 koresponden	Belum pernah mendengar
4122 koresponden	6417 koresponden	Pernah mendengar tetapi belum tertarik
9395 koresponden	2089 koresponden	Pernah mendengar dan sedang mempelajari

564 koresponden	6826 koresponden	Pernah menggunakan tetapi tidak ingin menggunakannya kembali
5810 koresponden	4817 koresponden	Pernah menggunakan dan akan menggunakannya kembali

Pada popularitas *Vue.js*, koresponden yang belum pernah mendengar tentang *framework Vue.js* berjumlah 263 atau 1.3% dari total kurang lebih 20.000 koresponden. Pada tahun 2018 koresponden paling banyak menjawab pernah mendengar dan sedang mempelajari *framework Vue.js* dengan total 9395 koresponden atau sekitar 46.6% dari total koresponden yang menjawab. *Vue.js* mendapatkan respon yang positif karena sepuluh kali lipat koresponden yang pernah menggunakan *Vue.js* ingin menggunakannya kembali.

Pada popularitas *Angular*, semua koresponden yang mengisi sudah pernah mendengar *Angular*. Sayangnya pada tahun 2018, *Angular* mendapat respon negatif walaupun termasuk kedalam 5 *framework* terbaik menurut suvey *stateofjs* tahun 2018, respon negatif karena koresponden paling banyak menjawab pernah mendengar tetapi tidak ingin menggunakannya kembali yaitu sebanyak 6826 koresponden atau sekitar 33.8% daripada respon positif sebanyak 4817 koresponden atau sekitar 23.9%.

Dapat disimpulkan bahwa koresponden lebih puas dengan *framework Vue.js* dari pada *Angular* sebagai *framework* untuk projek atau produk. Dan di tahun 2018 banyak koresponden yang sedang tertarik untuk mempelajari *Vue.js* yaitu sebanyak 9395 koresponden atau sekitar 46.6% dibandingkan dengan *Angular* yaitu sebanyak 2089 atau sekitar 10.4% koresponden.

4.4.4 Build Times

Pada pembahasan *build times*, peneliti akan tingkat kecepatan suatu *framework* berdasarkan data yang ada pada *rawgit* Tabel 4.13 mengenai popularitas *framework* tahun 2018 (Rawgit, 2019).

Tabel 4.13 *Built Times*

Nama	*Vue.js*-v2.6.2	*Angular*-v8.0.1
create rows creating 1,000 rows	162.33.4 (1.00)	164.37.6 (1.01)

Nama	*Vue.js*-v2.6.2	*Angular*-v8.0.1
replace all rows updating all 1,000 rows (5 warmup runs).	128.22.5 (1.00)	134.42.4 (1.05)
partial update updating every 10th row for 1,000 rows (3 warmup runs). 16x CPU slowdown.	230.96.1 (1.57)	147.07.3 (1.00)
select row highlighting a selected row. (5 warmup runs). 16x CPU slowdown.	104.71.7 (3.83)	27.32.3 (1.00)
swap rows swap 2 rows for table with 1,000 rows. (5 warmup runs). 4x CPU slowdown.	66.61.6 (1.00)	423.23.5 (6.36)
remove row removing one row. (5 warmup runs).	44.61.0 (1.11)	40.31.5 (1.00)
create many rows creating 10,000 rows	1,251.320.8 (1.00)	1,299.412.7 (1.04)
append rows to large table appending 1,000 to a table of 10,000 rows. 2x CPU slowdown	288.62.2 (1.05)	275.14.9 (1.00)
clear rows clearing a table with 1,000 rows. 8x CPU slowdown	160.44.6 (1.00)	244.515.3 (1.52)
slowdown geometric mean	1.24	1.30

4.4.5 *Performance Times*

Pada *Performance Times*, terdapat batasan yaitu hanya menampilkan dua *use case* saja, yaitu mencari dokter dan mencari rumah sakit. Adapun evaluasi halaman adalah untuk melihat *performance* test yang dihasilkan dari masing–masing *framework*. *Tools* yang digunakan untuk melihat *performance website* adalah *web*pagetest (*Web*pagetest, 2019).

Sebelum melakukan evaluasi, peneliti meng*hosting website* MedUp menggunakan *Vue.js* dan *website* MedUp menggunakan *Angular* pada *netlif. Netlify* adalah salah satu *platform* yang menyediakan layanan *build tools* sekaligus *continuous deployment* (Anshory, 2019).

Hasil dari hostingan menggunakan *netlify* adalah sebagai berikut:

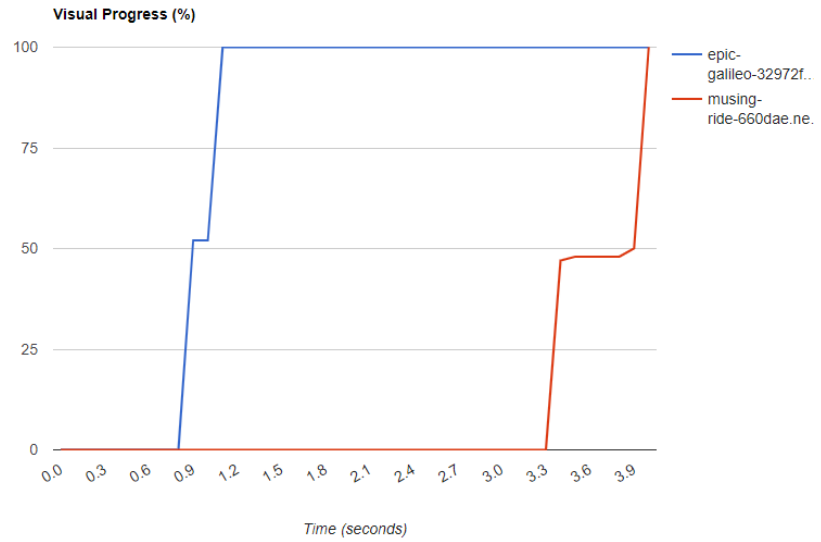
a. https://musing-ride-660dae.netlify.com adalah duplikat *website* MedUp menggunakan *Angular* s8
b. https://epic-galileo-32972f.netlify.com adalah duplikat *website* MedUp menggunakan *Vue.js*

Berikut evaluasi perbandingan dari tampilan yang sudah di buat pada duplikat *website* menggunakan *Angular* 8 dan *Vue.js*

a. Tampilan Utama *Website*

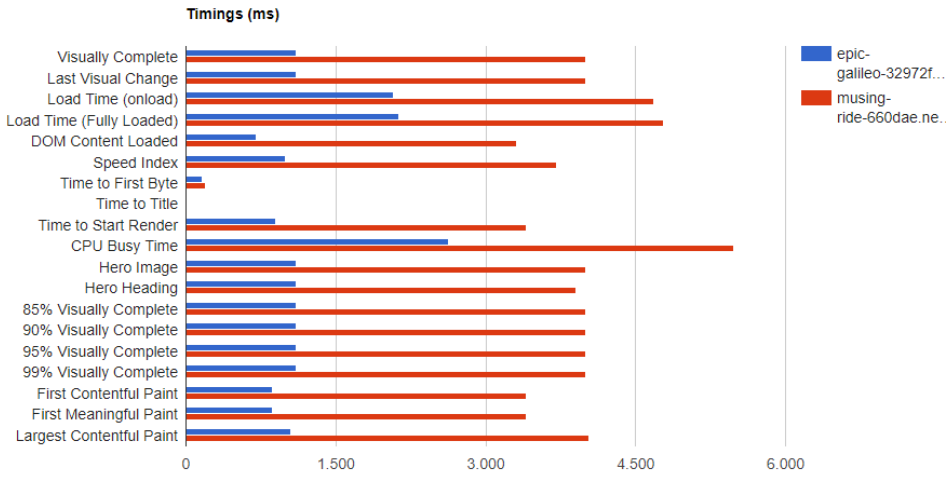
Pada tampilan utama berisi *input search* dokter atau rumah sakit yang akan diarahkan ke halaman *list* dokter ataupun fasilitas kesehatan. Selain *input search* dokter atau fasilitas kesehatan, terdapat informasi – informasi mengenai MedUp. Tampilan akan *visual progress* nya. Berikut hasil dari tampilan utama *website*:

Untuk hasil dari tampilan utama *website* menggunakan *Vue.js*, *visual progress* untuk dapat menampikan halaman utama secara penuh adalah 0.9 – 1.1 detik. Sedangkan hasil dari tampilan utama *website* menggunakan *Angular* 8, *visual progress* untuk dapat menampikan halaman utama secara penuh adalah 3.4 – 4.0 detik. Adapun grafik dari visual *progress* dapat dilihat pada Gambar 4.19



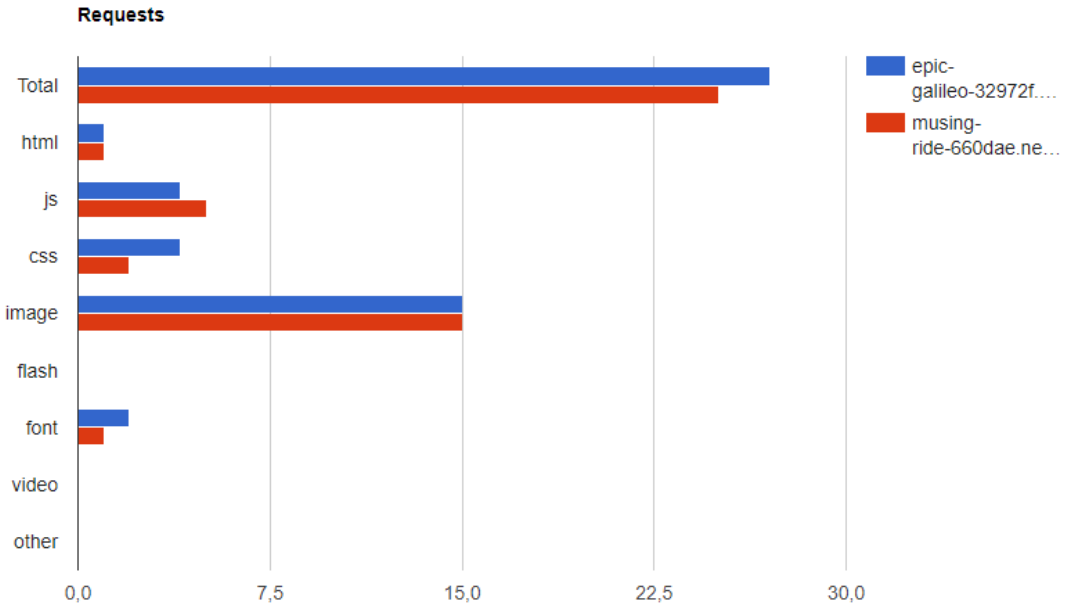
Gambar 4.19 Visual *Progress* Tampilan Utama

Pada Gambar 4.19 terlihat kecepatan dari *framework Vue.js* lebih cepat dibandingkan dengan *Angular* 8. Dan untuk detail *progress* yang dilakukan akan dilihatkan pada Gambar 4.20.



Gambar 4.20 Detail *Progress* Tampilan Utama

Hampir setiap kategori *Vue.js* jauh lebih cepat dibandingkan *Angular* 8, padahal untuk total *request Vue.js* lebih banyak dibandingkan dengan *Angular* 8, Gambar 4.21 untuk detail *request*.



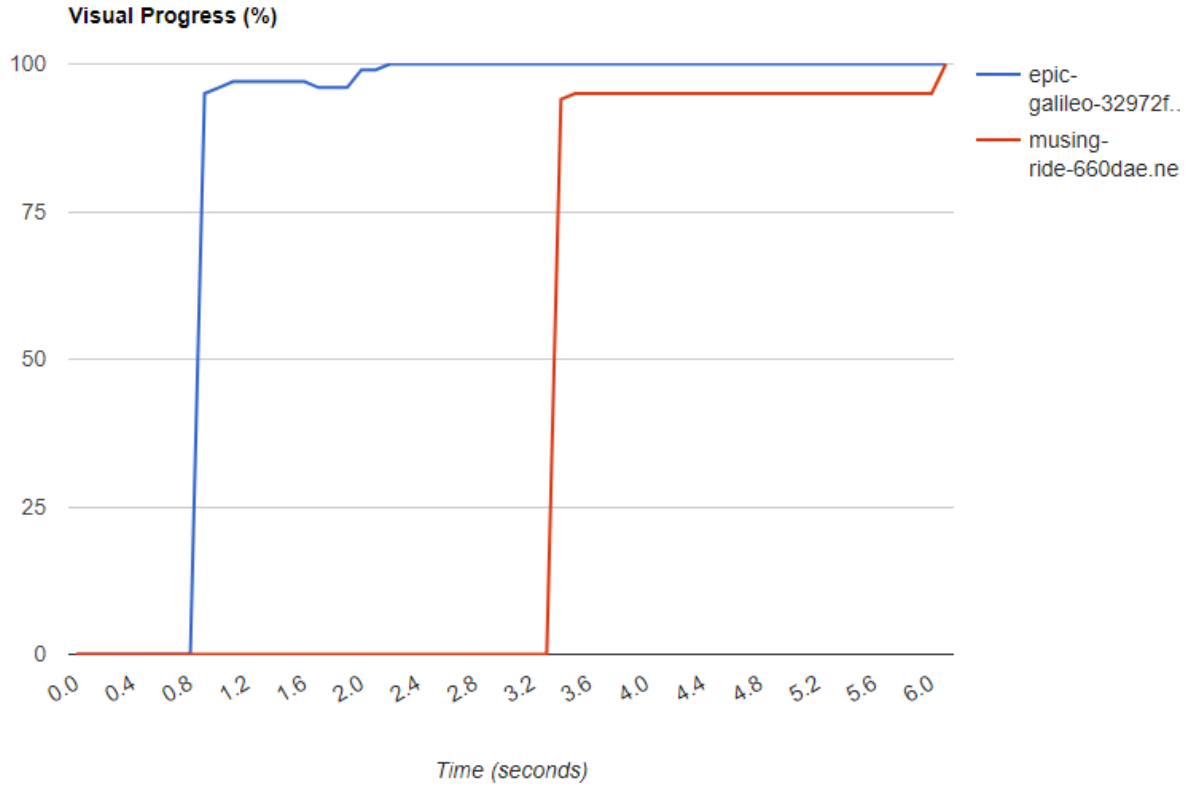
Gambar 4.21 Detail *Request* Tampilan Utama

Dengan demikian dapat disimpulkan untuk visual *progress* tampilan utama, duplikat *website* menggunakan *Vue.js* jauh lebih cepat dibandingkan menggunakan *Angular* 8. Walaupun total untuk *request Vue.js* lebih banyak yaitu 27 *requests* dibandingkan dengan *Angular* 8 yaitu 25 *requests.*

b. *Use case* Mencari Dokter

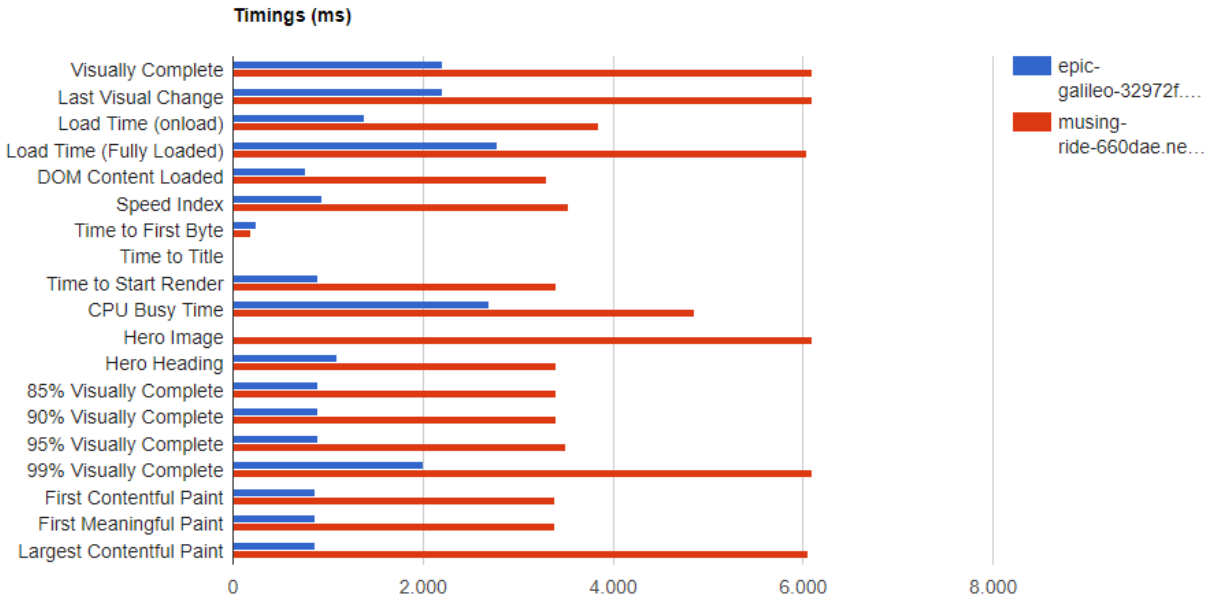
Pada tampilan mencari dokter berisi *input search* dokter, *filter* pada *sidebar* dan *list* untuk *card* dari dokter. Tampilan akan dievaluasi berdasarkan visual *progress* nya. Berikut hasil dari tampilan mencari dokter.

Untuk hasil dari *use case* mencari dokter menggunakan *Vue.js*, *visual progress* untuk dapat menampikan halaman mencari dokter secara penuh adalah 0.9 – 2.2 detik. Sedangkan hasil dari *use case* mencari dokter menggunakan *Angular* 8, *visual progress* untuk dapat menampikan halaman mencari dokter secara penuh adalah 3.4 - 6.1 detik. Adapun grafik dari visual *progress* dapat dilihat pada Gambar 4.22



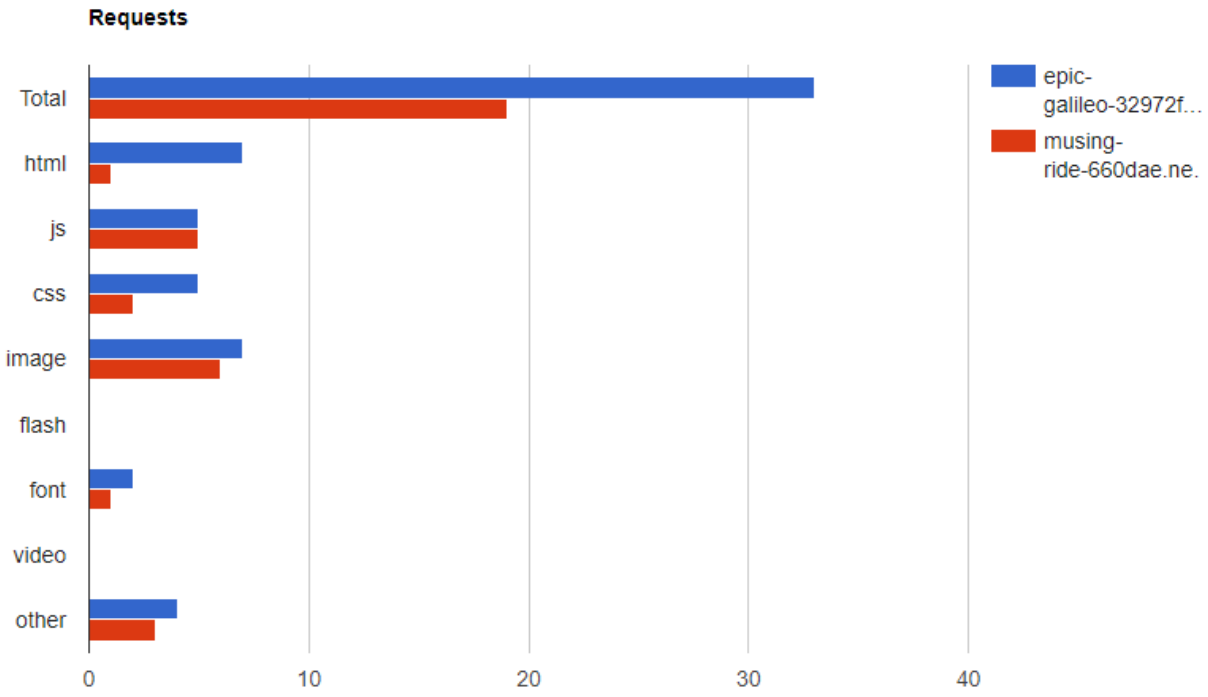
Gambar 4.22 Visual *Progress* Mencari Dokter

Pada Gambar 4.22 terlihat kecepatan dari *framework Vue.js* lebih cepat dibandingkan dengan *Angular* 8. Dan untuk detail *progress* yang dilakukan akan dilihatkan pada Gambar 4.23



Gambar 4.23 Detail *Progress* Mencari Dokter

Hampir setiap kategori *Vue.js* jauh lebih cepat dibandingkan *Angular* 8, padahal untuk total *request Vue.js* lebih banyak dibandingkan dengan *Angular* 8, Gambar 4.24 untuk detail *request*.



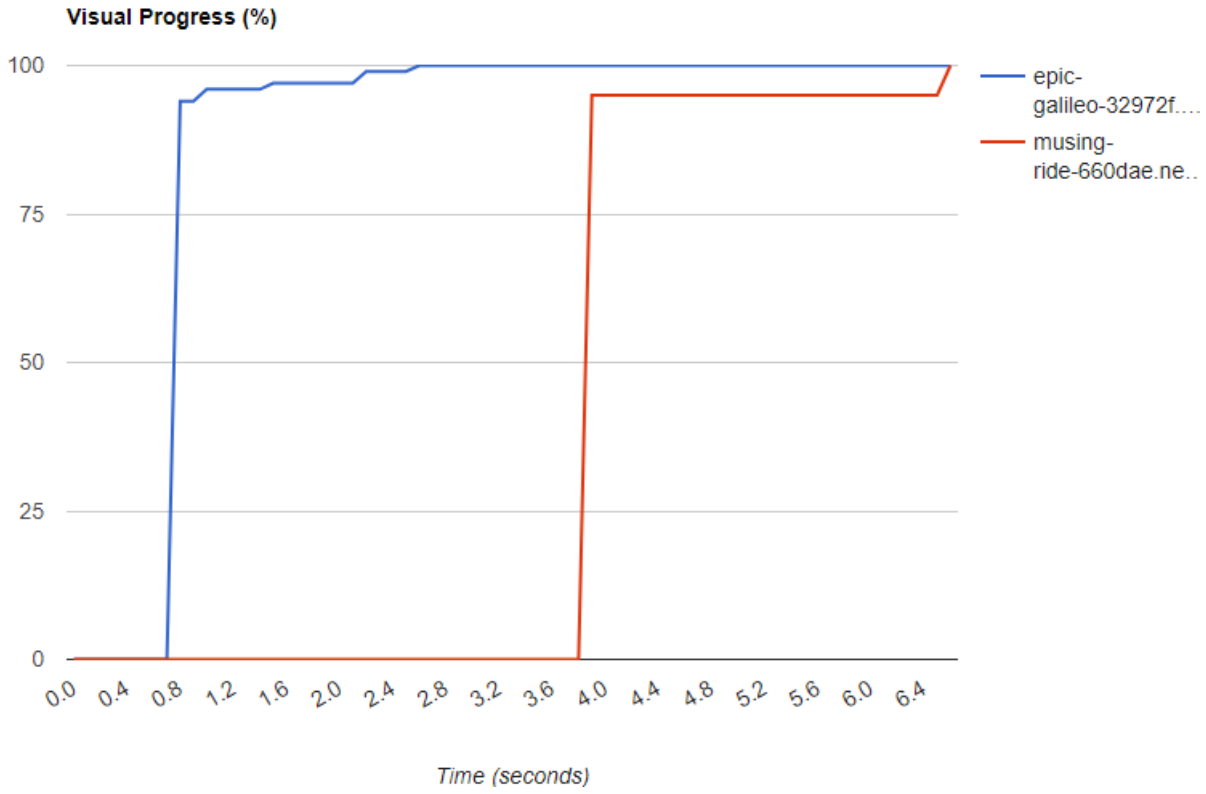
Gambar 4.24 Detail *Request* Mencari Dokter

Dengan demikian dapat disimpulkan untuk visual *progress* tampilan utama, duplikat *website* menggunakan *Vue.js* jauh lebih cepat dibandingkan menggunakan *Angular* 8. Walaupun total untuk *request Vue.js* lebih banyak yaitu 33 *requests* dibandingkan dengan *Angular* 8 yaitu 19 *requests.*

c. *Use case* Mencari Fasilitas Kesehatan

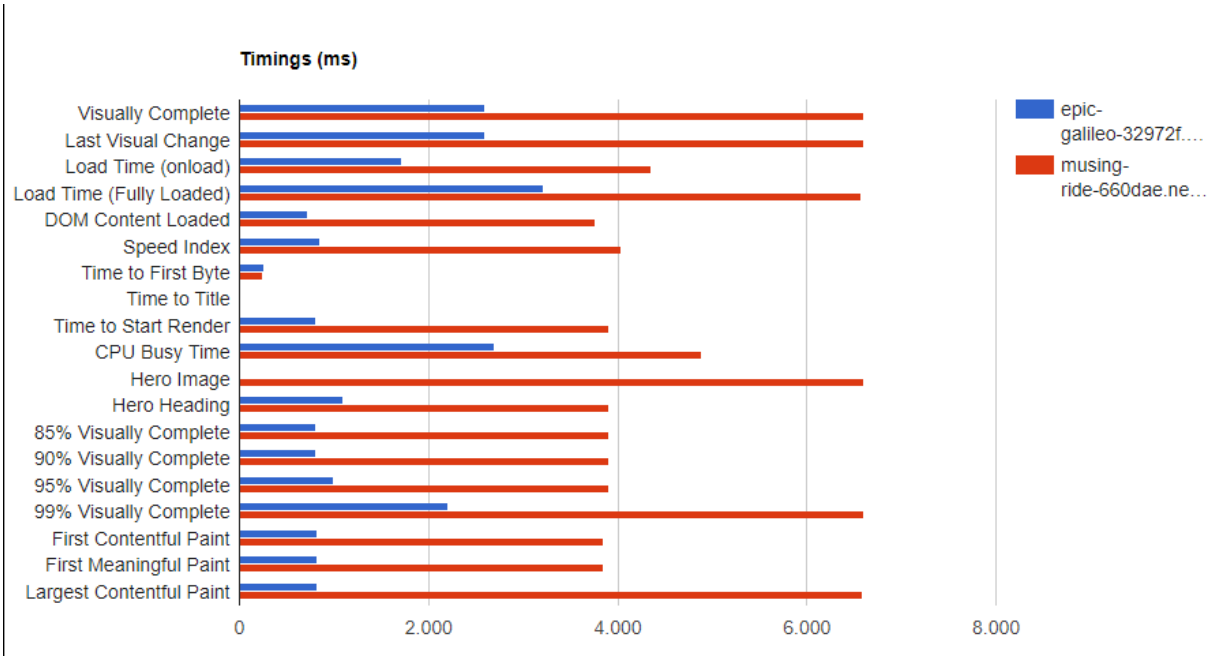
Pada tampilan mencari fasilitas kesehatan berisi *input search* fasilitas kesehatan, *filter* pada *sidebar* dan *list* untuk *card* dari fasilitas kesehatan. Tampilan akan dievaluasi berdasarkan visual *progress* nya. Berikut hasil dari tampilan mencari fasilitas kesehatan.

Untuk hasil dari *use case* mencari fasilitas kesehatan menggunakan *Vue.js*, *visual progress* untuk dapat menampikan halaman mencari fasilitas kesehatan secara penuh adalah 1.0 – 3.0 detik. Sedangkan hasil dari *use case* mencari fasilitas kesehatan menggunakan *Angular* 8, *visual progress* untuk dapat menampikan halaman mencari fasilitas kesehatan secara penuh adalah 4.0 - 7.0 detik. Adapun grafik dari visual *progress* dapat dilihat pada Gambar 4.25



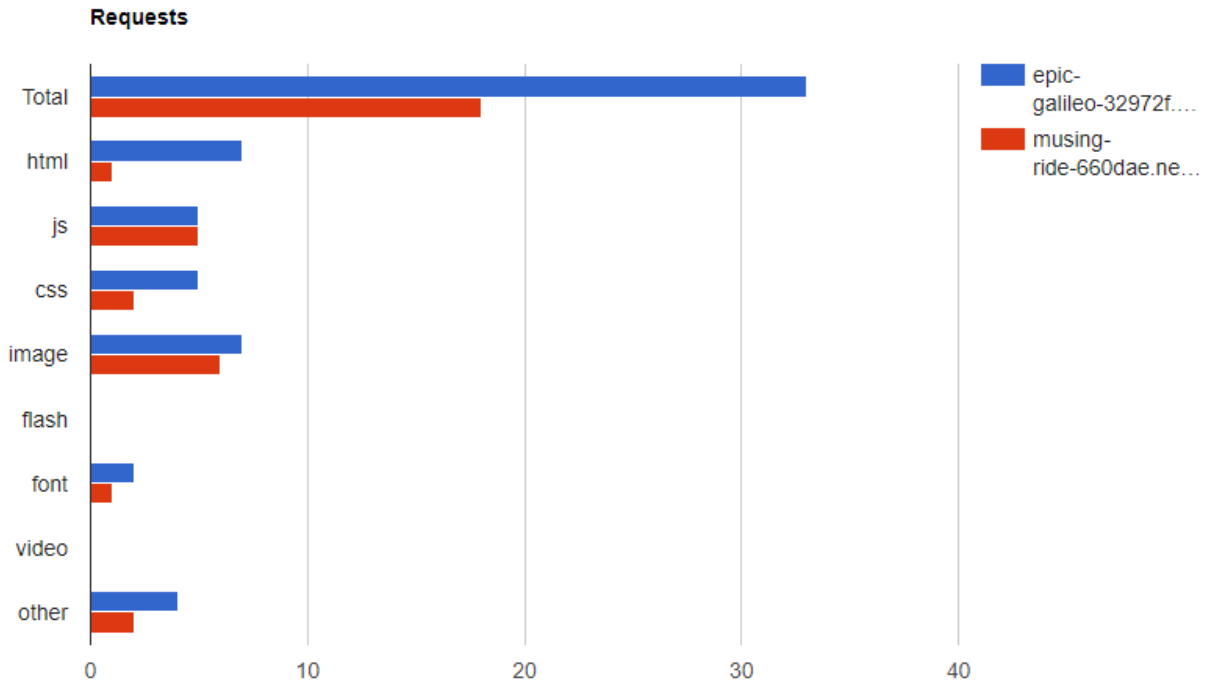
Gambar 4.25 Visual *Progress* Mencari Fasilitas Kesehatan

Pada Gambar 4.25 terlihat kecepatan dari *framework Vue.js* lebih cepat dibandingkan dengan *Angular* 8. Detail *progress* yang dilakukan akan dilihatkan pada Gambar 4.26.



Gambar 4.26 Detail *Progress* Mencari Fasilitas Kesehatan

Hampir setiap kategori *Vue.js* jauh lebih cepat dibandingkan *Angular* 8, padahal untuk total *request Vue.js* lebih banyak dibandingkan dengan *Angular* 8, Gambar 4.27 untuk detail *request*.



Gambar 4.27 Detail *Request* Mencari Fasilitas Kesehatan

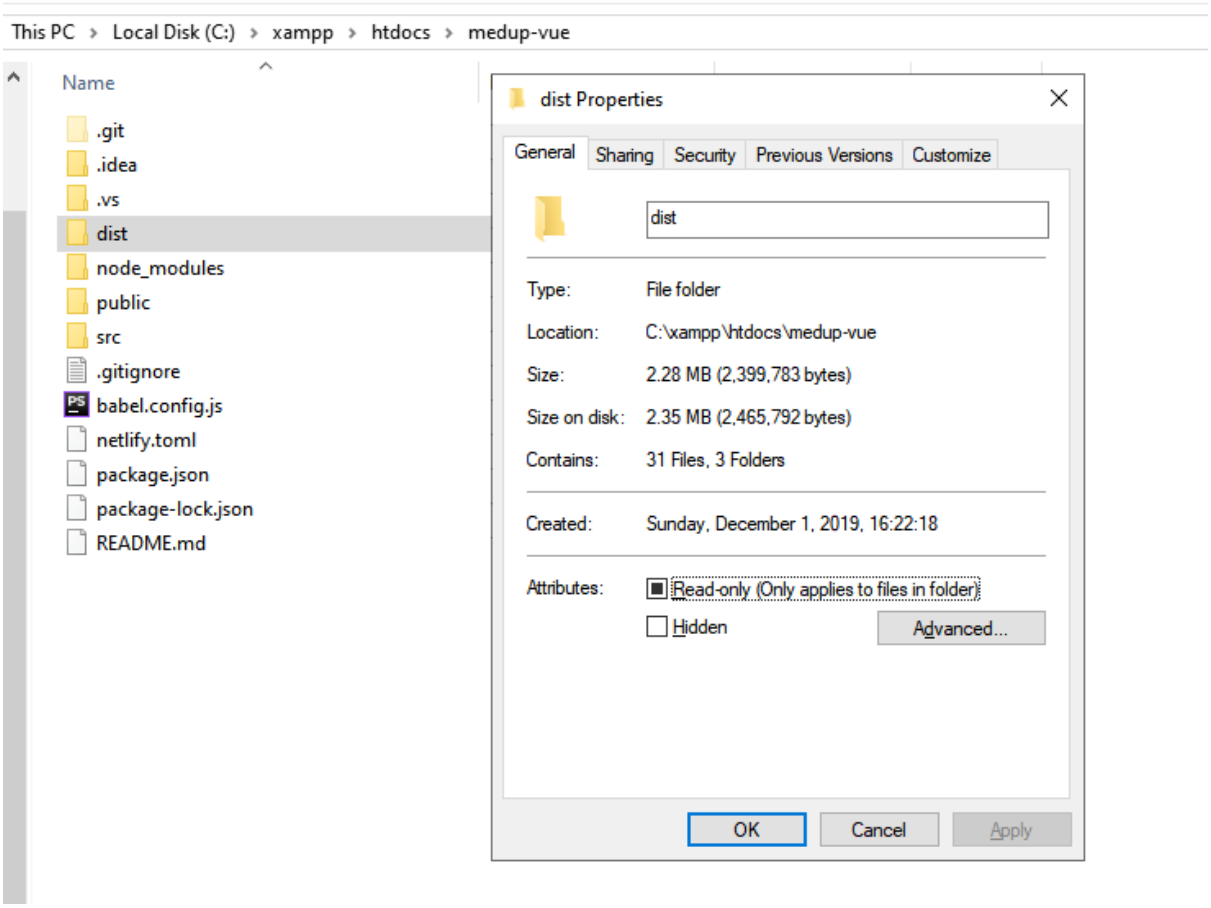
Dengan demikian dapat disimpulkan untuk visual *progress* tampilan utama, duplikat *website* menggunakan *Vue.js* jauh lebih cepat dibandingkan menggunakan *Angular* 8. Walaupun total untuk *request Vue.js* lebih banyak yaitu 33 *requests* dibandingkan dengan *Angular* 8 yaitu 18 *requests.*

4.4.6 *Size Metrics*

Pada pembahasan *size metrics*, peneliti menggunakan "*run build*" untuk mengetahui seberapa besar *file* dari masing–masing *framework* yang berjalan dan terpakai. Berikut penjelasan untuk size metrics dari masing–masing *framework* yang dijalankan:

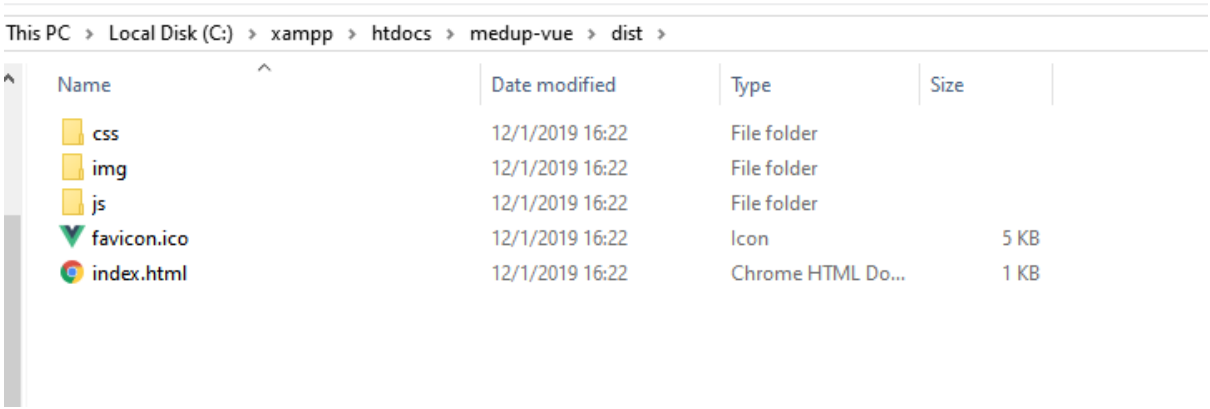
a. *Vue.js*

Pada *Vue.js* untuk membuat folder *dist* yaitu dengan cara menuliskan perintah "*npm run build*". Tunggu sebentar sampai build selesai, lalu buka folder dan lihat berapa ukuran foldernya. Adapun gambar dari folder *dist Vue.js*:



Gambar 4.28 Dist Folder Pada MedUp *Vue.js*

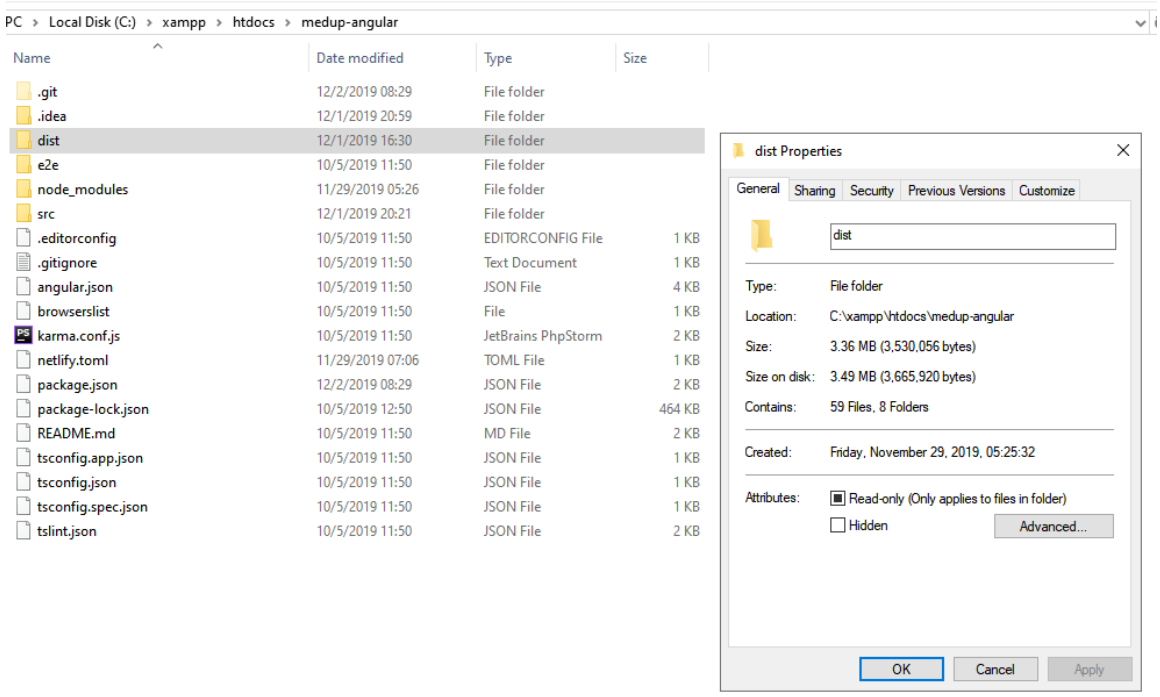
Pada Gambar 4.28 telah diketahui bahwa *npm* telah membangun projek MedUp *Vue.js* dengan ukuran 2.28 MB. Dan untuk isi dari folder berikut dilihatkan pada Gambar 4.29



Gambar 4.29 Isi Folder Dist Pada MedUp *Vue.js*

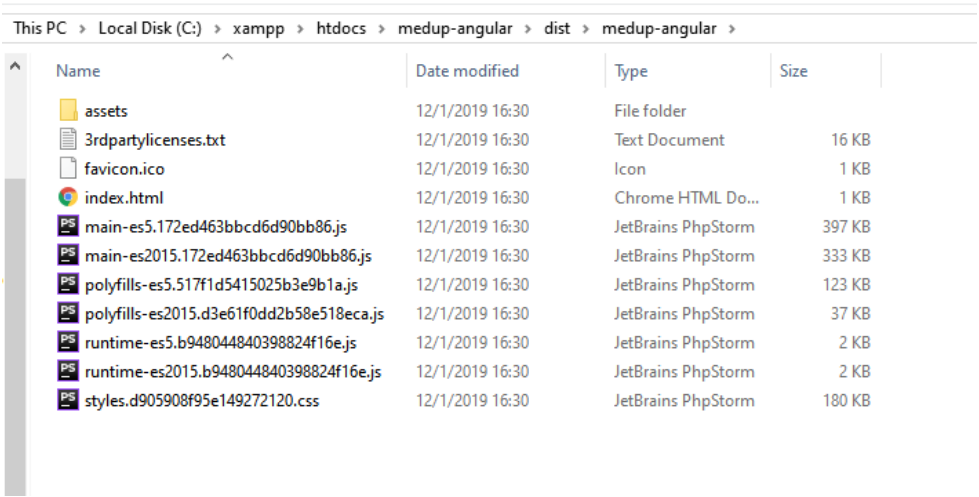
b. *Angular*8

Pada *Angular* untuk membuat folder *dist* yaitu dengan cara menuliskan perintah "*ng build --prod*". Tunggu sebentar sampai *build* selesai, lalu buka folder dan lihat berapa ukuran foldernya. Adapun gambar dari folder dist *Angular*.



Gambar 4.30 *Dist* Folder Pada MedUp *Angular*

Pada Gambar 4.30 telah diketahui bahwa *npm* telah membangun projek MedUp *Angular* dengan ukuran 3.36 MB. Dan untuk isi dari folder dilihatkan pada Gambar 4.31.



Gambar 4.31 Isi Folder *Dist* Pada MedUp *Angular*

Dengan demikian dapat disimpulkan, *Vue.js* lebih ringan dengan ukuran 2.28MB dibandingkan dengan *Angular*8 dengan ukuran 3.36MB dari segi *size metrics* nya.