Ejercicio 1

Enunciado.

Vamos a desarrollar una aplicación de Angular que se llame **JPcomponentes**. Intenta crear la aplicación base desde una consola que pregunte si quieres **enrutamiento**, para decirle que **sí** y ahorrar tiempo.

Origen de datos de artículos hardware

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):

1. Una *API REST* de "*My JSON Server*" en my-json-server.typicode.com y que se alimentará de un fichero de datos alojado en un repositorio de *GitHub* del profesor.
2. Si fallara el acceso a este último servicio entonces el profesor os proporcionará los datos para integrar directamente (*hard-coded*) en el código fuente.

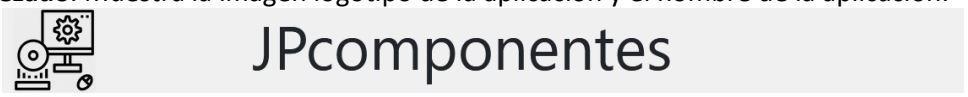
Hoja de estilos

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

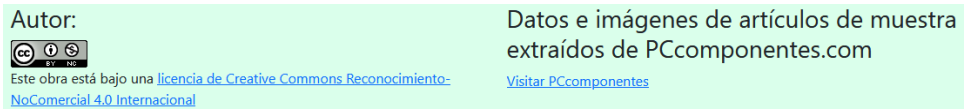
Componente raíz

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

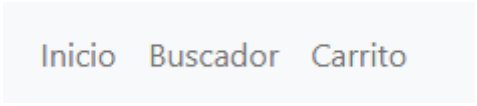
- **Encabezado**. Muestra la imagen logotipo de la aplicación y el nombre de la aplicación.



- **MenuNavegacion**. Ofrecerá la posibilidad de clicar en las secciones listadas en el apartado ***Navegación y enrutamiento*** más abajo.
- **Pie**. Muestra el nombre del autor o autora de la aplicación y cita la fuente de los datos.



Navegación y enrutamiento



Justo debajo del encabezado aparecerá una barra de navegación horizontal con los siguientes elementos:

- La sección **Inicio** mostrará el componente **Inicio**. Muestra los 3 artículos más recientes del catálogo.
- La sección **Buscador** mostrará el componente **Buscador**. El usuario podrá buscar artículos de las siguientes formas:
  - Escribir en un campo de texto para buscar por nombre de artículo
  - Seleccionando una categoría de la lista de categorías ofrecidas por el buscador
  - Seleccionando un fabricante de la lista de fabricantes ofrecidos por el buscador

Para mostrar en formato cuadrícula los resultados de las búsquedas, nos apoyaremos en el siguiente componente:

  - **Articulo**. Muestra en formato tarjeta información básica de un artículo (foto, nombre y precio) y un botón para añadirlo al carrito del usuario sólo en caso de que exista stock de dicho artículo. Si no hay stock, el botón no será clicable y su aspecto indicará claramente que no se puede añadir al carrito.

- La sección **Carrito** mostrará el componente **Carrito**. Muestra un listado en formato lista con los artículos que el usuario ha ido añadiendo a su carrito. Para conseguir este listado nos apoyaremos en el siguiente componente:
  - **ArticuloCarrito**. Muestra un artículo del carrito con los siguientes elementos:
    - su foto y su nombre
    - su precio unitario
    - cuántas unidades ha añadido al carrito
    - el subtotal que supone
    - la posibilidad de aumentar o reducir las unidades
    - la posibilidad de eliminar completamente dicho artículo

Si a la URL base de nuestra aplicación se le añade alguno de los siguientes sufijos, se mostrará lo que se indica a continuación:

- "`/home`": lleva a la sección **Inicio**.
- "`/search`": lleva a la sección **Buscador**.
- "`/cart`": lleva a la sección **Carrito**.
- "`/`": lleva a la sección **Inicio**.
- Cualquier otro sufijo: muestra el componente **Pagina404**.

Para implementar una página 404

- **Pagina404**. Este componente muestra un mensaje de error al usuario indicando que ha intentado acceder a una página que no existe.

Servicios

Tienes libertad para implementar servicios a tu criterio. Sin embargo, si te sirve de guía, para imitar exactamente la solución del profesor es necesario crear los siguientes servicios:

- **ConexionRemota**. Conoce la URL del servidor y es capaz de recuperar del servidor con llamadas HTTP las listas de categorías, fabricantes y artículos.
- **BaseDatosArticulos**. Es un servicio que centraliza la base de datos (categorías, fabricantes y artículos). Se inicializa en el constructor, haciendo uso del servicio **ConexionRemota**. Una vez que la base de datos está inicializada, es capaz de aportar al resto de componentes de la aplicación consultas sencillas, como:
  - getCategories: Devolver la lista completa de categorías
  - getManufacturers: Devolver la lista completa de fabricantes
  - getArticles: Devolver la lista completa de artículos

  Pero también consultas más elaboradas, como, por ejemplo:
  - getById: Localizar un artículo a partir de un ID.
  - searchByCategory: Obtener una lista de artículos de una categoría dada.
  - searchByManufacturer: Obtener una lista de artículos de un fabricante dado.
  - searchByName: Buscar una lista de artículos cuyo nombre incluya un patrón indicado.
- **Cart**. Es el lugar centralizado donde se almacena el carrito del usuario y el precio total de dicho carrito. Contiene métodos para todas las operaciones que puedan ser necesarias sobre el carrito:

- getCart: Obtener la lista de artículos que componen el carrito completo.
- getPrice: Obtener el precio total del carrito.
- addToCart: Añadir un artículo al carrito.
- removeFromCart: Eliminar un artículo del carrito.
- isAlreadyInCart: Comprobar si un artículo ya existe en el carrito.
- decrementUnits: Reduce el número de unidades de un artículo del carrito.
- incrementUnits: Aumenta el número de unidades de un artículo del carrito.
- recalculePrice: Es un método auxiliar que vuelve a calcular el precio total del carrito y debe ser llamado cada vez que se produzca cualquier tipo de cambio en la composición del carrito.

Modelos de *templates* entregados por el profesor

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

- Encabezado
- Pie
- Buscador y Artículo
- Carrito y ArtículoCarrito

Objetivos

Objetivos obligatorios	**OK**
Los contenidos del componente raíz se ajustan a lo pedido	
La barra de navegación funciona	
Un servicio obtiene datos del servidor mediante HTTP y dichos datos son usados correctamente en alguna parte de la aplicación	
El buscador muestra resultados correctos al buscar por nombre	
El buscador muestra resultados correctos al buscar por categoría o fabricante	
Los resultados de búsqueda pueden ser añadidos al carrito	
La sección Carrito lista los artículos añadidos por el usuario	
Los artículos del carrito pueden ser borrados individualmente	
El carrito muestra el precio total y se actualiza si se cambia el carrito	
Los precios son indicados con un filtro de unidades monetarias	
Objetivos opcionales	**OK**
La sección Inicio muestra los 3 artículos más recientes del catálogo	
En el carrito se tiene en cuenta cuántas unidades hay de cada artículo y el usuario puede modificarlas allí mismo	
Si se vuelve a insertar un mismo artículo sólo aparece una vez en el carrito, pero con un número mayor de unidades	
No se realizan llamadas HTTP innecesarias	
El aspecto está cuidado y los errores controlados	

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 un carrito de la compra flexible, con unidades de artículos

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 todos los objetivos opcionales son satisfechos.