Prueba Migración Angularjs a Angular 13

El objetivo de esta prueba es verificar el conocimiento de los candidatos en los dos frameworks utilizados en Iberia, angularjs y angular 13 en un contexto de migración. Se deben de demostrar conocimientos en typescript aparte de en los frameworks mencionados, de la librería @angular/upgrade y de testing unitario.

Los ejercicios siguen una escalada de dificultad progresiva con los diferentes casos que se pueden dar en una aplicación híbrida. El proyecto es una pequeña aplicación con Bootstrap donde la importancia la tiene el conocimiento en javascript y no la maquetación del mismo, se trata de montar una aplicación híbrida usando la pokeapi.

Al levantar la aplicación híbrida se observará el siguiente diseño, se trata de darle la funcionalidad. Se ha de seguir el estándar para nombrar los ficheros angular.

1. Creación servicio Angular: Crear un servicio Angular 13 que llame al siguiente endpoint de la pokeapi, obteniendo los primeros 20 pokemon, el servicio debe de inyectarse en la main-page en angularjs.

   https://pokeapi.co/api/v2/pokemon/?limit=20

   Los datos a mostrar en las cards será únicamente el nombre, el resto de datos se rellenará a posteriori.

2. Creación componente Angular: Pasar el componente card marcado en un comentario del html a angular 13, downgradear el componente a angularjs y pasar el dato del nombre mediante binding para mostrarlo.

3. Creación componente angularjs upgrade: Crear un pequeño componente en angularjs (en typescript) con el nombre del pokemon con el tag h5 que será consumido por el componente angular 13 de card anterior, pasando el nombre por binding.

4. RXJS: Modificar el servicio creado en el paso 1 para con la llamada a la api anterior hacer una petición a las url que venían en el resultado a la vez combinando el resultado de las llamadas. Esa información se utilizará para la imagen de la card cogiendo el campo sprites -> front_default y el peso de la card con el campo weight.
   Solamente se van a usar esos campos por lo que el servicio podría transformar los datos para solamente devolver los que interesan.

   Añadir una lógica adicional para reintentar las llamadas una segunda vez si el servicio falla la primera.

5. Formulario en angularjs (Opcional): Realizar un nuevo componente angularjs (en typescript) con el formulario de arriba con los dos inputs y el botón buscar. El primer input de texto es obligatorio, si el formulario es válido al presionar el botón buscar se debe de llamar al siguiente endpoint:
   https://pokeapi.co/api/v2/pokemon/{nombre o id}

Para ello crear un nuevo servicio en angularjs que utilice el $http propio de angularjs. El resultado imprimirlo en consola.

6. Inputs Angular en Angularjs Forms (Opcional): Demostrar los conocimientos en migración haciendo que el select del formulario sea un componente angular, pero a su vez esté ligado al formulario de angularjs gracias al ng-model de angularjs.
   Al pulsar el botón buscar si el input seleccionado está vacío se pondrá el nombre o el id del input 1, en caso de estar relleno será usado para llamar al servicio del ejercicio anterior.
7. Testing unitario (Opcional): Se pueden añadir test unitarios con karma tanto para angular o angularjs para asegurar la funcionalidad. Libremente.

   NOTA IMPORTANTE: Los tests angular 13 deben de terminar en .spec.ts y los tests angulajs en -test.ts.