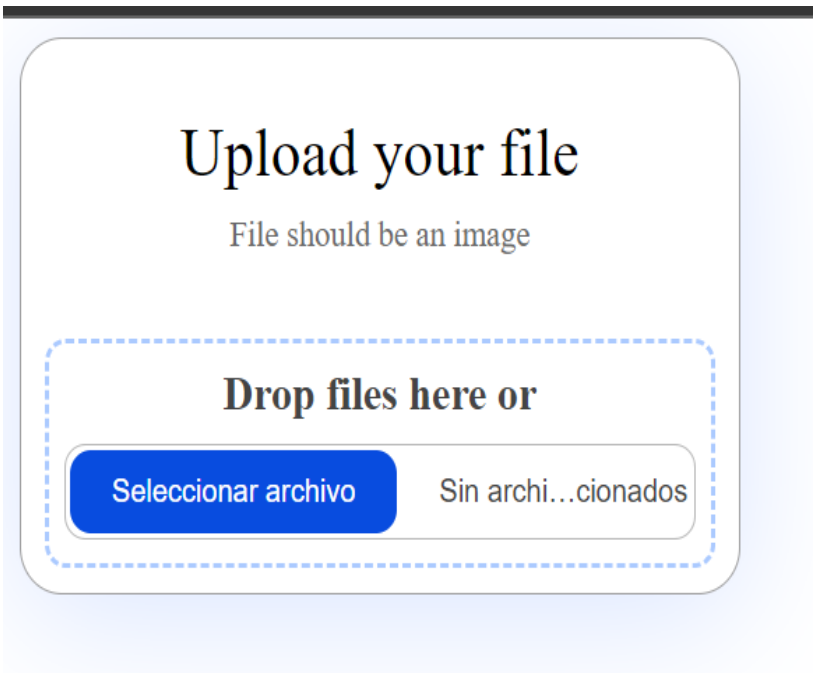
Proyecto 1

Componente de gestión de alojamiento de archivos en servidor (Cloud Storage)

Integrantes: Alan Garcia, Aldo Capurro, Mateo Roca, Santiago Pili, Beatriz Villanueva, Manuel Marques

Descripción y estructuración de la API V1.0



Para el cumplimiento del primer objetivo del proyecto utilizamos una clase fileHandler con un solo método encargado de subir el archivo dentro de un directorio local en el servidor, este devuelve una respuesta satisfactoria acompañada de la URL del archivo, o un código de error especificando el tipo de inconveniente

Endpoints

GET /:

- Descripción: Este endpoint es accesible desde la ruta raíz (http://localhost:3000/).
- Operación: Se utiliza la operación HTTP GET.
- Acción: Muestra un formulario HTML que permite a los usuarios seleccionar un archivo y enviarlo para su carga.
- Uso: Los usuarios pueden acceder a esta página para seleccionar un archivo y cargarlo.

POST /upload:

- Descripción: Este endpoint está diseñado para procesar la carga de archivos.
- Operación: Se utiliza la operación HTTP POST.
- Acción: Procesa el archivo cargado y lo guarda en el servidor en el directorio de carga especificado("./uploads").
- Uso: Cuando los usuarios envían un archivo utilizando el formulario en la página principal, este endpoint procesa la solicitud y almacena el archivo en el servidor.

Formato de serialización

En esta versión los datos no están serializados, trabajamos con los datos en bruto como archivos binarios.

GET /:

- No envía datos serializados en formato JSON o XML. En su lugar, proporciona un formulario HTML que permite a los usuarios seleccionar un archivo local para cargar.

POST /upload:

- Tampoco envía datos serializados en formato JSON o XML. En cambio, utiliza un formulario HTML con el atributo enctype="multipart/form-data", que permite a los usuarios seleccionar un archivo local y enviarlo al servidor en su formato original, como un archivo binario.

Detalles

Ejemplo de uso

Cargar un archivo

1. Realiza una solicitud GET a la ruta "/" para acceder al formulario de carga de archivos.
2. Selecciona un archivo local utilizando el campo de selección de archivos en el formulario.
3. Envía el formulario haciendo clic en el botón "Upload".
4. La API procesará la solicitud y almacenará el archivo en la carpeta "uploads" en el servidor.
5. Recibirás una respuesta indicando si la carga se realizó correctamente.

Notas adicionales

Asegúrate de que la carpeta "uploads" exista en el directorio donde se encuentra este servidor Node.js antes de usar la API. De lo contrario, la API creará la carpeta automáticamente.

La API no incluye autenticación ni autorización. Se recomienda implementar medidas de seguridad adicionales en un entorno de producción

Proyecto 1

Componente de gestión de alojamiento de archivos en servidor (Cloud Storage)

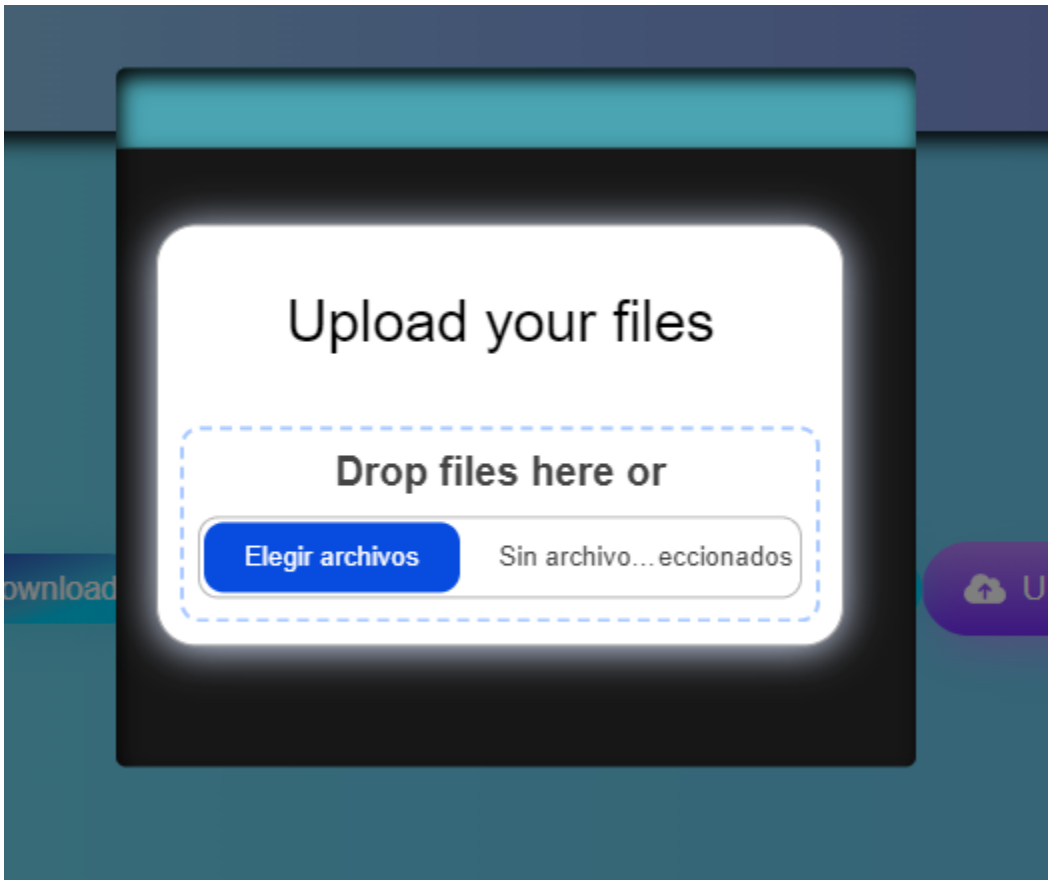
Integrantes: Alan Garcia, Aldo Capurro, Mateo Roca, Santiago Pili, Beatriz Villanueva, Manuel Marques

Descripción y estructuración de la API V1.1

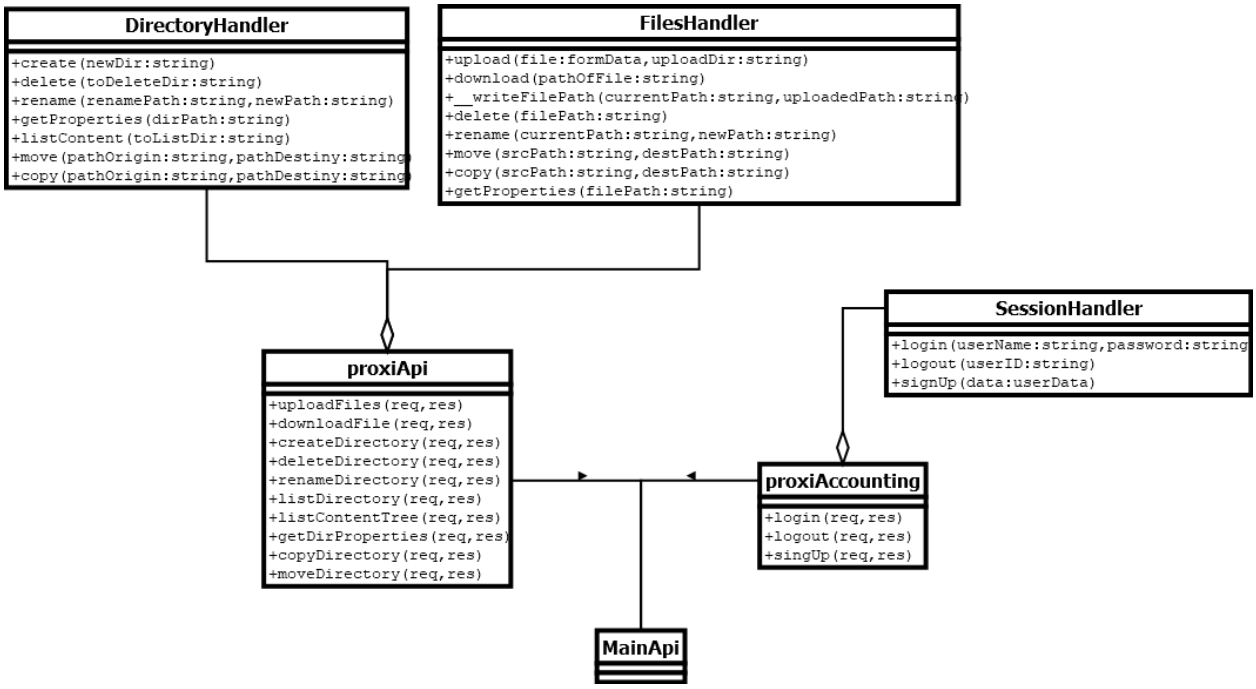
En el nuevo estado de la API tenemos 3 clases bases para la construcción encargadas de la sesion(sessionHandler), de los archivos(fileHandler) y de las carpetas(directoryHandler). El

sesionHandler cuenta con tres metodos, uno para el login, otro para el logout y un tercero para registrar usuarios nuevos, cuando ingrese al servicio lo primero que se va a encontrar el usuario va a ser esta pantalla de inicio. Una vez que el usuario ingrese se va a encontrar con su servidor de cloud storage. Desde esta pantalla va a poder tener control a todas las funcionalidades básicas de cualquier servicio de almacenamiento en la nube como manipular sus archivos y carpetas(renombrar, mover, borrar, crear carpetas nuevas y descargar) .

Para la carga de archivos desde el botón upload nos permite tanto seleccionar de una carpeta como arrastrarlos dentro de la ventana de carga de archivos, puede manejar la carga múltiple y tambien nos muestra el progreso de la carga con una barra.



Tenemos dos clase proxi para filtrar los llamados a los métodos separados en un proxi para todo lo que implica la carga de archivos y otro para todo lo que se trabaja con la sesion del cliente(se adjunta un UML para poder verse mas claro).



Endpoints

EndPoint	Método	Parámetros de solicitud	Respuesta Exitosa	Respuesta de Error	Cabeceras de la petición
/directoryHandler/create	“POST”	string;	{status: true, message: "Directory Created"}	{status: false, message: "Imposible Create Directory"}	*user-id*
/directoryHandler/delete	“POST”	string;	{status: true, message: "Directory	{status: false, message:	*user-id*

			Deleted Successfully"}	"Imposible Delete Directory"}	
/directoryHandler/rename	“POST”	oldName: string; newName: string;	{status: true, message: "Directorio renombrado correctamente"}	{status: false, message: ""}	*user-id*
/directoryHandler/move	“POST”	originPath: string; destinationPath: string;	{status: true, message: "Directorio movido correctamente"}	{status: false, message: ""}	*user-id*
/directoryHandler/list	“POST”	string;	{status: true,files:[]}	{status: false, message: ""}	*user-id*
/directoryHandler/copy	“POST”	originDir: string; newDir: string;	{status: true, message: "Directorio copiado correctamente"}	{status: false, message: ""}	*user-id*
/directoryHandler/properties	“POST”	propertiesDir: string;	{status: true, message: ""}	{status: false, message: ""}	*user-id*
fileHandler/upload	“POST”	formData	{status: true, message: "File uploaded"}	{status: false, message: “”}	*user-id*
fileHandler/rename	“POST”				*user-id*
fileHandler/download	“POST”	fileName:string;	{status: true, message: ""}	{status: false, message: ""}	*user-id*
fileHandler/properties	“POST”				*user-id*
sessionHandler/login	“POST”	userName:string; password:string;	{status: true, message: "login success"}	{status: false, message: "login error"}	*user-id*
sessionHandler/logout	“POST”	userId:string;	{status: true, message: "logout success"}	{status: false, message: "login error"}	*user-id*
sessionHandler/signUp	“POST”	userData:data;	{status: true, message: "sign up success"}	{status: false, message: "the user already exists", "sign up failed"}	

Formato de serialización

Detalles

Ejemplo de uso

Inicio de sesión

1. La primer pantalla que el usuario se encuentra es la de inicio de sesión.
2. Se pide el nombre de usuario y la contraseña del mismo.
3. Se envia la info y se corrobora con la info en la base de datos, si todo esta de acuerdo se ingresa al manejador de archivos de su perfil.

Registro

4. En caso de no tener un usuario registrado el cliente va a poder registrarse en el servidor.
5. Va a ingresar a una ventana donde le pide todos los datos necesarios para registrarse(nombre de usuario, nombre completo, email, etc).
6. Cuando el registro este completo se crea en el servidor una carpeta asociada a ese usuario en donde va a poder cargar sus archivos, modificarlos, borrarlos y descargarlos.

Carga de archivos

7. El usuario presiona el botón de Upload.
8. Permite seleccionar archivos desde la ventana de selección o arrastrarlos a la ventana.
9. Permite deseleccionar elementos que se agregan por error.
10. Una vez confirmado que todo es correcto se confirma la carga de archivos y se muestra el progreso de las cargas.

Manipulación de archivos

11. El usuario ya dentro de su directorio va a poder ver sus archivos y carpetas teniendo disponible un repertorio de acciones tipo CRUD.
12. Va a poder renombrar los archivos y carpetas que esten en su directorio, se va a poder mover entre sus directorios y tambien mover archivos y a otras carpetas.
13. Va a poder crear nuevos directorios dentro del mismo.
14. Tambien puede descargar archivos desde el storage a cualquier computadora desde la que se conecte.

Notas adicionales

Al cargar un archivo siempre se carga desde el directorio raíz del usuario, no desde el directorio en el que estemos parados al momento de hacer la carga de archivos.

Al estar en medio de una carga de archivos si apretamos fuera de la ventana modal de carga, se cancela pero sin ningún mensaje de advertencia, o seguir con la carga en segundo plano.