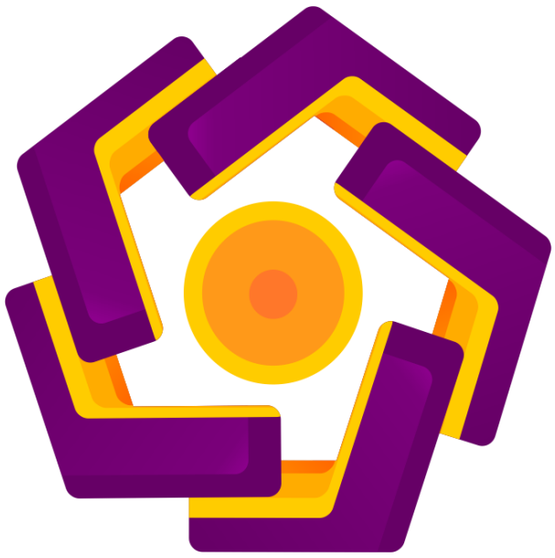
LAPORAN PRAKTIKUM
DEVELOPMENT FRAMEWORK

"QUERY BUILDER-SEEDER-FAKER-MIGRATION"



Disusun Oleh:

Nama	Kholid Anas Amrulloh
Nim	18.01.4207
Kelas	**D3 Teknik Informatika**
Nama Dosen	Arvin Claudy Frobenius, M.Kom

D3 – TEKNIK INFORMATIKA
FAKULTAS ILMU KOMPUTER
UNIVERSITAS AMIKOM
YOGYAKARTA
2020

30 Maret 2020	**LAPORAN PRAKTIKUM DEFELOPMENT FRAMEWORK**	05 April 2020
Praktikum ke-5		Laporan ke 4

A. Tujuan Praktikum

1. Mahasiswa dapat mengenal dan menggunakan query builder
2. Mahasiswa dapat mengenal dan membuat seeder & faker
3. Mahasiswa dapat mengenal dan membuat migration

B. Teori Singkat

1. **Query Builder**
   Adalah sebuah antar muka yang digunakan untuk mengenerasikan dan menjalankan query SQL. Query builder dapat digunakan untuk membuat aplikasi dengan tidak bergantung pada database tertentu.

2. **CRUD**
   Adalah sebuah singkatan dari Create, Read, Update, Delete yang sering digunakan pada aplikasi pengolahan data yang banyak menggunakan fungsi CRUD ini. Dengan Controller maka dapat menambahkan data, membaca data, menghapus data, dan juga mengupdate data.

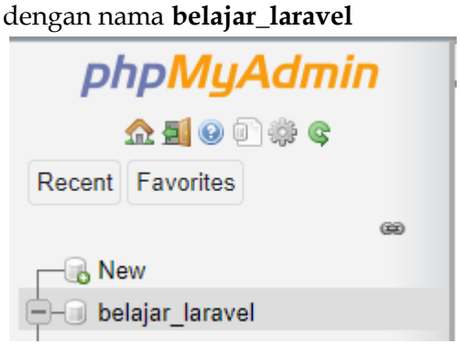
3. **Seeder & faker**
   Seeder adalah sebuat fitur dari framework Laravel untuk mengisi data pada database. Karena semakin berkembang suatu jaman & teknologi maka akan menemukan berbagai cara untuk mempermudah. Dengan seeding Laravel, maka menginput data akan lebih cepat. Sedangkan faker adalah sebuah library yang berguna untuk membuat data palsu (dummy).

4. **Migration**
   Migration adalah sebuah pengkontrolan daripada database yang akan kita buat. Yaitu dapat untuk membuat, mengubah, dan membagi skema database beserta tabelnya tanpa secara langsung dilakukan ke database yang sebenarnya.

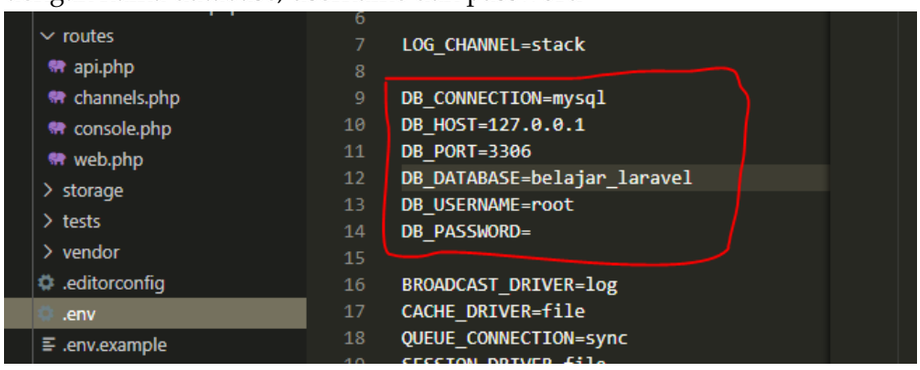
C. Hasil Praktikum

A. QUERY BUILDER

1. **Mengatur database pada laravel**
   Berikut langkah-langkah nya :
   1. Pertama, masuk ke phpMyAdmin, lalu membuat database baru pada mysql dengan nama **belajar_laravel**



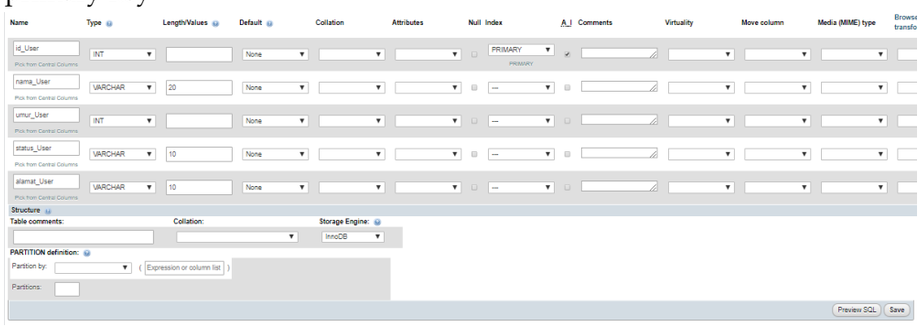
2. Selanjtnya buka file .env yang ada di dalam folder laravel yaitu 4207, sesuaikan dengan nama database, username dan password



2. Membuat database pada mysql user

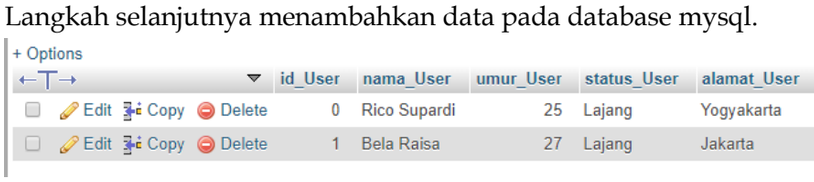
Berikut langkah-langkahnya:

1. Membuat database yang ssudah di seuaikan tadi di .env. Nama tabelnya adalah **user** dengan 5 kolom di dalamnya. Dan tetapkan id_User sebagai primary key



Selanjutnya save

2. Langkah selanjutnya menambahkan data pada database mysql.



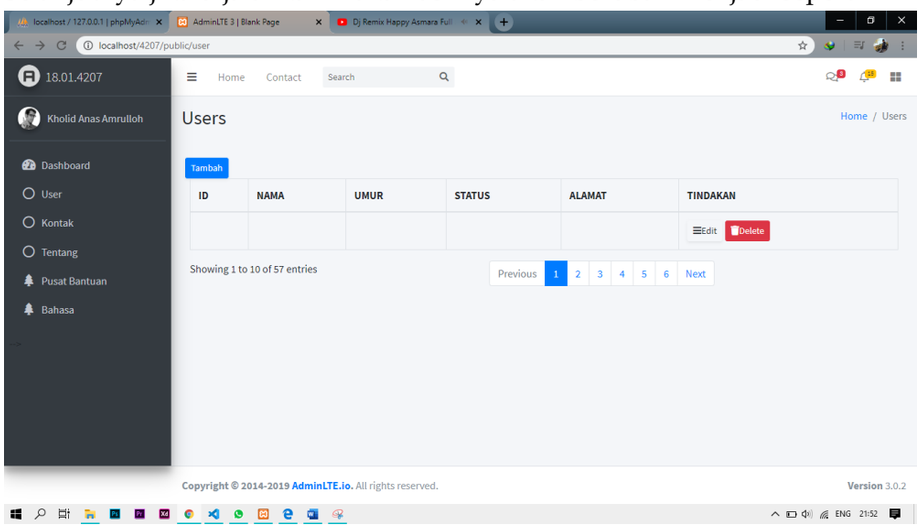
3. Menampilkan data pada menu user

Berikut langkah-langkahnya:

1. Mengubah sintak pada tamplate user sebelumnya menjadi seperti ini

```
                <div class="col-sm-12">
                    <table class="table table-bordered table-hover dataTable" id="example2" role="grid" aria-describedby="example2_info">
                        <thead>
                            <tr role="row">
                                <th tabindex="0" class="sorting_asc" aria-controls="example2" aria-sort="ascending" rowspan="1" colspan="1">ID</t
                                <th tabindex="0" class="sorting" aria-controls="example2" rowspan="1" colspan="1">NAMA</th>
                                <th tabindex="0" class="sorting" aria-controls="example2" rowspan="1" colspan="1">UMUR</th>
                                <th tabindex="0" class="sorting" aria-controls="example2" rowspan="1" colspan="1">STATUS</th>
                                <th tabindex="0" class="sorting" aria-controls="example2" rowspan="1" colspan="1">ALAMAT</th>
                                <th tabindex="0" class="sorting" aria-controls="example2" rowspan="1" colspan="1">TINDAKAN</th>
                            </tr>
                        </thead>
                        <tbody>
                            <tr class="odd" role="row">
                                <td></td>
                                <td></td>
                                <td></td>
                                <td></td>
                                <td></td>
                                <td>
                                    <a href="#" class="open_modal_ubah btn btn-sm btn_primary shadow-sm"><i class="fa fa-bars"></i>Edit</a>
                                    <a href="#" class="btn btn-danger btn-sm delete-link"><i class="fa fa-trash"></i>Delete</a>
                                </td>
                            </tr>
                        </tbody>
                    </table>
                </div>
            </div>
            <div class="row">
                <div class="col-sm-12 col-md-5">
                    <div class="dataTables_info" id="example2_info" role="status" aria-live="polite">Showing 1 to 10 of 57 entries</div>
                </div>
                <div class="col-sm-12 col-md-7">
                    <div class="dataTables_paginate paging_simple_numbers" id="example2_paginate">
                        <ul class="pagination">
                            <li class="paginate_button page-item previous disabled" id="example2_previous"><a href="#" tabindex="0" class="page-l
                            <li class="paginate_button page-item active"><a tabindex="0" class="page-link" aria-controls="example2" href="#" data
                            <li class="paginate_button page-item "><a tabindex="0" class="page-link" aria-controls="example2" href="#" data-dt-id
                            <li class="paginate_button page-item "><a tabindex="0" class="page-link" aria-controls="example2" href="#" data-dt-id
                            <li class="paginate_button page-item "><a tabindex="0" class="page-link" aria-controls="example2" href="#" data-dt-id
                            <li class="paginate_button page-item "><a tabindex="0" class="page-link" aria-controls="example2" href="#" data-dt-id
                            <li class="paginate_button page-item "><a tabindex="0" class="page-link" aria-controls="example2" href="#" data-dt-id
                            <li class="paginate_button page-item next" id="example2_next"><a tabindex="0" class="page-link" aria-controls="exampl
                        </ul>
                    </div>
                </div>
            </div>
        </div>
    </div>
    <!-- /.card-body -->

    </div>
    </div>

    <!-- /.card-footer-->
    <!-- /.card -->

</section>
<!-- /.content -->
@endsection
```

2. Selanjutnya jika dijalankan maka hasilnya akan berubah menjadi seperti ini



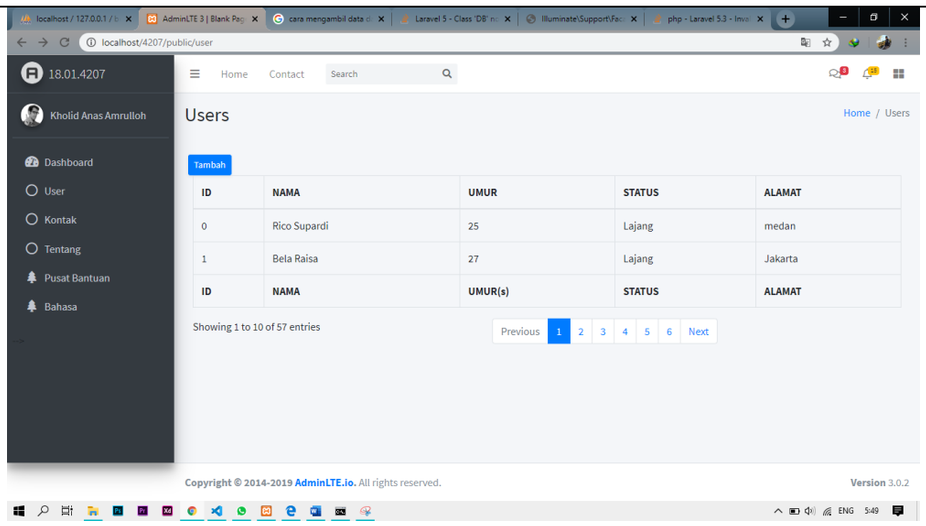
3. Untuk menampilkan data pada database menggunakan query builder maka ditambahkan skrip seperti berikut ini pada controller user

```php
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Suppport\Facades\DB;

class UserController extends Controller
{
    public function index()
    {
        // mengambil data dari tabel user
        $user = DB::table('user')->get();

        // mengirim data user ke view index.blade.php
        return view('index', ['user' => $user]);
    }
}
```

4. Selanjutnya mempassing data dari controller ke view. Menggunakan tamplate tadi, lalu ditambahkan script seperti berikut.

```blade
            </tr>
        </thead>
        <tbody>

            @foreach(@user as @u)
            <tr class="odd" role="row">
                <td>{{ $u->id_User }}</td>
                <td>{{ $u->nama_User }}</td>
                <td>{{ $u->umur_User }}</td>
                <td>{{ $u->status_User }}</td>
                <td>{{ $u->alamat_User }}</td>
            </tr>
            @endforeach
        </tbody>
        <tfoot>
            <tr>
                <th rowspan="1" colspan="1">ID</th>
                <th rowspan="1" colspan="1">NAMA</th>
                <th rowspan="1" colspan="1">UMUR(s)</th>
                <th rowspan="1" colspan="1">STATUS</th>
                <th rowspan="1" colspan="1">ALAMAT</th>
            </tr>
        </tfoot>
    </table>
'div>
```

5. Untuk menjalankannya menggunakan perintah php artisan serve



4. Membuat insert pada user

Berikut langkah-langkahnya:

1. Membuat Route untuk menambahkan data

```
Route::get('/user/tambah', 'UserController@tambah');
```

2. Selanjutnya membuat Controller pada UserController

```
public function tambah()
{
    // memanggil view tambah
    return view('user/tambah');
}
```

3. Lalu membuat view untuk input data dengan nama tambah.blade.php di dalam folder user pada folder views

index.blade.php | UserController.php | tambah.blade.php | web.php

resources > views > user > tambah.blade.php > section.content > div > div.card-body > div#example2_wrapper.dataTables_wrapper.dt-bootstrap4 > div.row

```
@section('content')

<!-- Main content -->
<section class="content">

    <!-- Default box -->
    <div>
        <div class="card-header">
            <h3 class="card-title">Users</h3>
        </div>

        <div class="card-body">
            <div class="dataTables_wrapper dt-bootstrap4" id="example2_wrapper">
                <div class="row">
                    <div class="col-sm-12 col-md-6"></div>
                    <div class="col-sm-12 col-md-6"></div>
                </div>
                <div class="row">
                    <div class="col-sm-12">
                        <a href="/user/tambah" class="open_modal_tambah d-none d-sm-inline-block btn btn-sm btn-primary shadow-sm">Tambah</a>
                        <form action="/user/store" method="post">
                            {{ csrf_field() }}
                            <div class="form-group">
                                <label for="nama">Nama User</label>
                                <input type="text" name="nama" class="form-control" placeholder="Nama User ..." require="required">
                            </div>
                            <div class="form-group">
                                <label for="umur">Umur User</label>
                                <input type="text" name="umur" class="form-control" placeholder="Umur User ..." require="required">
                            </div>
                            <div class="form-group">
                                <label for="status">Status User</label>
                                <input type="text" name="status" class="form-control" placeholder="Status User ..." require="required">
                            </div>
                            <div class="form-group">
                                <label for="Alamat">Alamat User</label>
                                <input type="text" name="alamat" class="form-control" placeholder="Alamat User ..." require="required">
                            </div>
                            <div class="modal-footer">
                                <button class="btn btn-success" type="submit" value="Simpan Data">Confirm</button>
                                <button type="reset" class="btn btn-danger" data-dismiss="modal" aria-hidden="true">Cancel</button>
                            </div>
                        </form>
                    </div>
                </div>
            </div>
            <!-- /.card-body -->
        </div>
        <!-- /.card -->

</section>
<!-- /.content -->
@endsection
```

4. Selanjutnya membuat form action dari user/store. Yang digunakan untuk menyimpan data ke database.

```
<a href="/user/tambah" class="open_modal_tambah d-
<form action="/user/store" method="post">
```

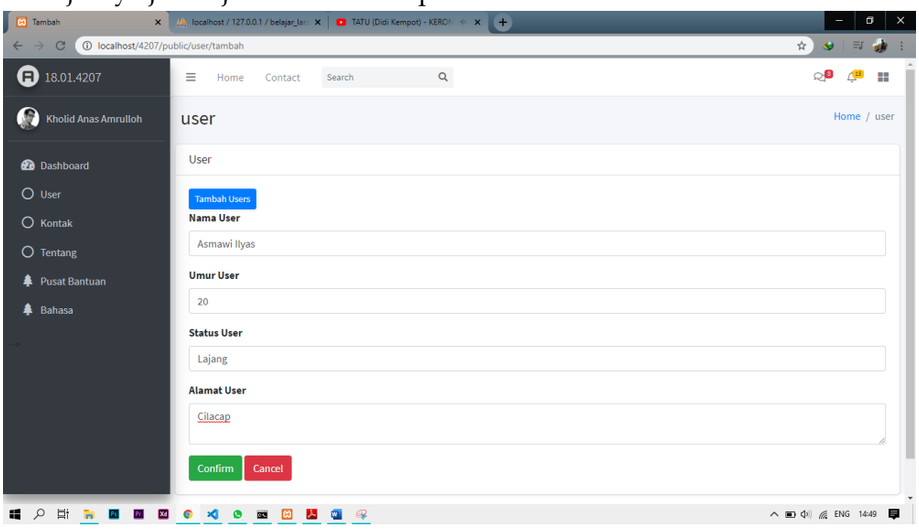
5. Lalu membuat Route store

```
Route::post('/user/store', 'UserController@store');
```

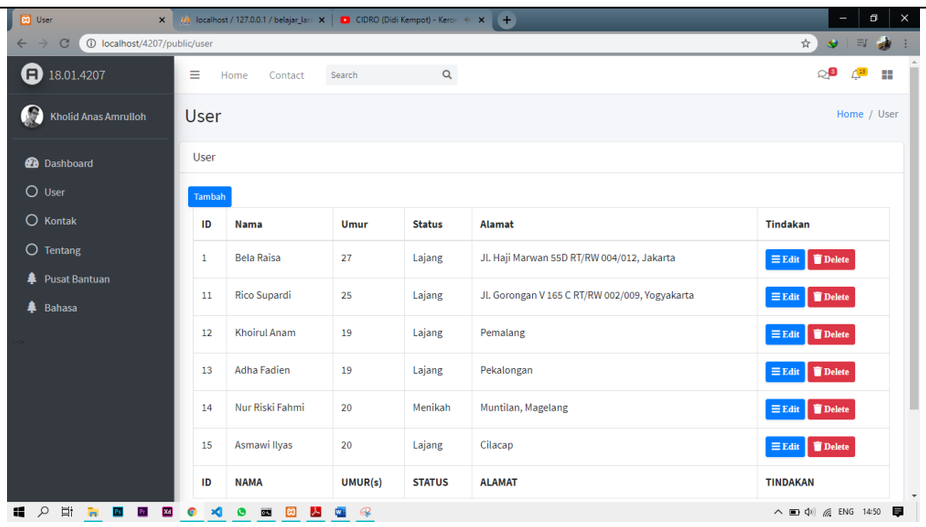
6. Langkah selanjutnya adalah membuat controller store(nama kolom pada tabel harus sesuai dengan yang ada di phpMyAdmin)

```
// method untuk insert data ke table user
public function store(Request $request)
{
    DB::table('user')->insert([
        'nama_User' => $request->nama,
        'umur_User' => $request->umur,
        'status_User' => $request->status,
        'alamat_User' => $request->alamat,
    ]);
    return redirect('/user');
}
```

7. Selanjutnya jika dijalankan akan seperti ini



8. Dan hasilnya jika ditambahkan maka akan urut dibawahnya karena ascending



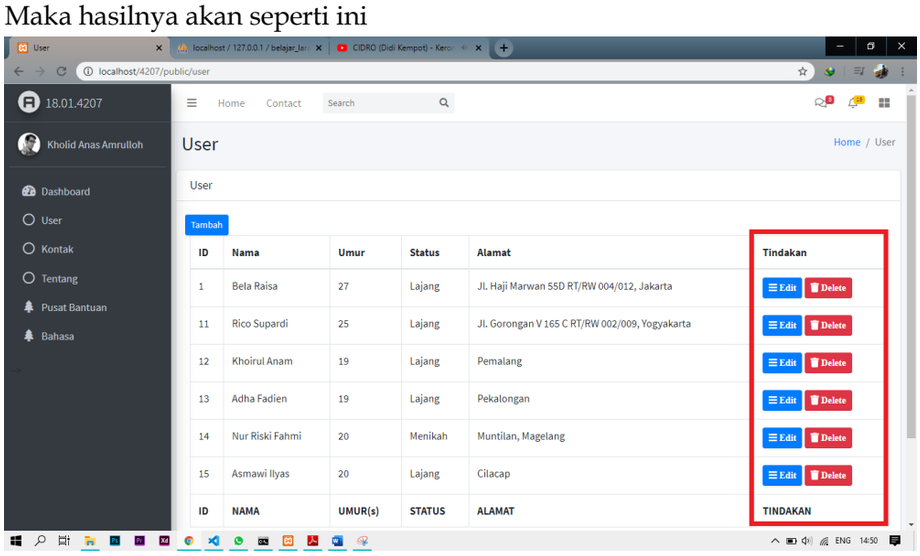
5. Membuat update dan edit pada user

Berikut langkah-langkahnya:

1. Menambahkan tombol edit dan hapus di dalam file index.blade.php. Lalu diarahkan pada route yang dibuat yaitu “user/edit/{{$u->id_User}}”

```
<tr class="odd" role="row">
    <td>{{ $u->id_User }}</td>
    <td>{{ $u->nama_User }}</td>
    <td>{{ $u->umur_User }}</td>
    <td>{{ $u->status_User }}</td>
    <td>{{ $u->alamat_User }}</td>
    <td>
        <a href="user/edit/{{$u->id_User}}" class="open_modul_ubah btn btn-sm btn-primary shadow-sm"><i class="fa fa-bars"> Edit</i></a>
        <a href="user/hapus/{{$u->id_User}}" class="btn btn-danger btn-sm delete-link"><i class="fa fa-trash"> Delete</i></a>
    </td>
</tr>
```

2. Maka hasilnya akan seperti ini



3. Langkah selanjutnya adalah membuat route untuk proses edit

```
Route::get('/user/edit/{id}', 'UserController@edit');
```

4. Lalu membuat controller edit yang digunakan untuk mengambil data pada database, setelah itu data di passingkan ke view edit.blade.php

```
//method untuk edit data user
public function edit($id)
{
    // mengambil data user berdasarkan id yang dipilih
    $user = DB::table('user')->where('id_User', $id)->get();
    // passing data user yang didapat ke view edit.blade.php
    return view('user.edit', ['user' => $user]);
}
```

5. Selanjutnya membuat view edit.blade.php seperti ini yang disimpan di folder user pada views

```
@extends('layout.master')
@section('title','Edit')
@section('content')
<!-- Content Header (Page header) -->
<section class="content-header">
    <div class="container-fluid">
        <div class="row mb-2">
            <div class="col-sm-6">
                <h1>Users</h1>
            </div>
            <div class="col-sm-6">
                <ol class="breadcrumb float-sm-right">
                    <li class="breadcrumb-item"><a href="#">Home</a></li>
                    <li class="breadcrumb-item active">Users</li>
                </ol>
            </div>
        </div>
    </div><!-- /.container-fluid -->
</section>

<!-- Main content -->
<section class="content">

    <!-- Default box -->
    <div class="card">
        <div class="card-header">
            <h3 class="card-title">Update Users</h3>
        </div>

        <div class="card-body">
            <div class="dataTables_wrapper dt-bootstrap4" id="example2_wrapper">
                <div class="row">
                    <div class="col-sm-12 col-md-6"></div>
                    <div class="col-sm-12 col-md-6"></div>
                    <div class="col-sm-12 col-md-6"></div>
                </div>
                <div class="row">
                    <div class="col-sm-12">

                        @foreach($user as $u)
                        <form action="{{ url('user/update/')}}" method="post">
                            {{ csrf_field() }}
                            <div class="form-group">
                                <input type="hidden" name="id" class="form-control" value="{{$u->id_User}}">
                            </div>
                            <div class="form-group">
                                <label for="nama">Nama User</label>
                                <input type="text" name="nama" class="form-control" value="{{$u->nama_User}}" require=required">
                            </div>
                            <div class="form-group">
                                <label for="umur">Umur User</label>
                                <input type="text" name="umur" class="form-control" value="{{$u->umur_User}}" require=" required">
                            </div>
                            <div class="form-group">
                                <label for="status">Status User</label>
                                <input type="text" name="status" class="form-control" value="{{$u->status_User}}" require=" required">
                            </div>
                            <div class="form-group">
                                <label for="alamat">Alamat User</label>
                                <textarea class="form-control" name="alamat" require="required">{{$u->alamat_User}}</textarea>
                            </div>
                            <div class="modal-footer">
                                <button class="btn btn-success" type="submit" value="Simpan Data">Confirm</button>
                                <button type="reset" class="btn btn-danger" data-dismiss="modal" aria-hidden="true">Cancel</button>
                            </div>

                        </form>
                        @endforeach
                    </div>
                </div>
            </div>
            <!-- /.card-body -->
        </div>
        <!-- /.card -->

</section>
<!-- /.content -->
@endsection
```

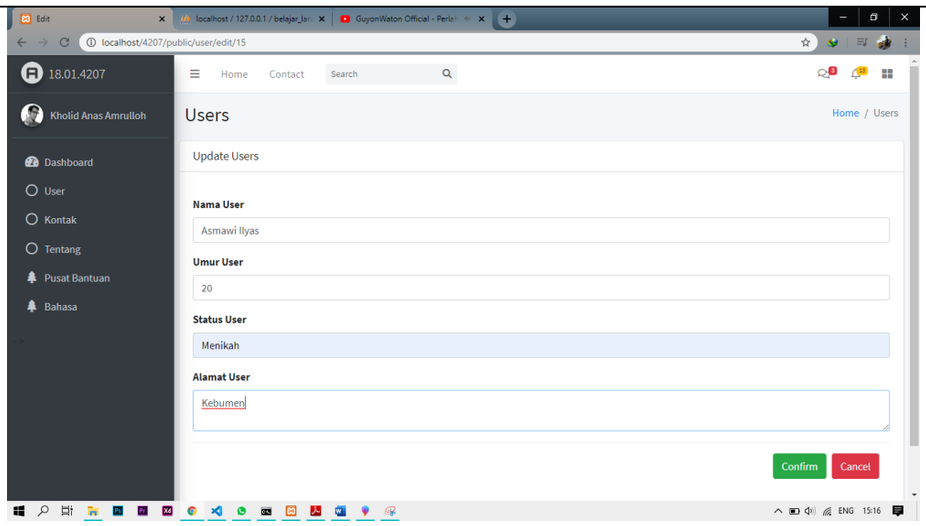
6. Membuat action pada fom dengan membuat route untuk action pada form edit

```
Route::post('/user/update', 'UserController@update');
```

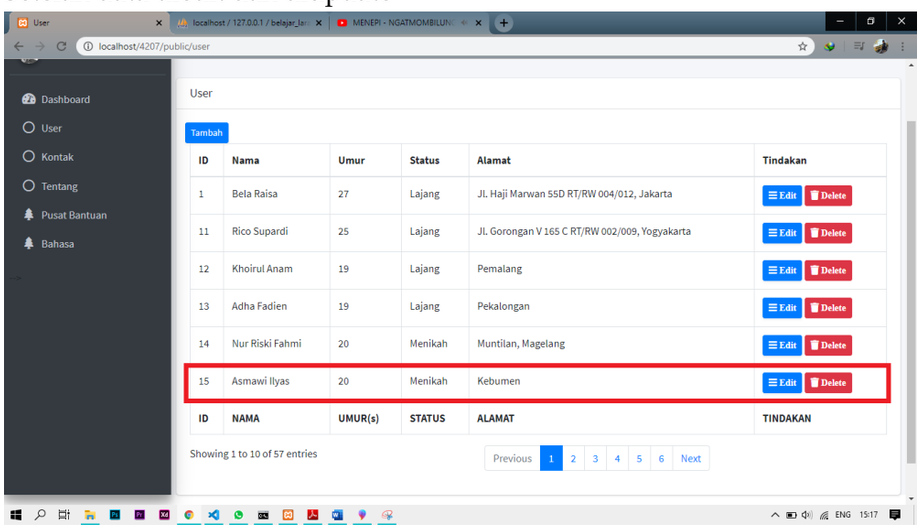
7. Membuat controller update

```
// update data user
public function update(Request $request)
{
    // update data user
    DB::table('user')->where('id_User', $request->id)->update([
        'nama_User' => $request->nama,
        'umur_User' => $request->umur,
        'status_User' => $request->status,
        'alamat_User' => $request->alamat,
    ]);
    //alihkan halaman ke halaman user
    return redirect('/user');
}
```

8. Jika dijalankan maka hasilnya seperti ini



9. Setelah data diedit dan diupdate



6. **Membuat Delete pada user**

   Berikut langkah-langkahnya:

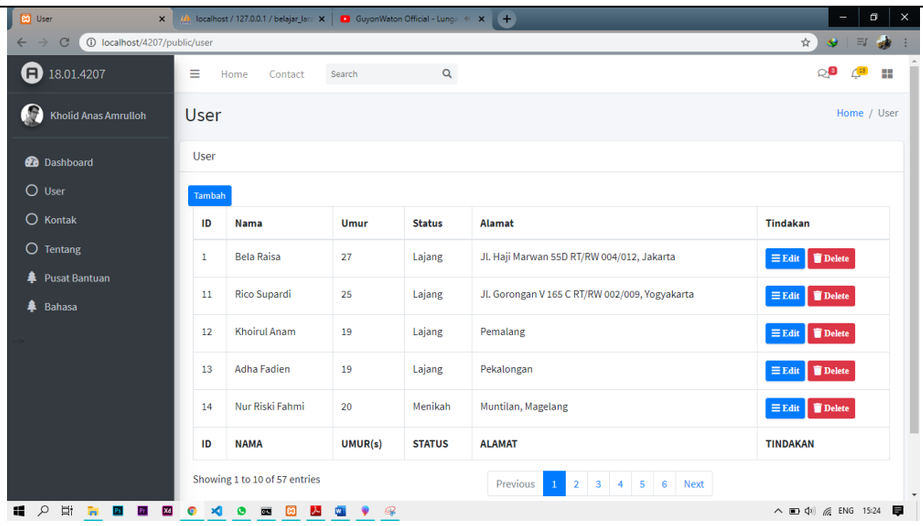
   1. Membuat route untuk proses /user/hapus{{$u->id_User}}

```
Route::get('/user/hapus/{id}', 'UserController@hapus');
```

   2. Membuat controller hapus pada UserController.php

```
    // method untuk menghapus data user
    public function hapus($id)
    {
        //menghapus data user berdasarkan id yang dipilih
        DB::table('user')->where('id_User', $id)->delete();

        //alihkan halaman ke halaman user
        return redirect('/user');
    }
}
```

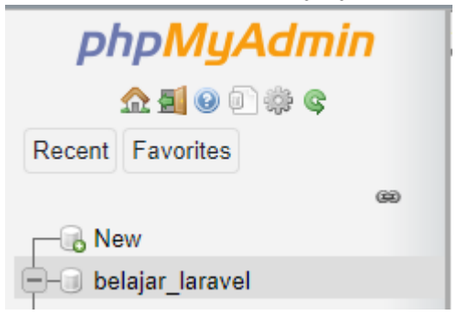
   3. Untuk hasilnya seperti ini



B. Membuat Seeder

Berikut langkah-langkahnya:

1. Menggunakan database belajar_laravel yang tadi sudah dibuat yang nantinya dapat membuat penginputan data testing pada pegawai dengan menggunakan seeder dari Laravel. Intinya yaitu membuat database di mysql.



2. Membuat seeder menggunakan perintah “php artisan make:seeder UserSeeder” dengan perintah cmd

```
C:\xampp\htdocs\4207>php artisan make:seeder UserSeeder
Seeder created successfully.
```

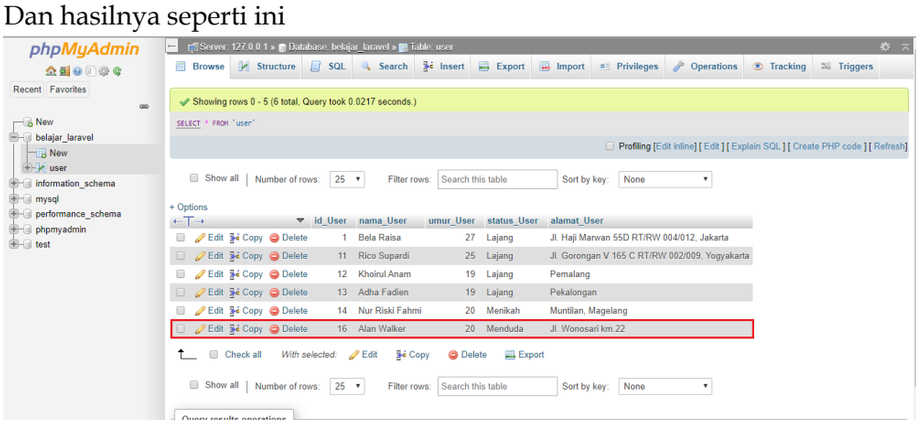
3. File tersebut tersimpan di 4207/database/seeds/UserSeeder, selanjutnya menambahkan sintak pada run

```php
<?php

use Illuminate\Database\Seeder;

class UserSeeder extends Seeder
{
    /**
     * Run the database seeds.
     *
     * @return void
     */
    public function run()
    {
        // insert data ke tabel user
        DB::table('user')->insert([
            'nama_User' => 'Alan Walker',
            'umur_User' => 20,
            'status_User' => 'Menduda',
            'alamat_User' => 'Jl. Wonosari km.22'
        ]);
    }
}
```

4. Untuk menjalankan perintah seed pada fungsi run, menggunakan perintah artisan seperti berikut

```
C:\xampp\htdocs\4207>php artisan db:seed --class=UserSeeder
Database seeding completed successfully.
```

5. Dan hasilnya seperti ini



C. Membuat Faker

Berikut langkah-langkahnya:

1. Membuka file seeder tadi, yaitu UserSeeder

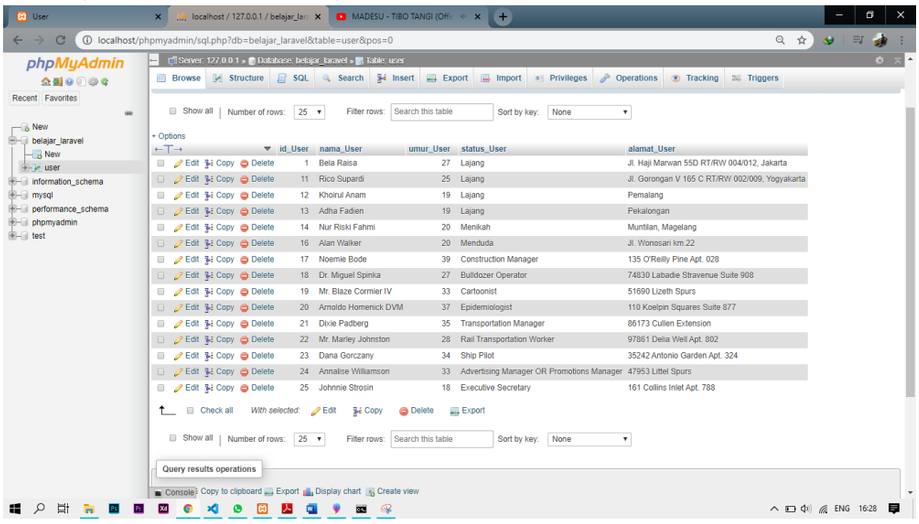
2. Lalu ubah sintak sebelumnya dengan sintak faker seperti di bawah ini

```php
    public function run()
    {
        // membuat faker dengan data Indonesia
        $faker = Faker::create();

        // looping digunakan untuk menampilkan data sebanyak 50 data
        for ($i = 1; $i <= 10; $i++) {


            // insert data ke tabel user dengan Faker
            DB::table('user')->insert([
                'nama_User' => $faker->name,
                'umur_User' => $faker->numberBetween(10, 20),
                'status_User' => $faker->jobTitle,
                'alamat_User' => $faker->streetAddress
            ]);
        }
    }
}
```

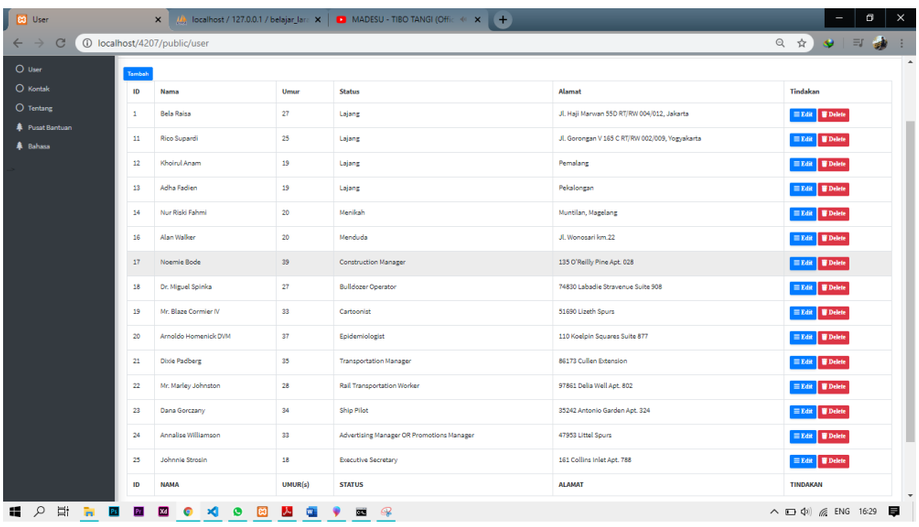
3. Untuk mengaktifkan seeder dengan data faker yaitu menggunakan perintah seperti gambar di bawah ini melalui cmd

```
C:\xampp\htdocs\4207>php artisan db:seed --class=UserSeeder
Database seeding completed successfully.
```

4. Hasilnya seperti ini



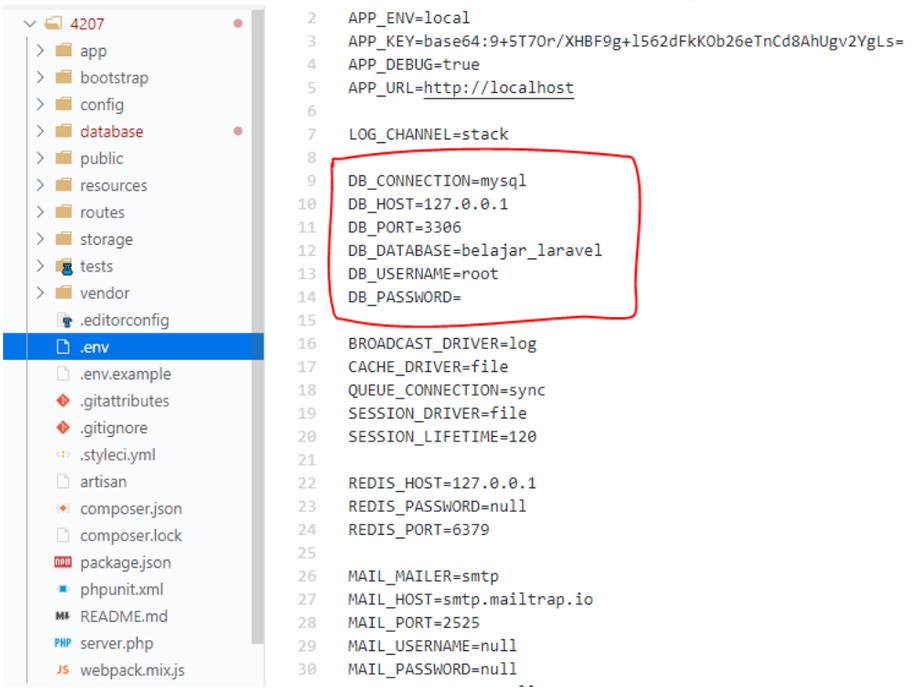
5. Pada halaman user



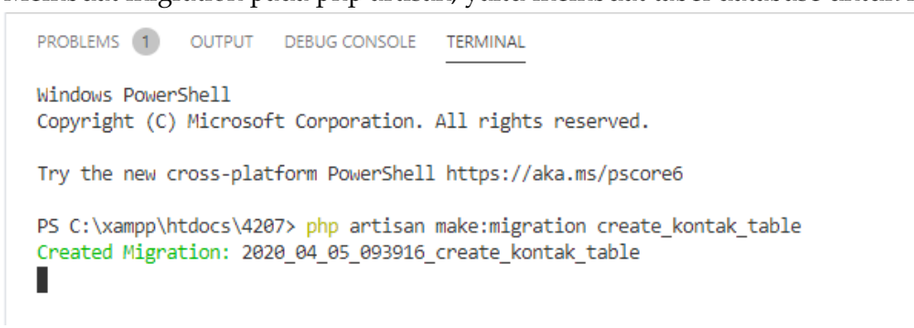
D. Membuat Migration

Berikut langkah-langkahnya:

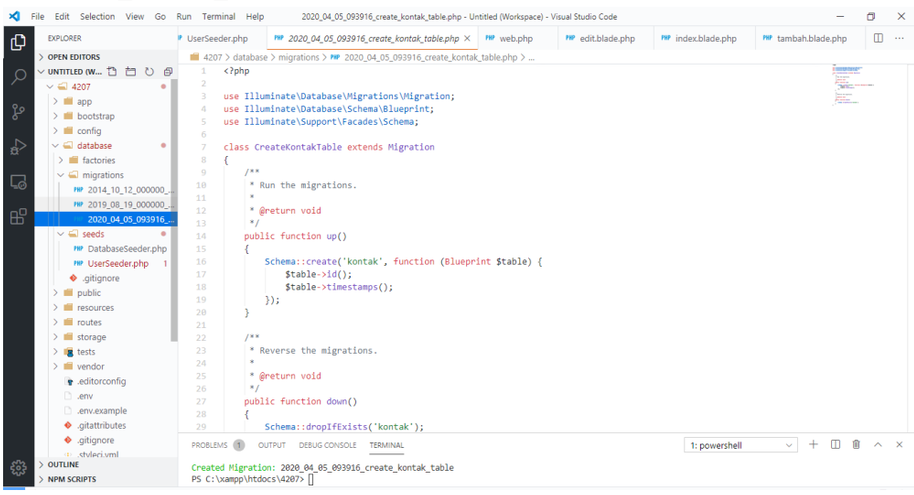
1. Sesuaikan koneksi database pada .env pastikan sama dengan database pada mysqli



2. Membuat migration pada php artisan, yaitu membuat tabel database untuk kontak



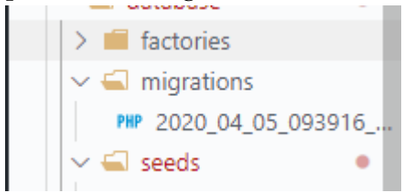
3. Selanjutnya kita lihat file yang suda dibuat tadi di folder database-> migrations -> file create_kontak_table.



4. Langkah selanjutnya adalah membuat kolom pada tabel

```php
     * @return void
     */

    /*MEMBUAT TABEL*/
    public function up()
    {
        Schema::create('kontak', function (Blueprint $table) {
            $table->increments('id');
            $table->string('nama');
            $table->string('telepon');
            $table->string('email');
        });
    }

    /**
     * Reverse the migrations.
     *
     * @return void
     */

    //MENGHAPUS TABEL
```

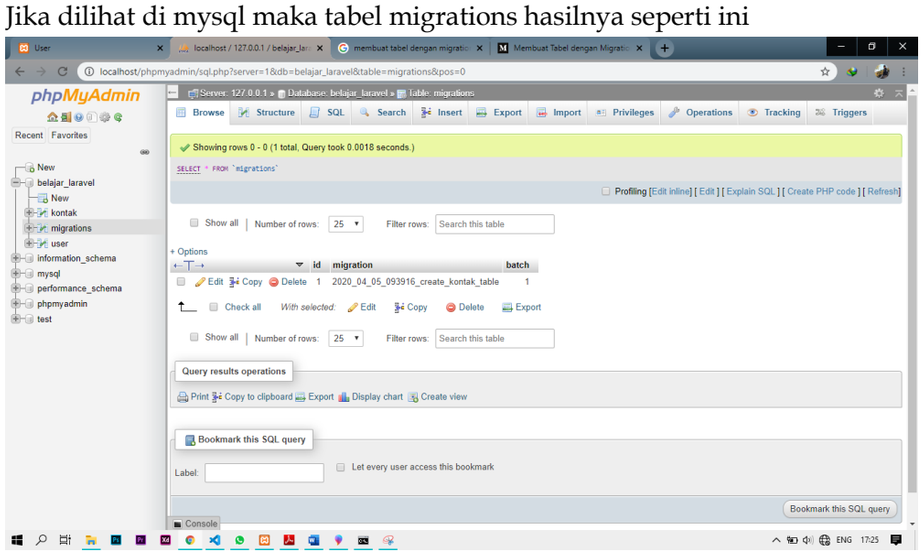
5. Hapus file migration yang lain, dan sisakan hanya file yang sudah kita buat tadi pada folder migrations



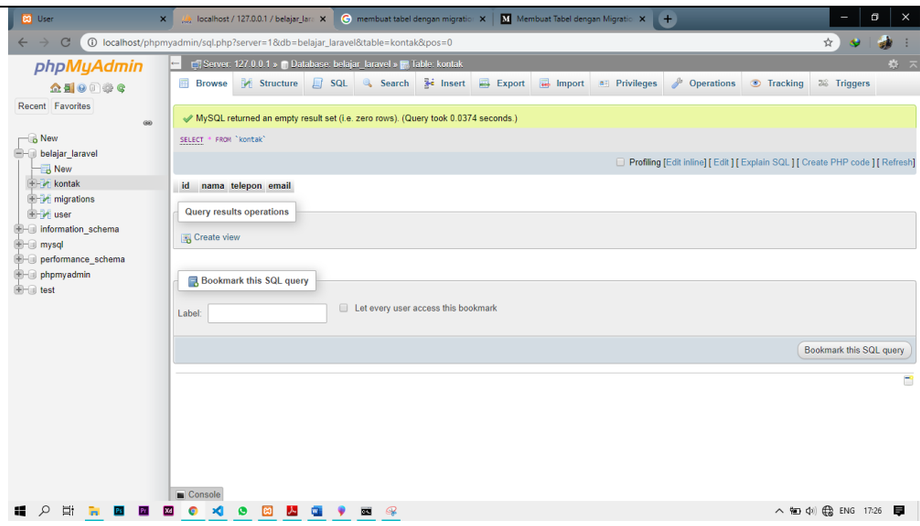
6. Selanjutnya menjalankan perintah pada terminal seperti ini

```
PS C:\xampp\htdocs\4207> php artisan migrate
Migration table created successfully.
Migrating: 2020_04_05_093916_create_kontak_table
Migrated:  2020_04_05_093916_create_kontak_table (1.01 seconds)
PS C:\xampp\htdocs\4207>
```

7. Jika dilihat di mysql maka tabel migrations hasilnya seperti ini



8. Selanjutnya tabel kontak



Merubah nama tabel dengan migration

Berikut langkah-langkahnya:

1. Masuk ke migration
2. Lalu masukkan sintak pada file tersebut

```
/*MEMBUAT TABEL*/
public function up()
{
    Schema::rename('kontak', 'kontak12');

    // Schema::create('kontak', function (Blueprint $table)
    //     $table->increments('id');
    //     $table->string('nama');
    //     $table->string('telepon');
    //     $table->string('email');
    // });
}
```

3. Lalu gunakan perintah pada cmd seperti berikut, sebelumnya karena tadi sudah di migrate. Maka lakukan reset terlebih dahulu lalu ulangi seperti dibawah ini

```
PS C:\xampp\htdocs\4207> Php artisan migrate:reset
Rolling back: 2020_04_05_093916_create_kontak_table
Rolled back:  2020_04_05_093916_create_kontak_table (0.11 seconds)
PS C:\xampp\htdocs\4207> php artisan migrate
Migrating: 2020_04_05_093916_create_kontak_table
Migrated:  2020_04_05_093916_create_kontak_table (0.6 seconds)
PS C:\xampp\htdocs\4207>
```

4. Hasilnya nama tabel berhasil diganti di mysql

Menghapus Tabel

Berikut langkah-langkahnya:

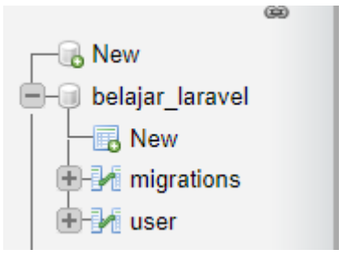
1. Masukkan sintak pada bagian dungsi down ()

```php
    //MENGHAPUS TABEL

    public function down()
    {
        //Hapus table
        Schema::drop('kontak12');

        //Hapus table jika ada
        Schema::dropIfExists('kontak12');
    }
}
```

2. Lalu gunakan perintah pada cmd seperti berikut ini, sebelumnya sudah di migrate maka lakukan reset terlebih dahulu

```
PS C:\xampp\htdocs\4207> php artisan migrate:reset
Rolling back: 2020_04_05_093916_create_kontak_table
Rolled back:  2020_04_05_093916_create_kontak_table (0.71 seconds)
PS C:\xampp\htdocs\4207> php artisan migrate
Migrating: 2020_04_05_093916_create_kontak_table
Migrated:  2020_04_05_093916_create_kontak_table (0 seconds)
PS C:\xampp\htdocs\4207>
```

3. Tabel “kontak12” sudah terhapus



Melakukan Rollback Migration Laravel

Berikut langkah-langkahnya:

1. Memasukkan perintah php artisan migrate:rollback pada cmd

```
PS C:\xampp\htdocs\4207> php artisan migrate:rollback
Rolling back: 2020_04_05_093916_create_kontak_table
```

LATIHAN

Berikut langkah-langkahnya:

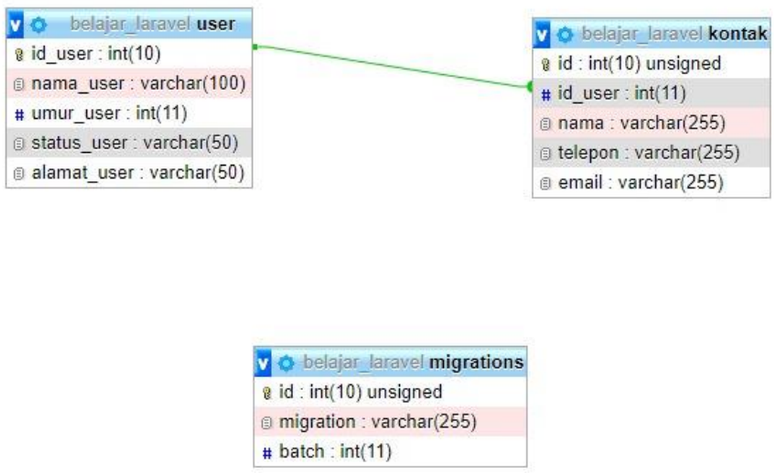
1. Lalu membuka masing” file migration nya tadi dan menambakan sintak seperti dibawah ini

```php
    /*MEMBUAT TABEL*/
    public function up()
    {
        schema::create('kontak', function (Blueprint $table) {
            $table->increments('id');
            $table->integer('id_User');
            $table->foreign('id_User')->references('id_User')->on('user');
            $table->string('nama');
            $table->string('telepon');
            $table->string('email');
        });
    }
```

2. Lalu ketikkan perintah seperti dibawah ini pada cmd

```
Nothing to rollback.
PS C:\xampp\htdocs\4207> php artisan migrate
Migrating: 2020_04_05_093916_create_kontak_table
Migrated:  2020_04_05_093916_create_kontak_table (4.85 seconds)
PS C:\xampp\htdocs\4207>
```

3. Dan hasil relasinya seperti ini



E. Kesimpulan

Jadi kesimpulan dari praktikum membuat CRUD dengan Query builder adalah, mahasiswa mampu mengetahui bagaimana fungsi-fungsi dari route, controller, dan view. Dengan membuat Crud maka mahasiswa mampu mengetahui bagaimana proses menampilkan sebuah view yang diawali dengan pembuatan route terlebih dahulu , selanjutnya pembuatan controller untuk pemanggilan view, dan juga pembuatan view itu sendiri yang di dalamnya menggunakan blade yang sudah diajarkan pertemuan sebelumnya. Dan juga mahasiswa mampu mengetahui dan melakukan bagaiman proses Seeder&faker untuk mengisi data pada database dan factory sebagai library yang menyimpan data palsu (dummy). serta Migration yang memungkinkan untuk dapat membuat , mengubah, dan memanipulasi serta menjalankan schema beserta tabelnya tanpa secara langsung dilakukan ke database sebenarnya.