1. **Instalar laravel breeze**

$ composer require laravel/breeze --dev

$ php artisan breeze:install

$ php artisan migrate

$ npm install

$ npm run dev

1. **Crear Middleware para Roles**

Crear middleware:

php artisan make:middleware RoleMiddleware

Agregar código php…

```
   public function handle(Request $request, Closure $next, ...$roles)
    {
        // Verifica si el usuario está autenticado
        if (!Auth::check()) {
            return redirect('/login'); // Redirige si no está logueado
        }

        // Obtiene el rol del usuario autenticado
        $userRole = Auth::user()->role->name ?? null; // Asegúrate de
tener la relación con roles en el modelo User

        // Verifica si el rol del usuario está permitido
        if (!in_array($userRole, $roles)) {
            return abort(404, 'No tienes permiso para acceder a esta
página.');
        }

        return $next($request);
    }
}
```

Luego Agregar middleware a kernel.php

En App/Http\kernel.php agregar en

protected $middlewareAliases = [:

'role' => \App\Http\Middleware\RoleMiddleware::class,

Establecer el middleware role en las rutas

```
// 🔹 Rutas accesibles solo para ADMINISTRADOR
Route::middleware(['auth', 'role:admin'])->group(function () {
    Route::get('/admin/dashboard', [AdminController::class, 'index'])->name('admin.dashboard');
});
```

Establecer el middleware role en los controladores:

Antes implementar la clase Auth:

```
use Illuminate\Support\Facades\Auth;
```

Agregar el construct():

```
 public function __construct()
    {
        //Solo el ADMINISTRADOR y MANTENEDOR pueden acceder a este
controlador
        $this->middleware(['auth', 'role:Administrador,OtroRol']);
    }
```

Proteger los menus del template en base al role:

```
@if (Auth::check() && in_array(Auth::user()->role->name,
['Administrador','Mantenedor']))
```

...codigo del menú

```
 @endif
```

Solucionando problema de redirección cuando se accede a la ruta raíz:

Usar un middleware para redirigir

Si prefieres no manejar la redirección en el controlador, puedes crear un middleware personalizado para redirigir a los usuarios según su rol. Esto es útil si deseas aplicar esta lógica en varias rutas.

1. Crear el middleware

Ejecuta el siguiente comando para crear un middleware:

```
php artisan make:middleware RedirectByRole
```

Luego, en el archivo generado (app/Http/Middleware/RedirectByRole.php), agrega la lógica de redirección:

```
namespace App\Http\Middleware;

use Closure;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;

class RedirectByRole
{
   public function handle(Request $request, Closure $next)
   {
      // Verificar si el usuario está autenticado
      if (Auth::check()) {

         $user = Auth::user();

         // Redirigir según el rol del usuario
         if ($user->role->name === 'Administrador') {
            return redirect()->route('dashboard');

         } elseif ($user->role->name === 'Cliente') {

            return redirect()->route('devices.index');
         }
      }

      // Si el usuario no está autenticado, continuar con la solicitud
      return $next($request);
   }
}
```

2. Registrar el middleware

En app/Http/Kernel.php, registra el middleware en el arreglo $routeMiddleware:

```
protected $ middlewareAliases = [
   // Otros middlewares
```

```
    'redirect.by.role' => \App\Http\Middleware\RedirectByRole::class,
];
```

3. Aplicar el middleware a la ruta raíz

En routes/web.php, aplica el middleware a la ruta raíz:

```
Route::get('/', [HomeController::class, 'index'])
    ->middleware('redirect.by.role')
    ->name('home');
```

4. Modificar el método index del HomeController

Ahora, el método index puede simplemente renderizar una vista, ya que el middleware se encargará de la redirección:

```
public function index()
{
    // Renderizar la vista de bienvenida
    return view(dashboard);
}
```