Día 5

Conceptos

- Alertas
- Formularios *template-driven*
- Formularios reactivos
- Servicios REST
- Introducción a los Observables

Explicación de uso de alertas

- Crear en la página Home un botón que cuando se pulse emergerá una alerta con un título, subtítulo, contenido y dos botones: aceptar y cancelar.
- La documentación se puede consultar en la página de ion-alert.
- Explicar cuál es la alternativa a async/await de ES7 usando promesas de ES6 (a las que probablemente estemos más familiarizados por su popularización con jQuery).

Formularios *template-driven*

- Se trabajará sobre la página *template-driven-form*.
- Crear un campo de texto.
- Ilustrar el uso de la propiedad value y el evento input.
- Mostrar el contenido del campo de texto en un h1 que se ubique abajo.
- Cambiar la solución usando *ngModel*.

Formulario reactivo

- Se trabajará sobre la página *reactive-form.*
- Preparar el formulario reactivo, sin validación, e inicializando los valores de los campos a vacío.
- Añadir a algunos de los campos validación.
- Gestionar el evento de submit del formulario, imprimiendo el objeto del formulario. Examinar el contenido de ese objeto (valores, controles, errores).

- Que el botón de submit se habilite sólo si el formulario es válido.
- Cuando se pulse el botón de submit, que se muestre en la sección *Summary* los datos del usuario que se introdujeron.
- Crear un validador *isNotTemporalEmailValidator* que validará si el campo email no contiene un email de un servicio de correos temporales.
- Crear un servicio de usuario con un método *getUser* que devuelva un usuario. Usar el método *patchValue* del formulario para actualizar el formulario con los datos del usuario.
- Arrancar el servidor de mocks, con un retraso en las respuestas de 2 segundos.
- Crear en el servicio de usuario un método *getRemoteUser* que obtenga el usuario del servidor de mocks.
- Obtener los datos en el controlador de la vista (introducción a los Observables). Ilustrar la gestión de errores.

Ejercicio 1

Desarrolle un componente *app-loading-feedback* que, a partir del parámetro de entrada *state*, que es una cadena que puede tener los valores “loading”, “loaded” o “error”, mostrará: el spinner de carga si el estado es “loading”; el contenido que se encierra entre las etiquetas si el estado es “loaded”; y un mensaje de error con un botón de reintentar si el estado es “error”. Cuando se pulse sobre el botón de reintentar, el componente emitirá un evento. Un ejemplo de uso del componente es:

```
<app-loading-feedback [state]="state" (retryPressed)="loadUser()">
  <form [formGroup]="form" (submit)="submitForm()">
    ...
  </form>
</app-loading-feedback>
```

Refactorice el código de la explicación para hacer uso de este componente. Este mismo componente podría reutilizarse para otras páginas (incluso en páginas de otros proyectos) que requieran la misma funcionalidad.

Ejercicio 2

Dote al formulario de funcionalidad para que se actualicen los datos del usuario remoto con los que se indican en el formulario cuando el usuario lo envíe. Mientras el servidor

lo está procesando, el botón de enviar formulario debe permanecer deshabilitado. Una vez que finalice el envío, se deberá informar al usuario del resultado de la operación con un toast.

Ejercicio 3

Retome la aplicación de la lista de frutas del día anterior.

Modifique la página de detalle de fruta para que desde la misma se pueda editar la descripción de la fruta. Tenga en cuenta que, según la solución que dé, cuando vuelva al listado la descripción puede que no esté actualizada con la que introdujo en el detalle. De ser así, cargue la lista de frutas siempre que se acceda a la pantalla (incluso cuando se vuelve a ella).

Incluya también el componente *app-loading-feedback* en la página de la lista de frutas y la página de detalle de fruta.