# Práctica JavaScript (I)

## Funcionalidad

Se desea implementar un servicio para la **gestión de un blog con persistencia y API REST**. Las **entradas** del blog podrán tener **comentarios** y para evitar la publicación de aquellos que puedan ser ofensivos, el servicio deberá incluir internamente un validador.

Las entidades de información que deberá gestionar el blog son:

- **Post**. Cada entrada contiene los siguientes campos: Nombre del autor, Nickname del autor, Titulo, Texto, Lista de comentarios.
- **Comment**. Cada comentario contiene los siguientes campos: Nickname del autor del comentario, Contenido, Fecha del comentario.
- **OffensiveWord:** Cada palabra tendrá asociada un campo “level” indicando la gravedad de la palabra de 1 a 5.

La aplicación ofrecerá los siguientes **endpoints REST**:

- Entradas y comentarios
    - Recuperación de todas las entradas (sin comentarios)
    - Recuperación de una entrada (con comentarios)
    - Creación de una nueva entrada (sin comentarios)
    - Borrado de una entrada existente (con todos sus comentarios)
    - Modificación de una entrada existente (se actualizará toda la información de la misma, excepto sus comentarios)
    - Adición de un nuevo comentario a una entrada
    - Modificación de un comentario existente
    - Borrado de un comentario existente en una entrada
- Palabras ofensivas
    - Creación, listado, borrado y modificación de una palabra ofensiva.

Si el comentario que se intenta incorporar contiene alguna de las palabras ofensivas registradas en la BBDD, el comentario no se podrá crear. La petición REST será rechazada con un código de error y se devolverá un JSON con información sobre la palabra ofensiva (o palabras) y su nivel. Las palabras ofensivas estarán almacenadas en la BBDD. Si al arrancar la aplicación se detecta que no hay palabras ofensivas en la BBDD, la aplicación deberá insertar un juego de palabras por defecto.

### Control de usuarios

La aplicación tiene que permitir varios tipos de usuarios:

- **Autenticados:**

    - **Administrador:** Los usuarios de tipo admin podrán realizar cualquier operación de la API REST.
    - **Publisher:** Los usuarios de tipo publisher podrán:
        - Crear entradas del blog.
        - Borrar y modificar únicamente las entradas que hayan creado ellos.
        - Borrar comentarios de sus entradas.
        - Añadir comentarios a otras entradas.
- **No autenticado:** Podrán consultar información pero no podrán añadir comentarios ni añadir posts.

Cualquier usuario podrá registrarse en la aplicación y se le asignará el rol de "publisher". Para ello, se creará un endpoint de creación de usuarios. Para simplificar la implementación, un usuario no se puede borrar ni modificar.

Se deberá implementar un script Node.js llamado **load_admins.js** que se conectará a MongoDB y creará los usuarios admin cuando se ejecute. Los usuarios estarán en un fichero de texto.

El mecanismo de autenticación de usuarios puede ser **Basic Auth + Tokens JWT** o **OAuth2**.

# Aspectos técnicos

La aplicación estará **dividida en, al menos, 4 módulos Node**:

- **app.js:** Tendrá el servidor Express.
- **controller.js**: Tendrá definidos los métodos de la API REST. Se usará un router[1] que será configurado en Express.
- **repository.js:** Módulo que contendrá el código de acceso a la base de datos.
- **validator.js:** Validador que verifica que el comentario no contiene ninguna palabra ofensiva.

**API REST:**

- La API REST deberá cumplir con el nivel de madurez 2 y el formato de las URLs deberá identificar recursos, no acciones.
- Se deberá entregar un fichero postman.json con al menos un ejemplo de cada una de los endpoints de la API REST.

**Persistencia:**

- La persistencia se implementará con MongoDB de forma obligatoria. Se podrán usar el driver oficial Mongo o el ODM Mongoose.
- Opcionalmente se podrá entregar la práctica con la persistencia de palabras ofensivas en una base de datos MySQL (con o si ORM).

[1] http://expressjs.com/es/4x/api.html#router

# Control de calidad

Se deberán implementar **tests unitarios** del validador con Jest. Estos tests, ya que son unitarios, no deberán acceder a la bbdd real. Es decir, hay que crear un doble del módulo que permite acceder a la base de datos. La funcionalidad que debería verificarse en los tests es:

- **Comentario con palabras ofensivas:** Con este test se debe verificar que si el comentario tiene una palabra ofensiva, se debe generar el error correspondiente.
- **Comentario sin palabras ofensivas:** Con este test se debe verificar que si el comentario no tiene palabras ofensivas debe obtenerse una validación OK.

# Mecanismo y fecha de entrega

Este documento describe la primera parte de la práctica. Este documento será ampliado con más funcionalidades a medida que se vaya avanzando en el curso. En concreto, se ampliará en:

- Control de usuarios
- Testing de sistema de la API REST
- Cliente implementado con Vue
- Testing de sistema del interfaz con Cypress.io o Selenium.

La práctica se entregará completa una vez acabado el curso. La fecha concreta está por confirmar, pero seguramente será el día 27 de Enero de 2019 a primera hora de la mañana.