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

D3.3 – Architecture and Revised Components, Microservices and APIs Designs v2.00

Version 1.1

Due date: 30.03.2018 **Delivery Date**: 19.07.2018 (updated)

Author(s): Dimitrios Miltiadou, Konstantinos Perakis, Stamatis Pitsios (UBITECH), Elisa Rossi, Alessandro Calcagno (GFT), Mahmoud Ismail, Alexandru A. Ormenisan (KTH), Yury Glikman, Fabian Kirstein, Fritz Meiners (Fraunhofer), Giannis Tsapelas, Panagiotis Kokkinakos, Spiros Mouzakitis, Christos Botsikas (NTUA), Sotiris Koussouris, Marios Phinikettos (SUITE5)

Editor: Dimitrios Miltiadou (UBITECH)

Lead Beneficiary of Deliverable: UBITECH

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

Internal Reviewers: Maurizio Megliola (GFT), Evmorfia Biliri (NTUA)

Remark: The versioning is only for the word documents in the formation phase and should be kept internally. Please delete the versioning before creating the final pdf that goes to the commission. It can be provided to the commission on request. Please document only major versions and such versions that indicate through the versioning, who (person and which partner) has contributed/was responsible for the different chapter, if this is feasible.

EXPLANATIONS FOR FRONTPAGE

Author(s): Name(s) of the person(s) having generated the Foreground respectively having written the content of the report/document. In case the report is a summary of Foreground generated by other individuals, the latter have to be indicated by name and partner whose employees he/she is. List them alphabetically.

Editor: Only one. As formal editorial name only one main author as responsible quality manager in case of written reports: Name the person and the name of the partner whose employee the Editor is. For the avoidance of doubt, editing only does not qualify for generating Foreground; however, an individual may be an Author – if he has generated the Foreground **-** as well as an Editor – if he also edits the report on its own Foreground.

Lead Beneficiary of Deliverable: Only one. Identifies name of the partner that is responsible for the Deliverable according to the AEGIS DOW. The lead beneficiary partner should be listed on the frontpage as Authors and Partner. If not, that would require an explanation.

Internal Reviewers: These should be a minimum of two persons. They should not belong to the authors. They should be any employees of the remaining partners of the consortium, not directly involved in that deliverable, but should be competent in reviewing the content of the deliverable. Typically this review includes: Identifying typos, Identifying syntax & other grammatical errors, Altering content, Adding or deleting content.

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 organizations 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 scope of D3.3 is to document the efforts undertaken within the context of the tasks 3.1, 3.2, 3.3, 3.4 and 3.5 of WP3. Towards this end, the scope of this deliverable is to build on top of the outcomes and knowledge extracted by D3.2 as well as the evaluation and feedback received for the first low fidelity, functional mock-up version of the AEGIS platform in order to define the necessary modifications and refinements on the components of the platform and the platform's workflows. The current deliverable provides a complementary documentation supplementing the information documented in deliverable D3.2 focusing on the updates introduced.

The document at hand is the revised version (v1.1) of the deliverable D3.3. The deliverable has been revised with the aim of documenting the updated information on the latest architectural decisions, providing also the technical architecture of the platform and highlighting the list of microservices included within the design of each component of the architecture.

More specifically, the objectives of the deliverable D3.3 are as follows:

- Provide a comprehensive description of the updated high-level architecture of the AEGIS platform as presented in deliverable D3.2, focusing on the architectural decisions, the updated components and their positioning within the architecture and the platform functionalities undertaken by each component. Additionally, the technical architecture of the AEGIS platform is presented, illustrating the functional decomposition of the components, the relationship between them and the corresponding data flow.
- Provide the updates and definitions with regards to the design and the specifications for each component of the platform.
- Outline the updated functionalities of each component of the platform along with the technologies and tools utilised in order to implement the aforementioned functionalities.
- Document the list of microservices and their corresponding functionalities for each component of the platform.
- Document the interfaces and exposed outcomes offered by each component to facilitate the interactions of the components as well as the execution of the workflows of the platform.
- Present the updated BPMN diagrams that are illustrating the AEGIS platform's workflow, focusing on the user perspective and on the interactions of the components on a high-level.

In the next steps the outcomes of this deliverable will drive the implementation activities of the project. As the project evolves, additional requirements will be received that will introduce new functionalities on the platform. Furthermore, once the upcoming version of the platform will be released, additional feedback will be received and evaluated. This will result in further updates and refinements on the platform and the platform's components that will be documented in D3.4 entitled "Architecture and Revised Components, Microservices and APIs Designs – v3.00".

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)
CSS	Cascading Style Sheets
CSV	Comma Separated Value files
D	Deliverable
DLT	Distributed ledger technology
DoW	Description of Work
DPF	Data Policy Framework
FLOSS	Free/Libre Open Source Software
HTML	Hypertext Markup Language
H2020	Horizon 2020 Programme
JSON	JavaScript Object Notation
JWT	JSON Web Token
NLP	Natural language processing
OSS	Open Source Software
PSPS	Public Safety and Personal Security
PU	Public
PM	Person Month
R	Report
RDF	Resource Description Framework
REST	Representational State Transfer
RTD	Research and Development
SQL	Structured Query Language
SSL	Secure Sockets Layer

T	Task
TLS	Transport Layer Security
UI	User Interface
URL	Uniform Resource Locator
WP	Work Package
XML	Extensible Markup Language

1. INTRODUCTION

1.1. Objective of the deliverable

The scope of D3.3 is to document the efforts within the context of the tasks 3.1, 3.2, 3.3, 3.4 and 3.5 of WP3. Towards this end, the scope of the current deliverable is to document the updates and refinements introduced on the AEGIS platform and platform's components, as well as the updates on the APIs and exposed outcomes of the components, based on the evaluation and feedback received for the first low fidelity, functional mock-up version of the AEGIS platform that was delivered in M14, as documented with deliverable D4.1.

The main objective of the current deliverable is to provide a complementary documentation supplementing the information documented in deliverable D3.2. More specifically, the current document is providing all the necessary updated descriptions on the design and specifications of the AEGIS components, the AEGIS platform workflows, as well as the updated high-level architecture. For reasons of coherency, the current document contains information included in deliverable D3.2 with the additions and modifications pointed out at the end of each section.

Following the approach used in the previous version, the updated high-level architecture is presented, describing how each component corresponds to a specific functionality of the platform with use of the specific technologies and tools. It should be noted at this point that in the current document there was no differentiation on the architecture as documented in D3.2, however the provided documentation is focusing on the positioning of the updated components within the architecture. In addition to the high-level architecture, the AEGIS platform's technical architecture is presented with the major focus being on the functional decomposition of the components, the relationship among them and the data flow.

The current document provides the updated detailed descriptions of the components of the AEGIS platform, outlining the functionalities and the interactions between them, the list of microservices designed within the context of each component, the technologies used, the technical interfaces and exposed outcomes offered. In addition to the aforementioned, the current document is presenting the BPMN diagrams illustrating the workflows of the AEGIS platform as documented in D3.2 with the necessary adaptations based on the updates of the components.

It should be noted that the document at hand is the revised version (v1.1) of the deliverable D3.3 that includes the latest information in terms of architectural decisions along with the technical architecture of the platform, as well as the list of microservices for each designed component of the platform.

1.2. Insights from other tasks and deliverables

The deliverable builds on top of the work reported in WP3 and WP4. In particular, the work performed in WP3, as reported in D3.1 and D3.2, provided valuable information concerning the functional and technical requirements collected, the initial design of the platform's components, the high-level architecture of the platform as well as the platform's workflows. Another useful insight is the work performed within the context of WP4, as reported in D4.1 where the first low fidelity, functional mock-up version of the platform was delivered. The first evaluation and feedback received from the project's demonstrators serves as the basis upon which the updates and refinements on the platform and the platform's components were built.

1.3. Structure

Deliverable D3.3 is organised in five main sections as indicated in the table of contents.

- The first section introduces the deliverable. It documents the objectives of the deliverable and the relation of the current deliverable with the other deliverables by describing how the outcomes of other deliverables and work-packages serves as input to the current deliverable. Finally, a brief description is provided on how the document is structured.
- The second section presents the high-level architecture of the AEGIS platform as documented in D3.2, focusing on the positioning of the updated components within the architecture. Moreover, the technical architecture of the AEGIS platform is presented, illustrating the functional decomposition of the components, their relationships and the respective data flow. Finally, in this section the decision to provide an integrated notebook containing three major components of the AEGIS platform is documented.
- The third section provides the updated documentation of the components of the AEGIS platform. In this section for each component an overview containing the component's updated functionalities along with the list of designed microservices is presented, as well as the technologies used for the implementation of the components and the APIs or the exposed outcomes of the components focusing on the updates after deliverable D3.2 (when available).
- The fourth section is presenting the BPMN diagrams that correspond to the provided functionalities of the AEGIS platform focusing on the user perspective and on summarising the component interactions in a high-level without the technical details.
- The fifth section concludes the deliverable. It outlines the main findings of the deliverable which will guide the future research and technological efforts of the consortium.

2. AEGIS ARCHITECTURE

2.1. High level architecture

In deliverable D3.2 the updated high-level architecture of the AEGIS platform was presented. This updated architecture included all the necessary refinements and adjustments towards the aim of enabling the designed workflows that will enable the data-driven innovation in the PSPS domains as envisioned by the consortium. The presented high-level architecture had driven the implementation and release of the first low fidelity, functional mock-up version of the AEGIS platform, which was presented in D4.1 in M14.

The AEGIS high-level architecture is a modular architecture composed of multiple key components, where each component was designed with a clear business context, scope and set of functionalities. Figure 2-1 illustrates the high-level architecture, which remained unaffected in terms of design, functionalities and interactions, as there were no additional requirements identified requiring the introduction of any additional updates or refinements.

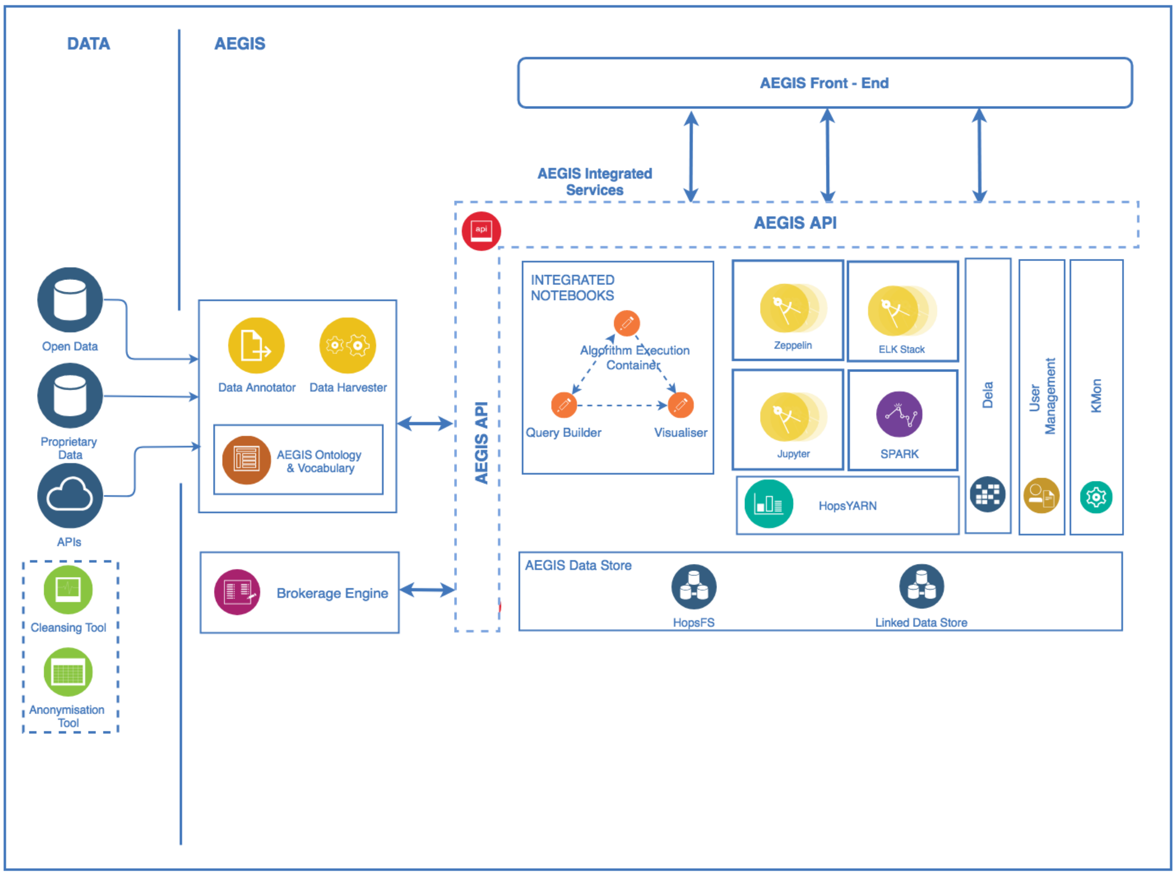


Figure 2-1: AEGIS high-level architecture

Residing at the location of the data, two optional components are offered by the AEGIS ecosystem, namely the Anonymisation tool and the Cleansing tool. The Anonymisation tool is an offline tool ensuring that sensitive or personal data are not uploaded in the platform and will address the privacy and anonymity requirements by applying a set of anonymisation techniques on the initial dataset. The Cleansing tool will provide the necessary cleansing processes with a variety of techniques that will be offered in both offline and online mode (through custom processes incorporated inside the integrated notebooks of the platform) depending on the context of the processes and required corrective actions.

The AEGIS Data Harvester and Annotator is providing the data entry point to the AEGIS platform offering the transformation, harmonisation and annotation functionalities required within the context of the platform as well as the rich metadata generation for the imported data. In the core of the AEGIS platform lays the AEGIS Data Store component, composed by the HopsFS and the Linked Data Store. HopsFS is a fast, reliable and scalable distributed file system that undertakes the responsibility for storing the imported datasets, while the Linked Data Store is responsible for storing the metadata generated using the AEGIS ontology and vocabulary for each dataset, as provided by the AEGIS Harvester and Annotator.

AEGIS Integrated Services consists of a list of services responsible for the data management and processing within the platform. In addition to the multi-tenant data management, data exploration, data parallel processing and resource management, these services implement as well the user management and service monitoring aspects of the AEGIS platform. The list of services in AEGIS Integrated Services includes the Apache Zeppelin and Jupyter services offering interactive notebooks, the ElasticSearch Logstash Kibana (ELK) stack, the Apache Spark and the HopsYARN, as well as the Dela, the User Management and the KMon services.

In addition to the AEGIS Integrated Services, the AEGIS platform incorporates three more components in the form of integrated notebooks using Apache Zeppelin and Jupyter, namely the Query Builder, the Algorithm Execution Container and the Visualiser that are supplementing the delivered functionalities of the AEGIS platform. More specifically, Query Builder is simplifying and empowering the querying capabilities of the platform by providing an intuitive graphical interface for powerful data pre-processing capabilities, data retrieval and view creation on the data in order to generate a new dataset or provide an input to Algorithm Execution Container and Visualiser. The Algorithm Execution Container is enabling the execution of the data analysis algorithms over multiple selected datasets in order to provide the data analysis results in the Visualiser. Visualiser is the component facilitating the visualisation functionalities of the platform for either the querying and filtering results as produced by the Query Builder or the analysis results as produced by the Algorithm Execution Container.

The Brokerage Engine is responsible for access control and recording of actions performed over the artefacts of the platform such as datasets, services and algorithms. More specifically, the Brokerage Engine is ensuring conformance with the Data Policy Framework of AEGIS while also utilising a distributed ledger supported by a blockchain implementation in order to record all transactions over these artefacts. Finally, the AEGIS Front-End is the component implementing the presentation layer of the platform using an innovative user-friendly interface to enable the easy navigation and exploitation of the platform services to the AEGIS stakeholders.

For each component, a detailed description documenting the functionalities and the technical details is elaborated in Section 3 of the current deliverable.

It should be noted that the description of the high-level architecture, as presented in the current revised version of the deliverable D3.3, includes the decision to remove the AEGIS Orchestrator component from the AEGIS platform architecture. The consortium came to the conclusion that based on the current design of the architecture of the platform an orchestration *engine is not required. According to the current design and specifications of the components of the platform, all the designed microservices of each component are incorporated and independently deployed within the context of each component and not exposed to the user or the rest of the components or services. For example, the HarvesterImporterService and the HarvesterTransformerService are registered microservices integrated within the Data Harvester component for the execution of the harvesting process and the Participant Registrant and Asset Registrant microservices are integrated within the Brokerage Engine for the data brokerage process, and even though a user could arbitrarily connect them this would not generate any added value service. Additionally, the various integrated services such as Apache Zeppelin and Jupyter are standalone services, that are exposed through their corresponding interfaces, and which could break down without ceasing the AEGIS platform operation. Since they are exposed and can be triggered at any time by the user, and given that the Query Builder, the Visualiser and the Algorithm Execution Container will be integrated in one notebook, there is no need for an orchestration engine that will undertake their sequential execution.*

2.2. Technical Architecture

In addition to high-level architecture presented in section 2.1, Figure 2-2 illustrates the functional decomposition of the components of the AEGIS platform, as well as the relationship of the components and the corresponding data flow during run-time. The details for the design and specification of each component are described in Section 3.

2.3. AEGIS Integrated Notebooks

In the course of the development of the Query Builder, the Algorithm Execution Container and the Visualiser the technical partners decided to leverage the capabilities and features provided by the Apache Zeppelin and Jupyter services that are already integrated within the AEGIS platform. Both services are providing functionalities for data ingestion, data discovery, data analytics and data visualisation to the data scientists, support for various languages such as Python and Scala, integration with data processing frameworks like Spark and support for user interface implementation in JavaScript.

Although the aforementioned components were developed as separate predefined notebooks containing several paragraphs, the integration of them into one complete notebook is foreseen towards the aim of offering a holistic toolset for data query and retrieval, data pre-processing, data analysis execution and advanced visualisations. Within this holistic notebook, all the described functionalities and features of the aforementioned three components will be integrated to enable the end users of the platform to perform all the desired tasks from this complete notebook, providing intuitive and advanced user experience. The integration of the aforementioned is an ongoing activity that will last until M24 when the AEGIS platform V3.00 will be released.

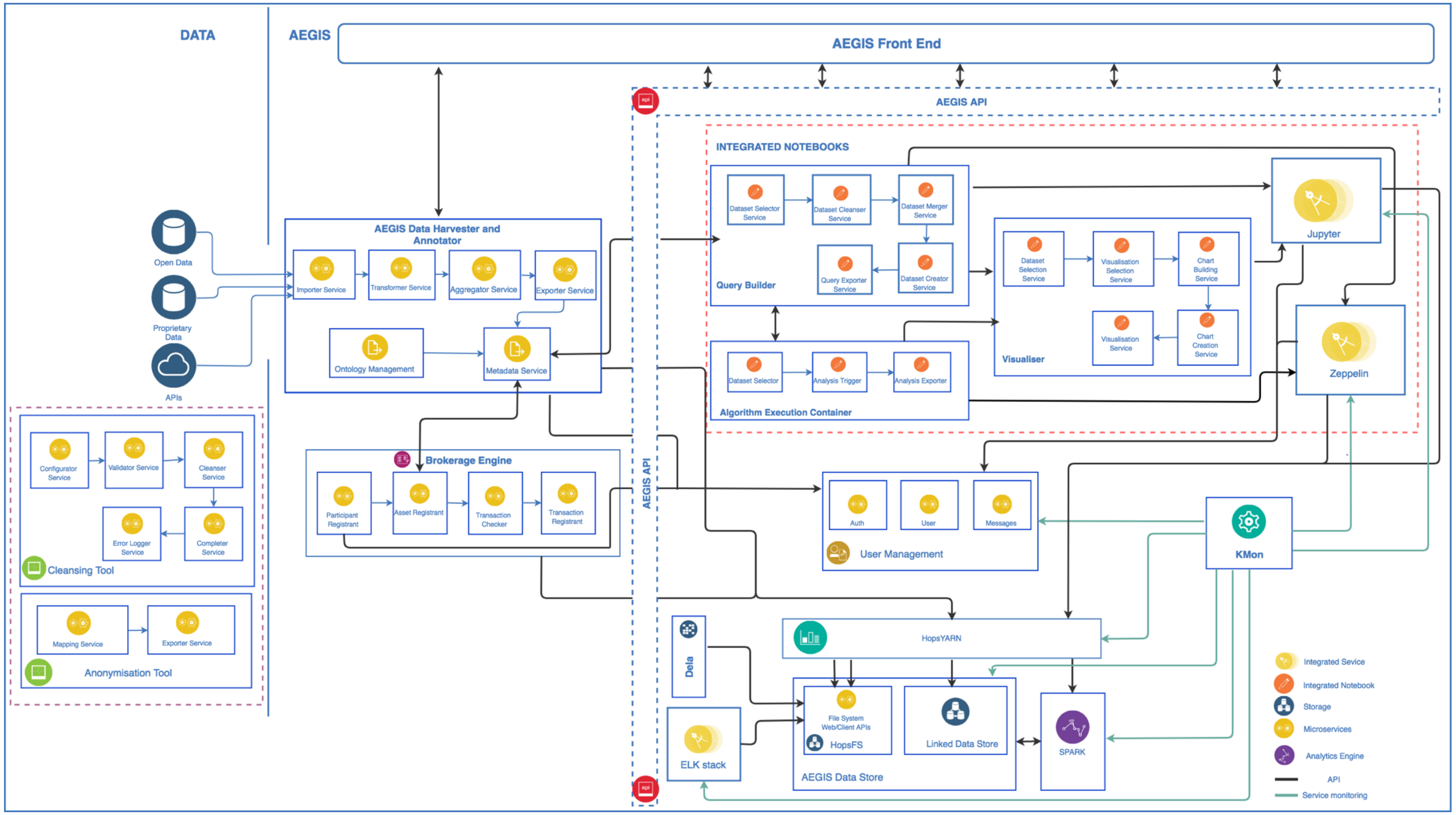


Figure 2-2: AEGIS Technical Architecture

3. AEGIS COMPONENTS AND APIS SPECIFICATIONS

3.1. Data Harvester and Annotator

3.1.1. Overview

The Data Harvester and Annotator is orchestrated out of several sub-components, including microservices and front-end modules. 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*. The Harvester interacts tightly with the Metadata Service of the AEGIS Linked Data Store and the AEGIS Data Store and is based on several basic concepts. In the following paragraphs these concepts are described in detail:

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

Annotation
An annotation constitutes metadata of a project, dataset or file within the AEGIS platform. Hence it uses the AEGIS vocabularies and ontologies (see deliverable D2.1 Ch.3).

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 includes detailed logging information.

These concepts are represented in the four microservices and the front-end of the Harvester. It is important to notice, that each microservice may have multiple instances or rather specialised implementations. E.g. they may be one importer for CSV data and one importer for JSON data.

Importer
An importer implements all functionality for retrieving data from a specific data source. It needs to specifically support the characteristic of that data source, including protocol, serialisation format, security etc. It has to export the harvested data as JSON to the next stage.

Transformer
A transformer converts the retrieved data from an importer into the target format of the AEGIS platform. Hence, a tabular format is specified for each data source.

Aggregator
An aggregator collects converted data from a transformer over a configurable time interval. It allows to adjust the granularity of the available data in one file within the AEGIS platform.

Exporter
The exporter uploads transformed and/or aggregated data to the AEGIS platform. In addition, it creates the corresponding metadata in the AEGIS Metadata Store. There will be only one implementation for the exporter.

Front-end
The front-end orchestrates the microservices and offers the visual interface for creating, editing and existing specific harvesting processes.

Figure 3-1 shows the process of harvesting data from a data source to the AEGIS platform.

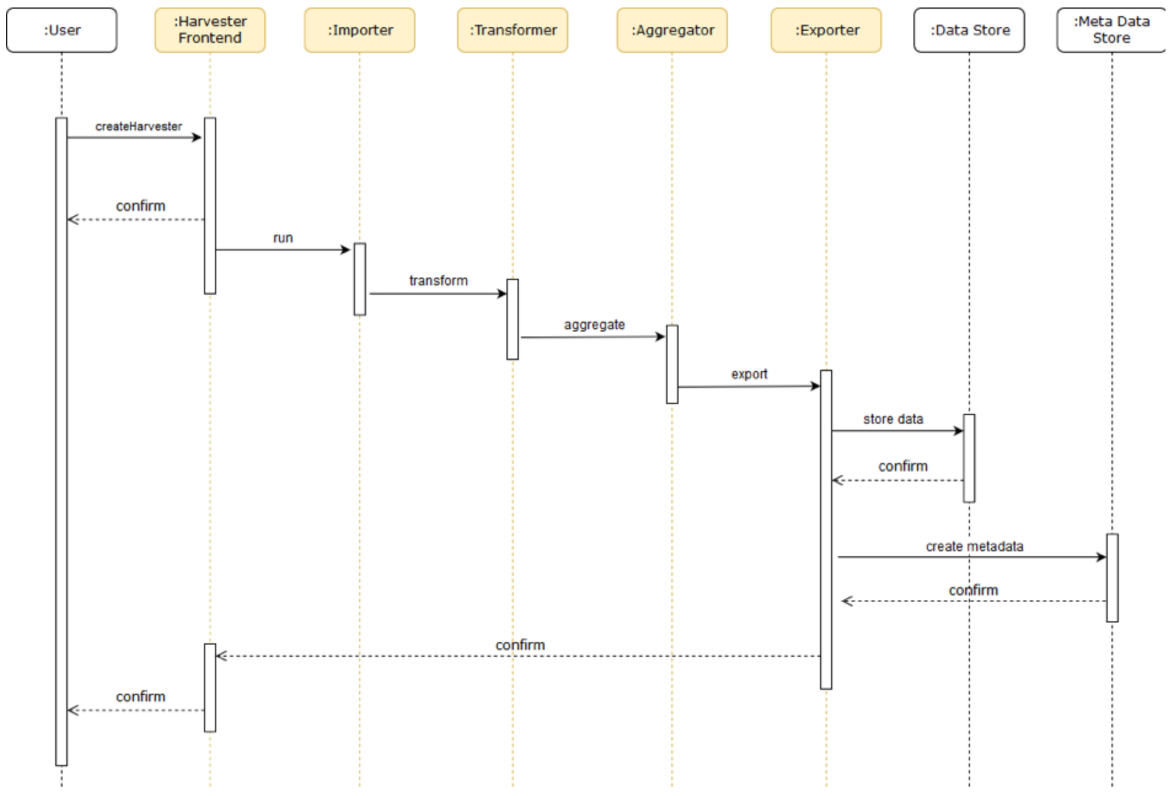


Figure 3-1: Sequence diagram of the Harvester component

Updates from V1.0:

The overall methodology did not change, but the specific application and implementation. The entire architecture of the Harvester was shifted towards a much more agile microservice approach.

3.1.2. List of microservices

For the Harvester component four microservices are developed. Each service depicts one distinct task within the harvesting process. The orchestration of the services is done via a single-page-application front-end, which will be tightly integrated into the AEGIS platform. All services expose their functionality via a RESTful-API.

Component Name	**Microservice Name**	**Functionalities**
AEGIS Data Harvester & Annotator	HarvesterImportService	• Handling of repositories, e.g. creation and modification • Management of specific repository connectors • Execution of the importing process • Logging of importing process • Transfer of the imported data to the transformer service
	HarvesterTransfomerService	• Management of transformations rules and scripts • Execution of the transformation from source data to the AEGIS data format • Logging of transformation process • Transfer of the transformed data to the aggregator or exporter service
	HarvesterAggregatorService	• Optional service for aggregating imported data for a specified time interval before exporting it to the AEGIS platform • Transfer of aggregated data to the exporter service
	HarvesterExporterService	• Handling of the export of the data to the AEGIS platform • Direct communication with the RESTful-API of AEGIS • Creation of the metadata in the Metadata-Service based on the given annotations

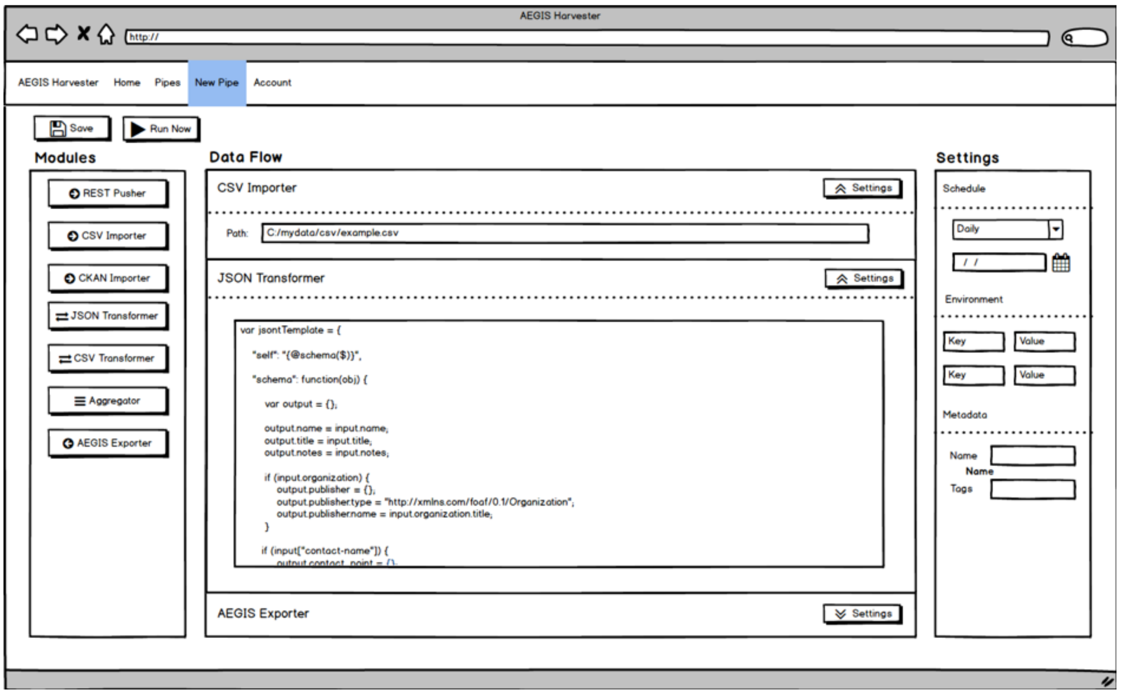
Table 3-1: Harvester list of microservices

3.1.3. Technologies to be used

The foundation of the AEGIS Harvester is the EDP Metadata Transformer Service (EMTS)[1], an open source solution for harvesting metadata from diverse Open Data sources.

For the purpose of the AEGIS platform, the EMTS is refined and updated to fit the needs of the project. This includes restructuring the application into small and scalable microservices. This will be done by extracting the respective functionalities into new standalone services. These services will be developed with the event-driven Java-framework Eclipse Vert.x[2]. This will allow a much tighter integration into the AEGIS platform and the straightforward extension with additional functionalities. Correspondently, the web front-end will be modified to single-page-application in order to act as an orchestrator of the various microservices. It will be implemented based on the JavaScript Vue.js[3] framework. Additionally, this will allow a better integration into the existing front-end of AEGIS platform.

Figure 3-2 shows a mock-up of the future front-end of the Harvester.



Path: C:/mydata/csv/example.csv

Daily

/ /

```
var jsontTemplate = {

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

  "schema": function(obj) {

    var output = {};

    output.name = input.name;
    output.title = input.title;
    output.notes = input.notes;

    if (input.organization) {
      output.publisher = {};
      output.publisher.type = "http://xmlns.com/foaf/0.1/Organization";
      output.publishername = input.organization.title;
    }

   if (input["contact-name"]) {
      output.contact_point = {};
```

Figure 3-2: The harvester interface

The orchestration architecture used for the AEGIS Harvester follows a “pipeline“ pattern, in which data is passed through several services, with each service manipulating the data in some sort. Each service is responsible for exactly one task. This permits a rather generic

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

[2] https://vertx.io/

[3] https://vuejs.org/

implementation of each service. The aim is to encourage a separation of concerns in order to enhance reusability, as well as allowing the dynamic scaling in times of high load. The latter is achieved by deploying additional instances of the services demanded most. Once new instances are spawned, request may dynamically be routed to the instance of a service with the least load. This is shown in Figure 3-3.

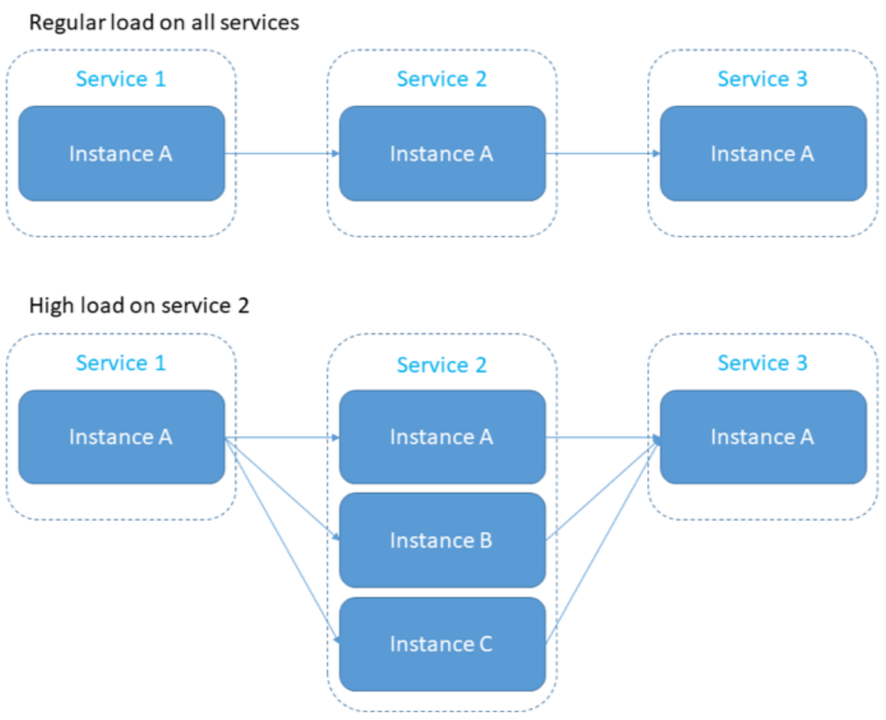


Figure 3-3: Harvester Orchestration Concept

The order and type of services participating in handling a certain use case is initially defined for later utilisation by the pipe implementation. The framework then builds the suitable requests (as well as the handling the concrete routing between instances) and provides the applicable configurations. This makes each service agnostic of its surroundings, aiding in the generic design mentioned earlier.

Updates from V1.0:

After a detailed evaluation, the application of the StreamSets Data Collector will be omitted. Practical tests have shown that the tool does not fit the needs of the AEGIS platform and required changes will be too extensive. Instead, the EMTS will be enhanced and extended to be suitable tool for harvesting and annotating data in the AEGIS project.

3.1.4. APIs and exposed outcomes

The following tables document the API of the Data Harvester and Annotator component.

Technical Interface	
Reference Code	AH#01
Function	Orchestrate, schedule and manage harvesting processed
Subsystems	EDP Metadata Transformer Service
Type, State	
RESTful API, Web Front-end	
Endpoint URI	
http://aegis-harvester.fokus.fraunhofer.de (Testing system)	
Input Data	
Harvester endpoints	
Output Data	
Status logs	

Table 3-2: Data Harvester and Annotator technical interface 2

3.2. Cleansing Tool

3.2.1. Overview

Data cleansing is an umbrella term for tasks that span from simple data pre-processing, 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.

The initial AEGIS decision on data cleansing was to not develop data cleansing tools from scratch and instead support the following two-fold approach: (a) simple data cleansing transformations will be applied through existing mature tools offline and (b) more complex data cleansing (e.g. outlier detection and removal) will be offered through custom cleansing processes developed within the available AEGIS tools for big data processing and algorithm execution. Based on the technical advancements of the project, the second option is now offered with the help of the Notebooks and Notebook-based components, which leverage the AEGIS processing and analytical capabilities.

However, the consortium has decided to extend the initial approach in the following two ways:

It has become obvious that cleansing tasks that are both easy/straightforward and at the same time computationally intense, may emerge as steps of the analysis to be performed, i.e. they may be dependent on the specific application and not on intrinsic characteristics of the original dataset. These tasks will need to be performed as part of the online data manipulation. Hence, in order to provide a more intuitive user experience and also leverage the computational power of the AEGIS system, it was decided to make certain simple data cleansing functionalities available to the user during the data query creation process, i.e. when he/she should be more confident about the desired data manipulation needed to perform in order to use the data for further analysis. Hence, some simple custom cleansing processes are incorporated inside Query Builder as part of the data selection process and will be described in the corresponding section.

Additionally, the consortium identified the added value of providing an offline cleansing tool that will offer a level of customisation to the users and will be easily adaptable to the user’s needs depending on the nature of the data source. Thus, it was decided to implement the offline cleansing tool that will enable data validation, data cleansing and data completion processes towards the aim of increasing the reliability, accuracy and completeness of the data that will be imported in the AEGIS platform. The tool will be customisable, in terms of rules definitions for validation, cleansing and missing data handling, by the user and will provide web-based user interface to display the cleansing process results.

The main functionalities of the offline cleansing tool are as follows:

- Definition of the rules for cleansing process (data validation, data cleansing, data completion).
- Provide a RESTful interface to facilitate the uploading of the dataset that will be used in the cleansing process and provide the cleaned data.
- Report the cleansing process results through a user interface.

The following figure shows the sequence diagram for the offline data cleansing. The sequence diagrams for data cleansing performed through other tools will be provided in the corresponding sections.

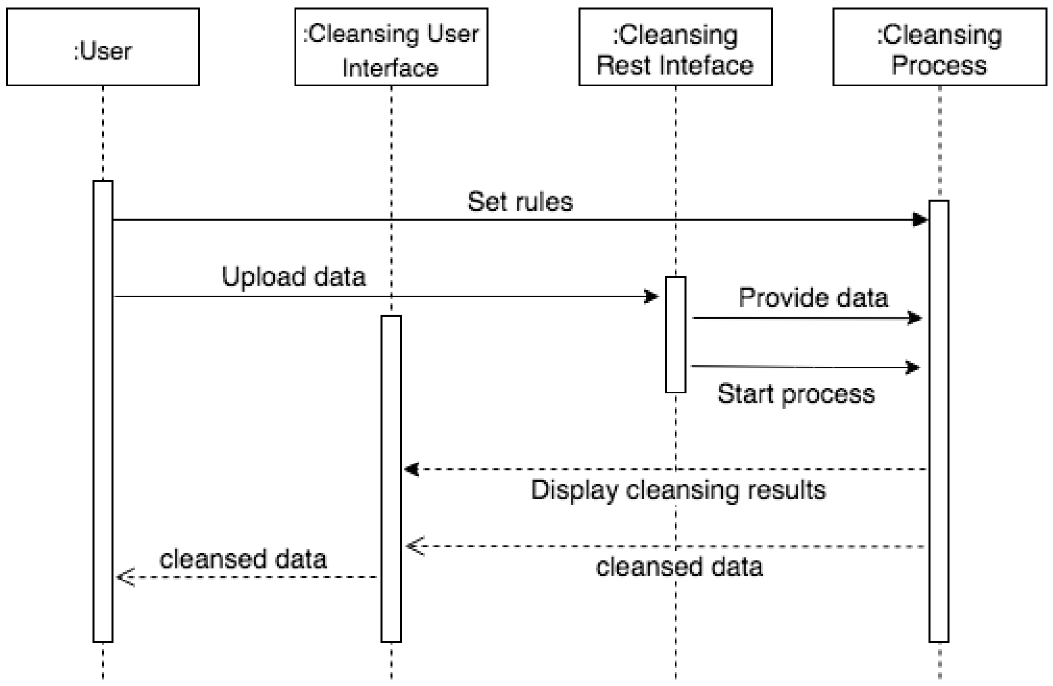


Figure 3-4: Offline data cleansing sequence diagram

Updates from V1.0:

A new approach to data cleansing is provided.

Specifically, an offline cleansing tool will be developed that will be customisable according to the user's needs for data validation, data cleansing and data completion.

Moreover, simple data filtering and cleansing functionalities will be provided through the Query Builder. In this way the user is offered a more intuitive workflow since data cleansing requirements may be not known prior to and independently of the query creation process. Furthermore, the computational power of the AEGIS system is fully leveraged, ultimately allowing the user to perform more quickly the necessary data cleansing tasks which may entail heavy processes when dealing with big data.

3.2.2. List of microservices

For the offline data cleansing a list of microservices will be developed and will be orchestrated towards the execution of the cleansing tasks and the successful handling of the incoming requests for data cleaning transformations and corrective actions. In particular, the Cleansing Process, as shown in Figure 3-4, is composed by four microservices. The first microservice, the ConfiguratorService is undertaking the management of the constraints/rules for validation and data completion, as well as the corrective actions/rules. Additionally, three microservices, the ValidatorService, the CleanserService and the CompleteteService, are responsible for the data validation, the data cleansing and the data completion respectively. The ErrorLoggerService is the microservice responsible for the collection and management of the log records that contain the identified errors and the corrective actions from the execution of the microservices of the Cleansing Process. Moreover, the ErrorLoggerService is providing the input for the Cleansing User Interface that reports the execution results to the user.

In total five microservices will be developed and are described in the following table:

Component Name	Microservice Name	Functionalities
Cleansing Tool (Offline)	ConfiguratorService	• Maintain and manage the constraints/rules for validation (e.g. specific data types, value representation, uniformity, range, regular expression patterns, cross-field validity) • Maintain and manage the corrective actions/rules (e.g. rejection of values, logical error identification) • Maintain and manage the data completion rules
	ValidatorService	• Perform data validation in accordance to the constrains/rules • Compile the list of identified errors identified in the validation • Log the errors in the appropriate log file • Provide interface for remote execution
	CleanserService	• Perform data cleaning based on the defined rules • Log the corrective actions in the appropriate log file • Provide interface for remote execution
	CompleterService	• Implement a list of methods / algorithms for data completion (e.x. Last Observation Carried Forward, Last Non-Zero, moving average, Linear regression, mean, median, k-NN). • Perform data completion based on the defined rules, algorithms and methods • Log the corrective actions in the appropriate log • Provide interface for remote execution

	ErrorLoggerService	• Create and display log records containing the errors and corrective actions • Manage log files generated by the rest of the microservices • Provide an interface for the rest of the microservices for log record creation

Table 3-3: Cleansing Tool list of microservices

For the cleansing functionalities that are offered through the Notebooks and Notebook-based components, the relevant microservices are described in the corresponding sections.

3.2.3. Technologies to be used

For the online data cleansing, either through the dedicated data cleansing UI in the Query Builder or through custom processes -implemented with the help of the Apache Zeppelin and Jupyter Notebooks (which are part of the AEGIS Integrated Services) will be used. More details are provided in the corresponding tools' sections.

For the offline cleansing processes, which will be applied before importing data in AEGIS with the aim of making the data more easily processable by subsequent components in the data flow, the microservices architecture is followed and the corresponding microservices, as described in section 0, are written in Python, using Flask microframework[4] and a set of libraries such as Pandas[5] and NumPy[6].

Updates from V1.0:

Query Builder is extended to support data cleansing functionalities, so technologies used by that tool are also relevant here. The technologies used to create the Query Builder user interface for data manipulation inside the two Notebooks (Apache Zeppelin and Jupyter) include JavaScript, HTML, PySpark, Python and AngularJS (only in the Zeppelin version).
Additionally, the offline cleansing tool is implemented utilising the Python microframework Flask, supported by a set of libraries such as Pandas and NumPy.

[4] http://flask.pocoo.org/

[5] https://pandas.pydata.org/

[6] http://www.numpy.org/

3.2.4. APIs and exposed outcomes

For the offline data cleansing a REST API interface is provided in order to enable the uploading of the dataset that will be cleansed and provide the cleaned dataset once the cleansing process is completed. The details of this interface are documented in Table 3-4.

Since the online data cleansing is performed with the help of other components, more information is available in the corresponding sections.

Technical Interface	
Reference Code	CT#01
Function	Upload the dataset that will be used in the cleansing process
Subsystems	Offline Cleansing Tool
Type, State	
RESTful-API	
Endpoint URI	
<server url:5000>/cleaner/api/clean	
Input Data	
The data that will be used in the cleansing process in JSON format	
Output Data	
The cleansed data in JSON format	

Table 3-4: Offline Cleansing tool technical interface

3.3. Anonymisation Tool

3.3.1. Overview

The anonymisation tool is an extensible, schema-agnostic plugin that allows real-time efficient data anonymisation. The anonymisation tool has been utilised for offline, private usage but offers the ability to output the anonymised data through a secured, web API. With emphasis on performance, the anonymisation syncs with private database servers and executes anonymisation functions on datasets of various sizes with little or no overhead. The purpose of the anonymisation is to enable the potential value of raw data in the system by accounting for privacy concerns and legal limitations.

The anonymisation process is optional to the AEGIS data flows and 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.

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 the individual sensitive records or subjects of the data cannot be re-identified.

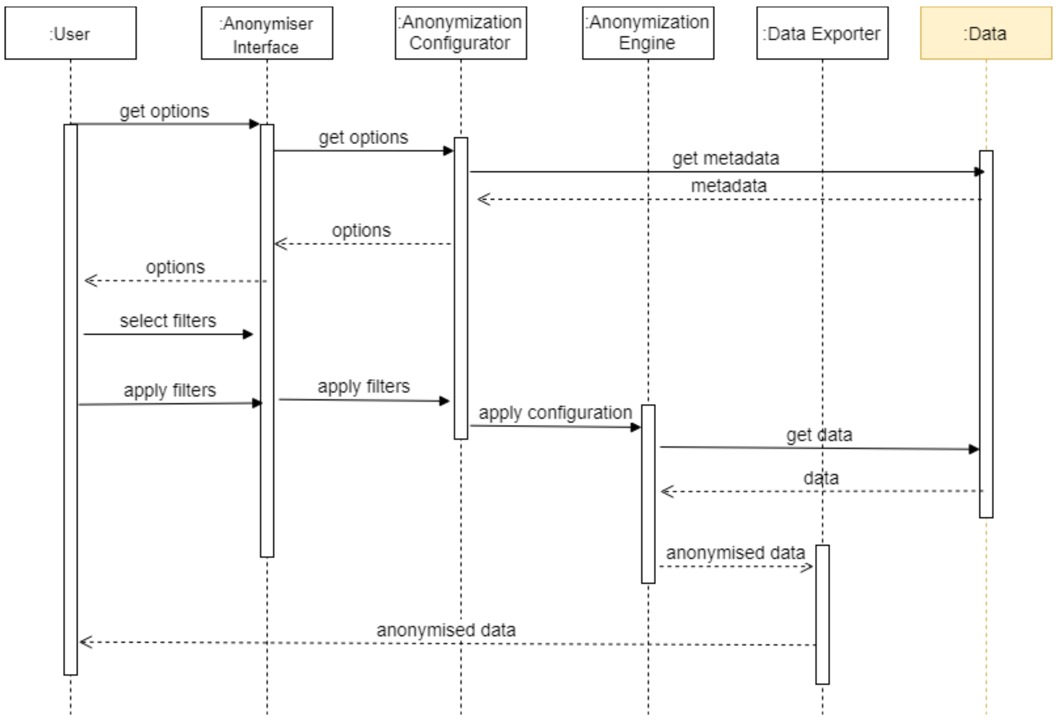


Figure 3-5: Data anonymisation sequence diagram

Updates from V1.0:

Added support for csv files for data input and output.

3.3.2. List of microservices

The anonymisation tool, i.e. the AEGIS Anonymiser, comprises two microservices which are orchestrated towards the execution of the anonymisation workflow. The first microservice (Mapping Service) includes the functionalities provided by the Anonymisation Configurator and Anonymisation Engine shown in Figure 3-5. In the same figure, the Data Exporter corresponds to the second microservice, i.e. the Exporter Service. The Anonymiser Interface orchestrates the two microservices towards applying the anonymisation process and constitutes the interaction point with the user where required.

Component Name	Microservice Name	Functionalities
Anonymiser	Mapping Service	• Connect to a database as data source • Read a csv file as data source • Provide anonymisation alternatives (e.g. generalisation, pseudonimity) per field • Apply the selected anonymisation action
	Exporter Service	• Provide the anonymised dataset through a REST API • Save the anonymised dataset as csv file

Table 3-5: Anonymisation Tool list of microservices

3.3.3. Technologies to be used

The anonymisation tool is based on Anonymiser, an anonymisation and persona-building tool, developed in the context of the European project CloudTeams.

The tool performs a type of generalisation, which can be used to achieve k-anonymity. 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 '≤ 18', the value '23' by '$20 < \text{Age} \leq 30$'. 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, using the Django web framework. These technologies will be also used to deliver the necessary updates and extensions in order to support the AEGIS anonymisation requirements. Specifically, the tool is extended to support csv files as data sources.

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 is extended and adapted to the project's requirements, but other open-source solutions may be also considered and evaluated, e.g. ARX, as complementary/supplementary tools.

Updates from V1.0:

No updates

3.3.4. APIs and exposed outcomes

The outcome of the Anonymisation tool is available through a REST API, documented in the following table.

Technical Interface	
Reference Code	AZ#01
Function	Retrieve the anonymised data
Subsystems	None / Standalone API
Type, State	
RESTful-API	
Endpoint URI	
<server url>: anonymizer/api/<secret key>/<parameters>	
Input Data	
Secret key: The secret access key generated for the user through the Anonymiser's user interface **Parameters:** Parameters for the data to be returned, including **limit**, **offset**, **filters on properties** and **count**	
Output Data	
The anonymised data in JSON format	

Table 3-6: Anonymisation tool technical interface

3.4. Brokerage Engine

3.4.1. Overview

The AEGIS Brokerage Engine has been modified to include both elements of the Data Policy and the Business Brokerage framework, while initial, generic models for those frameworks have been defined in order to roll-out a proof-of-concept implementation of the engine, that will be finalised after the delivery of D2.3 which will document the final models of the above mentioned frameworks, and will identify specific attributes that would be transferred to implementation and will be used by the engine to offer the features chosen by the consortium.

As identified above and mentioned in the previous deliverable, the Brokerage Engine includes both the Policy and Business Broker entities as shown in the next figure. In this context, the engine listens to activities that are to be performed on the AEGIS Cluster and the Data Store to prepare the Distributed Ledger Network records. These refer to adding new users on the cluster, which are also added to the ledger using the “Participant Registrant” microservice and to registering Assets on the Data Store, which are added to the ledger using the “Asset Registrant” microservice. Upon a transaction request, the “Transaction Checker” microservice checks each artefact’s metadata to conclude if a certain operation is possible. The first checks take place on the AEGIS Data Store, and the Brokerage Engine, in case the initial checks are ok, checks against its ledger to see whether a condition applies that does not permit the operation for the data artefact under observation. Finally, a successfully concluded transaction is marked on the ledger using the “Transaction Registrant” microservice.

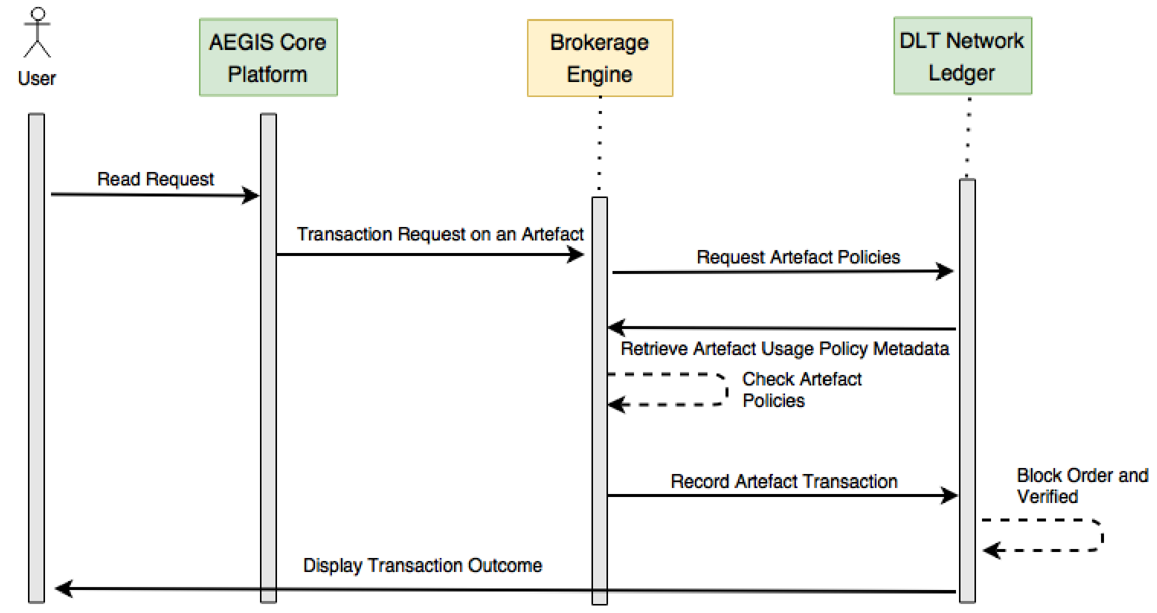

User
AEGIS Core Platform
Brokerage Engine
DLT Network Ledger
Read Request
Transaction Request on an Artefact
Request Artefact Policies
Retrieve Artefact Usage Policy Metadata
Check Artefact Policies
Record Artefact Transaction
Block Order and Verified
Display Transaction Outcome

Figure 3-6: Brokerage Engine sequence diagram

Updates from V1.0:
The main updates from V1.0 have to do with the design and the deployment of the DLT network that has been optimised and has been released as a set of (Docker) containers to be deployed as a core module to be co-hosted in the core AEGIS platform, and also be able to allow easy set-up and deployment of further nodes that will strengthen the networks integrity and credibility.

3.4.2. List of microservices

The microservices of the Brokerage Engine are tasked with the storage, checking and updating of data in the AEGIS Distributed Ledger network.

Component Name	Microservice Name	Functionalities
Brokerage Engine	Participant Registrant	• Listens to the registration facility of AEGIS • Registers the AEGIS users as participants of the Brokerage Engine of the AEGISDL network
	Asset Registrant	• Communicates with the Harvester and listens to datasets storing • Communicates with the Visualiser and listens to visualisations storing • Communicates with the Algorithm Execution Container and listens to analyses storing • Registers assets in the AEGIS Distributed Ledger network
	Transaction Checker	• Checks transaction details against details stored on the AEGIS Distributed Ledger
	Transaction Registrant	• Registers transactions in the Distributed Ledger • Updates the “wallets” of the transaction participants • Exposes executed transactions through a REST API

Table 3-7: Brokerage engine list of microservices

3.4.3. Technologies to be used

The prototype of the AEGIS Brokerage engine builds on top of the of Hyperledger Fabric[7] framework and provides an API that is consumed by the AEGIS platform for providing the interconnection between the core platform and the Brokerage Engine. The models of the Blockchain engine have been constructed based on the AEGIS DPF presented in deliverable D2.1, while Hyperledger Composer[8] is being used for testing and further optimising the overall engine, and for providing an interface to easily manage the overall network that has been deployed.

Updates from V1.0:

No change in technologies from the ones used in V1.0 has taken place

[7] https://www.hyperledger.org/projects/fabric

[8] https://hyperledger.github.io/composer/

3.4.4. *APIs and exposed outcomes*

The following tables present the most crucial interfaces used by the Brokerage Engine which are necessary for the interconnection with the AEGIS core platform.

Technical Interface	
Reference Code	BE#01
Function	User
Subsystems	Brokerage Engine
Type, State	
RESTful-API	
Indicative Endpoints	
GET /api/User **POST** /api/User **GET** /api/User/{id}	Get a list of all users registered with the brokerage engine Add a user to the brokerage engine Get user's details
Input Data (for POST)	

{
  "$class": "eu.aegis.User",
  "uid": "string",
  "balance": "0.0",
  "externalAssets": [
    {}
  ]
}	

Table 3-8: Brokerage Engine technical interface 1

Technical Interface	
Reference Code	BE#02
Function	AEGISAsset

Subsystems	Brokerage Engine
Type, State	
RESTful-API	
Indicative Endpoints	
GET /api/AEGISAsset **POST** /api/AEGISAsset **GET** /api/AEGISAsset /{id}	Get a list of all AEGIS assets Add an asset to the brokerage engine Get asset’s details
Input Data (for POST)	

{
  "$class": "eu.aegis.AEGISAsset",
  "aid": "string",
  "type": "Dataset",
  "cost": "0.0",
  "status": "Free",
  "exclusivity": "None",
  "contractText": "string",
  "owner": {}
}	

Table 3-9: Brokerage Engine technical interface 2

Technical Interface	
Reference Code	BE#03
Function	BuyAsset
Subsystems	Brokerage Engine
Type, State	
RESTful-API	
Indicative Endpoint	

POST /api/BuyAsset	Buy an asset
Input Data	

{
"$class": "eu.aegis.BuyAsset",
"buyer": {},
"relatedAsset": {},
"transactionId": "string",
"timestamp": "2018-03-21T10:10:29.343Z"
}	

Table 3-10: Brokerage Engine technical interface 3

Technical Interface	
Reference Code	BE#04
Function	ChangeAssetCost
Subsystems	Brokerage Engine
Type, State	
RESTful-API	
Indicative Endpoint	
POST /api/ChangeAssetCost	Change the cost of an AEGIS asset
Input Data	

{
"$class": "eu.aegis.ChangeAssetCost",
"newCost": 0,
"relatedAsset": {},
"transactionId": "string",
"timestamp": "2018-03-21T10:10:29.343Z"
}	

Table 3-11: Brokerage Engine technical interface 4

Technical Interface	
Reference Code	BE#05
Function	ChangeAssetStatus
Subsystems	Brokerage Engine
Type, State	
RESTful-API	
Indicative Endpoint	
POST /api/ChangeAssetStatus	Change the status of an AEGIS asset (i.e. free, paid etc)
Input Data	
{ "$class": "eu.aegis.ChangeAssetStatus", "newStatus": "Free", "relatedAsset": {}, "transactionId": "string", "timestamp": "2018-03-21T10:10:29.356Z" }	

Table 3-12: Brokerage Engine technical interface 5

Technical Interface	
Reference Code	BE#06
Function	ChangeExclusivity
Subsystems	Brokerage Engine
Type, State	
RESTful-API	
Indicative Endpoints	

POST /api/ChangeExclusivity	Change the exclusivity of an AEGIS asset (i.e. none, lifetime etc)
Input Data	

{
"$class": "eu.aegis.ChangeExclusivity",
"newPeriod": "None",
"relatedAsset": {},
"transactionId": "string",
"timestamp": "2018-03-21T10:10:29.368Z"
}	

Table 3-13: Brokerage Engine technical interface 6

Technical Interface	
Reference Code	BE#07
Function	LoadBalance
Subsystems	Brokerage Engine
Type, State	
RESTful-API	
Indicative Endpoints	
POST /api/LoadBalance	Loads currency to the balance of a user
Input Data	

{
"$class": "eu.aegis.LoadBalance",
"amount": 0,
"user": {},
"transactionId": "string",
"timestamp": "2018-03-21T10:10:29.384Z"
}	

Table 3-14: Brokerage Engine technical interface 7

3.5. AEGIS Data Store

3.5.1. Overview

The AEGIS Data Store has two distinct components; the HopsFS, which is the distributed file system mainly used for storing large amounts of data, as well as, the Linked Data Store, which is responsible for storing the metadata about the datasets.

3.5.2. HopsFS filesystem

The AEGIS Data Store component is responsible for storing data that were collected and curated by the Harvester. A distributed file system approach was chosen for flexibility, reliability, and scalability. The distributed file system will allow storing large amounts of data while enabling access to the data from other AEGIS supported services such as the Query Builder and the Visualiser. In particular, the distributed file system is primarily responsible for storing large files, that is, files ranging from megabytes to terabytes in size. However, as seen in many production Big Data clusters such as the ones at Yahoo and Spotify [1], it has been observed that almost 20% of the files in the cluster are less than 4 KB in size and as much as 42% of all the file system operations are performed on files less than 16 KB in size.

Under the hood, AEGIS uses HopsFS as the main file system to store the data. HopsFS is a reliable, highly scalable, and fault tolerant distributed file system. A file is stored as list of blocks that is triple replicated for fault tolerance. Unlike HDFS that stores the file system metadata in memory, HopsFS keeps all the file system metadata in an in-memory distributed database providing bigger clusters with higher throughput.

Updates from V1.0:

Efficient Storage of Small files less than 64 KB.

3.5.2.1. List of microservices

HopsFS runs as a service in the AEGIS cluster where users can interact with it using the AEGIS user interface and the REST API provided by Hopsworks. Under the hood, the AEGIS user interface communicates with HopsFS using the client APIs.

Component Name	Microservice Name	Functionalities
HopsFS	File System (Client/Web APIs)	• Perform file system operations such as create, mkdir, delete, append, etc

Table 3-15: HopsFS list of microservices

3.5.2.2. Technologies to be used

The AEGIS platform uses a file system, HopsFS, as the main store for Big Data. HopsFS is a drop-in replacement for Hadoop Distributed File System (HDFS). HopsFS is designed primarily to store large files, however, as reported most of production clusters contains a large number of small files (< 64KB). Therefore, we have extended HopsFS to efficiently manage large number of small files using a tiered storage solution. The tiers range from the highest tier where an in-memory database stores very small files (<1 KB), to the next tier where small files (<64 KB) are stored in Solid State Drives (SSDs), also using the database, to the largest tier, the existing Hadoop block storage layer for large files. Our approach is based on extending HopsFS with an inode stuffing technique, where we embed the contents of small files with the metadata and use database transactions and database replication guarantees to ensure the availability, integrity, and consistency of small files.

Updates from V1.0:
Efficient Storage of Small files less than 64 KB.
Some Bug Fixes for performance and usability.

3.5.2.3. APIs and exposed outcomes

The small files are handled transparently by the client and the file system without involving the users. It is recommended to interact with the data in HopsFS from the AEGIS user interface. However, HopsFS can be accessed using the command line, Java client APIs, and RESTful APIs.

Technical Interface	
Reference Code	EDS#01
Function	HopsFS FileSystem
Subsystems	HopsFS
Type, State	
RPC, Synchronous	
API Documentation	
https://hadoop.apache.org/docs/stable/api/org/apache/hadoop/fs/FileSystem.html	
Input Data	
Unsupported Calls: <ul><li>(get|set|unset)StoragePolicy</li><li>(get|set|list|remove)XAttr : At the moment adding extended metadata is done from Hopsworks</li><li>(set|get|remove)Acl</li><li>(create|rename|delete)Snapshot</li></ul>	

Output Data

Table 3-16: AEGIS Data Store technical interface 1

Technical Interface	
Reference Code	EDS#02
Function	HopsFS WebHDFS
Subsystems	HopsFS
Type, State	
RESTful-API, Synchronous	
API Documentation	
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	

Table 3-17: AEGIS Data Store technical interface 2

3.5.3. AEGIS Linked Data Store

The AEGIS Linked Data Store is responsible for storing the metadata associated with a particular dataset within the AEGIS platform. These metadata pose the foundation of the processing of the data within the AEGIS platform, since they offer detailed information about the semantic and syntax of the data. This allows choosing appropriate analysis and visualisation methods. The metadata will be stored as Linked Data, using the AEGIS ontology and vocabulary[9], which is based upon the DCAT-AP specifications. It will be developed as a service and integrated into the AEGIS Data Store.

Updates from V1.0:

The integration concept of the AEGIS Linked Data Store was revised. It will be available as a separate service and integrated via appropriate hooks within the AEGIS platform. This will allow a synchronisation of data and metadata. In addition, the interface of the service will be used from the AEGIS annotator, which will be integrated into the front-end of the platform. Besides the core functionality of managing the metadata associated with particular data the AEGIS Linked Data Store will offer a recommendation functionality. Based on given criteria, similar or relevant datasets will be returned. These criteria may include every available metadata, like the data structure, the spatial information or terms of use.

3.5.3.1. List of microservices

The AEGIS Linked Data Store is a combination of two microservices.

Component Name	Microservice Name	Functionalities
AEGIS Linked Data Store	MetadataService	• Creating and modifying the Linked Data metadata • Transform simple JSON to Linked Data • Recommendation engine for getting similar or suitable additional data
	OntologyManagementService	• Management of the AEGIS Linked Data vocabularies and ontologies • Exposes reusable namespaces for generating the metadata

Table 3-18: AEGIS Linked Data Store list of microservices

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

3.5.3.2. Technologies to be used

Updates from V1.0:

The used technologies were specified in more detail and already applied in the first prototype of AEGIS Linked Data Store. The service is written in Java. The core technologies are:

- Apache Jena Fuseki Triplestore
- Apache Jena Library
- Play Framework 2.6
- LinDA[10]

3.5.3.3. APIs and exposed outcomes

The AEGIS Linked Data Store will be exposed as one artefact and service.

Technical Interface	
Reference Code	AL#01
Function	Managing the AEGIS metadata
Subsystems	Triplestore
Type, State	
RESTful-API, SPARQL endpoint	
Endpoint URI	
http://aegis-metadata.fokus.fraunhofer.de/ (Testing system)	
Input Data	
Metadata as JSON or RDF	
Output Data	
Metadata as JSON or RDF	

Table 3-19: AEGIS Linked Data Store technical interface

[10] http://linda-project.eu/

3.6. AEGIS Integrated Services

3.6.1. Overview

The AEGIS platform provides a multi-tenant data management and processing services for Big Data. The multi-tenancy behaviour allows different users and services to securely and privately access and process their data. The AEGIS platform enables users to share their data with other users on the platform and allow access for specific services. In addition, users can use different data processing services that are supported by the platform to process and visualise their data. Under the hood, the data are mainly stored in the AEGIS Data Store; however, the AEGIS Data Store APIs are kept hidden from users. Instead, the AEGIS platform provides a Project/Dataset service to allow users to upload/download, explore, and do analysis on their data in a secure way without interacting with the AEGIS Data Store directly. To ensure secure and private access to the data, each user has an x509 certificate per project as well as a specific project user for the Data Store per project. The certificate has a CN field which contains the project specific username and that gives the platform the possibility to provide application level authorisation at the RPC server. For instance, any application executed within a YARN container will access the Data Store (HopsFS) as the user running this application. YARN acts as a proxy user for the user and accesses HopsFS (HDFS) through user impersonation. Thus, all accesses are seen as being done by the running user and storage access is limited to the files that can be accessed by this user. Moreover, each user has a specific project user for each of the projects that he/she can access. This means that any YARN application can only access files that are accessible for the running project and cannot normally access files cross projects, even if the project belongs to the same user as the one running the application. All applications running on top of YARN such as Spark, will be governed by the same storage access as described for a YARN container.

Under the hood, AEGIS builds upon Hopsworks to provide integrated support for different services such as interactive notebooks with Zeppelin and Jupyter that are used mainly by the Query Builder, Algorithm Execution Container and the Visualiser components. Other services such as Kafka, and ELK stack are also supported.

Updates from V1.0:

Some bug fixes for performance and usability.

Some bug fixes and improvements for certificates handling.

3.6.2. List of microservices

Hopsworks provides different integrated services that interact with each other and with users using the Hopsworks REST API.

Component Name	Microservice Name	Functionalities
Users	Auth	• Provides authentication functionality for users to login, logout, register, and recover password

	User	• Provides information about the current user
	Messages	• Provides an inbox functionality for users where they receive/send share requests for Datasets
Projects	Projects	• Provides information about the projects for a user, as well as details on each of the projects such as list of datasets, description, and team members
	Datasets	• Provides information about the datasets for a user. • It provides upload, download, and explore functionalities on the data

Table 3-20: AEGIS Integrated services list of microservices

3.6.3. Technologies to be used

AEGIS builds upon Hopsworks to provide multi-tenant data management and processing services for BigData. Hopsworks is a project-based multi-tenant platform for secure collaborative data science that runs on top of HopsFS. It provides an integrated support for different data parallel processing services such as Spark, Flink, and MapReduce, as well as a scalable messaging bus with Kafka, and interactive notebooks with Zeppelin and Jupyter. Hopsworks introduces new abstractions called Projects and Datasets that provide the basis for which users can securely upload and privately process data and securely collaborate with other users on the platform. A Dataset is a directory subtree in HopsFS that can be shared between projects. A Project is a collection of datasets, users, and notebooks (Zeppelin, Jupyter). In the AEGIS platform, Jupyter is mainly used by the Query Builder and the Visualiser components of the platform, while Zeppelin is mainly used by the Algorithm Execution Container. Most of the updates are bug fixes for usability and performance of the platform and the certificates handling.

Updates from V1.0:

Some bug fixes for performance and usability.

Some bug fixes and improvements for certificates handling.

3.6.4. APIs and exposed outcomes

The AEGIS platform APIs have not changed since most of the updates are bug fixes for platform usability and performance that will not require change of user facing APIs.

3.6.4.1. Users API

Hopworks provides a RESTful API to create users and to login to the platform. As documented in https://app.swaggerhub.com/apis/maismail/hopsworks-user-api/1.0.0

3.6.4.2. Projects and Datasets

Once logged in, users can create Project/Dataset, add member to a Project, share their Dataset, or upload/download/analyse their data. The RESTful API is documented in https://app.swaggerhub.com/apis/maismail/hopsworks-core-api/1.0.0

3.6.4.3. Interactive Notebooks

Users can create an interactive notebook in their Project using Zeppelin and Jupyter. Zeppelin and Jupyter are web-based notebooks that allow users to interactively analyse and visualise their data using different frameworks such as Spark. They only provide some basic charts; however, different JavaScript libraries could be loaded to support a more complex visualisation or the AEGIS Visualiser component could be utilised.

3.7. Query Builder

3.7.1. Overview

Query Builder is 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 is 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 is developed as interactive Notes inside Apache Zeppelin and Jupyter notebooks, offering intuitive data browsing, selection and manipulation facilitated through smart metadata usage in the background. As explained in section 3.2, the functionalities of Query Builder are not limited to the retrieval and combination of various datasets, but also support certain necessary processing tasks that cannot be known a priori, in terms of data filtering and cleansing. Thus, the tool incorporates functionalities that may conceptually be more relevant to the data cleansing tool. However, by integrating them in the current tool, there is a two-fold advantage: (a)the user is offered a more intuitive workflow, since data cleansing requirements may be not known prior to and independently of the query creation process and (b)the computational power of the AEGIS system is fully leveraged, as cleansing may be a very heavy process when dealing with big data.

The high-level functionalities to be offered by the Query Builder user interface are as follows:

- Dataset browsing powered by the available metadata
- Dataset selection and data preview
- Dataset merging and appending (metadata-enhanced)
- Data filtering, both row-wise and column-wise (metadata-enhanced)

- Various data manipulation and cleansing tasks, e.g. value replacement, fill-in of null values, changing column names, combining columns, removing duplicate entries etc. (metadata-enhanced)
- Save created view on data as new dataset
- Export the Python code that can be used to achieve the same data manipulation results that were created through the user's interaction with the UI
- Provide the Spark/Pandas DataFrame that corresponds to the created data view as input to Visualiser and/or Algorithm Execution Container
- Provide the Spark/Pandas DataFrame that corresponds to the created data view to the tech-savvy user that wants to directly use it in his/her code

The annotation "metadata-enabled" that is used in the above list refers to the fact that Query Builder leverages the metadata available for each file in the AEGIS system in order to provide its enhanced data selection and manipulation capabilities, i.e. enabling/disabling certain data merging and filtering options according to the data schema and also allowing the user to perform more targeted dataset exploration and retrieval based on the available metadata.

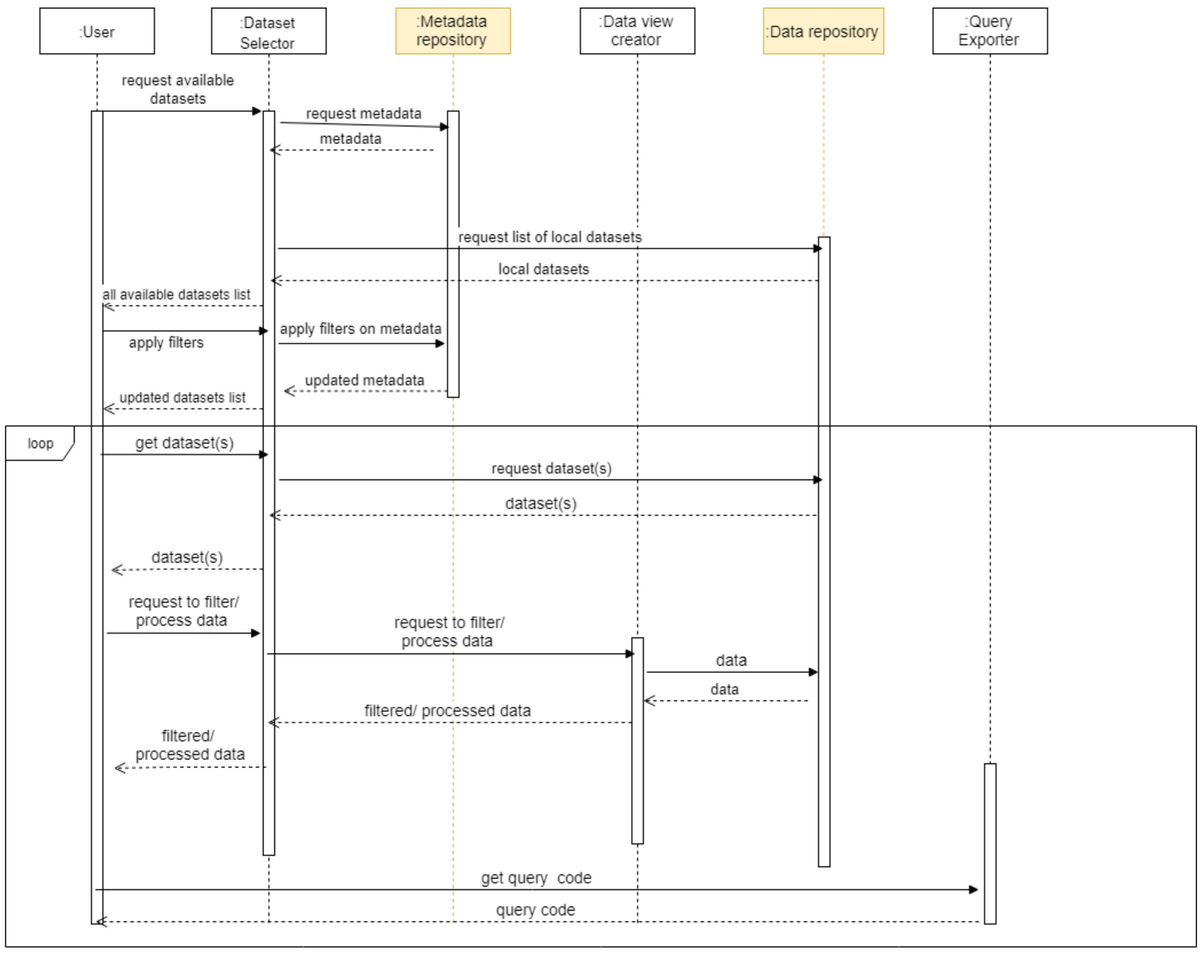


Figure 3-7: Query building and execution workflow

Updates from V1.0:

Query Builder is extended to offer simple data cleansing functionalities, such as replacing values, filling-in null values, removing duplicate entries etc.

3.7.2. List of microservices

Query Builder is one of the components developed inside a Notebook, as explained in Section 2.3. As such, the microservices of which it is composed correspond to specific functionalities also implemented inside the same Note of the Notebook. The microservices interact directly through Python code, as, in the normal workflow, they are all executed as parts of the same underlying process/job. Hence, the distinction of the five underlying microservices mostly corresponds to the conceptually separate tasks that are performed by each of them and the fact that in another context, i.e. externally to the Notebooks, they would constitute different services.

The first and last microservices (namely the Dataset Selector Service and the Query Exporter Service) correspond directly to sub-components of the Query Builder, which are shown in the Sequence diagram in Figure 3-7. The other three microservices (Cleanser Service, Merger Service and Dataset Creator Service) are integrated under the Data view creator part of the Query Builder (also shown in the corresponding sequence diagram).

Component Name	**Microservice Name**	**Functionalities**
Query Builder	Dataset Selector Service	• Interact with Data Harvester and Annotator to get the list of datasets and all related information accessible by the current user • Provide the available datasets as a list for the user to directly select and through a faceted search widget to progressively narrow down results • Acquire the list of available and accessible datasets from HopsFS of AEGIS Data Store • Provide informative metadata for each available dataset • Adjust the list of datasets shown to the user based on the performed choices and provide semantically enhanced suggestions • Retrieve the selected dataset from the filesystem and load it into a DataFrame
	Dataset Cleanser Service	• Remove/ Replace missing values • Perform data interpolation

		• Provide a preview of the applied actions • Provide aggregations and other statistics that help examine data integrity • Apply rule-based data transformations
	Dataset Merger Service	• Merge/Join datasets • Perform approximate joins
	Dataset Creator Service	• Apply aggregations on datasets • Select/Drop columns • Apply value replacing • Rename columns • Perform data interpolation • Apply selectors/ filters to dataset to refine the retrieved data • Save a dataset as a file in the filesystem • Provide a preview of the applied actions • Load created dataset in a DataFrame
	Query Exporter Service	• Translate data processing/filtering/merging actions performed so far into Python code that can be used externally to the tool to produce the same results • Export dataset to new file

Table 3-21: Query Builder list of microservices

3.7.3. Technologies to be used

Query Builder is developed as a preconfigured Note in the Apache Zeppelin Notebook and in Jupyter Notebook. Specifically for Zeppelin, its user interface is developed using JavaScript and AngularJS, whereas the backend functionalities are developed mainly in Python and PySpark. For certain tasks, mostly related to Zeppelin's internal way of sharing variables across paragraphs of different programming languages, Scala is also used. The Query Builder version created for Jupyter is implemented in Python and PySpark for the data processing and JavaScript for the user interface.

In order to provide effective big data querying and processing functionalities, Query Builder leverages the power of Apache Spark, which is available inside AEGIS Integrated Services (presented in section 3.6).

Updates from V1.0:

Although the initial decision was to utilise Apache Zeppelin for the implementation of the Query Builder, a transition to the Jupyter Notebook was decided, as experimentation proved it to be a more stable environment for building the required extensions. The Zeppelin version remains available in the platform; however it is expected that as the project progresses, the Jupyter version of Query Builder will be the only one used, due to its superior performance, which is related to Jupyter features.

3.7.4. APIs and exposed outcomes

Query Builder has two types of exposed outcomes:

1. The Python code that corresponds to the actions performed by the user through the tool's user interface. The generated code can be used then by the user independently in order to achieve the same results without having to repeat the performed steps and can also be directly edited by more tech-savvy users.
2. The DataFrame that contains the data view that was created through all the data manipulation tasks performed by the user. The DataFrame can be passed to the Visualiser and the Algorithm Execution Container or be directly manipulated inside Zeppelin/Jupyter through the user's custom code.

Although these are the main outcomes of the tool, there are also two more possible outcomes that may be produced through the user's usage of the tool:

1. New files may be created and stored in the local filesystem
2. The Zeppelin/Jupyter note that is created may itself serve as an outcome if the user chooses to save and keep it for future reference.

3.8. Visualiser

3.8.1. Overview

The Visualiser is the component enabling the visualisation capabilities of the AEGIS platform for the output of the querying and filtering results coming from the Query Builder as well as the output of the analysis results as produced from the Algorithm Execution Container. More specifically, the Visualiser is undertaking the necessary actions to address the advanced visualisation requirements of the AEGIS platform by offering a variety of visualisation formats, which span from simple static charts to interactive charts offering several layers of information and customisation.

The Visualiser is implemented as predefined Jupyter notebook, which is part of the AEGIS Integrated Services enabling the interactive web-based notebook functionality in the AEGIS platform. The Visualiser component consists of a set of functionalities which support the execution of the visualisation process. This set of functionalities includes the dataset selection, the dataset preview generation, the visualisation type selection, the visualisation configuration, the visualisation generation and the interactive dashboard. In the following list, the functionalities of the Visualiser are elaborated:

- Dataset selection: The list of available datasets within the project are presented to the user for selection[11].
- Dataset preview generation: Upon selecting the dataset, a preview of the dataset is displayed[11].
- Visualisation type selection: The list of available visualisations for the selected dataset is presented to the user for selection.
- Visualisation configuration: Based on the visualisation type selected for the desired dataset a set of parameters are displayed to the user to trim the visualisation.
- Visualisation generation: Once the visualisation type along with the parameters are set for the desired dataset, the visualisation generation is triggered. The results can be used in the current session for creating an interactive dashboard.
- Dashboard: The result of the visualisation generation is presented to the user into an interactive dashboard. This dashboard can contain also multiple generated visualisation results.

Figure 3-8 depicts the execution of the visualisation process.

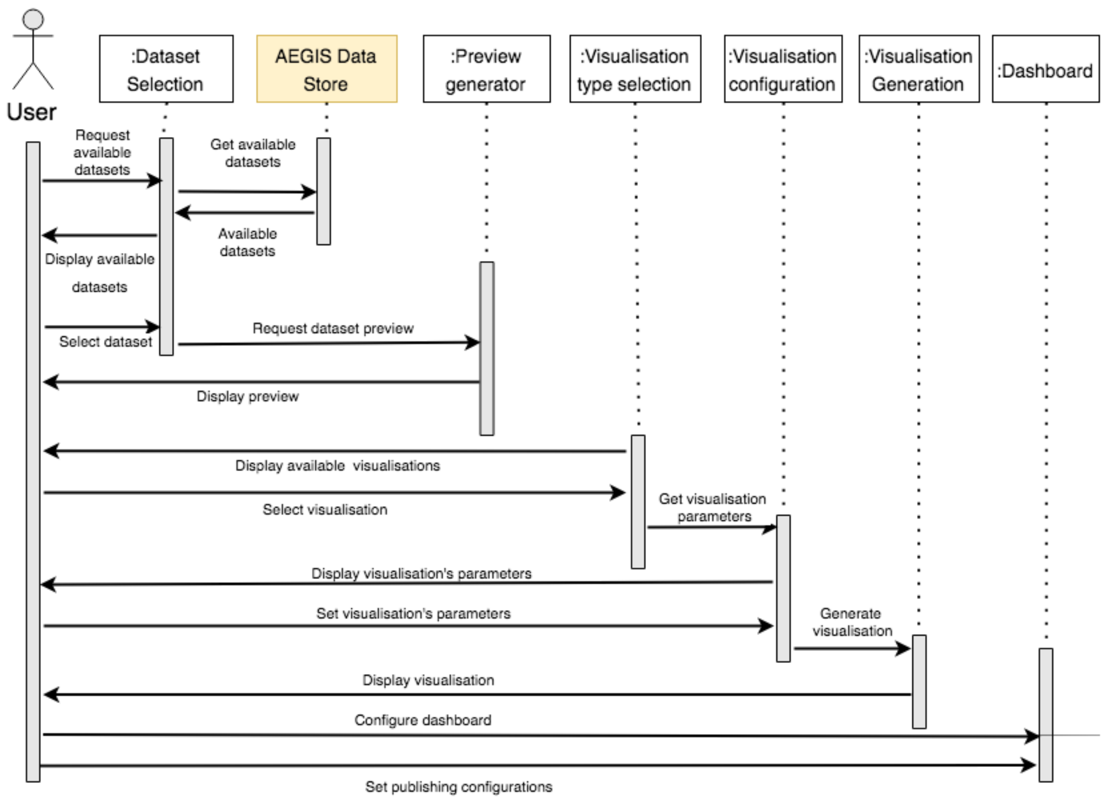

User
:Dataset Selection
AEGIS Data Store
:Preview generator
:Visualisation type selection
:Visualisation configuration
:Visualisation Generation
:Dashboard
Request available datasets
Get available datasets
Available datasets
Display available datasets
Select dataset
Request dataset preview
Display preview
Display available visualisations
Select visualisation
Get visualisation parameters
Display visualisation's parameters
Set visualisation's parameters
Generate visualisation
Display visualisation
Configure dashboard
Set publishing configurations

Figure 3-8: Sequence diagram of the visualiser component

In the first version of the Visualiser a variety of visualisation types is supported and will be further extended in the upcoming versions. The list of supported visualisation types includes

[11] This functionality will be obsolete once the AEGIS notebooks are integrated into one holistic notebook, as described in Section 2.3. Within this holistic notebook, the Visualiser will receive the input for visualisation directly from the Query Builder or the Algorithm Execution Container.

scatter plot, bar chart, pie chart, bubble chart, time series, heat map, map, box plot and histogram.

Updates from V1.0:

No updates in regards to the design of the Visualiser as documented in D3.2. In the upcoming versions of the Visualiser the list of available visualisation types will be further extended and the support for multiple datasets will be explored.

3.8.2. List of microservices

The Visualiser component, as explained in Section 2.3, is developed as a predefined Jupyter notebook and it is following the microservices architecture. The designed microservices are enabling the advanced visualisation capabilities of the AEGIS platform and are orchestrated towards the execution of the visualisation process, as described in the previous section. Each microservice is assigned with a specific functionality within the visualisation process and is implemented as a note in the same notebook, interacting through Python code.

In particular, the Dataset Selection and the Preview Generator, as shown in Figure 3-8, are undertaken by DatasetSelectionService microservice. Additionally, the microservice VisualisationSelectionService is responsible for the Visualisation Type Selection and the Visualisation Configuration. The Visualisation Generation is handled by the ChartBuildingService and ChartCreationService microservices. Finally, the microservice VisualisationService is handling the Dashboard functionality.

In total five microservices will be developed and are described in the following table:

Component Name	**Microservice Name**	**Functionalities**
Visualiser	DatasetSelectionService[12]	• Acquire the list of available and accessible datasets from HopsFS of AEGIS Data Store • Provide the list of available datasets for selection • Provide a preview of the selected dataset
	VisualisationSelectionService	• Provide the list of available visualisation types • Provide and manage the parameters (such as axis

[12] This microservice will be obsolete once the AEGIS notebooks are integrated into one holistic notebook, as described in Section 2.3. Within this holistic notebook, the Visualiser will receive the input for visualisation directly from the Query Builder or the Algorithm Execution Container.

		variables and titles) for each visualisation type
	ChartBuildingService	• Prepare the data in the appropriate format based on the selection of the visualisation type and set parameters
	ChartCreationService	• Generate the appropriate visualisation based on the data, visualisation type and parameters
	VisualisationService	• Display the generated visualisation as a UI component

Table 3-22: Visualiser list of microservices

3.8.3. Technologies to be used

As already described the Visualiser component is implemented as a predefined Jupyter notebook, a multipurpose interactive web-based notebook service for running Spark code on Hops YARN, which is part of the AEGIS Integrated Services. Jupyter offers functionalities for data visualisation out-of-the box in addition to data ingestion, data discovery and data analytics functionalities. In addition to Jupyter, the user interface is implemented using Python, JavaScript and HTML with the support of two Python libraries, namely the Folium[13] and the highcharts[14] libraries. These specific libraries were selected as they provide state-of-the-art visualisations specialised in charts and data visualisation.

Updates from V1.0:

Although the initial decision was to utilise Apache Zeppelin for the implementation of the Visualiser, it was decided to utilise Jupyter instead. The reason for this transition was the list of functionalities and features offered by Jupyter that facilitate the implementation of the Visualiser, as well as the stability and performance of the Jupyter service compared to the Zeppelin service. As both notebooks are Python based, the transition was executed with minimal effort.

In addition to the transition to a Jupyter notebook, two of the state-of-the-art charting libraries in Python, Folium and highcharts, were used to enable advanced visualisations.

[13] http://folium.readthedocs.io/en/latest/

[14] https://www.highcharts.com/

3.8.4. APIs and exposed outcomes

The Visualiser is providing as an exposed outcome the visualisation generated. The Visualiser is generating the visualisation tailored by the user, taking as input either the results of the query processing as facilitated by the Query Builder or the analysis results that are provided as the outcome of Algorithm Execution Container. The produced visualisation can be saved as an image or introduced in an interactive dashboard.

3.9. Algorithm Execution Container

3.9.1. Overview

Analytics in AEGIS are to be constructed with the use of the Algorithm Execution Container, which is a module that runs on top of a web-notebook and can be used either on its own, or as a follow-up to the Query Builder notebook.

With the aim to provide extra functionalities to both novel and non-expert users, this component features a UI that consists of an algorithm selection template, offering to users some basic information regarding each algorithm available in the big data analysis platform of AEGIS.

Initially, the user has to select a Dataset which will be the basis for the analysis, and the "Dataset Selector" microservice is triggered to retrieve the dataset from the AEGIS Data Store. Following this, the user proceeds with the selection of an algorithm (out of an algorithm family), specific parameters of each algorithm are presented, to provide to users the option to fill in all variables of the algorithm and perform an analysis over the platform. These actions are collected by the "Analysis Trigger" microservice, which is composed of a series of nested or interlinked notebook paragraphs and that is passing over the analysis request to the underlying Spark engine. The output of an analysis is then generated by the Algorithm Execution Container and the performance of each algorithm is being previewed in the same notebook and is saved back into the AEGIS Data Store using the "Analysis Exporter" microservice.

Moreover, the execution of a new algorithm using the outputs of the previously conducted analysis may lead to the enablement of "chainable" analyses.

The current design of the Algorithm Execution Container will support the following categories of analyses:

- Dimensionality Reduction/Feature Extraction/Selection
- NLP Functions
- Recommenders
- Clustering
- Classification/Regression

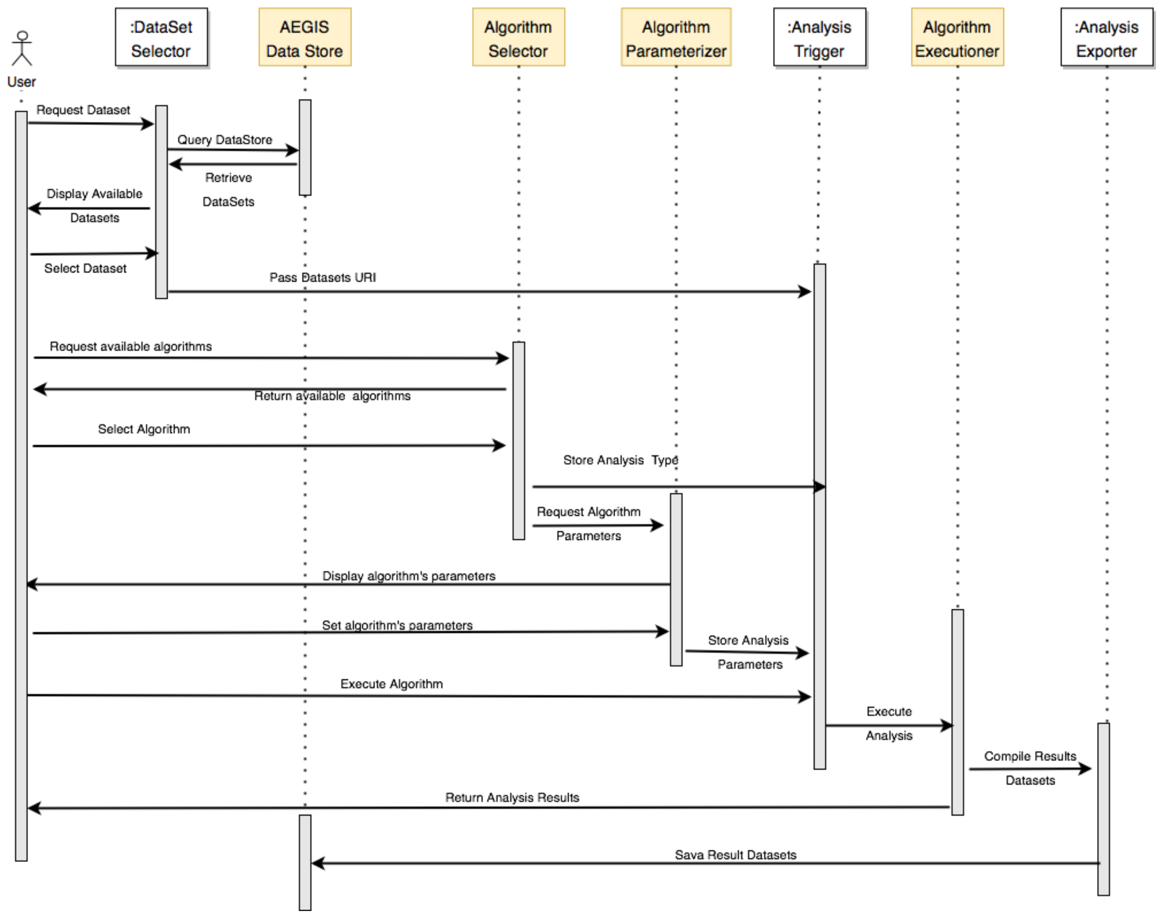


Figure 3-9: Algorithm Execution Container sequence diagram

Updates from V1.0: Updates from V1.0 include the way to load and save datasets into the filesystem, and the definition of the analysis outputs that are to be stored as well. Furthermore, chainable execution of algorithms has been made available, by repeatedly running algorithms as sequence after previous analyses.

3.9.2. List of microservices

The Algorithm Execution Container is a component that is developed inside a Zeppelin notebook and is part of the overall analysis functionality of the platform. The microservices interact with the backbone through Python code.

Component Name	Microservice Name	Functionalities
	Dataset Selector	• Interacts with the AEGIS storage and directly selects a dataset by its URI

Algorithm Execution Container	Analysis Trigger	• Selects the analysis to be performed by the user • Passes the analysis parameters to the analytics function • Triggers the analysis to be performed in the AEGIS Spark
	Analysis Exporter	• Stores the analysed dataframe in the AEGIS storage point • Stores the analysis outputs in the AEGIS storage point

Table 3-23: Algorithm Execution Container list of microservices

3.9.3. Technologies to be used

The Front-End of the Algorithm Execution Container is based on AngularJS framework running on top of an Apache Zeppelin notebook as already pre-defined paragraphs that present the UI to the user. MLlib is the core algorithm library to be supported. Due to certain constrains in the interoperation between AngularJS and MLlib under the environment of Zeppelin, Scala is being used to configure and execute the algorithms that are selected by the user. Python is being used as the language to interpret the paragraphs in Zeppelin.

Updates from V1.0:

No update in technology from version V1.0

3.9.4. APIs and exposed outcomes

No APIs are being provided by this component, as it directly interacts with the Zeppelin Notebook. The pre-generated Python code that corresponds to the actions performed by the user through the tool's user interface can be used then by the user independently in order to achieve the same results without having to repeat the performed steps and can also be directly edited by more tech-savvy users. The outcomes of the component are passed back to the AEGIS storage facility, if the user chooses to save and keep them for future reference.

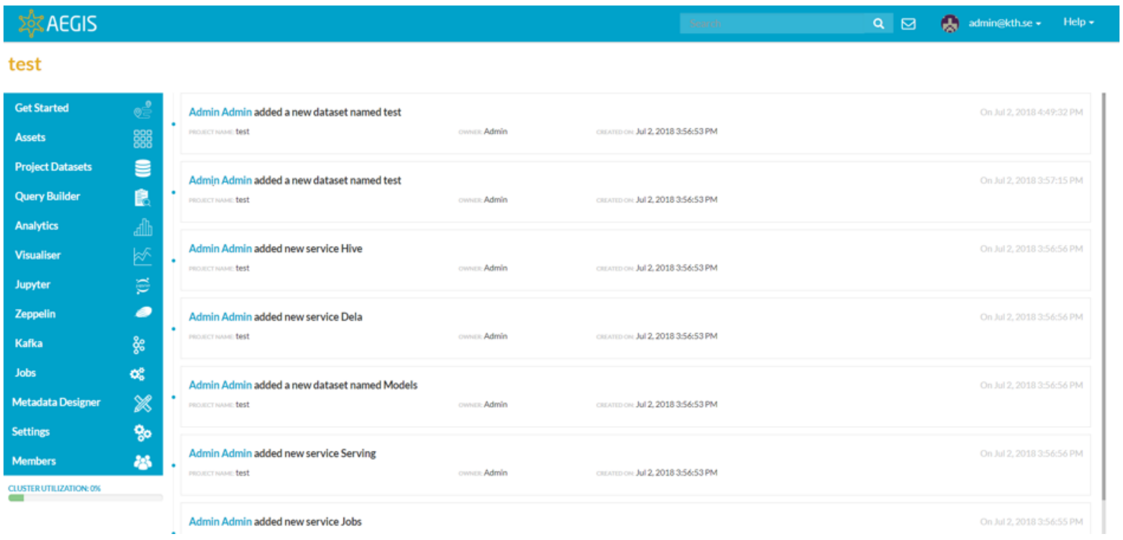
3.10. AEGIS Front-End

3.10.1. Overview

The AEGIS Front-End, as shown in Figure 3-10, is the upper layer of the whole AEGIS architecture, receiving and sending the outputs/inputs from/to the AEGIS API layer and from/to the AEGIS Harvester/Annotator.

The first step to access the platform is the creation/authentication of an account. Then the user will be able to browse the public assets (e.g. datasets, visualisations, etc.) according to the following filters: most popular, latest, and offers, or to select a project from a menu which includes the user and the public projects. Moreover, a new project can be created, with the option to specify the related members. At this stage, users can be assigned the roles of data owner/data scientist, with different permissions for the management of the projects/datasets. In next release of the platform, the users' roles could be reviewed according to the methodology developed in D1.3.

The Front-End facilitates all the AEGIS components which have an interaction with the user (e.g. Query Builder, Visualiser, Algorithm Execution Container). A common graphical environment has been developed, according to the look and feel of the AEGIS institutional web site, including direct links to the single web components corresponding to the AEGIS main functionalities, here integrated in the form of notebooks. In particular, the main menu of the AEGIS platform (see the figure below), for each project, presents the following items: Get Started, Assets, Project Datasets, Query Builder, Analytics, Visualiser, Jupyter, Zeppelin, Kafka, Jobs, Metadata Designer, Settings, Members. The applications related to Queries, Visualisations and Analytics have been implemented within Apache Zeppelin or Jupyter in form of notebooks.



Admin Admin added a new dataset named test — On Jul 2, 2018 4:49:32 PM
PROJECT NAME: test OWNER: Admin CREATED ON: Jul 2, 2018 3:56:53 PM

Admin Admin added a new dataset named test — On Jul 2, 2018 3:57:15 PM
PROJECT NAME: test OWNER: Admin CREATED ON: Jul 2, 2018 3:56:53 PM

Admin Admin added new service Hive — On Jul 2, 2018 3:56:56 PM
PROJECT NAME: test OWNER: Admin CREATED ON: Jul 2, 2018 3:56:53 PM

Admin Admin added new service Dela — On Jul 2, 2018 3:56:56 PM
PROJECT NAME: test OWNER: Admin CREATED ON: Jul 2, 2018 3:56:53 PM

Admin Admin added a new dataset named Models — On Jul 2, 2018 3:56:56 PM
PROJECT NAME: test OWNER: Admin CREATED ON: Jul 2, 2018 3:56:53 PM

Admin Admin added new service Serving — On Jul 2, 2018 3:56:56 PM
PROJECT NAME: test OWNER: Admin CREATED ON: Jul 2, 2018 3:56:53 PM

Admin Admin added new service Jobs — On Jul 2, 2018 3:56:55 PM

Figure 3-10: Main menu from the AEGIS platform

Updates from V1.0:

- New look and feel developed, according to the AEGIS institutional web site.
- Improved management of user account creation.
- Minor adjustments in the home page
- New Visualisations functionality added
- New Query Builder functionality added
- Bugs fixing.

3.10.2. List of microservices

A list of microservices will be developed in order to handle the selection of public assets (most popular, latest, offers). In total three microservices will be developed and are described in the following table:

Component Name	Microservice Name	Functionalities
Front-End	GetMostPopularAssets	• Return the sorted list of most popular public assets (e.g. datasets, projects, etc.)
	GetLatestAssets	• Return the sorted list of latest public assets (e.g. datasets, projects, etc.)
	GetOffers	• Return the sorted list of public assets characterised by special price conditions

Table 3-24: AEGIS Front-End list of microservices

3.10.3. Technologies to be used

The AEGIS Front-End is built on top of the Hopsworks platform. Hopsworks is a self-service User Interface 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 or Jupyter Notebooks (iPython notebook style web applications). While developing the AEGIS platform, a central role has been taken by the notebook technology, providing the technology required to implement in particular the following components: Query Builder, Visualiser and Algorithm Execution Container. Apache Zeppelin and Jupyter are the selected notebook frameworks (more details in section 3.6). Another important framework which has been used for the development of the graphical user interface is AngularJS[15].

AngularJS is a very powerful JavaScript based development framework to create Rich Internet Application (RIA[16]). It is used mostly in Single Page Application (SPA[17]) projects. It extends the HTML DOM with additional attributes and makes it more responsive to user actions. AngularJS is open source, completely free and used by thousands of developers around the world. It is licensed under the Apache License version 2.0. Applications written in AngularJS are cross-browser compliant. AngularJS automatically handles JavaScript code suitable for each browser and allows to implement the Model-View-Controller (MVC[18]) pattern on the client side using JavaScript.

[15] https://angularjs.org/
[16] https://en.wikipedia.org/wiki/Rich_Internet_application
[17] https://en.wikipedia.org/wiki/Single-page_application
[18] https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller

Updates from V1.0:

Usage of Jupyter notebooks.

3.10.4. APIs and exposed outcomes

No API will be provided in this version.

3.11. Holistic Security Approach

3.11.1. Overview

In the high-level architecture of the AEGIS platform the consortium identified the need for a holistic security approach that should be incorporated throughout the AEGIS platform and that will be applied in the whole lifecycle of the data exploitation safeguarding the security aspects of data in storage, in transit and in use. It should be noted at this point that the holistic security approach is not a standalone component, but rather a set of technologies and tools that are utilised within the components of the AEGIS platform in order to enable cross-platform security.

In the updated high-level architecture, as presented in D3.2, the main decision taken was the adoption of the Hopsworks[19] platform as the Big Data Processing Cluster of the AEGIS platform. Hopsworks is providing out-the-box the HopsFS, a new implementation of the Hadoop Filesystem (HDFS), covering the storage solution of the AEGIS platform. With respect to the security aspect for data in storage HopsFS is offering advanced security with a plethora of authentication mechanisms as well as data access control, data integrity and data consistency mechanisms. HopsFS is making use of checksum to ensure security and integrity control of the data in storage covering the envisioned by the consortium security aspect for data in storage. It should be noted at this point that several other solutions, like the use Symmetric Encryption Algorithms, Asymmetric Encryption Algorithms and Attribute-Based Encryption, have been evaluated by the consortium in order to address the security, privacy and integrity of the data stored in the AEGIS Data Store, however it was decided that those technologies are not ideal for big data ecosystems due to efficiency problems that will introduce within the data analysis process.

Concerning the security of data in transit or data in motion, which includes data transfer between the Hopsworks services and clients either within the internal network or through the internet, Hopsworks is providing data encryption via Secure Sockets Layer (SSL) and Transport Layer Security (TLS) at the RPC layer offering the required security level as envisioned by the AEGIS consortium. The third aspect of the holistic security approach is related to security of "data in use" 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). Although the AEGIS

[19] http://hops.io

consortium has already identified a list of candidate technologies, such as Homomorphic Encryption and Verifiable Computation, it was decided that the evaluation and adoption of such technologies will be included in the upcoming releases of the AEGIS platform.

The holistic security approach also covers the security aspects for the technical interfaces (e.g. REST) provided by the platform. This includes the interfaces provided by the components of the platform in regards to the authorisation, authentication and access approval mechanisms. Although Hopsworks REST API has already security mechanisms in place, more specifically it has session-based security with JSession tokens and x509 certificates for the user of every project, the consortium decided to introduce a token-based authentication with JSON Web Token (JWT)[20]. JWT is an open standard (RFC 7519) that defines a compact and self-contained way for securely transmitting information between parties as a JSON object.

For the introduction of the JWT as authentication and secure information exchange mechanism, a series of actions is needed. At first, a new service will be introduced and will be integrated in the backend of the AEGIS platform undertaking the token generation upon successful login, as well as token verification. The methods included within this service implement the access control mechanism that will be introduced in the AEGIS platform. In addition to the service, a new filter method will be introduced. This filter will replace the existing filter method used for the JSession tokens utilised currently in every technical interface and will perform the token verification for each incoming request. The integration of this new mechanism is an ongoing activity that will last till M24 when the new version of the AEGIS platform will be released.

The following table presents the holistic security approach of AEGIS platform for the data lifecycle security as described above.

Security Aspect	Proposed Approach	Adopted Approach	Remarks
Data in Storage	Symmetric Encryption Algorithms, Asymmetric Encryption algorithms and Attribute-Based Encryption	HopsFS mechanisms for authentication, authorisation and access control of stored data. Usage of checksums for data integrity.	The proposed technologies were evaluated and will not be introduced for performance and efficiency reasons.
Data in Transit	Secure Sockets Layer (SSL) and Transport Layer Security (TLS)	Hopsworks provides SSL and TLS data encryption and authentication at the RPC layer.	SSL and TLS encryption are the de-facto standard in the security of data in transit.

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

Data in Use	Homomorphic Encryption, Verifiable Computation	Currently none of the technologies has been adopted	The evaluation and adoption of the proposed technologies will be included in the upcoming versions of the platform.
Technical Interfaces	Token-based authentication and authorisation mechanism with JSON Web Token	JSON Web Token will be introduced.	The implementation and integration of JSON Web Token mechanism is an ongoing activity.

Table 3-25: Holistic Security Approach summary

Updates from V1.0:

There are no updates on the design and specification of the holistic security approach. The introduction of the JSON Web Token as an authentication and access control mechanism is an ongoing activity that will last till M24 when the new version of the AEGIS platform will be released.

3.11.2. Technologies to be used

The holistic security approach will be based on the following three technologies, one for each aspect of the approach. Concerning the data in storage aspect the usage of checksum was selected. Checksum is a small-sized datum derived as the outcome of the cryptographic hash function or checksum algorithm on a block of data or file. This outcome is utilised to identify data corruption errors or modifications and overall data integrity since even small changes will produce a different outcome.

With regards to the data in transit or data in motion security aspect the SSL and TLS cryptographic protocols are the de-facto standard for secure communication over the network. It ensures the secure connection by eliminating the unauthorised read and modification of the data in transit. TLS is an updated more secure version of SSL, introducing the symmetric cryptography with unique keys based on a shared secret for each connection. Each communicating parties is using a public-key to authenticate and the data integrity evaluation is performed with the use of message authentication code.

For the security of the technical interfaces of the platform JSON Web token (JWT) will be introduced for authentication and secure information exchange purposes. JSON Web Token (JWT) is an open standard (RFC 7519) utilising digitally signed using JSON Web Signature (JWS) and/or encrypted using JSON Web Encryption (JWE) JSON objects as a safe way to represent a set of information between two parties. As a consequence, this token is composed by a header, a payload and a signature. JWT is used for authentication purposes, as the token produced during the login authentication is defining the access level for the routes, services and resources of the platform. JWT can be also used for secure information transfer between communication parties.

Updates from V1.0:

No updates in terms of used technologies.

3.11.3. API

Not applicable.

4. User Interaction Workflows

In the current section, the main workflows that facilitate the data-driven innovation in the PSPS domains are presented, as documented in section of 4 of deliverable D3.2, with the necessary adaptations based on the updates of the components of the AEGIS platform.

All workflows are focusing on the user perspective and the purpose of this section is to hide the technical details on how the AEGIS components are interacting and on the internal processes of each component but rather illustrate the provided functionalities of AEGIS platform. All workflows are modelled in BPMN diagrams and on each workflow, a specific functionality is presented involving one or more components described 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 figure below shows the interactions between a user and the AEGIS user interface. A new user can create a new account by providing his/her name and password, and then wait for admin approval before being able to use the platform. In addition, a 2-factor authentication password could be used if enabled.

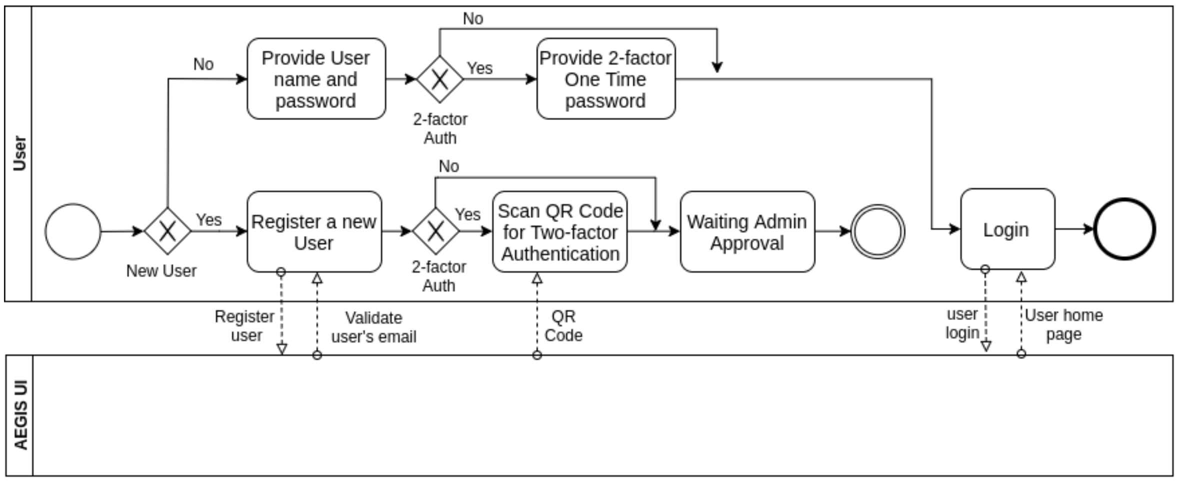


Figure 4-1: Sign-up and Login workflow

4.2. Data import

4.2.1. Importing data for a new dataset

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.

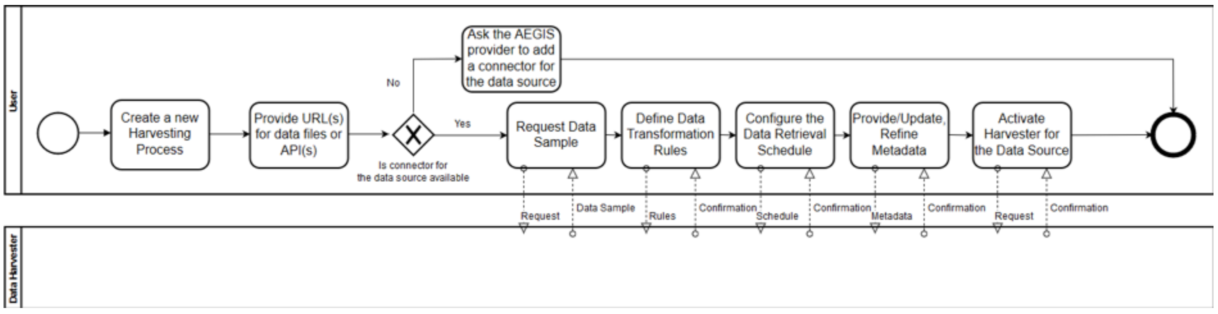

User
Data Harvester
Create a new Harvesting Process
Provide URL(s) for data files or API(s)
Is connector for the data source available
No
Yes
Ask the AEGIS provider to add a connector for the data source
Request Data Sample
Define Data Transformation Rules
Configure the Data Retrieval Schedule
Provide/Update, Refine Metadata
Activate Harvester for the Data Source
Request
Data Sample
Rules
Confirmation
Schedule
Confirmation
Metadata
Confirmation
Request
Confirmation

Figure 4-2: Importing data and metadata and registering them as a part of a new dataset

4.2.2. Anonymisation workflow

As explained in Section 3.3, anonymisation is performed offline, prior to uploading any potentially sensitive data to the core AEGIS platform. Anonymisation may be an iterative process, as several actions may be required until all personal information has been stripped off the original dataset. 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.

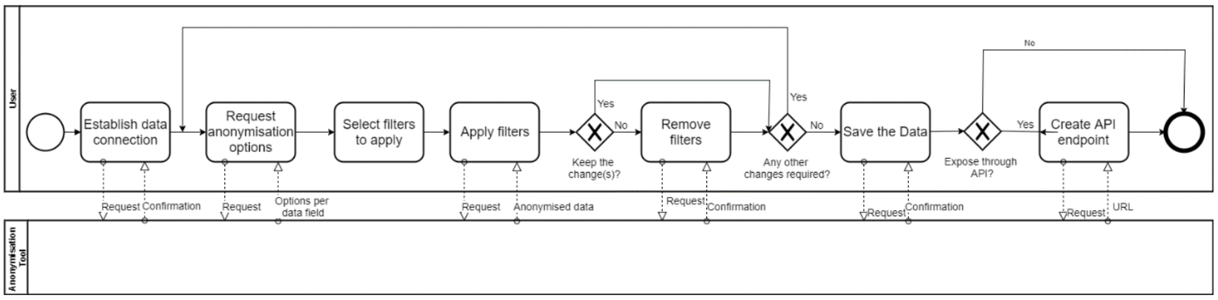


Figure 4-3: Data anonymisation workflow

4.2.3. Data cleansing workflow

As explained in section 3.2, AEGIS offers both online and offline cleansing functionalities, which may span from simple value replacements to more complex and computationally intense data manipulations. The offline data cleansing is performed through a dedicated AEGIS offline tool that offers an intuitive data cleansing workflow which is presented in the following figure.

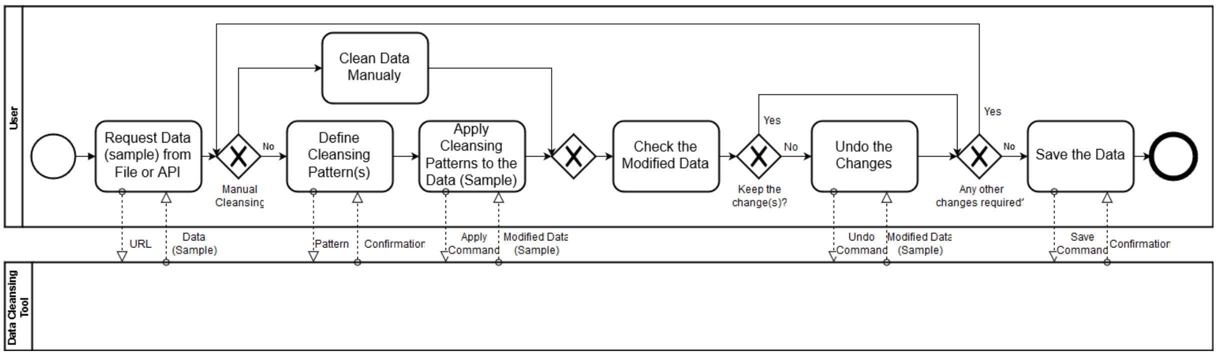


Figure 4-4: Data cleansing workflow

Regarding the online data cleansing, due to the flexibility offered by the Notebooks, there is no unique workflow to follow. However, the workflows that include the usage of Notebook-based components (e.g. the ones in sections 4.3.2 and 4.6) provide some insights on the expected user interaction.

4.3. Data and service exploration (search)

4.3.1. From the main AEGIS platform

The figure below presents the main actions the users can take to explore the data on the AEGIS platform. The user can request all his/her projects and datasets, and navigate to any project or dataset. Once in a dataset, the user can browse, upload, or download files.

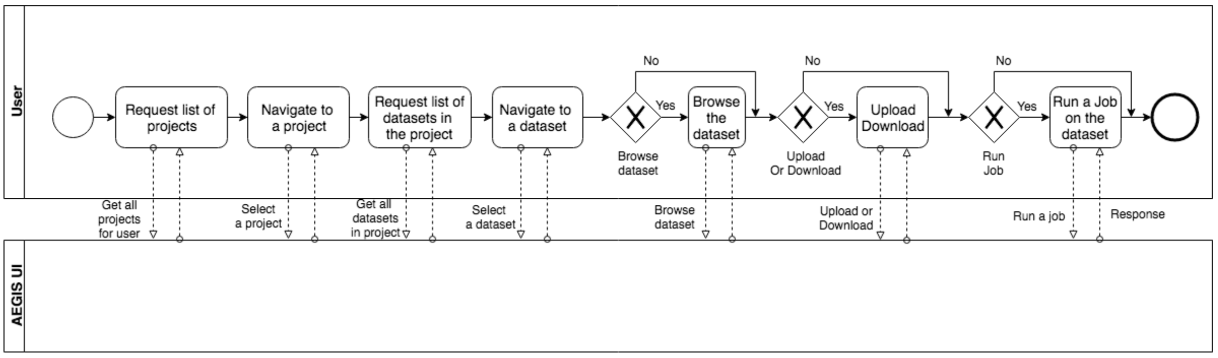


Figure 4-5: Data and service exploration workflow

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 (more accurately create a view on selected data) to be fed into analysis and/or visualisation or to be saved as a new dataset. The process of "creating an appropriate dataset" includes also the cleansing functionalities that have been integrated in the Query Builder (through the selection, configuration and application of the available filters). Although the user primarily interacts with the Query Builder component, two more components are utilised in the background. The AEGIS Linked Data Store to retrieve the metadata for the available datasets, which offer valuable information to the user and facilitate the data selection. The Brokerage Engine is involved in the dataset acquisition sub-process shown in the diagram, which is an instance of the artefact sharing process described in Section 4.5 and is external to the Query Builder utilisation process.

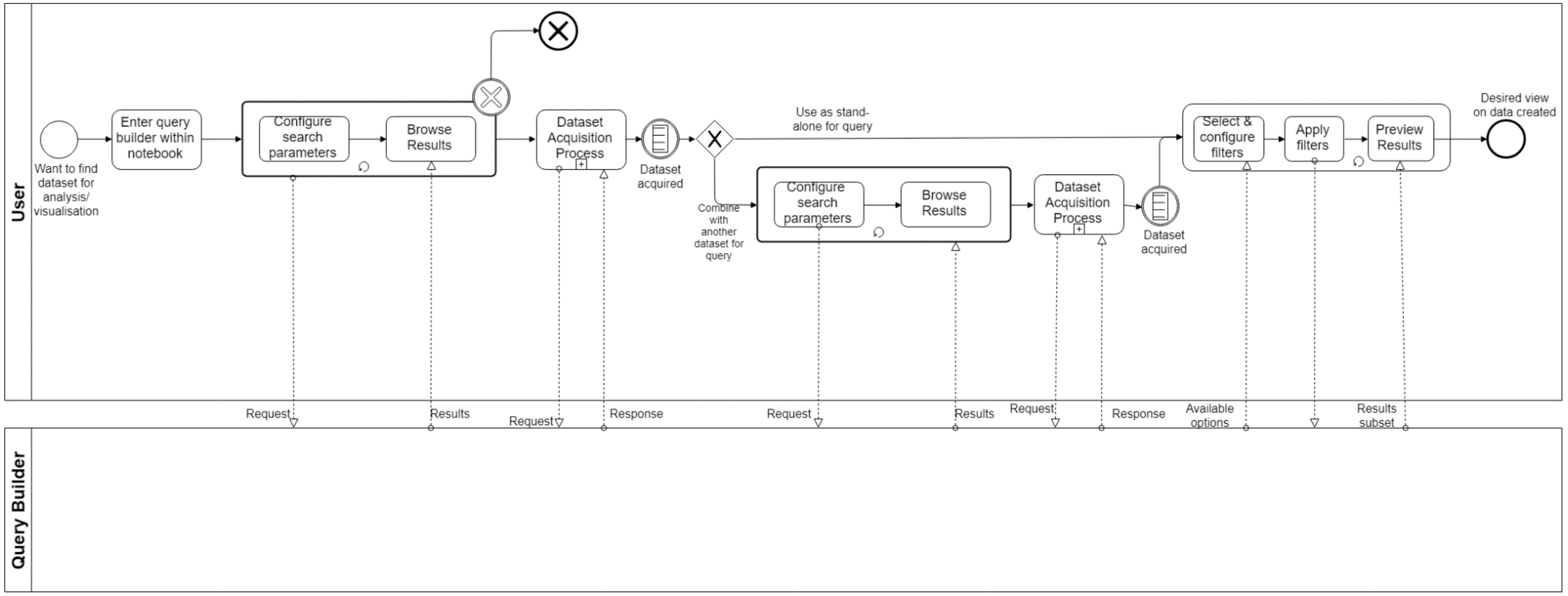

User
Query Builder
Want to find dataset for analysis/ visualisation
Enter query builder within notebook
Configure search parameters
Browse Results
Dataset Acquisition Process
Dataset acquired
Use as stand-alone for query
Combine with another dataset for query
Configure search parameters
Browse Results
Dataset Acquisition Process
Dataset acquired
Select & configure filters
Apply filters
Preview Results
Desired view on data created
Request
Results
Request
Response
Request
Results
Request
Response
Available options
Results subset

Figure 4-6: Dataset exploration through query builder workflow

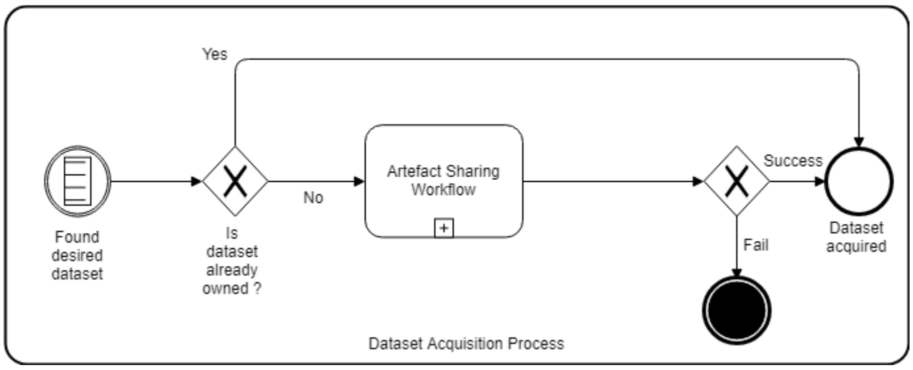

Yes
Found desired dataset
Is dataset already owned ?
No
Artefact Sharing Workflow
Success
Fail
Dataset acquired
Dataset Acquisition Process

Figure 4-7: Data acquisition sub-process workflow

4.4. Data export from AEGIS

The figure below presents the different ways for the user to export their data from the AEGIS platform. The user can download his/her files or share the whole dataset with other users within AEGIS.

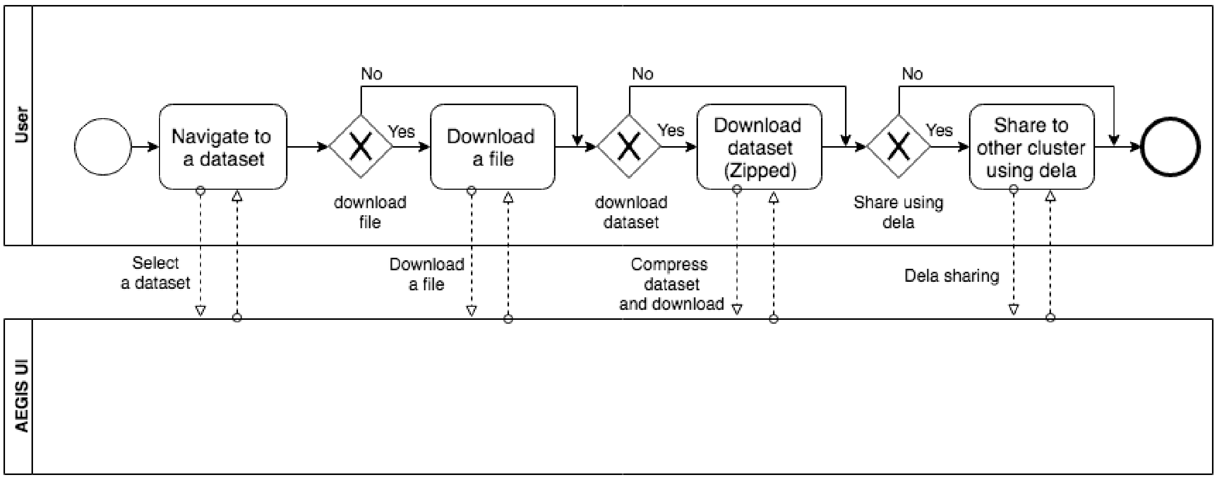


Figure 4-8: Data export workflow

4.5. Artefact sharing/reuse

The following figure shows the workflow for data sharing over the AEGIS platform. The operation to be performed involves both the core AEGIS platform as well as the Brokerage Engine, which will check if artefact sharing/reuse can be performed. At first level, the AEGIS platform checks whether the operation at high level is permitted (e.g. if the data asset exists, if the user has the right credentials to view the data artefact, if the user is logged in, etc.). If the access is possible and is permitted, then the Brokerage Engine is invoked. The Brokerage Engine checks the ledger to resolve the following situations:

- Identify whether the user requesting the data is capable of receiving it (e.g. if he/she has enough “credits” in case the data asset is not free), and
- Verify the availability of the data asset, comparing previous records in the ledger with the DPF elements that are attached to the data asset. This is essential only in case a data asset is provided with exclusivity rights (either permanently or within a specific timeframe), so that there is a check that no exclusivity rights have been transferred at the moment.

In case the above-mentioned check resolve that the data asset can be shared, then the transaction is inserted to the ledger, and the AEGIS platform is notified to release the data asset to the user.

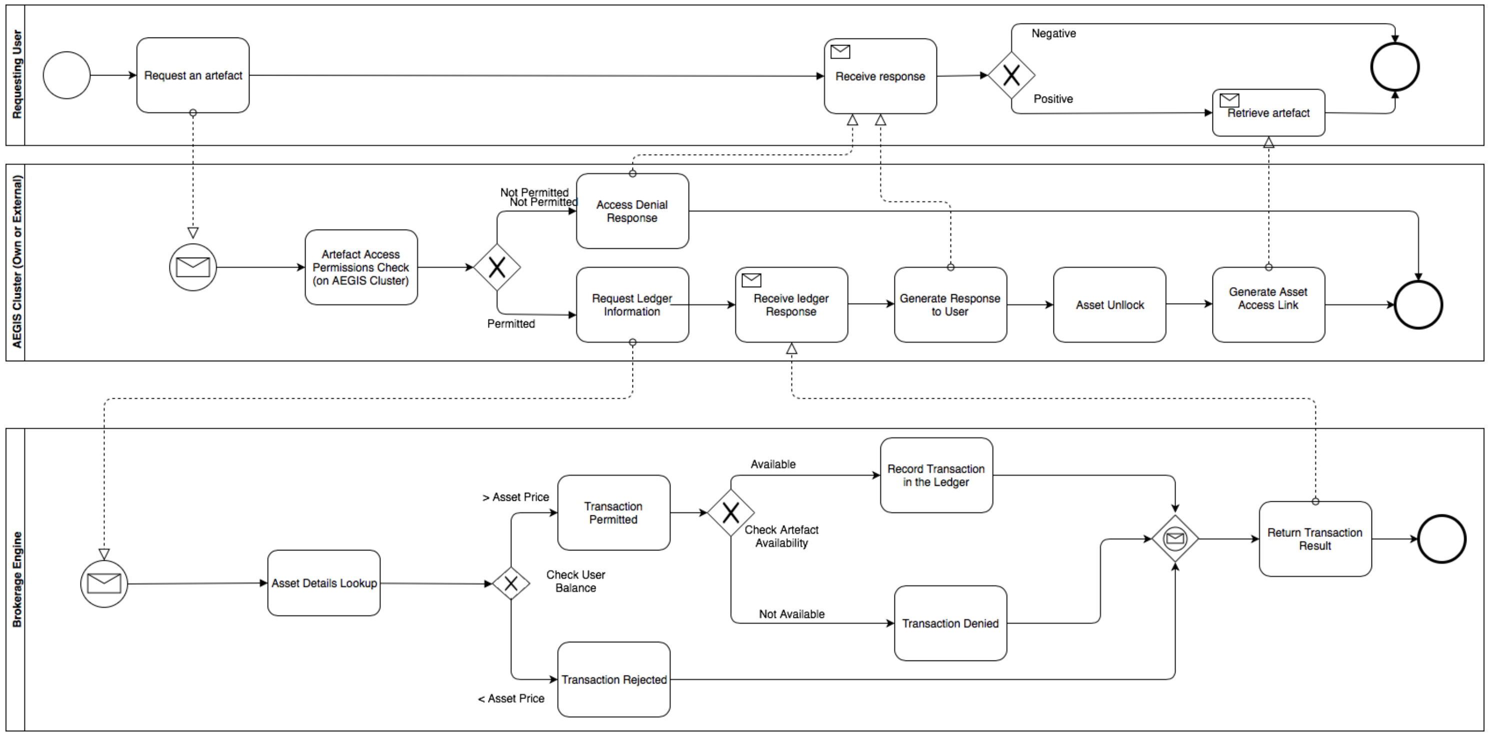


Figure 4-9: Artefact Sharing Workflow

4.6. Service creation

The following figure presents the workflow of the user's perspective when he/she will use the AEGIS services in order to either perform an analysis or generate a visualisation on an available dataset. In particular, the user interacts with the Algorithm Execution Container and the Visualiser components. The user is offered the option to request visualisation from the list of available visualisations in the Visualiser or a custom visualisation on an available dataset or on a dataset created as a result of the Query Builder execution. Additionally, the user is offered the option to perform a new analysis, multi-level analysis by chainable execution of algorithms, and request visualisation on the analysis results.

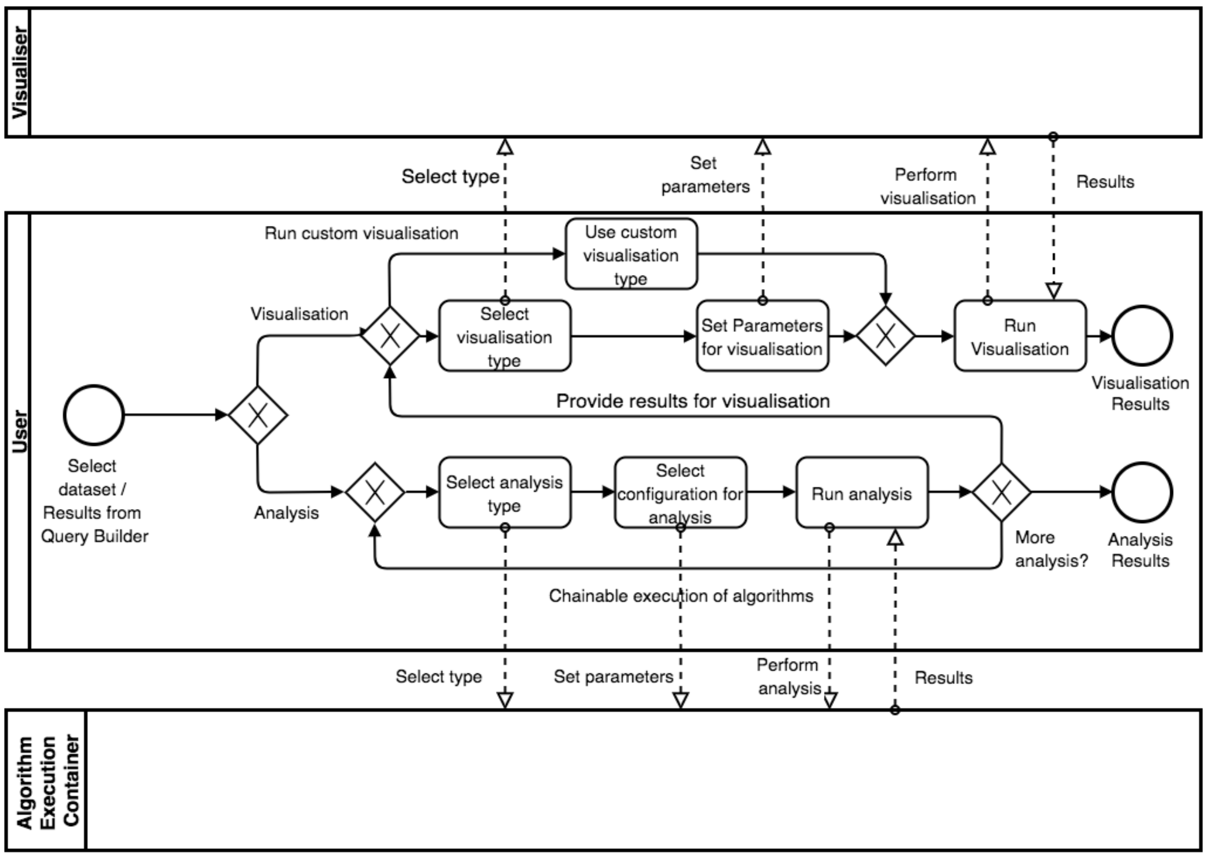


Figure 4-10: Service creation workflow

4.7. Service consumption

The following figure shows a workflow of the user's perspective when he/she will use the AEGIS platform to perform a general service consumption, which in this specific case includes the account creation/authentication, the search functionalities related to popular assets, latest assets and offers, the selection of a Dataset (together with its association with a Project) and the application to AEGIS main functionalities (Query Builder, Analytics, Visualiser).

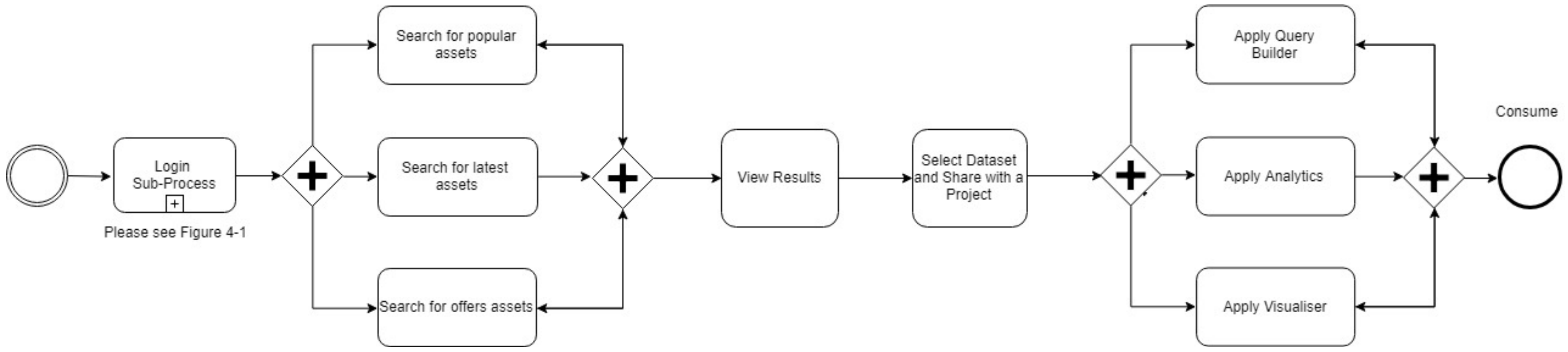


Figure 4-11: AEGIS Service consumption workflow

5. CONCLUSION

The current deliverable documented the efforts undertaken within the context of the tasks 3.1, 3.2, 3.3, 3.4 and 3.5 of WP3. The scope of this deliverable was to deliver a complementary documentation updating the information documented in deliverable D3.2 for the high-level and technical architecture of the platform, the platform's workflows as well as the components that compose the platform. Within the context of this deliverable, the necessary updates and adjustments on the platform's components and the platform's workflows are documented. These updates and adjustments are the results of a comprehensive analysis performed on the first evaluation and feedback received from the project's demonstrators on the first low fidelity, functional mock-up version of the AEGIS platform that was delivered in M14.

More specifically, the high-level architecture presented in deliverable D3.2 is further elaborated, focusing on the updated components and their positioning within the platform in regards to the functionalities undertaken by each component but also to the technologies and tools used in order to provide the aforementioned functionalities. Additionally, the technical architecture of the platform is presented providing deeper insight into the functional decomposition of components and the relationship between them along with the corresponding data flow.

In deliverable D3.2, the components of the AEGIS platform were defined and designed. Each component received several enhancements and refinements after the release of the first version of the platform. Following the approach used in the previous version, for each component the corresponding updated functionality was defined along with the corresponding information that will drive their updated implementation. For each component the list of the designed microservices and their functionalities are also documented. Moreover, the communication mechanisms between the components were refined based on the updated interfaces and the exposed outcomes offered by each component that facilitate the interactions of the components as well as the realisation of the workflows of the platform. In addition to the high-level architecture and the comprehensive description of the components of the architecture, the current document is presenting the updated AEGIS platform's workflows in the form of BPMN diagrams that facilitate the data-driven innovation in the PSPS domains as documented in D3.2 containing the necessary adaptations and modifications.

In the next steps, the outcomes of this deliverable will drive the implementation activities of the project. As the project evolves, the AEGIS platform and the components' design and specification will receive further adjustments and modifications as additional requirements will be received that will introduce new functionalities in the platform. Furthermore, the new version of the platform will be released and additional feedback from the project's demonstrators will be received that will result in further refinements that will be documented in D3.4.

APPENDIX A: LITERATURE

[1] S. Niazi, S. Haridi and J. Dowling, “Size Matters: Improving the Performance of Small Files in HDFS,” in *EuroSys*, 2017.