Fragalysis - #651 Fix data download

Author: Duncan Peacock
Version: v2.0
Date: 15/12/2021

Date	Who	Update
23/09	Duncan	Added estimate
05/10	Duncan	Document approved in meeting 05/10/2021
15/12	Duncan	Add appendix A with design change from ticket: Data download improvements #690

Introduction

Issue #651

Hard to get the coordinates *that you want*.

Minimal:

- Currently visible structures
- Structures with tag

Optional:

- Download builder dialog - probably generate temporary (or hidden) tags
    - (Maybe redundant, because the search dialog would stand in -
- Fix search tools (epic) #654)

Always by specific endpoint (fully restful), with toggles to configure (like for the direct view URL)

- (might include setting up a restful endpoint to create tags)
- needs backend work for assembling zipfile on FE request

Assumption

In this analysis, it is assumed that the user wants a subset of the information in the current download file (and not yet stuff from other tables). Once the basic process has been designed it could then be extended to pull, for example, other information from the Fragalysis database.

As-Is (Current)

When a target is uploaded, it is in the form of a zip file. This is unzipped and processed in the backend to load the Fragalysis tables. Some files (pdb/map/bound) are copied into common folders and direct links to these files are stored in the *Protein* table - but not all files included are directly referenced. After the file is processed an archive is created and it is this that is downloaded in its entirety when the “download structures” link is pressed in Fragalysis. The archive is also available under the target name in the media directory.

The available download options will be provided by the direct links and information in *Protein* as well as linked information provided in the *Target* and *Molecule* tables .

Protein direct links to files:

The Protein table has direct links to files in the media directory - so from a list of Fragalysis protein codes/required types for a target, a file containing the files that these links reference could be assembled and returned in a zip file.

pdb_info : varchar-255
bound_info : varchar-255
cif_info : varchar-255
mtz_info : varchar-255
map_info : varchar-255
diff_info : varchar-255
event_info : varchar-255
sigmaa_info : varchar-255

Metadata:

In addition to the individual file links there is also a link to the metadata.csv file that was uploaded with the target set in table *Target*.

If *metadata* is requested, this file should also be included in the download, but as is, in its entirety. No attempt at this stage will be made to extract information for individual proteins - but the code will be factored so that this can be added later.

Note that the *target_id* is available on the protein table.

SDF, Smiles and Transformation matrix *files*:

When a protein is uploaded with the target loader, a related *Molecule* record is created if ligand information exists (Molecule can be read using it’s foreign key: *prot_id*) with contents

extracted from an SDF file. Furthermore, the Smiles is also extracted from the sdf mol information and added to Molecule. The SDF and Smiles are desired to be part of the download.

The Transformation Matrix is not yet loaded into Fragalysis - but this is also desired to be in the download.

To simplify the solution, it has been agreed that links to the SDF file and Transformation Matrix (a CSV to be added to the Target upload) will be explicitly added to *viewer_protein* so that they can be requested in the same way as the other file types.

If <smiles> is requested, then the *smiles* information in the, potentially multiple following #652, molecule/component records attached to the protein will be put in a text file created for each molecule.

Tags

A tag can currently be created attached to a session project or a molecule. These have an additional_info field that is simply json. If we can work out a structure that is understandable to the backend, then maybe an extract template could be stored within the additional info and referred to by the tag.

Proposed Solution

Database Changes

Viewer_protein will be extended with the following fields:

sdf_info : varchar-255
trans_matrix_info : varchar-255

Target Uploader

The sdf file will now be explicitly saved in a new media subvolume ‘sdfs’ (as well as with the target as it is now) and have a link created in the *sdf_info* field in the protein record.

If a transformation matrix file is supplied in the zip file, it will be explicitly saved in a new media subvolume (as well as with the target) and have a link created in the *trans_matrix_info* field in the protein record.

Download API

A new API will be developed for this download - as a rough model I have based this on the existing *api/dicttocsv* API that was developed recently for Fragalysis.

The API will consist of:

- a POST form containing the requested fields/files. This will compile the zip file on the media server and return a link to it.
- A GET request that can be used to download the compiled zip file from the media server using the link.

Structure of the zip file

The zip file will be structured as follows:

- Each type of file will have it's own folder in the structure and
- Each filename will generally start with the protein code. File names should be unique as they are made unique on the upload. I suggest the smiles text file have a suffix of .smi.
- The metadata.csv file (if requested) will be placed on the root folder (named metadata.csv).

api/download_structures

POST ../api/download_structures/

- Construct a zip file based on the requested parameters and return a temporary download link for the file.

Params:

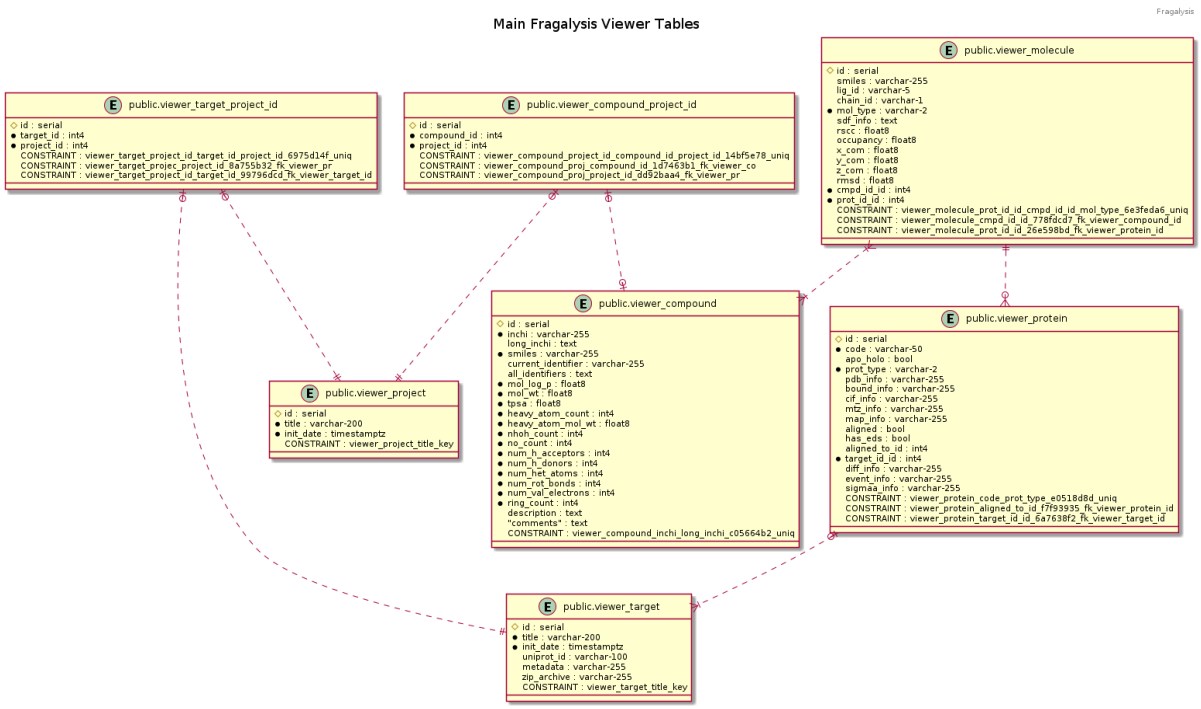
- Target name (string)
- List of *Protein.codes* (only the first part up to the colon will be used)
- pdb_info : boolean - when true, adds the file linked to in the *Protein.pdb_info* field to the zip file in the relevant folder (if the link works and file is present)
- bound_info : boolean - ditto for *Protein.bound_info*
- cif_info : boolean - ditto for *Protein.cif_info*
- mtz_info : boolean - ditto for *Protein.mtz_info*
- map_info : boolean - ditto for *Protein.map_info*
- diff_info : boolean - ditto for *Protein.diff_info*
- event_info : boolean - ditto for *Protein.event_info*
- sigmaa_info : boolean - ditto for *Protein.sigmaa_info*
- sdf_info : boolean - ditto for *Protein.sdf_info*
- transformation_matrix_info : boolean - ditto for *Protein.trans_matrix_info*
- metadata_info : boolean - add the metadata.csv file to the root folder.
- smiles_info : boolean - Create a text file containing the information in *Molecule_compound.smiles* and add to the archive in the protein folder with filename <protein-code>.smi

If any dead links are found (i.e. links that are not spaces, but where the file does not exist), an *error.csv* file will be created with a list of Protein code/Dead link, so that these can be investigated.

GET ../api/download_structures/<download link>

- Returns the file from the download location.

Data Model



Two new fields will be added to the *viewer_protein* table.

Estimates

Estimate includes API tests and documentation updates where appropriate.

I estimate the work described in this document to take around 5 days to complete.

Appendix A - Data download improvements #690

Updated design - Based on the discussions on Tuesday 8th December about:

- retaining the old zip file contents (what we've called _static_ files) vs regenerating the zip file each time (what we've called _dynamic_ files).
- Stopping unnecessary recreation of zip files
- Tidying up old data to reclaim disk space.

A new table will be created in the backend to manage all files that have been created by the download_structures calls as follows:

downloadlinks:

- - file_url (unique - returned to the user in the POST call)
- - user
- - target
- - proteins list
- - search booleans
- - static_link
- - zip_contents
- - zip_file boolean
- - keep_zip_until *(one hour from request)*

The download_structures POST will change as follows:

The API will have two new fields added to it:

- *static_link* (required boolean True = static file, False = dynamic file) and
- *file_url* (this only applies for static files)

Each time the POST is called to create a zip file:

Dynamic links

- Processing will check whether it has already been created by checking the search fields/protein list and static_link flag in *downloadlinks*
- If a file with the search already exists then the keep_zip_until is reset and the existing file_url is returned.
- If a record is found for the search, but the zip_file field is False (see below), then the zip file is recreated, the keep_until is reset and the existing file_url is returned
- If the static_link flag is set in the post request, then the contents of the zip file will be stored in *downloadlinks* and the static_link flag will be set to True.
- If no records exist then the zip file is created (if static_link is True, this will include the zip contents).

Static links

- Processing will check whether it has already been created by whether a static link with the file_url exists in *downloadlinks*.

- If a file with the search already exists then the keep_zip_until is reset and the existing file_url is returned.
- If a record is found for the search, but the zip_file field is False (see below), then the zip file is recreated _from the zip_contents_, the keep_until is reset and the existing file_url is returned
- If no records exist then the dynamic search code is called.

POST will also manage the records in *downloadlinks*:
- Each time it is called it will delete any zip files and set the zip_file to False where the _keep_zip_until_ is less than now.

The GET will be changed to check that is only returns files set up in *downloadlinks*

The changes will mean that:
- Frequently accessed dynamic files will available in the media directory
- Unused files will be deleted, but will be recreated either from the search fields using the latest data from the target or from the previous contents.