UNIVERSITY OF TARTU

FACULTY OF SOCIAL SCIENCES

NARVA COLLEGE

STUDY PROGRAM “INFORMATION TECHOLOGY SYSTEMS DEVELOPMENT”

Mihhail Matišinets

DEVELOPING A PYTHON LIBRARY FOR ESTONIAN OPEN DATA PORTAL’S API

Bachelor’s thesis

Supervisor: Andre Sääsk, M.Sc.

NARVA 2021

Olen koostanud töö iseseisvalt. Kõik töö koostamisel kasutatud teiste autorite tööd, põhimõttelised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

.....

Töö autori allkiri ja kuupäev

Non-exclusive licence to reproduce thesis

I, Mihhail Matišinets, 22.08.1996

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to

reproduce, for the purpose of preservation, including for addition to the DSpace digital archives until expiry of the term of validity of the copyright “Developing a Python library for Estonian Open Data Portal Api’s” supervised by Andre Sääsk, M. Sc

2. I am aware of the fact that the author retains the right referred to in point 1.

3. This is to certify that granting the non-exclusive licence does not infringe the intellectual property rights or rights arising from the Personal Data Protection Act.

Narva, **07.04.2022**

TABLE OF CONTENTS

Terms and abbreviations

API – Application Programming Interface.

URL – Uniform Resource Locator. Web address in the computer network or internet which points to some resource. For example, text file with some information (Berners-Lee, Uniform Resource Locators (URL), 1994).

JSON – JavaScript Notation Object. Human readable file format standard. Successor of XML, most common data format to exchange information between different types of web services (Bray, 2017).

VERSION CONTROL SYSTEM(VCS) – A tool that manages and tracks different versions of software or any other file (Loeliger & McCullough, Background, 2012).

GIT – Version control system software to track changes made in the files. Commonly used across software engineers to track and coordinate changes in the codebases (Chacon & Straub, 2014).

BRANCH – Is a split from a primary state of codebase, that allows for development to continue in other directions simultaneously and produce different versions of the project (Loeliger & McCullough, Branches, 2012).

ENDPOINT – Specific location for accessing service's resource using specific protocol or data format (The World Wide Web Consortium (W3C), 2004).

HTTP – Hypertext Transfer Protocol is an application-level protocol with the lightness and speed necessary for distributed, collaborative, hypermedia information systems. It is a generic, stateless, object-oriented protocol which can be used for many tasks, such as name servers and distributed object management systems, through extension of its request methods (commands) (Berners-Lee, Hypertext Transfer Protocol -- HTTP/1.0, 1996).

(DE)SERIALIZATION – Serialization is a process whereby a data structure is converted into a suitable format (JSON, text, bytes) to transfer over a network. Deserialization is a reverse process whereby received data in a given format is converted into an object of a programming language (Mozilla Developer Network, 2021).

API(REST) RESOURCE – API resource is a piece of information that is stored in the network and can be accessed via URL. Any information that can be named, can be a resource: text document, image, collection of other resources and so on (Fielding, 2000).

MOCK OBJECT(MOCKING) – In software development this is an object that acts as a substitution for real object in order to test object's behaviour in different circumstances. Likewise, mocking is a process of simulating real object's behaviour (Freeman & Pryce, 2010).

DEPLOYMENT PIPELINE(PIPELINE) – At an abstract level, deployment pipeline is a process of distributing software from version control repository to the machines of end users. That process can involve stages such as building the software, testing, deployment on some environment and release to the production (Humble & Farley, What Is a Deployment Pipeline?, 2010).

CONTINUOUS INTEGRATION (CI) – "is a software development practice where members of a team integrate their work frequently, usually each person integrates at least daily - leading to multiple integrations per day. Each integration is verified by an automated build (including test) to detect integration errors as quickly as possible." (Fowler, 2006).

CONTINUOUS DELIVERY (CD) – is a software delivery process which enables to get all types of changes to into the production environment safely, quickly and in sustainable way (Humble, What is Continuous Delivery?, n.d.).

PYTHON PACKAGE INDEX (PYPI) – is a Python's software register where software engineers can find or distributed packages for the programming language. (Python Software Foundation, 2022)

PULL/MERGE REQUEST – is a change or series of new changes proposed by a single or many contributors to the project codebase using git. Usually, those changes are then reviewed by maintainers of the project and decision is being made whether or not allow new changes to being pushed into the main branch of the project. (Loeliger & McCullough, Creating Pull Requests, 2012).

Introduction

Estonian Open Data portal is a platform where everyone can disclose any kind of data to the public whether it is data about markets, data gathered during scientific research or company's quarter revenue reports. This data can be composed into convenient datasets and presented on the website. The aim is to improve trust between general public and private or public institutions by making data transparent and available to everyone (Ministry of Economic Affairs and Communications, 2022).

Data provided by Open Data portal can be used for different kinds of purposes, for instance, to create data visualization applications to showcase analytics or to perform meta-analysis for some other research work.

Problem

Open Data portal provides two ways of interacting with datasets published on their website. First one is the standard one, through the web interface. The second way is to consume datasets by provided API endpoints. Those endpoints allow users to make HTTP requests directly to specific URL and utilize web resources laying on that endpoint. Be it file that relates to dataset or some other meta information about the dataset.

However, currently second approach for some users, especially the ones who would like to automate this process is a bit cumbersome, because, as at the moment of writing thesis, Open Data portal doesn't have any publicly available programming libraries, that provide convenient interface to consume endpoints programmatically. As a result, users must implement corresponding logic themselves, which might become potential bottleneck in their development process.

Solution

Solution for described problem above is to implement programming library and make it available to the public as an open-source project. This will allow anyone who has Open Data account to consume endpoints programmatically by using proposed solution. Moreover, by making this library open source, this will align with transparency mission that Open Data project carries and will ensure there is always someone who can contribute to the library one way or another.

Objectives and requirements

As stated in the section above the main goal of the project is to implement open-source library that covers functionality of all the endpoints provided by Open Data API, given, that the endpoints

themselves are functioning appropriately. It shall be stated that it is not in the scope of this work to ensure that API endpoints are functioning as expected, but rather provide programmatic interface for them and make this interface available to the general public.

Besides the main goal this programming library also should comply with set of requirements to make sure that it is meeting the definition of readiness. Requirements for the library are the following:

- The code of the library should be available to the public i.e. hosted as an open-source project on any of code hosting platform.
- The library must provide programmatic interface for every endpoint provided by Open Data API.
- The library must be easily installed through common registries for programming language that library is developed in.
- The library codebase should be covered with unit tests. At least 90% test coverage must be present.
- The library's programmatic methods names should resemble with Open Data API endpoints behaviour i.e. function names should be descriptive.

Outline

First chapter describes high-level analysis of Estonia Open Data API i.e., what is it and what capabilities it provides to users. Second chapter gives overview of design decision made on different levels of the project i.e., how library look on the high level and what is happening on deeper levels. Third chapter outlines development process of that library. In particular, in the third chapter can be found information about programming language choice, what dependencies were utilized and provides overview of development workflow. Finally, in the fourth chapter implemented solution is discussed, given consideration regarding future of the project and project's link is also located in the fourth chapter.

1 Analysis

This section provides high-level analysis of Open Data API portal and outlines researched solutions, that were used as a blueprint for the library.

1.1 Analysis of Open Data API

Open Data API provides possibility to retrieve, create, update and delete datasets. The advantage of using Open Data Portal API is that all those interactions can be automated and reduce human interaction potentially saving time and resources in the process.

Only authorized users are allowed to make requests to available endpoints. In order to authorize request, users have to create an API key and get access token upon every request. API key can be created on user's profile settings page. However, API keys is only valid for a year from moment of creation and user allowed to have up to 64 API keys. Access token for authorization is obtained by sending HTTP POST request with "Authorization" header to special endpoint. JSON format is used as a primary format for data transmission between user's and server (Ministry of Economic Affairs and Communications, 2022).

1.2 Endpoint's analysis

Endpoints are divided into 5 main categories: "Generic Datasets", "User's Datasets", "Organization's Datasets", "Authorization" and "Core".

"Generic Datasets" – category provides endpoints to access all publicly available datasets. However, for public datasets only read access is given. Endpoints for downloading files, that belong to datasets, are also available to the users. Files can be downloaded by sending HTTP POST request for special endpoint. Moreover, endpoints for applying for permissions and submitting privacy violations are given as well.

"User's Datasets" – this category of endpoints provides access to user's own datasets. Meaning user can freely read, create, modify and delete datasets and related information. On top of that, user can grant or decline different access injuries to datasets.

"Organization's Datasets – this category of endpoints is similar to user's described above, but scope is limited only for specific organization's datasets.

"Core" – this category of endpoints provides information regarding available categories, regions, languages, keywords and other static data, that can be used during creation of datasets.

"Authorization" category contains only a single endpoint. This endpoint can be used to obtain authorization token, which can be used to authorize user's requests.

1.3 Analysing similar solutions

Despite Open Data portal, at the moment of writing, didn't have any publicly available libraries for their API endpoints, programming libraries for different API services are quite widespread tool in the software engineering field and more important most of them are open to the public, meaning that anyone can explore their source code. Some of the biggest technology companies in the industry provide API endpoints to consume their web services programmatically and respective library written in popular programming languages.

This means that author doesn't have to come up with the unique solution, but rather build a library utilizing already well-established blueprints for similar problems by exploring open-sourced codebases. By using common techniques for building programmatic library author provides consistency between similar technologies, thus ensuring better developer experience.

For example, companies such as Twilio, Stripe, GitHub, Google, Amazon and many more provide web services along with API endpoints that can be accessed programmatically. In addition, they provide official libraries to consume API in most popular programming languages or promoting some third-party libraries created by their respective communities. Twilio, Stripe and GitHub libraries among all others were chosen and analysed as blueprints for Open Data portal's programming library. Things such as naming conventions, codebase structure, CI/CD practices, testing practices were analysed and later re-used or were used as an example to create this library own approaches for different issues that had to be tackled.

2 Design

This section describes design stage of the solution. Here is presented information about library's design decisions and description of what is happening on different levels of the processes.

2.1 High level overview

On a high-level the solution is rather simple. The library contains a set of functions that are mapped to respective HTTP endpoints. By using these functions, a developer can interact with Open Data service and retrieve necessary information. For example, a function that is mapped to GET endpoint would just read information from the server and return it to the developer in a human-readable format.

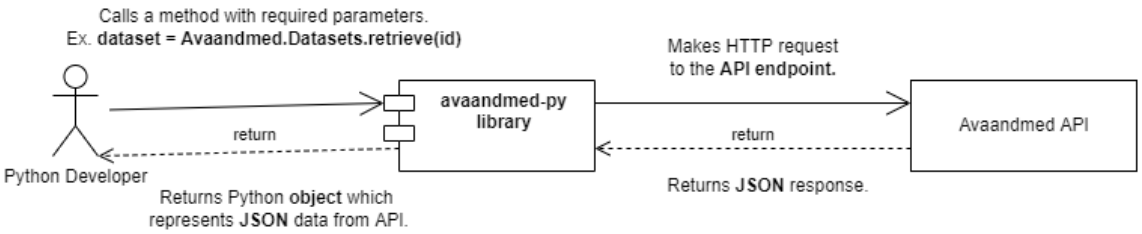

Figure 1. High-level overview of implementation. Created by Author.

2.2 Library level overview

On a library level process are getting a little bit more complex. One of the main library's responsibilities is to seamlessly serialize Python's data structures into transmissible format and pack that data into HTTP request that will be send to available endpoint.

Since Open Data API accepts data only in JSON format, library have to use the same format for serialization before sending it over to the server. When receiving data from the server library would perform the opposite. Deserializing received JSON from the server's response into Python's data structure and present it to the user.

In addition, Open Data requires to authorize each HTTP request to some of their endpoints, thus we have to perform authorization request before making a request to get an actual data from the server. All this communication is seamlessly handled by the library for developers. The overview of the whole process is presented on the **Figure 2.**

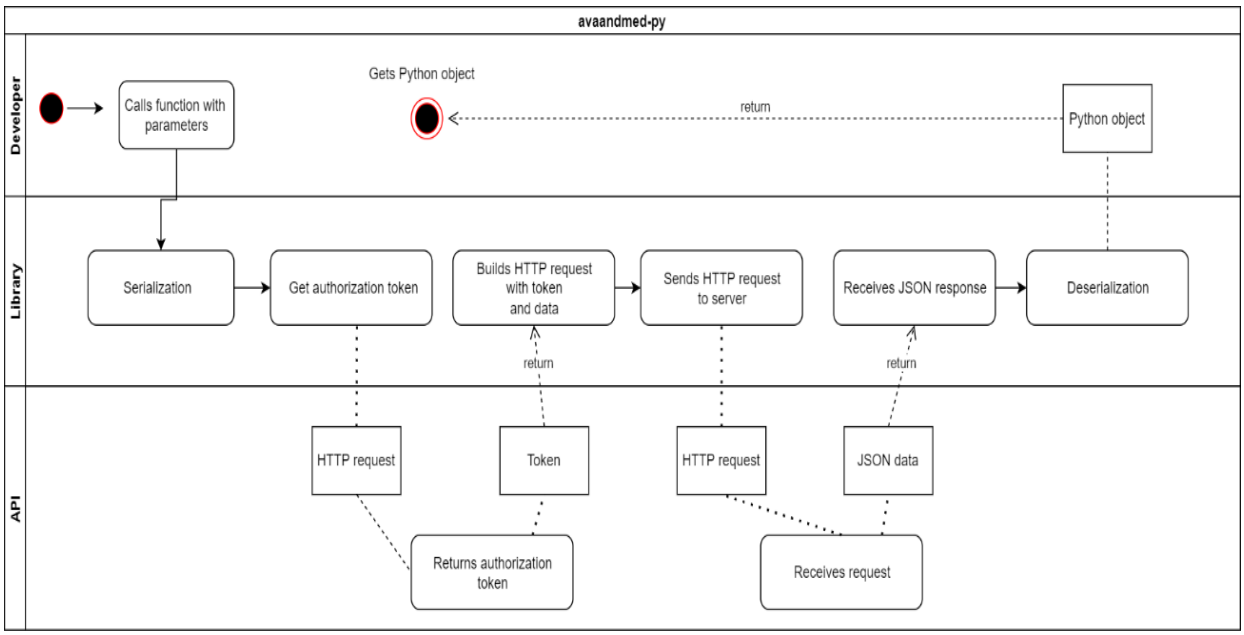

Figure 2. Library level overview. Created by Author.

2.3 Module level overview

This section describes how different modules inside the library are interacting between each other in order to achieve, what was described in the section above.

Library was organized into set of different modules to separate responsibilities and keep the codebase consistent with common software engineering principles. One of those principles that was followed is called — "Single Responsibility principle". This principal advocate that each module, class or a function should only have one responsibility to fulfil (Martin, 2014).

As a result, library has four main modules:

- **http** – module which is responsible for composing HTTP requests and other logic that relates to making connections to the API.
- **exceptions –** module which stores custom exceptions that can be used to display to a developer whenever some error occurs during library execution.
- **api_resources –** module contains Python classes that are going to be used for serialization/deserialization of JSON data.
- **utils –** is a support module, that is containing functions that are commonly utilised across the library codebase.

Graphical overview of modules interconnections is presented on **Figure 3.**

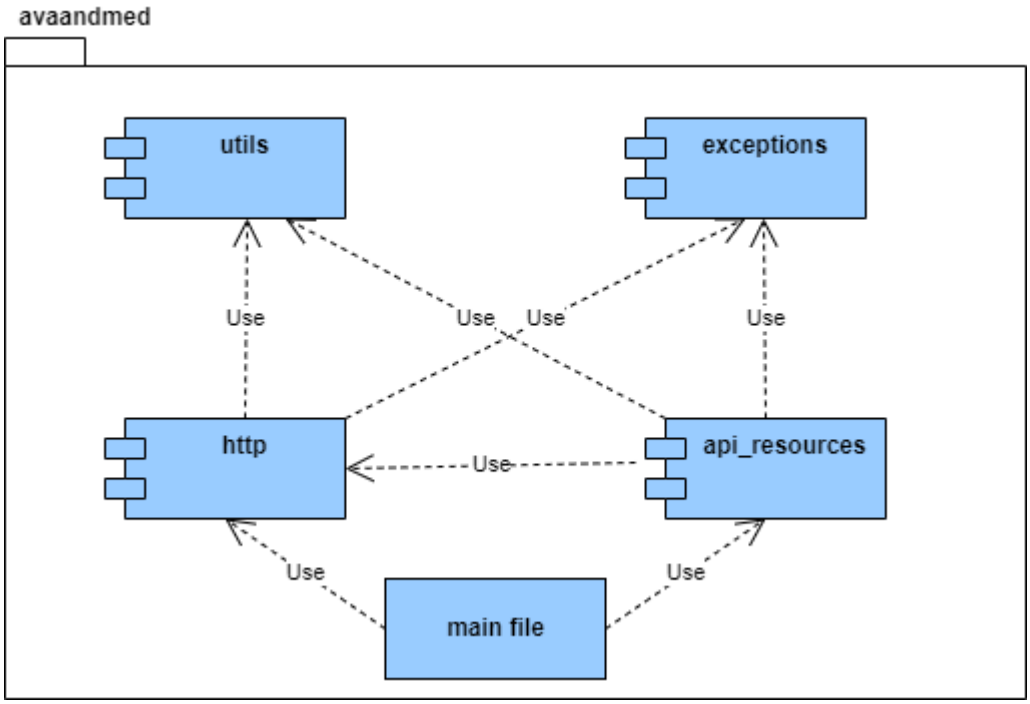

Figure 3. Modules structure's overview. Created by Author.

2.4 API resources module

While other modules contain at most one or two Python files with functions, **api_resources** module is more complex than that and deserves it is own section to be described.

During the analysis of the Open Data API endpoints, it became clear that putting all the logic inside one module, would let to poorly organized code structure and might be a cause of developer's frustration in the future when attempting to introduce any new changes to the codebase. Hence, it was decided to split **api_resources** module into submodules.

Organization of the submodules was followed by the same principle described above. Whereby producing 4 additional submodules in total: **users**, **organizations**, **datasets**, **entities**. Each submodule resembles actual API resources that are accessible via endpoints. Therefore, for example, **users** submodule responsible for providing means for the developer to interact with resources that are stored under **/users** API endpoints. Other submodules have similar function, except for **entities** submodule.

Submodule **entities** stores common Python classes, that are utilized by other submodules in order to serialize and deserialize data between Python's data structure and JSON format. Also, to provide type annotations for the developers to facilitate usage of provided data structures.

2.5 Classes level overview

Modules contains code that is organized into set of different classes. Classes were created by following the same principle described above i.e., each class only has one responsibility to fulfil. As a result, this helped to keep necessary data close to each class and make codebase more encapsulated. Consequently, a class **Avaandmed** was created that acts as an entry point for interacting with Open Data API endpoints. Classes such as **Datasets**, **Users** and **Organizations** contains logic that is mapped to respective category of endpoints. For instance, through class Users, developer can access endpoints under "Users" category described in **1.1.1** section.

As described previously, the class Avaandmed acts as an entry point for a library, hence through Avaandmed class developer can access other classes like: Datasets, Users and Organizations to interact with their respective endpoints. In addition, Avaandmed class contains methods to retrieve information from endpoints in "Core" category as described per **1.2.** section.

Since categories of endpoints such as "Generic Datasets", "Users Datasets" and "Organization Datasets" have significant overlap in provided functionality, it made sense to encapsulate methods with the same usage for different classes in one class – **DatasetRepository**. This class contains methods, that other classes such as **UserDataset** and **OrganizationDataset** can use to interact with data related to datasets from respective endpoints. Therefore, avoiding repetitiveness between similar classes.

Logic responsible for HTTP requests is encapsulated into separate **HttpClient** class. This class also exposes **request()** and **download()** generic methods. Method **request()** can be used to send all types of HTTP requests with necessary data to the endpoints. Having one generic method for all HTTP requests helped to avoid repetitiveness. Likewise, custom exception class was designed to outline errors, that might be thrown during HTTP requests.

Graphical overview of classes can be seen on **Figure 4**.

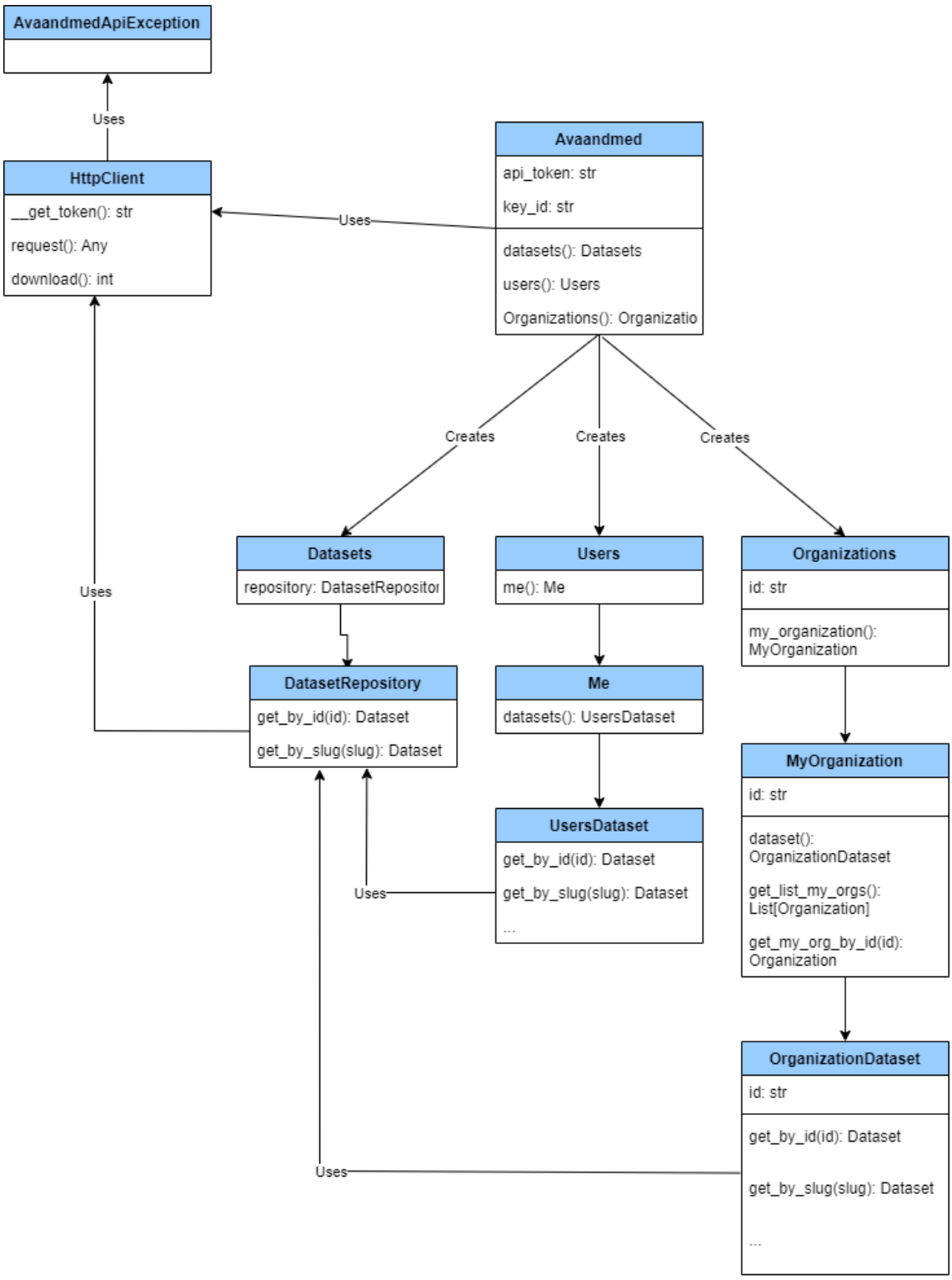

Figure 4. Classes level overview. Created by Author.

3 Development

This section describes the development process. Here is outlined information about what technologies or third-party libraries were used, how code was tested and other development related decisions.

3.1 Programming language choice

3.1.1 Python

Python Software Foundation describes Python programming language as follows: “Python is an interpreted, interactive, object-oriented programming language. It incorporates modules, exceptions, dynamic typing, very high-level dynamic data types, and classes.” (Python Software Foundation, 2022).

Python is also being recognized as one of the popular programming languages, if not the most. According to data from Google Trends, around 30% of search requests is people looking for Python’s tutorials (Carbonnelle, 2022). Likewise, Python is among first languages that is being taught in universities all across the globe.

3.1.2 Why?

The enormous popularity of the language, its multipurpose nature and community that was built around the language over the years, were the main reasons why this particular programming language was chosen as a primary development language for the library. The popularity of the language will ensure, that library would be getting more attention from other developers. Moreover, the cost effectiveness of finding someone to maintain or contribute to this library in the future is much lower compared to the other languages. Language’s multipurpose provided author with flexible development environment, where most of the necessary tools were already developed and could be utilized. Finally, with big community around the language, it is always possible to find help on how to solve different issues when it comes to development of the library, which greatly improves overall experience.

3.2 Utilized technologies

On top of the standard Python’s library some of the third-party solutions had to be used in order to reduce codebase complexity and reduce development timeline on implementing logic that is already developed by other developers and can be used out of the box as a dependency.

This section contains information about third-party libraries and technologies that were used to facilitate development. These third-party libraries and technologies were used for the following

purposes: serialization/deserialization between JSON and Python's data structures, creating HTTP requests, mocking Open Data API endpoints to automate testing processes during development, creating a CI/CD pipeline for testing and deployment of the library to the PYPI register.

3.2.1 Requests

Requests – is a simple library for Python that allows sending HTTP request. On top of that, it provides many convenient features, which can be used to build efficient HTTP clients for Web API services (Reitz, 2022). This library is considered to be go-to library when it comes to creating HTTP requests with Python.

This library was used to create HTTP client for Open Data library, that is used to compose HTTP requests. Its simplicity and community around allowed quickly to start developing necessary HTTP functionality required for the library.

3.2.2 Responses

For every request there should be a response. For this reason, "responses" library was used to simulate HTTP responses during testing that can be received from Open Data endpoints.

Responses – is a utility library that was created for mocking "requests" library described above (Sentry, 2022). This utility allowed author to easily create mock objects, that were used to simulate behaviour of Open Data API endpoints under different scenarios and develop necessary logic to act accordingly.

3.2.3 Pydantic

Pydantic – is a data validation, parsing and management library (Colvin, 2022). It allows to create data models, that can be efficiently converted into other formats like JSON. This library was used for serialization and deserialization between Python's data structures and JSON data received from the endpoints.

3.2.4 Pytest

Pytest – is framework that allows for painless creation of complex functional tests for Python's codebases (Krekel, pytest, 2004). This framework along with "requests" library was used to create and organize unit and functional tests for Open Data library.

3.2.5 Tox

Tox – is a tool that helps to automate testing processes in Python (Krekel, Tox, 2022). It allows to run tests in different environments. For example, to run tests with different version of Python or on different operating systems in order to see how code is behaving under different conditions.

This utility was used to automate testing processes in local development environment and inside CI/CD pipelines. It allowed to raise confidence in the quality of Open Data library's codebase and improve overall quality of the project.

3.2.6 Git

Git was used as a main version control system to track codebase's changes during the development. Tool was chosen primarily because author had previous experience with it, and it is considered mostly the standard way of managing versions of code in software development industry that most developer as familiar with.

3.2.7 GitHub

GitHub – is a most popular software hosting platform in the world with more than 70 million users (Github Inc, 2022). Given its popularity and wide use among developers it was chosen to be as a code hosting platform for Open Data library.

3.2.8 GitHub Actions

GitHub Actions – is a GitHub's feature that can be used to automate some of software development processes such as testing, delivery and many others. Essentially providing to the project free CI/CD pipeline capabilities. This feature was used to automatically run test before merging code changes into main branch and generate test coverage reports.

3.3 Development workflow overview

This section summarizes how technologies described in the section above was used for development of Open Data library and overall development process.

3.3.1 Environment's local setup

On the local machine the development environment was set up using Python's virtual environment. Python's virtual environment helped to isolate development environment settings from system's default settings, which allowed to avoid any potential issue relating to managing different dependencies. Having isolated development environment is considered to be one of the standards of software development practices.

3.3.2 Tracking changes

As mentioned earlier, changes were tracked using version control system such as git. Local git repository was initialized with the primary branch named "main". "Main" branch's responsibility is to store stable and tested version of the library's code.

3.3.3 Remote code hosting

GitHub was used as a code hosting platform to host remote git repository to ensure, that project would not disappear completely in case of the total failure of local machine e.g., hard drive failure or some other critical issue which could lead to data corruption. Moreover, GitHub provides visibility to other people who wish to interact with the project one way or another.

3.3.4 CI/CD pipeline

GitHub Actions were used to setup CI/CD pipelines to run functional regression tests upon each push and merge request into the "main" branch. In addition, pipeline was used to create test coverage reports to visualize how much of the codebase is covered with tests. Furthermore, continuous delivery feature of the GitHub Actions allowed to upload source code of the main branch directly to the PYPI register whenever there is a new change in the main branch, therefore making latest version always available for installation.

3.4 Workflow

Firstly, an issue or feature that is needs to be implemented was identified. Then new issue was open on the GitHub project's page with respective description. Description contained information such as what must be implemented and what is the expected outcome along with other completion criteria if applicable. Next on the local machine separate git branch with relevant name was created to address new issue. In addition to implementing the solution, at least one test needed to be implemented to confirm correctness of the new logic. After that, if test was successfully passed, the regression tests were executed using tox library, thus verifying that none of the previous logic was broken by introducing new changes to the codebase. When regression testing was done, documentation was created to describe new changes. Library version also was updated to indicate that there were changes in the code. Performed work was then committed using git and new changes were pushed into the remote repository on the GitHub. Once pull request was submitted, using GitHub Actions, the same regression tests were executed once more, but on different operating systems with different Python's versions to verify that new changes are working as expected across different environments. When all checks were passed, tests report was created to visualize how much of the codebase was covered with tests. If none of the new issues were found during testing stage, then code was merged into the main branch and uploaded to PYPI registry so developers could update library to a new version.

Graphical overview of workflow process can be viewed on **Figure 5.**

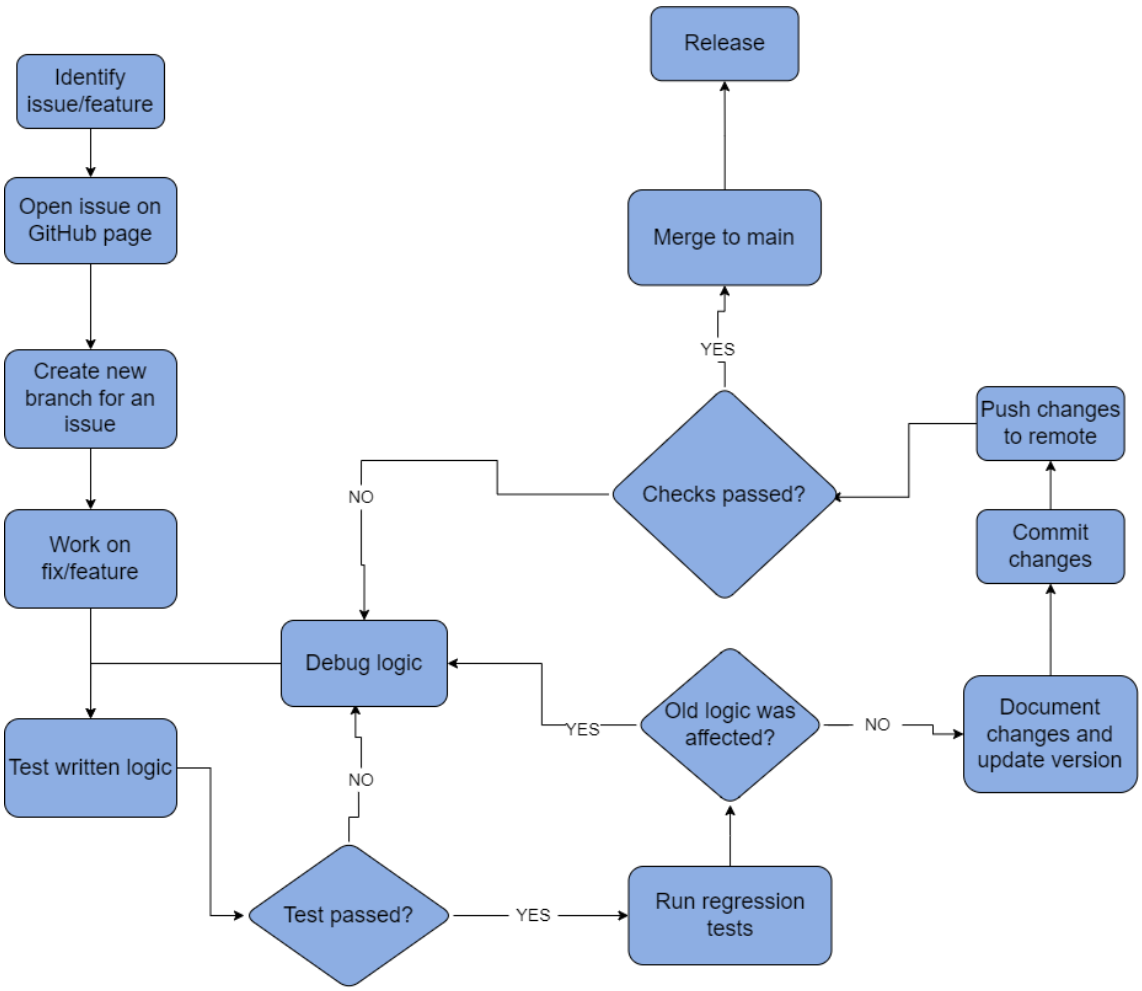

Figure 5. Workflow diagram. Created by Author.

4 Implemented solution

This section describes final outcome of the performed work. In particular, this section contains information about open sourcing the library, licencing, how library is being distributed, how library can be installed and general information about library's capabilities.

4.1 Open source

The source code of the library is available for the public on the GitHub page of the project.

By open sourcing this project, author makes sure that other developers can contribute to the project either by implementing new feature, improving existing code or fixing an issue. This ensures that there always be someone who could maintain library code even when author is incapable of doing so. Furthermore, in case project would be completely abandoned, others could simply copy the project by creating their own fork in GitHub account and continue to develop their version of the library.

4.2 License choice

MIT licence was chosen for this library. This license allows others to use this library for commercial purposes and gives permission to modify the source code if required. Moreover, it signals to developers, that there is no warranty or any other obligation to provide further support of the library. However, in case of using or modifying library's source code, copyright and licence notices should be preserved.

Since this library is not an official implementation created by Open Data portal, but rather one of many possible solutions created by the author individually, a MIT license is the best fit for such individual projects. This way author is being legally protected from any sort of issues, but also can receive credit whenever code of the library is being used.

4.3 Distribution

The library is being distributed via the Python Package Index (PYPI). This is the standardized way of distribution of Python's libraries, hence it was decided to use it as well, because Python developers are most familiar with this resource and that would greatly reflect on user experience overall.

4.4 Installation

Library can be installed on any operating system that has a version of Python at least 3.6 or above. There are no plans to support lower versions as some of the dependencies used for development require to have a version of Python 3.6 or higher as well. And ensuring backward compatibility with a much older version of Python would increase the complexity of the software overall, which would lead to potential bottlenecks and other issues.

Library can be installed with following commands: **pip install avaandmed.** This will install the library and required dependencies.

4.5 Capabilities

Implemented solution allows users to perform almost all of the same actions with Open Data datasets programmatically as they would by using the portal's website. The capability of library is only limited by the provided API endpoints. And surely if Open Data decides to add to functionality to the portal and expose it to users via their API, the same functionality also could be added to the library.

Registered users and organizations can create, update, delete, modify and retrieve their own datasets. Moreover, they can programmatically grant access permissions or resolve privacy violations complains if there are any. Public datasets can also be retrieved by any user; however, API provides only read access, because of that it won't be possible to perform any other action besides retrieving information on them.

Among other capabilities, users can upload different files for their datasets by using respective functions. Datasets also can be rated, which helps to increase overall quality of the dataset. In addition, library provides means by which static information such as keywords, regions or other data can be retrieved, that later can be utilized for different purposes. Complete set of provided functionality can be viewed on the GitHub project's page.

4.6 Potential use cases

Open Data portal provides different variety of data, consequently opening up possibility for large number of different data analytical projects and with Python being the most popular language in Data Science field at the moment, this library will come very handy for anyone who is interested in data analysis. Furthermore, data visualization applications also can be built by using this library. This library also will be useful for anyone who needs to perform meta-analysis of the datasets in automated

manner using Python, they now can achieve that with much less effort as all the functionality is already being implemented for them.

4.7 Future considerations

Despite project has been completely implemented functionality for programmatically using Open Data API, maintenance of the project is still an open question. The author will try to do their best to provide support for the project. Meaning if there is any new feature that needs to be implemented or issue that needs to be addressed the author will try to resolve it.

Moreover, since this project is being open-sourced, this opens a possibility for other developers or students to contribute to this project and possibly make their own thesis out of it.

For example, as library was built with simplicity in mind some aspects of optimization practices were neglected to avoid complexity. Since code optimization is one of most popular subjects in software engineering, code optimization of this library could be one of the potential bachelor's thesis topics during which students would optimize code and perform benchmarking activities to explore by how much library's performance was improved and if there were any significant improvements at all.

4.8 Project's link

Project is available on the GitHub via the following link:

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

Conclusion

For this bachelor thesis the potential problem for Python developers who would like to consume Open Data API was identified. As part of the solution the Python library was created by an independent author, that provides functionality to interact with the API. The library was then uploaded to a popular registry in Python's ecosystem to facilitate the distribution of it and make it easier for others to install. In addition, library's source code was published on the code hosting platform GitHub and made available to the public with an intention to attract other contributors and create community around to facilitate the maintenance of the project.

Overall, all objectives described in the section "Objectives and Requirements" of this thesis were successfully fulfilled. To iterate over what was done: library provides methods to interact with every endpoint provided by API, 100% test coverage for the codebase was achieved, library can be easily installed with the help of Python's common package manager and finally the library's source code was made available to the public on the GitHub page. This gives confidence to the author to conclude, that project was successfully completed with accordance to set requirements.

Resümee

Hetkel Eesti Avaandmete portaal ei paku oma kasutajale ühtegi teeki, mis võimaldaks programmiliselt kasutada nende API, ning ka Internetist ei olnud leitud ühtegi avalikku lahendust selle kohta. Seoses sellega, et autor pidas Avaandmete portaali kasutamise julgustamist ühiskondlikult oluliseks, otsustas ta teha oma bakalaureusetöö eesmärgiks Pythoni teegi loomist Eesti avaandmete portaali API jaoks, et kasutajad, kes soovivad automatiseerida oma andmestike töötlemise protsesse API kaudu, oleksid võimelised mugavalt ja ajasäästlikumalt rakendada selliseid protsesse. Vastasel juhul kogu loogikat, mis võimaldaks programmiliselt kasutada API, peaks kasutaja looma ise.

Lõputöö on jagatud neljaks peatükiks, mis annavad ülevaade teegi arendamise protsessist. Esimeses peatükis on kirjeldatud Eesti avaandmete API analüüs. Teises peatükis on toodud arhitektuuri kujunemise protsess. Kolmandas peatükis räägitakse arendamise osast ja kasutatud tehnoloogiatest. Neljas peatükk kirjeldab loodud teegi võimalusi ja kuidas seda tarnitakse. Neljanda peatüki lõpus jagab autor oma ideid, kuidas teeki võiks tulevikus edasi arendada.

Teegi loomiseks oli kasutatud populaarne tänapäeval programmeerimiskeel Python ja selle kogukonna poolt pakutud abitööriistad.

Lõputöö tulemuseks on loodud avatud lähtekoodiga GitHub projekt, kus iga vabatahtlik arendaja saab teha ettepanekuid teeki edasise arendamise ja leitud vigade kohta. Teek oli laaditud “Python Package Index” registrisse, et lihtsustada installeerimise protsessi Pythoni arendajate jaoks.

References

Berners-Lee, T. (1994, December). *Uniform Resource Locators (URL).* Retrieved February 2, 2022, from https://datatracker.ietf.org/doc/html/rfc1738

Berners-Lee, T. (1996, May). *Hypertext Transfer Protocol -- HTTP/1.0.* Retrieved February 2, 2022, from https://datatracker.ietf.org/doc/html/rfc1945

Bray, T. (2017, December). *The JavaScript Object Notation (JSON) Data Interchange Format.* Retrieved February 2, 2022, from https://datatracker.ietf.org/doc/html/rfc8259

Carbonnelle, P. (2022). Retrieved March 7, 2022, from PYPL PopularitY of Programming Language Index: https://pypl.github.io/PYPL.html

Chacon, S., & Straub, B. (2014). *About Version Control.* Retrieved February 2, 2022, from Git: https://git-scm.com/book/en/v2/Getting-Started-About-Version-Control

Colvin, S. (2022). *pydantic*. Retrieved March 8, 2022, from https://pydantic-docs.helpmanual.io/

Fielding, R. T. (2000). *CHAPTER 5 Representational State Transfer (REST).* Retrieved February 26, 2022, from Architectural Styles and the Design of Network-based Software Architectures: https://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm#sec_5_2

Fowler, M. (2006, May 1). *Continuous Integration.* Retrieved March 10, 2022, from martfinfowler: https://martinfowler.com/articles/continuousIntegration.html

Freeman, S., & Pryce, N. (2010). Support for TDD with Mock Objects. In *Growing Object-Oriented Software, Guided by Tests* (pp. 19-20). Retrieved March 7, 2022

Github Inc. (2022). *About*. Retrieved March 8, 2022, from Github: https://github.com/about

Humble, J. (n.d.). *What is Continuous Delivery?* Retrieved March 10, 2022, from Continuous Delivery: https://continuousdelivery.com/

Humble, J., & Farley, D. (2010). What Is a Deployment Pipeline? In *Continuous Delivery* (pp. 106-109). Pearson Eduation, Inc. Retrieved March 10, 2022

Krekel, H. (2004). *pytest*. Retrieved March 8, 2022, from https://docs.pytest.org/en/latest/index.html

Krekel, H. (2022). *Tox*. Retrieved March 8, 2022, from https://tox.wiki/en/latest/

Loeliger, J., & McCullough, M. (2012). Background. In *Version Control with Git* (2nd ed., p. 1). O'Reilly Media Inc. Retrieved March 8, 2022

Loeliger, J., & McCullough, M. (2012). Branches. In *Version Control with Git* (2nd ed., p. 89). O'Reily Media Inc. Retrieved March 8, 2022

Loeliger, J., & McCullough, M. (2012). Creating Pull Requests. In *Version Control with Git* (2nd ed., pp. 394-395). O'Reily Media Inc. Retrieved March 26, 2022

Martin, R. C. (2014). SRP: The Single-Responsibility. In *Agile Software Development: Principles, Patterns, and Practices* (pp. 95-98). Pearson Education Limited.

Ministry of Economic Affairs and Communications. (2022, February 1). Retrieved February 1, 2022, from Estonian Open Data Portal: https://avaandmed.eesti.ee/

Ministry of Economic Affairs and Communications. (2022). *OpenData Dataset API*. Retrieved February 28, 2022, from https://avaandmed.eesti.ee/api/dataset-docs/#/

Mozilla Developer Network. (2021). *Serialization*. Retrieved February 18, 2022, from Mozilla Developer Network: https://developer.mozilla.org/en-US/docs/Glossary/Serialization

Python Software Foundation. (2022). Retrieved March 26, 2022, from Python Package Index: https://pypi.org/

Python Software Foundation. (2022, March 7). *General Python FAQ*. Retrieved from Python Documentation: https://docs.python.org/3/faq/general.html#what-is-python

Reitz, K. (2022). *Requests*. Retrieved March 7, 2022, from Requests: HTTP for Humans: https://docs.python-requests.org/en/latest/

Sentry. (2022). *Responses*. Retrieved March 8, 2022, from Github: https://github.com/getsentry/responses

The World Wide Web Consortium (W3C). (2004). *Web Services Glossary*. Retrieved February 2, 2022, from https://www.w3.org/TR/ws-gloss/