Servicio de directorio

Revisión 1

1. Introducción

Se deberá implementar una API REST que permita la creación de estructuras similares a un sistema de ficheros. El lenguaje de implementación es libre, aunque se recomienda utilizar Python 3. Además, deberá crearse una librería y un cliente en Python 3 para poder utilizar el servicio sin necesidad de conocer la propia API REST.

2. Implementación del servicio

El servicio se apoyará en una base de datos para almacenar todos los directorios que se creen en el sistema. El alumno puede utilizar la tecnología de base de datos que desee, aunque se recomienda utilizar alguna de las disponibles en la biblioteca estándar de Python 3, como por ejemplo SQL Lite. En cualquier caso, la base de datos deberá estar almacenada localmente de manera que se pueda indicar mediante un *argumento opcional* la ruta de la misma.

La base de datos almacenará información general del árbol de directorios: el identificador del directorio raíz. Además, contendrá una lista de directorios, cada uno de ellos tendrá un identificador único, que puede ser un *UUID* o un token *url-safe*. Por cada directorio se almacenará: un nombre identificativo del directorio, el identificador del directorio padre, una lista de directorios hijos y una lista de tuplas con la forma (“nombre”, <URL>). Ambas listas pueden ser vacías. Además, cada directorio tendrá otras dos listas “*readable_by*” y “*writable_by*”, que contendrán los usuarios que, respectivamente, pueden leer y escribir en dicho directorio.

Se deberá crear una clase que implemente la persistencia de datos, es decir, que abstraiga de todos los detalles de la base de datos, de manera que tenga métodos similares a:

- Establecer un directorio raíz: como parte de la inicialización de la base de datos deberá crearse un directorio vacío, con el nombre por defecto “/”. Además, las listas “readable_by” y “writable_by” a [“admin”].
- Crear un directorio: dado el identificador de un directorio, un nombre de usuario y un nombre de directorio, se creará un nuevo directorio, estableciendo el identificador dado como directorio padre. Se inicializarán las listas “readable_by” y “writable_by” a [“<nombre de usuario>”]. El directorio tomará como nombre el valor pasado como <nombre de directorio>. Si el directorio padre ya tuviera un hijo con el mismo nombre de directorio, la operación generaría una excepción.
- Borrar un directorio: dado el identificador de un directorio, un nombre de directorio y un nombre de usuario; si el nombre de usuario aparece en “writable_by”, se procederá a eliminar el directorio hijo del padre dado por el identificador cuyo nombre de directorio coincida con <nombre de directorio>. Si el nombre de usuario no aparece en la lista “writable_by” se lanzará una excepción. También se lanzará si no existiera un hijo así con nombre <nombre de directorio>.
- Añadir permisos de lectura: dado el identificador de un directorio, un nombre de usuario “propietario” y un nombre de usuario “invitado”; si el usuario “propietario” está en la lista “writable_by” del directorio, se añadirá el nombre de usuario “invitado” a la lista

"readable_by" del directorio. Generará una excepción si no existe dicho directorio y otra en caso de que el usuario "propietario" no esté en la lista "writable_by".

- Quitar permisos de lectura: dado el identificador de un directorio, un nombre de usuario "propietario" y un nombre de usuario "invitado"; si el usuario "propietario" está en la lista "writable_by" del directorio, se eliminará el usuario "invitado" de la lista "readable_by" del directorio. Se generará una excepción si el usuario "propietario" no está en la lista "writable_by". También se generará otra excepción si el identificador del directorio no existe o si el usuario "invitado" no está en la lista "readable_by". Si se intenta quitar permisos a "admin" también se rechará con una excepción.
- Añadir permisos de escritura: dado el identificador de un directorio, un nombre de usuario "propietario" y un nombre de usuario "invitado"; si el usuario "propietario" está en la lista "writable_by" del directorio, se añadirá el nombre de usuario "invitado" a la lista "writable_by" del directorio. Generará una excepción si no existe dicho directorio y otra en caso de que el usuario "propietario" no esté en la lista "writable_by".
- Quitar permisos de escritura: dado el identificador de un directorio, un nombre de usuario "propietario" y un nombre de usuario "invitado"; si el usuario "propietario" está en la lista "writable_by" del directorio, se eliminará el usuario "invitado" de la lista "writable_by" del directorio. Se generará una excepción si el usuario "propietario" no está en la lista "writable_by". También se generará otra excepción si el identificador del directorio no existe o si el usuario "invitado" no está en la lista "writable_by". Si se intenta quitar permisos a "admin" también se rechará con una excepción.
- Añadir fichero: dado el identificador de un directorio, un nombre de usuario, un nombre identificativo y una URL; si el nombre de usuario está en la lista "writable_by" del directorio, se añadirá una nueva tupla a la lista de ficheros de la forma: ("nombre identificativo", URL). Si ya existiese una tupla con dicho "nombre identificativo" o un directorio hijo con el mismo nombre, se rechazará la ejecución generando una excepción. También se lanzará una excepción si el usuario no está en la lista "writable_by".
- Borrar fichero: dado el identificador de un directorio, un nombre de usuario y un nombre identificativo; si el nombre de usuario está en la lista "writable_by" del directorio y en la lista de ficheros existe una tupla cuyo primer término sea "nombre identificativo", se eliminará de la lista dicha tupla. Si no existiese dicha tupla, se lanzará una excepción. También se lanzará si el usuario no está en la lista de "writable_by".

Finalmente se creará una API REST que utilizará dicha clase para implementar su funcionalidad. Los métodos y recursos que la API debe exponer son los siguientes:

- /v1/directory/root (GET): Se acepta la petición si incluye la cabecera de administrador o un user token cuyo dueño esté en la lista "readable_by". En ese caso se devolverá un código 200 y un JSON con la siguiente información:

  {"dir_id": "<identificador directorio>", "childs": [<nombres de directorios hijos>]}

  En caso contrario, se devolverá el error 401.

- /v1/directory/<dir_id>/<nombre_hijo> (GET): Se acepta la petición si incluye la cabecera de administrador o un user token cuyo dueño esté en la lista "readable_by". En ese caso se devolverá un código 200 y un JSON con la siguiente información:

  {"childs_ids": [<identificadores de directorios hijos>]}

Si no se incluyen cabeceras administrativas ni de autenticación de usuario, se devolverá el error 401. Si no existe un directorio con dicho identificador o si el directorio no tiene un hijo con ese nombre, un 404.

- /v1/directory/<dir_id>/<nombre_hijo> (PUT): Se acepta la petición si incluye la cabecera de administrador o un user token cuyo dueño esté en la lista "writable_by". En ese caso se devolverá un código 200 y se creará un nuevo directorio. Se devolverá un JSON con la siguiente información:

  {"dir_id": <identificador de directorio>}

  En caso contrario, se devolverá el error 401. Si no existe un directorio con dicho identificador, un error 404. Si ya existe un directorio hijo con dicho nombre, un código 409.

- /v1/directory/<dir_id>/<nombre_hijo> (DELETE): Se acepta la petición si incluye la cabecera de administrador o un user token cuyo dueño esté en la lista "writable_by". En ese caso se devolverá un código 204 y se eliminará el directorio hijo. Si no existen las cabeceras o no son válidas, se devolverá el error 401. Si no existe un directorio con dicho identificador o existe, pero no tiene un hijo con dicho nombre, un error 404.
- /v1/files/<dir_id> (GET): Se acepta la petición si incluye la cabecera de administrador o un user token cuyo dueño esté en la lista "readable_by". En ese caso se devolverá un código 200 y un JSON con la siguiente información:

  {"files": [<nombres de ficheros]}

  Si no se incluyen cabeceras o no son válidas, se devolverá error 401. Si no existe un directorio con ese identificador, un 404.

- /v1/files/<dir_id>/<filename> (GET): Se acepta la petición si incluye la cabecera de administrador o un user token cuyo dueño esté en la lista "readable_by". En ese caso se devolverá un código 200 y la URL asociada al fichero. Si no se incluyen cabeceras o no son válidas, se devolverá error 401. Si no existe un directorio con ese identificador, un 404.
- /v1/files/<dir_id>/<filename> (PUT): Se acepta la petición si incluye la cabecera de administrador o un user token cuyo dueño esté en la lista "writable_by". En ese caso se devolverá un código 200 y la URL asociada al fichero. Si no se incluyen cabeceras o no son válidas, se devolverá error 401. Si no existe un directorio con ese identificador, un 404. Para que la petición sea aceptada, también deberá acompañar un JSON con la siguiente información:

  {"URL": "<URL asociada al fichero>"}

  En caso contrario se devolverá un código 400.

- /v1/files/<dir_id>/<filename> (DELETE): Se acepta la petición si incluye la cabecera de administrador o un user token cuyo dueño esté en la lista "writable_by". En ese caso se devolverá un código 204 y se eliminará la tupla asociada al fichero en el directorio dado. Si no se incluyen cabeceras o no son válidas, se devolverá error 401. Si no existe un directorio con ese identificador o existe, pero no tiene un fichero con ese nombre, un 404.

2.1 Cabeceras

Las siguientes cabeceras tienen que ser reconocidas por el servicio:

- admin-token: <token>

- user-token: <token>

3. El servidor

El servicio de autenticación se implementará mediante un servidor, que aceptará las siguientes opciones:

- “-a <token>” o “--admin <token>”: establece un token de administración. Si no se establece, se generará uno automáticamente y se mostrará por salida estándar.
- “-p <puerto> o “--port” <puerto>”: establece un puerto de escucha, si no se establece por defecto será el 3002.
- “-l <dirección>” o “--listening <dirección>”: establece una dirección de escucha, por defecto se usará “0.0.0.0”.
- “-d <ruta>” o “--db <ruta>”: establece la ruta de la base de datos, por defecto se usará el *current working directory* o CWD.

Requiere un argumento obligatorio, la URL de la API del servicio de autenticación. Antes de iniciar el servidor, verificará que el token administrativo establecido es válido mediante la URL. En caso de token inválido, el servició cancelará la ejecución y saldrá con un código de error 1.

4. Pruebas

Se crearán una serie de pruebas unitarias que deberán poder ejecutarse de forma automática dentro del entorno virtual apropiado. Por tanto, el proyecto incluirá un archivo *requirements.txt* con los datos necesarios para crear ese entorno. Las pruebas deberán tener las siguientes características:

- Las clases de la capa de negocio y persistencia deberán probarse con una cobertura de al menos un 75%.
- El código del servidor y la API deberá estar cubierto en al menos un 50%.