

# Guardianes en Angular



## Guardianes en Angular

En esta lección, aprenderemos cómo agregar guardianes a nuestra aplicación para proteger las rutas y asegurarnos de que los usuarios no autenticados no puedan acceder a ciertas secciones.

### Paso 1: Crear el Guardián de Autenticación

Primero, crearemos el servicio de autenticación, en la carpeta de servicios:

```
ng g s servicios/login-guardian --skip-tests
```

Código del archivo `login-guardian.service.ts`:

```
import { Injectable } from '@angular/core';
import { authState, Auth } from '@angular/fire/auth';
import { CanActivate, Router } from '@angular/router';
import { map, Observable } from 'rxjs';

@Injectable({
  providedIn: 'root'
})
export class LoginGuardianService implements CanActivate{
```

```

constructor(
  private authService: Auth,
  private router: Router
) { }

canActivate(): Observable<boolean> {
  return authState(this.authService).pipe(
    map(auth => !!auth || (this.router.navigate(['/login']), false))
  );
}
}

```

### Explicación del código:

- Operador `!!auth`:** Convierte el valor de `auth` en un booleano. Si el usuario está autenticado (es decir, `auth` existe), se convierte en `true`. Si no, será `false`.
- Ternario con acción de redirección:** En lugar de usar un `if`, se utiliza el operador `||`. Si `auth` es falso (es decir, el usuario no está autenticado), se redirige al usuario a la página de login (`this.router.navigate(['/login'])`) y se retorna `false`. Esto bloquea el acceso a la ruta protegida.

Este enfoque compacto mantiene la misma lógica de autenticación, pero de una forma más moderna y simplificada.

### Flujo de autenticación:

- Si el usuario está autenticado (`auth` tiene un valor), la expresión `!!auth` devuelve `true`.
- Si el usuario **no** está autenticado (`auth` es `null` o `undefined`), el guardián redirige al usuario a la página de login y bloquea el acceso retornando `false`.

### Código equivalente con `if`:

Si prefieres usar un `if` en lugar del operador `||`, el código sería el siguiente:

```

canActivate(): Observable<boolean> {
  return authState(this.authService).pipe(
    map((auth) => {
      if (!auth) {
        this.router.navigate(['/login']);
        return false;
      }
      return true;
    })
  )
}

```

```
 );
```

Ambas versiones son funcionalmente equivalentes, pero la primera opción es más moderna y compacta.

## Paso 2: Modificar el archivo `app.route.ts`

Ahora vamos a proteger nuestras rutas usando el guardián que acabamos de crear. Para ello, modificamos el archivo `app.route.ts`.

### Código del archivo `app.route.ts`:

```
import { Routes } from '@angular/router';
import { TableroComponent } from './componentes/tablero/tablero.component';
import { LoginComponent } from './componentes/login/login.component';
import { EditarClienteComponent } from './componentes/editar-cliente/editar-cliente.component';
import { NoEncontradoComponent } from './componentes/no-encontrado/no-encontrado.component';
import { LoginGuardianService } from './servicios/login-guardian.service';

export const routes: Routes = [
  {path: '', component: TableroComponent, canActivate: [LoginGuardianService]},
  {path: 'login', component: LoginComponent},
  {path: 'cliente/editar/:id', component: EditarClienteComponent,
    canActivate: [LoginGuardianService]},
  {path: '**', component: NoEncontradoComponent}
];
```

### Explicación del código:

- `canActivate: [LoginGuardianService]`: Añadimos esta propiedad a las rutas que queremos proteger. Esto asegura que solo los usuarios autenticados puedan acceder a esas rutas.
- Las rutas que hemos protegido son la ruta de inicio (`TableroComponent`) y la de edición de cliente (`EditarClienteComponent`).

## Resultado Final

Con este guardián en funcionamiento, las rutas de inicio y de edición de cliente ahora están protegidas. Si un usuario intenta acceder a estas rutas sin haber iniciado sesión, será redirigido automáticamente a la página de login.

The screenshot shows a web application interface. At the top, there is a header bar with the title "Control Clientes" and a "Login" button. Below this is a "Login" form with fields for "Email" (containing "ubaldoacosta@gmail.com") and "Password" (containing "....."). A "Login" button is at the bottom of the form. The background of the main content area is dark grey. Below the login form is a yellow footer bar with the text "© GlobalMentoring.com.mx". The main content area contains a "Control de Clientes" section with a "Listado de Clientes" table and two summary boxes: "Saldo Total" (\$600.00) and "Total Clientes" (3). The "Listado de Clientes" table has columns "Nombre" and "Saldo", listing three clients: Carlos Lara (\$300.00), Juan Perez (\$100.00), and Karla Gomez (\$200.00). Each row has an "Editar" button. The "Control de Clientes" section also includes a "Agregar Cliente" button. The bottom of the main content area has another yellow footer bar with the text "© GlobalMentoring.com.mx".

Recuerden que pueden descargar el código actualizado en la sección de recursos de esta lección.

¡Continúa así , el esfuerzo vale la pena!

Saludos

Ing. Ubaldo Acosta

Fundador de [GlobalMentoring.com.mx](http://www.globalmentoring.com.mx)