berlinger
feel safe

Berlinger & Co. AG
Mitteldorfstrasse 2
9608 Ganterschwil
Switzerland

Tel. +41 71 982 88 11
Fax +41 71 982 88 39
info@berlinger.com
www.berlinger.com

The Goal

We want to create an API capable to do simple operations around images and images metadata. The customer should be able to bulk load metadata (like image description, location) and then the system should be able to deliver either metadata, or the images themselves (acting as a local cache system).

Client Requirements

As a client I want to be able to upload my data in a csv form to a specific URL. An example of this is provided along with this file. The structure remains always the same, but the data can be inconsistent.

I also want to be able to retrieve all the metadata for the already uploaded data, and also an individual picture based on the id.

Technical description

The API should be built to support three different operations:

- Upload of the data, by POSTing a CSV file
- Retrieving the metadata for ALL the already uploaded pictures (JSON response preferred)
- Retrieving one single image based on his unique identifier

The saved data structure will contain at least these fields (but it's not limited to)

- picture_title – unique element, case insensitive
- picture_url (mandatory)
- picture_description (optional)

CSV delimiter is always a vertical bar |. CSV don't have an imposed upper limit in terms of records/size, but large ones can be expected. For simplicity we're assuming that the server is able to handle big POST data. CSV has always a header which should be ignored.

If a consecutive load of the data contains already existing elements, the data will be overwritten with the newest version (for example picture_url will be updated, or picture_description).

Because the pictures might not be always available at the original locations (URLs are not reliable), it's required to create a local copy for those. The API call to retrieve one single image should use this local file and not the remote one.

Remarks

The API can be built using any framework or tool of choice (obviously PHP based).

The code should be available to us for review using a GitHub/BitBucket repository (or a similar account). No zip files accepted.

A proof of a working API should be provided to us.

The API should consider the cases where the remote pictures are not available anymore, or downloading them locally takes a lot of time.

Any DB system can be used in the backend.

Using a Dockerfile for the setup of the project/environment is a nice to have, but not required.

A separation of different environments in the code is nice to see (e.g. development/production at least) but not required.