



HORIZON 2020 - ICT-14-2016-1

AEGIS

Advanced Big Data Value Chains for Public Safety and Personal Security

WP4 – AEGIS Infrastructure Implementation and Rollout



D4.3 - AEGIS Platform - Release 3.00

Version 1.0

Due date: 31.12.2018

Delivery Date: 21.12.2018

Author(s): Maurizio Megliola, Alessandro Calcagno (GFT), Dimitrios Miltiadou, Konstantinos Perakis (UBITECH), Evmorfia Biliri, Spiros Mouzakis, Christos Botsikas (NTUA), Fritz Meiners, Fabian Kirstein (Fraunhofer), Marios Phinikettos, George Bikas (SUITE5), Alexandru Ormenisan (KTH)

Editor: Maurizio Megliola (GFT)

Lead Beneficiary of Deliverable: GFT

Dissemination level: PU **Nature of the Deliverable:** Demonstrator

Internal Reviewers: Yury Glikman (Fraunhofer), Sotiris Koussouris (SUITE5)

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

Disclaimer: AEGIS is a project co-funded by the European Commission under the Horizon 2020 Programme (H2020-ICT-2016) under Grant Agreement No. 732189 and is contributing to the BDV-PPP of the European Commission.

The information and views set out in this publication are those of the author(s) and do not necessarily reflect the official opinion of the European Communities. Neither the European Union institutions and bodies nor any person acting on their behalf may be held responsible for the use which may be made of the information contained therein.

© Copyright in this document remains vested with the AEGIS Partners

EXECUTIVE SUMMARY

This deliverable, as part of WP4 which is responsible for the implementation of the AEGIS platform, consists of the third and pre-final version of the software package of the AEGIS core platform and its APIs with corresponding supporting documentation on the deployment of each component and usage of the APIs.

This accompanying document describes the current status of implementation of the different AEGIS components of this third prototype release.

Table of Contents

EXPLANATIONS FOR FRONTPAGE.....	2
AEGIS KEY FACTS	3
AEGIS PARTNERS.....	3
EXECUTIVE SUMMARY.....	4
LIST OF FIGURES	5
ABBREVIATIONS	8
1. INTRODUCTION.....	9
1.1. INSIGHTS FROM OTHER TASKS AND DELIVERABLES	9
1.2. STRUCTURE.....	9
2. AEGIS PLATFORM	10
2.1. URL	10
2.2. DATA HARVESTER AND ANNOTATOR.....	11
2.3. CLEANSING TOOL.....	15
2.4. ANONYMISATION TOOL.....	18
2.5. BROKERAGE ENGINE.....	22
2.6. AEGIS INTEGRATED SERVICES.....	24
2.7. METADATA SERVICE	25
2.8. QUERY BUILDER	28
2.9. VISUALISER.....	31
2.10. ALGORITHM EXECUTION CONTAINER	36
2.11. AEGIS FRONT-END.....	42
3. CONCLUSION.....	47

LIST OF FIGURES

Figure 1: AEGIS platform architecture.....	10
Figure 2: Mainpage of the Harvester Frontend	13
Figure 3: Sub Menu for a Data File	14
Figure 4: Data Annotator for CSV Files	14
Figure 5: Data Annotator for Data Sets.....	15
Figure 6: Offline cleansing tool rules editing	16
Figure 7: Offline cleansing tool upload form for CSV files	17
Figure 8: Offline cleansing tool results overview	17
Figure 9: Anonymiser configuration screen.....	18

Figure 10: Anonymiser data selection.....	19
Figure 11: Anonymiser indicative functions.....	19
Figure 12: Anonymiser custom user-defined function.....	20
Figure 13: Anonymiser output	20
Figure 14: Anonymiser exposed API.....	21
Figure 15: Anonymiser API response	22
Figure 16: List of User's Recent Transactions.....	23
Figure 17: List of User's Full Transactions	23
Figure 18: Transaction Details	24
Figure 19: SPARQL User Interface of Fuseki	26
Figure 20: AEGIS Linked Data Example	27
Figure 21: Extract of the AEGIS Ontology documentation.....	27
Figure 22: Simple JSON Representation of an AEGIS Dataset	28
Figure 23: Query Builder Temp dataset preview	29
Figure 24: Query Builder data filters view	30
Figure 25: Query Builder data filters view	30
Figure 26: Visualiser Heatmaps on Maps	32
Figure 27: Visualiser Maps with markers with custom labels and colours.....	32
Figure 28: Visualiser - dataset selection	33
Figure 29: Visualiser - visualisation type selection	34
Figure 30: Visualiser - visualisation parameters.....	34
Figure 31: Visualiser – scatter plot	35
Figure 32: Visualiser - bubble chart.....	35
Figure 33: Visualiser - time series	36
Figure 34: Visualiser – map	36
Figure 35: AEC Input File Tab	37

Figure 36: AEC Data Preview.....	38
Figure 37: AEC Algorithm Configuration Tab	39
Figure 38: AEC Output File tab	39
Figure 39: AEC Overview Tab	40
Figure 40: AEC Overview Tab with Results	41
Figure 41: AEGIS web site	43
Figure 42: Home page	44
Figure 43: Query Builder User Guide	44
Figure 44: Project Datasets	45
Figure 45: Splash page mockup	45
Figure 46: AEGIS Marketplace mockup.....	46

ABBREVIATIONS

CO	Confidential, only for members of the Consortium (including the Commission Services)
D	Deliverable
DoW	Description of Work
H2020	Horizon 2020 Programme
FLOSS	Free/Libre Open Source Software
GUI	Graphical User Interface
IPR	Intellectual Property Rights
MGT	Management
MS	Milestone
OS	Open Source
OSS	Open Source Software
O	Other
P	Prototype
PU	Public
PM	Person Month
R	Report
RTD	Research and Development
WP	Work Package
Y2	Year 2

1. INTRODUCTION

The present deliverable is released within the context of Workpackage 4 “AEGIS Infrastructure Implementation and Rollout” and is in particular associated with Task 4.4 “Continuous Integration and Testing Activities”. Within this task, the third release of the AEGIS platform has been developed.

1.1. Insights from other tasks and deliverables

This deliverable is strictly related to the work done in WP3, more precisely to the work reported in D3.4, regarding two main features: from the one hand, the detailed information of the AEGIS components, comprising an overview of its functionalities and positioning in the overall architecture, the technologies used for its implementation and the API (when available) that it exposes in order for other components to interact with. On the other hand, the BPMN diagrams that correspond to the main tasks a user will perform through the AEGIS, as seen from the user perspective, but also outlining component interactions, which provided the basis for the development of the AEGIS prototype. Another important source is the work done in WP2, as reported in D2.1/D2.2, especially the information about the features that must be offered through certain components, clarifying how important parts of the data value chain should be supported in practice, e.g. data policies, data harmonisation, metadata handling, knowledge extraction, and visualisation. Finally, the user perspective when utilising AEGIS, depicted by the usage scenarios reported in D1.2, has also driven the design and development of the AEGIS platform functionalities.

1.2. Structure

Section 2 of this document reports, for each foreseen platform component, the status within the prototype or -in case it is not yet integrated in the prototype- the component concept and application screenshots.

Section 3 concludes the deliverable and illustrates the next steps in the work package.

2. AEGIS PLATFORM

In this chapter, we provide the references to the AEGIS project software source code as well as the related documentation about the component deployment and API usage.

Next, for each platform components, its status within the integrated prototype is described, in terms of its functionalities and how it works. If not yet integrated in the prototype, the component concept is reported, together with a set of mockup/application pictures.

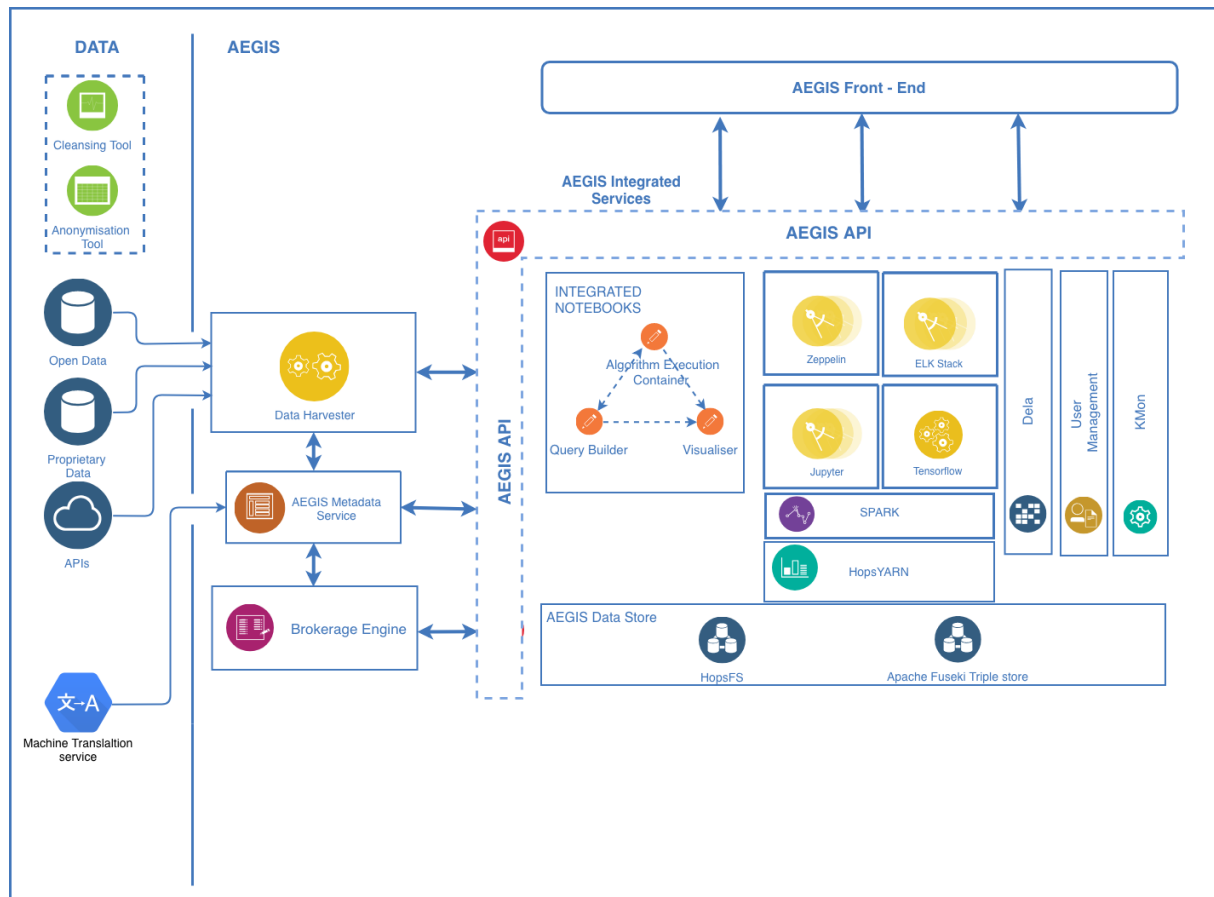


Figure 1: AEGIS platform architecture

2.1. URL

The AEGIS integrated prototype is available at the following URL:

<http://bbc6.sics.se:8080/hopsworks/#!/login>¹

Some services are not yet integrated into the prototype, but deployed separately. The appropriate URLs are indicated within the respective chapters.

¹ User: test@aegisbigdata.com; Password: Test01

The software core packages are available in AEGIS code repository at Github.com, at the following link:

<https://github.com/aegisbigdata>

The documentation related to the deployment of each component and usage of the APIs is provided on the project Github wiki, at the following link:

<https://github.com/aegisbigdata/documentation/wiki>

2.2. Data Harvester and Annotator

The following sections describe the status and implementation details of the Data Harvester and Data Annotator components in the third release of the prototype.

Data Harvester Microservices

The Data Harvester constitutes a microservice architecture, consisting out of the following four basic services:

- **Importer** - implements all functionality for retrieving data from a specific data source
- **Transformer** - converts the retrieved data from an importer into the target format of the AEGIS platform
- **Aggregator** - collects converted data from a transformer over a configurable time interval
- **Exporter** - uploads transformed and/or aggregated data to the AEGIS platform

These services represent the fundamental harvesting pipeline. For the third release of the AEGIS platform the following instances of the services have been implemented and orchestrated:

Importer

Two basic importers have been implemented. The first one enables the import of data from RESTful JSON-APIs and the second one allows the provision of CSV data.

Transformer

A basic transformer is available, allowing to provide scripts (JavaScript) for creating custom transformations from source to target. In addition, a basic CSV-to-CSV transformer was implemented.

Aggregator

One aggregator for accumulating CSV data over a specified time interval was developed.

Exporter

One exporter for uploading CSV files to the AEGIS platform was implemented.

To name a concrete example: Weather data may be imported hourly from an external service as JSON, which is then transformed into CSV with values being converted into the metric

system. The results are then aggregated for 24 hours and the final file exported to an analytics platform.

In order for this “chain” of microservices to work as intended, each service needs to send the result of its computation to the next service. This is handled by an overarching concept, called a *pipe*. 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 and provides applicable configurations. This makes each service agnostic of its surroundings, aiding in the generic design discussed earlier.

Another advantage of the described architecture is the easy scaling of each component in times of high load. For example, should the transformer pose to be a bottleneck when computing large amounts of data, another instance can simply be spawned. Once registered at the pipe framework, future transformation requests can be sent to the newly launched transformer instance. This ensures maximum availability and throughput even in big data scenarios like AEGIS. In addition, this architecture allows the simple extension of the data harvester by developing new instances for the respective services.

Also, the reuse in different contexts is encouraged by keeping the implementation of each component as generic as possible. Given a suitable script, the transformer could also be used to anonymize sensitive data. Tests have already been conducted on the new software infrastructure, showing promising results.

Data Harvester Technology

In order to reuse as much as code as possible, the microservices introduced in the previous section have also been written in the Java programming language. However, this does not mean that future services must also be written using the same language, as each service is designed to be an independent piece of software. To aid with the creation of each service the framework Eclipse Vert.x² has been chosen. Apart from propagating an asynchronous programming paradigm ensuring a high availability and throughput, Eclipse Vert.x also provides a lot of built in support for managing microservices. Concepts that come to use are circuit breakers, health checks, and performance metrics.

The final services are deployed via containerization, in which each service is isolated from its host and other service’s operating systems. The services are thereby packaged separately as *images*. The technology used for building these images is called Docker³. Aside from enhancing security this method also ensures platform independency, which means that these images (and consequently the services in question) will run on any infrastructure providing a Docker runtime. The concept of containerization also greatly helps with the dynamic launching of service instances mentioned previously.

² <https://vertx.io/>

³ <https://www.docker.com/>

To handle the frictionless interaction between various microservices their APIs have been defined using the OpenAPI 3⁴ specification, which allows the precise definition of RESTful interfaces in a commonly understood format. The previously mentioned Eclipse Vert.x framework supports loading an OpenAPI document and exposing the endpoints defined, including request validation.

Data Harvester Frontend

The new approach for implementing the Data Harvester requires a specialised frontend, which allows the orchestration, configuration and execution of a specific harvesting process (pipe). The first basic version of this frontend was developed for the third release of the AEGIS platform. Figure 2 shows the main page of the application.

Figure 2: Mainpage of the Harvester Frontend

Data Annotator

The Data Annotator (Metadata) was integrated into the AEGIS platform core frontend, by extending the AngularJS frontend. It allows the simple provision of detailed and semantic valuable metadata for projects, data sets and files in the AEGIS platform. Figure 3 shows the access to the Data Annotator in the context menu of an asset in the AEGIS platform. Figure 4 and Figure 5 show the actual annotation interfaces.

⁴ <https://github.com/OAI/OpenAPI-Specification>

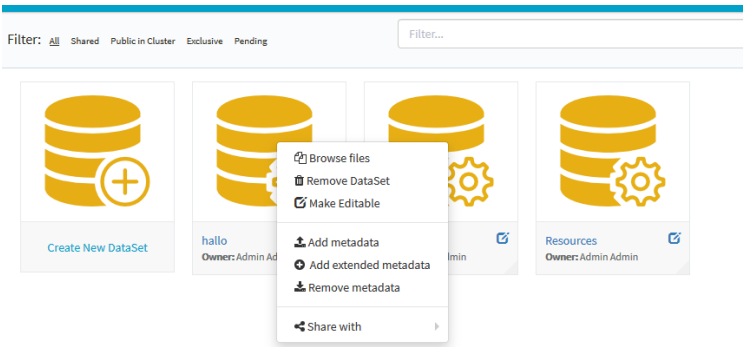


Figure 3: Sub Menu for a Data File

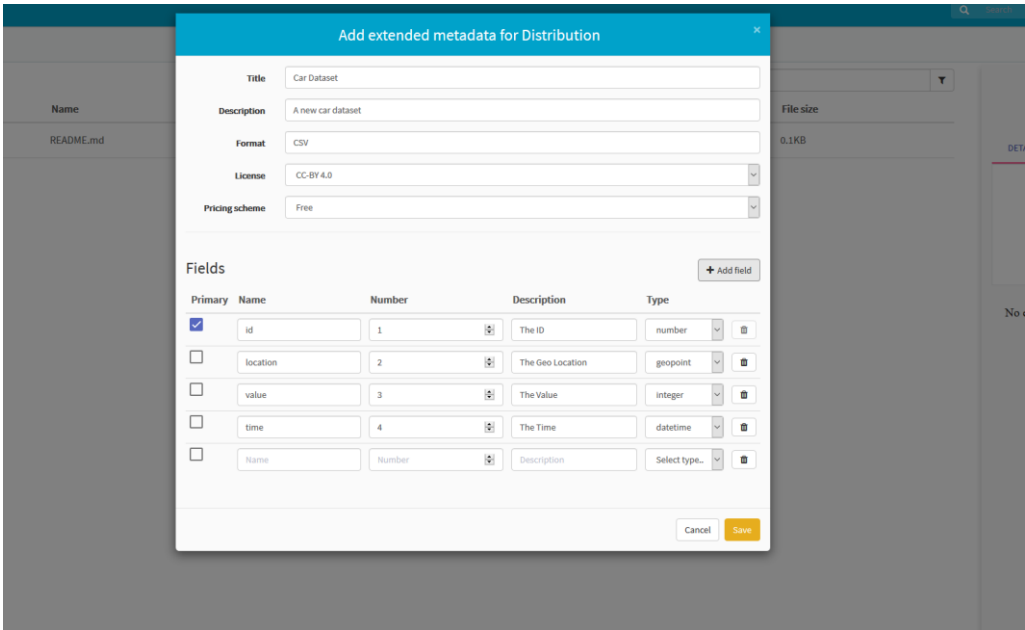


Figure 4: Data Annotator for CSV Files

Figure 5: Data Annotator for Data Sets

In the following releases the Data Annotator will be extended to support more detailed metadata.

2.3. Cleansing Tool

Within the context of the AEGIS platform, the data cleansing tasks, as it have been documented also in the deliverables of WP3, are implemented following a two-fold approach: (a) an offline cleansing tool, residing where the data are located, that is offering a variety of cleansing processes and functionalities covering all the crucial and required data cleansing aspects with regard to data validation, data cleansing and data completion that can applied to any datasets prior to importing them in the AEGIS platform and (b) an online cleansing tool suitable for data cleansing and manipulation tasks that are executed during the data query creation process, capable of addressing certain simple cleansing tasks that are time computationally intense leveraging the computational power of the AEGIS platform.

With regard to the online cleansing tool, the described functionalities are the offered functionalities that are incorporated inside the Query Builder component and will be described in the corresponding section (See 2.8).

The AEGIS offline cleansing tool is designed and implemented with the aim of facilitating the users to the accomplish the required cleansings tasks. For this reason, the tool is easily customisable and adaptable to the user's needs taking into consideration the variety of the heterogeneous sources that will be imported in the AEGIS platform. Following the design that is documented in the deliverables of WP3, the offline cleansing tool is implemented as a

standalone application written in Python and following the microservices architecture using Flask microframework⁵ and a set of libraries such as Pandas⁶ and NumPy⁷.

In the current version of the offline cleansing tool, a list rules for data validation, data cleansing and missing data handling are offered to the user and can be set according the user's needs and the nature of the data. The list of supported rules was expanded from the previous version in order to accommodate additional cases as per the feedback received from the AEGIS stakeholders. The tool is offering an intuitive user interface where the user is able to review the results of the execution of the cleansing tasks. Several refinements were also introduced in the user interface towards the aim of offering a better user experience to the users.

The offline cleansing tool is also offering a REST-API with the appropriate endpoints facilitating the uploading of the dataset that will be used in the cleansing process and the execution of the cleansing tasks. The REST-API documentation is now available using the Swagger framework. In addition to this, a new user interface with an upload form suitable from the cleansing of CSV files was also introduced. Furthermore, a series of optimisations were introduced in the tool in order to achieve significant performance improvement for the handling of large datasets. Finally, the tool is now available also as a Docker image, facilitating the easy installation on the premises of the user.

In the upcoming versions of the tool, the focus will be on providing several enhancements and updates in terms of cleansing functionalities, rules definition and performance.

The following figures illustrate some indicative examples from the execution of the offline cleansing tool.

The image shows a web-based interface titled "Missing Value Rules Editor". It contains three input fields: "Variable:" with the value "unit", "Rule:" with a dropdown menu showing "FILL_WITH_VALUE", and "Arguments:" with the value "mmHg". At the bottom right, there are two buttons: "Cancel" and "Save Changes".

Figure 6: Offline cleansing tool rules editing

⁵ <http://flask.pocoo.org/>

⁶ <https://pandas.pydata.org/>

⁷ <http://www.numpy.org/>

AEGIS Data Cleaner | Cleaning Page [Home](#) [Datasets](#) [Rules](#) [Logs](#) [Clean](#) [Sign out](#)

Select Provider

Select Dataset

Select file to clean (CSV or Excel)
 No file chosen

Figure 7: Offline cleansing tool upload form for CSV files

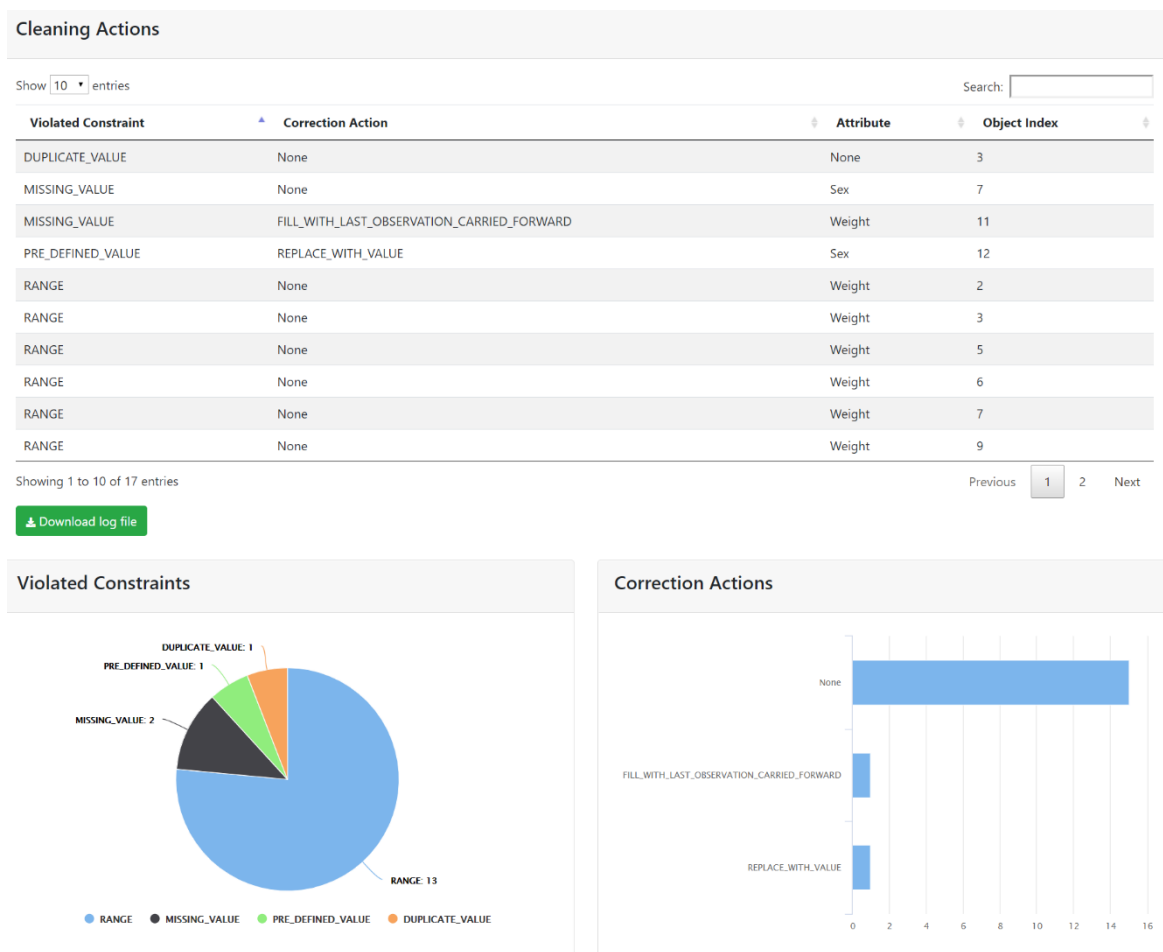


Figure 8: Offline cleansing tool results overview

2.4. Anonymisation Tool

The anonymisation tool is an extensible, schema-agnostic plugin that allows real-time efficient data anonymisation. The anonymisation tool has been utilized for offline, private usage but offers the ability to output the anonymized data through a secured, web API. The anonymization either syncs with private database servers or imports files from the filesystem and executes anonymisation functions on datasets of various sizes with little or no overhead. The purpose of the anonymisation is to allow the exploitation of the raw data in the system by accounting for privacy concerns and legal limitations.

For the third release, risk assessment for anonymization mappings were introduced and several bugs were fixed.

The first screen of the tool allows you to setup or edit an existing, saved configuration. Each configuration is basically a separate anonymization project to various database back-ends or text files, with different executed anonymization functions. By creating a new configuration the system will prompt the user to connect to the private database backend or select the file to open and select the entities / tables to anonymise.

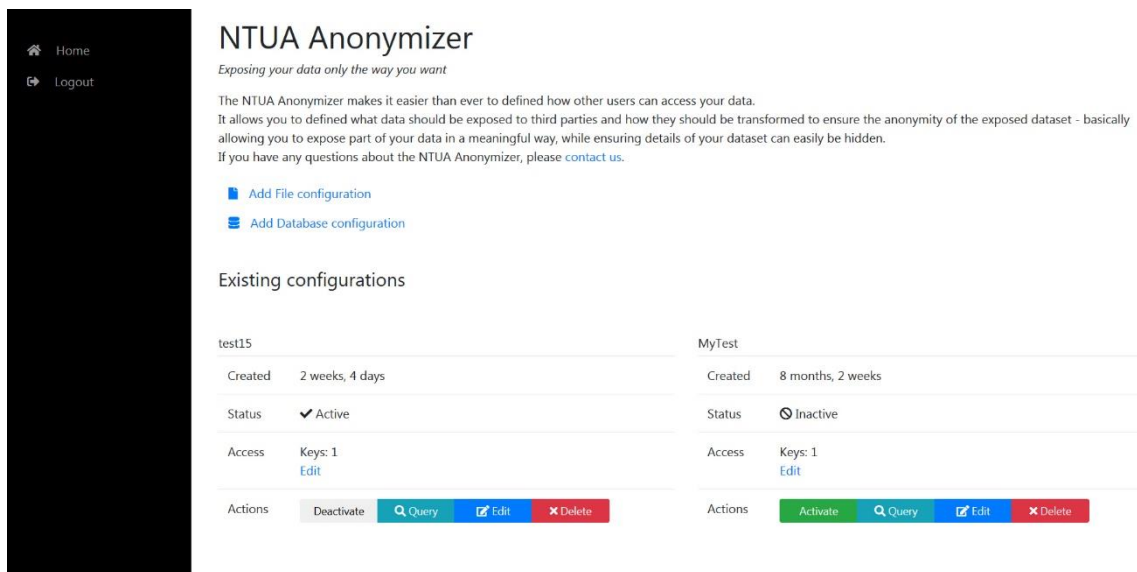


Figure 9: Anonymiser configuration screen

The system then prompts the user to select which fields from the data source to expose to the anonymised set, as well as the anonymisation function to be performed.

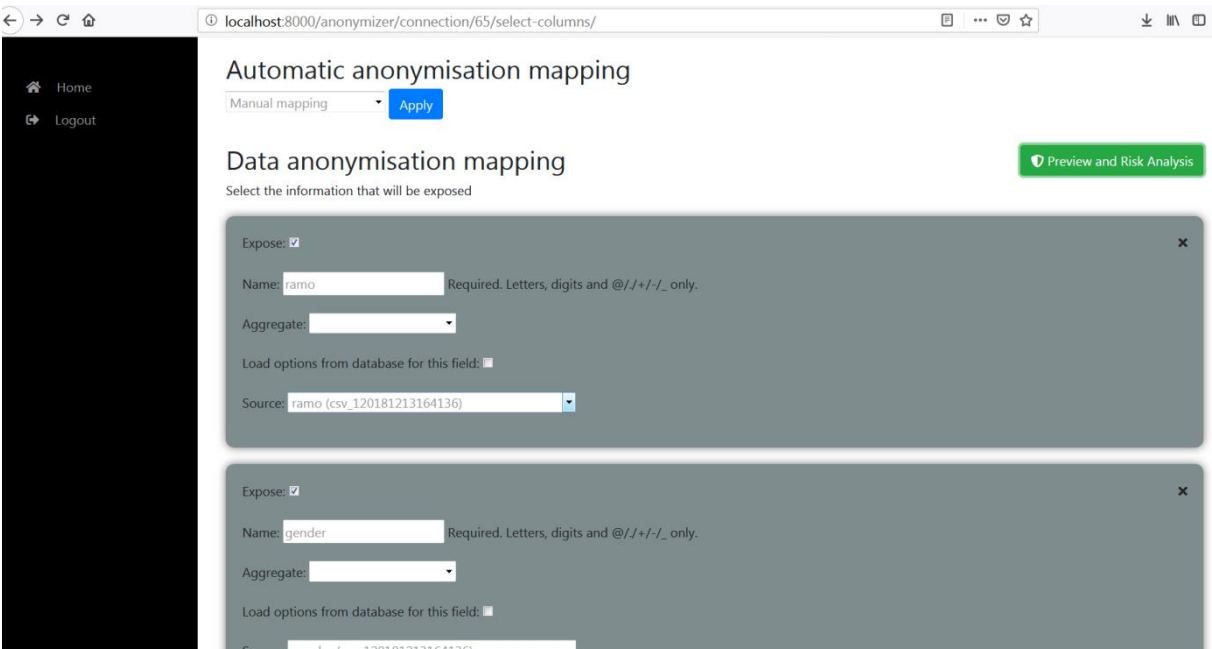


Figure 10: Anonymiser data selection

The anonymisation system comes with a list of predefined anonymisation functions that can be used directly (e.g. city from an exact address, range of values from an integer), as well as a list of aggregation functions (e.g. average).

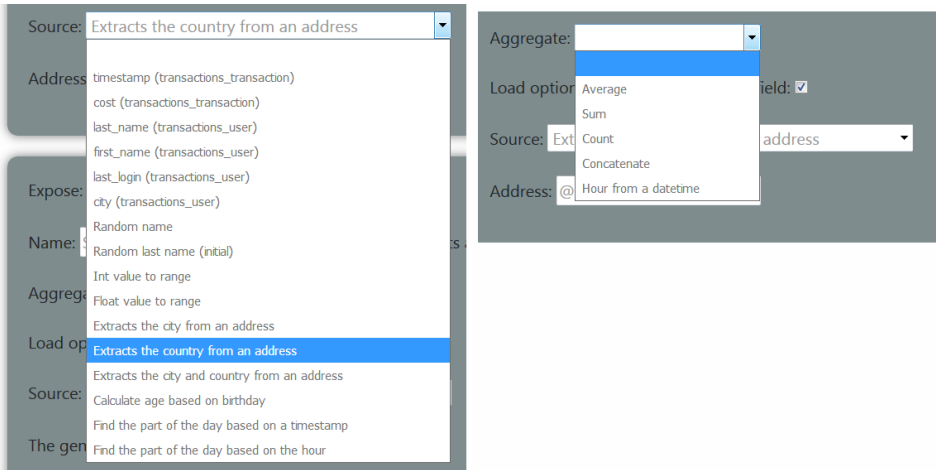


Figure 11: Anonymiser indicative functions

The tool can be easily extended with any new, custom anonymisation functions defined by the user in a Python module.

```
def address_to_city_helper(address, info=None):
    if not info:
        info = get_address_info(address)

    if not info['results']:
        return '', info

    city_name = ''
    for component in info['results'][0]['address_components']:
        if 'administrative_area_level_3' in component['types']:
            city_name = component['long_name']
        elif 'administrative_area_level_5' in component['types']:
            city_name = component['long_name']

        if city_name:
            break

    return city_name, info
```

Figure 12: Anonymiser custom user-defined function

The user is able to execute queries and test the anonymised output through the integrated console of the tool as seen in the following screenshot.

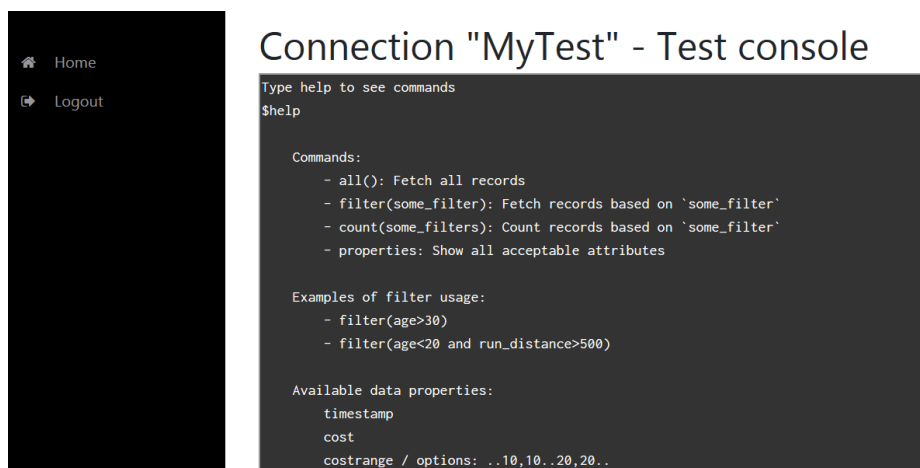
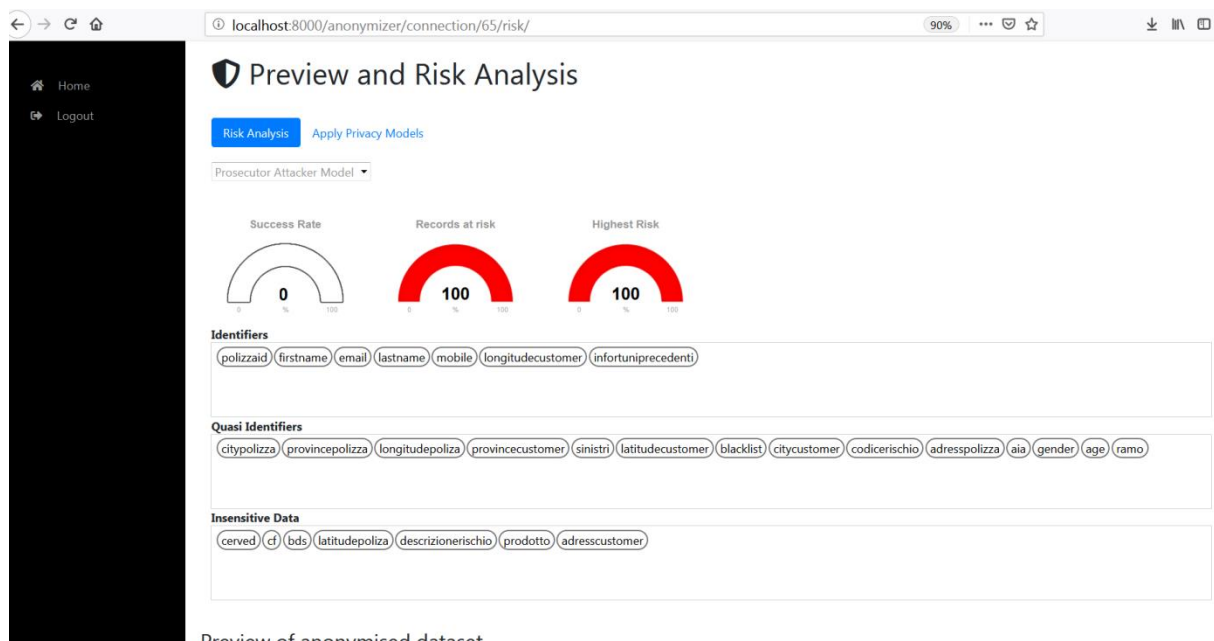


Figure 13: Anonymiser output

By clicking Preview and Risk analysis the user is able to assess the risk levels of his anonymization mapping as shown in the following interface:



The anonymised (output) data can be exposed through API to external parties in a secure way through the provision of access keys.

Access Keys

Here you can manage the available access keys for the **MyTest** connection.

External parties who want to access data must do this through the NTUA Anonymizer API with a valid access key.

Access keys can be revoked at any time.

You can entirely close this connection by deactivating it from the [Home](#) page.

[+ Create new key](#)

Name	Created	Key	Status	Last usage	Actions
AEGISkey2	Oct. 9, 2017, 6:22 p.m.	0f0af4fbcc68	✓ Active	Never	Revoke

Figure 14: Anonymiser exposed API

Users can access the anonymised data in JSON format through API provided by the anonymization tool using their private access keys.

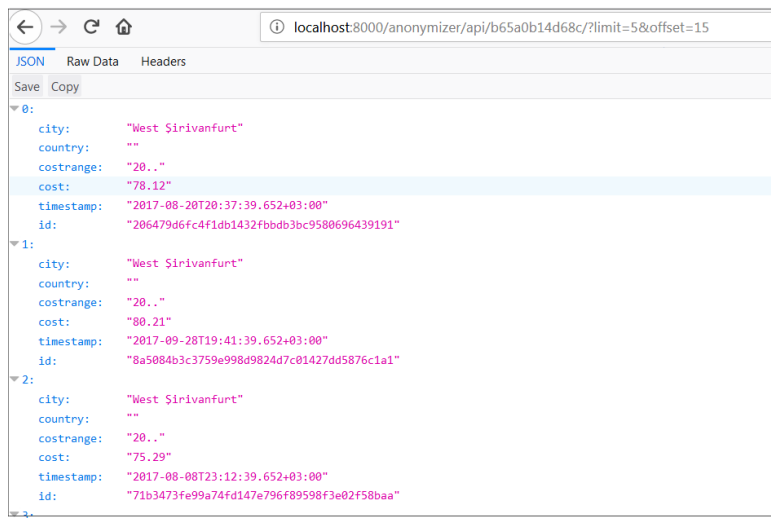


Figure 15: Anonymiser API response

2.5. Brokerage Engine

Following the implementation and testing of the AEGIS brokerage engine, and the finalisation of deliverable D2.3 which denotes the final version of the Data Brokerage Framework, a revision of the backend infrastructure of the Brokerage engine was necessary to be performed. This was required to match the changes (and simplifications) imposed in the DBF, which dictate to store less information on the distributed ledger, and shift non-critical information to the AEGIS platform, as metadata that accompany each data asset.

Moreover, the introduction of the “public dataset” feature of the AEGIS platform, allowed the concept of the distributed ledger network to be realised in a more natural and logical manner, and the backend system has been set up in a way that new nodes (e.g. AEGIS clusters) can join the network and become part of it, having access and read/write permissions on the ledgers.

The AEGIS Brokerage engine is at the moment being installed in the premises of GFT which will host the main (publicly demonstrable) AEGIS platform, while other nodes will be installed in the premises of other partners to test the infrastructure prior to the final release of the platform.

Apart from the back-end development, the re-design of the AEGIS portal front end brought to the surface the request to develop a simple information page that will list all the transactions that a user has conducted. For this purpose, mock-ups have been designed (Figure 16: List of User’s Recent Transactions) that allow a user to access the latest and the whole list of his transaction, through his profile page.

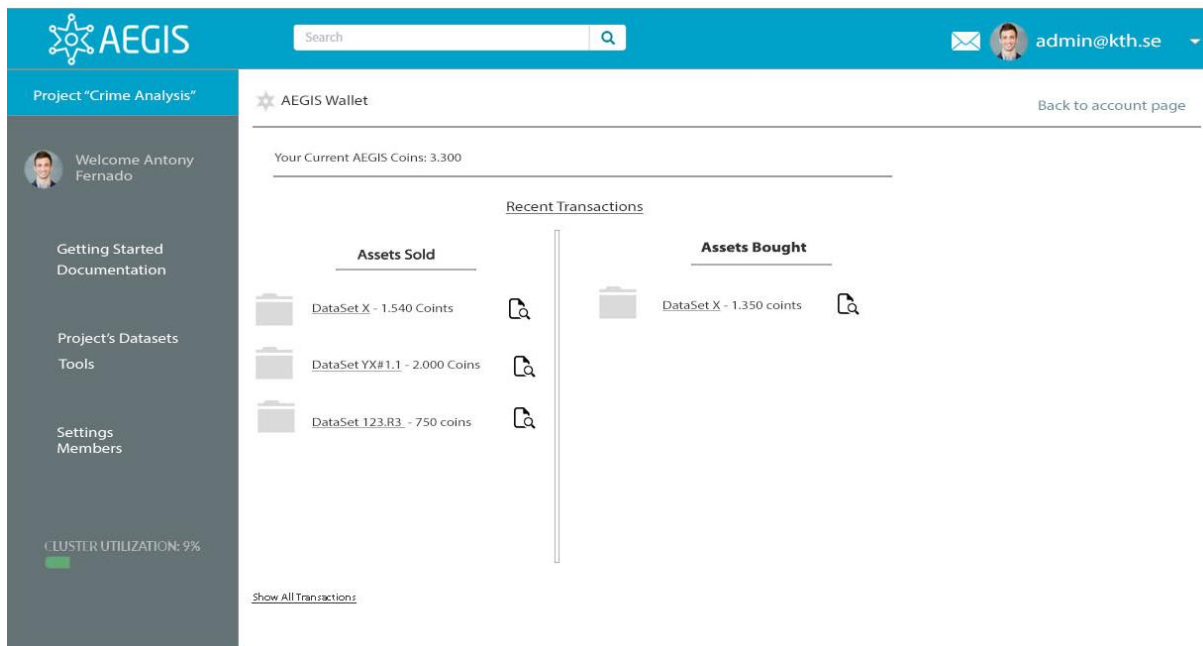


Figure 16: List of User's Recent Transactions

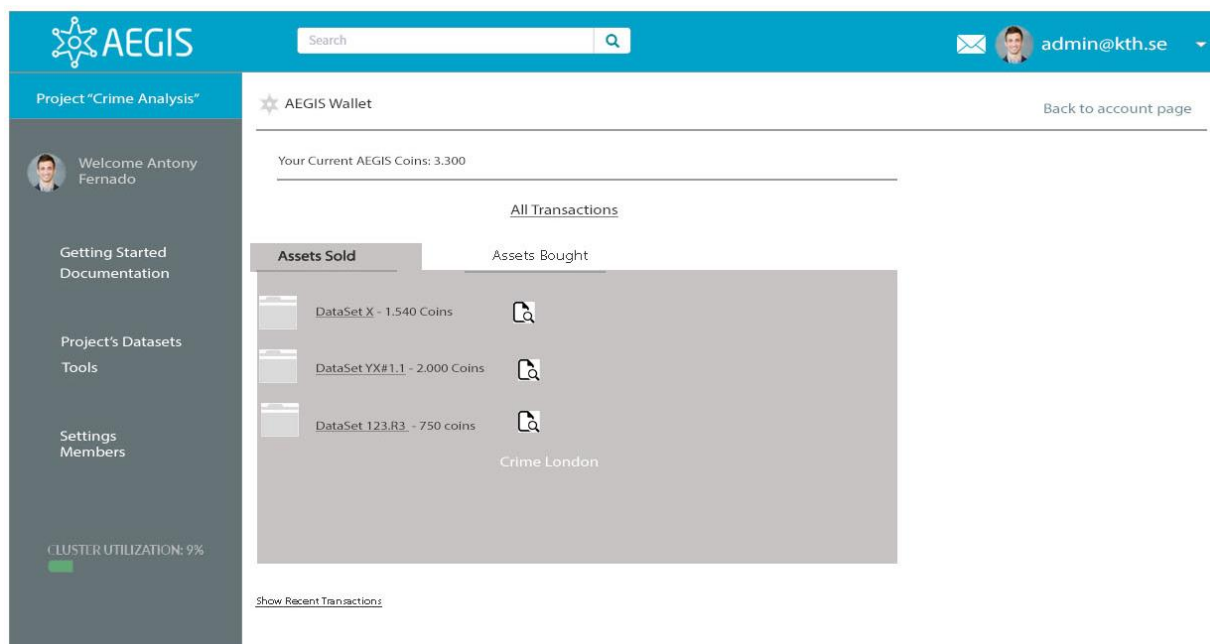


Figure 17: List of User's Full Transactions

Furthermore, an "Asset Transaction" specific page will be developed, that will retrieve (Figure 18: Transaction Details) that will retrieve from the Brokerage engine's API the information that follows a past transaction.

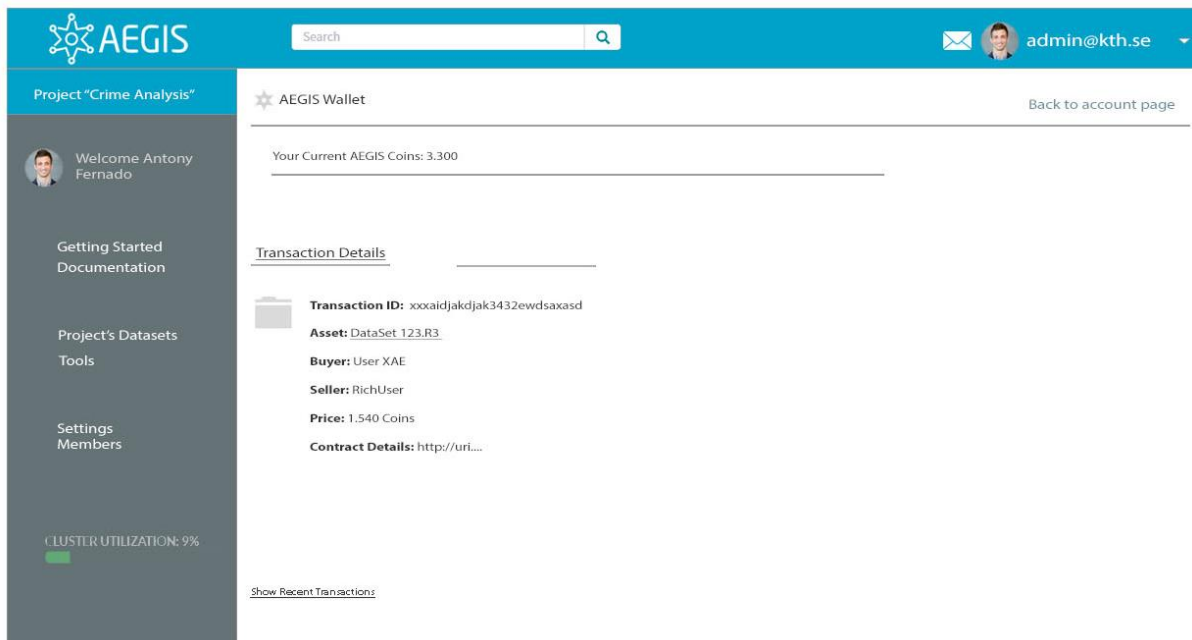


Figure 18: Transaction Details

2.6. AEGIS Integrated Services

The AEGIS platform provides data management and processing, user management, and service monitoring through the use of Hopsworks integrated services. Hopsworks introduces the notion of Project, Dataset, and User to enable multi-tenancy within the context of data management. Data processing includes data parallel processing services such as MapReduce, Spark, and Flink, as well as interactive analytics using notebooks such as Jupyter.

Full-text search capability is offered by the included ELK stack component (Elasticsearch). Real time analytics is enabled by the use of the included Kafka service. Deep learning is enabled by the use of the included Tensorflow service. TensorFlow is an open source software library released in 2015 by Google to make it easier to design, build, and train deep learning models. Although TensorFlow is only one of several options available, we choose to use it here because of its good design, ease of use and large community of adopters.

Full-text Search

Elasticsearch, one of the ELK stack components, is used to provide full text search capabilities to explore projects and datasets within the AEGIS platform. The search space available to each of the users depends on the context from which the user searches. For example, within the context of the home page, the search space includes public datasets, projects, and private datasets. When inside the project, the scope of the search is reduced to the project's datasets including the shared datasets from other projects. Thus, the search function as shown in Figure 1 includes all the projects, where a user is involved, as well as all the datasets included in the projects, shared or owned by the project, as well as public datasets

Tensor Flow

TensorFlow is an ecosystem for developing deep learning models, containing all the tools from building to deployment. TensorFlow has 3 main components:

1. TensorFlow(API) - contains the API's to define the models and train the models with the data.
2. TensorBoard - helps to analyse, visualize, and debug TensorFlow graphs.
3. TensorFlow Serving - helps to deploy the pre-trained models.

2.7. Metadata Service

The Metadata Service (AEGIS Linked Data Store in D4.1) is responsible for storing the metadata associated with a particular dataset within the AEGIS platform. This metadata poses the foundation of the processing of the data within the AEGIS platform. It is based on the AEGIS ontology and vocabulary. For the third release, only minor bug fixes were applied.

Triplestore

The foundation of the Metadata Service is the Apache Fuseki Triplestore. It can be directly accessed here:

<http://aegis-store.fokus.fraunhofer.de>

It offers multiple standardised Linked Data interfaces, like SPARQL or the Graph Store HTTP Protocol. These interfaces can be utilised from other components of the AEGIS platform, especially the Query Builder. Figure 19 shows the SPARQL interface of Fuseki.

Dataset: /ds

query upload files edit info

SPARQL query

To try out some SPARQL queries against the selected dataset, enter your query here.

EXAMPLE QUERIES

Selection of triples Selection of classes

PREFIXES

rdf rdfs owl xsd

SPARQL ENDPOINT: http://aegis-store.fokus.fraunhofer.de/ds/query

CONTENT TYPE (SELECT): JSON

CONTENT TYPE (GRAPH): Turtle

```

2
3 SELECT ?subject ?predicate ?object
4 WHERE {
5   GRAPH ?g {
6     ?subject ?predicate ?object
7   }
8 }
9 LIMIT 25

```

QUERY RESULTS

Table Raw Response

Showing 1 to 25 of 25 entries

Search: Show 50 entries

Figure 19: SPARQL User Interface of Fuseki

Fuseki only acts as a storage layer and is not supposed to be accessed directly by the users or any other component, with an exception to the SPARQL interface, which can be used for executing complex queries against the metadata of the AEGIS platform. Therefore, the AEGIS ontologies are publicly available: <http://aegis.fokus.fraunhofer.de/>. Figure 20 shows an example for the metadata stored in the triplestore. In Figure 21 an extract of the AEGIS ontology documentation can be seen.

```

<http://www.aegis-bigdata.eu/md/dataset/austria>
  a
  dct:description "Weather data for austria" ;
  dct:publisher [ foaf:homepage "http://www.fokus.fraunhofer.de" ;
                 foaf:name "Fraunhofer FOKUS"
               ] ;
  dct:title "austria" ;
  hops:datasetId "3754629" ;
  dcat:contactPoint [ vcard:hasEmail "Fabian.Kirstein@fokus.fraunhofer.de" ;
                     vcard:hasName "Fabian Kirstein"
                   ] ;
  dcat:distribution <http://www.aegis-bigdata.eu/md/dataset/austria/distribution/Weather-Data-for-Graz-19-05-2018> , <http://www.aegis-bigdata.eu/md/dataset/austria/distribution/Weather-Data-for-Graz-19-05-2018> ;
  dcat:keyword "weather" , "austria" , "temperature" , "humidity" ;
  dcat:theme <http://publications.europa.eu/resource/authority/data-theme/ENVI> .

<http://www.aegis-bigdata.eu/md/dataset/austria/distribution/Weather-Data-for-Graz-19-05-2018>
  dct:description "Weather Data for Graz" ;
  dct:format "CSV" ;
  dct:license "CC-BY" ;
  dct:title "Weather Data for Graz 19-05-2018" ;
  aegis:hasField [ aegis:description "The Humidity of the entry" ;
                  aegis:name "Humidity" ;
                  aegis:number "6" ;
                  aegis:type "number"
                ] ;
  aegis:hasField [ aegis:description "The Wind Direction of the entry" ;
                  aegis:name "Wind Direction" ;
                  aegis:number "11" ;
                  aegis:type "number"
                ] ;
  aegis:hasField [ aegis:description "The Avg. Temperature of the entry" ;
                  aegis:name "Avg. Temperature" ;
                  aegis:number "4" ;
                  aegis:type "number"
                ] ;
  aegis:hasField [ aegis:description "The Min. Temperature of the entry" ;
                  aegis:name "Min. Temperature" ;
                  aegis:number "7" ;
                  aegis:type "number"
                ] ;
  aegis:hasField [ aegis:description "The location of the entry (human-readable)" ;
                  aegis:name "Location" ;
                  aegis:number "1" ;
                  aegis:type "string"
                ] ;

```

Figure 20: AEGIS Linked Data Example

2. AEGIS Big Data Vocabulary: Overview

This ontology has the following classes and properties.

Classes

[Catalog](#) [Dataset](#) [Distribution](#) [Field](#) [FieldType](#) [TabularDistribution](#)

Object Properties

[hasField](#) [type](#)

Data Properties

[datasetId](#) [description](#) [field](#) [hasPrimaryKey](#) [name](#) [number](#) [projectId](#)

Named Individuals

[Any](#) [Array](#) [Boolean](#) [Date](#) [Datetime](#) [Duration](#) [Geosjon](#) [Geopoint](#) [Integer](#) [Number](#) [Object](#) [String](#) [Time](#) [WebAdresses](#) [Year](#) [Yearmonth](#)

Figure 21: Extract of the AEGIS Ontology documentation

Metadata Service

The Triplestore only offers (complex) Linked Data interfaces and no rich management functionalities. Therefore, an additional service is required, providing additional functionalities for the management of the metadata. This includes particularly the straight-forward creation of metadatasets. A first prototype is available here:

<http://aegis-metadata.fokus.fraunhofer.de>

It interacts with the Fuseki triplestore and offers a simple JSON-based REST-API for creating, deleting and updating metadata. It maps the JSON input to the Linked Data structures defined by the AEGIS ontology. Figure 22 illustrates such a simple JSON object, which can be posted to the service. For the second release, a basic recommendation service was implemented. It suggests suitable and similar datasets based on an input dataset. Therefore, several characteristics of the dataset are matched against the stored metadata, e.g. keywords or the

semantic tabular information. For future releases this feature will be extended and improved. The metadata service is developed in Java, based on the Play Framework.

```
{
  "id": "car_demo_trip1",
  "title": "Car Scenario Demo Trip 1",
  "description": "This dataset includes multiple data from one car trip",
  "keywords": [
    "cars",
    "safety"
  ],
  "distributions": [
    {
      "accessURL": "hdfs:///Projects/VIF/DemoTrip1/trip1_acceleration.csv",
      "description": "Acceleration data of the car",
      "format": "CSV",
      "keyFactors": [
        {
          "columnNumber": 1,
          "columnHeader": "acceleration_id",
          "factorType": "NominalKeys"
        }
      ],
      "valueFactors": [
        {
          "columnNumber": 2,
          "columnHeader": "trip_id",
          "factorType": "NominalKeys"
        },
        {
          "columnNumber": 3,
          "columnHeader": "x_value",
          "factorType": "Measurements"
        }
      ]
    }
  ]
}
```

Figure 22: Simple JSON Representation of an AEGIS Dataset

Integration

The Metadata Services requires tight integration into the AEGIS platform, since the metadata is present throughout the entire data value chain, from providing until visualizing data. For creating the metadata, the Metadata Service was integrated into the AEGIS frontend via the Data Annotator.

2.8. Query Builder

Query Builder provides the capability to interactively define and execute queries on data available in the AEGIS system. It is primarily addressed to the AEGIS users with limited technical background, but potentially useful for all, as it simplifies and accelerates the process of retrieving data and creating views on them. As explained also in Section 2.3, Query Builder also offers some simple data cleansing functionalities.

As already stated in D4.1 and D4.2, the name of the component may be misleading, since the word query can be interpreted as the part of the data value chain responsible for retrieving the appropriate data. However, when trying to build the dataset on which an analysis or a visualisation will be applied, the workflow to extract the meaningful parts of the data may include certain processing tasks that cannot be known a priori, in terms of filtering and

cleansing. Therefore, and in order to leverage the computational power of the AEGIS system, Query Builder includes various such functionalities, e.g. null value replacement, filtering based on value, statistics through aggregation functions etc.

In its previous version, as also explained in D4.2, the tool switched to the Jupyter Notebook. The tool, potentially in slightly different flavours (to allow for more customization of the data manipulation processes) will be directly accessible inside every newly created project through the Jupyter Notebook. For the third release, the following updates have been performed:

- Support for AEGIS shared datasets
- New (~25) operations available
- User interface Improvements
- Support for various CSV separators

The user can initially select an interesting dataset and then browse its files in order to choose which one to load. The new version allows the selection of AEGIS shared datasets and the ability to choose csv separators. In the next version the embedded metadata browser inside the Query Builder will be removed in order to be integrated to the core platform.

Once the selected file is opened, it is loaded as a Spark Dataframe, called temp dataset (tempDF), and it is available for further data manipulation. At all times, the user can have up to two different datasets active in Query Builder: the temporary (temp) and the master. The temporary is the one currently being used and changed, whereas the master is used as a “storage point” for intermediate results while the user is processing data and as the final result once all data manipulation is over and the user is satisfied with the outcome.

Type: Table Pie Scatter Line Area Bar

#CF	LastName	Firstname	Gender	Age	Adress Customer	City Customer	Province Customer	Latitude Customer	Longitude Customer	Email	Mobile	Sinis
414	*****	*****	Maschio	35	*****	ROMA	RM	41.851704	12.484327	*****	*****	
540	*****	*****	Femmina	34	*****	ROMA	RM	41.916513	12.573291	*****	*****	
568	*****	*****	Femmina	47	*****	ROMA	RM	41.928984	12.521653	*****	*****	
371	*****	*****	Maschio	31	*****	ROMA	RM	41.789951	12.352493	*****	*****	

Figure 23: Query Builder Temp dataset preview

Filter

Column

Operator

Value

Figure 24: Query Builder data filters view

A number of filters and data processing methods are available for the user to select and apply on the temporary dataset, through the “Controls” panel. Indicatively, the user can fill in null values, filter out entries based on values, rename columns, replace values, select/drop columns etc.

Temp Dataset Preview

City_Name	Country
Skomielna Czarna	PL
Orsha	BY
Overtoarne	SE
Huraydah	YE
Pérez	AR
Stavropol'	RU
Tân Phú	VN
Kuala Lumpur	MY
Zhukovo	RU
Letpandan	MM

Figure 25: Query Builder data filters view

As shown in the image, the new version has 25 more operations organized in Basic operations, Aggregation operations, Joins, Time Conversions, Mathematic operations, Cleansing operations and others.

The user may also merge or append the temporary dataset with the master dataset. A list of selected data manipulation actions, either already applied or pending application, is always visible under the “Selected filters” panel. A preview of the data processing result is always available upon clicking the “Refresh temp” button.

When the result of a series of data processing tasks on the temporary dataset is satisfactory, it can be saved as the master dataset. The user may continue processing the same temporary dataset or open a new one or, when the query creation process is complete, can save the master dataset as a new csv file or export the query and continue to directly change the generated code. This code can be leveraged (a) by the advanced user as an easily acquired starting point to further elaborate on for more complex queries and (b) by the less technically skilled user as a means to understand the underlying code and facilitate learning. Finally, the result of the data manipulation, i.e. the master dataset, can be directly passed as input to more high-level AEGIS tools, like the Visualiser and the Algorithm Execution Container.

2.9. Visualiser

The Visualiser is the component enabling the advanced visualisation capabilities of the AEGIS platform. In accordance to the latest design and the specification of the component, as documented in deliverable D3.4, the purpose of the Visualiser remains two-fold: (1) to provide visualisations of the results generated by the Algorithm Execution Container and (2) to provide visualisations of the results generated by the queries composed and executed by the Query Builder.

For the realisation the advanced visualisation capabilities of the AEGIS, the Jupyter⁸ notebook development environment, incorporated in the AEGIS platform as part of the AEGIS integrated services, has been explored. Jupyter supports multiple programming languages and is integrated with data processing frameworks such as Spark, which is utilised in the AEGIS platform. The Visualiser is implemented as a predefined Jupyter notebook following the microservices architecture. As such, a series of microservices were developed and orchestrated through Jupyter in order to implement the functionalities of the Visualiser, which includes the dataset selection, the dataset preview generation, the visualisation type selection, the visualisation configuration, the visualisation generation and the interactive dashboard. The Visualiser can be accessed through Jupyter that is integrated in the AEGIS Front-End as a service.

In addition to Jupyter, two Python libraries were utilised, namely the Folium⁹ and the highcharts¹⁰ libraries. These libraries are two state-of-the-art open source charting libraries that facilitate the generation of a large variety of interactive charts and visualisations and are used within the Jupyter notebook.

To facilitate the advanced dataset or results visualisation process, an intuitive and easy-to-use user interface has been implemented guiding the user across all the execution workflow of the Visualiser component, as it was designed and documented in the deliverable D3.4. Visualiser offers a variety of visualisation formats which spans from simple static charts to interactive charts with multiple layers of information and several customisation options.

⁸ <http://jupyter.org/>

⁹ <http://folium.readthedocs.io/en/latest/>

¹⁰ <https://www.highcharts.com/>

In the current version the focus was on the improving the Maps visualisation type support. More specifically, the Visualiser is now offering support for Heatmaps on top of Maps (Figure 26) and enhanced map visualisation with support for markers with custom labels and colours (Figure 27). Moreover, support for FastMarkerCluster on Maps is now available. Moreover, a list of improvements have been introduced in the Visualiser with regard to visualisation parameters.

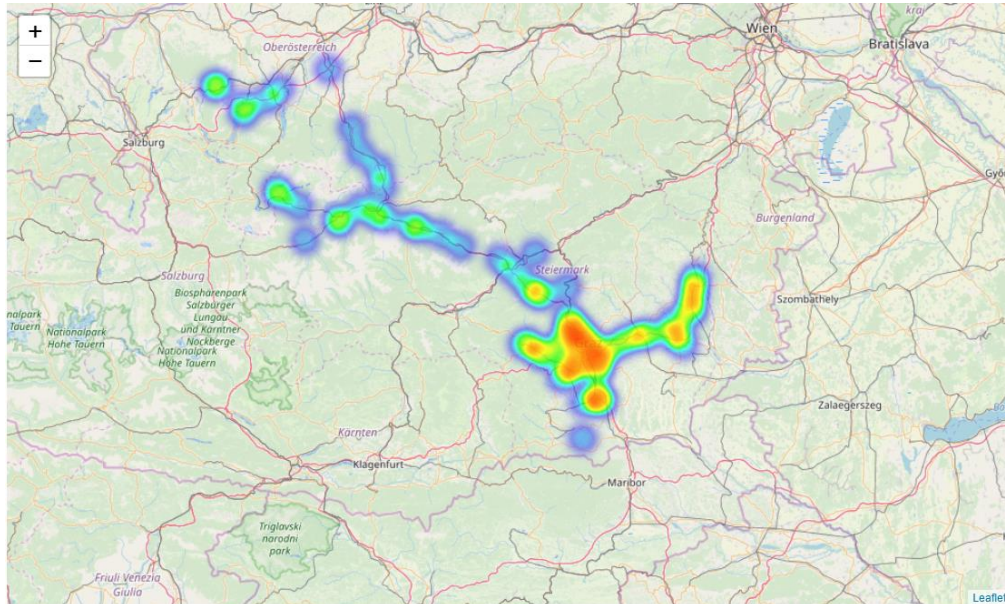


Figure 26: Visualiser Heatmaps on Maps

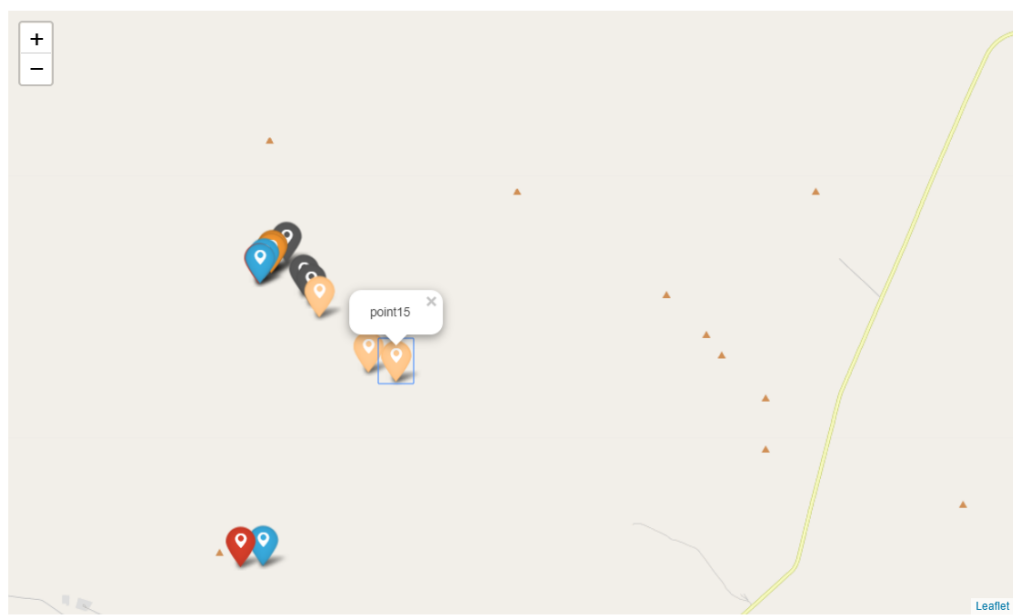


Figure 27: Visualiser Maps with markers with custom labels and colours

The current implementation of the Visualiser component supports the following visualisation types:

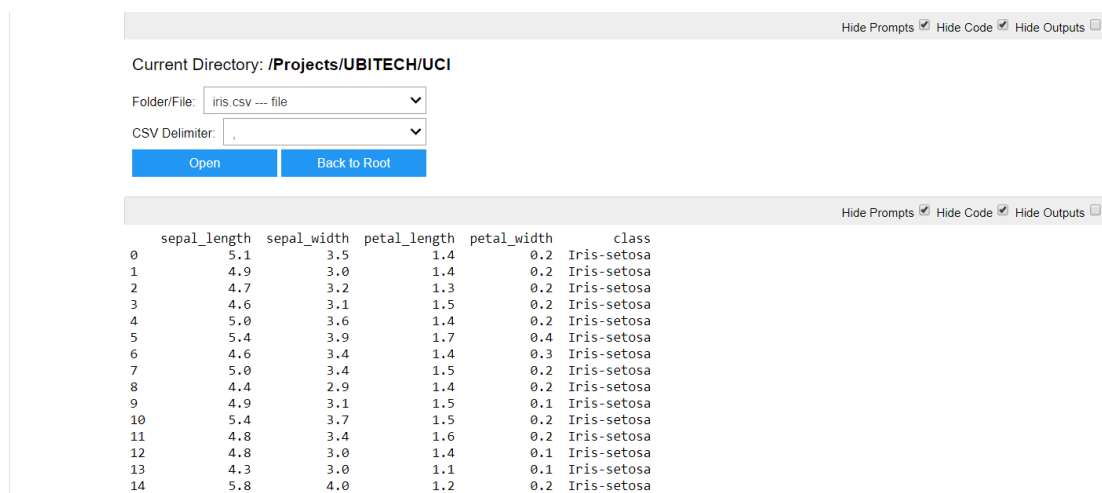
- Scatter plot

- Pie chart
- Bar chart
- Line chart
- Box plot
- Histogram
- Time series
- Heatmap
- Bubble chart
- Map (with support for HeatMaps on Maps, markers with custom labels and colours and FastMarkerCluster)

Other improvement included in the current version is the exception handling that was introduced in order to inform the user for any failure during the visualisation process with the prompt message. Finally, the Visualiser is now supporting the export of the generated visualisation in the form of HTML file that can be saved in the AEGIS Data Store as a new asset of the user's project.

The execution workflow has not changed since deliverable D4.2, however the description of the steps, as described in D4.2, are included below for coherency reasons. The steps are as follows:

1. At first, when the Visualiser is loaded as an interactive notebook the user is presented with a list of options in order to define the dataset that will be utilised for the visualisation creation. The user is able to navigate through the list of available datasets within the project's folders and select the desired dataset. Upon selecting the desired dataset, a preview of the dataset in tabular format is presented to the user (Figure 28).



	sepal_length	sepal_width	petal_length	petal_width	class
0	5.1	3.5	1.4	0.2	Iris-setosa
1	4.9	3.0	1.4	0.2	Iris-setosa
2	4.7	3.2	1.3	0.2	Iris-setosa
3	4.6	3.1	1.5	0.2	Iris-setosa
4	5.0	3.6	1.4	0.2	Iris-setosa
5	5.4	3.9	1.7	0.4	Iris-setosa
6	4.6	3.4	1.4	0.3	Iris-setosa
7	5.0	3.4	1.5	0.2	Iris-setosa
8	4.4	2.9	1.4	0.2	Iris-setosa
9	4.9	3.1	1.5	0.1	Iris-setosa
10	5.4	3.7	1.5	0.2	Iris-setosa
11	4.8	3.4	1.6	0.2	Iris-setosa
12	4.8	3.0	1.4	0.1	Iris-setosa
13	4.3	3.0	1.1	0.1	Iris-setosa
14	5.8	4.0	1.2	0.2	Iris-setosa

Figure 28: Visualiser - dataset selection

2. In the next step, the user is presented with the list of available visualisation types (Figure 29). Once the desired visualisation type is selected, the user is presented with the list of available parameters for the specific visualisation type. The list of parameters includes a variety of options that spans from the variables that will be used in the visualisation

and the titles that will be displayed in the visualisation axis to the selected visualisation’s type specific parameters such as the aggregation function or class variable. An example of the visualisation parameters selection is displayed in Figure 30.

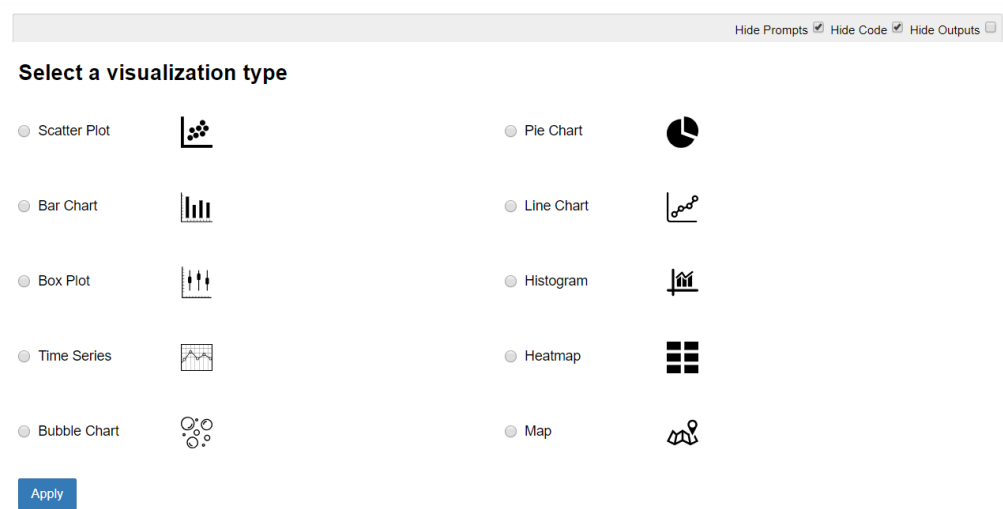


Figure 29: Visualiser - visualisation type selection

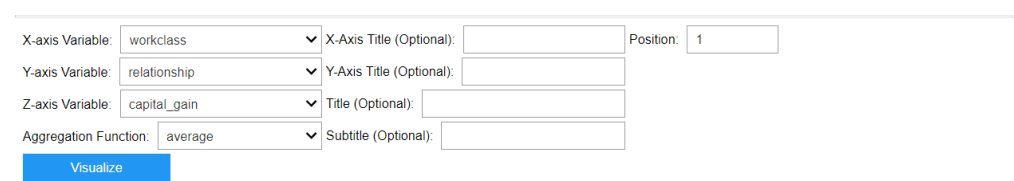


Figure 30: Visualiser - visualisation parameters

3. Once the visualisation type has been selected and the corresponding parameters have been set, the user can trigger the visualisation creation. The following figures illustrate some examples of the visualisations offered by the Visualiser.

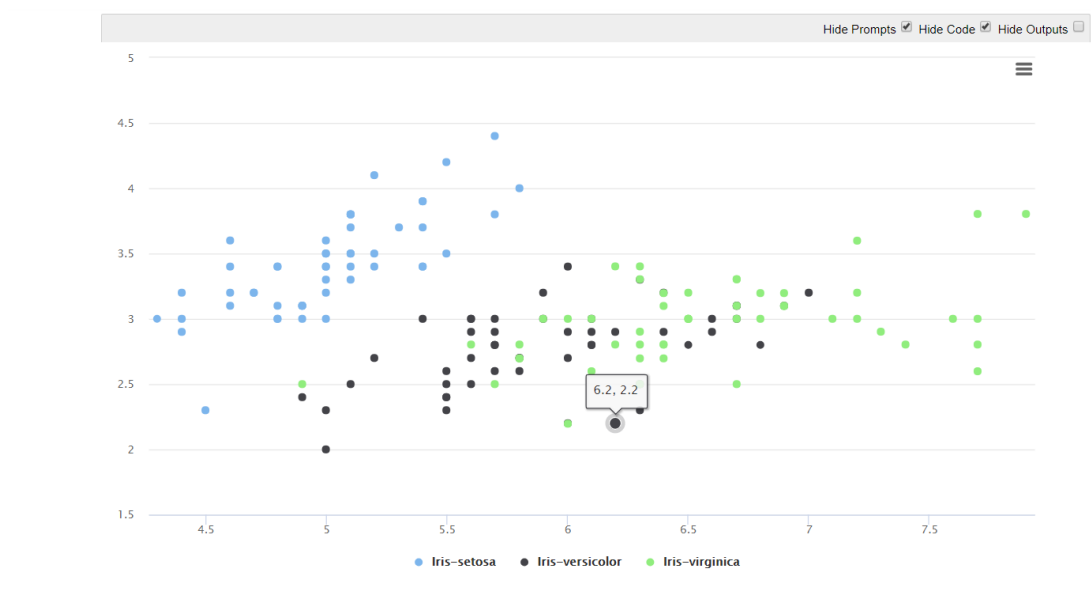
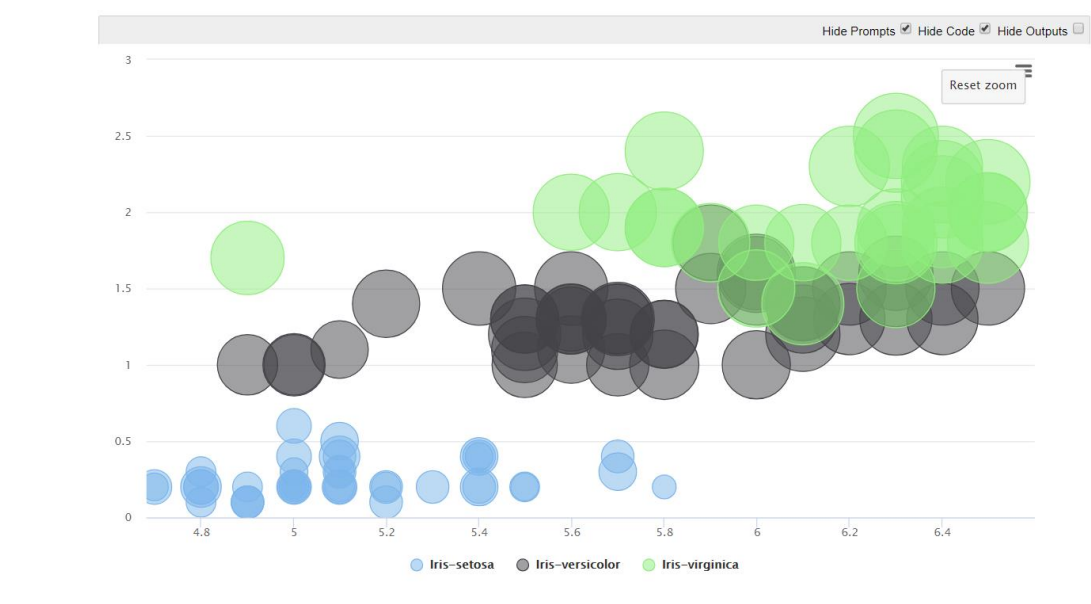
**Figure 31: Visualiser – scatter plot****Figure 32: Visualiser - bubble chart**



Figure 33: Visualiser - time series

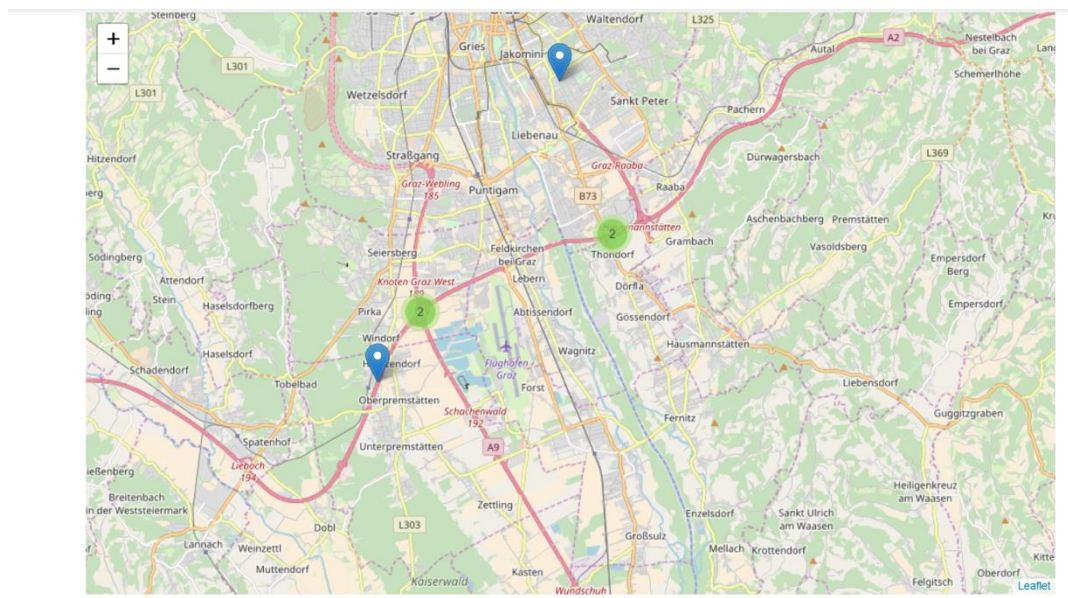


Figure 34: Visualiser – map

2.10. Algorithm Execution Container

Following the release of the Zeppelin version of the AEGIS Algorithm Execution Container and the upgrade of the underlying HOPS platform, a decision was taken to shift implementation efforts to a Jupyter version of the Algorithm Execution Container, which is better supported in HOPS and that will allow integration and interoperation with the other analytics tools (Query Builder and Visualiser) that make already use of Jupyter notebooks.

The current Jupyter version of the container is based on Python and JavaScript. For this new version, the algorithms that have been implemented for the past version (Zeppelin version) and

that were originally written in Scala, were re-written into Python, specifically using the PySpark MLlib package.

Upon launch of the corresponding Jupyter Notebook, the Algorithm Execution Container (hereinafter container) is initialised through a dedicated button. The initialisation ensures that the Spark interpreter of the AEGIS platform is started and also creates the basic UI of the container, which is a simple four-tab window:

- An overview tab showing the current user selections and, when ready, the execution results
- An input file selection tab
- An algorithm selection and configuration tab
- An output folder selection tab

Apart from the overview tab, the other three correspond to the basic steps towards applying an algorithm through the container. The first step is to select the file that contains the data to be used by the algorithm, through the interface shown in the next figure.

Overview **Input File** Algorithm Selection & Configuration Output File

Available Datasets Samples Apply Refresh

Available Files hdfs://172.16.0.6:8020/Projects/s5/Samples/FloodDamageData.csv

CSV Separator , Apply Refresh Preview

Figure 35: AEC Input File Tab

Once a file is loaded, a small preview is available, as shown in Figure 36:

Overview **Input File** Algorithm Selection & Configuration Output File

Available Datasets

Available Files

CSV Separator

Water Level	Damage	Duration	Contact
52.9	760551	1	1
37.2	698802	2	0
60.6	306466	4	1
36.2	630917	2	1
63.6	157173	3	0
35.5	320183	2	1
56.8	764451	2	1
65.1	235739	4	0
44.8	497692	3	1
34.8	626316	4	1]

Figure 36: AEC Data Preview

The next step is to select the algorithm to be applied. The provided algorithms are grouped under five categories (algorithm families). Once an algorithm is selected, a form from which to configure its parameters is shown to the user. A basic form validation is done for the case that a selected value for a parameter is not within the specified boundaries (Figure 37).

Overview *Input File* **Algorithm Selection & Configuration** *Output File*

Algorithm Family CLASSIFICATION/REGRESSION

Algorithm Logistic Regression

Logistic regression is a popular method to predict a categorical response. It is a special case of Generalized Linear models that predicts the probability of the outcomes. In spark.ml logistic regression can be used to predict a binary outcome by using binomial logistic regression, or it can be used to predict a multiclass outcome by using multinomial logistic regression.

Configuration

Label Column Contact

Feature Columns Water Level
Damage
Duration
Contact

Elastic Net Parameter 1

Regularisation Parameter 17

Maximum Iterations 10

Number of Folds for Cross-Validation 3

Figure 37: AEC Algorithm Configuration Tab

The model that will be created when an algorithm is applied is saved in the user's datasets, in a folder specified in the last container tab, as shown below:

Overview *Input File* *Algorithm Selection & Configuration* **Output File**

Select Dataset MLResults Update Browser

Result Folder Name LogisticRegression_Floods

Include Header ☒

Apply

Figure 38: AEC Output File tab

Once this last step is concluded, by pressing the “Apply” button, the user is taken to the Overview Tab:

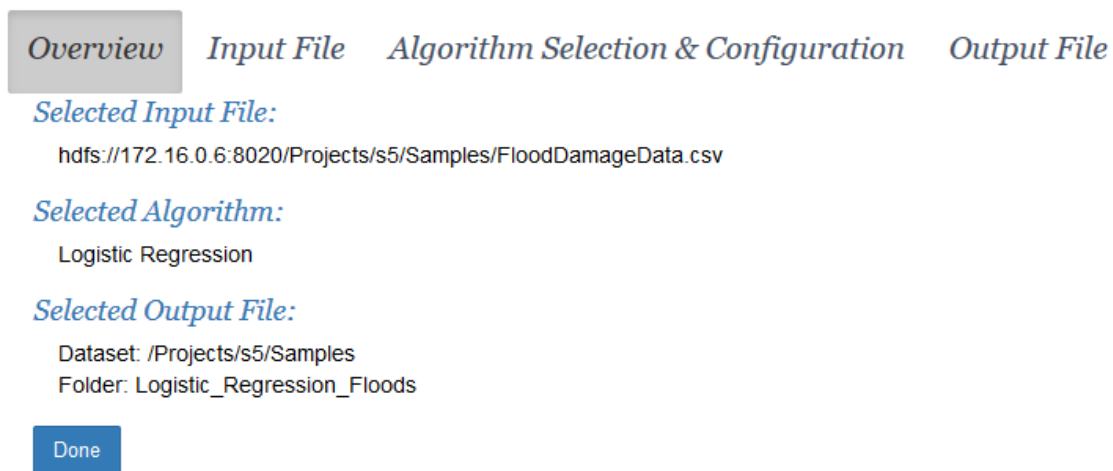


Figure 39: AEC Overview Tab

There, by pressing Done, the algorithm is applied and when the execution is completed, some algorithm-dependent results are provided to the user in the same tab (Figure 40).

The final output of the analysis is stored back in the AEGIS Data Store, while the model that was used for the analysis, is also stored alongside with the analysis results.

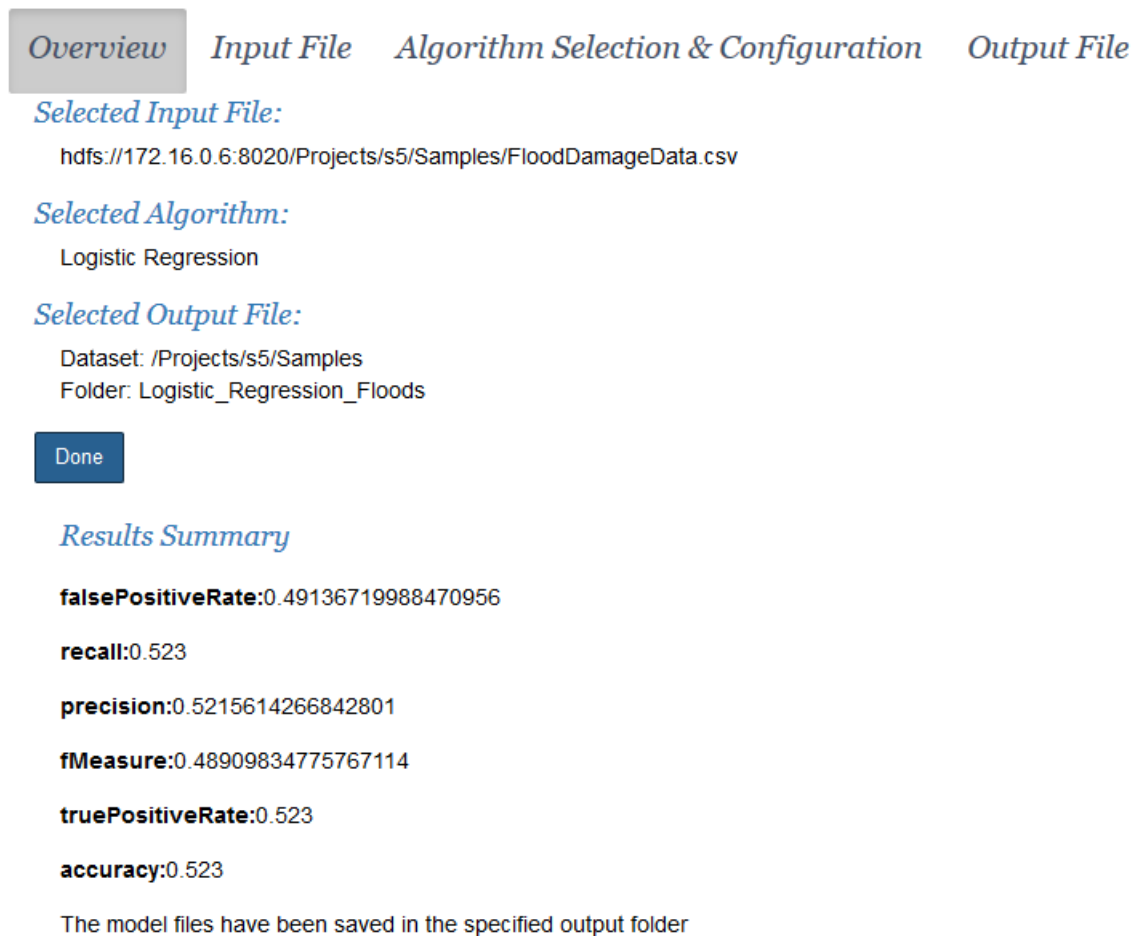


Figure 40: AEC Overview Tab with Results

As with the previous version, it needs to be mentioned, that in case an analyst is willing to run more complex and customised analyses, the module allows the direct edit of the underlying code.

According to the project development plan, the final version of the Algorithm Execution Container will include error checking and will allow for the creation of a unified analytics flow within the platform by interconnecting the container to the Query Builder and the Visualiser.

Finally, it needs to be mentioned that the Zeppelin version of the AEGIS Execution Container is publicly available in the AEGIS Github repository and can be used in earlier HOPS versions, while it can be loaded in any Zeppelin notebook (with small modifications regarding the input/output interfaces), to help individuals automate certain analysis aspects.

2.11. AEGIS Front-end

Following the look and feel of the AEGIS institutional web site¹¹ an updated GUI/front-end for the AEGIS third integrated prototype has been developed, on top of Hopsworks, using as main technologies HTML and AngularJS¹².

The updates consist in many UI graphical improvements and adjustments, partially due to the upgrade of Hopsworks versions. The home page has been redefined (see Figure 42: Home page) and now features one tab for the Assets and one tab for the Public datasets, which are now clickable. The Projects menu on the right side has been as well changed and simplified. In the project page (see Figure 44: Project Datasets), besides, the functionalities linked to a dataset have been enhanced and the user/dataset interaction logic has been modified as follows:

- the hamburger menu icon removed;
- left-click button on the gray part enter the datasets (browse files);
- left-click button on the blue bar show details on the column on the right;
- right-click anywhere opens the menu.

User guides have been included relating to the Visualizer, Query Builder and Algorithm Execution Container (see Figure 43: Query Builder User Guide). Also, a “Get started” page has been inserted as an introductory guide to the platform scope and functionalities in order to address the diverse needs of its users. At the same time, it has been provided a Getting started workflow, describing the “first visit” step by step and introducing the advanced options offered by the platform.

Finally, an EU footer has been placed on every page with a link to AEGIS project web page

¹¹ <https://www.aegis-bigdata.eu/>

¹² <https://angularjs.org/>

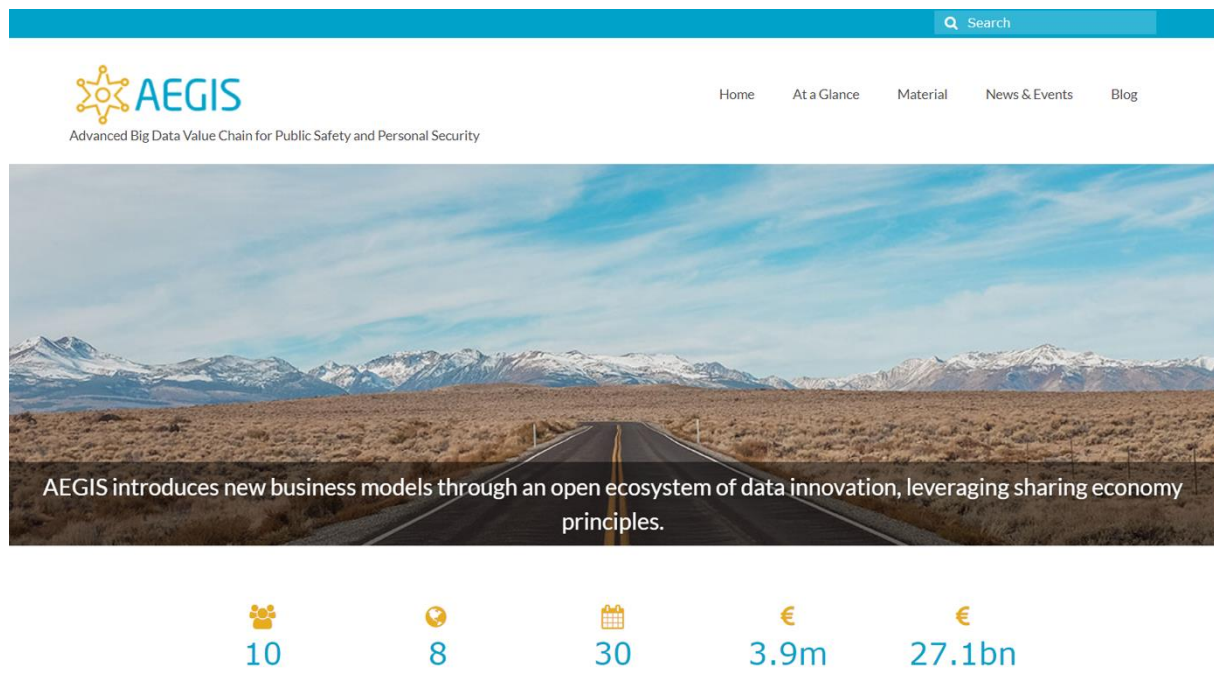


Figure 41: AEGIS web site

The first step of the Front-end is the user account registration, followed by a login/logout mechanism. Once the user is logged into the platform, he/she will be able to browse all the available assets (i.e. datasets, projects) and to browse the public datasets, or to select a project from a menu on the right side of the page.

Moreover, a new project can be created, with the option to specify the related members. After a project has been selected, a new page shows the related activity history and the main menu on the left side, now including the following items: Get Started, Assets, Project Datasets, Project Metadata, Query Builder, Analytics, Visualiser, Jupyter, Kafka, Jobs, Metadata Designer, Settings, Members. In addition, on the top-right corner of the page, a full search functionality is available.

Major technical details about the Front-end technologies have been provided in AEGIS- D3.4 – *Architecture and Revised Components, Microservices and APIs Designs v3.00*.

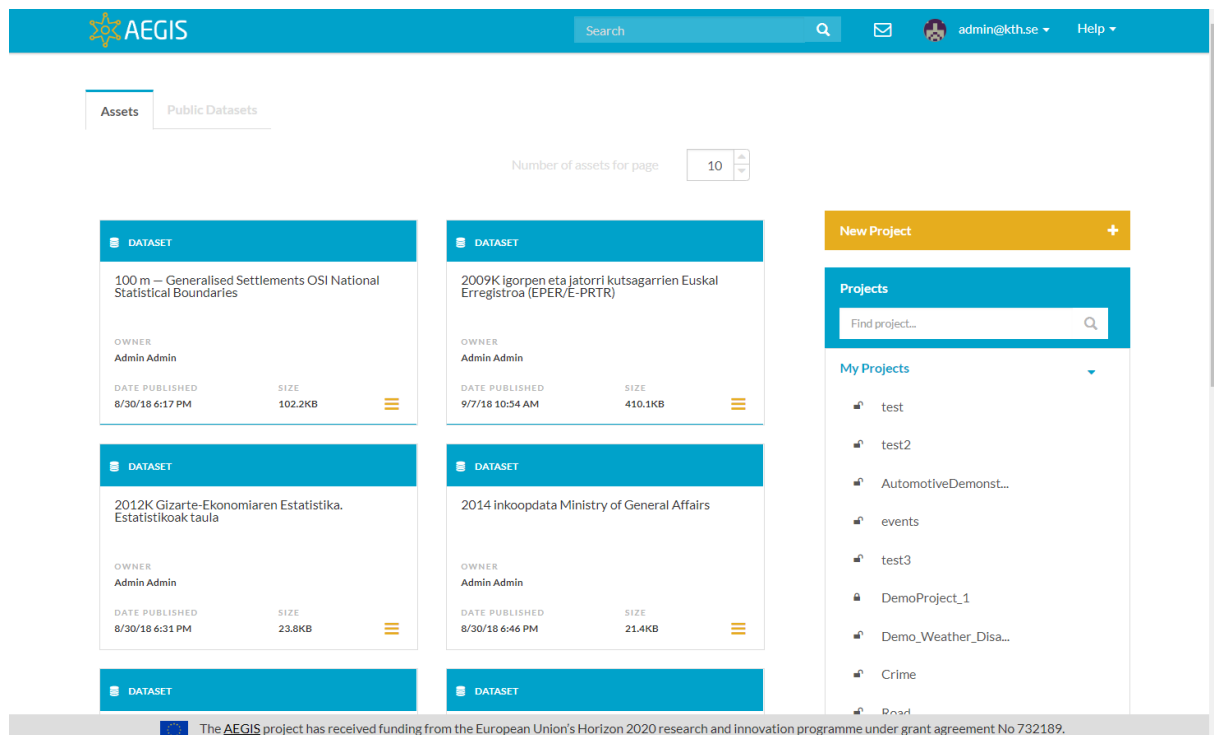


Figure 42: Home page

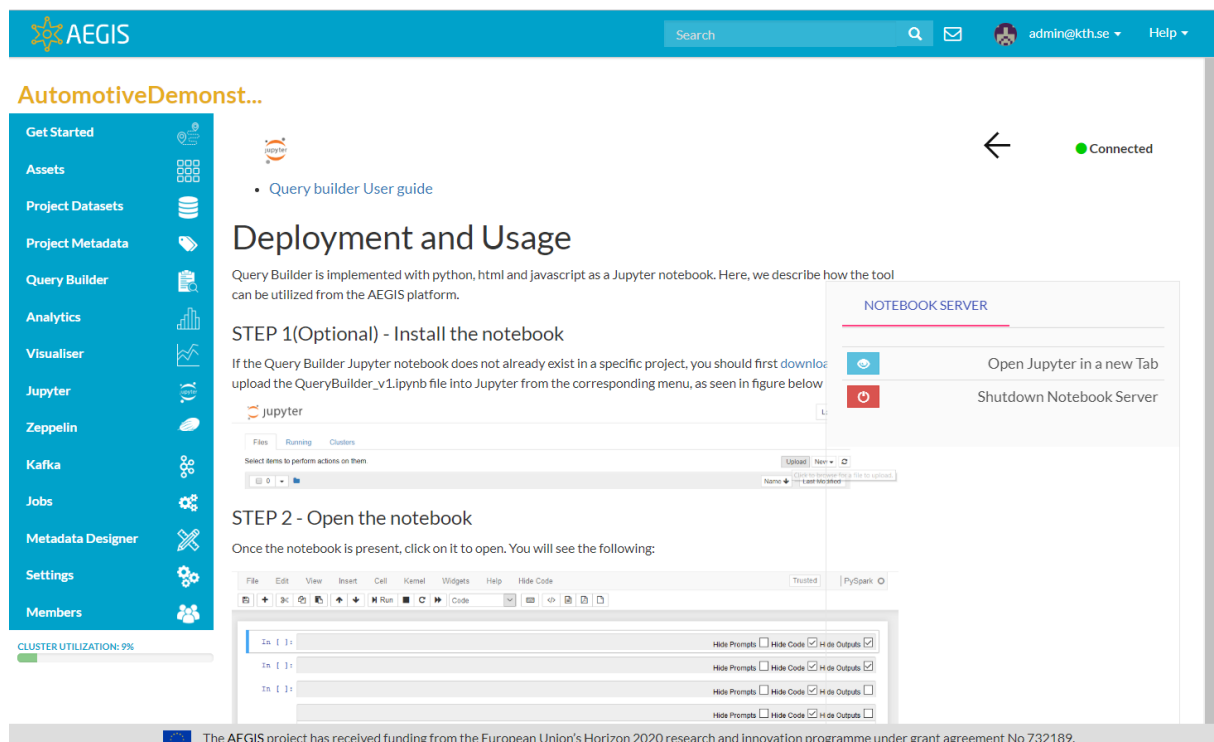


Figure 43: Query Builder User Guide

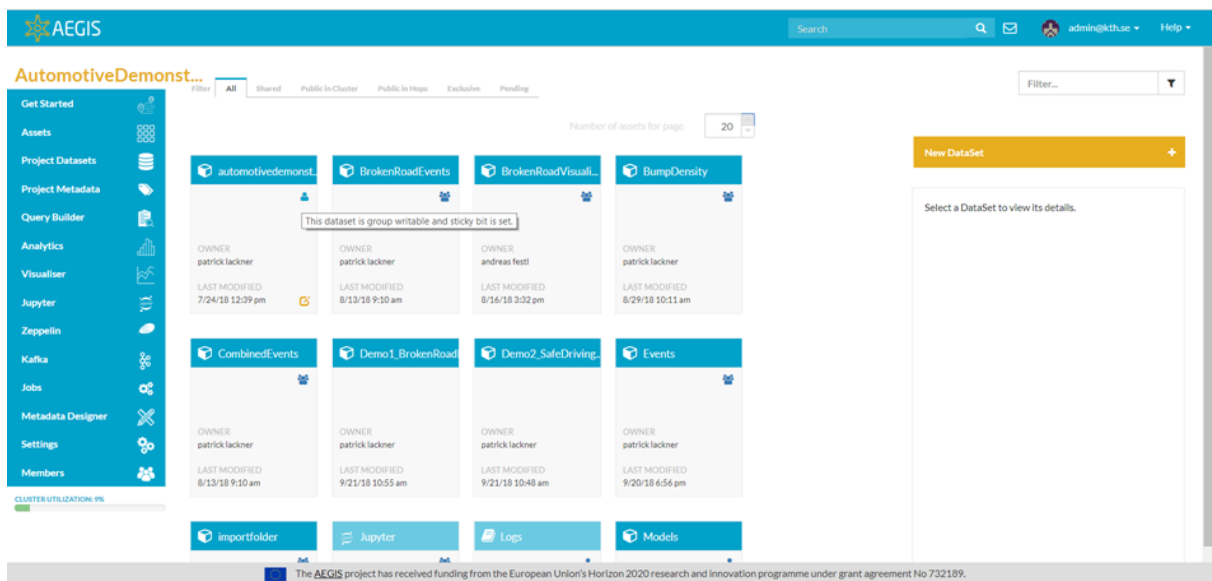


Figure 44: Project Datasets

For the next release of the AEGIS integrated platform, it is foreseen a final restyling of the GUI. Some initial mockups have been produced and samples are depicted in the following figures.

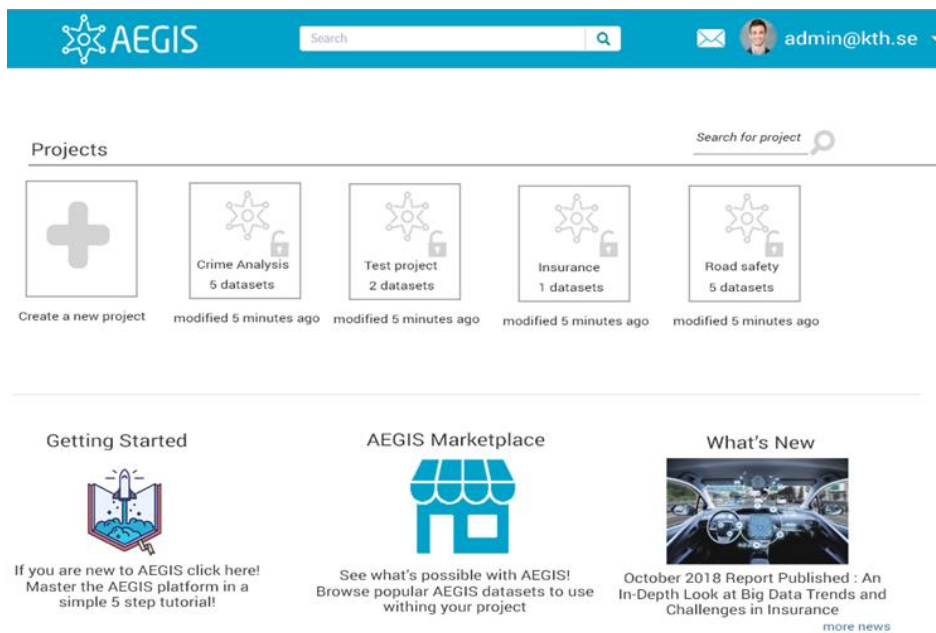


Figure 45: Splash page mockup



Figure 46: AEGIS Marketplace mockup

3. CONCLUSION

The document accompanies the third release of the AEGIS integrated prototype and provides information about the realization of the foreseen software prototype connected to a deployed version of platform. A full description of the platform functionalities developed is provided, as well as references to the software package of the core platform and its API, with corresponding supporting documentation on the deployment of each component and usage of the APIs.

Until the next foreseen deadline at M36, the prototype will be upgraded and refined by providing the final software package of the core platform and its API, with corresponding supporting documentation on the deployment of each component and usage of the APIs.