HORIZON 2020 - ICT-14-2016-1

AEGIS

Advanced Big Data Value Chains for Public Safety and Personal Security

WP3 - System Requirements, User stories, Architecture and MicroServices

D3.2 – AEGIS Components, Microservices and APIs Design v1.00

Version 1.0

Due date: 30.11.2017 **Delivery Date**: 30.11.2017

Author(s): Maurizio Megliola, Elisa Rossi, Alessandro Calcagno (GFT), Spiros Mouzakitis, Evmorfia Biliri, Christos Botsikas (NTUA), Sotiris Koussouris, Marios Phinikettos (SUITE5), Konstantinos Perakis, Dimitrios Miltiadou (UBITECH), Mahmoud Ismail, Alexandru A. Ormenisan (KTH), Yury Glikman, Fabian Kirstein (Fraunhofer)

Editor: Spiros Mouzakitis (NTUA)

Lead Beneficiary of Deliverable: NTUA

Dissemination level: Public **Nature of the Deliverable:** Report

Internal Reviewers: Fabian Kirstein (Fraunhofer), Spyridon Kousouris (SUITE5)

AEGIS KEY FACTS

Topic:	ICT-14-2016 - Big Data PPP: cross-sectorial and cross-lingual data integration and experimentation
Type of Action:	Innovation Action
Project start:	1 January 2017
Duration:	30 months from **01.01.2017** to **30.06.2019** (Article 3 GA)
Project Coordinator:	Fraunhofer
Consortium:	10 organisations from 8 EU member states

AEGIS PARTNERS

Fraunhofer	Fraunhofer-Gesellschaft zur Förderung der angewandten Forschung e.V.
GFT	GFT Italia SRL
KTH	Kungliga Tekniska högskolan
UBITECH	UBITECH Limited
VIF	Kompetenzzentrum - Das virtuelle Fahrzeug , Forschungsgesellschaft-GmbH
NTUA	National Technical University of Athens – NTUA
EPFL	École polytechnique fédérale de Lausanne
SUITE5	SUITE5 Limited
HYPERTECH	HYPERTECH (CHAIPERTEK) ANONYMOS VIOMICHANIKI EMPORIKI ETAIREIA PLIROFORIKIS KAI NEON TECHNOLOGION
HDIA	HDI Assicurazioni S.P.A

EXECUTIVE SUMMARY

The document at hand, entitled "AEGIS Components, Microservices and APIs Design v1.00", constitutes a report of the performed work and the produced results of Task 3.4 "Components' and APIs' Definition and Design" and T3.5 "Design of AEGIS MicroServices and Orchestrator". The scope of the current report can be described in the following axes:

- The high-level architecture of the AEGIS solution, reported in D3.1, is updated based on the refined user-requirements and collected feedback from the consortium discussions. It should be noted that the work performed in T3.4 and T3.5 has inherent connections to the other three tasks of WP3, as it is interdependent with the overall AEGIS platform architecture.
- The AEGIS technical architecture is designed in detail. Each of the conceptual components is mapped to specific technologies and underlying tools which will be leveraged to provide the required functionalities.
- The way components interact is described and all the APIs that will be provided in the first version of the AEGIS platform are documented.
- The way the various data related tasks are performed and the way these are seamlessly organised in data flows and workflows is reported from a technical perspective and sequence diagrams are provided.
- BPMN diagrams are provided to show how, from a user perspective, AEGIS enables the required workflows, as these have been identified from the work performed during the project so far, including the collected user stories and requirements, the MVP definition, usage scenarios and updated information from the demonstrators.

The results of the current deliverable, mainly the technical architecture of the AEGIS platform as a whole and per component and the designed interaction and workflow realisation mechanisms (including APIs), will be leveraged as input to the implementation tasks of the project. Updates to the work and results described here will be presented in D3.3 entitled "AEGIS Components, Microservices and APIs Design v2.00".

Table of Contents

LIST OF FIGURES

LIST OF TABLES

ABBREVIATIONS

API	Application programming interface
BPMN	Business Process Model and Notation
CO	Confidential, only for members of the Consortium (including the Commission Services)
D	Deliverable
DoW	Description of Work
DPF	Data Policy Framework
FLOSS	Free/Libre Open Source Software
GUI	Graphical User Interface
HTML	Hypertext Markup Language
H2020	Horizon 2020 Programme
IPR	Intellectual Property Rights
JSON	JavaScript Object Notation
JWT	JSON Web Token
MGT	Management
MS	Milestone
MVP	Minimum Viable Product
OS	Open Source
OSS	Open Source Software
O	Other
P	Prototype
PSPS	Public Safety and Personal Security
PU	Public
PM	Person Month
R	Report
REST	REpresentational State Transfer
RTD	Research and Development

SSL	Secure Sockets Layer
T	Task
TR	Technical Requirement
WP	Work Package
XML	Extensible Markup Language
XSLT	eXtensible Stylesheet Language
Y1	Year 1

1. INTRODUCTION

1.1. Objective of the deliverable

The current deliverable is related to the activities performed during the first iteration of Task 3.4 and Task 3.5 regarding the definition and design of the components that make up AEGIS as a whole and the design of the AEGIS microservices. More specifically, the objectives of the deliverable are as follows:

- Update the high-level architecture of the AEGIS solution which was reported in D3.1 based on the refined user-requirements and collected feedback from the consortium discussions
- Provide the technical architecture of AEGIS and show how each of the components in the conceptual architecture corresponds to concrete tools, implemented with the use of specific technologies.
- Provide more detailed descriptions of the functionalities offered by each of the components and the ways they interact.
- Show how, from a technical perspective, the various data related tasks are performed through the services offered by AEGIS and how these are seamlessly organised in data flows and workflows.
- Show how, from a user perspective, AEGIS enables the necessary workflows, as these were identified during the previous steps of the project, e.g. through the collected user stories and requirements, as well as in the MVP definition, usage scenarios and updated information from the demonstrators.

1.2. Insights from other tasks and deliverables

The deliverable builds directly on top of the work reported in D3.1, particularly the initial insights regarding the collected technical and functional requirements, the high-level architecture of the AEGIS framework and the actor types and interactions. Another important input source is the work performed in WP2, as reported in D2.1 and D2.2, since they both provide valuable information about the functionalities that need to be offered through certain components, clarifying how important parts of the data value chain should be supported in practice, e.g. regarding metadata handling, data policies, data harmonisation, knowledge extraction and visualisation. Finally, the usage scenarios reported in D1.2 served as a basis for outlining the user perspective when utilising AEGIS, so as to more effectively design the expected workflows.

1.3. Structure

Deliverable 3.2 is organised in four sections. Following the first introductory section, section 2 presents the user roles and high-level architecture of the AEGIS solution, updated and refined from the initial description that was presented in D3.1. Section three includes detailed information of the AEGIS components. The provided description for each component comprises an overview of its functionalities and positioning in the overall architecture, the technologies used for its implementation and the API (when available) that it exposes in order for other components to interact with. Section 4 presents the BPMN diagrams that correspond to the main tasks a user will perform through the AEGIS, as seen from the user perspective, but also outlining component interactions. Finally, in Appendix A the technical requirements of the

AEGIS solution are provided (originally presented in D3.1), together with their mappings to the components responsible for addressing them.

2. HIGH-LEVEL ARCHITECTURE

2.1. Identified actors

In section 2.2 of D3.1 the list of actors of the AEGIS platform were identified along with the interactions with the AEGIS platform as opposed to the AEGIS Value Chain. Table 1 presents the list of actors as documented in D3.1.

Actor	Description
Administrator	The Administrator is both the Tools Administrator and the System Administrator. He is responsible for: 1) setting up the proper configuration of the AEGIS tools to be easily adopted by the end users; 2) setting up the AEGIS solution and the monitoring of the proper functionality of the overall execution environment. He is the supervisor of the AEGIS Core.
Service provider	The service provider is the mid-actor between the stakeholders and the AEGIS tools. He promotes the long-term interests of end users or services supplied, the efficiency and competitiveness.
Data provider	The data provider is the provider of datasets that may be available on a specific format or even raw data. He is responsible for: 1) data source identification; 2) type of data provided; 3) objective of the data; and 4) data format.
Data curator	The data curator is responsible of the processes and activities related to the organisation and integration of data collected, annotation, publication, presentation and maintenance of the data. The data curator is responsible for: status / maintenance; 2) improvement of data accessibility and quality; and 3) data reliability, reusability.
Data analyst	The data analyst is the person that takes advantage of the available data for the realisation of an analysis based on his business needs.
Developer	The developer is: 1) the developer of components of the AEGIS tools, responsible for the development of the desired functionalities as well as the proper interfaces; 2) the Demonstrator developer of the API/tools that interact with the AEGIS Core Services
Non-expert user	The non-expert user is the end-user that visualises the results of an analysis or browse datasets.

Table 1: AEGIS actors

The list of actors remains unaffected since there were no additional requirements identified during the design of the components of the AEGIS platforms that indicate the need for modification of the list. In addition to Table 1, Figure 2-1 is illustrating their interactions with AEGIS platform and it also provided for reference. It should be noted that the user can undertake multiple roles (where roles of the actors are described in Table 1), for example a user which is the data owner can be both data provider and a service provider and a data analyst.

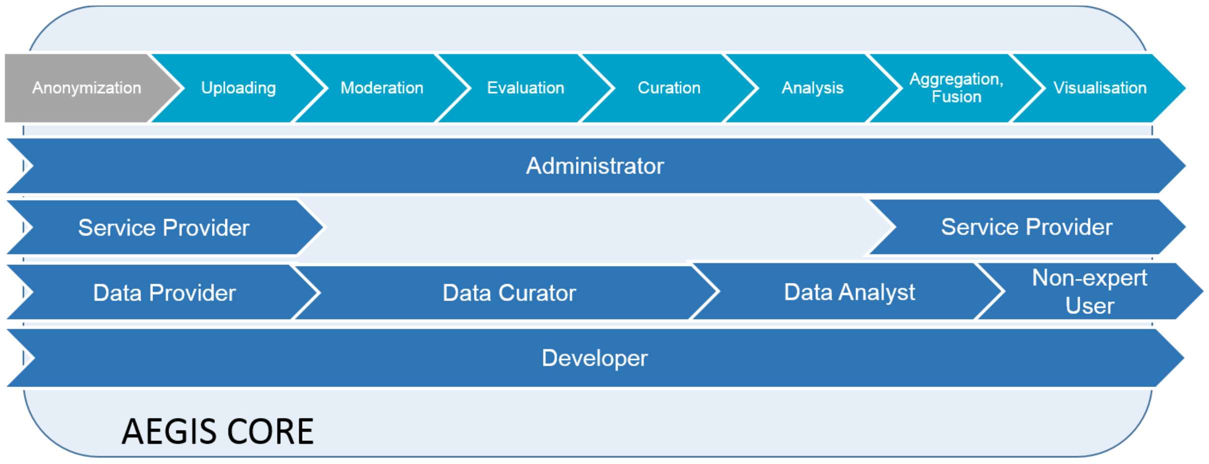


Figure 2-1: AEGIS actors' interaction

2.2. Conceptual Architecture

In deliverable D3.1 the conceptual architecture of the first version of the integrated AEGIS platform was presented. This conceptual architecture was designed in a modular way assuring that all the technical requirements, also documented in D3.1, were addressed, and comprises a list of key components. Within this context, the key components were described along with their functionalities and interactions.

As the project evolved from M6 where the first version of the integrated AEGIS platform was presented, AEGIS consortium further analysed this conceptual architecture along with the components design and made the necessary updates on the conceptual architecture towards the aim of providing a more solid and concrete architecture that will drive the implementation of the integrated AEGIS platform. The updated conceptual architecture is illustrated in Figure 2-2.

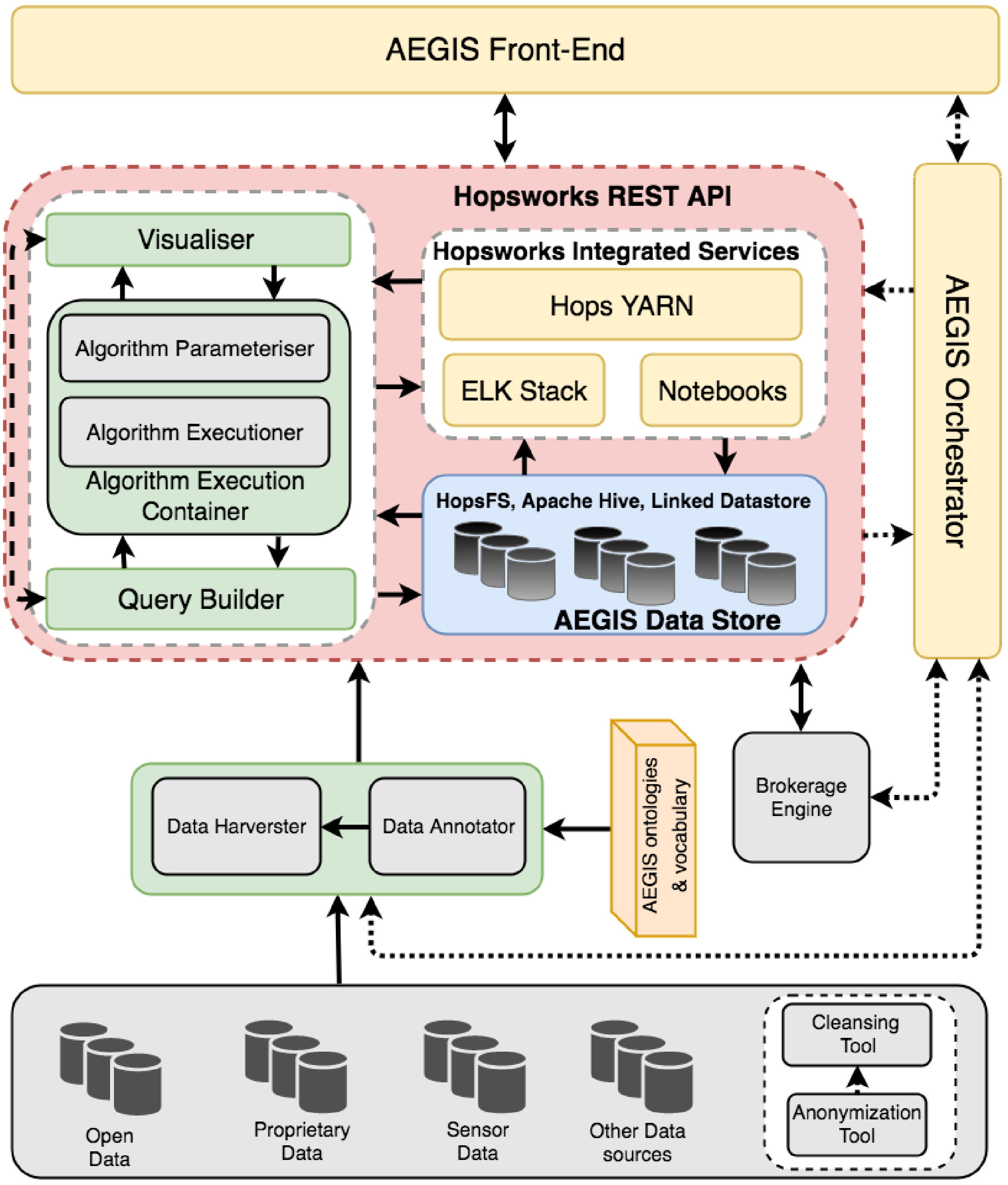


Figure 2-2: AEGIS conceptual architecture (updated)

In the updated conceptual architecture, several decisions were made with the main decision being the adoption of Hops platform[1] as the Big Data Processing Cluster of AEGIS platform. Hops is a next-generation distribution of Apache Hadoop[2] supporting Hadoop as a Service,

[1] http://www.hops.io/

[2] http://hadoop.apache.org/

project-Based Multi-Tenancy, secure sharing of datasets across projects, extensible metadata that supports free-text search using Elasticsearch[3] and YARN[4] quotas for projects among other features and services. Hopsworks is the User Interface built around Hops providing graphical access to the integrated services such as Spark, YARN, Elasticsearch, Kafka and Apache Zeppelin. HopsYARN is undertaking the responsibility of the resource management of the cluster. HopsYARN is a distributed stateless Resource Manager that enables Hops to have no down-time, providing also efficient resource management with consistent operations, security and data governance tools.

The storage capabilities of the platform will be provided by the AEGIS Data Store. The AEGIS Data Store will be based on HopsFS, a new implementation of the Hadoop Filesystems (HDFS) provided by Hops. HopsFS enables more scalable clusters as it supports multiple stateless NameNodes where the metadata are stored in an in-memory distributed database increasing performance dramatically. It should be noted that the AEGIS Data Store will be also extended with additional storage solutions, such as solutions for storing linked metadata. The evaluation will be performed during the implementation phase of the AEGIS Data Store.

The Brokerage Engine instantiates the process responsible for applying the policies concerning read and execution permissions as defined in AEGIS Data Policy Framework regarding artefacts such as datasets, services and algorithms. The Brokerage Engine is also responsible for maintaining the records of any relevant action performed over these artefacts.

The AEGIS orchestrator is the process responsible for the interconnection of various services of the AEGIS platform, facilitating the flow of information between the services and the execution of the workflows involving several distinct services of the platform, especially the ones involving any integrated service of Hopsworks with the rest of the services of the platform. Thus, the AEGIS orchestrator will act as a mediator between services upon needs, utilising the exposed interfaces of the services.

The AEGIS Front-End is the upper layer of the AEGIS platform providing an innovative User Interface for the AEGIS stakeholders. AEGIS Front-End will provide a user-friendly interface, facilitating the navigation between the AEGIS platform functionalities in a flexible, easy-to-use and secure way.

The rest of the components were introduced in D3.1 and their names and main functionalities remain the same, hence are not discussed here. More in-depth descriptions and technical details for each of them are provided in Section 3.

Table 4 in Appendix A documents the updated mapping of the technical requirements, as identified in D3.1, with the AEGIS components. A description for each of the AEGIS components illustrated on Figure 2-2 is elaborated in the forthcoming sections.

[3] https://www.elastic.co/products/elasticsearch

[4] https://hadoop.apache.org/docs/current/hadoop-yarn/hadoop-yarn-site/YARN.html

3. AEGIS COMPONENTS AND APIS SPECIFICATIONS

3.1. Data Harvester and Annotator

The following sections describe the high-level implementation details, interfaces and used technologies of the Data Harvester and Data Annotator components, which were specified in deliverable 3.1. The detailed design process of the implementation and the workflow has yielded to the conclusion that the two components are tightly coupled. In connection they represent the process of harvesting, transforming, harmonising, annotating and providing the required data and metadata for the AEGIS platform. Therefore, they will be described as one component and from here on simply denominated as *Harvester*.

3.1.1. Overview

The Harvester will be orchestrated out of several logical subcomponents, where each depicts one distinct task and domain within the process of the data acquisition and can be seen as a self-contained resources. In the following the underlying concepts of these main resources are described:

Repository
A repository represents a specific data source and handles the respective connection to it. Each repository holds descriptive and required data about the data source, where the address (in most cases a URL) is the most significant one.

Annotation
An annotation is a prototype or class of the subsequent metadata of a dataset within the AEGIS platform. Hence it uses the same vocabularies and ontologies (see D2.1 chap.3) defined for describing the datasets.

Transformation
A transformation describes all processing rules for converting the source data to the suitable target format. This may include mapping of fields, harmonisation and any converting.

Harvester
A harvester describes a concrete instance of retrieving data from a data source. It holds metadata about the harvesting process itself, like the execution schedule. One harvester is linked to a repository, the corresponding annotation and responsible transformation.

Run
A run depicts the single execution of a harvester, where the data and the metadata are generated and harvested respectively. It stores metadata about the performance and success of a harvesting process, which included detailed logging information.

These resources will be represented as appropriate API endpoints, where the combination of all endpoints constitutes the Harvester component in its entirety. Other minor resources may be added in the course of the development. Figure 3-1 illustrates how the different resources will interact with each other during a harvesting process.

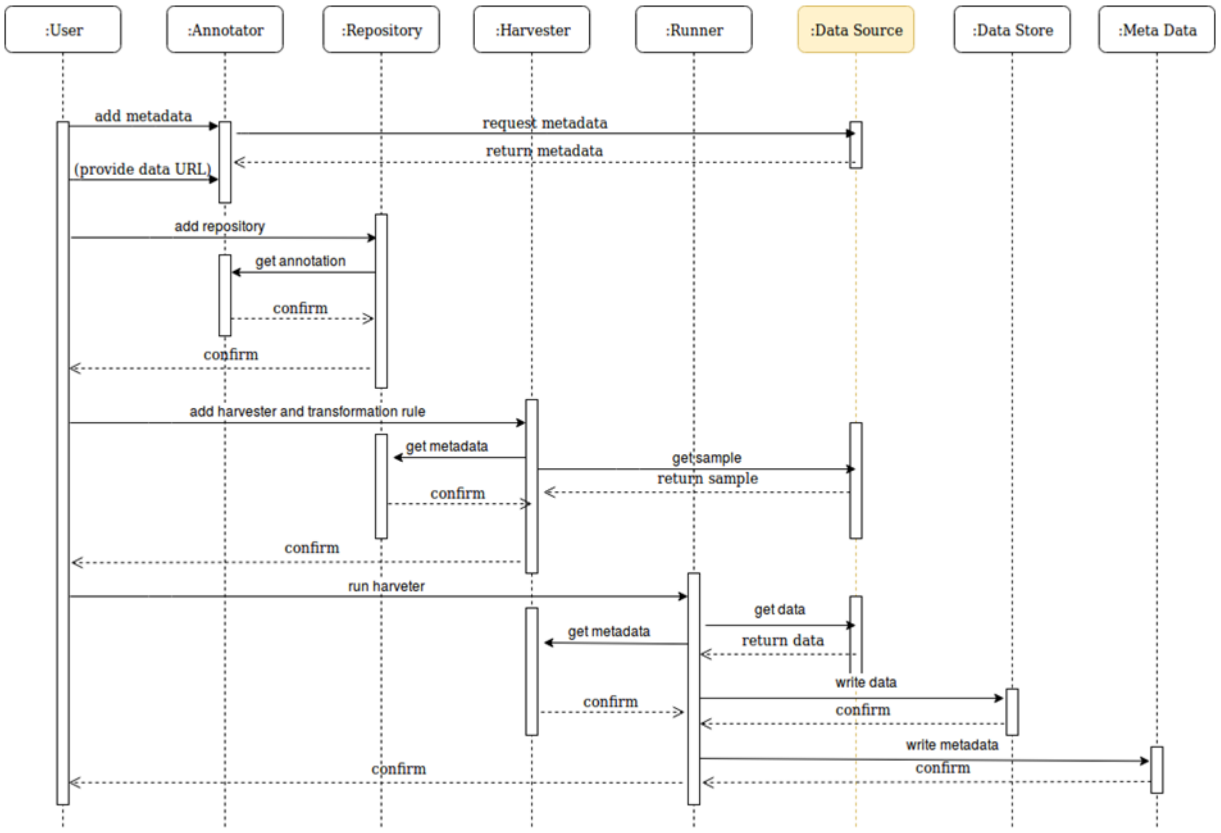


Figure 3-1 - Sequence diagram of the harvester component

3.1.2. Technologies to be used

The foundation of the AEGIS harvester will be the EDP Metadata Transformer Service (EMTS)[5], an open source solution for harvesting metadata from diverse Open Data sources. The web application allows the scheduled fetching of metadata, their rule-based transformation and the export to a target platform. It was specifically designed for harvesting metadata based on a metadata catalogue methodology, where each catalogue offers a list of datasets. Although this methodology cannot be directly applied to the requirements of the AEGIS platform, the fundamental concepts and technologies can be transferred to build the AEGIS harvester component. Currently the EMTS uses the following main technology stack:

- JavaEE[6] as the applications main framework
- PicketLink[7] for security and identity management

[5] The original source code can be found here: (https://gitlab.com/european-data-portal/MetadataTransformerService)

[6] http://www.oracle.com/technetwork/java/javaee/overview/index.html

[7] http://picketlink.org/

- Quartz Scheduler[8] as job scheduling library
- Apache Jena[9] for dealing with Linked Data and RDF
- JavaServer Faces[10] for building the user interface
- Bootstrap[11] 3 for styling the user interface
- WildFly[12] as an application server for deployment

The core architecture of the EMTS is divided into three general sections:

Importer
An importer acts as an adapter for a specific data source. The EMTS has already built-in support for various protocols and data formats, like JSON, XML, RDF etc.

Transformer
A transformer defines a specific transformation and harmonisation process from one data format into another. The EMTS supports several transformation mechanisms, mostly based on simple transformation script written in XLST or JavaScript.

Export
An exporter acts as an adapter for a specific data target. The EMTS supports among others the export to SPARQL endpoints or the CKAN API.

The consolidation of these three sections is managed and scheduled by the EMTS.

In addition to the EMTS, the open source Java-based solution StreamSets Data Collector[13] (SDC) will be employed. This application is designed to assist in building data flows from arbitrary data sources to arbitrary data destinations. It emphasises on a rich graphical user interface for assembling custom data pipelines. Furthermore, it comes with a variety of connectors and processors for well-established interface protocols and data formats and offers a plug-in system for customisation. It lacks a scheduling functionality, but since it provides a REST-API it can be easily integrated within the AEGIS platform. Conceptually SDC will be mainly responsible for the harvesting and transformation processes and the EMTS will be orchestrating, scheduling and managing these transformation processes.

In order to support the requirements of AEGIS the EMTS needs to be modified accordingly. Momentarily each concrete instance of an importer, transformer or exporter are Java classes, tightly integrated into the application. Therefore, it requires direct modifications and recompiling, when adding support for new protocols and data formats. In compliance with

[8] http://www.quartz-scheduler.org/

[9] https://jena.apache.org/

[10] http://www.oracle.com/technetwork/java/javaee/javaserverfaces-139869.html

[11] https://getbootstrap.com/

[12] http://wildfly.org/

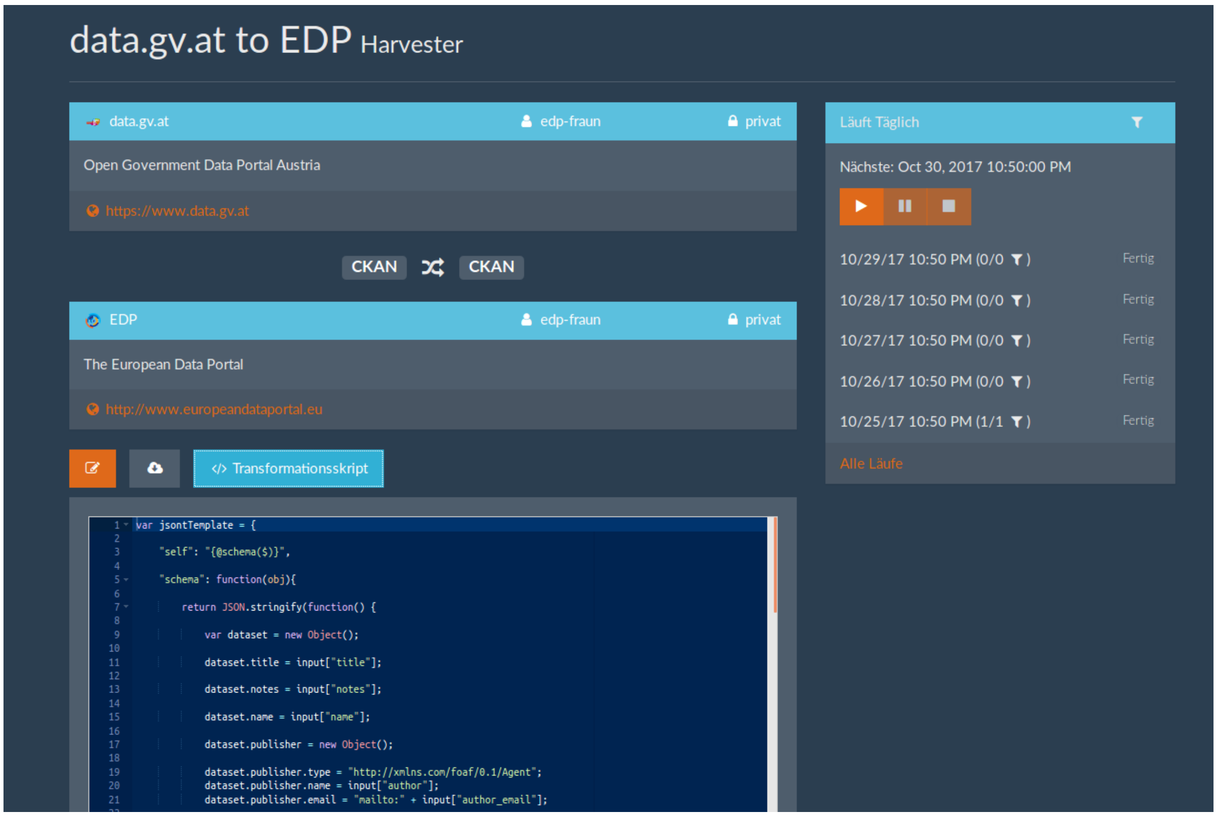
[13] https://streamsets.com/products/sdc/

overall AEGIS architecture this process will be updated to a microservice methodology. Hence a new importer, transformer or exporter will be developed as a distinct and independent service, communicating via well-defined RESTful API. Accordingly, the extension of the EMTS with a new set of importers, transformers and exporters is just the registration of new services. For example, one might a develop one service for getting a Excel sheet from a specific resource which requires specific authentication, another service for transforming the data of the sheet into the AEGIS standard data format and a third one for exporting it to the AEGIS data store. The EMTS will then orchestrate and execute these services as a specific pipeline.

Such services can be either developed from scratch in any desired programming language or technology stack. It just has to follow the guidelines for the RESTful-API, which will be defined during the course of work on this component. Alternatively, SDC can be used to graphically define a specific data pipeline, either in its entirety, including import, transformation and export or just a specific section. Based on the RESTful-API of SDC these pipeline can registered in the EMTS and executed.

When combined, the EMTS and the SDC will allow the development of very customised and individual data harvesting pipelines and an easy graphical way to compose such pipelines. As basis some basic importers, transformers and exporters for the pilot scenarios and the Hops platform will be available.

Figure 3-2 and Figure 3-3 show examples of the current user interfaces of the EMTS and SDC. Currently these are stand-alone interfaces, which will be tightly integrated into the AEGIS platform during the development. This will be mostly done by exposing all functionalities to a RESTful-API, which can then be utilised with the AngularJS frontend of AEGIS. However, the principle structure and layout will remain.



data.gv.at to EDP Harvester

data.gv.at | edp-fraun | privat

Open Government Data Portal Austria

https://www.data.gv.at

CKAN ⇄ CKAN

EDP | edp-fraun | privat

The European Data Portal

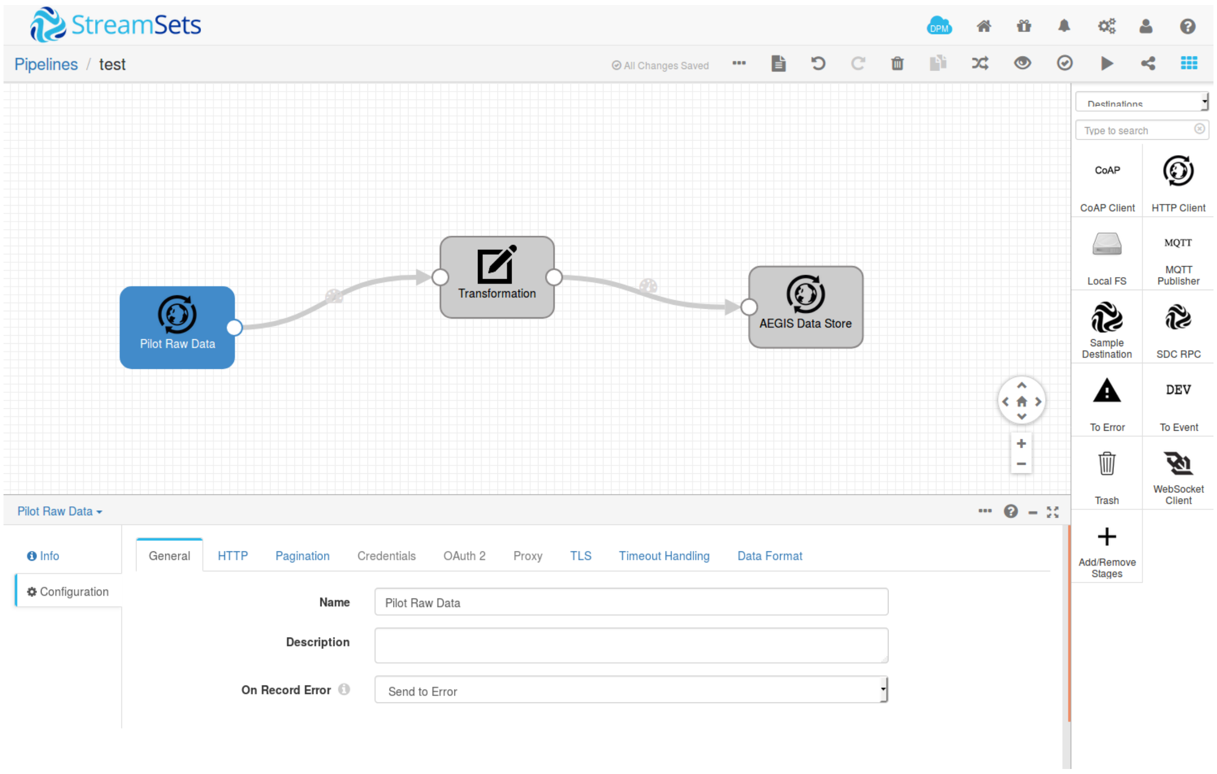
http://www.europeandataportal.eu

Transformationsskript

Läuft Täglich	
Nächste: Oct 30, 2017 10:50:00 PM	
10/29/17 10:50 PM (0/0)	Fertig
10/28/17 10:50 PM (0/0)	Fertig
10/27/17 10:50 PM (0/0)	Fertig
10/26/17 10:50 PM (0/0)	Fertig
10/25/17 10:50 PM (1/1)	Fertig
Alle Läufe	

```
var jsontTemplate = {

    "self": "{@schema($)}",

    "schema": function(obj){

        return JSON.stringify(function() {

            var dataset = new Object();

            dataset.title = input["title"];

            dataset.notes = input["notes"];

            dataset.name = input["name"];

            dataset.publisher = new Object();

            dataset.publisher.type = "http://xmlns.com/foaf/0.1/Agent";
            dataset.publisher.name = input["author"];
            dataset.publisher.email = "mailto:" + input["author_email"];
```

Figure 3-2: Exemplary view of the current harvester interface



Name	Pilot Raw Data
Description	
On Record Error	Send to Error

Figure 3-3: Building a data pipeline in StreamSets Data Collector

3.1.3. API

The API

Technical Interface	
Reference Code	XX#01
Function	Harvest, transform and harmonise data
Subsystems	Annotator, Repository Manager, Harvester, Runner
Type, State	
REST-full API, User Interface	
Endpoint URI	
https://app.swaggerhub.com/apis/aegis/Harvester/0.1 (Draft)	

3.2. Cleansing Tool

3.2.1. Overview

The cleansing tool was presented in the initial high-level AEGIS architecture described in D3.1 as an optional component, residing where the data are located before they become available in the AEGIS platform. Data cleansing is an umbrella term for tasks that span from simple data preprocessing, like restructuring, predefined value substitutions and reformatting of fields (e.g. dates) to more advanced processes, such as outliers' detection and elimination from a dataset. Particularly in the AEGIS context of big data processing and analysis, cleansing may, by itself, be a process requiring big data technologies to be applied.

As such, AEGIS will not follow a uniform approach in all data flows that involve the application of cleansing techniques. More specifically, a two-fold approach will be followed:

1. For the tasks that can be performed by applying simple predefined data transformation rules, tools will be proposed to the users to install and use offline in order to cleanse their data prior to importing them in the AEGIS platform through the harvester.
2. For the tasks that require heavier processing and analysis, data cleansing will be performed through dedicated cleansing processes that will be made available through the Algorithm Execution Container, described in section 3.9 of the current deliverable. An example of such cleansing methods could be the outlier detection and removal mentioned before, or data imputation (which could be also performed using the offline tool), in case the user prefers to benefit from the advanced capabilities of the core platform, i.e. for more efficient - time-wise - cleansing.

It should be noted that none of the approaches corresponds to new cleansing tools developed by the AEGIS consortium. In the first case, a mature and tested tool will be utilised, whereas the second approach will be offered through another component that enables the user to leverage the AEGIS processing and analytical capabilities towards coding an advanced and fully customised data cleansing process. Hence, for the second approach, the corresponding sequence diagram is the same as the one of the Algorithm Execution Container, presented in subsequent section. The sequence diagram for the first approach is presented below.

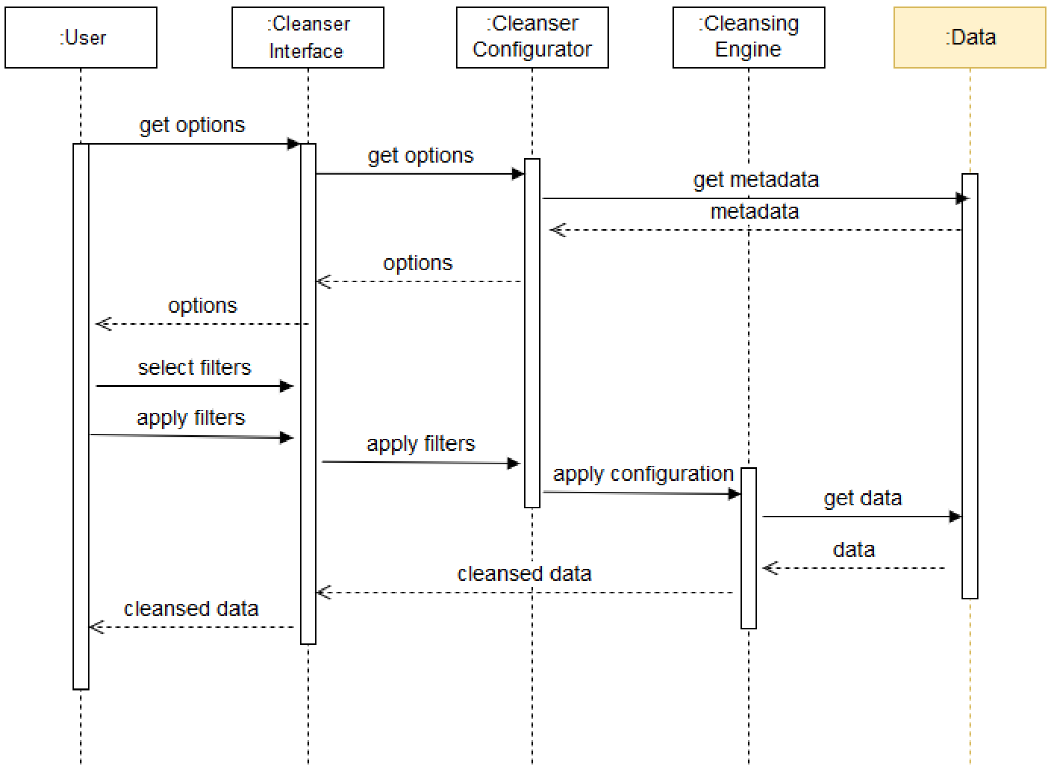


Figure 3-4: Offline data cleansing sequence diagram

3.2.2. Technologies to be used

As explained above, it is not currently in the scope of AEGIS to develop and deliver a dedicated cleansing tool, particularly since in most big data applications, cleansing is achieved inside the core platforms through the provided Jupyter and Apache Zeppelin notebooks (web applications in the form of simple online text editors, that allows to create and share documents that contain live code, equations, visualisations and narrative text), which are described in section 3.6. This is particularly relevant when cleansing emerges as an important step of the analysis to be performed, i.e. it may be also heavily dependent on the specific application and not on intrinsic characteristics of the original dataset. Modern approaches in the domain[14], that share the same technology stack with AEGIS in terms of data processing with Spark, also point to the adoption of a similar cleansing strategy.

[14] https://hioptimus.com/

For the offline cleansing processes which will be applied before importing data in AEGIS, primarily aiming to make the data more easily processable by subsequent components in the data flow, AEGIS will utilise existing mature cleansing solutions, including TRIFACTA Wrangler[15], sampleclean[16] and, for small scale transformations, OpenRefine[17]. Finally, the usage of StreamSets Data Collector, presented in the AEGIS Harvester section, will be explored.

3.2.3. API

Not applicable in the first iteration.

3.3. Anonymisation Tool

3.3.1. Overview

The inherent sensitivity of data in the PSPS domain renders the provision of data anonymisation functionalities imperative in certain cases. Indeed, the provision of an anonymisation tool was also envisioned in the initial conceptual architecture reported in D3.1, however no specific solution could be drafted at that stage. Hence, although the need for an anonymisation tool appeared in the requirements tables provided in D3.1, it will be described here in detail for the first time.

The anonymisation process is optional to the AEGIS data flows and, as depicted inFigure 2-2, the tool is external to the core AEGIS platform, residing where the data to be anonymised are located. This decision ensures that no potentially sensitive data leave company premises, i.e. by-design eliminates any vulnerability risks entailed in uploading the initial eponymised, thus sensitive, data to the platform. Therefore, the AEGIS anonymisation solution will be used offline, offering connection mechanisms with the core AEGIS platform only to facilitate the import of the produced anonymised data in the AEGIS data store. It should be noted that the offline usage of the anonymisation tool is depicted in the identified technical and functional requirements presented in D3.1.

Table 2: Requirements related to the anonymisation needs, as reported in D3.1

ID	Need	Description	Component
CRF48		AEGIS should be able to anonymise personal data	
TR29	Anonymise datasets	AEGIS should be able to offer (offline) tool for (sensitive / personal) data anonymisation.	Anonymisation Tool

15 https://www.trifacta.com/products/wrangler/

16 http://sampleclean.org/index.html#About

17 http://openrefine.org/

TR59	Support local deployment of the anonymisation tool	AEGIS should be able to support and provide deployment options for the anonymisation tool on a local platform, and provide the interfaces for the management of the tools and its execution options	Anonymisation Tool

The main functionalities of the anonymisation tool are as follows:

- Connection to various data sources, including PostgreSQL, MySQL and csv files.
- Provision of anonymisation alternatives (generalisation, k-anonymity, pseudonimity), depending on the data schemas, the data values and the user's intended usage of the anonymised dataset
- Export of anonymised data in files and as RESTful services, if desired.

Overall, the tool will help the user generate an anonymised dataset as an output, making sure that that the individual sensitive records or subjects of the data cannot be re-identified.

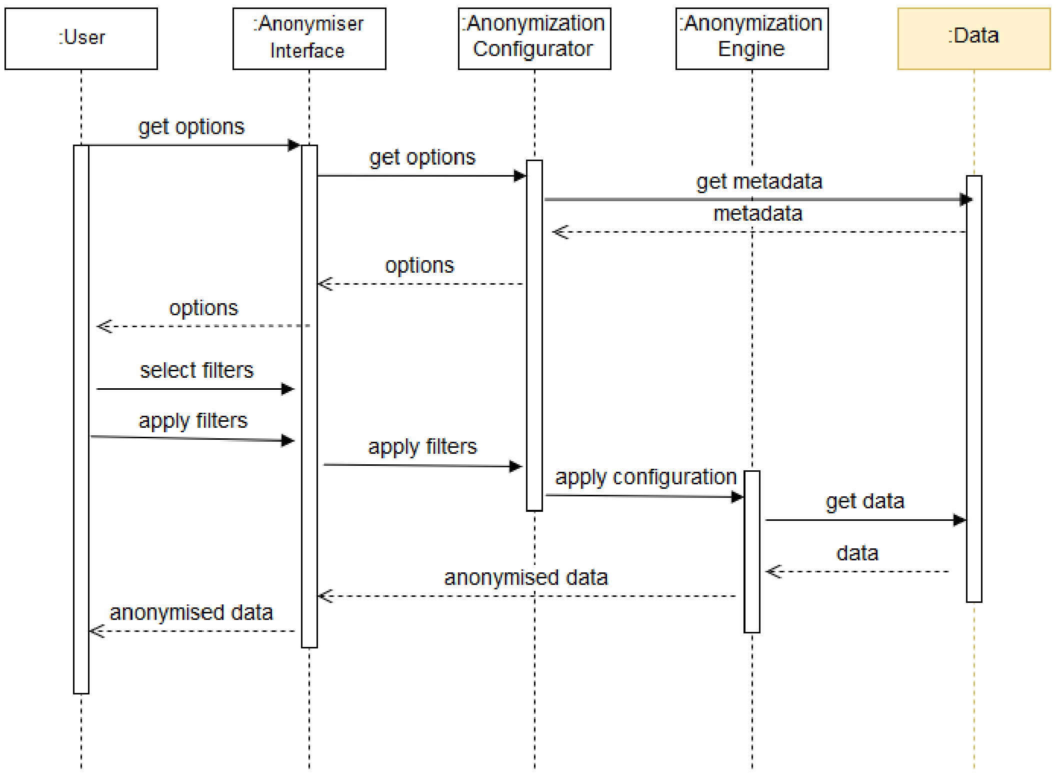


Figure 3-5: Data anonymisation sequence diagram

3.3.2. Technologies to be used

The anonymisation tool will be based on Anonymiser, an anonymisation and persona-building tool[18], developed in the context of the European project CloudTeams[19].

The tool currently performs a type of generalisation, which can be used to achieve k-anonymity. More specifically, it allows users to customise the level of anonymisation per data field, i.e. sensitive data fields can be completely stripped out or suppressed from the output with asterisks or can be generalised. With the generalisation mapping, individual values of input data fields are replaced by a broader category. For example, the value '15' of the attribute 'Age' may be replaced by ' $\leq$ 18', the value '23' by '20 < Age $\leq$ 30'. The user may then apply a threshold (k) on the minimum number of entries with the same value, thus ensuring k-anonymity. A pseudonimity functionality is also available to hide personal information and all data fields can be masked with ranged data.

The original tool is written in Python and more specifically using the Django web framework[20]. These technologies will be also used to deliver the necessary updates and extensions in order to support the AEGIS anonymisation requirements. More specifically, the tool will be extended (a)to support csv files as data sources, (b) to provide recommendations on the configuration of the anonymisation technique to be applied (e.g. recommended value ranges for generalisation, as in the current workflow it is expected from the user to know beforehand and manually insert the appropriate ranges). Furthermore, the provision of more anonymisation alternatives will be explored.

It should be noted that, as the anonymisation tool is not integrated with the other AEGIS components but only offers limited interaction points, there is a flexibility in diversifying the provided anonymisation solution. Hence, in the course of the project, the tool will be extended and adapted to the project’s requirements, but other open-source solutions may be also considered and evaluated, e.g. ARX[21], as complementary/supplementary tools.

3.3.3. API

Not applicable in the first iteration.

[18] https://github.com/epu-ntua/Anonymizer

[19] https://cloudteams.eu/projects/

[20] https://www.djangoproject.com/

[21] http://arx.deidentifier.org/

3.4. Brokerage Engine

3.4.1. Overview

The Brokerage Engine is the component that checks execution/read permissions and afterwards records actions that are performed in relation with the various artefacts that are to be found over the AEGIS platform (being Datasets, Services or Algorithms). This component is therefore responsible for applying the policies described in the Data Policy Framework (DPF) of AEGIS, and for that recording transactions over the different data artefacts, in a distributed ledger, which is supported by a blockchain implementation. As such, this component safeguards specific privacy, IPR and security related aspects that are relevant to the data and the other outputs shared over the AEGIS platform, guarantees that transactions are properly recorded and cannot be tapped, while it provides a future extension towards interconnecting various installations/distributions of the final platform under the umbrella of a common environment that supports data and sharing.

Under transactions, the following activities are foreseen:

- Uploading (a dataset/algorithm)
- Executing (an analysis/visualisation)
- Downloading (a dataset/analysis result/algorithm/visualisation result)
- Copying/Replicating (a dataset/analysis configuration/analysis result/visualisation configuration/visualisation result)
- Using (a dataset/algorithm) for conducting an analysis

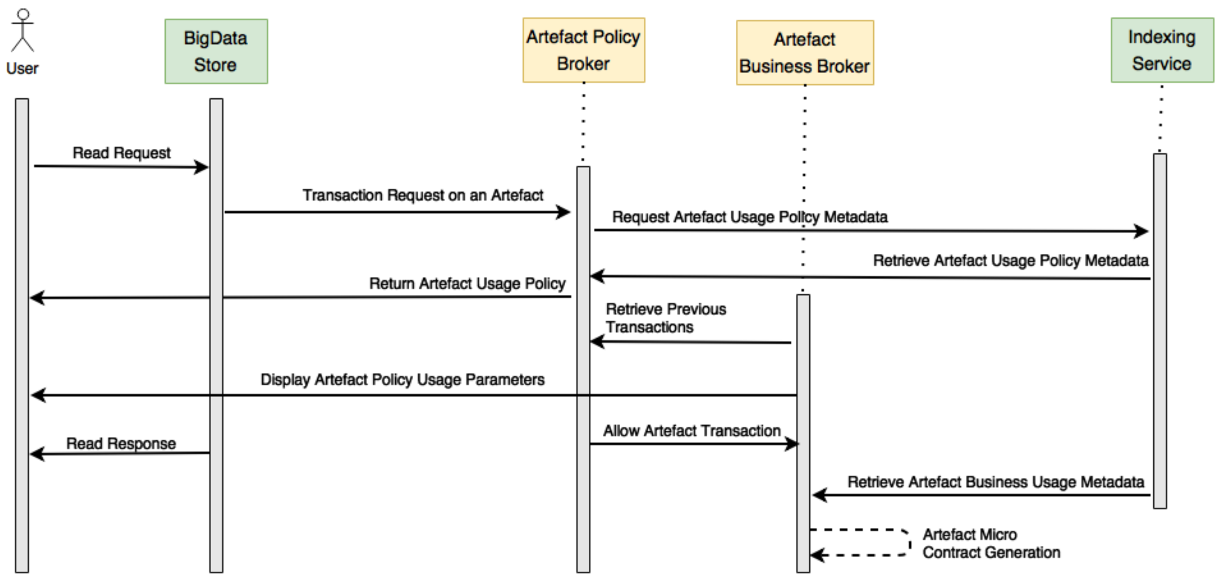

User
BigData Store
Artefact Policy Broker
Artefact Business Broker
Indexing Service
Read Request
Transaction Request on an Artefact
Request Artefact Usage Policy Metadata
Retrieve Artefact Usage Policy Metadata
Return Artefact Usage Policy
Retrieve Previous Transactions
Display Artefact Policy Usage Parameters
Allow Artefact Transaction
Read Response
Retrieve Artefact Business Usage Metadata
Artefact Micro Contract Generation

Figure 3-6: Brokerage Engine sequence diagram

As the sequence diagram above reveals, the brokerage engine is comprised out of two main brokers, the Artefact Policy Broker and the Artefact Business Broker. The Engine listens to activities that are to be performed to the AEGIS Data Store, and the Artefact Policy Broker is taking up control to identify the data policies that are relevant to an artefact. As such, checks are performed on the data policy metadata of each artefact, and conclusions are drawn related to how this artefact (if allows) can be used by a user. These are also checked against previous transactions on the ledger, and are returned to the User. As such, in case a transaction is allowed, then the business broker is engaged, and the transaction details are stored to the ledger. In the opposite case, the user is presented with a justification on why an artefact cannot be used.

The initially characterisation of an artefact with specific data policy metadata is not a responsibility of the Brokerage Engine, but of the Annotator (see above). However, the Brokerage Engine holds the responsibility of updating policies based on the transactions recorded in the ledger (e.g. if an artefact is allowed to be used in an exclusive manner by a user for a specific timeframe, then the Brokerage Engine has to if this artefact has been already registered to another user, and if the timeframe for its usage has expired or not).

It needs to be noted, that no "monetary" or "marketplace" features are foreseen at this stage, which would require the presence of a dedicated component that should interact with the business broker in order to facilitate the transfer of "monetary" value between a user and the owner of an artefact. However, future implementations of the platform, depending also on the exploitation strategy of the project, may result in the introduction of such a component.

1.1.1. Technologies to be used

The brokerage engine will be based on a hybrid approach which utilises Python to implement part of the DPF, as well as an implementation of Hyperledger Iroha[22] for the blockchain part of the engine, and especially the Iroha-Python distribution[23] (a python library for Iroha). The overall approach will infiltrate to the different calls and methods provided by HOPSworks in order to capture the related events and stroe them in blockchain, with accompanying data that comes from the metadata describing the data policies of each asset. As such, the brokerage engine will consist of:

- A listener, that will identify activities of interest on the platform and register them to the blockchain using the Hyperledger Iroha ledger
- A policy validation engine, that will:
    - precede the blockchain entry to identify whether a transaction is valid and can be performed
    - run on the background to crosscheck entries in the block chain and enforce policies (typically time-related policies (e.g. subscription expirations of licenses, etc.))

3.4.2. API

No API will be provided in the first version.

[22] http://iroha.tech

[23] https://github.com/hyperledger/iroha-python

3.5. AEGIS Data Store

3.5.1. Overview

AEGIS Data Store component is responsible for storing the data that was collected and curated by the Harvester. A filesystem approach was chosen as the most flexible approach. Working with Big Data requires a fast, reliable, scalable distributed file system. This file system will allow storing large amounts of data and will also enable parallel access to this data from within the AEGIS supported services. The processing services are integrated in a single multi-tenant data management platform that runs on top of the data store, and provides different data parallel processing scenarios.

1.1.1. Technologies to be used

AEGIS uses HopsFS as the main store for Big Data. HopsFS is a drop-in replacement alternative to HDFS. HopsFS overcome HDFS limitations by storing the namespace metadata into an external distributed database 'MySQL Cluster' that allowed HopsFS to adapt a multiple stateless namenodes architecture. HopsFS is compatible with HDFS clients, but it is recommended to use HopsFS clients since they provide a better load balancing between the namenodes. HopsFS provide the same HDFS APIs, however, only a subset of features are not supported such as snapshotting, heterogeneous storage, and access control lists.

To simplify data processing tasks, a multi-tenant data management platform is provided on top of HopsFS that is called Hopsworks. Hopsworks Integrated services are explained in more details in the next sections of the document.

It is recommended to access the files through Hopsworks since it provides an integrated view over the files and services. Also, it provides an intuitive design to add extended attributes to files/directories/datasets that are searchable within the whole cluster, project or dataset.

HopsFS and HDFS clients can be used in a Java project. Also, a RESTful API is supported 'WebHDFS' that can be used to manipulate the namespace.

1.1.2. API

Technical Interface	
Reference Code	XX#01
Function	HopsFS FileSystem
Subsystems	HopsFS
Type, State	
RPC, Synchronous	
Endpoint URI	

https://hadoop.apache.org/docs/stable/api/org/apache/hadoop/fs/FileSystem.html
Input Data
Unsupported Calls: ● (get|set|unset)StoragePolicy ● (get|set|list|remove)XAttr : At the moment adding extened metadata is done from Hopsworks ● (set|get|remove)Acl ● (create|rename|delete)Snapshot
Output Data

Technical Interface	
Reference Code	XX#01
Function	HopsFS WebHDFS
Subsystems	HopsFS
Type, State	
RESTfull-API, Synchronous	
Endpoint URI	
https://hadoop.apache.org/docs/stable/hadoop-project-dist/hadoop-hdfs/WebHDFS.html	
Input Data	
Unsupported Call: ● (set|get|unset)StoragePolicy ● (get|set|remove)XAttr ● (set|get|remove)Acl ● (create|rename|delete)Snapshot	
Output Data	

3.6. Hopsworks Integrated Services

3.6.1. Overview

The AEGIS platform provides data management and processing services for Big Data. Such data can be of sensitive nature and thus data access of different users and services has to be carefully managed. This behaviour is known as multi-tenancy and the AEGIS platform contains a service that manages this. The AEGIS data-store API is not exposed to outside users directly and instead a Project/Dataset Service is provided in the REST API to allow the exploration, import and export of data within one cluster. This service is complemented by another service that allows the exploration, import and export of data between different clusters using AEGIS. Free text search is also provided as a service to augment the data exploration capabilities of AEGIS. In order to manage computation on the AEGIS cluster, a resource manager service is also provided, accompanied by a number of data processing services. The integrated service contains a large number of services that require monitoring to detect that state of each of them and this is provided as a service itself - the monitoring service.

3.6.2. Technologies to be used

AEGIS provides data management and processing using Hopsworks integrated services. Hopsworks is a multi-tenant data management platform running on top of HopsFS and HopsYARN. It provides an integrated support for different data parallel processing such as Spark, Flink, and MapReduce, as well as scalable messaging bus with Kafka, and interactive notebooks with Zeppelin and Jupyter. Hopsworks is built on three main abstractions; Dataset, Project, and User. A Dataset is a directory subtree in HopsFS that can be shared between projects. A Project is a collection of datasets and users, as well as Kafka topics, Jobs (Flink, Spark, MapReduce), and notebooks (Zeppelin, Jupyter). Hopsworks supports two levels of users; platform-users and project-users. A platform-user can have as many project-users as number of projects he involved with. A project-user is used to enforce security requirement from Hopsworks on HopsFS.

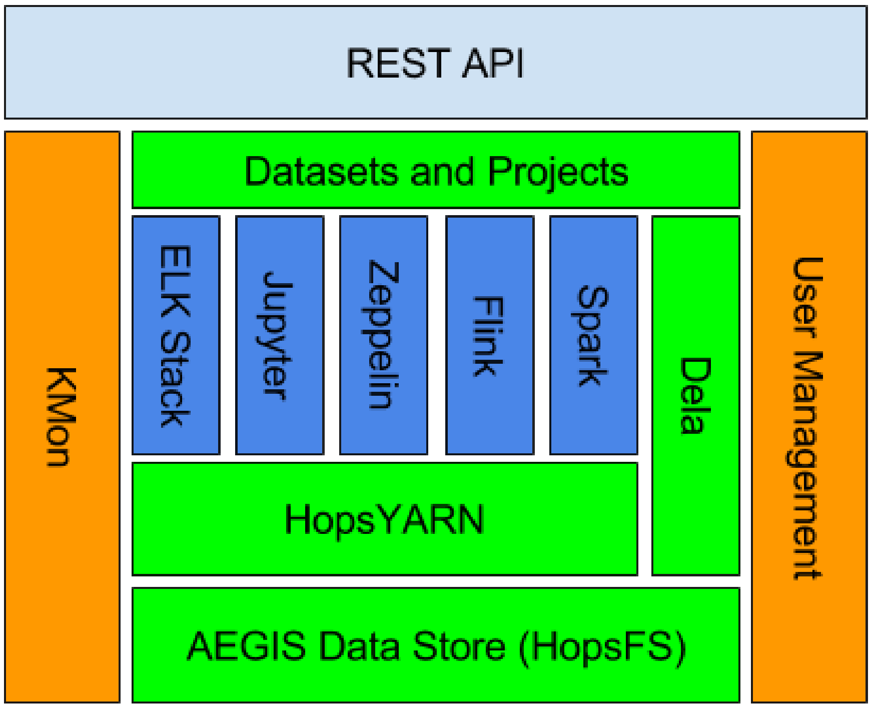


Figure 3-7: Hopsworks integrated services overview

3.6.3. Services

3.6.3.1. Users

3.6.3.1.1. Overview

Users can create accounts on Hopsworks cluster. Then, they can login using a username and password or with a 2-factor authentication. Hopsworks provides a messaging service for users where they can send and receive sharing requests to other datasets and projects within the cluster or across clusters.

3.6.3.1.2. API

The REST API documentation can be found in https://app.swaggerhub.com/apis/maismail/hopsworks-user-api/1.0.0

3.6.3.2. Projects and Datasets

3.6.3.2.1. Overview

A user can create projects, datasets, add members to projects, share datasets, run jobs, write notebooks, etc. A project contains multiple datasets, where a user can upload/download data into/from them.

3.6.3.2.2. API

The REST API documentation can be found in https://app.swaggerhub.com/apis/maismail/hopsworks-core-api/1.0.0

3.6.3.3. KMon - service monitoring

3.6.3.3.1. Overview

KMon or the Hopsworks monitoring service allows the retrieval of information of three different types: per host, per service, per role. The host provides resource utilisation like CPU usage for the whole machine that is running different services. The service provides resource utilisation like CPU per service like HopsFS, YARN, etc. The role provides resource utilisation per subservice, like hopsfs - namenode, datanode, etc.

Currently monitored services include: HDFS, NDB, YARN, dela, ELK, zookeeper, kafka, spark history server, map_reduce, livy, epipe.

3.6.3.3.2. API

The REST API documentation can be found in https://app.swaggerhub.com/apis/o-alex/kmon/1.0.0

3.6.3.4. Dela - transfer service

3.6.3.4.1. Overview

Dela transfer service provides capabilities to transfer and search for datasets between different instances of the Hopsworks platform running on separate clusters. There are three subservices included: Cluster, Dela and Hopssite. The Cluster and Dela subservice are both related to projects as the shared datasets are from within a parent project. The REST calls that are bound to a project will be found under a path prefixed by "/project/{id}".

The Cluster subservice allows the publishing and retrieval of datasets published for users outside the parent project, but still within the local cluster.

The Dela subservice allows the publishing and retrieval of datasets between clusters connected and registered with the Hopssite service. This service also allows search for datasets as well as retrieving information about the status of different dataset ongoing transfers.

The Hopssite subservice allows registration of clusters as well as allowing comments and ratings on published datasets.

3.6.3.4.2. API

The REST API documentation can be found in https://app.swaggerhub.com/apis/o-alex/dela/1.0.0

3.7. AEGIS Linked Data Store

3.7.1. Overview

The AEGIS Linked Data is responsible for storing the metadata associated with a particular dataset within the AEGIS platform. This metadata poses the foundation of the processing of the data within the AEGIS platform, since it offers detailed information about the semantic and syntax of the data itself. This allows to choose appropriate analysis and visualisation methods. The metadata will be stored as Linked Data, using the AEGIS ontology and vocabulary[24], which is based upon the DCAT-AP specifications. It will be developed as a service and integrated into the AEGIS data store.

3.7.2. Technologies to be used

The basis for the AEGIS Linked Data Store will be the Apache Jena Fuseki Triplestore[25]. For convenient access the access the SPARQL-endpoint of Fuseki will wrapped by a lightweight Java application based on Apache Jena. This application will offer a RESTful-API allowing the easy manipulation of the Linked Data.

3.7.3. API

Not applicable in the first iteration.

3.8. Query Builder

3.8.1. Overview

Query Builder was described in D3.1 as the component that provides the capability to interactively define and execute queries on data available in the AEGIS system. Query Builder is primarily addressed to the AEGIS users with limited technical background, but potentially useful for all, as it will simplify and accelerate the process of retrieving data and creating views on them, which could be then saved as new datasets or used as input for more high-level AEGIS tools, like the Visualiser and the Algorithm Execution Container.

The tool will be delivered as a widget inside the Apache Zeppelin notebook (presented in section 3.6), offering intuitive data browsing and selection facilitated through metadata-driven recommendations, both for individual datasets and combinations that can be performed. Apart from selecting the dataset(s), the user will be able also to refine the particular subsets to be included in the data view, if desired. This workflow corresponds to what the tool perceives as a query, which will be then executed in order to fetch the query result, i.e. a tabular view on the underlying dataset(s) to be consumed directly or used as input for visualisations and algorithm executions.

More specifically, regardless of the architectural changes, the tool will provide the same core features, described in D3.1 and briefly presented here updated:

- Allow users to create complex queries from a simple, easy-to-use GUI.
- Leverage AEGIS metadata schema in order to offer recommendations during query building.
- Provide an extensible, schema and database agnostic browser to minimise developer intervention when/if changes occur in the underlying AEGIS structures.

[24] https://github.com/aegisbigdata/aegis-ontology

[25] https://jena.apache.org/documentation/serving_data/

- Exploit common patterns for optimised querying of large volume data.

The high-level Query Builder workflow, also presented in Figure 3-8, is as follows:

a) Datasets available to the user are presented with some high-level information per each. This corresponds to the retrieval of metadata for datasets that are either already accessible by the user (owned by him or public) or private but open for discovery (i.e. private datasets with publicly published metadata allowing discoverability).
b) The metadata information that corresponds both to common dataset features (e.g. publisher) and specific features (e.g. field names and types) are exploited to guide the user through the selection of most appropriate for his needs dataset. Subsequently, if desired, the user may refine the selected dataset to include only a subset of the original data.
c) Selected data can be further combined with data from additional datasets, following a process similar to the one described above, but this time receiving recommendations adjusted to the already selected data view in order to ensure that linking points exist.
d) The previously defined filters and transformations are applied and the final data view is retrieved. This corresponds to the query execution, after which its results will be previewed and available for further use in the system.

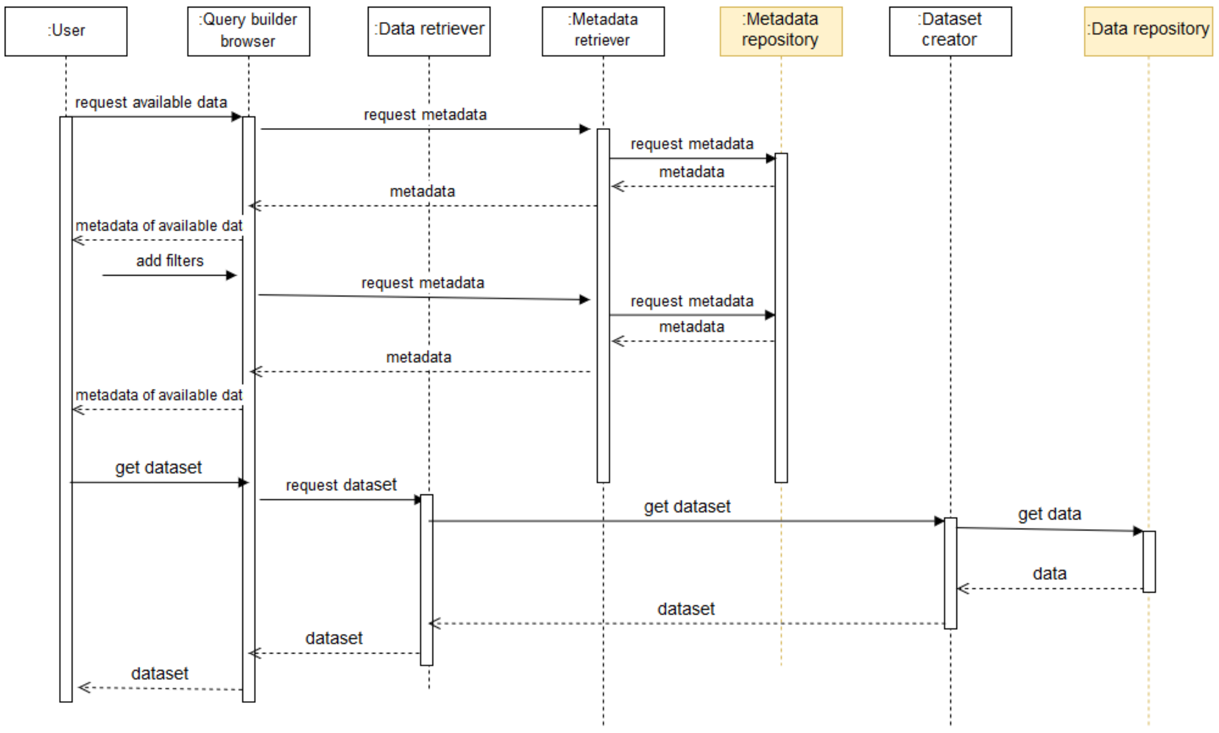


Figure 3-8: Query building and execution workflow

3.8.2. Technologies to be used

In D3.1 it was stated that Query Builder will be developed as an extension of the LinDA Query Designer[26], a web platform enabling intuitive drag and drop query building and execution on

[26] http://linda.epu.ntua.gr/query-designer/

RDF data. Having identified the challenges imposed by the volume of data in AEGIS and the need to support non-RDF data, a big data enabled solution was originally designed, aiming to restructure the backend to improve query efficiency, adopting and adapting the existing intuitive UI.

However, based on the decision to utilise Hopsworks Integrated Services (presented in section 3.6) and, consequently, to leverage the capabilities of the provided notebooks and more specifically Apache Zeppelin, Query Builder has been adapted to the updated architecture to enable big data query building and result generation capabilities. Therefore, modules for recommendations and data compatibility from LinDa, developed in Python, will be integrated in the AEGIS Query Builder, which will be implemented as extensions to the Apache Zeppelin notebook, using JavaScript, Python and the provided notebook APIs.

3.8.3. API

No API will be provided in the first version. The provision of an API will be reconsidered in the course of the project in case any new needs emerge that require interaction mechanisms with the Query Builder or if the functionalities of the tool can be leveraged in some way external to the current, integrated with the notebooks, workflows.

3.9. Visualiser

In the following sections the component design, the technologies that will be used, as well as the interface specifications of the Visualiser component are described. The Visualiser component was specified in deliverable D3.1 as the component that aims to provide visualisation capabilities for the output of the analysis results coming from the Algorithm Execution Container component. The delivery of the Visualiser component will address the current challenges for advanced visualisations by the AEGIS platform, providing a variety of visualisation formats which spans for simple static charts to interactive charts offering several layers of information and customisation.

3.9.1. Overview

The Visualiser component is composed of several internal subcomponents, each one supporting the execution of the visualisation process with the aim of providing advanced visualisations to the end-user. Each subcomponent provides the necessary interface to support the interconnection with the rest of the subcomponents. The Visualiser component comprises the following main services:

- Recommender service: The purpose of this service is to provide recommendations for the appropriate visualisation type based on the selected dataset before the visualisation process is started
- Selector service: This service is responsible for providing the list of the available visualisation of the AEGIS platform to user and for starting the visualisation process.

- Configurator service: The purpose of this service is to display the available parameters based on the selected visualisation format. Also, the service is responsible for providing recommendations on the parameter values based on predefined values depending on the selected dataset type.
- Generator service: This service will perform the visualisation generation based on the input provided by the Selector and Configurator service. This service is responsible also for storing the generated visualisations for later use in dashboards in the upcoming steps of the process.
- Dashboard Compiler service: This service will undertake the responsibility of creating interactive dashboards where several results stored by the Generator service will be displayed in multiple ways.
- Dashboard Publisher service: This service is responsible for publishing the dashboards created by the user as AEGIS services which can be consumed by other stakeholders.

The Visualiser service will be orchestrating these services towards the execution of the visualisation process. Figure 3-9 depicts the visualisation process and how the several services interact with each other.

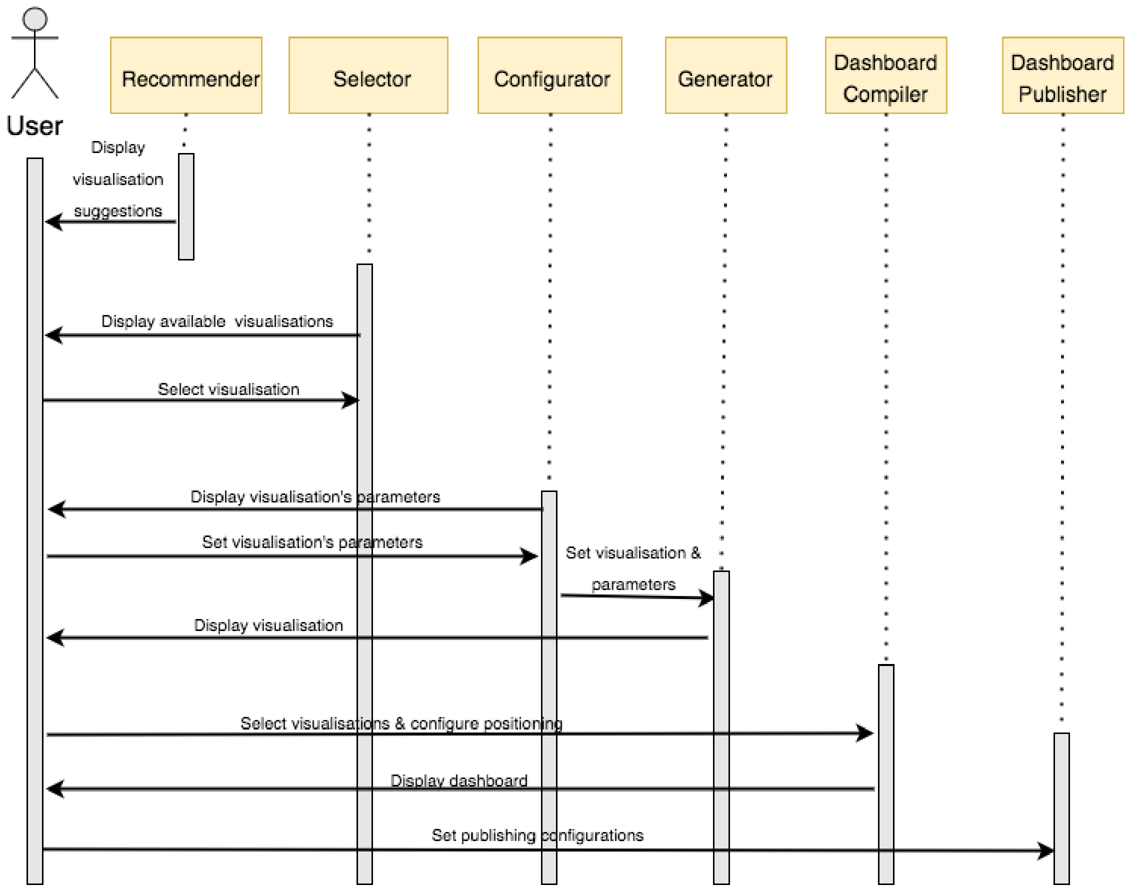

User
Recommender
Selector
Configurator
Generator
Dashboard Compiler
Dashboard Publisher
Display visualisation suggestions
Display available visualisations
Select visualisation
Display visualisation's parameters
Set visualisation's parameters
Set visualisation & parameters
Display visualisation
Select visualisations & configure positioning
Display dashboard
Set publishing configurations

Figure 3-9 - Sequence diagram of the visualiser component

3.9.2. Technologies to be used

AEGIS Visualiser will be a Web application which will based on the Apache Zeppelin[27] that is already integrated in Hopsworks. Apache Zeppelin is a multipurpose interactive web-based notebook for running Spark or Flink code on Hops YARN. Zeppelin is enabling data engineers, data analysts and data scientists to be more productive by providing functionalities for data visualisation along with functionalities for data ingestion, data discovery and data analytics. Zeppelin supports the notebook functionality which allows users to develop, organise, execute and share data code and visualising results but also to interactively work with long workflows.

Zeppelin currently supports Python programming language along with a growing list of other programming languages or data processing backends such as Scala, Hive, SparkSQL and R by implementing the interpreter concept. Interpreters allow any language or data processing backend to be plugged into Zeppelin. Apart from the growing list of the available interpreters, Zeppelin provides an easy way to develop new custom interpreters upon needs.

For the purposes of the AEGIS platform, Visualiser will utilise the functionalities of Zeppelin through the API provided by Zeppelin. The REST API provided by Zeppelin supports interaction and remote activation of Zeppelin's functionalities. Among others, it supports providing configurations to Zeppelin core engine or the interpreters and notebook related functionalities (like List, Create, Get, Delete, Clone, Run, Import, Export).

Zeppelin offers the following out-of-the-box visualisations:

- Tabular Data
- Bar chart
- Area chart
- Line chart
- Pie chart
- Scatter plot

For more advanced visualisations the following python libraries will used:

- **matplotlib**[28]: Matplotlib is Python 2D plotting library which produces high quality visualisations in a variety of formats and can generate plots, histograms power spectra, bar charts, error charts, scatterplots and many more.
- **ggplot**[29]: Ggplot is plotting system for Python based on R's and it is capable of creating highly customised data visualisations.

[27] http://zeppelin.apache.org/

[28] https://matplotlib.org/

[29] http://ggplot.yhathq.com/

- **plotly[30]**: Plotly is an interactive, browser-based graphing library for Python. Plotly is a high-level, declarative charting library with over 30 chart types, including scientific charts, 3D graphs, statistical charts, SVG maps, financial charts, and more.

Apart from Apache Zeppelin which will be the base for the Visualiser on the first version, during the implementation of the second version of the Visualiser the integration with Shiny[31] will be evaluated. Shiny is a framework, provided as an R package, for developing interactive charts, data visualisations and building interactive web applications using the R language. It consists of a library which renders HTML and JavaScript and a server-side application; this server-side application allows R code to be interpreted and updated based on interactions with the browser-embedded Shiny element. It supports highly customisable widgets and prebuilt output widgets for displaying interactive plots and dashboards.

The following R libraries can be used in combination with Shiny:

- **networkD3[32]**: networkD3 provides tools for creating D3 JavaScript network graphs from R.
- **Leaflet for R**[33]: Leaflet provides interactive maps with features like interactive panning/zooming, maps composed using arbitrary combinations of map tiles, markers, polygons and GeoJSON
- **Google Charts with googleVis**[34]: googleVis package provides an interface between R and the Google's charts tools. It allows users to create web pages with interactive charts based on R data frames.
- **ggplot2[35]**: ggplot2 is a plotting system for R, based on the grammar of graphics. It takes care of making plots in an easy and efficient way as well as providing a powerful model of graphics that makes it easy to produce complex multi-layered graphics.

3.9.3. API

In the first version of the Visualiser component an API will not be provided. However, Visualiser will exploit the API provided by Zeppelin and the relevant information is provided in the table below for reference.

Technical Interface	
Reference Code	XX#01

[30] https://plot.ly/d3-js-for-python-and-pandas-charts/

[31] https://shiny.rstudio.com/

[32] http://www.htmlwidgets.org/showcase_networkD3.html

[33] http://rstudio.github.io/leaflet/

[34] https://github.com/mages/googleVis/

[35] http://ggplot2.org/

Function	Zeppelin integration
Subsystems	Visualiser
Type, State	
RESTfull-API	
API Documentation	
http://zeppelin.apache.org/docs/0.7.3/rest-api/rest-notebook.html	
Indicative Endpoints Create a new note Get note info List of the notes Delete a note Run all paragraphs Stop all paragraphs Create paragraph Delete paragraph Update paragraph configuration Run paragraph asynchronously Run paragraph synchronously	

3.10. Algorithm Execution Container

3.10.1. Overview

Algorithm execution in AEGIS will take place over notebooks, which will be used by users to select analysis data and then perform analyses using the underlying AEGIS infrastructure.
The Algorithm Execution Container is regarded as an upper-layer of the algorithm execution methods available in notebooks. In order to provide extra functionalities to both novel and non-expert users, this component will feature an algorithm selection template, offering to users some basic information regarding each algorithm available in the big data analysis platform.

Moreover, the selection will be also extended to specific parameters of each algorithm, to let users better understand and trim parameters for their analyses.
Finally, algorithm execution will be performed over the notebook's module, and results will be returned to it, which can then be used for visualisation or be stored.

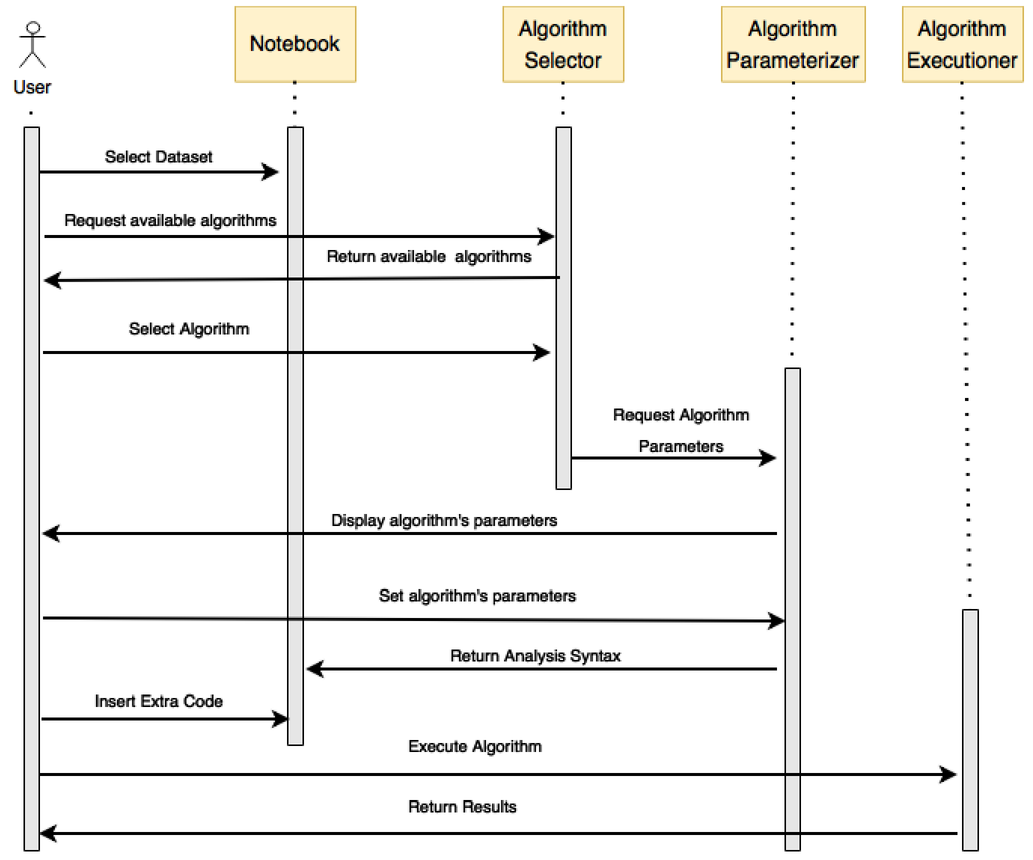


Figure 3-10: Algorithm Execution Container sequence diagram

1.1.2. Technologies to be used

The algorithm execution container will be based on Python, extending the Apache Zeppelin notebook with an easy to use algorithm library that will contain pre-configured algorithms, exposing selected configurable parameters that could be changed by the user.

3.10.2. API

No API will be provided in the first version.

3.11. AEGIS Orchestrator

3.11.1. Overview

AEGIS Orchestrator is the component responsible for the integration of services and components within the context of the AEGIS platform, which includes integrated in Hopsworks or external to Hopsworks services, via the exposed APIs of each of the components. Thus, Orchestrator will build the automated and managed flow of information between the various services and components of the AEGIS platform allowing the administrator of the dataflow to send, receive and route data in an automated way. The Orchestrator will provide the aggregation of various services and components by enabling "data pipes" of information from different sources and the ability to setup the rules on how that information should be transmitted and modified. To achieve this, the Orchestrator is highly dependent on the exposed APIs of the AEGIS components as it will utilise the APIs to communicate with the rest of the components for the execution of the dataflow. The Orchestrator will provide a list of basic conversions, for example from JSON to XML and vice versa, in order to facilitate the integration of the components in the dataflow.

The Orchestrator will provide the means to the administrator configure the desired flow of information via a web-based user interface. Through this interface, the administrator will be able to manage these flows in a user-friendly way by incorporating a set of fine-grained data provenance tools. This will provide an easy to use and reliable orchestration tool for processing and distributing data acting as a mediator between the components. It should be noted at this point that the Orchestrator will be provided only to the administrator of the AEGIS platform thus the functionalities and the described user interface will not be available for the rest of the users of the platform.

The concept of the orchestrator will be to create flows. Flows will be made up of a series of actions each with a specific task. These flows can be saved and can be reused to produce more complex flows easily. The Orchestrator will also focus on providing sufficient security, interactivity, scalability and data provenance measuring details on data origins, how, where and why data are transmitted.

At first, the Orchestrator will focus on providing the flow information between the components of Query Builder, Algorithm Execution Container and the Visualiser. The Orchestrator will provide the functionalities needed to interconnect those components via their exposed APIs. For the first version, these components will be tightly coupled with notebooks, which are more suitable for developers and expert users. To support the non-expert users the interconnection between those components is needed and the Orchestrator will provide the means to achieve this in an abstract and more user-friendly way for the non-expert users. Also, as the project evolves support for more components of the AEGIS platform will be evaluated such as the Cleansing Tool, the Anonymisation Tool and the Data Harvester.

In the first version of the AEGIS platform since most of the components mentioned above will not provide their APIs, the efforts will be focused on the implementation of the Orchestrator and the preparation steps needed so that the Orchestrator will be ready to use once these APIs are available in the upcoming versions of the platform.

3.11.2. Technologies to be used

The implementation of the Orchestrator will be based on specifically Java 8 and the Spring Framework[36]. Spring Framework is a powerful lightweight application framework that makes it easy to create Java enterprise applications in an efficient and modular way. Spring supports a wide range of application scenarios by providing an easy and elegant way to use the existing technologies built around Java.

Thymeleaf is a Java template engine for processing and creating HTML, XML, JavaScript, CSS, and text. Thymeleaf is extremely extensible and its templating capability ensures templates can be prototyped easily without a back-end which makes development very fast compared to other popular template engines. Thymeleaf provides modules for Spring Framework enabling their integration easily.

3.11.3. API

In the first version of the platform the Orchestrator will not provide any API. In the course of the project and based on the upcoming needs of the project, this decision will be reconsidered.

3.12. AEGIS Front-End

One of the AEGIS platform goals is to provide to the AEGIS stakeholders a user-friendly interface to surf easily in the platform while exploiting its services; from the technical requirements (D3.1) the following (Table 3) concern the most important presentation layer requirements.

The Requirements Table presented in D3.1 did not explicitly include references to the Front-end component, as the architecture described there was an early version of the AEGISs platform. In addition, some new technical requirements have been identified. These new requirements are presented here and have been also added in the updated AEGIS requirements table presented in Appendix A.

Table 3: Technical Requirements related to the AEGIS Front-End component from D3.1

Id	Component	Description
TR52	Data Management, Visualiser	Support overview of resources
TR53	Visualiser	Preview a small selection of the results of the generated query

[36] https://spring.io/

TR64	Visualiser, Query Builder, Data Harvester, Data Annotator	Provide secure user-friendly interface and flexible navigation
TR70	Front-End	Support a set of functionalities for the processing of the output result such as save, export, share etc.
TR71	Front-End	Provide means for processing the selected datasets and include constraints towards the preparation of the desired output.
TR72	Front-End	The AEGIS Front-End will be designed in order to allow even a non-expert user to experience with the AEGIS services.
TR73	Front-End	The cross-browser and cross-platform compatibility (accessibility) will be ensured by responsive web design. Another key point of the development to be ensured, is the performance (render time).

3.12.1. Overview

The AEGIS Front-End, as shown in Figure 2-2, is the upper layer of the whole AEGIS architecture, receiving and sending the outputs/inputs from/to the Aegis services module and from/to the AEGIS Orchestrator.

The first step to access the platform will be the creation/authentication of an account and the creation/selection of a project. Each user will have a specific role that allows the exploitation of some AEGIS functionalities while blinding other functionalities.

The Front-End will facilitate all the Aegis platform web components which have an interaction with the user (e.g. Harvester, Query Builder, Visualiser, Orchestrator), starting with a common graphical environment which provides direct links to the single web components, exploiting their graphical user interfaces (GUIs) rather than re-build the GUIs within the Front-End.

The following pictures represent the sequence diagram related to the AEGIS Front-End.

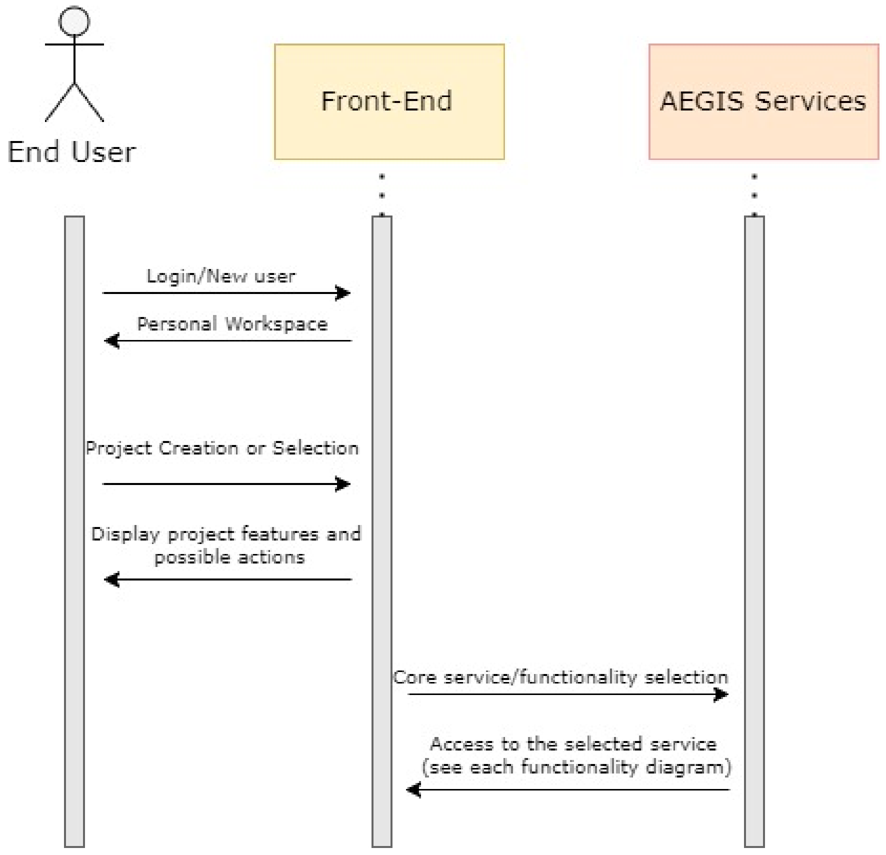

End User
Front-End
AEGIS Services
Login/New user
Personal Workspace
Project Creation or Selection
Display project features and possible actions
Core service/functionality selection
Access to the selected service (see each functionality diagram)

Figure 3-11: AEGIS front-end

3.12.2. Technologies to be used

The AEGIS Front-End will be built on top of the Hopsworks platform. Hopsworks is a self-service User Interface (UI) for Hops Hadoop, which introduces new concepts needed for project-based multi-tenancy: projects, users, and datasets. All jobs and interactive analyses are run from the HopsWorks UI and Apache Zeppelin (an iPython notebook style web application). For further details, please see D1.1 – Domain Landscape Review and Data Value Chain Definition paragraph 2.8.3. Scalable Data stores.

The Front-End module will be developed exploiting mainly three specific technologies:

- AngularJS framework
- Java Server Faces (JSF)
- PrimeFaces (implementation of JSF)

AngularJS is a toolset for building the framework most suited to application development. It is fully extensible and works well with other libraries. Every feature can be modified or replaced to suit unique development workflow and feature needs. AngularJS is often used for creating single-page applications, where only certain portions of the page (sub-views) are updated as a result of the user's actions or data being sent from the server. Other features include two-way data-binding, reusable components, deep linking, built-in services for backend communication and localisation support. Mobile Angular is another User Interface framework that is built on AngularJS and Bootstrap for mobile-friendly application development.

JavaServer Faces (JSF) is a Java specification for building component-based user interfaces for web applications and was formalised as a standard through the Java Community Process being part of the Java Platform, Enterprise Edition. It is also a MVC web framework that simplifies construction of user interfaces for server-based applications by using reusable UI components in a page. JSF 2 uses Facelets as its default templating system. Other view technologies such as XUL or plain Java can also be employed. In contrast, JSF 1.x uses JavaServer Pages (JSP) as its default templating system.

PrimeFaces is an open source user interface component library for JavaServer Faces based applications. It features over 100 UI components, Ajax Framework, Push Framework, Dialog Framework, Client Side Validation, Theme Engine and Search Expression Framework.

3.12.3. API

No API will be provided in the first version.

3.13. Holistic Security Approach

3.13.1. Overview

In deliverable D3.1 a holistic security approach that should be incorporated throughout the AEGIS platform was described. This approach included security aspects applicable to the whole lifecycle of the data exploitation covering the security aspects of data in storage, in transit and in use.

With the adoption of Hopsworks some of the aspects described in D3.1 were covered. For the data in storage which includes all raw data and metadata stored in the platform infrastructure Hopsworks has embraced a new implementation of Hadoop Filesystem (HDFS) called HopsFS. In deliverable D3.1 a number of candidate technologies for securing data in storage have been presented such as Symmetric Encryption Algorithms, Asymmetric Encryption Algorithms and Attribute-Based Encryption. These candidates were proposed to address the preservation of the security, privacy and integrity of data that is stored within the AEGIS infrastructure. After further evaluation, it was decided that such technologies are not ideal for big data infrastructures and for platforms performing data analysis. The main reason is that these technologies introduce efficiency problems and delays in data handling in the course of security. Big data analytics frameworks are however highly dependent in performance and in accessing data for the analysis in order to perform the analysis in a timely and efficient manner. For this reason, for the security and integrity control the effort of introducing the usage of checksum will be evaluated for the data in storage. Checksum is used to verify the integrity of data by verifying that the data have not been altered or corrupted. The value of checksum is usually calculated using cyclic redundancy check (CRC) algorithms and cryptographic hash functions.

For the security of data in transit or data in motion, which includes data actively moving across the internet or through the private network Hops supports Secure Sockets Layer (SSL) and Transport Layer Security (TLS) encryption at the RPC layer providing encryption and authentication in all internal communication links. SSL/TLS encryption was also one of the candidate technologies identified by the AEGIS technical partners.

The third aspect of the holistic security approach is related to security of "data in use" which refers to data at-rest state, residing on one particular node of the network (for example, in resident memory, swap, processor cache or disk cache). In deliverable D3.1 the AEGIS technical partners have pointed out a number of candidate technologies including Homomorphic Encryption and Verifiable Computation to guarantee controlled access to this data and privacy preservation. These candidates will not be included in the current version and will be further evaluated in the upcoming versions of the AEGIS platform.

The holistic security approach includes the security aspects for the technical interfaces (e.g. REST) as exposed by the various AEGIS components and the data access control which includes authorisation, authentication and access approval. Currently Hopsworks REST API is session based with JSession tokens and every registered user holds a x509 certificate for every project by the user and JDBC Realm or LDAP for user authentication. For the authentication and authorisation purposes of the AEGIS platform along with data access control operations the effort to introduce a token based authentication with JSON Web Token[37] (JWT) will be evaluated. JWT defines a compact and self-contained way for securely transmitting information between communicating parties as a JSON object. This information can be verified and trusted since it is digitally signed.

3.13.2. Technologies to be used

- Checksum is a small-sized datum derived from a block of data, usually a file for the purpose of detecting errors or modifications that have been introduced when the file in transit or in storage. Checksum is the outcome of running an algorithm, called cryptographic hash function or checksum algorithm, on the block of data. For almost every checksum algorithm, even for small changes the outcome will be different. Checksums will be used to identify data corruption error and overall data integrity.
- TLS and SSL are cryptographic protocols enabling communication security over the network. It is the standard technology for keeping an internet connection secure and safeguarding any data in transit between two communicating parties, preventing unauthorised read and modification of any information transferred. TLS is an updated, more secure version of SSL. More specific, TLS provides privacy and data integrity by using symmetric cryptography to encrypt data in transit with unique keys for each connection created which are based on a shared secret negotiated at the start of the session. The identity of the communicating parties is authenticated using public-key cryptography while for the data transmitted integrity checks are performed against undetected loss or modification with the use of message authentication code.
- JSON Web Token (JWT) is an open standard (RFC 7519) is a means of representing claims to be transferred between two parties. The claims in a JWT are encoded as a JSON object that is digitally signed using JSON Web Signature (JWS) and/or encrypted using JSON Web Encryption (JWE). JWT is used for authentication purposes, once the user is logged in, each subsequent request will include the JWT, allowing the user to

[37] JSON Web Tokens, https://jwt.io/

access routes, services and resources that are permitted with that token. JWT is also used for information exchange, more specific for securely transmitting information between communicating parties. JWTs can be signed using public/private keys providing authentication for identity of the sender. Additionally, as the signature is calculated using the payload and the header providing verification that the content has not been tampered.

3.13.3. API

Not applicable.

4. USER INTERACTION WORKFLOWS

The current section presents the main workflows that AEGIS will enable towards data-driven innovation in the PSPS domains. All workflows are modelled in BPMN diagrams and are focused on the user perspective. The target here is not to show the technical details, which were covered in section 3, hence the diagrams do not show how AEGIS components interact exactly and how each component internally processes its tasks. Each workflow corresponds to a specific provided functionality of the AEGIS framework, which may be based on one or more of the components presented in section 3. By chaining and combining these workflows, all AEGIS scenarios and identified user requirements are covered.

4.1. Sign-up and Login

The following figure shows the interaction between the user and the AEGIS UI. A user first signs-up for an account by filling the form on the UI, then the AuthService consult the CustomAuthRealm for authenticating the user, which in turns sends and email to the user's email for verification. Once the user verifies his/her email, he/she can access their account through the login form on the AEGIS UI.

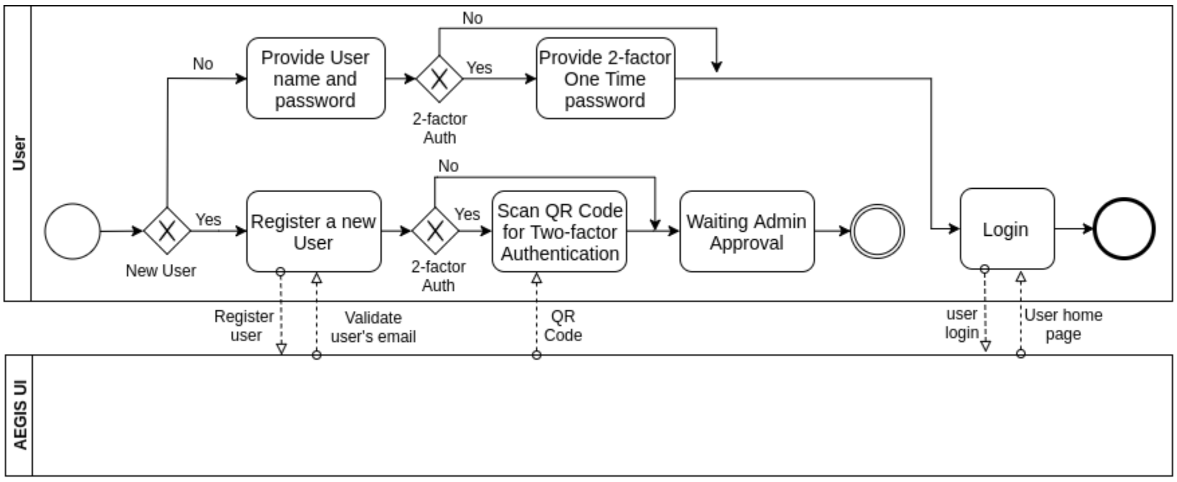


Figure 4-1: Sign-up and Login workflow

4.2. Data import

4.2.1. Main workflow

The Figure below presents a workflow of user interaction with the AEGIS harvester for registering a new dataset in AEGIS. The workflow shows the required user actions for configuring the harvester for a dataset metadata registration/import as well as its data import and transformation to the target format.

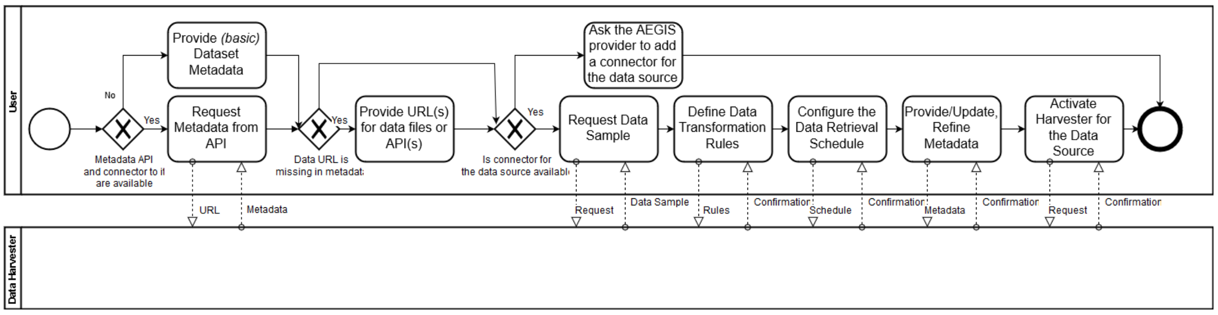

User
Data Harvester
Metadata API and connector to it are available
No
Yes
Provide (basic) Dataset Metadata
Request Metadata from API
Data URL is missing in metadata
Provide URL(s) for data files or API(s)
Is connector for the data source available
Ask the AEGIS provider to add a connector for the data source
Request Data Sample
Define Data Transformation Rules
Configure the Data Retrieval Schedule
Provide/Update, Refine Metadata
Activate Harvester for the Data Source
URL
Metadata
Request
Data Sample
Rules
Confirmation
Schedule
Request

Figure 4-2: Registering a new dataset

4.2.2. Anonymisation workflow

The figure below shows the actions undertaken by the user in order to anonymise their data through the provided anonymisation tool, prior to importing them to the AEGIS web platform.

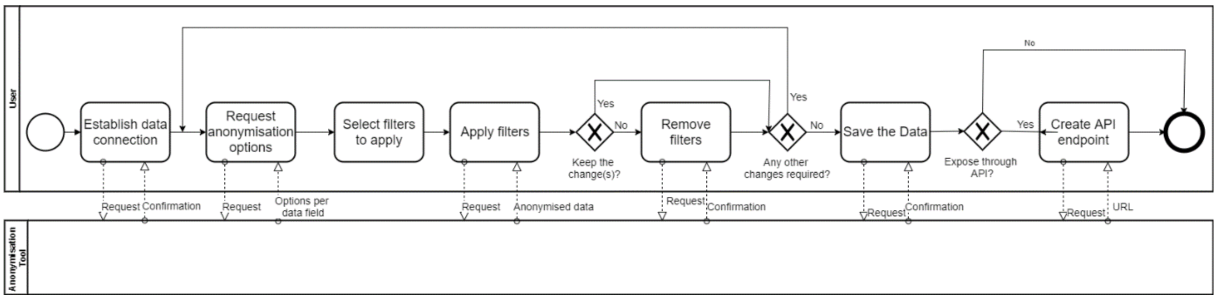


Figure 4-3: Data anonymisation workflow

4.2.3. Data cleansing workflow

The Figure below presents the user interaction with the Data Cleansing tool to cleanse the data. As explained in section 3.2, AEGIS will follow a two-fold approach for data cleansing: the first entails offline data processing using existing mature data cleansing tools, whereas the second one will be achieved through leveraging the Algorithm Execution Container component, hence the workflow for the latter is covered through the service creation workflow presented in section 0. For the first approach, multiple tools will be proposed and tested, hence slight differentiations in the workflow may occur. However, cleansing tools generally support a common intuitive workflow, which is presented in the following figure.

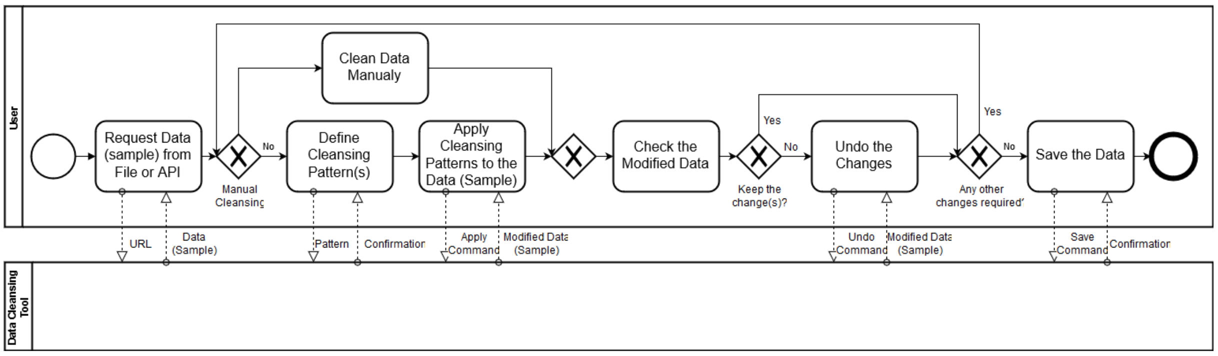


Figure 4-4: Data cleansing workflow

4.3. Data and service exploration: search and filter

4.3.1. From the main AEGIS platform

The AEGIS platform uses the Hopsworks integrated service as described in section 3 to provide the capability to explore data both through a REST API as well as through the graphical interface.

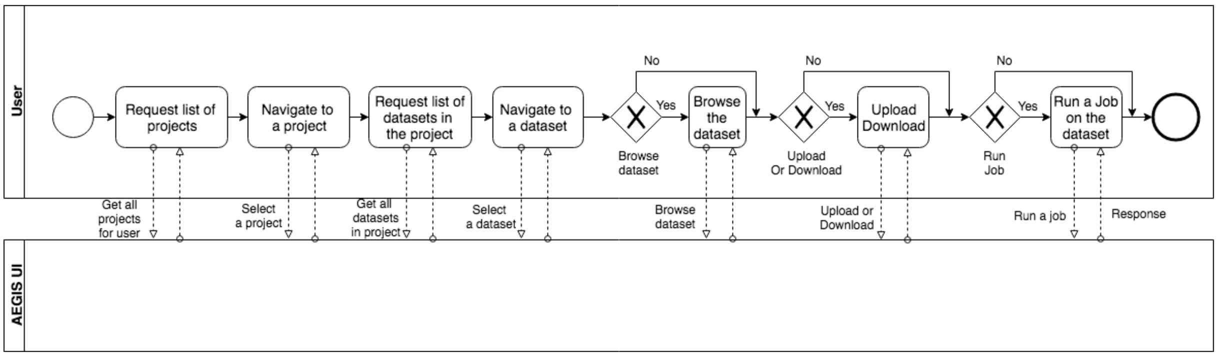


Figure 4-5: Data and service exploration workflow

4.3.2. Using query builder

The following two figures show the user's perspective when using query builder to find data and create an appropriate dataset to be fed into analysis and/or visualisation or to be saved as a new dataset. Although the user primarily interacts with the Query Builder component, the AEGIS Linked Data Store and the Brokerage Engine are also utilised in the background. Dataset acquisition sub-process shown in the diagram is an instance of the artefact sharing process described in Figure 4-9.

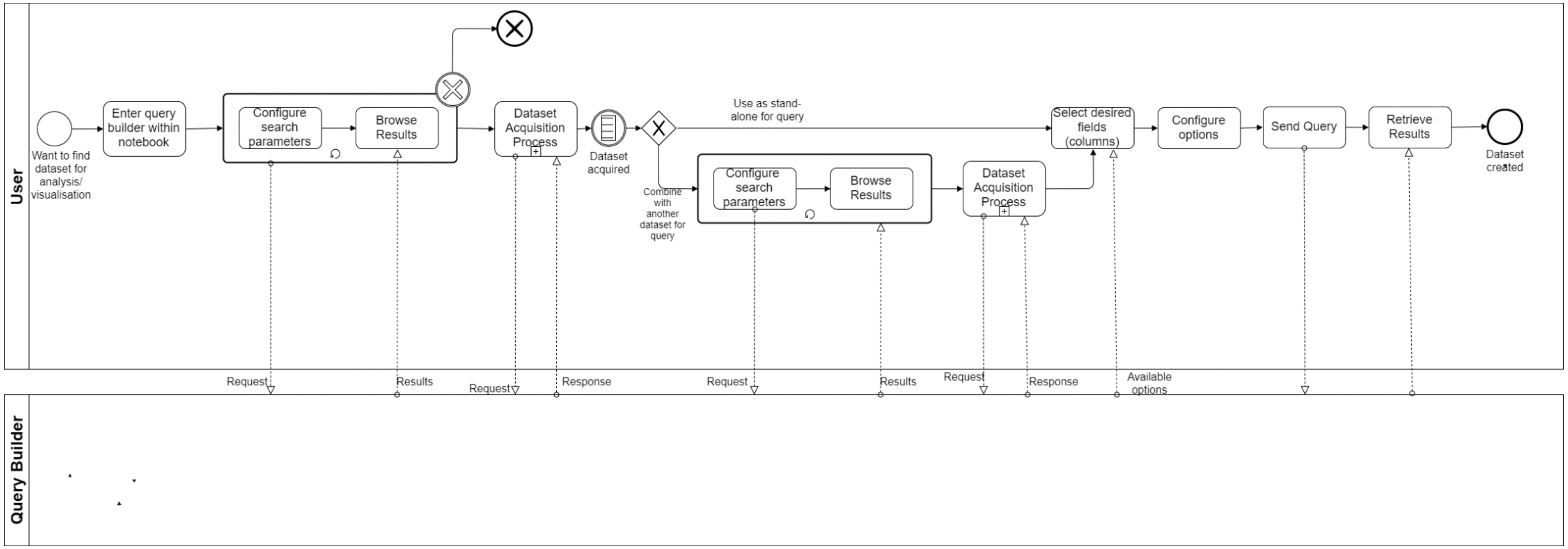

User
Query Builder
Want to find dataset for analysis/ visualisation
Enter query builder within notebook
Configure search parameters
Browse Results
Dataset Acquisition Process
Dataset acquired
Use as stand-alone for query
Combine with another dataset for query
Configure search parameters
Browse Results
Dataset Acquisition Process
Select desired fields (columns)
Configure options
Send Query
Retrieve Results
Dataset created
Request
Results
Request
Response
Request
Results
Request
Response
Available options

Figure 4-6: Dataset exploration through query builder workflow

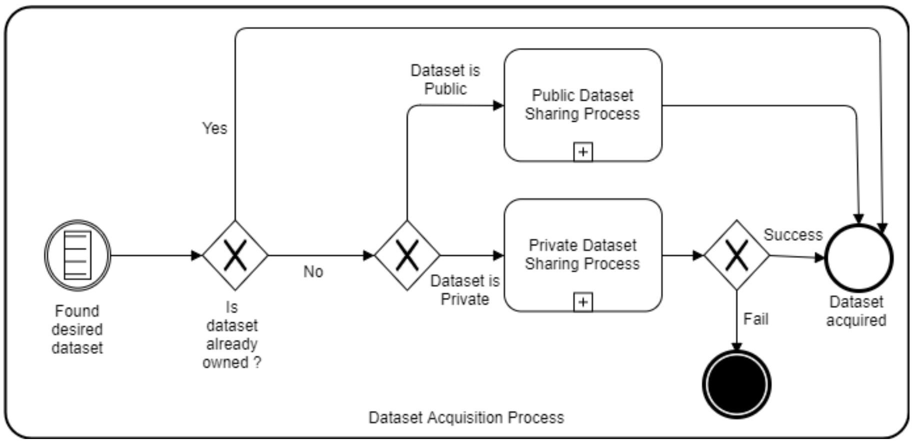

Yes
Dataset is Public
Public Dataset Sharing Process
No
Private Dataset Sharing Process
Dataset is Private
Success
Fail
Found desired dataset
Is dataset already owned ?
Dataset acquired
Dataset Acquisition Process

Figure 4-7: Data acquisition sub-process workflow

4.4. Data export from AEGIS

The following figure shows the different alternatives for a user to get their data exported from with AEGIS into either their local computer through a local download or to other clusters using Dela sharing service provided by AEGIS.

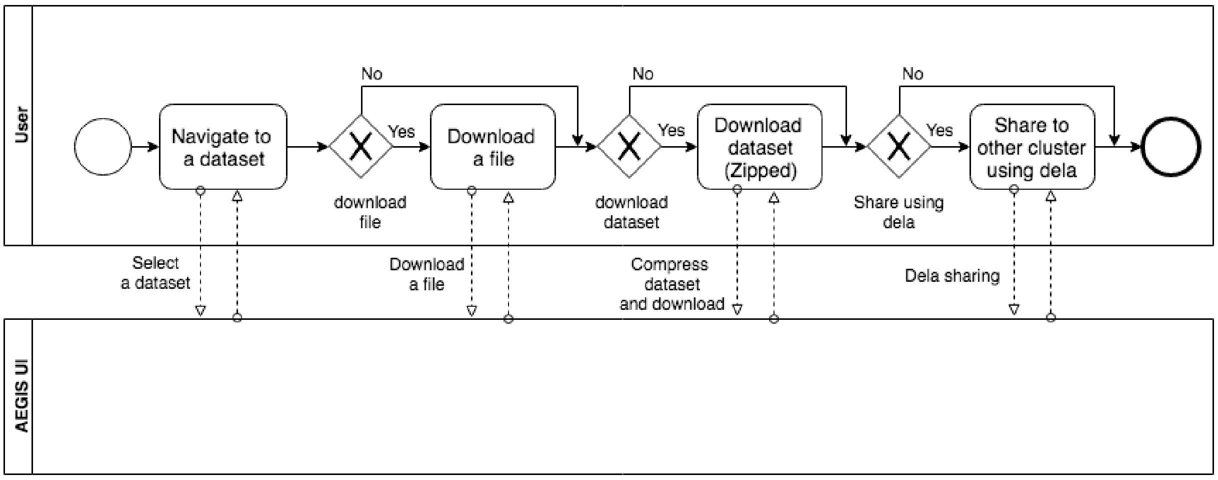


Figure 4-8: Data export workflow

4.5. Artefact sharing/reuse

The following figure shows the workflow for data sharing over the AEGIS platform. The operation to be performed will invoke the Brokerage Engine, which will check if artefact sharing/reuse can be performed.

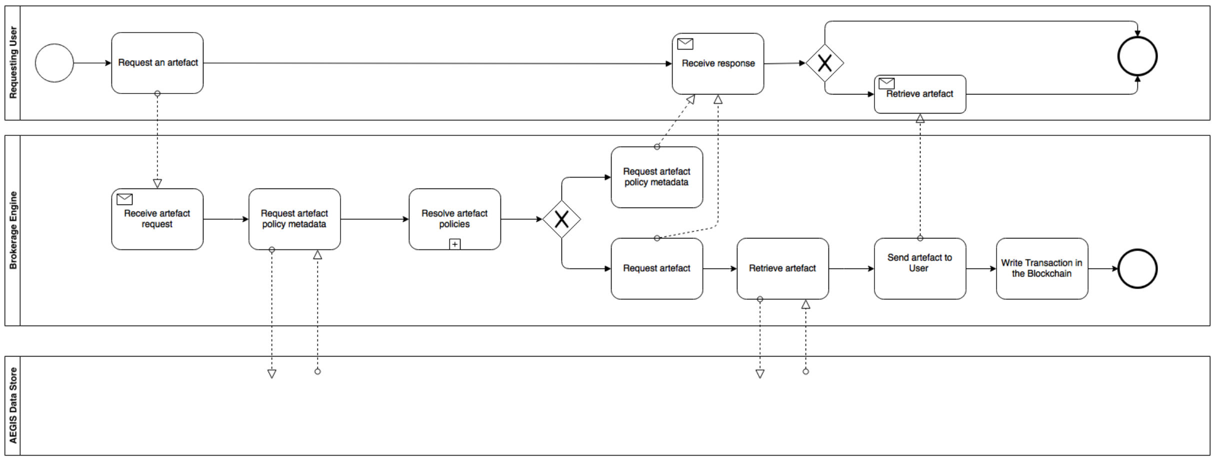


Figure 4-9: Artefact Sharing Workflow

4.6. Service creation

The following figure presents the user interaction for service creation, which involves Algorithm Execution Container and Visualiser. The user is able to request visualisation from the list of available visualisations by the AEGIS platform or a custom visualisation, on a selected dataset or on the dataset created from the Query Builder execution. Moreover, the user is able to perform analysis, also by chaining execution of algorithms, and request visualisation on the analysis results.

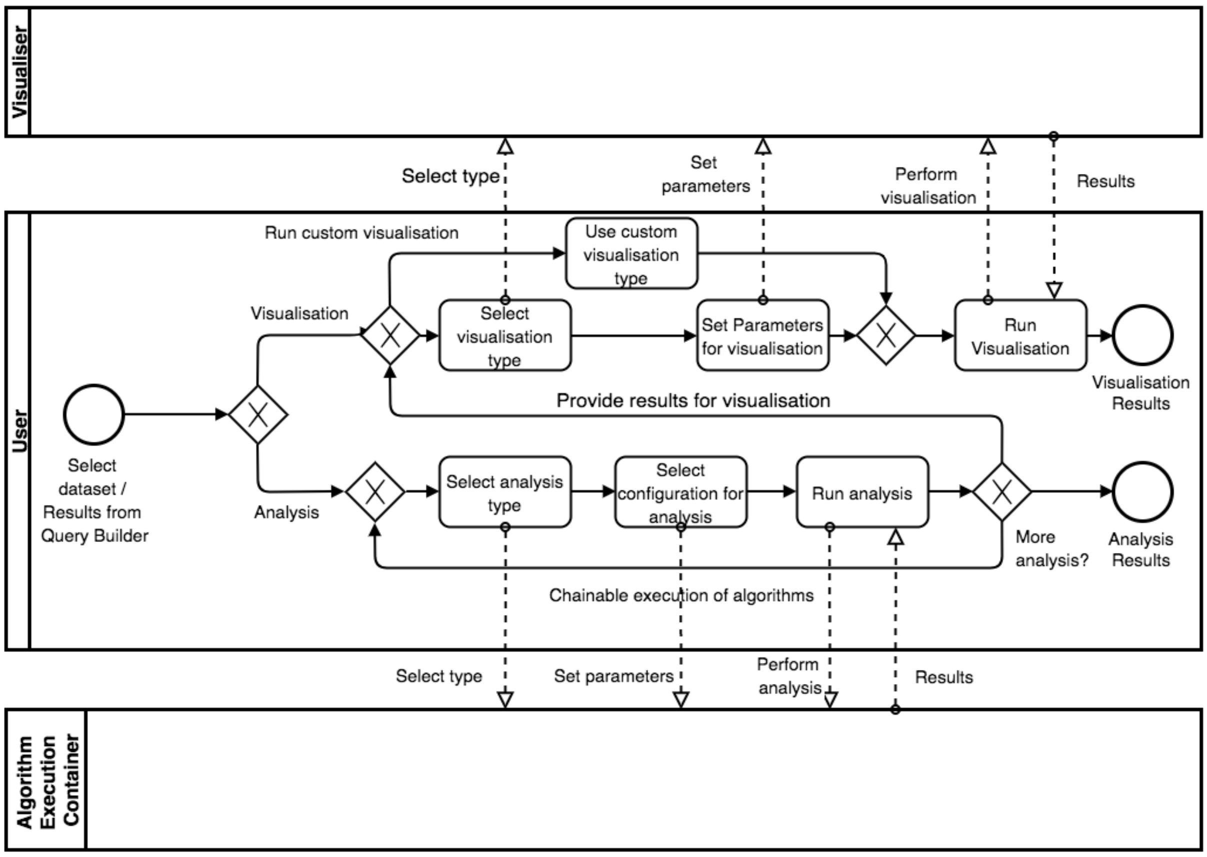


Figure 4-10: Service creation workflow

4.7. Service Consumption

The following figure shows a first draft of the user's perspective when using the AEGIS platform to perform a service consumption workflow, including account creation/authentication, project creation/selection and search functionalities related to data, visualisations and/or reports.

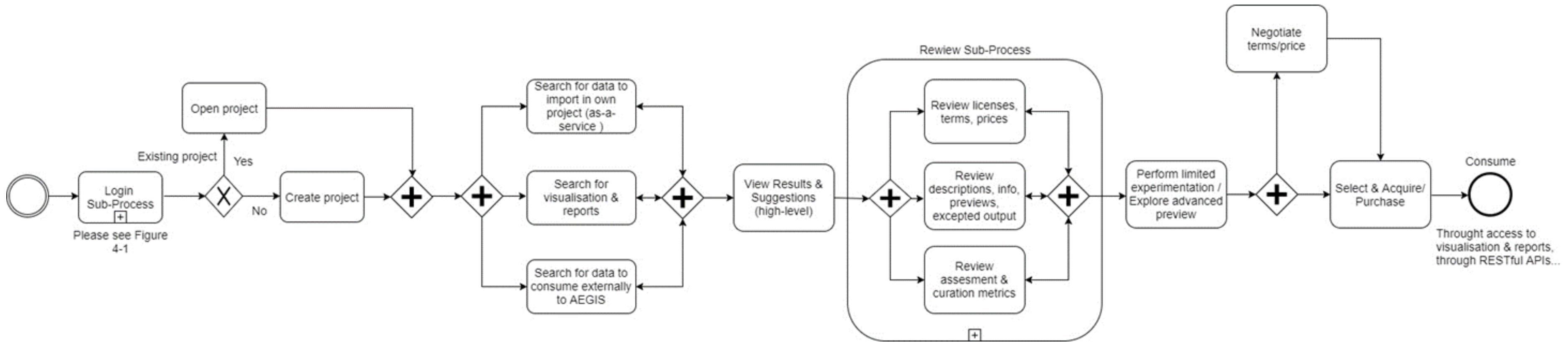

Login Sub-Process
Please see Figure 4-1
Existing project
Yes
No
Open project
Create project
Search for data to import in own project (as-a-service)
Search for visualisation & reports
Search for data to consume externally to AEGIS
View Results & Suggestions (high-level)
Rewiew Sub-Process
Review licenses, terms, prices
Review descriptions, info, previews, excepted output
Review assesment & curation metrics
Perform limited experimentation / Explore advanced preview
Negotiate terms/price
Select & Acquire/ Purchase
Consume
Throught access to visualisation & reports, through RESTful APIs...

Figure 4-11: AEGIS Services Workflow

5. CONCLUSION

The current deliverable reports on the work performed in Tasks 3.4 and 3.5 and builds on top of the work and outputs of other tasks and deliverables, in order to define and design the components of the holistic technical AEGIS approach, as well as the way those components interact, and their provided services are planned and coordinated towards enabling the workflows that lead to (big) data-driven innovation in the PSPS domains.

The conceptual architecture of AEGIS, initially presented in D3.1, has been updated and refined with more in-depth technical details. Towards this, the decision that determined the majority of the architectural choices, was the adoption of Hops platform as the AEGIS big data processing cluster, which will serve as the implementation backbone.

Following this decision, a big data solution was designed, composed of multiple high-level components. Residing at the location of the data, an anonymisation tool ensures that sensitive information is not published. The AEGIS Harvester constitutes the entrance point to the AEGIS web platform and is responsible also for transforming, harmonising and annotating data, providing reach metadata to the AEGIS platform for all imported data. The Data Store component, based on HopsFS, is then responsible for storing the collected and curated data. Data cleansing is envisioned both offline and online, in order to support the diverse cleansing needs of big-data enabled applications. Query Builder, Algorithm Execution Container and Visualiser deliver the functionalities related to the subsequent steps of the data value chain, enabling the intuitive querying, filtering, processing and visualising data and data analysis results. In the background, the policies defined by the AEGIS DPF are applied through the Brokerage Engine, which also records transactions over the different data artefacts, in a distributed ledger, supported by a blockchain implementation. All provided services leverage the Hopsworks Integrated Services and will be brought together under a common user interface and a holistic security approach. Finally, the AEGIS orchestrator ensures that all processes are smoothly executed across components.

Although most of the components were introduced in D3.1, the current deliverable goes beyond refining their functionality and adds information regarding the way they will be implemented, i.e. the underlying tools and technologies used. It should be noted that the provision of more specific information regarding the functionalities of certain components was made possible through the work reported in D2.1 and D2.2. More specifically, the identified semantic representations and vocabularies to be used, as well as the algorithms and other business intelligence services to be supported helped define in greater detail the way the core components of the data value chain bus should function. Furthermore, the communication mechanisms among components are provided here, so that it is clear both which components should interact and through which interfaces.

Complementary to the technical details, the work performed aimed to ensure that the designed solution will efficiently deliver all required workflows, as these have been reported in the high-level scenarios, the concrete user stories and the initially identified demonstrator needs. Therefore, BPMN diagrams were provided to show the desired user perspective when performing the core functionalities that AEGIS offers. Both simple actions, e.g. user signup, and complex actions, e.g. big data analysis and result visualisation, have been included, ensuring that all complex activities can be delivered by appropriately combining the provided diagrams as building blocks.

The AEGIS platform and tools, as a whole and per component, will be further refined based on collected feedback during the subsequent phases of the project and updates will be reported in D3.3.

APPENDIX A: TECHNICAL REQUIREMENTS AND COMPONENTS MAPPING

ID	Need	Priority	Component
TR1	Process measurements from “sensor nodes”	High	Data Harvester, Data Annotator, Algorithm Execution Container, AEGIS Data Store
TR2	Process measurements from social media sources	High	Data Harvester, Data Annotator, Algorithm Execution Container, AEGIS Data Store
TR3	Process data from mobile devices	High	Data Harvester, Data Annotator, Algorithm Execution Container, AEGIS Data Store
TR4	Process data from other open sources	High	Data Harvester, Data Annotator, Algorithm Execution Container, AEGIS Data Store
TR5	Process data from relational databases	High	Data Harvester, Data Annotator, Algorithm Execution Container, AEGIS Data Store
TR6	Process data from NoSQL databases	High	Data Harvester, Data Annotator, Algorithm Execution Container, AEGIS Data Store
TR7	Process data from structured sources	High	Data Harvester, Data Annotator, Algorithm Execution Container, AEGIS Data Store
TR8	Process data from semi-structured sources	High	Data Harvester, Data Annotator, Algorithm Execution Container, AEGIS Data Store
TR9	Connect to data sources through well-defined APIs.	High	Data Harvester, Data Annotator, AEGIS Data Store
TR10	Configure data sampling / update period	High	Data Harvester
TR11	Store big data	High	Data Harvester, Data Annotator, AEGIS Data Store
TR12	Produce RDF Triples	High	Data Harvester, Data Annotator
TR13	Store RDF Triples	High	AEGIS Data Store
TR14	Handle big data scalability	High	AEGIS Data Store, Query Builder, Algorithm Execution Container, Big Data Processing Cluster
TR15	Ensure data availability	High	AEGIS Data Store
TR16	Ensure data persistence	High	AEGIS Data Store

TR17	Ensure data consistency	High	AEGIS Data Store, Data Harvester, Data Annotator
TR18	Offer distributed storage	High	AEGIS Data Store
TR19	Offer public storage	High	AEGIS Data Store
TR20	Offer private storage	High	AEGIS Data Store
TR21	Provide ability to users to securely upload datasets	High	AEGIS Data Store, Data Harvester, Data Annotator
TR22	Provide ability to declare IPR on uploaded datasets and set access restrictions	High	Brokerage Engine
TR23	Access data	High	AEGIS Data Store, Query Builder
TR24	Select data	High	AEGIS Data Store
TR25	Implement a data management policy	High	AEGIS Data Store, Brokerage Engine
TR26	Support authentication mechanisms for access to data	High	AEGIS Data Store
TR27	Secure datasets	High	AEGIS Data Store
TR28	Clean datasets	Medium	Data Harvester, Data Annotator
TR29	Anonymise datasets	Medium	Anonymisation Tool
TR30	Support interlinking of AEGIS triplestore with SLOD	Medium	AEGIS Data Store
TR31	Provide SPARQL Endpoints	High	AEGIS Data Store
TR32	Support conversion of multiple source formats to RDF	High	Data Harvester, Data Annotator
TR33	Support conversion of multiple source formats to multiple formats	Medium	Data Harvester, Data Annotator
TR34	Support interlinking of datasets from different sources.	High	Data Harvester, Data Annotator
TR35	Support the reuse of available canonical standard vocabularies and ontologies.	High	Data Harvester, Data Annotator
TR36	Deal with corrupted, inconsistent data or misconfiguration of mapping rules	Medium	Data Harvester, Data Annotator

TR37	Support management of data sources	High	Data Harvester, Data Annotator, AEGIS Data Store
TR38	Add metadata, enriching the existing data sources	Medium	Data Harvester, Data Annotator
TR39	Select / Search for / Define vocabularies	Medium	Data Annotator
TR40	Inspect data sources	Medium	Data Harvester, Data Annotator
TR41	Ability to handle queries spanning multiple datasets	High	Algorithm Execution Container, Query Builder
TR42	Support format conversion	High	Data Harvester
TR43	Manually balance transformation automation and semantic soundness	Low	Data Harvester, Data Annotator
TR44	Support data analytics on distributed big data	High	Algorithm Execution Container, Query Builder, Big Data Processing Cluster
TR45	Support data analytics on distributed big data	High	Algorithm Execution Container, Big Data Processing Cluster
TR46	Support initialisation of algorithms	High	Algorithm Execution Container
TR47	Support the processing of data in multiple formats.	High	Algorithm Execution Container
TR48	Support a set of different types of output formats.	High	Algorithm Execution Container
TR49	Customise the analytics process	High	Algorithm Execution Container
TR50	Support real-time analysis		Algorithm Execution Container
TR51	Manage the results of the big data enabled analytics	High	Algorithm Execution Container, Visualiser
TR52	Support overview of resources	High	Data Management, Visualiser
TR53	Preview a small selection of the results of the generated query	Medium	Visualiser

TR54	Support a set of different types of visualisations	High	Visualiser
TR55	Create advanced graphs based upon queries spanning multiple datasets	High	Visualiser
TR56	Support secure communication among the AEGIS components	High	Data Harvester, Data Annotator, Data Indexing Engine, Algorithm Execution Container, Query Builder, Visualiser, Big Data Processing Cluster, Brokerage Engine, AEGIS Data Store, AEGIS Orchestrator
TR57	Save analytics results	High	Visualiser, AEGIS Data Store
TR58	Support authorised sharing of saved results	Medium	Visualiser, Brokerage Engine
TR59	Support local deployment of the anonymisation tool	Low	Anonymisation Tool
TR60	Support local deployment of the cleansing tool	Low	Cleansing Tool
TR61	Support local deployment of the harvesting tool	Low	Data Harvester
TR62	Support local deployment of the transformation tool	Low	Data Harvester
TR63	Provide push notifications	High	Data Harvester
TR64	Provide secure user-friendly interface and flexible navigation.	High	Visualiser, Query Builder, Data Harvester, Data Annotator
TR65	Provide system management dashboard	High	AEGIS Front-End
TR66	Support geospatial data analysis	High	Algorithm Execution Container
TR67	Provide a solid deployment workflow of the system	Medium	
TR68	Support uploading user’s custom algorithm implementation	Low	Algorithm Execution Container
TR70	Support a set of functionalities for the processing of the output result	High	Front-End

	such as save, export, share etc.		
TR71	Provide means for processing the selected datasets and include constraints towards the preparation of the desired output.	High	Front-End
TR72	Allow even a non-expert user to experience the AEGIS services.	Medium	Front-End
TR73	Provide cross-browser and cross-platform compatibility (accessibility), ensure performance (render time).	Medium	Front-End

Table 4: Mapping Technical Requirements to Components