Manual Técnico

Introducción:
A continuación se tratará de explicar de forma detallada la estructura y el funcionamiento de cada una de las clases, componentes, módulos y archivos en general usados para la elaboración de el frontend y el backend del proyecto “CloudArch”.

Tecnologías usadas:
- Node 18.17.1
- Angular 16.2.1
- express
- mongodb 7.0.3
- Docker

Requerimientos:
- Node (última versión lts recomendable).
- Angular
- npm
- mongodb
- docker

Backend:
El backend se encuentra estructurado de la siguiente manera:

- carpeta config.
- carpeta controllers.
- carpeta routes.
- carpeta models.
- index.js

El archivo index.js es el encargado de escuchar las peticiones http del frontend y relacionar dichas peticiones con los archivos en la carpeta routes.

Carpeta routes:
Esta carpeta contiene todos los archivos que contienen las distintas rutas soportadas por el backend para el correcto funcionamiento del backend. Estos archivos.js son los encargados de relacionar las peticiones con los controllers. Entre los archivos tenemos:

- **auth.routes.js:** Este archivo contiene la petición “get” para verificar las credenciales del usuario a ingresar en el sistema (login).

- **user.routes.js:** Este archivo contiene las distintas peticiones CRUD para la creación, lectura, actualización y eliminación de los usuarios en el sistema.
- **file.routes.js:** Este archivo contiene las distintas peticiones CRUD para la creación, lectura, actualización y eliminación de los archivos creados por los usuarios en el sistema.
- **directory.routes.js:** Este archivo contiene las distintas peticiones CRUD para la creación, lectura y eliminación de los distintos directorios creados por los usuarios en el sistema.
- **sharedFile.routes.js:** Este archivo contiene las distintas peticiones CRUD para la creación, lectura, modificación y eliminación de los archivos compartidos por los usuarios del sistema.
- **deletedFile.routes.js:** Este archivo contiene las distintas peticiones CRUD para el manejo de la papelera al momento de eliminar archivos en el sistema.

Carpeta models:
Esta carpeta contiene todos los modelos usados en la base de datos. Entre los modelos encontramos:

- **File:** Este modelo hace referencia a un archivo creado por el usuario. La información que este almacena es: nombre de usuario, nombre del archivo, contenido del archivo, ruta del archivo.
- **Directory:** Este modelo hace referencia a una carpeta creada por el usuario. La información que este almacena es: nombre del usuario, nombre de la carpeta, ruta de la carpeta.
- **User:** Este modelo hace referencia a los usuarios en el sistema. La información contenida es: nombre de usuario, nombre del empleado, tipo de empleado y contraseña.
- **SharedFile:** Este modelo hace referencia a los archivos compartidos entre los usuarios del sistema. La información contenida es: nombre del usuario que lo compartió, nombre del usuario a quien se compartió, nombre del documento, fecha de compartido, nombre del archivo, contenido del archivo.
- **DeletedFile:** Este modelo hace referencia a los archivos eliminados por los usuarios del sistema. La información que este modelo almacena es: nombre del usuario que lo eliminó, nombre del archivo, contenido del archivo.

Carpeta config:
Esta carpeta contiene un único archivo que posee toda la configuración necesaria para conectar la aplicación con la base de datos en mongoDB.

Carpeta controllers:
Esta carpeta contiene todos los archivos que contienen la lógica del backend. Los controllers son los siguientes:

- **file.controller.js:** Este archivo contiene los siguientes métodos:
    - **findUserFiles:** este método encuentra todos los archivos del usuario.
    - **findFileByName:** este método encuentra la información de un archivo en base a su nombre.
    - **saveFile:** este método guarda un nuevo archivo en la base de datos.
    - **updateFile:** este método actualiza un archivo existente en la base de datos.
    - **deleteFileByName:** este método elimina un archivo existente en la base de datos.
    - **moveFile:** este método cambia la ruta anterior de un archivo a una nueva.
    - **copyFile:** este archivo crea una copia de un archivo con una ruta diferente.
- **user.controller.js:** Este archivo contiene los siguientes métodos:
    - **findUser:** este método encuentra la información del usuario en base al nombre y la contraseña.
    - **saveUser:** este método guarda un nuevo usuario en la base de datos.
    - **deleteUser:** este método elimina un usuario existente en la base de datos.
    - **findAll:** este método encuentra todos los usuarios en la base de datos.
    - **updateUser:** este método actualiza un usuario existente en la base de datos.
    - **findUserByUsername:** este método encuentra la información de un usuario en base a su nombre de usuario.
- **directory.controller.js:** este archivo contiene los siguientes métodos:
    - **findUserDirectories:** este método encuentra todos los directorios de un usuario.
    - **createDirectory:** este método crea un directorio nuevo en la base de datos.
    - **findAllWithout:** este método encuentra todos los directorios excepto uno (método usado para mover directorios, ya que no puede mover el directorio dentro de sí mismo).
    - **moveDirectory:** este método cambia la ruta del directorio en la base de datos.
    - **deleteDirectory:** este método elimina al directorio de la base de datos.
    - **copyDirectory:** este método copia el directorio a una nueva ruta.
- **sharedFile.controller.js:** este archivo cuenta con los siguientes métodos:
    - **findUserSharedFiles:** este método encuentra todos los archivos que se han compartido a un usuario.
    - **shareFileTo:** este método comparte un archivo de un usuario a otro.

    - **findSharedFile:** este método encuentra la información de un archivo compartido.
    - **updateSharedFile:** este método actualiza la información de un archivo compartido.
    - **deleteSharedFile:** este método elimina un archivo compartido de la base de datos.
- **deletedFile.controller.js:** este archivo cuenta con los siguientes métodos:
    - **findAll:** este método encuentra todos los archivos eliminados en la base de datos.

Frontend:
El frontend se encuentra estructurado de la siguiente manera:

- core
- modules
- services
- shared

Core:
Esta carpeta (no es tan importante) cuenta con un par de modelos usados en el frontend para evitar el uso de any (en algunas ocasiones).

Modules:
Los módulos son la parte más importante del sistema, estos nos ayudan a poder hacer carga perezosa y así evitar que se carguen componentes que no se utilizarán hasta determinado momento. los módulos usados en el sistema fueron:

- **auth:** este módulo fue utilizado para la autenticación en el sistema. Contiene un único componente que vendría siendo el authPage.
- **employee:** este módulo fue utilizado para el apartado Empleados en el sistema. Contiene una única página que es: employeePage. Además cuenta con un par de componentes para evitar que la página principal tuviera toda la información, los componentes extra son: addEmployee que contiene el formulario para la creación de nuevos usuarios y employeesTable que muestra una tabla con todos los usuarios registrados en el sistema.
- **home:** este módulo fue utilizado en el apartado Documentos, por lo que es el más extenso. Este módulo cuenta con las siguientes páginas:
    - **HomePage:** este componente es la página principal del módulo, simplemente contiene un dashboard que redirecciona a showDocumentsPage, ShowDeletedFilesPage y ShowSharedDocumentsPage.

- **ShowDeletedFilesPage:** este componente muestra todos los archivos que han sido eliminados en el sistema.
- **ShowDocumentsPage:** este componente muestra todos los archivos y directorios que el usuario ha creado en el sistema.
- **ShowFilePage:** este componente muestra la información de un archivo al que el usuario quiera acceder o que acaba de crear.
- **ShowSharedDocumentsPage:** este componente muestra todos los archivos que han sido compartidos al usuario.
- **ShowSharedFilePage:** este componente muestra la información del archivo compartido seleccionado por el usuario.

Además de eso cuenta con algunos componentes extra en la carpeta components. Estos componentes son utilizados como plantillas para dialog usados para mover y copiar archivos y directorios.

Services:
Esta carpeta cuenta con archivos utilizados para realizar las peticiones http por medio de HttpClient al backend.