H2020 - INDUSTRIAL LEADERSHIP - Leadership in enabling and industrial technologies - Information and Communication Technologies (ICT)

ICT-14-2016: Big Data PPP: cross-sectorial and cross-lingual data integration and experimentation

BIG DATA OCEAN

BigDataOcean

"Exploiting Oceans of Data for Maritime Applications"

D4.3 BigDataOcean Platform Architecture, Components Design and APIs – v2.00

Workpackage:	WP4 – BigDataOcean Service Requirements, User Stories, APIs definition, Architecture
Authors:	Konstantinos Perakis, Dimitrios Miltiadou, Stamatis Pitsios (UBITECH), Giannis Tsapelas, Spiros Mouzakitis, Panagiotis Kokkinakos, Dimitris Askounis (NTUA), Ioanna Lytra, Fabrizio Orlandi (UBONN), Konstantinos Chatzikokolakis (EXMILE)
Status:	Final
Date:	29/03/2018
Version:	1.00
Classification:	Public

BigDataOcean Project Profile

Grant Agreement No.: 732310

Acronym:	BigDataOcean
Title:	Exploiting Oceans of Data for Maritime Applications
URL:	http://www.bigdataocean.eu/site/
Start Date:	01/01/2017
Duration:	30 months

Partners

	National Technical University of Athens (NTUA), Decision Support Systems Laboratory, DSSLab Co-ordinator	Greece
	Exmile Solutions Limited (EXMILE)	United Kingdom
	Rheinische Friedrich-Wilhelms-Universitt Bonn (UBONN)	Germany
	Centro de Investigacao em Energia REN – State Grid, S.A. – R&D Nester (NESTER)	Portugal
	Hellenic Centre for Marine Research (HCMR)	Greece
	Ubitech Limited (UBITECH)	Cyprus
	Foinikas Shipping Company (FOINIKAS)	Greece
	Istituto Superiore Mario Boella (ISMB)	Italy
	Instituto de Desenvolvimento de Novas Tecnologias (UNINOVA)	Portugal
	Anonymi Naftiliaki Etaireia Kritis (ANEK)	Greece

Document History

Version	Date	Author (Partner)	Remarks
0.10	01/03/2018	Dimitrios Miltiadou (UBITECH)	ToC
0.20	05/03/2018	Dimitrios Miltiadou, Konstantinos Perakis (UBITECH)	Contributions to chapter 1, 2
0.3	14/03/2018	Dimitrios Miltiadou, Konstantinos Perakis, Stamatis Pitsios (UBITECH)	Contributions to chapter 2, 3
0.4	16/03/2018	Giannis Tsapelas, Spiros Mouzakitis, Panagiotis Kokkinakos, Dimitris Askounis (NTUA), Konstantinos Chatzikokolakis (EXMILE)	Contributions to chapter 3
0.5	19/03/2018	Ioanna Lytra, Fabrizio Orlandi (UBONN)	Contributions to chapter 3
0.6	23/03/2018	Dimitrios Miltiadou (UBITECH)	Consolidation of contributions, preparation of review ready version
0.7	27/03/2018	Fani Panagopoulou (ANEK)	Internal review
0.8	27/03/2018	Jose Ferreira (UNINOVA)	Internal review
1.0	28/03/2018	Dimitrios Miltiadou (UBITECH)	Final version

Executive Summary

The scope of D4.3 is to document the efforts undertaken within the context of all tasks of WP4, namely the Task 4.1 – Platform Technology Requirements and Integration Components Analysis, the Task 4.2 – User Stories from Pilots for Big Maritime Data Exploitation, the Task 4.3 – Components' and APIs' Definition and Design and the Task 4.4 – BigDataOcean Scalable Architecture Design.

The current document builds directly on top of the work reported within the context of D4.2, in particular the initial version of the conceptual architecture of the integrated platform and the definition of the platform's components. As the project evolved and the first mock-up version of the integrated platform was released, the feedback received from the project's pilot served as basis for the introduction of several optimisations and adjustments in the platform and the platform's components.

Towards this end, the primary objective of the current deliverable is to provide the updated documentation that will supplement the information documented in deliverable D4.2. In more details, the additional technical requirements that were elicited from the analysis of the feedback received are presented. These new technical requirements were translated into several optimisations and customisations in the high-level architecture of the platform with the aim of addressing the new stakeholders' needs. Thus, the new updated high-level architecture is presented focusing on the platform's functionalities, as well as the roles and responsibilities of each component within the platform in order to enable these functionalities. Furthermore, in the current document all essential updates on the documentation of the platform's components are presented in terms of design and specifications. For each component the addressed requirements and the functionalities developed are documented. Additionally, the technical interfaces that facilitate the implementation of the integrated platform are documented.

The design of the architecture of the BigDataOcean platform is a living process that will last until M26, as additional technical requirements will be collected and will be translated into further optimisations and customisations on the platform and the platform's components in the course of addressing new stakeholders' needs. Moreover, as the projects matures and new versions of the platform will be delivered additional feedback will be collected that will drive these new optimisations and customisations.

Table of Contents

List of Figures

List of Tables

Abbreviations

Abbreviation	Description
API	Application Programming Interface
CSV	Comma-separate values
D	Deliverable
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
M	Month
RDF	Resource Description Framework
SQL	Structured Query Language
TLS	Transport Layer Security
URI	Uniform Resource Identifier
WP	Workpackage
XML	Extensible Markup Language

1 Introduction

1.1 Objective of the deliverable

The scope of D4.3 is to document the efforts undertaken within the context of all tasks of WP4, namely the Task 4.1 – Platform Technology Requirements and Integration Components Analysis, the Task 4.2 – User Stories from Pilots for Big Maritime Data Exploitation, the Task 4.3 – Components' and APIs' Definition and Design and the Task 4.4 – BigDataOcean Scalable Architecture Design.

D4.3 is building directly on top of the outcomes and knowledge extracted by D4.2 where the main components of the BigDataOcean platform were defined and documented and also the first version of the conceptual architecture of the integrated platform was presented. The design of the components, where the characteristics and the functionalities of each component were defined, had driven the development activities of the components while the conceptual architecture, where all identified components were integrated, had driven the implementation and release of the first mock-up version of the BigDataOcean platform that was delivered in M14, as documented in D5.3. The feedback received by the project's pilots on this first version of the platform resulted in several optimisations and adjustments on the platform and components' architecture.

Towards this end, the objective of this deliverable is to provide supplementary documentation on top of the information documented in deliverable D4.2. In more details, the current document is presenting the new additional technical requirements elicited from the evaluation of the feedback received. These new requirements served as input for the analysis performed for the update of the high-level architecture of the platform that will be presented within the context of this deliverable. Moreover, the current document will present all essential updates on the documentation of the components in terms of design and functionalities undertaken, as well as document the technical interfaces implemented by each component to facilitate the integration of the components towards the implementation of the integrated platform. It should be noted that the current document contains also information included in D4.2 for coherency reasons.

The design of the architecture of the BigDataOcean platform is a living process that will last until M26, and as the project evolves new updates will be incorporated in the architecture design as new technical requirements will be elicited from additional functionalities that will be designed and planned to be integrated in the platform. Additionally, as new releases of the platform will be delivered the feedback that will be received by the project's pilots will also introduce new updates and customisations on the platform and components' architecture accordingly.

1.2 Structure of the deliverable

Deliverable 4.3 is organised in five main sections as indicated in the table of contents.

1. The first section introduces the deliverable. It documents the scope of the deliverable and briefly describes how the document is structured. It also documents the positioning of the deliverable in the project, namely the relation of the current deliverable with the other deliverables, and how the knowledge produced in the other deliverables and work-packages served as input to the current deliverable.
2. Following the introduction, section 2 documents the additional technical requirements elicited from the analysis of the feedback received from the project's pilot for the first mock-up

version of the BigDataOcean platform. Moreover, the updated high-level architecture of the platform is presented by describing the roles and responsibilities undertaken by each component highlighting the updates on the components.

3. Section 3 provides the updated documentation of the components composing the platform's architecture. For each component the updated design and specifications are documented as well as the functionalities of the platform that the component is implementing. The technical requirements are mapped to the components and the interactions between the components are documented with the use of sequence diagrams. In this section, the interfaces implemented by each component (when available) are also documented.
4. Section 4 reports the conclusions of the deliverable outlining the main findings of the deliverable, which will guide the future research and technological efforts of the consortium.
5. At the end of the current document two annexes are included. Annex I is documenting the mapping between the technical requirements and the platform components while Annex II is documenting the mapping between the platform components and the technical requirements. Both annexes are the updated versions of the annexes presented in D4.2 with the addition of the new requirements documented in section 2.

1.3 Positioning within the project

Deliverable D4.3 is the outcome of the efforts undertaken within the context of all the tasks of WP4, namely the Task 4.1, the Task 4.2, the Task 4.3 and the Task 4.4. D4.3 builds upon the outcomes of the efforts carried out within the context of WP4 till M7 when D4.2 was delivered. D4.3 receives as input the preliminary version of the conceptual architecture of the BigDataOcean platform, as well as the platform components identified along with the specified design and specifications. Moreover, the feedback received on the first mock-up version of the BigDataOcean platform that was released in M14, as documented in D5.3, serves as an input for the optimisations and refinements introduced on both the platform and the platform's components.

2 High level architecture

2.1 Technical Requirements

In deliverable D4.1 the technical requirements of the BigDataOcean platform were documented with respect to the BigDataOcean Data Value Chain. These technical requirements served as input in the analysis performed for the design of the platform's components and the design of the high-level architecture of the platform.

During the development activities of the first version of the BigDataOcean platform and the implementation phase of the first version of the platform's components, new additional technical requirements were identified by the consortium. These new technical requirements resulted in both updates and refinements being introduced on the design of some of the components of the platform in order to properly address them, as well as provide the requested functionalities and features on the platform.

In the following table, the additional technical requirements are presented. The updated complete list of technical requirements, originally documented in D4.2, is provided in Annex I which also includes the mapping of the technical requirements with the components.

Data Value Chain Step	Code	Technology Requirement	Priority
Data Curation	**DC-TR-7**	The system should be able to parse and normalise data in structured format (for example netCDF files).	High
	DC-TR-8	The system should be able to parse and normalise data in semi-structured format (for example CSV, Excel, KML files).	High
Data Storage	**DS-TR-8**	The system should be able to optimise the storage of spatio-temporal data facilitating their more efficient query execution.	High
Data Usage	**DU-TR-7**	The system should be able to perform simple and advanced queries over different datasets originating from different source formats.	High
	DU-TR-8	The system should be able to perform spatio-temporal query execution	High

Table 2-1: BigDataOcean additional technical requirements

2.2 High level architecture description

Within the context of deliverable D4.2, the first version of the high-level architecture description has been provided as the outcome of the translation of the technical requirements into the design and specification of the BigDataOcean components. The purpose of this first version was to address all the technical requirements and provide the necessary functionalities envisioned for the first version of the BigDataOcean platform.

However, as the project evolved and based on the initial evaluation and feedback received, the consortium performed a re-evaluation of the high-level architecture and agreed on several necessary refinements and enhancements on the components towards the aim of addressing new end user requirements and providing additional features and functionalities to the BigDataOcean platform. As a result, the high-level architecture received the necessary updates and is illustrated in Figure 2-1.

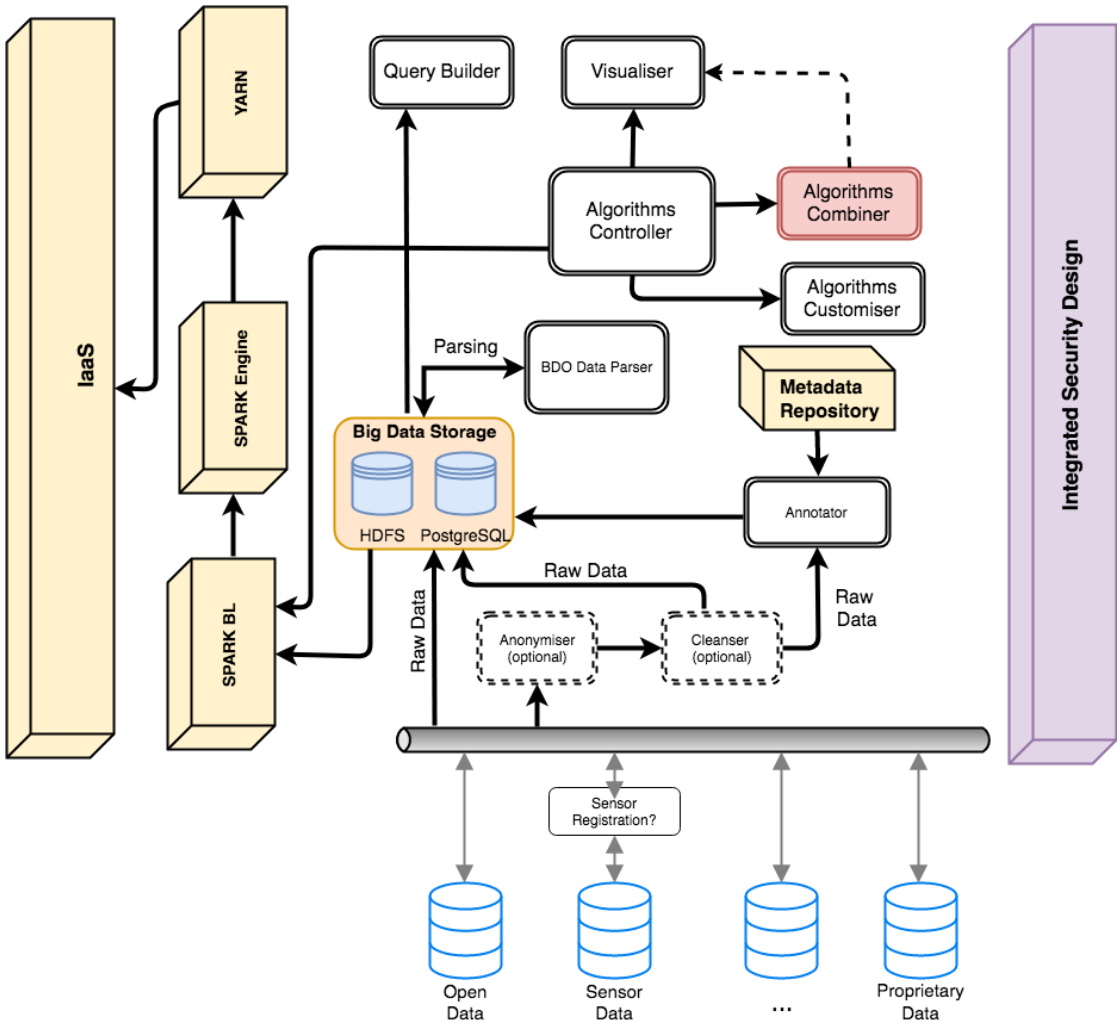

Figure 2-1: BigDataOcean updated High Level architecture

In the updated high-level architecture one of the most important updates is the decision to provide the storage capabilities of the platform with two distinct storage solutions, the Hadoop Distributed File System (HDFS) and the PostgreSQL, with each storage solution being utilised for different purposes. In particular, HDFS is used to store raw data and possibly semantically enriched data received by the platform. HDFS is capable of storing large amount of structured or unstructured data in a fast, reliable and efficient way. On the other hand, PostgreSQL is used to store the results of the BDO Data Parser component. PostgreSQL is object-relational database management system (ORDBMS) focusing on security and efficiency for handling a variety of workloads. BDO Data Parser is designed and implemented to parse and normalise the raw data originally stored in the HDFS. More specifically, BDO Data Parser is responsible for extracting the available information from the various file formats, such as netCDF, CSV and Excel files, and transforming the extracted information according to the schema defined in the PostgreSQL.

Concerning the data extraction process in the updated high-level architecture two optional components, the Anonymiser and the Cleanser, are offered. The Anonymiser, as described also in D4.2, will provide the anonymisation processes required dealing with privacy issues, protection of personal, sensitive data and risk elimination of unintended disclosure. The Cleanser is the second optional offline tool offering the cleansing functionalities for the detection and correction of corrupted, incomplete or incorrect records from a dataset. The Cleanser will provide the necessary mechanisms to replace, modify or eliminate the identified erroneous data.

Following the optional components for anonymisation and cleansing purposes, the Annotator encapsulates the semantic annotation of the extracted dataset utilising the ontologies and vocabularies provided by the Metadata Repository. The Annotator is providing the semantic enrichment of the dataset by mapping the contents of the dataset with the well-defined semantics of a domain, introducing semantic relationships to provide information for the dataset and its relationship with other datasets. The ontologies and vocabularies are stored, managed and provided to the Annotator component by the Metadata Repository. The Metadata Repository will enable importing, managing, querying and interlinking vocabularies for the maritime domain.

On the top of the data storage of the BigDataOcean lays the Query Builder. The Query Builder is the innovative graphical user interface enabling the exploration, combination and expression of simple or complex queries on top of the available datasets. Query Builder is eliminating the need of experience with formal query languages or knowledge of the structure of the underlying data, providing an easy, efficient and friendly way to define queries over multiple datasets with a variety of predefined filters. The results of the queries can be passed as input to other components such as the Algorithms Controller and the Visualiser.

The next component in the foreseen logical flow is the Algorithms Controller, which is responsible for the handling of the execution of the analysis algorithms. Algorithms Controller is responsible for initiating and monitoring the requests to Spark framework and for providing the results of the analysis to the Visualiser component. For the customisation of the algorithms, the Algorithm Customiser is undertaking the responsibility of collecting the parameter values required and providing them to the Algorithm Controller, while the Algorithm Combiner is extending the analysis capabilities by enabling the multiple algorithm execution in a chainable way.

The Visualiser is enabling the visualisation capabilities of the BigDataOcean platform. The Visualiser offers a variety of interactive data visualisations that can be utilised also within customisable and flexible dynamic dashboards. The Visualiser is receiving the results of the Query Builder or the Algorithms Controller as input and enables the appropriate visual representation of the results.

For the purposes of BigDataOcean, Apache Spark has been selected as the cluster-computing framework. Spark is offering the processing engine suitable for analysis with a combination of speed, performance, reliability and efficiency. Spark is also offering cluster management with data parallelism and fault-tolerance suitable for the needs of the BigDataOcean platform. Apache Hadoop YARN is supplementing the cluster management offered by Spark. YARN is providing a centralised platform responsible for the resource management and scheduling. YARN is offering the support of multiple data processing engines enabling Spark with extended functionalities like enhanced resource management, job scheduling and monitor tools over the cluster nodes and the necessary security mechanisms.

3 BigDataOcean Components specifications

3.1 Anonymiser

3.1.1 Overview

The Anonymiser is an offline tool with the purpose of data anonymisation. Data anonymisation is a very crucial aspect when datasets need to be uploaded or shared on a platform. It is usual that without data anonymisation the transfer of information across a boundary, which could be either a company network, an organisation or a personal perspective when sharing personal data, might not be possible or acceptable. The Anonymiser tool must deal with privacy issues, eliminate the risk of unintended disclosure and protect sensitive personal data. It must be able to deal with sensitive attributes in a way that privacy threats like attribute disclosure or identity disclosure are handled properly. Several anonymisation techniques like data masking, data encryption or data scrambling will be considered during implementation having in mind that the tool much achieves a balance between data utility and privacy.

The Anonymiser does not meet any specific technical requirements since it was not deemed a necessity during the stakeholders' needs' analysis to anonymise sensitive information. Nevertheless, the Anonymiser has been included as an optional component, because such a need may arise in the future, and thus the delivery of such a tool may be deemed beneficial for a stakeholder wishing to share a data source through the BDO platform.

The Anonymiser will receive datasets from the data sources in order to offer an effective way of filtering the original data upon the user needs. For this reason, the user will be able to choose and customise the applicable risk models, privacy models, transformation models and data utility measures before the tool will perform the anonymisation process. The output of the anonymised datasets will be offered to the optional Cleanser tool.

Figure 3-1 displays the execution flow of Anonymiser.

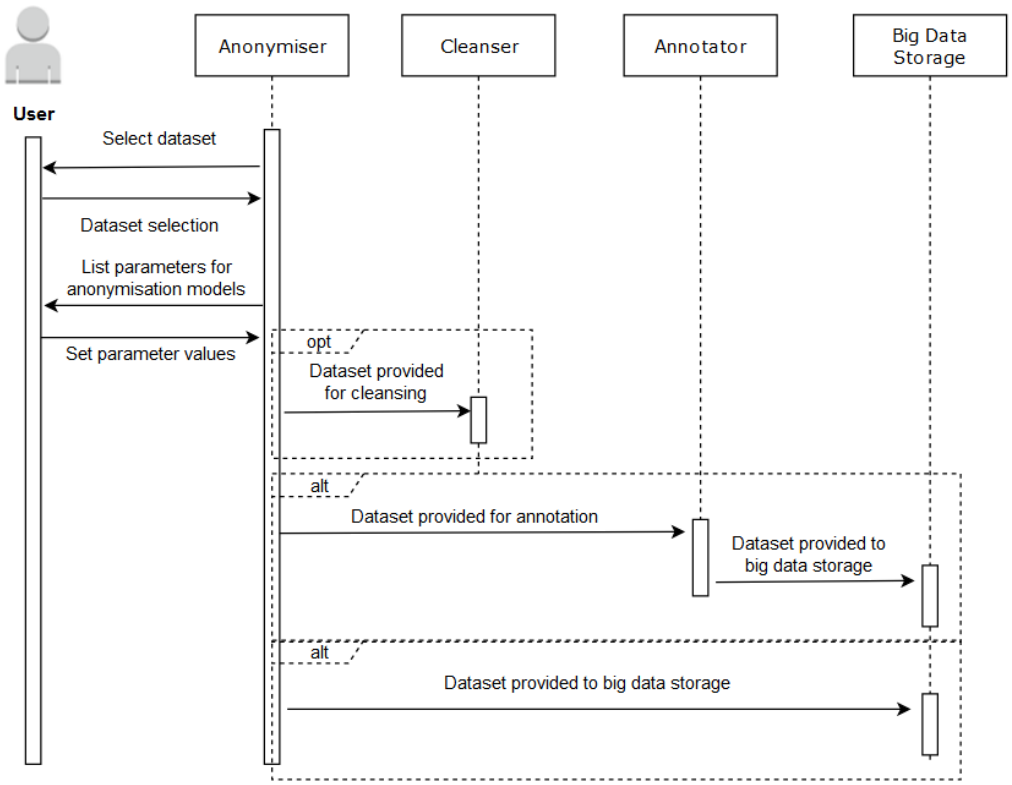

Figure 3-1: Anonymiser Sequence Diagram

3.1.2 Interfaces

In the first version of the Anonymiser, the availability of an API is not foreseen. However, as the project evolves and based on the upcoming needs of the project, this decision will be reconsidered.

3.2 Cleanser

3.2.1 Overview

The Cleanser is an optional offline tool addressing the data cleansing requirements of the stakeholders of the BigDataOcean platform. The Cleanser will implement the processes that will assure that the datasets received by the platform are clean and complete according to the standards set by the platform. Within this context, the data cleaning process will detect and correct (or even remove) corrupted or inaccurate records from a dataset in order to assure that the data are consistent, complete and accurate. The Cleanser will identify incomplete, inaccurate, incorrect and irrelevant records from datasets with the aim of providing the necessary actions to replace, modify or delete erroneous records. In addition to this, the Cleanser will handle also duplicate and missing entries in order to safeguard the appropriateness and reliability of the information provided in the dataset.

The purpose of the Cleanser is to provide the assurance on the consistency of the datasets stored in the platform although they are originating from various heterogeneous data sources. The Cleanser will focus on the assurance of the data integrity, data accuracy, data precision and data reliability. The Cleanser will provide several functionalities in the course of the execution of the data cleaning process, which include the data identification and analysis of the content, the selection of the part of

the dataset to be cleansed and the specification of the cleansing parameters and rules from a list of predefined operations.

With regard to the design and specifications defined as presented in the deliverable D4.2, there were no updates or modifications since there were no additional requirements identified that would introduce any updates or refinements. It should be noted at this point that the implementation of the Cleanser component was not included in the first release of the BigDataOcean platform that was delivered in M14 with deliverable D5.3 but will be included in the upcoming versions of the platform.

The Cleanser will address the following requirements:

1. **DC-TR-1:** The Cleanser facilitates the effective and efficient cleansing of the datasets (e.g. remove duplicates, inconsistent values, identify outliers, values with wrong types).
2. **DC-TR-3:** The Cleanser enables the cleansing processes in the course of data curation.
3. **DC-TR-6:** The Cleanser will be offered as an offline tool that can be used on the client side.

Figure 3-2 displays the execution flow of Cleanser.

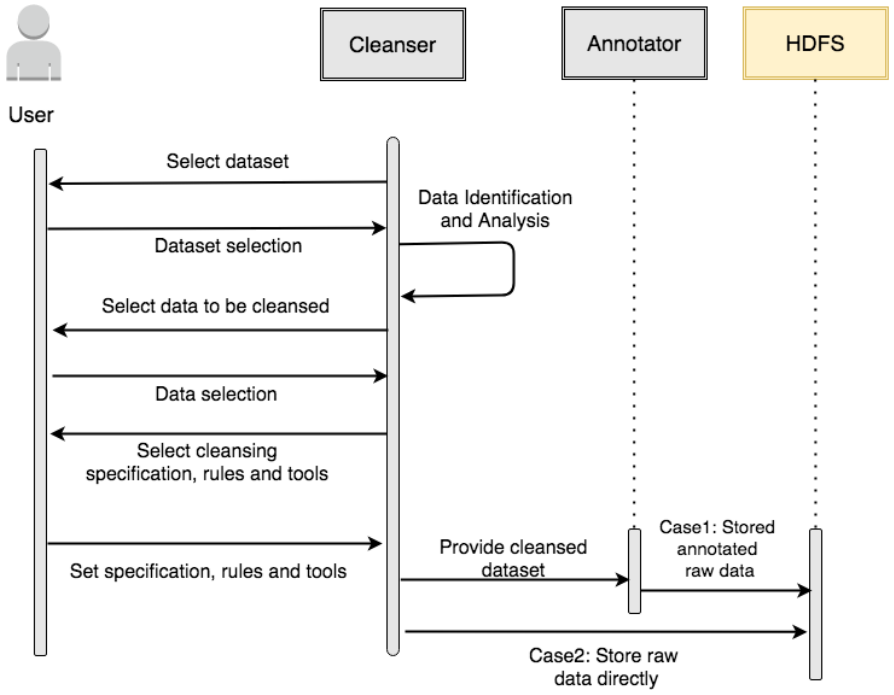

Figure 3-2: Cleanser sequence diagram

3.2.2 Interfaces

In the first version of the Cleanser the availability of an API is not foreseen. However, as the project evolves and based on the upcoming needs of the project, this decision will be reconsidered.

3.3 Metadata Repository

3.3.1 Overview

The Metadata Repository allows users to import, manage, query, and interlink vocabularies related to the maritime domain. It aims at publishing this information online for the maritime domain experts and exposes APIs. These APIs are provided to the Annotator for retrieving related ontologies and vocabularies from the repository and mapping them to maritime datasets. The vocabularies/

ontologies are stored in a triple store and represented as RDF data according to the Semantic Web standards. Therefore, they can also be queried using the SPARQL query language.

The Metadata Repository addresses the following requirements:

- **DC-TR-4**: The transformation of data (converting values to other formats, normalising and de-normalising) will be possible with the use of ontologies and vocabularies stored in the Metadata Repository.
- **DC-TR-5**: The Metadata Repository will provide the tools for realising the semantic curation and reconciliation of datasets.

As illustrated in Figure 3-3 the main components of the BigDataOcean Metadata Repository are the following:

- **Vocabulary Importer**: This component is responsible for importing ontologies and vocabularies of different formats;
- **Metadata Management**: Is responsible for CRUD (Create, Read, Update and Delete) operations regarding all metadata; it also manages the versioning of the vocabularies;
- **Statistics**: It calculates and persists statistics about the vocabulary metadata persisted in the metadata repository;
- **Quality Metrics**: It calculates and persists quality metrics about the vocabulary metadata persisted in the metadata repository;
- **Search/ Query Engine**: Provides search and query capabilities on top of the metadata persisted in the metadata repository;
- **UI**: Graphical User Interface for interaction of vocabulary publishers and consumers with the metadata repository.

On the bottom level, the vocabulary metadata refers to vocabularies that are published as Linked Open data. The BigDataOcean Metadata Repository provides a SPARQL endpoint and API that can be used as well by other BigDataOcean tools providing Semantic Enrichment, Data Linking, and Data Management. More concretely, the Metadata Repository is connected to the BDO Annotator (or Harmonisation tool) which retrieves information about existing terms/ concepts and annotates the imported BDO datasets.

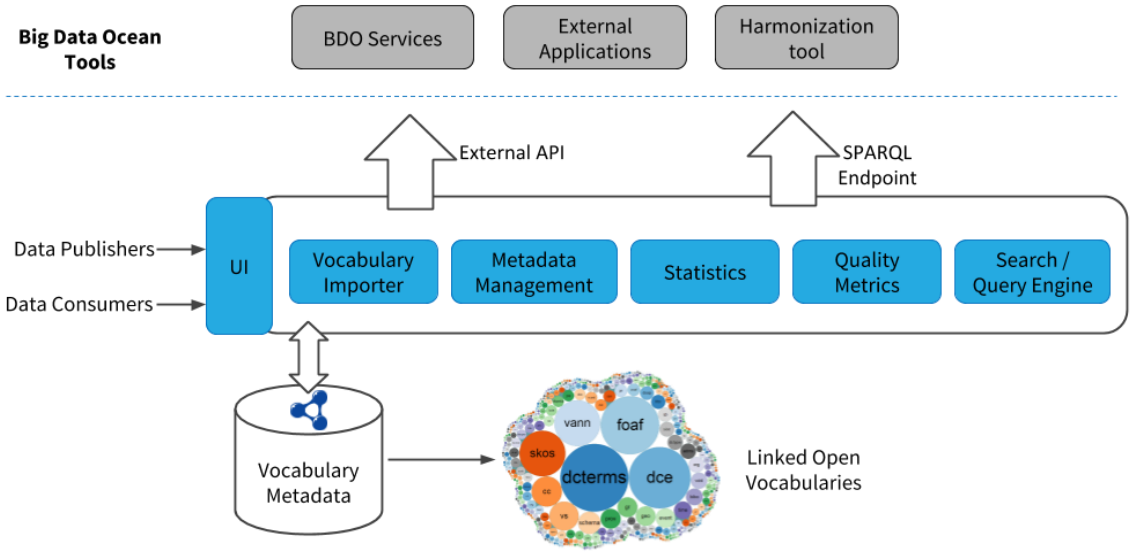

Figure 3-3: Architecture of the Metadata Repository

3.3.2 Interfaces

BigDataOcean Metadata Repository offers the following API methods. All methods defined in BigDataOcean Metadata Repository, divided in four categories, can be called with an HTTP GET request.

Vocabulary Term (Class, Property, Datatype, Instance)

<table>
<tr><td>Endpoint Name</td><td>Search Term API v2</td></tr>
<tr><td>Description</td><td>The Search Term API allows a user to search over BigDataOcean Repository for a vocabulary term (class, property, datatype or instance).</td></tr>
<tr><td>HTTP Method</td><td>GET</td></tr>
<tr><td>Endpoint URL</td><td>http://server_ip:port/dataset/lov/api/v2/term/search</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td>q</td><td>string</td><td>Full-text query.</td></tr>
<tr><td>page_size</td><td>int</td><td>Maximum number of results to return per page (default: 10).</td></tr>
<tr><td>page</td><td>int</td><td>Result page to display starting from 1 (default: 1).</td></tr>
<tr><td>type</td><td>string</td><td>Filter query results based on their type. Possible values: [class, property, datatype, instance]. Multiple values allowed (use comma without space to separate them).</td></tr>
<tr><td>vocab</td><td>string</td><td>Filter query results based on the vocabulary it belongs to (e.g. "foaf"). Expecting only one value.</td></tr>
<tr><td>vocab_limit</td><td>int</td><td>Number of elements to display in the vocabulary facet (default: 10).</td></tr>
<tr><td>tag</td><td>string</td><td>Filter query results based on their tag (e.g. "event"). Multiple values allowed, use comma as a separator (e.g. "event, time").</td></tr>
<tr><td>tag_limit</td><td>int</td><td>Number of elements to display in the tag facet (default: 10).</td></tr>
</table>
</td></tr>
<tr><td>Request Body</td><td>-</td></tr>
</table>

Table 3-1: Search Term API v2 interface

<table>
<tr><th>Endpoint Name</th><th>Autocomplete Term API v2</th></tr>
<tr><td>Description</td><td>The Autocomplete Term API allows a user to get autocompletion recommendations for terms in BigDataOcean Repository. The autocompletion is performed on terms URI (e.g. http://xmlns.com/foaf/0.1/Person) or terms prefixed URI (foaf:Person). The user can ask for autocompletion from 1 character and filter by type class or property.</td></tr>
<tr><td>HTTP Method</td><td>GET</td></tr>
<tr><td>Endpoint URL</td><td>http://server_ip:port/dataset/lov/api/v2/term/search</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td>q</td><td>string</td><td>URI or prefixed URI to complete (from 1 character).</td></tr>
<tr><td>type</td><td>string</td><td>Filter query results type: "property" or "class" (if no type is given, will search over property AND class types.)</td></tr>
<tr><td>page_size</td><td>int</td><td>Maximum number of results to return per page (default: 10).</td></tr>
</table>
</td></tr>
<tr><td>Request Body</td><td>–</td></tr>
</table>

Table 3-2: Autocomplete Term API v2 interface

<table>
<tr><th>Endpoint Name</th><th>Suggest Term API v2</th></tr>
<tr><td>Description</td><td>The Suggest Term API allows a user to get suggestion of vocabulary terms label based on the labels stored in BigDataOcean Repository. It is useful for instance to help fix some typos for a search engine.</td></tr>
<tr><td>HTTP Method</td><td>GET</td></tr>
<tr><td>Endpoint URL</td><td>http://server_ip:port/dataset/lov/api/v2/term/suggest</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td>q</td><td>string</td><td>URI or prefixed URI to complete (from 1 character).</td></tr>
<tr><td>type</td><td>string</td><td>Filter suggestions based on their type. Possible values: [class, property, datatype, instance]. Multiple values allowed (use comma without space to separate them).</td></tr>
<tr><td>page_size</td><td>int</td><td>Maximum number of results to return</td></tr>
</table>
</td></tr>
</table>

			per page (default: 10).
Request Body	–		

Table 3-3: Suggest Term API v2 interface

Vocabulary

Endpoint Name	List Vocab API v2
Description	The List Vocab API allows a user to list the vocabularies part of the BigDataOcean Repository.
HTTP Method	GET
Endpoint URL	http://server_ip:port/dataset/lov/api/v2/vocabulary/list
Parameters	–
Request Body	–

Table 3-4: List Vocab API v2 interface

Endpoint Name	Search Vocab API v2
Description	The Search Vocab API allows a user to search over BigDataOcean Repository for a Vocabulary based on its title or prefix.
HTTP Method	GET
Endpoint URL	http://server_ip:port/dataset/lov/api/v2/vocabulary/search
Parameters	

Parameter	Type	Description
q	string	Full text query.
page_size	int	Maximum number of results to return per page (default: 10).
page	int	Result page to display starting from 1 (default: 1).
tag	string	Filter query results based on their tag (e.g. "event"). Multiple values allowed, use comma as a separator (e.g. "event, time").
tag_limit	int	Number of elements to display in the tag facet (default: 10).
lang	string	Filter query results based on their language (e.g. "french"). Multiple values allowed, use comma as a separator (e.g. "english", "french").

	lang_limit	int	Number of elements to display in the language facet (default: 10).
	pilot	string	Filter query results based on their pilot (e.g. "PILOT_I"). Multiple values allowed, use comma as a separator (e.g. "PILOT_I, PILOT_II").
	pilot_limit	int	Number of elements to display in the pilot facet (default: 10).
Request Body	–		

Table 3-5: Search Vocab API v2 interface

Endpoint Name	Autocomplete Vocab API v2		
Description	The Autocomplete Vocab API allows a user to get autocompletion recommendations for vocabularies in BigDataOcean Repository. The autocompletion is performed on vocabulary URI (e.g. http://www.w3.org/2003/01/geo/wgs84_pos) or vocabulary prefix (geo). The user can ask for autocompletion from one character.		
HTTP Method	GET		
Endpoint URL	http://server_ip:port/dataset/lov/api/v2/vocabulary/autocomplete		
Parameters	**Parameter**	**Type**	**Description**
	q	string	URI or prefixed URI to complete (from 1 character).
	page_size	int	Maximum number of results to return per page (default: 10).
Request Body	–		

Table 3-6: Autocomplete Vocab API v2 interface

Endpoint Name	Info Vocab API v2		
Description	The Info Vocab API allows a user to get details about one vocabulary.		
HTTP Method	GET		
Endpoint URL	http://server_ip:port/dataset/lov/api/v2/vocabulary/info		
Parameters	**Parameter**	**Type**	**Description**
	vocab	string	Prefix, URI or Namespace of a vocabulary in BigDataOcean. This parameter is mandatory

Request Body	–

Table 3-7: Info Vocab API v2 interface

Agent

Endpoint Name	List Agent API v2
Description	The List Agent API allows a user to list all agents (Person or Organisation) in BigDataOcean Repository.
HTTP Method	GET
Endpoint URL	http://server_ip:port/dataset/lov/api/v2/agent/list
Parameters	–
Request Body	–

Table 3-8: List Agent API v2 interface

Endpoint Name	Search Agent API v2
Description	The Search Agent API allows a user to search over BigDataOcean Repository for an agent (person or organisation).
HTTP Method	GET
Endpoint URL	http://server_ip:port/dataset/lov/api/v2/agent/search
Parameters	**Parameter** / **Type** / **Description** q / string / Full-text query. page_size / int / Maximum number of results to return per page (default: 10). page / int / Result page to display starting from 1 (default: 1). type / string / Filter query results based on their type. Possible values: [person, organisation]. tag / string / Filter query results based on their tag (e.g. "event"). Multiple values allowed, use comma as a separator (e.g. "event, time"). tag_limit / int / Number of elements to display in the tag facet (default: 10).
Request Body	–

Table 3-9: Search Agent API v2 interface

<table>
<tr><th>Endpoint Name</th><th>Autocomplete Agent API v2</th></tr>
<tr><td>Description</td><td>The Autocomplete Agent API allows a user to get autocompletion recommendations for agents (Person or Organisation) in BigDataOcean Repository.
The autocompletion is performed on agent name (e.g. Food and Agriculture Organisation) or agent URI
(e.g. http://212.101.173.34:3333/dataset/lov/agents/Food%20and%20Agriculture%20Organization
).
The user can ask for autocompletion from 1 character.</td></tr>
<tr><td>HTTP Method</td><td>GET</td></tr>
<tr><td>Endpoint URL</td><td>http://server_ip:port/dataset/lov/api/v2/agent/autocomplete</td></tr>
<tr><td>Parameters</td><td><table><tr><th>Parameter</th><th>Type</th><th>Description</th></tr><tr><td>q</td><td>string</td><td>URI or prefixed URI to complete (from 1 character).</td></tr></table></td></tr>
<tr><td>Request Body</td><td>–</td></tr>
</table>

Table 3-10: Autocomplete Agent API v2 interface

<table>
<tr><th>Endpoint Name</th><th>Info Agent API v2</th></tr>
<tr><td>Description</td><td>The Info Agent API allows a user to get details about one agent in BigDataOcean.</td></tr>
<tr><td>HTTP Method</td><td>GET</td></tr>
<tr><td>Endpoint URL</td><td>http://server_ip:port/dataset/lov/api/v2/agent/info</td></tr>
<tr><td>Parameters</td><td><table><tr><th>Parameter</th><th>Type</th><th>Description</th></tr><tr><td>agent</td><td>string</td><td>URI, AltURI or FullName (as stated in BigDataOcean) of an agent in BigDataOcean. This parameter is mandatory</td></tr></table></td></tr>
<tr><td>Request Body</td><td>–</td></tr>
</table>

Table 3-11: Info Agent API v2 interface

Pilot

<table>
<tr><th>Endpoint Name</th><th>List Pilot API v2</th></tr>
<tr><td>Description</td><td>The List Pilot API allows a user to list all pilots in BigDataOcean Repository.</td></tr>
<tr><td>HTTP Method</td><td>GET</td></tr>
</table>

Endpoint URL	http://server_ip:port/dataset/lov/api/v2/pilot/list
Parameters	–
Request Body	–

Table 3-12: List Pilot API v2 interface

Endpoint Name	Search Pilot API v2
Description	The Search Pilot API allows a user to search over BigDataOcean Repository for a pilot.
HTTP Method	GET
Endpoint URL	http://server_ip:port/dataset/lov/api/v2/pilot/search
Parameters	**Parameter** / **Type** / **Description** q / string / Full-text query. page_size / int / Maximum number of results to return per page (default: 10). page / int / Result page to display starting from 1 (default: 1). type / string / Filter query results based on their type. Possible value: [pilot]. tag / string / Filter query results based on their tag (e.g. "event"). Multiple values allowed, use comma as a separator (e.g. "event, time"). tag_limit / int / Number of elements to display in the tag facet (default: 10).
Request Body	–

Table 3-13: Search Pilot API v2 interface

Endpoint Name	Autocomplete Pilot API v2
Description	The Autocomplete Pilot API allows a user to get autocompletion recommendations for pilots in BigDataOcean Repository. The autocompletion is performed on pilot name (e.g. PILOT_IV). The user can ask for autocompletion from 1 character.
HTTP Method	GET
Endpoint URL	http://server_ip:port/dataset/lov/api/v2/pilot/autocomplete

Parameters	<table><tr><th>Parameter</th><th>Type</th><th>Description</th></tr><tr><td>q</td><td>string</td><td>Name (from 1 character).</td></tr></table>
Request Body	–

Table 3-14: Autocomplete Pilot API v2 interface

Endpoint Name	Info Pilot API v2
Description	The Info Pilot API allows a user to get details about one pilot in BigDataOcean.
HTTP Method	GET
Endpoint URL	http://server_ip:port/dataset/lov/api/v2/pilot/info
Parameters	<table><tr><th>Parameter</th><th>Type</th><th>Description</th></tr><tr><td>pilot</td><td>string</td><td>Name (as stated in BigDataOcean) of a BigDataOcean pilot. This parameter is mandatory</td></tr></table>
Request Body	–

Table 3-15: Info Pilot API v2 interface

3.4 Annotator/ Harmonisation tool

3.4.1 Overview

The goal of the Annotator, also called Harmonisation tool, is to semantically annotate datasets in input utilising metadata vocabularies available in the Metadata Repository. The datasets are provided in input as raw data, ideally by the Cleanser component after the Anonymiser module. The Metadata Repository contains ontologies/ vocabularies for describing both metadata about the datasets themselves and the data contained in these datasets as well. Domain-specific vocabularies are used for representing the meaning of the attributes inside each dataset. The component responsible for associating a meaning to (or annotating) each attribute of the datasets is the Harmonisation tool.

The Annotator requires the input of domain experts in order to perform the annotation of the data attributes which are being imported into the BDO platform. The domain expert has the role of choosing from the Metadata Repository which terms best match the meaning of the attributes in the dataset. Some of the annotations can be automated, i.e. they would be done automatically without user input whenever possible. A GUI for the Harmonisation tool supports users in specifying the annotations. The annotations are received by the Big Data Framework and Storage which can translate and store these metadata descriptions of the datasets into the Big Data Storage.

The Annotator addresses the following BDO requirements already defined in D4.2:

- **DAC-TR-5:** The system should be able to link big data sources to BigDataOcean Infrastructure.

- **DC-TR-5:** The system should be able to program the semantic curation/ reconciliation of datasets.
- **DC-TR-6:** The data curation tools can be used offline at the client side.

and also the following two additional requirements defined here in Section 2.1:

- **DC-TR-7:** The system should be able to parse and normalise data in structured format (for example netCDF files).
- **DC-TR-8:** The system should be able to parse and normalise data in semi-structured format (for example CSV, Excel, KML files).

Figure 3-4 displays the execution flow of the Annotator.

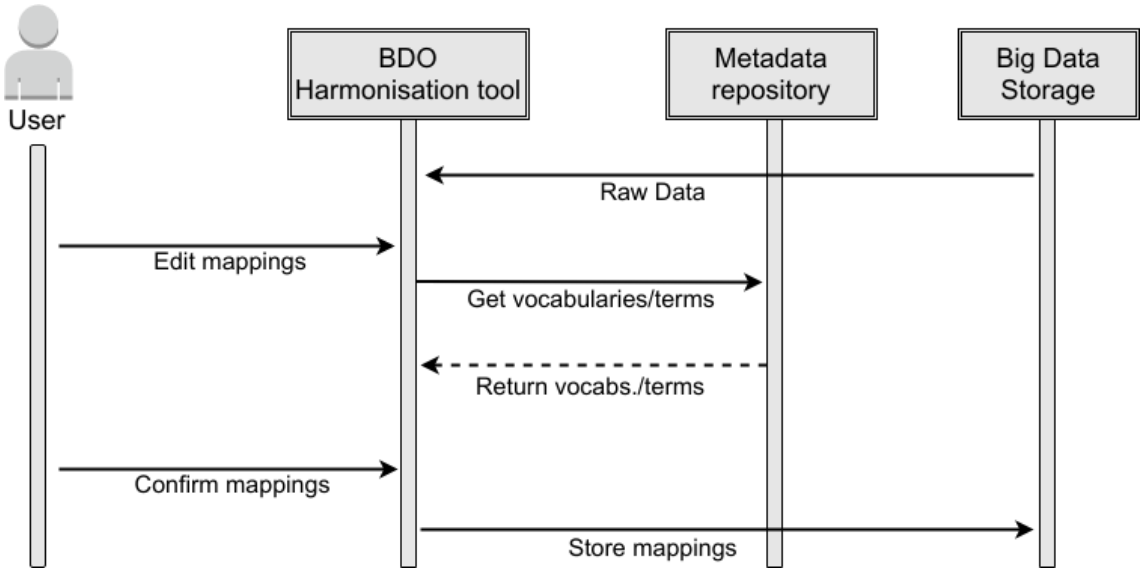

Figure 3-4: Annotator/ Harmonisation tool Sequence diagram

3.4.2 Interfaces

BigDataOcean Harmonisation Tool offers the following API methods. All methods defined in BigDataOcean Harmonisation Tool can be called with an HTTP GET request.

Endpoint Name	List Dataset v1
Description	It lists all datasets for which metadata are stored in the BigDataOcean Harmonisation Tool.
HTTP Method	GET
Endpoint URL	http://server_ip:port/api/v1/dataset/list
Parameters	–
Request Body	–

Table 3-16: List Dataset v1 interface

Endpoint Name	Search Dataset v1

Description	It searches a dataset for which metadata is stored in the BigDataOcean Harmonisation Tool given a dataset id.
HTTP Method	GET
Endpoint URL	http://server_ip:port/api/v1/dataset/searchDataset
Parameters	**Parameter** / **Type** / **Description** id / string / URI of the dataset to search for.
Request Body	–

Table 3-17: Search Dataset v1 interface

Endpoint Name	Search Dataset (Subject) v1
Description	It searches a dataset for which metadata is stored in the BigDataOcean Harmonisation Tool given a subject or a list of subjects.
HTTP Method	GET
Endpoint URL	http://server_ip:port/api/v1/dataset/searchSubject
Parameters	**Parameter** / **Type** / **Description** search / string / Name of dataset to search for. More subjects are separated by comma.
Request Body	–

Table 3-18: Search Dataset (Subject) v1 interface

Endpoint Name	Search Dataset (Keyword) v1
Description	It searches a dataset for which metadata is stored in the BigDataOcean Harmonisation Tool given a keyword or a list of keywords.
HTTP Method	GET
Endpoint URL	http://server_ip:port/api/v1/dataset/searchKeyword
Parameters	**Parameter** / **Type** / **Description** keyword / string / Keyword of dataset to search for. More keywords are separated by comma.
Request Body	–

Table 3-19: Search Dataset (Keyword) v1 interface

Endpoint Name	Search Dataset (Geo Location) v1
Description	It searches a dataset for which metadata are stored in the BigDataOcean Harmonisation Tool given geographical location.
HTTP Method	GET
Endpoint URL	http://server_ip:port/api/v1/dataset/searchGeoLocation
Parameters	**Parameter** / **Type** / **Description** geolocation / string / Geographical location of dataset to search for.
Request Body	–

Table 3-20: Search Dataset (Geo Location) v1 interface

Endpoint Name	Search Dataset (Geo Coverage) v1
Description	It searches a dataset for which metadata is stored in the BigDataOcean Harmonisation Tool given geographical coverage.
HTTP Method	GET
Endpoint URL	http://server_ip:port/api/v1/dataset/searchGeoCoverage
Parameters	**Parameter** / **Type** / **Description** west / number / Limit on the west. east / number / Limit on the east. south / number / Limit on the south. north / number / Limit on the north.
Request Body	–

Table 3-21: Search Dataset (Geo Coverage) v1 interface

Endpoint Name	Search Dataset (Vertical Coverage) v1
Description	It searches a dataset for which metadata is stored in the BigDataOcean Harmonisation Tool given vertical coverage.
HTTP Method	GET
Endpoint URL	http://server_ip:port/api/v1/dataset/searchVerticalCoverage

Parameters	<table><tr><th>Parameter</th><th>Type</th><th>Description</th></tr><tr><td>from</td><td>number</td><td>Lowest value of the depth.</td></tr><tr><td>to</td><td>number</td><td>Highest value of the depth.</td></tr></table>
Request Body	–

Table 3-22: Search Dataset (Vertical Coverage) v1 interface

Endpoint Name	Search Dataset (Temporal Coverage) v1
Description	It searches a dataset for which metadata is stored in the BigDataOcean Harmonisation Tool given temporal coverage.
HTTP Method	GET
Endpoint URL	http://server_ip:port/api/v1/dataset/searchTemporalCoverage
Parameters	<table><tr><th>Parameter</th><th>Type</th><th>Description</th></tr><tr><td>begin</td><td>datetime</td><td>Beginning of time to be covered.</td></tr><tr><td>end</td><td>datetime</td><td>End of time to be covered.</td></tr></table>
Request Body	–

Table 3-23: Search Dataset (Temporal Coverage) v1 interface

Endpoint Name	List Variable v1
Description	It lists all variables of the datasets for which metadata are stored in the BigDataOcean Harmonisation Tool.
HTTP Method	GET
Endpoint URL	http://server_ip:port/api/v1/variable/list
Parameters	–
Request Body	–

Table 3-24: List Variable v1 interface

Endpoint Name	Search Dataset (Variables) v1
Description	It searches for datasets having a specific variable or a list of variables.
HTTP Method	GET

<table>
<tr><td>Endpoint URL</td><td>http://server_ip:port/api/v1/dataset/searchVariable</td></tr>
<tr><td>Parameters</td><td>
<table>
<tr><th>Parameter</th><th>Type</th><th>Description</th></tr>
<tr><td>search</td><td>string</td><td>Name of variable. More variables can be given separated by comma.</td></tr>
</table>
</td></tr>
<tr><td>Request Body</td><td>–</td></tr>
</table>

Table 3-25: Search Dataset (Variables) v1 interface

3.5 Big Data Framework & Storage

3.5.1 Overview

The success and efficiency of every Big Data platform is heavily dependent on the incorporated cluster computing framework and storage solution of the platform and as a consequence, their role on the platform is very important.

As described also in deliverable D4.2, the consortium selected Apache Spark as the cluster-computing framework for the BigDataOcean platform. The cluster-computing framework must provide the processing engine required to perform effective and efficient big data analysis as well as deal with the cluster management and scheduling aspects required. Apache Spark is offering a powerful processing engine capable of executing distributed applications by performing cluster management with data parallelism with fault-tolerance. At its core, Spark is implementing a master/ slave architecture built on the concept of the resilient distributed dataset (RDD) utilised for job distribution and data parallelism. Spark implements an effective cluster management mechanism for job distribution and monitoring with dynamic allocation of resources across the nodes of the cluster upon needs. Spark supports a large variety of algorithms, which includes machine learning training algorithms, to facilitate exploratory data analysis. Apache Hadoop YARN was also selected in order to undertake the role of cluster management, supplementing the functionalities offered by Spark. YARN is enabling the support of multiple data processing engines by offering an enhanced toolset for resource management, job scheduling and monitoring. Besides the cluster management, YARN is offering the missing security mechanisms from Spark. The purpose of the big data framework of the BigDataOcean platform is to effectively and efficiently handle the requests for data analysis and execution of algorithms over multiple datasets as received by the Algorithm Controller and report the results of the analysis back to the Algorithms Controller.

The Big Data Storage of the BigDataOcean is the component responsible for the storage capabilities of the platform. It is composed of two different storage solutions, each utilised for different purpose, the Hadoop Distributed File System (HDFS) and the PostgreSQL. HDFS is a distributed file system designed to be deployed on commodity hardware, is highly fault-tolerant providing high throughput access and capable for handling and storing large data sets with high reliability and high availability. HDFS is highly scalable and is easily configurable supporting configurations for many installation types. A HDFS cluster consists of one NameNode that manages the file system metadata and DataNodes that store the actual data. Within HDFS, the data are divided into block and copies of these blocks are stored on different nodes across the HDFS cluster. As a consequence, a file is

actually stored as smaller blocks replicated across multiple nodes in the cluster. To access the file, the NameNode is contacted for file metadata and actual I/O is performed directly on the DataNodes. HDFS supports effective and rapid transfer between the nodes of the cluster, enabling highly efficient parallel processing. HDFS is also supporting shell-like command via its native console. Within the context of the BigDataOcean platform the HDFS is used to store the incoming raw data in their original format (for example in netCDF, CSV or Excel file format) as well as the semantically enriched data as provided by the Annotator.

In addition to the HDFS, PostgreSQL is also used in the Big Data Storage component. PostgreSQL is a powerful open source object-relational database management system (ORDBMS) ensuring data integrity and correctness with high reliability and high availability. Besides efficiency PostgreSQL has advanced support for security. PostgreSQL is fully ACID compliant with full support for foreign keys, joins, views, triggers and stored procedures, as well as support for binary larger objects storage. PostgreSQL offers a variety of features such as the Multi-Version Concurrency Control (MVVC), asynchronous replication, nested transactions, write ahead logging for fault tolerance and foreign key referential integrity. Moreover, it supports user-defined types, table inheritance and provides a sophisticated query planner and optimiser. PostgreSQL also supports a variety of extensions offering advanced features. One of the most valuable extensions is the PostGIS extension. PostGIS is enabling the support for geospatial objects in PostgreSQL facilitating the option to use PostgreSQL as a spatial database for geographic data. Within the context of the BigDataOcean platform, PostgreSQL will be used to store normalised information as parsed and extracted by the BDO Data Parser component.

In deliverable D4.2 the requirements addressed by the Big Data Framework and Storage were documented. In addition to these requirements, the Big Data Framework and Storage is also addressing a new requirement, namely the **DS-TR8** as documented in section 2.1.

The following list is the complete list of requirements addressed for coherency reasons:

1. **DAC-TR-1**: The Big Data Storage is able to connect to (either in a direct manner through custom developed connectors or through the data check in process) and acquire data from sensors and IoT devices of different manufacturers in real-time or intervals.
2. **DAC-TR-2:** The Big Data Storage is able to connect to (either in a direct manner through custom developed connectors or through the data check in process) and acquire data from AIS receivers, transponders or base stations.
3. **DAC-TR-3:** The Big Data Storage is able to connect to (either in a direct manner through custom developed connectors or through the data check in process) and acquire data from large and popular oceanographic databases including Copernicus through APIs.
4. **DAC-TR-4:** The Big Data Storage allows uploading big data datasets on the BigDataOcean infrastructure.
5. **DAN-TR-3:** The Big Data Framework & Storage allows the performance of the desired data analytics on a scalable cluster-computing framework, programming entire clusters with implicit data parallelism and fault-tolerance.
6. **DC-TR-2:** The Big Data Storage allows for the curation of large-scale (big data) datasets in a timely and efficient manner.
7. **DS-TR-1:** The Big Data Storage facilitates the effective and efficient storage and management of large datasets (big data).

8. **DS-TR-2:** The Big Data Framework & Storage allows the parallel setup and operation of different data storage environments in order to accommodate different storage needs (Relational Databases, NoSQL, TripleStores).
9. **DS-TR-3:** The Big Data Framework & Storage has a high degree of scalability.
10. **DS-TR-4:** The Big Data Framework & Storage allows for replication and high availability with no single point of failure.
11. **DS-TR-5:** The Big Data Framework & Storage supports distributed storage and auto-sharing.
12. **DS-TR-7:** The Big Data Storage has a direct pipeline with the chosen cluster computing framework and analytics engine, namely the Big Data Framework
13. **DS-TR-8:** The Big Data Storage facilitates the effective and efficient storage and management of spatio-temporal data.

Figure 3-5 displays the execution flow of Big Data Framework and Storage.

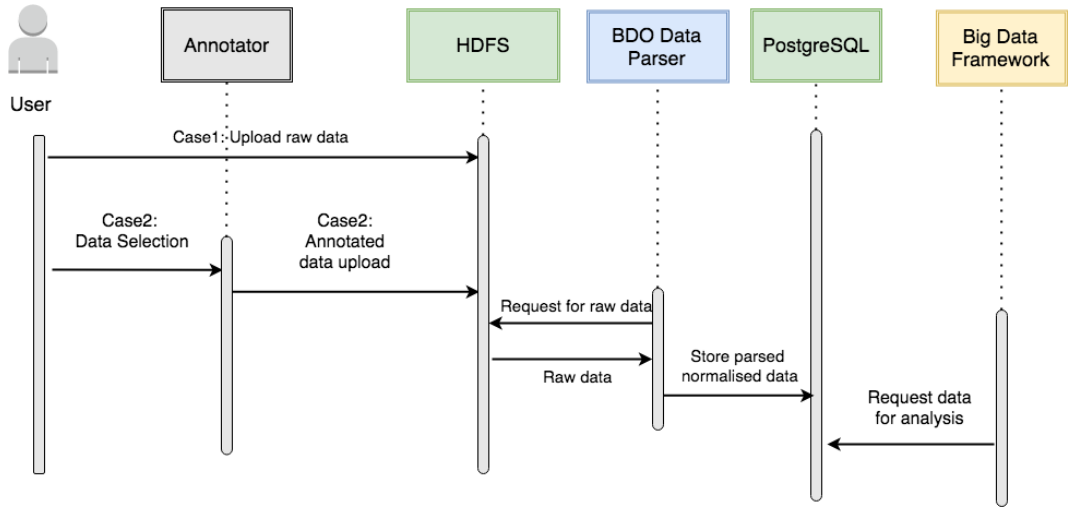

Figure 3-5: Big Data Framework and Storage sequence diagram

3.5.2 Interfaces

The availability of an API in the Big Data Framework and Storage component is not foreseen. However, as the project evolves and based on the upcoming needs of the project, this decision will be reconsidered.

3.6 BDO Data Parser

3.6.1 Overview

BDO Data Parser is the component where the data sources that have been stored in raw format on the Big Data Storage and specifically in the HDFS storage solution, are parsed and normalised. The purpose of the BDO Data Parser is to perform the extraction of the information included in the datasets provided in the raw format in order to normalise and transform the information in a format that can be stored in the second storage solution of the Big Data Storage, the PostgreSQL.

One of the core parts of the BDO Data Parser is the definition of the BigDataOcean Context model. This model is based on the Climate and Forecast (CF) Metadata Convention standard[1]. The definition included various steps in order to create the model that will contain all the variables included in the datasets that will be available initially by the BigDataOcean partners in conformance to the CF. At first, all the variables were collected and the naming conflicts between common variables were resolved. In the next step, the variables were mapped to the CF compliant standard names. Following the mapping of the naming of the variables, the unit conflicts were resolved again in conformance to the CF. The collected variables were mapped to the CF categories and finally the BigDataOcean Context model was defined. This context model serves as input for the schema defined in the PostgreSQL storage solution.

BDO Data Parser receives as input the structured file formats, such as netCDF, and semi-structured file format, such as CSV, Excel, KML, that are stored in HDFS. BDO Data Parser extracts the information from these files, that are afterward parsed and normalised in accordance to the BigDataOcean Context model towards the aim of preparing the available information for insertion into the database provided by PostgreSQL. The results of the process are stored in PostgreSQL and are available to the BigDataOcean for query execution and analysis.

The BDO Data Parser will address the following requirements:

- **DC-TR-2:** The BDO Data Parser facilitates the timely and efficient curation of large scale datasets within the parsing and normalisation mechanisms that implements.
- **DC-TR-7:** The BDO Data Parser enables the parsing and normalisation of structured data in accordance with the BigDataOcean Context model.
- **DC-TR-8:** The BDO Data Parser enables the parsing and normalisation of semi-structured data in accordance with the BigDataOcean Context model.
- **DS-TR-1:** The BDO Data Parser enables the storage and management of large datasets via the parsing and normalisation mechanisms that implements.

Figure 3-6 displays the execution flow of BDO Data Parser.

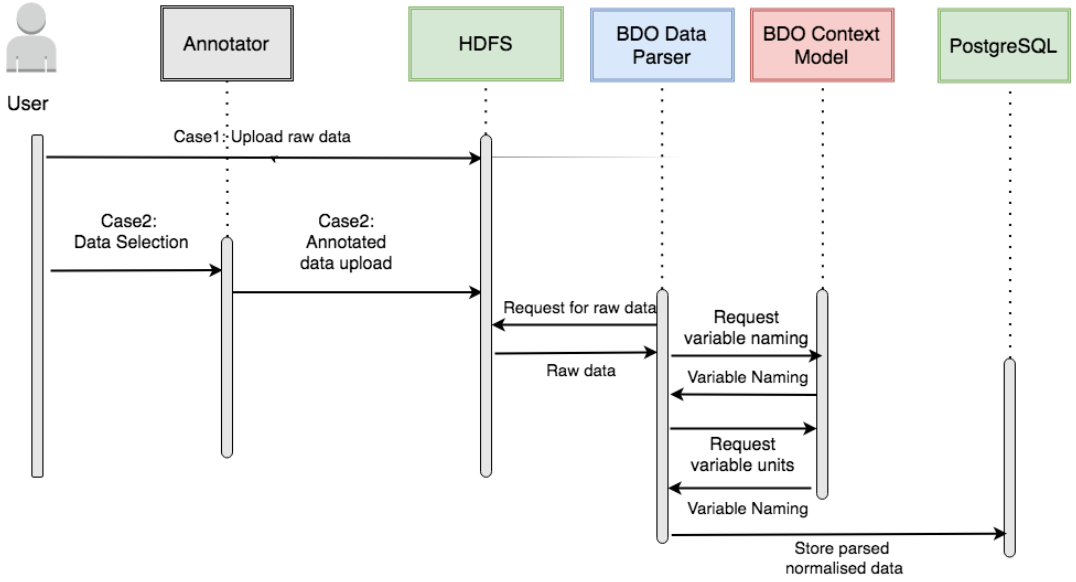

Figure 3-6: BDO Data Parser

[1] http://cfconventions.org/index.html

3.6.2 Interfaces

The following tables document the interface offered by the BDO Data Parser:

Endpoint Name	login
Description	Login to retrieve a valid JWT needed for all other endpoints
HTTP Method	POST
Endpoint URL	http://server_ip:port/login
Parameters	-

Request Body	{
	"username": "user",
	"password": "password"
	}

Table 3-26: BDO Data Parser login interface

Endpoint Name	uploadData
Description	Uploads a new dataset to HDFS
HTTP Method	POST
Endpoint URL	http:// server_ip:port/file/upload
Parameters	The actual file is needed in multipart/form-data form

Request Body	{
	"dataType": "csv",
	"observation": "forecast",
	"source": "hcmr"
	}

Table 3-27: Upload data interface

Endpoint Name	downloadData
Description	Store a dataset to HDFS, by downloading it from an external source
HTTP Method	POST
Endpoint URL	http:// server_ip:port/file/download
Parameters	-

Request Body	{
	"observation": "forecast",
	"source": "hcmr",
	"dataType": "netcdf",
	"downloadUrl": "http://....",
	"fileName": "test.nc"
	}

Table 3-28: Download data interface

Endpoint Name	findAllFiles
Description	Returns a list of all the files stored in HDFS
HTTP Method	GET
Endpoint URL	http:// server_ip:port/file
Parameters	-
Request Body	-

Table 3-29: Find all files interface

Endpoint Name	findFileById
Description	Returns information about a stored file, based on its unique ID
HTTP Method	GET
Endpoint URL	http:// server_ip:port/file/{id}
Parameters	id: The id of the file to look for
Request Body	-

Table 3-30: Find file by id interface

Endpoint Name	findFilesWithoutMetadata
Description	Returns a list of files stored in HDFS, that haven't been associated yet with any metadata.
HTTP Method	GET
Endpoint URL	http:// server_ip:port/file/metadata/empty
Parameters	-

Request Body	-

Table 3-31: Find files without metadata interface

Endpoint Name	updateMetadataId
Description	Updates the metadata repository ID of a given file
HTTP Method	PUT
Endpoint URL	http:// server_ip:port/file/{fileId}metadata/{metadataId}
Parameters	fileId: The id of the file to update metadataId: The id of the associated metadata repository
Request Body	-

Table 3-32: Upload metadata id interface

Endpoint Name	findAllParsableFiles
Description	Returns a list of all the files that are stored to HDFS and are ready to be sent to the parser, i.e. that some metadata linked with them
HTTP Method	GET
Endpoint URL	http:// server_ip:port/file/parsable
Parameters	-
Request Body	-

Table 3-33: Find all parsable files interface

Endpoint Name	findAllParsableFilesByObservation
Description	Returns a list of all the files that are stored to HDFS and are ready to be sent to the parser, matching the given observation type.
HTTP Method	GET
Endpoint URL	http:// server_ip:port/file/parsable/{observation}
Parameters	observation: To the observation to look for, e.g. "forecast", "timeseries", etc.
Request Body	-

Table 3-34: Find all parsable files by observation interface

Endpoint Name	findParsedFiles

Description	Returns a list of files that have been successfully parsed
HTTP Method	GET
Endpoint URL	http:// server_ip:port/file/parsed
Parameters	-
Request Body	-

Table 3-35: Find parsed files interface

Endpoint Name	findParsedFilesByObservation
Description	Returns a list of files that have been successfully parsed, matching the given observation type
HTTP Method	GET
Endpoint URL	http:// server_ip:port/file/parsed/{observation}
Parameters	observation: To the observation to look for, e.g. "forecast", "timeseries", etc.
Request Body	-

Table 3-36: Find parsed files by observation interface

Endpoint Name	parse
Description	Parses a file already stored in the HDFS and linked with metadata
HTTP Method	POST
Endpoint URL	http:// server_ip:port/parse/{fileId}
Parameters	fileId: The ID of the file to be parsed

Request Body	{
	"parallel": true
	}

Table 3-37: Parse interface

Endpoint Name	findAllVariables
Description	Returns a list of all the variables under consideration (the ones existing in BDO context model)
HTTP Method	GET

Endpoint URL	http:// server_ip:port/variable
Parameters	-
Request Body	-

Table 3-38: Find all variables interface

Endpoint Name	findVariableByName
Description	Returns information about a single variable, based on the variable's original name
HTTP Method	GET
Endpoint URL	http:// server_ip:port/variable/name/{name}
Parameters	name: The name of the variable to look for
Request Body	-

Table 3-39: Find variable by name interface

Endpoint Name	findVariableByCanonicalName
Description	Returns information about a single variable, based on the variable's canonical name
HTTP Method	GET
Endpoint URL	http:// server_ip:port/variable/canonicalName/{name}
Parameters	-
Request Body	-

Table 3-40: Find variable by canonical name interface

3.7 Query Builder

3.7.1 Overview

The Query Builder is the component responsible for allowing its users to explore, combine and express complex queries on BigDataOcean datasets. The user interface's ease of use ensures that end users unfamiliar with formal query languages or the structure of the underlying data will still be able to describe complex queries on top of multiple datasets. The produced queries will then be used in order to retrieve raw data from BigDataOcean. Moreover, these queries will be used by other components of the platform, such as the Visualiser, the Algorithms Customiser and the Algorithms Combiner, in order to retrieve the required data for a specific analysis or visualisation and they can also be used as templates that can be parameterised by the final users of the services so as to get

customised results. The Query Builder will also ensure that the generated queries will be used to provide data without any violation of data access policies defined within the platform.

The Query Builder will address the following requirements:

1. **DU-TR-1**: Query Builder will allow end users to perform simple and advanced queries over the stored big data.
2. **DU-TR-7**: Query Builder will allow end users to perform simple and advanced queries over different dataset originating from different source formats.
3. **DU-TR-8**: Query Builder will allow end users to execute geospatial queries.

The purpose of the Query Builder is to be simple to use, yet powerful enough in order to be able to describe the vast majority of queries that can be expected. The tool's strength will be the use of metadata provided by Metadata Repository in order to better predict the needs and intentions of users, help them with most common tasks and provide better insights in the domain of interlinking data from distinct datasets.

The execution flow of the Query Builder can be broken into the following parts, and an example is used to better demonstrate the proposed workflow:

- The user can search for datasets (to which he has access) or data variables that he wants to explore *(for example, the user search for "Wave Forecast" and finds related datasets)*.
- After that, he can select one or more variables to add them to the "Workbench" area, effectively adding them to the query *(in the example, the user picks the wave significant height variable and drags it into the Workbench)*.
- For the selected variables he can choose to get all the query results or if he wants to group them based on a dimension (i.e. timestamp) and use an aggregations function (i.e. average).
- Then, the query is executed and the retrieved results are displayed on a proper visualisation type that the user selects (*in the example, he selects a line chart*) or on a table.
- The selected variables can be filtered based on their different attributes of the variable *(for example, in the added wave significant height variable the user applies a chronological filter for a specific period on the time attribute and then orders the result from the largest to the smallest variable values)*.
- Moreover, the user can select to add more variables into the constructed query. When that is done, the different variables are joined on all their common dimensions (*in the example, the user selects to add the sea water temperature variable and the results are joined based on location and time, in order to get the values or the two variables for the same dimensions*).
- At any time, the user may again select to group or not the results based on a dimension or use an aggregation function.
- Finally, the user can perform some query-level configuration and execute the query to get the requested results (*for example, they may limit the number of results to 100, view a visualisation of the results and send the query to an analysis*).

A diagrammatic representation of the workflow described above is provided in the figure bellow:

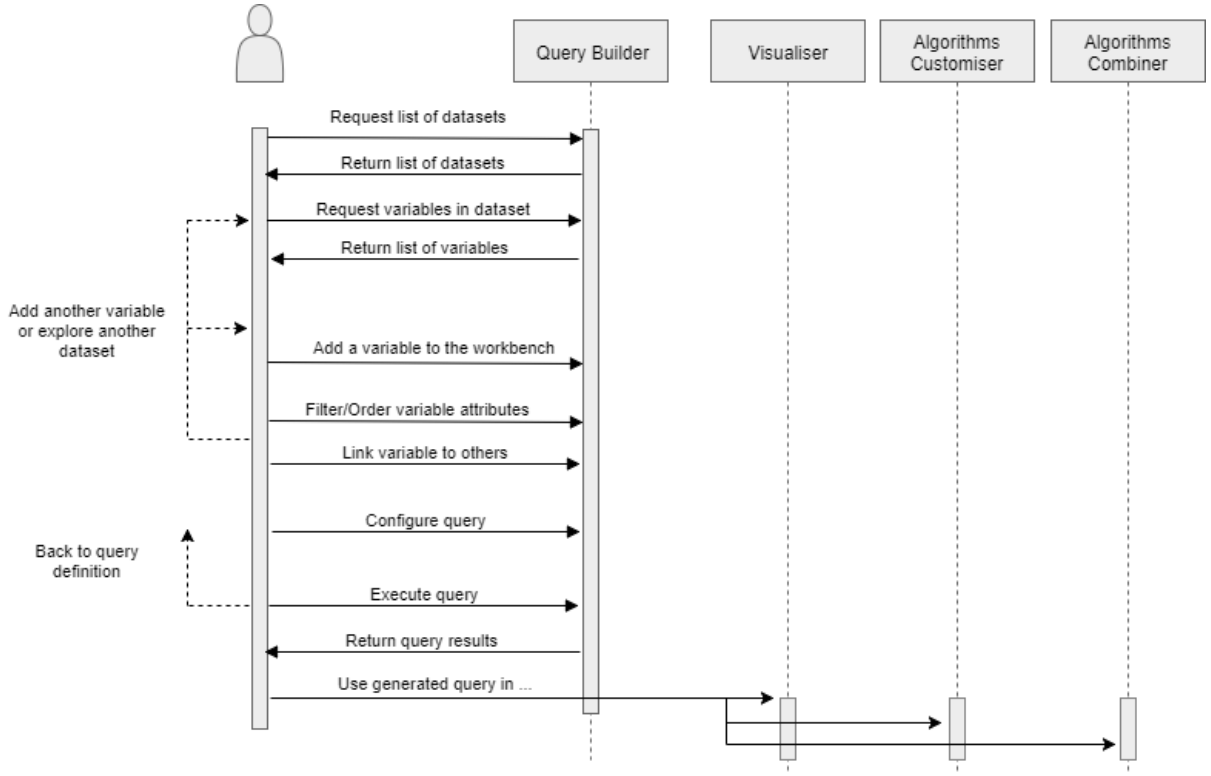

Figure 3-7: Query Builder Sequence diagram

3.7.2 Interfaces

The following tables document the interface offered by the Query Builder:

List the available datasets	
Description	Retrieve a list of all the available datasets that are available on the BigDataOcean platform. For each dataset, some metadata are provided, such as its title, the source/ provider of the dataset and a small description.
HTTP Method	GET
Endpoint URL	/queries/datasets/
Parameters	**-**
Request Body	-

Table 3-41: List available datasets interface

List the variables of a dataset	
Description	Retrieve a list with all the variables that are included in a selected dataset. For each variable, some metadata are provided, such as its name, the unit of measurement and its type (i.e. numeric).

HTTP Method	GET
Endpoint URL	/queries/datasets/<dataset_id>/variables/
Parameters	**dataset_id**: The identifier of the dataset from which the variables are obtained.
Request Body	-

Table 3-42: List variables of a dataset interface

Count the rows of a variable	
Description	Count the number of rows of a specific variable, inside a specific dataset.
HTTP Method	GET
Endpoint URL	/queries/datasets/<dataset_id>/variables/<variable_id>/count/
Parameters	**dataset_id**: The identifier of the dataset from which the variables are obtained. **variable_id**: The identifier of the variable that is counted.
Request Body	-

Table 3-43: Count the rows of a variable interface

List the saved queries	
Description	Get a list of all the saved queries on the platform. For each query, its id and name are listed, along with its creation date and the date of its most recent modification.
HTTP Method	GET
Endpoint URL	/queries/list/
Parameters	**-**
Request Body	-

Table 3-44: List saved queries interface

Get the variables and dimensions of a query	
Description	For a specific query, get all the variables and dimensions that it returns when executed. For each variable and dimension, its name is followed by the alias that is used in the SQL query statement.
HTTP Method	GET
Endpoint URL	/queries/get_query_variables/

Parameters	**id:** the identifier of the query that is used to get its variables and dimensions
Request Body	-

Table 3-45: Get variables and dimensions of a query interface

Execute a query	
Description	Execute a specific query on the platform. When the query is executed the results are obtained, which are the following: • Headers: some metadata about the execution of the query, such as the execution time in milliseconds and the columns returned. For each column, information about the corresponding variable or dimension is listed. • Results: a list of the rows that are retrieved by the query execution
HTTP Method	GET
Endpoint URL	/queries/execute/<query_id>/
Parameters	**query_id:** the identifier of the query that is executed
Request Body	-

Table 3-46: Execute a query interface

3.8 Algorithms Controller

3.8.1 Overview

The Algorithms Controller is the component responsible for initiating and monitoring the execution of the requested algorithm or service. This component is acting as the supervisor of the algorithm execution for the data analysis. The role of this component is very crucial for addressing the stakeholder's needs. Algorithms Controller should be able to perform advanced analytics over multiple datasets that are available on the Big Data Storage with a wide range of algorithms and provide results in a timely and efficient manner.

The algorithms controller addresses the following requirements:

1. **DAN-TR-1**: The algorithms controller facilitates the performance of big data analytic algorithms on the data stored/ linked in the BigDataOcean platform including descriptive, predictive, and prescriptive analytics.
2. **DAN-TR-2**: The algorithms controller facilitates the storage of results of the analyses as datasets that can be further analysed with other algorithms.
3. **DAN-TR-3**: The algorithms controller is performed on a scalable cluster-computing framework, programming entire clusters with implicit data parallelism and fault-tolerance.
4. **DAN-TR-4**: The algorithms controller has the capability to perform streaming analytics.

A diagrammatic representation of the workflow described above is provided in the figure bellow:

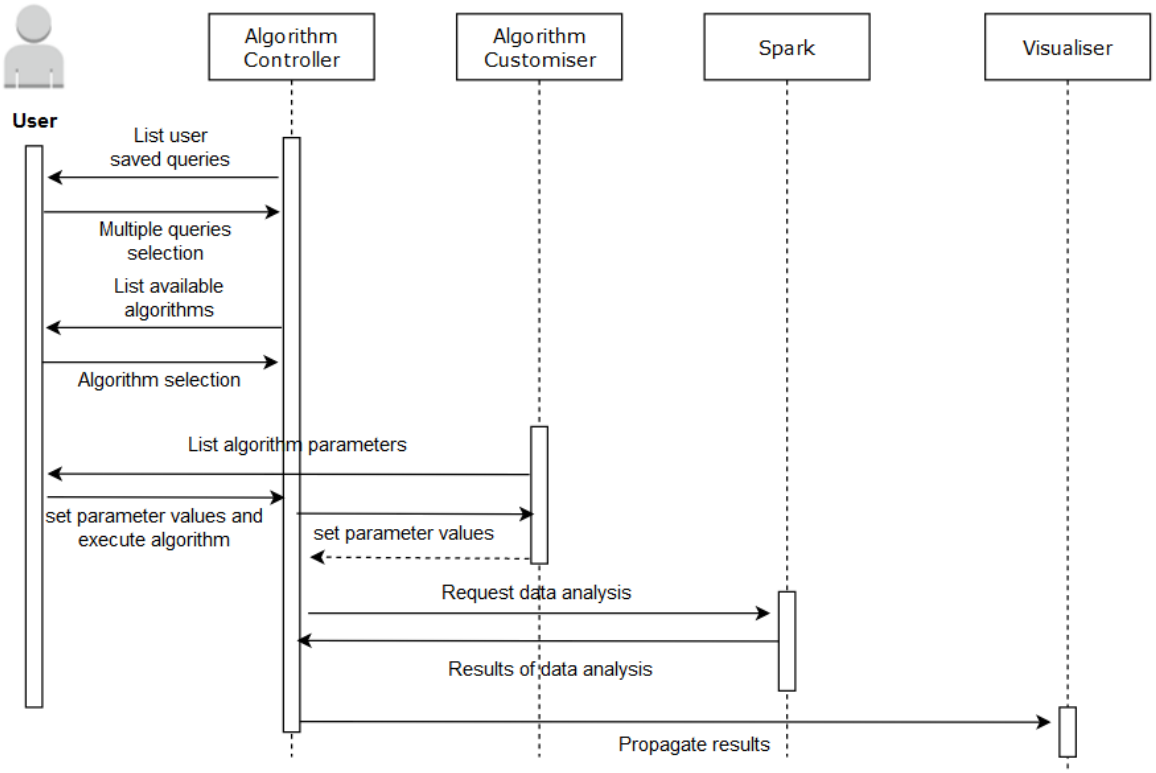

Figure 3-8: Algorithms Controller Sequence diagram

3.8.2 Interfaces

The following tables document the interface offered by the Algorithms Controller:

List the available algorithms	
Description	Retrieve a list of all the available algorithms that are available on the BigDataOcean platform. For each algorithm, some metadata are provided, such as its title, a description and the provider.
HTTP Method	GET
Endpoint URL	/services/
Parameters	**-**
Request Body	-

Table 3-47: List available algorithms interface

Algorithm execution	
Description	Execute a specific algorithm of the platform. The selected algorithm is customised with the selected values of the required arguments and then is executed. The response includes either the results of the algorithm execution or a link to the page where the result is available (if for example is a visualisation)

HTTP Method	POST
Endpoint URL	/services/<algorithm_id>/execute/
Parameters	**algorithm_id:** the identifier of the selected algorithm. argument_name:
Request Body	{ argument_name_1: value_1, argument_name_2: value_2, … }

Table 3-48: Algorithm execution interface

3.9 Algorithms Customiser

3.9.1 Overview

The Algorithms Customiser is a supplementary component of the Algorithms Controller that is responsible for providing the parameter values of the algorithm selected by the user, provided that the algorithm requires parameters. The role of Algorithms Customiser is very concrete and crucial as it will provide the capability of customising the algorithm's parameters thus the ability of personalising the algorithm upon the user's needs towards the extraction of valuable results and analysis.

The core functionality of this component will be to initially provide a set of default values for the parameters of the algorithm. Following the suggestion of the default values, the user will be able to customise the parameter values according to the needs of the user's analysis. Furthermore, it shall be possible for the user to save the parameter values used for later reference or reuse.

The Algorithms Customiser addresses the following requirements:

1. **DAN-TR-1**: The Algorithms Customiser facilitates the customisation of big data analytic algorithms to be performed on the data stored/ linked in the BigDataOcean platform.
2. **DAN-TR-4**: The Algorithms Customiser has the capability to customise algorithms supporting streaming analytics.

Figure 3-9 displays the execution flow of the Algorithms Customiser.

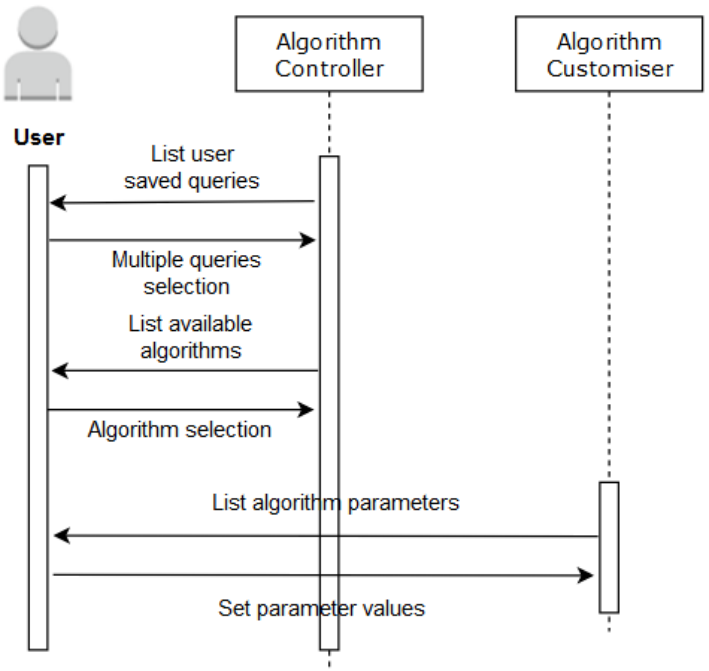

Figure 3-9: Algorithms Customiser Sequence Diagram

3.9.2 Interfaces

The following table documents the interface offered by the Algorithms Customiser:

List the arguments of an algorithm	
Description	Retrieve a list with all the arguments that are required for the customisation and execution of specific algorithms. For each argument, information is provided like its name, its type, a description and the default value.
HTTP Method	GET
Endpoint URL	/services/<algorithm_id>/arguments/
Parameters	**algorithm_id:** the identifier of the selected algorithm.
Request Body	-

Table 3-49: List the arguments of an algorithm interface

3.10 Algorithms Combiner

3.10.1 Overview

The Algorithms Combiner is another supplementary component of the Algorithms Controller that is offering the capability of executing algorithms in a chainable way. This will add further value to the BigDataOcean platform as it extends the analysis that can be performed.

More specific, the Algorithms Combiner will provide the mechanism to accomplish further analysis by executing a selected algorithm on the results of the execution of a previously selected algorithm, thus

creating a chain of algorithms execution. Once the results of the selected algorithm are ready the Algorithms Combiner will present the option to execute a new algorithm over the produced results.

The Algorithms Combiner addresses the following requirements:

1. **DAN-TR-1**: The Algorithms Combiner facilitates the performance of big data analytic algorithms on the data stored/ linked in the BigDataOcean platform including descriptive, predictive, and prescriptive analytics.
2. **DAN-TR-2**: The Algorithms Combiner facilitates the storage of results of the analyses as datasets that can be further analysed with other algorithms.
3. **DAN-TR-3**: The Algorithms Combiner is performed on a scalable cluster-computing framework, programming entire clusters with implicit data parallelism and fault-tolerance.
4. **DAN-TR-4**: The Algorithms Combiner has the capability to perform streaming analytics.

The following figure displays the execution flow of the Algorithm Combiner.

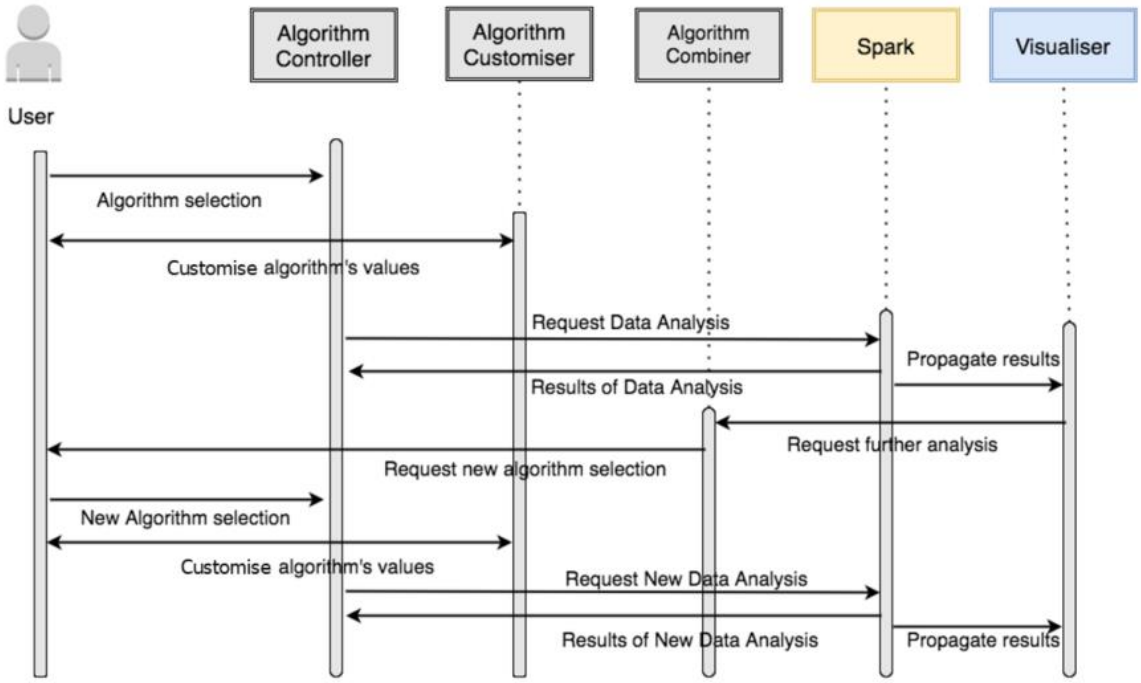

Figure 3-10: Algorithms Combiner Sequence Diagram

3.10.2 Interfaces

In the first version of the Algorithms Combiner, the availability of an API is not foreseen. However, as the project evolves and based on the upcoming needs of the project, this decision will be reconsidered.

3.11 Visualiser

3.11.1 Overview

The Visualiser is the component responsible for the visual representation of the results produced from the other components. It addresses the following requirements:

1. **DU-TR-2:** Visualiser is able to visualise stored and real-time data using common chart types (bar chart, line chart, pie chart, etc.).
2. **DU-TR-3:** Visualiser refers to user-defined queries in order to access the underlying data.
3. **DU-TR-4:** Various data such as data aggregations, real-time data and spatial datasets can be fed into the Visualiser.
4. **DU-TR-5:** Visualiser facilitates the creation of static and dynamic dashboards and reports.
5. **DU-TR-6:** Visualiser allows users to export data and results of visualisations through APIs.

Computer-based visualisation systems provide visual representations of queries designed to help people carry out tasks more effectively. Particularly this component will offer a variety of visual representations including bar and scatter plots, pie charts, heat maps, etc. based on the execution output of queries composed by the Query Builder component. It will also display the results of the analysis executed by the Algorithm Controller component. The visualisations will be provided within various contexts within the BigDataOcean platform, including:

- The Query Builder in order to allow users to generate visualisations of the explored data.
- Analytics will use the Visualiser to provide insights into the results of the executed analysis.
- Reports and dashboards will include results of the Visualiser in order to become more interactive and intuitive for the end user.
- The results of this component will be also available for export as static images or embeddable assets.

The following figure displays the execution flow of the Visualiser.

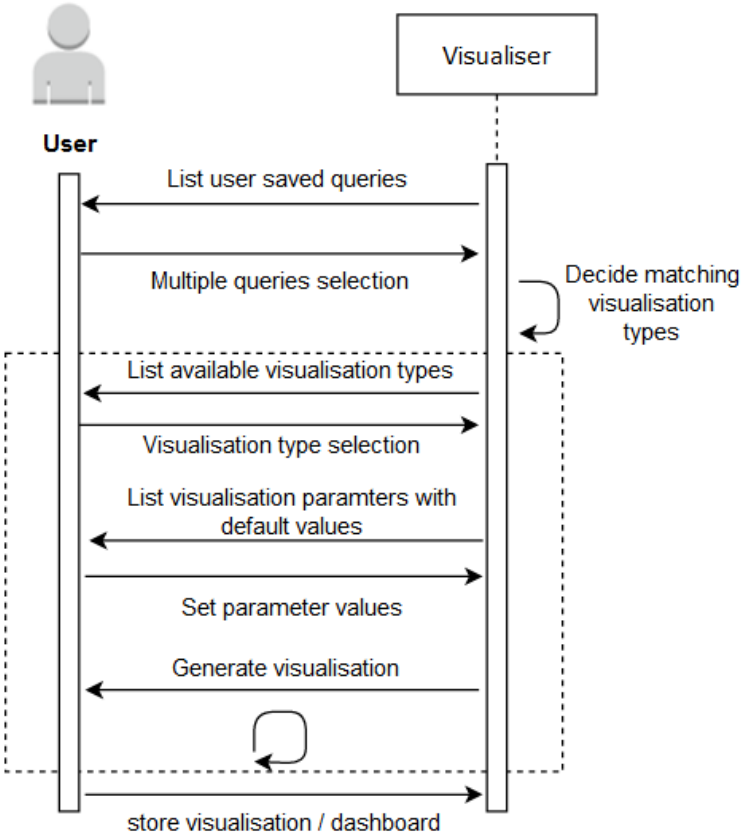

Figure 3-11: Visualiser Sequence Diagram

3.11.2 Interfaces

The following tables document the interface offered by the Visualiser:

List the available visualisation types	
Description	Retrieve a list of all the visualisation types that are available on the BigDataOcean platform. For each algorithm, some metadata are provided, such as its title and a description.
HTTP Method	GET
Endpoint URL	/visualisations/list/
Parameters	**-**
Request Body	-

Table 3-50: List available visualisation types interface

List the parameters of a visualisation type	
Description	Retrieve a list with all the parameters that are required for the customisation of a specific visualisation type. For each parameter, information is provided like its name, its type, a description and the default value.
HTTP Method	GET
Endpoint URL	/visualisations/<visualisation_type_id>/parameters/
Parameters	**visualisation_type_id:** the identifier of the selected visualisation type.
Request Body	-

Table 3-51: List the parameters of the visualisation type

Line chart	
Description	Returns a line-chart visualisation that includes all the data retrieved from the execution of the query the user created or a dataframe after an algorithm execution.
HTTP Method	GET
Endpoint URL	/visualisations/get_line_chart/
Parameters	**query:** the identifier of the query to be used to retrieve the data of the visualisation. **df:** the identifier of the spark or pandas dataframe from which the data of the visualisation will be retrieved. **note_id:** the identifier of the note where the dataframe. (the df and note_id parameters are used instead of query)

	y_var: the variable(s) to be used for the vertical axis of the visualisation. More than one variables can be provided, separated by commas. **x_var:** the variable/ dimension to be used for the vertical axis of the visualisation.
Request Body	-

Table 3-52: Line chart interface

Column chart	
Description	Returns a column-chart visualisation that includes all the data retrieved from the execution of the query the user created or a dataframe after an algorithm execution.
HTTP Method	GET
Endpoint URL	/visualisations/get_column_chart/
Parameters	**query:** the identifier of the query to be used to retrieve the data of the visualisation. **df:** the identifier of the spark or pandas dataframe from which the data of the visualisation will be retrieved. **note_id:** the identifier of the note where the dataframe. (the df and note_id parameters are used instead of query) **y_var:** the variable(s) to be used for the vertical axis of the visualisation. More than one variables can be provided, separated by commas. **x_var:** the variable/ dimension to be used for the vertical axis of the visualisation.
Request Body	-

Table 3-53: Column chart interface

Pie chart	
Description	Returns a pie-chart visualisation that includes all the data retrieved from the execution of the query the user created or a dataframe after an algorithm execution.
HTTP Method	GET
Endpoint URL	/visualisations/get_pie_chart/
Parameters	**query:** the identifier of the query to be used to retrieve the data of the visualisation. **df:** the identifier of the spark or pandas dataframe from which the data of the visualisation will be retrieved. **note_id:** the identifier of the note where the dataframe. (the df and note_id parameters are used instead of query) **key_var:** the variable/ dimension to be used to separate the data into categories.

	value _var: the variable to be used as value for each category.
Request Body	-

Table 3-54: Pie chart interface

Filled contours on the map	
Description	Returns a contour map visualisation created according to the data retrieved from the execution of the query the user created or a dataframe after an algorithm execution
HTTP Method	GET
Endpoint URL	/visualisations/get_map_contour/
Parameters	**query:** the identifier of the query to be used to retrieve the data of the visualisation. **df:** the identifier of the spark or pandas dataframe from which the data of the visualisation will be retrieved. **note_id:** the identifier of the note where the dataframe. (the df and note_id parameters are used instead of query) **feat:** the name of the physical quantity that is presented in the contours. **step:** carries the value of the step used in the division of the area of interest into a grid. **agg**: the aggregation function to be used for the aggregation of data that fall into the same grid's cell. **n_contours:** the number of colour samples that are generated for the different contours.
Request Body	-

Table 3-55: Filled contours on a map interface

Vessel course on the map	
Description	Returns a map visualisation created according to the data retrieved from the execution of the query the user created or a dataframe after an algorithm execution, where a number of markers are displayed according to the most recent positions of a vessel that are present on the data.
HTTP Method	GET
Endpoint URL	/visualizations/map_course/
Parameters	**query:** the identifier of the query to be used to retrieve the data of the visualisation. **df:** the identifier of the spark or pandas dataframe from which the data of the visualisation will be retrieved. **note_id:** the identifier of the note where the dataframe. (the df and note_id parameters are used instead of query)

	markers: the number of the most recent positions to be displayed. **vessel_id:** the identifier of the vessel whose positions are displayed.
Request Body	-

Table 3-56: Vessel course on a map interface

3.12 Integrated Security Design

3.12.1 Overview

In deliverable D4.2 the integrated security design that will span across the whole platform and not be developed as an autonomous and individual component was presented. This design is assuring that every designed component will incorporate the security mechanisms and protocols identified in order to provide the required high-level of security as envisioned by the consortium. Moreover, this security design also covers all the designed workflows between the various components towards the aim of covering all security aspects which include security of data in storage, in transit and in use.

The security aspect of data in storage refers to any dataset or metadata stored within the BigDataOcean platform, particularly in Big Data Storage. The Big Data Storage is composed of two storage solutions, namely the Hadoop Distributed File System (HDFS) and the PostgreSQL, and each of these solutions offers the required level of security with a variety of mechanisms for authentication, access control, service level authorisation and data confidentiality that are exploited in order to cover this security aspect. Several other encryption technologies such as Symmetric and Asymmetric Encryption Algorithms as well as Attribute-Based Encryption that were initially identified as candidates were not adopted as they are considered not suitable for big data platforms causing a significant decrease in the performance of the platform, especially in the data analysis process.

For the security aspect of data in transit which refers to any data being transferred over the internal or the external network of the BigDataOcean platform and that is considered nowadays as the most critical aspect of any platform the consortium has identified the security mechanisms of IPsec and TLS. These mechanisms have been already adopted by some the components of the platform like Big Data Framework and the Big Data Storage in the first version of platform and will be incorporated in the remaining components in the upcoming versions of the platform.

The third aspect of the integrated security design is referring to the security of data in use, in particular to any data currently stored locally usually for internal use like in resident memory, or swap, or processor cache or disk cache. Although two security mechanisms have been identified, namely the Homomorphic Encryption and Verifiable Computation, the evaluation and incorporation of these mechanisms in the BigDataOcean platform will be part of the upcoming versions of the platform.

The Integrated Security Design addresses the following requirement:

1. **DS-TR-6:** The Integrated Security Design offers native security and encryption mechanisms.

4 Conclusions

The scope of D4.3 was to document the efforts undertaken in the context of all tasks of WP4, namely the Task 4.1, the Task 4.2, the Task 4.3 and the Task 4.4.

The main objective of the current deliverable was to provide the supplementary documentation on top of the information provided within the context of D4.2 by presenting the necessary updates on the technical requirements and the high-level architecture of the platform together with the updates on the design and functionalities of the platform's components. These updates and optimisations were introduced as a result of the comprehensive analysis of the feedback received by the pilots of the projects on the first mock-up version of the platform that was released in M14.

At first, the new technical requirements that have been identified were presented. These additional requirements were the direct output of the analysis of the feedback received by the pilots. The high-level architecture of the BigDataOcean platform received the necessary optimisations and adjustments in order to properly address these requirements and safeguard that the new stakeholder's needs are included in the platform. The description of the updated high-level architecture was focused on the roles and responsibilities undertaken by each component in the course of providing the designed platform functionalities to the end-users. Furthermore, the current document presented the updated documentation of the platform's components taking into consideration the optimisations and customisations introduced in the design and specifications of the components. Additionally, the technical interfaces of the components that enabled their integration into the BigDataOcean platform were documented.

The design of the architecture of the BigDataOcean platform is a living process that will last until M26. As the platform matures additional technical requirements will be collected that will be translated into new platform functionalities. Thus, the platform's architecture will receive further optimisations and customisations to handle the new stakeholders' needs. Moreover, additional feedback will be collected for the upcoming releases of the platform that will drive the updates and customisations on both the platform and the components' architecture in the future.

Annex I: Mapping Requirements to Components

Code	Technology Requirement	Priority	Components Responsible
DAC-TR-1	The system should be able to connect and acquire data from sensors and IoT devices of different manufacturers in real-time or intervals.	Medium	Big Data Framework & Storage
DAC-TR-2	The system should be able to obtain data from AIS receivers, transponders or base stations.	High	Big Data Framework & Storage
DAC-TR-3	The system should be able to acquire data from large and popular oceanographic databases including Copernicus through APIs.	High	Big Data Framework & Storage
DAC-TR-4	The system should be able to upload big data datasets on BigDataOcean infrastructure.	High	Big Data Framework & Storage
DAC-TR-5	The system should be able to link big data sources to BigDataOcean Infrastructure.	High	Annotator
DAN-TR-1	The system should be able to perform big data analytic algorithms on the data stored/ linked in the BigDataOcean platform including descriptive, predictive, and prescriptive analytics.	High	Algorithms Controller, Algorithms Customiser, Algorithms Combiner
DAN-TR-2	The system should be able to save results of the analyses as datasets that can be further analysed with other algorithms.	High	Algorithms Controller, Algorithms Combiner
DAN-TR-3	The data analysis framework should be performed on a scalable cluster-computing framework, programming entire clusters with implicit data parallelism and fault-tolerance.	High	Big Data Framework & Storage, Algorithms Controller, Algorithms Combiner
DAN-TR-4	The BDO analytics should have the capability to perform streaming analytics.	Medium	Algorithms Controller, Algorithms Customiser, Algorithms Combiner
DC-TR-1	The system should be able to cleanse the datasets (e.g. remove duplicates, inconsistent values, identify outliers, values with wrong types).	Medium	Cleanser
DC-TR-2	The system should be able to curate large scale (big data) datasets in a timely and efficient manner.	Medium	Big Data Framework & Storage, BDO Data Parser

DC-TR-3	The system should be able to store the processes followed for the data curation.	Medium	Cleanser, Annotator
DC-TR-4	The system should allow for the efficient transformation of data: converting values to other formats, normalising and de-normalising.	Medium	Metadata Repository
DC-TR-5	The system should be able to program the semantic curation/ reconciliation of datasets.	Medium	Metadata Repository, Annotator
DC-TR-6	The data curation tools can be used offline on the client side.	Medium	Annotator, Cleanser
DC-TR-7	The system should be able to parse and normalise data in structured format (for example netCDF files).	High	BDO Data Parser, Annotator
DC-TR-8	The system should be able to parse and normalise data in semi-structured format (for example CSV, Excel, KML files).	High	BDO Data Parser, Annotator
DS-TR-1	The system should be able to store and manage large datasets (big data).	High	Big Data Framework & Storage, BDO Data Parser
DS-TR-2	Setup of different data storage environments should be allowed in parallel in order to accommodate different storage needs (Relational Databases, NoSQL, TripleStores).	High	Big Data Framework & Storage
DS-TR-3	The system should a have high degree of scalability.	High	Big Data Framework & Storage
DS-TR-4	The system should allow for replication and high availability with no single point of failure.	High	Big Data Framework & Storage
DS-TR-5	The system should support distributed storage and auto-sharing.	Medium	Big Data Framework & Storage
DS-TR-6	The system should offer native security and encryption mechanisms.	High	Integrated Security Design
DS-TR-7	The system should have a direct pipeline with the chosen cluster computing framework and analytics engine.	Medium	Big Data Framework & Storage
DS-TR-8	The system should be able to optimise the storage of spatio-temporal data facilitating their more efficient query execution.	High	Big Data Framework & Storage
DU-TR-1	The system should be able to perform simple and advanced queries over the stored big data.	High	Query Builder

DU-TR-2	The system should be able to visualise stored and real-time data using common chart types (bar chart, line chart, pie chart, etc.).	High	Visualiser
DU-TR-3	The visualisations should be linked to the queries performed over the data.	High	Visualiser
DU-TR-4	The system should be able to visualise statistics, real-time data and spatial datasets.	High	Visualiser
DU-TR-5	The system should be able to create and save static and dynamic dashboards and reports.	High	Visualiser
DU-TR-6	The system should be able to export data and results of visualisations through APIs.	High	Visualiser
DU-TR-7	The system should be able to perform simple and advanced queries over different dataset originating from different source formats.	High	Query Builder
DU-TR-8	The system should be able to perform spatio-temporal query execution	High	Query Builder

Annex II: Mapping Components to Requirements

Component	Requirement Code	Requirement Description
Anonymiser	N/A	The Anonymiser currently does not address any specific technical requirements, nevertheless it has been included as an optional component because such a need may arise in the future for the provision of a service through the platform.
Cleanser	DC-TR-1	The system should be able to cleanse the datasets (e.g. remove duplicates, inconsistent values, identify outliers, values with wrong types).
	DC-TR-3	The system should be able to store the processes followed for the data curation.
	DC-TR-6	The data curation tools can be used offline at the client side.
Metadata Repository	DC-TR-4	The system should allow for efficient transformation of data: converting values to other formats, normalising and de-normalising.
	DC-TR-5	The system should be able to program the semantic curation/ reconciliation of datasets.
	DAC-TR-5	The system should be able to link big data sources to BigDataOcean Infrastructure.
Annotator	DC-TR-3	The system should be able to store the processes followed for the data curation.
	DC-TR-5	The system should be able to program the semantic curation/ reconciliation of datasets.
	DC-TR-6	The data curation tools can be used offline at the client side.
	DC-TR-7	The system should be able to parse and normalise data in structured format (for example netCDF files).
	DC-TR-8	The system should be able to parse and normalise data in semi-structured format (for example CSV, Excel, KML files).
Big Data Framework & Storage	DAC-TR-1	The system should be able to connect and acquire data from sensors and IoT devices of different manufactures in real-time or intervals.

	DAC-TR-2	The system should be able to obtain data from AIS receivers, transponders or base stations.
	DAC-TR-3	The system should be able to acquire data from large and popular oceanographic databases including Copernicus through APIs.
	DAC-TR-4	The system should be able to upload big data datasets on BigDataOcean infrastructure.
	DAN-TR-3	The data analysis framework should be performed on a scalable cluster-computing framework, programming entire clusters with implicit data parallelism and fault-tolerance.
	DC-TR-2	The system should be able to curate large scale (big data) datasets in a timely and efficient manner.
	DS-TR-1	The system should be able to store and manage large datasets (big data).
	DS-TR-2	Setup of different data storage environments should be allowed in parallel in order to accommodate different storage needs (Relational Databases, NoSQL, TripleStores).
	DS-TR-3	The system should have high degree of scalability.
	DS-TR-4	The system should allow for replication and high availability with no single point of failure.
	DS-TR-5	The system should support distributed storage and auto-sharing.
	DS-TR-7	The system should have a direct pipeline with the chosen cluster computing framework and analytics engine.
	DS-TR-8	The system should be able to optimise the storage of spatio-temporal data facilitating their more efficient query execution
BDO Data Parser	DC-TR-2	The system should be able to curate large scale (big data) datasets in a timely and efficient manner.
	DC-TR-7	The system should be able to parse and normalise data in structured format (for example netCDF files).
	DC-TR-8	The system should be able to parse and normalise data in semi-structured format (for example CSV, Excel, KML files).
	DS-TR-1	The system should be able to store and manage large

		datasets (big data).
Query Builder	DU-TR-1	The system should be able to perform simple and advanced queries over the stored big data.
	DU-TR-7	The system should be able to perform simple and advanced queries over different dataset originating from different source formats.
	DU-TR-8	The system should be able to perform spatio-temporal query execution
Algorithms Controller	DAN-TR-1	The system should be able to perform big data analytic algorithms on the data stored/ linked in the BigDataOcean platform including descriptive, predictive, and prescriptive analytics.
	DAN-TR-2	The system should be able to save results of the analyses as datasets that can be further analysed with other algorithms.
	DAN-TR-3	The data analysis framework should be performed on a scalable cluster-computing framework, programming entire clusters with implicit data parallelism and fault-tolerance.
	DAN-TR-4	The BDO analytics should have the capability to perform streaming analytics.
Algorithms Customiser	DAN-TR-1	The system should be able to perform big data analytic algorithms on the data stored/ linked in the BigDataOcean platform including descriptive, predictive, and prescriptive analytics.
	DAN-TR-4	The BDO analytics should have the capability to perform streaming analytics.
Algorithms Combiner	DAN-TR-1	The system should be able to perform big data analytic algorithms on the data stored/ linked in the BigDataOcean platform including descriptive, predictive, and prescriptive analytics.
	DAN-TR-2	The system should be able to save results of the analyses as datasets that can be further analysed with other algorithms.
	DAN-TR-3	The data analysis framework should be performed on a scalable cluster-computing framework, programming entire clusters with implicit data parallelism and fault-tolerance.
	DAN-TR-4	The BDO analytics should have the capability to

		perform streaming analytics.
Visualiser	DU-TR-2	The system should be able to visualise stored and real-time data using common chart types (bar chart, line chart, pie chart, etc.).
	DU-TR-3	The visualisations should be linked to the queries performed over the data.
	DU-TR-4	The system should be able to visualise statistics, real-time data and spatial datasets.
	DU-TR-5	The system should be able to create and save static and dynamic dashboards and reports.
	DU-TR-6	The system should be able to export data and results of visualisations through APIs.
Integrated Security Design	DS-TR-6	The system should offer native security and encryption mechanisms.