Challenge Frontend - Angular 🚀

Situación Inicial 📍

“A la carta” es un restaurante que ofrece platos de gran calidad y que satisface a sus clientes hace más de 40 años brindando un menú variado que alcanza los gustos de los diferentes consumidores. Dicho restaurante, se acerca a tí para solicitarle el desarrollo de una aplicación que aloje su menú virtual.

Objetivo 🎯

Desarrollar una aplicación para crear una carta de opciones de menús para el restaurante “A la carta” que consumirá una API externa y mostrará diferentes atributos a nivel individual de cada plato y del menú finalizado.

👉 Consumir los endpoints de la siguiente API para realizar las distintas operaciones. Deberás autenticarte en la plataforma para obtener una ApiKey y poder realizar las peticiones.

⚠️ ¡No es indispensable hacer todo!
Mientras más completes, mayor puntaje obtendrás, pero puedes enviar la app hasta el estadío que tengas en base a tu conocimiento actual. Recuerda que el objetivo del challenge es entender tu nivel de conocimiento actual. Recomendamos realizar los **requerimientos técnicos** en el orden en el que están dados y tener en cuenta los **criterios a evaluar** al final de este documento.

Requerimientos funcionales ⚙️

En la pantalla de Home se deberá mostrar, además de los platos que conforman el menú:

- Acumulativo de precio del menú.
- Promedio de tiempo de preparación entre todos los platos.
- Promedio de Healt Score entre todos los platos.
- El menú debe tener 4 platos. Debe haber 2 veganos y 2 que no lo sean. Esto debe validarse al intentar agregar un nuevo plato.
- Se deberá poder eliminar un plato del menú, lo que generará nuevamente los promedios y acumulativos (utilizando Servicios deberán estar almacenados en un estado de este)

Requerimientos técnicos 🔧

Aprovechando las características de Angular, deberán crearse las siguientes secciones, y modularizar las mismas en componentes reutilizables.

Además, para el manejo de peticiones HTTP deberá utilizarse HttpClient. Y el sitio deberá ser

responsive, y utilizar Bootstrap o cualquier otro framework como punto de partida para aprovechar las características.

1. Formulario de Login

El formulario se deberá renderizar al ingresar a cualquier ruta si el usuario no está autenticado, conteniendo los campos:

- Email.
- Password.
- Botón de “Enviar”.

Al hacer click en “Enviar”, se deberá validar con ReactiveFormsModule que ambos campos no estén vacíos, y mostrar un mensaje al usuario si lo estuviesen. Caso contrario, se deberá realizar una petición POST a la siguiente url, con los campos email y password en el BODY.

Los datos válidos para obtener un token son:

- Email: challenge@alkemy.org
- Password: react

Se debe mostrar algún tipo de feedback al usuario mientras se está procesando la petición, no permitiendo que vuelva a accionar el botón de login hasta obtener una respuesta.

En el caso de obtener un error de la API, se deberá mostrar una alerta (utilizando sweet alert), mientras que si es satisfactorio deberá redirigir al Home y almacenar el token obtenido en localStorage.

2. Platos

El Home de la aplicación mostrará los platos del menú en un listado. Cada ítem (el cuál debe ser un componente separado) del listado contendrá:

- Nombre del plato.
- Imagen.
- Características del plato.
- Acciones para ver el detalle o eliminarlo del menú.

3. Buscador de Platos

Para agregar un plato al menú, se deberá visualizar un formulario que realice una petición GET al endpoint de búsqueda y muestre los resultados disponibles en un grid, utilizando el componente de ítem del punto anterior.

El formulario deberá buscar únicamente si hay más de 2 caracteres en el filtro utilizando debounce, caso contrario no debe mostrar nada.

4. Detalle del Plato

Al hacer click en un plato del menú, se mostrarán los detalles de los campos acumulados y promediados en el menú.

5. Navegación entre secciones

Las diferentes secciones que tendrá la app deberán protegerse verificando que el usuario autenticado disponga de un token que se almacenará en localStorage. El mismo, se obtendrá de una API con datos de muestra. Si un usuario intenta ingresar a cualquier ruta sin estar autenticado, deberá ser redirigido al login y en caso de cerrar la sesion, el token deberá borrarse. Para el manejo de rutas se deberá utilizar RouterModule y Guards.

Criterios a evaluar

- Almacenamiento y consulta del token en local storage
- Peticiones a los endpoints de autenticación de la API
- Utilizar servicios para el manejo de estado de la aplicación.
- Actualizar estado de la aplicación si el usuario está autenticado
- Generar un mensaje para informar al usuario mientras hace una operación.
- Crear componentes de formularios con campos tipo texto y numérico que persistan el input del usuario en el estado. Por ejemplo: nombre de usuario, password, datos personales.
- Validar el contenido de los campos de formularios.
- Desarrollar componentes que puedan mostrar datos recibidos por propiedades (Input/Output).
- Renderizar una lista recibida por input realizando una iteración sobre la misma.
- Desarrollar la navegación con las validaciones correspondientes.
- Renderizar el contenido de forma dinámica según la ruta actual de la aplicación.
- Utilizar un framework de frontend para estandarizar los estilos de los elementos visuales en la aplicación (por ejemplo Bootstrap, Bulma, Tailwind, etc).
- Utilizando HttpClient, realizar peticiones HTTP desde los componentes. Por ejemplo, GET y POST.
- Manejar las excepciones en el caso de que las peticiones no puedan realizarse correctamente.
- Agregar elementos visuales (como alerta, editor de texto enriquecido) utilizando un SDK específico en base a los requerimientos.

¡Bonus track!

Si terminaste los puntos anteriores y queres destacarte optimizando la resolución tu challenge, te invitamos a realizar (esto es opcional) los siguientes requerimientos:

- **Test,** en caso de conocer el procedimiento se pueden agregar tests unitarios para validar los elementos de la app:

    - Verificación de usuario autenticado al ingresar a una ruta.
    - Validación de campos en submit de formulario de login o búsqueda.

- Manejo de excepciones al obtener errores de la API.

Para la implementación de los tests deberán utilizarse Jasmine con Karma

- **Validación de formularios** por medio de ReactiveFormsModule, con entidades relacionadas e imágenes según el caso. Maneja el contenido de los formularios de forma dinámica. Por ejemplo, listas desplegables anidadas a otras categorías. Permite la reutilización de los componentes de formularios.

- **Componentes de contenido:** Crea componentes reutilizables que permitan mostrar contenido de forma dinámica con campos básicos y complejos. Estandarizar estilos y tipologías de los componentes de la aplicación.

- **Componentes de listados:** Crea componentes reutilizables que permitan renderizar listados de recursos con campos simples y campos complejos.

- **Componentes de navegación:** Partiendo del ruteo base, desarrollar elementos visuales para la transición entre rutas y optimizar la navegación utilizando lazy loading. En caso que no exista o no se encuentre la ruta, mostrar mensaje apropiado.

- **Variables de entorno:** Utiliza las prácticas recomendadas para centralizar el uso de datos comunes de la aplicación y almacenarlos en variables de entorno. Por ejemplo, las URLs de los endpoints, entorno actual, versión del proyecto.