JSBO1 - Proyecto final

- Fecha de entrega: 01/05/2019

Requisitos no funcionales

- ❑ Se valorará que el HTML sea lo más válido y semántico posible (en la medida que el framework lo permita) y que el CSS funcione bien en todos los navegadores y la mayoría de tamaños de dispositivos

- ❑ Se valorará la calidad y organización del código (ejemplo: Charla Clean Code)

- ❑ Declarar variables correctamente. Esto incluye:

    - Diferenciar entre let y const.
    - Buena nomenclatura.

- ❑ Usar los siguientes **tipos de variable**:

    - number
    - string
    - boolean
    - null
    - object
    - array

- ❑ **Conversión entre tipos**.
- ❑ **Operadores**.
- ❑ **Comparaciones**.
- ❑ **Condicionales**. Se requiere el uso de ternarios si es conveniente.
- ❑ **Operadores lógicos** (||, &&, !).
- ❑ **Bucles**.
- ❑ Usar **funciones***:

    - Se prestará atención a la nomenclatura.
    - Concepto de DRY.
    - Parámetros por defecto.
    - Separación de conceptos.
    - Arrow functions.

*El proyecto debe estar documentado. (Véase jsdoc):

- Requisito tener todas las funciones bien documentadas.

- ❏ Usar métodos de los primitivos:

    - Métodos de números.
    - Métodos de Strings.
    - Métodos de Arrays.

- ❏ Usar algún bucle de la **programación funcional** (.map, .filter, .reduce...).
- ❏ Usar **destructuring**.
- ❏ Usar **fechas**.
- ❏ Usar **JSON**.
- ❏ Usar el **operador spread**(...)
- ❏ Usar mínimo 1 **clase**.
- ❏ **Try/catch**.
- ❏ Errores customizados.
- ❏ Usar promesas/async/await.

Backend

- ❏ La aplicación a desarrollar tiene que ser una **API REST** (entradas y salidas en formato JSON).
- ❏ Al menos 3 **endpoints** diferentes (entre GET, POST, PUT, PATCH, DELETE)
- ❏ Cada módulo, tiene que tener su ruta propia para organizar el código.
- ❏ Utilizar al menos una base de datos (Ver especificaciones en punto “Infraestructura”)
- ❏ Validación de los datos de entrada.
- ❏ Sistema de **autenticación**.
- ❏ La configuración tiene que estar en un archivo externo configurable (configuración no hardcodeada en el código).

- Bonus point:

- ❏ Sistema de autorización (admin, guest…)

Frontend

- ❏ La aplicación Angular debe ser responsive
- ❏ Usar flexbox para crear layouts

La aplicación a desarrollar tiene que ser una **Single Page Application** (SPA)

- ❏ Deben incluirse varias **rutas** (Por lo menos 3).
- ❏ Cada ruta debe configurarse en un **NgModule** diferente.
- ❏ La aplicación debe constar de varios **NgModule**, normalmente uno por ruta además del módulo principal. Puede haber módulos que no se correspondan con ninguna ruta en concreto como por ejemplo módulos de autenticación o con artefactos compartidos (SharedModule).
- ❏ Deberíamos diferenciar componentes de acceso a datos y lógica de negocio de componentes presentacionales (**Smart vs Dumb components**).
- ❏ La aplicación debería incluir al menos un **formulario** (Reactive o Template-driven)
- ❏ Debemos gestionar la **validación** de los formularios.
- ❏ La aplicación debe comunicarse con un **API REST** y definir los servicios de acceso al mismo de forma clara y ordenada.
- ❏ Incluir una capa de **autenticación** en la aplicación (Login, Registro).

- Bonus point:

- ❏ Utilizar un **patrón de State Management** es un bonus aunque no es obligatorio.
- ❏ El uso de **Lazy Loading.**
- ❏ Incluir **validaciones personalizadas** en los formularios.

Infraestructura

Requisitos base de datos:

Mysql

- ❏ Todas las tablas deben tener **Primary Key** (nombradas PK_nombre_tabla).
- ❏ Las **Foreing Key** deben estar correctamente nombradas (FK_tabla_hijo_tabla_padre)
- ❏ Uso de **campo de auditoría** (mínimo fecha de creación y fecha de actualización)(uso de triggers)

MongoDB

- ❏ Al menos una colección que los documentos tengan una lista
- ❏ Proceso de búsqueda sobre los documentos (al menos un criterio)
- ❏ La parte avanzada realmente es complicar el modelo

- Bonus point:

Mysql

- ❏ Utilización de algún tipo de datos especial (geografia, GIS)

MongoDB

- ❏ Almacenamiento de un fichero binario (imagen o documento pdf o word
- ❏ Requisitos de modelado
- ❏ Utilización de herencia entre clases