LAPORAN INSTALASI LARAVEL

**INSTALASI LARAVEL DAN KONEKSI KE DATABASE MYSQLI MENGGUNAKAN PDO**

KELOMPOK 3 :

Rahmad Nakula Z

Felix Vannial

Akmal

Salsabilla Zahrani Purnama



**PENDAHULUAN**

* 1. **Pengertian Laravel**

Laravel adalah sebuah framework PHP yang dirilis dibawah lisensi MIT, dibangun dengan konsep MVC (model view controller). Laravel adalah pengembangan website berbasis MVP yang ditulis dalam PHP yang dirancang untuk meningkatkan kualitas perangkat lunak dengan mengurangi biaya pengembangan awal dan biaya pemeliharaan, dan untuk meningkatkan pengalaman bekerja dengan aplikasi dengan menyediakan sintaks yang ekspresif, jelas dan menghemat waktu.

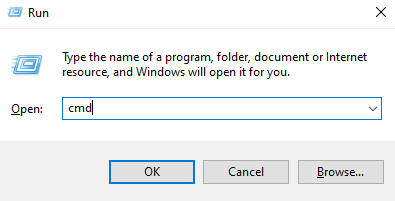
MVC adalah sebuah pendekatan perangkat lunak yang memisahkan aplikasi logika dari presentasi. MVC memisahkan aplikasi berdasarkan komponen- komponen aplikasi, seperti : manipulasi data, controller, dan user interface.

1. Model, Model mewakili struktur data. Biasanya model berisi fungsi-fungsi yang membantu seseorang dalam pengelolaan basis data seperti memasukkan data ke basis data, pembaruan data dan lain-lain.
2. View, View adalah bagian yang mengatur tampilan ke pengguna. Bisa dikatakan berupa halaman web.
3. Controller, Controller merupakan bagian yang menjembatani model dan view.
   1. **Fitur-fitur Laravel**
4. Bundles yaitu sebuah fitur dengan system pengemasan modular dan berbagai bundle telah tersedia untuk di gunakan dalam aplikasi Anda.
5. Eloquent ORM merupakan penerapan PHP lanjutan dari pola “active record”, menyediakan metode internal untuk mengatasi kendala hubungan antara objek database. Pembangun query Laravel’s Fluent ini didukung oleh Eloquent.
6. Application logic merupakan bagian dari aplikasi yang dikembangkan, baik menggunakan Controllers atau sebagai bagian dari deklarasi Route. Sintak yang digunakan untuk mendefinisikannya mirip dengan yang dugunakan oleh framework Sinatra.
7. Reverse routing, mendefinisikan hubungan antara Link dan Route, sehingga jika suatu saat ada perubahan pada route secara otomatis akan tersambung dengan link yang relevan. Ketika Link yang dibuat dengan menggunakan nama-nama dari Route yang ada, secara otomatis Laravel akan membuat URI yang sesuai.
8. Restful controllers, memberikan sebuah option (pilihan) untuk memisahkan logika dalam melayani HTTP GET dan permintaan POST.
9. Class auto loading, menyediakan otomatis loading untuk class-class PHP, tanpa membutuhkan pemeriksaan manual terhadap jalur masuknya. Fitur ini mencegah loading yang yang tidak perlu.
10. View composers adalah kode unit logical yang dapat dieksekusi ketika sebuah View diload.
11. IoC Container memungkinkan untuk objek baru yang dihasilkan dengan mengikuti prinsip control pembalik, dengan pilhan contoh dan referensi dari objek baru sebagai singletons.
12. Migrations menyediakan versi sistem control untuk skema database, sehingga memungkinkan untuk menghubungkan perubahan dalah basis kode aplikasi dan keperluan yang dibutuhkan dalam merubah tata letak database, mempermudah dalam penempatan dan memperbarui aplikasi.
13. Unit Testing mempunyai peran penting dalan framework Laravel, dimana unit testing ini mempunyai banyak tes untuk medeteksi dan mencegah regresi. Unit testing ini dapat dijalankan melalui utilitas “artisan command-line”
14. Automatic pagination menyederhanakan tugas dari penerapan halaman, menggantikan penerapan yang manual dengan metode otomatis yang terintregrasi ke Laravel.
    1. **Instalasi Laravel**

Dalam melakukan installasi laravel di bagi menjadi dua bagian yaitu via install laravel dan via composer. Pada installasi laravel kali ini kita menggunakan via composer, dan pada instalasi laravel via composer terdapar dua cara, yang pertama menggunakan composer create project yang kedua menggunakan composer install.

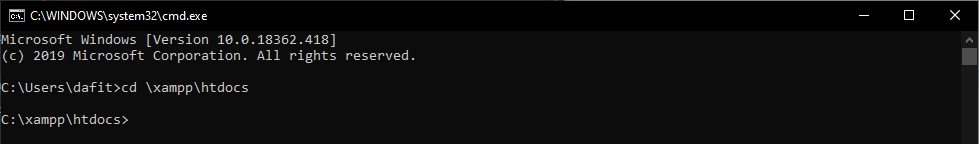
* + - 1. **Masuk Command Prompt**

Langkah pertama dalam install Laravel adalah masuk Command Prompt. Caranya adalah klik Win+R lalu ketik cmd dan klik OK



* + - 1. **Masuk ke Folder XAMPP**

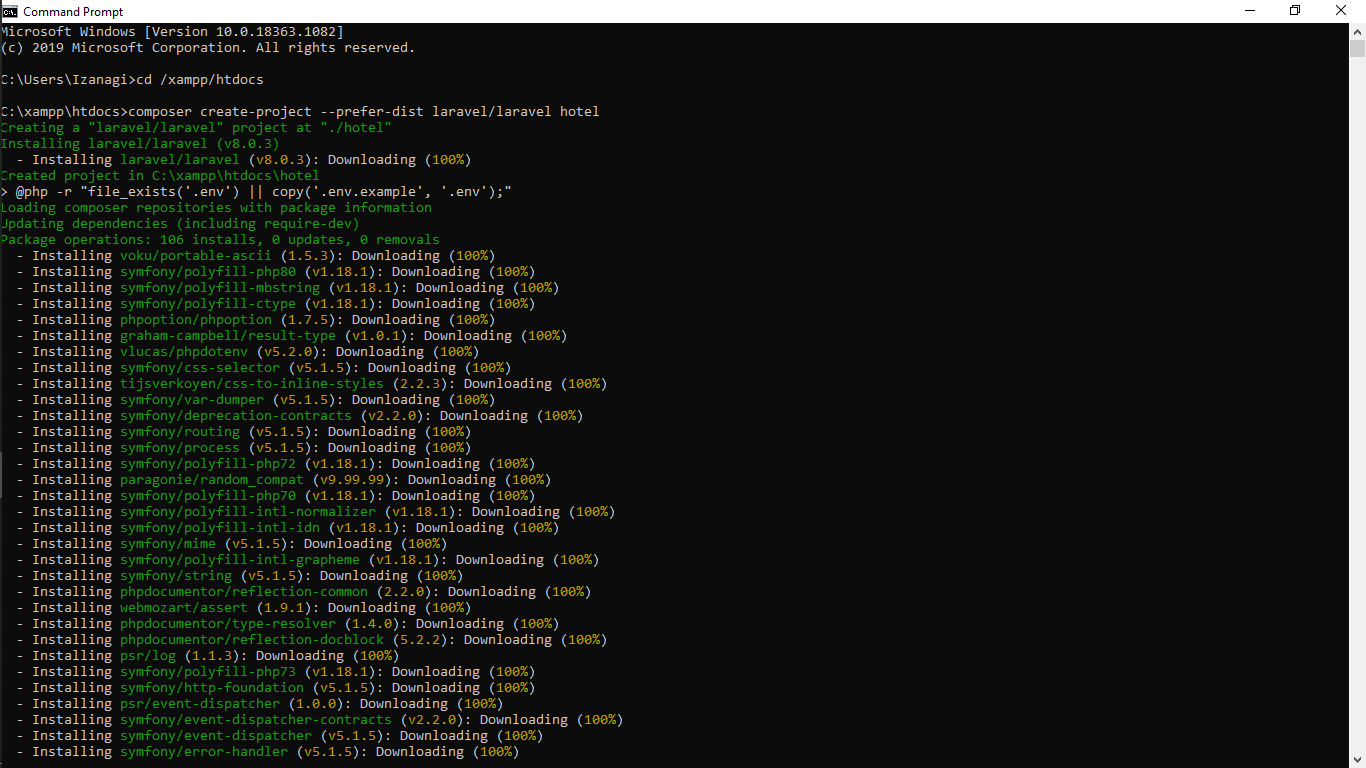
Sebelum melakukan instalasi Laravel, arahkan Command Prompt atau terminal menuju direktori file server. Lokasi file server pada XAMPP secara default berada pada direktori xampp/htdocs. Masukan perintah ini pada jendela Command Prompt untuk masuk ke direktori htdocs.



* 1. **Mulai Proses Install Laravel**

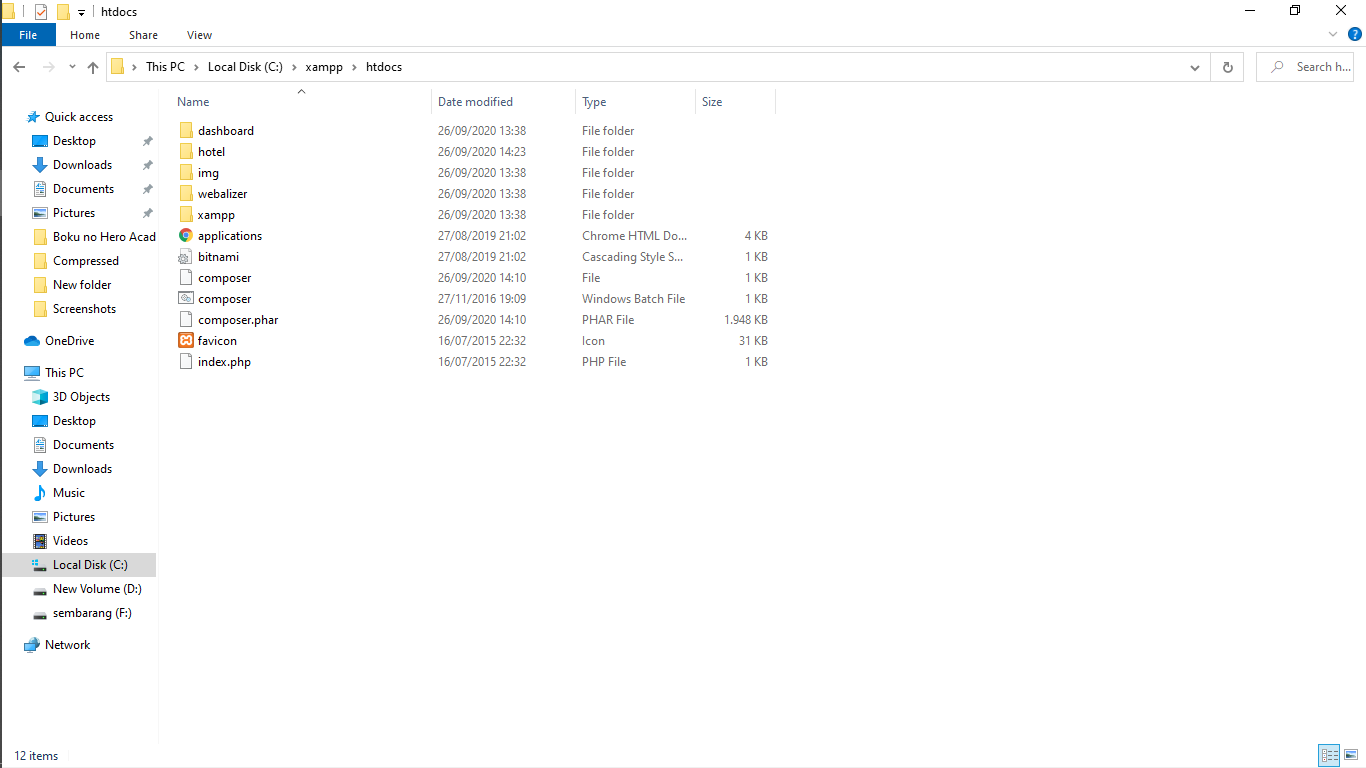
Selanjutnya jika sudah masuk direktori htdocs, Anda harus membuat request untuk mengambil (serta menginstall) file Laravel yang telah disediakan dalam repositori Github. Gunakan perintah ini untuk melakukan request:

composer create-project --prefer-dist laravel/laravel nama\_projectmu Jika perintah telah berhasil dimasukkan, Composer akan mulai melakukan proses pengambilan data serta instalasi Laravel ke dalam direktori yang telah Anda tentukan. Pastikan bahwa koneksi internet dalam keadaan stabil agar tidak terjadi gangguan pada saat proses pengambilan data Laravel.



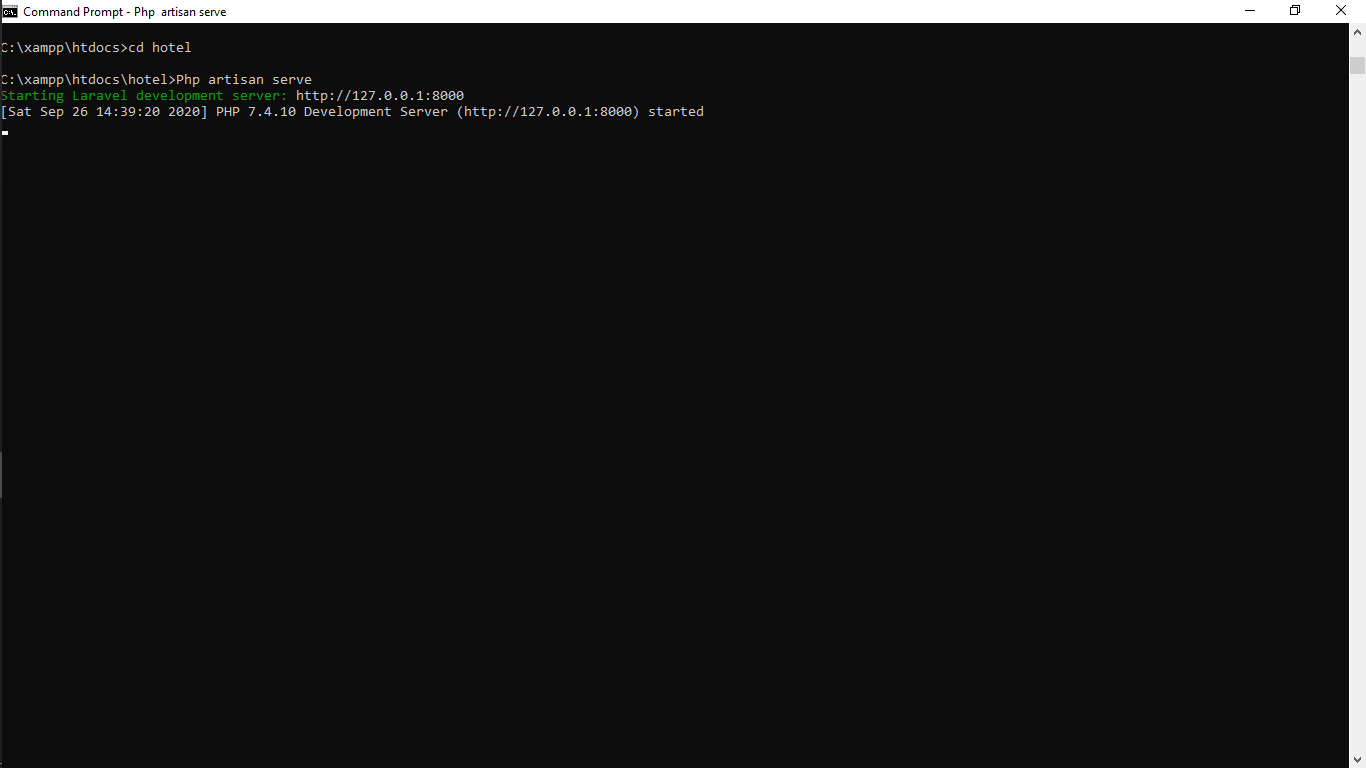
* 1. **Cek Instalasi Laravel di Web Browser**

Setelah proses download file Laravel selesai, nantinya akan ada folder baru pada direktori file server dengan nama sesuai nama project yang telah Anda tentukan sebelumnya pada folder /xampp/htdocs.

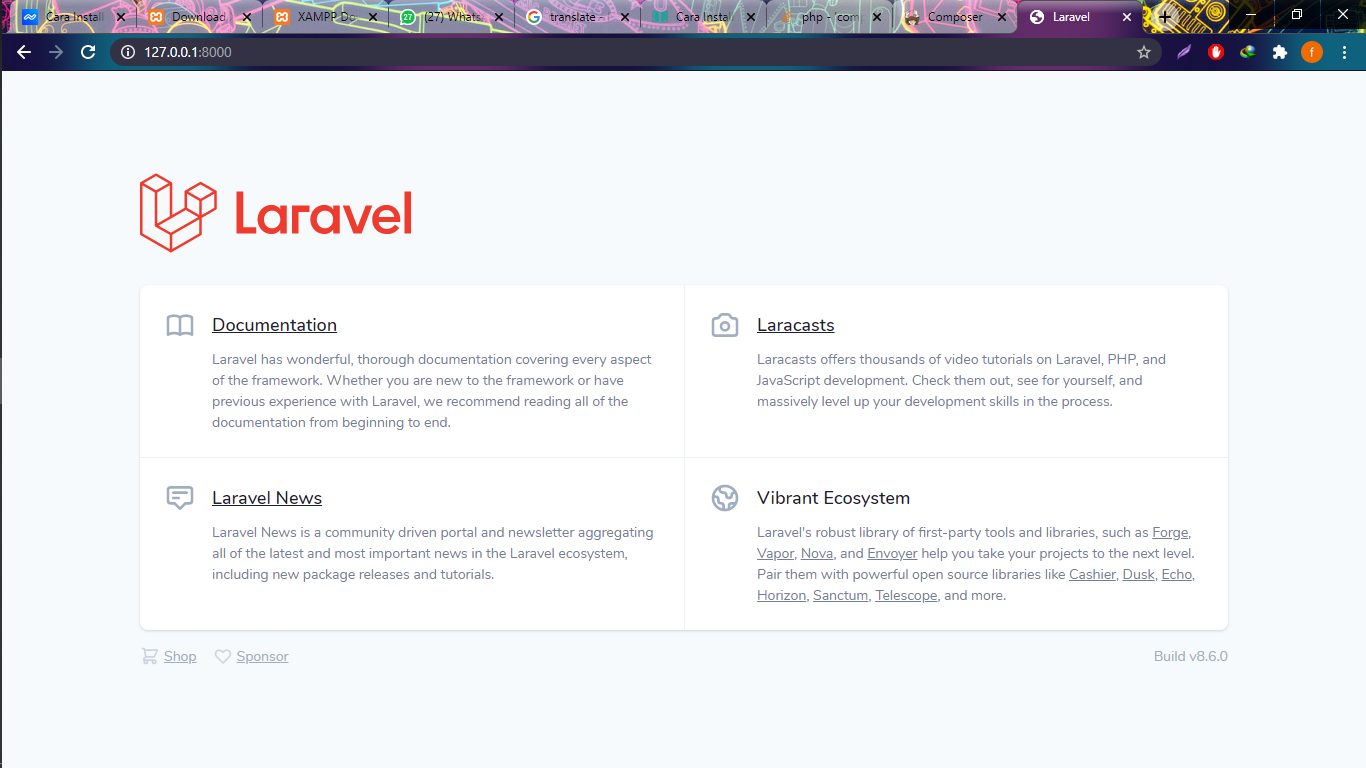


Untuk memastikan bahwa Laravel sukses terinstall dan siap untuk digunakan, arahkan Command Prompt atau Terminal menuju direktori yang telah Anda buat sebelumnya. Lalu, masukkan perintah berikut ke dalam Command Prompt atau Terminal:

Php artisan serve



Jika muncul tulisan Laravel development server started pada Command Prompt atau Terminal, langkah selanjutnya adalah membuka link yang telah disediakan oleh Laravel. Secara default, Anda akan diarahkan menuju alamat server,yaitu 127.0.0.1:8000. Nantinya, akan muncul tampilan homepage dengan tulisan Laravel di bagian tengah seperti pada gambar di bawah ini:



* 1. **Koneksi menggunakan PDO pada halaman page php dan html biasa**

Karena **PDO** adalah fungsi/fitur PHP yang ditulis menggunakan objek, maka untuk membuat koneksi antara PHP dengan MySQL, kita menggunakan **PDO constructor**, yakni dengan membuat objek baru dari class PDO (menggunakan keyword: **new**).

Berikut adalah struktur dasar cara penulisan *PDO constructor* untuk database MySQL:



Variabel **$dbh** adalah variabel yang akan menjadi **object PDO**. Object ini biasa disebut sebagai **“Database Handler”**. Object inilah yang nantinya akan kita gunakan untuk menjalankan perintah-perintah PDO. Nama variabel boleh bebas, dan tidak harus **$dbh**.

Argumen dari *constructor* PDO terdiri dari 3 bagian, bagian pertama berisi **nama database server** (misalnya:mysql) kemudian diikuti dengan **alamat server** dan **nama database**, semuanya di dalam 1 string. Untuk argumen kedua diisi dengan **nama user**, dan argumen ketiga berisi **password user**.

Sebagai contoh, untuk masuk ke dalam MySQL server di komputer **localhost**, dengan user **root,** password:**12345**, nama database: **universitas**, kode PDO yang digunakan adalah sebagai berikut:

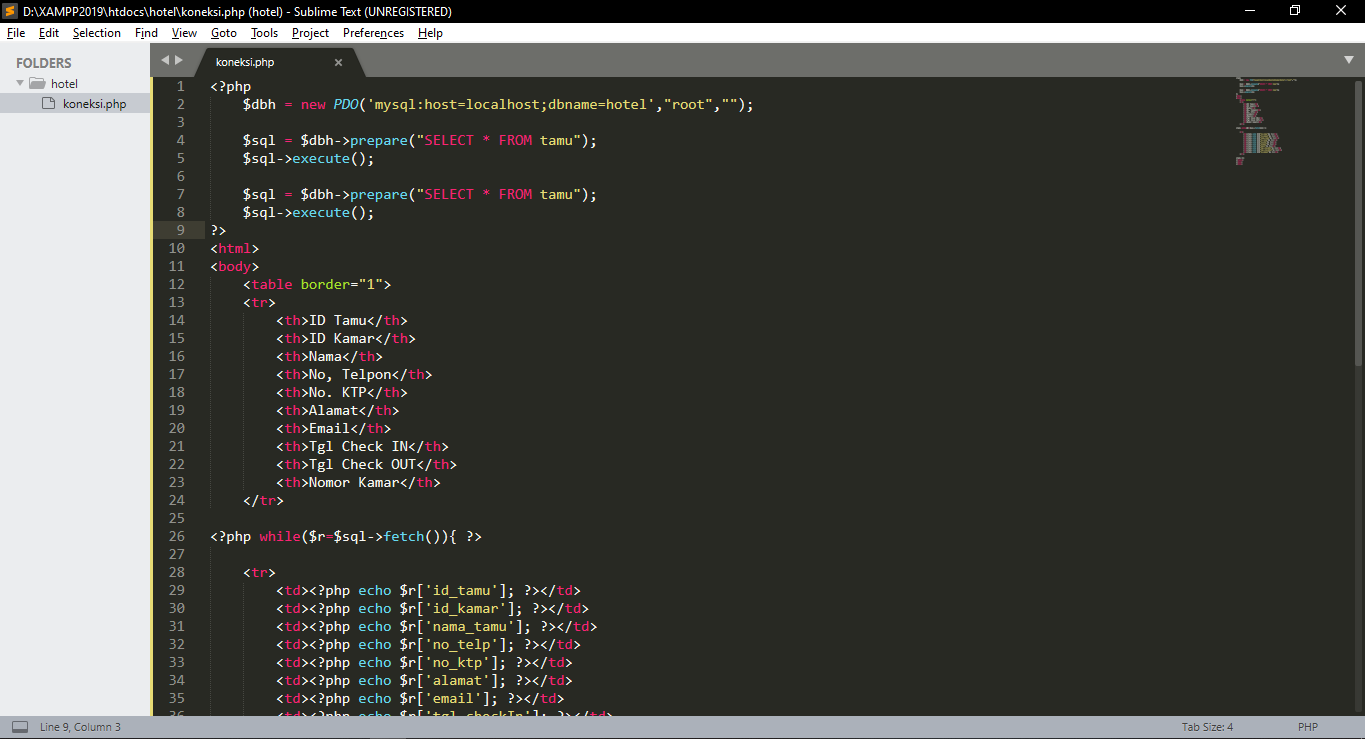


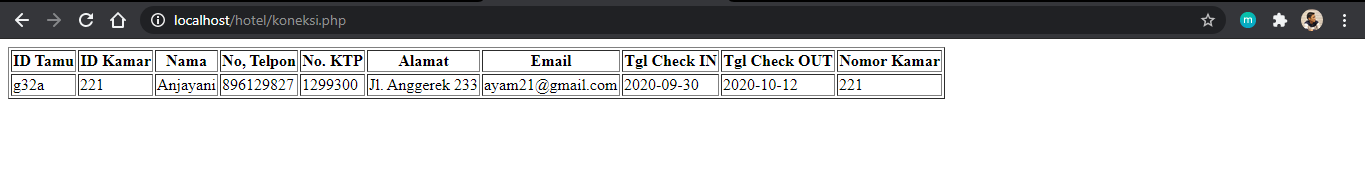
Setelah membuat koneksi PDO, untuk menghapus koneksi kita tinggal memberikan nilai “**null**” kepada variabel **“Database Handler”** sebagai berikut:



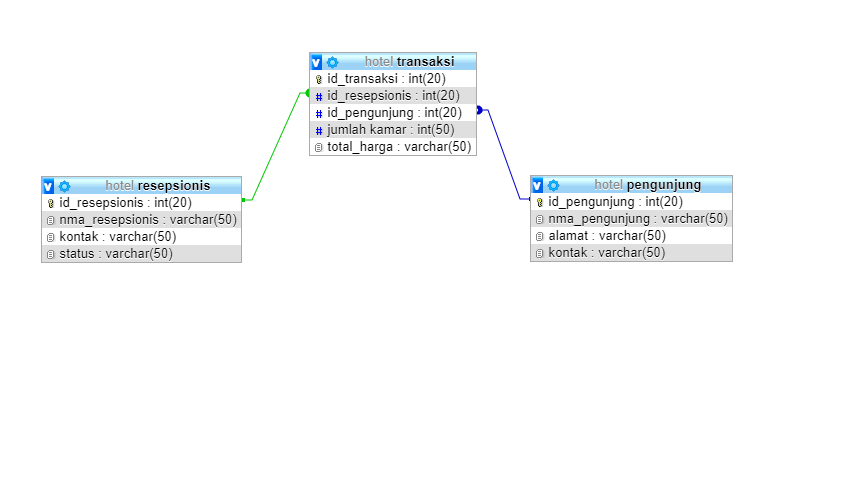
Sama seperti **mysqli**, proses penutupan koneksi ini bersifat opsional, dan PHP secara otomatis akan menghapus koneksi pada saat halaman PHP selesai dijalankan.

Hasil :

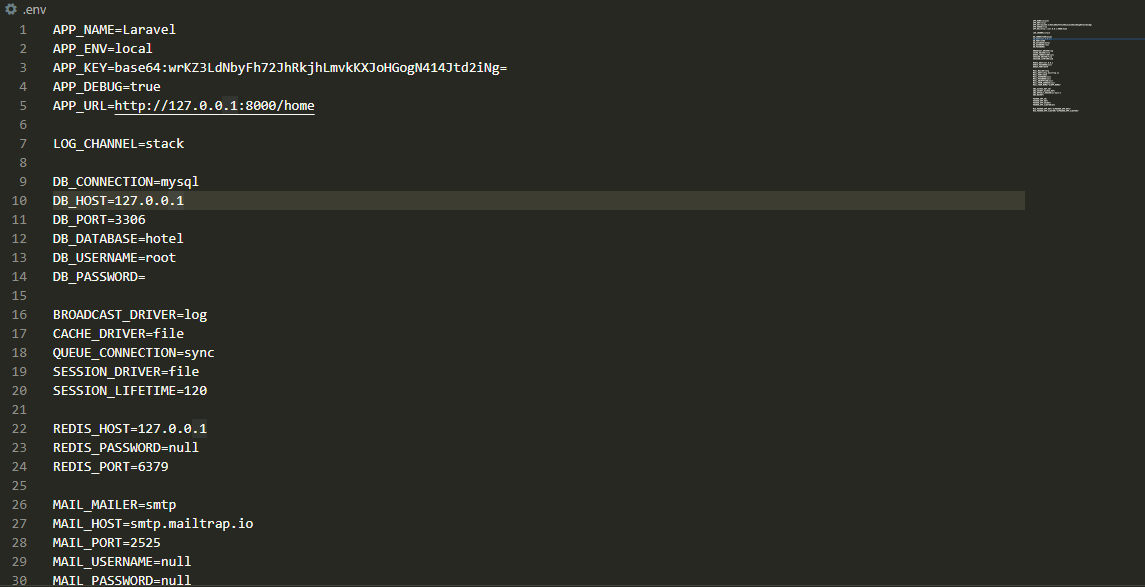




* 1. **Koneksi Database MYSQLI menggunakan PDO di Laravel**

****

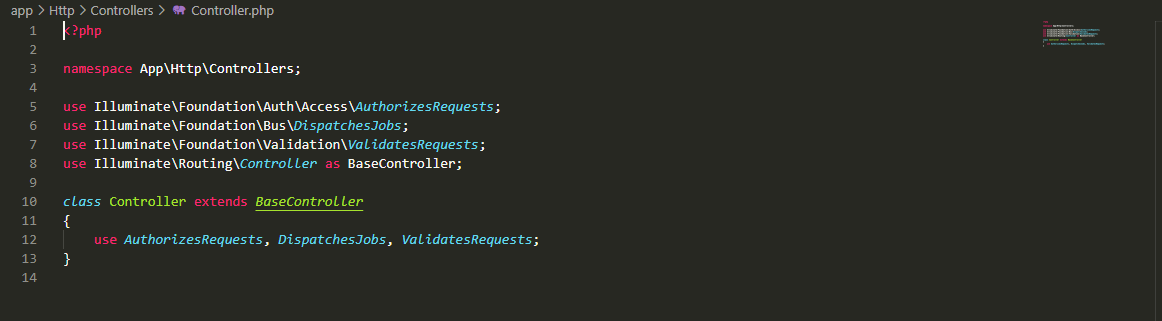
1. **Set Database pada laravel dengan mengubah di environment (.env)**



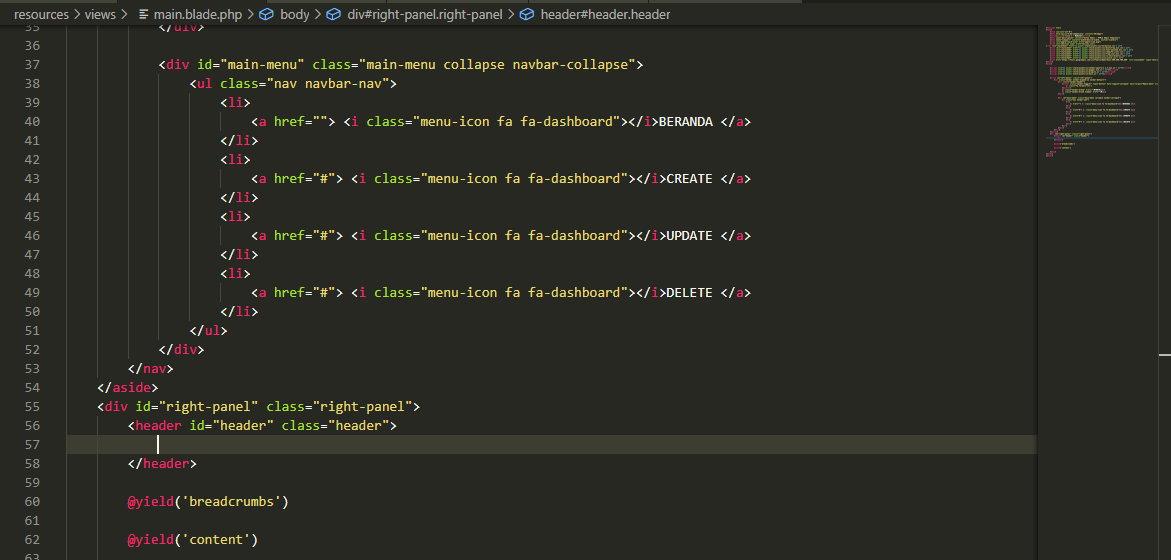
1. **Pada Folder routes buka file bernama web.php dan ubah code di dalamnya**



1. **Setelah menambahkan kode untuk web.php buat file baru di folder Controller dengan nama controller.php lalu masukan kode di dalam filenya**



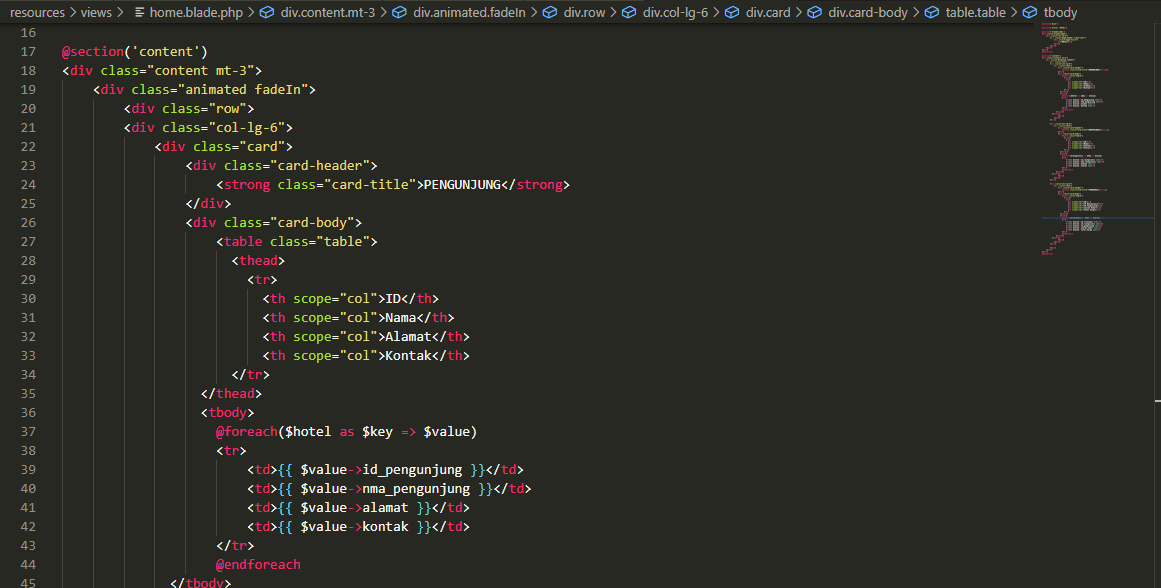
1. **Setelah membuat file controller.php silahkan membuat file index.php untuk menampilkan data di dalam folder resources>view dan isi dengan code yang telah dibuat**



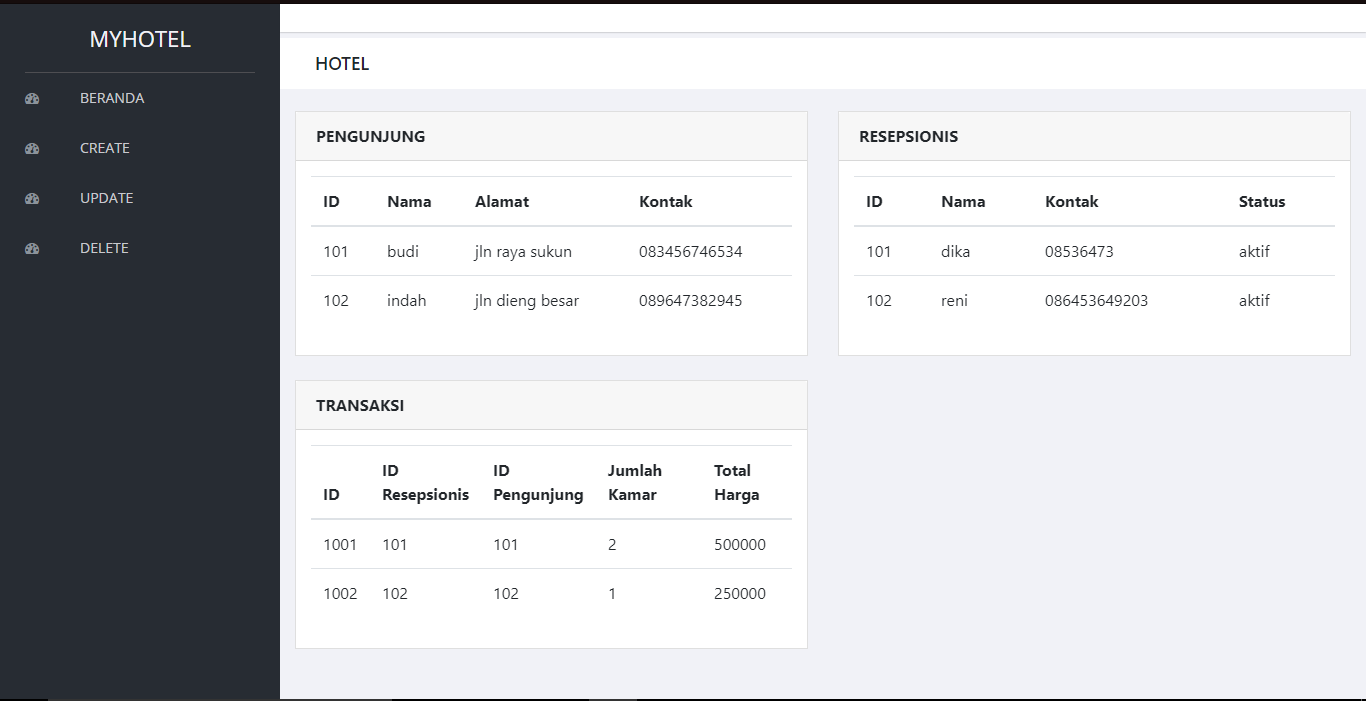
1. **Setelah semua telah selesai buat code PDO untuk memanggil atau mengambil datanya di dalam database yang telah di buat**



1. **Terakhir untuk menampilkan datanya buat file baru di dalam folder yang sama seperti index.php dengan nama yang berbeda lalu masukan kode nya**



1. **Hasil tampilan dari kode di atas bisa di buka dengan memalui browser**



* 1. **Pengerjaan proses CRUD**

1. **Halaman Create Data Transaksi**

@extends('main')

@section('title','HOTEL')

@section('breadcrumbs')

<div class="breadcrumbs">

    <div class="col-sm-4">

        <div class="page-header float-left">

            <div class="page-title">

                <h1>TAMBAH</h1>

            </div>

        </div>

    </div>

</div>

<div class="tambah">

    <form action="{{route('create\_resepsionis')}}" method="POST">

        {{ csrf\_field() }}

      <label for="country">Nama Resepsionis</label>

      <select id="country" name="id\_resepsionis">

        @foreach ($resepsionis as $rsp)

          <option value="{{$rsp->id\_resepsionis}}">{{$rsp->nma\_resepsionis}}</option>

        @endforeach

      </select>

      <label for="country">Nama Pengunjung</label>

      <select id="country" name="id\_pengunjung">

        @foreach ($pengunjung as $peng)

          <option value="{{$peng->id\_pengunjung}}">{{$peng->nma\_pengunjung}}</option>

        @endforeach

      </select>

      <label for="lname">Jumlah Kamar</label>

      <input type="number" id="lname" name="jumlah\_kamar" placeholder="Jumlah Kamar...">

      <label for="lname">Total Harga</label>

      <input type="number" id="lname" name="total\_harga" placeholder="Total Harga...">

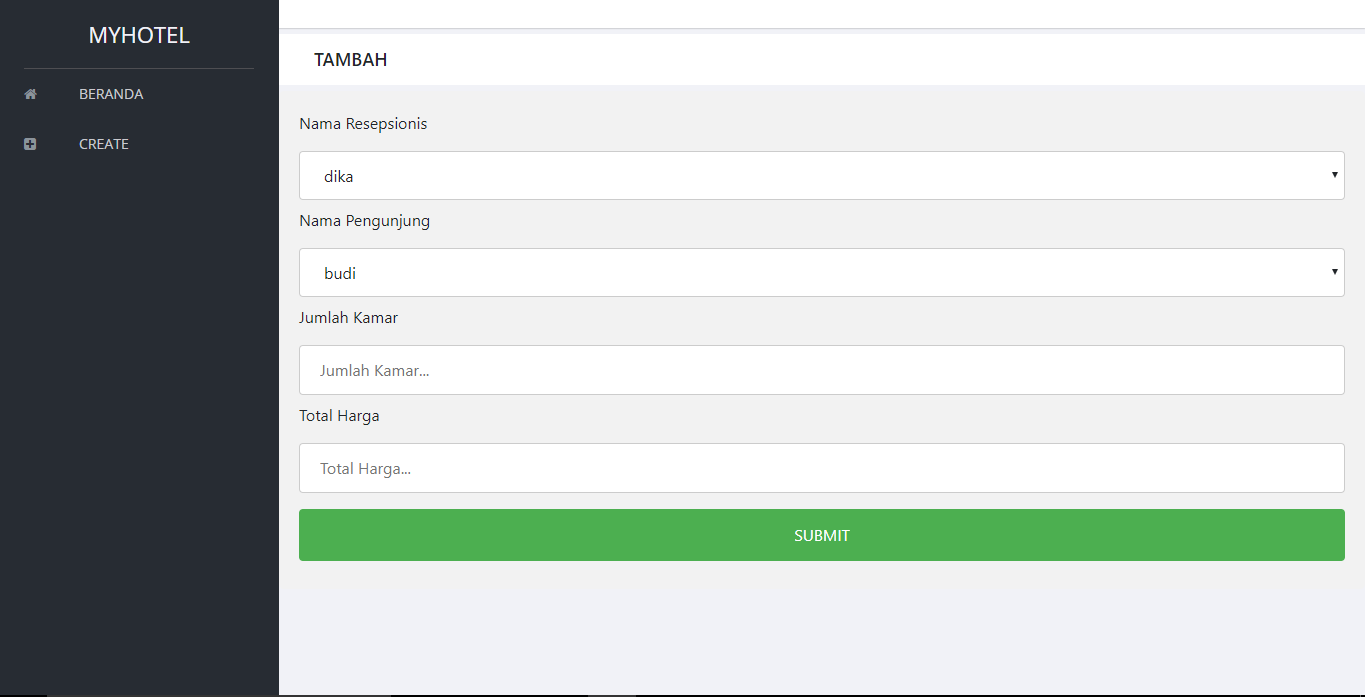
      <input type="submit" value="SUBMIT">

    </form>

  </div>

@endsection

**HASIL :**



**PENJELASAN :**

Pada kode diatas terdapat kode pemanggilan main content untuk sidebar yang ada di sebelah pinggir kiri yang berfungsi sebagai pilihan apa yang akan kita lakukan dengan kode .

@extends('main')

Pada div class breadcrumbs untuk memanggil section breadcrumbs dengan teks bertuliskan TAMBAH pada atas form. Lalu setelah itu terdapat kode untuk membuat table form untuk menambah data yang di inginkan dengan menggunakan kode untuk memanggil dari model pada controller, dan untuk isi di dalam form nya memanggil data di dalam table database yang telah dibuat menggunakan variabel yang telah di buat pada controller dengan code.

<form action="{{route('create\_resepsionis')}}" method="POST">

        {{ csrf\_field() }}

      <label for="country">Nama Resepsionis</label>

      <select id="country" name="id\_resepsionis">

        @foreach ($resepsionis as $rsp)

          <option value="{{$rsp->id\_resepsionis}}">{{$rsp->nma\_resepsionis}}</option>

        @endforeach

      </select>

Begitu juga dengan form nama pengunjung tetapi menggunakan pemanggilan variabel dan kolom table yang berbeda, selanjutnya untuk jumlah kamar dan harga total hanya memanggil data dalam table database yang sudah ada dengan code sebagai berikut tanpa adanya variabel.

<label for="lname">Jumlah Kamar</label>

      <input type="number" id="lname" name="jumlah\_kamar" placeholder="Jumlah Kamar...">

1. **Penggunaan Model dengan nama Transaksi**

<?php

namespace App;

use Illuminate\Database\Eloquent\*Model*;

*class* Transaksi extends *Model*

{

    protected $table = 'transaksi';

    protected $primaryKey = 'id\_transaksi';

    public $timestamps = false;

    public *function* resepsionis() {

        return $this->belongsTo('App\Resepsionis', 'id\_resepsionis');

    }

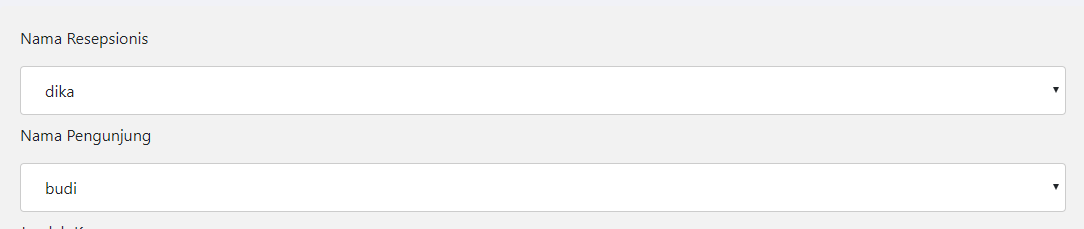
    public *function* pengunjung() {

        return $this->belongsTo('App\Pengunjung', 'id\_pengunjung');

    }

}

**HASIL :**



**PENJELASAN :**

Pada kode diatas untuk membuat variabel yang fungsinya untuk memanggil data yang ada pada database kita tetapi dengan menggunakan kata sesuai kemauan lalu melakukan pemanggilan data seperti umumnya pada laravel seperti ini.

 protected $table = 'transaksi';

    protected $primaryKey = 'id\_transaksi';

    public $timestamps = false;

Setelah itu membuat function untuk memanggil data yang akan di edit untuk di tampilkan ke halaman edit atau update data dengan code.

 public *function* resepsionis() {

        return $this->belongsTo('App\Resepsionis', 'id\_resepsionis');

    }

Begitu juga untuk id\_pengunjung .

1. **Model Resepsionis**

    <?php

    namespace App;

    use Illuminate\Database\Eloquent\*Model*;

    use *DB*;

*class* Resepsionis extends *Model*

    {

        protected $table = 'resepsionis';

        protected $primaryKey = 'id\_resepsionis';

        public *function* getPengunjung()

        {

            return *DB*::table('pengunjung')->get();

        }

        public *function* getResepsionis()

        {

            return *DB*::table('resepsionis')->get();

        }

        public *function* insertTransaksi($data = array())

        {

            return *DB*::table('transaksi')->insert($data);

        }

        public *function* delete\_data($id)

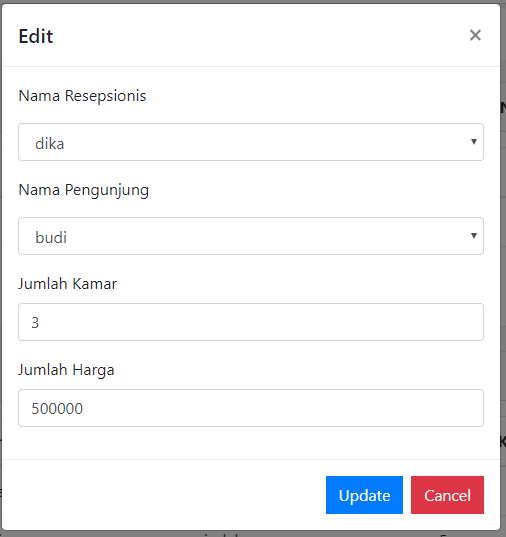
        {

            return *DB*::table('transaksi')->where('id\_transaksi',$id)->delete();

        }

    }

**HASIL :**



**PENJELASAN :**

Pada Kode diatas model berfungsi untuk pemanggilan data di dalam database yang telah dibuat menggunakan function yang berbeda sesuai dengan fungsinya.

protected $table = 'resepsionis';

    protected $primaryKey = 'id\_resepsionis';

menggunakan pemanggilan variabel yang sama seperti sebelumnya untuk mengambil data kemudian menggunakan function GET untuk meng return data dari tabel .

public *function* getPengunjung()

    {

        return *DB*::table('pengunjung')->get();

    }

Untuk insert data menggunkana function sebagai berikut dengan mengambil variabel array dan function array() ke dalam tabel tersebut.

public *function* insertTransaksi($data = array())

    {

        return *DB*::table('transaksi')->insert($data);

    }

1. **Model Hotel**

<?php

namespace App;

use Illuminate\Database\Eloquent\*Model*;

use Illuminate\Support\Facades\*DB*;

use *PDO*;

*class* Hotel extends *Model*

{

    public *function* pengunjung(){

        $pdo=*DB*::getPdo();

        $stmt=$pdo->prepare("select \* from pengunjung");

        $stmt->execute();

        return $stmt->fetchAll(*PDO*::FETCH\_OBJ); //$key[''] $key->

    }

    public *function* resepsionis(){

        $pdo=*DB*::getPdo();

        $stmt=$pdo->prepare("select \* from resepsionis");

        $stmt->execute();

        return $stmt->fetchAll(*PDO*::FETCH\_OBJ);

    }

    public *function* transaksi(){

        $pdo=*DB*::getPdo();

        $stmt=$pdo->prepare("select a.\*, b.nma\_resepsionis, c.nma\_pengunjung from transaksi a JOIN resepsionis b ON b.id\_resepsionis = a.id\_resepsionis JOIN pengunjung c ON c.id\_pengunjung = a.id\_pengunjung");

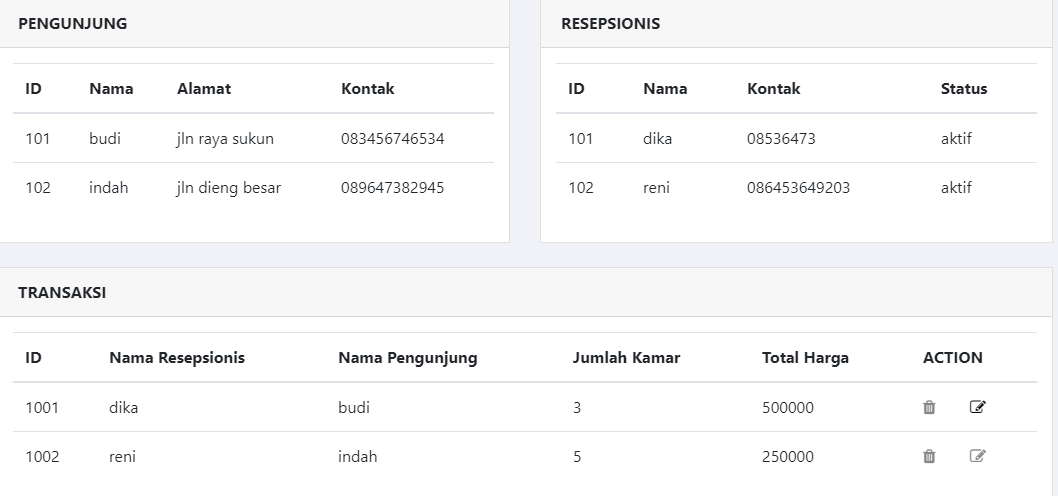
        $stmt->execute();

        return $stmt->fetchAll(*PDO*::FETCH\_OBJ);

    }

}

**HASIL :**



**PENJELASAN :**

Pada kode diatas untuk membuat function yang berfungsi memanggil data dan kemudian melakukan JOIN tabel untuk mendapatkan nama dari tabe lain nya, untuk function pengunjung itu berfungsi sebagai pemanggilan data tabel pengunjung dengan kode.

public *function* pengunjung(){

        $pdo=*DB*::getPdo();

        $stmt=$pdo->prepare("select \* from pengunjung");

        $stmt->execute();

        return $stmt->fetchAll(*PDO*::FETCH\_OBJ); //$key[''] $key->

    }

Membuat variabel sesuai keinginan lalu memanggil dengan PDO ke dalam table database yang ada, dan untuk table transaksi karna terdapat JOIN table sedikit berbeda dengan kode function pengunjung.

public *function* transaksi(){

        $pdo=*DB*::getPdo();

        $stmt=$pdo->prepare("select a.\*, b.nma\_resepsionis, c.nma\_pengunjung from transaksi a JOIN resepsionis b ON b.id\_resepsionis = a.id\_resepsionis JOIN pengunjung c ON c.id\_pengunjung = a.id\_pengunjung");

        $stmt->execute();

        return $stmt->fetchAll(*PDO*::FETCH\_OBJ);

    }

Di dalam prepare memasukan kode JOIN table seperti di atas dengan pemanggilan ID masing-masing table lalu di ubah menjadi nama dalam tampilan di dasboardnya.

1. **EDIT page**

<div class="modal-dialog modal-md">

    <div class="modal-content">

        <form id="frmEdit" action="{{ url()->current() . '/process' }}" method="POST" autocomplete="off">

            {{ csrf\_field() }}

            <input type="hidden" name="id\_transaksi" value="{{ $transaksi->id\_transaksi }}">

            <div class="modal-header">

                <h5 class="modal-title">Edit</h5>

                <button type="button" class="close" data-dismiss="modal" aria-label="Close">

                    <span aria-hidden="true">&times;</span>

                </button>

            </div>

            <div class="modal-body">

                <div class="form-group">

                    <label>Nama Resepsionis</label>

                    <select name="id\_resepsionis" class="form-control">

                        @foreach($resepsionis as $key => $value)

                        <option value="{{ $value->id\_resepsionis }}" {{ $value->id\_resepsionis == $transaksi->id\_resepsionis ? 'selected' : null }}>{{ $value->nma\_resepsionis }}</option>

                        @endforeach

                    </select>

                </div>

                <div class="form-group">

                    <label>Nama Pengunjung</label>

                    <select name="id\_pengunjung" class="form-control">

                        @foreach($pengunjung as $key => $value)

                        <option value="{{ $value->id\_pengunjung }}" {{ $value->id\_pengunjung == $transaksi->id\_pengunjung ? 'selected' : null }}>{{ $value->nma\_pengunjung }}</option>

                        @endforeach

                    </select>

                </div>

                <div class="form-group">

                    <label>Jumlah Kamar</label>

                    <input type="text" name="jml\_kmr" class="form-control" value="{{ $transaksi->jumlah\_kamar }}">

                </div>

                <div class="form-group">

                    <label>Jumlah Harga</label>

                    <input type="text" name="jml\_hrg" class="form-control" value="{{ $transaksi->total\_harga }}">

                </div>

            </div>

            <div class="modal-footer">

                <button type="submit" class="btn btn-primary">Update</button>

                <button type="button" class="btn btn-danger" data-dismiss="modal">Cancel</button>

            </div>

        </form>

    </div>

</div>

<script>

    $('#frmEdit').submit(*function*(*e*){

        e.preventDefault();

*let* formData = new FormData(this);

        $.ajax({

            url: $(this).attr('action'),

            method: $(this).attr('method'),

            dataType: 'json',

            data: formData,

            processData: false,

            contentType: false,

            success: *function*(*response*){

                if(response.RESULT == 'OK'){

                    alert(response.MESSAGE);

                    setTimeout(*function*(){

                        window.location.reload();

                    }, 1000);

                }else{

                    alert(response.MESSAGE);

                }

            }

        }).fail(*function*(){

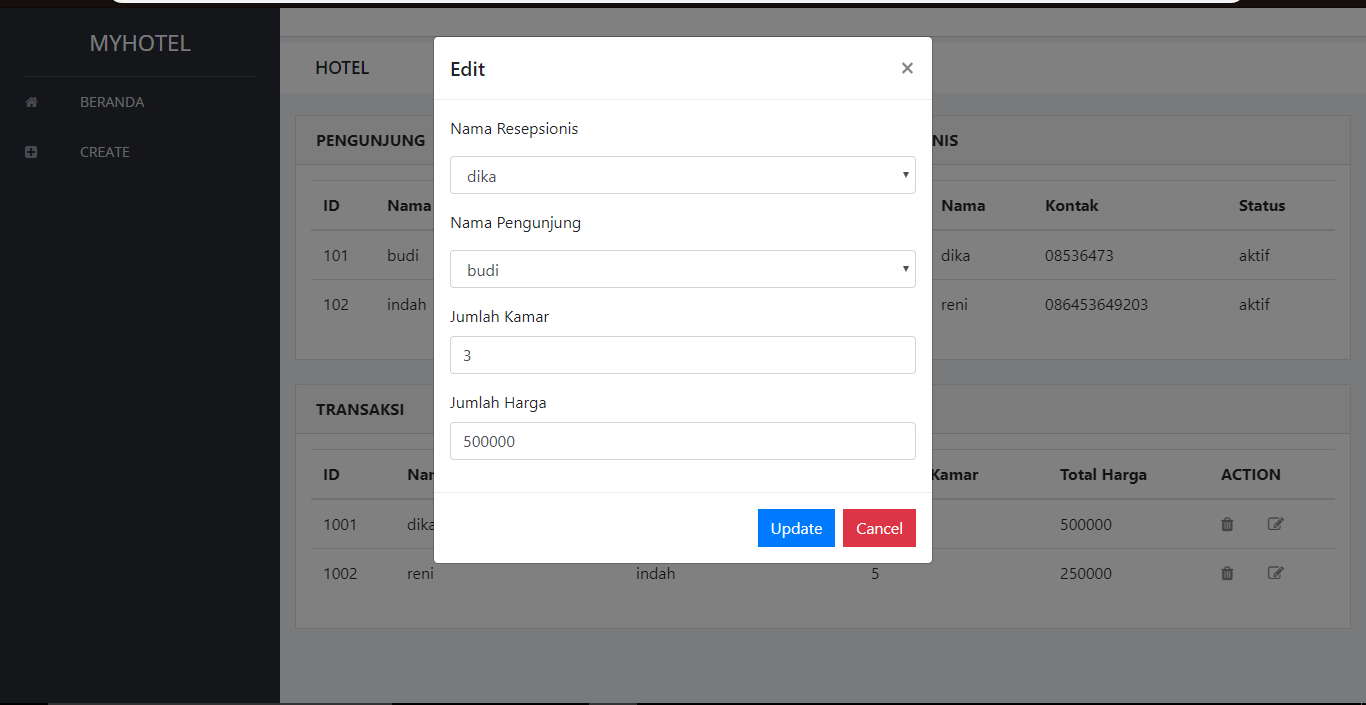
            alert('Error');

        });

    });

</script>

**HASIL :**



**PENJELASAN :**

Pada kode diatas menggunakan MODEl yang terdapat pada bootstrap untuk menampilkan form edit atau update data nya lalu dengan mengisi form menggunakan pemanggilan variabel yang telah dibuat pada model sebelumnya dan nama kolom table pada database .

<input type="hidden" name="id\_transaksi" value="{{ $transaksi->id\_transaksi }}">

<div class="modal-body">

                <div class="form-group">

                    <label>Nama Resepsionis</label>

                    <select name="id\_resepsionis" class="form-control">

                        @foreach($resepsionis as $key => $value)

                        <option value="{{ $value->id\_resepsionis }}" {{ $value->id\_resepsionis == $transaksi->id\_resepsionis ? 'selected' : null }}>{{ $value->nma\_resepsionis }}</option>

                        @endforeach

                    </select>

                </div>

Pada kode ini melakukan pemanggilan variabel pada controller dengan nama $resepsionis, $ key dan lain nya untuk menampilkan data pada table begitu juga dengan lain nya yang berfungsi sebagai update data otomatis pada table database. Dan untuk menjalankan fungsi button menggunakan ajax dan jSon agar mendapat notif ketika melakukan update atau edit data dengan menaruh kode di paling bawah seperti berikut.

<script>

    $('#frmEdit').submit(*function*(*e*){

        e.preventDefault();

*let* formData = new FormData(this);

        $.ajax({

            url: $(this).attr('action'),

            method: $(this).attr('method'),

            dataType: 'json',

            data: formData,

            processData: false,

            contentType: false,

            success: *function*(*response*){

                if(response.RESULT == 'OK'){

                    alert(response.MESSAGE);

                    setTimeout(*function*(){

                        window.location.reload();

                    }, 1000);

                }else{

                    alert(response.MESSAGE);

                }

            }

        }).fail(*function*(){

            alert('Error');

        });

    });

</script>

Pada kode ajax dan Json di atas ketika button edit di klik otomatis akan ajax akan melakukan pengiriman data ke dalam table database dengan menggunkana .attr lalu akan mendapat message menggunakan jSon untuk meyakinkan update data.

1. **DELETE data**

<script>

*function* delete\_data(*id*) {

        if (confirm("Apakah Anda Yakin Ingin Menghapus Data ini?")) {

            $.ajax({

                url:'delete',

                method:'GET',

                data:{id:id},

            }).done(*function* (*data*) {

                if (data == 'success') {

                    location.reload();

                } else {

                    alert("gagal");

                }

            })

        }

    }

*function* edit\_data(*id*){

        $.ajax({

            url: '{{ url("edit") }}',

            method: 'POST',

            dataType: 'json',

            data: {

                \_token: '{{ csrf\_token() }}',

                id: id

            },

            success: *function*(*response*){

                if(response.RESULT == 'OK'){

                    $('#ModalGlobal').html(response.CONTENT);

                }else{

                    alert(response.MESSAGE);

                }

            }

        }).fail(*function*(){

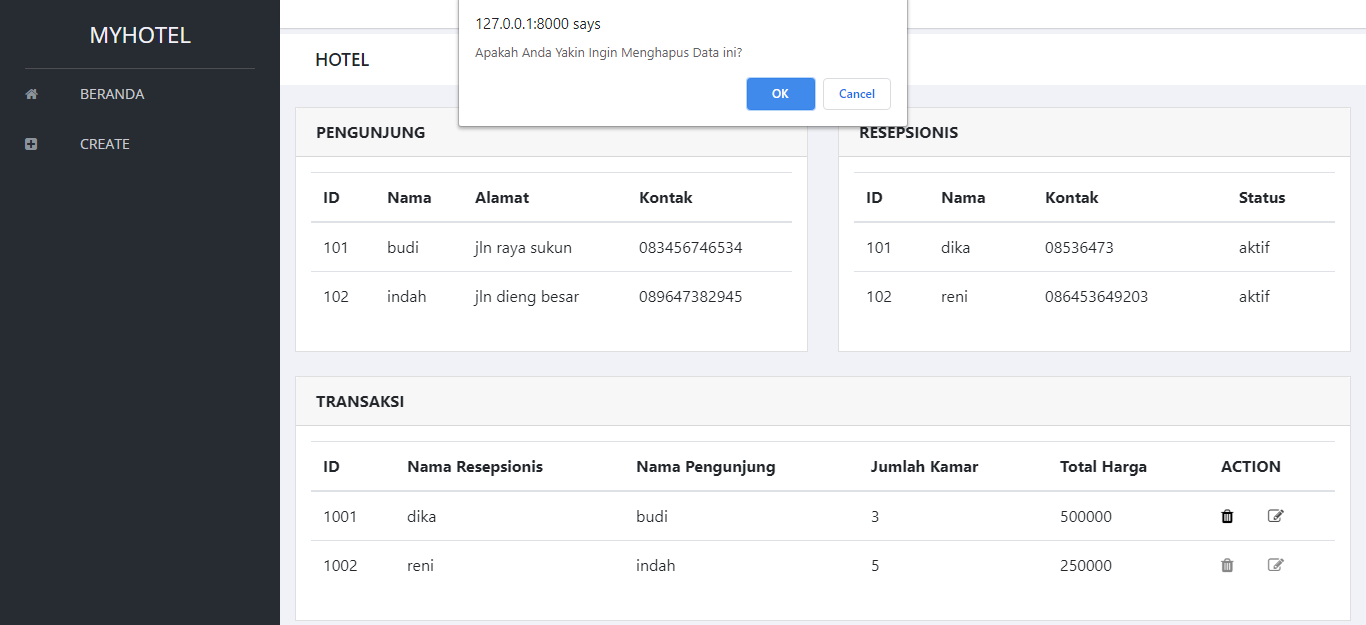
            alert('Error');

        });

    }

</script>

**HASIL :**



**PENJELASAN :**

Pada kode diatas menggunakan ajax untuk notif message

*function* delete\_data(*id*) {

        if (confirm("Apakah Anda Yakin Ingin Menghapus Data ini?")) {

            $.ajax({

                url:'delete',

                method:'GET',

                data:{id:id},

            }).done(*function* (*data*) {

                if (data == 'success') {

                    location.reload();

                } else {

                    alert("gagal");

                }

            })

        }

    }

Kode diatas digunakan untuk mendapat notif message jika kita ingin menghapus data yang ada di dalam dashboard menggunakan url delet dengan method GET untuk mendapatkan data pada database lalu jika data sukses dihapus akan reload ke page dashboard namun jika data gagal akan muncul alert atau peringatan GAGAL .

1. **ROUTE data pada web**

<?php

use Illuminate\Support\Facades\*Route*;

use App\*Hotel*;

/\*

|--------------------------------------------------------------------------

| Web Routes

|--------------------------------------------------------------------------

|

| Here is where you can register web routes for your application. These

| routes are loaded by the RouteServiceProvider within a group which

| contains the "web" middleware group. Now create something great!

|

\*/

*Route*::get('/', *function* () {

    return view('welcome', ['title' => 'PROJECT LARAVEL 7 - KELOMPOK 3']);

});

*Route*::get('create','ResepsionisController@index');

*Route*::get('home', *function* () {

    $hotel=new *Hotel*();

    $data = array();

    $data['hotel'] = $hotel->pengunjung();

    $data['resepsionis'] = $hotel->resepsionis();

    $data['transaksi'] = $hotel->transaksi();

    return view('home',$data);

});

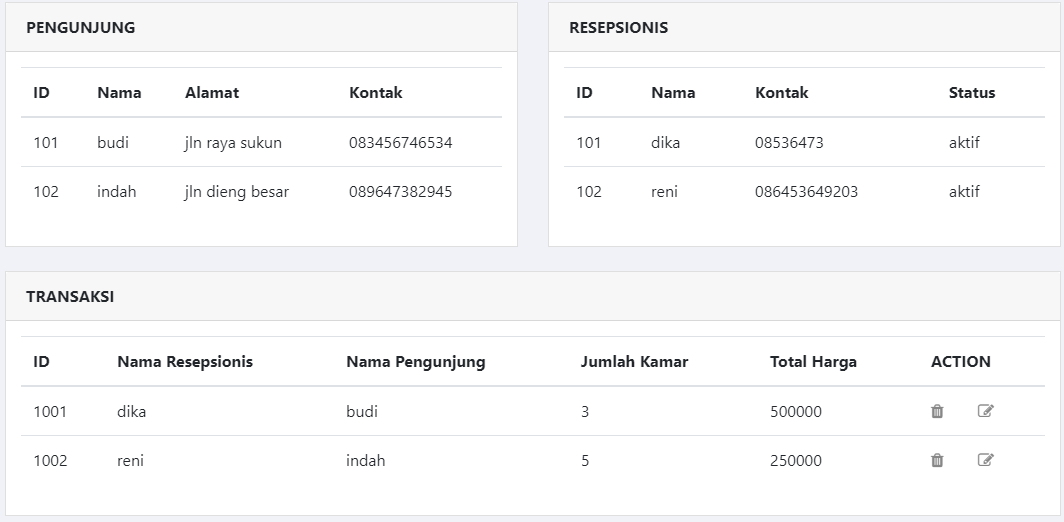
*Route*::post('create-resepsionis','ResepsionisController@create\_resepsionis')->name('create\_resepsionis');

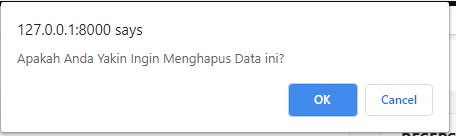
*Route*::get('delete','ResepsionisController@delete');

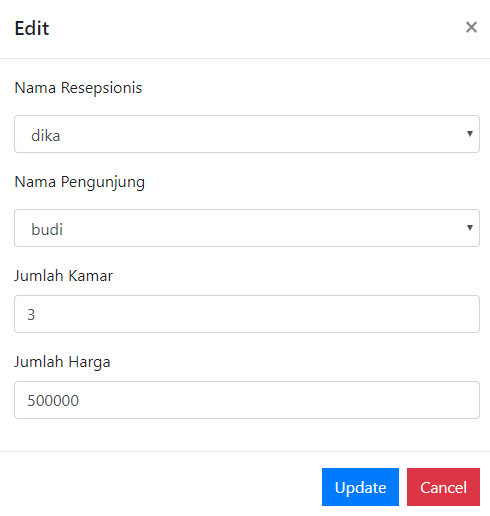
*Route*::post('edit', 'ResepsionisController@modal\_edit');

*Route*::post('edit/process', 'ResepsionisController@process\_edit');

**HASIL :**







**PENJELASAN :**

Pada Kode di atas route berfungsi untuk menentukan kemana data akan di bawa dan kemana data akan masukan ke dalam kode lain nya datau ke dalam tampilan dashboard.

*Route*::get('home', *function* () {

    $hotel=new *Hotel*();

    $data = array();

    $data['hotel'] = $hotel->pengunjung();

    $data['resepsionis'] = $hotel->resepsionis();

    $data['transaksi'] = $hotel->transaksi();

    return view('home',$data);

});

Kode ini berfungsi untuk membawa data ke dalam page dashboard dengan mengambil di dalam database yang telah dibuat dengan menggunakan variabel lalu memanggil kolom table dan function.

*Route*::post('create-resepsionis','ResepsionisController@create\_resepsionis')->name('create\_resepsionis');

*Route*::get('delete','ResepsionisController@delete');

*Route*::post('edit', 'ResepsionisController@modal\_edit');

*Route*::post('edit/process', 'ResepsionisController@process\_edit');

Lalu route ini berfungsi untuk melakukan GET dan POST data untuk di bawa ke dalam fungsi button atau kode delete, create dan edit sebelumnya dengan menngunakan route laravel pada umumnya.

1. **CONTROLLER utama**

<?php

namespace App\Http\Controllers;

use Illuminate\Http\*Request*;

use App\*Resepsionis*;

use App\*Transaksi*;

use App\*Pengunjung*;

*class* ResepsionisController extends *Controller*

{

    public *function* index()

    {

        $rsp=new *Resepsionis*();

        $pengunjung = $rsp->getPengunjung();

        $resepsionis = $rsp->getResepsionis();

        return view('create', [

            'title' => 'PROJECT LARAVEL 7 - ZANUAR',

            'pengunjung' => $pengunjung,

            'resepsionis' => $resepsionis

            ]);

    }

    public *function* create\_resepsionis(*Request* $request)

    {

        $rsp=new *Resepsionis*();

        $data=array();

        $data['id\_resepsionis']=$request->input('id\_resepsionis');

        $data['id\_pengunjung']=$request->input('id\_pengunjung');

        $data['jumlah\_kamar']=$request->input('jumlah\_kamar');

        $data['total\_harga']=$request->input('total\_harga');

        $proses = $rsp->insertTransaksi($data);

        if ($proses) {

            return redirect('home');

        } else {

            return "alert('Error')";

        }

    }

    public *function* delete(*Request* $request)

    {

        $rsp=new *Resepsionis*();

        $pros = $rsp->delete\_data($request->get('id'));

        if ($pros < 0) {

            return 'error';

        } else {

            return 'success';

        }

    }

    public *function* modal\_edit(*Request* $request){

        $id\_transaksi = $request->post('id');

        $transaksi = *Transaksi*::find($id\_transaksi);

        $resepsionis = *Resepsionis*::all();

        $pengunjung = *Pengunjung*::all();

        if($transaksi == null){

            echo json\_encode(array(

                'RESULT' => 'FAILED',

                'MESSAGE' => 'Data tidak ditemukan'

            ));

            exit;

        }

        $data = array();

        $data['transaksi'] = $transaksi;

        $data['resepsionis'] = $resepsionis;

        $data['pengunjung'] = $pengunjung;

        echo json\_encode(array(

            'RESULT' => 'OK',

            'CONTENT' => view('edit', $data)->render()

        ));

        exit;

    }

    public *function* process\_edit(*Request* $request){

        $id\_transaksi = $request->post('id\_transaksi');

        $nama\_rsp = $request->post('id\_resepsionis');

        $nama\_pnj = $request->post('id\_pengunjung');

        $jml\_kmr = $request->post('jml\_kmr');

        $jml\_hrg = $request->post('jml\_hrg');

        $transaksi = *Transaksi*::find($id\_transaksi);

        if($transaksi == null){

            echo json\_encode(array(

                'RESULT' => 'FAILED',

                'MESSAGE' => 'Data tidak ditemukan'

            ));

            exit;

        }

        $transaksi->id\_resepsionis = $nama\_rsp;

        $transaksi->id\_pengunjung = $nama\_pnj;

        $transaksi->jumlah\_kamar = $jml\_kmr;

        $transaksi->total\_harga = $jml\_hrg;

        $update = $transaksi->save();

        if($update){

            echo json\_encode(array(

                'RESULT' => 'OK',

                'MESSAGE' => 'Sukses update'

            ));

            exit;

        }else{

            echo json\_encode(array(

                'RESULT' => 'FAILED',

                'MESSAGE' => 'Gagal update'

            ));

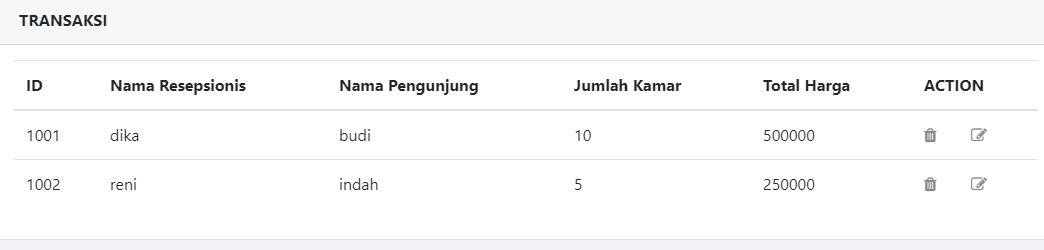
            exit;

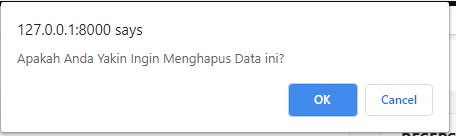
        }

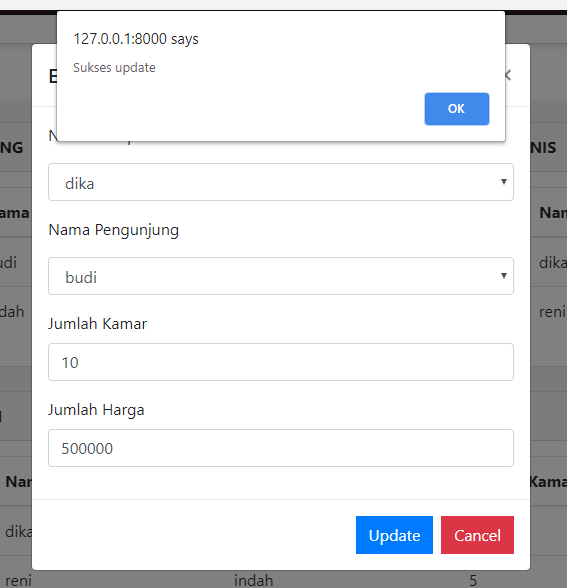
    }

}

**HASIL :**







**PENJELASAN :**

Pada Kode diatasberfungsu sebagai controller atau mengatur jalannya data untuk di panggil atau di tampilkan ke dalam kode dan page.

public *function* index()

    {

        $rsp=new *Resepsionis*();

        $pengunjung = $rsp->getPengunjung();

        $resepsionis = $rsp->getResepsionis();

        return view('create', [

            'title' => 'PROJECT LARAVEL 7 - ZANUAR',

            'pengunjung' => $pengunjung,

            'resepsionis' => $resepsionis

            ]);

    }

function index disini adalah sebagai menampilkan data table datri database ke dalam page dashboard dan untuk menampilkan bagian awal atau terdepan dengan menulis variabel lalu get nama table dan kemudian untuk kembali ke halaman awal dengan return view untuk kode title hanya untuk menjadi judul awal page atau home page.

public *function* create\_resepsionis(*Request* $request)

    {

        $rsp=new *Resepsionis*();

        $data=array();

        $data['id\_resepsionis']=$request->input('id\_resepsionis');

        $data['id\_pengunjung']=$request->input('id\_pengunjung');

        $data['jumlah\_kamar']=$request->input('jumlah\_kamar');

        $data['total\_harga']=$request->input('total\_harga');

        $proses = $rsp->insertTransaksi($data);

        if ($proses) {

            return redirect('home');

        } else {

            return "alert('Error')";

        }

    }

Function ini berfungsi untuk membuat create data pada page sebelumnya dengan mengambil atau membuat variabel dan kemudian input di dalam kolom table database lalu insert ke dalam table transaksi jika selesai akan redirect ke home dan jika gagala tau error akan tetap pada halaman tersebut.

public *function* delete(*Request* $request)

    {

        $rsp=new *Resepsionis*();

        $pros = $rsp->delete\_data($request->get('id'));

        if ($pros < 0) {

            return 'error';

        } else {

            return 'success';

        }

    }

Function tersebut berfungsi sebagai delete data pada tabel transaksi dengan function delete lalu menggunakan variabel yang dibuat dengan perintah delete\_data lalu menagambil id dengan GET.

public *function* modal\_edit(*Request* $request){

        $id\_transaksi = $request->post('id');

        $transaksi = *Transaksi*::find($id\_transaksi);

        $resepsionis = *Resepsionis*::all();

        $pengunjung = *Pengunjung*::all();

        if($transaksi == null){

            echo json\_encode(array(

                'RESULT' => 'FAILED',

                'MESSAGE' => 'Data tidak ditemukan'

            ));

            exit;

        }

        $data = array();

        $data['transaksi'] = $transaksi;

        $data['resepsionis'] = $resepsionis;

        $data['pengunjung'] = $pengunjung;

        echo json\_encode(array(

            'RESULT' => 'OK',

            'CONTENT' => view('edit', $data)->render()

        ));

        exit;

    }

Function di atas untuk menampilkan data di dalam modal bootstrap sebelumnya dengan membuat variabel lalu mengambil data pada table transaksi kemudian di json ketika gagal akan muncul message failed dan jika berhasil akan langsung ke page dashboard.

public *function* process\_edit(*Request* $request){

        $id\_transaksi = $request->post('id\_transaksi');

        $nama\_rsp = $request->post('id\_resepsionis');

        $nama\_pnj = $request->post('id\_pengunjung');

        $jml\_kmr = $request->post('jml\_kmr');

        $jml\_hrg = $request->post('jml\_hrg');

        $transaksi = *Transaksi*::find($id\_transaksi);

        if($transaksi == null){

            echo json\_encode(array(

                'RESULT' => 'FAILED',

                'MESSAGE' => 'Data tidak ditemukan'

            ));

            exit;

        }

        $transaksi->id\_resepsionis = $nama\_rsp;

        $transaksi->id\_pengunjung = $nama\_pnj;

        $transaksi->jumlah\_kamar = $jml\_kmr;

        $transaksi->total\_harga = $jml\_hrg;

        $update = $transaksi->save();

        if($update){

            echo json\_encode(array(

                'RESULT' => 'OK',

                'MESSAGE' => 'Sukses update'

            ));

            exit;

        }else{

            echo json\_encode(array(

                'RESULT' => 'FAILED',

                'MESSAGE' => 'Gagal update'

            ));

            exit;

        }

    }

Function diatas untuk melakukan proses edit dengan membuat variabel lalu mengambil data pada kolom table yang akan di ambil lalu mengganti dengan variabel yang diinginkan jika tidak ditemukan akan muncul notif failed atau data tidak ditemukan dan jika berhasil akan langsung muncul sukses update.

**PEMBAGIAN TUGAS**

**-** Rahmad Nakula Z

* pengerjaan tugas 2,pembuatan aplikasi hotel dan database

**-** Felix Vannial

* pengerjaan tugas 1,pembuatan desain dan lapoan instalasi laavel

**-** Akmal

* pengerjaan tugas 3,penyempurnaan aplikasi dan laporan

**-** Salsabilla

* pengerjaan tugas 3,penyempurnaan aplikasi dan laporan

**HASIL DISKUSI**

Sistem penyewaan kamar hotel menggunakan database untuk menyimpan data-data pemesanan kamar secara komputerisasi sehingga kemungkinan terjadinya terulangnya data yang sama dalam sebuah database sangat kecil Dan juga dapat memudahkan pegawai dalam hal memesukan data penginap dari manual ke komputerisasi sehingga pekerjaan yang sebelumnya lebih efisien dan terdata dengan baik, memudahkan pegawai dalam mengatur kemejemenan perhotelan. Rancang bangun aplikasi penyewaan kamar hotel belum maksimal sehingga kedepannya dapat dikembangan lagi dengan berbasis web. Dan aplikasi ini menghubungakan dari pihak hotel ke ruang publik.