## Social media content repository

To support the social media data storage in Pheme, WP6 provides the data collection framework named **Capture** developed by Atos. Capture data is stored in NoSQL databases (Apache HBase and Apache Solr), that provide necessary scalability and data distribution over multiple nodes in order to handle large volumes of data.

The syntactic repository is part of the Capture module for data collection, further explained in section 7.1. Figure 7 shows the high-level architecture of Capture.

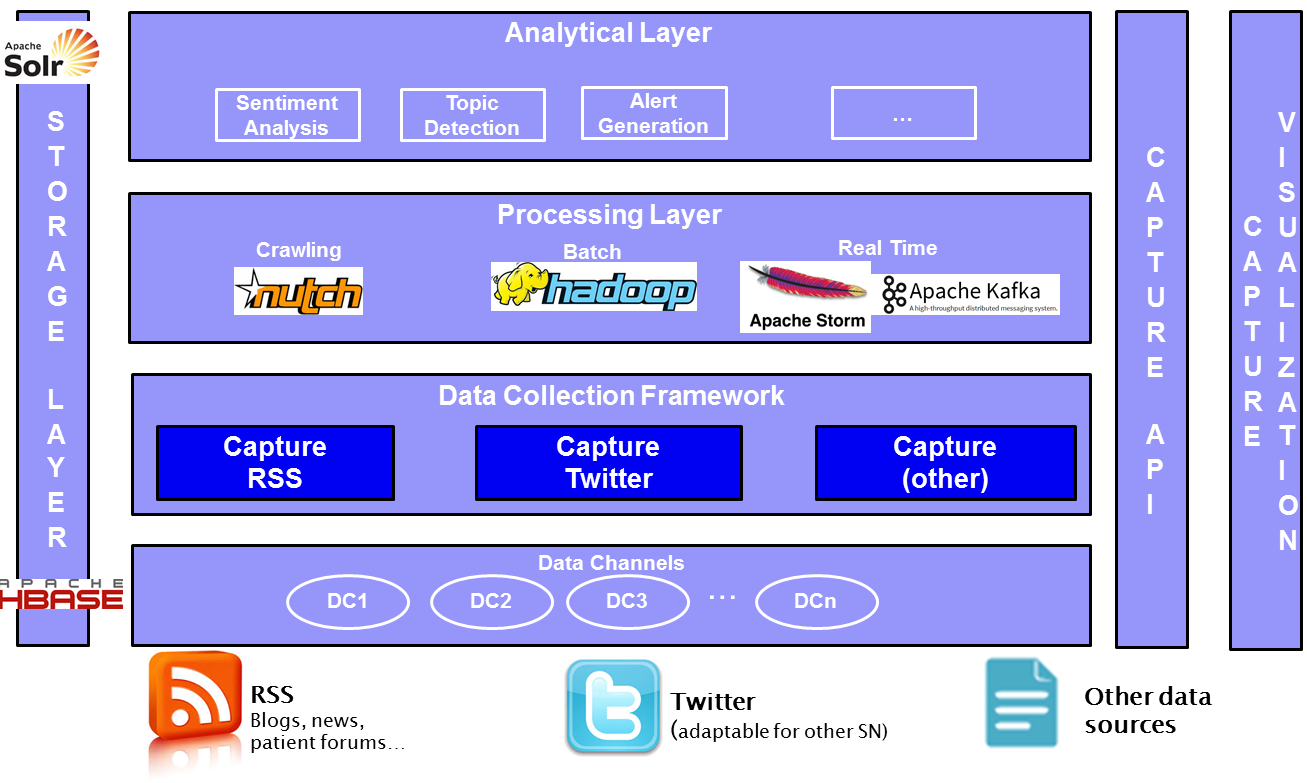


Figure 7. Data collection framework (Capture) high-level overview

From the repository perspective, the main element from the Capture architecture is the storage layer. Capture is using Apache HBase as storage repository combined with a powerful indexing mechanism based on Apache Solr. This way, fast information retrieval for the collected data (i.e. tweets) is ensured. The interaction with the repository is done using a RESTful service layer (the Capture API). This API layer provides methods for the management of the data channels and search,

It is also worth mentioning that Capture enables pipelining different analytical components both batch or real time, giving an extra integration flavour to the data collection tool. Capture provides the possibility of interacting with real time streams of tweets. To that extent, the use by other components developed within the project of the real time processing framework provided by Capture is possible without having to interact directly with the repository. More detailed explanation of the Capture API and how to use it for data integration is provided in section 7.1.

During the first year of the project, the objective was to carry out the data gathering from Twitter. To that extent, Atos provided an initial version of the Capture framework, developed in the scope of commercial proof of concepts with Atos customers. Capture had an initial data model enabling the acquisition of tweets. However, the acquisition tweets was not deemed enough for the requisites of the Pheme use cases for rumour and veracity detection. Therefore, the Capture model was extended to be able to cope with different data sources (Twitter, RSS and in the future any other social network). In particular, the Twitter model in Capture was also extended to be able to not only retrieve the pure JSON representation of a tweet, but also to store and index more raw info, such as getting user profiles, list of followers and followees, etc. The Capture model for Twitter data is now characterized by the followings attributes:

* TweetID: The representation of the unique identifier for the Tweet.
* CreatedAt: UTC time when this Tweet was created.
* FavouriteCount: Indicates approximately how many times this Tweet has been “[favorited](https://dev.twitter.com/rest/reference/post/favorites/create)” by Twitter users.
* HashTags: Tweet hash tags.
* InReplyToId: If the represented Tweet is a reply, this field will contain the integer representation of the original Tweet’s ID. Needed for the conversation chains.
* Latitude: The latitude of the Tweet’s location.
* Longitude: The longitude of the Tweet’s location.
* OriginalTweetId: Original identifier of the Tweet.
* Place: It is a specific, named location with corresponding geo coordinates.
* RetweetCount: Number of times this Tweet has been retweeted.
* Source: Utility used to post the Tweet, as an HTML-formatted string. Tweets from the Twitter website have a source value of web.
* Text: Tweet text
* UserDescription: The user-defined text describing their account.
* UserFollowers: The number of followers this account currently has. Under certain conditions of duress, this field will temporarily indicate “0.”
* UserFollowees: The number of users this account is following. Under certain conditions of duress, this field will temporarily indicate “0.
* UserID: The representation of the unique identifier for the User.
* UserName: User name.
* UserScreenName: The screen name of the user.

Besides Twitter data, Pheme use case partners have a set of requirements with which the social data repository should deal with:

* Collection of data from RSS feeds: Both case studies expressed the need of acquiring data from news and blogs for different purposes. A standard way of acquiring data is getting the content based on querying RSS feeds provided by many sites. In order to create a common acquisition framework, Capture will integrate this RSS feed content gathering using a similar approach. The approach would be to generate a new data source model of type RSS, similar to the case of Twitter mentioned above. This new type of data source will give the possibility of expressing the RSS feeds (the actual URL of the feeds) and filters (keywords or similar) to be acquired. In this way, for instance, Capture will be able to acquire news from the pages retrieved from the feeds that contains a set of keywords, not all the pages. This is still work in progress at the time of writing this deliverable.
* Specific connectors to some forums: As a requirement from the eHealth case study, some patient forums that are not open or do not provide RSS feeds would need ad-hoc scraping in order to get content. This might require also agreements with the forum’s owners and it is still unclear to what extend WP6 will provide support for this type of functionality.

Speaking in terms of enhancements or Pheme-related improvements to the Capture framework, it is planned to work in the following aspects in the coming months:

* Define a future strategy for snapshots of user profiles. Capture allows the possibility of getting and storing Twitter user profiles. However, the question of automating what profiles should be extracted is still unclear and will depend on the needs of the analysis to be done in the scope of the Pheme veracity framework. Therefore, the concrete strategy to get Twitter user profiles is not set yet. Currently, it is a manual procedure (a service getting one or several profiles in a service invocation). A tentative strategy could consist in getting the profiles on demand.
* Show the retweets history.
* Links/Sources download and indexing: Getting the links (URL mentioned in a specific tweet) could be of interest to get more knowledge related to specific rumours. However, retrieving all the links in all tweets could become storage and computationally expensive. Therefore, a strategy to manually select and get the content for some key tweets will be put into place within Capture.
* Try to maximise the amount of tweets gathered by making an intelligent use of the Twitter APIs. Capture is currently managing the limits of queries posed by the different open Twitter APIs. However, there is still room for improvement.

The section below reports the main methods of the Capture API.

## Data collection framework: Capture

## Description

User-generated content and social networks are highly dynamic, thus the PHEME data collection tools will track over time the content created by a given user, exchanges with other users, as well as changes in their profiles and social networks. For that purpose, the partner ATOS provides in the context of Pheme a data collection tool named “*Capture*”.

From the conceptual point of view, the main concept in Capture is the ***Data Channel***. A data channel is the way users can group query results below a single umbrella. Data channels allow defining several queries or *Data sources* to social networks. All results are stored and indexed associated to the channel, giving the user the possibility of knowing and querying for their results, rather than having them all in a single storage.

Data Channels have associated metadata that describes them. This metadata is defined by means of the set of attributes:

* channelID: unique identifier of a data channel
* description: data channel description
* name: data channel name or Title
* creationDate: Date/time when the data channel has been created
* updateDate: Date/time when the data channel has been updated last time
* startCaptureDate: Date/time to start the collection of data for the data channel
* endCaptureDate: Date/time to finalize the collection of data for the data channel

The second entity that enriches the definition of Data Channel is the ***Data Source***. A Data Source represents a specific web resource (i.e. Twitter, RSS, Reddit, Facebook, etc.) and the definition of queries or filters that the system will perform. Conceptually, a Data Channel could be composed of 1 to N Data Sources, giving the possibility of making several queries to the same or different resources in the same data channel (i.e two different queries to Twitter and RSS grouped in the same data channel). The Data Source is defined by the following attributes:

* type - The type of the data source, currently supported for twitter and RSS
* sourceID - The ID of the data source
* keywords - The actual query definition.

## Technical Perspective

Capture is a solution for social data collection that is based on big data technologies. It relies on the concept of gathering content using dedicated data channels. A data channel listens to data from different data sources (i.e. Twitter), meaning that data channels implement configurable user queries (based on keywords, hashtags, locations, etc.) to gather data (tweets) associated to the channel. The users are therefore able to set up several data channels for different purposes (i.e. to listen to specific events, or search for mentions of legal highs in Twitter), providing that the search limits provided by the APIs of the social networks (i.e. the limits of the public Twitter search and/or streaming APIs) are respected. Therefore, the data can be collected in very flexible ways. Capture also provides a search API to query for the data collected. It is also worth mentioning that Capture enables pipelining different analytical components both in batch or real time, giving an extra integration flavour to the data collection tool.

Figure 13 depicts the main building blocks of the Capture module along with some of the provided functionalities.

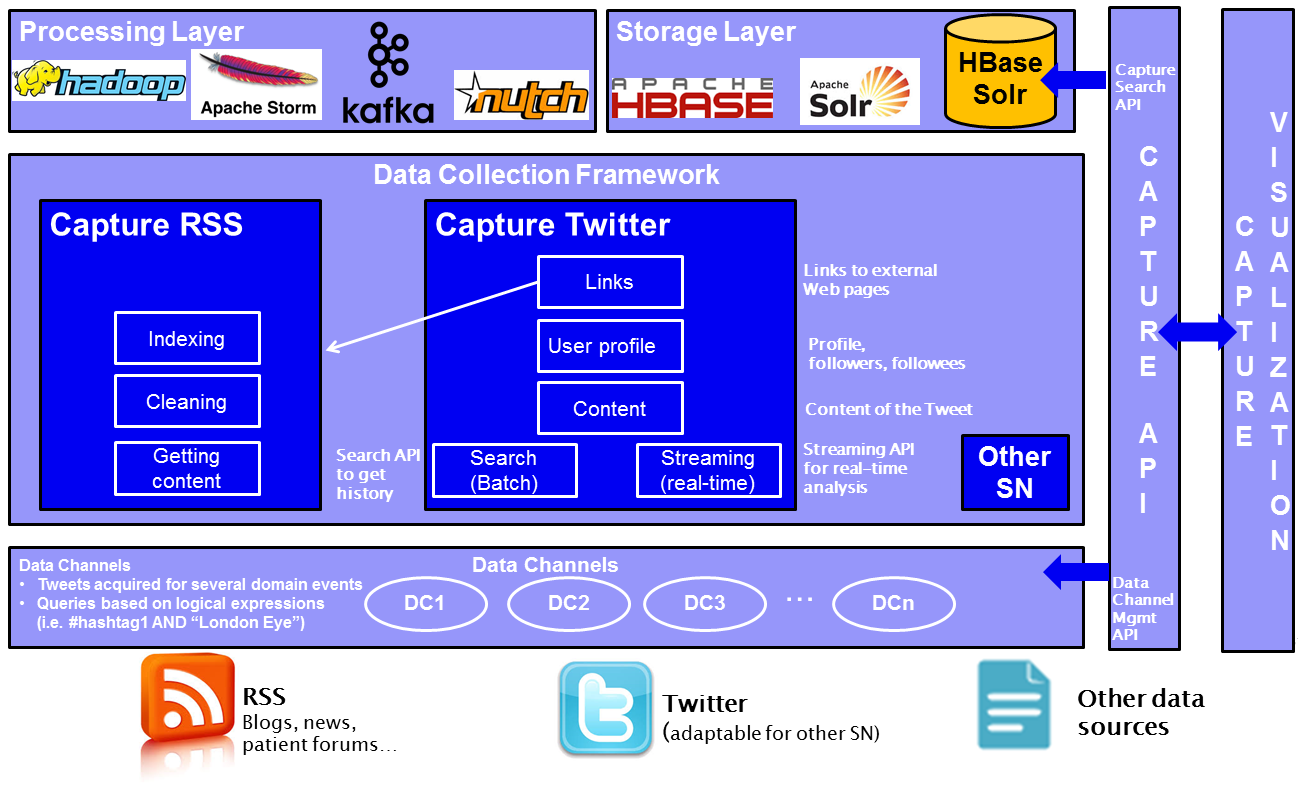


Figure 13. Detailed overview of the Data Collection framework (Capture)

## Deployment Environment

Capture is deployable in any UNIX environment with the following characteristics:

* Tomcat 6-7 or Jetty 9
* Java 7
* Apache Kafka 0.8.2.2 or later,
* Apache HBase 1.1.1 or later,
* Hadoop 2,
* Apache Solr 5 or later
* Apache Flink 0.10

The recommended minimal physical architecture is: 4 CPU cores, 8 GB RAM and 500 GB HD (extensible if the storage needs grows in time)

Capture expose two sets of user interfaces: (1) Capture REST API and (2) Capture Web GUI.

## Invocation guidelines

The Data Collection tool Capture is deployed as RESTful services. The signature of the service is provided in this section.

The examples below show how to use the Capture REST API for the following actions. Note that the URL is relative. For the Pheme deployment, the root URL is:

<http://pheme-capture.gate.ac.uk/CaptureREST>

Since the last version of the data retrieval services there have been few subtle modifications, related to the data retrieval from data channels and pagination.

1. Data channel manipulation services
   * Create a Data channel (DC) with data sources (DSs):
     + *Step 1: Create DC and DSs*
       - URL: [/rest/datachannel/](http://95.211.84.96:8080/CaptureREST/rest/datachannel/)
       - HTTP operation: POST
       - Content:

<dataChannel>

<channelID>a59fb3dc-94cd-444c-a730-10c8bb18464c</channelID>

<creationDate>2014-09-05 10:59:05.477</creationDate>

<updateDate>2014-09-05 10:59:05.477</updateDate>

<description>Refined Data channel for legal highs</description>

<startCaptureDate>2014-09-05 10:30:00.000</startCaptureDate>

<endCaptureDate>2014-09-12 15:45:00.000</endCaptureDate>

<name>Data collection for Pheme</name>

<status>active</status>

<type>search</type>

<dataSources>

<twitter>

<type>Twitter</type>

<sourceID>deeddc3e-dc26-4da0-a10e-5eecc6c09fd3</sourceID>

**<keywords>**

**lang:en geocode:51.5085300,-0.1257400,30km "WHITE MAGIC" OR "MIAOW MIAOW" OR "MEOW MEOW" OR MEPH OR "M-SMACK" OR "M-CAT" OR "MEPHEDRONE"**

**</keywords>**

</twitter>

</dataSources>

</dataChannel>

* *Step 2: Querying for the acquired data*
  + To get data using pagination and a specific page size: /rest/datachannel/{dataChannelID}/data?fromId=348173591752810&numResults=150  
    fromId field can be obtained automatically from “lastTweetId” property of the “tweet\_list” field of each non-empty result.
  + By default fromId is empty and numResults = 100
    - Get All Data Channels in current Capture instance:/rest/datachannel/
    - HTTP operation: GET
    - Get Data Channel Configuration:/rest/datachannel/{dataChannelID}
    - HTTP operation: GET
    - Update Data Channel:/rest/datachannel/{dataChannelID}
    - HTTP operation: PUT
    - Content:

<dataChannel>

<channelID>325be839-6585-4e98-bb26-5018222fe464</channelID>

<creationDate>2014-07-09 11:31:38.271</creationDate>

<dataSources>

<twitter>

<type>Twitter</type>

**<sourceID>29413386-8351-42e2-a51f-a32f78877d3c</sourceID>**

<keywords>Tour 2014</keywords>

</twitter>

</dataSources>

<description>Tour DC Updated</description>

<endCaptureDate>2014-07-09 15:26:00.000</endCaptureDate>

<name>Lab TEST</name>

<startCaptureDate>2014-06-12 17:57:51.401</startCaptureDate>

<status>active</status>

<type>search</type>

<updateDate>2014-07-09 11:31:38.271</updateDate>

</dataChannel>

* + - Delete Data Channel (logically):/rest/datachannel/{dataChannelID}
    - HTTP operation: DELETE
    - Get Twitter User Profile:/rest/datachannel/twitter/user/{userId}
    - HTTP operation: GET
    - NOTE: This method allows to retrieve the user profile (with followers and followees) using a given user handler. It stores the user if it is not in our KB

1. Faceted Query services
   * Faceted query services allow to look at the stored data through the following facets:
     + “dcID” – data channel Id a document belongs to
     + “createdAt” – creation time of a document, in ISO format,   
       e.g.: 2015-11-22T06:26:50+00:00
     + “text” – textual content of the document (allowing full text search)
     + “favouriteCount” – twitter specific field, number of times a tweet was starred as favourite
     + “tweetID” – twitter specific field, twitter assigned ID of a message
     + “hashTags” – twitter specific field, list of hashtags in the tweet text
     + “retweetCount” – twitter specific field, number of retweets of an original message
     + “originalTweetId” – twitter specific field, id of an orginal tweet in case that the message is a retweet
     + “inReplyToId” – twitter specific field, id of a parent message, in case that the tweet is a reply to other tweet
     + “sourceUrls” – twitter specific field, list of URLs contained in the tweet text
     + “userFollowers” – twitter specific field, number of followers of the author of the message
     + “userFollowes” – twitter specific field, number of the followees of the author of the message
     + “userID” – twitter specific field, numerical ID of the twitter user
     + “userScreenName” – twitter specific field, textual user name, as presented by twitter
     + “latitude” – twitter specific field, latitude part of the geolocation data (if exists)
     + “longitude” – twitter specific field, longitude part of the geolocation data (if exists)
     + “dcTweeetID” – capture specific field, a composed id (dc id + tweet id)
     + “typeDS” – capture specific field, social networks name (“twitter” for Twitter messages)
   * URL endpoint: /rest/datachannel/data
   * Parameters:
     + filterExpression – facet(s) to use as a filter. Different facets are separated by “AND” operators. e.g.: (createdAt:[2016-01-10T00:00:00.000Z TO 2016-01-13T00:00:00.000Z] *AND* sentiment\_feature:1.0)
       - “text” field:
         * text:vriendinnetjes AND/OR liefste  “vriendinnetjes” and/or “liefste” in any position in the text
         * text:”vriendinnetjes liefste”  “liefste vriendinnetjes” in the text
         * text:Vriendin\*  “Vriendin” in any position in the text as a word or as a lema
         * text:Vriendin\* AND/OR Gezellig  “Vriendin” in any position and/or the word “Gezellig” in any position.
     + sorter – field to use as a sorting order. Default sorting field: tweetID.
     + mode – sorting mode: asc or desc. Default: desc
     + id – limit results to concrete data channel
     + typeDataSource - limit results to concrete social network type. Default: Twitter.
     + numResults – number of results per page
     + page – number of the page
     + fields – coma separated list of fields to return
   * example:  
     Retrieve all documents from data channel “1f55451f”, with creation date between 2015-04-13 07:00:00 UTC and 2015-04-13 08:00:00 UTC, sort by “retweetCount” field, and include the following fields only: retweetCount, userID, text, tweetID, createdAt:  
       
     /rest/datachannel/data?filterExpression=(createdAt:[2015-04-13T07:00:00.000Z TO 2015-04-13T08:00:00.000Z])&id=1f55451f&sorter=retweetCount&mode=desc&fields=retweetCount,userID,text,tweetID,createdAt
   * example:

Retrieve all documents from data channel “1773632b”, with text containing “esplendido” and “video” words in any place, with positive sentiment, and including all fields:

rest/datachannel/data?filterExpression=((text:esplendido AND video) AND sentiment\_feature:1.0)&id=1773632b

1. Aggregated Services
   * getTagCloud.
     + Description: Get the most repeated words for a concrete(s) data pool(s) for a specific period of time and with a concrete periodicity
     + Parameters:
       - dataPoolIdL 🡪 Desired data pool Id. To get for multiples data pools include additional “dataPoolIdL” parameters.
       - date 🡪 Desired date. Following pattern: “Thu, 19 May 2015 00:00:00 GMT” codificado en la url
       - per 🡪 Periodicity (frequency measurement): {minutes1, hourly, daily}. Nowadays for search just hourly and daily. For streaming minutes1.
       - Result: It will get as much result as data pools desired. For each data pool will get the 20 most repeated words (with the number of word apparitions).
     + Example:
       - Get the most repeated words for the data pool “08593b4” for the whole day 20 July 2016.
         * …/AnalyticREST/rest/analytic/tagCloud?dataPoolIdL=08593b49&date=Wed%2C%2027%20Jul%202016%2000%3A00%3A00%20GMT&per=daily
       - Get the most repeated words for the data pool “[4ce776b8](http://154.48.153.6/CaptureREST/detailsDataPool?id=4ce776b8)” for the whole minute Sun, 10 Jul 2016 19:18:00 GMT.
         * …/AnalyticREST/rest/analytic/tagCloud?dataPoolIdL=4ce776b8&date=Sun%2C%2010%20Jul%202016%2019%3A18%3A00%20GMT%0A&per=minutes1
   * getVolume.
     + Description: Get the number of tweets for a concrete(s) data pool(s) for a specific period of time and with a concrete periodicity
     + Parameters:
       - dataPoolIdL 🡪 Desired data pool Id. To get for multiples data pools include additional “dataPoolIdL” parameters.
       - initDate 🡪 Init date from the period of time. Following pattern: “Thu, 19 May 2015 00:00:00 GMT”
       - endDate 🡪 End date from the period of time. Following pattern: “Thu, 19 May 2015 00:00:00 GMT” codificado en la url
       - per 🡪 Periodicity (frequency measurement): {seconds15, minutes1, minutes5, minutes15, minutes30, hourly, daily}. Nowadays for search just hourly and daily. For streaming minutes1.
       - Result: A time series with as much data points as slots between initDate y endDate for the desired periodicity. (Ej.: 8 days, hourly will get 192 data point). For each data point, will get as much result as data pools desired. For each data pool will get the number of tweets.
     + Example:
       - Retrieve number of tweets for following data pools: “08593b49”. 29805cac7” and “a60723a6”. Between 00:00:00 (GMT/UTC) 11th July to 23:59:59 (GMT/UTC) 18th July with a “hourly” periodicity.
         * …AnalyticREST/rest/analytic/newVolume?dataPoolIdL=08593b49&dataPoolIdL=9805cac7&dataPoolIdL=a60723a6&initDate=Sun%20Jul%2010%202016%2022%3A00%3A00%20GMT&endDate=Mon%20Jul%2018%202016%2021%3A59%3A59%20GMT&per=hourly
   * getSentiment.
     + Description: Get the number of positive and negative tweets for a concrete(s) data pool(s) for a specific period of time and with a concrete periodicity
     + Parameters:
       - dataPoolIdL 🡪 Desired data pool Id. To get for multiples data pools include additional “dataPoolIdL” parameters.
       - initDate 🡪 Init date from the period of time. Following pattern: “Thu, 19 May 2015 00:00:00 GMT” codificado en la url
       - endDate 🡪 End date from the period of time. Following pattern: “Thu, 19 May 2015 00:00:00 GMT” codificado en la url
       - per 🡪 Periodicity (frequency measurement): {seconds15, minutes1, minutes5, minutes15, minutes30, hourly, daily}. Nowadays for search just hourly and daily. For streaming minutes1.
       - Result: A time series with as much data points as slots between initDate y endDate for the desired periodicity. (Ej.: 8 days, hourly will get 192 data point). For each data point, will get as much result as data pools desired. For each data pool will get the number of positive tweets (position 0) and negative tweets (position 1).
     + Examples:
       - Retrieve positive and negative number of tweets for the data pool “08593b49 between 00:00:00 (GMT/UTC) 10th July to 23:59:59 (GMT/UTC) 10th July with a “hourly” periodicity
         * …/AnalyticREST/rest/analytic/sentiment?dataPoolIdL=08593b49&initDate=Sun%2C%2010%20Jul%202016%2000%3A00%3A00%20GMT%0A&endDate=Sun%2C%2010%20Jul%202016%2023%3A59%3A59%20GMT%0A&per=daily
   * getSentimentDegree
     + Description: Get the index sentiment (total positive - total negatives/total positives + total negatives) for a concrete(s) data pool(s) for a specific period of time and with a concrete periodicity
     + Parameters:
       - dataPoolIdL 🡪 Desired data pool Id. To get for multiples data pools include additional “dataPoolIdL” parameters.
       - initDate 🡪 Init date from the period of time. Following pattern: “Thu, 19 May 2015 00:00:00 GMT” codificado en la url
       - endDate 🡪 End date from the period of time. Following pattern: “Thu, 19 May 2015 00:00:00 GMT” codificado en la url
       - per 🡪 Periodicity (frequency measurement): {seconds15, minutes1, minutes5, minutes15, minutes30, hourly, daily}. Nowadays for search just hourly and daily. For streaming minutes1.
       - Results: A single data point (set to endDate) with as much result as data pools desired. For each data pool will get its corresponding sentiment degree.
     + Examples:
       - Retrieve the sentiment degree for following data pools: “08593b49”, “9805cac7”, “4a673994”, “cb52dc79” and “cb52dc79” between 00:00:00 (GMT/UTC) 10th July to 23:59:59 (GMT/UTC) 10th July.
         * …/AnalyticREST/rest/analytic/sentimentDegree?dataPoolIdL=08593b49&dataPoolIdL=9805cac7&dataPoolIdL=4a673994&dataPoolIdL=cb52dc79&dataPoolIdL=1ddfbd05&initDate=Sun%2C%2010%20Jul%202016%2000%3A00%3A00%20GMT%0A&endDate=Sun%2C%2010%20Jul%202016%2023%3A59%3A59%20GMT%0A&per=daily
   * getVolumePer:
     + Description: Get the popularity (based on the number of tweets and expressed in %) for a concrete data pool from a set of data pools in a specific period of time and with a concrete periodicity. Popularity shows the percentage of tweets from the candidate data pool with respect the total number of tweets.
     + Parameters:
       - dpCandidateIdL 🡪 Desired data pool Id.
       - dataPoolIdL 🡪 Set of data pools to measure with. Each data pool id will be in a “dataPoolIdL” parameter.
       - initDate 🡪 Init date from the period of time. Following pattern: “Thu, 19 May 2015 00:00:00 GMT” codificado en la url
       - endDate 🡪 End date from the period of time. Following pattern: “Thu, 19 May 2015 00:00:00 GMT” codificado en la url
       - per 🡪 Periodicity (frequency measurement): {seconds15, minutes1, minutes5, minutes15, minutes30, hourly, daily}. Nowadays for search just hourly and daily. For streaming minutes1.
       - Result: Results: A single data point (set to endDate) showing a single data pool (that indicated in dpCandidateIdL) with its popularity.
     + Example:
       - Retrieve the popularity for the data pool “08593b49” from a set of following data channels: “08593b49”, “9805cac7” and “a60723a6” between 00:00:00 (GMT/UTC) July 11th to 23:59:59 (GMT/UTC) 18th July
         * …/AnalyticREST/rest/analytic/overalPopularity?dataPoolIdL=08593b49&dataPoolIdL=9805cac7&dataPoolIdL=a60723a6&dpCandidateIdL=08593b49&initDate=Sun%20Jul%2010%202016%2022%3A00%3A00%20GMT&endDate=Mon%20Jul%2018%202016%2021%3A59%3A59%20GMT&per=daily
   * getTopVolume
     + Description: Get the N data pools with higher number of tweets from a set of data pools in a specific period of time and with a concrete periodicity.
     + Parameters:
       - dataPoolIdL 🡪 Set of data pools to include in the computation. Each data pool id will be in a “dataPoolIdL” parameter
       - initDate 🡪 Init date from the period of time. Following pattern: “Thu, 19 May 2015 00:00:00 GMT” codificado en la url
       - endDate 🡪 End date from the period of time. Following pattern: “Thu, 19 May 2015 00:00:00 GMT” codificado en la url
       - per 🡪 Periodicity (frequency measurement): {seconds15, minutes1, minutes5, minutes15, minutes30, hourly, daily}. Nowadays for search just hourly and daily. For streaming minutes1.
       - Nor 🡪 number of result (N). By default is 3.
       - Results: A single data point (set to endDate) with the top N data pools
     + Example:
       - Get the 2 data pools with higher volume from a set of following data pools: “08593b49”, “9805cac7”, “4a673994”, “cb52dc79” and “cb52dc79” between 00:00:00 (GMT/UTC) 10th July to 23:59:59 (GMT/UTC) 10th July
         * …/AnalyticREST/rest/analytic/populars?dataPoolIdL=08593b49&dataPoolIdL=9805cac7&dataPoolIdL=4a673994&dataPoolIdL=cb52dc79&dataPoolIdL=1ddfbd05&initDate=Sun%2C%2010%20Jul%202016%2000%3A00%3A00%20GMT%0A&endDate=Sun%2C%2010%20Jul%202016%2023%3A59%3A59%20GMT%0A&per=daily&nor=2