

# PANDUAN SINKRONISASI WEBSITE KE PORTAL AJI



## 0. SESUAIKAN DATABASE TABEL USER,DEPARTEMENT,DETAIL DEPARTEMENT DAN POSISI SEPERTI MILIK WEB WEB AJI

Ketentuan database table dapat melihat halaman 6 sampai 8

## 1. MENAMBAHKAN URL API DI .ENV

API\_BASE\_URL = <http://10.14.143.89/portal-AJI/public> (SESUAIKAN URL WEB MAIN AJI, tanpa "/" diakhir)

## 2. MEMBUAT CONTROLLER BARU UNTUK LOGIN

php artisan make:controller LoginAJIController

## 3. MASUK KE CONTROLLER DAN BUAT FUNCTION UNTUK LOGIN DARI WEB PORTAL AJI

Source Code Controller 'loginPortalAJI' ada pada halaman ke 2 sampai 4

## 4. BUAT ROUTE LOGIN API DI ROUTE WEB

Route::post('/loginPortalAJI', [LoginApiController::class, 'loginPortalAJI'])->name('EHS.Patrol.loginPortalAJI');

## 6. TAMBAHKAN ROUTE PADA ROUTE API UNTUK MENERIMA UPDATE DATA DARI WEB AJI

Route::post('/updateDataAPI',[LoginApiController::class, 'updateDataAPI'])->name('updateDataAPI');

## 7. BUAT FUNCTION "updatedataapi" PADA "loginajicontroller"

Code untuk Route updateDataApi ada di controller LoginApiController pada halaman ke 2 sampai 4

## 8. MATIKAN CRSF UNTUK ROUTE "/loginPortalAJI" DAN ROUTE API DI "Middleware/VerifyCsrfToken.php"

Code untuk Middleware/VerifyCsrfToken.php ada di halaman 5

## 9. DAFTARKAN ROLE,PERMISSION(JIKA ADA) DAN URL LINK WEBSITE ANDA KE ADMIN WEBSITE PORTAL AJI, SELESAI.

NOTE :

SEBELUM SINKRONISASI WEBSITE DISARANKAN UNTUK MEMBACKUP TERLEBIH DAHULU DATA DB DAN PROJECTNYA!

# Source Code Controller 'loginPortalAJI"

NOTE :

SESUAIKAN BESAR KECIL DAN EJAAN PENULISAN MODEL!

```
<?php

namespace App\Http\Controllers;

use App\Models\department;
use App\Models\detail_departement;
use App\Models\position;
use App\Models\User;
use GuzzleHttp\Client;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Illuminate\Support\Facades\Hash;
use Illuminate\Support\Facades\Log;
use Spatie\Permission\Models\Permission;
use Spatie\Permission\Models\Role;

class LoginApiController extends Controller
{
    public function loginPortalAJI(Request $request)
    {
        // Mengambil token yang dikirim dari Portal AJI
        $encryptedToken = $request->token; // Mengambil token yang telah dienkripsi dari permintaan

        $token = base64_decode($encryptedToken); // Mendekode token yang dienkripsi menggunakan base64
        $client = new Client(); // Membuat instance baru dari Client untuk melakukan permintaan HTTP

        try {
            $response = $client->get(env('API_BASE_URL') . '/api/user', [
                'headers' => [
                    'Authorization' => 'Bearer ' . $token,
                    'Accept' => 'application/json',
                ],
            ]);

            $data = json_decode($response->getBody()->getContents(), true);

            if (isset($data['status']) && $data['status'] === 'success') {
                $userData = $data['user'];

                $user = User::where('username', $userData['username'])->first();
                if ($user) {
                    Auth::loginUsingId($user->id);
                    $request->session()->regenerate();
                    //redirect ke halaman utama web (diganti sesuai website)
                    return redirect()->intended('/dashboard');
                }

                // Jika autentikasi gagal, redirect kembali ke web 1 dengan pesan error
                return redirect()->away(env('API_BASE_URL') . '?error=Password atau Username Salah');
            } else {
                return redirect()->away(env('API_BASE_URL') . '?error=Request API Gagal');
            }
        } catch (\Exception $e) {
            // Jika terjadi kesalahan, redirect kembali ke web 1 dengan pesan error
            return redirect()->away(env('API_BASE_URL') . '?error=Terjadi Kesalahan dalam masuk sub Website, Pastikan data sudah di Update');
        }
    }
}
```

# Source Code Controller 'loginPortalAJI'

```
public function updateDataAPI(Request $request)
{
    try {
        // Validasi input data
        $request->validate([
            'users' => 'required|array',
            'departments' => 'required|array',
            'detail_departments' => 'required|array',
            'positions' => 'required|array',
        ]);

        // Mengupdate model Department
        foreach ($request->input('departments') as $deptData) {
            department::updateOrCreate(
                ['id' => $deptData['id']], // Sesuaikan dengan kolom unik
                [
                    'code' => $deptData['code'],
                    'name' => $deptData['name'],
                    'email_depthead' => $deptData['email_depthead'] ?? null,
                    'email_spv' => $deptData['email_spv'] ?? null,
                    'email_members' => $deptData['email_members'] ?? null,
                    'deleted_at' => $deptData['deleted_at'] ?? null,
                ],
            );
        }

        // Mengupdate model Detail_departement
        foreach ($request->input('detail_departments') as $detailDeptData) {
            detail_departement::updateOrCreate(
                ['id' => $detailDeptData['id']], // Sesuaikan dengan kolom unik
                [
                    'departement_id' => $detailDeptData['departement_id'],
                    'name' => $detailDeptData['name'],
                    'code' => $detailDeptData['code'] ?? null,
                    'email_depthead' => $detailDeptData['email_depthead'] ?? null,
                    'email_director' => $detailDeptData['email_director'] ?? null,
                    'email_spv' => $detailDeptData['email_spv'] ?? null,
                    'email_members' => $detailDeptData['email_members'] ?? null,
                ],
            );
        }

        // Mengupdate model Position
        foreach ($request->input('positions') as $positionData) {
            position::updateOrCreate(
                ['id' => $positionData['id']], // Sesuaikan dengan kolom unik
                [
                    'position' => $positionData['position'],
                    'code' => $positionData['code'],
                ],
            );
        }

        // Ambil data roles dan permissions dari request
        $allRole = $request->input('allRole');
        $allPermission = $request->input('allPermission');
        // Bersihkan semua data role yang ada
        Role::query()->delete();

        // Bersihkan semua data permission yang ada
        Permission::query()->delete();
    }
}
```

# Source Code Controller 'loginPortalAJI'

```
foreach ($allPermission as $permissionData) {
    Permission::firstOrCreate(['name' => $permissionData['name']]);
}

foreach ($allRole as $roleData) {
    $newRole = Role::firstOrCreate([
        'name' => $roleData['name'],
    ]);
}

$permissions = [];
foreach ($roleData['permissions'] as $permissionData) {
    $NewPermissions = Permission::firstOrCreate([
        'name' => $permissionData['name'],
    ]);
    $permissions[] = $NewPermissions['name'];
}
// Sync permissions
$newRole->givePermissionTo($permissions);
}

// Mengupdate model User
foreach ($request->input('users') as $userData) {
    if ($userData['deleted_at'] != null) {
        $deleteUser = User::where('npk', $userData['npk'])->first();
        if ($deleteUser) {
            $deleteUser->delete();
        }
    } else {
        $user = User::updateOrCreate(
            ['npk' => $userData['npk']],
            [
                'name' => $userData['name'],
                'email' => $userData['email'],
                'npk' => $userData['npk'] ?? null,
                'username' => $userData['username'] ?? null, // Tambahkan null jika tidak ada
                'gender' => $userData['gender'] ?? null,
                'tgl_masuk' => $userData['tgl_masuk'] ?? null,
                'tgl_lahir' => $userData['tgl_lahir'] ?? null,
                'dept_id' => $userData['dept_id'] ?? null,
                'position_id' => $userData['position_id'] ?? null,
                'detail_dept_id' => $userData['detail_dept_id'] ?? null,
                'golongan' => $userData['golongan'] ?? null,
                'email_verified_at' => $userData['email_verified_at'] ?? null,
                'password' => isset($userData['password']) ? $userData['password'] : null,
            ],
        );
    }
    // Mengupdate roles dan permissions jika ada
    if (isset($userData['role']) || isset($userData['permissions'])) {
        $user->syncRoles($userData['role'] ?? []);
        $user->syncPermissions($userData['permissions'] ?? []);
    }
}

return response()->json(['success' => 'Data updated successfully']);
} catch (\Exception $e) {
    // Log error untuk pengecekan jika dibutuhkan
    Log::error('Data update error: ' . $e->getMessage());
}

return response()->json(['error' => $e->getMessage()]);
}
}
```

# Code untuk Middleware/VerifyCsrfToken.php

```
<?php

namespace App\Http\Middleware;

use Illuminate\Foundation\Http\Middleware\VerifyCsrfToken as Middleware;

class VerifyCsrfToken extends Middleware
{
    /**
     * The URIs that should be excluded from CSRF verification.
     *
     * @var array
     */
    protected $except = [
        '/loginPortalAJI',
        '/api/updateDataAPI'
    ];
}
```

## Ketentuan Tabel 'departments' (migrations)

```
Schema::create('departments', function (Blueprint $table) {
    $table->id();
    $table->string("code");
    $table->string("name");
    $table->string("email_depthead")->nullable();
    $table->string("email_spv")->nullable();
    $table->string("email_members")->nullable();
    $table->timestamps();
    $table->timestamp("deleted_at")->nullable();
});
```

## Ketentuan Tabel 'detail\_departements' (migrations)

```
Schema::create('detail_departements', function (Blueprint $table) {
    $table->id();
    $table->integer("departement_id")->references('id')->on('departments');
    $table->string("name");
    $table->string("code")->nullable();
    $table->string("email_depthead")->nullable();
    $table->string("email_director")->nullable();
    $table->string("email_spv")->nullable();
    $table->string("email_members")->nullable();
    $table->timestamps();
});
```

## Ketentuan Tabel 'positions' (migrations)

```
Schema::create('positions', function (Blueprint $table) {
    $table->id();
    $table->string("position");
    $table->string("code");
    $table->timestamps();
});
```

## Ketentuan Tabel Users (migrations)

```
Schema::create('users', function (Blueprint $table) {
    $table->id();
    $table->string('name');
    $table->string('email')->unique();
    $table->string('npk')->unique()->nullable();
    $table->string('username')->unique();
    $table->string('gender')->nullable();
    $table->date('tgl_masuk')->nullable();
    $table->date('tgl_lahir')->nullable();
    $table->integer('dept_id')->references('id')->on('departement')->nullable();
    $table->integer('position_id')->references('id')->on('position')->nullable();
    $table->integer('detail_dept_id')->references('id')->on('detail_departement')->nullable();
    $table->string('golongan')->nullable();
    $table->timestamp('email_verified_at');
    $table->string('password');
    $table->rememberToken();
    $table->timestamps();
    $table->string("deleted_at")->nullable();
});
```

### Catatan :

1. Pastikan pembuatan tabel data urut ('departments'->'detail\_departements'->'positions'->'users')
2. Pada Model cukup gunakan Guarded ke id

```
protected $guarded = [
    'id'
];
```

3. Pastikan sudah mengunduh spatie untuk role dan permission updatenya.
4. sesuaikan code controller dengan website anda (modal,relasi,dll)
5. Tambahkan Tombol kembali ke portal pada Navigasi Website Anda

Untuk Code dapat dilihat di halaman 10.

6. Link File sample konfigurasi : [ungu.in/sample-sinkronisasi-web-aji](http://ungu.in/sample-sinkronisasi-web-aji)

# Default Relasi Model User, Department , Detail Department dan Position.

## 1. RELASI DI MODEL USER

```
public function department(){
    return $this->belongsTo(Department::class, 'dept_id');
}

public function position(){
    return $this->belongsTo(Position::class, 'position_id');
}

public function detailDepartment(){
    return $this->belongsTo(Detail_departement::class, 'detail_dept_id');
}
```

## 2. RELASI DI MODEL DEPARTMENT

```
public function detail() {
    return $this->belongsTo(detail_departement::class, 'departement_id');
}

public function user() {
    return $this->hasMany(user::class, 'dept_id');
}
```

## 3. RELASI DI MODEL DETAIL DEPARTEMEN

```
public function detail() {
    return $this->hasMany(department::class, 'departement_id');
}

public function user() {
    return $this->hasMany(user::class, 'detail_dept_id');
}
```

## 4. RELASI DI MODEL POSITION

```
public function user() {
    return $this->hasMany(user::class, 'position_id');
}
```

## Tambahan : Membuat Command untuk mengirim permissions bedasarkan route name ke Web Main AJI.

### 1. MEMBUAT COMMAND DI WEBSITE ANDA

```
php artisan make:command RouteNameToPermission
```

### 2. MEMBUKA FILE "RouteNameToPermission.php" DAN ATUR COMMAND SEPERTI BERIKUT

```
protected $signature = 'routes:send-api';
protected $description = 'Send all route names with web middleware to the main website';
public function handle()
{
    // Mendapatkan semua route yang terdaftar
    $routes = Route::getRoutes()->getRoutes();

    foreach ($routes as $route) {
        // Memeriksa apakah route memiliki nama dan menggunakan middleware web
        if ($route->getName() != "" && $route->getAction()['middleware'][0] == 'web') {
            // Mengirim data route ke API di website utama
            $response = Http::post(env('API_BASE_URL') . '/api/store-route', [
                'name' => $route->getName(),
            ]);

            if ($response->successful()) {
                $this->info("Route {$route->getName()} sent successfully to main website.");
            } else {
                $this->error("Failed to send route {$route->getName()} to main website.");
            }
        }
    }

    $this->info('All routes have been processed.');
}
```

### 3. JALANKAN COMMAND UNTUK MENGIRIM ROUTENAME KE PERMISSION WEB MAIN AJI

```
php artisan routes:send-api
```

## Membuat Tombol Kembali ke Website Portal AJI

### 1. BUKA FILE BLADE NAVIGASI ANDA LALU TAMBAHKAN BUTTON “KE PORTAL” DIBAWAH (DAPAT DILETAKKAN DISAMPING LOGOUT)

```
<a href="{{ route('to.portal') }}" class="p-0 pt-2"><button type="button" class="btn btn-secondary">Ke Portal</button></a>
```

### 2. MEMBUAT ROUTE “to.portal” DI ROUTE WEB (DISARANKAN ROUTENYA KE CONTROLLER YANG MENGATUR AUTH)

```
Route::get('/to-portal', [UserController::class, 'toPortal'])->name('to.portal');
```

### 3. MEMBUAT FUNCTION “toPortal” DI CONTROLLER

```
public function toPortal(Request $request){  
    $request->session()->invalidate();  
  
    $request->session()->regenerateToken();  
    Auth::logout();  
    return redirect(env('API_BASE_URL'));  
}
```

### 4. SELESAI