

**TECNOLÓGICO DE ESTUDIOS SUPERIORES DE ECATEPEC**

# DIVISIÓN DE INGENIERÍA EN SISTEMAS COMPUTACIONALES

**BASE DE DATOS PARA DISPOSITIVOS MÓVILES**

**ESTRUCTURA PRINCIPAL DEL MÓDULO**

**GRUPO: 5851**

**EQUIPO ##**

**PRESENTA:**

López Alcántara Alan Jesús

Lucio Alba Brandon Daniel

**PROFESOR:**

Cortes Barrera Griselda

ENERO 18, 2022

**BLOQUEO DE USUARIOS**

**REQUERIMIENTOS Y NECESIDADES DEL MÓDULO A TRABAJAR**

El presente módulo a trabajar surge de la necesidad de contemplar el acceso restringido a determinado tipo de usuarios que cuenten con los permisos y privilegios para la visualización y uso del espacio de trabajo de sitios específicos de la aplicación en curso. Así mismo como el **bloqueo** de usuarios que resulten denominados por la empresa por y estos no puedan tener ningún tipo de acceso a sitios de la aplicación y por tanto resguardar la información de la empresa.

Es posible enlistar las necesidades como:

1. Bloqueo de usuarios en el sistema.
2. Notificación de usuario no registrado (logueado).
3. Restringir el acceso a usuarios que no cuenten con permisos establecidos para determinados sitios.

Los requerimientos del módulo engloban principalmente:

* Una base de datos que describa los usuarios bloqueados con su identificador (correo).
* Un módulo de Login en donde implementar la funcionalidad del bloqueo en el proceso de acceso de la cuenta.
* Un servicio que permita bloquear el acceso de los usuarios a las rutas de redirección de los sitios.

**PROBLEMÁTICA**

La empresa que solicita el diseño de la aplicación nos describe la necesidad de mantener un acceso protegido a la información de la página por medio del acceso de las cuentas de usuarios. Es decir, es necesario mantener un registro de los usuarios que pueden tener acceso a determinados sitios de la aplicación y usuarios que definitivamente no puedan acceder al sistema por medio de un bloqueo.

Tomando en cuenta que la aplicación trata con diferentes tipos de usuario, se identifica que cada uno de ellos tiene acceso a determinado tipo de información y de espacio de trabajo, por lo tanto la funcionalidad del módulo permitirá que los usuarios únicamente tengan posibilidad de acceder a sus espacios correspondientes de trabajo. Así mismo como identificar usuarios que se encuentren bloqueados y no cuenten con acceso alguno al sistema.

**OBJETIVO GENERAL**

Crear un módulo en el cual se administrar usuarios ya sea de un portal escolar, empresa etc. Etiquetando a personal con ciertos permisos y restricciones dependiendo de su posición que se le otorgue.

**OBJETIVOS ESPECIFICOS**

* Identificar que funciones son las que ocuparemos para la implementación del modulo
* Diseñar un modulo de pruebas de dichas funciones
* Codificar para el proyecto final

**DISEÑO DE LA PROPUESTA**

Nuestra propuesta principal consiste en la implementación de una función en Angular denominada “Guard”, este tipo de middlewares funcionan como un bloqueador de rutas, es decir que cuando queremos acceder a algún sitio que nos redirija a otro sitio el Guard podrá bloquear esta ruta y evitar que el usuario tenga acceso. En sencillos pasos Guard funciona como un resultado booleano, responde true o false, si la respuesta es true se da acceso, de lo contrario se notifica al usuario que no puede acceder a ese sitio en general.

**¿Qué es Guard?**

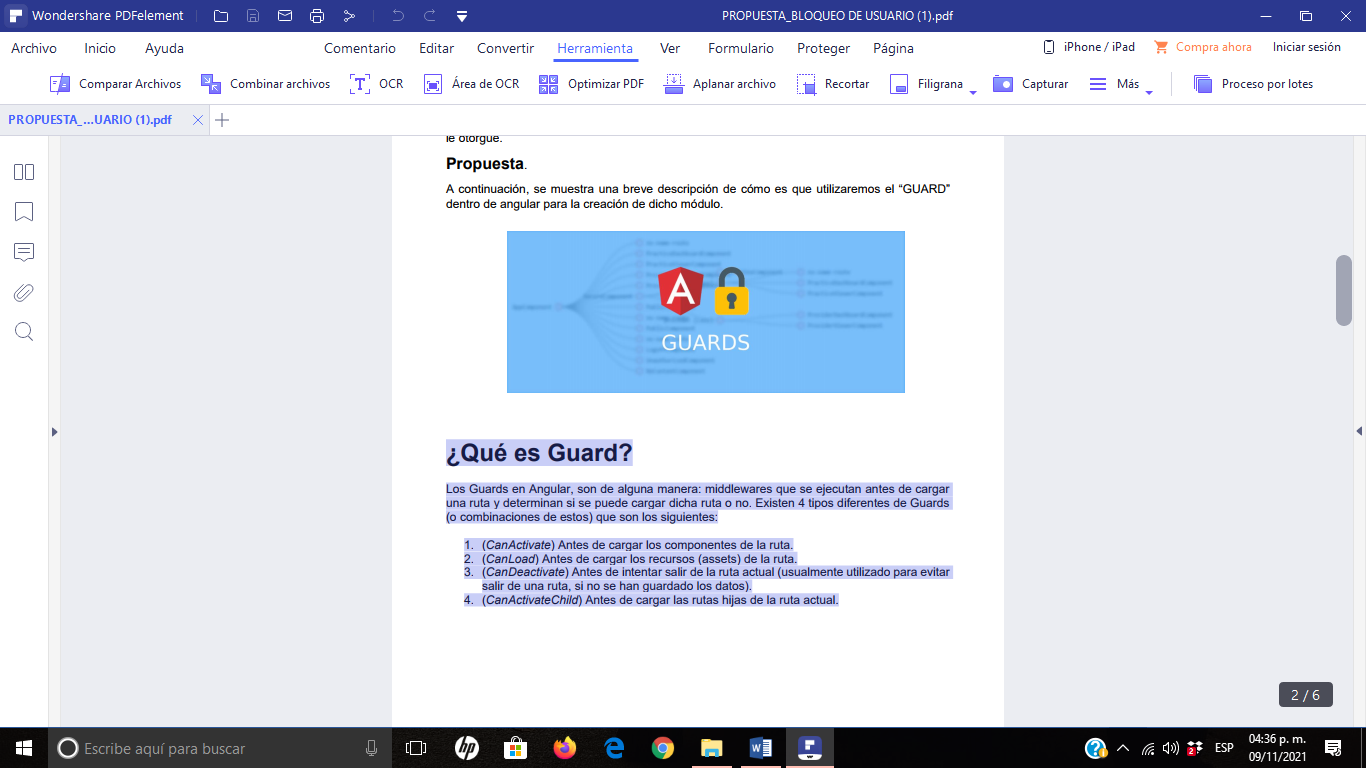
Los Guards en Angular, son de alguna manera: middlewares que se ejecutan antes de cargar una ruta y determinan si se puede cargar dicha ruta o no. Existen 4 tipos diferentes de Guards (o combinaciones de estos) que son los siguientes:

1. (**CanActivate**) Antes de cargar los componentes de la ruta.

2. (**CanLoad**) Antes de cargar los recursos (assets) de la ruta.

3. (**CanDeactivate**) Antes de intentar salir de la ruta actual (usualmente utilizado para evitar salir de una ruta, si no se han guardado los datos).

4. (**CanActivateChild**) Antes de cargar las rutas hijas de la ruta actual.

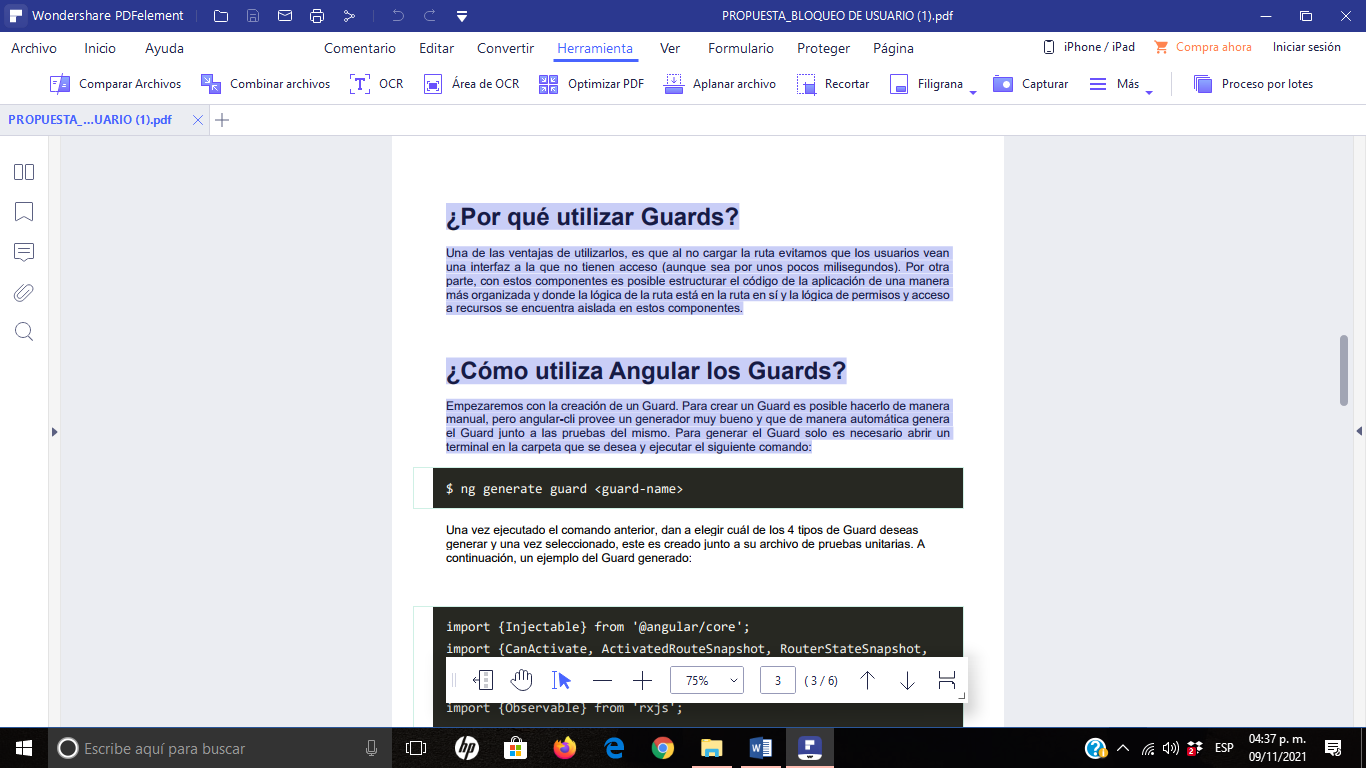


**¿Por qué utilizar Guards?**

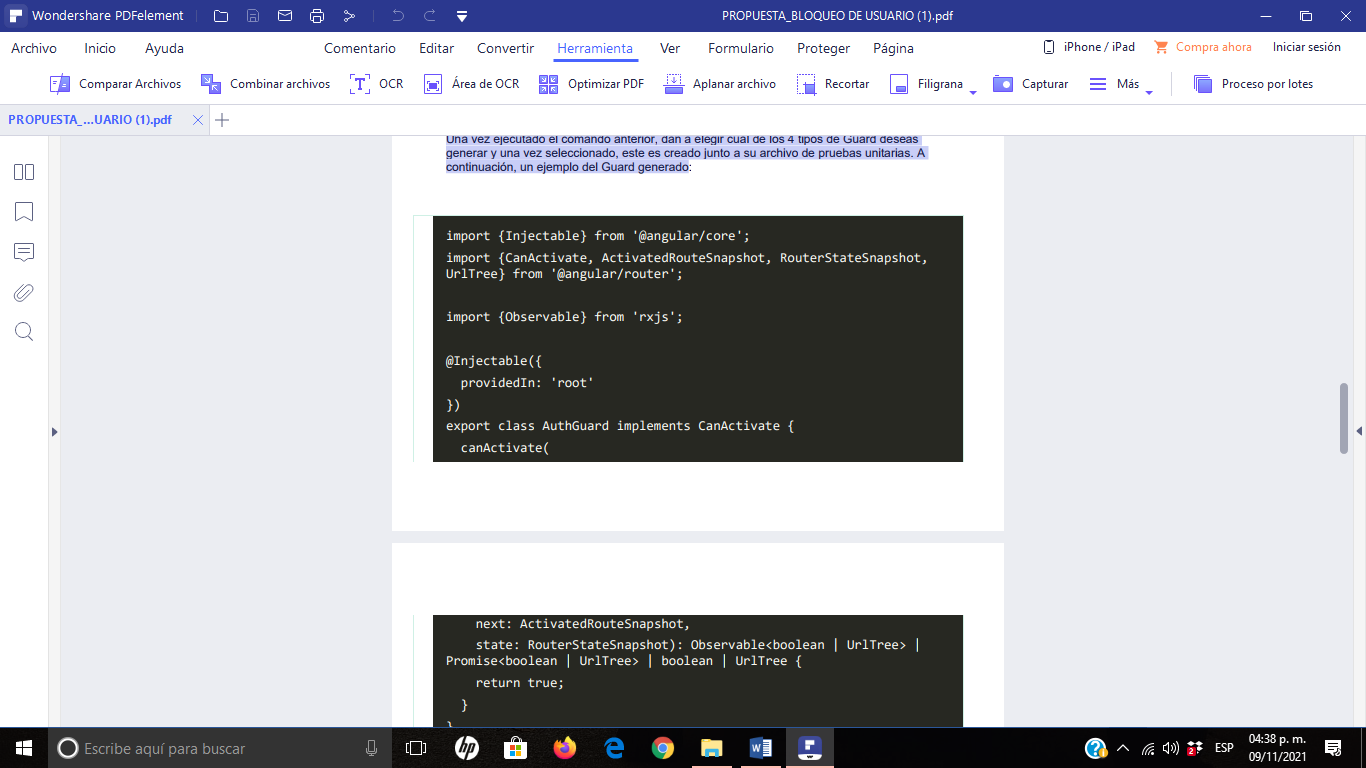
Una de las ventajas de utilizarlos, es que al no cargar la ruta evitamos que los usuarios vean una interfaz a la que no tienen acceso (aunque sea por unos pocos milisegundos). Por otra parte, con estos componentes es posible estructurar el código de la aplicación de una manera más organizada y donde la lógica de la ruta está en la ruta en sí y la lógica de permisos y acceso a recursos se encuentra aislada en estos componentes.

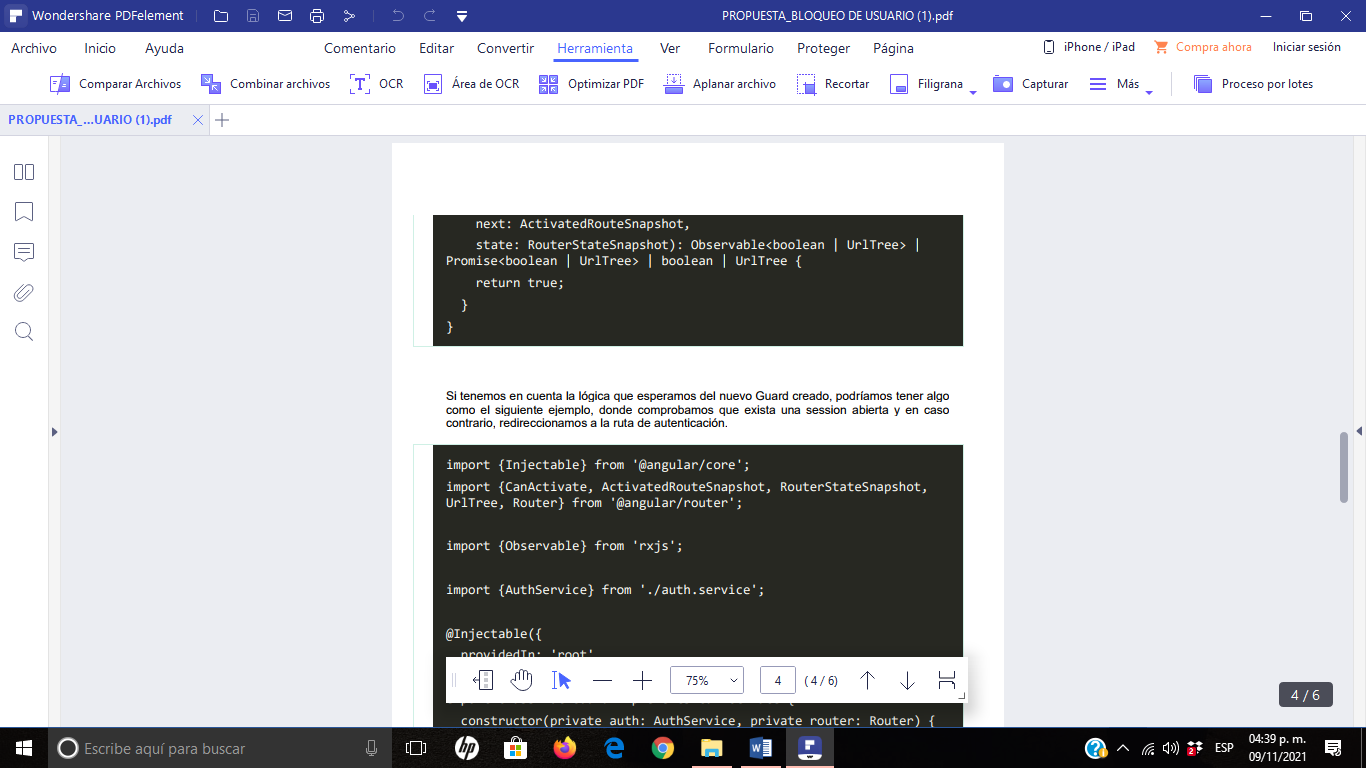
**¿Cómo utiliza Angular los Guards?**

Empezaremos con la creación de un Guard. Para crear un Guard es posible hacerlo de manera manual, pero angular-cli provee un generador muy bueno y que de manera automática genera el Guard junto a las pruebas del mismo. Para generar el Guard solo es necesario abrir un terminal en la carpeta que se desea y ejecutar el siguiente comando:

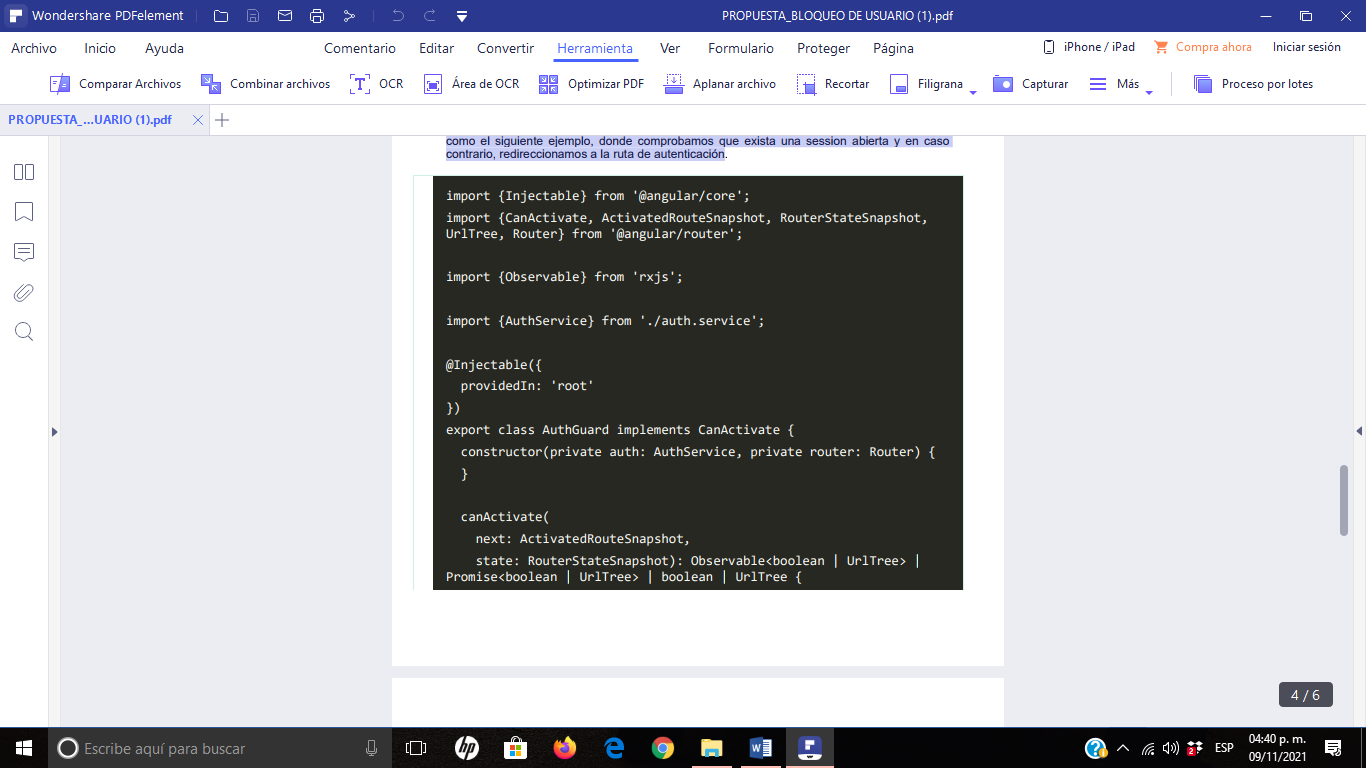


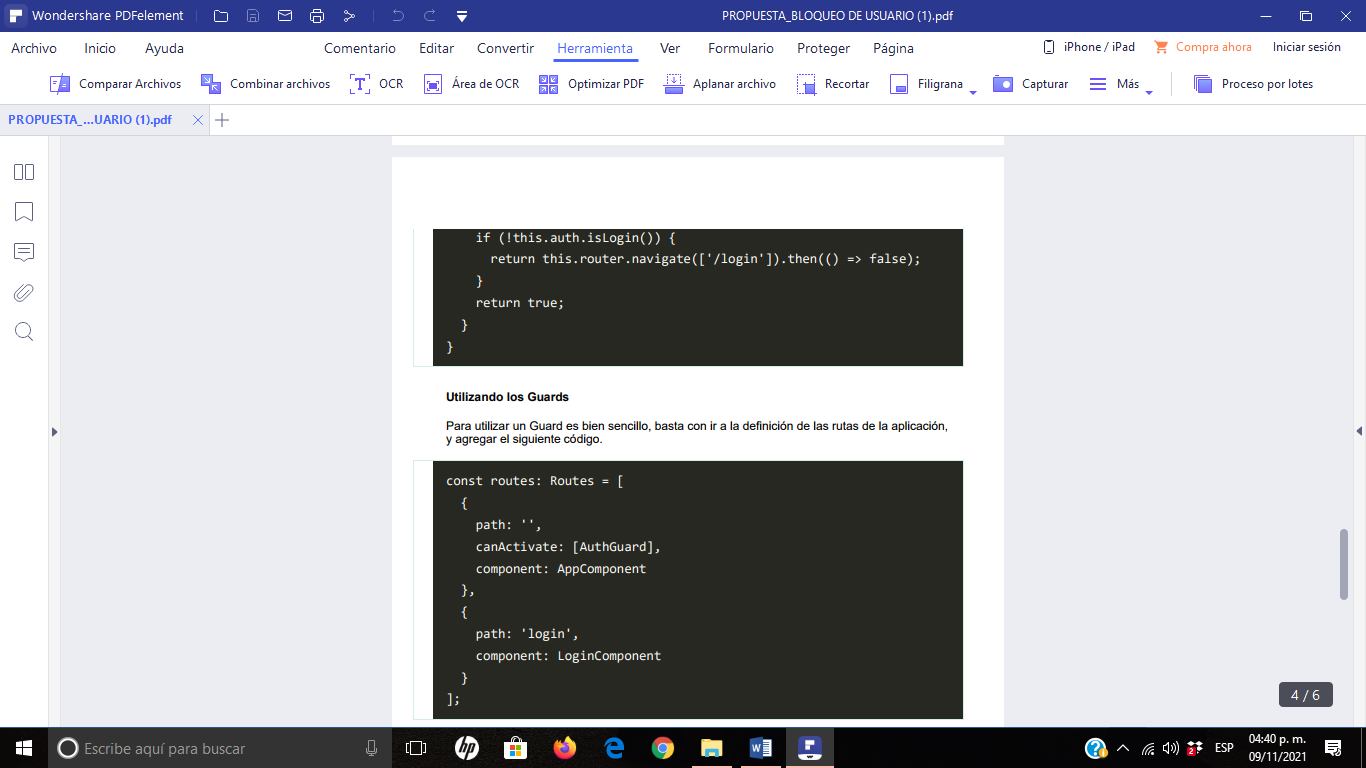
Una vez ejecutado el comando anterior, dan a elegir cuál de los 4 tipos de Guard deseas generar y una vez seleccionado, este es creado junto a su archivo de pruebas unitarias. A continuación, un ejemplo del Guard generado:





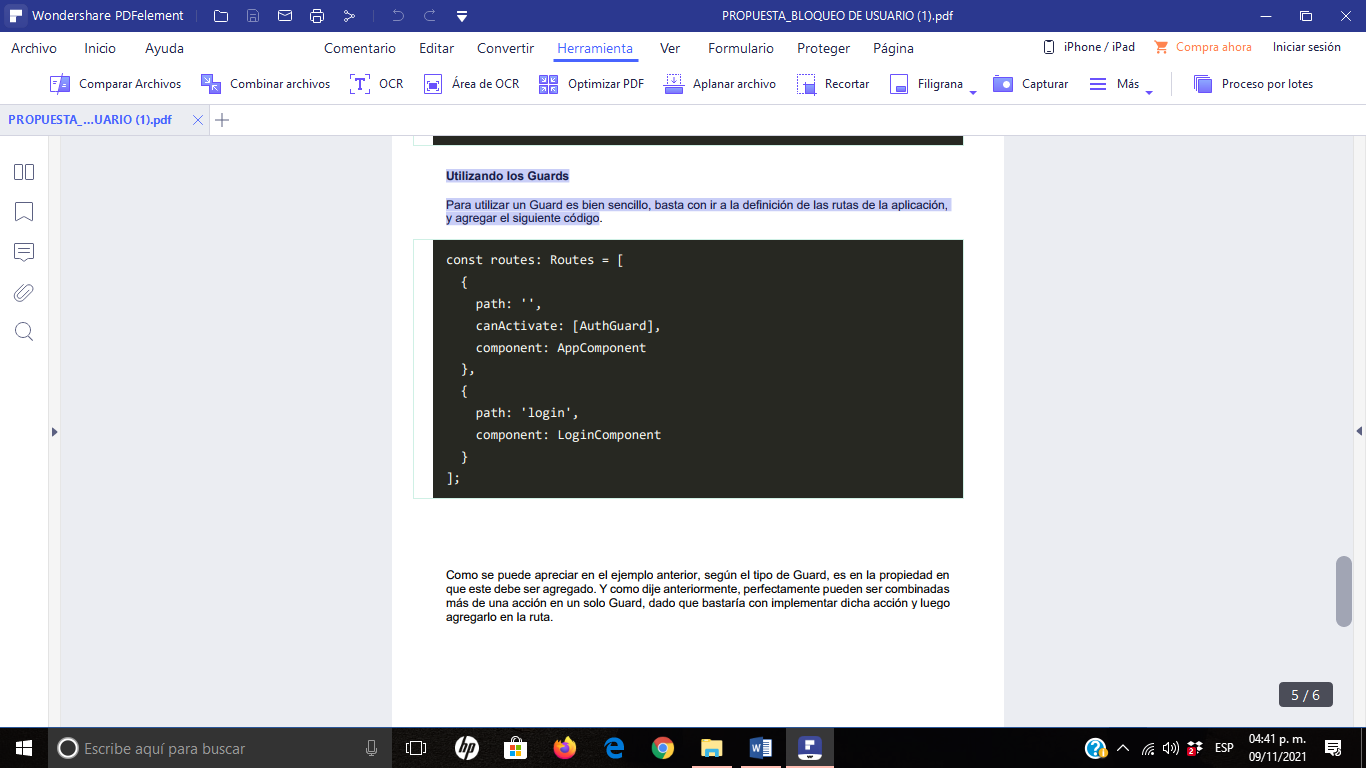
Si tenemos en cuenta la lógica que esperamos del nuevo Guard creado, podríamos tener algo como el siguiente ejemplo, donde comprobamos que exista una sesión abierta y en caso contrario, redireccionamos a la ruta de autenticación:





**Utilizando los Guards**

Para utilizar un Guard es bien sencillo, basta con ir a la definición de las rutas de la aplicación, y agregar el siguiente código.

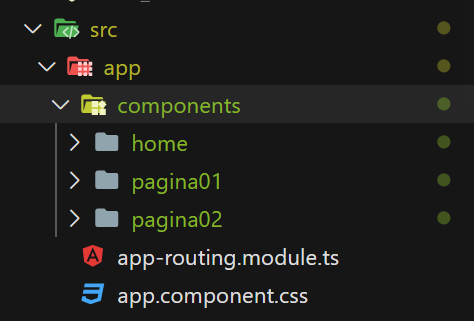


Como se puede apreciar en el ejemplo anterior, según el tipo de Guard, es en la propiedad en que este debe ser agregado. Y como dije anteriormente, perfectamente pueden ser combinadas más de una acción en un solo Guard, dado que bastaría con implementar dicha acción y luego agregarlo en la ruta.

**Ejemplo con un proyecto sencillo**

Para el ejemplo vamos a crear tres componentes, cada uno en una ruta diferente. Básicamente crearemos una pequeña navegación básica. Se puede crear de la forma que mejor te parezca, crear otro componente «Menú» o agregar un botón a cada componente que lleve a la otra ruta.

Vamos a nuestra consola y ejecutamos los comandos para hacerlo. ng g c components/pagina01, ng g c components/pagina02, ng g c components/home.



Ahora vamos al archivo app-routing.module.ts y definamos algunas rutas que nos lleven a cada uno de los [componentes](https://www.codigocorrecto.com/angular/correcto-uso-de-componentes-de-software/). Antes las crearemos sin el **CanActive**, para probar el funcionamiento normal



Tenemos 3 rutas, una que lleva a cada componente, para realizar las pruebas puedes escribir los paths en la barra de direcciones de tu navegador: http://localhost:4200/pagina02.

Esto debería mostrarte el típico mensaje Pagina02 works. Si no lo vez puede que este hasta abajo de todo de la página que crear por defecto Angular, si quieres velo mejor ve al archivo app.components.html y borra todo el contenido. Solo deja el <router-outlet></router-outlet>.

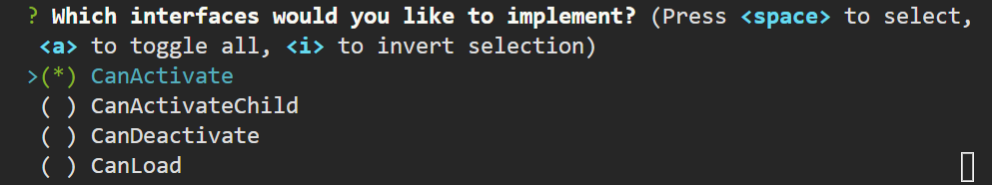
Se debería apreciar una pantalla completamente blanca en tu navegador solo con las palabras Pagina02 works. Es decir, lo que haya en el componente pagina02 que creamos antes. Si pones el resto de direcciones podrías ver los componentes también. Excelente. Ya tenemos nuestras rutas. Ahora creemos nuestro Guard.

Creacion del Guard

Para crear nuestro Guard podemos hacerlo de manera manual. Para hacerlo usamos el siguiente comando: ng g guard guards/horaAcceso

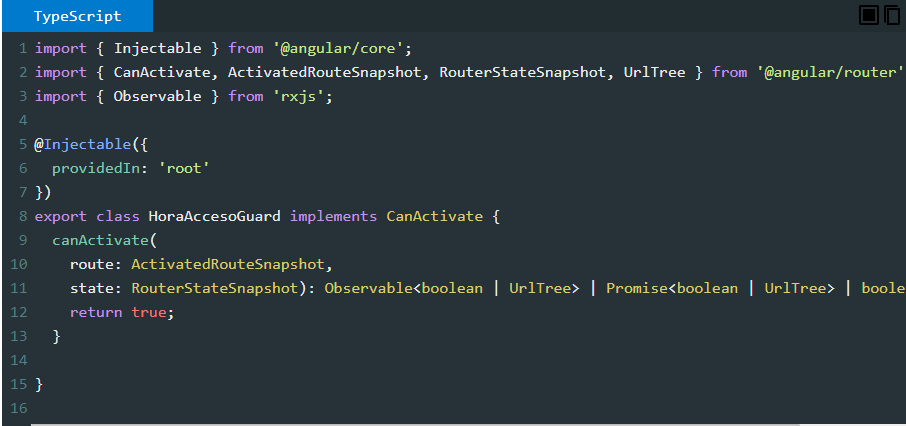
Si ya has trabajado con comandos ng verás que es muy intuitivo. g es de generar, guard porque queremos generar un Guard, guard/ es el directorio donde queremos guardar nuestro archivo y horaAcceso es el nombre del Guard

Cuando ejecutas el comando te aparece algo así como un menú donde tienes que seleccionar el tipo de Guard que quieres crear. Por defecto viene seleccionada CanActivate, pero si quieres cambiarlo puede usar las flechas de navegación y seleccionar el tipo de Guard usando la barra espaciadora del teclado.



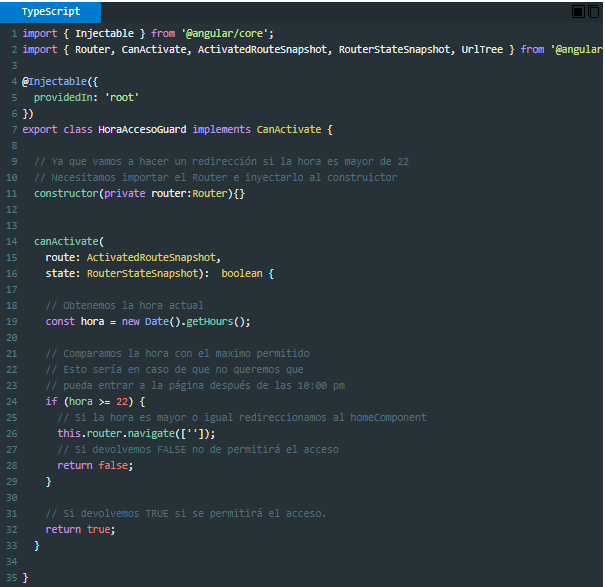
Para crearlo solo debes dar Enter y listo, **Angular CLI** se encargará de crear el directorio y el archivo. También hará todas las importaciones necesarias para que el **Guard**funcione. Nosotros solo tendremos que modificarlo para que funcione como nosotros queremos.

El archivo generado tiene que estar en la carpeta guards/ tenemos que acceder al archivo con el nombre hora-acceso.guard.ts, que es nuestro **Guard**, el otro archivo que termina con .spec.ts no lo tocaremos por el momento. Actualmente quedémonos en el .guard.ts



Nosotros tenemos que editarlo para que cumpla nuestra necesidad. El ejemplo que se me ocurre es verificar la hora y que no permita el acceso a la página si ya es más de las 10:00 pm. Claro que se pueden hacer todo tipo de **validaciones**, **verificar que existe un token**,**verificar el rol del usuario activo**. Verficando un Service el límite es tu imaginación.

Pero, para este ejemplo que tiene por objetivo demostrarte como funcionan los **Guard**, creo que meterle más complejidad sería innecesario. Solo tienes que saber como funciona, luego puedes hacer todo tipo de validaciones. El archivo debe quedar así:



Lo más importante sucede en la línea número 24 y en la línea 28. En la línea 24 estamos verificando si la condición para **NO cargar la página** se cumple. Y en la línea 28 devolvemos un false en el caso de que suceda.

El router es necesario para hacer una redirección, de lo contrarío veríamos una página blanca, sin con componente pero vacia, no tendría mucho sentido para el usuario. Así que usamos el router para redirigir al usuario al Home. Otra opción sería crear un componente de AccesoDenegado y redirigir ahí cuando la condición se cumpla.

Si la condición no se cumple entonces devolvemos un true, eso le dice a Angular que puede cargar la ruta sin problemas. Bueno. Ya tenemos nuestro Guard configurado, ahora tenemos que aplicarlo a la ruta que necesitamos verificar. Así que agregamos lo siguiente al archivo de rutas. Específicamente a la ruta que queremos afectar:



Lo nuevo es el valor canActivate: [ HoraAccesoGuard ] este argumento aplica los **Guard**. Como puedes ver canActivate recibe un array de Guard, lo que significa que perfectamente puedes enviar más de uno, seperándolos por comas ( , ) como cualquier **Array**.

Ya con eso estamos aplicando el Guard a la ruta, puedes comprobarlo entrando a la ruta http://localhost:4200/pagina02/ claro que para que la condición para mostrar la ruta es que sea antes de las 10:00 pm. Así que cuando lo pruebes puedes cambiar el valor del if (hora >= 22 ) { en el Guard. Si pon una hora que ya haya pasado, por ejemplo, si lo estás haciendo a las 9:00 am, puedes poner if (hora >= 8) { con eso, Angular no te dejaría entrar a la página y te redireccionaría al Home.