SharePoint & TagMyDoc (Convoflo) sync

Sidenote: we are in the process of renaming TagMyDoc to Convoflo in the next month. You may see references to TagMyDoc and/or Convoflo, but they are the same.

The What

One of our clients (LECSOR) needs to have their SharePoint Online drive synced with our in-house application filesystem. Any addition/changes/deletion of files and folders on the SharePoint drive needs to happen on our platform and any addition/changes/deletion of files and folders on TagMyDoc needs to happen on SharePoint.

The Why

TagMyDoc is a client-portal for our customers. They use the application to better communicate with their clients such as sending files, messages, request e-signatures, request payments and request files. LECSOR wants to use TagMyDoc to communicate with their clients through SharePoint. For example, one of their clients uploads a file on TagMyDoc, that same file needs to be sent to SharePoint in the correct folder as well.

The How

We need a PHP/Laravel library that our application can talk to for sending files, creating folders, updating file versions, deleting items on SharePoint. That same library should be able to configure webhooks or registering deltas to track changes made on SharePoint. The Laravel implementation should use queues to execute jobs with fail safes, retries and logging.

The library should be able to:

- Connect to Microsoft's Graph API
- Maintain / renew authentication tokens automatically via Laravel's task scheduling.
- Connect to the drive on the tenant.
- Upload a file on the SharePoint drive. Return its ID and properties for TagMyDoc to store on our end. If there's a conflict, do not replace the file. Simply add (1) to its name.
- Rename an item on the SharePoint drive.
- Create a new version of a file on the SharePoint drive. Return its ID and properties for TagMyDoc to store on our end.
- Create a folder on the SharePoint drive. Return its ID and properties for TagMyDoc to store on our end.
- Delete an item on the SharePoint drive.
- Download a file from the SharePoint drive.
- Track changes of the SharePoint via webhooks sent to TagMyDoc or poll the delta from the drive to see what items have changed. Make sure that when we add files from TagMyDoc to SharePoint, these files are ignored when they appear in webhooks/delta polling.
- Have a clear list of what needs to be added, modified and deleted from that API call, so TagMyDoc can apply these changes to its system. Upon a change, the library should have a callback to its listeners where data is passed as follows:
    - For new items, include:
        - the name
        - the type (file or folder)
        - the contents (if it's a file)
        - the location (Sharepoint folder ID)
    - For changed files, include:

    - the Sharepoint item ID
    - the new name (if changed)
    - the new contents (if changed)
    - the new location (if moved)
  - For deleted files, include:
    - the Sharepoint item ID
    - the type (file or folder)
  - For example: *Create a file named test.pdf with these binary contents in the folder with this ID.* And then, the client code (TagMyDoc's side) will execute this. TagMyDoc's side doesn't need to be in the library, we can do it on our end, but what we really need is an indication of what needs to be executed.
- If webhooks are used, make sure to renew their subscriptions and not let them expire. Probably use Laravel's task scheduling or listen for Webhooks lifecycle notifications.

A first draft of the library is published at: *https://github.com/tagmydoc/Sharepoint-SDK*. This SDK is built on top of Saloon. A basic implement is done as:

```
use Saloon\Http\Auth\AccessTokenAuthenticator;
use TagMyDoc\SharePoint\SharePointClient;

$auth = AccessTokenAuthenticator::unserialize(get_auth_token_from_storage());

$client = new SharePointClient('client_id', 'client_secret', 'tenant_id');
$client->authenticate($auth);

$response = $client
     ->drive('drive_id')
     ->getItemByPath('99-999V99 Syndicat Test 01/Assurances (EC-A)/Contrats');

$item = $response->json();
```