Guardianes en Angular

Implementación de Guardianes en Angular

En esta lección, vamos a implementar **guardianes** en nuestra aplicación de Angular para proteger las rutas. Los guardianes garantizan que solo los usuarios autenticados puedan acceder a determinadas rutas, redirigiendo a la página de login si es necesario.

Introducción a los Guardianes en Angular

¿Qué son los guardianes?

Los guardianes en Angular son servicios que pueden controlar el acceso a las rutas en la aplicación. Permiten definir condiciones para decidir si se puede navegar a una ruta o no. Este control es esencial para proteger rutas que requieren autenticación u otras validaciones antes de permitir el acceso.

Sintaxis Básica

Un guardián debe implementar una de las interfaces del módulo de rutas, como `CanActivate`, `CanDeactivate`, `CanLoad`, entre otras.

Ejemplo Básico de Guardián con `CanActivate`

```
import { Injectable } from '@angular/core';
import { CanActivate, Router } from '@angular/router';
import { LoginService } from './login.service';

@Injectable({
  providedIn: 'root',
})
export class LoginGuardianService implements CanActivate {
  constructor(private loginService: LoginService, private router: Router) {}

  // Verifica si el usuario está autenticado antes de activar la ruta
  canActivate(): boolean {
    if (this.loginService.isAutenticado()) {
      return true; // Permitir el acceso si está autenticado
    } else {
      this.router.navigate(['login']); // Redirigir al login si no está autenticado
      return false;
    }
  }
}
```

- **`CanActivate`**: Evalúa si se puede activar una ruta antes de permitir el acceso.
- **`authService.isLoggedIn()`**: Método que verifica si el usuario está autenticado.
- **Redirección**: Si no está autenticado, se redirige al login.

Aplicación del Guardián en las Rutas

Después de crear el guardián, debemos aplicarlo a las rutas en el archivo de configuración de rutas.

Ejemplo:

```
const routes: Routes = [
  { path: 'inicio', component: InicioComponent, canActivate: [AuthGuard] },
  { path: 'login', component: LoginComponent },
  { path: '**', redirectTo: 'login' },
];
```

En este caso, la ruta `/inicio` solo será accesible si el guardián `AuthGuard` devuelve `true`.

Implementación en nuestra Aplicación de Tienda Online

Paso 1: Crear el Servicio de Guardián de Login

Vamos a crear un servicio llamado `LoginGuardianService` que actuará como nuestro guardián de rutas para validar que el usuario esté autenticado antes de acceder a las páginas protegidas.

Comando para generar el servicio:

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

Código: `login-guardian.service.ts`

```
import { Injectable } from '@angular/core';
import { CanActivate, Router, MaybeAsync, GuardResult } from '@angular/router';
import { LoginService } from './login.service';

@Injectable({
  providedIn: 'root',
})
export class LoginGuardianService implements CanActivate {
  constructor(private loginService: LoginService, private router: Router) {}

  canActivate(): boolean {
    if (this.loginService.isAutenticado()) {
      return true;
    } else {
      this.router.navigate(['login']);
      return false;
    }
  }
}
```

Explicación del Código:

- **`canActivate()`**: Evalúa si se puede activar la ruta. Si el usuario está autenticado, retorna `true` y permite el acceso. De lo contrario, redirige al login y retorna `false`.

Paso 2: Asegurar las Rutas con el Guardián

Una vez que el guardián esté implementado, es necesario aplicarlo a las rutas que queremos proteger. Esto se hace mediante la propiedad `canActivate` en la configuración de rutas.

Código: `app.routes.ts`

```
import { Routes } from '@angular/router';
```

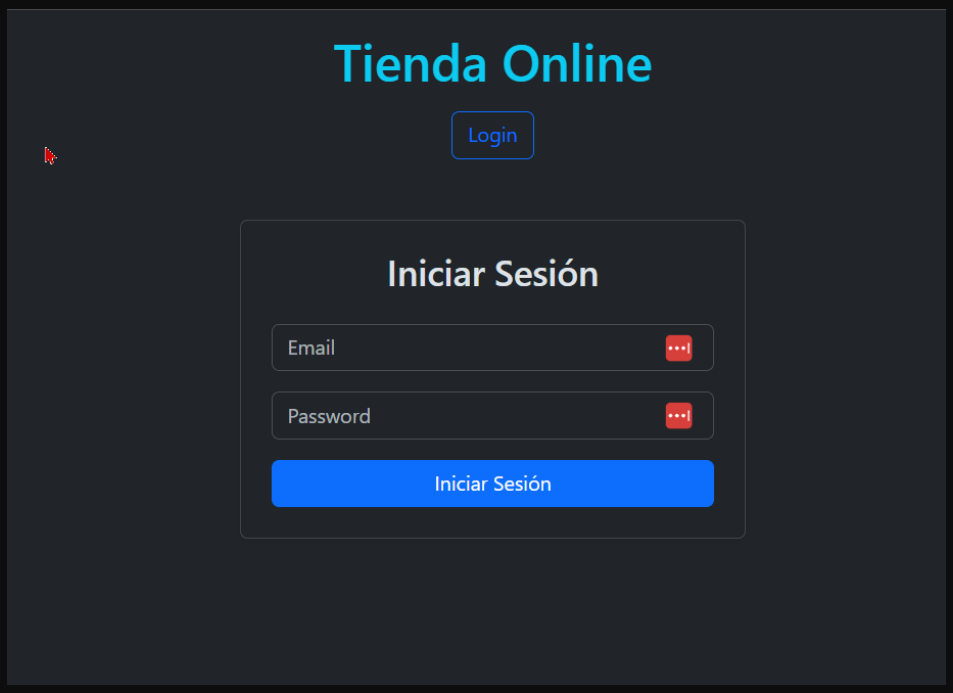
```
import { ListadoProductosComponent } from './listado-productos/listado-
productos.component';
import { FormularioComponent } from './formulario/formulario.component';
import { ErrorComponent } from './error/error.component';
import { LoginComponent } from './login/login.component';
import { LoginGuardianService } from './login-guardian.service';

export const routes: Routes = [
    {path: '', component: ListadoProductosComponent, canActivate:[LoginGuardianService]},
    {path: 'listado', component: ListadoProductosComponent,
canActivate:[LoginGuardianService]},
    {path: 'agregar', component: FormularioComponent,
canActivate:[LoginGuardianService]},
    {path: 'editar/:llave', component: FormularioComponent,
canActivate:[LoginGuardianService]},
    {path: 'login', component: LoginComponent},
    // Ruta comodín para cualquier otra ruta
    { path: '**', component: ErrorComponent }
];
```

Explicación del Código:

- **`canActivate: [LoginGuardianService]`**: Protege las rutas, asegurándose de que solo se puedan acceder si el usuario ha iniciado sesión.
- **Rutas Protegidas**: Las rutas `listado`, `agregar`, y `editar` están protegidas por el guardián. Solo se puede acceder a ellas si el usuario está autenticado.
- **Ruta de Login**: La ruta de `login` no está protegida por el guardián, ya que debe ser accesible sin autenticación.

Resultado Final:



Conclusión

En esta lección, hemos aprendido a implementar guardianes en Angular para proteger rutas. Los guardianes nos permiten controlar el acceso a diferentes secciones de la aplicación, garantizando que solo los usuarios autenticados puedan acceder a ciertas rutas. Esto mejora la seguridad de nuestra aplicación y la experiencia del usuario.

¡Saludos!

Ing. Ubaldo Acosta
Fundador de GlobalMentoring.com.mx