

PRAKTIKUM MODUL 05
PEMROGRAMAN FRAMEWORK
LARAVEL DATABASE TAHAP DASAR



DISUSUN OLEH:

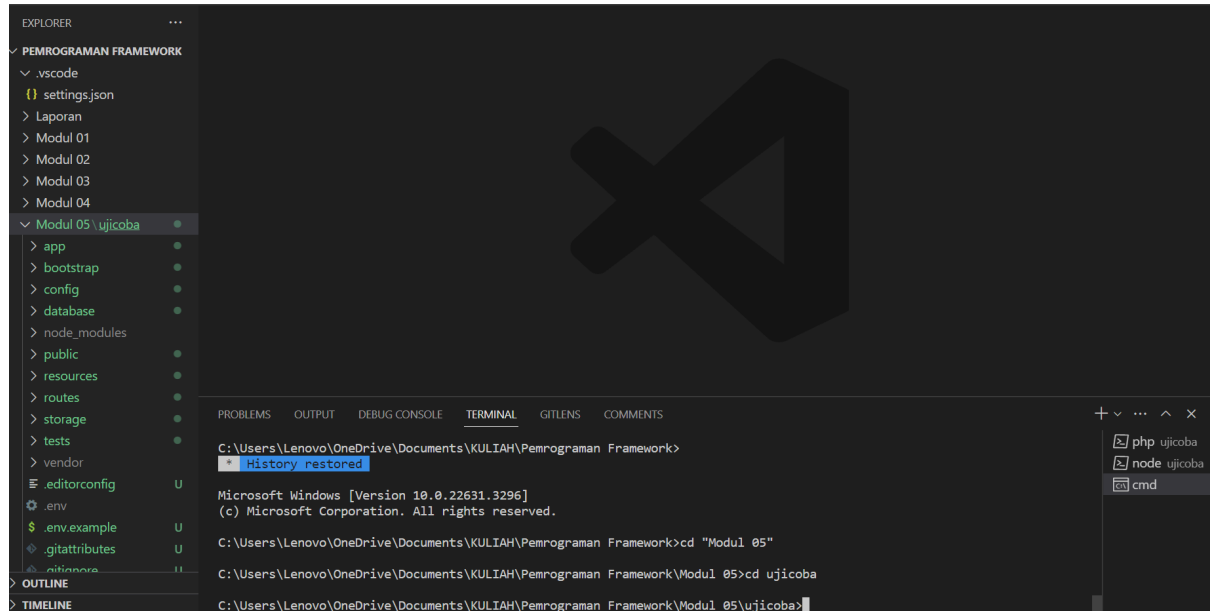
1204200081 RAFAEL FILLAH FIRMANSYAH

PROGRAM STUDI SISTEM INFORMASI
FAKULTAS REKAYASA INDUSTRI
TELKOM UNIVERSITY SURABAYA
2024

Generate Laravel Project

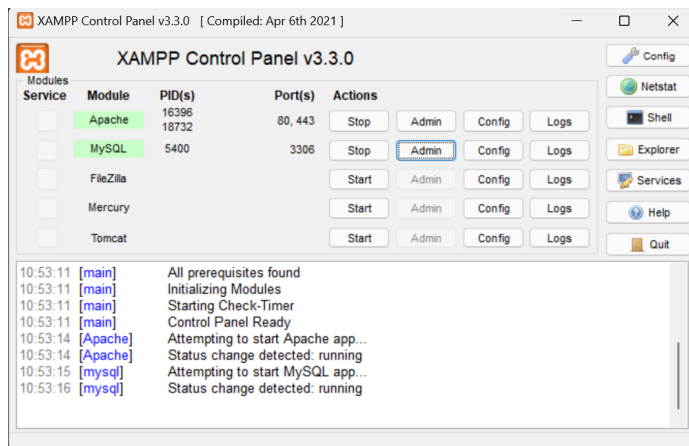
Link Github: <https://github.com/elfill03/Pemrograman-Framework>

1. Saya menggunakan project sebelumnya yang telah digunakan pada modul 4 sehingga tidak perlu melakukan set up project laravel ulang.

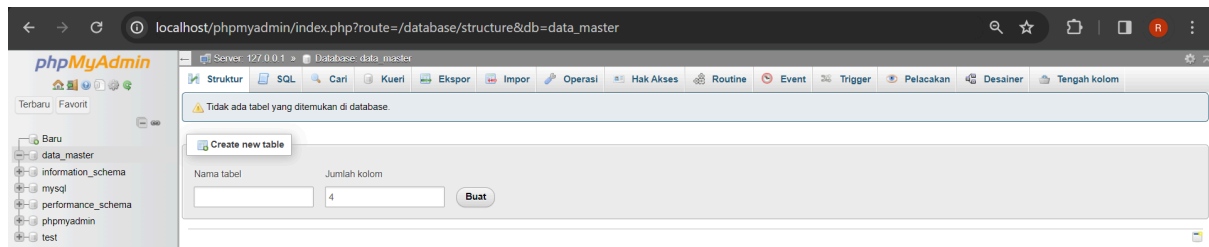


Aktivasi Service MySQL dan PhpMyAdmin

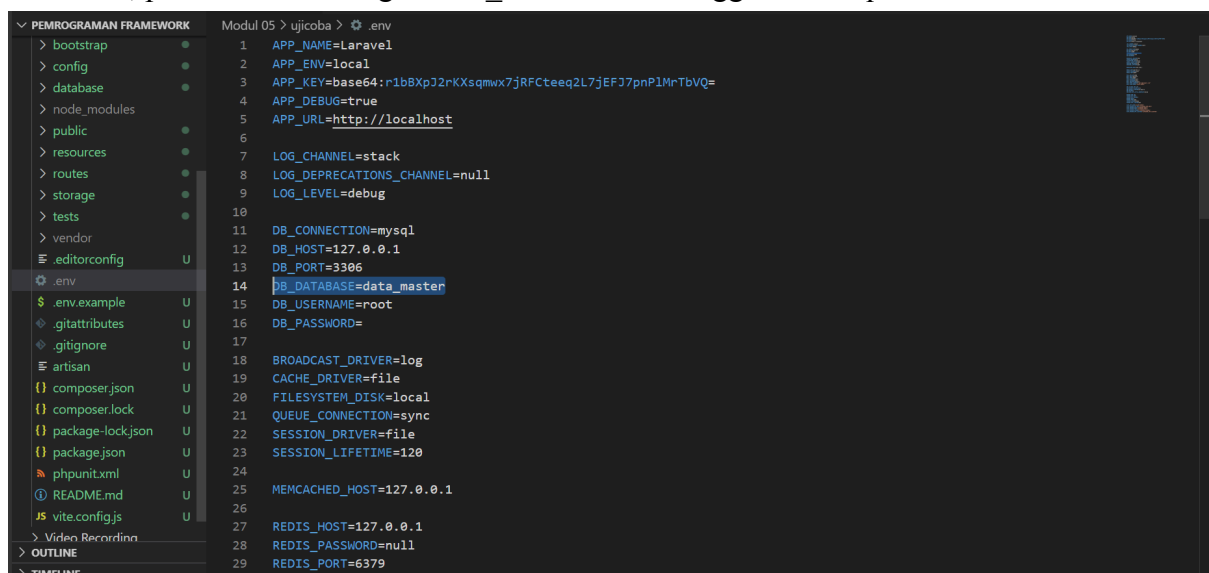
1. Buka XAMPP Control Panel lalu aktifkan apache dan MySQL



2. Kemudian buka phpmyadmin dan buat sebuah database dengan nama “data_master” yang nantinya akan digunakan untuk data employee pada halaman employee list



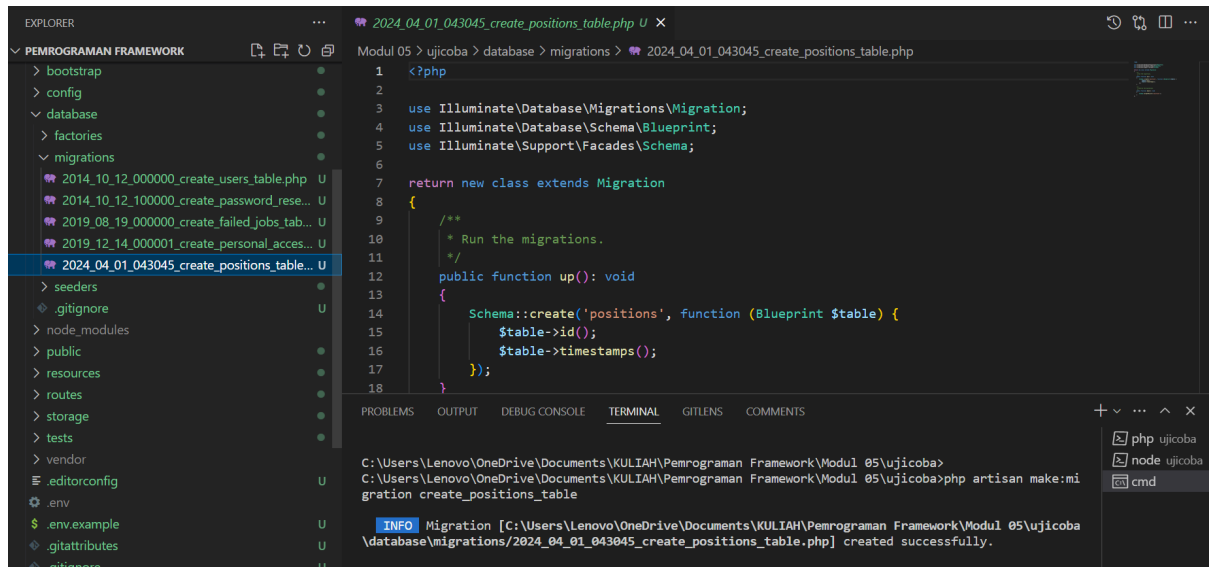
3. Selanjutnya melakukan konfigurasi database pada file .env. Pada line 14 terdapat DB_Database yang digunakan, apabila masih default dengan nama laravel ubah menjadi data_master. Selain itu saya masih menggunakan laravel 10 yang sedikit berbeda dengan laravel 11, pada laravel 11 bagian DB_Connection menggunakan sqlite



Membuat Skema Database Menggunakan Migration

Saya akan membuat skema database sederhana yang dimana terdapat 2 tabel yaitu employees dan positions. Tabel tersebut memiliki hubungan one-to-many, dimana pada tabel employees terdapat kolom position_id yang merujuk pada kolom id pada tabel positions.

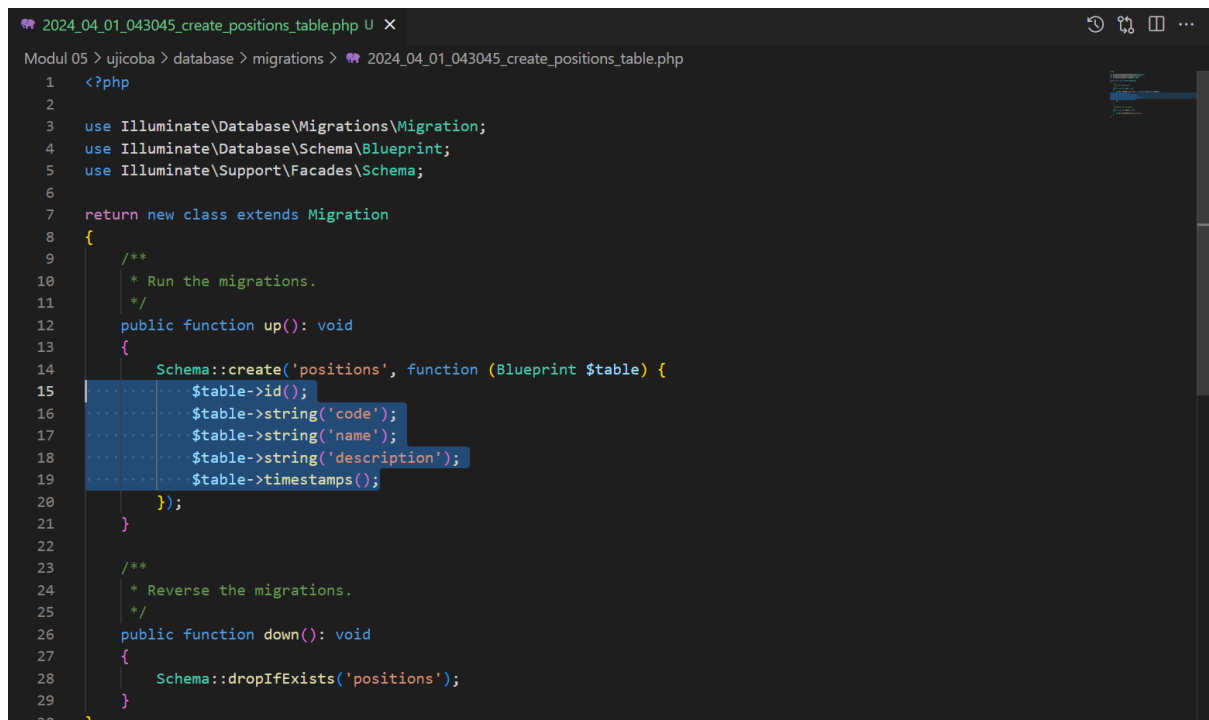
1. Membuat file migration untuk tabel positions



The screenshot shows the VS Code interface with the Explorer panel on the left displaying the project structure. The file explorer shows a directory named 'migrations' containing several PHP files. The file '2024_04_01_043045_create_positions_table.php' is selected. The main editor shows the content of this file, which is a Laravel migration class. The class extends 'Migration' and implements the 'up()' method. The 'up()' method uses the 'Schema::create()' method to create a table named 'positions'. The 'Schema::create()' method is called with a closure that defines the table's structure: it has an 'id' column (primary key), a 'code' column (string), a 'name' column (string), a 'description' column (string), and a 'timestamps()' column. The 'down()' method is also implemented, using 'Schema::dropIfExists()' to drop the table. The terminal at the bottom shows the command 'php artisan make:migration create_positions_table' and the output 'INFO Migration [C:\Users\Lenovo\OneDrive\Documents\KULIAH\Pemrograman Framework\Modul 05\ujicoba\database\migrations\2024_04_01_043045_create_positions_table.php] created successfully.'

```
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('positions', function (Blueprint $table) {
15             $table->id();
16             $table->string('code');
17             $table->string('name');
18             $table->string('description');
19             $table->timestamps();
20         });
21     }
22
23     /**
24      * Reverse the migrations.
25      */
26     public function down(): void
27     {
28         Schema::dropIfExists('positions');
29     }
30 }
```

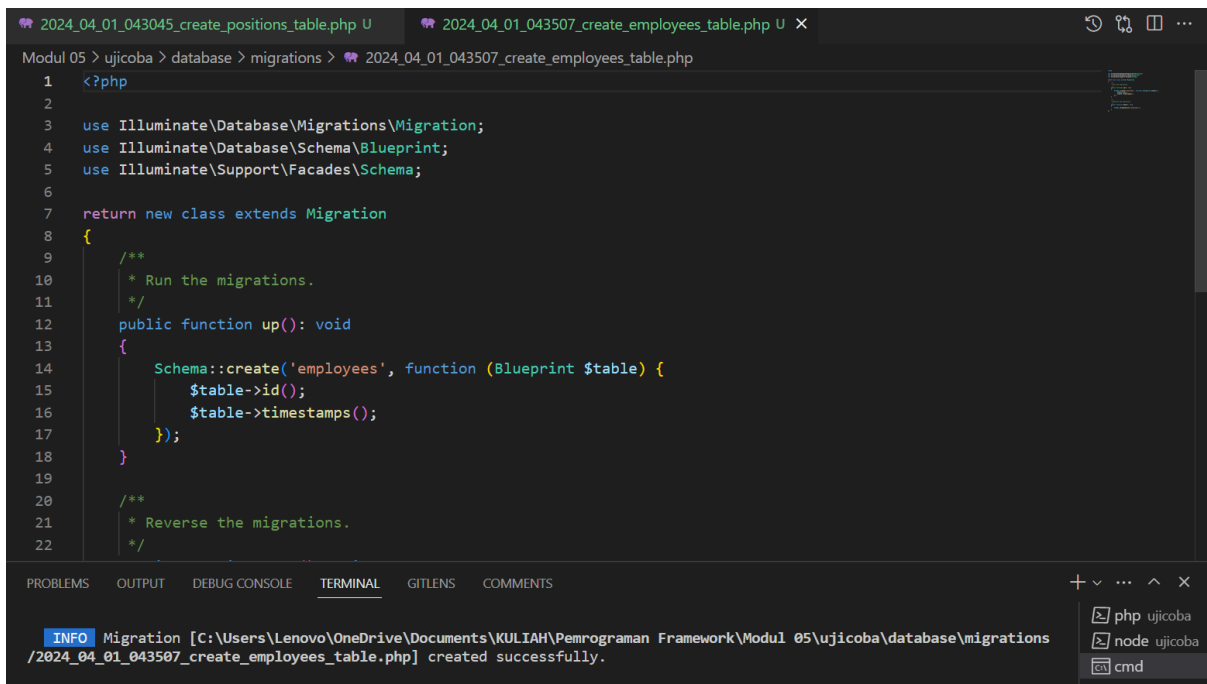
2. Kemudian definisikan kolom-kolom yang akan digunakan pada file migration tersebut (lane 15-19)



The screenshot shows the same VS Code interface as the previous one, but with the 'up()' method of the migration class expanded. The columns defined in the 'Schema::create()' closure are highlighted with blue boxes. The columns are: 'id' (primary key), 'code' (string), 'name' (string), 'description' (string), and 'timestamps()' (timestamps). The 'down()' method is also visible, showing the 'dropIfExists()' method call.

```
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('positions', function (Blueprint $table) {
15             $table->id();
16             $table->string('code');
17             $table->string('name');
18             $table->string('description');
19             $table->timestamps();
20         });
21     }
22
23     /**
24      * Reverse the migrations.
25      */
26     public function down(): void
27     {
28         Schema::dropIfExists('positions');
29     }
30 }
```

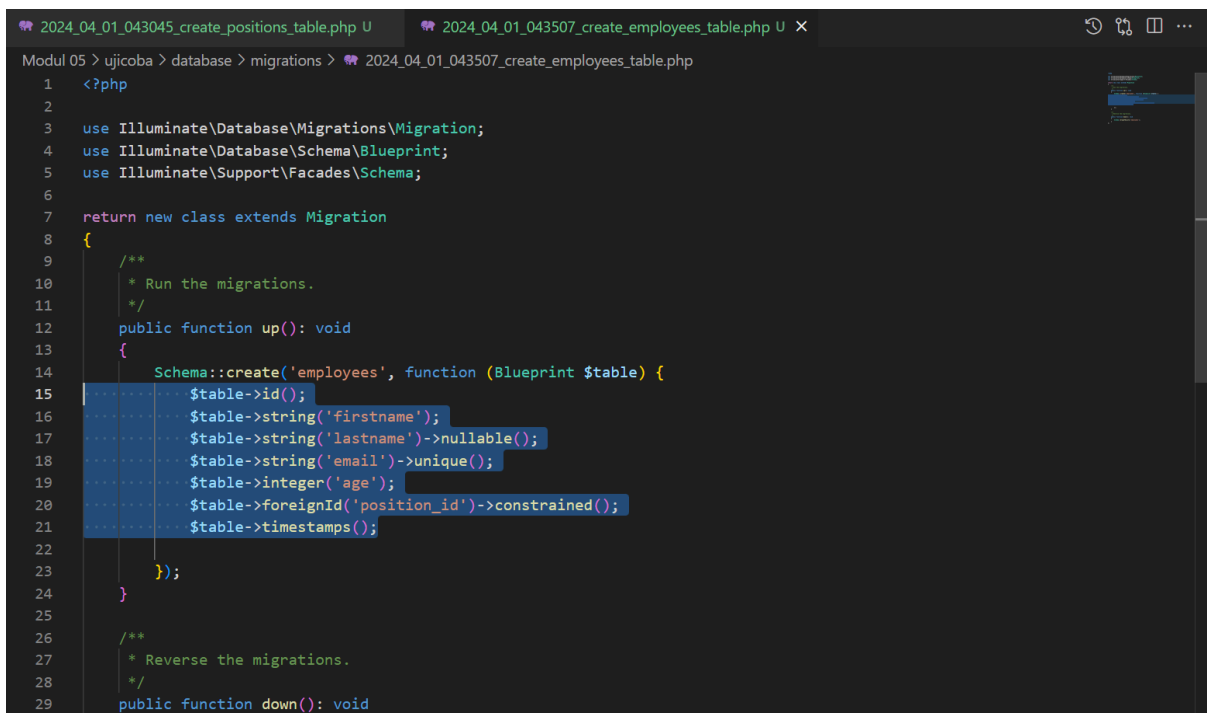
3. Buat satu lagi file migration untuk tabel employees



The screenshot shows a VS Code editor with a file named `2024_04_01_043507_create_employees_table.php` open. The file contains a PHP class that extends `Illuminate\Database\Migrations\Migration`. The `up()` method uses `Schema::create()` to create a table named 'employees' with columns for `id` and `timestamps`. The `down()` method is also present but empty. The terminal at the bottom shows a successful message: `Migration [C:\Users\Lenovo\OneDrive\Documents\KULIAH\Pemrograman Framework\Modul 05\ujicoba\database\migrations\2024_04_01_043507_create_employees_table.php] created successfully.`

```
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->timestamps();
17         });
18     }
19
20     /**
21      * Reverse the migrations.
22      */
23     public function down(): void
24     {
25     }
26 }
```

4. Lalu definisikan kolom-kolom yang akan digunakan pada file migration tersebut (lane 15-21)



The screenshot shows the same VS Code editor with the same file, but now the `up()` method includes definitions for several columns: `id`, `firstname`, `lastname` (nullable), `email` (unique), `age` (integer), `position_id` (foreign key constrained to `position_id`), and `timestamps`. The `down()` method remains empty.

```
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     }
31 }
```

5. Selanjutnya eksekusi file migration yang telah dibuat dengan cara “php artisan migrate” pada cmd

```
C:\Users\Lenovo\OneDrive\Documents\KULIAH\Pemrograman Framework\Modul 05\ujicoba>php artisan migrate
```

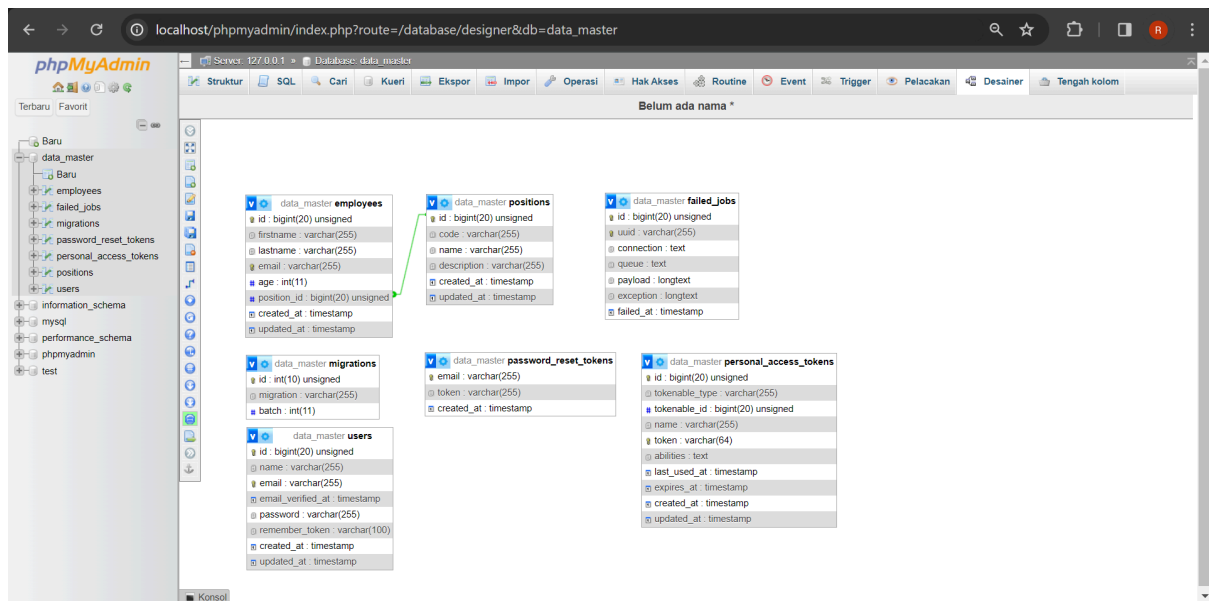
```
INFO Preparing database.
```

```
Creating migration table ..... 210ms DONE
```

```
INFO Running migrations.
```

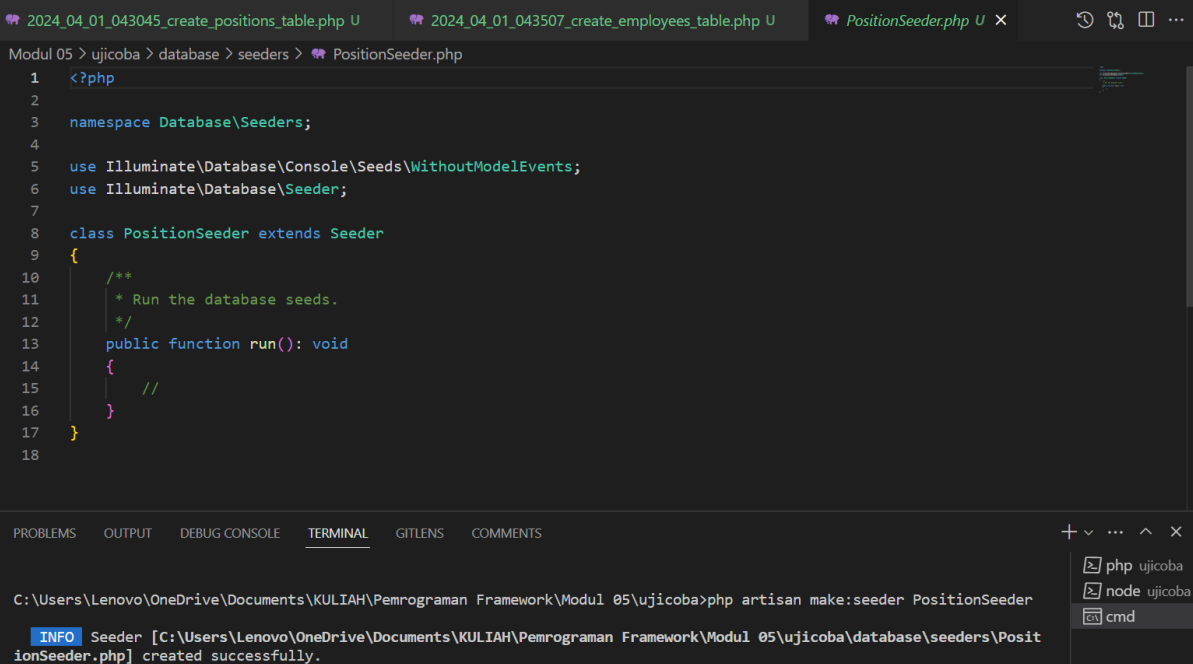
```
2014_10_12_000000_create_users_table ..... 107ms DONE
2014_10_12_100000_create_password_reset_tokens_table ..... 7ms DONE
2019_08_19_000000_create_failed_jobs_table ..... 65ms DONE
2019_12_14_000001_create_personal_access_tokens_table ..... 33ms DONE
2024_04_01_043045_create_positions_table ..... 8ms DONE
2024_04_01_043507_create_employees_table ..... 107ms DONE
```

6. Schema database yang sudah dibuat (Peletakan default tabel tidak seperti ini, ini setelah dirapikan sedikit)



Membuat Data Dummy Menggunakan Seeder

1. Pertama buat sebuah seeder dengan nama PositionSeeder yang akan digunakan untuk membuat data dummy pada tabel positions



The screenshot shows a code editor with three tabs: `2024_04_01_043045_create_positions_table.php U`, `2024_04_01_043507_create_employees_table.php U`, and `PositionSeeder.php U`. The active file is `PositionSeeder.php`. The code defines a `PositionSeeder` class extending `Seeder` from the `Database\Seeders` namespace. It includes the `Illuminate\Database\Console\Seeds\WithoutModelEvents` trait and the `Illuminate\Database\Seeder` class. The `run()` method is currently empty. The terminal at the bottom shows the command `php artisan make:seeder PositionSeeder` being executed successfully.

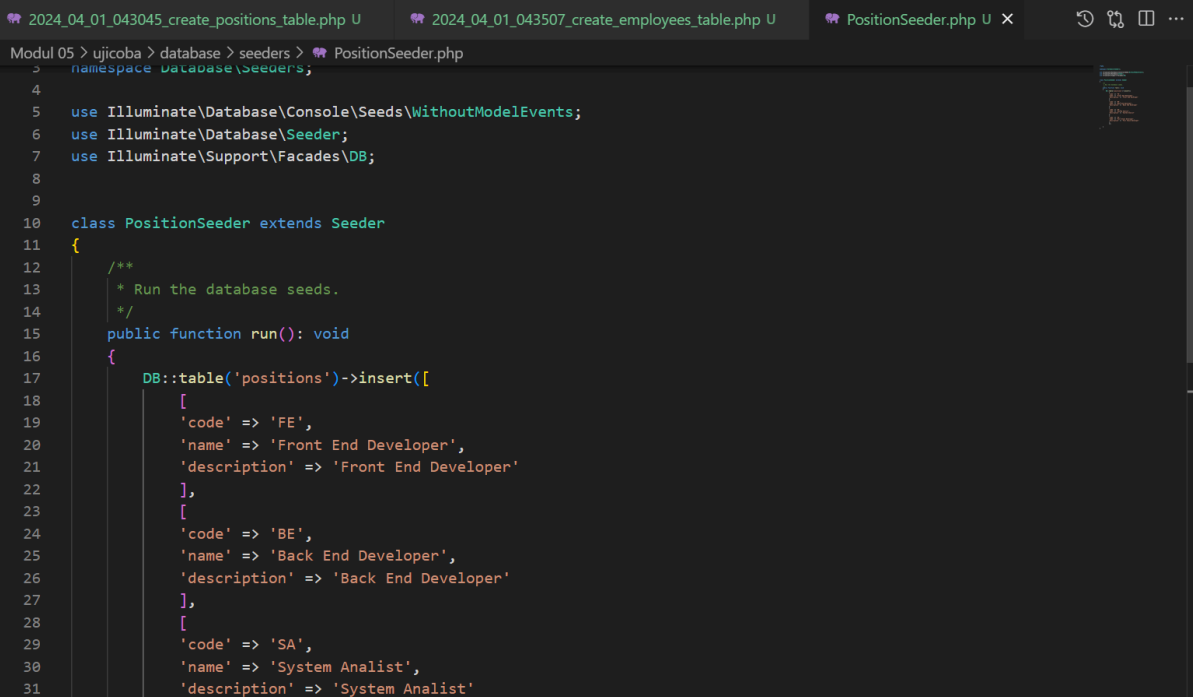
```
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7
8 class PositionSeeder extends Seeder
9 {
10     /**
11      * Run the database seeds.
12      */
13     public function run(): void
14     {
15         //
16     }
17 }
18
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL GITLENS COMMENTS

C:\Users\Lenovo\OneDrive\Documents\KULIAH\Pemrograman Framework\Modul 05\ujicoba>php artisan make:seeder PositionSeeder

INFO Seeder [C:\Users\Lenovo\OneDrive\Documents\KULIAH\Pemrograman Framework\Modul 05\ujicoba\database\seeders\PositionSeeder.php] created successfully.

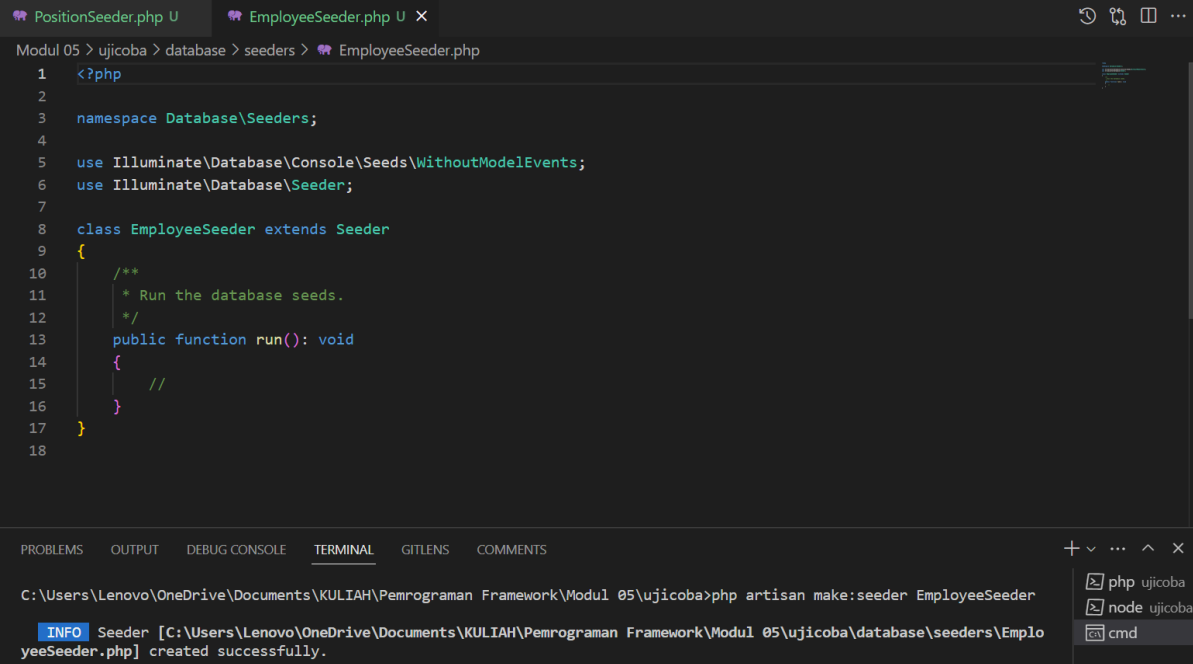
2. Kemudian tambahkan kode program pada file seeder untuk tabel positions seperti berikut. Tambahkan import untuk databasenya (lane 7) dan tambahkan isi dari tabel positions (lane 17-32)



The screenshot shows the same IDE with the `PositionSeeder.php` file updated. Line 7 now includes `use Illuminate\Support\Facades\DB;`. The `run()` method (lines 17-32) contains a `DB::table('positions')->insert()` call with an array of three data entries for 'FE', 'BE', and 'SA' roles.

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

3. Buat lagi seeder dengan nama EmployeeSeeder yang akan digunakan untuk membuat data dummy pada tabel Employees



The screenshot shows a Visual Studio Code editor with two tabs: 'PositionSeeder.php U' and 'EmployeeSeeder.php U X'. The active file is 'EmployeeSeeder.php'. The code in the editor is as follows:

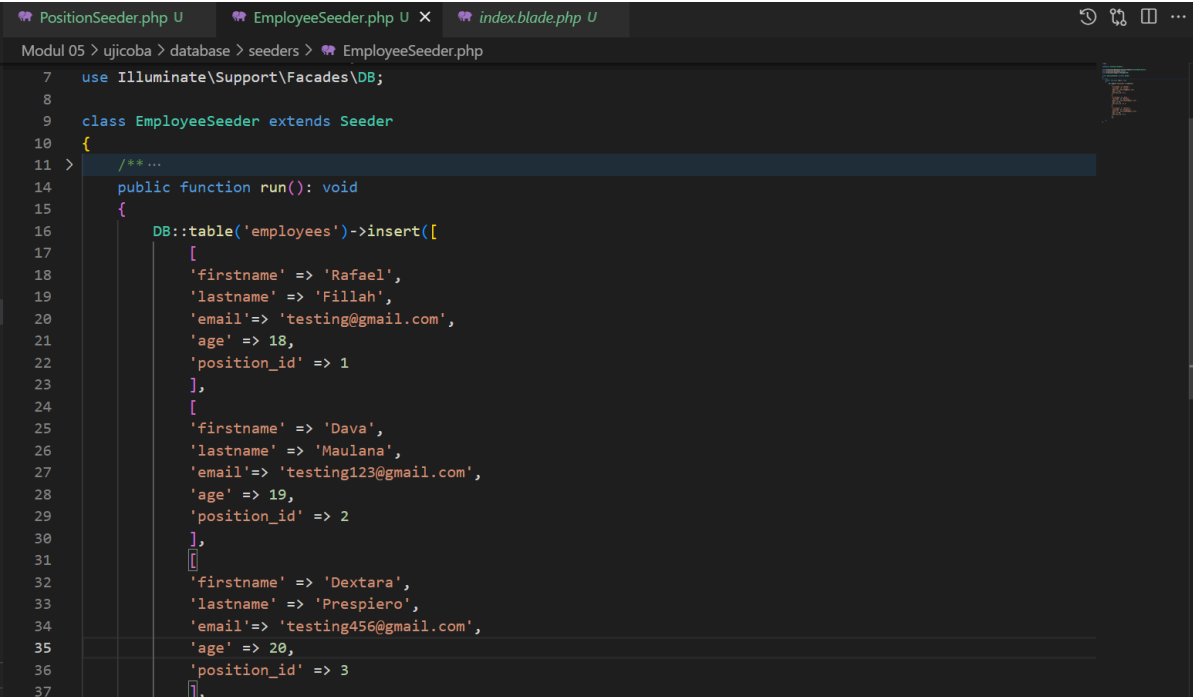
```
1 <?php
2
3 namespace Database\Seeders;
4
5 use Illuminate\Database\Console\Seeds\WithoutModelEvents;
6 use Illuminate\Database\Seeder;
7
8 class EmployeeSeeder extends Seeder
9 {
10     /**
11      * Run the database seeds.
12      */
13     public function run(): void
14     {
15         //
16     }
17 }
18
```

Below the editor, the TERMINAL tab is active, showing the command execution:

```
C:\Users\Lenovo\OneDrive\Documents\KULIAH\Pemrograman Framework\Modul 05\ujicoba>php artisan make:seeder EmployeeSeeder
```

An INFO message states: 'Seeder [C:\Users\Lenovo\OneDrive\Documents\KULIAH\Pemrograman Framework\Modul 05\ujicoba\database\seeders\EmployeeSeeder.php] created successfully.'

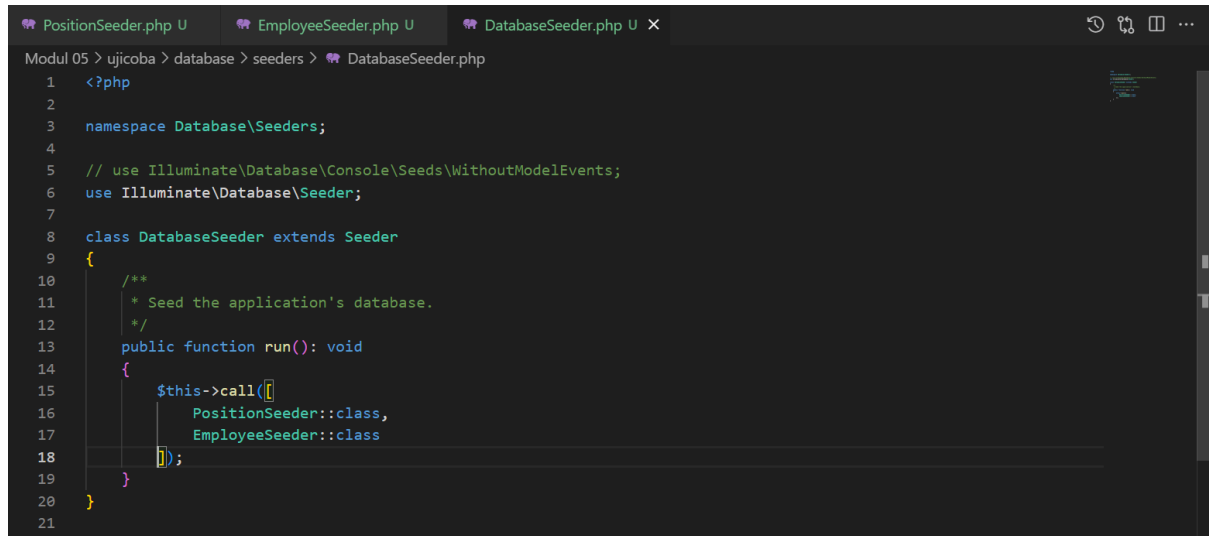
4. Kemudian tambahkan import untuk databasenya (lane 7) dan tambahkan isi dari tabel employee (lane 16-37)



The screenshot shows the same Visual Studio Code editor with three tabs: 'PositionSeeder.php U', 'EmployeeSeeder.php U X', and 'index.blade.php U'. The active file is 'EmployeeSeeder.php'. The code in the editor is as follows:

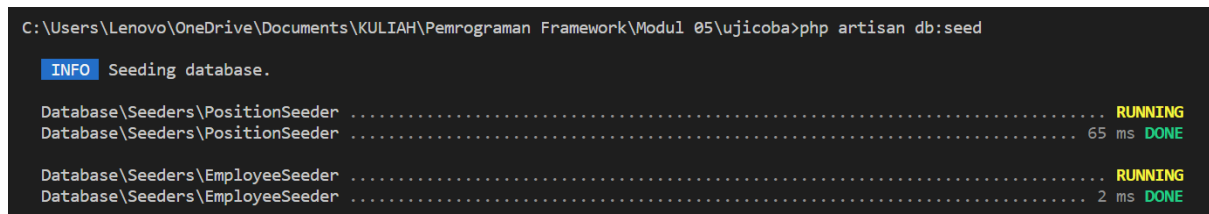
```
7 use Illuminate\Support\Facades\DB;
8
9 class EmployeeSeeder extends Seeder
10 {
11     /** ...
12     */
13     public function run(): void
14     {
15         DB::table('employees')->insert([
16             [
17                 'firstname' => 'Rafael',
18                 'lastname' => 'Fillah',
19                 'email'=> 'testing@gmail.com',
20                 'age' => 18,
21                 'position_id' => 1
22             ],
23             [
24                 'firstname' => 'Dava',
25                 'lastname' => 'Maulana',
26                 'email'=> 'testing123@gmail.com',
27                 'age' => 19,
28                 'position_id' => 2
29             ],
30             [
31                 'firstname' => 'Dextara',
32                 'lastname' => 'Prespiero',
33                 'email'=> 'testing456@gmail.com',
34                 'age' => 20,
35                 'position_id' => 3
36             ],
37         ]);
38     }
39 }
```


5. Selanjutnya melakukan define seeder pada file DatabaseSeeder agar dapat ditemukan untuk menambahkan data dummy pada database data_master



```
Modul 05 > ujicoba > 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
```

6. Kemudian eksekusi file seeder yang telah dibuat dengan “php artisan db:seed” pada cmd



```
C:\Users\Lenovo\OneDrive\Documents\KULIAH\Pemrograman Framework\Modul 05\ujicoba>php artisan db:seed

INFO Seeding database.

Database\Seeders\PositionSeeder ..... RUNNING
Database\Seeders\PositionSeeder ..... 65 ms DONE

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

Menampilkan List Data dari Database

1. Pertama buat sebuah SQL QUERY pada file EmployeeController. Pada bagian function index tambahkan variable employees yang berisi data dari MySQL (lane 18-22). Kemudian import untuk databasenya (lane 7). Tambahkan juga variable passing data dari employee controller ke view (lane 24)

```
PositionSeeder.php U EmployeeSeeder.php U 2024_04_01_043507_create_employees_table.php U EmployeeController.php U x
Modul 05 > ujicoba > app > Http > Controllers > EmployeeController.php
1 <?php
2
3 namespace App\Http\Controllers;
4
5 use Illuminate\Http\Request;
6 use Illuminate\Support\Facades\Validator;
7 use Illuminate\Support\Facades\DB;
8
9 class EmployeeController extends Controller
10 {
11     /**
12      * Display a listing of the resource.
13      */
14     public function index()
15     {
16         $pageTitle = 'Employee List';
17
18         // RAW SQL QUERY
19         $employees = DB::select('
20 select *, employees.id as employee_id, positions.name as position_name
21 from employees left join positions on employees.position_id = positions.id
22 ');
23
24         return view('employee.index', ['pageTitle' => $pageTitle, 'employees' => $employees]);
25
26     }
27 }
```

2. Selanjutnya ubah file view untuk employee list yang awalnya masih statis diubah menjadi connect database MySQL untuk datanya (lane 64-97). Pada tag <tbody> merupakan data employee yang sebelumnya masih statis atau berdasarkan code sekarang diubah berdasarkan database, terdapat @foreach yang berguna untuk setiap employee yang terdapat pada tabel employees akan ditampilkan pada halaman index. Lalu pada lane 65-70 merupakan pemanggilan datanya mulai dari nama employee sampai nama posisi










```
EmployeeSeeder.php U EmployeeController.php U index.blade.php U x
php > html > body > div.container.mt-4 > div.table-responsive.border.p-3.rounded-3 > table.table.table-bordered.table-hover.table-striped.mb-0.bg-white > tbody
63 <tbody>
64     @foreach ($employees as $employee)
65     <tr>
66         <td>{{ $employee->firstname }}</td>
67         <td>{{ $employee->lastname }}</td>
68         <td>{{ $employee->email }}</td>
69         <td>{{ $employee->age }}</td>
70         <td>{{ $employee->position_name }}</td>
71     </tr>
72     <div class="d-flex">
73         <a href="{{ route('employees.show', ['employee' => $employee->employee_id]) }}"
74             class="btn btn-outline-dark btn-sm
75             me-2"><i
76                 class="bi-person-lines-fill"></i></a>
77         <a href="{{ route('employees.edit', ['employee' => $employee->employee_id]) }}"
78             class="btn btn-outline-dark btn-sm
79             me-2"><i
80                 class="bi-pencil-square"></i></a>
81     </div>
82     <form
83         action="{{ route('employees.destroy', ['employee' =>
84             $employee->employee_id]) }}"
85         method="POST">
86         @csrf
87         @method('delete')
88         <button type="submit"
89             class="btn
90             btn-outline-dark btn-sm me-2"><i
91             class="bi-trash"></i></button>
```

```
91         </form>
92     </div>
93 </div>
94 </td>
95 </tr>
96 @endforeach
97 </tbody>
```

3. Hasil dari halaman employee list ketika sudah terhubung dengan MySQL

The screenshot shows a web browser at localhost:8000/employees. The application has a blue header with 'Data Master' and links to 'Home' and 'Employee List'. A 'My Profile' button is in the top right. The main content area is titled 'Employee List' and has a 'Create Employee' button. Below is a table with 5 columns: First Name, Last Name, Email, Age, and Position. It contains 3 rows of employee data, each with edit and delete icons.

First Name	Last Name	Email	Age	Position	
Rafael	Fillah	testing@gmail.com	18	Front End Developer	
Dava	Maulana	testing123@gmail.com	19	Back End Developer	
Dextara	Prespiero	testing456@gmail.com	20	System Analyst	

First Name	Last Name	Email	Age	Position	
Rafael	Fillah	testing@gmail.com	18	Front End Developer	  
Dava	Maulana	testing123@gmail.com	19	Back End Developer	  
Dextara	Prespiero	testing456@gmail.com	20	System Analyst	  

Input Data ke Database

1. Pertama buat Raw SQL Query pada method create() di dalam file EmployeeController (lane 35-36) untuk pilihan “Position” pada Form Create Employee. Kemudian Passing data tersebut dari controller ke View (lane 38)

```
EmployeeController.php U X index.blade.php U create.blade.php U
Modul 05 > ujicoba > app > Http > Controllers > EmployeeController.php
31 public function create()
32 {
33     $pageTitle = 'Create Employee';
34
35     // RAW SQL Query
36     $positions = DB::select('select * from positions');
37
38     return view('employee.create', compact('pageTitle', 'positions'));
39 }
```

2. Buat sebuah form select untuk positions. Dengan pilihan posisi sesuai dengan id yang ada pada MySQL. Disini saya hilangkan notifikasi validasi error karena untuk selection tidak ada error

```
EmployeeController.php U index.blade.php U create.blade.php U X
e > create.blade.php > html > body > div.container-sm.mt-5 > form > div.row.justify-content-center > div.p-5.bg-light.rounded-3.border.col-xl-6 > div.row
74 </div>
75 @enderror
76 </div>
77 <div class="col-md-6 mb-3">
78     <label for="age" class="form-label">Age</label>
79     <input class="form-control @error('age') is-invalid @enderror" type="text" name="age"
80         id="age" value="{{ old('age') }}" placeholder="Enter Age">
81     @error('age')
82     <div class="invalid-feedback">
83         {{ $message }}
84     </div>
85     @enderror
86 </div>
87 <div class="col-md-12 mb-3">
88     <label for="position" class="form-label">Position</label>
89     <select name="position" id="position" class="form-select">
90         @foreach ($positions as $position)
91             <option value="{{ $position->id }}"
92                 {{ old('position') == $position->id ? 'selected' : '' }}>
93                 {{ $position->code . ' - ' . $position->name }}
94             </option>
95         @endforeach
96     </select>
97 </div>
```

3. Kemudian buat Query untuk insert value form kedalam database MySQL (lane 62-70)

```
EmployeeController.php U X index.blade.php U create.blade.php U
Modul 05 > ujicoba > app > Http > Controllers > EmployeeController.php
43  */
44  public function store(Request $request)
45  {
46      $messages = [
47          'required' => ':Attribute harus diisi.',
48          'email' => 'Isi :attribute dengan format yang benar',
49          'numeric' => 'Isi :attribute dengan angka'
50      ];
51      $validator = Validator::make($request->all(), [
52          'firstName' => 'required',
53          'lastName' => 'required',
54          'email' => 'required|email',
55          'age' => 'required|numeric',
56      ], $messages);
57      if ($validator->fails()) {
58          return redirect()->back()->withErrors($validator)->withInput();
59      }
60
61      // INSERT QUERY
62      DB::table('employees')->insert([
63          'firstname' => $request->firstName,
64          'lastname' => $request->lastName,
65          'email' => $request->email,
66          'age' => $request->age,
67          'position_id' => $request->position,
68      ]);
69
70      return redirect()->route('employees.index');
71  }
```

4. Hasil akhir halaman create employee



Create Employee

First Name

Last Name

Email

Age

Position

FE - Front End Developer

FE - Front End Developer

BE - Back End Developer

SA - System Analyst

FS - Full Stack Developer

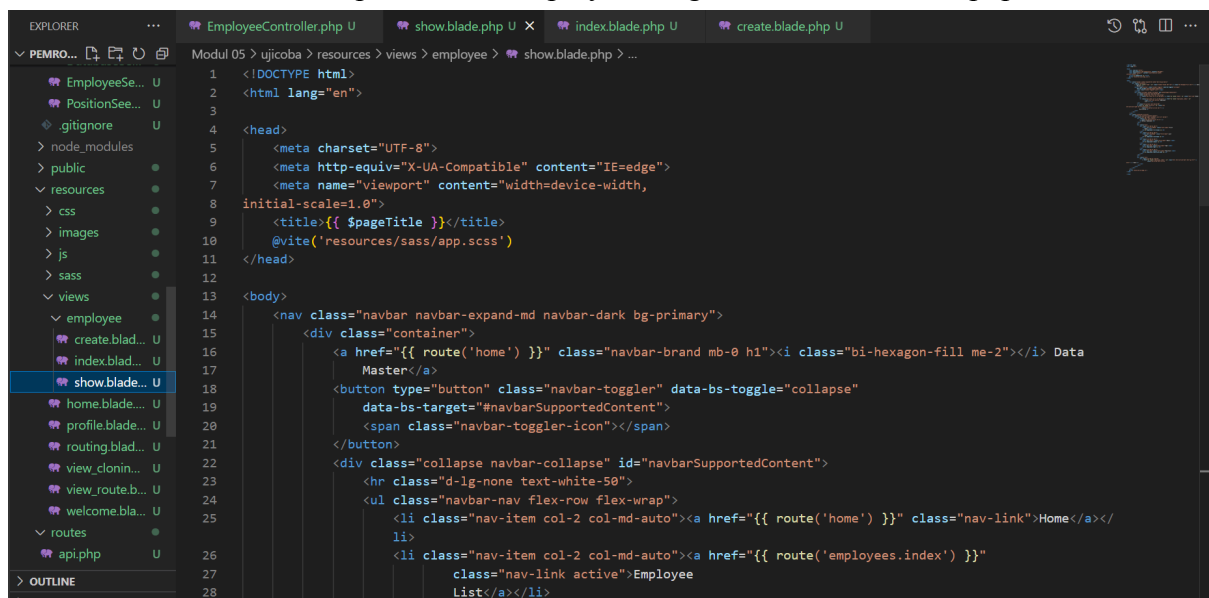
Menampilkan Detail Data dari Database

1. Buat Raw SQL Query pada method show() di dalam file EmployeeController dan Passing data employee dari controller ke View



```
EmployeeController.php U x index.blade.php U create.blade.php U
Modul 05 > ujicoba > app > Http > Controllers > EmployeeController.php
77 public function show(string $id)
78 {
79     $pageTitle = 'Employee Detail';
80
81     // RAW SQL QUERY
82     $employee = collect(DB::select('
83     select *, employees.id as employee_id, positions.name as position_name
84     from employees left join positions on employees.position_id = positions.id
85     where employees.id = ? ',
86     [$id]))->first();
87
88     return view('employee.show', compact('pageTitle', 'employee'));
89 }
```

2. Buat halaman view baru pada folder employee dengan nama show.blade.php

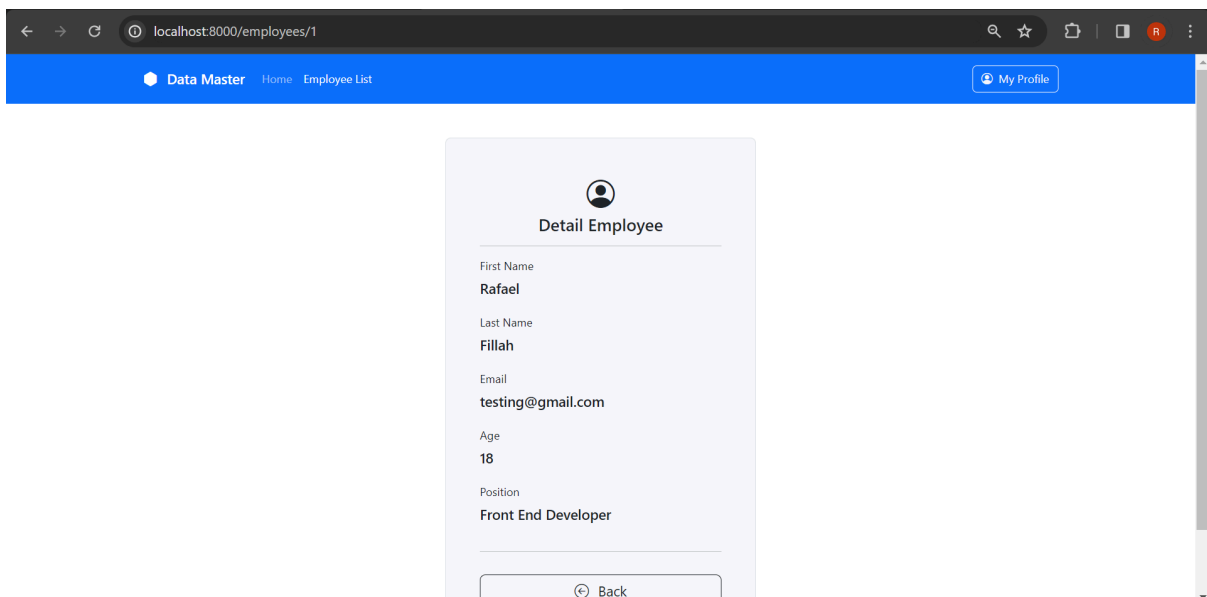


```
EXPLORER ... EmployeeController.php U show.blade.php U x index.blade.php U create.blade.php U
Modul 05 > ujicoba > resources > views > employee > show.blade.php > ...
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5     <meta charset="UTF-8">
6     <meta http-equiv="X-UA-Compatible" content="IE=edge">
7     <meta name="viewport" content="width=device-width,
8     initial-scale=1.0">
9     <title>{{ $pageTitle }}</title>
10     @vite('resources/sass/app.scss')
11 </head>
12
13 <body>
14     <nav class="navbar navbar-expand-md navbar-dark bg-primary">
15         <div class="container">
16             <a href="{{ route('home') }}" class="navbar-brand mb-0 h1"><i class="bi-hexagon-fill me-2"></i> Data
17             Master</a>
18             <button type="button" class="navbar-toggler" data-bs-toggle="collapse"
19             data-bs-target="#navbarSupportedContent">
20                 <span class="navbar-toggler-icon"></span>
21             </button>
22             <div class="collapse navbar-collapse" id="navbarSupportedContent">
23                 <hr class="d-lg-none text-white-50">
24                 <ul class="navbar-nav flex-row flex-wrap">
25                     <li class="nav-item col-2 col-md-auto"><a href="{{ route('home') }}" class="nav-link">Home</a></li>
26                     <li class="nav-item col-2 col-md-auto"><a href="{{ route('employees.index') }}"
27                         class="nav-link active">Employee
28                         List</a></li>
29                 </ul>
30             </div>
31         </div>
32     </nav>
33
34     <div class="container">
35         <div class="card">
36             <div class="card-body">
37                 <table border="1">
38                     <thead>
39                         <tr>
40                             <th>ID</th>
41                             <th>Name</th>
42                             <th>Position</th>
43                         </tr>
44                     </thead>
45                     <tbody>
46                         <tr>
47                             <td>1</td>
48                             <td>John Doe</td>
49                             <td>Software Engineer</td>
50                         </tr>
51                         <tr>
52                             <td>2</td>
53                             <td>Jane Smith</td>
54                             <td>Product Manager</td>
55                         </tr>
56                         <tr>
57                             <td>3</td>
58                             <td>Mike Johnson</td>
59                             <td>Data Analyst</td>
60                         </tr>
61                     </tbody>
62                 </table>
63             </div>
64         </div>
65     </div>
66 </body>
67 </html>
```

```
EXPLORER ... EmployeeController.php U show.blade.php U X index.blade.php U create.blade.php U
Modul 05 > ujicoba > resources > views > employee > show.blade.php > ...
EmployeeSe... U 29
PositionSee... U 30
.gitignore U 31
node_modules 32
public 33
resources 34
css 35
images 36
js 37
sass 38
views 39
employee 40
create.blad... U 41
index.blad... U 42
show.blade... U 43
home.blade... U 44
profile.blade... U 45
routing.blad... U 46
view_clonin... U 47
view_route.b... U 48
welcome.bla... U 49
routes 50
api.php U 51
OUTLINE 52
TIMELINE 53
29 </ul>
30 <hr class="d-lg-none text-white-50">
31 <a href="{{ route('profile') }}" class="btn
32 btn-outline-light my-2 ms-md-auto"><i
33 class="bi-person-circle me-1"></i>
34 My Profile</a>
35 </div>
36 </div>
37 </nav>
38 <div class="container-sm my-5">
39 <div class="row justify-content-center">
40 <div class="p-5 bg-light rounded-3 col-xl-4 border">
41 <div class="mb-3 text-center">
42 <i class="bi-person-circle fs-1"></i>
43 <h4>Detail Employee</h4>
44 </div>
45 <hr>
46 <div class="row">
47 <div class="col-md-12 mb-3">
48 <label for="firstName" class="form-label">First
49 Name</label>
50 <h5>{{ $Employee->firstname }}</h5>
51 </div>
52 <div class="col-md-12 mb-3">
53 <label for="lastName" class="form-label">Last
54 Name</label>
55 <h5>{{ $Employee->lastname }}</h5>
56 </div>
57 <div class="col-md-12 mb-3">
```


```
EXPLORER ... EmployeeController.php U show.blade.php U X index.blade.php U create.blade.php U
Modul 05 > ujicoba > resources > views > employee > show.blade.php > ...
EmployeeSe... U 58
PositionSee... U 59
.gitignore U 60
node_modules 61
public 62
resources 63
css 64
images 65
js 66
sass 67
views 68
employee 69
create.blad... U 70
index.blad... U 71
show.blade... U 72
home.blade... U 73
profile.blade... U 74
routing.blad... U 75
view_clonin... U 76
view_route.b... U 77
welcome.bla... U 78
routes 79
api.php U 80
OUTLINE 81
TIMELINE 82
58 <label for="email" class="form-label">Email</label>
59 <h5>{{ $Employee->email }}</h5>
60 </div>
61 <div class="col-md-12 mb-3">
62 <label for="age" class="form-label">Age</label>
63 <h5>{{ $Employee->age }}</h5>
64 </div>
65 <div class="col-md-12 mb-3">
66 <label for="age" class="form-label">Position</label>
67 <h5>{{ $Employee->position_name }}</h5>
68 </div>
69 <hr>
70 <div class="row">
71 <div class="col-md-12 d-grid">
72 <a href="{{ route('employees.index') }}" class="btn btn-outline-dark btn-lg mt-3"><i
73 class="bi-arrow-left-circle
74 me-2"></i> Back</a>
75 </div>
76 </div>
77 </div>
78 </div>
79 </div>
80 @vite('resources/js/app.js')
81 </body>
82 </html>
83
84
85
```

3. Hasil akhir halaman detail employee



Menghapus Data dari Database

1. Buat sebuah query Builder untuk menghapus employee, pada file EmployeeController bagian function destroy



```
109  */
110  public function destroy(string $id)
111  {
112      // QUERY BUILDER
113      DB::table('employees')
114          ->where('id', $id)
115          ->delete();
116
117      return redirect()->route('employees.index');
118  }
119
120 }
121
```


Membuat Fitur Edit Employee

1. Pertama buat sebuah route untuk mengarah ke halaman edit pada file web.php

```
EmployeeController.php U  edit.blade.php U  web.php U x  show.blade.php U  index.blade.php U  create.blade.php U  ...
Modul 05 > ujicoba > routes > web.php
83
84 // Modul 05
85 Route::get('/employees/{id}/edit', [EmployeeController::class, 'edit'])->name('employees.edit');
86
```

2. buat sebuah logic untuk melakukan edit pada function edit by id EmployeeController

```
EmployeeController.php U x  edit.blade.php U  web.php U  show.blade.php U  index.blade.php U  create.blade.php U  ...
Modul 05 > ujicoba > app > Http > Controllers > EmployeeController.php
94 public function edit(string $id)
95 {
96     $pageTitle = 'Edit Employee';
97     $employee = DB::table('employees')->where('id', $id)->first();
98     $positions = DB::table('positions')->get();
99
100     return view('employee.edit', compact('pageTitle', 'employee', 'positions'));
101 }
102
```

3. Kemudian buat logic untuk update value dari employee, sertakan validationnya dan message yang akan ditampilkan

```
EmployeeController.php U x  edit.blade.php U  web.php U  show.blade.php U  index.blade.php U  create.blade.php U  ...
Modul 05 > ujicoba > app > Http > Controllers > EmployeeController.php
106 public function update(Request $request, string $id)
107 {
108     $messages = [
109         'required' => ':Attribute harus diisi.',
110         'email' => 'Isi attribute dengan format yang benar',
111         'numeric' => 'Isi attribute dengan angka'
112     ];
113     $validator = Validator::make($request->all(), [
114         'firstName' => 'required',
115         'lastName' => 'required',
116         'email' => 'required|email',
117         'age' => 'required|numeric',
118     ], $messages);
119
120     if ($validator->fails()) {
121         return redirect()->back()->withErrors($validator)->withInput();
122     }
123
124     // UPDATE QUERY
125     DB::table('employees')->where('id', $id)->update([
126         'firstname' => $request->firstName,
127         'lastname' => $request->lastName,
128         'email' => $request->email,
129         'age' => $request->age,
130         'position_id' => $request->position,
131     ]);
132
133     return redirect()->route('employees.index');
134 }
```

4. Selanjutnya buat sebuah halaman view untuk fitur edit, disini saya menyamakan dengan fitur create employee. Terdapat beberapa bagian yang harus diubah dari halaman create menjadi halaman edit, yang paling utama adalah isi value dari setiap form input, judul halaman, dan route form yang diubah menjadi employees.update

```
EmployeeController.php U edit.blade.php X web.php U show.blade.php U index.blade.php U create.blade.php U
Modul 05 > ujicoba > resources > views > employee > edit.blade.php > html > body > div.container-sm.mt-5 > form > div.row.justify-content-center > div.p-5
1 <!DOCTYPE html>
2 <html lang="en">
3
4 <head>
5 <meta charset="UTF-8">
6 <meta http-equiv="X-UA-Compatible" content="IE=edge">
7 <meta name="viewport" content="width=device-width, initial-scale=1.0">
8 <title>{{ $pageTitle }}</title>
9 @vite('resources/sass/app.scss')
10 </head>
11
12 <body>
13 <nav class="navbar navbar-expand-md navbar-dark bg-primary">
14 <div class="container">
15 <a href="{{ route('home') }}" class="navbar-brand mb-0 h1"><i class="bi-hexagon-fill me-2"></i> Data
16 Master</a>
17 <button type="button" class="navbar-toggler" data-bs-toggle="collapse"
18 data-bs-target="#navbarSupportedContent">
19 <span class="navbar-toggler-icon"></span>
20 </button>
21 <div class="collapse navbar-collapse" id="navbarSupportedContent">
22 <hr class="d-lg-none text-white-50">
23 <ul class="navbar-nav flex-row flex-wrap">
24 <li class="nav-item col-2 col-md-auto"><a href="{{ route('home') }}" class="nav-link">Home</a></li>
25 <li class="nav-item col-2 col-md-auto"><a href="{{ route('employees.index') }}"
26 class="nav-link">Employee List</a></li>
27 </ul>
28 <hr class="d-lg-none text-white-50">
29 <a href="{{ route('profile') }}" class="btn btn-outline-light my-2 ms-md-auto"><i
30 class="bi-person-circle me-1"></i> My Profile</a>
31 </div>
32 </div>
33 </nav>
34 <div class="container-sm mt-5">
35 <form action="{{ route('employees.update', $employee->id) }}" method="POST">
36 @csrf
37 @method('PUT')
38 <div class="row justify-content-center">
39 <div class="p-5 bg-light rounded-3 border col-xl-6">
40 <div class="mb-3 text-center">
41 <i class="bi-person-circle fs-1"></i>
42 <h4>Edit Employee</h4>
43 </div>
44 <hr>
45 <div class="row">
46 <div class="col-md-6 mb-3">
47 <label for="firstName" class="form-label">First Name</label>
48 <input class="form-control @error('firstName') is-invalid @enderror" type="text"
49 name="firstName" id="firstName" value="{{ $employee->firstname }}"
50 placeholder="Enter First Name">
51 @error('firstName')
52 <div class="invalid-feedback">
53 {{ $message }}
54 </div>
55 @enderror
56 </div>
57 <div class="col-md-6 mb-3">
```

```

58     <label for="lastName" class="form-label">Last Name</label>
59     <input class="form-control @error('lastName') is-invalid @enderror" type="text"
60         name="lastName" id="lastName" value="{{ $employee->firstname }}"
61         placeholder="Enter Last Name">
62     @error('lastName')
63         <div class="invalid-feedback">
64             {{ $message }}
65         </div>
66     @enderror
67 </div>
68 <div class="col-md-6 mb-3">
69     <label for="email" class="form-label">Email</label>
70     <input class="form-control @error('email') is-invalid @enderror" type="text"
71         name="email" id="email" value="{{ $employee->email }}" placeholder="Enter Email">
72     @error('email')
73         <div class="invalid-feedback">
74             {{ $message }}
75         </div>
76     @enderror
77 </div>
78 <div class="col-md-6 mb-3">
79     <label for="age" class="form-label">Age</label>
80     <input class="form-control @error('age') is-invalid @enderror" type="text" name="age"
81         id="age" value="{{ $employee->age }}" placeholder="Enter Age">
82     @error('age')
83         <div class="invalid-feedback">
84             {{ $message }}
85         </div>
86     @enderror

```

```

87 </div>
88 <div class="col-md-12 mb-3">
89     <label for="position" class="form-label">Position</label>
90     <select name="position" id="position" class="form-select">
91         @foreach ($positions as $position)
92             <option value="{{ $position->id }}"
93                 {{ old('position') == $position->id ? 'selected' : '' }}>
94                 {{ $position->code . ' - ' . $position->name }}
95             </option>
96         @endforeach
97     </select>
98 </div>
99
100 </div>
101 <hr>
102 <div class="row">
103     <div class="col-md-6 d-grid">
104         <a href="{{ route('employees.index') }}" class="btn btn-outline-dark btn-lg mt-3"><i
105             class="bi-arrow-left-circle me-2"></i> Cancel</a>
106     </div>
107     <div class="col-md-6 d-grid">
108         <button type="submit" class="btn btn-dark btn-lg mt-3"><i class="bi-check-circle
109             me-2"></i> Save</button>
110     </div>
111 </div>
112 </div>
113 </div>
114 </form>

```


```

115 </div>
116 @vite('resources/js/app.js')
117 </body>
118
119 </html>
120

```

5. Hasil akhir halaman edit





Edit Employee

First Name	Last Name
<input type="text" value="Rafael"/>	<input type="text" value="Rafael"/>
Email	Age
<input type="text" value="testing@gmail.com"/>	<input type="text" value="18"/>
Position	
<input type="text" value="FE - Front End Developer"/>	

Mengubah Raw SQL Query Menjadi Query Builder

1. Perubahan pada function index

```
EmployeeController.php U X edit.blade.php U index.blade.php U
Modul 05 > ujicoba > app > Http > Controllers > EmployeeController.php

14 public function index()
15 {
16     $pageTitle = 'Employee List';
17
18     // RAW SQL QUERY
19     // $employees = DB::select('
20     // select *, employees.id as employee_id, positions.name as position_name
21     // from employees left join positions on employees.position_id = positions.id
22     // ');
23
24     // QUERY BUILDER
25     $employees = DB::table('employees')
26         ->select('employees.*', 'positions.name as position_name')
27         ->leftJoin('positions', 'employees.position_id', '=', 'positions.id')
28         ->get();
29
30     return view('employee.index', ['pageTitle' => $pageTitle, 'employees' => $employees]);
31 }
32
```

2. Perubahan pada function create

```
37 public function create()
38 {
39     $pageTitle = 'Create Employee';
40
41     // RAW SQL Query
42     // $positions = DB::select('select * from positions');
43
44     // QUERY BUILDER
45     $positions = DB::table('positions')->get();
46
47     return view('employee.create', compact('pageTitle', 'positions'));
48 }
```

3. Perubahan pada function show

```
86 public function show(string $id)
87 {
88     $pageTitle = 'Employee Detail';
89
90     // RAW SQL QUERY
91     // $employee = collect(DB::select('
92     // select *, employees.id as employee_id, positions.name as position_name
93     // from employees left join positions on employees.position_id = positions.id
94     // where employees.id = ? ',
95     // [$id]))->first();
96
97     // QUERY BUILDER
98     $employee = DB::table('employees')
99         ->select('employees.*', 'positions.name as position_name')
100         ->leftJoin('positions', 'employees.position_id', '=', 'positions.id')
101         ->where('employees.id', $id)
102         ->first();
103
104     return view('employee.show', compact('pageTitle', 'employee'));
105 }
```