API design

Principles of the API design:

1. All documented routes should be appended to https://ddd.cjvt.si/api/.
2. All the routes are available as POST calls, even if they do not result in changes in the database, because:
   - some routes will have non-trivial input parameters (structured data, arbitrary strings), which are difficult and clunky to encode as path parameters, and expecting request body parameters in GET calls can be problematic and misleading
   - we can have short clear URLS for all routes, and there are limits on URL length in some contexts.
3. Some routes also have a GET counterpart, which behave the same way as the POST call but do not allow for response body parameters (default values are used instead).
4. All request parameters are provided as JSON request body parameters, except for the object's id, which is used to identify a given object and provided as an obligatory path parameter for certain types of calls (e.g., retrieve).
5. The following HTTP response codes are used:
   - **200**: for most successful requests
   - **201**: for successful get-or-create requests where no matching object was found and a new one was created
   - **400**: an error occurred due to invalid or unexpected request parameters or combinations
   - **401**: authorisation denied (suitable credentials are needed for routes which write to the database)
   - **404**: objects were not found for the value (usually id) provided
   - **501**: the specifications for this route are designed but it has not yet been implemented
6. Each route falls under a particular type of operation identified with a particular verb as the first part of the route. The verbs include:
   - **retrieve**: return data for a given object
   - **search**: return all the objects which match the set of search parameters
   - **export**: return all object ids by minimal filter and with minimal data OR return objects en masse for performance-sensitive data, using cursor pagination
   - **get-or-create**: get the object matching the parameters provided, creating one if necessary, along with any other missing objects it depends on
   - **update**: update the properties of a given object based on the parameters provided
   - **delete**: delete a given object
   - **attach**: attach the given object to a particular resource, if not yet attached
   - **detach**: detach the given object from a particular resource, if attached
   - **process**: process the input data with an appropriate independent tool (e.g., the CLASSLA NLP library)

7. If the operation verb has a "-batch" suffix, it differs from its non-batch counterpart as follows:
   - users can make 1 API call instead of N API calls for N items
   - the input data should be a list, with each element in the format expected by the non-batch route
   - the output data is a list, with each element corresponding to the element at the same position in the input, where each element has three fields:
     - status: the HTTP response code that would be used if the element was processed in a non-batch call
     - message: a message describing the results of the operation (e.g., whether an object was found or created, or the cause of the warning or error)
     - data: the output data (for successful calls), in the same format as non-batch output
8. Routes which do not change data in the database (retrieve, search, export, process) are publicly available. Routes which may result in changes in the database (get-or-create, update, delete, attach, detach) require authentication credentials.

Revision #9
Created 20 June 2022 14:04:23 by Cyprian Laskowski
Updated 3 April 2025 07:59:45 by Cyprian Laskowski