

Laporan Praktikum Pemrograman Web Lanjut P7



Disusun oleh :

Octrian Adiluhung Tito Putra

NIM. 2341720078

Dosen Pengampu Mata Kuliah :

Dimas Wahyu Wibowo, ST., MT.

KELAS 2-A

D-4 TEKNIK INFORMATIKA

JURUSAN TEKNOLOGI INFORMASI

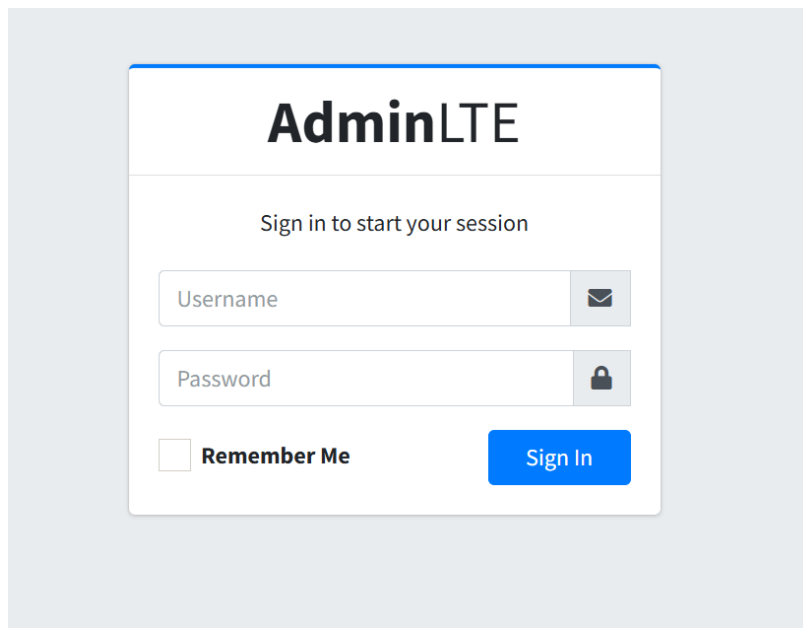
POLITEKNIK NEGERI MALANG

2025

Praktikum 1 : Implementasi Authentication

- Melakukan modifikasi file konfigurasi autentikasi di config/auth.php agar sesuai dengan model untuk tabel m_user.
- Melakukan Modifikasi UserModel.php agar mendukung proses autentikasi.
- Membuat AuthController.php untuk memproses login dan logout.
- Membuat view untuk halaman login di resources/views/auth/login.blade.php dengan tampilan yang diambil dari template AdminLTE.

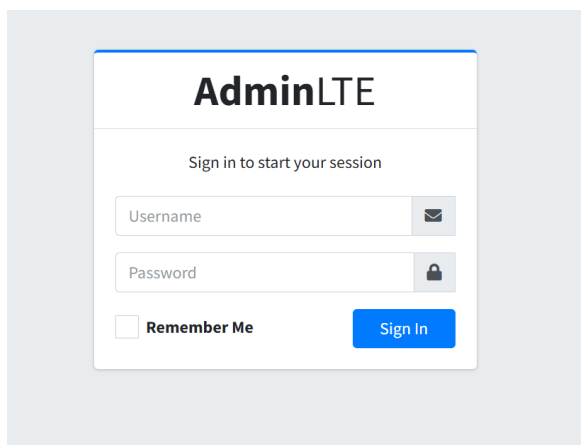
- **Hasil:**



Tugas 1 : Implementasi Authentication

1. Silahkan implementasikan proses login pada project kalian masing-masing

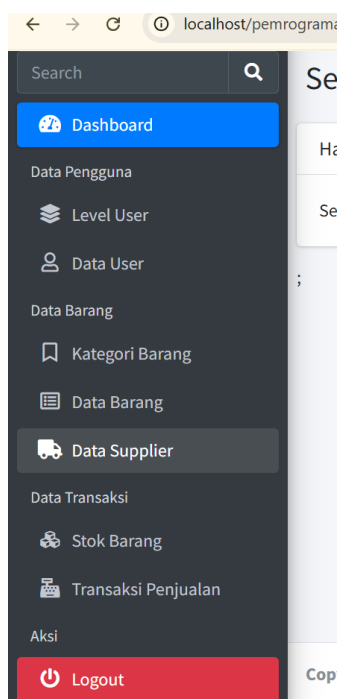
- Proses Login



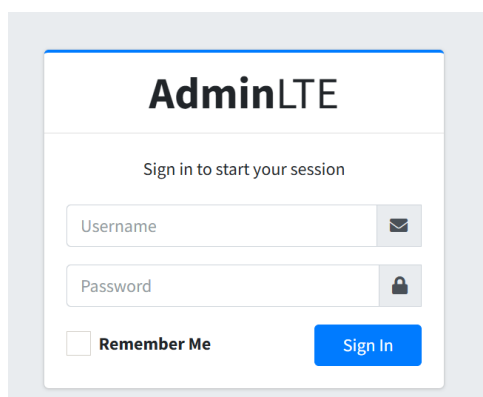
2. Silahkan implementasi proses logout pada halaman web yang kalian buat

```
header.blade.php 66 </li>
sidebar.blade.php 67 <li class="nav-header">Aksi</li>
                  68 <li class="nav-item bg-danger rounded">
template.blade.php 69 <a href="{{ url(path: '/logout') }}" class="nav-link">
> level           70 <i class="nav-icon fas fa-power-off"></i>
> supplier        71 <p>Logout</p>
> user            72 </a>
kategori.blade.php 73 </li>
```

- Menambahkan fitur logout di sidebar dengan url yang mengarah ke routes /logout agar bisa memanggil fungsi logout dari AuthController melalui routes web.



- Fitur logout telah muncul di sidebar

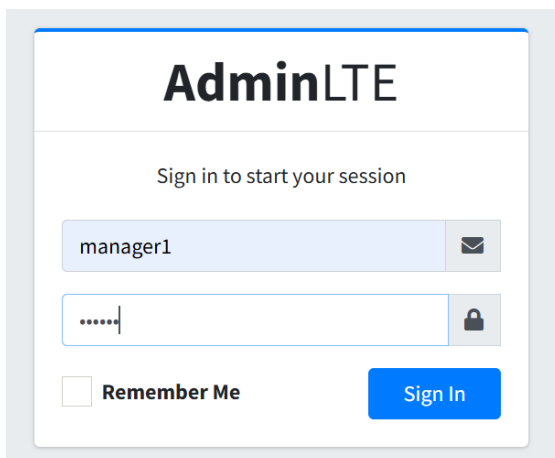


- Berhasil kembali ke halaman login lagi setelah logout

3. Amati dan jelaskan tiap tahapan yang kalian kerjakan, dan jabarkan dalam laporan
- Memodifikasi config/auth untuk mengakses UserModel
 - Memodifikasi UserModel dengan menambahkan variable hidden yang fungsi nya untuk tidak menampilkan password saat di select dan variable casts yang melakukan hash pada password
 - Membuat AuthController untuk memproses login user
 - Membuat view untuk login user
 - Dan yang terakhir membuat routes untuk proses login user
4. Submit kode untuk impementasi Authentication pada repository github kalian. (Sudah)

Praktikum 2 : Implementasi Authorizaton di Laravel dengan Middleware

- Melakukan modifikasi UserModel.php dengan menambahkan properti/relasi yang mendukung pengecekan role/level.
- Membuat middleware AuthorizeUser.php menggunakan perintah:
php artisan make:middleware AuthorizeUser
- Mengedit middleware AuthorizeUser.php
- Melakukan registrasi middleware di file app/Http/Kernel.php.
- Melakukan modifikasi file routes/web.php untuk menerapkan middleware pada route yang ingin dilindungi, misalnya dengan menambahkan keterangan seperti authorize:ADM untuk akses administrator.
- **Hasil:**



403

FORBIDDEN. KAMU TIDAK PUNYA AKSES KE HALAMAN INI

- Saya mencoba login menggunakan akun selain Admin (ADM), dan hasilnya Ketika saya ingin mengakses menu level user muncul sebuah error 403 seperti gambar diatas, itu berarti authorization berhasil dilakukan.

Tugas 2 : Implementasi Authorization

1. Apa yang kalian pahami pada praktikum 2 ini?

- Pada praktikum 2 ini kita diajarkan bagaimana cara untuk mengatur hak akses tiap user, pada praktikum ini hak akses untuk menu level user telah dibatasi. Hanya level user dengan kode ADM atau admin saja yang dapat mengakses menu tersebut, sementara user lainnya seperti manajer dan staff mendapatkan pesan error Ketika mencoba untuk mengakses menu level user.

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

```
Tabnine | Edit | Test | Explain | Document | 0 references | 0 overrides
public function getRoleName(): string {
    return $this->level->level_nama;
}

/**
 * Cek apakah user memiliki role tertentu
 */
Tabnine | Edit | Test | Explain | Document | 0 references | 0 overrides
public function hasRole($role): bool {
    return $this->level->level_kode == $role;
}
```

- Menambahkan function baru di usermodel untuk mengambil data level_nama dan level_kode

```
PS C:\laragon\www\pemrograman_web_lanjutan\Minggu7\PWL_POS> php artisan make:middleware AuthorizeUser

INFO Middleware [C:\laragon\www\pemrograman_web_lanjutan\Minggu7\PWL_POS\app\Http\Middleware\AuthorizeUser.php] created successfully.
```

- Membuat middleware baru

```

public function handle(Request $request)
{
    $user = $request->user();
    if ($user->hasRole($role)) {
        return $next($request);
    }
    // jika tidak punya role, maka
    abort(403, 'Forbidden');
}

```

- middleware yang telah dibuat tadi digunakan untuk mengambil data user dan mengecek apakah user mempunyai role yang diinginkan.

```

0 references
protected $middlewareAliases = [
    'auth' => \App\Http\Middleware\Authenticate::class,
    'authorize' => \App\Http\Middleware\AuthorizeUser::class,
];

```

- Mendaftarkan middleware yang baru ke file kernel.php

```

// artinya semua route di dalam group ini harus punya role ADM (Administrator)
Route::middleware(['authorize:ADM'])->group(function() {
    Route::get('/level', function() { return view('level.index'); });
    Route::post('/level/list', function() { return view('level.list'); });
    Route::get('/level/create', function() { return view('level.create'); });
    Route::post('/level', function() { return view('level.store'); });
    Route::get('/level/create_ajax', function() { return view('level.create_ajax'); });
    Route::post('/level/ajax', function() { return view('level.store_ajax'); });
    Route::get('/level/{id}', function() { return view('level.show'); });
    Route::get('/level/{id}/show_ajax', function() { return view('level.show_ajax'); });
    Route::get('/level/{id}/edit', function() { return view('level.edit'); });
    Route::get('/level/{id}', function() { return view('level.update'); });
    Route::get('/level/{id}/edit_ajax', function() { return view('level.edit_ajax'); });
    Route::put('/level/{id}/update_ajax', function() { return view('level.update_ajax'); });
    Route::get('/level/{id}/delete_ajax', function() { return view('level.delete_ajax'); });
    Route::delete('/level/{id}/delete_ajax', function() { return view('level.delete_ajax'); });
    Route::delete('/level/{id}', function() { return view('level.destroy'); });
});

```

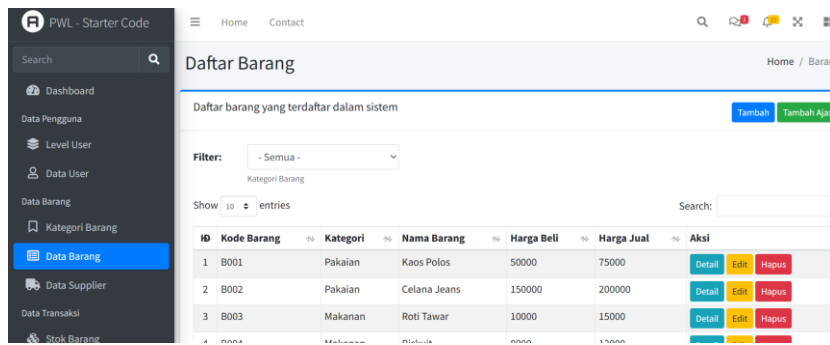
- Menentukan routes mana saja yang hanya bisa diakses oleh level kode “ADM”.

3. Submit kode untuk implementasi Authorization pada repository github kalian. (sudah)

Praktikum 3 : Implementasi Multi-Level Authorization di Laravel dengan Middleware

- Melakukan modifikasi UserModel.php dengan menambahkan fungsi getRole() untuk mendapatkan kode level dari user.
- Penyesuaian pada middleware AuthorizeUser.php agar mendukung pengecekan multi-level authorization.
- Melakukan perbaikan dan pengaturan ulang route di routes/web.php agar dapat mengatur hak akses untuk beberapa level user secara dinamis.

- **Hasil:**



- User manajer (MNG) dan admin (ADM) bisa mengakses menu data barang karena mendapat authorization yang diatur di routes middleware.



- Sementara user staff (STF) tidak bisa mengakses menu data barang karena tidak diberikan authorization.

Tugas 3 : Implementasi Multi-Level Authorization

1. Silahkan implementasikan multi-level authorization pada project kalian masing-masing

```
Route::middleware(middleware: ['authorize:STF,ADM,MNG'])->group(callback: function(): void{
    Route::group(attributes: ['prefix' => 'barang'], routes: function(): void{
        Route::get(uri: '/', action: [BarangController::class, 'index']); // Menampilkan
        Route::post(uri: '/list', action: [BarangController::class, 'list']); // Menampilkan
        Route::get(uri: '/create', action: [BarangController::class, 'create']); // Menampilkan
        Route::post(uri: '/', action: [BarangController::class, 'store']); // Menyimpan data
        Route::get(uri: '/create_ajax', action: [BarangController::class, 'create_ajax']); // Menampilkan
        Route::post(uri: '/ajax', action: [BarangController::class, 'store_ajax']); // Menyimpan data
        Route::get(uri: '/{id}/show_ajax', action: [BarangController::class, 'show_ajax']); // menampilkan
        Route::get(uri: '/{id}', action: [BarangController::class, 'show']); // Menampilkan
        Route::get(uri: '/{id}/edit', action: [BarangController::class, 'edit']); // Menampilkan
        Route::put(uri: '/{id}', action: [BarangController::class, 'update']); // Menyimpan data
        Route::get(uri: '/{id}/edit_ajax', action: [BarangController::class, 'edit_ajax']); // Menampilkan
        Route::put(uri: '/{id}/update_ajax', action: [BarangController::class, 'update_ajax']); // Menyimpan data
        Route::get(uri: '/{id}/delete_ajax', action: [BarangController::class, 'confirm_ajax']); // Menampilkan
        Route::delete(uri: '/{id}/delete_ajax', action: [BarangController::class, 'delete_ajax']); // Menghapus data
        Route::delete(uri: '/{id}', action: [BarangController::class, 'destroy']); // Menghapus data
    });
});

Route::group(attributes: ['prefix' => 'kategori'], routes: function(): void{
    Route::get(uri: '/', action: [KategoriController::class, 'index']); // Menampilkan
    Route::post(uri: '/list', action: [KategoriController::class, 'list']); // Menampilkan
    Route::get(uri: '/create', action: [KategoriController::class, 'create']); // Menampilkan
    Route::post(uri: '/', action: [KategoriController::class, 'store']); // Menyimpan data
    Route::get(uri: '/create_ajax', action: [KategoriController::class, 'create_ajax']); // Menampilkan
    Route::post(uri: '/ajax', action: [KategoriController::class, 'store_ajax']); // Menyimpan data
    Route::get(uri: '/{id}/show_ajax', action: [KategoriController::class, 'show_ajax']); // menampilkan
    Route::get(uri: '/{id}', action: [KategoriController::class, 'show']); // Menampilkan
    Route::get(uri: '/{id}/edit', action: [KategoriController::class, 'edit']); // Menampilkan
    Route::put(uri: '/{id}', action: [KategoriController::class, 'update']); // Menyimpan data
    Route::get(uri: '/{id}/edit_ajax', action: [KategoriController::class, 'edit_ajax']); // Menampilkan
    Route::put(uri: '/{id}/update_ajax', action: [KategoriController::class, 'update_ajax']); // Menyimpan data
    Route::get(uri: '/{id}/delete_ajax', action: [KategoriController::class, 'confirm_ajax']); // Menampilkan
    Route::delete(uri: '/{id}/delete_ajax', action: [KategoriController::class, 'delete_ajax']); // Menghapus data
    Route::delete(uri: '/{id}', action: [KategoriController::class, 'destroy']); // Menghapus data
});
});
```

- Multilevel authorization untuk menu data barang dan kategori yang bisa di akses oleh 3 user (admin, manager, dan staff).

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

```
Tabnine | Edit | Test | Explain | Document | 0 references | 0 overrides
public function getRole(): mixed {
    return $this->level->level_kode;
}
```

- Menambahkan fungsi getRole pada UserModel

```
Tabnine | Edit | Test | Explain | Document | 0 references | 0 overrides
public function handle(Request $request, Closure $next, ... $roles): Response
{
    // $user = $request->user(); // ambil data user yang login, fungsi user() di
    // if ($user->hasRole($role)) { // cek apakah user mempunyai role yg diinginkan
    //     return $next($request);
    // }

    $user_role = $request->user()->getRole(); // ambil data level_kode dari user yang login
    if (in_array($user_role, $roles)) { // cek apakah level_kode user login ada di array $roles
        return $next($request); // jika ada, maka lanjutkan request
    }
    // jika tidak punya role, maka akan ditampilkan error 403
    abort(code: 403, message: 'Forbidden. Kamu tidak punya akses ke halaman ini');
}
```

- Mengubah pengecekan role pada AuthorizeUser yang sebelumnya hanya bisa mengecek satu role menjadi bisa mengecek array yang berisi role tertentu untuk multilevel authorization.

```
Route::middleware('middleware: ['authorize:ADM,MNG'])->group(callback: function(): void{
    Route::group(attributes: ['prefix' => 'supplier'], routes: function(): void{
        Route::get(uri: '/', action: [SupplierController::class, 'index']); // Menampilkan index
        Route::post(uri: '/list', action: [SupplierController::class, 'list']); // Menampilkan list
        Route::get(uri: '/create', action: [SupplierController::class, 'create']); // Menampilkan form create
        Route::post(uri: '/', action: [SupplierController::class, 'store']); // Menyimpan data
        Route::get(uri: '/create_ajax', action: [SupplierController::class, 'create_ajax']); // Menampilkan form create ajax
        Route::post(uri: '/ajax', action: [SupplierController::class, 'store_ajax']); // Menyimpan data ajax
        Route::get(uri: '/{id}/show_ajax', action: [SupplierController::class, 'show_ajax']); // Menampilkan data ajax
        Route::get(uri: '/{id}', action: [SupplierController::class, 'show']); // Menampilkan data
        Route::get(uri: '/{id}/edit', action: [SupplierController::class, 'edit']); // Menampilkan form edit
        Route::put(uri: '/{id}', action: [SupplierController::class, 'update']); // Menyimpan data
        Route::get(uri: '/{id}/edit_ajax', action: [SupplierController::class, 'edit_ajax']); // Menampilkan form edit ajax
        Route::put(uri: '/{id}/update_ajax', action: [SupplierController::class, 'update_ajax']); // Menyimpan data ajax
        Route::get(uri: '/{id}/delete_ajax', action: [SupplierController::class, 'confirm_ajax']); // Menampilkan form delete ajax
        Route::delete(uri: '/{id}/delete_ajax', action: [SupplierController::class, 'delete_ajax']); // Menyimpan data delete ajax
        Route::delete(uri: '/{id}', action: [SupplierController::class, 'destroy']); // Menghapus data
    });
});
```

- Mengatur routes agar bisa menerapkan multilevel authorization.

3. Implementasikan multi-level authorization untuk semua Level/Jenis User dan Menu menu yang sesuai dengan Level/Jenis User 4. Submit kode untuk impementasi Authorization pada repository github kalian.

1. Membuat authorization untuk admin

```
Route::middleware(middleware: ['authorize:ADM'])->group(callback: function(): void{
    Route::group(attributes: ['prefix' => 'level'], routes: function(): void{
        Route::get(uri: '/', action: [LevelController::class, 'index']);
        Route::post(uri: '/list', action: [LevelController::class, 'list']);
        Route::get(uri: '/create', action: [LevelController::class, 'create']);
        Route::post(uri: '/', action: [LevelController::class, 'store']);
        Route::get(uri: '/create_ajax', action: [LevelController::class, 'create_ajax']);
        Route::post(uri: '/ajax', action: [LevelController::class, 'store_ajax']);
        Route::get(uri: '/{id}', action: [LevelController::class, 'show']);
        Route::get(uri: '/{id}/show_ajax', action: [LevelController::class, 'show_ajax']);
        Route::get(uri: '/{id}/edit', action: [LevelController::class, 'edit']);
        Route::put(uri: '/{id}', action: [LevelController::class, 'update']);
        Route::get(uri: '/{id}/edit_ajax', action: [LevelController::class, 'edit_ajax']);
        Route::put(uri: '/{id}/update_ajax', action: [LevelController::class, 'update_ajax']);
        Route::get(uri: '/{id}/delete_ajax', action: [LevelController::class, 'confirm_ajax']);
        Route::delete(uri: '/{id}/delete_ajax', action: [LevelController::class, 'delete_ajax']);
        Route::delete(uri: '/{id}', action: [LevelController::class, 'destroy']);
    });
});

Route::group(attributes: ['prefix' => 'user'], routes: function(): void{
    Route::get(uri: '/', action: [UserController::class, 'index']);
    Route::post(uri: '/list', action: [UserController::class, 'list']);
    Route::get(uri: '/create', action: [UserController::class, 'create']);
    Route::post(uri: '/', action: [UserController::class, 'store']);
    Route::get(uri: '/create_ajax', action: [UserController::class, 'create_ajax']);
    Route::post(uri: '/ajax', action: [UserController::class, 'store_ajax']);
    Route::get(uri: '/{id}/show_ajax', action: [UserController::class, 'show_ajax']);
    Route::get(uri: '/{id}', action: [UserController::class, 'show']);
    Route::get(uri: '/{id}/edit', action: [UserController::class, 'edit']);
    Route::put(uri: '/{id}', action: [UserController::class, 'update']);
    Route::get(uri: '/{id}/edit_ajax', action: [UserController::class, 'edit_ajax']);
    Route::put(uri: '/{id}/update_ajax', action: [UserController::class, 'update_ajax']);
    Route::get(uri: '/{id}/delete_ajax', action: [UserController::class, 'confirm_ajax']);
    Route::delete(uri: '/{id}/delete_ajax', action: [UserController::class, 'delete_ajax']);
    Route::delete(uri: '/{id}', action: [UserController::class, 'destroy']);
});
```

- Admin dapat mengakses semua menu yang ada di website, tetapi dalam pengaturan authorization khusus untuk admin ini hanya ada akses ke level dan user yang memang diperuntukkan hanya kepada admin saja. Untuk fitur lainnya akan berbagi hak akses dengan user lainnya.

2. Membuat Authorization untuk Admin dan Manager

```
Route::middleware(middleware: ['authorize:ADM,MNG'])->group(callback: function(): void{
    Route::group(attributes: ['prefix' => 'supplier'], routes: function(): void{
        Route::get(uri: '/', action: [SupplierController::class, 'index']);
        Route::post(uri: '/list', action: [SupplierController::class, 'list']);
        Route::get(uri: '/create', action: [SupplierController::class, 'create']);
        Route::post(uri: '/', action: [SupplierController::class, 'store']);
        Route::get(uri: '/create_ajax', action: [SupplierController::class, 'create_ajax']);
        Route::post(uri: '/ajax', action: [SupplierController::class, 'store_ajax']);
        Route::get(uri: '/{id}/show_ajax', action: [SupplierController::class, 'show_ajax']);
        Route::get(uri: '/{id}', action: [SupplierController::class, 'show']);
        Route::get(uri: '/{id}/edit', action: [SupplierController::class, 'edit']);
        Route::put(uri: '/{id}', action: [SupplierController::class, 'update']);
        Route::get(uri: '/{id}/edit_ajax', action: [SupplierController::class, 'edit_ajax']);
        Route::put(uri: '/{id}/update_ajax', action: [SupplierController::class, 'update_ajax']);
        Route::get(uri: '/{id}/delete_ajax', action: [SupplierController::class, 'confirm_ajax']);
        Route::delete(uri: '/{id}/delete_ajax', action: [SupplierController::class, 'delete_ajax']);
        Route::delete(uri: '/{id}', action: [SupplierController::class, 'destroy']);
    });
});
```

- Disini hanya ada satu menu yang bisa diakses hanya admin dan manager, yaitu data supplier. Sementara user staff dan customer tidak bisa mengakses nya.

3. Membuat authorization untuk Admin, Manager, dan Staff

```
Route::middleware(middleware: ['authorize:STF,ADM,MNG'])->group(callback: function(): void{
    Route::group(attributes: ['prefix' => 'barang'], routes: function(): void{
        Route::get(uri: '/', action: [BarangController::class, 'index']); // Menampilkan
        Route::post(uri: '/list', action: [BarangController::class, 'list']); // Menampilkan
        Route::get(uri: '/create', action: [BarangController::class, 'create']); // Menampilkan
        Route::post(uri: '/', action: [BarangController::class, 'store']); // Menyimpan da
        Route::get(uri: '/create_ajax', action: [BarangController::class, 'create_ajax']); // Menampil
        Route::post(uri: '/ajax', action: [BarangController::class, 'store_ajax']); // Menyimpan data
        Route::get(uri: '/{id}/show_ajax', action: [BarangController::class, 'show_ajax']); // menamp
        Route::get(uri: '/{id}', action: [BarangController::class, 'show']); // Menampilkan
        Route::get(uri: '/{id}/edit', action: [BarangController::class, 'edit']); // Menampilkan
        Route::put(uri: '/{id}', action: [BarangController::class, 'update']); // Menyimpan pa
        Route::get(uri: '/{id}/edit_ajax', action: [BarangController::class, 'edit_ajax']); //
        Route::put(uri: '/{id}/update_ajax', action: [BarangController::class, 'update_ajax']); //
        Route::get(uri: '/{id}/delete_ajax', action: [BarangController::class, 'confirm_ajax']); // Me
        Route::delete(uri: '/{id}/delete_ajax', action: [BarangController::class, 'delete_ajax']); //
        Route::delete(uri: '/{id}', action: [BarangController::class, 'destroy']); // Menghapus da
    });
});

Route::group(attributes: ['prefix' => 'kategori'], routes: function(): void{
    Route::get(uri: '/', action: [KategoriController::class, 'index']); // Menampilkan
    Route::post(uri: '/list', action: [KategoriController::class, 'list']); // Menampilkan
    Route::get(uri: '/create', action: [KategoriController::class, 'create']); // Menampilkan
    Route::post(uri: '/', action: [KategoriController::class, 'store']); // Menyimpan
    Route::get(uri: '/create_ajax', action: [KategoriController::class, 'create_ajax']); // Menampil
    Route::post(uri: '/ajax', action: [KategoriController::class, 'store_ajax']); // Menyimpan data
    Route::get(uri: '/{id}/show_ajax', action: [KategoriController::class, 'show_ajax']); // menamp
    Route::get(uri: '/{id}', action: [KategoriController::class, 'show']); // Menampilkan
    Route::get(uri: '/{id}/edit', action: [KategoriController::class, 'edit']); // Menampilkan
    Route::put(uri: '/{id}', action: [KategoriController::class, 'update']); // Menyimpan
    Route::get(uri: '/{id}/edit_ajax', action: [KategoriController::class, 'edit_ajax']); //
    Route::put(uri: '/{id}/update_ajax', action: [KategoriController::class, 'update_ajax']); //
    Route::get(uri: '/{id}/delete_ajax', action: [KategoriController::class, 'confirm_ajax']); // Me
    Route::delete(uri: '/{id}/delete_ajax', action: [KategoriController::class, 'delete_ajax']); //
    Route::delete(uri: '/{id}', action: [KategoriController::class, 'destroy']); // Menghapus
});
});
```

- Terdapat 2 menu yang dapat diakses oleh 3 user yaitu menu data barang dan kategori barang.

Tugas 4 : Implementasi Form Registrasi

1. Silahkan implementasikan form untuk registrasi user.

- Routes

```
Route::get(uri: 'register', action: [AuthController::class, 'register'])->name(name: 'register');
Route::post(uri: 'register', action: [AuthController::class, 'postregister']);
```

- Authcontroller

```
Tabnine | Edit | Test | Explain | Document | 1 reference | 0 overrides
public function register(): Factory|Redirector|RedirectResponse|View{
    if(Auth::check()){ // jika sudah login, maka redirect ke halaman home
        return redirect(to: '/');
    }

    $level = LevelModel::select(columns: 'level_id','level_nama')->get();

    return view(view: 'auth.register', data: ['level' => $level]);
}

Tabnine | Edit | Test | Explain | Document | 1 reference | 0 overrides
public function postregister(Request $request): JsonResponse|mixed{
    if($request->ajax() || $request->wantsJson()){
        $rules = [
            'level_id' => 'required|integer',
            'username' => 'required|string|min:3|unique:m_user,username',
            'nama' => 'required|string|max:100',
            'password' => 'required|min:5'
        ];

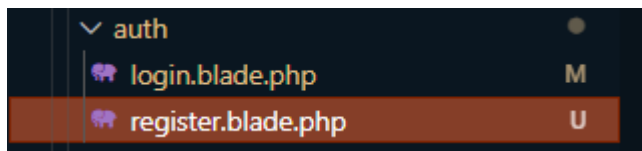
        $validator = Validator::make(data: $request->all(), rules: $rules);

        if ($validator->fails()) {
            return response()->json(data: [
                'status' => false,
                'message' => $validator->errors()->first(),
                'msgField' => $validator->errors()
            ]);
        }

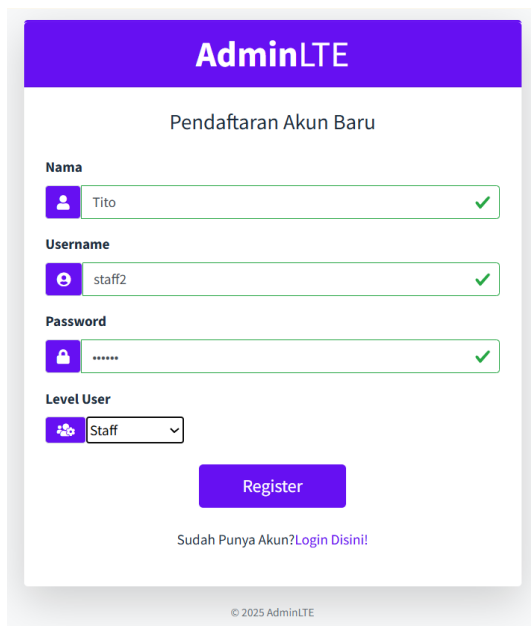
        UserModel::create(attributes: $request->all());

        return response()->json(data: [
            'status' => true,
            'message' => 'Data user berhasil disimpan',
            'redirect' => url(path: '/login'),
        ]);
    }
}
```

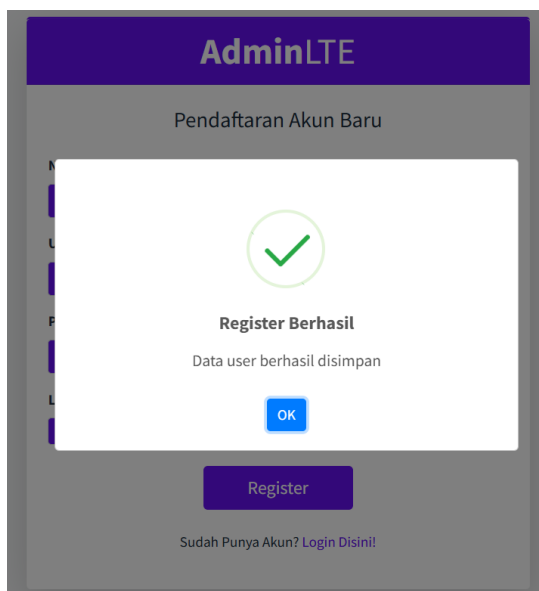
- View register



2. Screenshot hasil yang kalian kerjakan



The screenshot shows the 'AdminLTE' registration page. The title is 'Pendaftaran Akun Baru'. The form includes fields for 'Nama' (filled with 'Tito'), 'Username' (filled with 'staff2'), and 'Password' (masked with '*****'). A 'Level User' dropdown is set to 'Staff'. A green checkmark is visible at the end of each input field. A purple 'Register' button is at the bottom. Below the button, it says 'Sudah Punya Akun? [Login Disini!](#)'. The footer shows '© 2025 AdminLTE'.



3. Commit dan push hasil tugas kalian ke masing-masing repo github kalian (Sudah)