BAB III
PELAKSANAAN KERJA PROFESI

3.1 Bidang Kerja

Pada pelaksanaan Kerja Profesi ini praktikan bertugas sebagai IT atau di tempatkan pada divisi IT yang melaksanakan tugas sebagai Backend Developer dalam membangun aplikasi website company profile. Dalam proses kegiatan Kerja Profesi praktikan tidak mempunyai pembimbing kerja yang membuat praktikan dan rekan rekan yang lain langsung berkoordinasi denga IT manager dan direktur oprasional. Kegiatan praktikan dalam membangun website company profile di bagian backend, seperti membangun *database* yang akan digunakan oleh website company profile seperti menganalisa *database* apa yang cocok digunakan lalu strukturnya seperti apa dan mengimplementasi *database* yang telah di Analisa dan dirancang strukturnya.Kemudian praktikan juga membantu frontend dalam menjelaskan koneksi antara *database* dan halaman user yang memerlukan koneksi ke *database*. Dan praktikan juga membuat CRUD property untuk di tampilkan ke halaman user yang akan digunakan admin untuk menginput data property. Terakhir praktikan melakukan test untuk aplikasi website yang sudah berjalan, apakah berjan baik atau mempunyai bug jika ada bug segera diperbaiki.

3.2 Pelaksanaan Kerja

Pada tanggal 6 Juni 2022 sampai 6 September 2022, para praktisi melakukan pekerjaan profesi, mengikuti jadwal yang telah ditentukan. Praktikan bekerja sebagai backend developer di divisi IT di PT Bumi Rejeki Agung, di mana praktikan mengimplementasikan pembuatan situs web profil perusahaan. Pekerjaan dimulai dengan berdiskusi dengan tim frontend developer, system analyst, dan API developer beserta dengan tim mobile developer untuk menentukan jalan dan arah proyek ini di bangun.

Berikut adalah dokemntasi dari diskusi pertama praktikan seperti di gambar 3.1

Gambar 3. 1 Diskusi Pertama dan Pengenalan

Sumber : dokumentasi perusahaan

Disana praktikan beserta rekan-rekan melakukan diskusi pertama dengan membahas kebutuhan *user* apa yang user butuhkan dari aplikasi yang akan praktikan dengan rekan bangun serta menentukan *scope* dari aplikasi dan juga tahapan apa saja yang akan praktikan dan rekan rekan semua terapkan kedepan. Setelah itu pembagian tugas yang dipecah menjadi dua tim yaitu *mobile developer* yang berfokus membangun aplikasi *mobile* dan *web developer* yang bertugas membuat aplikasi *website.* Kemudian masing masing tim membagi pekerjaan yang akan dilakukan setiap orang. Selanjutnya setiap orang bertugas sesuai dengan pembagian tugasnya masing masing dan praktikan sendiri bertugas dan melakukan pekerjaan di bagian *Backend* untuk membuat *database*, halaman admin, dan juga koneksi untuk halaman *user* dan *database* yang ada.

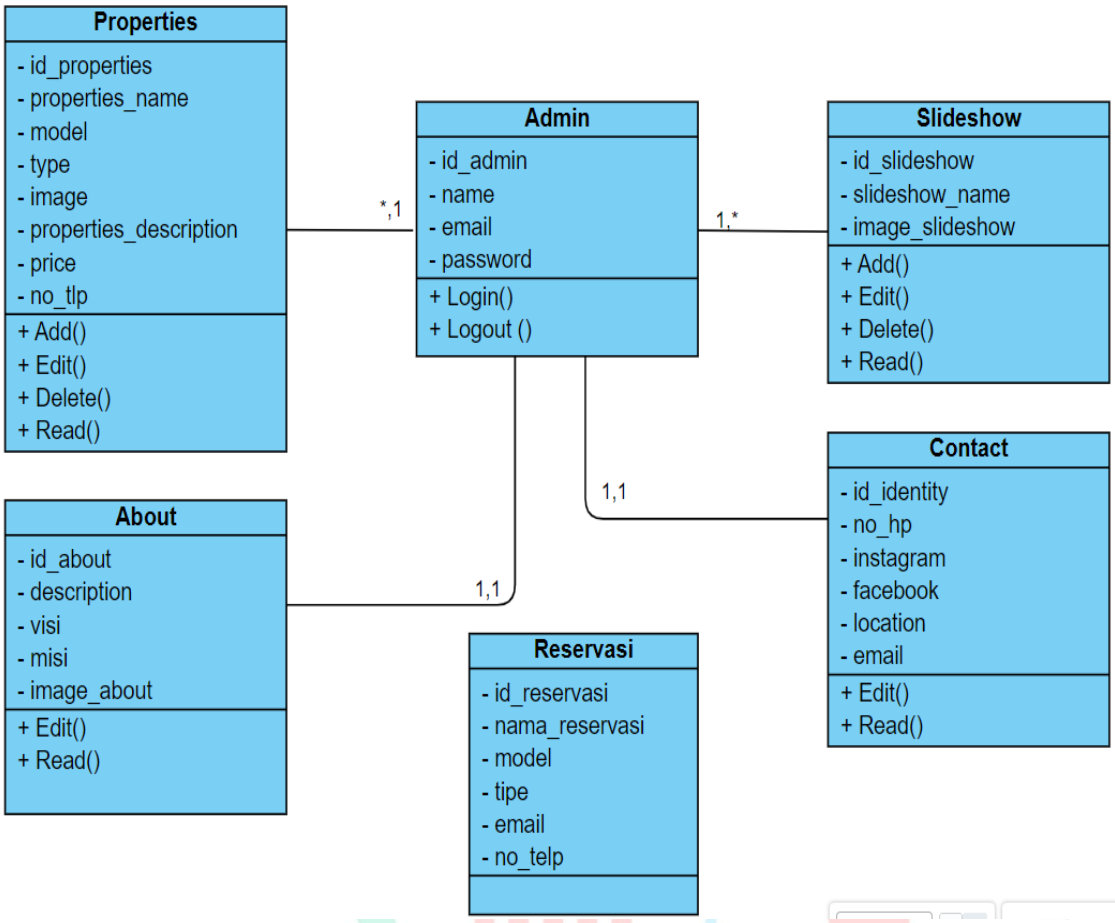
3.2.1 Merancang *database* dan menganalis

Pada tahap pertama melakukan pekerjaan Langkah pertama yang praktisi ambil Bersama rekan adalah membuat perancangan apa saja yang dirancang di tahap pertama diantaranya:

1. Merancang jalannya aplikasi seperti apa UI/UX akan membuat desain dan analyst membuat alur aplikasi yang akan dibuat berjalan seperti yang diinginkan oleh user.
2. Analyst akan melakukan desain sesuai alur aplikasi menggunakan beberapa metode seperti use case, activity dan lain sebagainya yang akan menjadi acuan untuk praktisi
3. Merancang *database*nya terlebih dahulu yang mana *database* sudah dibuat strukturnya oleh analyst menggunakan *class diagram* jadi praktikan membuat *database* sesuai dengan struktur yang telah dirancang dengan *class diagram* yang sudah dibuat oleh analyst.

Setelah itu praktisi memulai pekerjaan untuk membangun backend pada aplikasi company profile ini dengan tahapan sebagai berikut:

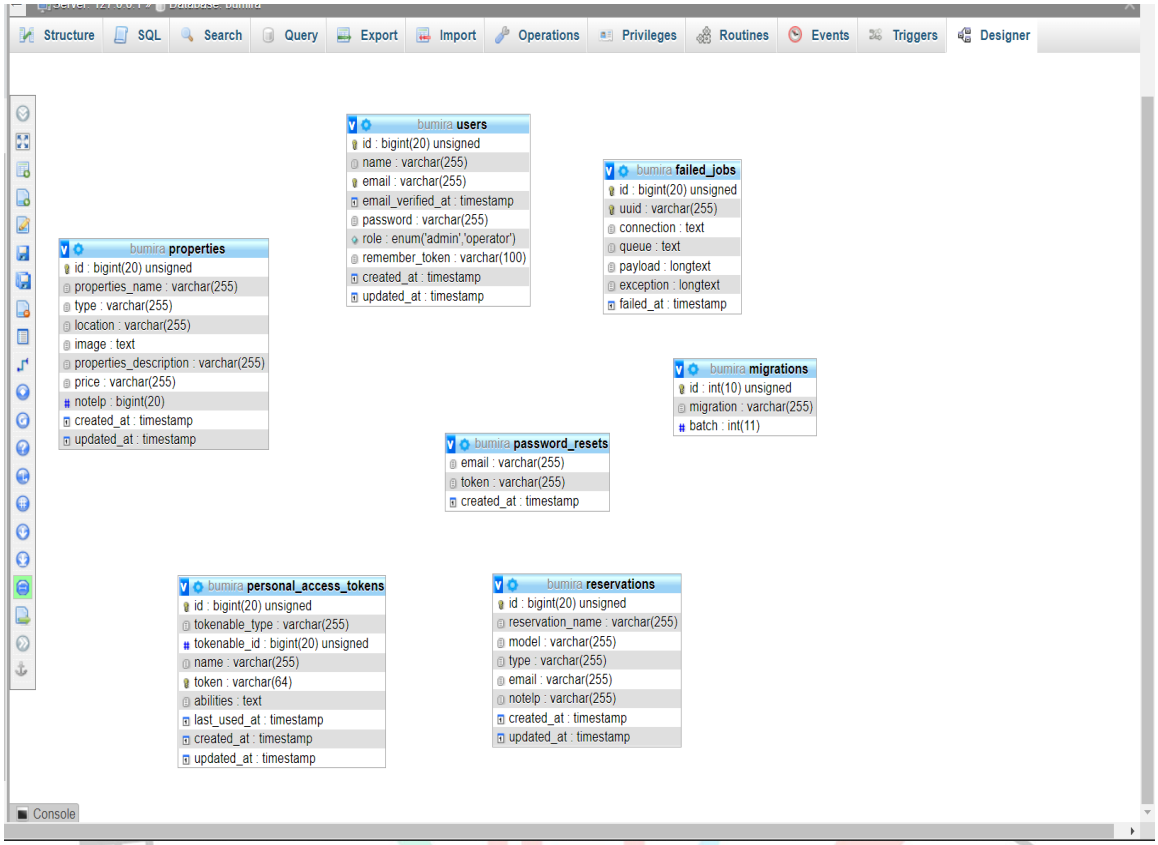
1. Membuat *database* yang sesuai dari *class diagram* yang analyst buat.
2. Membuat Login admin untuk memastikan admin yang dapat masuk adalah admin yang telah mempunyai username dan password di dalam *database*
3. Merancang CRUD untuk properties yang bisa di input oleh admin dan akan muncul di halaman user
4. Mengkoneksikan properties halaman user dan admin dan beserta *database*nya
5. Mengintegrasikan opsi reservasi untuk costumer yang ingin merservasi di halaman utama agar bisa masuk ke dalam *database* dan terdata di *database* admin.



Gambar 3. 2 Class Diagram

Sumber : hasil kerja analis

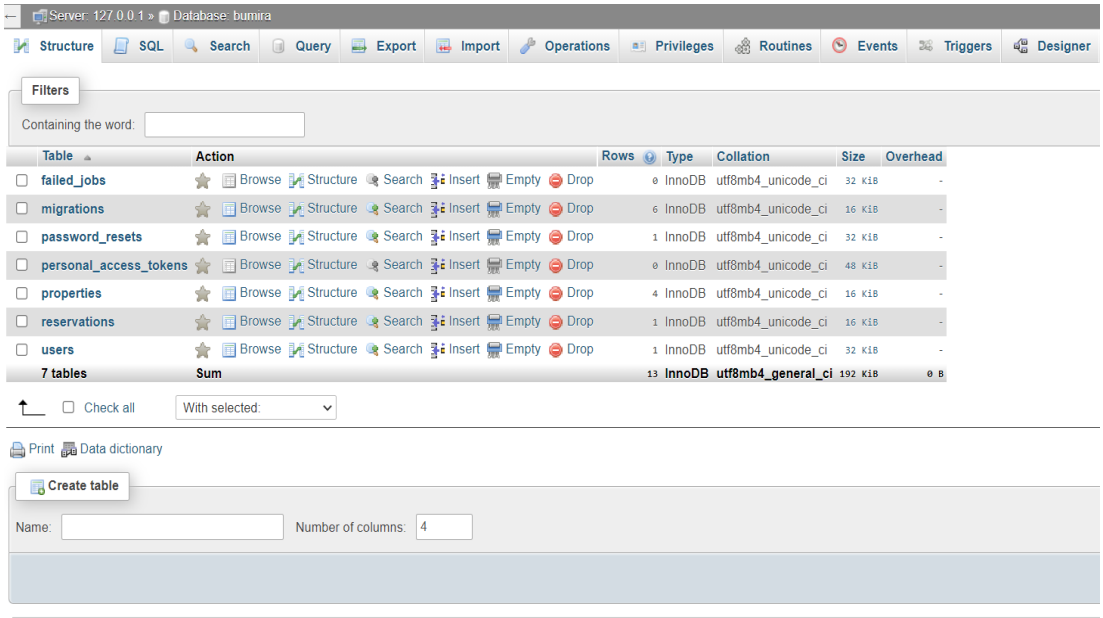
Seperti yang ada di gambar 3.2 yang ada diatas ini merupakan *class diagram* yang dibuat oleh analis yang kemudian akan praktikan implementasikan *class diagram* ini ke kedalam bentuk *database* menggunkan *database* dengan struktur data mysql dikarenakan *database* mysql merupakan struktur data yang cocok dan juga familiar untuk praktikan implementasikan kedalam aplikasi website ini. Berikut ini merupakan contoh struktur designer mysql yang praktikan perlihatkan di gambar 3.3 sebagai berikut.



Gambar 3. 3 Designer Mysql

Sumber : hasil kerja praktikan

Dapat dilihat dari gambar diatas yang merupakan desain dari database yang digunakan dari aplikasi *company profile* yang dibuat berdasarkan dari desain yang telah dirancang oleh analis kemudian praktikan implementasikan kedalam bentuk database di *mysql* dan menghubungkan database ke halaman admin dan juga halaman *user.* Data tersebut dapat dirubah oleh admin untuk mengubah isi dari *properties* sehingga dapat ditampilkan di halaman utama sehingga user dapat melihat apa saja isi dari *properties* yang telah admin buat. *User* dapat melakukan reservasi di halaman utama dan admin juga dapat melihat hasil reservasi yang telah dibuat oleh user.



Gambar 3. 4 Tabel Database

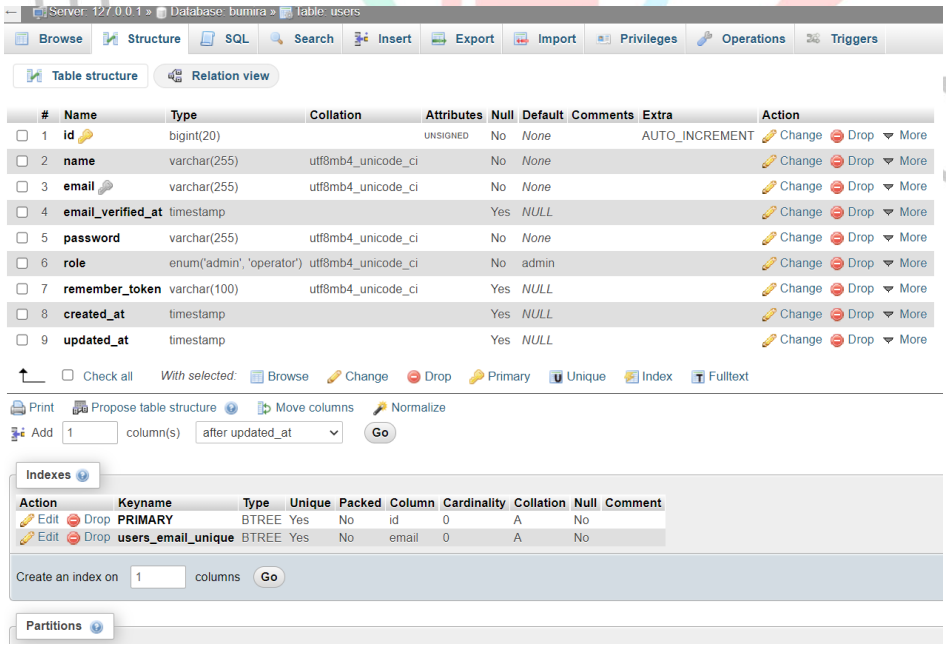
Sumber : hasil kerja praktikan

Dari gambar diatas dapat dilihat tabel dari database untuk aplikasi *website company profile* yang dibuat. Data-data yang diperoleh dalam tabel ini merupakan data untuk ditampilkan di halaman utama dan juga untuk menyimpan informasi yang dibuat di halaman admin. Admin bertugas untuk membuat isi konten yang akan ditampilkan di halaman utama, disini admin dikatagorikan sebagai *user.* Dan didalam tabel user terdapat 2 katagori yang mendeskripsikan perannya masing-masing, ada katagori *user admin* dan katagori *user operator* yang memiliki perbedaan akses dan dapat di ubah oleh admin. Dan operator sendiri bertugas untuk mengatur konten apa saja yang dapat ada didalam halaman utama. Untuk isi dari tabel admin dapat dilihat pada gambar dibawah ini.

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('users', function (Blueprint $table) {
            $table->id();
            $table->string('name');
            $table->string('email')->unique();
            $table->timestamp('email_verified_at')->nullable();
            $table->string('password');
            $table->enum('role', ['admin', 'operator'])->default('admin');
            $table->rememberToken();
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('users');
    }
```

Gambar 3. 5 Tabel User

Sumber : hasil kerja praktikan



Gambar 3. 6 Database Tabel User

Sumber : hasil kerja praktikan

Hasil dari pekerjaan praktikan terhadap tabel pengguna untuk aplikasi online profil perusahaan dapat dilihat pada dua gambar di atas. Selain itu, Gambar 3.7 di bawah ini menampilkan data konten pada layar beranda.

```php
<?php

use Illuminate\Database\Migrations\Migration;
use Illuminate\Database\Schema\Blueprint;
use Illuminate\Support\Facades\Schema;

return new class extends Migration
{
    /**
     * Run the migrations.
     *
     * @return void
     */
    public function up()
    {
        Schema::create('properties', function (Blueprint $table) {
            $table->id();
            $table->string('properties_name');
            $table->string('type');
            $table->string('location');
            $table->text('image')->nullable();
            $table->string('properties_description');
            $table->string('price');
            $table->string('notelp');
            $table->timestamps();
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */
    public function down()
    {
        Schema::dropIfExists('properties');
```

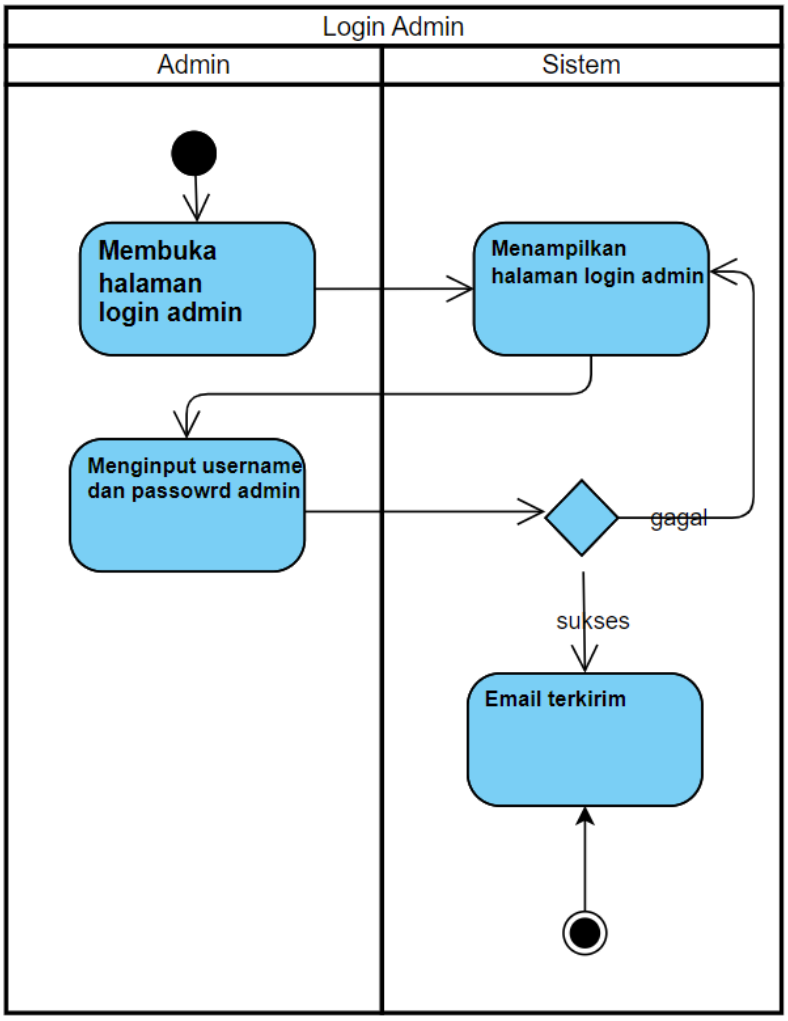
Gambar 3. 7 Tabel Properties

Sumber : hasil kerja praktikan

Berdasarkan gambar dan penjelasan diatas merupakan contoh *database* yang dibuat oleh praktikan. Yang menggunakan jenis *database* sql dengan mysql praktikan memlilih *database* jenis ini dikarenakan tipe *database* ini yang familiar untuk praktikan gunakan. Sehingga praktikan dapat mempercepat proses implementasi database untuk segera digunakan.

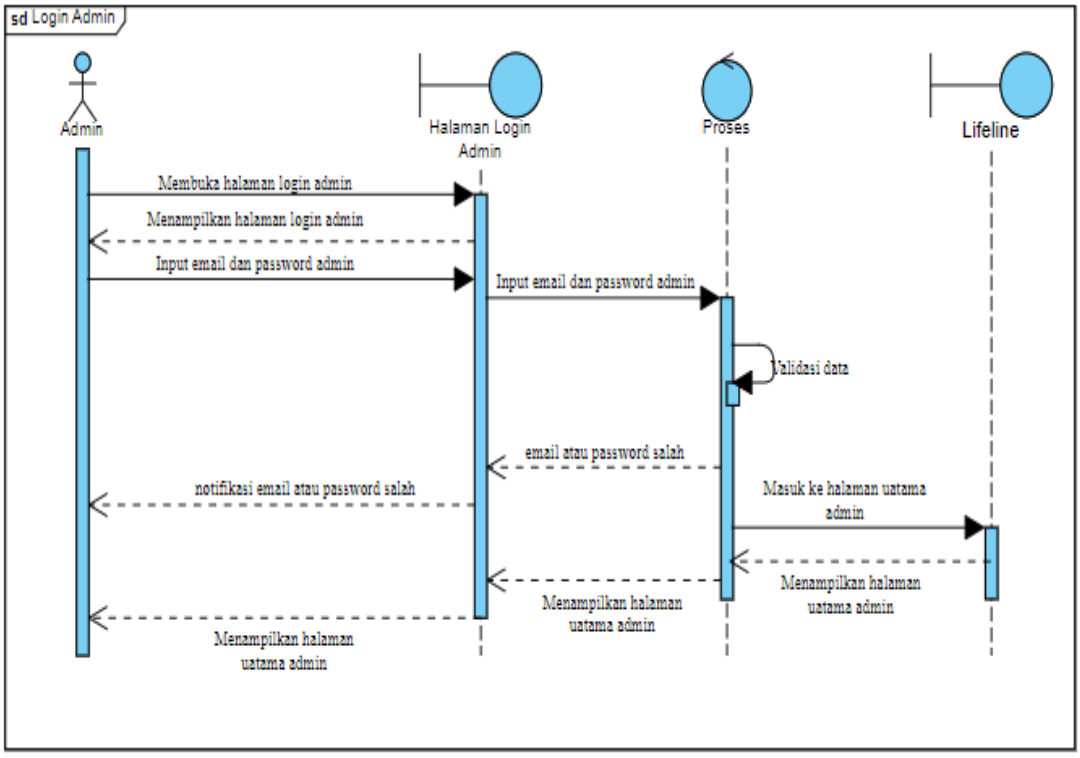
3.2.2 Implemetasi CRUD

Setelah membuat *database* praktikan membuat CRUD properties untuk admin karena ini dibutukan jika admin ingin mengubah isi konten di website tidak perlu mengubah langsung dari source code yang akan membuat binggung untuk admin yang tidak mempunyai basic sistem informasi jadi praktikan beserta rekan rekan sepakat untuk membuat CRUD untuk diakses oleh admin. Hal pertama yang praktikan buat untuk fungsi admin adalah login terlebih dahulu yang telah di desain oleh analis dan dibuat activity diagram dan juga *sequence diagram* seperti berikut.



Gambar 3. 8 Activity Diagram Login

Sumber : hasil kerja analis



Gambar 3. 9 Sequence Diagram Login

Sumber : hasil kerja analis

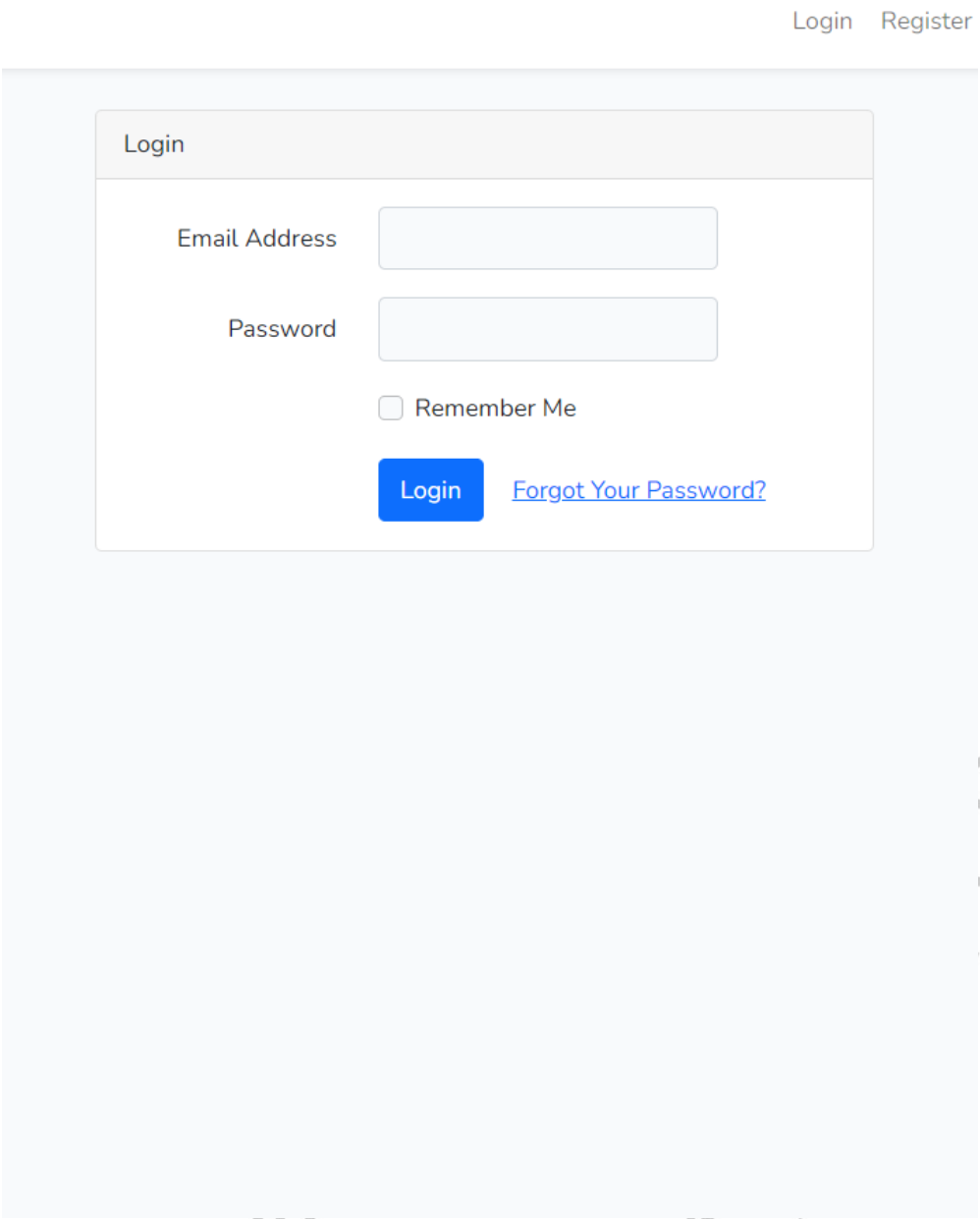
Dari *activity diagram* dan juga *sequence diagram* yang telah dirancang oleh sistem analais praktikan menganalsa bagaimana untuk mengimplementasikan kedalam bentuk sistem yang akan digunakan di aplikasinya dan bersama rekan analis praktikan mulai merancang untuk tampilan login beserta database login itu sendiri mulai dari membuat *source code* untuk login lalu membuat database login itu sendiri dan dari *source code* yang praktikan buat dapat dilihat contohnya pada gambar dibawah ini.

```
@extends('layouts.app')

@section('content')
<div class="container">
    <div class="row justify-content-center">
        <div class="col-md-8">
            <div class="card">
                <div class="card-header">{{ __('Login') }}</div>

                <div class="card-body">
                    <form method="POST" action="{{ route('login') }}">
                        @csrf

                        <div class="row mb-3">
                            <label for="email" class="col-md-4 col-form-label text-md-end">{{ __('Email Address') }}</label>

                            <div class="col-md-6">
                                <input id="email" type="email" class="form-control @error('email') is-invalid @enderror" name="email"

                                @error('email')
                                    <span class="invalid-feedback" role="alert">
                                        <strong>{{ $message }}</strong>
                                    </span>
                                @enderror
                            </div>
                        </div>

                        <div class="row mb-3">
                            <label for="password" class="col-md-4 col-form-label text-md-end">{{ __('Password') }}</label>

                            <div class="col-md-6">
                                <input id="password" type="password" class="form-control @error('password') is-invalid @enderror" name

                                @error('password')
                                    <span class="invalid-feedback" role="alert">
                                        <strong>{{ $message }}</strong>
                                    </span>
                                @enderror
                            </div>
                        </div>

                        <div class="row mb-3">
                            <div class="col-md-6 offset-md-4">
                                <div class="form-check">
                                    <input class="form-check-input" type="checkbox" name="remember" id="remember" {{ old('remember') ?

                                    <label class="form-check-label" for="remember">
                                        {{ __('Remember Me') }}
                                    </label>
                                </div>
                            </div>
                        </div>

                        <div class="row mb-0">
                            <div class="col-md-8 offset-md-4">
                                <button type="submit" class="btn btn-primary">
                                    {{ __('Login') }}
                                </button>

                                @if (Route::has('password.request'))
                                    <a class="btn btn-link" href="{{ route('password.request') }}">
                                        {{ __('Forgot Your Password?') }}
                                    </a>
                                @endif
                            </div>
                        </div>
                    </form>
```

Gambar 3.10 Source Code Login

Sumber : hasil kerja praktikan

Source code untuk login admin dapat ditemukan pada gambar 3.10 di atas. Dengan menggunakan kode ini akan menghasilkan halaman login yang ditampilkan di bawah ini.



Kode sumber untuk login admin dapat ditemukan pada gambar 3.10 di atas. Dengan menggunakan kode ini akan menghasilkan halaman login yang ditampilkan di bawah ini.

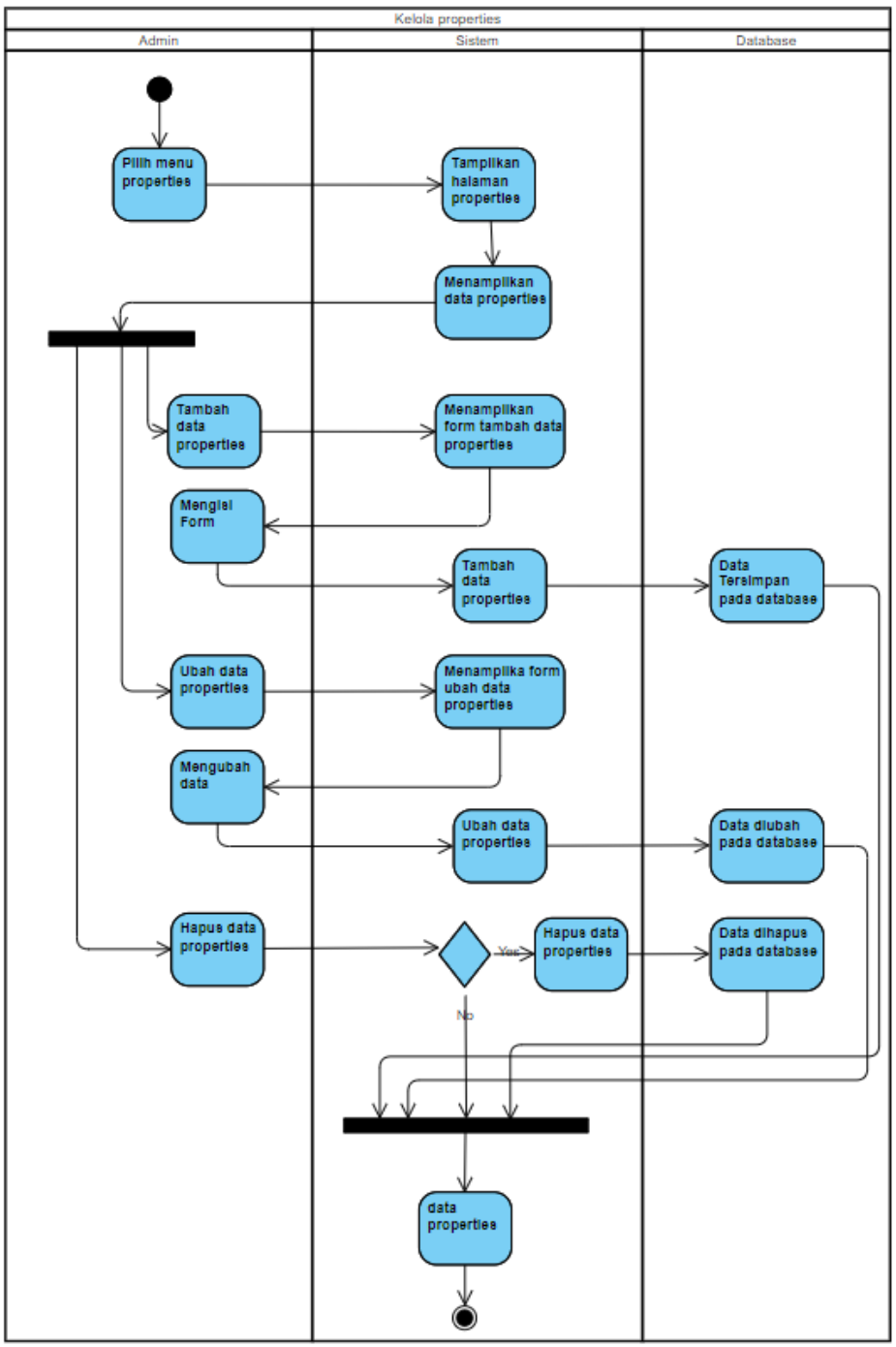
Gambar 3. 11 Login Page

Sumber : hasil kerja praktikan

Berdasarkan pada gambar 3.10 dan 3.11 yang merupakan login page untuk admin yang mana dibuat berdasarkan dari hasil analisa praktikan terhadap *activity*

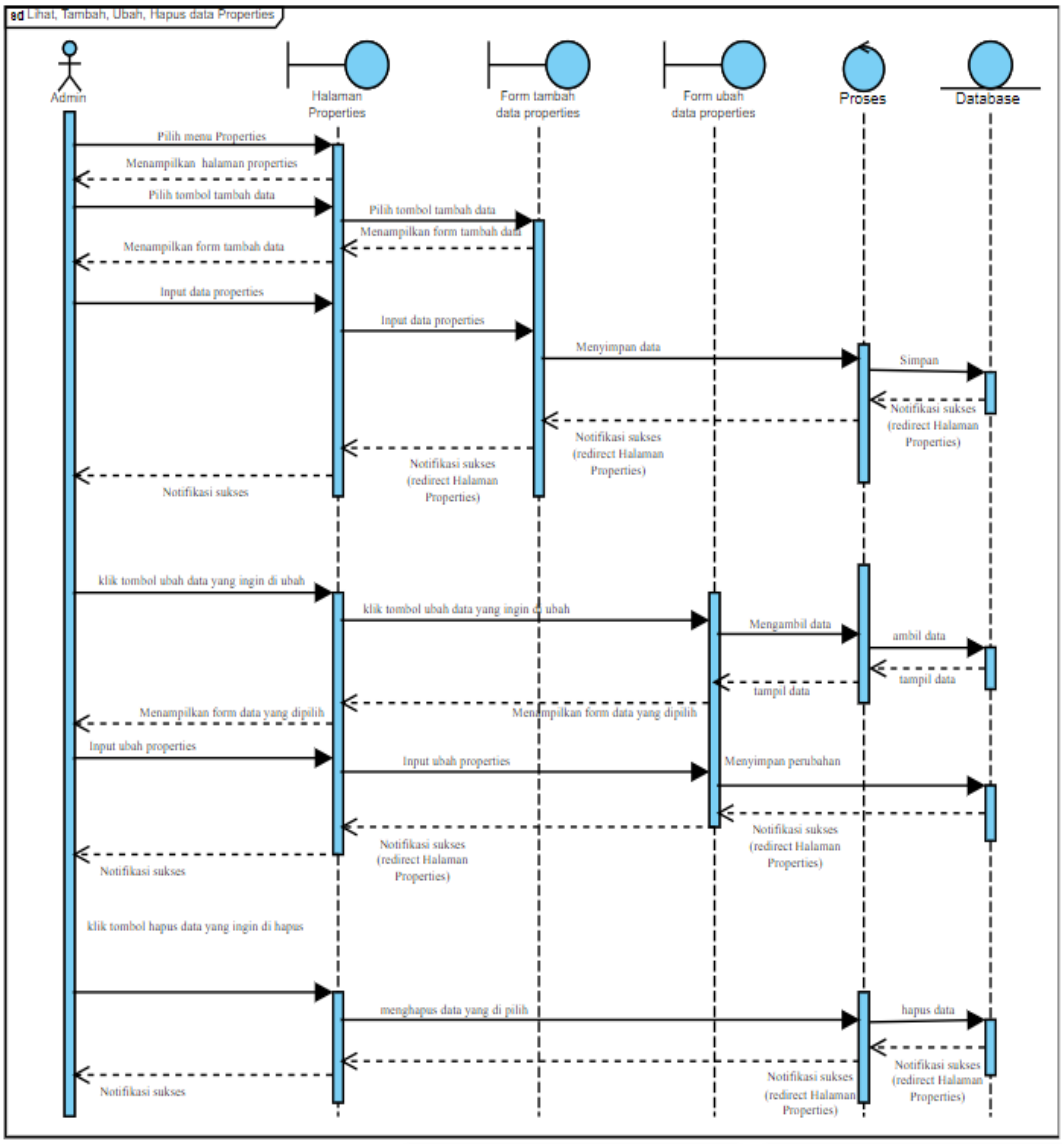
diagram dan *sequence diagram* yang dibuat oleh analis dan telah disetujui juga oleh analis terhadap *form* login yang praktian buat.

Setelah membuat login page praktikan melanjutkan untuk masuk ke tahap selanjutnya yakni membuat sistem untuk admin. Dimana sistematis alur aplikasinya dapat dilihat pada gambar 3.12 ini.



Gambar 3. 12 Activity Diagram Properties

Sumber : hasil kerja analis



Gambar 3. 13 Sequence Diagram Properties

Sumber : hasil kerja analis

Seusai menganalisa activity diagram dan sequence diagram yang telah dibuat oleh analis dan menentukan *requirement* apa saja yang dibutuhkan di tahap selanjutnya yaitu menu *porperties.* Praktikan melakukan pembuatan halaman *properties* sesuai dengan ketentuan yang telah dibuat oleh sistem analis

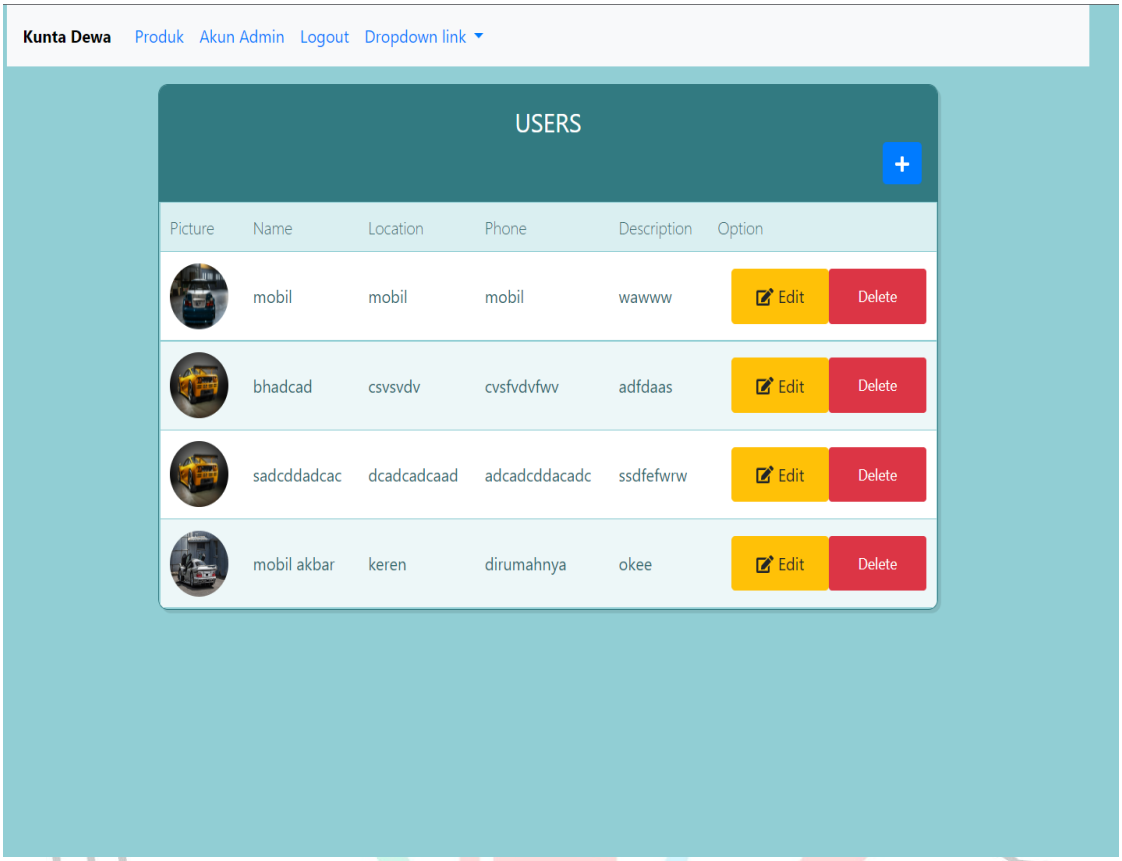
resources > views > controls > admins > index-properties.blade.php

```
@include('controls/admins.outline.header')

<div class="table-users">
   <div class="header">Users
      <div class="d-flex justify-content-end">
         <a href="/add-properties" class="btn btn-primary ">
            <i class="fas fa-plus">
            </i>
         </a>
      </div>
   </div>

   <table cellspacing="0">

      <tr>
         <th>Picture</th>
         <th>Name</th>
         <th>Location</th>
         <th>Phone</th>
         <th>Description</th>
         <th>Option</th>
      </tr>

      @foreach($properties as $property)
      <tr>
         <td>
            @php $property_images = json_decode($property->image); @endphp
            <img src="{{ asset('images/blueprints/'. $property_images[0]) }}">
         </td>
         <td>{{ $property->properties_name }}</td>
         <td>{{ $property->type }}</td>
         <td>{{ $property->location }}</td>
         <td>{{ $property->properties_description }}</td>
         <td>
            <div class="w-100 d-flex justify-content-end">
               <button style="width: 100px; height: 50px;" href="/update-properties/{{ $property-
               <form method="post" class="deleteConfirm" data-route="{{route('destroyProperties',
                  @csrf
                  @method('delete')
                  <button style="width: 100px; height: 50px;" type="submit" class="btn btn-danger
               </form>
            </div>
         </td>
      </tr>
```

Gambar 3. 14 Source Code Index Properties

Sumber : hasil kerja praktikan

Dari struktur code diatas merupakan struktur code untuk *index properties* yang menjadi halaman utama dari menu atau halaman admin untuk menampilkan apa saja isi konten yang ada di halaman utama *company profile* atau halaman untuk user dan di halaman ini berisi fungsi atau *button* untuk *edit, delete,* dan *update* untuk isi konten yang ada di halaman user. Yang nantinya akan dibuat dan ditampilkan oleh *frontend*.



Gambar 3. 15 Halaman Index Properties

Sumber : hasil kerja praktikan

Gambar diatas merupakan tampilan dari halaman *index properties*. Dapat terlihat isi dari halaman *index properties* yang menampilkan isi dari konten yang telah di buat dan isi konten di halaman ini juga ditampilkan oleh *frontend* di halaman utama. Di halaman ini terdapat *navbar* yang berfungsi untuk navigasi di halaman admin. Terdapat juga *button edit, update,*dan *delete* yang berfungsi untuk menambahkan, mengubah atau menghapus isi konten.

Setelah membuat halaman *index properties* seperti yang ditampilkan pada gambar 3.14 dan gambar 3.15 diatas selanjutnya yang praktikan lakukan adalah membuat halaman add properties seperti yang terlampir pada gambar 3.16 dan 3.17 dibawah ini.

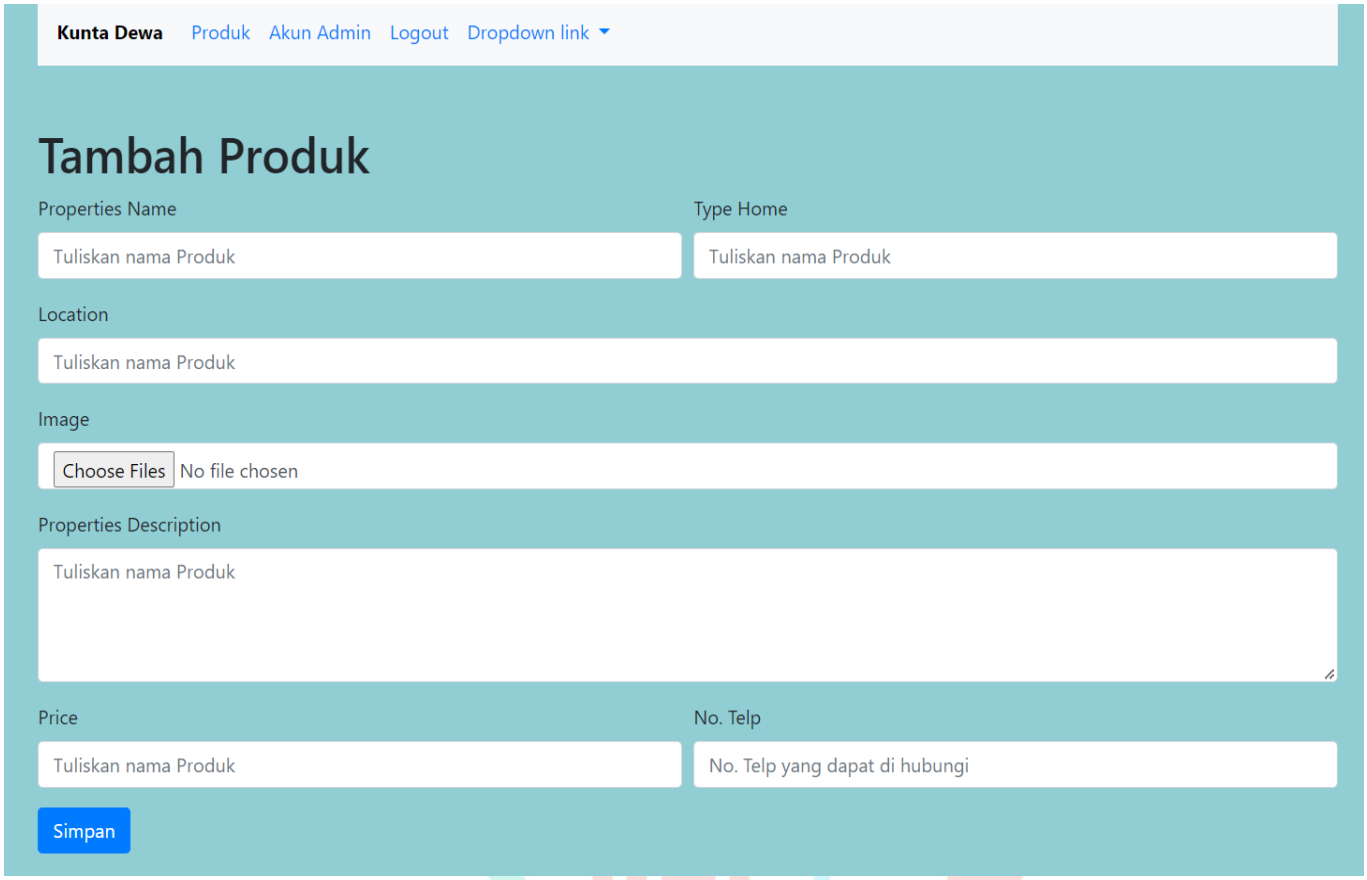
resources > views > controls > admins > add-properties.blade.php

```
@include('controls/admins.outline.header')

<div class="container mt-5">
    <h1>Tambah Produk</h1>
    <form action="/add-properties" method="POST" enctype="multipart/form-data" class="form">
        @csrf
        <div class="form-row">
            <div class="form-group col-6">
                <label for="properties_name" class="form-label">Properties Name</label>
                <input type="text" class="form-control" id="properties_name" name="properties_name" placeholder="Tuliskan
            </div>

            <div class="form-group col-6">
                <label for="type" class="form-label">Type Home</label>
                <input type="text" class="form-control" id="type" name="type" placeholder="Tuliskan nama Produk" required>
            </div>
        </div>

        <div class="form-row">
            <div class="form-group col-12">
                <label for="location" class="form-label">Location</label>
                <input type="text" class="form-control" id="location" name="location" placeholder="Tuliskan nama Produk">
            </div>
        </div>

        <div class="form-row">
            <div class="form-group col-12">
                <label for="image">Image</label>
                <input type="file" class="form-control" accept="image/*" id="image" name="image[]" multiple>
            </div>

            <div class="form-group col-12">
                <label for="properties_description" class="form-label">Properties Description</label>
                <textarea class="form-control" rows="4" id="properties_description" name="properties_description" placehol
            </div>
        </div>

        <div class="form-row">
            <div class="form-group col-6">
                <label for="price" class="form-label">Price</label>
                <input type="number" class="form-control" id="price" name="price" placeholder="Tuliskan nama Produk" requi
            </div>

            <div class="form-group col-6">
                <label for="notelp" class="form-label">No. Telp</label>
                <input type="number" class="form-control" id="notelp" name="notelp" placeholder="No. Telp yang dapat di hu
            </div>
        </div>
        <button type="submit" class="btn btn-primary">Simpan</button>
    </form>
```

Gambar 3. 16 Source Code Add Properties

Sumber : hasil kerja praktikan

Selanjutnya disini ada *source code add properties* dari *activity diagram* dan *sequence* yang dibuat oleh sistem analis terdapat fungsi untuk menambahkan isi konten untuk halaman utama *website company profile*. Yang bertujuan untuk mengisi konten properti apa saja yang ada dan disediakan oleh perusahaan yang dapat dilihat atau di reservasi oleh user. Halaman yang dihasilkan dari code diatas berupa halaman seperti gambar berikut ini.



Gambar 3. 17 Halaman Add Properties

Sumber : hasil kerja praktikan

Dari *source code* yang ada di gambar 3.16 menghasilkan halaman tambah produk seperti pada gambar 3.17 diatas yang mana berfungsi untuk menambahkan produk atau properti yang akan ditampilkan pada halaman utama untuk user. Admin dapat membuat isi konten baru di halaman ini dengan tujuan untuk ditampilkan di halaman utama. Admin dapat menginput nama properti, tipe dari properti, lokasi, contoh gambar dari properti deskripsi, harga beserta nomor telpon yang dapat dihubungi guna mendapatkan informasi lebih lanjut dari properti tersebut.

Dan setelah praktikan membuat halaman *add properties* selanjutnya praktikan membuat halaman untuk update produk yang praktikan tertera digambar 3.18 dan 3.19 sebagai berikut.

app > Http > Controllers > ReservationController.php

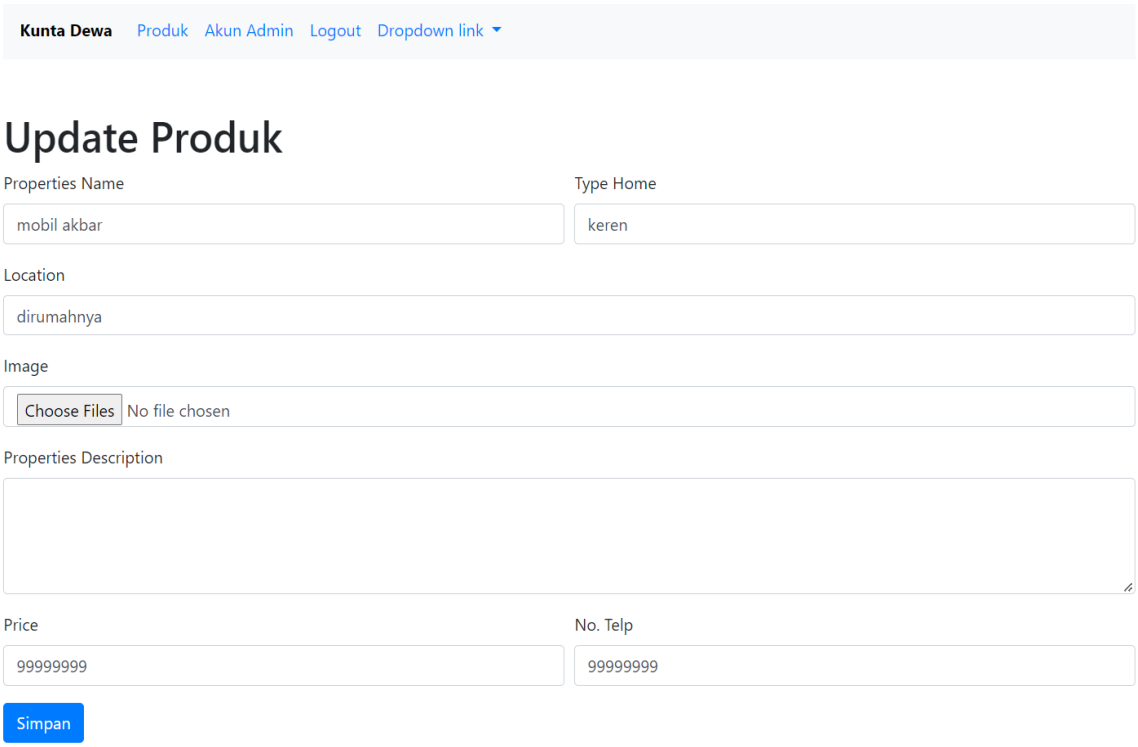
```php
<?php

namespace App\Http\Controllers;

use App\Models\Reservation;
use Illuminate\Http\Request;

class ReservationController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
    {
        $request->validate([
            'reservation_name' => 'required',
            'model' => 'required',
            'type' => 'required',
            'email' => 'required',
            'notelp' => 'required',
            'captcha' => 'required|captcha',
        ]);

        Reservation::create($request->all());

        return redirect("/")
            ->with('success', 'You have successfully created the properties.');
    }

    /**
     * Display the specified resource.
     *
```

Gambar 3. 18 Source Code Update Properties

Sumber : hasil kerja praktikan

Dari gambar 3.18 ini terdapat struktur kode dari *update properties* kode ini berfungsi untuk mengubah isi konten yang sebelumnya telah diinput pada halaman *add properties* yang ada pada gambar 3.16 dan 3.17 dan admin dapat mengubah isi konten yang ada sehingga tidak perlu menghapus kemudian menambhakan ulang isi konten yang mana akan merepotkan dan juga menghabiskan waktu dan tidak efisien.

Dari struktur kode yang ada diatas akan membuat tampilan seperti pada gambar 3.19 dibawah ini.



Gambar 3. 19 Halaman Update Properties

Sumber : hasil kerja praktikan

Seperti yang sudah dijelaskan diatas fungsi dari *update properties* adalah untuk mengubah isi dari konten yang sebelumnya telah ditambahkan oleh admin. Sehingga admin tidak perlu untuk menhapus konten yang sudah ada kemudian membuat ulang lagi. Dengan fitur ini admin dapat mengubah saja konten yang sudah ada sebelumnya. Tampilan pada menu *update properties.* Serupa dengan menu *add properties* seperti pada gambar 3.17 hanya ada sedikit perbedaan di warna background untuk membedakan agar admin tidak keliru.

Itu adalah beberapa CRUD yang praktikan kerjakan untuk pada proyek aplikasi *web company profile* di PT. Bumi Rejeki Agung.

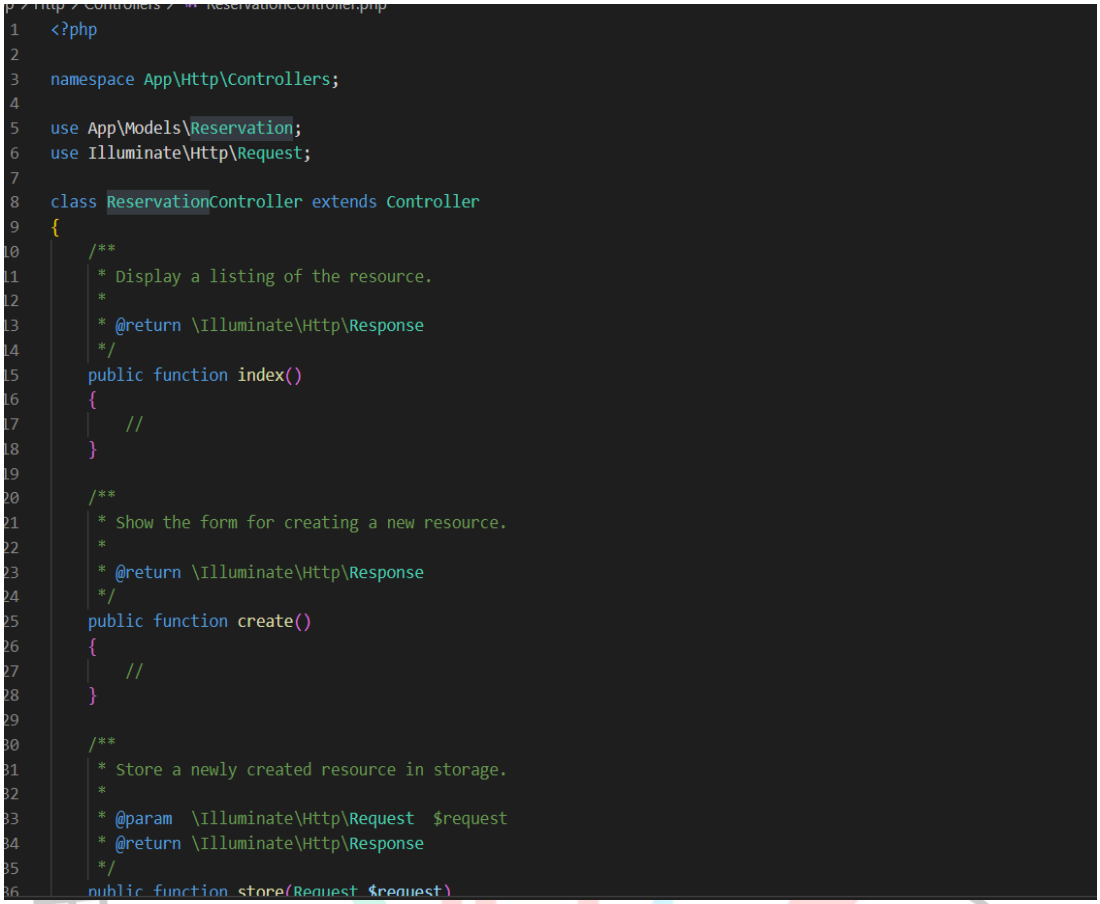
3.2.3 Controller

Terakhir ada controller untuk halaman User yang praktikan buat. Controller yang dimaksud adalah untuk properties dan juga untuk reservasi untuk mengkoneksikan admin dan user dengan *database* yang sudah dibuat maka harus dibutuhkan controller. Berikut ini adalah controller yang praktikan buat seperti yang ada di gambar 3.20 dan 3.21.

```php
<?php

namespace App\Http\Controllers;

use App\Models\Properties;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\View;

class PropertiesController extends Controller
{

    public function properties()
    {
        $title = 'Home';
        $properties = Properties::all();
        return View::make('index')
            ->with('title', $title)
            ->with(compact('properties'));
    }

    public function indexProperties()
    {
        $title = 'Detail Properties';
        $properties = Properties::all();
        return View::make('controls.admins.index-properties')
            ->with('title', $title)
            ->with(compact('properties'));
    }

    public function addProperties(Request $request)
    {
        $validated = $request->validate([
            'properties_name' => 'required',
            'type' => 'required',
            'location' => 'required',
            'image' => 'required|array',
            'image.*' => 'required|image|mimes:jpg,jpeg,png,gif|max:10240',
            'properties_description' => 'required',
            'price' => 'required',
            'notelp' => 'required',
        ]);

        $imgName = [];
        foreach ($request->file('image') as $img) {
```

Gambar 3. 20 Controller Properties

Sumber : hasil kerja praktikan

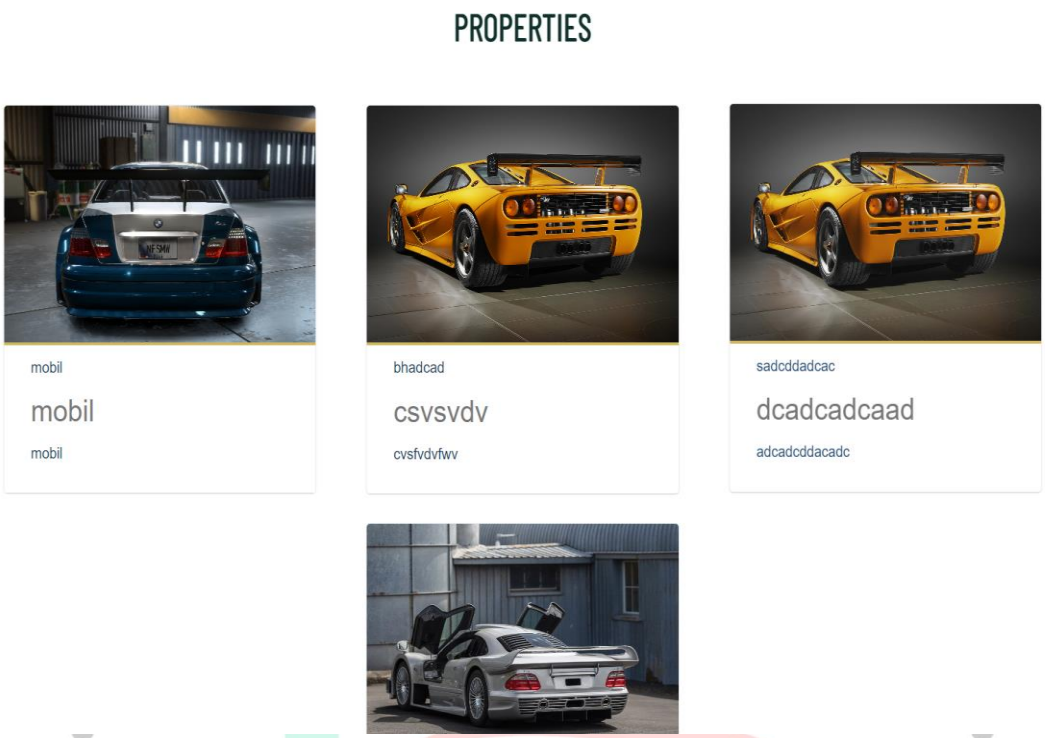


```php
<?php

namespace App\Http\Controllers;

use App\Models\Reservation;
use Illuminate\Http\Request;

class ReservationController extends Controller
{
    /**
     * Display a listing of the resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function index()
    {
        //
    }

    /**
     * Show the form for creating a new resource.
     *
     * @return \Illuminate\Http\Response
     */
    public function create()
    {
        //
    }

    /**
     * Store a newly created resource in storage.
     *
     * @param  \Illuminate\Http\Request  $request
     * @return \Illuminate\Http\Response
     */
    public function store(Request $request)
```

Gambar 3. 21 Controller Reservation

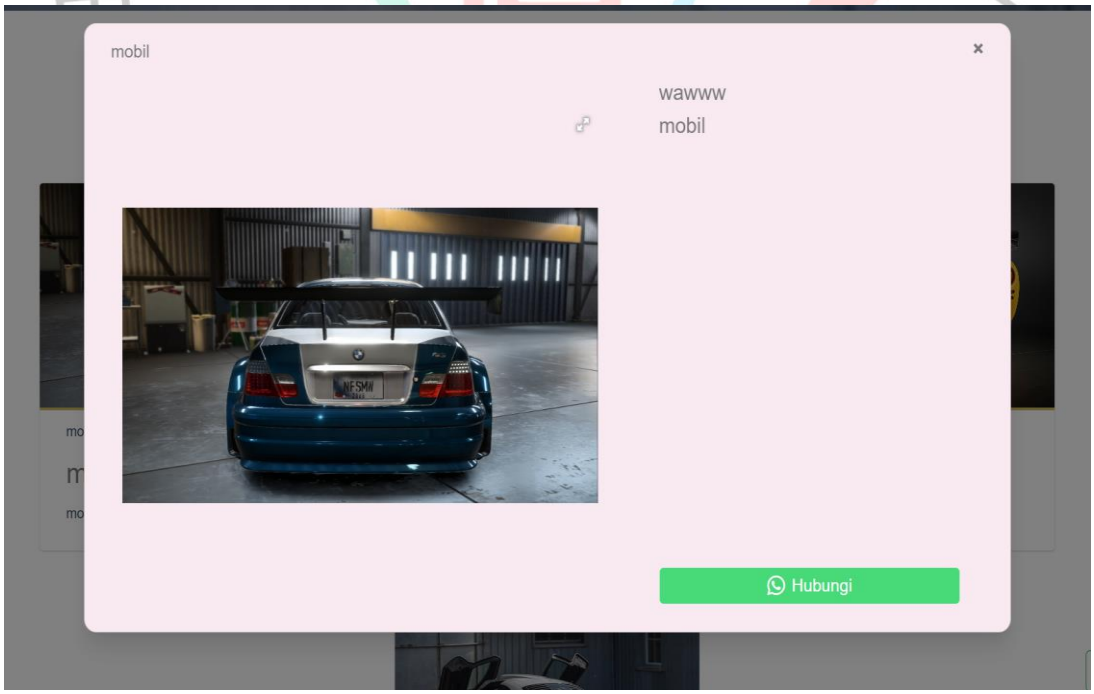
Sumber : hasil kerja praktikan

Itulah controller yang praktikan buat adanya controller dikarenakan menggunakan model MVC dan menggunakan framework laravel. Karena MVC dengan framework lebih mudah digunakan jika dikerjakan Bersama sebab dengan adanya folder yang peletakannya sudah disusun default oleh framework dapat memudahkan Backend dan Frontend dalam bekerja sama. Mengapa menggunakan Laravel dikarenakan framework ini sedang umum dan mudah mencari dokumentasi di internet sehingga lebih mudah dipelajari dan dipahami dan juga dengan basis PHP yang dimana cukup dikuasai oleh praktikan dan rekan-rekan sehingga pengerjaan jauh lebih cepat. Dan dari controller yang telah praktisi buat terkoneksi menuju halaman utama yang dibuat oleh bagian frontend yang terlihat digambar 3.22, gambar 3.23, dan juga gambar 3.24.



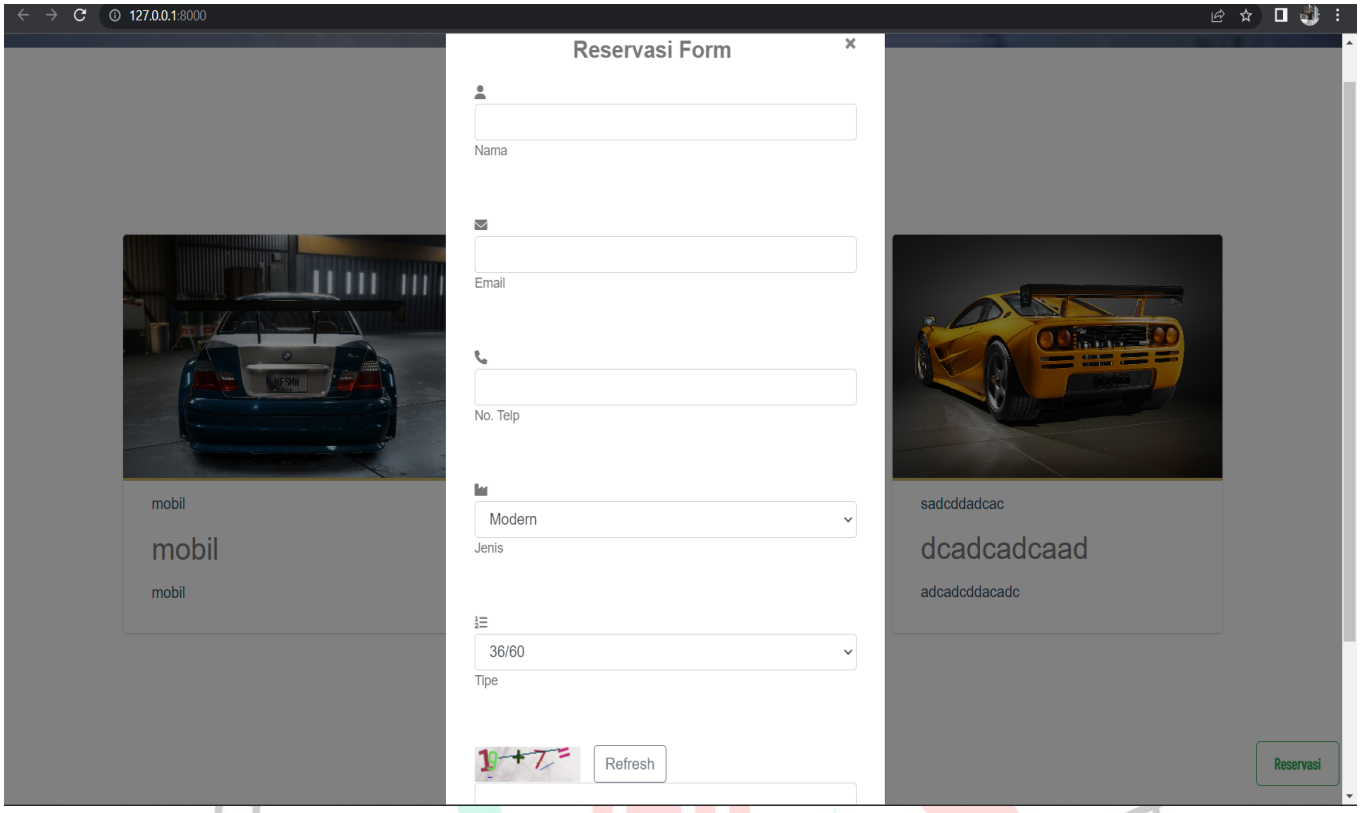
Gambar 3. 22 Frontend Properties

Sumber : hasil kerja frontend



Gambar 3. 23 Frontend Detail Properties

Sumber : hasil kerja frontend



Gambar 3. 24 Frontend Reservasi

Sumber : hasil kerja frontend

3.3 Kendala Yang Dihadapi

Dari pekerjaan yang praktikan lakukan selama 3 bulan di PT. Bumi Rejeki Agung terdapat beberapa kendala. Terdapat beberapa kendala yang dialami oleh praktikan, maka dari itu praktikan menjabarkan apa saja kendala yang dialami dan anatara lain:

1. Lokasi kantor yang jauh, ini tidak terlalu fatal tapi merupakan kendala yang cukup menyulitkan dikarenakan jika ingin ke lokasi praktikan dan rekan harus menempuh jarak kurang lebih 40 kilometer membuat praktikan kelelahan dan mengurangi efektifitas waktu serta tenaga.
2. Kurangnya tim developer untuk memanage waktu dengan baik, sehingga proyek berjalan tidak tepat waktu.

3. Komunikasi dan koordinasi yang kurang berjalan dengan baik. Tim developer selalu saling menunggu sehingga komunikasi selalu stuck, dan tidak ada orang yang berinisiatif untuk mengawali komunikasi dan koordinasi sehingga sering terjadi kesalahpahaman pada komunikasi.

Itu adalah beberapa kendala yang dialami praktikan dalam menjalankan proyek perancangan aplikasi web company profile di PT. Bumi Rejeki Agung.

3.4 Cara Mengatasi Kendala

Dari kendala yang telah disebutkan sebelumnya. Praktikan menguraikan strategi berikut untuk mengatasi hambatan-hambatan tersebut:

1. Melakukan pekerjaan dengan sistem hybrid. Dengan cara hybrid bisa juga memenuhi target dengan membuat timeplan dan workplan menggunakan gantt chart dan memiliki IT manager yang selalu meremind kerjaan harian dari parktikan dan rekan.
2. Melakukan pencatatan apa yang sudah dilakukan, agar masing masing individu dapat mengelola waktu dan pekerjaan yang dilakukan dengan baik dan efisien

Itulah beberapa cara yang efektif untuk mengatasi kendala tersebut agar proyek dapat berjalan dengan baik dan juga sesuai dengan scope yang diinginkan oleh PT Bumi Rejeki Agung.

3.5 Pembelajaran Yang Diperoleh dari Kerja Profesi

Pelajaran yang dapat praktikan ambil dari Kerja Profesi ini adalah belajar bertanggung jawab dengan pekerjaan yang dilakukan. Karena jika tidak adanya rasa tanggung jawab dengan apa yang seharusnya dikerjakan akan menimbulkan banyak masalah dikedepannya seperti menyulitkan rekan-rekan yang mana telah berdedikasi tinggi dalam proyek ini, menghilangkan kepercayaan orang lain juga menjadi resiko jika tidak bertanggung jawab, dan juga dapat menghambat berkembangnya orang lain karena seharusnya jika proyek ini berjalan dengan baik membuat praktikan dan rekan-rekan dapat berkembang dan memiliki ilmu baru. Kemudian ada juga pelajaran tentang manajemen waktu yang ternyata

tidak mudah dilakukan dalam sebuah proyek yang melibatkan banyak orang dengan pemikiran dan cara kerja yang berbeda beda. Disini praktikan dan rekan harus menyatukan tujuan atau visi dan misi yang selaras dan komitmen yang sama sehingga jalannya proyek akan tepat waktu. Yang terakhir adalah pelajaran bagaimana bekerjasama dan berkordinasi dengan baik dengan orang lain. Mungkin pengalaman ini tak hanya didapatkan di sebuah pekerjaan pengalaman ini juga bisa didapatkan dalam berorganisasi di lingkunan kampus. Walapun pengalaman ini bisa didapatkan di kampus akan tetapi ada beberapa perbedaan yang di dapatkan di dunia industri ini dimana scope perusahaan jauh lebih besar dan goals yang dituju juga untuk memajukan perusahaan. Kemudian hal yang dapat dipelajari adalah penyesuaian di dunia industri yang mana praktikan sendiri masih kurang pengalaman di sebuah perusahaan, menjadikan pelajaran berharga pengalaman yang praktikan dapat selama bekerja 3 bulan di PT. Bumi Rejeki Agung ini. Itulah beberapa pelajaran yang bisa praktikan ambil dari pelaksanaan Kerja Profesi ini.