Nama: Yonanda Mayla Rusdiaty

Kelas: SIB 2A/30

# JOBSHEET 07

**Authentication dan *Authorization* di Laravel**

Laravel Authentication dipergunakan untuk memproteksi halaman atau fitur dari web yang hanya diakses oleh orang tertentu yang diberikan hak. Fitur seperti ini biasanya ditemui di sistem yang memiliki fitur administrator atau sistem yang memiliki pengguna yang boleh menambahkan datanya.

Laravel membuat penerapan otentikasi sangat sederhana dan telah menyediakan berbagai fitur yang dapat dimanfaatkan tanpa perlu melakukan penambahan instalasi modul tertentu. File konfigurasi otentikasi terletak di config / auth.php, yang berisi beberapa opsi yang terdokumentasi dengan baik untuk mengubah konfigurasi dari layanan otentikasi.

Pada intinya, fasilitas otentikasi Laravel terdiri dari “*guard*s” dan “*provider*s”. *Guard*s menentukan bagaimana pengguna diautentikasi untuk setiap permintaan. Misalnya, Laravel mengirim dengan *guard*s untuk sesi dengan menggunakan penyimpanan session dan cookie.

## Middleware

**Middleware** adalah lapisan perantara antara permintaan ***route HTTP*** yang masuk dan ***action*** dari Controller yang akan dijalankan. **Middleware** memungkinkan kita untuk meakukan berbagai tugas baik itu sebelum ataupun sesudah tindakan dilakukan. Kita juga dapat menggunakan *tool* **CLI** untuk membuat sebuah **Middleware** dalam **Laravel**. Beberapa contoh penggunaan **Middleware** meliputi autentikasi, validasi, manipulasi permintaan, dan lainnya. Berikut di bawah ini adalah manfaat dari **Middleware :**

* **Keamanan** : dalam **Middleware** memungkinkan kita untuk memverifikasi apakah pengguna sudah diautentikasi sebelum mengakses halaman tertentu. Dengan demikian, kita dapat melindungi data sensitif dan mengontrol hak akses pengguna.
* **Pemfilteran Data** : **Middleware** dapat digunakan untuk memanipulasi data permintaan sebelum sebuah ***action*** dalam *controller* dilakukan. Misalnya, kita dapat memeriksa terlebih dahulu data yang dikirim oleh pengguna sebelum data tersebut diproses lebih

lanjut atau kita ingin memodifikasi data yang akan dikirim lalu kita dapat memeriksa ulang data yang akan dikirim oleh pengguna sebelum data tersebut diproses.

* **Logging dan Audit** : **Middleware** juga dapat digunakan untuk mencatat aktivitas pengguna atau melakukan audit terhadap permintaan yang masuk. Ini dapat membantu dalam pemantauan dan analisis aplikasi.
* **NOTES**:
* Authentication: proses untuk **memverifikasi identitas pengguna** yang mencoba mengakses sistem (misalnya, dengan memeriksa kredensial seperti username dan password). Menentukan "siapa" pengguna tersebut
* Authorization: proses yang menentukan **hak akses** atau izin yang dimiliki pengguna setelah mereka berhasil diautentikasi. Ini berkaitan dengan "apa" yang boleh dilakukan pengguna dalam sistem. Mengatur dan membatasi akses pengguna sesuai dengan peran (role) atau level mereka

**INFO**

Kita akan menggunakan Laravel Auth secara manual seperti <https://laravel.com/docs/10.x/authentication#authenticating-users>

Sesuai dengan **Studi Kasus PWL.pdf**.

Jadi project Laravel 10 kita masih sama dengan menggunakan repositori **PWL\_POS.**

*Project* **PWL\_POS** akan kita gunakan sampai pertemuan 12 nanti, sebagai project yang akan kita pelajari

# Implementasi Manual Authentication di Laravel

Autentikasi adalah proses untuk memverifikasi identitas pengguna yang mencoba mengakses sistem. Dalam konteks aplikasi web, autentikasi memastikan bahwa pengguna yang mencoba login memiliki hak akses yang sesuai berdasarkan kredensial seperti email dan password. Proses autentikasi berbeda dengan **otorisasi**, yang merupakan langkah lanjutan untuk menentukan hak akses apa yang dimiliki pengguna setelah mereka berhasil diautentikasi. **Konsep Autentikasi di Laravel**

Laravel menawarkan sistem autentikasi yang sangat fleksibel. Laravel menyediakan mekanisme autentikasi bawaan melalui layanan authentication scaffolding seperti Laravel *Jetstream* dan *Breeze*, yang dapat secara otomatis menghasilkan halaman dan logika autentikasi. Namun, terkadang pengembang memerlukan implementasi autentikasi yang lebih manual untuk memberikan kontrol penuh terhadap setiap aspek dari proses tersebut.

Beberapa komponen penting dalam sistem autentikasi Laravel meliputi:

* *Guard*: Komponen yang mengatur bagaimana pengguna diautentikasi untuk setiap permintaan. *Guard* default menggunakan sesi dan cookie.
* *Provider*: Mengatur bagaimana pengguna diambil dari database atau sumber data lainnya. *Provider* default mengambil data pengguna dari database dengan menggunakan Eloquent ORM.
* *Session*: Laravel menggunakan sesi untuk menyimpan status autentikasi pengguna. Sesi memungkinkan sistem untuk mengingat pengguna yang sudah login di antara permintaan HTTP yang berbeda.

Alur umum dari autentikasi meliputi:

* 1. *Login*: Pengguna mengirimkan kredensial (biasanya berupa email dan password).
  2. *Verifikasi Kredensial*: Sistem memeriksa apakah kredensial yang diberikan sesuai dengan data di database.
  3. *Pembuatan Sesi*: Jika kredensial benar, sistem akan membuat sesi untuk pengguna yang akan disimpan di server.
  4. *Akses ke Halaman yang Dilindungi*: Pengguna yang terautentikasi dapat mengakses halaman-halaman yang dilindungi oleh *middleware* auth.
  5. *Logout*: Pengguna bisa keluar dari sistem dan sesi mereka akan dihapus.

## Middleware Autentikasi

Middleware auth di Laravel digunakan untuk melindungi rute atau halaman agar hanya dapat diakses oleh pengguna yang sudah terautentikasi. Jika pengguna mencoba mengakses rute yang memerlukan autentikasi tanpa login, mereka akan diarahkan ke halaman login.

* + - ***Guard*** bertanggung jawab untuk menangani proses autentikasi pengguna. Laravel secara default menggunakan *guard* berbasis sesi untuk autentikasi web, namun juga mendukung *guard* berbasis token (seperti API).
    - ***Provider*** bertugas untuk mengambil pengguna dari database. Laravel menyediakan *provider* default yang menggunakan Eloquent, namun juga mendukung *provider* lain seperti Query Builder.

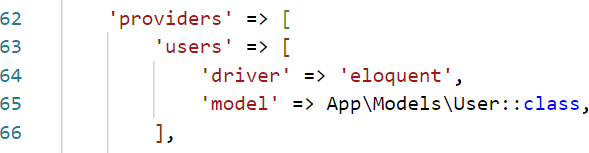
## Implementasi di Laravel 10

Kita akan menerapkan penggunakan authentication di Laravel. Dalam penerapan ini, kita akan mencoba membuat otentikasi secara di Laravel, agar kita paham langkah-langkah dalam membuat Authentication

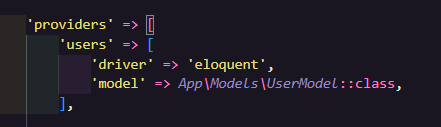
**Praktikum 1** – Implementasi Authentication :

1. Kita buka project laravel **PWL\_POS** kita, dan kita modifikasi konfigurasi aplikasi kita di

config/auth.php



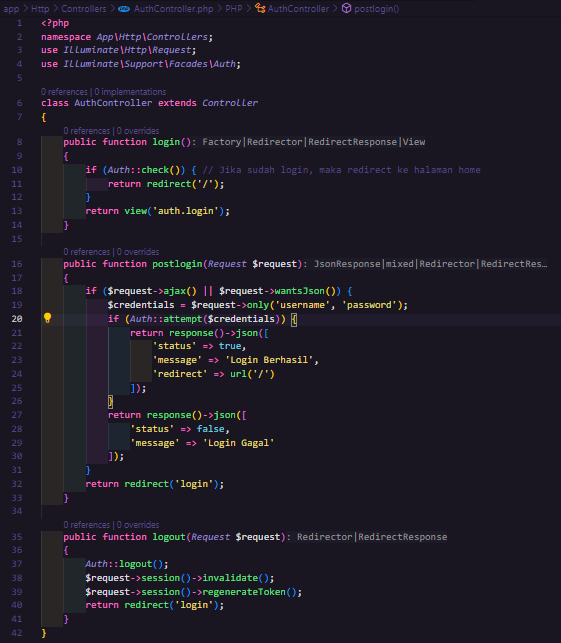
Pada bagian ini kita sesuaikan dengan Model untuk tabel m\_user yang sudah kita buat



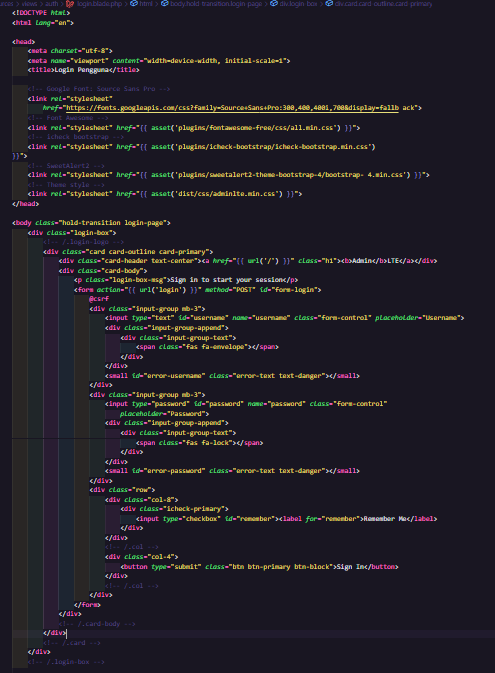
1. Selanjutnya kita modifikasi sedikit pada UserModel.php untuk bisa melakukan proses otentikasi



1. Selanjutnya kita buat AuthController.php untuk memproses login yang akan kita lakukan



1. Setelah kita membuat AuthController.php, kita buat view untuk menampilkan halaman login. View kita buat di auth/login.blade.php , tampilan login bisa kita ambil dari contoh login di template **AdminLTE** seperti berikut (pada contoh login ini, kita gunakan page login-V2 di **AdminLTE**)



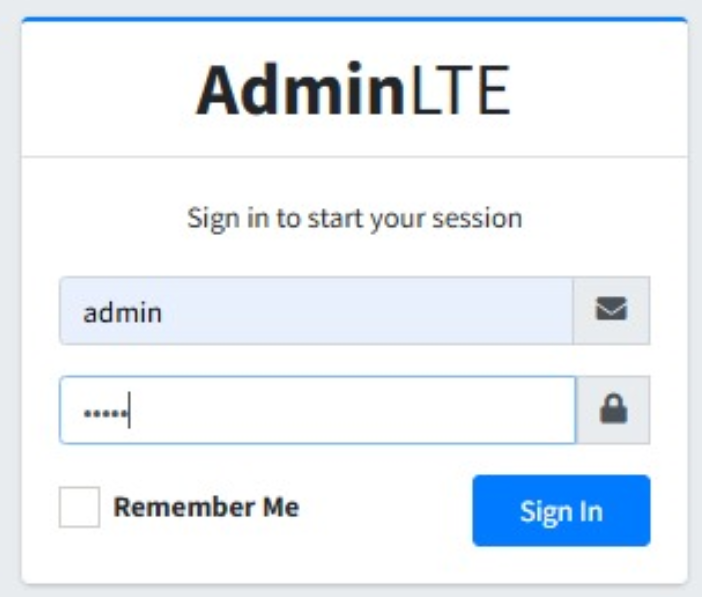
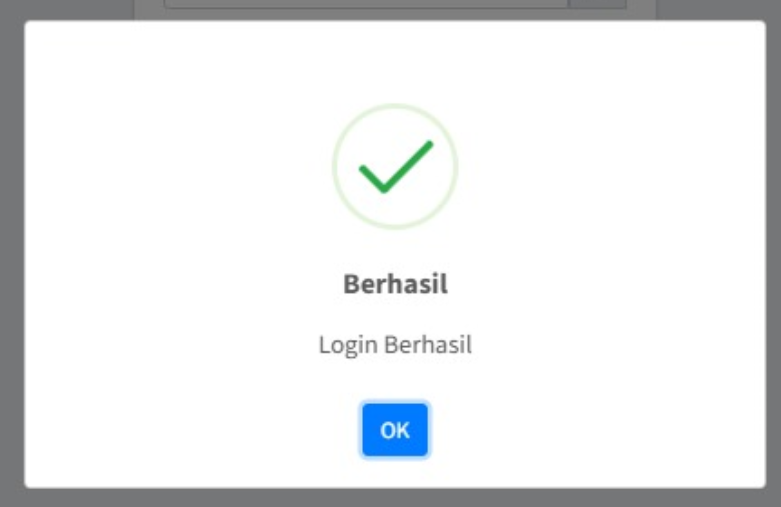


1. Kemudian kita modifikasi route/web.php agar semua route masuk dalam auth

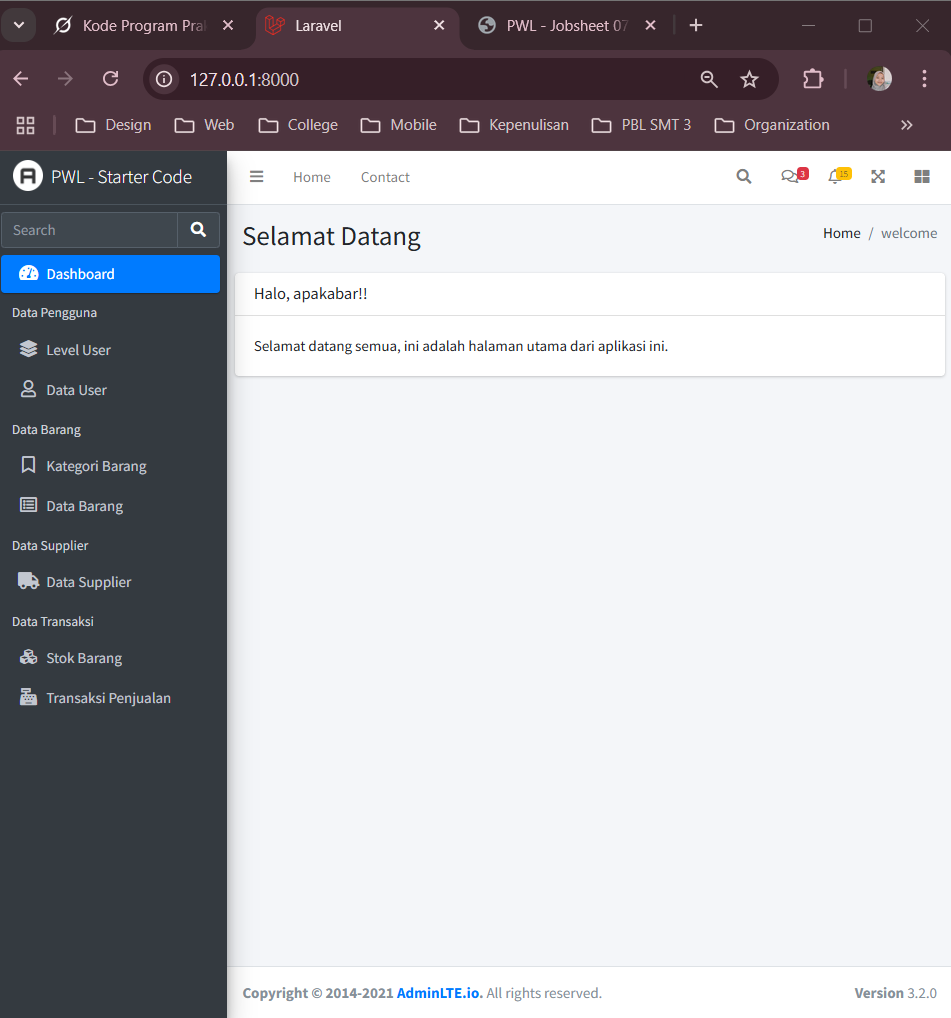
****

1. Ketika kita coba mengakses halaman localhost/PWL\_POS/public makan akan tampil halaman awal untuk login ke aplikasi

**Jawab:**

Jika kita berhasil login, maka akan meroute ke halaman dashboard

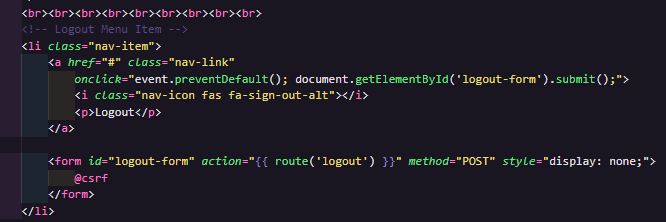


**Tugas 1** – Implementasi Authentication :

1. Silahkan implementasikan proses login pada project kalian masing-masing
2. Silahkan implementasi proses logout pada halaman web yang kalian buat

**Jawab:**

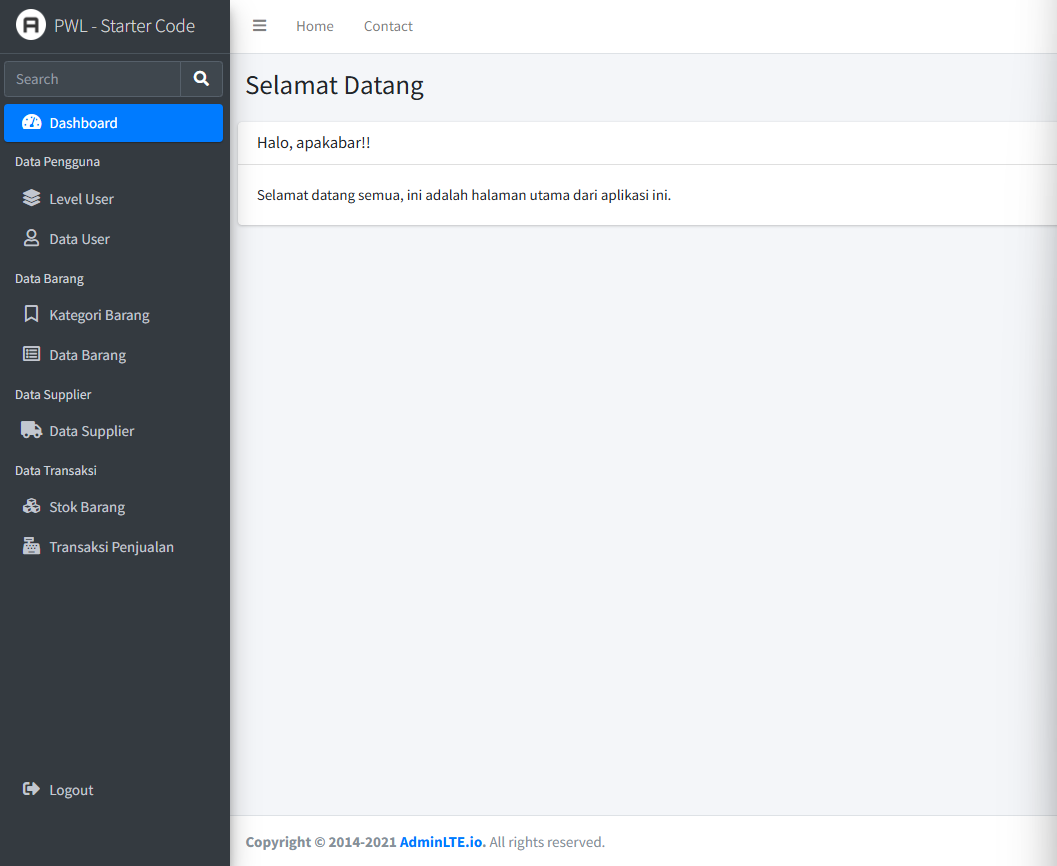
1. Menambahkan button logout pada sidebar.blade



1. Memodifikasi route pada web.php menjadi POST



1. Tampilan logout sidebar



Jika diklik logout otomatis akan keluar dan kembali ke halaman login

1. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan

**Jawab:**

1. Pengguna mengklik tombol "Logout" di sidebar
2. Route::post('/logout', [AuthController::class, 'logout'])->name('logout'); pada web.php digunakan untuk proses logout dari sistem. Disini dia menggunakan method POST bukan GET karena logout adalah tindakan untuk mengakhiri sesi.
3. Route ini akan memanggil method /logout yg ada dalam AuthController (method yg menghapus data sesi pengguna dan mengakhiri status login)
4. AuthController menjalankan proses logout dan mengarahkan pengguna ke halaman login
5. Submit kode untuk impementasi Authentication pada repository github kalian.
6. **Implementasi *Authorization* di Laravel**

*Authorization* merupakan proses setelah authentication berhasil dilakukan (dalam kata lain,kita berhasil login ke sistem). *Authorization* berkenaan dengan hak akses pengguna dalam menggunakan sistem. *Authorization* memberikan/memastikan hak akses (ijin akses) kita, sesuai dengan aturan (role) yang ada di sistem. *Authorization* sangat penting untuk membatasi akses pengguna sesuai dengan peruntukannya.

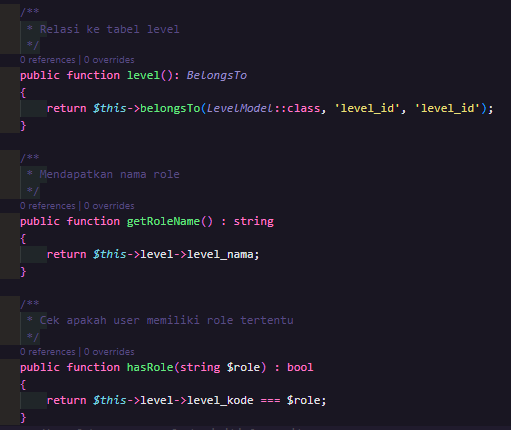
**Contoh** ketika kita mengakses LMS dengan akun (*username* dan *password*) yang bertipe Mahasiswa. Saat berhasil melakukan authentication, maka hak akses kita juga akan diberikan selayaknya mahasiswa. Seperti melihat kursus (course), melihat materi, men-download file materi, mengerjakan/meng-upload tugas, mengikuti ujian, dll. Kita tidak akan diberikan hak akses oleh sistem untuk membuat materi, membuat soal ujian, membuat tugas, memberikan nilai tugas karena hak akses tersebut masuk ke ranah akun tipe Dosen/Pengajar.

Selain menyediakan layanan otentikasi bawaan, Laravel juga menyediakan cara sederhana untuk mengotorisasi tindakan pengguna terhadap sumber daya tertentu. Misalnya, meskipun pengguna diautentikasi, mereka mungkin tidak berwenang untuk memperbarui atau menghapus model Eloquent atau rekaman database tertentu yang dikelola oleh aplikasi Anda. Fitur otorisasi Laravel menyediakan cara yang mudah dan terorganisir untuk mengelola jenis pemeriksaan otorisasi ini.

**Praktikum 2** – Implementasi *Authorizaton* di Laravel dengan Middleware

Kita akan menerapkan *authorization* pada project Laravel dengan menggunakan Middleware sebagai pengecekan akses. Langkah-langkah yang kita kerjakan sebagai berikut:

* 1. Kita modifikasi UserModel.php dengan menambahkan kode berikut

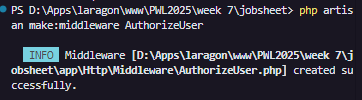


* 1. Kemudian kita buat *middleware* dengan nama AuthorizeUser.php. Kita bisa buat

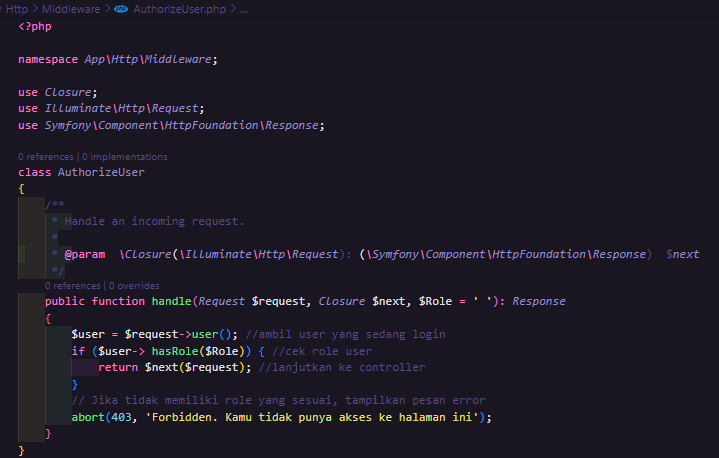
*middleware* dengan mengetikkan perintah pada terminal/CMD

php artisan make:*middleware* AuthorizeUser

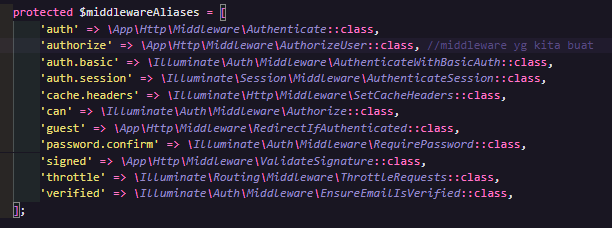
File *middleware* akan dibuat di app/Http/Middleware/AuthorizeUser.php



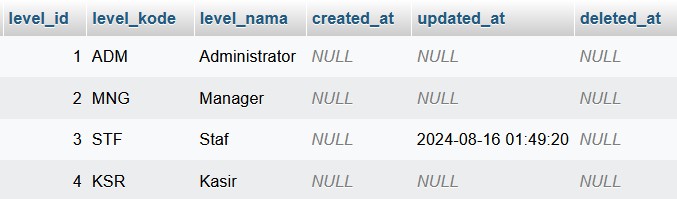
* 1. Kemudian kita edit *middleware* AuthorizeUser.php untuk bisa mengecek apakah pengguna yang mengakses memiliki Level/Role/Group yang sesuai

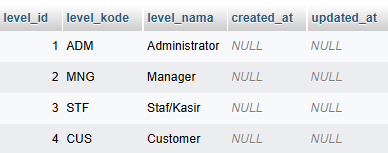


* 1. Kita daftarkan ke app/Http/Kernel.php untuk *middleware* yang kita buat barusan



* 1. Sekarang kita perhatikan tabel m\_level yang menjadi tabel untuk menyimpan level/group/role dari user ada





* 1. Untuk mencoba *authorization* yang telah kita buat, maka perlu kita modiifikasi route/web.php untuk menentukan route mana saja yang akan diberi hak akses sesuai dengan level user

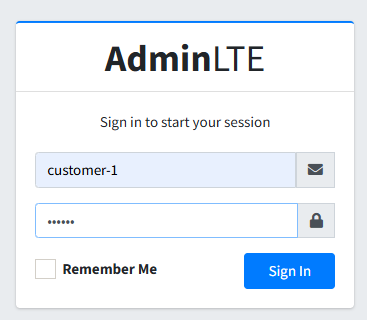


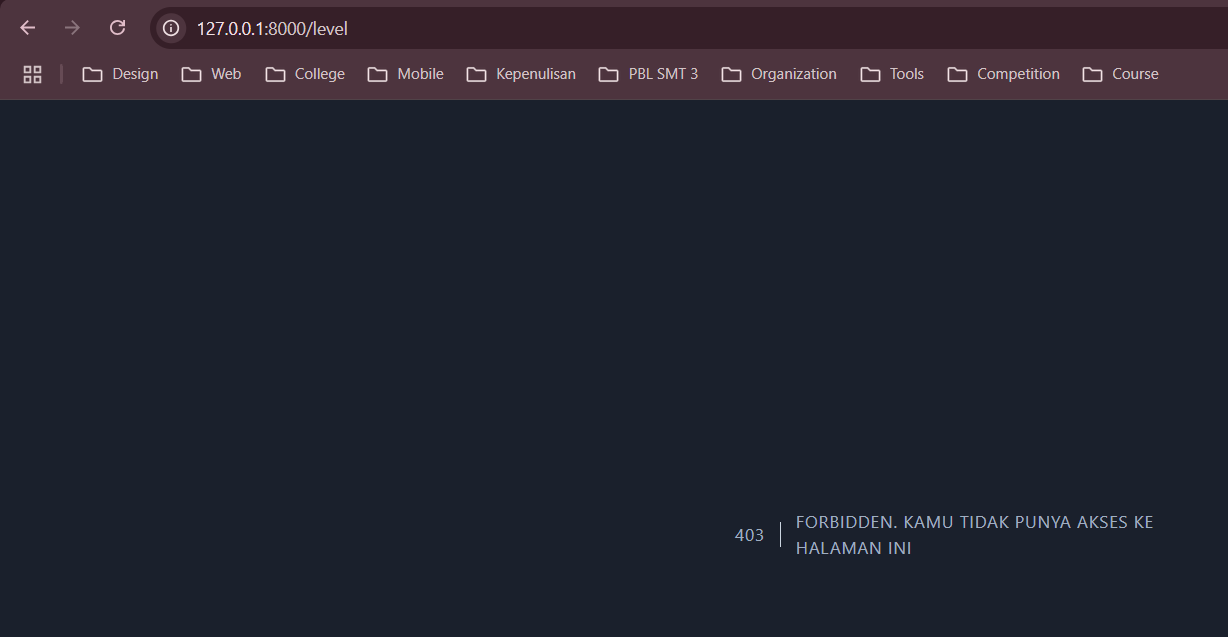
Pada kode yang ditandai merah, terdapat authorize:ADM . Kode ADM adalah nilai dari level\_kode pada tabel m\_level. Yang artinya, user yang bisa mengakses route untuk manage data level, adalah user yang memiliki level sebagai Administrator.

* 1. Untuk membuktikannya, sekarang kita coba login menggunakan akun selain level administrator, dan kita akses route menu level tersebut

**Jawab:**

Disini saya mencoba login menggunakan akun customer, dan akses ditolak





**Tugas 2** – Implementasi Authoriization :

1. Apa yang kalian pahami pada praktikum 2 ini?

**Jawab:** Authorization adalah proses yang dilakukan setelah authentication berhasil, untuk menentukan hak akses pengguna dalam sistem sesuai dengan peran (role) atau level yang dimilikinya. alam konteks ini, kita membatasi akses ke rute atau fitur tertentu berdasarkan level pengguna, seperti Administrator, Manager, Staf, atau Cust

1. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan

**Jawab:**

1. Memodifikasi UserModel dengan menambahkan function untuk relasi ke tabel m\_level (mendapatkan role sesuai yg ada di tabel m\_level dan memeriksa role hasRole()) 🡪 bertujuan untuk mengelola role pengguna agar bisa diakses oleh middleware
2. Membuat Middleware AuthorizeUser 🡪 untuk memeriksa otorisasi berdasarkan role
3. Edit Middleware AuthorizeUser.php 🡪 Menambahkan logika pengecekan role dan pengalihan jika tidak punya hak akses.
4. Daftarkan Middleware di Kernel.php 🡪 Mendaftarkan middleware agar bisa digunakan di route.
5. Tabel m\_level 🡪 Memahami struktur data role apa saja yang akan digunakan untuk otorisasi
6. Modifikasi Route 🡪 Mengatur route agar hanya pengguna dengan role tertentu yang bisa mengakses.
7. Submit kode untuk impementasi Authorization pada repository github kalian.

# Multi-Level Authorization di Laravel

Bagaimana seandainya jika terdapat level/group/role satu dengan yang lain memiliki hak akses yang sama. Contoh sederhana, user level Admin dan Manager bisa sama-sama mengakses menu Barang pada aplikasi yang kita buat. Maka tidak mungkin kalau kita buat route untuk masing-masing level user. Hal ini akan memakan banyak waktu, dan proses yang lama.

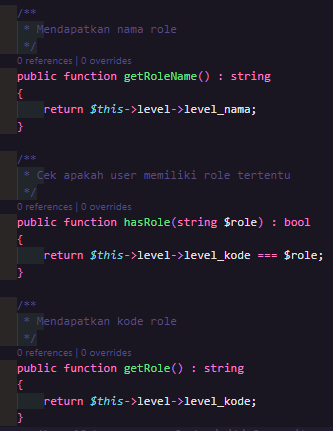


Hal ini jadi kendala ketika kita mau mengganti hak akses, maka kita akan mengganti sebagian besar route yang sudah kita tulis. Untuk itu, kita perlu mengelola middleware agar bisa mendukung penambahan hak akses secara dinamis.

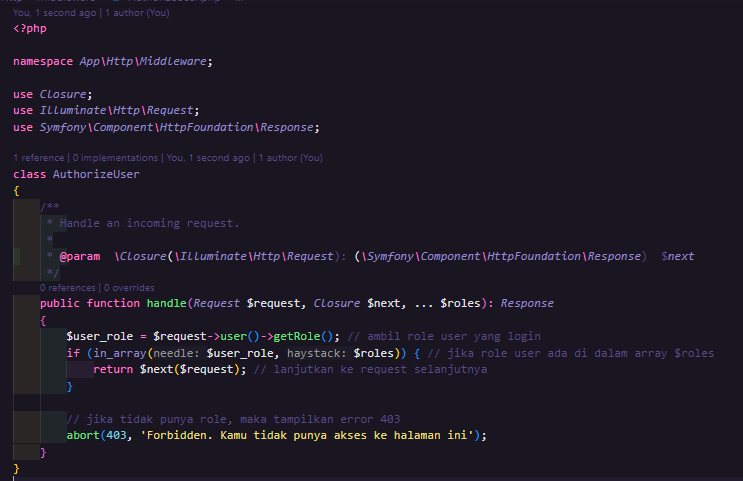
**Praktikum 3** – Implementasi Multi-Level Authorizaton di Laravel dengan Middleware

Kita akan menerapkan multi-level authorization pada project Laravel dengan menggunakan Middleware sebagai pengecekan akses. Langkah-langkah yang kita kerjakan sebagai berikut:

* 1. Kita modifikasi UserModel.php untuk mendapatkan level\_kode dari user yang sudah login. Jadi kita buat fungsi dengan nama getRole()



* 1. Selanjutnya, Kita modifikasi middleware AuthorizeUser.php dengan kode berikut



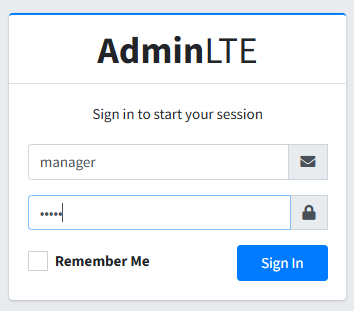
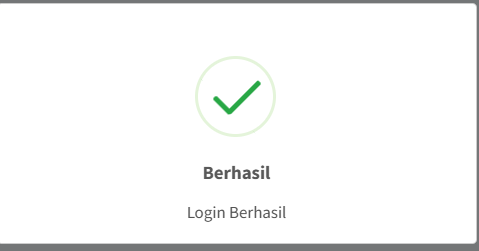
* 1. Setelah itu tinggal kita perbaiki route/web.php sesuaikan dengan role/level yang diinginkan. Contoh

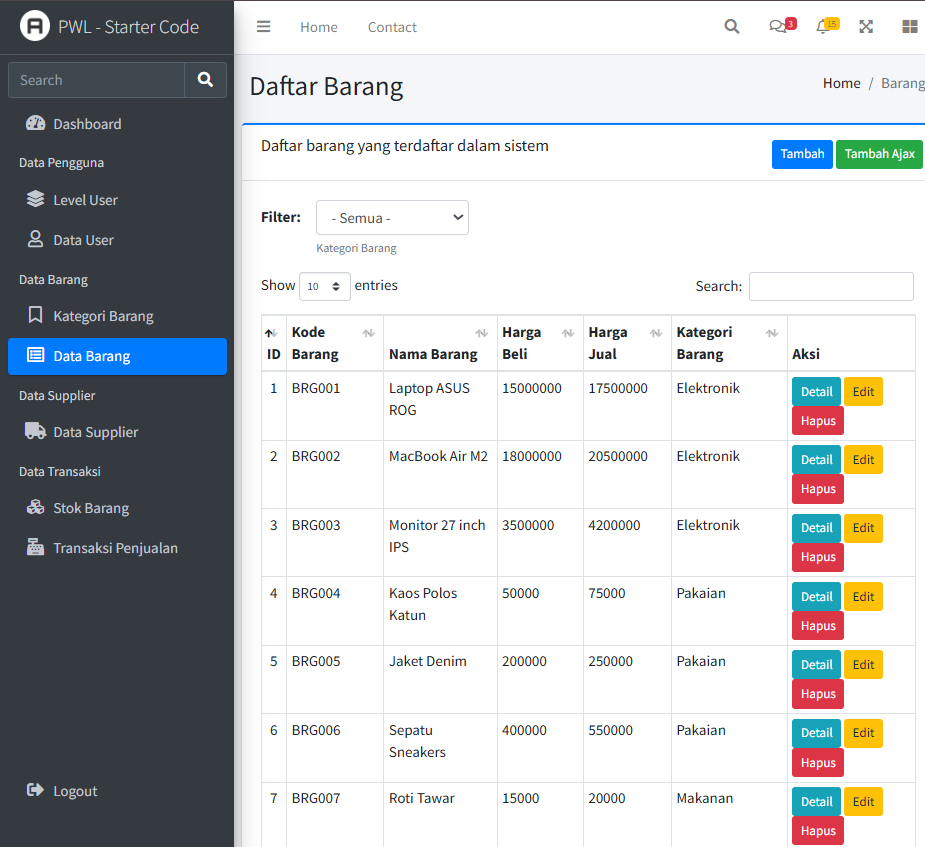


* 1. Sekarang kita sudah bisa memberikan hak akses menu/route ke beberapa level user

**Jawab:**

1. Disini saya mencoba login dengan akun manager untuk membuka halaman barang



Dan hasilnya bisa dibuka, karena autorisasi/hak akses pengguna disini yaitu admin dan manager

**Tugas 3** – Implementasi Multi-Level Authorization :

1. Silahkan implementasikan multi-level authorization pada project kalian masing-masing
2. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
3. Implementasikan multi-level authorization untuk semua Level/Jenis User dan Menu- menu yang sesuai dengan Level/Jenis User

**Jawab:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Menu | | Level Admin (ADM) | Level Manager (MNG) | Level Staff (STF) |
| Level User |  |  |  |
| Data User |  |  |  |
| Data Kategori |  |  |  |
| Data Barang |  |  |  |
| Data Supplier |  |  |  |

**Notes:** Staf hanya dapat read pada data barang. Tidak bisa create, update, dan delete

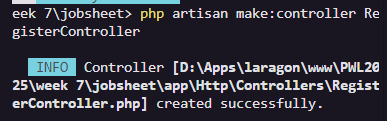
1. Submit kode untuk impementasi Authorization pada repository github kalian.

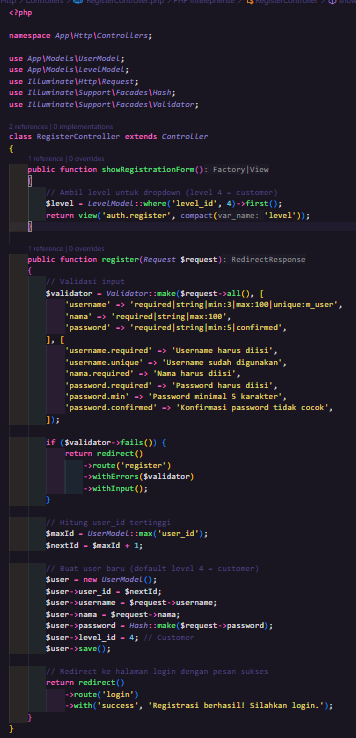
**Tugas 4** – Implementasi Form Registrasi :

1. Silahkan implementasikan form untuk registrasi user.

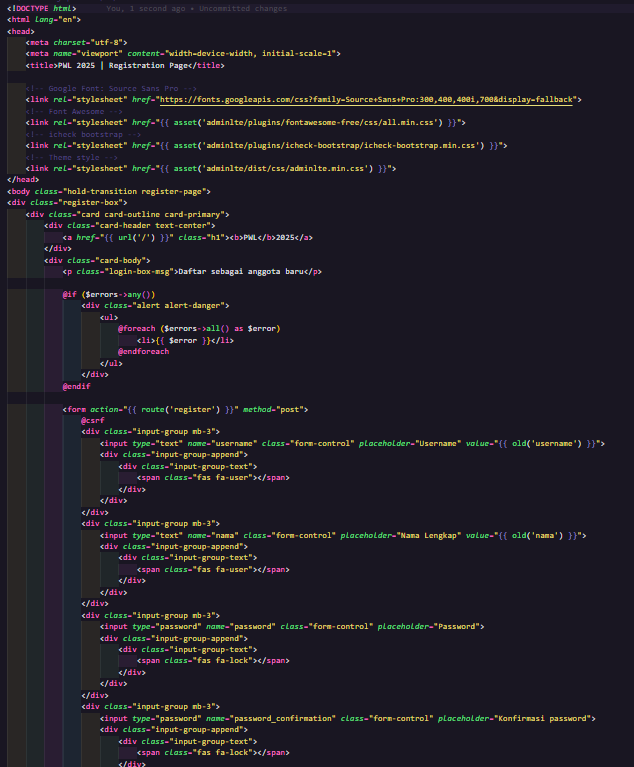
**Jawab:**

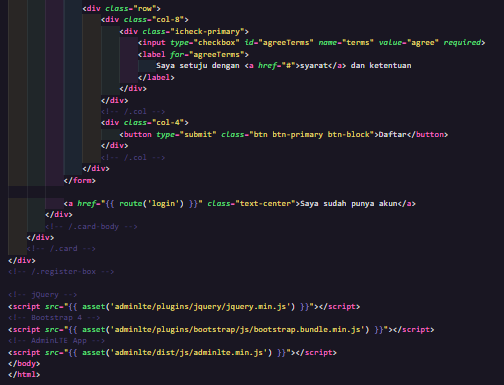
1. Membuat controller untuk menangani register





1. View untuk registrasi





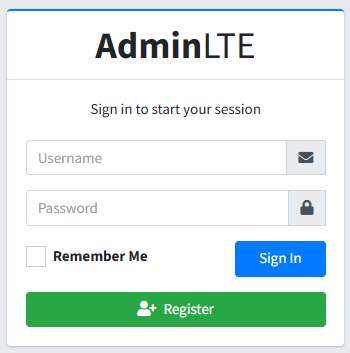
1. Route untuk mengakses halaman registrasi



1. Screenshot hasil yang kalian kerjakan

**Jawab:**

1. Berikut adalah tampilan form register



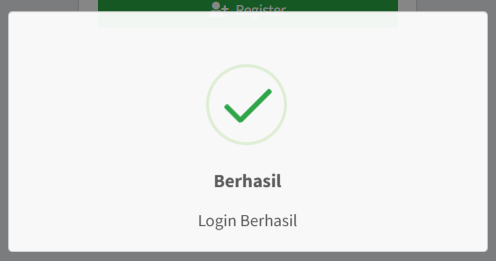
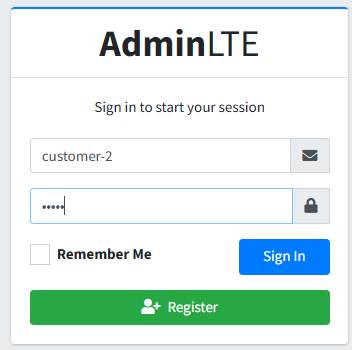
1. Saat klik register maka akan muncul halaman form register

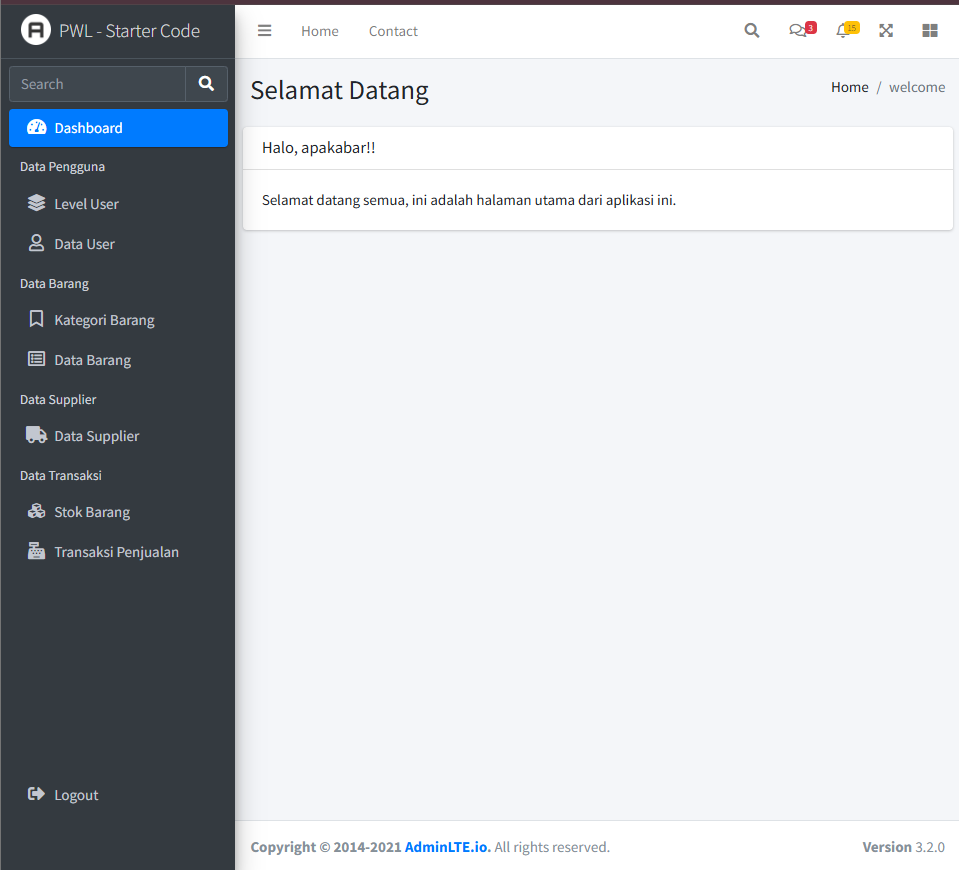


1. Saya coba daftar dengan username customer-2

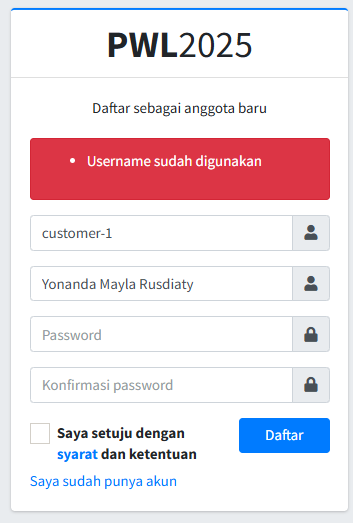


1. Saya coba login menggunakan username customer-2





1. Berikut adalah tampilan jika pengguna membuat akun dengan username yg sudah ada



1. Commit dan push hasil tugas kalian ke masing-masing repo github kalian

*\*\*\* Sekian, dan selamat belajar \*\*\**