UNIVERSIDAD CAECE

Proyecto Final de Arquitectura Web

Alumnos

Romero Arregin, Guillermo Daniel

Corvalán, Gabriel Enrique

Materia

Arquitectura Web

Docentes

Mendez-Garabetti, Miguel

Piray, Eduardo Enrique

Indice

Aplicación en Angular

Siguiendo la consigna, se decidió implementar una aplicación Angular basada en el funcionamiento de las aplicaciones que permiten sacar turnos médicos. Estas funcionalidades son típicamente provistas por aplicaciones pertenecientes a seguros médicos.
Este proyecto se comenzó desde cero con el comando “ng new”, pero utilizamos la hoja de estilos global del “getting started” para contar con un archivo CSS global como punto de partida, aunque agregamos modificaciones a la misma, y algunos componentes implementan modificaciones en sus propios CSS.
Cada componente y servicio se creó desde consola con los comandos “ng g c” y “ng g s”, aunque se eliminaron los archivos “.spec” generados ya que no se utilizaron. Para poder desplegar el proyecto se necesita instalar el módulo de Auth0 declarado en “package.json” con “npm install”.
Dentro del módulo principal se realizaron las declaraciones correspondientes a todos los componentes y rutas de RouterLink necesarios. Además se declararon los módulos necesarios, incluyendo el servicio de Auth0 con el ClientID y Tenant, y dos servicios propios que utilizamos en esta aplicación, TurnosService y userLoginService.

Este proyecto se desplegó a través de Netlify en la siguiente dirección:

- https://romerocorvalanarqweb.netlify.app/

El código de este proyecto se encuentra almacenado en Github:

- https://github.com/gromeroarregin/arqweb-final

Componentes y Servicios

userLoginService

Este servicio se encarga de las funciones de login, logout y getters necesarios para obtener la identidad del usuario al realizar el inicio de sesión simulado por ReactiveForms. Como atributos incluye los datos del usuario y un flag para indicar si la sesión se encuentra activa o no.

UserLogin

Es el componente que utiliza el servicio de login para iniciar sesión. Implementa un ReactiveForm y un botón para enviar la información que, en caso de ser exitoso, envía los datos ingresados por consola y regresa a la página principal luego de limpiar el formulario. También verifica que los campos ingresados no estén vacíos.

Top Bar

Muestra la barra superior que se ve de manera constante en la aplicación. Tiene un enlace para ir a la página principal, y a la derecha dos botones, uno para realizar el login simulado por ReactiveForm y uno para realizar el login por Auth0.

Bottom Bar

Muestra la barra inferior que se ve de manera constante en la aplicación.
Permite acceder a tres componentes, “Sacar Turno”, “Perfil” y “Contacto”. Verifica que la sesión esté iniciada (por userLogin o Auth0) para activar los botones correspondientes.

Welcome

Este es el componente que actúa como página principal. Si no hay un login realizado, invita al usuario a iniciar, en caso de estar iniciado se muestra el nombre de usuario correspondiente.

Profile y Contact

El perfil permite al usuario modificar sus datos de perfil (excepto sus credenciales, cuyos campos están desactivados) e imprimirlos por consola. En el contacto se encuentra información útil para el usuario.

TurnosService

Es el servicio que realiza las consultas al JSON para obtener información. El archivo JSON se encuentra en la carpeta “assets” del proyecto y contiene un árbol donde se detallan especializaciones médicas, ubicaciones donde se atienden esas especializaciones y los médicos que atienden allí.
El servicio implementa una interfaz para cada objeto dentro del JSON, e implementa las funciones necesarias para obtener todas las especializaciones, así también como lugares y médicos para especializaciones y ubicaciones específicas ingresadas como parámetro.

Sacar turno

Utiliza el servicio de turnos para mostrar las distintas opciones disponibles para los usuarios. Muestra cada especialización, ubicación y médico correspondiente en formato de acordeón, que se expande (o retrae según corresponda) cuando el usuario hace clic sobre uno de ellos y utiliza los Id de los elementos seleccionados como parámetro para las funciones del servicio. También se sirve de las interfaces definidas en el servicio para declarar los tipos de las variables y recibirlos correctamente, además implementa los botones del componente de reserva en su propio HTML.

Reservar

Este componente sirve para mostrar un botón de reserva para cada médico. Dependiendo de la disponibilidad del médico (indicado en el JSON y recibido como Input), se activa o desactiva el botón. Al presionar el botón se emite un evento como Output.

Conclusiones del Proyecto

Durante el desarrollo de este proyecto tuvimos la oportunidad de aprender a desplegar una aplicación SPA de Angular y entender cómo interactúan los distintos componentes y servicios entre sí, además de entender los archivos que forman parte de cada componente, y cuando utilizar cada uno de ellos, además de cómo aumentar la seguridad de nuestra aplicación, especialmente sin contar con un backend propio donde gestionar las credenciales de los usuarios.
Adicionalmente a los requerimientos de la consigna, interactuamos con algunos elementos adicionales:

- En varias situaciones donde se requería una estructura If - Else en el HTML, nos encontramos con que la directiva ngIf no nos era suficiente, y podíamos optar por implementar una estructura con **ng-container + ng-template** para solucionarlo.
- Debido a la estructura del JSON, aprendimos a obtener objetos anidados dentro de otros, utilizando las funciones **pipe() y map()** para obtener los datos que necesitamos.
- Al interactuar con el JSON, y principalmente con Auth0Service, fue necesario utilizar las funciones **subscribe()** para los **Observables**. Se hizo evidente la necesidad de manejar de manera manual la función unsubscribe() o asegurarse que las funciones sean Async, ya que puede llevar a errores y congelamientos en las páginas de no hacerse correctamente debido al monitoreo constante y uso excesivo de los recursos.
- Para este proyecto específico, debido a que múltiples componentes interactúan con los dos servicios de login quizás se hubiese podido utilizar un servicio para unificar y simplificar ambos, especialmente las funciones getters, donde antes se suele preguntar cuál es el servicio activo para saber a cuál servicio consultar, y en el caso de Auth0Service se debe trabajar con los observables, agregando un poco más de complejidad.
- Como mejora sería muy útil y atractivo para los usuarios integrar un mapa de Google que muestre los lugares donde se encuentren las sucursales de la empresa y los lugares donde ofrecen cobertura, de manera que un usuario pueda seleccionarlo allí de manera más sencilla.

Seguridad

Análisis de la seguridad

Al completar el inicio de sesión simulado a través de ReactiveForms, se hace evidente la necesidad de contar con un backend capaz de gestionar la seguridad de la aplicación, ya sea para procesar las credenciales ingresadas o almacenarlas. La ausencia del backend provoca que todas las credenciales y procesos de seguridad se encuentren expuestos en el frontend, y por lo tanto son vulnerables al analizar el código fuente de la aplicación. La matriz de Mitre identifica este problema como “Credentials From Password Stores”.

Implementación de un Sistema IAM

La implementación de un backend para almacenar y procesar credenciales requiere de mucho esfuerzo, tiempo y conocimiento apropiado para realizarse correctamente y con la menor cantidad de vectores de ataque posibles. Una solución a este problema es delegar el aspecto de seguridad a un sistema IAM, dejando que se encargue de la gestión de credenciales y los procesos de autenticación y autorización. De esta manera, la aplicación de Angular no tendrá que almacenar ni procesar datos sensibles, y por lo tanto no estarán expuestos en la aplicación. Para nuestra aplicación elegimos utilizar el sistema IAM Auth0.

Auth0

Al integrar Auth0 en la aplicación, se puede notar que al momento de gestionar la autenticación, la aplicación de Angular redirecciona a una dirección dentro del dominio de Auth0, y por lo tanto las credenciales ingresadas y los procesos de autenticación ocurren dentro del backend provisto por Auth0. Sin embargo, esto también agrega la necesidad de comunicarse con el backend, y por lo tanto agrega posibles vectores de ataque adicionales que debemos tener en cuenta.
Para integrar Auth0 en la aplicación, sólo se debe instalar el módulo de Auth0 a través de NPM, importarlo en el archivo app.module.ts como tal e iniciar la instancia del módulo que se utilizará en los componentes. Para inicializar la instancia se debe proveer un clientID (la ID de la app en Auth0) y el dominio (el “Tenant” que crea la app en Auth0) a utilizar por el módulo. Estos datos se pueden crear y obtener dentro de la página de gestión de Auth0 en su sitio web.

Vectores de Ataque Identificados

Para la identificación de vectores de ataque nos enfocamos en la sección de “Credential Access” en la matriz de Mitre.

- **Credentials From Password Stores, Brute Force, Exploitation for Credential Access, Modify Authentication Process, OS Credential Dumping, Unsecured Credentials:** La autenticación se gestiona dentro del backend de Auth0, y por lo tanto las credenciales están protegidas dentro su dominio (O de los identity providers utilizados, como Google).
- **Adversary in the Middle, Network Sniffing:** Este vector se agrega el requerir una comunicación con el backend de Auth0. Auth0 implementa el protocolo OAuth2 y OpenID, y al integrarse en una aplicación Angular sin backend propio (app pública), resulta imposible almacenar el “client secret” para encriptar las comunicaciones necesarias. Para solucionar esto, se utiliza el flow “Authorization Code Flow With Proof Key for Code Exchange” que permite encriptar las comunicaciones y asegurar que cualquier información que provenga de fuentes desconocidas no cuente con la verificación necesaria para ser aceptada. Adicionalmente, el uso de JWT evita cualquier intento de manipulación de paquetes ya que al ser modificados las verificaciones no serán correctas.
- **Forge Web Credentials, Steal Application Access Token, Steal Web Session Cookie:** El uso de JWT combinado con la correcta declaración de Origin URIs permiten encriptar la información enviada, negar el acceso desde fuentes no autorizadas, y descartar los intentos de utilizar credenciales modificadas.

Posibles Vulnerabilidades

Estos vectores de ataque explotan vulnerabilidades externas a la interacción entre la app y Auth0, y por lo tanto pueden ser viables para obtener información.

- **Input Capture**
- **Multi-Factor Authentication Interception**
- **Multi-Factor Authentication Request Generation**