Avaandmed-py

Overview

Avaandmed-py is an open-source **Python** project designed and developed for **Estonian Open Data**(Avaandmed) portal as part of bachelor thesis.

This library provides convenient way to programmatically utilize public Open Data portal's **API**. Prior to that all the logic to for API usage had to be implemented by users individually which would increase overall development time and decrease developer experience.

Library is published on the **Python Package Index** (PYPI) registry and can be installed with common **pip install** command. Project itself is available on the **GitHub**.

Capabilities

Library enables to programmatically interact with Open Data resources. Meaning, you can perform following actions: **read**, **update**, **delete**, **create**, **download**, **upload** resources (for some actions permissions granted by owner of a resource is required).

Furthermore, you can automate such actions as granting or declining access, policy violations and much more.

Future Considerations

• Potential bachelor **thesis** for future students. For example, code optimization. Perform benchmarking before code optimization and after and provide results.

• Build **community** around to facilitate maintenance and developing of new features.

• Make library **officially** supported by Open Data portal(?).

Author:

Mihhail Matišinets,
Tartu University Narva College,
IT Systems Development (3rd year)

Supervisor:

Andre Sääsk, MSc Physics

GitHub:

https://github.com/mihhail-m/avaandmed-py

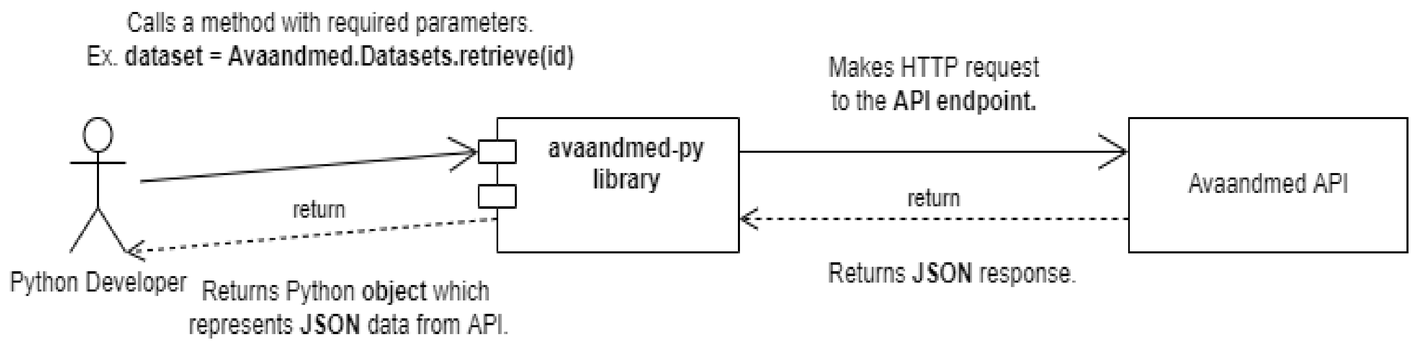
PYPI:

https://pypi.org/project/avaandmed/

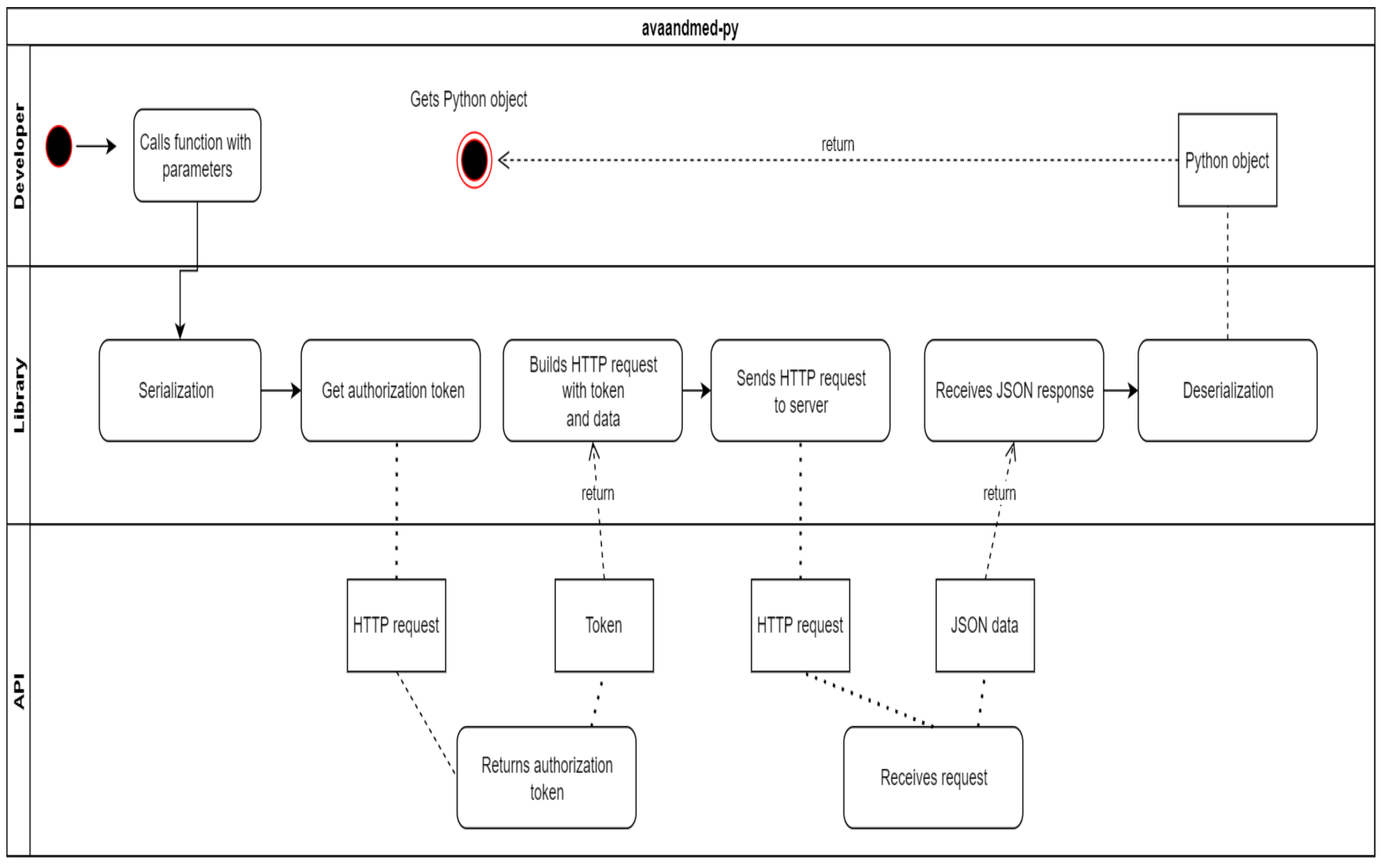
Open Data:

https://avaandmed.eesti.ee/

High Level Overview



Library Level Overview



Usage Examples

```python
key_id = 'key_id_value'
token = 'token_value'
client = Avaandmed(api_token=token, key_id=key_id)
```

```python
me: Me = client.users.me
# List all user's dataset
my_dataset: UserDataset = me.dataset

# Retrieves specific dataset
specific_ds: Dataset = me.get_by_id('dataset_id')
# Delete specific dataset
my_dataset.delete('dataset_id')
# Update dataset
# To update dataset you need to provide a dictionary with fields you wish to update.
# Fields should be camelCased as per official Swagger document.
updateParams = {
    "maintainerPhone": "37255511122",
    "maintenerEmail": "maintainer@mail.com"
}
my_dataset.update(updateParams)

# Create dataset metadata
# To create dataset's metadata you can use DatasetMetadata model.
# For user's convinience parameters can be provided in camelCase manner to match
# naming convention of the JSON that is used by Open Data.
# There are fields that are required to be filled and some don't. Optional fields can be left
# out.
metadata: DatasetMetadata = DatasetMetadata(nameEn="new dataset", nameEt="new dataset"...)
my_dataset.create_dataset_metadata(metadata)
# You can consider specific privacy violations for your dataset
my_dataset.consider_privacy_violation('id')

# Or you can discard it
my_dataset.discard_privacy_violation('id')

# Approve access permission by its ID
my_dataset.approve_access_permission('permission_id')

# Decline access permission by its ID
my_dataset.decline_access_permission('permission_id')
```


UNIVERSITY OF TARTU
Institute of Computer Science
1632