HORIZON 2020 - ICT-14-2016-1

AEGIS

Advanced Big Data Value Chains for Public Safety and Personal Security

WP4 – AEGIS Infrastructure Implementation and Rollout

D4.2 - AEGIS Platform - Release 2.00

Version 1.0

Due date: 30.06.2018 **Delivery Date: 17.07.2018**

Author(s): Maurizio Megliola, Alessandro Calcagno (GFT), Dimitrios Miltiadou, Konstantinos Perakis (UBITECH), Evmorfia Biliri, Spiros Mouzakitis, Christos Botsikas (NTUA), Fritz Meiners, Fabian Kirstein (Fraunhofer), Sotiris Koussouris, 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), Mahmoud Ismail (KTH)

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

This deliverable consists of a software package of a core platform and its APIs with corresponding supporting documentation on the deployment of each component and usage of the APIs. This deliverable includes a partially functional high fidelity software prototype connected to a deployed version of the platform. The initial interface includes the basic UI/UX (updated from D4.1) for the users of the platform.

Table of Contents

List of Figures

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
Y1	Year 1

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, a partially functional high fidelity software prototype connected to a deployed version of the platform is supplied. The initial interface includes the basic UI/UX (updated from D4.1) for the users of the platform.

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 D3.3, 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 start providing 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.

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.

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 second release of the prototype. During the further development using the StreamSets platform presented in deliverable D4.1 a few use cases hindering the adoption of said platform were encountered. First and foremost, the StreamSets platform proved to not be flexible enough to enable a generic use within the current ecosystem. Also, occasional stability problems became apparent while testing. The creation of custom adapters suitable for use within the StreamSets environment demanded utilisation of their plugin API, which posed problems when trying to reuse code. Thus, despite being a powerful tool, StreamSets was dropped in favour of extending the software that has previously been written by Fraunhofer FOKUS.

Data Harvester Microservices

In order to tackle the mentioned problems a different approach was chosen. Instead of using a third-party software, the Java based EDP Metadata Transformer Service (EMTS) has been split up into its components, transforming the monolithic application into a microservice architecture, consisting out of the following four basic services:

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

- **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 second 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.x[2] 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 Docker[3]. 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.

To handle the frictionless interaction between various microservices their APIs have been defined using the OpenAPI 3[4] 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.

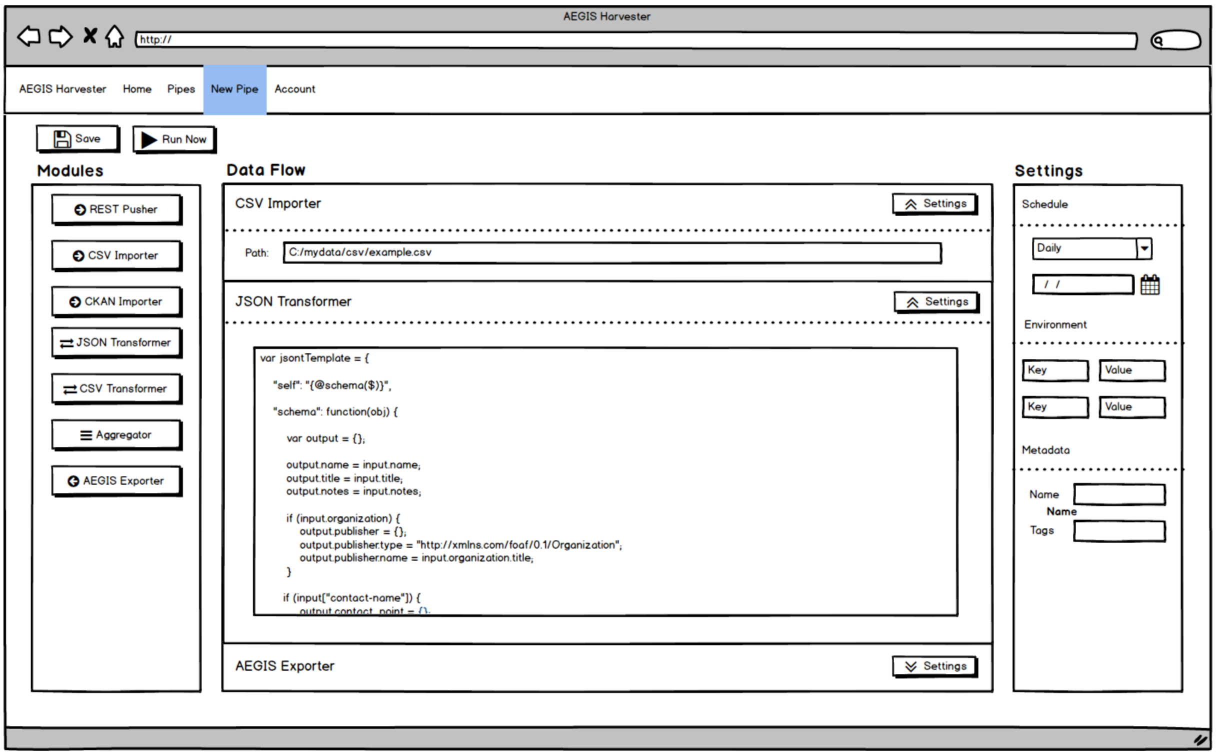
Data Harvester Frontend

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). This frontend is to be developed for the next release of the AEGIS platform. Figure 1 and Figure 2 illustrate the envisioned frontend. Users will be able to interactively create harvesting pipes out of the set of available services.

[2] https://vertx.io/

[3] https://www.docker.com/

[4] https://github.com/OAI/OpenAPI-Specification



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

```
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.publisher.name = input.organization.title;
    }

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

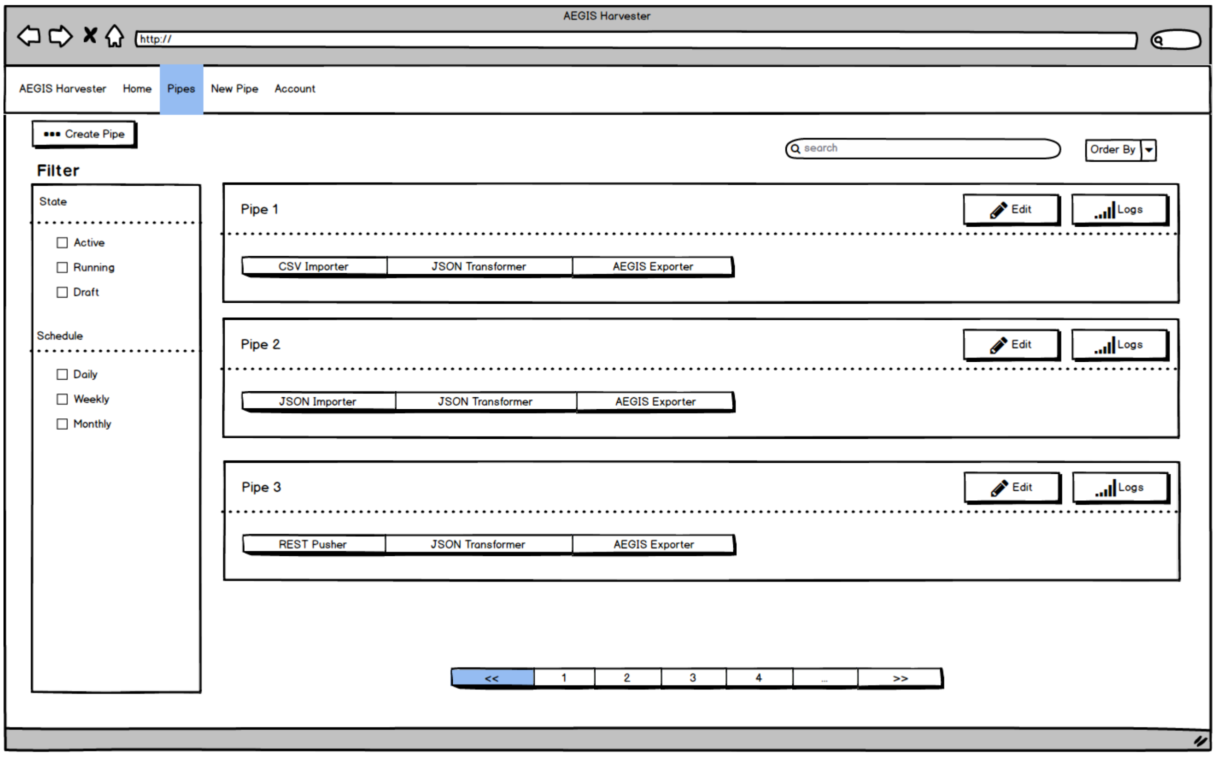
Schedule: Daily

Date: / /

Key	Value
Key	Value
Key	Value

Name:
Name
Tags:

Figure 1: Mockup for Creating a Harvesting Pipe



Pipe 1: CSV Importer | JSON Transformer | AEGIS Exporter

Pipe 2: JSON Importer | JSON Transformer | AEGIS Exporter

Pipe 3: REST Pusher | JSON Transformer | AEGIS Exporter

Figure 2: Overview of Existing Harvesting Pipes

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.

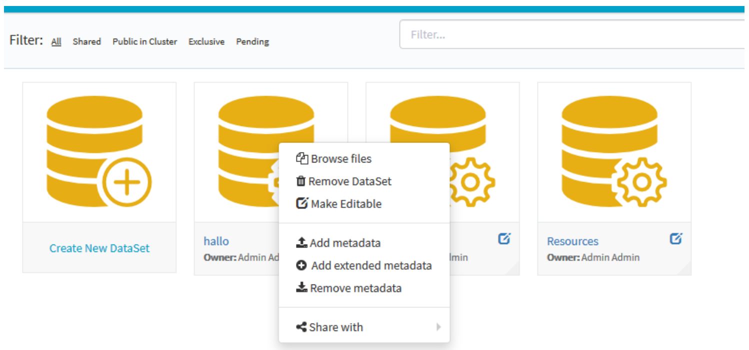
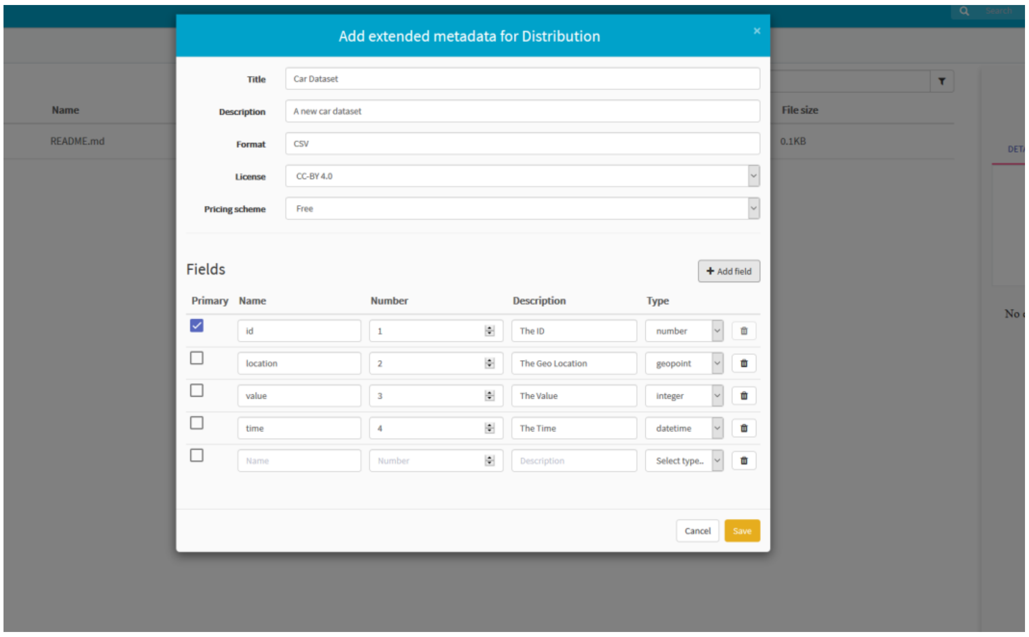


Figure 3: Sub Menu for a Data File

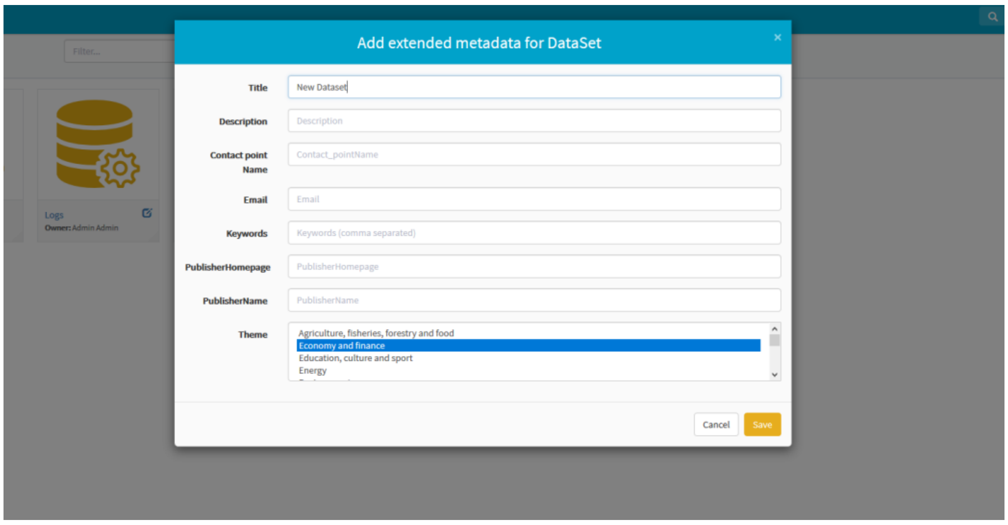


Title	Car Dataset
Description	A new car dataset
Format	CSV
License	CC-BY 4.0
Pricing scheme	Free

Fields

Primary	Name	Number	Description	Type
✓	id	1	The ID	number
☐	location	2	The Geo Location	geopoint
☐	value	3	The Value	integer
☐	time	4	The Time	datetime
☐	Name	Number	Description	Select type..

Figure 4: Data Annotator for CSV Files



Add extended metadata for DataSet

Title	New Dataset
Description	Description
Contact point Name	Contact_pointName
Email	Email
Keywords	Keywords (comma separated)
PublisherHomepage	PublisherHomepage
PublisherName	PublisherName
Theme	Agriculture, fisheries, forestry and food [✓ Economy and finance] Education, culture and sport Energy

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

In the deliverable D3.3, the revised approach for the data cleansing tool that will be offered by the AEGIS platform has been documented. More specifically, for the data cleansing tool a two-fold approach is followed: (a) an offline cleansing tool, residing where the data are located, that provides the necessary cleansing processes and functionalities with a variety of techniques to enable data validation, data cleansing and data completion to the data prior to importing them in the AEGIS platform and (b) an online cleansing tool for data cleansing and manipulation with a set of functionalities offered during the data query creation process, addressing certain simple cleansing tasks that are time computationally intense leveraging the computational power of the AEGIS platform.

Concerning the online cleansing tool, as already described in D3.3, the offered functionalities are incorporated inside the Query Builder component and will be described in the corresponding section (See 2.8).

The purpose of the offline cleansing tool is to provide an easily customisable and adaptable to the user's needs tool that will enhance user experience and facilitate the users to accomplish the required cleansing tasks. The implementation of the offline cleansing tool is following the design and specification documented in D3.3. As such, the tool is implemented as a standalone application written in Python using the Flask microframework[5] and a set of libraries such as Pandas[6] and NumPy[7]. In the first prototype version of the offline cleansing tool the rules for data validation, data cleansing and missing data handling can be set according to the user's

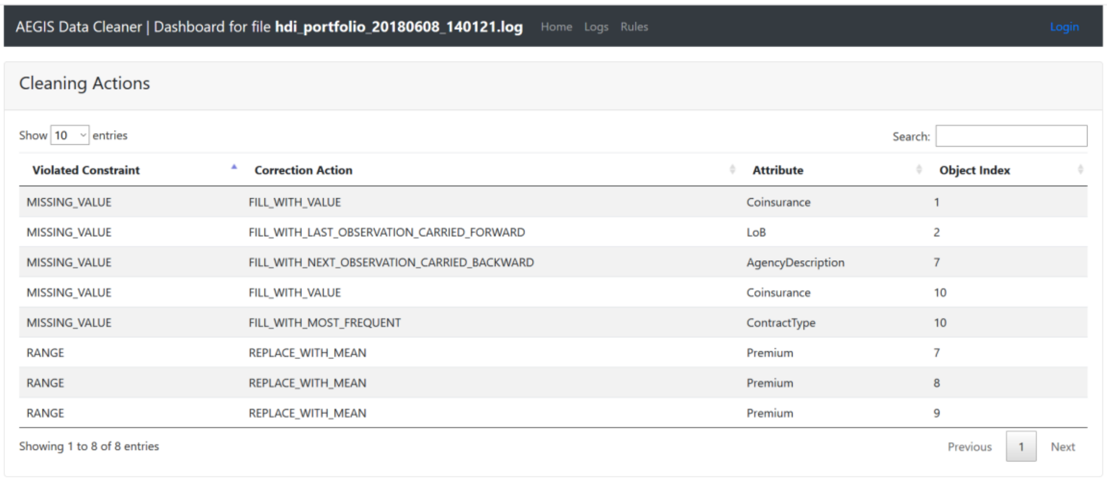
[5] http://flask.pocoo.org/

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

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

needs. Additionally, the user is able to review the results of the execution of the cleansing tasks through the user interface. 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 upcoming versions of the tool will contain several enhancements and updates in terms of cleansing functionalities and rules definition.

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


AEGIS Data Cleaner | Dashboard for file **hdi_portfolio_20180608_140121.log**
Home
Logs
Rules
Login

Cleaning Actions

Show 10 entries Search:

Violated Constraint	Correction Action	Attribute	Object Index
MISSING_VALUE	FILL_WITH_VALUE	Coinsurance	1
MISSING_VALUE	FILL_WITH_LAST_OBSERVATION_CARRIED_FORWARD	LoB	2
MISSING_VALUE	FILL_WITH_NEXT_OBSERVATION_CARRIED_BACKWARD	AgencyDescription	7
MISSING_VALUE	FILL_WITH_VALUE	Coinsurance	10
MISSING_VALUE	FILL_WITH_MOST_FREQUENT	ContractType	10
RANGE	REPLACE_WITH_MEAN	Premium	7
RANGE	REPLACE_WITH_MEAN	Premium	8
RANGE	REPLACE_WITH_MEAN	Premium	9

Showing 1 to 8 of 8 entries

Figure 6: Offline cleansing tool cleansing tasks

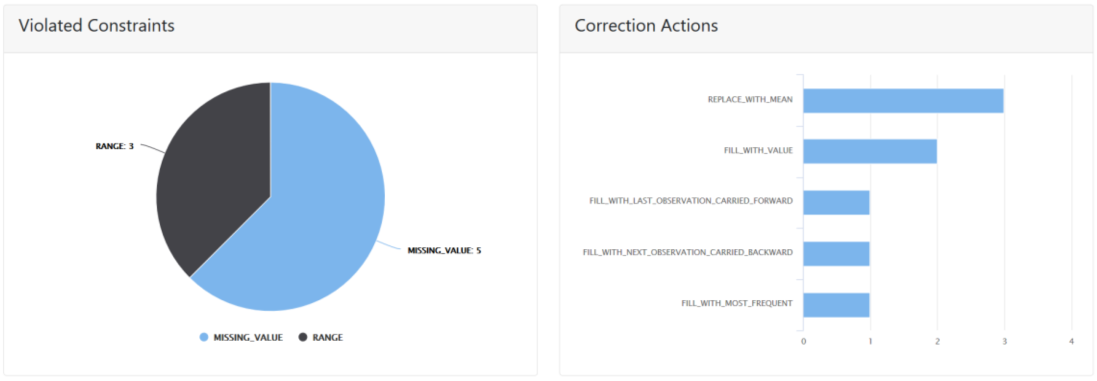


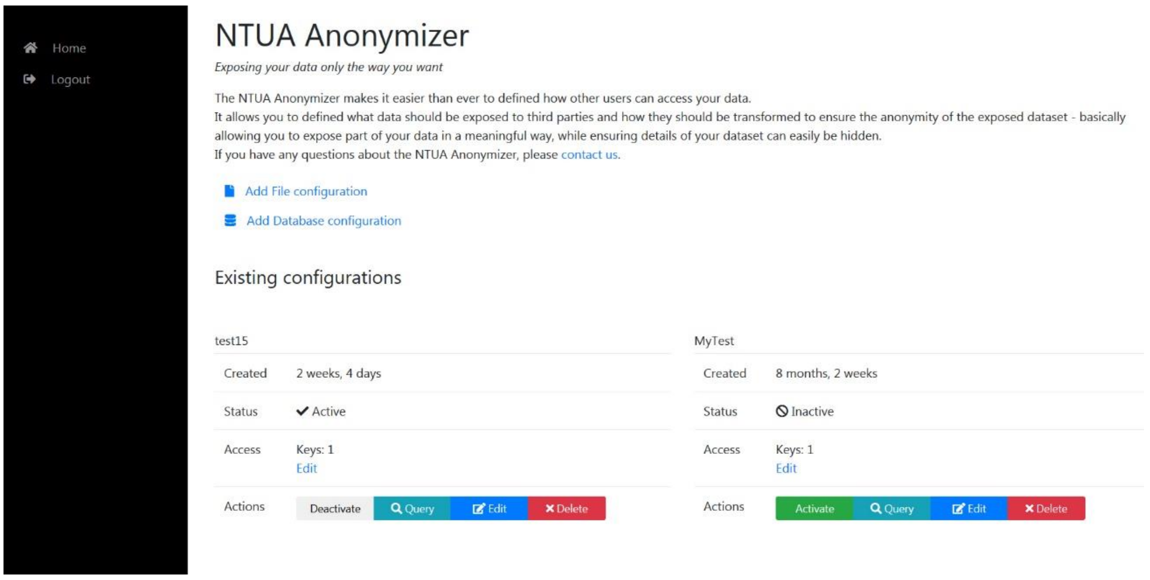
Figure 7: Offline cleansing tools results statistics

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.

Deliverable D3.3 - AEGIS Components, Microservices and APIs Design v2.00 provides further description of the anonymization tool.

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.



NTUA Anonymizer

Exposing your data only the way you want

The NTUA Anonymizer makes it easier than ever to defined how other users can access your data.
It allows you to defined what data should be exposed to third parties and how they should be transformed to ensure the anonymity of the exposed dataset - basically allowing you to expose part of your data in a meaningful way, while ensuring details of your dataset can easily be hidden.
If you have any questions about the NTUA Anonymizer, please contact us.

Add File configuration
Add Database configuration

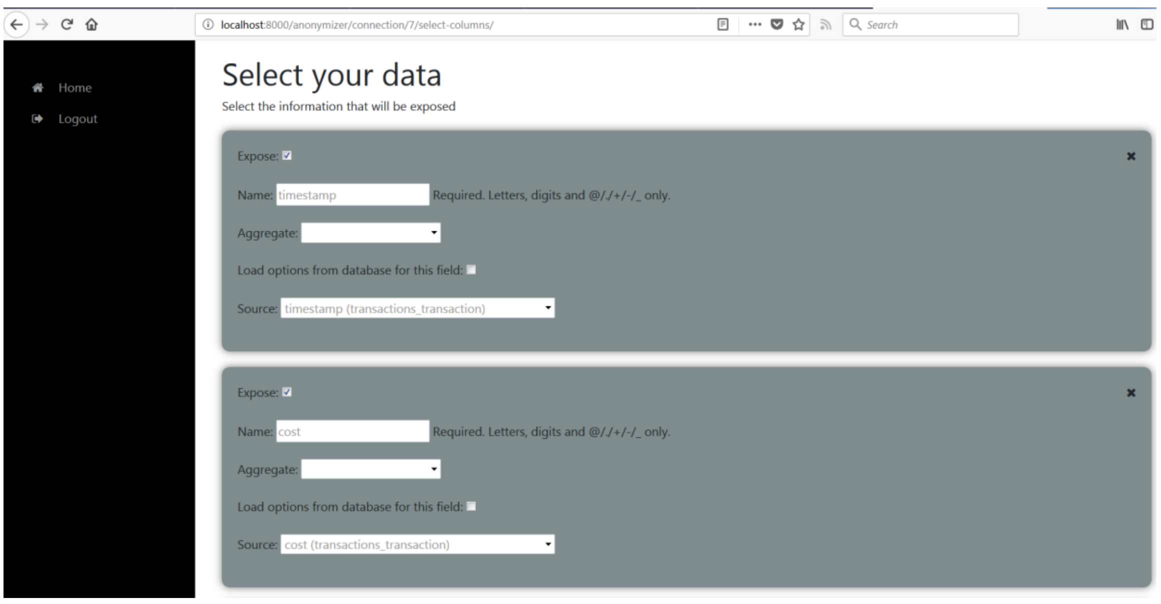
Existing configurations

test15	
Created	2 weeks, 4 days
Status	✓ Active
Access	Keys: 1 Edit
Actions	Deactivate Query Edit Delete

MyTest	
Created	8 months, 2 weeks
Status	Inactive
Access	Keys: 1 Edit
Actions	Activate Query Edit Delete

Figure 8: 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.


localhost:8000/anonymizer/connection/7/select-columns/

Home
Logout

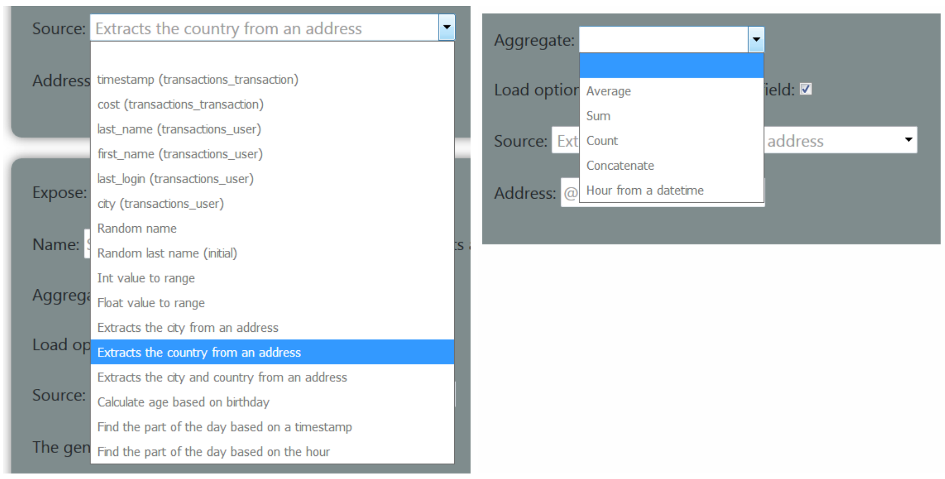
Select your data
Select the information that will be exposed

Expose: ☑
Name: timestamp Required. Letters, digits and @/./+/-/_ only.
Aggregate:
Load options from database for this field: ☐
Source: timestamp (transactions_transaction)

Expose: ☑
Name: cost Required. Letters, digits and @/./+/-/_ only.
Aggregate:
Load options from database for this field: ☐
Source: cost (transactions_transaction)

Figure 9: 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).


Source: Extracts the country from an address

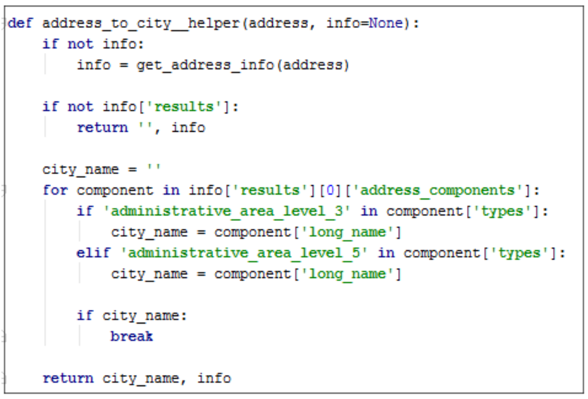
timestamp (transactions_transaction)
cost (transactions_transaction)
last_name (transactions_user)
first_name (transactions_user)
last_login (transactions_user)
city (transactions_user)
Random name
Random last name (initial)
Int value to range
Float value to range
Extracts the city from an address
Extracts the country from an address
Extracts the city and country from an address
Calculate age based on birthday
Find the part of the day based on a timestamp
Find the part of the day based on the hour

Aggregate:

Average
Sum
Count
Concatenate
Hour from a datetime

Figure 10: Anonymiser indicative functions

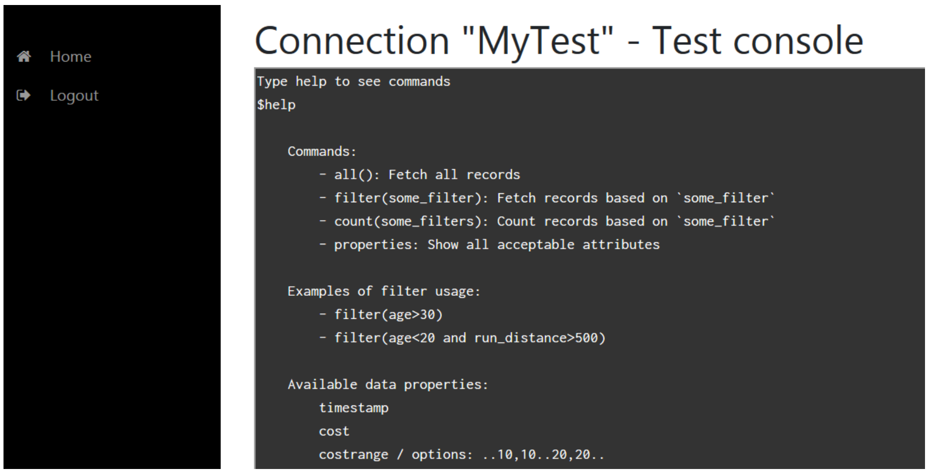
The tool can be easily extended with any new, custom anonymisation functions defined by the user in a python module.



```python
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 11: 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.

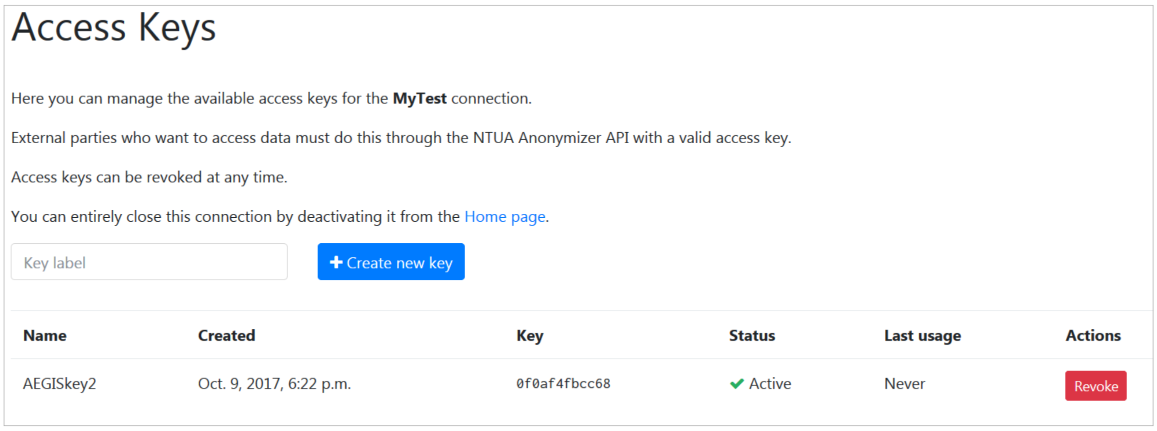


Connection "MyTest" - Test console

```
Type help to see commands
$help

    Commands:
        - all(): Fetch all records
        - filter(some_filter): Fetch records based on `some_filter`
        - count(some_filters): Count records based on `some_filter`
        - properties: Show all acceptable attributes

    Examples of filter usage:
        - filter(age>30)
        - filter(age<20 and run_distance>500)

    Available data properties:
        timestamp
        cost
        costrange / options: ..10,10..20,20..
```

Figure 12: Anonymiser output

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.

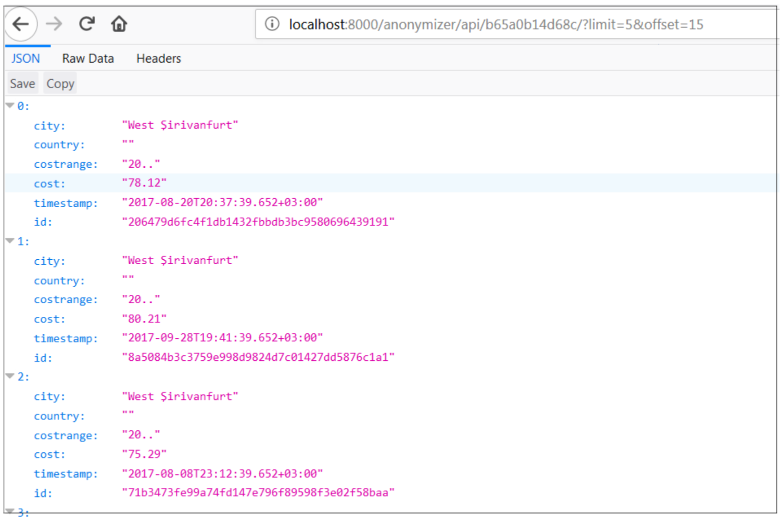
Access keys can be revoked at any time.

You can entirely close this connection by deactivating it from the Home page.

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

Figure 13: Anonymiser exposed API

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



localhost:8000/anonymizer/api/b65a0b14d68c/?limit=5&offset=15
JSON
Raw Data
Headers
Save
Copy

```
0:
    city:        "West Şirivanfurt"
    country:     ""
    costrange:   "20.."
    cost:        "78.12"
    timestamp:   "2017-08-20T20:37:39.652+03:00"
    id:          "206479d6fc4f1db1432fbbdb3bc9580696439191"
1:
    city:        "West Şirivanfurt"
    country:     ""
    costrange:   "20.."
    cost:        "80.21"
    timestamp:   "2017-09-28T19:41:39.652+03:00"
    id:          "8a5084b3c3759e998d9824d7c01427dd5876c1a1"
2:
    city:        "West Şirivanfurt"
    country:     ""
    costrange:   "20.."
    cost:        "75.29"
    timestamp:   "2017-08-08T23:12:39.652+03:00"
    id:          "71b3473fe99a74fd147e796f89598f3e02f58baa"
3:
```

Figure 14: Anonymiser API response

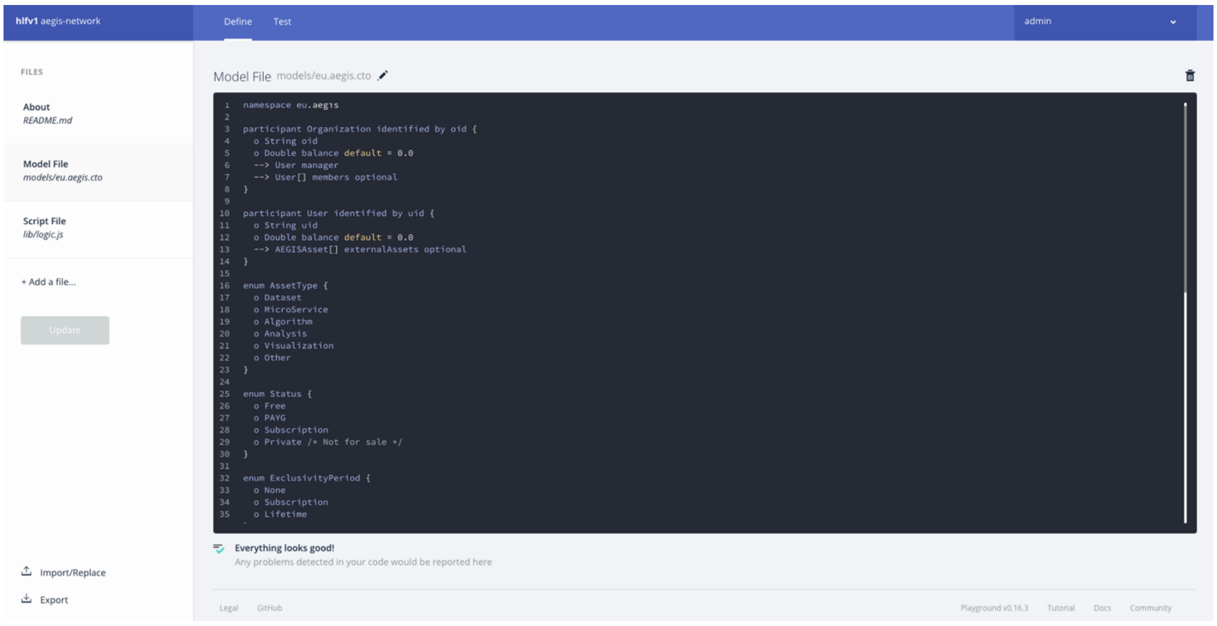
2.5. Brokerage Engine

The brokerage engine of AEGIS acts as a trusted way to record and keep a log of transactions over the AEGIS platforms, which mostly have to do with the sharing of the different data assets as those have been defined in deliverables D2.1 and in D2.3.

The current implementation of the brokerage engine is based on the Hyperledger Fabrik open source distribution and the models that are supported come from deliverable D2.1. However, as D2.3 has recorded a shift in the way of models to be recorded in the blockchain ledger, following the ODRL ontology and distinguishing elements that can be served directly by the AEGIS platform and that can be served by the Blockchain Engine, the models of the overall engine will be revised and delivered in the next version of the platform, that is under D4.3.

However, this change does not affect the integration logic nor the business logic of the overall blockchain engine.

In this context, the current versions of the transaction models supported are based on the Data Policy Framework (deliverable D2.1) and have been optimised to reflect the main points of the Business Brokerage Framework. These models are to be updated in the deliverable D2.3 and also the most important descriptors to be exposed through the AEGIS metadata editor, in order to accompany each data asset with the necessary model attributes for the brokerage engine to perform.



```
namespace eu.aegis

participant Organization identified by oid {
  o String oid
  o Double balance default = 0.0
  --> User manager
  --> User[] members optional
}

participant User identified by uid {
  o String uid
  o Double balance default = 0.0
  --> AEGISAsset[] externalAssets optional
}

enum AssetType {
  o Dataset
  o MicroService
  o Algorithm
  o Analysis
  o Visualization
  o Other
}

enum Status {
  o Free
  o PAYG
  o Subscription
  o Private /* Not for sale */
}

enum ExclusivityPeriod {
  o None
  o Subscription
  o Lifetime
```

Everything looks good!
Any problems detected in your code would be reported here

Figure 15: Snippet of Brokerage Engine Script for AEGIS Data Asset Models coming from D2.1

The Blockchain Engine is deployed on GFT servers and goes together with the overall AEGIS distribution, facilitating in this manner the transactions that are happening over the platform. Connection with the platform is based on the REST API interface exposed by the Blockchain Engine, where essential actions on the platform, such as user creation and dataset sharing are also recorded in the Blockchain ledger.

The Brokerage engine, after its customisation, has been containerised with the use of Docker, in order to allow for easier and faster deployment. Moreover, this approach will allow multiple nodes of the brokerage engine to be deployed in the different AEGIS clusters that can be set up in the future, to allow the interconnection of those into a single AEGIS distributed ledger in case they are all connected to the same public network and then with specific peer and ordered keys are issued to facilitate interconnection and service discovery.

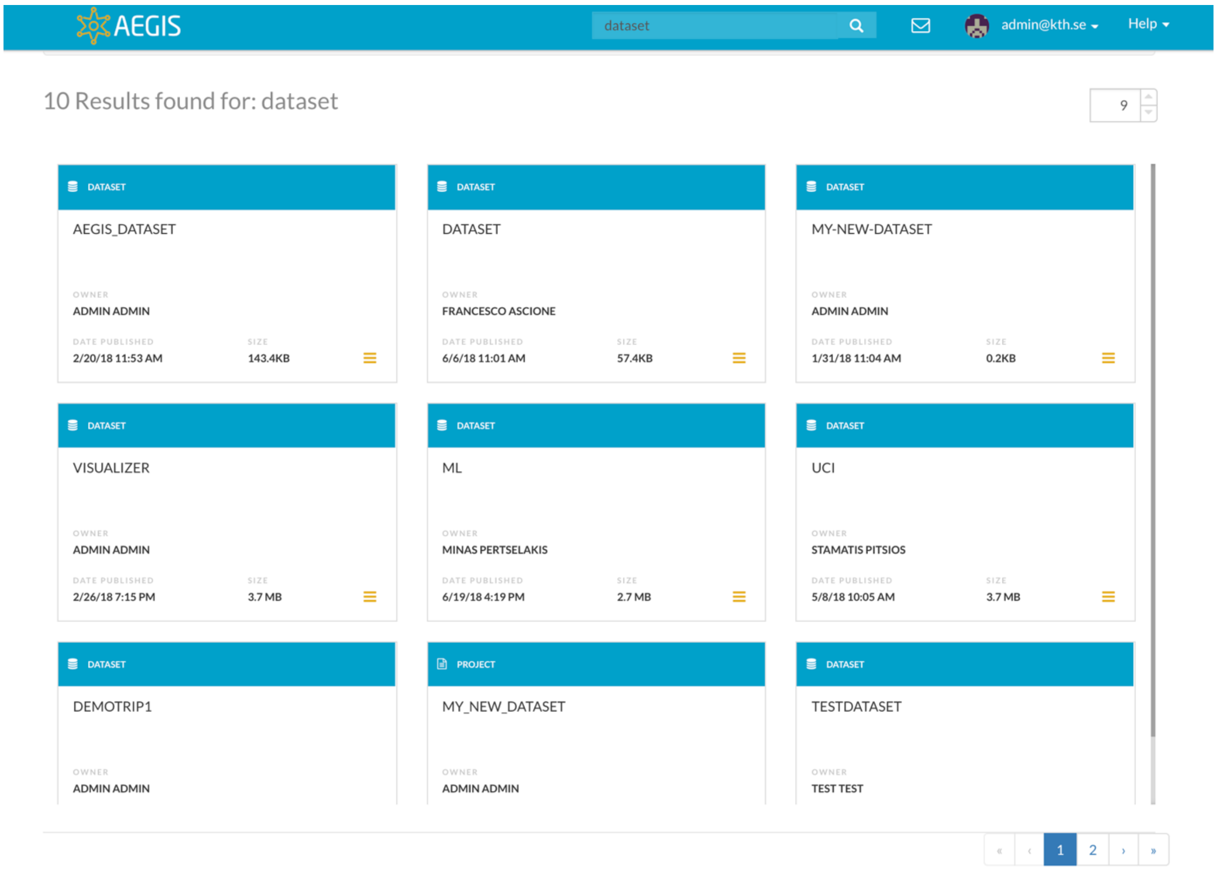
Moreover, a testbed platform has been deployed to be used for performing tests during integration with the front-end of AEGIS and of the rest of the AEGIS platform, in order not to interfere with the production deployment of the blockchain ledger that is serving the 1st prototype version of the AEGIS platform.

2.6. Hopsworks 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 Zeppelin and 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.

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 16 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.

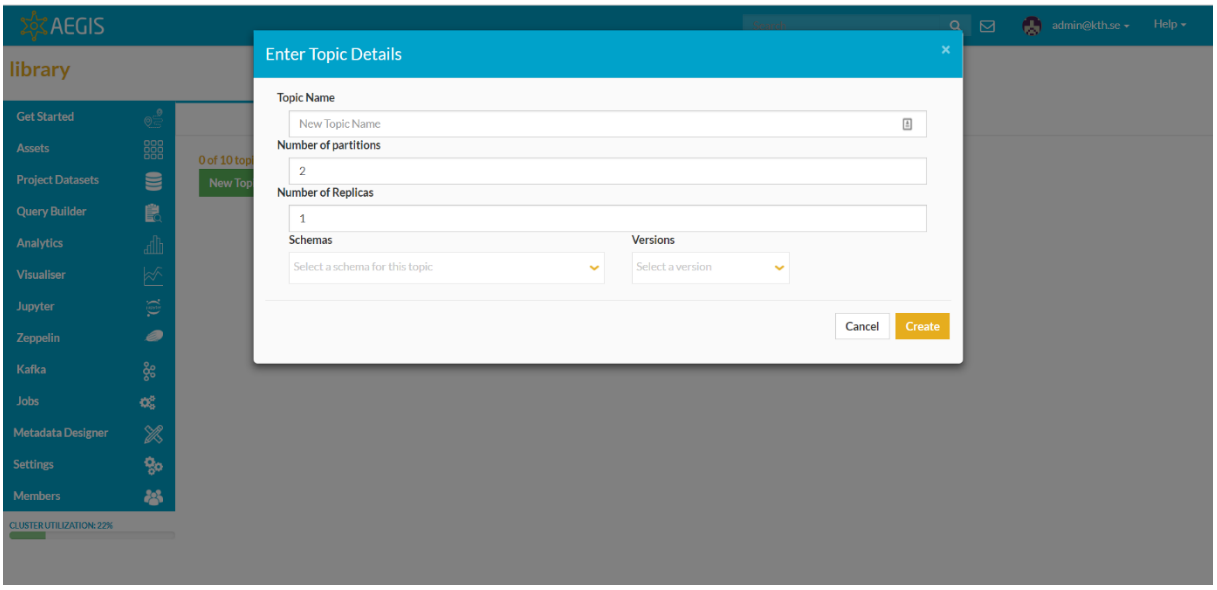


10 Results found for: dataset

Type	Name	Owner	Date published	Size
DATASET	AEGIS_DATASET	ADMIN ADMIN	2/20/18 11:53 AM	143.4KB
DATASET	DATASET	FRANCESCO ASCIONE	6/6/18 11:01 AM	57.4KB
DATASET	MY-NEW-DATASET	ADMIN ADMIN	1/31/18 11:04 AM	0.2KB
DATASET	VISUALIZER	ADMIN ADMIN	2/26/18 7:15 PM	3.7 MB
DATASET	ML	MINAS PERTSELAKIS	6/19/18 4:19 PM	2.7 MB
DATASET	UCI	STAMATIS PITSIOS	5/8/18 10:05 AM	3.7 MB
DATASET	DEMOTRIP1	ADMIN ADMIN		
PROJECT	MY_NEW_DATASET	ADMIN ADMIN		
DATASET	TESTDATASET	TEST TEST		

Figure 16: Full-text search support using Elasticsearch

Kafka



Enter Topic Details

Topic Name: New Topic Name

Number of partitions: 2

Number of Replicas: 1

Schemas: Select a schema for this topic

Versions: Select a version

Figure 17: KAFKA topic details

2.7. Metadata Service

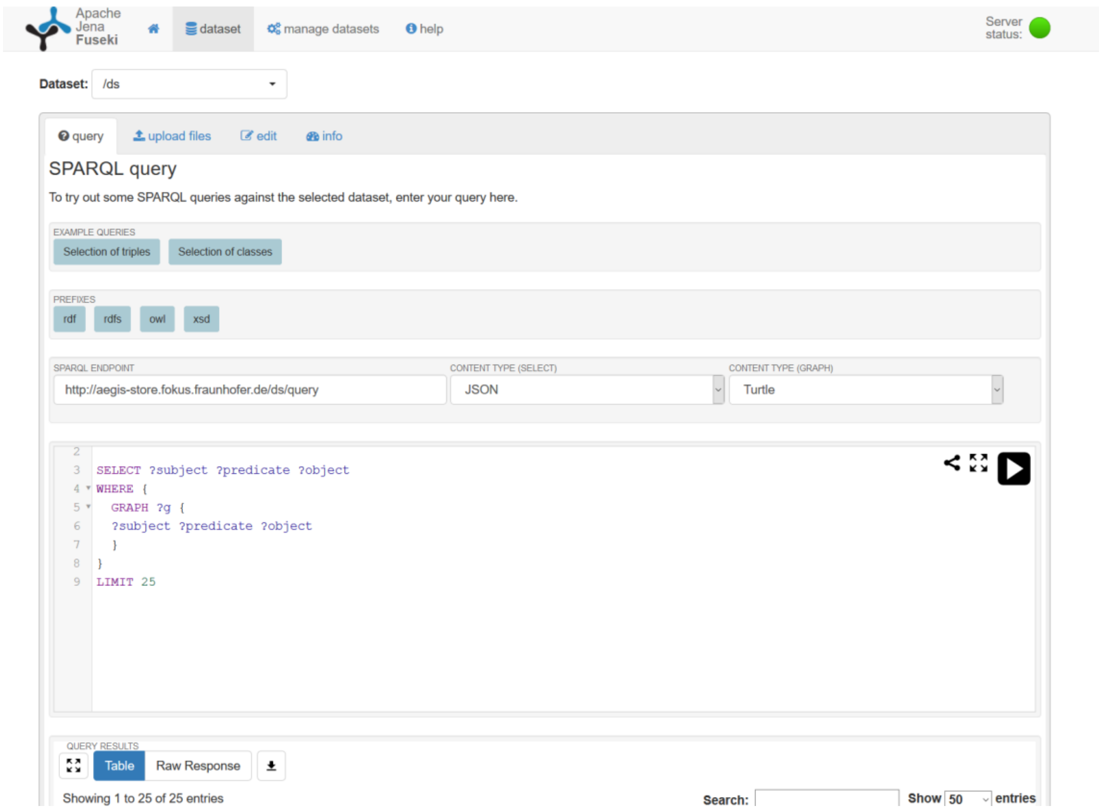
The Metadata Service (AEGIS Linked Data Store in D4.1) is responsible for storing the metadata associated with a particular dataset within the AEGIS platform. This metadata poses the foundation of the processing of the data within the AEGIS platform. It is based on the AEGIS ontology and vocabulary. For the second release, the component was further developed, enhanced and better integrated.

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 18 shows the SPARQL interface of Fuseki.



Dataset: /ds

SPARQL query

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

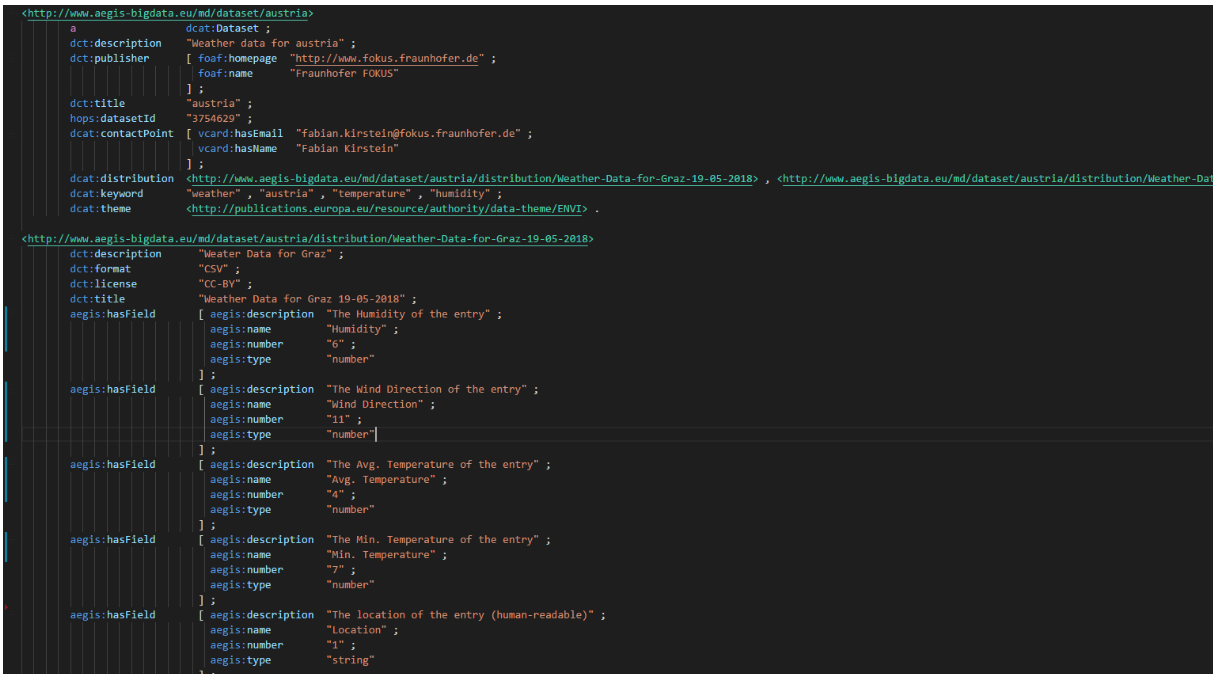
SPARQL ENDPOINT: http://aegis-store.fokus.fraunhofer.de/ds/query
CONTENT TYPE (SELECT): JSON
CONTENT TYPE (GRAPH): Turtle

```

SELECT ?subject ?predicate ?object
WHERE {
  GRAPH ?g {
  ?subject ?predicate ?object
  }
}
LIMIT 25
```

Showing 1 to 25 of 25 entries

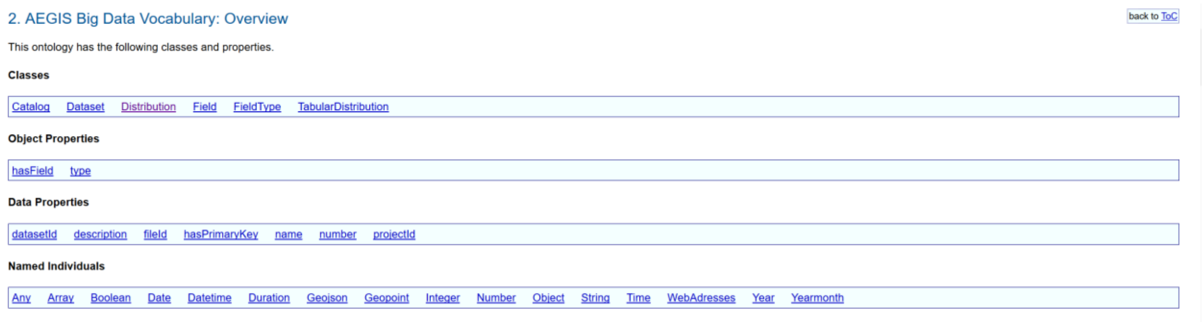
Figure 18: SPARQL User Interface of Fuseki

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 19 shows an example for the metadata stored in the triplestore. In Figure 20 an extract of the AEGIS ontology documentation can be seen.



```
<http://www.aegis-bigdata.eu/md/dataset/austria>
        a                 dcat:Dataset ;
        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-Dat
        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     "Weater 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 19: AEGIS Linked Data Example



2. AEGIS Big Data Vocabulary: Overview

back to ToC

This ontology has the following classes and properties.

Classes

Catalog Dataset Distribution Field FieldType TabularDistribution

Object Properties

hasField type

Data Properties

datasetId description fileld hasPrimaryKey name number projectId

Named Individuals

Any Array Boolean Date Datetime Duration Geojson Geopoint Integer Number Object String Time WebAdresses Year Yearmonth

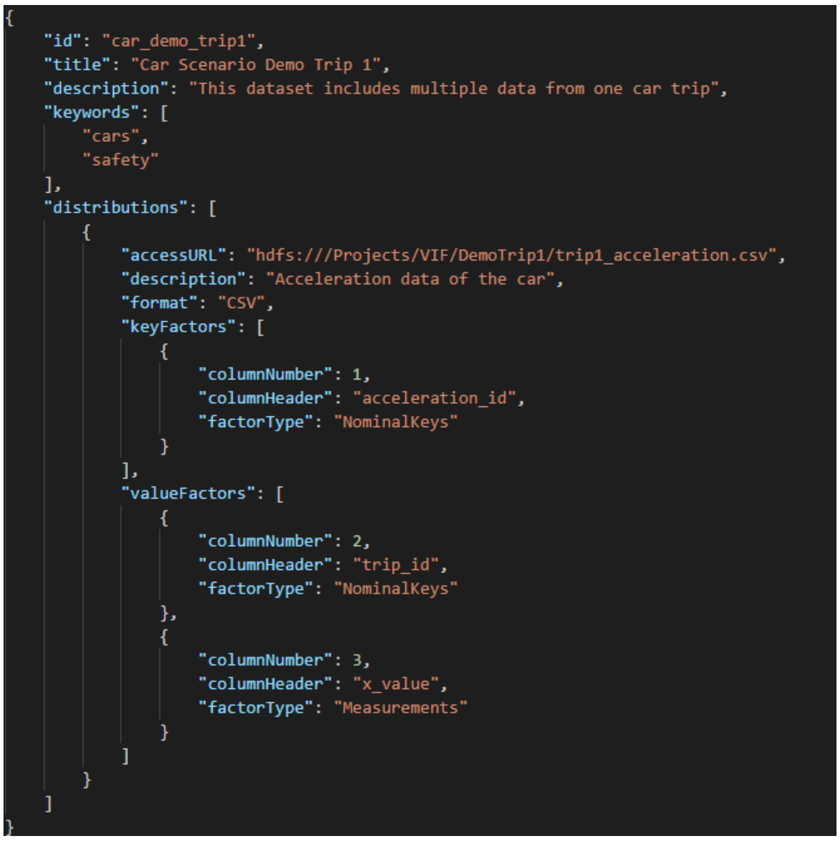
Figure 20: Extract of the AEGIS Ontology documentation

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 21 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 21: 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.

Since the metadata and the actual data are store in different service a synchronisation mechanism will be integrated, ensuring that for each metadata record the respective data is available and vice versa. This will be done by implementing hooks into the AEGIS core platform Hopsworks, which fire events to the metadata service whenever data is modified or deleted. In addition, a garbage collection will be added to the metadata service, which constantly checks the availability of referenced data. In addition, the Metadata Service will also be integrated into the Data Harvester, where each data export will trigger the creation of corresponding metadata.

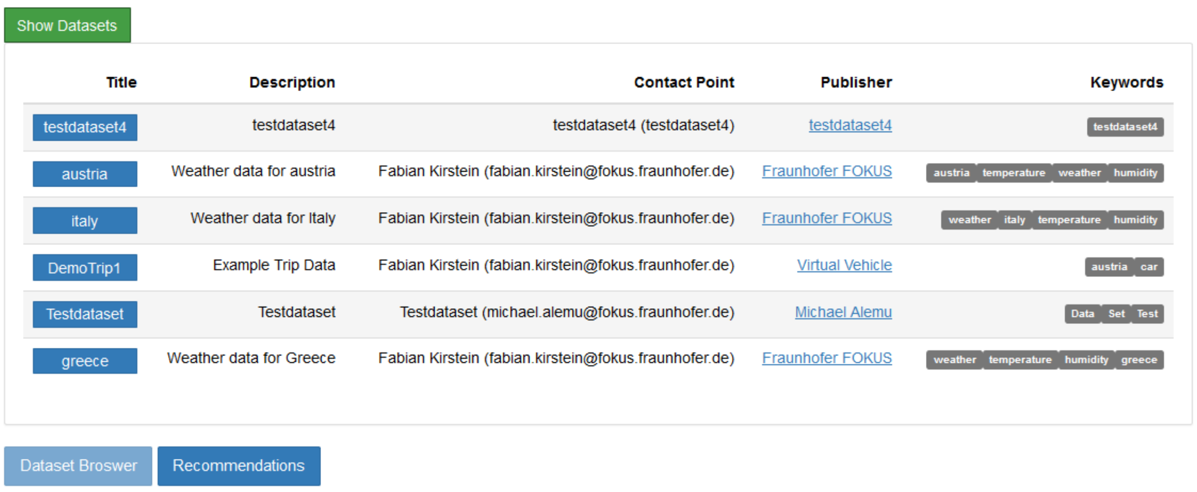
2.8. Query Builder

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, 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, Query Builder was implemented in the form of a note inside the Apache Zeppelin notebook, using mainly PySpark for the data manipulation and Javascript and Angular JS for the user interface. In the next version, as also explained in D3.3, the tool switched to the Jupyter Notebook as more appropriate. The tool, potentially in slightly different flavours (to allow for more customization of the data manipulation processes) will directly accessible inside every newly created project through the Jupyter Notebook (although the old Zeppelin version which was documented in D4.1 remains accessible).

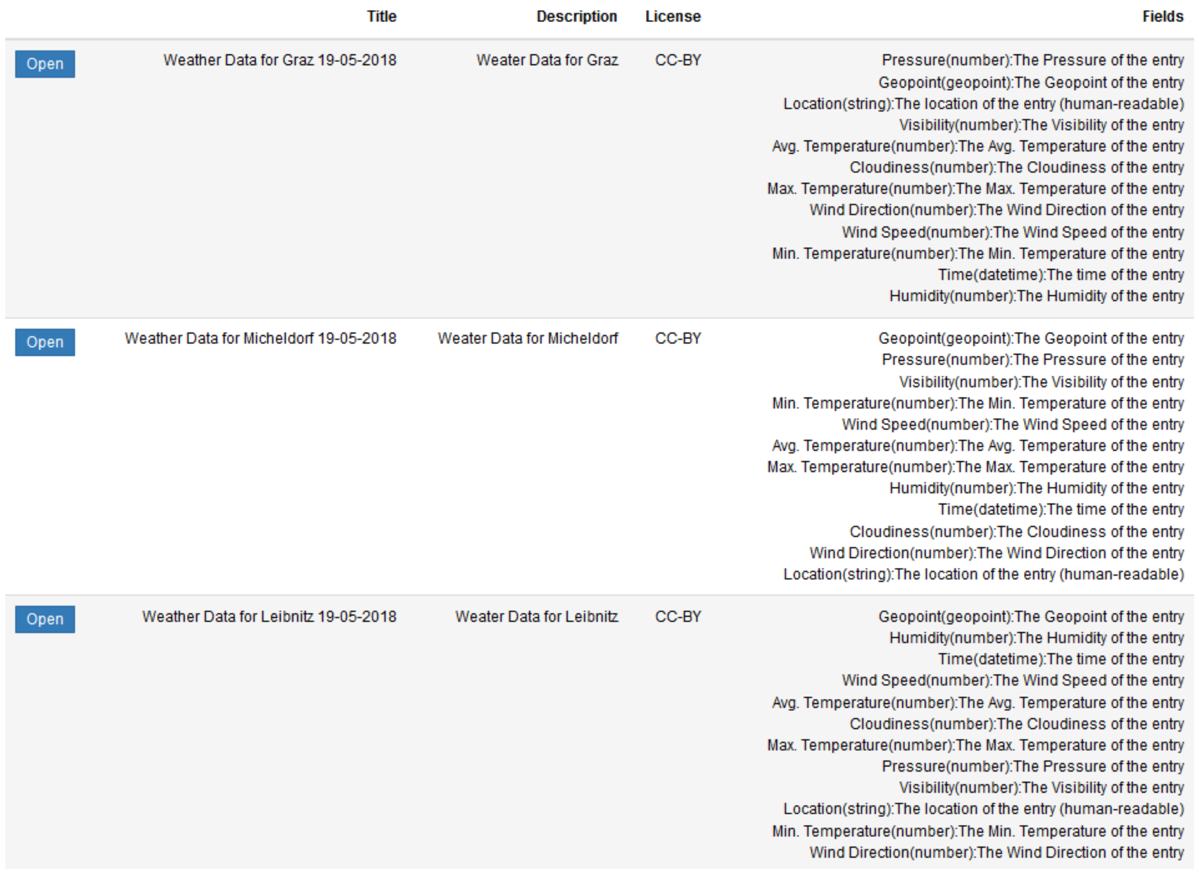
Query Builder leverages the metadata available for each file in the AEGIS system in order to provide its enhanced data selection and manipulation capabilities, i.e. enabling/disabling certain data merging and filtering options according to the data schema and also allowing the user to perform more targeted dataset exploration and retrieval based on the available metadata.



Title	Description	Contact Point	Publisher	Keywords
testdataset4	testdataset4	testdataset4 (testdataset4)	testdataset4	testdataset4
austria	Weather data for austria	Fabian Kirstein (fabian.kirstein@fokus.fraunhofer.de)	Fraunhofer FOKUS	austria temperature weather humidity
italy	Weather data for Italy	Fabian Kirstein (fabian.kirstein@fokus.fraunhofer.de)	Fraunhofer FOKUS	weather italy temperature humidity
DemoTrip1	Example Trip Data	Fabian Kirstein (fabian.kirstein@fokus.fraunhofer.de)	Virtual Vehicle	austria car
Testdataset	Testdataset	Testdataset (michael.alemu@fokus.fraunhofer.de)	Michael Alemu	Data Set Test
greece	Weather data for Greece	Fabian Kirstein (fabian.kirstein@fokus.fraunhofer.de)	Fraunhofer FOKUS	weather temperature humidity greece

Figure 22: Query Builder Data Browser (Jupyter version)

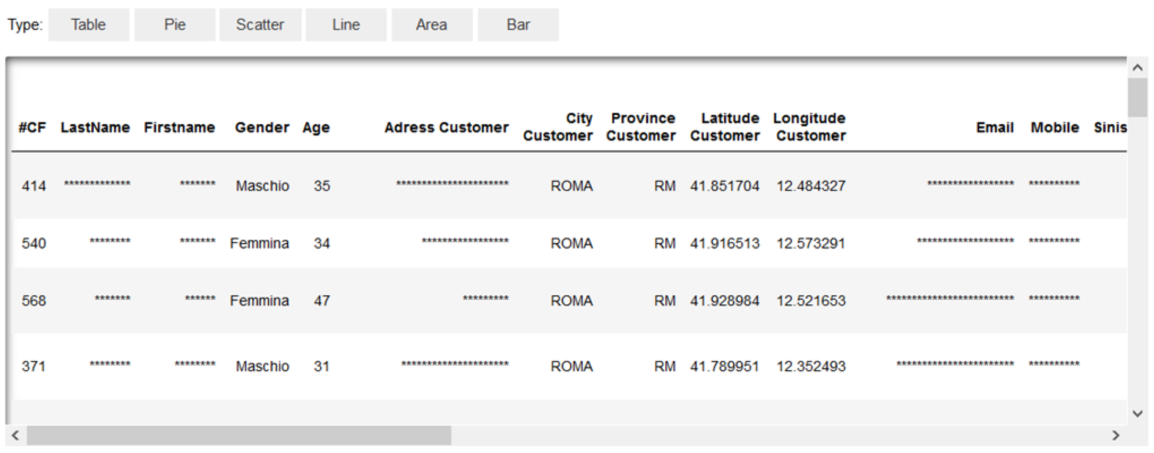
The new Query Builder version is fully integrated with the Metadata Service, hence the available datasets and corresponding information are taken from the service and presented to the user (Figure 22). The user can initially select an interesting dataset and then browse its files in order to choose which one to load.



	Title	Description	License	Fields
Open	Weather Data for Graz 19-05-2018	Weater Data for Graz	CC-BY	Pressure(number):The Pressure of the entry Geopoint(geopoint):The Geopoint of the entry Location(string):The location of the entry (human-readable) Visibility(number):The Visibility of the entry Avg. Temperature(number):The Avg. Temperature of the entry Cloudiness(number):The Cloudiness of the entry Max. Temperature(number):The Max. Temperature of the entry Wind Direction(number):The Wind Direction of the entry Wind Speed(number):The Wind Speed of the entry Min. Temperature(number):The Min. Temperature of the entry Time(datetime):The time of the entry Humidity(number):The Humidity of the entry
Open	Weather Data for Micheldorf 19-05-2018	Weater Data for Micheldorf	CC-BY	Geopoint(geopoint):The Geopoint of the entry Pressure(number):The Pressure of the entry Visibility(number):The Visibility of the entry Min. Temperature(number):The Min. Temperature of the entry Wind Speed(number):The Wind Speed of the entry Avg. Temperature(number):The Avg. Temperature of the entry Max. Temperature(number):The Max. Temperature of the entry Humidity(number):The Humidity of the entry Time(datetime):The time of the entry Cloudiness(number):The Cloudiness of the entry Wind Direction(number):The Wind Direction of the entry Location(string):The location of the entry (human-readable)
Open	Weather Data for Leibnitz 19-05-2018	Weater Data for Leibnitz	CC-BY	Geopoint(geopoint):The Geopoint of the entry Humidity(number):The Humidity of the entry Time(datetime):The time of the entry Wind Speed(number):The Wind Speed of the entry Avg. Temperature(number):The Avg. Temperature of the entry Cloudiness(number):The Cloudiness of the entry Max. Temperature(number):The Max. Temperature of the entry Pressure(number):The Pressure of the entry Visibility(number):The Visibility of the entry Location(string):The location of the entry (human-readable) Min. Temperature(number):The Min. Temperature of the entry Wind Direction(number):The Wind Direction of the entry

Figure 23: Query Builder available files inside a dataset

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.



#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 24: Query Builder Temp dataset preview

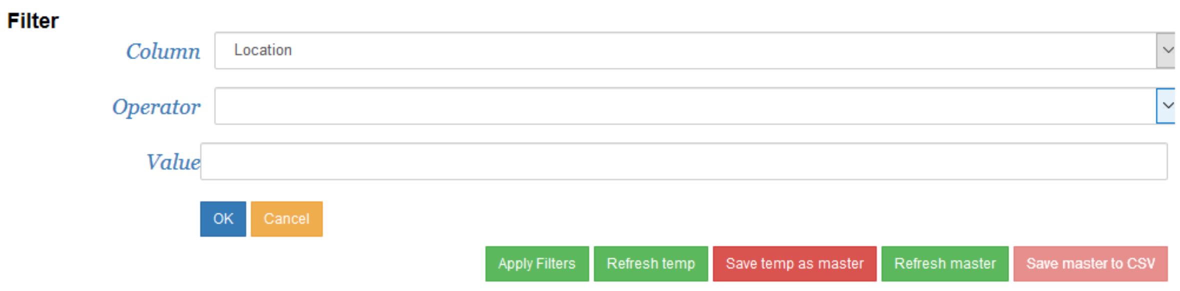


Figure 25: Query Builder data filters view

A number of filters and data processing methods is 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. 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 D3.3, 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.

The Visualiser through its intuitive and easy-to-use user interface is offering a variety of visualisation formats which spans from simple static charts to interactive charts with multiple layers of information and several customisation options. 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

As documented also in the deliverable D3.3, the Visualiser is implemented as a predefined Jupyter[8] notebook. Jupyter is a multipurpose interactive web-based notebook service, which is already integrated in the AEGIS platform as part of the AEGIS integrated services, that facilitates the execution of data analysis and data visualisation, supporting a variety of programming languages and integration with data processing frameworks. In addition to Jupyter, two Python libraries were utilised, namely the Folium[9] and the highcharts[10] 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.

Within the AEGIS platform the Visualiser can be accessed through Jupyter, that is integrated as a service within the AEGIS Front End. The implementation of the execution workflow of the Visualiser component, as documented in deliverable D3.3, is explained in the following steps:

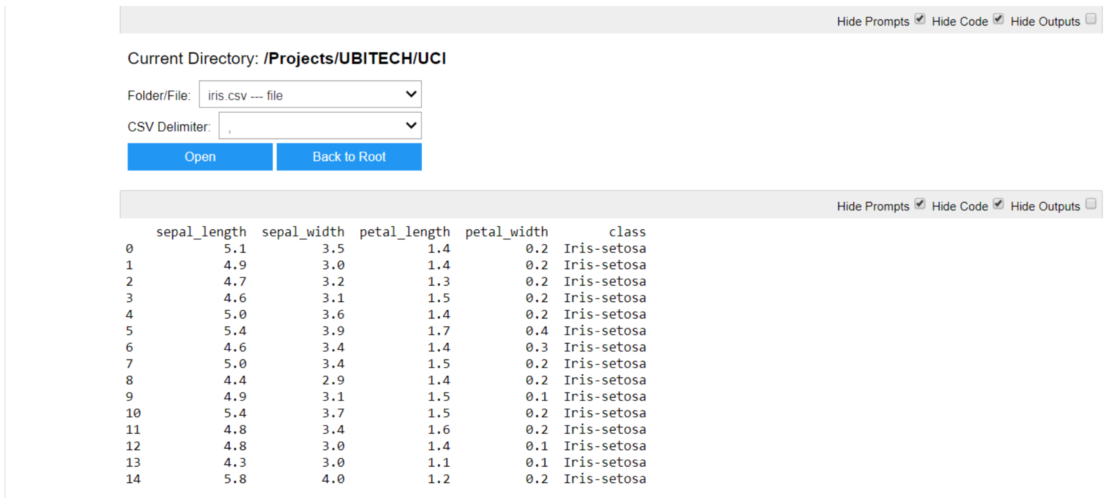
1. At first, when the Visualiser is loaded as an interactive notebook the user is presented with a list of options in order to define the dataset that will be utilised for the

[8] http://jupyter.org/

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

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

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 26).



Current Directory: **/Projects/UBITECH/UCI**

Folder/File: iris.csv --- file

CSV Delimiter: ,

```
    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 26: Visualiser - dataset selection

2. In the next step, the user is presented with the list of available visualisation types (Figure 27). 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 28.

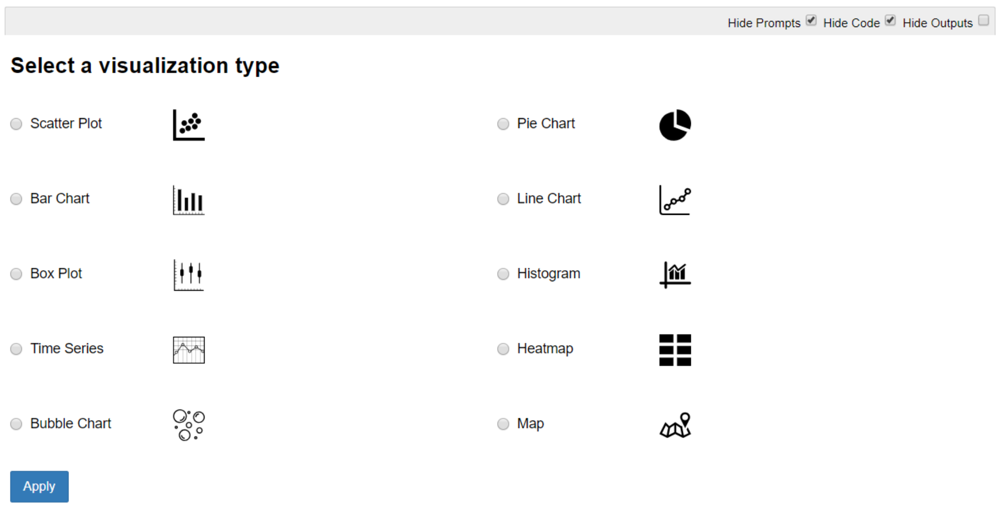
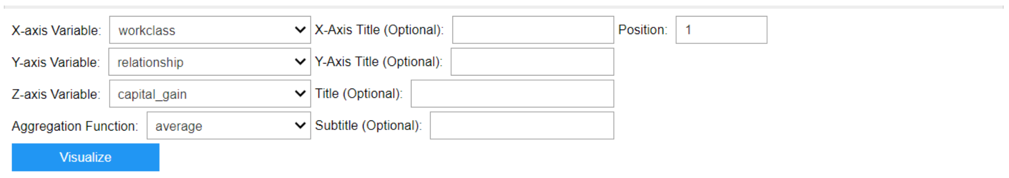


Figure 27: Visualiser - visualisation type selection



X-axis Variable:	workclass	X-Axis Title (Optional):		Position:	1
Y-axis Variable:	relationship	Y-Axis Title (Optional):			
Z-axis Variable:	capital_gain	Title (Optional):			
Aggregation Function:	average	Subtitle (Optional):			

Figure 28: 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.

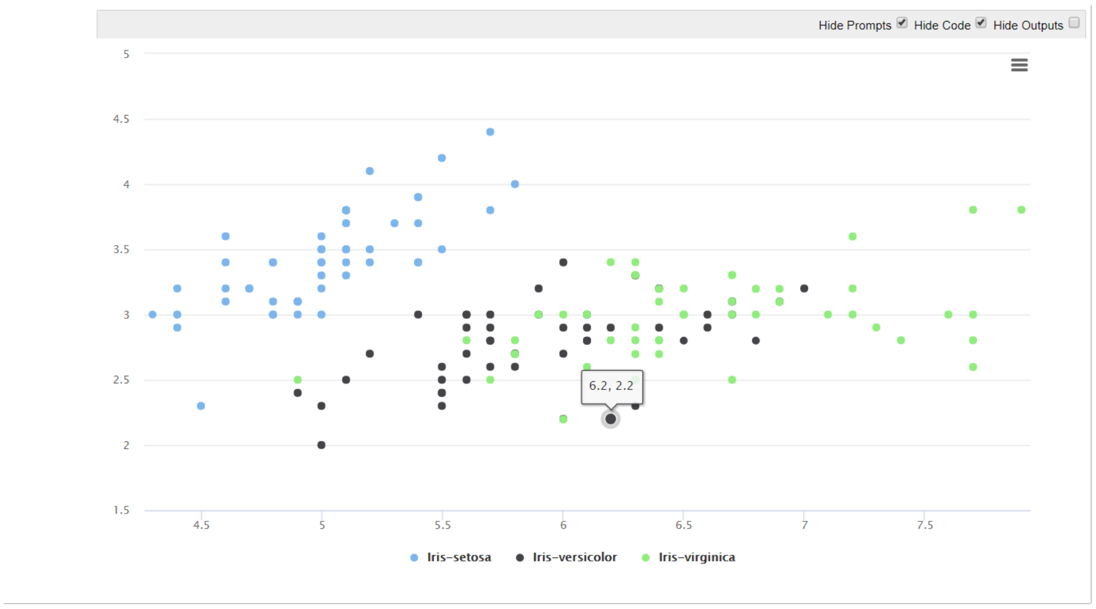


Figure 29: Visualiser – scatter plot

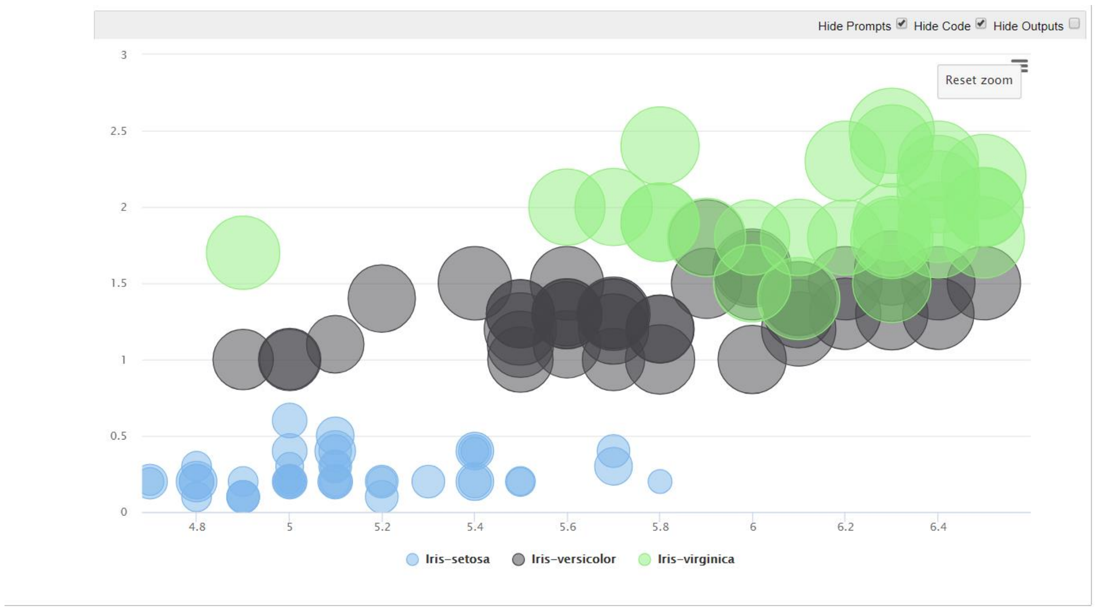


Figure 30: Visualiser - bubble chart

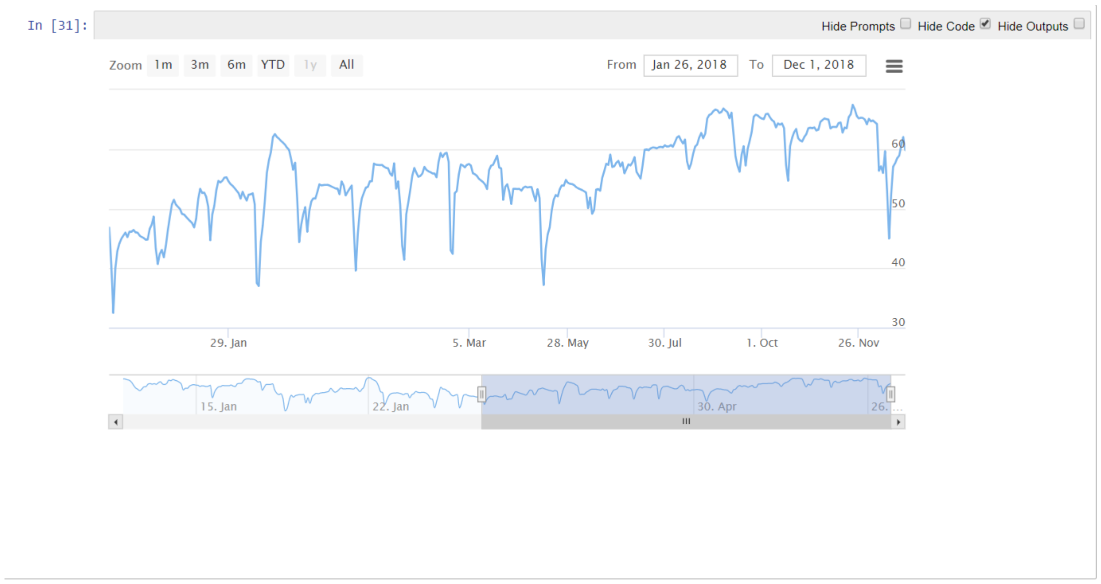


Figure 31: Visualiser - time series

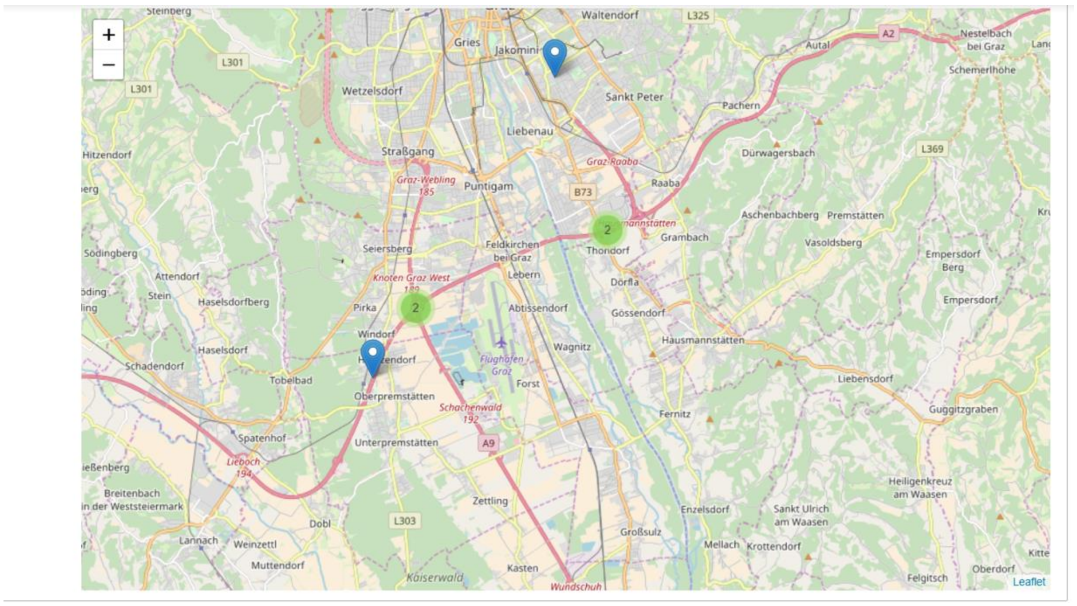


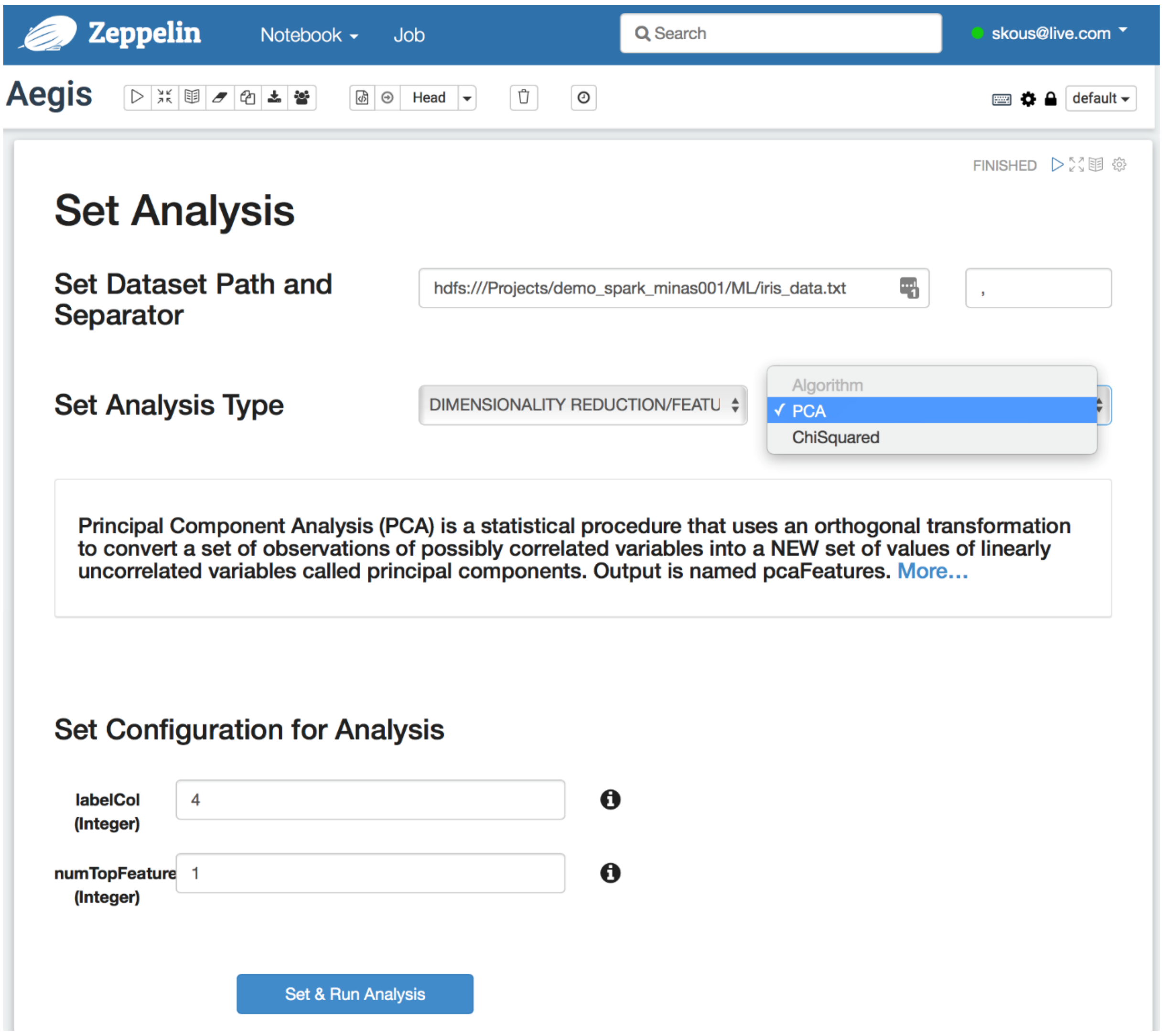
Figure 32: Visualiser - map

2.10. Algorithm Execution Container

The Algorithm Execution Container of the 1st AEGIS integrated prototype platform version is an interface for data analysis that resides on top of Zeppelin which is one of the most popular notebooks used by data analysts. The overall concept of this module is to accelerate analysis execution by simplifying the steps that data analysts perform, through eliminating the need to author code directly into the notebook.

The current implementation of Algorithm Execution Container is based on Angular JS and Pythion. In terms of algorithms, the container exploits the MLlib machine learning library and exposes 16 different algorithms, which are grouped in 5 different algorithmic families, as shown in the screenshot below.

Upon launch of the container, the Spark interpreter of the AEGIS platform is fired up to power the code that needs to be executed by the Zeppelin notebook. The container takes as input a dataset which has to be formatted accordingly to be ready to be used by the selected algorithm. A point to the specific URL of the input dataset is provided, which has to be a dataset stored in the AEGIS Data Store and should be accessible to the user that is performing the analysis.



Set Analysis

Set Dataset Path and Separator: hdfs:///Projects/demo_spark_minas001/ML/iris_data.txt ,

Set Analysis Type: DIMENSIONALITY REDUCTION/FEATU

Algorithm
✓ PCA
ChiSquared

Principal Component Analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a NEW set of values of linearly uncorrelated variables called principal components. Output is named pcaFeatures. More...

Set Configuration for Analysis

labelCol (Integer): 4

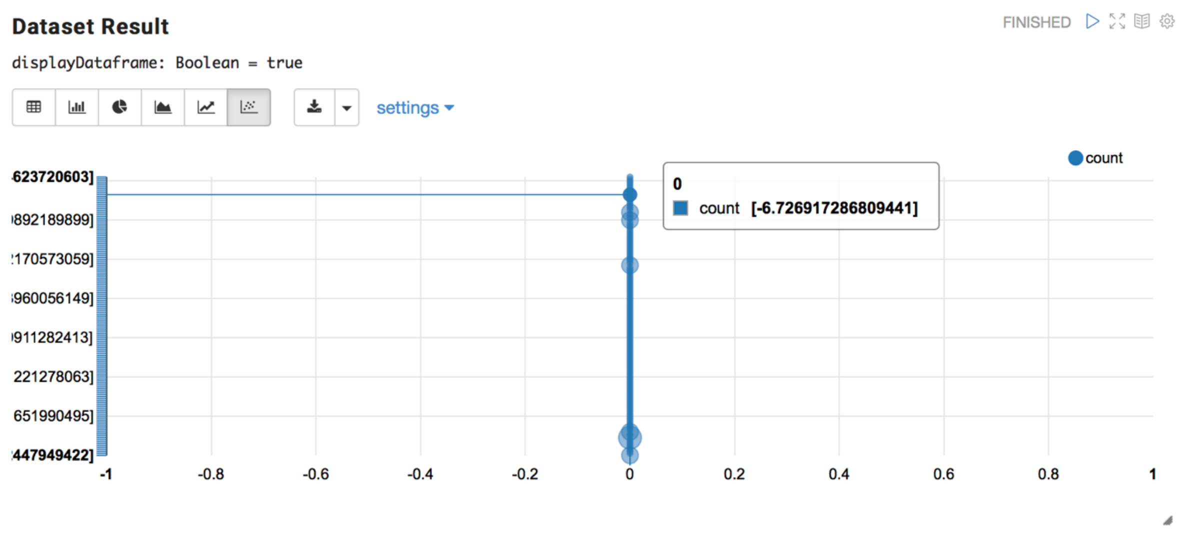
numTopFeature (Integer): 1

Set & Run Analysis

Figure 33: Algorithm Execution Container – Algorithmic Families and Algorithms

The user is then able to select the algorithmic family and the specific algorithm to run, and he is presented with the parameters that are relevant to the analysis, so he can provide his preferences.

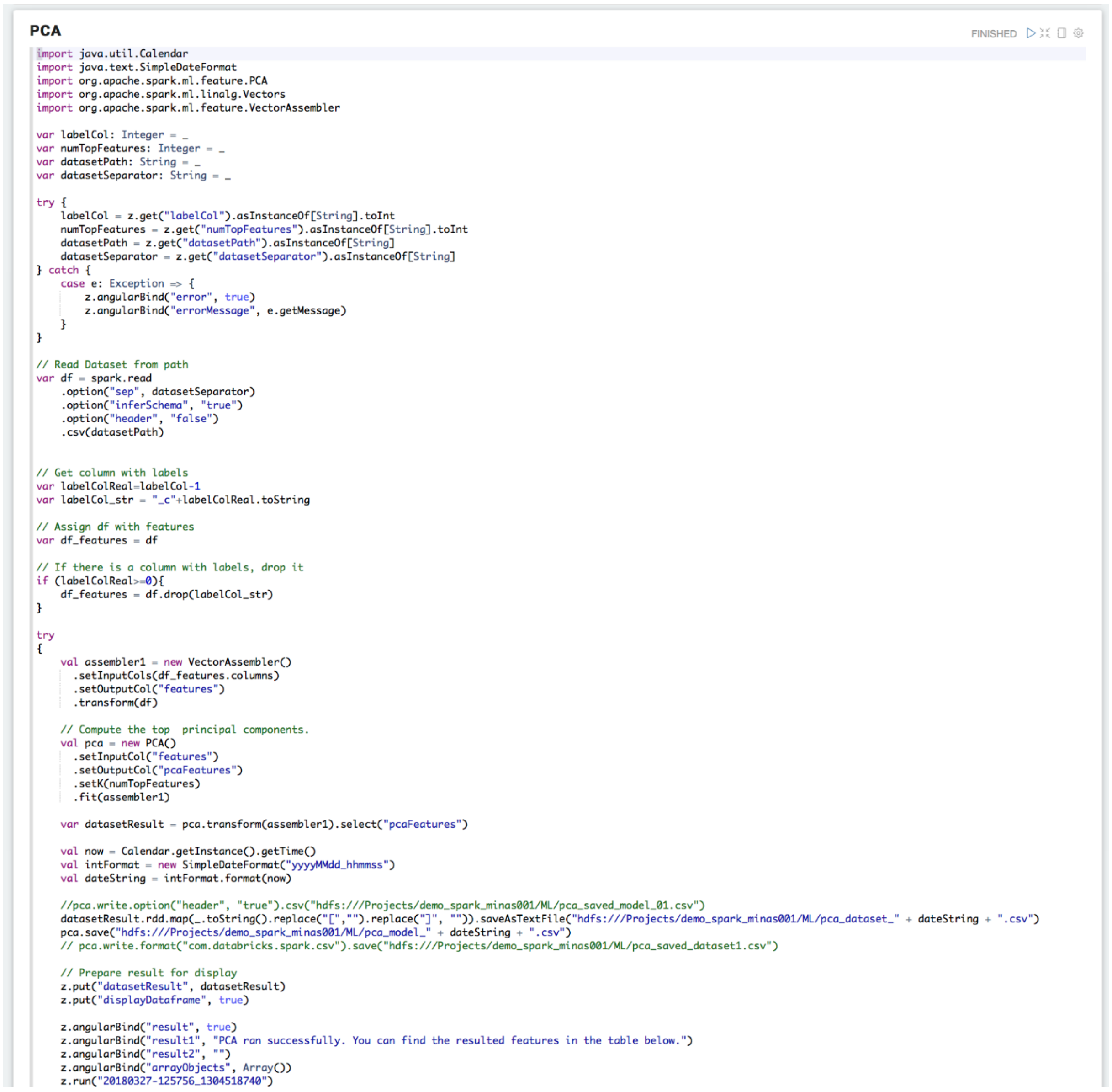
Upon execution, the necessary Zeppelin paragraphs are executed in the background and the user is presented with the final result of his analysis. Simple, predefined visualisation options are also provided by the Zeppelin notebook, as seen in Figure 34.


Dataset Result

displayDataframe: Boolean = true

Figure 34: Sample Analytics Output Predefined Visualisations

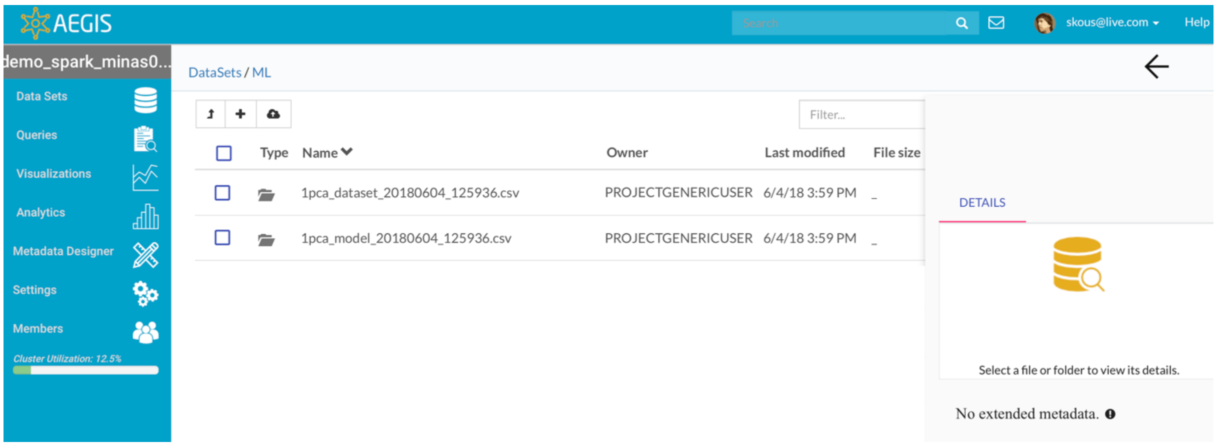
It needs to be mentioned, that in case an analyst is willing to run more complex and customised analyses, the module allows the direct edit of the underlying code, by simply choosing the appropriate notebook paragraph and editing it.



```
import java.util.Calendar
import java.text.SimpleDateFormat
import org.apache.spark.ml.feature.PCA
import org.apache.spark.ml.linalg.Vectors
import org.apache.spark.ml.feature.VectorAssembler

var labelCol: Integer = _
var numTopFeatures: Integer = _
var datasetPath: String = _
var datasetSeparator: String = _

try {
    labelCol = z.get("labelCol").asInstanceOf[String].toInt
    numTopFeatures = z.get("numTopFeatures").asInstanceOf[String].toInt
    datasetPath = z.get("datasetPath").asInstanceOf[String]
    datasetSeparator = z.get("datasetSeparator").asInstanceOf[String]
} catch {
    case e: Exception => {
        z.angularBind("error", true)
        z.angularBind("errorMessage", e.getMessage)
    }
}

// Read Dataset from path
var df = spark.read
    .option("sep", datasetSeparator)
    .option("inferSchema", "true")
    .option("header", "false")
    .csv(datasetPath)


// Get column with labels
var labelColReal=labelCol-1
var labelCol_str = "_c"+labelColReal.toString

// Assign df with features
var df_features = df

// If there is a column with labels, drop it
if (labelColReal>=0){
    df_features = df.drop(labelCol_str)
}

try
{
    val assembler1 = new VectorAssembler()
      .setInputCols(df_features.columns)
      .setOutputCol("features")
      .transform(df)

    // Compute the top  principal components.
    val pca = new PCA()
      .setInputCol("features")
      .setOutputCol("pcaFeatures")
      .setK(numTopFeatures)
      .fit(assembler1)

    var datasetResult = pca.transform(assembler1).select("pcaFeatures")

    val now = Calendar.getInstance().getTime()
    val intFormat = new SimpleDateFormat("yyyyMMdd_hhmmss")
    val dateString = intFormat.format(now)

    //pca.write.option("header", "true").csv("hdfs:///Projects/demo_spark_minas001/ML/pca_saved_model_01.csv")
    datasetResult.rdd.map(_.toString().replace("[","").replace("]", "")).saveAsTextFile("hdfs:///Projects/demo_spark_minas001/ML/pca_dataset_" + dateString + ".csv")
    pca.save("hdfs:///Projects/demo_spark_minas001/ML/pca_model_" + dateString + ".csv")
    // pca.write.format("com.databricks.spark.csv").save("hdfs:///Projects/demo_spark_minas001/ML/pca_saved_dataset1.csv")

    // Prepare result for display
    z.put("datasetResult", datasetResult)
    z.put("displayDataframe", true)

    z.angularBind("result", true)
    z.angularBind("result1", "PCA ran successfully. You can find the resulted features in the table below.")
    z.angularBind("result2", "")
    z.angularBind("arrayObjects", Array())
    z.run("20180327-125756_1304518740")
```

Figure 35: Predefined NoteBook paragraphs for the Algorithm

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



Type	Name	Owner	Last modified	File size
	1pca_dataset_20180604_125936.csv	PROJECTGENERICUSER	6/4/18 3:59 PM	_
	1pca_model_20180604_125936.csv	PROJECTGENERICUSER	6/4/18 3:59 PM	_

Figure 36: Analysis Output and Model stored in AEGIS

According to the project development plan, the next version of the Algorithm Execution Container is going to be deployed over the Jupiter notebook. This will allow the coverage of the two most popular notebook implementations available, while it will allow for the creation of a unified analytics flow within the platform by interconnecting the container to the Query Builder and the Visualiser.

2.11. AEGIS Front-end

Following the look and feel of the AEGIS institutional web site[11] an updated GUI/front-end for the AEGIS second integrated prototype has been developed, on top of Hopsworks, using as main technologies HTML and AngularJS[12].

[11] https://www.aegis-bigdata.eu/

[12] https://angularjs.org/

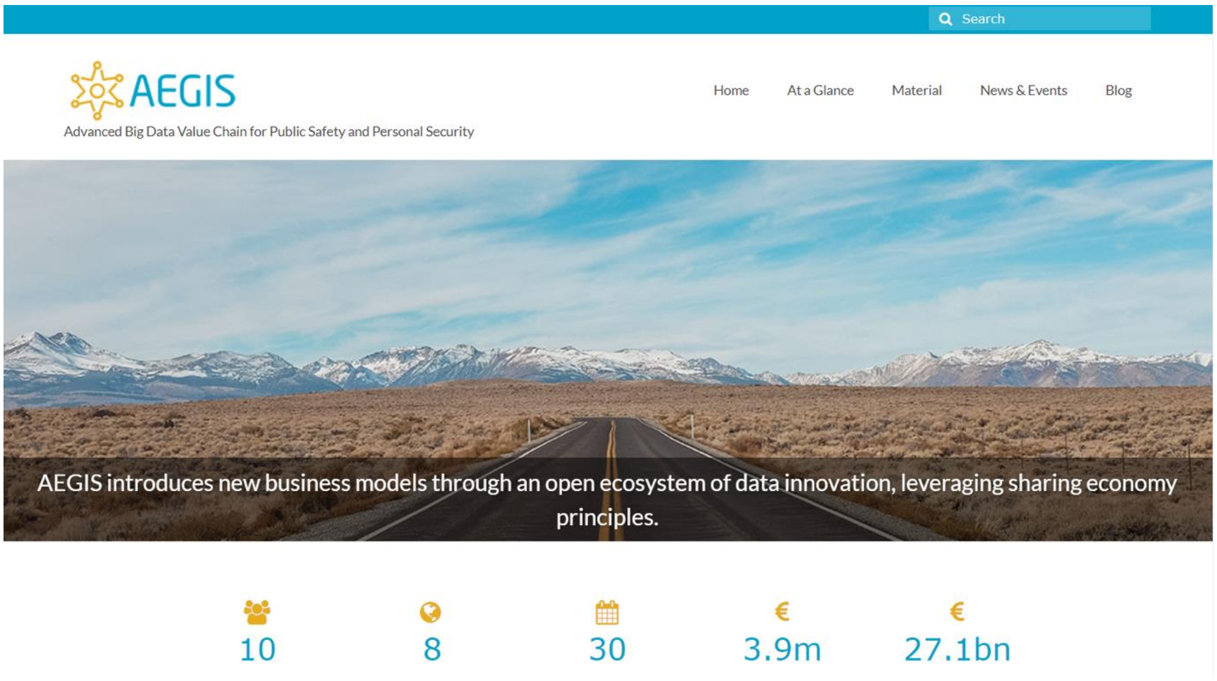
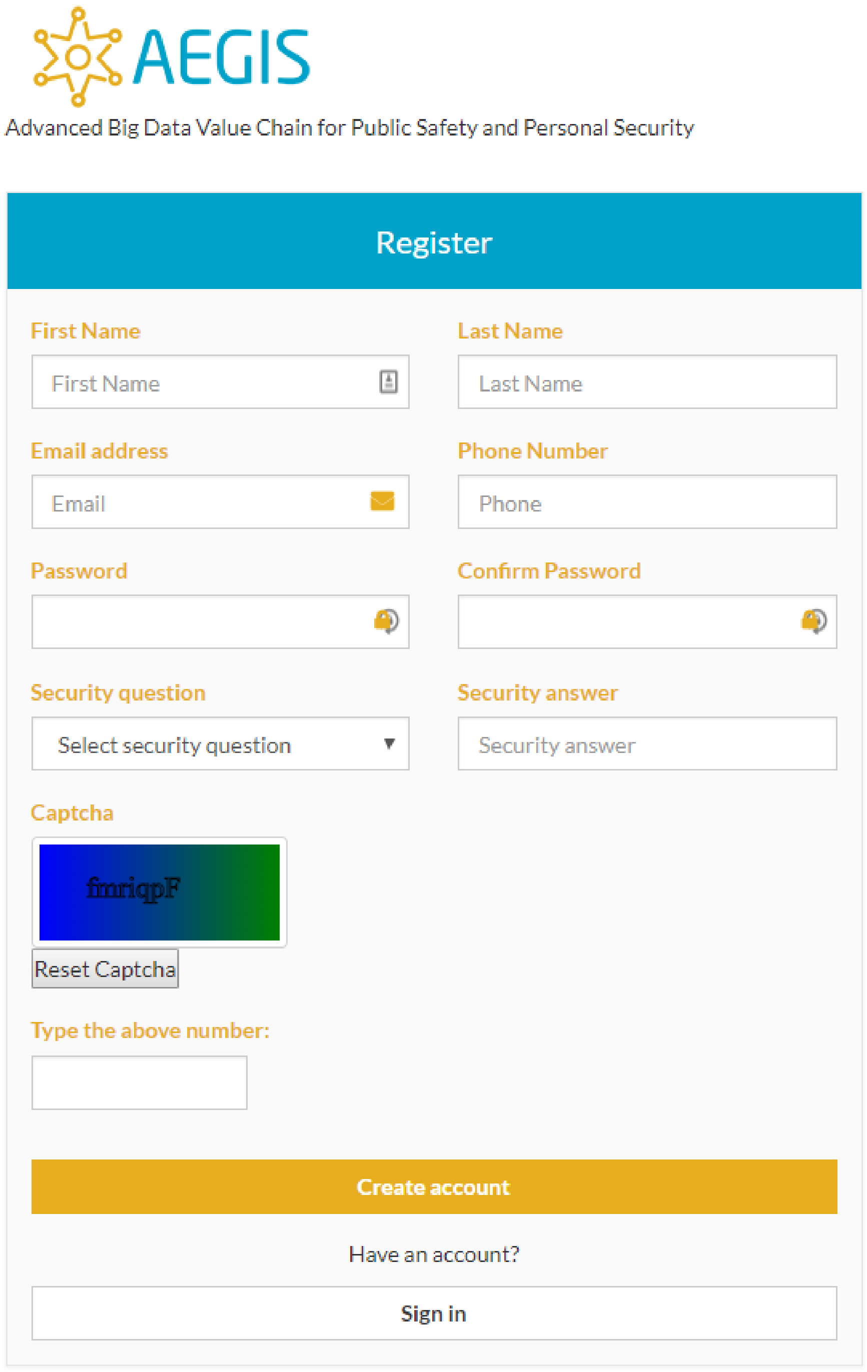

Search
AEGIS
Advanced Big Data Value Chain for Public Safety and Personal Security
Home
At a Glance
Material
News & Events
Blog
AEGIS introduces new business models through an open ecosystem of data innovation, leveraging sharing economy principles.
10
8
30
3.9m
27.1bn

Figure 37: AEGIS web site

The first step of this component features the user account registration (Figure 38), followed by a login/logout mechanism. The registration module has been enhanced with a captcha mechanism and features security questions/answers. Once the user is logged into the platform, the main page shows three tabs with the options to filter the AEGIS content by the popular assets, the latest assets, and the offers (to be fully implemented in the following version of the platform). On the right side of the page, it is showing the list of the currently available projects (personal and public), as well as the button to creating a new one, with the option to specify its members. After a project has been selected, a new page shows the related activity history and the main menu on the left side, including the following items: Get Started, Assets, Project Datasets, Query Builder, Analytics, Visualiser, Jupyter, Zeppelin, Kafka, Jobs, 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-D3.3-*AEGIS Components Microservices and APIs Design.*


AEGIS
Advanced Big Data Value Chain for Public Safety and Personal Security

Register

First Name	Last Name
First Name	Last Name
Email address	Phone Number
Email	Phone
Password	Confirm Password
Security question	Security answer
Select security question ▼	Security answer

Captcha

fmriqpF

Reset Captcha

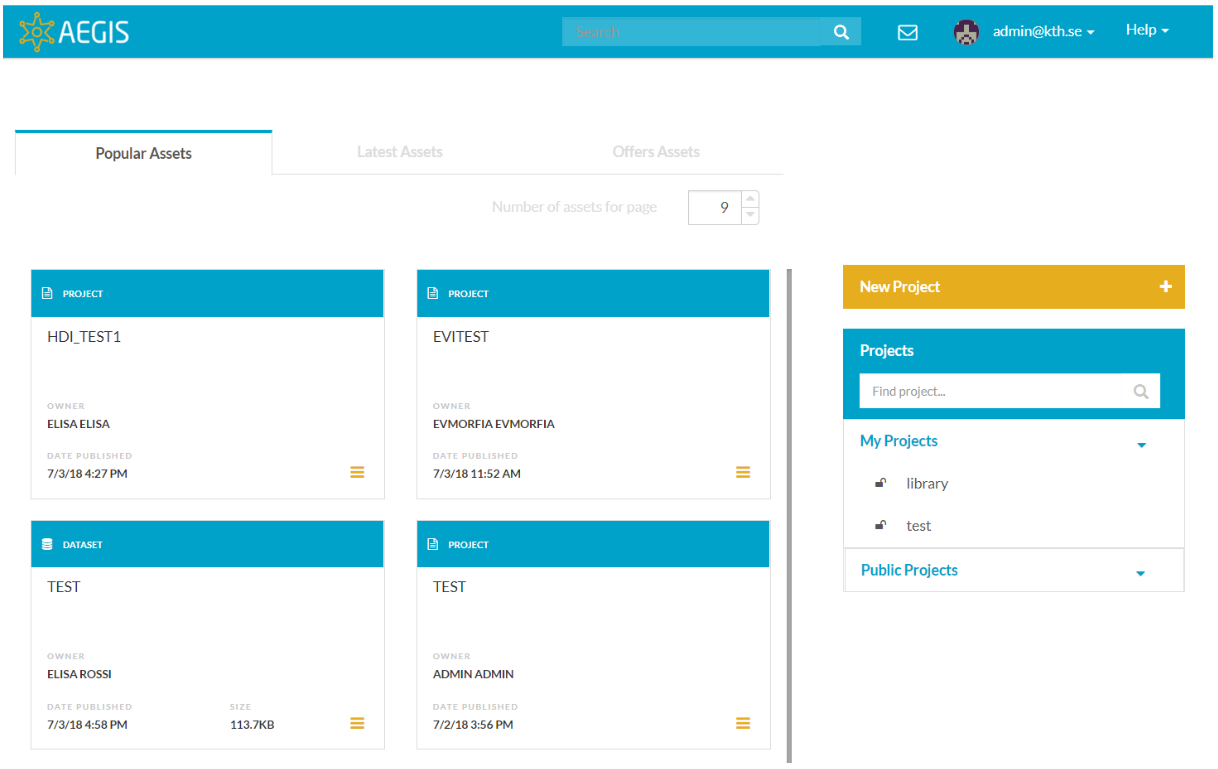
Type the above number:

Create account

Have an account?

Sign in

Figure 38: Registration page



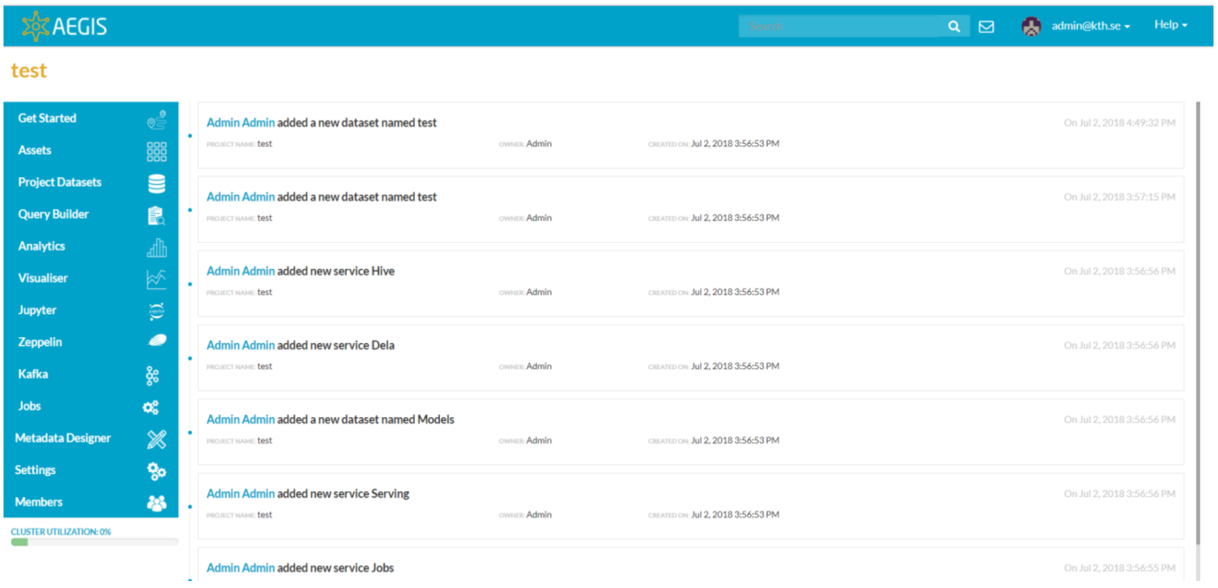
PROJECT
HDI_TEST1
OWNER
ELISA ELISA
DATE PUBLISHED
7/3/18 4:27 PM

PROJECT
EVITEST
OWNER
EVMORFIA EVMORFIA
DATE PUBLISHED
7/3/18 11:52 AM

DATASET
TEST
OWNER
ELISA ROSSI
DATE PUBLISHED
7/3/18 4:58 PM
SIZE
113.7KB

PROJECT
TEST
OWNER
ADMIN ADMIN
DATE PUBLISHED
7/2/18 3:56 PM

Figure 39: Home page



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

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

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

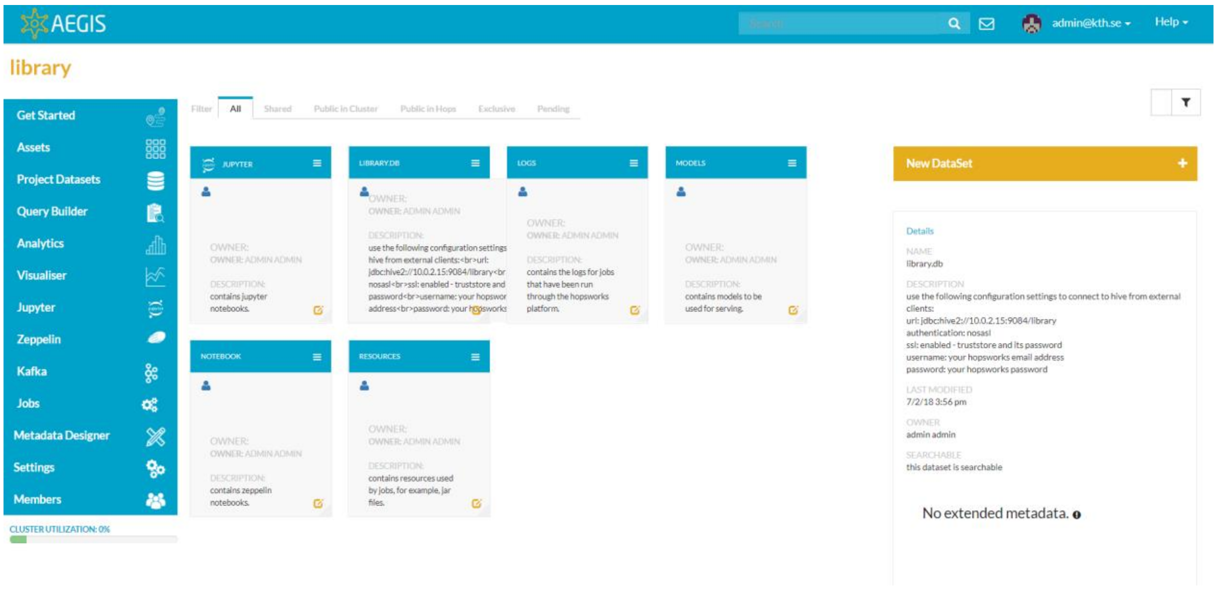
Admin Admin added new service Dela
PROJECT NAME: test OWNER: Admin CREATED ON: Jul 2, 2018 3:56:53 PM
On Jul 2, 2018 3:56:56 PM

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

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

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

Figure 40: Project Home page



JUPYTER
OWNER:
OWNER: ADMIN ADMIN
DESCRIPTION:
contains jupyter notebooks.

LIBRARY.DB
OWNER:
OWNER: ADMIN ADMIN
DESCRIPTION:
use the following configuration settings hive from external clients:
url: jdbc:hive2://10.0.2.15:9084/library<br nosasl
ssl: enabled - truststore and password
username: your hopswor address
password: your hopsworks

LOGS
OWNER:
OWNER: ADMIN ADMIN
DESCRIPTION:
contains the logs for jobs that have been run through the hopsworks platform.

MODELS
OWNER:
OWNER: ADMIN ADMIN
DESCRIPTION:
contains models to be used for serving.

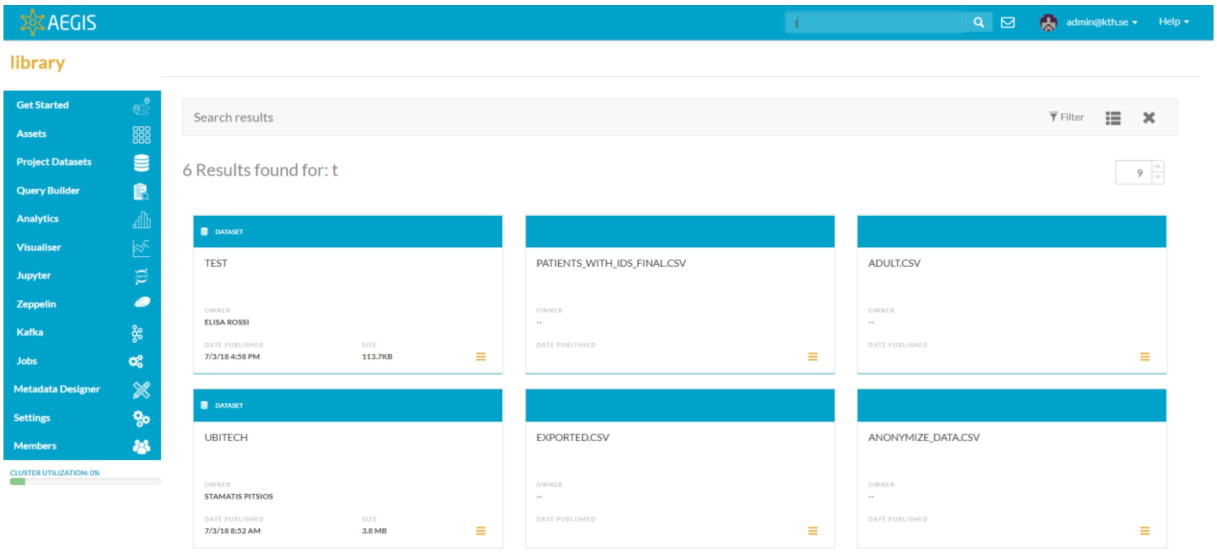
NOTEBOOK
OWNER:
OWNER: ADMIN ADMIN
DESCRIPTION:
contains zeppelin notebooks.

RESOURCES
OWNER:
OWNER: ADMIN ADMIN
DESCRIPTION:
contains resources used by jobs, for example, jar files.

Details
NAME
library.db
DESCRIPTION
use the following configuration settings to connect to hive from external clients:
url: jdbc:hive2://10.0.2.15:9084/library
authentication: nosasl
ssl: enabled - truststore and its password
username: your hopsworks email address
password: your hopsworks password
LAST MODIFIED
7/2/18 3:56 pm
OWNER
admin admin
SEARCHABLE
this dataset is searchable

No extended metadata.

Figure 41: Project Datasets



6 Results found for: t

TEST
OWNER
ELISA ROSSI
DATE PUBLISHED 7/3/18 4:58 PM
SIZE 113.7KB

PATIENTS_WITH_IDS_FINAL.CSV
OWNER
--
DATE PUBLISHED

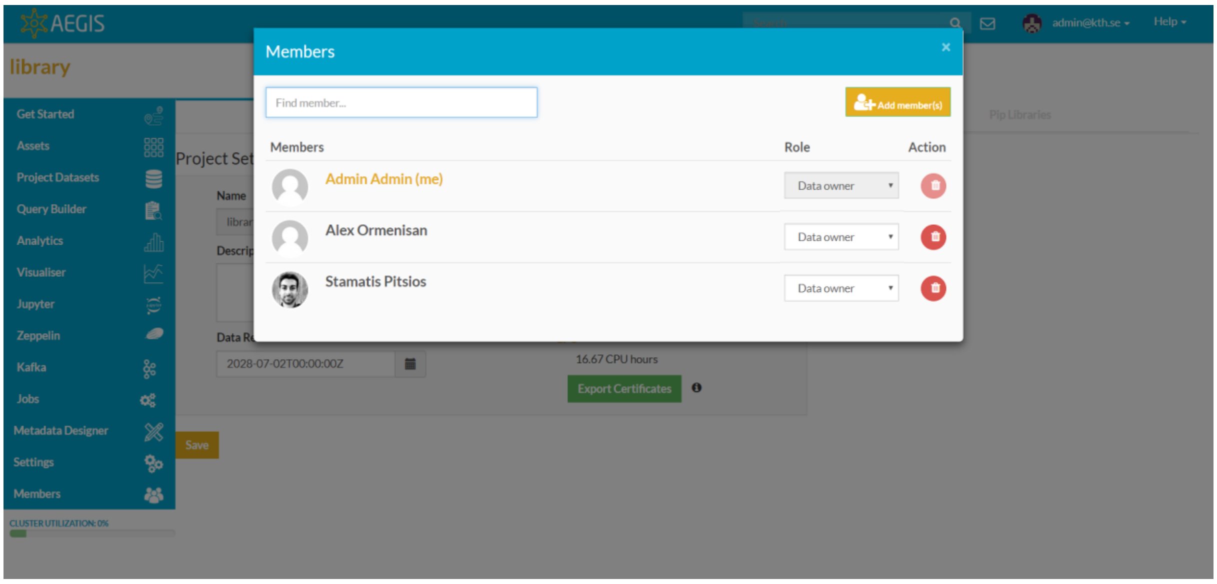
ADULT.CSV
OWNER
--
DATE PUBLISHED

UBITECH
OWNER
STAMATIS PITSIOS
DATE PUBLISHED 7/3/18 8:52 AM
SIZE 3.8 MB

EXPORTED.CSV
OWNER
--
DATE PUBLISHED

ANONYMIZE_DATA.CSV
OWNER
--
DATE PUBLISHED

Figure 42: Search results



Members	Role	Action
Admin Admin (me)	Data owner	
Alex Ormenisan	Data owner	
Stamatis Pitsios	Data owner	

Figure 43: Members

3. CONCLUSION

The document accompanies the second release of the AEGIS integrated prototype and provides information about the realization of the foreseen partially functional high fidelity software prototype connected to a deployed version of platform. The initial interface includes the basic UI/UX (updated from D4.1) for the users of the 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 M24, the prototype will be upgraded and refined by providing a software package of the core platform and its API, with corresponding supporting documentation on the deployment of each component and usage of the APIs.