PUBLIC

CounteR

Privacy-First Situational Awareness Platform for Violent Terrorism and Crime Prediction, Counter Radicalisation and Citizen Protection

Project Acronym	CounteR
Project Full Title	Privacy-First Situational Awareness Platform for Violent Terrorism and Crime Prediction, Counter Radicalisation and Citizen Protection
Grant Agreement no	101021607
Project Duration	36 months (starting May 1st 2021)

Deliverable number 3.2

Blogs, forums and website data collection

Work Package	WP3 - Data Acquisition & Management
Task	Task 3.2 - Collection of data from blogs and forums
Lead Beneficiary	CINI
Due Date	30/04/2022
Submission Date	17/05/2022
Deliverable Status	1.0
Deliverable Type	DEM (Demonstrator)
Dissemination Level	PU
Document Name	D3.2 – Blogs, forums and website data collection

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 101021607

Disclaimer

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 101021607

Editors

First name	Surname	Beneficiary
Sandra	Cardoso	INS
Jose Miguel	Sanchíz	INS

Contributors

First name	Surname	Beneficiary
Georgi	Kostadinov	IMA
Mariano	Martín Zamorano	ETI

Reviewers

First name	Surname	Beneficiary
Marco	Anisetti	UMIL
Adrian	Onu	AST

History of Changes

Version	Date	Change	Modified by
0.1	7-01-2022	Structure	Sandra Cardoso
0.2	17-01-2022	Draft version	Sandra Cardoso, Jose Miguel Sanchíz
0.3	28-03-2022	Ready for internal review	Sandra Cardoso, Jose Miguel Sanchíz
0.4	30-04-2022	External version ready for review	Sandra Cardoso, Jose Miguel Sanchíz
0.5	29-04-2022	External review	Marco Anisetti, Adrian Onu
0.6	1-04-2022	Revised version	Sandra Cardoso, Jose Miguel Sanchíz
0.7	14-04-2022	Second revision ready for submission	Sandra Cardoso, Jose Miguel Sanchíz
1.0	28-04-2022	Final Version for review	Sandra Cardoso
1.1	13-05-2022	Final Version after partner review	Sandra Cardoso

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 101021607

Table of Contents

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 101021607

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 101021607

Executive Summary

The D3.2 is part of the WP3 Data Acquisition & Management focused specifically in the collection of data obtained in blogs, forums and websites that are not social media or deep or dark web. This document will detail the sources, the tools developed and their use so that they can be fully used and integrated into the Counter platform.

It starts by explaining the task as a part of the WP3 as well as the methodology and architecture inherited from D3.1. After that, it details the web collection components and how the solution was approached according to their specific peculiarities. Lastly, it will explain the APIs and give examples about their usage to collect data.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 101021607

List of Abbreviations

Abbreviation	Explanation
ACK	Acknowledge
API	Application Programming Interface
D	Deliverable
DoA	Description of Action
ETL	Extract Transform Load
EU	European Union
GA	Grant Agreement
GDPR	General Data Protection Regulation
HTML	Hypertext Markup Language
ID	Identification
JSON	JavaScript Object Notation
LEA	Law Enforcement Agency
PHP	Hypertext Preprocessor
Phpbb	PHP Bulletin Board.
pub/sub or pubsub	Publisher/Subscriber
Redis	Remote Dictionary Server
REST	Representational State Transfer
T	Task
URL	Uniform Resource Locator
WP	Work Package

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 101021607

List of Figures

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 101021607

List of Tables

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement no. 101021607

1 Introduction

The goal of this deliverable is to describe the software components developed for the collection of data obtained in blogs, forums, and websites.

General description of WP3 collection architecture. In Chapter2, it shall briefly describe the overall architecture of WP3 and how T3.2 integrates within, from a general perspective. The specifics and greater discussion about the methodology and architecture can be found in D3.1.

Blogs, forums, and websites collectors. Secondly, it will describe the components developed. The Collectors access to web sources to carry out a data collection campaign retrieving data suitable for the CounteR needs. There had to be an analysis of each source in order to decide the method used for the collection of the data. The sources have their own structure and presents limitations of their own. Most of the sources do not offer an official API, since they are basically webpages. It was also noticed that some sources did not use a user account in order to gain access to the content, while others do have login restrictions. The collectors receive as input the collection request (e.g., keywords to be searched on the forum, blogs or websites) and any other parameter specific for the source. Chapter 3 will provide the documentation for these sources' collections.

APIs and usage examples. APIs description and usage examples are presented in the deliverable in order to describe how to use the collectors and how to request a data collection campaign.

1.1 Relation with other tasks and deliverables

As part of WP3, all the data collectors are the base of several WPs such as WP4 and WP6 and are intimately related to others such as WP2. All this can be seen in Figure 1 below.

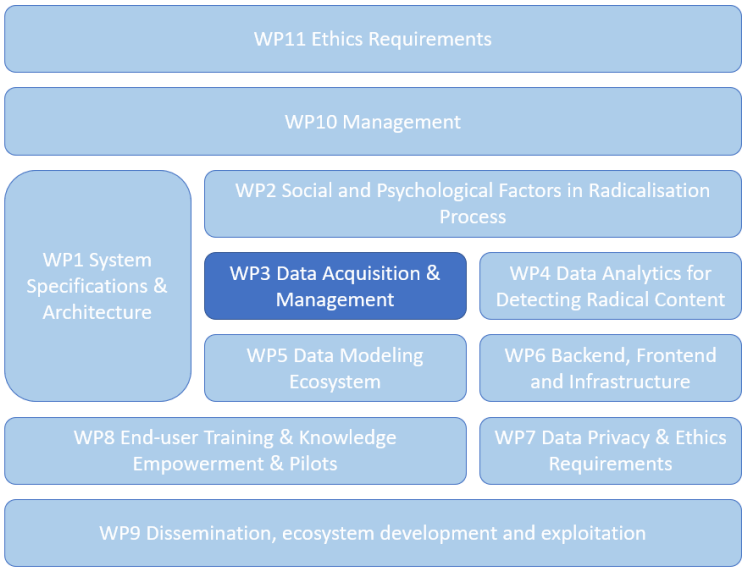

Figure 1 - Relationship between Working Packages

The scope of this deliverable is to design a Web Collector that can be easily integrated with the design defined for the Data Collectors in WP3 T3.1. All tasks from WP3 will be fully integrated with

the infrastructure, frontend and backend defined for the system in WP6. As well as obtain the identified keywords provided by WP2 to procure the datasets for the training of the models in WP4. Lastly, it is important to highlight that by WP6 all the data collected in WP3 will be sanitized before carrying out any analytics by WP4. In Figure 2 we can see an abstract data flow for data collection considering the relevant Work Packages.

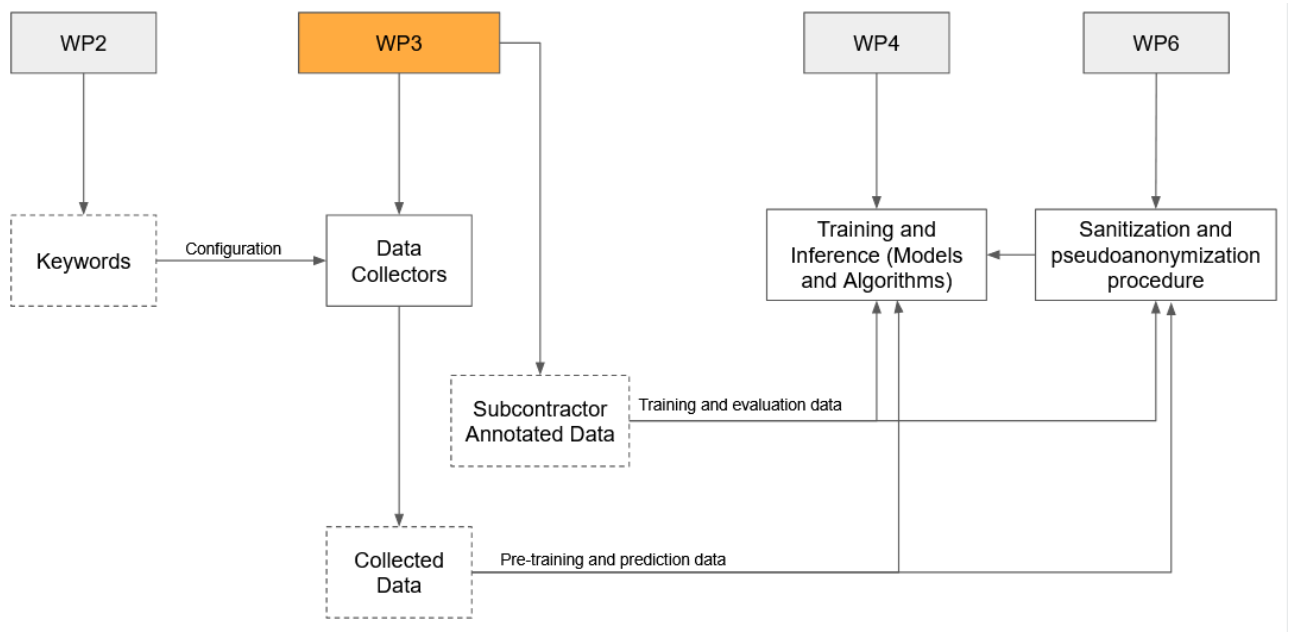

Figure 2 - Overall abstract data view in relation with the relevant WPs[1]

This deliverable focuses on tools development; therefore, it refers primarily to deliverables detailing requirements to be considered.

To decide the sources that had to be developed for this task, the insight was taken from the LEA requirements presented in the D1.1. Also, at a technical level, the D1.2 presented the LEA requirements.

This task inherits the methodology and architecture from the D3.1, where it is defined the general architecture of the WP3 and its submodules in order to be flexible for integration. Therefore, the Data Collector module requires a full technical understanding of D3.1.

Table 1 - Relation to other deliverables

Deliv.#	Deliverable Title	Nature of Relation
D1.1	LEA requirements, use cases and scenarios	D1.1 presents requirements as perceived by the LEAs including, use cases and scenarios.
D1.2	Technical Requirements	D1.2 presents technical requirements showing a direct impact for the collection architecture.
D3.1	Social Media Data collection	D3.1 presents the methodology and architecture for the integration of the tools developed in this task.

[1] *Source of the image is D3.1. Social Media Data Collector*

1.2 Document objectives

The main objective of this deliverable is to document the developed tool and the API usage so it can be integrated into the common architecture defined in D3.1 for the WP3 collectors. More specifically:

- **Explanation of the role of the Web Data Collector on the full architecture.** Based on the proposed architecture in D3.1, this document presents the web collection module as part of the architecture.
- **Definition and implementation of Web Data Collectors.** The document focused on blogs, forums, and websites collection processes and presents the documentation of the relevant components. The scope is to document the developed components in order to ease the integration process with the rest of the CounteR platform.

1.3 Deliverable structure

This deliverable is structured as follows:

1. **Introduction** – describing the focus of the deliverable in a nutshell;
2. **Collection Architecture** – focused on the fit of the Web Collector module inside the complete data collection architecture of WP3;
3. **Web Data Collection Components**– documenting the components developed for blogs, forums and webpages data collection;
4. **API documentation** – showing the data collection API pattern and technical details on the social media data collection API designed;
5. **Usage Examples** – presenting usage examples as use cases.

1.4 Methodology

The content of this document has been elaborated based on the discussions conducted between the technical partners from WP3. The DoA included in the GA was studied and discussed to decide the approach that would have to be taken in order to achieve the objectives of CounteR.

The subject of debate was to deliver an architecture to be used as a framework that was modular enough that could be flexible and easily integrable with all of the parts. The overall architecture of the WP3 was agreed upon and described in D3.1.

Using this framework, a more detailed analysis was done for all the tools, in this case the Web Collector. The internal modules of the tool should use the framework, but with the flexibility of adapting it according to the needs of each specific source.

From D1.1 we learned that the Web Collector needs to be able to gather publicly available data and a first list of sources was provided. Since then, a larger list of relevant sources has been supplied in order of relevance by some LEAs and ISP. Because of the length of this list and the time that

consumes the technical evaluation of each source, only the most relevant sources were analysed. This scrutiny made possible the definition of the architecture and functionality of the Web Collector and the further development and testing of this.

Also due to data protection law, not all the sources provided are available for scraping. A legal review and analysis of the Terms and Conditions of each website has to be performed before being eligible to be incorporated in the Counter platform.

2 Collection architecture

2.1 Overview

In D3.1 - Social Media Data collection it was defined the overall logical architecture for WP3 based on requirements enumerated in D1.2 - Technical Requirements such as modularity (C6), parallel campaigns (P1, P10) and security best practices (S3, S6, S7, S8, S9, S12, S13, S15). Figure 3 shows the ETL architecture of the data flow from collecting, transforming, and storing to the Data Lake for its later use.

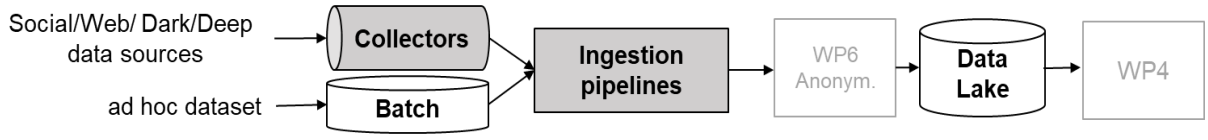

Figure 3 - Overview of the data flow architecture[2]

The most relevant modules of this architecture for this particular task are:

- **Collectors**: Modules specialized in collecting data from different data sources in which case we centre the attention specifically to the Web Collector that scrapes web sites, blogs, and forums.
- **Ingestion pipelines:** a set of procedures called pipelines to treat data after collection to prepare raw data for the rest of the CounteR platform.

Each of the tasks will be responsible for the collectors and ingestion pipelines needed for their sources. Figure 4 shows a more zoom in view of all the modules from the different tasks of WP3 as envisioned in D3.1.

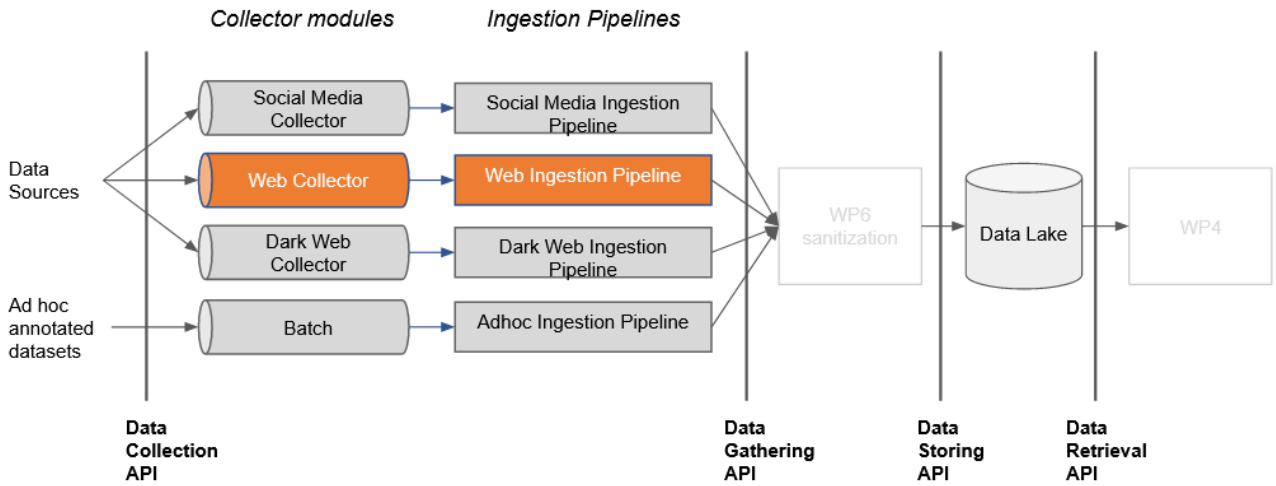

Figure 4 - Architecture and APIs. In Orange the components that are part of this deliverable.

[2] *Source of the image is D3.1. Social Media Data Collector*

2.2 API module

The Web Collection API is the orchestrator of the jobs that have to be performed by the Collector. This module is in charge of the inbound and outbound communication between the general Data Collector API and the Web Collectors. It has several endpoints; for example, to create jobs and notify of the status of a specific job. A detailed explanation of the API will be described in Chapter 4 API documentation.

2.3 Collectors

Having done the analysis of the different sources, we found that we had two different scenarios. The first one is the small pool of webpages that offers a public API. The second one is the vast majority of blogs, forums, and webpages that do not offer an API to gather data.

In the case of sites with public APIs, Reddit was selected as a source since it is a relevant and very popular forum that is commonly requested for analysis given its varied content. For this, a specific collector and ingestion pipeline have to be developed to deal with the peculiarities of the requests and the data structure obtained from its API.

On the other hand, for the rest of webpages that do not provide an API, we would have to create custom-made scrapers for each one of the sources.

The webpages can be created under numerous languages, protocols, and tools making an infinite possibility of structures and content interfaces. This would make it almost impossible to have one generic scraper that would be able to deal with any format of webpage that exists over the internet and detect and identify the relevant data requested. However, as part of the innovation for T3.2 we have decided to create only one scraper that can be dynamically configured to work with a big majority of blogs, news, forums, and webpages.

What this means is that the scraper will not be for any specific website, but instead will receive as an input parameter the configuration of the page it will scrape. The addition of new webpages will require the analysis of the webpage beforehand by someone with basic HTML technical knowledge to create the configuration JSON needed for the purpose. A detailed explanation of this procedure and the JSON structure will be explained in Chapter 3 Web Data Collectors.

Figure 5 shows an abstract view of the internal architecture for the Web Collector implemented in this deliverable.

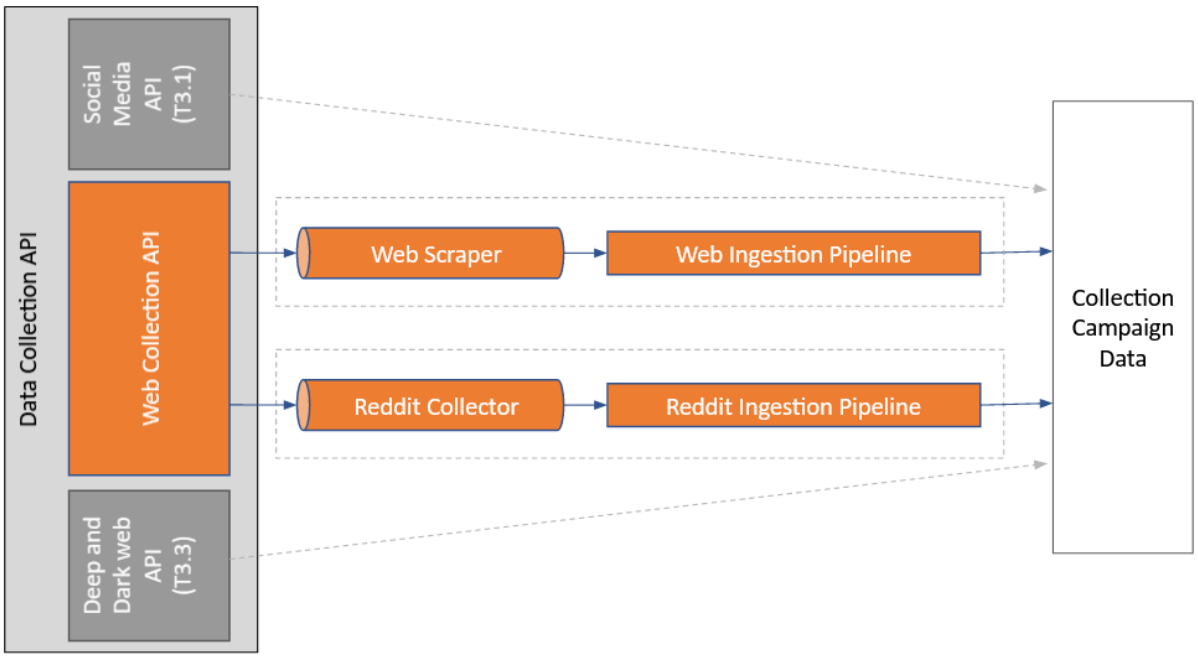

Figure 5 - Web Collector

3 Web Data Collectors

3.1 Overview

The final internal architecture of the Web Data Collector can be drawn as shown in **Error! Reference source not found.**.

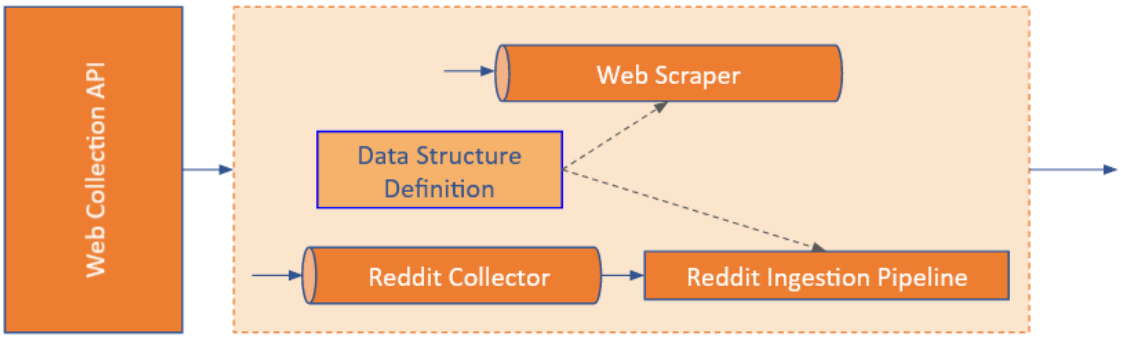

Figure 6 - Architecture of Data Collector

In the following, we describe the components of the Data Collector detailing their peculiarities and how we implement them considering the different sources examined in this deliverable.

3.2 Data Collectors

In this chapter we will describe both types of data collectors that have been developed for this tool. On one side, the Web Scraper for generic webpages and secondly, the API collector for Reddit.

3.2.1 Web Scraper (generic web page)

The main problem with the forum scrapers is that they don't provide an official API and we can't create a simple API REST service to request information. In our case, we needed to develop a web scraper crawler. This scraper was built using a library that helps to connect to a PhpBB[3] forum and to extract the information. The last step is to connect with this service using a headless browser, in this case Chromium[4], with the Puppeteer[5] library and extract the needed information from the HTML tags. Another important necessity is to get the pagination information in order to be able to obtain all the different posts and get all the comments related to the requested keyword.

In the first iteration of this task, our intention was to have independent scrapers for each webpage since all of them have their peculiarities and specific formats. Given the large number of sources provided, this task became too complicated to develop and maintain in the future for it to be scalable. But having done the analysis of the first webpages, we could see that there was the possibility of developing a more neutral web scraper. This scraper has a separate endpoint to upload

[3] https://www.phpbb.com/
[4] https://www.chromium.org/chromium-projects/
[5] https://pptr.dev/
https://github.com/puppeteer/puppeteer/
https://developers.google.com/web/tools/puppeteer/

new website configurations and, thanks to this new configuration, change its behaviour and be able to scrap the information needed for this new website. The endpoint accepts a JSON configuration object. See an example of the configuration in Appendix 8.1.

In order to have asynchronous requests between the API and the scraper, the communication is done through pubsub messages with Redis[6]. The Redis database is also used to manage the status of the different jobs of the scraper service and to temporarily store the scraped data until retrieved at the end of the job.

3.2.2 API collector (Reddit)

For the Reddit scraper we use the official API from Reddit and we are able to do a search directly to the Reddit API[7]. In this case we are exposing an endpoint that receives by parameters the Keyword to search in Reddit.

Table 2 - Official Reddit Data API used

Reddit Data API v3	Description
`https://www.reddit.com/api/search`	Returns a list of reddits and subreddits that contain a provided keyword.

Reddit pipeline: Reddit Data API provides as output JSON objects that are structured depending on the request. Our Reddit pipeline parses the JSON object in order to: i) reorganize the structure if needed, ii) retrieve reddits, subreddits and messages to be downloaded and download them. The final JSON with messages and users' information is parsed and passed to the main API caller.

3.3 Pipeline Ingestion

The Ingestion Pipeline component described in D3.2 is different in this task since we don't need to have an independent Ingestion Pipeline for each source. After the decision of having a unique scraper for all the webpages, the acquisition of the data is done in a single manner and the structure of the gathered data can be defined by ourselves from the beginning in a way that fits the requirements of raw data for Counter's ingestion.

This approach is easy to maintain since the Data Structure will be defined in the starting point of the Web Data Collector and be used all throughout the flow. In the case that the raw data structure requirements change over time, the modifications that should be done inside the Ingestion Pipeline described in D3.2's architecture would be similar to the modifications that would have to be realized under this approach.

On the other hand, for the purpose of this project we will only use one source that provides a public API and that is Reddit. The data received from it will have the specific format that the API delivers. So, in this case, a Reddit Ingestion Pipeline has been developed in order to transform the data into

[6] https://redis.io/
https://redis.com/redis-best-practices/communication-patterns/pub-sub/
[7] https://www.reddit.com/dev/api/
Specific description available at: https://www.reddit.com/dev/api#GET_search

the RAW data that can be ingested by Counter. Having said this, the JSON format used as the final data structure of the Ingestion pipeline will be the same as the data structure initially defined for the web scrapers. In this manner, the maintenance of this code is simpler in case there would be a need for modifications.

3.4 Data structure

As explained above, in this task we have identified two scenarios. On one hand, we have the sources where the data has to be obtained manually with ad hoc scraping. In this case, since we will have a unique collector and it has been completely developed from scratch by ourselves, we will only have one data structure for all the webpages.

On the other hand, we have the sources that give access to their official API in which we obtain the data with the structure given by the API directly. For these cases, we will create a specific ingestion pipeline for it.

In all cases, the format used for the data structure is a JSON. Examples on the data structure is available in Appendix A and an example in Appendix B.

4 API documentation

4.1 Data Collection API

The Data Collection API module contains the API to trigger and handle WP3 data collection campaigns. It is designed in order to have a uniform API structure for all the collectors based on the REST paradigm. This API module will be developed and detailed in the framework of WP3 T3.1 deliverable.

4.2 Web Data Collection API

Our web data collection API exposes different endpoints to manage the different options of the scraper. We have configuration related endpoints and job-related endpoints.

4.2.1 Web scraper

Our web data collection API exposes different endpoints to manage the different options of the scraper. We have configuration related endpoints and job-related endpoints.

Configuration endpoints

As mentioned above, a configuration file has to be created for each source. The format and content of the file is explained in Appendix A, section 8.1. For creating this file, it is necessary to have some technical knowledge of HTML to analyse the webpage and transfer the findings into the JSON configuration file.

We have two endpoints for working with the different web configurations of the scraper. One to create/update a JSON configuration for one platform (website) and another one to get the JSON configuration of a saved platform.

POST /api/v1/config/{name_of_the_platform}

This is the endpoint to save or update a configuration object. We need to send a JSON object with the fields. Please, find the JSON scheme of an example in Appendix B

GET /api/v1/config/{name_of_the_platform}

With this endpoint we can get the configuration object of a platform. The object in the response is the same that we have in the POST request

POST /api/v1/{name_of_the_platform}

This endpoint will create a new scraping job for a platform. We need to have the configuration for this platform before we´re able to send a job to the scraper. If the scraper doesn´t have the configuration for this platform the response will be a 404 error.

In the body of the request, we need to send an object with these fields, please find this JSON scheme in Appendix A:

The response of this request will be a JSON object that has only one field:

projectId: this is the ID of the job inside the scraper scope. We will need this ID to view the status of this job or to download the scraped information.

GET /api/v1/status/{projectId}

This endpoint will query the status of the job that we send in the request. The result JSON object will contain these fields, please find these fields in the Appendix A:

GET /api/v1/data/{projectId}

This endpoint will return the scraped information of the job that we send in the request. The result will be an array of objects with the scraped information. Every object has the next fields, please find the JSON scheme of this in Appendix A and an example in Appendix B

4.2.2 Reddit

This is the main endpoint to create a Reddit job

POST /api/v1/reddit/newjob

The main endpoint to create a Reddit scraper work needs a JSON object like this. Please, find a JSON scheme in Appendix A.

5 Usage Examples

Use cases of data collections describing how to use APIs. With some examples of data retrieved.

5.1 Web scraper keyword search

Let's consider a client asking for a collection campaign focused on collecting a list of messages of a given keyword. As an example, let us consider as target keyword "Paris" and the platform an extreme right blog called *Egalite et Réconciliation*[8], which we renamed as *egaliteetreconciliation* in the configuration.

The client calls the API for */api/v1/egaliteetreconciliation/* indicating Paris as the requested keyword in the request body. The call should include the hook details on where to notify the collection completion. The client receives back as ACK the project ID to be used, for instance, to monitor the campaign status and as a way to retrieve data when available.

When the collection campaign is completed, the client receives on the provided hook a notification and can receive the collected data that is structured as a folder called with the campaign ID and compressed into one or more files depending on the size. The folder in this example contains JSON files with a list of messages that contains these keywords and the author of these messages. Please find the JSON scheme of this configuration object in Appendix A and an example in Appendix B.

5.2 Reddit keyword search

Let's consider a client asking for a collection campaign focused on collecting a list of messages of a given keyword. As an example, let us consider as the target keyword "Paris".

The client calls the API for */api/v1/reddit/newjob* indicating Paris as the requested keyword in the request body. The call should include the hook details on where to notify the collection completion. As an ACK, the client receives the project ID assigned, this should be used for instance to monitor the campaign status and as a way to retrieve data when available.

When the collection campaign is completed, the client receives on the provided hook a notification and can receive the collected data that is structured as a folder called with the campaign ID and compressed into one or more files depending on the size. The folder in this example contains JSON files with a list of messages that contains this keyword and the author of these messages.

Please find the JSON scheme of this configuration object in Appendix A and an example in Appendix B.

[8] https://www.egaliteetreconciliation.fr/

6 Security and Ethics

The collector described in this deliverable collects only publicly available data that its procurement is compliant with the GDPR.[9]

For the sources that offer public API, such as Reddit, the usage of this provided API is compliant by design with the provider's terms of usage. The tool will not collect anything else or in this case use any other method of acquisition but the provided by the source.

In the case of webpages using the crawler for scraping data, on account of GDPR the tool will not acquire data from pages that the user and the site do not give their explicit consent to the acquisition of their data by third parties. This is not only related to the data being publicly available or the sites not being password-restricted, but the terms and conditions of the webpage.

Therefore, it is imperative that before doing the technical analysis of a webpage for the scraper, the Consortium needs to use an appropriate legal basis for scraping or check whether the user's consent extends to cover the scraping activities. Only then, the website should be incorporated to the sources of the Impetus project.

As stated in D3.1. "The collection components will not be used as independent stand-alone tools. They will be wrapped within the CounteR platform which is responsible to provide most of the security and privacy features such as authentication, authorization, secure channels, and privacy integrity preserving storage of the collected data to name but a few. Such security and privacy features will be verified and implemented in the framework of integration between WP3 and WP6.

When integrated by WP6, the WP3 the [..] collectors will be compliant with the Security requirements presented in D1.2 (see details in Section 2) and will fully consider the data protection requirements of D7.1."[10]

[9] *DIRECTIVE (EU) 2016/680 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016 on the protection of natural persons with regard to the processing of personal data by competent authorities for the purposes of the prevention, investigation, detection or prosecution of criminal offences or the execution of criminal penalties, and on the free movement of such data, and repealing Council Framework Decision 2008/977/JHA* https://eur-lex.europa.eu/eli/dir/2016/680/oj/
REGULATION (EU) 2016/679 OF THE EUROPEAN PARLIAMENT AND OF THE COUNCIL of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation) http://data.europa.eu/eli/reg/2016/679/oj/

[10] *Source D3.1. Social Media Data Collector*

7 Conclusions

In this deliverable, we presented the design and development of a modular and flexible tool that is able to acquire data from diverse forums, blogs, and new webpages. Using the proposed architecture defined in D3.1, the Web Collector resolves the requirements stated in D1.1 and can be easily integrated with the other modules from tasks of WP3 and overall, to the CounteR Platform.

Since most of the sources for this task are webpages, blogs, and forums that do not provide a public API, it was very important to have a tool that was flexible enough to acquire data from most webpages without too much hassle and in the most dynamic possible way.

Adding new sources to Counter's collector will be easier with the method described in this document than having a separate scraper for each new source. Since the scraper is open to any kind of input configuration given, there is no need for downtime of the system to add new sources. It will only require the analysis of the specific webpage and the addition of the configuration to the main Counter's configuration database in order for the Web Collector to accept and start acquiring data from this source.

Another advantage of this approach is that if a webpage changes the format, structure or content, there would be no need for downtime of the whole Web Collector system in order to fix it. It would only require updating the configuration in the database.

There are some limitations of sources that can be acquired due to security reasons:

- For the purpose of this project, abiding by the GDPR law, the tool does not scrape webpages that require user credentials to access.
- There are some pages that detect the use of proxies and block the access. The proxy use in the configuration of the webpage should be taken into consideration in this and it will be the responsibility of the users to agree on the usage of a scraper without a proxy.
- Webpages, forums, blogs, and news pages that do not have a Search feature available for us to emulate the search of the requested keywords cannot be scrapped.

There could arise some other technical problems in some webpages that will not be dealt with for the scope of this project:

- Use of captchas to access the data.
- Some interstitial ads or pop-ups that block access until further interaction from the user.
- The data that is not rendered in the webpage cannot be extracted. This could be the case of some hidden data behind JavaScript requests (i.e. PhantomJS).

In summary, the purpose of this document is to describe how to use our collectors with some usage examples and some appendixes with relevant JSON schemas.

8 Appendix A

8.1 Scrapper's configuration object example

```
{
 "platform": String,
 "settings": {
   "url": String,
   "proxy": String,
   "waitUntil": String,
   "search": Boolean,
   "searchParameters": {
     "searchField": String,
     "searchAction": String
   },
   "waitForResults": String,
   "resultsPerPage": Int,
   "waitTime": Int,
   "contentLanguage": String,
   "searchResults": {
     "content": String,
     "date": String,
     "authorName": String,
     "authorResponses": String,
     "messagesViews": String
   },
   "pagination": {
     "paginationNext": String,
     "paginationContent": String
   }
 }
}
```

platform: name of the platform, for example google

settings: In the next fields it will be defined the configuration for the scraper:

url: The url of the website

proxy: This field is optional and it allows the scraper to use a proxy URL with this website.

waitUntil (String): The scraper needs to wait the page is loaded before can work and extract information. We use this field to send a valid class or ID of the HTML structure to the scraper, and the scraper will wait until this HTML tag is loaded and is visible in the webpage. Usually, we want a HTML tag related with the information we want to scrap (lists, tables, etc)

searchParameters: This field is an object an have two different fields. In this fields we need to send the tags of the search form input and the search button tags. The scraper will simulate the interaction of one user and fill the input search form and click in the search button.

waitForResults (String): This field is very similar to the waitUntil field, but in this case is related to the results page when a search is done. We need a HTML tag related to the search results. Good candidates will be tags related to the number of results or pagination of the results.

resultsPerPage (Number): The quantity of search results per page

waitTime (number): The scraper has a simple delay function to add a time between pagination or page loads. With this delay added we can avoid webpage bans. The webpages can have mechanisms to detect if a user is doing a lot of requests in a short time frame and ban this user.

contentLanguage (string | international country code (ES, US, FR): We use this field to the library in charge of translating the datetime format, usually in the local language of the website, to a valid ISO format. We will use this field to send in the result messages too.

searchResults: this is an object used to send to the scraper the fields that contains the info that we want. Have different fields related to the information we want to extract:

content: the tag that contains the message text itself.

date: the tag that contains the information about the date creation of the message

authorName: the tag related to the name of the author of the message.

authorResponses: the tag related of the responses to this message.

messageViews: the tag that contains the information about the number of views of this message.

IMPORTANT NOTE: *These fields are optional. If does not have this data, the scraper will not search for this information.*

pagination: this is an object related to the pagination of the information. This object has two fields:

paginationNext: The tag of the button or link to move to the next page of the search results

paginationContent: this is the tag that contains the number of results in this search.

8.2 Web scraper create job json body example

```
{
 "keyword": String,
 "proxies": Array[String],
 "webhook": String
}
```

keyword: The keyword to search into the webpage.

proxy: Optionally, we can send a proxy for this job. If the job petition does not have a proxy, the scraper will use the proxy in the source configuration, if there is such.

Webhook: The URL that the scraper should call when a job is done

8.3 Reddit scraper create job JSON body example

```
{
 "query": Array[String],
 "requested_languages": Array[String],
 "count": Int
}
```

These are the requested parameters:

"query": An array of words to be searched in the webpage

"requested_languages": An array of the languages wanted to be searched.

"count": A limit number of messages to be returned.

8.4 Job status endpoint response

```
{
 "status": String,
 "keyword": String
}
```

status: This field have the status of the job. The possible results are:

pending: The job is waiting for a free scraper.

inprogress: The scraper is working on this job.

error: The scraper had an error doing the job. We can have an error message.

done: The scraper has already done the job.

keyword: The keyword that the scraper has in the job.

8.5 Scraper output data example

```
[
 {
   "text": String,
   "authorName": String,
   "authorResponses": Integer,
   "messageViews": Integer,
   "date": ISO8601Date
 }
]
```

text: The text of the message

date: The creation date of the message in ISO format (UTC).

authorName: The name of the author of the message

authorResponses: The number of responses to this message, if available.

messageViews: The number of views of this message, if available.

9 Appendix B

9.1 Data response example


```
[
 {
     "text": "Un \"journaliste\" ukrainien appelle à tuer tous les Russes, femmes (...) C'est étrange que, quasiment personne dans les commentaires, ne relève que Fakhrudin Sharafmal, ça sonne pas vraiment Ukrainien de souche. Quand on voit sa tronche, il ressemble plutôt à un Syrien. (...)",
     "date": "2022-03-20T13:27:45.000Z",
     "authorName": "abcd1234",
     "authorResponses": 0,
     "messageViews": 0
 },
 {
     "text": "Bunkerisation : les riches seront-ils les seuls à survivre à l'hiver (...) Apparemment il y a des serres sous terre avec tout ce qu'il faut pour survivre.. je ne me rappelle plus mais il y a un nom. Ce dont je suis sûre, c'est que ça ne sera certainement pas Paris qui (...)",
     "date": "2022-03-14T17:52:34.000Z",
     "authorName": "abcd1234",
     "authorResponses": 0,
     "messageViews": 0
 },
 {
     "text": "Arrivée massive de mercenaires en Ukraine. Les mercenaires étrangers devaient être exécutés d'après le tsar. Pologne et Roumanie risquent d'être bombardés aussi.",
     "date": "2022-03-14T12:26:58.000Z",
     "authorName": "abcd1234",
     "authorResponses": 0,
     "messageViews": 0
 }
]
```


9.2 Egaliteetconciliation webscraper config example


```
{
 "platform": "egaliteetreconciliation",
 "settings": {
   "url": "https://www.egaliteetreconciliation.fr/",
   "proxy": "",
   "headless": true,
   "waitUntil": ".barre_rech",
   "auth": false,
   "search": true,
   "searchParameters": {
     "searchField": "#recherche",
     "searchAction": ".submit"
   },
   "waitForResults": ".cher_block",
   "resultsPerPage": 10,
   "waitTime": 500,
   "contentLanguage": "fr",
   "searchResults": {
     "content": "#wrappermiddle > div:nth-child(4) > ul > li",
     "date": "#wrappermiddle > div:nth-child(4) > ul > li > i",
     "authorName": "name",
     "authorResponses": "messages",
     "messagesViews": "views"
```

```
    },
    "pagination": {
      "paginationNext": "a.lien_pagination.pagination_page_suivante",
      "paginationContent": "#wrappermiddle > div:nth-child(4) > h2 > sup"
    }
  }
}
```