# Proyecto BACK-END

Realizado por:

- ☐ Iker García López
- ☐ Pablo Alcolea Aguilar
- ☐ Oier Barrutiabengoa Errandonea
- ☐ Mikel Elola Echechipia

Universidad del País Vasco
Euskal Herriko Unibertsitatea

Descripción:

En esta segunda parte del proyecto estamos diseñando una aplicación web, en el ámbito de back-end, donde tenemos que trabajar con audios jugando con una base de datos, en este caso MongoDB. Además, terminaremos diferentes funcionalidades que en la anterior parte solo se trató su parte en el front-end y poder autenticarnos al entrar a la aplicación.

Hemos implementado las siguientes API's:

- 1: API/List
- 2: API/Upload
- 3: API/Delete
- 4: API/Play
- 5: Iniciar sesión (autenticación)

También hemos implementado la funcionalidad cleanup para limpiar los audios antiguos.

A continuación, vamos a desarrollar cada una de estos puntos.

1. API/List:

En esta api, se recoge el identificador de usuario mediante la URL, y esta nos devolverá una serie de ficheros asociados a dicho usuario. Además, mediante la función handleList, cogiendo el parámetro del identificador, nos devolverá desde la base de datos los últimos cinco audios que presente este usuario, ordenados por fecha.

Para ello, hemos implementado en main.js un fetch que permite obtener la lista de audios del usuario.

En la ruta index.js, tratamos ese fetch y llamamos a la función mencionada anteriormente, handleList. En ella, se pasa por parámetro el ID del usuario y se obtienen las últimas 5 grabaciones.

2. API/Upload:

La API/Upload es una interfaz que facilita la carga de archivos de audio en una aplicación web. Esta gestiona la subida de grabaciones de audio a la aplicación y realiza operaciones asociadas en la base de datos y en el sistema de archivos.

Implementamos un fetch de tipo POST, que nos permite subir a la base de datos del usuario un audio grabado por él. Se pasa por parámetro el usuario.

En index.js, tratamos ese audio tratado mediante POST, utilizando también multer, que es un Middleware de Express. Gracias a esto solo esperamos un archivo llamado “recording”, y lo trataremos con req.file. A continuación, extraemos el ID del usuario, se obtiene el blob, asignandole el valor “audioInfo”, que recoge los parámetros “name”, ”filename”, ”date”, ”accessed” y lo inserta tanto en la base de datos como en el directorio “./recordings”.

3. API/Delete :

En esta API, cuando nosotros desde la aplicación queremos borrar un audio, se representa en una URL que tiene como parámetros el usuario que ha realizado esa operación así como el audio a eliminar.

Todo esto se trata en el POST, que mediante estos parámetros, borra de la base de datos el audio de la colección de ese usuario, además de borrarlo en la carpeta recordings.

Por último, devolverá “/list” del usuario, es decir, las últimas cinco grabaciones de este.

4. API/Play :

Define una ruta con un parámetro de ruta "filename". Lee de forma sincrónica un archivo de audio ".ogg" correspondiente al valor del parámetro y actualiza un documento en una colección MongoDB. La actualización se basa en el parámetro "filename" y modifica el campo "accessed" con la marca de tiempo actual. Cuando se llama a esta api se carga una nueva página llamada play.ejs en la que solo nos muestra el botón de reproducción

5. Iniciar sesión:

La implementación permite a los usuarios autenticarse utilizando Google OAuth. Se genera un identificador único para cada usuario y se utiliza Passport para gestionar el flujo de autenticación, interactuando con las APIs de Google para la autenticación y almacenando la información del perfil del usuario.

CleanUp

Por último, hemos implementado el método cleanup mediante la función setInterval para ejecutar un proceso de limpieza periódico. Cada hora (3600000 milisegundos), el código busca y elimina datos antiguos en la base de datos en el directorio de grabaciones. Primero, utiliza la fecha actual menos cinco días (432000000 milisegundos) como referencia. Luego, usa esta referencia para eliminar documentos de la base de datos cuya fecha sea anterior a los cinco días. Posteriormente, examina los archivos en el directorio de grabaciones y elimina aquellos cuya última modificación sea anterior a los cinco días.

Problemas

Nos hemos topado con varios problemas a la hora de implementar el api/upload, puesto que no acabábamos de lograr que el audio grabado se guardara de forma correcta, para arreglar esto usamos el blob y lo

convertimos en un objeto tipo buffer para posteriormente escribirlo en memoria.
También había un problema al pasar los datos del audio al realizar el api/play. Para arreglar eso decidimos convertir el audio a una cadena de texto para pasarlo por parámetro a la página play.ejs y después tratarlo allí. Más allá de estos 2 casos no hemos tenido grandes problemas y al final, hemos realizado el proyecto en su totalidad.

Enlace
https://github.com/RosherB/PracticaBackEnd