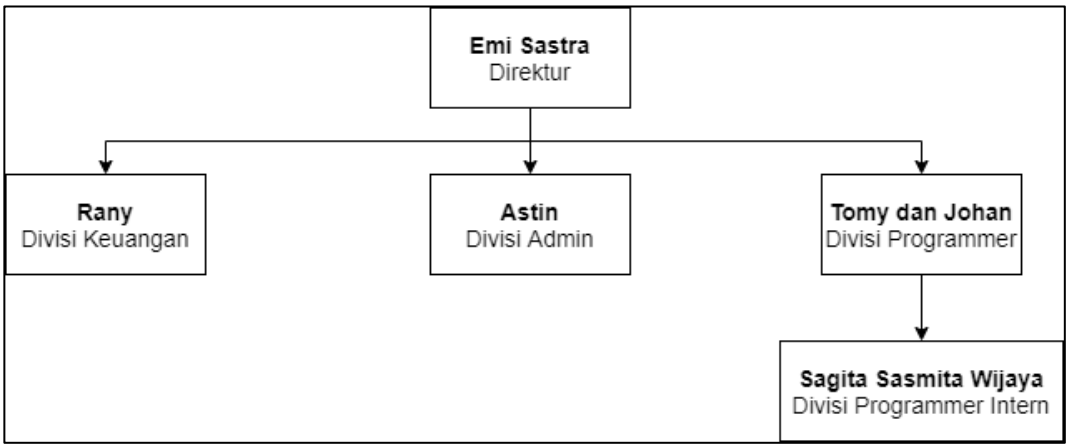
BAB 3

PELAKSANAAN KERJA MAGANG

3.1 Kedudukan dan Koordinasi

Dalam melaksanaan kerja magang di ERV Software, penulis memegang jabatan sebagai programmer intern di divisi programmer, dengan supervisor dari Bapak Emi Sastra selaku Direktur ERV Software. Bapak Emi berperan dalam memberi instruksi dan bimbingan dalam proses pembuatan CRUD data master mini market. Berikut ini merupakan struktur dari kedudukan penulis di ERV Software:



Gambar 3.1 Struktur Kedudukan Penulis di ERV Software

(Sumber: ERV Software, 2020)

3.2 Tugas yang Dilakukan

Tugas yang dilakukan selama praktek kerja magang yaitu mengembangkan aplikasi voucher mini market berbasis *website* menggunakan *framework* Laravel sebagai *backend* dan VueJs sebagai *frontend*. Pengembangan

aplikasi disesuaikan dengan rancangan serta kebutuhan klien, seperti dapat melihat data, menambah data, menghapus data dan memperbaharui data.

3.3 Uraian Pelaksanaan Kerja Magang

Empat minggu pertama dari praktek kerja magang digunakan untuk mempelajari dasar-dasar Laravel dalam penemapatan MVC (Model View Controller), Javascript, PHP, dan dasar-dasar dalam menggunakan Gitlab. Seluruh pembelajaran dilakukan dengan tatap muka secara langsung selama beberapa hari dimulai pada hari pertama magang, kemudian melakukan pembelajaran secara *online* atau *work from home* melewati aplikasi *Whatsapp*. Selanjutnya mulai dari minggu kelima sampai dengan minggu kesembilan dalam praktek kerja magang digunakan untuk mempelajari dasar-dasar VueJS baik *library* maupun cara mengimplementasikan dengan Laravel diikuti dengan beberapa latihan dalam menggunakan *library* VueJS. Pada minggu kesepuluh hingga minggu ketiga belas dalam praktek kerja magang digunakan untuk membuat *CRUD* untuk *table master* mini market dengan menggunakan Laravel sebagai *backend* untuk mengambil data dari *database* dan VueJS sebagai *front end* untuk menampilkan data. Seluruh tugas yang dikerjakan selama mengerjakan proyek data master mini market dijabarkan dalam Tabel 3.1 dan Tabel 3.2.

Tabel 3.1 Deskripsi Pekerjaan Pembuatan Data Master Mini Market

Minggu	Deskripsi Pekerjaan
1	Pengenalan struktrur *framework* Laravel dalam menggunakan Model, View dan Controller.
2	Mempelajari cara untuk seeding data kedalam database menggunakan migrations di Laravel.
3	Melakukan latihan dasar implementasi menggunakan Javascript dan PHP.
4	Melakukan latihan dasar implementasi menggunakan Javascript dan PHP. (lanjutan).
5	Pengenalan struktur framework VueJS.
6	Pengenalan struktur framework VueJS (lanjutan), mempelajari bagaimana menselaraskan VueJS dengan Laravel.
7	Melakukan instalasi yang dibutuhkan untuk membuat CRUD data mini market master.
8	Memperbaiki *bug* dan *error* yang tedapat pada proyek saat instalasi *library* berhasil dilakukan.
9	Memperbaiki *bug* dan *error* yang tedapat pada proyek saat instalasi *library* berhasil dilakukan.(lanjutan).
10	Membuat fungsi tambah, *edit*, dan *delete* di *Controller* Laravel dan mengatur *Route* untuk masing-masing fungsi tersebut.
11	Melakukan *debugging* untuk memeriksa apakah data dapat di *fetch* atau tidak.
12	Memperbaiki *error* pada saat menampilkan data menggunakan *Axios* didalam VueJS.
13	Membuat *modal* suntuk fungsi tambah data kemudian menampilkannya menggunakan *Axios*.

Tabel 3.2 Jadwal Kerja Magang

Nama Kegiatan	Minggu												
	1	2	3	4	5	6	7	8	9	10	11	12	13
Pengenalan *framework* Laravel menggunakan MVC													
Mempelajari cara untuk seeding menggunakan migrations													
Melakukan latihan dasar implementasi menggunakan Javascript													
Melakukan latihan dasar implementasi menggunakan Javascript dan PHP (continue)													
Pengenalan struktur *framework VueJS*													
Mempelajari untuk menselaraskan VueJS dengan Laravel													
Instalasi *library* VueJS													
Memperbaiki *bug* dan *error* setelah instalasi													
Memperbaiki *bug* dan *error* setelah instalasi													
Membuat fungsi dan *route* tambah, *edit,* dan *delete*													
Debugging untuk menampilkan data													
Memperbaiki error saat mengambil data dengan *axios* di VueJS													
Membuat *modal* untuk tambah dan menampilkannya dengan													

Berdasarkan pada Table 3.1 dan Tabel 3.2 yang telah dijabarkan, pada minggu pertama dan keempat penulis melakukan *planning* sekaligus melakukan analisis *requirement* dalam pembuatan aplikasi data master mini market dibagian *backend*. Kemudian pada minggu kelima sampai minggu ketuujuh, penulis melakukan tahap analisis *requirement* yang dibutuhkan dalam pembuatan pada bagian *frontend* aplikasi data master mini market. Selanjutnya pada minggu kedelapan dan kesembilan penulis melakukan *testing* untuk memeriksa apabila terdapat kerusakkan dalam *requirement* yang dibutuhkan dalam proses pembuatan aplikasi. Pada minggu kesepuluh penulis melakukan tahap *development* aplikasi data master mini market baik pada bagian *frontend* maupun

backend, dilanjutkan dengan melakukan *testing* aplikasi data master mini market pada minggu kesebelas dan kedua bales untuk memeriksa apabila terjadi error saat implementasi dan pada minggu ketiga belas penulis melakukan tahap *development* untuk beberapa fungsi yang belum terbuat pada tahap sebelumnya.

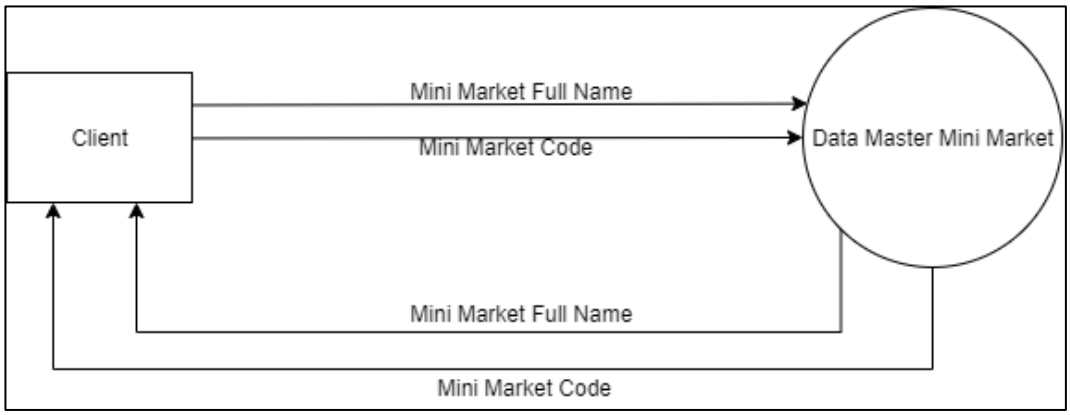
3.3.1. Analisa Kebutuhan

Pada rancang bangun *website* data master mini market, tedapat kebutuhan yang harus dipenuhi, yaitu:

1. Sistem dapat menambah data ke dalam database.
2. Sistem dapat menampilkan data.
3. Sistem dapat menghapus data .
4. Sistem dapat melakukan edit dan update data.

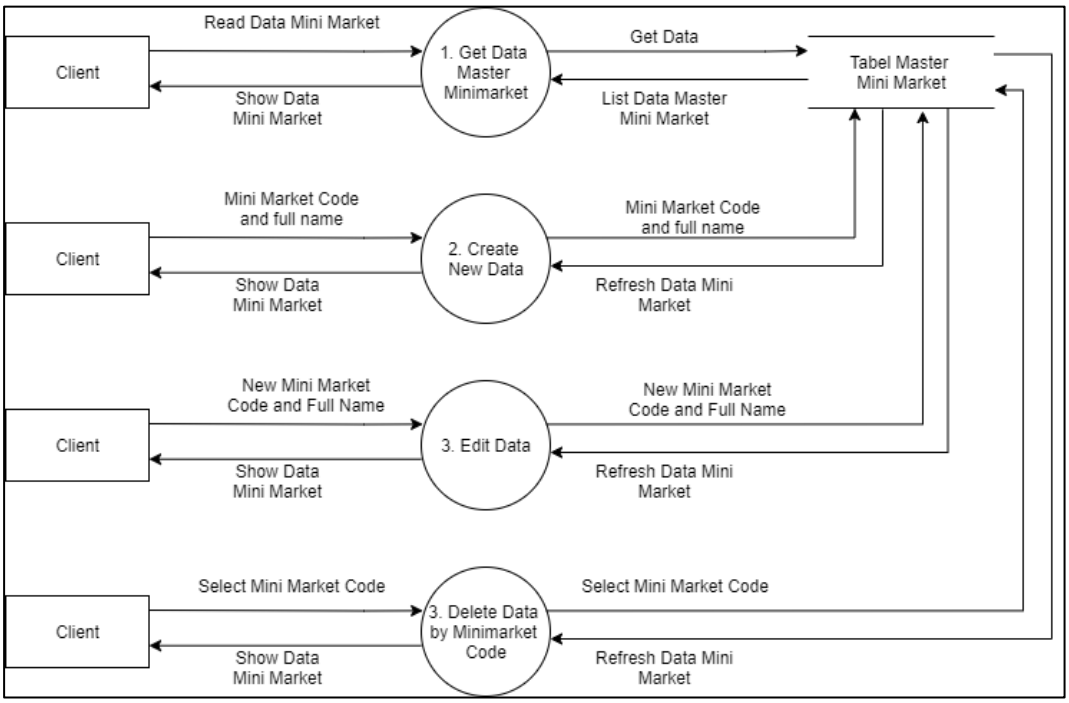
3.3.2. Perancangan Aplikasi

Berdasarkan hasil dari analisa kebutuhan yang telah dilakukan, dibuat perancangan aplikasi yang akan dibuat berdasarkan kebutuhan yang dijabarkan. Berikut ini adalah *Context Diagram* dan *Data Flow Diagram Level 1*. *Context Diagram* menjelaskan alur sistem dari data master mini market secara umum.

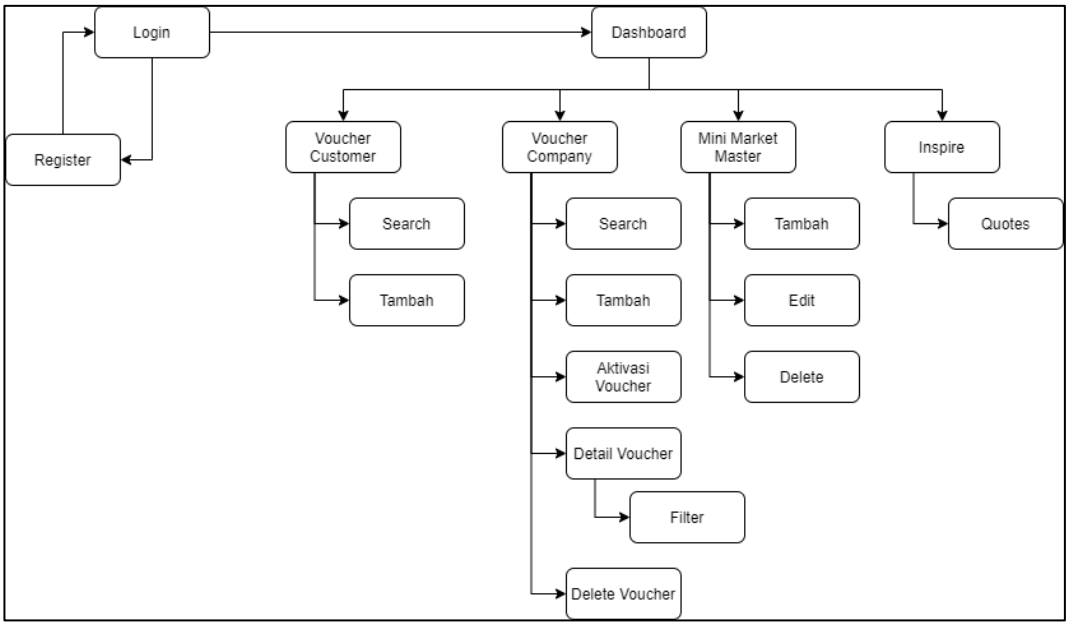


Gambar 3.2 Context Diagram Data Master Mini Market

Pada Gambar 3.2 menjelaskan bahwa *client* dapat memasukkan *Mini Market Code* dan *Mini Market Full Name* pada data master mini market. Sistem akan mengirim kembali kode dan *full name* ketika data yang diinput user telah masuk kedalam *database*. Kemudian pada Gambar 3.3 menjelaskan alur data master mini market diawali dengan klien sebagai *entity* yang dapat berinteraksi dengan 4 sistem baik dari sisi *read* data, tambah, *edit* maupun *delete*. Kemudian semua data yang telah dimasukkan akan disimpan kedalam *database*. *Site Map* dari aplikasi Voucher Mini Market dapat dilihat pada Gambar 3.4.



Gambar 3.3 Data Flow Diagram Level 1 Mini Market Master

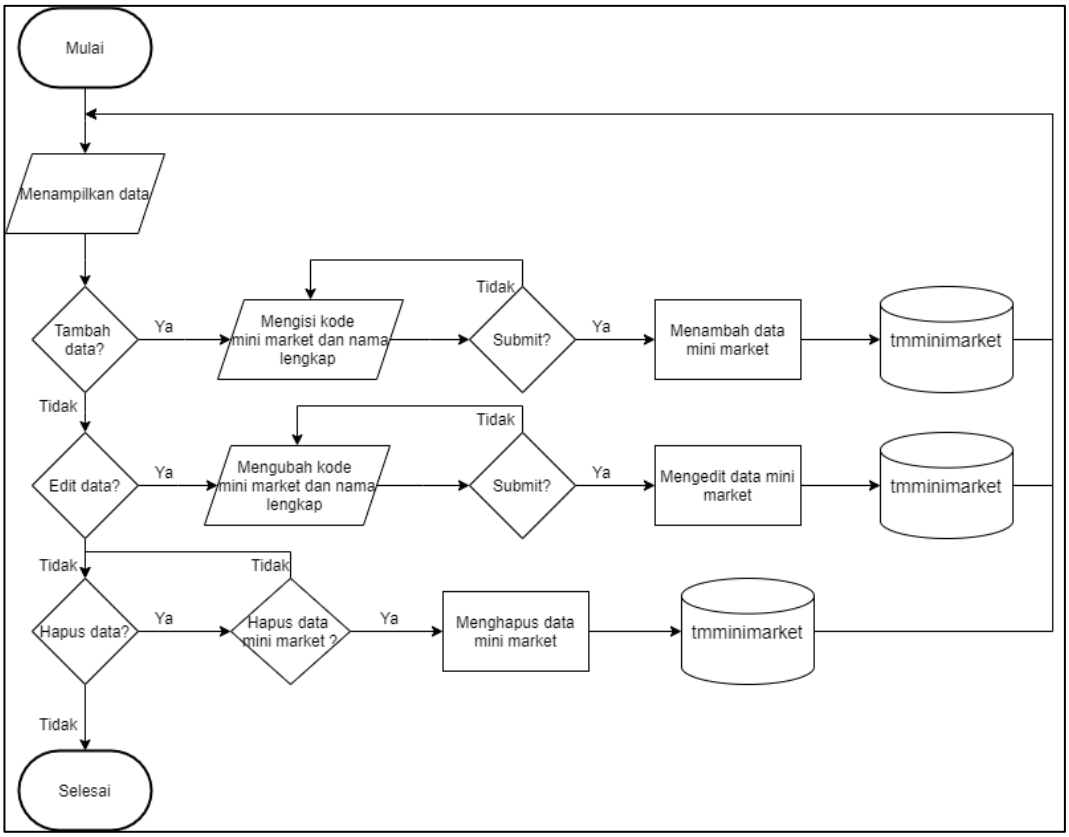


Gambar 3.4 Diagram Site Map Aplikasi Voucher Mini Market

Perancangan alur kerja aplikasi voucher mini market direpresentasikan melalui *flowchart*. Seluruh halaman memiliki *flowchart*-nya masing-masing, namun pada laporan kerja magang ini hanya disertakan 1 *flowchart* yang merepresentasikan 1 halaman dari aplikasi voucher mini market sesuai dengan jumlah halaman yang dikerjakan selama praktek kerja magang. Jumlah halaman yang dikerjakan terbatas karena proyek yang dikerjakan cukup rumit sehingga sering terdapat *error* dan *bug* yang menghambat proses pengerjaan proyek. 1 halaman yang dikerjakan dan akan dijabarkan pada bagian ini yaitu halaman mini market master.

A. Halaman Mini Market Master

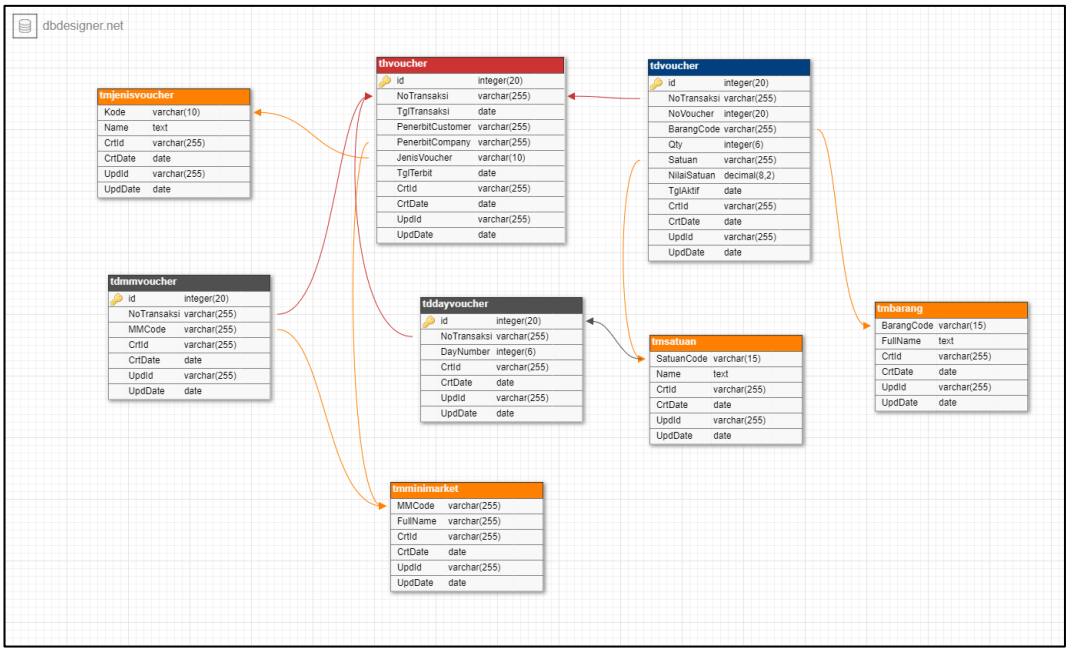
Halaman Mini Market Master terdiri atas komponen *Card* atau yang biasa digunakan untuk membuat *layout* menjadi *constraint*, tombol tambah pada bagian kanan atas untuk melakukan navigasi memunculkan *modal*, serta 2 *input fields* untuk memasukkan kode mini market dan nama mini market. Jika kode mini market serta nama mini market telah dimasukkan sesuai dengan validasi data yang ada pada *database* maka sistem akan melakukan pembuatan data baru yang telah dimasukkan. Jika data telah berhasil dibuat maka sistem melakukan navigasi kembali ke halaman Mini Market Master untuk melihat apakah data telah terbaharui. Kemudian tombol *edit* yang terdapat pada setiap data berfungsi untuk melakukan navigasi memunculkan *modal* dengan 2 *input fields* untuk memasukkan memasukkan kode mini market dan nama mini market, Jika kode mini market dan nama mini market telah dimasukkan, pengguna dapat menekan tombol "*Save*" untuk menyimpan data yang telah diubah dan tombol "*Cancel*", lalu sistem melakukan navigasi kembali ke halaman Mini Market Master untuk melihat apakah data yang diperbarui telah berubah atau tidak. Dan tombol terakhir adalah tombol *delete* yang akan menampilkan sebuah *alert* untuk meyakinkan pengguna untuk menghapus data atau tidak.



Gambar 3.5 Flowchart Tampilan Mini Market Master

3.3.3. Database Schema

Terdapat 8 *table* yang saling terhubung satu dengan lainnya termasuk *table mini market*. *Table* yang terhubung merupakan beberapa kumpulan macam *table* yang memiliki *primary key* dan *foreign key* seperti *table master, table detail*, dan *table header*. Berikut adalah *Database Schema* Mini market Master yang terhubung dengan *table* lainnya.



Gambar 3.6 Database Schema Mini Market Master

3.3.4. Perancangan Tampilan

Perancangan tampilan antarmuka dilakukan oleh penulis menggunakan *Adobe XD*. Tampilan perancangan antarmuka yang disertakan dalam laporan ini menyesuaikan dengan *flowchart* yang telah disertakan pada bagian perencanaan aplikasi, diawali dengan menmapilkan data kemudian disertai beberapa *modal* yang dijabarkan atau komponen dan fungsi yang dikerjakan, yaitu *modal* tambah, *modal edit* dan *alert delete*.

A. Halaman Mini market Master

Sesuai dengan perancangan aplikasi, tampilan halaman Mini Market Master memiliki 3 tombol untuk tambah, *edit*, dan *delete*. Rancangan tampilan antarmuka yang dibuat adalah berbasis website dan sudah *responsive*. Tampilan antarmuka dapat dilihat pada Gambar 3.7

Gambar 3.7 Tampilan Antarmuka Halaman Mini Market Master

B. Modal Tambah, Edit, dan Delete

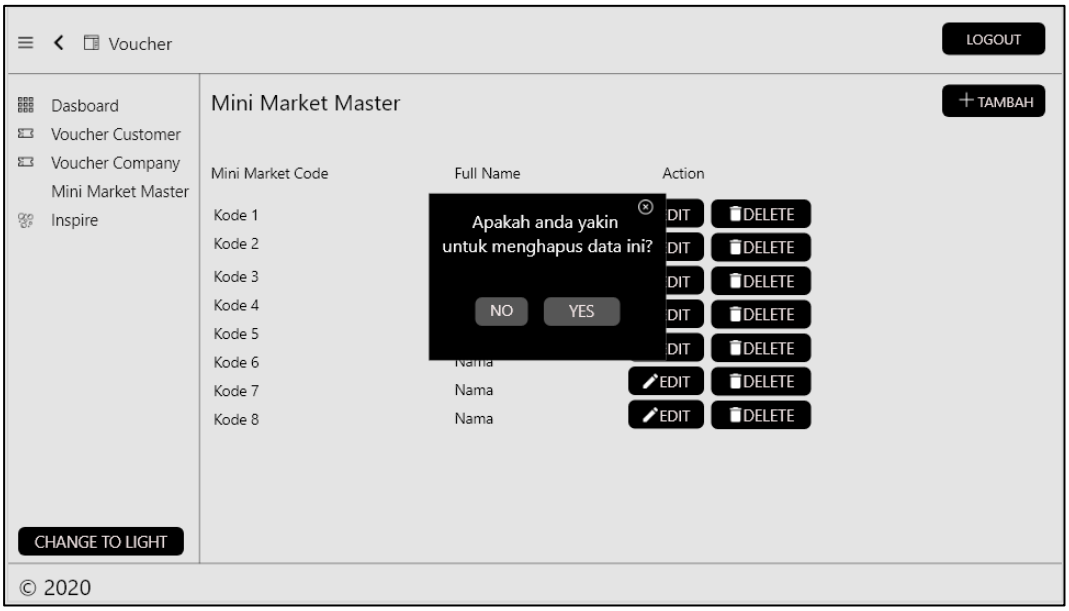
Sesuai dengan perancangan aplikasi, pada saat tombol tambah ditekan maka akan muncul *modal* yang terdapat 2 *input field* untuk memasukkan kode mini market dan nama mini market disertai tombol *create* dan *cancel*. Pada bagian *modal edit* memiliki 2 *input field* yang sama dengan tombol tambah hanya saja untuk tombol yang ada hanya *save* dan *cancel*. Sedangkan untuk *alert delete* terdapat pesan untuk menyatakan bahwa pengguna akan menghapus data tersebut disertai dengan tombol "*N*o" dan "*Yes*". Tampilan *modal* dapat dilihat pada Gambar 3.8, Gambar 3.9, dan Gambar 3.10

Gambar 3.8 Tampilan Modal Tambah Mini Market

Gambar 3.8 merupakan *mockup* tampilan pada modal Tambah yang mewajiban pengguna untuk mengisi 2 (dua) *input fields* yaitu "Masukkan Kode Mini Market" dan "Masukkan Nama Mini Market". *Input fields* pada "Masukkan Kode Mini Market" terbentuk dalam format *varchar* sehingga pengguna hanya dapat memasukkan alpfabet dan angka saja. Kemudian untuk "Masukkan Nama Mini Market" juga terbentuk dalam format *varchar* untuk memudahkan pengguna dalam memasukkan nama mini market.

Gambar 3.9 Tampilan Modal Edit Mini Market

Gambar 3.9 merupakan hasil *mockup* tampilan *modal* untuk menu *edit* data mini market, pada halaman ini pengguna dapat mengubah kode mini market serta nama mini market yang pernah ditambah sebelumnya. Didalam melakukan *edit* data mini market, pengguna hanya dapat memasukkan alfabet dan angka saja baik dalam memasukkan kode mini market maupun nama mini market. Jika pengguna ingin membatalkan perubahan pada data mini market maka dapat menekan tombol "Cancel", sedangkan tombol "Save" akan menyimpan data yang telah diubah oleh pengguna dan kemudian sistem akan memperbarui data yang telah di *edit*.



Gambar 3.10 Tampilan Alert Delete Mini market

Gambar 3.10 merupakan *alert* yang akan tampil ketika pengguna ingin menghapus data mini market. Pengguna dapat menghapus data mini market apabila data tersebut tidak signifikan atau sudah tidak *valid*, pada saat pengguna menekan tombol "Delete" pada data yang ingin di hapus akan muncul *alert* untuk memastikan bahwa pengguna ingin menghapus data tersebut. Jika pengguna tidak ingin menghapus data tersebut maka dapat menekan tombol "No" atau icon *close* pada alert, sebaliknya pengguna dapat menekan tombol "Yes" untuk menghapus data tersebut. Kemudian sistem akan melakukan *refresh* pada tampilan data untuk memastikan bahwa data tersebut sudah terhapus.

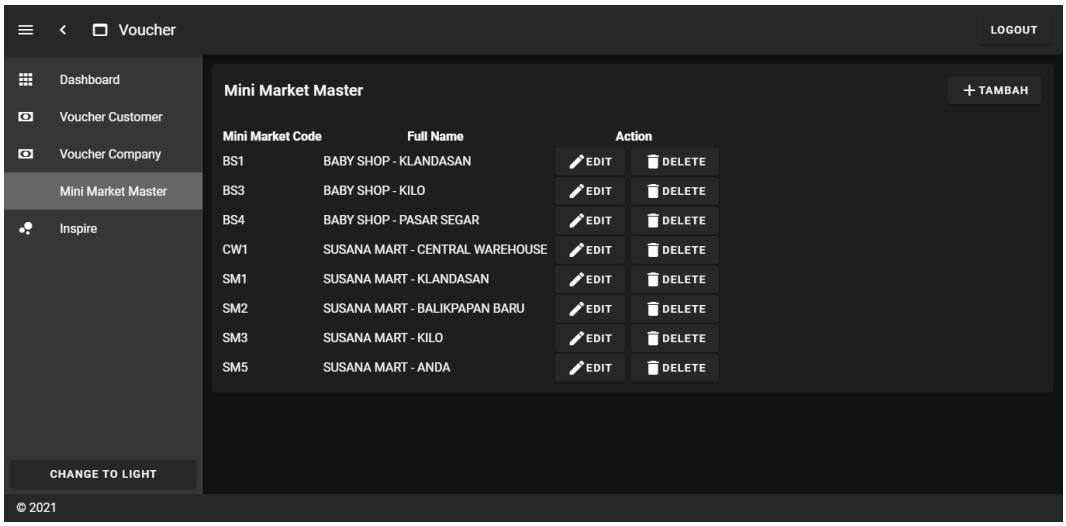
3.3.5. Hasil Implementasi Halaman

Berikut merupakan hasil implementasi dari aplikasi untuk 1 halaman yang telah disebutkan pada bagian perancangan aplikasi, yaitu Mini Market Master. Perlu diketahui data pada gambar yang dicantumkan pada laporan ini bersifat tetap karena sudah diselaraskan dengan data dari *database*, namun pada proses

tambah, *edit* dan *delete* belum dapat di implementasikan karena membutuhkan waktu yang cukup lama sehingga masih terdapat *error* dan *bug*. Seluruh gambar yang disertakan pada bagian ini merupakan hasil *screenshot* pada *browser Microsoft Edge*.

A. Mini Market Master

Halaman Mini Market diimplementasikan sesuai dengan rancangan tampilan antarmuka. Ketika pengguna telah melakukan *login* dan *registrasi* maka dapat melihat tampilan pada Gambar 3.11 yang berisikan *list* data yang sudah diselaraskan dengan *database*.



Gambar 3.11 Implementasi Halaman Mini Market Master

Implementasi halaman Mini Market Master mengalami perubahan beberapa kali terkait pesan *error* pada saat menampilkan data yang di ambil dari *database* karena membutuhkan instalasi *library* bernama *axios*. Fungsi *axios* ini digunakan untuk memanggil *route* di *file Controller* yang ada pada bagian *backend*. Berikut adalah *source code* dalam mengimplementasikan pengambilan data mini market master.

```php
class MinimarketController extends Controller
{

    public function ambildata(){
        //get
        $minimarketdata = DB::table('tmminimarket')->get();

        return response()->json($minimarketdata);
    }
```

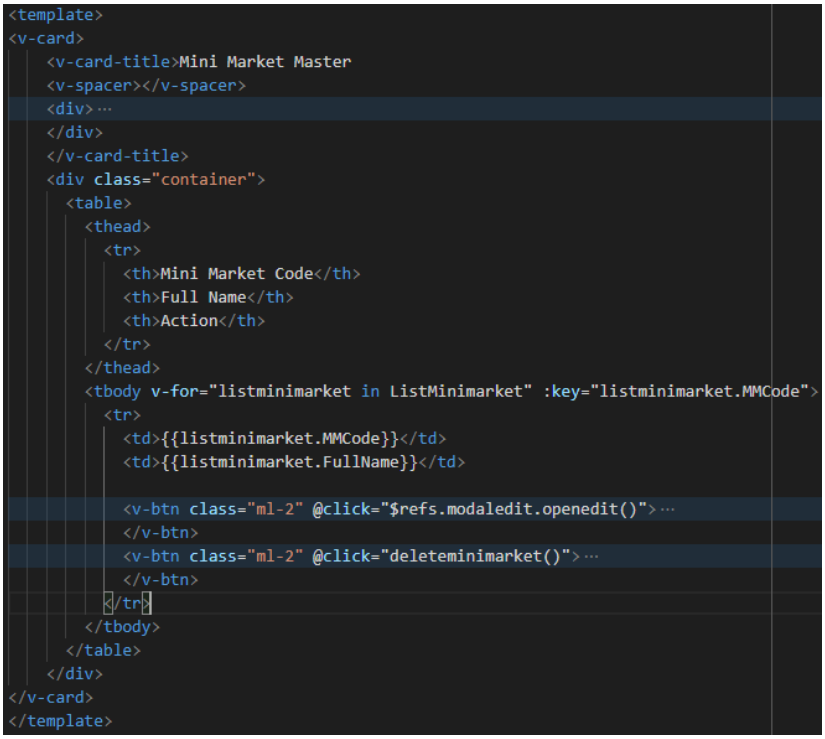
Gambar 3.12 Potongan Kode Memanggil Data Menggunakan Controller

Fungsi ambildata() akan memanggil data di *table tmminimarket* yang terdapat didalam *database* kemudian mengembalikan data tersebut kedalam *format* JSON atau biasa digunakan untuk mengubah data menjadi sebuah API. Fungsi yang terlampir pada Gambar 3.12 merupakan sebuah lokasi data API yang dapat diambil dan mengimplementasikannya pada bagian *frontend*.

```php
Route::group([], function(){
    Route::get('/showminimarket', 'MinimarketController@ambildata');
});
```

Gambar 3.13 Potongan Kode Route Memanggil Data

Fungsi Route::group akan menggabungkan *route* yang berkaitan dengan *MinimarketController* agar tidak tergabung atau tercampur dengan *route* lainnya yang dapat menyebabkan *error* pada sistem karena tidak dapat melacak *route* dengan benar. Kemudian Route::get akan mengambil nama *route* dari fungsi yang ada pada *MinimarketController* sehingga dapat diimplementasikan untuk bagian *frontend*.



Gambar 3.14 Potongan Kode Menampilkan Data Mini Market.

Gambar 3.14 merupakan potongan *source code* pada bagian *frontend* untuk menampilkan data. Dibutuhkan *looping* pada saat menampilkan data, maka dari itu pada bagian *table body (tbody)* diperlukannya fungsi *v-for* yang berfungsi untuk melakukan *looping* sebanyak data yang ada di *database*. Kemudian *:key* memilki fungsi untuk mengambil *primary key* dari table *tmminimarket*, untuk menampilkan data maka diawali dengan kurung kurawal ({}) disetiap nama *variable*.

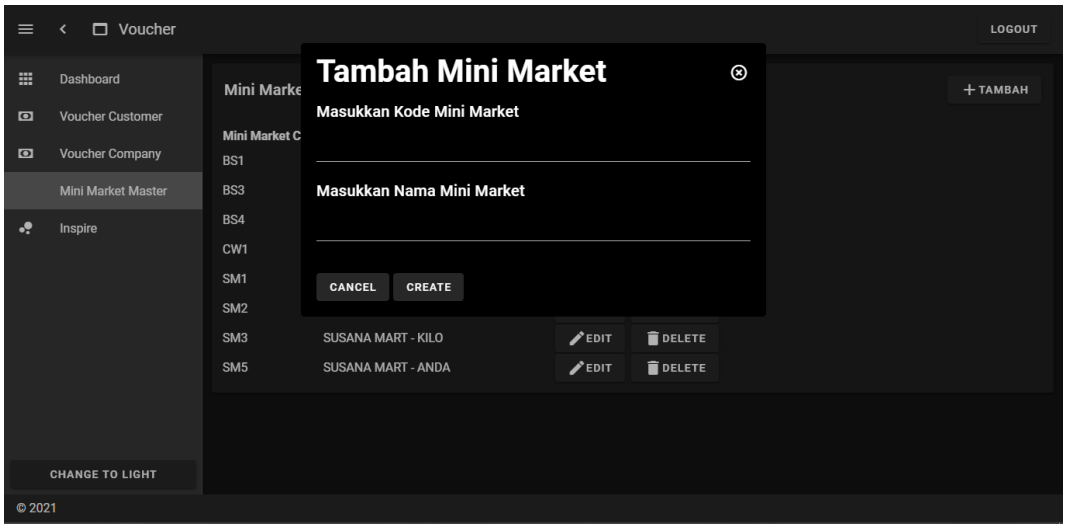
```
<script>
import axios from 'axios';
import Modal from '~/components/Modal';
import EditMinimarket from '~/components/EditMinimarket';
export default {
    components: {
        Modal,
    },
    data(){
        return{
        ListMinimarket: [],
        MMCode: '',
        FullName: '',
        }
    },
    methods:{
        getMinimarket(){
            axios.get('api/showminimarket').then(({data}) => {
                this.ListMinimarket = data
            })
            .catch((error) => console.error(error));
        },
    },
    mounted(){
        this.getMinimarket();
       // this.tampilkandata();
    },
    icons: {
        iconfont: 'mdi'
    }
}
</script>
```

Gambar 3.15 Potongan Kode Menampilkan Data Mini Market (lanjutan).

Gambar 3.15 merupakan lanjutan potongan kode dalam menampilkan data mini market dimana VueJS menggunakan sebuah *library* bernama *axios.get* untuk memanggil *route* yang berkaitan dengan fungsi menampilkan data di Laravel. Karena data mini market terbentuk dalam format *array* maka perlu ditambahkan kurung siku pada bagian fungsi data(), setelah membuat fungsi untu memanggil *route* di Laravel, maka fungsi tersebut harus di *mounted* agar dapat berjalan dengan baik.

B. Modal Tambah, Edit, dan Delete

Dari ketiga modal yang disebutkan hanya modal tambah yang dapat diimplementasikan sesuai dengan perancangan aplikasi dan perancangan tampilan antarmuka yang telah dijabarkan sebelumnya namun proses yang telah dilakukan tidak mengalami keberhasilan sehingga data yang di *input* tidak bertambah pada database. Proses pada bagian *backend* untuk ketiga *modal* telah di buat secara teratur berserta *routing* dari masing-masing fungsi tersebut. Berikut adalah *source code* yang telah diimplementasikan.



Gambar 3.15 Tampilan untuk Modal Tambah Mini Market

```php
public function TambahMinimarket(){
    $tambahminimarket = Minimarket::all();

    return response()->json('Added Data');
}

public function TambahMinimarketPost(Request $request){
    error_log('Some message here.');
    var_dump($request->all());
    $tambahdata = $request->validate([
        'MMCode' => 'required',
        'FullName' => 'required'
    ]);

    $cekdata = Minimarket::create($tambahdata);
    return response()->json('Data successfully added!');
}
```

Gambar 3.16 Potongan Kode Untuk Menambah Data

```php
Route::get('/tambahminimarket', 'MinimarketController@TambahMinimarket');
Route::post('/posttambah', 'MinimarketController@TambahMinimarketPost');
```

Gambar 3.17 Potongan Kode Untuk Routing Menambah Data

```vue
<div>
  <v-btn class="ml-2" @click="$refs.modaltambah.tampilkanmodal()">
    <v-icon>mdi-plus</v-icon> Tambah
  </v-btn>

  <modal ref="modaltambah">
    <template v-slot:header>
      <h1>Tambah Minimarket</h1>
    </template>

    <template v-slot:body>
      <div id="tambahminimarket">
        <label>Masukkan Kode Minimarket</label><br>
        <v-text-field type="text" v-model="MMCode"/>


        <label>Masukkan Nama Minimarket</label><br>
        <v-text-field type="text" v-model="FullName"/>
      </div>
    </template>

    <template v-slot:footer>
      <div>
        <v-btn @click="$refs.modaltambah.tutupmodal()">Cancel</v-btn>
        <v-btn @click="tampilkandata()">Create</v-btn>
      </div>
    </template>
  </modal>
</div>
```

Gambar 3.18 Potongan Kode untuk Form Tambah Data

```
tampilkandata(){
    let data = new FormData();
    data.append('MMCode', this.MMCode);
    console.log(this.MMCode);
    data.append('FullName', this.FullName);
    console.log(this.FullName);
    axios.post('api/posttambah', {
      data: data
    }).then((response) => {
      console.log(response)
    });

},
```

Gambar 3.19 Potongan Kode untuk Memanggil Route di Laravel

```
public function EditMinimarket($minimarket){

    $getminimarketdata = Minimarket::where('MMCode', $MMCode)->first();

    return response()->json($getminimarketdata);
}


public function EditMinimarketPost(){
    $MMCode = request()->input('MMCode');

    $editminimarket = [
        'MMCode' => request()->input('MMCode'),
        'FullName' => request()->input('FullName'),
    ];

    $saveedit = Minimarket::where('MMCode', $MMCode)->update($editminimarket);

    return response()->json('Data successfully edited!');
}
```

Gambar 3.20 Potongan Kode untuk Edit Data

Potongan kode diatas fungsi EditMarket($minimarket) merupakan fungsi untuk mengambil data yang akan di *edit* berdasarkan ‘MMCode’, kemudian dilanjutkan dengan fungsi kedua yaitu EditMini marketPost() yang digunakan

untuk melakukan validasi data baru yang telah dimasukkan kedalam *input fields* dan menyimpan data baru tersebut kedalam *database.*

```
Route::get('/editminimarket', 'MinimarketController@EditMinimarket');
Route::post('/saveedit', 'MinimarketController@EditMinimarketPost');
```

Gambar 3.21 Potongan Kode dari Route untuk Fungsi Edit

```
public function DeleteMinimarket($MMCode){
    $deleteminimarket = Minimarket::find($MMCode);
    $deleteminimarket->delete();
    return response()->json('Minimarket deleted successfully');
}
```

Gambar 3.22 Potongan Kode untuk Menghapus Data

Fungsi DeleteMinimarket($MMCode) akan menghapus data berdasarkan 'MMCode' yang terdapat didalam *database*, jika fungsi berhasil di jalankan maka sistem akan mengembalikan pesan JSON dengan pesan menyatakan bahwa data telah berhasil dihapus.

3.3.6. Kendala yang Ditemukan

Dalam pembuatan aplikasi data master mini market di perusahaan ERV Software terdapat beberapa kendala yang dialami, yaitu sebagai berikut:

1. Kurangnya pengalaman dalam menggunakan *framework* terkait sehingga memerlukan waktu yang lebih lama untuk menyelesaikan pekerjaan, teruatama ketika adanya penerapan pada kedua platform.
2. Adanya beberapa halaman atau fitur yang belum selesai dikembangkan, namun belum dapat dilanjutkan lagi sehingga menghambat progress lain.
3. Adanya kesulitan dalam melakukan penerapan *framework* terkait sehingga sering terjadi *error* dan *bug* yang menghambat progress.

3.3.7. Solusi Atas Kendala yang Ditemukan

Solusi yang dilakukan untuk mengatasi kendala dalam pembuatan aplikasi data master mini market adalah:

1. Melakukan pembelajaran mandiri melalui dokumentasi *framework*, menonton video, dan melakukan pencarian pada *website* Stack Overflow, W3School, dan Medium, terkait masalah yang ditemui. Selain itu, dapat bertanya dan berdiskusi dengan supervisor.
2. Menggunakan referensi proyek yang telah ada untuk menyelesaikan suatu permasalahan. Jika solusi yang ada masih belum pasti, dapat melakukan pencarian atau membahas masalah yang ada dengan supervisor.