

---

# Praktikum 8

---

## Gambaran Umum

Mempelajari dasar pemrograman web menggunakan Laravel  
Materi meliputi relasi database

## Capaian Pembelajaran

1. Mahasiswa mampu memahami pembuatan relasi database
2. Mahasiswa mampu memahami pembuatan authentication
3. Mahasiswa mampu memahami pembuatan middleware

## Kegiatan

1. Mahasiswa mengerjakan semua contoh yang diberikan dalam modul
2. Mahasiswa mengerjakan tugas
3. Mahasiswa membuat laporan

## Evaluasi

1. Penilaian kehadiran mahasiswa pada saat pelaksanaan pembelajaran berdasarkan waktu yang telah ditentukan
2. Penilaian kemampuan mahasiswa dinilai dari tugas dan laporan yang dikumpulkan sesuai batas pengumpulan laporan yang telah dilakukan
3. Mahasiswa yang tidak hadir dan tidak mengerjakan tugas beserta laporan tidak akan mendapat nilai
4. Tidak ada pengumpulan susulan laporan bagi mahasiswa yang tidak mengumpulkan berdasarkan waktu yang telah ditentukan

Relasi Database

Authentication

Middleware

28 Nopember – 1 Desember 2022

---

## Alat dan Bahan

1. Aplikasi editor
2. Web browser
3. Web server

## Referensi

1. <https://git-scm.com/downloads>
2. <https://laravel.com>
3. [youtube.com](https://youtube.com)

## Catatan

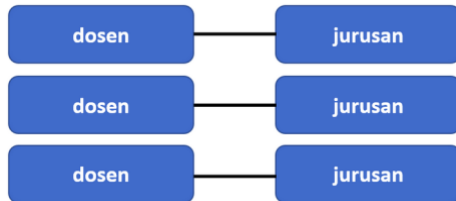
1. Mahasiswa yang mengalami kesulitan dan error pada saat mengerjakan dianjurkan menghubungi dosen melalui WA
2. Mahasiswa tidak boleh mengcopy paste pekerjaan mahasiswa lain
3. Mahasiswa mengerjakan laporan berdasarkan template laporan yang telah disediakan

## 1. Relasi Database

Relasi antar table merupakan hubungan satu table dengan table yang lainnya pada database. Jenis dari relasi yang digunakan disesuaikan dengan kebutuhan pada aplikasi yang dibangun. Pada eloquent (itir Laravel yang dapat dimanfaatkan untuk mengakses dan memanipulasi data yang tersimpan di dalam database dengan perintah yang lebih singkat ) terdapat 3 macam relasi table, yaitu:

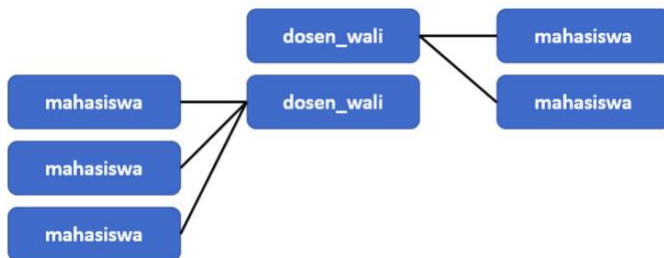
### a. One to One

Relasi yang mana setiap satu baris data pada tabel pertama hanya berhubungan dengan satu baris pada tabel kedua. Pada eloquent relasi ini menggunakan menggunakan method `hasOne()` pada class model untuk table pertaman dan terdapat method `belongsTo()` pada class model untuk table kedua.



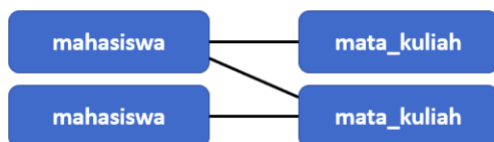
### b. One to Many

Relasi yang mana setiap satu baris data pada tabel pertama berhubungan dengan lebih dari satu baris pada tabel kedua. Pada eloquent relasi ini menggunakan menggunakan method `hasMany()` pada class model. Selanjutnya terdapat **One To Many (Inverse)** menggunakan method `belongsTo()` yang merupakan kebalikan dari `hasMany()`, di mana relasi dibuat pada model class child yang dapat memanggil dari parent.




### c. Many to Many

Relasi *Many to Many* adalah relasi yang mana setiap lebih dari satu baris data dari tabel pertama berhubungan dengan lebih dari satu baris data pada tabel kedua. Artinya, kedua tabel masing-masing dapat mengakses banyak data dari tabel yang direlasikan.



Pada praktikum sebelumnya kita telah mempunyai table produk. Pada praktikum relasi, kita akan menghubungkan table produk dengan table baru yaitu kategori. Oleh karena itu, sebelum melakukan relasi buatlah table kategori dengan menggunakan artisan, dengan struktur sebagai berikut:

#	Name	Type
1	id 	bigint(20)
2	nama_kategori	varchar(255)
3	created_at	timestamp
4	updated_at	timestamp

Kemudian isi data kategori secara manual seperti berikut:

id	nama_kategori
1	Smartphone
2	Accesories
3	Notebook
4	Personal Computer

Dari table produk dan kategori akan dilakukan relasi, dalam hal ini relasi yang dimiliki oleh dua table tersebut adalah one to many, 1 kategori mempunyai banyak produk.

Untuk membuat relasi produk dan kategori, ikuti langkah berikut ini:

- 1) Buat migrasi untuk mengubah table produk, dengan perintah **php artisan make:migration tambah\_kolom\_tabel\_produk**
- 2) Buka file hasil migrasi, tambahkan kode pada method up seperti berikut:

```
migrations > 2022_11_27_102550_tambah_kolom_tabel_produk.php
| */
| public function up()
| {
|     Schema::table('produk', function(Blueprint $table){
|         $table->string('kategori_id')->after('nama_produk');
|     });
| }
```

- 3) Lakukan migrasi dan cek pada database manager apakah migrasi sudah berhasil.

#	Name	Type
1	id 	bigint(20)
2	nama_produk	varchar(255)
3	kategori_id	varchar(255)
4	harga	int(11)
5	stock	varchar(255)
6	created_at	timestamp
7	updated_at	timestamp

- 4) Tambahkan data pada kategori\_id, dengan angka-angka yang disimpan pada table kategori.

## 5) Tambahkan kolom untuk kategori pada **views/produk/index.blade.php**

```
resources > views > produk > index.blade.php
13 <h2>Tabel Data Produk</h2>
14 <a href="{{route('produk.create')}}" class="btn btn-success">+Tambah Data</a>
15 <table class="table table-bordered table-striped" id="tabel">
16     <thead>
17         <tr>
18             <th style="width:5%">No.</th>
19             <th style="width:7%">Kode Produk</th>
20             <th style="width:12%">Nama Produk</th>
21             <th style="width:12%">Kategori</th>
22             <th style="width:12%">Harga</th>
23             <th style="width:8%">Stok</th>
24             <th style="width:8%">Aksi</th>
25         </tr>
26     </thead>
27     <tbody>
28         @foreach ($dataProduk as $data)
29             <tr>
30                 <td {{ $loop->iteration }} </td>
31                 <td {{ $data->id }} </td>
32                 <td {{ $data->nama_produk }} </td>
33                 <td {{ $data->kategori_id }} </td>
34                 <td {{ number_format($data->narga, 0, ',', '.') }} </td>
35                 <td {{ $data->stock }} </td>
```

Jalankan url 127.0.0.1:8000/tampil-produk, akan tampil hasil sementara seperti berikut:

**Tabel Data Produk**

+Tambah Data

Show 10 entries Search:

No.	Kode Produk	Nama Produk	Kategori	Harga	Stok	Aksi
1	2189	Samsung Galaxy A53	1	5.799.000	1	<button>Edit</button> <button>Delete</button>
2	3210	Iphone 13 Pro 256 GB	1	18.000.000	3	<button>Edit</button> <button>Delete</button>

## 6) Tambahkan function produk() seperti berikut pada **app/Models/Kategori.php**

```
app > Models > Kategori.php
1 <?php
2
3 namespace App\Models;
4
5 use Illuminate\Database\Eloquent\Factories\HasFactory;
6 use Illuminate\Database\Eloquent\Model;
7
8 class Kategori extends Model
9 {
10     use HasFactory;
11     protected $table = 'Kategori';
12     protected $primaryKey = 'id';
13     protected $fillable = ['id', 'nama_kategori'];
14
15     public function produk(){
16         return $this->hasMany(Produk::class);
17     }
18 }
```

- **Baris 15**, merupakan method yang menyatakan bahwa kategori mempunyai relasi one to many dengan table produk

7) Tambahkan function kategori() seperti berikut pada **app/Models/Produk.php**

```
app > Models > Produk.php
1  <?php
2
3  namespace App\Models;
4
5  use Illuminate\Database\Eloquent\Factories\HasFactory;
6  use Illuminate\Database\Eloquent\Model;
7
8  class Produk extends Model
9  {
10     use HasFactory;
11     protected $table = 'Produk';
12     protected $primaryKey = 'id';
13     protected $fillable = ['id', 'nama_produk', 'harga', 'stock'];
14
15     public function kategori(){
16         return $this->belongsTo(Kategori::class, 'kategori_id');
17     }
18 }
```

- Baris 15, belongsTo digunakan untuk invers relasi one to many oleh class Kategori dengan menyebutkan foreign key yang digunakan untuk relasi

8) Coba cek hasil relasi produk dan kategori dengan menggunakan perintah pada terminal **php artisan tinker**

```
Kuntis-Macbook:Project macbookair$ php artisan tinker
Psy Shell v0.10.9 (PHP 8.0.12 - cli) by Justin Hileman
>>> $produk = Produk::first()
[!] Aliasing 'Produk' to 'App\Models\Produk' for this Tinker session.
=> App\Models\Produk {#3538
    id: 2189,
    nama_produk: "Samsung Galaxy A53",
    kategori_id: "1",
    harga: 5799000,
    stock: "1",
    created_at: null,
    updated_at: null,
}
>>> $produk->kategori
=> App\Models\Kategori {#3542
    id: 1,
    nama_kategori: "Smartphone",
    created_at: null,
    updated_at: null,
}_
```

9) Ubah kode kolom kategori pada **views/produk/index.blade.php**

```
resources > views > produk > index.blade.php
28  @foreach ($dataProduk as $data)
29      <tr>
30          <td> {{ $loop->iteration }} </td>
31          <td> {{ $data->id }} </td>
32          <td> {{ $data->nama_produk }} </td>
33          <td> {{ $data->kategori->nama_kategori }} </td>
34          <td> {{ number_format($data->harga, 0, ',', '.') }} </td>
35          <td> {{ $data->stock }} </td>
```

- Baris 33, kategori merupakan function kategori() pada model Produk, nama\_kategori merupakan nama kolom dari table kategori yang datanya ingin kita tampilkan

## 2. Authentication

Authentication adalah proses dimana seorang user (melalui berbagai macam akses fisik berupa komputer , melalui jaringan , atau melalui remote access ) mendapatkan hak akses kepada suatu entity.

Berikut cara menampilkan form login pada Laravel seperti tampilan di atas:

- Pastikan di computer sudah terinstal nodejs, jika belum maka download terlebih dahulu di <https://nodejs.org/en/> , kemudian install

- Buka terminal, install package laravel/ui dengan menjalankan perintah composer berikut:

```
composer require laravel/ui (untuk Laravel 9 ke atas)
```

```
composer require laravel/ui:^3.1.4 (untuk Laravel 8)
```

- Menerapkan bootstrap pada Laravel dengan perintah berikut:

```
php artisan ui bootstrap tunggu hingga selesai
```

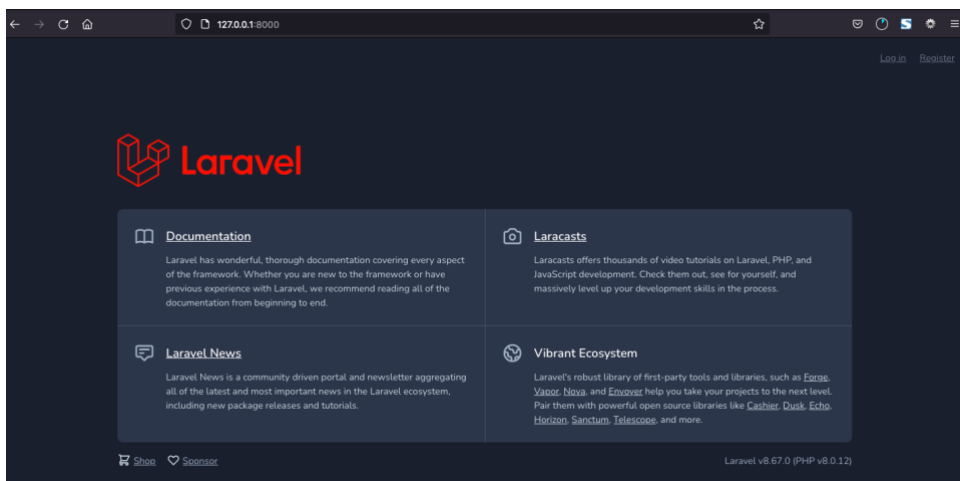
- Lalu saatnya kita terapkan bootstrap, di mana dengan perintah di bawah ini akan mengupdate file public/css/app.css dan public/js/app.js.

```
npm install && npm run dev tunggu hingga selesai
```

- Selanjutnya adalah bagian inti, yaitu membuat autentikasi login di laravel.

```
php artisan ui bootstrap --auth
```

- Jalankan <https://127.0.0.1:8000/>



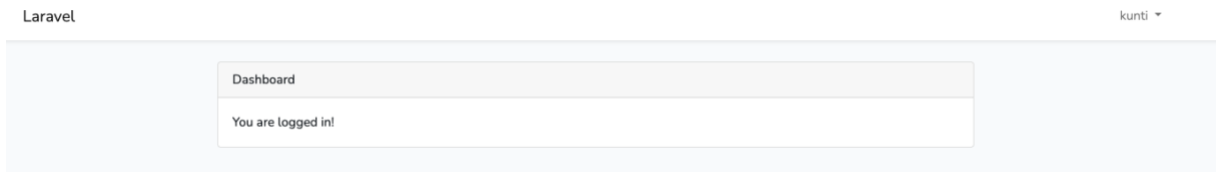
- Jika dalam database db\_project belum terdapat table user, maka buat table user dengan menggunakan migration. Dengan struktur skema seperti berikut:

```

public function up()
{
    //
    Schema::create('users', function (Blueprint $table) {
        $table->bigIncrements('id');
        $table->string('name');
        $table->string('email')->unique();
        $table->timestamp('email_verified_at')->nullable();
        $table->string('password');
        $table->timestamps();
    });
}

```

- h. Kemudian cobalah untuk melakukan register pada tombol register, kemudian coba untuk login. Jika berhasil login maka akan tampil seperti berikut:



### 3. Menggabungkan Halaman Login Laravel dengan Navbar

Ikuti langkah-langkah berikut:

- 1) Buka file [home.blade.php](#) pada folder **resources/views/** kemudian ubah kode:

@extends('layouts.app') menjadi :

```
resources > views > home.blade.php
1 @extends('layouts.master')
```

- 2) Buka file **resources/views/layouts/app.blade.php**, copy code pada baris 33-72, kemudian paste ke file **resources/views/layouts/navbar.blade.php**.

Keseluruhan dari file navbar.blade.php menjadi seperti berikut:

```
resources > views > layouts > navbar.blade.php
1 <nav class="navbar navbar-expand-lg navbar-dark bg-info">
2 <div class="container-fluid">
3 <a class="navbar-brand" href="#">Toko Komputer</a>
4 <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNav" aria-controls="navbarNav" aria-expanded="false" aria-label="Toggle navigation">
5 <span class="navbar-toggler-icon"></span>
6 </button>
7 <div class="collapse navbar-collapse" id="navbarNav">
8 <ul class="navbar-nav">
9 <li class="nav-item">
10 <a class="nav-link active" aria-current="page" href="{{url('home')}}">Home</a>
11 </li>
12 <li class="nav-item">
13 <a class="nav-link" href="{{url('barang')}}">Barang</a>
14 </li>
15 <li class="nav-item">
16 <a class="nav-link" href="{{url('kategori')}}">Kategori</a>
17 </li>
18 <li class="nav-item">
19 <a class="nav-link" href="{{url('transaksi')}}">Transaksi</a>
20 </li>
21 </ul>
22 </div>
23
24
25 <div class="collapse navbar-collapse" id="navbarSupportedContent">
26 <!-- Left Side Of Navbar -->
27 <ul class="navbar-nav me-auto">
28
29 </ul>
30
31 <!-- Right Side Of Navbar -->
32 <ul class="navbar-nav ms-auto">
33 <!-- Authentication Links -->
34 @guest
35 @if (Route::has('login'))
36 <li class="nav-item">
37 <a class="nav-link" href="{{ route('login') }}">{{ __('Login') }}</a>
38 </li>
39 @endif
40
41 @if (Route::has('register'))
42 <li class="nav-item">
43 <a class="nav-link" href="{{ route('register') }}">{{ __('Register') }}</a>
44 </li>
45 @endif
46 @else
47 <li class="nav-item dropdown">
48 <a id="navbarDropdown" class="nav-link dropdown-toggle" href="#" role="button" data-bs-toggle="dropdown" aria-haspopup="true" aria-expanded="false" v-
49 {{ Auth::user()->name }}
50 </a>
51
52 <div class="dropdown-menu dropdown-menu-right" aria-labelledby="navbarDropdown">
53 <a class="dropdown-item" href="{{ route('logout') }}"
54 onclick="event.preventDefault();
55 document.getElementById('logout-form').submit();"
56 {{ __('Logout') }}
57 </a>
58
59 <form id="logout-form" action="{{ route('logout') }}" method="POST" class="d-none">
60 @csrf
61 </form>
62 </div>
63 </li>
64 @endguest
65 </ul>
66 </div>
67 </nav>
```



3) Ubah routes untuk url '/dashboard' pada web.php

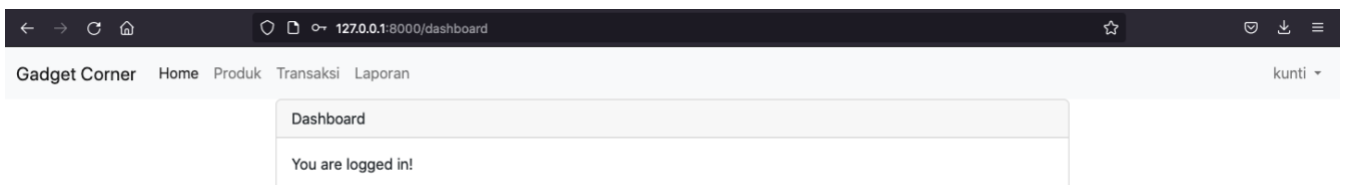
Dari semula seperti ini:

```
routes > web.php
20
21 Route::get('/dashboard', function () {
22     return view('dashboard');
23 }->middleware(['auth'])->name('dashboard');
--
```

Menjadi seperti berikut:

```
routes > web.php
20
21 Route::get('/dashboard', [App\Http\Controllers\HomeController::class, 'index'])->name('home')
22     ->middleware(['auth'])->name('dashboard');
--
```

Coba jalankan hasilnya dengan login dan apabila berhasil maka akan masuk ke halaman berikut:



#### 4. Membuat Authentication dengan Middleware

Middleware berperan sebagai filter request user/HTTP yang masuk ke aplikasi.

Sebelumnya tambahkan kolom level pada table user.

- 1) Tambahkan kolom dengan menggunakan migrasi:

```
database > migrations > 2022_11_27_233709_ubah_tabel_users.php
13  */
14  public function up()
15  {
16      //
17      Schema::table('users', function(Blueprint $table){
18          $table->string('level')->after('password');
19      });
20  }
```

- 2) Tambahkan kolom input level untuk registrasi pada **views/auth/register.blade.php**

```
resources > views > auth > register.blade.php
63
64  <div class="row mb-3">
65      <label for="password-confirm" class="col-md-4 col-form-label text-md-end">{{ __('Level') }}</label>
66
67      <div class="col-md-6">
68          <input id="level" type="text" class="form-control" name="level" >
69      </div>
70  </div>
```

- 3) Tambahkan kode berikut pada **RegisterController.php**

```
app > Http > Controllers > Auth > RegisterController.php
50  protected function validator(array $data)
51  {
52      return Validator::make($data, [
53          'name' => ['required', 'string', 'max:255'],
54          'email' => ['required', 'string', 'email', 'max:255', 'unique:users'],
55          'password' => ['required', 'string', 'min:8', 'confirmed'],
56          'level' => ['required', 'string', 'max:255'],
57      ]);
58  }
59
60  /**
61   * Create a new user instance after a valid registration.
62   *
63   * @param array $data
64   * @return \App\Models\User
65   */
66  protected function create(array $data)
67  {
68      return User::create([
69          'name' => $data['name'],
70          'email' => $data['email'],
71          'password' => Hash::make($data['password']),
72          'level' => $data['level'],
73      ]);
74  }
```

- 4) Tambahkan fillable pada model **User.php**

```
app > Models > User.php
19  */
20  protected $fillable = [
21      'name',
22      'email',
23      'password',
24      'level',
25  ];
```

- 5) Jalankan form register pada browser

Register

Name

Email Address

Password

Confirm Password

Level

Register

Nantinya akan terdapat 2 level user, yaitu kasir dan admin.

id	name	email	email_verified_at	password	level
1	kunti	kunti.class@gmail.com	NULL	\$2y\$10\$3igUJ5CxVq6fGhQLUgqv5OURXO5rCcrsWWUD4/IB9Ez...	kasir
2	lisa	lisa@gmail.com	NULL	\$2y\$10\$jgR5pURMutr/qMB2vMmhG.izd2TjEiZ9Lvptk2ulNuE...	admin

Skenario:

- Level admin dapat mengakses halaman Home, Produk dan Laporan. Admin tidak dapat mengakses halaman Transaksi.
- Level kasir dapat mengakses halaman Home dan Transaksi

Berikut langkah-langkahnya:

- Buat terlebih dahulu controller untuk kasir dan laporan, yaitu KasirController dan LaporanController
- Pada KasirController, ProdukController dan LaporanController tambahkan method construct

app > Http > Controllers > ProdukController.php

```

10 class ProdukController extends Controller
11 {
12     public function __construct(){
13         $this->middleware('admin');
14     }

```

app > Http > Controllers > KasirController.php

```

7 class KasirController extends Controller
8 {
9     //
10     public function __construct(){
11         $this->middleware('kasir');
12     }
13 }

```

app > Http > Controllers > ProdukController.php

```

9 class ProdukController extends Controller
10 {
11     public function __construct(){
12         $this->middleware('admin');
13     }
14 }

```

- Melakukan proteksi navigation bar, buka file **navbar.blade.php**

```

1 <nav class="navbar navbar-expand-lg bg-light">
2   <div class="container-fluid">
3     <a class="navbar-brand" href="#">Gadget Corner</a>
4     <button class="navbar-toggler" type="button" data-bs-toggle="collapse" data-bs-target="#navbarNavAltMarkup"
5       aria-controls="navbarNavAltMarkup" aria-expanded="false" aria-label="Toggle navigation">
6       <span class="navbar-toggler-icon"></span>
7     </button>
8     <div class="collapse navbar-collapse" id="navbarNavAltMarkup">
9       <div class="navbar-nav">
10        <a class="nav-link active" aria-current="page" href="{{url('home')}}">Home</a>
11
12        @if (Auth::check() && Auth::user()->level == 'admin')
13        <a class="nav-link" href="{{url('tampil-produk')}}">Produk</a>
14        @endif
15
16        @if (Auth::check() && Auth::user()->level == 'kasir')
17        <a class="nav-link" href="{{url('transaksi')}}">Transaksi</a>
18        @endif
19
20        @if (Auth::check() && Auth::user()->level == 'admin')
21        <a class="nav-link" href="{{url('laporan')}}">Laporan</a>
22        @endif
23      </div>

```

- Baris 12, `@if (Auth::check() && Auth::user()->level == 'admin')`, jika login sebagai admin tampilkan Produk
- d. Membuat middleware Kasir dan Admin, dengan cara masuk ke terminal kemudian tulis perintah berikut:

**php artisan make:middleware Kasir**  
**php artisan make:middleware Admin**

Buka file hasil middleware bernama **Kasir.php** dan **Admin.php** pada folder **app/Http/Middleware**. Tambahkan **use Auth;** pada awal file tersebut.

```

1 <?php
2
3 namespace App\Http\Middleware;
4
5 use Closure;
6 use Illuminate\Http\Request;
7 use Auth;
8
9 class Kasir
10 {

```

Kemudian sesuaikan kode pada method handle seperti berikut:

```
app > Http > Middleware > Kasir.php
18 public function handle(Request $request, Closure $next)
19 {
20     if(!Auth::user()->level == 'kasir'))
21     {
22         return redirect()->back();
23     }
24     {
25         return $next($request);
26     }
27 }

app > Http > Middleware > Admin.php
18 */
19 public function handle(Request $request, Closure $next, ...$level)
20 {
21
22     if(!Auth::user()->level == 'admin'))
23     {
24         return redirect()->back();
25     }
26     {
27         return $next($request);
28     }
29 }
30 }
```

- e. Daftarkan middleware yang telah dibuat pada file **Kernel.php**, agar dapat dikenali oleh Laravel. Buka **app/Http/Kernel.php**, kemudian cari bagian protected \$routeMiddleware, tambahkan kode seperti berikut:

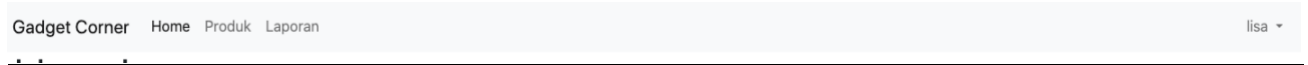
```
app > Http > Kernel.php
54 */
55 protected $routeMiddleware = [
56     'auth' => \App\Http\Middleware\Authenticate::class,
57     'auth.basic' => \Illuminate\Auth\Middleware\AuthenticateWithBasicAuth::class,
58     'bindings' => \Illuminate\Routing\Middleware\SubstituteBindings::class,
59     'cache.headers' => \Illuminate\Http\Middleware\SetCacheHeaders::class,
60     'can' => \Illuminate\Auth\Middleware\Authorize::class,
61     'guest' => \App\Http\Middleware\RedirectIfAuthenticated::class,
62     'password.confirm' => \Illuminate\Auth\Middleware\RequirePassword::class,
63     'signed' => \Illuminate\Routing\Middleware\ValidateSignature::class,
64     'throttle' => \Illuminate\Routing\Middleware\ThrottleRequests::class,
65     'verified' => \Illuminate\Auth\Middleware\EnsureEmailIsVerified::class,
66     'admin' => \App\Http\Middleware\Admin::class,
67     'kasir' => \App\Http\Middleware\Kasir::class,
```

- f. Masukkan semua route untuk produk pada group middleware pada **web.php** menjadi seperti berikut:

```
routes > web.php
47
48 //Route untuk aksi produk dengan middleware
49 Route::middleware(['auth'])->group(function () {
50
51     Route::get('/dashboard', [HomeController::class, 'index']);
52     Route::get('tampil-produk', [ProdukController::class, 'index']);
53     Route::get('tambah-produk', [ProdukController::class, 'create'])->name('produk.create');
54     Route::post('tambah-produk', [ProdukController::class, 'store'])->name('produk.store');
55     Route::get('/produk/edit/{id}', [ProdukController::class, 'edit'])->name('produk.edit');
56     Route::post('/produk/update/{id}', [ProdukController::class, 'update'])->name('produk.update');
57     Route::post('/produk/delete/{id}', [ProdukController::class, 'destroy'])->name('produk.destroy');
58     Route::get('transaksi', function () { return view('transaksi'); });
59     Route::get('laporan', function () { return view('laporan'); });
60 });
```

- g. Jalankan login dengan menggunakan level admin dan kasir, hasilnya adalah sebagai berikut:

### KASIR



### ADMIN

