ADI_2223

Práctica evaluable 1: Diseño e implementación de APIs REST

Parte 2: Implementación del API

En esta fase 2 debéis realizar la implementación del API que habéis diseñado en la fase 1. Si al realizar la implementación os dais cuenta de que necesitáis hacer algún cambio sobre el diseño original, entregadlo junto con la implementación (solo los cambios al diseño, no el diseño entero).

> Recordad que por el momento vamos a diseñar **solamente el API**, no el sitio web. Este se desarrollará en la práctica 3. Eso quiere decir que por el momento **no tendremos interfaz de usuario**.

Requisitos mínimos de la implementación

Como mínimo todas las prácticas deben cumplir estos requisitos para poder aprobar. Los requisitos mínimos se valorarán con un 4 máximo y la documentación de la fase inicial con 1.5 puntos máximo.

Implementación de las llamadas al API

Se deben implementar al menos 7 llamadas distintas al API (más la de autentificación, o *login*, que se describe en el apartado siguiente). Debe haber al menos dos casos de GET (1 de lectura de un recurso sabiendo su `id` y 1 de lectura de una colección), y al menos un caso de POST, PUT o PATCH y DELETE. Esto nos da 5 llamadas. Las dos restantes pueden ser del tipo que queráis.

Autentificación

El API debe permitir autentificación mediante JSON web token. Para ayudaros en la implementación solo podéis usar el paquete jwt-simple pero no librerías adicionales (ya que el objetivo es aprender cómo funciona JWT).

En el contexto de JSON Web token "hacer login en la aplicación" es realmente "obtener el token". Mapeadlo con una petición POST a la URL que queráis, pasando en la petición HTTP el

login y el password en JSON y obteniendo como resultado en *token* en el cuerpo de la respuesta. Esta es una llamada adicional a las otras 7 que debéis implementar.

De momento no implementaremos el *logout* ya que no será tarea del servidor, en la práctica del cliente veremos cómo implementar este caso de uso.

Persistencia de los datos

Para alcanzar los requisitos mínimos no hace falta usar una base de datos, **los datos pueden ser estáticos, no modificables y estar almacenados en memoria**. Para almacenarlos podéis usar la estructura de datos que queráis. Es decir desde el punto de vista de los datos el API no es más que un *mockup*.

De este modo las peticiones de tipo GET siempre devolverían los mismos datos fijos y las peticiones que modifiquen datos (POST/PUT/DELETE) devolverán al cliente el código de estado correspondiente pero en realidad no modificarán nada (por ejemplo devolver un 201 si se solicita borrar una entidad aunque en la realidad no se borre y al hacer luego GET siga existiendo. No obstante el API debería comprobar que la operación tiene sentido (por ejemplo devolver un 404 si se intenta borrar una entidad que no existe).

> Para inicializar los datos en memoria o en la BD podéis ayudaros de herramientas como `faker.js`, que os permite generar datos como nombres, apellidos, ciudades, etc...con apariencia de "reales".

Requisitos "adicionales"

Para poder puntuar estos requisitos es necesario haber implementado correctamente los requisitos mínimos. De este apartado podéis elegir los requerimientos que deseéis.

(hasta 1.5 puntos) Implementar la persistencia del API con base de datos. Podéis usar para ello tanto el motor de base de datos como las librerías Node que queráis. En caso de implementar este apartado no es necesario guardar además los datos en memoria.

(hasta 1 punto) *Testing* automatizado (NO manual) de todas las llamadas al API. Podéis usar las herramientas que deseéis, por ejemplo con Postman se pueden hacer *scripts* para *tests* con un código JS muy sencillo. La documentación original está en inglés pero también tenéis algún que otro tutorial en castellano.

(hasta 1 punto) Documentar el API usando alguna de las herramientas que se comentarán brevemente en clase de teoría (Swagger/OpenAPI, RAML, API Blueprint), o cualquier otra que conozcáis.

(hasta 1 punto) Que se puedan subir archivos binarios al API y no solo enviar texto. Esto se puede implementar de varias formas. La más común es usando un `Content-Type` en la petición llamado `multipart/form-data`. Por ejemplo en este tutorial podéis encontrar más

información, o podéis simplemente buscar en Google "file upload nodejs multipart". No deberíais necesitar de momento HTML para probar la funcionalidad. En el tutorial referenciado se hace con Postman, pero también podríais buscar en Google cómo hacerlo con Curl u otra herramienta en línea de comandos.

(hasta 1 punto) Implementar *paginado* en el API. Podéis hacerlo basado en *offset* o en cursores, como queráis. Lo usaréis en el GET que lea una colección de recursos. En el resultado devuelto hay que incluir como metadatos al menos el total de resultados, el número de resultados en la página actual y la URL que me devolvería la página anterior y la siguiente.

Cualquier otra idea que se os ocurra como ampliación a los requisitos mínimos consultadla antes con el profesor, para ver cuánto se podría valorar en el baremo.

Plazo de entrega y baremo de evaluación

El plazo de entrega de la práctica concluye el **lunes día 24 de octubre a las 23:59**. La entrega se realizará en moodle enviando un único archivo comprimido en .zip o similar.

Resumiendo, la calificación será:

- Hasta 4 puntos implementación requisitos mínimos
- Hasta 1.5 puntos la documentación de casos de uso, relaciones entre recursos, (no hace falta que lo volváis a entregar, ya lo entregáis en la fase inicial de la práctica)
- Hasta 4.5 puntos los requerimientos optativos. Elegid los que queráis de entre los propuestos. También podéis proponer cualquier ampliación que se os ocurra aunque no esté en la lista, pero consultadlo antes conmigo para que veamos si es razonable y cuánto se podría valorar en el baremo.

Se entregará:

- Un proyecto Node con un `package.json` con las dependencias. **No debéis incluir el directorio `node_modules`** en la entrega.
- Modificaciones al diseño inicial, si las hubiera (solo si habéis hecho algún cambio en el diseño del API con respecto a la entrega del día 8 de octubre, y solo hace falta entregar la parte que haya cambiado)
- Un `LEEME.txt` donde listéis los requerimientos adicionales que hayáis implementado y dónde están en el proyecto (por ejemplo "implementado paginado con cursores en el listado de clientes, en el archivo `api.js`, asociado a la ruta `/api/clientes`")