

PRAKTIKUM PENGEMBANGAN APLIKASI WEB

MODUL 13 Laravel Database Tahap Dasar



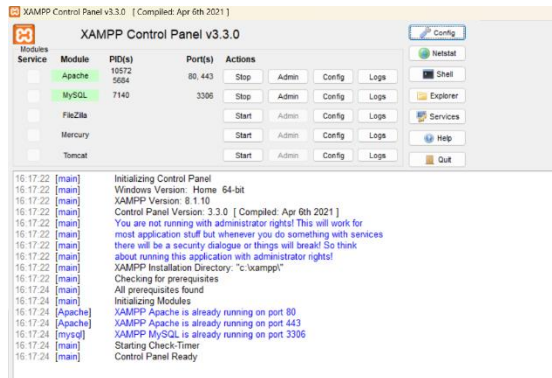
Sistem Informasi
Telkom University
Surabaya

Oleh :
Azrabelva Juventia-IS 06 02 (1204230083)

**PROGRAM STUDI S1 SISTEM INFORMASI
FAKULTAS REKAYASA INDUSTRI
TELKOM UNIVERSITY SURABAYA
2024**

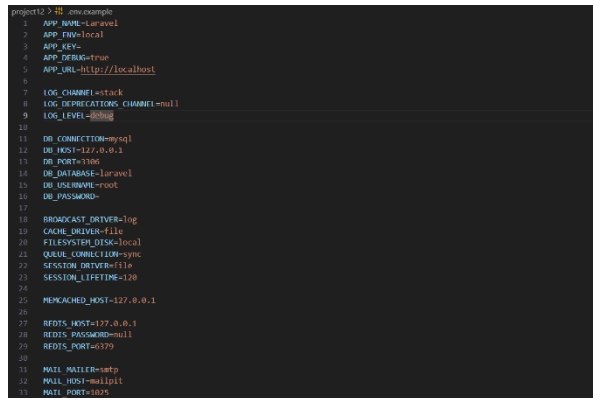
1. Aktifkan Service MySQL dan PhpMyAdmin

- Buka XAMPP Control Panel

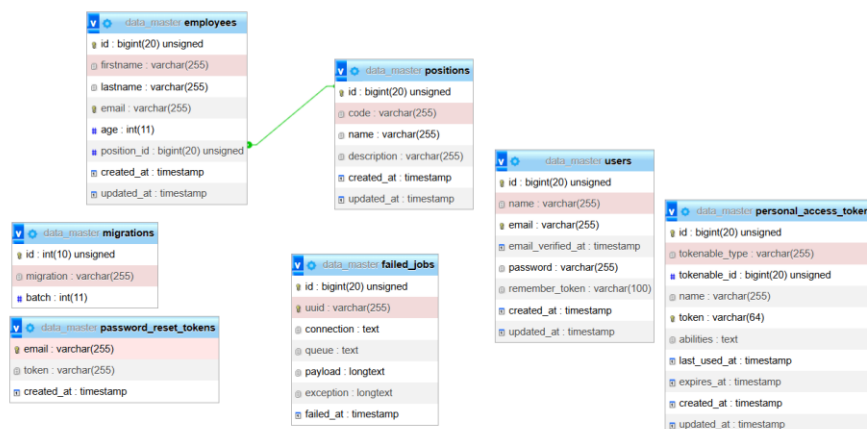


2. Setup Database Config

-



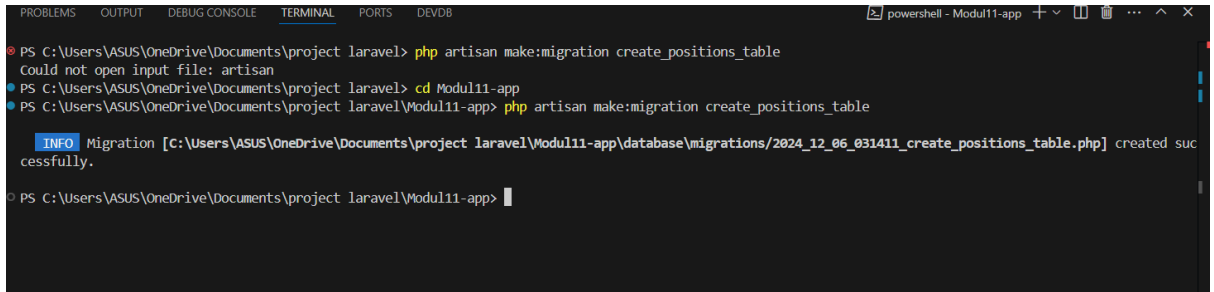
3. Skema Database



4. Membuat Skema Database Menggunakan Migration

- Generate file migration untuk tabel **positions** via Artisan

Perintah php artisan make:migration create_position_table digunakan dalam kerangka kerja laravel yang berguna untuk membuat file migrasi baru yang akan bertugas untuk membuat sebuah table position di database kita

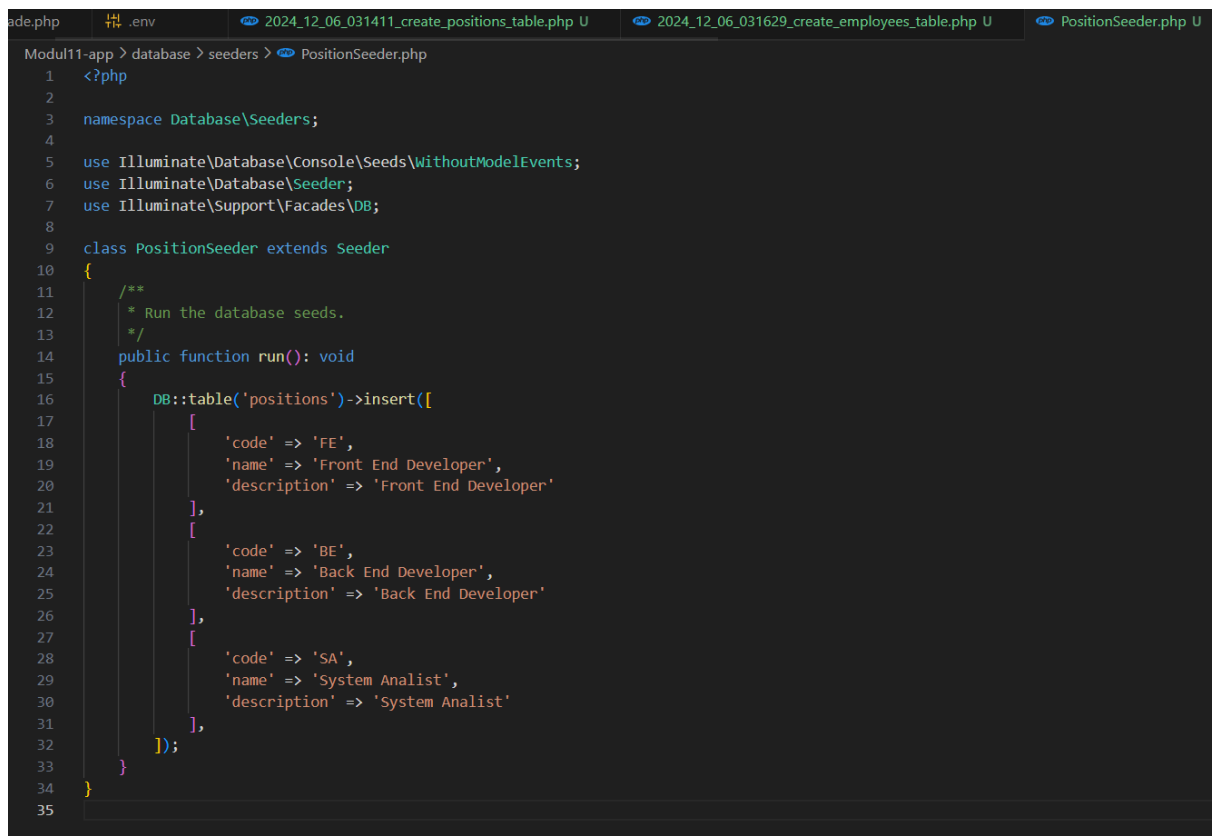


```
PS C:\Users\ASUS\OneDrive\Documents\project laravel> php artisan make:migration create_positions_table
Could not open input file: artisan
PS C:\Users\ASUS\OneDrive\Documents\project laravel> cd Modul11-app
PS C:\Users\ASUS\OneDrive\Documents\project laravel\Modul11-app> php artisan make:migration create_positions_table

[INFO] Migration [C:\Users\ASUS\OneDrive\Documents\project laravel\Modul11-app\database\migrations\2024_12_06_031411_create_positions_table.php] created successfully.
PS C:\Users\ASUS\OneDrive\Documents\project laravel\Modul11-app>
```

- kode program pada file migration yang mendefinisikan tabel **positions** seperti di bawah ini.

Dalam kode ini berguna untuk mengelola tabel database, kode ini akan membuat table positions pada database kita yang akan digunakan untuk menyimpan data pada sebuah posisi tertentu.



```
Modul11-app > database > seeders > PositionSeeder.php
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7 use Illuminate\Support\Facades\DB;
8
9 class PositionSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13     */
14     public function run(): void
15     {
16         DB::table('positions')->insert([
17             [
18                 'code' => 'FE',
19                 'name' => 'Front End Developer',
20                 'description' => 'Front End Developer'
21             ],
22             [
23                 'code' => 'BE',
24                 'name' => 'Back End Developer',
25                 'description' => 'Back End Developer'
26             ],
27             [
28                 'code' => 'SA',
29                 'name' => 'System Analyst',
30                 'description' => 'System Analyst'
31             ]
32         ]);
33     }
34 }
35
```

- Generate file migration untuk tabel **employees** via Artisan

```
PS C:\Users\ASUS\OneDrive\Documents\project laravel\Modul11-app> php artisan make:seeder EmployeeSeeder
```

```
INFO Seeder [C:\Users\ASUS\OneDrive\Documents\project laravel\Modul11-app\database\seeders\EmployeeSeeder.php] created successfully.
```

- Buat kode program pada file migration yang telah dibuat yang mendefinisikan tabel **employees** seperti di bawah ini.

Pada migration ini membuat table employees yang digunakan untuk meyimpan data karyawan dengan dihubungkan ke tabel position. Fungsi dari metode down digunakan untuk membatalkan perubahan yang dilakkan oleh up maka tabel employees akan dihapus.

```
Modul11-app > database > migrations > 2024_12_06_031629_create_employees_table.php
1  <?php
2
3  use Illuminate\Database\Migrations\Migration;
4  use Illuminate\Database\Schema\Blueprint;
5  use Illuminate\Support\Facades\Schema;
6
7  return new class extends Migration
8  {
9      /**
10       * Run the migrations.
11       */
12      public function up(): void
13      {
14          Schema::create('employees', function (Blueprint $table) {
15              $table->id();
16              $table->string('firstname');
17              $table->string('lastname')->nullable();
18              $table->string('email')->unique();
19              $table->integer('age');
20              $table->foreignId('position_id')->constrained();
21              $table->timestamps();
22          });
23      }
24
25      /**
26       * Reverse the migrations.
27       */
28      public function down(): void
29      {
30          Schema::dropIfExists('employees');
31      }
32  };
33
```

5. Membuat Data Dummy Menggunakan Seeder

Seeder ini digunakan untuk menambahkan data karyawan yang berjumlah 3 orang ke tabel employees. Setiap orang memiliki atribut seperti nama depan, nama belakang, email, umur dan ID posisi.

```
Modul11-app > database > seeders > PositionSeeder.php
1  <?php
2
3  namespace Database\Seeders;
4
5  use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6  use Illuminate\Database\Seeder;
7  use Illuminate\Support\Facades\DB;
8
9  class PositionSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         DB::table('positions')->insert([
17             [
18                 'code' => 'FE',
19                 'name' => 'Front End Developer',
20                 'description' => 'Front End Developer'
21             ],
22             [
23                 'code' => 'BE',
24                 'name' => 'Back End Developer',
25                 'description' => 'Back End Developer'
26             ],
27             [
28                 'code' => 'SA',
29                 'name' => 'System Analyst',
30                 'description' => 'System Analyst'
31             ]
32         ]);
33     }
34 }
35
```

- Generate file seeder untuk tabel **employees** via Artisan

```
PS C:\Users\ASUS\OneDrive\Documents\project laravel\Modul11-app> php artisan make:seeder EmployeeSeeder
INFO Seeder [C:\Users\ASUS\OneDrive\Documents\project laravel\Modul11-app\database\seeders\EmployeeSeeder.php] created successfully.
```

- Buat kode program pada file seeder yang telah dibuat untuk tabel **employees** seperti di bawah ini.

Data pada seeder ini dapat digunakan untuk menguji relasi antara table employees dan positions dan juga fitur aplikasi yang akan memanfaatkan data karyawan

```
2024_12_06_031411_create_positions_table.php U 2024_12_06_031629_create_employees_table.php U PositionSeeder.php U EmployeeSeeder.php U
Modul11-app > database > seeders > EmployeeSeeder.php
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7 use Illuminate\Support\Facades\DB;
8
9 class EmployeeSeeder extends Seeder
10 {
11     /**
12      * Run the database seeds.
13      */
14     public function run(): void
15     {
16         DB::table('employees')->insert([
17             [
18                 'firstname' => 'Purnama',
19                 'lastname' => 'Anaking',
20                 'email'=> 'purnama.anaking@gmail.com',
21                 'age' => 20,
22                 'position_id' => 1
23             ],
24             [
25                 'firstname' => 'Adzanil',
26                 'lastname' => 'Rachmadhi',
27                 'email'=> 'adzanil.rachmadhi@gmail.com',
28                 'age' => 25,
29                 'position_id' => 2
30             ],
31             [
32                 'firstname' => 'Berlian',
33                 'lastname' => 'Rahmy',
```

- Definiskan file seeder yang akan dieksekusi pada function **run()** di dalam file **DatabaseSeeder.php**

Databaseseeder.php berfungsi untuk engelola dan menjalankan seeder dalam satu tempat dan memudahkan untuk mengisi tabel database dengan data yang diperlukan. Dengan memanggil positionsedder dan employeeeeder akan emastikan bahwa data tabel position dan employees dapat dimuat degan mudah

```
Modul11-app > database > seeders > DatabaseSeeder.php
1 <?php
2
3 namespace Database\Seeders;
4
5 // use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7
8 class DatabaseSeeder extends Seeder
9 {
10     /**
11      * Seed the application's database.
12      */
13     public function run(): void
14     {
15         $this->call([
16             PositionSeeder::class,
17             EmployeeSeeder::class
18         ]);
19     }
20 }
21
```

- Eksekusi file **seeder** yang telah dibuat via Artisan

```
PS C:\Users\ASUS\OneDrive\Documents\project laravel\Modul11-app> php artisan db:seed

INFO Seeding database.

Database\Seeders\PositionSeeder ..... RUNNING
Database\Seeders\PositionSeeder ..... 59 ms DONE

Database\Seeders\EmployeeSeeder ..... RUNNING
Database\Seeders\EmployeeSeeder ..... 2 ms DONE

PS C:\Users\ASUS\OneDrive\Documents\project laravel\Modul11-app> |
```

6. Menampilkan List Data dari Database

Method index ini meng query data karyawan dan posisi mereka menggunakan RAW SQL dan mengirimkannya ke tampilan employee.index untuk ditampilkan dalam tabel

```
public function index()
{
    $pageTitle = 'Employee List';

    // RAW SQL QUERY
    $employees = DB::select('
        select *, employees.id as employee_id, positions.name as position_name
        from employees
        left join positions on employees.position_id = positions.id
    ');

    return view('employee.index', [
        'pageTitle' => $pageTitle,
        'employees' => $employees
    ]);
}
```

```
namespace App\Http\Controllers;
use Illuminate\Support\Facades\DB;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Validator;
use App\Models\Employee;
use App\Models\Position;
```

- Tampilkan data employee pada file View.

Secara keseluruhan kode ini akan menghasilkan sebuah tabel yang akan menampilkan sebuah data karyawan dan akan ada aktivitas seperti mengedit, melihat dan menghapus data

```

<table class="table table-bordered table-hover table-striped mb-0 bg-white">
  <thead>
    <tr>
      <th>First Name</th>
      <th>Last Name</th>
      <th>Email</th>
      <th>Age</th>
      <th>Position</th>
      <th></th>
    </tr>
  </thead>
  <tbody>
    @foreach ($employees as $employee)
      <tr>
        <td>{{ $employee->firstname }}</td>
        <td>{{ $employee->lastname }}</td>
        <td>{{ $employee->email }}</td>
        <td>{{ $employee->age }}</td>
        <td>{{ $employee->position_name }}</td>
        <td>
          <div class="d-flex">
            <a href="{{ route('employees.show', ['employee' => $employee->employee_id]) }}" class="btn btn-outline-dark btn-sm me-2">
            <a href="{{ route('employees.edit', ['employee' => $employee->employee_id]) }}" class="btn btn-outline-dark btn-sm me-2">
            <div>
              <form action="{{ route('employees.destroy', ['employee' => $employee->employee_id]) }}" method="POST">
                @csrf
                @method('delete')
                <button type="submit" class="btn btn-outline-dark btn-sm me-2"><i class="bi-trash"></i></button>
            </div>
          </div>
        </td>
      </tr>
    </foreach>
  </tbody>
</table>

```

7. Input Data ke Database

Method create() ini mengambil data dari tabel position menggunakan RAW dan mengirim data ke tampilan employee. Didalam tampilan data posisi digunakan untuk membuat dropdown pilihan posisi saat membuat karyawan baru

```

public function create()
{
    $pageTitle = 'Create Employee';
    // RAW SQL Query
    $positions = DB::select('select * from positions');

    return view('employee.create', compact('pageTitle', 'positions'));
}

```

- Baca data **positions** pada file View.


Kode inni juga merupakan bagian dari form HTML untuk memilih posisi karyawan. Form ini akan menampilkan dropdown untuk memilih dari table yang diambil pada position. Jika ada kesalahan validasi terkait dengan posisi maka pengguna akan menerima pesan error


```

<div class="col-md-12 mb-3">
  <label for="position" class="form-label">Position</label>
  <select name="position" id="position" class="form-select">
    @foreach ($positions as $position)
      <option value="{{ $position->id }}" {{ old('position'
    @endforeach
  </select>
  @error('position')
    <div class="text-danger"><small>{{ $message }}</small></div>
  @enderror
</div>

```

- Hasil Akhir **Form Create Employee** adalah seperti di bawah ini.



Create Employee

First Name

Last Name

Email

Age

Position

FE - Front End Developer
▼

⬅ Cancel

✔ Save

- membuat Query Insert pada method **store()** di dalam **EmployeeController**. Kemudian **redirect** Route ke halaman Employee List.

Validasi input sebelum data disimpan dalam database kode ini akan memastikan bahwa data yang diisi pengguna sudah valid.

```
public function store(Request $request)
{
    $messages = [
        'required' => ':Attribute harus diisi.',
        'email' => 'Isi :attribute dengan format yang benar',
        'numeric' => 'Isi :attribute dengan angka'
    ];

    $validator = Validator::make($request->all(), [
        'firstName' => 'required',
        'lastName' => 'required',
        'email' => 'required|email',
        'age' => 'required|numeric',
    ], $messages);

    if ($validator->fails()) {
        return redirect()->back()->withErrors($validator)->withInput();
    }

    // INSERT QUERY
    DB::table('employees')->insert([
        'firstname' => $request->firstName,
        'lastname' => $request->lastName,
        'email' => $request->email,
        'age' => $request->age,
        'position_id' => $request->position,
    ]);

    return redirect()->route('employees.index');
}
```

8. Menampilkan Detail Data dari Database

```
public function show(string $id)
{
    $pageTitle = 'Employee Detail';

    // RAW SQL QUERY
    $employee = collect(DB::select('
        select *, employees.id as employee_id, positions.name as position_name
        from employees
        left join positions on employees.position_id = positions.id
        where employees.id = ?
    ', [$id]))->first();

    return view('employee.show', compact('pageTitle', 'employee'));
}
```

- Buat file baru di **/views/employee/show.blade.php**. Tampilkan data employee pada file View tersebut.

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>{{ $pageTitle }}</title>
    @vite('resources/sass/app.scss')
</head>
<body>
    <nav class="navbar navbar-expand-md navbar-dark bg-primary">
        <div class="container">
            <a href="{{ route('home') }}" class="navbar-brand mb-0 h1"><i class="bi-hexagon-fill me-2"></i> Data Master</a>

            <button type="button" class="navbar-toggler" data-bs-toggle="collapse" data-bs-target="#navbarSupportedContent">
                <span class="navbar-toggler-icon"></span>
            </button>

            <div class="collapse navbar-collapse" id="navbarSupportedContent">
                <hr class="d-lg-none text-white-50">

                <ul class="navbar-nav flex-row flex-wrap">
                    <li class="nav-item col-2 col-md-auto"><a href="{{ route('home') }}" class="nav-link">Home</a></li>
                    <li class="nav-item col-2 col-md-auto"><a href="{{ route('employees.index') }}" class="nav-link active">Employee List</a>
                </ul>

                <hr class="d-lg-none text-white-50">

                <a href="{{ route('profile') }}" class="btn btn-outline-light my-2 ms-md-auto"><i class="bi-person-circle me-1"></i> My Profile
            </div>
        </div>
    </nav>
```

9. Menghapus Data dari Database

- Buat Builder Query pada method **destroy()** di dalam file **EmployeeController** kemudian redirect Route ke halaman Employee List.

```
public function destroy(string $id)  
{  
    // QUERY BUILDER  
    DB::table('employees')  
        ->where('id', $id)  
        ->delete();  
  
    return redirect()->route('employees.index');  
}
```

TUGAS

```
1 public function edit(string $id)
2 {
3     $pageTitle = 'Edit Employee';
4
5     // ELOQUENT
6     $positions = Position::all();
7     $employee = Employee::find($id);
8
9     return view('employee.edit', compact('pageTitle', 'positions', 'employee'));
10 }
11
12 public function update(Request $request, string $id)
13 {
14     $messages = [
15         'required' => ':Attribute harus diisi.',
16         'email' => 'Isi :attribute dengan format yang benar',
17         'numeric' => 'Isi :attribute dengan angka'
18     ];
19
20     $validator = Validator::make($request->all(), [
21         'firstName' => 'required',
22         'lastName' => 'required',
23         'email' => 'required|email',
24         'age' => 'required|numeric',
25     ], $messages);
26
27     if ($validator->fails()) {
28         return redirect()->back()->withErrors($validator)->withInput();
29     }
30
31     // ELOQUENT
32     $employee = Employee::find($id);
33     $employee->firstname = $request->firstName;
34     $employee->lastname = $request->lastName;
35     $employee->email = $request->email;
36     $employee->age = $request->age;
37     $employee->position_id = $request->position;
38     $employee->save();
39
40     return redirect()->route('employees.index');
41 }
42
```

- Metode `edit` pada baris 1 hingga 9 berguna untuk menampilkan form untuk mengedit data karyawan. Metode ini menerima sebuah parameter ID karyawan. Dan mengambil daftar posisi yang telah

tersedia dan juga data karyawan yang akan diedit lalu kemudian mengembalikan tampilan atau view dengan data yang diperlukan

- Metode update pada baris 12 hingga 41 digunakan untuk memproses pengiriman form yang digunakan untuk memperbarui data karyawan. Dalam metode ini akan mengvalidasi data yang diterima dan memperbarui data karyawan lalu menyimpannya pada database
- Validasi memastikan bahwa data yang dimasukkan pengguna sudah benar atau belum sebelum akhirnya data tersebut diperbarui
- Jika validasi gagal pengguna akan dikembalikan ke form dengan pesan error jika berhasil akan diarahkan pada halaman daftar karyawan