**Minimal ADC server (mADC) - Specification**

**Summary**

The minimal ADC will implement a subset of the [AIRR Data Commons API](https://docs.airr-community.org/en/latest/api/adc_api.html) that is sufficient to allow the available repertoires to be interrogated and a specified repertoire to be downloaded.

AIRR Data Commons API Support

The following API Endpoints will be supported:

|  |  |  |  |
| --- | --- | --- | --- |
| **Endpoint** | **Type** | **HTTP** | **Description** |
| /v1 | Service status | GET | Returns success if API service is running. |
| /v1/info | Service information | GET | Upon success, returns service information such as name, version, etc. |
| /v1/repertoire/{repertoire\_id} | Retrieve a repertoire given its repertoire\_id | GET | Upon success, returns the Repertoire information in JSON according to the [Repertoire schema](https://docs.airr-community.org/en/latest/datarep/metadata.html#repertoireschema). |
| /v1/repertoire | Query repertoires | POST | Upon success, returns a list of Repertoires in JSON according to the [Repertoire schema](https://docs.airr-community.org/en/latest/datarep/metadata.html#repertoireschema). |
| /v1/rearrangement | Query rearrangements | POST | Upon success, returns a list of Rearrangements in JSON or AIRR TSV format according to the [Rearrangement schema](https://docs.airr-community.org/en/latest/datarep/rearrangements.html#rearrangementschema). |

**Implementation**

REST API implementation

The mADC will be written in Python 3.10, [Flask 2.3.3](https://flask.palletsprojects.com/en/2.3.x/) and [Flask-RESTX](https://flask-restx.readthedocs.io/en/latest/) (Flask 3 was released in September 2023. As there aren’t any compelling new features for this application, it’s a little too early in its lifecycle to consider adoption).

[This article](https://towardsdatascience.com/creating-restful-apis-using-flask-and-python-655bad51b24) provides a simple introduction to REST APIs in Flask.

[This article](https://medium.com/analytics-vidhya/swagger-ui-dashboard-with-flask-restplus-api-7461b3a9a2c8) is slightly more in-depth and explains the use of [Flask-RESTX](https://flask-restx.readthedocs.io/en/latest/).

[This article](https://www.google.com/search?client=firefox-b-d&q=13+tips+and+tehniques+for+modern+flask+apps) provides an update on some more modern approaches in Flask and is well worth reading once the basics are understood.

The Flask application should embody the following features:

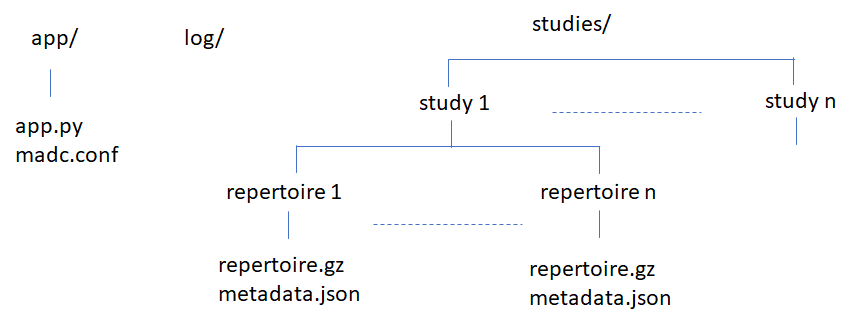
* Logging of API requests using app.logger
* Implementation of requests in separate files using blueprints (put the first two endpoints in service.py, and the second two in repertoire.py)
* Swagger UI, auto-generated by Flask-RESTX
* Configuration of host, file paths, info required by the /v1/info endpoint, etc via app.config.from\_file

Implementation of these features is described in the articles listed above.

Hosting

At this point, the aim is to produce code that will run on a developer’s machine. Once the code is complete, it will be encapsulated in Docker and can use the front-end infrastructure shared by OGRDB and VDJbase.

File Structure



The location of the log and studies directories should be specified in madc.conf..

API restriction

The repertoire and rearrangement endpoints, as specified in the ADC API documentation, allow the response to be filtered using the filters keyword. The mADC will only provide minimal filtering as described below.

*Repertoire endpoint*

The repertoire endpoint *may* contain a filter specification identifying a study id as below, and *may* contain a fields specification:

*repertoire*

{

"filters": {

"op": "=",

"content":

{

"field": "study\_id",

"value": <study\_id>

}

},

"fields": [

"repertoire\_id",

"subject.species.id",

"sample.pcr\_target.pcr\_target\_locus"

]

}

Requests that specify other filter parameters or options will be rejected as invalid. If the filter is not specified, the request will return metadata for all repertoires. If the fields specification is not specified, the request will return all available fields.

*Rearrangement endpoint*

Rearrangement requests *must* contain a filter specification identifying the repertoires to be provided, as below. This filter must be present in rearrangement requests. Requests that are missing this filter, or that specify other filter parameters or options, will be rejected as invalid.

{

"filters": {

"op": "=",

"content":

{

"field": "repertoire\_id",

"value": <repertoire\_id>

}

},

"format": "tsv"

}

Business Logic

On startup, the application should scan the studies directory and create an in-memory data structure enumerating the repertoires and paths to their files. This will be used to satisfy the requests, in conjunction with the files themselves. It can be assumed that the studies file tree will not change during execution - if repertoires are added or removed, the app will be restarted.

Additional Utilities

Metadata.json will contain metadata in compliance with the MiAIRR schema. A utility is required to create this metadata from study- and subject- level metadata held in a simpler and less repetitive format (probably a csv file).

A utility is required that will validate a metadata file to confirm that it conforms to the schema.

The AIRR Python Reference Library will be useful for these two utilities.It is guaranteed to write a metadata file that conforms to the schema, and validates against the schema when data is read.