Documentación del backend/API

Estándares

Definición de arquitectura de archivos y estándares a utilizar en el desarrollo del backend del proyecto

```
api/
├─index.js
├─firebase.js
├─controllers/
├ └─ auth.controllers.js
├──  suscriptions.controllers.js
├──  logs.controllers.js
├─models/
├ └─ File.js
├──  User.js
├──  Log.js
├─routes/
├─└─ auth.js
├──  files.js
├──  subs.js
├──  users.js
├──  log.js
```

- index.js: Punto de arranque de la API
- firebase.js: Funciones y métodos necesarios para la conexión con Firebase, esto para el manejo de archivos
- auth.controllers.js: Funciones y métodos para autenticación
- subscriptions.controllers.js: Funciones y métodos para manejo de suscripciones
- File.js: Definición de Esquema de Archivo
- User.js: Definición de Esquema de Usuario
- auth.js: Definición de Rutas

Controladores

La defindción de los controladores fue basada en el requerimiento de tener la posibilidad de hacer revisión de la autenticación y estado de la suscripciones de los usarios en multiples endpoints de la API. También el manejo del registro de logs fue una necesidad presentada en

multiples direcciones por lo que fue agregado un controlador especial para el procesmiento de esto

Definición

- **auth.controller**: Contraldor para autenticación, este controlador se encarga de crear y verficar las fichas para autenticar a los usuarios utilizando librerías y técnicas seguras para la encriptación/desencriptación de información de los usuarios. Tambien en este controlador se realiza la creación y control de la sesiones de usuario, también se provee al frontend de la información necesaria para el manejo de las sesiones.
    - **createAccessToken**: Crea los tokens de sesión basados en la información de usuarios.
    - **checkSession**: Dado un token verifica la autenticidad del mismo y que aún represente una sesión con un tiempo de vida válido.
    - **decryptUserData**: Dato un token desencripta el token y consigue a partir de este la información del usuario
- **subscriptions.controller**: Controlador para el manejo de las suscripciones, el trabajo de este controlador es la verificación del estado de la suscripción del usuarios
    - **hardCheckSubscription**: Cuando un usuario realiza una petición a un endpoint inaccesible sin suscripción este manejador retorna al frontend un mensaje de error en el cual se determina que requiere una suscripción activa para seguir el proceso.
    - **softCheckSubscription**: Agrega a la petición información sobre el estado de la suscripción del usuario, esta funciona permite seguir con el proceso, en un contexto en el que se determinó el estado de la suscripción.

Base de datos

MongoBD

Base de datos no relacional utilizando el servicio de base de datos MongoDB donde se almacena la información de los usuarios y accesos encriptados a los archivos.

Esquemas

User

Tabla de usuarios en la que se definen los usuarios, suscripciones, relaciones de amistad y archivos compartidos.

Campos:

- username: Nombre de usuario
- email: Correo electrónico
- password: Contraseña
- subscription: Fecha de vencimiento de la suscripción
- followers: Seguidores
- following: Usuarios a quienes sigo
- sharedFiles: Usuarios compartidos

File

Esquema de archivos en la cual se guardan de forma encriptadas los accesos a los archivos de los usuarios.

Campos:

- userID
- desc
- hash
- baseDir

Bitácora

Esquema para el almacenamiento de los registros de bitácora los cuales guardan los procedimientos dirigidos por usuarios

Campos:

- userID
- Date
- Type
- action

FireStorage

Solución de Google para el almacenamiento de archivos de forma segura. Esta solución resolvió el problema del almacenamiento de los archivos de los usuarios y obtenerlos de forma segura y eficiente con gran ancho de banda para garatizar la mejor experiencia de subida y bajada de archivos

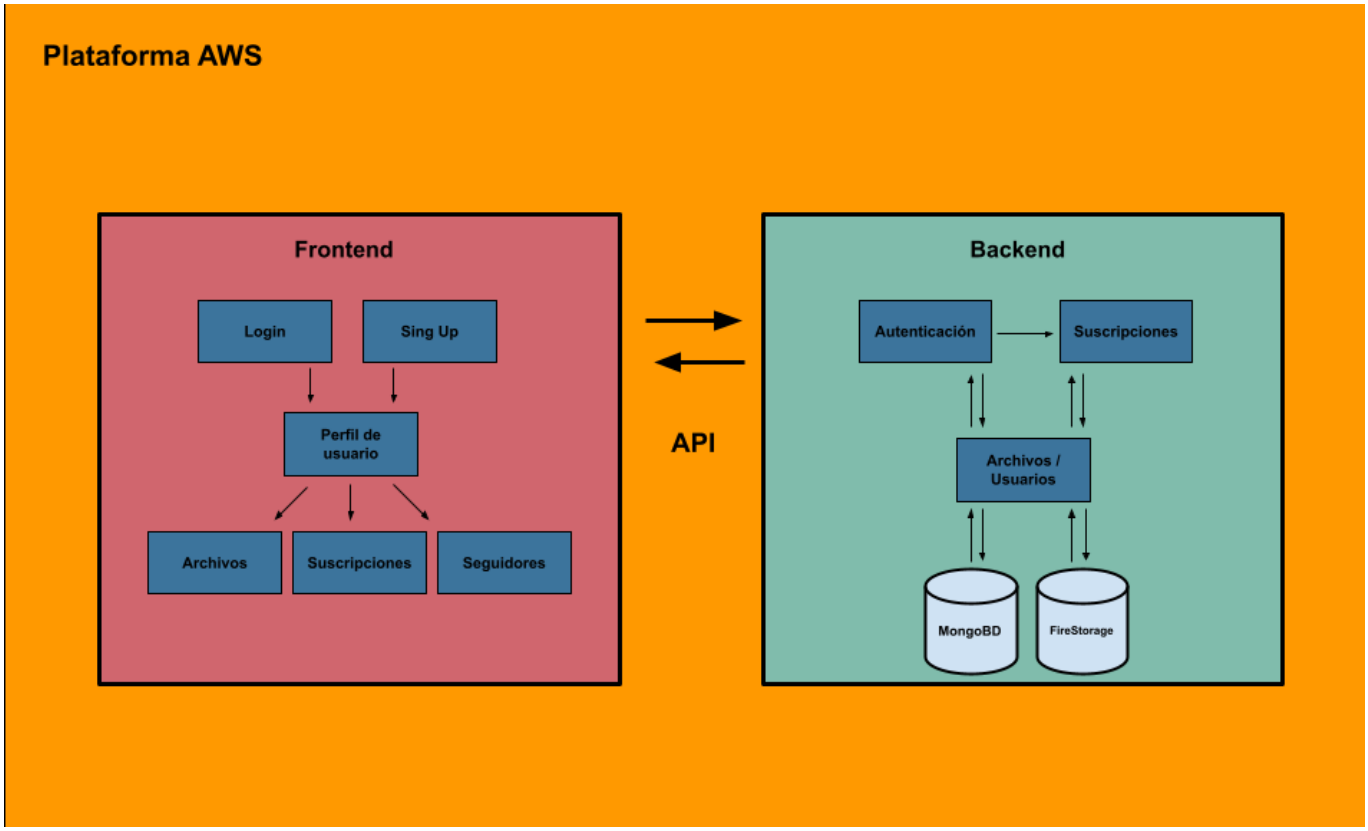
Arquitectura

Para el diseño de la arquitectura de nuestra aplicación nos basamos en los modelos Cliente/Servidor separando nuestro proyecto en dos grandes componentes:

- Frontend: Diseño y desarrollo web ejecutado en el cliente del usuario, esta es lo que coloquialmente conocemos como "La parte visual" del proyecto ya que es con lo que el usuario interactura directamente. Utilizamos las técnologías Vue y Typescript para el desarrollo de este gran componente

- Backend: Procesamiento y lógica de negocio, encargado del manejo de usuario, suscripciones y archivos.

Para el intercambio entre estos dos componentes utilizados el concepto de una API, la cual es el intermediario encargado de la transferencia de información.



API

Registro

Registra los usuarios en la base de datos.

URL: host:8800/api/auth/register

Método: POST

Headers de solicitud: Ninguno en especifico

Body de solicitud:

```
{
    "email": "email-user-test1",
    "password": "user-test2",
    "username": "user-test2"
}
```

Status de respuesta:

- 200 Solicitud correcta, usuario creado.
- 400 Representa error de la solicitud o datos erróneos
- 500 Error del backend

Headers de respuesta esperada: Ninguno notable

Body de respuesta:

- En status 200:

```
{
"accessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...v0xmpX8sdSxj7phffcsc--
AqhjiVrN1ZbnIlTVjtaCI"
}
```

- En status 400: Ejemplo al existir un usuario

```
{
    "driver": true,
    "name": "MongoError",
    "index": 0,
    "code": 11000,
    "keyPattern": {
        "username": 1
    },
    "keyValue": {
        "username": "user-test2"
    }
}
```


- En estatus 500: Ejemplo de existir un error en el backend


```
    {}
```


LOGIN

Verifica las credenciales de los usuarios existentes y les genera un token de acceso.

URL: host:8800/api/auth/login

Método: POST

Headers de solicitud: Ninguno en especifico

Body de solicitud:


```
    {
        "email": "email-user-test1",
        "password": "user-test1"
    }
```


Status de respuesta:

- 200 Solicitud correcta, login correcto.
- 400 Representa error de la solicitud o datos erróneos
- 500 Error del backend

Headers de respuesta esperada: Ninguno notable

Body de respuesta:

- En status 200:

```
{
"accessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...v0xmpX8sdSxj7phffcsc--
AqhjiVrN1ZbnIlTVjtaCI"
}
```

- En status 400: Ejemplo al no existir un usuario

```
    User not found
```

- En estatus 500: Ejemplo de existir un error en el backend

```
    {}
```

Actualizar a usuario

Actualiza la información del usuario, busca el usuario por ID y luego se encarga de actualizar su contraseña.

URL: host/api/users/:id

Ejemplo: localhost:8800/api/users/62df5890b010114e2c4d813b

Método: PUT

Headers de solicitud:

```
{
    authorization: eyJhbGciOiJIUzI1NiIsI...nR5cCI6IkpXVCJ9J1c-rFb8
}
```

Body de solicitud:

El body puede llevar cualquiera de los valores del usuario a actualizar.

Ejemplo con los posibles campos a actualizar

```
{
    "email": "email-user-test2",
    "password": "user-test2",
    "username": "user-test2"
}
```

Status de respuesta:

- 200 Solicitud correcta, Actualización correcta
- 400 Se trata de actualizar a otro usuario
- 500 Error del backend

Headers de respuesta esperada: Ninguno notable

Body de respuesta:

Se envía token de sesión actualizado, este token debe remplazar al antiguo

- En status 200:

```
{
"accessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...v0xmpX8sdSxj7phffcsc--
AqhjiVrN1ZbnIlTVjtaCI"
}
```

- En status 400: Al tratar de actualizar a otro usuario

```
    "You can only update your account"
```

- En estatus 500: Ejemplo de existir un error en el backend

```
    {}
```

Eliminar usuario

Busca un usuario por su ID, verifica que el usuario pueda eliminar solo su cuenta y luego la elimina.

URL: host/api/users/:id

Ejemplo: localhost:8800/api/users/62df5890b010114e2c4d813b

Método: DELETE

Headers de solicitud:

```
{
    authorization: eyJhbGciOiJIUzI1NiIsI...nR5cCI6IkpXVCJ9J1c-rFb8
}
```

Body de solicitud:

Ninguno

Status de respuesta:

- 200 Solicitud correcta, Usuario eliminado
- 500 Error del backend
- 403 Solicitud incorrecta, cuenta incorrecta

Headers de respuesta esperada: Ninguno notable

Body de respuesta:

- En status 200:

```
{
  "message": "Account deleted"
}
```

- En estatus 500: Ejemplo de existir un error en el backend

```
{
  "message": "Error"
}
```

- En estatus 403:

```
{
  "message": "You can only delete your account"
}
```

Seguir a Usuario

Busca a un usuario por su ID y al encontrarlo agrega al usuario actual a su lista de seguidos y viceversa.

URL: host/api/users/:id/follow

Ejemplo: localhost:8800/api/users/62e0b51d6432d04c700f2561/follow

Método: PUT

Headers de solicitud:

```
{
    authorization: eyJhbGciOiJIUzI1NiIsI...nR5cCI6IkpXVCJ9J1c-rFb8
}
```

Body de solicitud:

Ninguno

Status de respuesta:

- 200 Solicitud correcta, Follow correcto
- 403 Error en los datos de la solicitud
- 500 Error del backend

Headers de respuesta esperada: Ninguno notable

Body de respuesta:

Se envía token de sesión actualizado, este token debe remplazar al antiguo

- En status 200:

```
{
"accessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...v0xmpX8sdSxj7phffcsc--
AqhjiVrN1ZbnIlTVjtaCI"
}
```

- En status 403: Se trata de seguir a un usuario ya seguido o se trata de seguir a si mismo.

```
    "You can't follow yourself""
```

```
    "You already follow this user""
```

- En estatus 500: Ejemplo de existir un error en el backend

```
    {}
```

Dejar de seguir a Usuario

Busca a un usuario por su ID y al encontrarlo elimina al usuario actual de su lista de seguidos y viceversa.

URL: host/api/users/:id/unfollow

Ejemplo: localhost:8800/api/users/62e0b51d6432d04c700f2561/unfollow

Método: PUT

Headers de solicitud:

```
{
    authorization: eyJhbGciOiJIUzI1NiIsI...nR5cCI6IkpXVCJ9J1c-rFb8
}
```

Body de solicitud:

Ninguno

Status de respuesta:

- 200 Solicitud correcta, Unfollow correcto
- 403 Error en los datos de la solicitud
- 500 Error del backend

Headers de respuesta esperada: Ninguno notable

Body de respuesta:

Se envía token de sesión actualizado, este token debe remplazar al antiguo

- En status 200:

```
{
"accessToken":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9...v0xmpX8sdSxj7phffcsc--
AqhjiVrN1ZbnIlTVjtaCI"
}
```

- En status 403: Se trata de dejar de seguir a un usuario ya seguido o se trata de dejar de seguir a si mismo.

```
    "You can't unfollow yourself""
```

```
    "You don't follow this user""
```

- En estatus 500: Ejemplo de existir un error en el backend

```
    {}
```

Obtener información de un usuario

Se puede obtener información a partir del id o del username, esto puede hacer que el id o el username se omitan pero uno de los dos deben existir

URL: host:8800/api/users?userId=:id&username:Username

Ejemplo: localhost:8800/api/users?userId=62df5890b010114e2c4d813b

Método: GET

Headers de solicitud:

```
{
    authorization: eyJhbGciOiJIUzI1NiIsI...nR5cCI6IkpXVCJ9J1c-rFb8
}
```

Body de solicitud:

Ninguno

Status de respuesta:

- 200 Solicitud correcta, Unfollow correcto
- 500 Error del backend o no se obtuvo ningún usuario a partir de los parámetros

Headers de respuesta esperada: Ninguno notable

Body de respuesta:

Ejemplo


```
{
    "followers": [],
    "following": [
        "62e0b51d6432d04c700f2561"
    ],
    "isAdmin": false,
    "_id": "62df5890b010114e2c4d813b",
    "username": "user-test1",
    "email": "email-user-test1"
}
```


Subir archivo

Se encarga de subir los archivos a firebase storage y crear una referencia en mongoDB conteniendo el ID del dueño del archivo y el URL encriptado.
En caso de no tener una subscripción activa, el endpoint denegara la subida del archivo.

URL: host:8800/api/files/

Método: POST

Headers de solicitud:


```
{
    authorization: eyJhbGciOiJIUzI1NiIsI...nR5cCI6IkpXVCJ9J1c-rFb8
}
```


Body de solicitud:
form-data body

```
{
    'uploadedFile': File,
    'description': 'File description'
}
```

Body de respuesta:

Status 200:

```
{
    'url': "FileURL"
}
```

Obtener todos los archivos

Obtiene todos los archivos de un usuario. Se encarga de obtener todos los archivos dependiendo de la subscripción, si el usuario tiene una subscripción activa,
el mismo endpoint retornara los archivos personales y los archivos compartidos. En caso de tener una subscripción
vencida, solo retornara los archivos personales.

Obtener todos los archivos del usuario tanto los propios como los compartidos.

URL: localhost:8800/api/files/

Método: GET

Headers de solicitud:

```
{
  authorization: eyJhbGciOiJIUzI1NiIsI...nR5cCI6IkpXVCJ9J1c-rFb8
}
```

Body de solicitud:

Ninguno

Body de respuesta:

- Status 200: Archivos del usuario

```
[
    {
        "_id": "62f341398cd3420511f8f3b9",
        "userID": "62e0b751d8e5cc5e906f5dcd",
        "desc": "Una descripcion",
        "hash":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1cmwiOiJodHRwczovL2ZpcmViYXNlc3
RvcmFnZS5nb29nbGVhcGlzLmNvbS92MC9iL3ZhdWx0ZG9tZS5hcHBzcG90LmNvbS9vL2EzZW
VjYTdmLWM4MmEtNGUyZi04ZGQyLTY4ODVlNzYwZjliNDEudHh0P2FsdD1tZWRpYSZ0b2tlbj
1kYTFhMTBiZS1iZmQ4LTQ5OWMtYmUyYi0xMDcxYWU5ZGU4Y2UiLCJpYXQiOjE2NjAxMDkxMT
N9.zwlYLCk47saQtmCPpJITmcZoXdEopff71VqszIX2Za0",
        "baseDir": "a3eeca7f-c82a-4e2f-8dd2-6885e760f9b41.txt",
        "createdAt": "2022-08-10T05:25:13.087Z",
        "updatedAt": "2022-08-10T05:25:13.087Z",
        "__v": 0
    },
    {
        "_id": "62f3413d8cd3420511f8f3ba",
        "userID": "62e0b751d8e5cc5e906f5dcd",
        "desc": "Una descripcion",
        "hash":
"eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9.eyJ1cmwiOiJodHRwczovL2ZpcmViYXNlc3
RvcmFnZS5nb29nbGVhcGlzLmNvbS92MC9iL3ZhdWx0ZG9tZS5hcHBzcG90LmNvbS9vLzhlNz
hjZjhlLWYyN2MtNGFkNC1iOGZiLTBjMzYwNzhiYmVkNDIudHh0P2FsdD1tZWRpYSZ0b2tlbj
1kYTkwM2ViOC04MzNmLTQ2NjMtOGNiNS1iZmNmMTU2ODQ2MTUiLCJpYXQiOjE2NjAxMDkxMT
d9.67-GhPEScOBtTu2R7AbNPJyr1PpL6slInaSPtOn-bwc",
        "baseDir": "8e78cf8e-f27c-4ad4-b8fb-0c36078bbed42.txt",
        "createdAt": "2022-08-10T05:25:17.996Z",
        "updatedAt": "2022-08-10T05:25:17.996Z",
        "__v": 0
    }
]
```


- En status 500: Ejemplo de existir un error en el backend


```
 {}
```


Compartir un archivo

Se encarga de compartir archivos a otros usuarios siguiendo dos condiciones, la primera que el usuario tenga una subscripción activa, y la segunda
que los usuarios se sigan entre sí.

URL: localhost:8800/api/files/share

Método: POST

Headers de solicitud:

```
{
    authorization: eyJhbGciOiJIUzI1NiIsI...nR5cCI6IkpXVCJ9J1c-rFb8
}
```

Body de solicitud:
form-data body

```
{
    "fileID": "62f3416f8cd3420511f8f3bb",
    "sharedUserID": "62e0b751d8e5cc5e906f5dcd"
}
```

Body de respuesta:

- Status 200: Archivo compartido

  ```
  {
    "message": "File shared"
  }
  ```

- Status 403: Archivo ya fue compartido

  ```
   {
  "message": "File was already shared"
  }
  ```

- Status 500: Archivo no se pudo compartir, el usuario no es dueño del archivo

  ```
  {
    "message": "You can't share files that you don't own"
  }
  ```

Obtener un archivo

Obtiene un archivo por su ID, y verifica que el usuario tenga un acceso valido al mismo.

URL: localhost:8800/api/files/:id

Ejemplo: localhost:8800/api/files/80efs8a611786165pc

METHOD: GET

Headers de solicitud:

```
{
    authorization: eyJhbGciOiJIUzI1NiIsI...nR5cCI6IkpXVCJ9J1c-rFb8
}
```

Body de solicitud:

Ninguno

Body de Respuesta:

- Status 200: URL del archivo

  ```
  {
    "url":
  "https://firebasestorage.googleapis.com/v0/b/vaultdome.appspot.com/o/a3e
  eca7f-c82a-4e2f-8dd2-6885e760f9b41.txt?alt=media&token=da1a10be-bfd8-
  499c-be2b-1071ae9de8ce"
  }
  ```

- Status 403: Acceso denegado

  ```
  {
  "message": "User has no access to the file"
  }
  ```

Subscripción

Le agrega más tiempo de subscripción al usuario, por meses.

URL: localhost:8800/api/subs

METHOD: GET

Headers de solicitud:

```
{
    authorization: eyJhbGciOiJIUzI1NiIsI...nR5cCI6IkpXVCJ9J1c-rFb8
}
```

Body de solicitud:

Ninguno

Body de Respuesta:

- Status 200: Subscripción completada

  ```
  {
    "message": "Successful"
  }
  ```

- Status 500: Error de servidor

  ```
  {
    "message": "Internal Server Error"
  }
  ```