Universidad ORT Uruguay

Facultad de Ingeniería

Diseño de Aplicaciones 2
Obligatorio 1:
Documentacion Api

Juliette Ruchel - 203942
Francisco Martinez - 233126

Docente: `Ignacio Valle`

Entregado como requisito de la materia Diseño de Aplicaciones 2

https://github.com/ORT-DA2/233126-203942.git

10 de octubre de 2019

Declaraciones de autoría

Nosotros, Juliette Ruchel y Francisco Martinez, declaramos que el trabajo que se presenta en esa obra es de nuestra propia mano. Podemos asegurar que:

- La obra fue producida en su totalidad mientras realizábamos Diseño de aplicaciones 2;
- Cuando hemos consultado el trabajo publicado por otros, lo hemos atribuido con claridad;
- Cuando hemos citado obras de otros, hemos indicado las fuentes. Con excepción de estas citas, la obra es enteramente nuestra;
- En la obra, hemos acusado recibo de las ayudas recibidas;
- Cuando la obra se basa en trabajo realizado conjuntamente con otros, hemos explicado claramente qué fue contribuido por otros, y qué fue contribuido por nosotros;
- Ninguna parte de este trabajo ha sido publicada previamente a su entrega, excepto donde se han realizado las aclaraciones correspondientes.

Resumen

El presente documento tiene el propósito de especificar la Web api desarrollada en el ultimo mes. De la misma forma justificar nuestras decisiones respecto a su desarrollo.

Índice general

1. Estándares

Se intentaron seguir los estándares establecidos por REST.[2] Esto consiste en:

- Aislar al cliente de la implementación.
- Organizar la Api segun los recursos del sistema. En nuestro caso se implementaron 5 controllers. uno por cada recurso: Sessiones, Writers, Folder, Elements y Reports.
- Las urls se deben basar en subjetivos y no verbos.
  Por ejemplo *Localhost:4002//api/Writer/Friend/2*
- Las colecciones de objetos tienen su proprio url.
- Definir las operaciones en base a los metodos de HTTP. Los usados fueron : GET, PUT, POST & DELETE
- Se utilizaron los HTTP Status Code para indicar el estado de la respuesta la usuario

2. Mecanismos de Autenticación

Como mecanismos de autenticación se utilizó algo básico como el login, pero adicionalmente se decidió agregar dos filtros para autenticar y controlar que quienes accedan a los end points sean las personas adecuadas.

2.1. Authorization Filter

El Authorization filter fue creado con la intención de verificar que quienes quieran acceder a un end point tuvieran condiciones necesarias. Las condiciones son :

- Tener un token es decir estar loggeado
- Que ese token pertenezca a un usuario en el sistema
- El rol del usuario sea Administrador

De esta forma los administradores tienen acceso a ciertos end point del sistema que un usuario común no. Permitiéndoles consumir otro tipo de información sin necesidad de estar preguntando en cada action que rol tiene el usuario a ejecutar la misma.

Los requerimientos para usar este filtro son un Rol, y un usuario loggeado. Se deberá usar siempre que se necesite permitir a un conjunto exclusivo de usuarios a realizar una acción.

2.2. Claim Filter

En nuestra solución del problema, establecimos que cada usuario o writer, puede tener "Permisos.º en el sistema ÇustomClaims". Estos se aplican a los recursos que se manejan en TwoDrive, las carpetas y los archivos. Los permisos establecidos fueron los siguientes:

- Read: Se le permite al writer leer o visualizar datos
- Write: Se le permite al writer crear o modificar recursos
- Delete: Se le permite al writer eliminar elementos del sistema
- Share: Se le permite al writer compartir elementos con otros writers amigos del sistema. Compartir implica darle a un amigo alguno de los permisos anteriores sobre un archivo o carpeta que le pertenezca al writer

Cada Acción del sistema se puede identificar con un claim, lo que permitió que se pudiera implementar un filtro, que dado un elemento y el usuario en sesión, que verifique que el usuario tiene los permisos necesarios para realizar esa acción.

Este filtro también incluye roles, ya que el administrador tiene capacidades especiales en cuanto a la lectura y eliminación de archivos, siendo así que puede leer o eliminar cualquier archivo del sistema. Esto implico que para los permisos de Read y Delete, el administrador no se vea afectado por el filtro.

Los requerimientos para este filtro son, un tipo de permiso y un usuario loggeado. El objetivo de este filtro es reducir el acceso y modificación de recursos únicamente aquellos a los que se les otorgaron los permisos adecuados.

3. Codigos de error utilizados

- OK Status code : 200 Se utilizó para indicar que la acción realizada se ejecuto correctamente
- Bad Request Status Code : 400 Se utilizó para indicarle al usuario de un error en la request de la acción.
- Not Found Status Code : 404 Se utilizo para indicarle al usuario que ciertos recursos no fueron encontrados

4. Especificacion de end points

La tabla con los datos de cada endpoint se encuentra al final de este documento ya que es un pdf exportado de excel para que los datos se puedan visualizar mejor.

Bibliografía

[1] Universidad ORT Uruguay. (2013) Documento 302 - Facultad de Ingeniería. [Online]. Available: http://www.ort.edu.uy/fi/pdf/documento302facultaddeingenieria.pdf

[2] Microsoft Azure. (2018) API design. [Online]. Available: https://docs.microsoft.com/en-us/azure/architecture/best-practices/api-design

Resource	Descripcion	Endpoint		Parametros	Responses	Filter	Header
		Verbo	Url				
Session	Permite al usuario loggearse con su nombre de usuario y contraseña, devuleve un token de autorizacion	POST	Url/api/Token	LoginModel: Este modelo contiene el usuario y contraseñas ingresadas	OK 200: Si existe un usuario y contraseña en la base de datos que corresponda un writer. Devuelve el token asignado Bad Request 400: Si no existe ningun usuario con esos datos, se retorna un bad request con un mensaje	Ninguno	Header = "Authotirzation" Value = GUID
Session	Permite al usuario cerrar la session actual. Esto eliminara la session de la base de datos y con ella el token usado	DELETE	Url/api/Token	No recibe parametros	OK 200: Si la session se elimino de manera correcta Bad Request 400: Si hubo algun error al eliminar la session	Ninguno	Header = "Authotirzation" Value = GUID
Writer	Permite crear un nuevo usuario del sistema, con todos sus datos. El usuario creado puede ser administrador o writer	POST	Url/api/Writer	Writer Model: Contiene los datos ingresados por un usuario administrador para crear un nuevo writer	OK 200: Si se creo el writer con exito. Bad Request 400: Hubo un error de validacion con los datos ingresados	[AuthorizeFilter(Role.Administrator)]	Header = "Authotirzation" Value = GUID
Writer	Permite eliminar cualquier usuario de la base datos. Mientras quien realice la request sea administrador	DELETE	Url/api/Writer/id	id: recibe el identificador del usuario a eliminar	OK 200: Si el usuario fue eliminado correctamente Bad Request 400: Si hubo algun problema en el proceso de eliminar el usuario	[AuthorizeFilter(Role.Administrator)]	Header = "Authotirzation" Value = GUID
Writer	Permite obtener un writer dado. Solo puede ser utilizado por administradores	GET	Url/api/Writer/id	id: recibe el identificador del usuario a mostrar	OK 200: Retorna los datos del usuario obtenido Not Found 404: Si no encuentra un usuario para mostrar	[AuthorizeFilter(Role.Administrator)]	Header = "Authotirzation" Value = GUID
Writer	Permite visualizar todos los datos de los usuarios en el sistema. Solo para administradores	GET	Url/api/Writer	No recibe parametros	OK 200: Retorna los datos de todos los writer del sistema Not Found 404: Si no encuentra usuarios en el sistema	[AuthorizeFilter(Role.Administrator)]	Header = "Authotirzation" Value = GUID
Writer	Permite modificar los datos de un usuario dado. Solo disponible para administrador	PUT	Url/api/Writer/Id	Id: identificador del usuario WriterModel: Contiene los datos a modificar del usuario	OK 200: Si se modifico al writer de forma correcta. Bad Request 400: Si hubo algun error de validacion luego de los cambios. O en caso de que el cuerpo de la request llegue null	[AuthorizeFilter(Role.Administrator)]	Header = "Authotirzation" Value = GUID
Writer	Le permite a un usuario loggeado agregar a otro usuario como amigo	PUT	Url/api/Writer/Friend/Id	Id: identifica al usuario al que se quiere agregar como amigo	OK 200: Si se agrega al usuario deseado como amigo. Bad Request 400: Si el writer no se encuentra loggeado, o no se encuentra a su amigo. Tambien el caso de que el usuario ya es amigo del writer a agregar. Retorna en el caso de que falle alguna validacion	Ninguno	Header = "Authotirzation" Value = GUID

Writer	Le permite a un usuario eleminar a otro writer de su lista de amigos	DELETE	Url/api/Writer/Friend/Id	Id: identifica al usuario al que se quiere eliminar como amigo	OK 200: Si se agrega al usuario deseado como amigo. Bad Request 400: Si el writer no se encuentra loggeado, o no se encuentra a su amigo. Tambien el caso de que el usuario no fuera amigo del writer. Retorna en el caso de que falle alguna validacion	Ninguno	Header = "Authotirzation" Value = GUID
Writer	Permite visualizar la lista de todos los amigos de un usuario dado	GET	Url/api/Writer/Friends/Id	Id: identifica al usuario del que se quiere ver sus amigos	OK 200: Se envia la lista de amigos como una lista de WriterModels. O envia mensaje avisando que para ese usuario, no tiene amigos	Ninguno	Header = "Authotirzation" Value = GUID
Folder	Permite crear una carpeta nueva dentro de alguna carpeta padre	POST	Url/api/Folder/id	Id: Es el id que identifica la carpeta padre donde guardar la nueva carpeta. FolderViewModel: Posee los datos para crear la carpeta	OK 200: Si se crea la carpeta sin problemas Bad Request 400: Si se detencta problemas de validacion	[ClaimFilter(ClaimType.Write)]	Header = "Authotirzation" Value = GUID
Folder	Permite eliminar un carpeta dada y todos sus hijos	DELETE	Url/api/Folder/id	id: Es el id que identifica la carpeta a eliminar	OK 200: Se elimina la carpeta y todos sus hijos con exito Bad Request 400: Si al eliminar hubo algun error	[ClaimFilter(ClaimType.Delete)]	Header = "Authotirzation" Value = GUID
Folder	Permite ver los datos de una carpeta dada	GET	Url/api/Folder/id	id: Es el id que identifica la carpeta a visualizar	OK 200: Se retorna los valores de la carpeta encontrada Not Found 404: Si no encuentra ninguna carpeta para mostrar	[ClaimFilter(ClaimType.Read)]	Header = "Authotirzation" Value = GUID
Folder	Permite ver el arbol de hijos de una carpeta dada	GET	Url/api/Folder/id	id: Es el id que identifica la carpeta a mostrar el arbol	OK 200: Muestra el arbol de hijos de una carpeta Not Found 404: Si no encuentra la carpeta para mostrar el hijo	[ClaimFilter(ClaimType.Read)]	Header = "Authotirzation" Value = GUID
Folder	Permite modificar los datos de una carpeta dada.	PUT	Url/api/Folder/id	id: Es el id que identifica la carpeta a modificar FolderModel: Contiene los datos para modificar la carpeta	OK 200: De vuelve la carpeta con los cambios realizados Bad Request 400: Errores de validacion	[ClaimFilter(ClaimType.Write)]	Header = "Authotirzation" Value = GUID
Folder	Permite mover una carperta a otra carpeta mientras ambas pertenezcan la mismo usuario	POST	Url/api/Folder/folderToMoveId/folderDestination	folderToMoveId: Es el identificador de la carpeta que quiero moves. folderDestination: Es el identificador de la carpeta en que se encuentra	OK 200: Si se pudo mover la carpeta de lugar con exito Not Found 404: Si no se encuentra usuario loggeado o no se encuentra el folder a mover o el folder de destino Bad Request 400: Si hay algun error interno		Header = "Authotirzation" Value = GUID
Folder	Permite dejar de compartir una carpeta con un amigo. Mientras el usuario loggeado sea el dueño	DELETE	url/api/Folder/folderToShareId/friendID	folderToShareId: Es el identiifcador de la carpeta que se quiere dejar de compartir friendId: Es el identificador del amigo con el que quiero dejar de compartir	OK 200: Si se pudo dejar de comparti con exito Not Found 404: Si no se encuentra usuario loggeado o no se encuentra el folder, o el amigo a compartir Bad Request 400: Si el usuario con quien se intent dejar compartir no es amigo	[ClaimFilter(ClaimType.Share)]	Header = "Authotirzation" Value = GUID

Folder	Permite compartir una carpeta con un amigo. Mientras el usuario loggeado sea el dueño	PUT	url/api/Folder/folderToShareId/friendID	folderToShareId: Es el identiifcador de la carpeta que se quiere dejar de compartir friendId: Es el identificador del amigo con el que quiero dejar de compartir	OK 200: Si se pudo compatirr la carpeta de lugar con exito Not Found 404: Si no se encuentra usuario loggeado o no se encuentra el folder, o el amigo a compartir Bad Request 400: Si el usuario con quien se intenta compartir no es amigo	[ClaimFilter(ClaimType.Share)]	Header = "Authotirzation" Value = GUID
File	Permite crear un archivo dentro de una carpeta padre	POST	url/api/File/id	id: Es el identificador de la carpeta padre donde crear el archivo	OK 200: Si se crea el archivo sin problemas Not Found 404: Si la carpeta padre no se encuentra Bad Request 400: Si no hay usuario loggeado, si el usuario no es el dueño de la carpeta padre, o si hay un error de validacion	[ClaimFilter(ClaimType.Write)]	Header = "Authotirzation" Value = GUID
File	Permite eliminar un archivo	DELETE	url/api/File/id	id: Es el identificador del archivo a eliminar	OK 200: Si el archivo se elimino de manera correcta Bad Request 400: Si hubo algun error al eliminar el archivo	[ClaimFilter(ClaimType.Delete)]	Header = "Authotirzation" Value = GUID
File	Permite visualizar los datos de un archivo	GET	url/api/File/id	id: Es el identificador del archivo a visualizar	OK 200: Se envian los datos del archivo Not Found 404: No se encontro un archivo	[ClaimFilter(ClaimType.Read)]	Header = "Authotirzation" Value = GUID
File	Permite ver todos los archivos de un usuario, con la opcion de utilizar filtros. Si no se utilizan filtros se retornan todos los datos desordenados. Si el usuario es adminisitrador se muestran todos los archivos del sistema, si no solo los del usuario loggeado	GET	url/api/File (si no se quiere ordernar) url/api/File?name=value&isOrderDescending=bool&isOrderByName=bool&isOrderByCreationDate=bool&isOrderByModificationDate=bool	**Optional parameters:** Name: string name of user to filter by IsOrderDescending: When true the list will be ordered as descending else as asceding IsOrderByName: Indicates where we should order the list by name of writer IsOrderByCreationDate: When true, it will filter and order the list by the created date IsOrderByModificationDate: When true, it will filter and order the list by the modified date	OK 200: retorna la lista filtrada por los parametros seleccionados Not Found 404: Si no encuentra al usuario loggeado, o si no hay ninung archivo en el sistema		Header = "Authotirzation" Value = GUID
File	Permite modificar los datos de un archivo dado	PUT	url/api/File/id	id: Es el identificador del archivo que se quiere modificar	OK 200: File was updated correclty Bad Request 400: si hubo algun error de validacion	[ClaimFilter(ClaimType.Write)]	Header = "Authotirzation" Value = GUID

File	Permite compartir un archivo con un amigo. Mientras el usuario loggeado sea el dueño	PUT	url/api/File/id/friendId	Id: es el identificador del archivo a compartir. friendId: Es el id del writer al que le quiero comparti	OK 200: Si se pudo compatirr la archivo de lugar con exito Not Found 404: Si no se encuentra usuario loggeado o no se encuentra el la carepeta, o el amigo a compartir Bad Request 400: Si el usuario con quien se intenta compartir no es amigo	[ClaimFilter(ClaimType.Share)]	Header = "Authotirzation" Value = GUID
File	Permite dejar de compartir una archivo con un amigo. Mientras el usuario loggeado sea el dueño	DELETE	url/api/File/id/friendId	Id: es el identificador del archivo a compartir. friendId: Es el id del writer al que le quiero comparti	OK 200: Si se pudo dejar de comparti con exito Not Found 404: Si no se encuentra usuario loggeado o no se encuentra el folder, o el amigo a compartir Bad Request 400: Si el usuario con quien se intent dejar compartir no es amigo	[ClaimFilter(ClaimType.Share)]	Header = "Authotirzation" Value = GUID
File	Permite mover un archivo de una carpeta a otra	POST	url/api/File/folderToMoveId /folderDestionation	Id: es el identificador del archivo a mover. folderDestination: Es el id de la carpeta destino	OK 200: Se cambio de lugar la carpeta de forma correcta Not Found 404: Si el writer loggeado o el parent folder o el archivo a mover no se encuentra Bad Request 400: Si el usuario loggeado no es el dueño de la carpeta o el archvio	[ClaimFilter(ClaimType.Write)]	Header = "Authotirzation" Value = GUID
File	Retorna la cantidad de modificaciones realizadas por un usuario a sus archivos	GET	url/api/Report	start: Inicio del periodo de tiempo en el que buscar mofícaciones end: fin del periodo de tiempo en el que buscar modificaciones	OK 200: retorna una lista de usuarios y su cantidad de modificaciones	[AuthorizeFilter(Role.Administrator)]	Header = "Authotirzation" Value = GUID
File	Retorna los TOP 10 usuarios con mayor cantidad de archivos en el sistema	GET	url/api/Report	No recibe parametros	OK 200: retorna lista de usuarios y su cantiad de archivos	[AuthorizeFilter(Role.Administrator)]	Header = "Authotirzation" Value = GUID