Summer Semester 2020

Datenbanken und Web-Techniken

Project Term Paper

SUBMITTED BY

Shovra Das
Matriculation # 550659
Study Program: Master Web Engineering

June 28, 2020

PROJECT NAME

ProxyAlly

This project corresponds to a web application which aggregates HTTP(S) proxies from variety of sources around the web and make them available under a single interface with an extensive set of features.

TECHNISCHE UNIVERSITÄT
CHEMNITZ

Contents

Figures

Tables

1 Introduction

Location based access control or restriction on resources is common in the context of Web. For numerous reasons, a service provider may control the access including the restrictions imposed by concerned authority. This temporary or permanent unavailability may affect users to utilize individual offers based on their geographical location. Using proxies can be considered as one of the possible solutions to avoid this restriction. Proxies can pretend to be a proper client to the requested service and after getting the response back it forwards the answer to end user. When using proxies, however, the problem arises that a user must know the proxy address and many proxies are often permanently not available [1].

Project **ProxyAlly** has been developed with the aim to aggregate proxy lists from various free proxy list providers which allow end users to access these proxies form a common interface. While processing the data it also evaluates the basic functionality of the individual HTTP(S) proxies. Additionally, ProxyAlly help users to determine if any proxy is functional against a desired (restricted) service.

2 Project: ProxyAlly

2.1 Requirements

The top-level requirements for the project include a backend which is capable of aggregating and processing data from various proxy list. While processing, a database is used to store the aggregated data which can be later accessed by the end user using an interactive frontend. Frontend consumes the data from data store using a REST based interface. The diagram below describes the scenario even better.

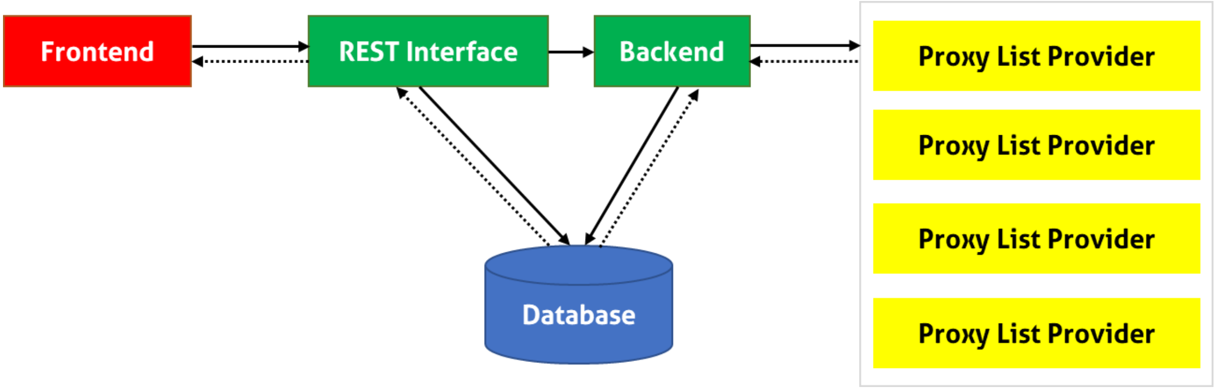


Figure-1: Project Structure

2.1.1 Requirements Breakdown

The table below illustrates the requirements in more detail.

APPLICATION STACK	TODOs	REMARK
Front End	UI to display list of proxy providers	
	UI to display the detail of a provider	Includes the list of related proxies
	UI to create a new proxy provider	Includes the possibility to choose sync proxies immediately or later
	UI to edit existing proxy providers	
	UI to remove existing proxy providers	Also removes related proxies
	UI to display available test URLs	These are the service URLs to test against the proxies
	UI to display collected proxies with details	Includes filters
	UI to export filtered proxies	
	UI for manipulating configurations	
Back End **REST Interface**	Sync new or existing proxy providers with the updated proxy list.	Prevents multiple fetches within 10 minutes. Prevents multiple URL access within 1 seconds for a single fetch.
	Test the basic functionality	
	Validate test URLs against proxies	
	Remove the stale proxies	If not updated for more than a week
	Expose all functionalities as REST API	A set of endpoints
Database	Conceptualize the Schema	

Table-1: Requirements breakdown

2.2 Implementation

2.2.1 Database

ProxyAlly highly depends on the collected data from the web which means the data structure may vary depending on the provider. However, the diagram below shows the minimal conceptual schema supporting the current implementation.

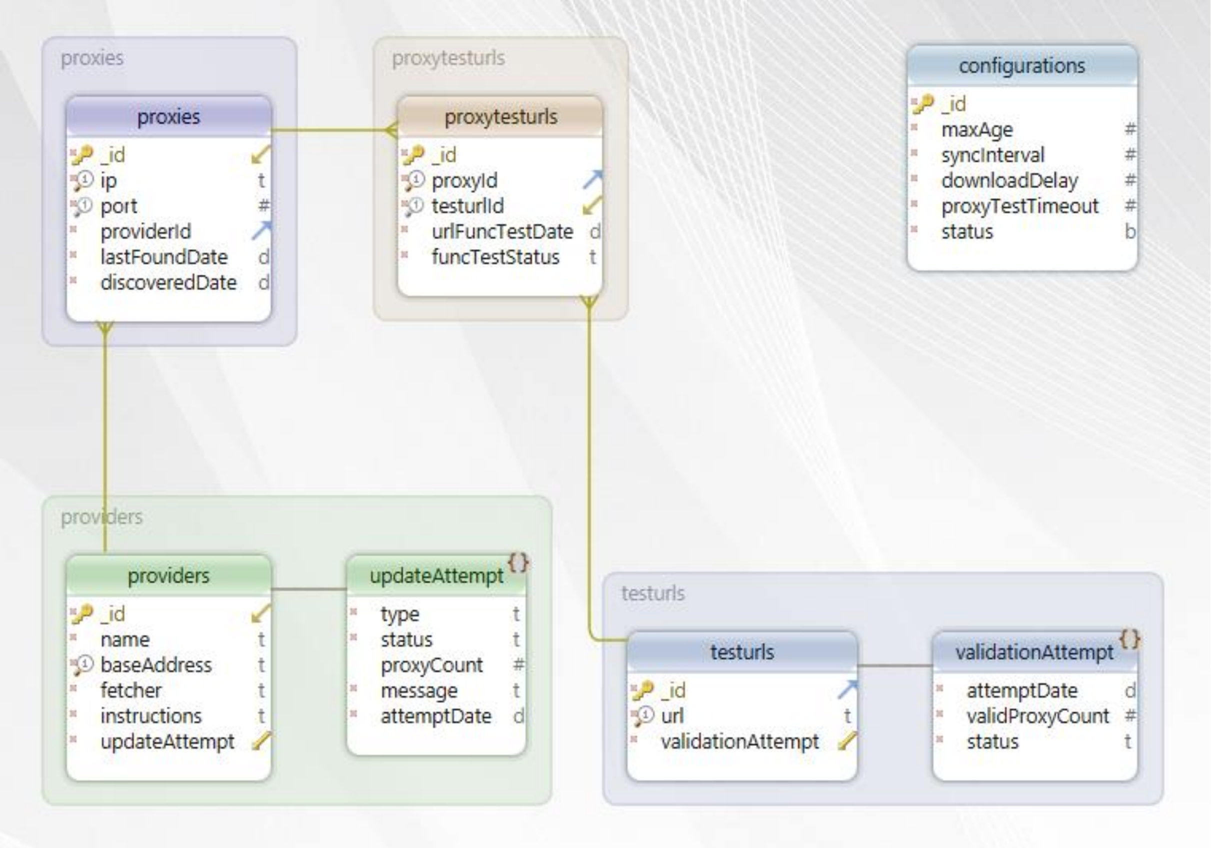


Figure-2: Database Diagram

In general, a total of 4 collections and 1 relational collection has been created to serve the application need.

2.2.2 Application Architecture

The overall project has been broken down into two separate web applications where "webapp" refers to frontend and top level "webapi" serves as backend.

The "**core**" package inside "webapi" is the **aggregator** module which deals with scraping or fetching data from providers. This package also contains the "proxy_checker" module which evaluates certain attributes such as "Anonymity" of a given proxy.

The "**webapi**" subdirectory inside "webapi" holds all the **REST interfaces** (also known as resources) necessary for the frontend. Some interfaces are created proactively so that they can be used for future enrichment of the project. (Please refer to the Appendix for further detail)

```
proxyally
|___ webapp                           [ Frontend ]
|
|___ webapi
        |___ core                     [ Backend Core ]
        |       |___ proxy_fetchers
        |       |___ proxy_checker.py
        |
        |___ webapi                   [ REST Interface ]
                |___ resources
```

2.2.3 Functionality Explanation

This section contains the detail of some basic logics which are applied to implement the core part of the application.

Fetching the Proxies:

"Scrapy" spiders are used to fetch proxies from the source URL. Only the base URL is provided. Rest of the navigation is configured for automated crawling. Mechanism for controlling the search depth is also implemented through "max_page" guard variable. An example spider is provided below as reference.

```
class ProxyScrapSpider(scrapy.Spider):
  name = 'freeproxylistsnet'
  page_number=2
  max_page=3

  def start_requests(self):
      yield scrapy.Request('http://www.freeproxylists.net/', self.parse)


  def parse(self, response):
    table = response.xpath('//table[@class="DataGrid"]')[0]
    trs = table.xpath('tr')
    for tr in trs:
      tds = tr.xpath('td')
      if len(tds)>1:
        ip_text = tds[0].xpath('script/text()').extract_first()
        if ip_text:
          ip_text = unquote(ip_text.split('"')[1])
          ip_text = scrapy.Selector(text = ip_text)
          yield {
            'ip': ip_text.xpath('//a/text()').extract_first(),
            'port': int(tds[1].xpath('text()').extract_first()),
            'https': tds[2].xpath('text()').extract_first()
          }

     next_page = f'http://www.freeproxylists.net/?page={ProxyScrapSpider.page_number}'
     if ProxyScrapSpider.page_number <= ProxyScrapSpider.max_page:
        ProxyScrapSpider.page_number += 1
        yield response.follow(next_page, callback=self.parse)
```

Source Code: A Scrapy Spider Example

Testing Basic Functionality of Proxies:

To test the functionality of a given proxy the help of below proxy judges is taken. IP providers are utilized to get the real time IP address of the client and compared with the result retrieved from the judges.

```
PROXY_JUDGES = [
  'http://proxyjudge.us/azenv.php',
  'http://mojeip.net.pl/asdfa/azenv.php',
  'https://azenv.net/',
  'http://www.proxy-listen.de/azenv.php',
  'http://httpheader.net/azenv.php'
]

IP_PROVIDERS = [
  'https://api.ipify.org/?format=json',
  'https://ip.seeip.org/jsonip',
  'https://api.myip.com/',
  'https://ip.seeip.org/jsonip',
  'https://api.my-ip.io/ip.json'
]
```

Checking the Anonymity of Proxies:

To check the anonymity the below headers were searched in the retrieved from the judges. If they exist, then the proxy is no more “Elite” one. They are then categorized as “Anonymous” proxies.

```
ANONYMOUS_HEADERS = [
    'ACCPROXYWS',
    'CDN-SRC-IP',
    'CLIENT-IP',
    'CLIENT_IP',
    'CUDA_CLIIP',
    'FORWARDED',
    'FORWARDED-FOR',
    'REMOTE-HOST',
    'X-CLIENT-IP',
    'X-COMING-FROM',
    'X-FORWARDED',
    'X-FORWARDED-FOR',
    'X-FORWARDED-FOR-IP',
    'X-FORWARDED-HOST',
    'X-FORWARDED-SERVER',
    'X-HOST',
    'X-NETWORK-INFO',
    'X-NOKIA-REMOTESOCKET',
    'X-PROXYUSER-IP',
    'X-QIHOO-IP',
    'X-REAL-IP',
    'XCNOOL_FORWARDED_FOR',
    'XCNOOL_REMOTE_ADDR'
]
```

Restricting the Update in too small interval:

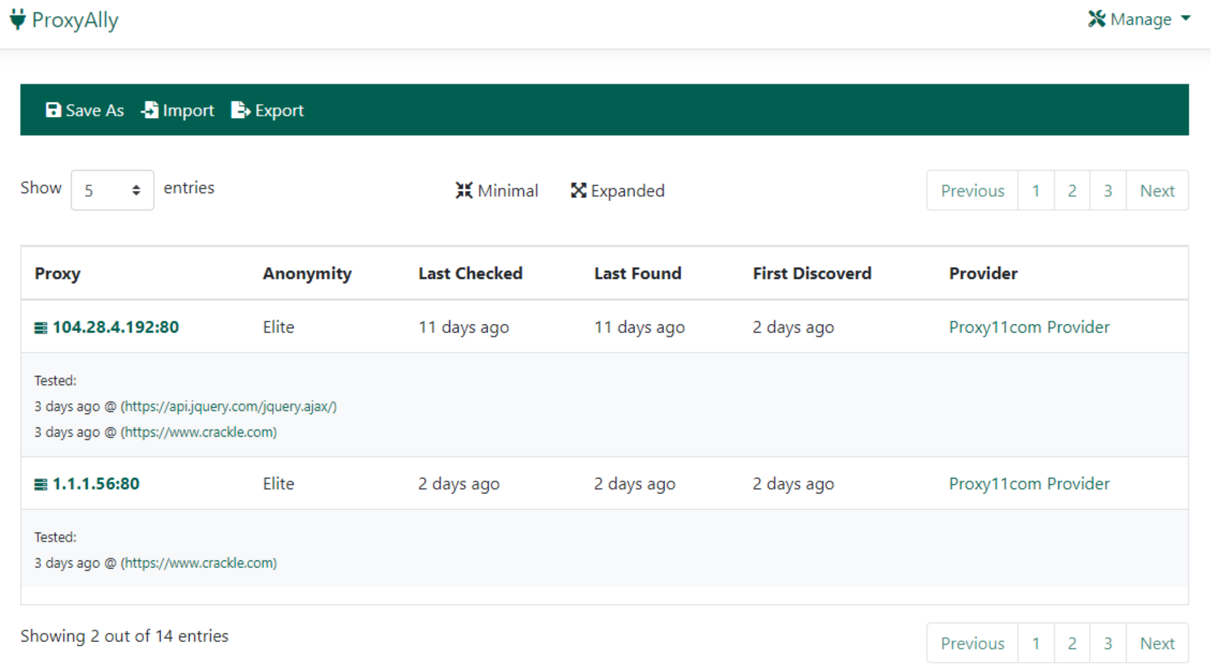
The “syncInterval” field has been stored in database under configuration. Whenever there is an update request it restricts the request comparing the value set for “syncInterval”. Default is set o 10 minutes.

```
if 'updateAttempt' in provider:
    duration = datetime.datetime.now() - provider['updateAttempt']['attemptDate']
    syncInterval = datetime.timedelta(0, 0, 0, 0, cfg['syncInterval'], 0, 0)
    if duration<syncInterval:
        abort(429, message="Too many requests", status="error")
```

Restricting the requests of same host:

This feature is handled through Scrapy supplying the “DOWNLOAD_INTERVAL” configuration item. This interval is also stored in database. Default is set to 1 second.

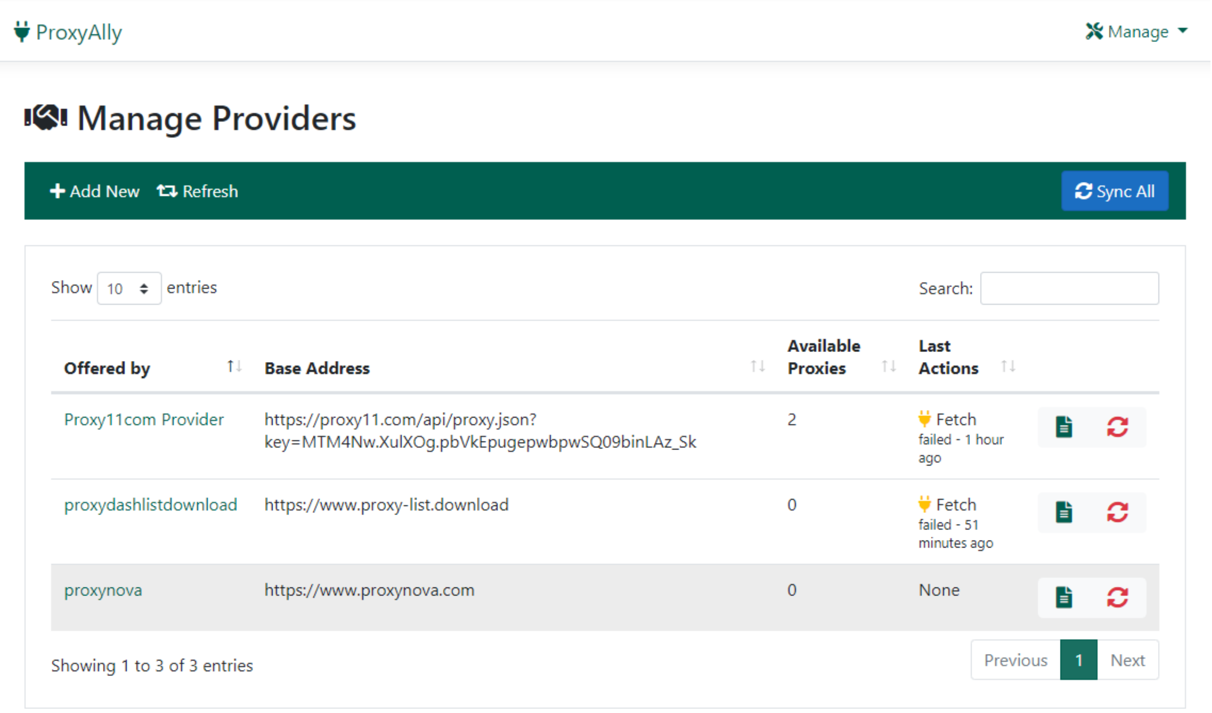
2.2.4 Sample Screenshots



Proxy	Anonymity	Last Checked	Last Found	First Discoverd	Provider
104.28.4.192:80	Elite	11 days ago	11 days ago	2 days ago	Proxy11com Provider
Tested: 3 days ago @ (https://api.jquery.com/jquery.ajax/) 3 days ago @ (https://www.crackle.com)					
1.1.1.56:80	Elite	2 days ago	2 days ago	2 days ago	Proxy11com Provider
Tested: 3 days ago @ (https://www.crackle.com)					

Showing 2 out of 14 entries

Figure-3: Home page



Manage Providers

Offered by	Base Address	Available Proxies	Last Actions	
Proxy11com Provider	https://proxy11.com/api/proxy.json?key=MTM4Nw.XulXOg.pbVkEpugepwbpwSQ09binLAz_Sk	2	Fetch failed - 1 hour ago	
proxydashlistdownload	https://www.proxy-list.download	0	Fetch failed - 51 minutes ago	
proxynova	https://www.proxynova.com	0	None	

Showing 1 to 3 of 3 entries

Figure-4: Manage Provider List

2.3 Deployment

2.3.1 Package Overview

The application is distributed as a zip archive (proxyally.zip) which travels along with this document. After extracting the zip file two directory titled “webapp” and “webapi” can be noticed. “webpp” refers to the frontend application and “webapi” serves as the backend along with a REST interface. A database is not needed to be created since it is handled through backend itself. A glimpse of the package is illustrated below for reference.

```
proxyally
|___ webapp              [ frontend ]
|      |___ README.md
|      |___ runserver.py
|      |___ requirements.txt
|      |___ other artifacts ...
|
|___ webapi               [ backend ]
|      |___ README.md
|      |___ runserver.py
|      |___ requirements.txt
|      |___ other artifacts ...
|
|___ README.md
```

2.3.2 Installation and Prerequisites

Installation is as simple as extracting the zip archive to somewhere in the disk. In general, Python-3 with pip as package manager and MongoDB-4 is needed to successfully serve the applications. The table below shows further required packages which are needed to be installed before attempting to run.

APPS	REQUIRED PACKAGES
webapp	Flask==1.1.2 requests==2.24.0 timeago==1.0.14
webapi	Flask==1.1.2 Flask-RESTful==0.3.8 Flask-PyMongo==2.3.0 Flask-Cors==3.0.8 flask-marshmallow==0.13.0 webargs==6.1.0 Scrapy==2.1.0 pycurl==7.43.0.5

Table-2: Required Package for Installation

It is always recommended to create virtual environments for the applications, two separate one for each of the applications is even better. However, to install the dependencies navigate to each of the application directories from command prompt and issue the following command.

```
pip install -r requirements.txt
```

2.3.3 Configure

There are few optional configurations which can be considered before running the application. Both applications take some input as a form of environment variables. If not supplied, then application will continue with the default values. In that case the availability of ports is a must have to successfully run the applications.

Configuration Parameters: Backend

```
SERVER_HOST=localhost
SERVER_PORT=5001
MONGO_URI=mongodb://localhost:27017/proxyAllyDB
```

Configuration Parameters: Frontend

```
SERVER_HOST=localhost
SERVER_PORT=5000
API_ROOT=http://localhost:5001/api/v1
```

2.3.4 How to Run

Navigate to the directory "webapi" and "webapp" consecutively from the command prompt and simply run the following command.

```
python runserver.py
```

To quick run both the applications a run script can be found in the corresponding application directories along with a README.txt file with an extended technical detail.

3 Technology Overview

3.1 MongoDB

MongoDB is a cross-platform NoSQL database program. It is a document-oriented database software which uses JSON-like documents with optional schemas. MongoDB is developed by MongoDB Inc. and licensed under the Server Side Public License (SSPL) [3]. It can be obtained from https://www.mongodb.com/try/download/

MongoDB has been used for storing and manipulating the data generated by the application. Since ProxyAlly highly depends on the scraped or API consumed data it is often difficult to decide on a common schema, at least for the proxy providers and proxies. Moreover, the application itself is not heavily transactional i.e. consistency problem is very less likely to happen. Hence to fit with this semi structured data and foster simplicity with platform independence, MongoDB is seemed to be the better choice compared to a relational one.

3.2 Python

Python is an interpreted, high-level, general-purpose programming language. Its language constructs and object-oriented approach aim to help programmers write clear, logical code for small and large-scale projects [4]. It is available at https://www.python.org/downloads/

Python's built-in high-level data structures, modularity and community driven open source libraries made it very attractive for Rapid Application Development. To serve the purpose of project ProxyAlly a nicely handled scraping library was a must have along with the support for REST API development. **Scrapy** and **Flask** packages (see below for more) have been used as a backbone of the whole backend correspondingly. Flask is even used in the frontend as well to act as a template server and sometimes as a mediator between the backend and UI. Last but not the least, deployment sometimes become a scary nightmare even after successful completion of a software project. Python is again preferred as the tool for this project to make deployment process simple and maintainable. Python-3 has been used as the python version.

3.2.1 Scrapy

Scrapy is a fast and high-level web crawling and web scraping framework, used to crawl websites and extract structured data from their pages [5].

ProxyAlly has highly utilized Scrapy to implement a bunch of spiders for couple of providers. This package is used for scraping, consuming json API, extracting RSS feeds and even to read text files. The exact package identifier for pip is "Scrapy==2.1.0".

3.2.2 Flask

Flask is a lightweight WSGI web application framework. It is designed to make getting started quick and easy, with the ability to scale up to complex applications [6]. It began as a simple wrapper around Werkzeug [19] and Jinja [18] and has become one of the most popular Python web application frameworks.

Besides scrapy ProxyAlly has highly utilized Flask as the backbone of REST API modules. The exact package identifier for pip is "Flask==1.1.2".

3.2.2.1 Flask-RESTful

Flask-RESTful is an extension for Flask that adds support for quickly building REST APIs. Flask-RESTful encourages best practices with minimal setup [7].

ProxyAlly has utilized this package to ensure minimal configuration for developing REST APIs. This package helps designing the resources in a object oriented manner. The package identifier for pip is "Flask-RESTful==0.3.8".

3.2.2.2 Flask-Marshmallow

Flask-Marshmallow is a thin integration layer for Flask and Marshmallow [20] that adds additional features to marshmallow, including URL and Hyperlinks fields for HATEOAS-ready APIs [8].

Flask-Marshmallow has been utilized for object serialization and deserialization including field validation and HATEOAS. The package identifier is "flask-marshmallow==0.13.0".

3.2.2.3 Flask-PyMongo

Flask-PyMongo bridges Flask and PyMongo and provides some convenience helpers [9]. PyMongo is the recommended way to work with MongoDB from Python [23].

ProxyAlly has used "Flask-PyMongo==2.3.0" package for communicating with the database.

3.2.2.4 Flask-Cors

Flask-Cors is another Flask extension for handling Cross Origin Resource Sharing (CORS), making cross-origin AJAX possible [10].

Since the frontend extensively uses jQuery to request the APIs, CORS support is a must have to allow requests from certain origin. ProxyAlly has used "Flask-Cors==3.0.8" for its purpose.

3.2.3 Webargs

"webargs" is a Python library for parsing and validating HTTP request objects, with built-in support for many popular web frameworks [11].

ProxyAlly has used "webargs==6.1.0" both exclusively and in cooperation with the Marshmallow Schema for input parsing and validation.

3.2.4 Pycurl & Requests

PycURL is a Python interface to libcurl. PycURL can be used to fetch objects identified by a URL from a Python program [12]. Requests is an elegant and simple HTTP library for Python, built for human beings as the documentation suggests [13].

ProxyAlly has used "pycurl==7.43.0.5" in backend and "requests==2.24.0" in the frontend sometimes to communicate with the proxy providers and ProxyAlly REST API endpoint correspondingly.

3.2.5 Timeago

A very simple python lib used to format datetime with time ago statement [14]. "timeago==1.0.14" has been used in server side of the frontend to implement filters for Jinja [18] template engine.

3.3 jQuery

jQuery is a fast, small, and feature-rich JavaScript library. It makes things like HTML document traversal and manipulation, event handling, animation, and Ajax much simpler with an easy-to-use API that works across a multitude of browsers. With a combination of versatility and extensibility, jQuery has changed the way that millions of people write JavaScript [15].

ProxyAlly is highly dependent on jQuery (version 3.3.1) both for consuming the APIs directly from the client side and user-friendly data visualization and interaction.

3.3.1 Datatables

DataTables is a plug-in for the jQuery Javascript library. It is a highly flexible tool, built upon the foundations of progressive enhancement, that adds all these advanced features to any HTML table [26].

ProxyAlly has used Datatables (version 1.10.21) very intensively in several pages where list of data is needed to display.

3.3.2 Timeago.js

"timeago.js" is just the JavaScript version of previously referred Timeago python package [14]. This jQuery plugin (version 1.1.0) is highly used to humanize the frontend for end user to ensure better user experiences [16].

3.3.3 Underscore.js

Underscore [17] is another jQuery plugin which has been utilized while the frontend needed to perform aggregation operations like groupBy, indexBy, countBy etc. Underscore version 1.9.1 has been used for the designated purpose.

3.4 Bootstrap

Bootstrap is a free and open-source CSS framework directed at responsive, mobile-first front-end web development [24].

This component is used to simplify the user interface development and to facilitate an appealing look to the application. The version 4.0 is used in the application

3.5 Font Awesome

Font Awesome is a font and icon toolkit based on CSS and LESS [25]. This tool is used to add some visual smoothness using set of related icons to the application. Font Awesome version 5 is used in this project.

4 Conclusion

The overall development experience of ProxyAlly was absolutely satisfying. Especially the opportunity to explore the python world to a further extent is worth stating. Working with MongoDB was another fun part which helped a lot to differentiate between relational and NoSQL databases in the context of a practical application. The well-defined set of requirements made it even helpful to deal with some misperceptions during the development period. However, as a concluding statement it can be confidently declared that, ProxyAlly is capable of aggregating proxies from heterogeneous sources, checking freshness, and last but not the least can validate proxies against restricted services with an extensible user-friendly web interface.

4.1 Future Scope

Due to **project scope** and several other constraints some features are scheduled for the next incremental releases. These features include:

- Implementing login feature, enabling proper administration
- Personalizing proxy management per user
- Implementing some statistical feature

Future releases can be tracked at https://github.com/shovradas/proxyally

References

[1] Richter D. (2020. May 25). “DBW Project” [PDF File]. Retrieved from https://www.tu-chemnitz.de/informatik/DVS/lehre/DBW/Projekt/DBW%20Project.pdf

[2] Richter D. (2020. June 27). “Questions and Answers concerning project task in Datenbanken und Web-Techniken 2020” [Text File]. Retrieved from https://www.tu-chemnitz.de/informatik/DVS/lehre/DBW/Projekt/DBW%20Project%20QA.txt

[3] MongoDB. (2020, June 28). Retrieved from https://en.wikipedia.org/wiki/MongoDB

[4] What is Python? Executive Summary. (2020, June 28). Retrieved from https://www.python.org/doc/essays/blurb/

[5] Scrapy 2.2 documentation. (2020, June 28). Retrieved from https://docs.scrapy.org/en/latest/

[6] Flask. (2020, June 28). Retrieved from https://palletsprojects.com/p/flask/

[7] Flask-RESTful — Flask-RESTful 0.3.8 documentation. (2020, June 28). Retrieved from https://flask-restful.readthedocs.io/en/latest/

[8] Flask + marshmallow for beautiful APIs. (2020, June 28). Retrieved from https://flask-marshmallow.readthedocs.io/en/latest/

[9] Flask-PyMongo. (2020, June 28). Retrieved from https://flask-pymongo.readthedocs.io/en/latest/

[10] Flask-CORS. (2020, June 28). Retrieved from https://flask-cors.readthedocs.io/en/latest/

[11] webargs. (2020, June 28). Retrieved from https://webargs.readthedocs.io/en/latest/

[12] PycURL Home Page. (2020, June 28). Retrieved from http://pycurl.io/

[13] Requests: HTTP for Humans™. (2020, June 28). Retrieved from https://requests.readthedocs.io/en/master/

[14] timeago . PyPI . (2020, June 28). Retrieved from https://pypi.org/project/timeago/

[15] jQuery. (2020, June 28). Retrieved from https://jquery.com/

[16] timeago.js. (2020, June 28). Retrieved from https://timeago.org/

[17] Underscore.js. (2020, June 28). Retrieved from https://underscorejs.org/

[18] Jinja — Jinja Documentation (2.11.x). (2020, June 28). Retrieved from https://jinja.palletsprojects.com/en/2.11.x/

[19] Werkzeug | The Pallets Projects. (2020, June 28). Retrieved from https://palletsprojects.com/p/werkzeug/

[20] marshmallow: simplified object serialization (2020, June 28). Retrieved from https://marshmallow.readthedocs.io/en/stable/

[21] proxy-checker-python/proxy_checker.py at master · ricerati/proxy-checker-python (2020, June 28). Retrieved from https://github.com/ricerati/proxy-checker-python/blob/master/proxy_checker/proxy_checker.py

[22] Proxy Anonymity Level. (2020, June 28). Retrieved from https://www.proxynova.com/proxy-articles/proxy-anonymity-levels-explained/

[23] PyMongo (2020, June 28). Retrieved from https://docs.mongodb.com/drivers/pymongo

[24] Bootstrap (front-end framework). (2020, June 28). Retrieved from https://en.wikipedia.org/wiki/Bootstrap_(front-end_framework)

[25] Font Awesome (front-end framework). (2020, June 28). Retrieved from https://en.wikipedia.org/wiki/Font_Awesome

[26] DataTables | Table plug-in for jQuery (2020, June 28). Retrieved from https://datatables.net/

Appendix

API Documentation

List of Endpoints

Here is an exhaustive list of all the endpoints. Please refer to the next pages for details.

RESOURCES	ENDPOINTS	
Configuration	GET	/configurations
	POST	/configurations
	GET	/configurations/{id}
	PUT	/configurations/{id}
	DELETE	/configurations/{id}
Provider	GET	/providers
	POST	/providers
	GET	/providers/{id}
	PUT	/providers/{id}
	DELETE	/providers/{id}
Proxy	GET	/proxies
	POST	/proxies
	GET	/proxies/{id}
	PUT	/proxies/{id}
	DELETE	/proxies/{id}
Testurl	GET	/testurls
	POST	/testurls
	GET	/testurls/{id}
	PUT	/testurls/{id}
	DELETE	/testurls/{id}

Endpoints’ Details

RESOURCE: Configuration --

GET /configurations		
Request	Headers	
	Path Params	
	Query Params	limit: int (optional), offset: int (optional)
	Sample Body	
Response	Status	200 (Success), 500 (Error)
	Sample Body	

```
{
   "count": 2,
   "items": [
      {
         "_links": {
            "self": {
               "href": "/configurations/5ef7aaee6d2a6b2019bda897",
               "title": "configuration detail"
            },
            "collection": {
               "href": "/configurations",
               "title": "list of configurations"
            }
         },
         "id": "5ef7aaee6d2a6b2019bda897",
         "maxAge": 1,
         "syncInterval": 10,
         "downloadDelay": 1,
         "proxyTestTimeout": 3,
         "status": false
      },
      {
         "_links": {
            "self": {
               "href": "/configurations/5ef7ab270df252f4d6930534",
               "title": "configuration detail"
            },
            "collection": {
               "href": "/configurations",
               "title": "list of configurations"
            }
         },
         "id": "5ef7ab270df252f4d6930534",
         "maxAge": 1,
         "syncInterval": 10,
         "downloadDelay": 1,
         "proxyTestTimeout": 3,
         "status": false
      }
   ]
}
```

POST /configurations		
Request	Headers	Content-Type: application/json
	Path Params	
	Query Params	

	Sample Body	{
		"syncInterval":10,
		"maxAge":1,
		"downloadDelay": 2,
		"status": true,
		"proxyTestTimeout": 3
		}

	Status	201 (Success), 500 (Error)
Response	Headers	location: {newly created resource url}

	Sample Body	{
		"_links": {
		"self": {
		"href": " /configurations/5ef8cbc1790518968552e08a",
		"title": "configuration detail"
		},
		"collection": {
		"href": "/configurations",
		"title": "list of configurations"
		}
		},
		"id": "5ef8cbc1790518968552e08a",
		"maxAge": 1,
		"syncInterval": 10,
		"downloadDelay": 2,
		"proxyTestTimeout": 3,
		"status": true
		}

GET /configurations		
Request	Headers	
	Path Params	id: string (required)
	Query Params	
	Sample Body	
Response	Status	200 (Success), 404 (Error), 500 (Error)

	Sample Body	{
		"_links": {
		"self": {
		"href": " /configurations/5ef7aaee6d2a6b2019bda897",
		"title": "configuration detail"
		},
		"collection": {
		"href": "/configurations",
		"title": "list of configurations"
		}
		},
		"id": "5ef7aaee6d2a6b2019bda897",
		"maxAge": 1,
		"syncInterval": 10,
		"downloadDelay": 1,
		"proxyTestTimeout": 3,
		"status": false
		}

PUT /configurations		
Request	Headers	Content-Type: application/json
	Path Params	id: string (required)
	Query Params	

	Sample Body	{
		"syncInterval":10,
		"maxAge":1,
		"downloadDelay": 2,
		"status": true,
		"proxyTestTimeout": 3
		}

Respons	Status	204 (Success), 404 (Error), 500 (Error)
	Sample Body	

DELETE /configurations		
Request	Headers	
	Path Params	id: string (required)
	Query Params	
	Sample Body	
Response	Status	204 (Success), 500 (Error)
	Sample Body	

RESOURCE: Provider ---

GET /providers		
Request	Headers	
	Path Params	
	Query Params	limit: int (optional), offset: int (optional), embed: bool (optional)
	Sample Body	
Response	Status	200 (Success), 500 (Error)

	Sample Body	{
		  "count": 2,
		  "items": [
		    {
		      "_links": {
		        "self": {
		          "href": "/providers/5ef6a930bd4fb14acc6014e3",
		          "title": "provider detail"
		        },
		        "collection": {
		          "href": "/providers",
		          "title": "list of providers"
		        }
		      },
		      "id": "5ef6a930bd4fb14acc6014e3",
		      "name": "Proxy11com Provider",
		      "baseAddress": "https://proxy11.com/api/proxy.json”,
		      "fetcher": "proxydashlistdownload",
		      "instructions": "TBA",
		      "updateAttempt": {
		        "type": "syncDB_funcTest",
		        "status": "partial",
		        "proxyCount": "221",
		        "message": "Error occured while updating database",

```
              "attemptDate": "2020-06-27 21:17:09.941000"
            }
        },
        {
            "_links": {
              "self": {
                "href": "/providers/5ef8d09836ce2e6ecc7de8d5",
                "title": "provider detail"
              },
              "collection": {
                "href": "/providers",
                "title": "list of providers"
              }
            },
            "id": "5ef8d09836ce2e6ecc7de8d5",
            "name": "Proxy11com Provider",
            "baseAddress": "https://proxylisten.de",
            "fetcher": "proxydashlistdownload",
            "instructions": "TBA",
            "updateAttempt": {
              "type": "syncDB_funcTest",
              "status": "partial",
              "proxyCount": "221",
              "message": "Error occured while updating database",
              "attemptDate": "2020-06-27 21:17:09.941000"
            }
        }
    ]
}
```

POST /providers		
Request	Headers	Content-Type: application/json
	Path Params	
	Query Params	syncTest: int {1, 2} (optional)

	Sample Body	{
		"name": "Proxy11com Provider",
		"baseAddress": "https://proxy-listen.de",
		"fetcher": "proxy11com",
		"instructions": "Some Instructions"
		}

	Status	201 (Success), 500 (Error)
Response	Headers	location: {newly created resource url}

	Sample Body	{
		"_links": {
		"self": {
		"href": "/providers/5ef8d10d790518968552e08c",
		"title": "provider detail"
		},
		"collection": {
		"href": "/providers",
		"title": "list of providers"
		}

```
    },
    "id": "5ef8d10d790518968552e08c",
    "name": "Proxy11com Provider",
    "baseAddress": "https://proxy-listen.de",
    "fetcher": "proxylisten",
    "instructions": "Some Instructions",
    "updateAttempt": {
        "type": "fetch",
        "status": "failed",
        "message": "Error occured while fetching",
        "attemptDate": "2020-06-28 19:19:09.510000"
    }
}
```

GET /providers

Request	Headers	
	Path Params	id: string (required)
	Query Params	embed: bool (optional)
	Sample Body	
Response	Status	200 (Success), 404 (Error), 500 (Error)

Sample Body

```
{
    "_links": {
        "self": {
            "href": "/providers/5ef8d09836ce2e6ecc7de8d5",
            "title": "provider detail"
        },
        "collection": {
            "href": "/providers",
            "title": "list of providers"
        }
    },
    "id": "5ef8d09836ce2e6ecc7de8d5",
    "name": "Proxy11com Provider",
    "baseAddress": "https://proxylisten.de",
    "fetcher": "proxydashlistdownload",
    "instructions": "TBA",
    "updateAttempt": {
        "type": "syncDB_funcTest",
        "status": "partial",
        "proxyCount": "221",
        "message": "Error occured while updating database",
        "attemptDate": "2020-06-27 21:17:09.941000"
    }
}
```

PUT /providers		
Request	Headers	Content-Type: application/json
	Path Params	id: string (required)
	Query Params	

	Sample Body	{
		"name": "New Provider",
		"baseAddress": "https://proxy11.com/api/proxy.json",
		"fetcher": "proxydashlistdownload",
		"instructions": "Changed Instruction"
		}

Respons	Status	204 (Success), 404 (Error), 500 (Error)
	Sample Body	

DELETE /providers		
Request	Headers	
	Path Params	id: string (required)
	Query Params	
	Sample Body	
Response	Status	204 (Success), 500 (Error)
	Sample Body	

RESOURCE: Proxy --

GET /proxies		
Request	Headers	
	Path Params	
	Query Params	limit: int (optional), offset: int (optional) , embed: bool (optional)
	Sample Body	
R	Status	200 (Success), 500 (Error)

Sample Body

```
{
  "count": 2,
  "items": [
    {
      "_links": {
        "self": {
          "href": " /proxies/5ef6b37db36260771983cef5",
          "title": "proxy detail"
        },
        "provider": {
          "href": " /providers/5ef6a930bd4fb14acc6014e3",
          "title": "provider detail"
        },
        "collection": {
          "href": " /proxies",
          "title": "list of proxies"
        }
      },
      "id": "5ef6b37db36260771983cef5",
      "providerId": "5ef6a930bd4fb14acc6014e3",
      "ip": "159.8.114.37",
      "port": 80,
      "funcTestDate": "2020-06-27T21:16:51.074000",
      "lastFoundDate": "2020-06-27T21:17:09.915000",
      "discoveredDate": "2020-06-27T04:48:29.128000",
      "anonymity": "Elite"
    },
    {
      "_links": {
        "self": {
          "href": " /proxies/5ef6b37db36260771983cef7",
          "title": "proxy detail"
        },
        "provider": {
          "href": " /providers/5ef6a930bd4fb14acc6014e3",
          "title": "provider detail"
        },
        "collection": {
          "href": " /proxies",
          "title": "list of proxies"
        }
      },
      "id": "5ef6b37db36260771983cef7",
      "providerId": "5ef6a930bd4fb14acc6014e3",
      "ip": "169.57.1.85",
      "port": 8123,
      "funcTestDate": "2020-06-27T21:16:52.027000",
      "lastFoundDate": "2020-06-27T21:17:09.936000",
      "discoveredDate": "2020-06-27T04:48:29.130000",
      "anonymity": "Elite"
    }
  ]
}
```

POST /proxies

Request	Headers	Content-Type: application/json
	Path Params	
	Query Params	

	Sample Body	{
		"providerId": "5ef6a930bd4fb14acc6014e3",
		"ip": "127.0.0.1",
		"port": 8080,
		"anonymity": "Elite"
		}

	Status	201 (Success), 500 (Error)
Response	Headers	location: {newly created resource url}

	Sample Body	{
		"_links": {
		"self": {
		"href": " /proxies/5ef8d560bfd5ed38ba14a4cc",
		"title": "proxy detail"
		},
		"provider": {
		"href": " /providers/5ef6a930bd4fb14acc6014e3",
		"title": "provider detail"
		},
		"collection": {
		"href": " /proxies",
		"title": "list of proxies"
		}
		},
		"id": "5ef8d560bfd5ed38ba14a4cc",
		"providerId": "5ef6a930bd4fb14acc6014e3",
		"ip": "127.0.0.1",
		"port": 8080,
		"anonymity": "Elite"
		}

GET /proxies

Request	Headers	
	Path Params	id: string (required)
	Query Params	embed: bool (optional)
	Sample Body	
Response	Status	200 (Success), 404 (Error), 500 (Error)

	Sample Body	{
		"_links": {
		"self": {
		"href": " /proxies/5ef6b37db36260771983cef5",
		"title": "proxy detail"
		},

		"provider": {
		"href": " /providers/5ef6a930bd4fb14acc6014e3",
		"title": "provider detail"
		},
		"collection": {
		"href": " /proxies",
		"title": "list of proxies"
		}
		},
		"id": "5ef6b37db36260771983cef5",
		"providerId": "5ef6a930bd4fb14acc6014e3",
		"ip": "159.8.114.37",
		"port": 80,
		"funcTestDate": "2020-06-27T21:16:51.074000",
		"lastFoundDate": "2020-06-27T21:17:09.915000",
		"discoveredDate": "2020-06-27T04:48:29.128000",
		"anonymity": "Elite"
		}

PUT /proxies		
Request	Headers	Content-Type: application/json
	Path Params	id: string (required)
	Query Params	

	Sample Body	{
		"providerId": "5ef6a930bd4fb14acc6014e3",
		"ip": "127.0.0.1",
		"port": 8080,
		"anonymity": "Elite"
		}

Respons	Status	204 (Success), 404 (Error), 500 (Error)
	Sample Body	

DELETE /proxies		
Request	Headers	
	Path Params	id: string (required)
	Query Params	
	Sample Body	
Response	Status	204 (Success), 500 (Error)
	Sample Body	

RESOURCE: Testurl --

GET /testurls		
Request	Headers	
	Path Params	
	Query Params	limit: int (optional), offset: int (optional) , embed: bool (optional)
	Sample Body	
Response	Status	200 (Success), 500 (Error)

	Sample Body	{
		"count": 2,
		"items": [
		{
		"_links": {
		"self": {
		"href": " /testurls/5eece56c44c66a36f015bdbe",
		"title": "test-url detail"
		},
		"collection": {
		"href": " /testurls",
		"title": "list of test urls"
		}
		},
		"id": "5eece56c44c66a36f015bdbe",
		"url": "http://www.hippo.com/file.txt",
		"description": "na"
		},
		{
		"_links": {
		"self": {
		"href": "/testurls/5eed15028d1adb77a98e6158",
		"title": "test-url detail"
		},
		"collection": {
		"href": " /testurls",
		"title": "list of test urls"
		}
		},
		"id": "5eed15028d1adb77a98e6158",
		"url": "http://www.filetuc.com/file.txt",
		"description": "NA"
		}
		]
		}

POST /testurls

Request	Headers	Content-Type: application/json
	Path Params	
	Query Params	

	Sample Body	{
		  "url": "http://www.filetuc.com/file.txt",
		  "description": "Some Description"
		}

	Status	201 (Success), 500 (Error)
Response	Headers	location: {newly created resource url}

	Sample Body	{
		  "_links": {
		    "self": {
		      "href": " /testurls/5ef8d763bfd5ed38ba14a4cd",
		      "title": "test-url detail"
		    },
		    "collection": {
		      "href": " /testurls",
		      "title": "list of test urls"
		    }
		  },
		  "id": "5ef8d763bfd5ed38ba14a4cd",
		  "url": "http://www.filetuc.com/file.txt",
		  "description": "Some Description"
		}

GET /testurls

Request	Headers	
	Path Params	id: string (required)
	Query Params	embed: bool (optional)
	Sample Body	
Response	Status	200 (Success), 404 (Error), 500 (Error)

	Sample Body	{
		  "_links": {
		    "self": {
		      "href": " /testurls/5eed15028d1adb77a98e6158",
		      "title": "test-url detail"
		    },
		    "collection": {
		      "href": "http://localhost:5001/api/v1/testurls",
		      "title": "list of test urls"
		    }
		  },
		  "id": "5eed15028d1adb77a98e6158",
		  "url": "http://www.filetuc.com/file.txt",
		  "description": "NA"

		}

PUT /testurls		
Request	Headers	Content-Type: application/json
	Path Params	id: string (required)
	Query Params	

	Sample Body	{
		"url": "http://www.filetuc.com/file.txt",
		"description": "Some Description"
		}

Respons	Status	204 (Success), 404 (Error), 500 (Error)
	Sample Body	

DELETE /testurls		
Request	Headers	
	Path Params	id: string (required)
	Query Params	
	Sample Body	
Response	Status	204 (Success), 500 (Error)
	Sample Body	

-- End of Document --