Fase 1: crear app e implementar enrutamiento

Enunciado.

Vamos a desarrollar una aplicación de Angular con **enrutamiento** (*routing*) que se llamará **Despotify**. La aplicación permitirá consultar información sobre canciones de una base de datos. También permitirá que un usuario elabore una lista de reproducción.

Hoja de estilos

De manera general, la aplicación usará ***Bootstrap***, por lo que es necesario enlazar los archivos requeridos. Los estilos que sean exclusivos de un componente estarán en el CSS de dicho componente.

Origen de datos

Los datos van a ser obtenidos de las siguientes fuentes por orden de prioridad, es decir, se intentará la fuente (1) y si hay algún obstáculo insalvable pasaremos a la fuente (2) y así sucesivamente:

1. La ***API*** de ***My JSON Server*** con la siguiente URL base: https://my-json-server.typicode.com/luismiguel-fernandez/examen
2. Si fallara el acceso a este último servicio entonces el profesor os proporcionaría un archivo ***JSON*** para integrar en nuestra app como recurso local.

Las URL para consultar datos son:

URL	URL base + "`/songs`"
Parámetros	JSON con listado de canciones
URL	URL base + "`/playlists`"
Parámetros	JSON con listado de *playlists*

Modelos de *templates* entregados por el profesor

Para ahorrar tiempo, se entregan algunas *templates* con código HTML genérico que ayudan mucho, pero deben ser adaptadas previamente para funcionar bien.

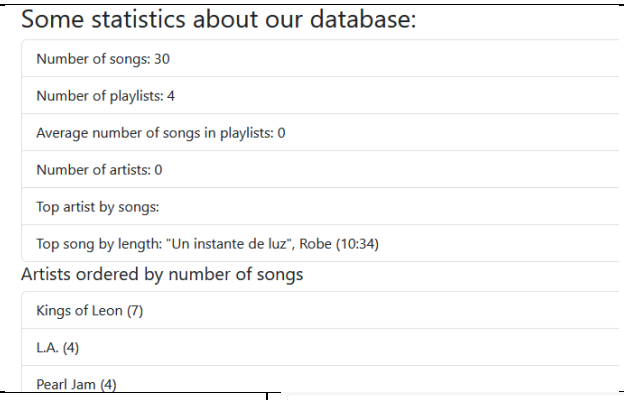
Componente raíz

A continuación, se indica una lista de los 3 componentes que será necesario crear para dar contenido al componente raíz:

- **Componente para el encabezado**. Muestra la imagen logotipo de la aplicación y el nombre de la aplicación.
- **Componente para la barra de navegación**. Ofrecerá la posibilidad de clicar en las secciones listadas en el apartado ***Navegación y enrutamiento*** más abajo.
- **Componente para el pie**. Muestra el nombre del autor o autora de la aplicación, es decir, tu nombre. También cita la fuente de los datos.

Navegación y enrutamiento

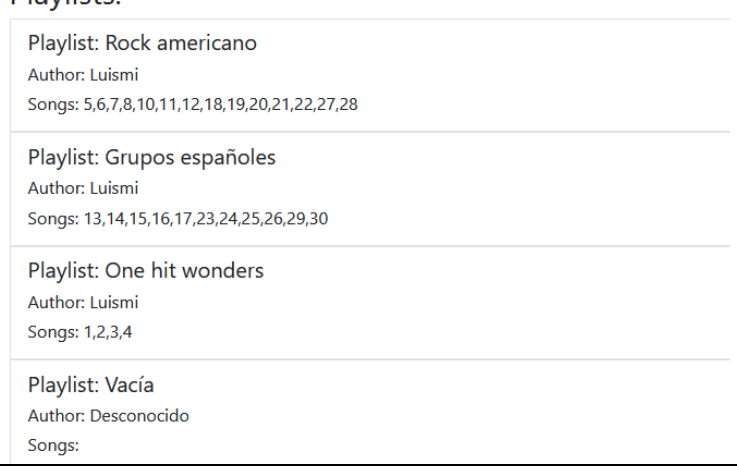
Justo debajo del encabezado aparecerá una barra de navegación horizontal con varias secciones:



Sección *Home*: muestra algunas estadísticas sobre la base de datos. Además de unas cifras, también muestra un listado ordenado de artistas de mayor a menor número de canciones.

Sección *Songs*: muestra un campo de texto para buscar por nombre de artista o título de canción. Listará resultados al presionar la tecla Intro.

La vista se puede cambiar con el *switch*. Esta opción debe guardarse en *Localstorage*.



Sección *Playlists*: muestra las listas de reproducción. La última lista está vacía para poder añadir canciones en esta lista.

Fase 2: desarrollar servicios y filtros

Servicios

Tienes libertad para implementar servicios a tu criterio. Sin embargo, si te sirve de guía, aquí tienes una sugerencia:

- **Un servicio para obtener datos del origen de datos**. Es capaz de hacer llamadas *HTTP* para obtener los datos en formato *JSON* necesarios para alimentar a la aplicación.

Las canciones añadidas por el usuario a la última lista de reproducción no deben perderse, por lo que hay que gestionar la **persistencia** en el almacenamiento local del navegador (***local storage***).

Filtros

Implementar los siguientes filtros:

- Filtro para que se muestren únicamente las canciones cuyo título o nombre de artista incluye una cadena patrón introducida por el usuario.
- Filtro para convertir duraciones de canciones de segundos al formado **mm:ss** (donde *mm* son minutos y *ss* son segundos).

Objetivos

Objetivos obligatorios	**OK**
Los contenidos del componente raíz se ajustan a lo pedido. [0,25p]	
La barra de navegación funciona. [0,75p]	
Un servicio obtiene datos del servidor mediante HTTP y dichos datos son usados y mostrados correctamente en alguna parte de la aplicación. [1p]	
El usuario puede añadir canciones a la última lista de reproducción (inicialmente es una lista vacía y no puede contener canciones repetidas). [0,5p]	
Se muestran las estadísticas básicas (datos individuales) en la página de inicio de la aplicación. [1p]	
Los filtros han sido implementados: El buscador muestra resultados correctos al buscar por patrón sin importar mayúsculas o minúsculas. [0,75p] La duración se muestra en formato mm:ss. [0,75p]	
Objetivos opcionales	**OK**
No se realizan llamadas HTTP innecesarias y el aspecto está cuidado y los errores en la consola controlados. [0,25p]	
La *playlist* del usuario y el formato de listado se conserva en *LocalStorage*. [0,75p]	
Se puede cambiar la lista de canciones entre formato tarjetas y formato filas. [1,5p]	
Se muestran las estadísticas avanzadas (artista con más canciones y listado de artistas ordenado por n.º de canciones) en la página de inicio de la aplicación. [1p]	
Se puede eliminar canciones de las *playlists*. [1,5p]	

Conocimientos examinados

En este examen se **requiere** que el alumno o la alumna sepa:

- Crear aplicaciones web con Angular
- Insertar nuevos componentes en la aplicación
- Capturar eventos producidos por el usuario sobre ciertos elementos
- Definir enrutamiento para navegar dentro de la aplicación
- Establecer vías de comunicación entre componentes
- Añadir servicios e inyectarlos en los componentes que los requieran
- Realizar llamadas HTTP asíncronas para interactuar con una base de datos

En este examen se **valora** que el alumno o la alumna sepa:

- Detectar y manejar los posibles errores en tiempo de ejecución
- Ahorrar ancho de banda
- Implementar aspectos avanzados en aplicaciones web para aumentar la funcionalidad de la misma

Calificación del examen

La correcta resolución de todos los objetivos obligatorios otorga una calificación de 5 sobre 10. Solamente si todos los objetivos obligatorios han sido resueltos adecuadamente se procederá a sumar puntuación adicional por cada uno de los objetivos opcionales solucionados con éxito, hasta un máximo de 5 puntos adicionales si se implementan 5 de los objetivos opcionales.