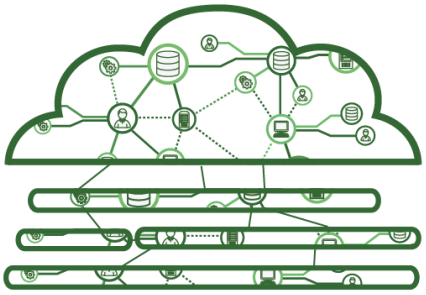


DITAS

Data-intensive applications Improvement by moving daTA and computation in mixed cloud/fog environmentS

D3.2 Data Virtualization SDK prototype (initial version)

Project Acronym	**DITAS**
Project Title	Data-intensive applications Improvement by moving daTA and computation in mixed cloud/fog environmentS
Project Number	731945
Instrument	Collaborative Project
Start Date	01/01/2017
Duration	36 months
Thematic Priority	ICT-06-2016 Cloud Computing
Website:	http://www.ditas-project.eu

Dissemination level: Public

Work Package	**WP3 Data Virtualization**
Due Date:	M16
Submission Date:	30/04/2018
Version:	1.0
Status	Final
Author(s):	Alexandros Psychas, Achilleas Marinakis, George Chatzikyriakos (ICCS), David García Pérez, Jose Antonio Sanchez (ATOS), Marco Peise, Sebastian Werner (TUB), Maya Anderson (IBM), Mattia Salnitri, Giovanni Meroni (POLIMI)
Reviewer(s)	Pierluigi Plebani (POLIMI), Peter Gray (CS)

This project has received funding by the European Union's Horizon 2020 research and innovation programme under grant agreement No. 731945

Version History

Version	Date	Comments, Changes, Status	Authors, contributors, reviewers
0.1	05/03/2018	ToC creation	Alexandros Psychas (ICCS)
0.2	22/03/2018	Blueprint sections 1,5	Achilleas Marinakis (ICCS)
0.3	10/04/2018	Blueprint section 2, Data Utility Resolution Engine, Running Example	Mattia Salnitri, Giovanni Meroni (POLIMI)
0.4	12/04/2018	Blueprint Section 2	Marco Peise, Sebastian Werner (TUB)
0.5	16/04/2018	Blueprint Section 3, Blueprint Filtering	Alexandros Psychas (ICCS), Mattia Salnitri (POLIMI)
0.6	18/04/2018	Blueprint Repository Engine Operations	Achilleas Marinakis, George Chatzikyriakos (ICCS)
0.7	19/04/2018	Blueprint section 1, VDC implementation	Maya Anderson (IBM)
0.8	19/04/2018	Blueprint section 4	David García Pérez, Jose Antonio Sanchez (ATOS)
0.9	20/04/2018	Internal review version	Alexandros Psychas, Achilleas Marinakis, George Chatzikyriakos (ICCS)
0.91	24/04/2018	Internal review comments	Pierluigi Plebani (POLIMI), Peter Gray (CS)
0.92	26/04/2018	Addressed internal review comments	Alexandros Psychas, Achilleas Marinakis, George Chatzikyriakos (ICCS)
1.0	27/04/2018	Final version for submission	Alexandros Psychas, Achilleas Marinakis, George Chatzikyriakos (ICCS)

© Main editor and other members of the DITAS consortium

Contents

Version History 2
List of Figures 4
List of tables 4
Executive Summary 5
1 Introduction 6
1.1 Glossary of Acronyms 8
2 Running Example 10
2.1 General Description 10
2.2 Dataset Publication 10
2.3 Access Policies 11
2.4 Application Designers 11
2.5 Application Developer 11
2.6 Application Execution 12
2.7 Focus of this Deliverable 12
3 VDC Blueprint Schema 13
3.1 Internal Structure (Blueprint Section 1) 13
3.2 Data Management (Blueprint Section 2) 16
3.3 Abstract Properties (Blueprint Section 3) 20
3.4 Cookbook Appendix (Blueprint Section 4) 22
3.5 Exposed API (Blueprint Section 5) 26
4 VDC Blueprint Repository & Resolution Engine 28
4.1 VDC Blueprint Repository Engine 28
4.1.1 Blueprint Creation 28
4.1.2 Blueprint Retrieval 28
4.1.3 Compenet API 29
4.2 VDC Blueprint Resolution Engine 33
4.2.1 Content based resolution 34
4.2.2 Data utility Resolution Engine 36
4.2.3 Product Based Recommendation 38
5 VDC Implementation 41
6 Conclusions 43
References 44
Annex A. VDC Blueprint Complete Schema 45
Annex B. VDC Blueprint Example 54

© Main editor and other members of the DITAS consortium

List of Figures

Figure 1: VDC Blueprint Lifecyle 7
Figure 2: Representation of a goal model 22
Figure 3: VDC Blueprint creation sequence 28
Figure 4: VDC Blueprint filtering processes 34
Figure 5: Content based resolution architecture 34
Figure 6: Data Utility resolution sequence diagram 36
Figure 7: Product based recommendations architecture 39
Figure 8: Virtual access to data in the federated store 41

List of tables

Table 1. Acronyms 9
Table 2. Internal Structure parameters(Overview) 15
Table 3. Internal Structure parameters (Data Sources) 16
Table 4. Internal Structure parameters (Flow) 16
Table 5. Internal Structure parameters (Testing Output Data) 16
Table 6. Data Management parameters for data quality 18
Table 7. Data Management parameters for privacy and security 20
Table 8. Abstract Properties parameters 22
Table 9. Cookbook Appendix parameters 25
Table 10. Exposed API parameters(Methods) 27
Table 11. Get Blueprint method documentation 29
Table 12. Get Blueprint by id method documentation 30
Table 13. Post Blueprint method documentation 31
Table 14. Update Blueprint method documentation 33
Table 15. Delete Blueprint method documentation 33
Table 16. Search Blueprint method documentation 36
Table 17. Rank Blueprint method documentation 38

© Main editor and other members of the DITAS consortium

Executive Summary

This document is responsible for describing all the technical information about the components created in order to describe and manage the Virtual Data Container (VDC). More specifically, is contains a thorough technical description of said components as well as the API for the communication, the architectural position and any interactions between components. First of all, a major part of this document contains information about the VDC Blueprint Schema which is created in order to unify and formally describe all the essential information about the VDC structure. The Blueprint is separated in different sections and each and every one of them is described in this document by what their utility is on the overall Blueprint and a strictly defined JSON schema. Every section is created in order to describe different aspects and functionalities of the Blueprint. Moreover, the mechanism that enables the storage of all this information as well as the CRUD operations and methods that are exposed in order to be used by other components are thoroughly analyzed. Another crucial subject that is being examined in this document is all the process end mechanisms that are created or going to be created in order to help the Application Developer in selecting the appropriate VDC that corresponds to the application needs and his/her requirements. Finally, there is a detailed description of the running example and how the VDC is actually implemented based on that.

© Main editor and other members of the DITAS consortium

1 Introduction

Before going in detail on how the Virtual Data Container (VDC) is described in the form of a Blueprint and how all the processes in order to create, retrieve and select a VDC are realized it is important to highlight what the VDC is and what purpose it fulfills. As it is stated in the deliverable D3.1 [D3.1] the VDC provides an abstraction layer that takes care of retrieving, processing and delivering data with the proper quality level, while in parallel putting special emphasis on data security, performance, privacy, and data protection. The VDC, acting as a middleware, lets the developers simply define the requirements on the needed data, expressed as data utility, and takes the responsibility for providing these data timely, securely and accurately by hiding the complexity of the underlying infrastructure. The infrastructure could consist of different platforms, storage systems, and network capabilities. The orchestration of these services could be based on the Node-RED programming model which DITAS extends by introducing a business logic API.

Following the VDC definition, in order to fully understand the usage of all the components involved in this document, it is important to comprehend the life cycles, the interoperability and the communications between them. First and foremost, it is important to describe the VDC Blueprint which is the vessel in which all the appropriate information about the data source and deployment methods are stored. All of the components involved in this document affect or are affected by this Blueprint. A crucial factor that affects these components and the VDC Blueprint itself is the fact that the VDC is used, described and developed as a product. Being a product, a VDC Blueprint can be created discovered and consumed. Taking under consideration the life-cycle of said Blueprint the architectural components and interconnection are created.

© Main editor and other members of the DITAS consortium

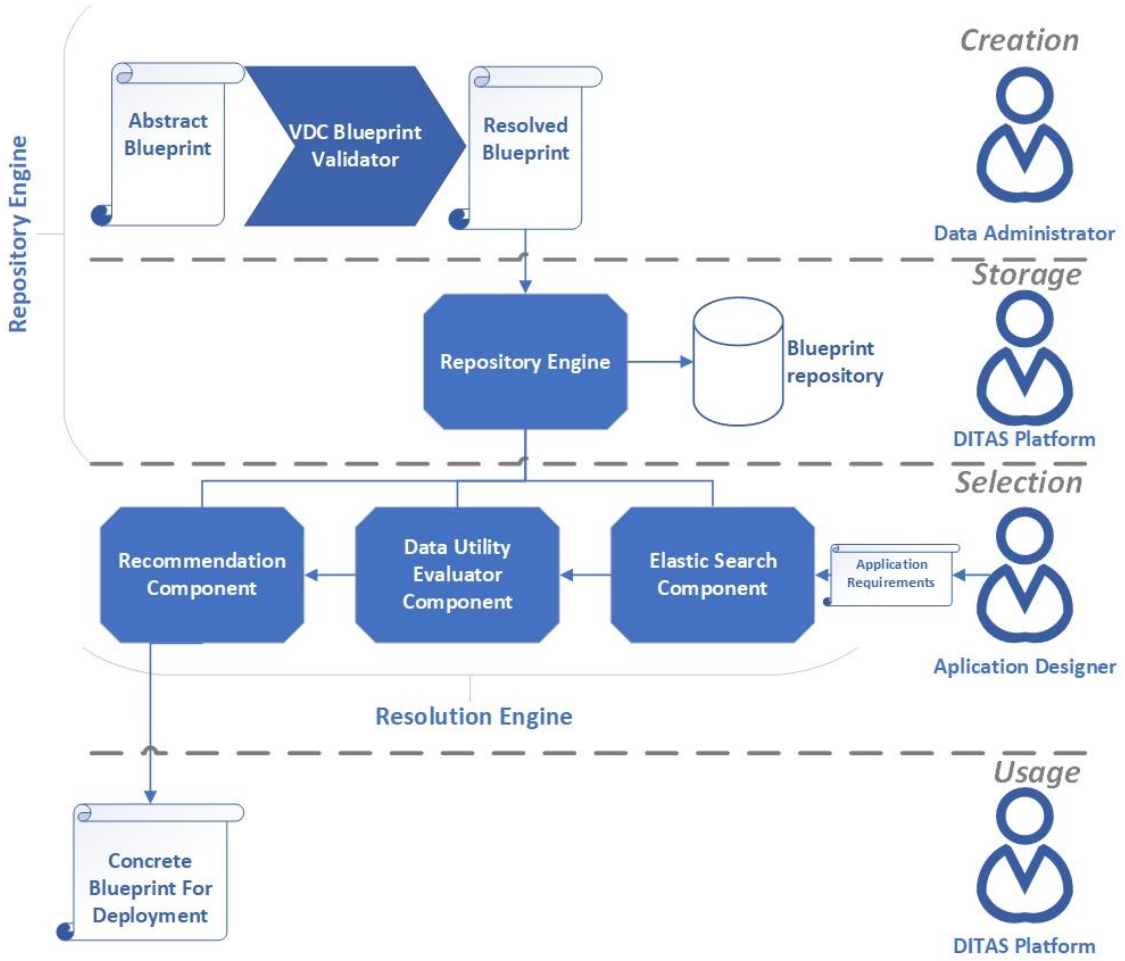


Figure 1: VDC Blueprint Lifecyle

Depicted above is the architectural diagram for the components that are developed in order to fulfill the purpose of WP3. This diagram is separated in 4 phases each one identifying every separate state of the Blueprint life-cycle. Also, it contains two major components; the Repository Engine and the Resolution Engine. Both of these components have functionalities that extend in more than one phases of the life cycle. In order to better describe the architectural concepts, it is better to analyze these phases of the architecture separately, starting from the creation of a blueprint and ending at the final deployment.

- **Creation:** In order to create a functional blueprint an *Abstract Blueprint* is created and it contains all the appropriate information that the Data Administrator should provide (All the sections of the Blueprint described in Chapter 3). This. More specifically, Data Administrators create a JSON file that has to meet the standards and the specifications of the *Blueprint*. This *Abstract Blueprint* is evaluated by the VDC Blueprint Validator which not only checks the structure and the validity of the JSON file but also creates the appropriate fields in order to construct the *Resolved Blueprint*. The *Resolved Blueprint* derives from the *Abstract Blueprint* with the difference that it also contains fields and values that are generated by the system. These auto-generated fields concern mainly the Data Management and the Abstract Properties sections of the Blueprint (more information about these can be found in the VDC Blueprint Schema Chapter).

© Main editor and other members of the DITAS consortium

- **Storage:** The storage layer contains the Blueprint Repository as well as the Repository Engine. The Blueprint Repository is responsible for storing all the blueprints as well as extra information needed for the resolution process. The Repository Engine is the middleware responsible for all the CRUD operations needed to be performed in the repository.
- **Selection:** A VDC is a very complex mechanism to extract data for Data-Intensive Applications (DIA). In order to select the most appropriate Blueprint, the Application Designer should follow a sequence of steps to select a Blueprint that fits the needs of the DIA that is being developed. The Resolution Engine component is involved in the selection phase and consists of three sub-components. Each and every one of these components is created to refine and narrow down the number of blueprints based on the Application requirements and aim to deliver the best candidates to the Application Developer. In particular, the application requirements which are used as input in this phase are expressed using a goal-based modelling language, described in the deliverable D2.2 [D2.2], which hides the complexity of the configuration of parameters and it allows the application developer to define macro-objectives the VDC should achieve. For example, the application developer can specify the need to have a "fast service" or a "reliable service". However, an expert developer will be allowed to tune the macro-objectives by modifying specific parameters to match his/her application requirements. Following the definition of Application Requirements the three sub-components involved in this process are the Elasticsearch, the Data Utility Evaluator and the Recommendation Component. All of these components will be analyzed in another chapter.
- **Usage**: The last step of the VDC life cycle is the actual deployment of the Blueprint Selected. Given all the information about the deployment a blueprint contains, the deployment sequence is initialized.

As far as the structure of the document is concerned, Chapter 2 describes the running example, Chapter 3 describes the Blueprint in terms of the fields and the values that enfolds, Chapter 4 describes the process of creating, validating and also the selection (Resolution) process of the Blueprint and finally Chapter 5 describes the implementation phase of the VDC based on the e-health case study.

1.1 Glossary of Acronyms

Acronym	Definition
ACL	Access Control List
AES	Advanced Encryption Standard
API	Application Programming Interface
CAF	Common Accessibility Framework
CPU	Central Processing Unit
CRUD	Create Read Update Delete
DIA	Data-Intensive Application
GDPR	General Data Protection Regulation
HTTP	Hypertext Transfer Protocol
JDBC	Java Database Connectivity
JSON	JavaScript Object Notation
k-NN	k-Nearest Neighbours
N/A	Not Applicable
OS	Operating System

© Main editor and other members of the DITAS consortium

QoS	Quality of Service
RAM	Random Access Memory
REST	Representational State Transfer
SDK	Software Development Kit
SLA	Service-Level Agreement
SQL	Structured Query Language
SSH	Secure Shell
SSN	Social Security Number
TLS	Transport Layer Security
UI	User Interface
URI	Uniform Resource Identifier
UUID	Universally Unique IDentifier
VDC	Virtual Data Container
VDM	Virtual Data Manager
VM	Virtual Machine
WP	Work Package

Table 1. Acronyms

© Main editor and other members of the DITAS consortium

2 Running Example

In this section, we introduce a running example – inspired by one of the main case studies adopted in the project – that includes most of the main functions that the DITAS platform is able to provide. This running case study helps the understanding of the DITAS approach in terms of data management life-cycle, overall framework and the interaction among the different software components. The case study has been also adopted to drive the implementation of the first DITAS platform release also in terms of components test, integration test, and acceptance test.

2.1 General Description

To improve their internal processes and to provide additional services to its patients and employed doctors, a hospital decided to digitalized the results of all the blood tests that had been taken in its laboratories from the 60s till 90s. That way, being from the 90s all the exams already digitized, the hospital can enlarge its information heritage to make it available to the family doctors that want to quickly access some historical data about their patients.

Based on this decision, the hospital realizes that this data set is a valuable data asset as there could be also external actors (e.g., medical researchers) that are interested in it. To expose these data efficiently, while ensuring the respect of the privacy constraints, the hospital decides to adopt the DITAS approach.

2.2 Dataset Publication

According to the DITAS approach, the hospital holds the role of Data Administrator, i.e., the owner of the data that is in charge of defining and publishing to DITAS the Abstract VDC Blueprint containing all the information about the data set to be exposed, which in our case corresponds to the data stored in two different sources:

- Patient biographical data (SSN, last name, first name, gender, email, age). These data are stored in a classical relational database (e.g., MySQL). All the attributes are mandatory for each tuple.
- Blood tests (SSN, lastname, firstname, gender, date, cholesterol, triglyceride, hepatitis (yes/no)). These data are stored in a noSQL database (e.g., Minio or Cassandra). Not all the attributes have values on each tuple. It depends on the illness that the doctor wants to investigate.

The access to this data set is mediated by the CAF (Common Accessibility Framework) that defines REST-based APIs through which the data sources can be queried:

- Given the SSN returns all the details of the patient (response must be returned encrypted).
- Given the SSN returns the last values for all the exams.
- Given the SSN and a specific blood test (e.g., cholesterol) returns the timeseries of the value of this test for the given patient.
- Given a specific blood test (e.g., cholesterol) and a range of age (e.g., 35-60) it returns the average of this value.

While the last method does not require any authentication, the other ones can be accessed only by medical doctors, because sensitive data of a patient can be viewed only by his/her family doctor.

© Main editor and other members of the DITAS consortium

2.3 Access Policies

To protect the data from any kind of misusing, the following privacy policies must be enforced:

For these data sources the following policies must be enforced:

- If the data sources are accessed for medical purpose (e.g., the family doctor), all the data about patients of which the querying doctor is in charge can be accessed. Information of other patients cannot be accessed.
- If the data sources are accessed for research purpose, the blood tests data source is available except of the SSN, lastname, and firstname, while for the patient data source only the age is available.

2.4 Application Designers

In our running example, we assume that two different Application Designers are involved: one aiming to design an application for medical doctors, while the other to design application for researchers.

In both cases, the Application Designers can also express some QoS requirements. In particular:

- The requirements for the medical doctor app:
  - Availability > 99%
  - Response time < 1 sec (regardless the methods)
- The requirements for the research app:
  - Blood tests should cover ten years of observations:
    - timeliness > 0.6
    - volume > 10'000 tuple
  - Every patient should have in the average 4 blood tests:
    - Process completeness > 90%

Using the DITAS VDC Resolution Engine, the Application Designers are in charge of finding the best VDC satisfying both the functional and non-functional constraints. Although this step appears quite trivial in this example, it is worth noting that we assume that the hospital could be one out of many potential Data Administrators that have published a VDC Blueprint on the VDC Blueprint Repository. Thus, providing retrieval facilities is fundamental.

2.5 Application Developer

Related to the two Application Designers introduced in the previous paragraph, there are two Application Developers:

- The developer of the medical app that implements the client of the VDC CAF for the following three methods:
  - Given the SSN returns all the details of the patient.
  - Given the SSN returns the last values for all the exams.
  - Given the SSN and a specific blood test (e.g., cholesterol) returns the timeseries of the value of this test for the given patient.
- The developer of the research app, that implements the client stub for the fourth methods:
  - Given a specific blood test (e.g., cholesterol) and a range of age (e.g., 35-60) it returns the average of this value.

© Main editor and other members of the DITAS consortium

2.6 Application Execution

Once the applications are running, the following two scenarios are depicted:

- Medical app:
  - The family doctor starts calling some of the methods.
  - Availability goes under 99% for several times.
  - DITAS realizes that it is better to move the VDC, initially deployed on the medical doctor premises, to cloud resources in order to enforce the availability issue.
- Research app
  - The researcher starts calling the selected method against the VDC that is initially deployed on the researcher premises.
  - Timeliness becomes lower than 0.6, volume < 10'000 tuples since the storage on the premises is limited.
  - DITAS realizes that it is better to move the VDC to the Cloud.

2.7 Focus of this Deliverable

This deliverable (D3.2) focuses on the **Data Virtualization** concept. For the specific case of this **running example** it is crucial for the **Data Administrator** to have knowledge of the structure, the **schema** and the specifications that a **VDC Blueprint** follows in order to correctly publish the **dataset** and its **methods** (subchapter 3.2) on the DITAS Platform. This deliverable gives a crystal-clear image of said Blueprint and strictly defines all the necessary information needed for the **VDC Blueprint creation** on chapter 3. Furthermore, details for the **implementation of the VDC** for the running example can be found on chapter 5.

© Main editor and other members of the DITAS consortium

3 VDC Blueprint Schema

In order to form the Blueprint that will reflect the structure and the function of a VDC, it is essential to create a format that will describe the body of this Blueprint, defining all the information and fields that it could possibly include. Based on this format, there are five core sections that a Blueprint contain: the Internal Structure, the Data Management, the Abstract Properties, the Cookbook Appendix and finally the Exposed API section.

- The Internal Structure section records general information related to the VDC composition system, like a business description of the VDC, data sources that could be used, the category and scope of the VDC etc.
- The Data Management is a section dedicated to the VDC that plans on measuring the performance or monitoring the system for optimization and business purposes. Also, this section is responsible for enfolding information about the data quality. Quality of service aspects along with data quality aspects constitutes the definition of data utility for the exposed data [D2.1, D2.2].
- In the Abstract Properties section, all the properties that define the VDC composition system as a product are listed, in order to be commercially available. These indicators are used to conclude Service Level Agreements (SLAs) between the end-user and the Data Administrator.
- The Cookbook Appendix section contains all the necessary information for the orchestration and the deployment of each module/microservice (aka CookBook recipes). Also, it provides software configuration management, agnostic in nature, in order to support multiple tools such as Puppet and Chef.
- The Exposed API section describes the details of the methods that are offered by the VDC and the way they can be accessed.

3.1 Internal Structure (Blueprint Section 1)

DITAS aims to facilitate a company which intends to develop a DIA, focusing only on the application logic and relying on the VDC which, acting as a middleware, takes care of retrieving, processing and delivering data with the proper quality level. To this end, before starting developing the DIA, the product manager who has complete knowledge of the business scopes and goals of the company, investigates whether there are VDCs, the business characteristics of which are aligned with his/her expectations. Consequently, there is a need for the Data Administrator to somehow provide an **abstract description** of his/her VDC Blueprint to enable the manager's work and this is where the Internal Structure section shows its value. Indeed, most of its fields, and in particular those nested inside the **"Overview"** field, are of textual and descriptive type, in order to characterize the VDC as a **product** that provides its functionalities as outsource services.

From the architecture point of view, the Internal Structure section is used from the **Resolution Engine** functional component to search for the most appropriate VDC Blueprint based on its **"Overview.description"** and its **"Overview.tags"** that define the **content** of the data that the VDC provides to the application.

This section is strongly **related** to the **Exposed API section** of the Blueprint, since the **data sources** and also the **VDC methods** are referred in both sections and thus they need to be consistent with each other.

© Main editor and other members of the DITAS consortium

The **"Data_Sources"** field defines the data sources available to the VDC. It defines their connection parameters and their scheme, if relevant. It is used by the **Deployment Engine in WP4** when creating a VDC instance. In addition, it is used by the **Policy Enforcement Engine** in DITAS, defined in **WP2,** for the enforcement of privacy policies, if it is configured in the **Data Management** section of the Blueprint. Its exact structure is defined by the **Data Administrator**.

Finally, this section includes the **Data Flow** of the VDC, so it has **dependencies** that need to be satisfied by the **Cookbook Appendix Section** which, among others, contains all the necessary information for the **orchestration** and the **deployment** of each component that takes part in the flow.

The JSON file below depicts the high level JSON schema of the "Internal Structure" Blueprint section:

```
{
   "type":"object",
   "description":"General information about the VDC Blueprint",
   "properties":{
      "Overview":{
         "type":"object"
      },
      "Data_Sources":{
         "type":"array",
         "minItems":1,
         "uniqueItems":true
      },
      "Flow":{
         "type":"object",
         "description":"The data flow that implements the VDC"
      },
      "Testing_Output_Data":{
         "type":"array",
         "minItems":1,
         "uniqueItems":true
      }
   },
   "additionalProperties":false,
   "required":[
      "Overview",
      "Data_Sources",
      "Flow",
      "Testing_Output_Data"
   ]
}
```

The following tables present thoroughly all the fields that are listed in the above schema:

Overview		
Name	**JSON Format**	**Description**
name	{ "type":"string" }	This field should contain the name of the VDC Blueprint **(mandatory field)**
description	{ "type":"string" }	This field should contain a short description of the VDC Blueprint **(mandatory field)**

© Main editor and other members of the DITAS consortium

tags	{
	"type":"array",
	"items":{
	"type":"object",
	"properties":{
	"method_name":{ },
	"tags":{ }
	},
	"additionalPropeties":false,
	"required":[
	"method_name",
	"tags"
	]
	},
	"minItems":1,
	"uniqueItems":true
	}
		Each element of this array should contain some keywords that describe the function-ality of each one ex-posed VDC method **(mandatory field)** (The two rows below present its nested fields)

tags.method_name	{
	"type":"string"
	}
		This name is the same with the one included in the Exposed API section; EX-POSED_API/Methods **(mandatory field)**

tags.tags	{
	"type":"array",
	"items":{
	"type":"string"
	},
	"minItems":1,
	"uniqueItems":true
	}
		(mandatory field)

Table 2. Internal Structure parameters(Overview)

Data Sources		
Name	**JSON Format**	**Description**

name	{
	"type":"string"
	}
		A name that uniquely identifies the data-source in the VDC, e.g. MinioBloodTests, MySQLPatientDetails **(mandatory field)**

type	{
	"type":"string"
	}
		The type of the datasource, e.g. JDBC, S3 **(optional field)**

parameters	{
	"type":"object"
	}
		Connection parameters, e.g. for MySQL:
		{
		"jdbc.url":
		"jdbc:mysql://localhost:3306/sakila?pro-fileSQL=true",
		"driver" : "com.mysql.jdbc_5.1.5.jar"
		}
		
		Minio example:
		{
		"spark.hadoop.fs.s3a.endpoint" :
		"http://MINIO:9000",
		"spark.hadoop.fs.s3a.access.key" : "ACCESS_KEY",
		"spark.hadoop.fs.s3a.secret.key": "SECRET_KEY",
		"spark.hadoop.fs.s3a.path.style.access" :
		"true",
		"spark.hadoop.fs.s3a.impl" : "org.apache.ha-doop.fs.s3a.S3AFileSystem",
		"s3.filename" : "s3a://ditas.ehealth-sample/di-tas-blood-tests.parquet"
		}
		
		(optional field)

© Main editor and other members of the DITAS consortium

schema	{ "type":"object" }	**(optional field)**

Table 3. Internal Structure parameters (Data Sources)

Flow		
Name	**JSON Format**	**Description**
platform	{ "enum":["Spark", "Node-RED"] }	Spark or Node-RED **(mandatory field)**
parameters	{ "type":"object" }0	Platform details, e.g. for Spark: { "spark.master" : "spark://SPARK:7077", "spark.app.name" : "PatientBloodTestsVDC", "spark.jars" : "mysql-connector-java-5.1.45/mysql-connector-java-5.1.45-bin.jar,hadoop-aws-3.0.0.jar,aws-java-sdk-bundle-1.11.271.jar" } **(optional field)**
source_code	N/A (blank JSON object)	**(optional field)**

Table 4. Internal Structure parameters (Flow)

Testing Output Data		
Name	**JSON Format**	**Description**
method_name	{ "type":"string" }	This name is the same with the one included in the Exposed API section; EXPOSED_API/Methods **(mandatory field)**
zip_data	{ "type":"string" }	The URI to the zip testing output data for each one exposed VDC method **(mandatory field)**

Table 5. Internal Structure parameters (Testing Output Data)

3.2 Data Management (Blueprint Section 2)

The Data Management section of the Blueprint specifies, properties (called metrics) of the methods offered by the VDC and, for each method, the guaranteed levels of data quality, security and privacy. Such information will be used by the DITAS platform: (i) for filtering the blueprints that don't fit the Application Developer requirements; (ii) for the data and computation movement; (iii) as specifications of metric thresholds agreed with the application developer. In particular, such section will be used by the Data Utility Resolution Engine (described later in this document) for filtering blueprints, by the Decision Systems for Data and Computation Movement [D4.2] for the selection of the data and computation movement to be enacted, and by the SLA manager to check whether the VDC violated the defined metrics.

Metrics are defined using the following JSON schema in Table 6. Each metric is identified by an *id*, and two optional parameters a *name* and a *type*, i.e., the conventional name of the metric. For each metric, at least one property is defined. A property defines a measurable value, and it is specified with a *name*, a

© Main editor and other members of the DITAS consortium

measure *unit*. A metric can set a specific *value* or a range between the *maximum* and *minimum* values.

```
{
   "type":"object",
   "properties":{
      "id":{
         "description":"id of the metric",
         "type":"string"
      },
      "name":{
         "description":"name of the metric",
         "type":"string"
      },
      "type":{
         "description":"type of the metric",
         "type":"string"
      },
      "properties":{
         "description":"properties related to the metric",
         "type":"array",
         "items":{
            "type":"object",
            "properties":{
               "name":{
                  "description":"name of the property",
                  "type":"string"
               },
               "unit":{
                  "description":"unit of measure of the property",
                  "type":"string"
               },
               "maximum":{
                  "description":"lower limit of the offered property",
                  "type":"number"
               },
               "minimum":{
                  "description":"upper limit of the offered property",
                  "type":"number"
               },
               "value":{
                  "description":"value of the property",
                  "anyOf":[
                     {
                        "type":"string"
                     },
                     {
                        "type":"object"
                     },
                     {
                        "type":"array"
                     }
                  ]
               }
            }
         }
      }
   }
}
```

Table 6 describe the **metrics** used for **data quality** while Table 7, specified the properties for **privacy** and **security**.

© Main editor and other members of the DITAS consortium

Metric	JSON Example	Description
Availability	{ "id": "1", "name":"Availability 95-99", "type":"Availability", "properties": [{ "name":"Availability", "unit": "percentage", "minimum":95, "maximum":99 }] }	This metric specifies the level of availability of a method.
Response Time	{ "id": "2", "name": "ResponseTime 1", "type": "ResponseTime", "properties": [{ "name": "ResponseTime", "maximum":1, "unit": "second" }] }	This metric specifies the maximum amount of time that the method will take to answer a request.
volume	{ "id": "3", "name": "volume 10000", "type": "volume", "properties": [{ "name": "volume", "value":"10000", "unit": "tuple" }] }	This metric specifies the amount of volume in tuples that the method will return.
Timeliness	{ "id": "4", "name": "Timeliness", "type": "Timeliness", "properties": [{ "name": "Timeliness", "maximum":2, "unit": "day" }] }	This metric specifies how long is the time difference between data capture and the real world event being captured.
Process completeness	{ "id": "5", "name": "Process completeness 90", "type": "Process completeness", "properties": [{ "name": "Process Completeness", "minimum":90, "unit": "percentage" }] }	This metric specifies the completeness of the dataset with respect the application request.

Table 6. Data Management parameters for data quality

Property	JSON Example	Description
Transport Encryption	{ "name": "transport encryption", "properties":[{	Available transport Encryption, if a VDC has this property, it is mandatory to use the minimal available

© Main editor and other members of the DITAS consortium

	"name":"Protocol", "unit":"enum", "value":"TLS" }, { "name":"Version", "unit":"number", "value":"1.2" }, { "name":"Algorithm", "unit":"enum", "value":"AES" }, { "name":"Keylength", "unit":"number", "value":"128" }] }	protocol. The Algorithm and key length can be upgraded upon request (using an intermediary service).
Storage Encryption	{ "name": "encryption", "properties":[{ "name":"Algorithm", "unit":"enum", "value":"AES" }, { "name":"Keylength", "unit":"number", "value":"128" }] }	Available storage Encryption, if a VDC has this property, it is mandatory and cannot be downgraded. The decryption key or authentication have to be send with each request in order to read/write the VDC data. Can only be available in combination with transport encryption. This property is static based on the data sources selected.
Tracing & Access Monitoring	{ "name": "tracing", "properties":[{ "name":"level", "unit":"enum", "value":"datasource" }, { "name":"samplerate", "unit":"percentage", "value":"10" }, { "name":"instrumentation", "unit":"enum", "value":"VDC" }] }	Tracing describes the level of detail the DITAS tracing system observes within for a given request. This monitoring resolution can range from only monitoring every 10'000th request to monitoring every request. And from looking at a VDC as a complete black box to a minute overview of all processing steps, timings, data transferred and even data source processing steps recorded on a request by request basis (depends highly on the data sources and vdc). Only some data sources will allow low level tracing for instance. The tracing system can be triggered either by an application, or the VDC. If triggered by an application the sampling rate is ignored.
Access Control	{ "name": "acl", "properties":[{ "name":"protocol", "unit":"enum",	Available type of access control for a given VDC. It also includes available credential hints, that

© Main editor and other members of the DITAS consortium

	"value":"username/pasword" }, { "name":"credentials", "unit":"list", "value":["b31d032cfdcf47a399990a71e43c5d2a","d63d0e21fdc05f618d55ef306c54af82","..."] }] }	can be used to check if an application has credentials for the given VDC without checking it at runtime.
Purpose Control	{ "name ": "purpose_control", "properties": [{ "name": "available purpose", "unit": "object", "value": { "Jeder": ["NGO", "oeffentl. Institute"], "NGO": ["OpenKnowlageFoundation", "OpenWeatherMap"], "OpenKnowlageFoundation": [], "OpenWeatherMap": [], "oeffentl. Institute": ["TU Berlin"] } }, { "name": "guarantor", "unit": "list", "value": ["d63d0e21fdc05f618d55ef306c54af82", "..."] }] }	Purpose Control indicates if the VDC provides purpose control as required by the GDPR. If that the case the application needs to provide that purpose on every request. High level of tracing and access monitoring is also recommended if purpose control is enabled. Furthermore the application needs to provide a guarantor. A record that allows a VDC to validate a purpose claim. (i.e.: a certificate).
Mutation Control	{ "name": "mutation control", "properties":[{ "name":"announcementAdress", "unit":"string", "value":"https://..." }] }	A system that allows a data source to announce changes of previously accessed data. This enables an application that consumed data from this data source to obey the GDPR even in case of changes being made to the original data (i.e.: accuracy article 5(1d), right to be forgotten).

Table 7. Data Management parameters for privacy and security

3.3 Abstract Properties (Blueprint Section 3)

This section enfolds all the rules in the form of goal trees that are going to be used by the **SLA Manager** in order to construct the **SLA contract**. These rules have values that the **provider** and the **consumer agreed** on and must then be verified at **run-time** and describe the VDC offers as a **product**. The **Data Administrator** is responsible for filling these with the appropriate values and update them if needed. Therefore, these parameters can be used by the product manager (as mentioned in subchapter 3.1) that is interested in consuming the given Blueprint to analyse the cost and compensation actions in case of an **SLA violation**.

© Main editor and other members of the DITAS consortium

Furthermore, during the second iteration of the project, this section of the Blueprint will be extended with parameters that describe the product quality of the VDC. They will help **Application Designer** to select the most appropriate Blueprint based on product based parameters by enforcing the **Recommendation System** designed for the purposes of the **Resolution Engine** described in subchapter 5.2.

The following schema specifies how to represent the goal model specified in deliverable 2.2 [D2.2]. The schema is recursive: each node contains the decomposition type (i.e., "AND" or "OR"), a list of goals and a list of children. Each children is a node itself while goals are IDs that refer to the goals (set of metrics) defined in Section 2 (subchapter 3.2).

```
{
   "treeStructure":{
      "type":"object",
      "properties":{
         "type":{
            "description":"type of node (AND or OR)",
            "type":"string"
         },
         "children":{
            "type":"array",
            "items":{
               "$ref":"#/properties/DATA_MANAGEMENT/properties/methods/items/properties/constraints/properties/dataUtility/properties/goals/items"
            }
         },
         "leaves":{
            "type":"array",
            "items":{
               "type":"string"
            }
         }
      }
   }
}
```

Name	JSON Format	Description
treeGoal	{ "treeStructure":{ "type":"AND", "children":[{ "type":"OR", "leaves":["1", "4"] }, { "type":"AND", "children":[{ "type":"OR", "leaves":["3", "5"] }], "leaves":["2"] }] } }	This is an example of goal tree that specifies the aggregation of metrics that can be used for identifying when the SLA is violated. In particular, it specifies a goal tree as represented in Figure 2, i.e., that a SLA is violated when goals 1 OR goal 4 are violated AND when goal 2 is violated AND Goals 3 OR 4.

© Main editor and other members of the DITAS consortium

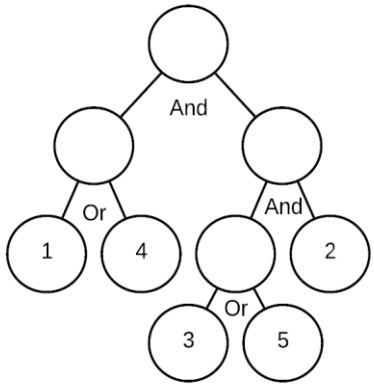
	}	 **Figure 2: Representation of a goal model**

Table 8. Abstract Properties parameters

3.4 Cookbook Appendix (Blueprint Section 4)

This part describes the deployment information for a VDM+VDC(s), this includes both Edge and Cloud parts of the deployment and detailed information for **DI-TAS Deployment Engine** to be able to deploy the DITAS infrastructure in a Fog environment. The structure of the JSON document is as follows:

```
{
   "type":"object",
   "description":"Deployment Information of the VDC Blueprint",
   "properties":{
      "name":{
         "type":"string"
      },
      "description":{
         "type":"string"
      },
      "Infrastructure":{
         "type":"array",
         "description":"an array with the description of the infrastructure",
         "items":{
            "title":"an infrastructure definition",
            "type":"object",
            "properties":{
               "name":{
                  "type":"string"
               },
               "description":{
                  "type":"string"
               },
               "type":{
                  "type":"string"
               },
               "on-line":{
                  "type":"boolean"
               },
               "api_endpoint":{
                  "type":"string"
               },
               "api_type":{
                  "type":"string"
               },
               "keypair_id":{
```

© Main editor and other members of the DITAS consortium

```
                "type":"string"
             },
             "resources":{
                "type":"array",
                "items":{
                   "type":"object",
                   "properties":{
                      "name":{
                         "type":"string"
                      },
                      "type":{
                         "type":"string"
                      },
                      "cpus":{
                         "type":"string"
                      },
                      "ram":{
                         "type":"string"
                      },
                      "disk":{
                         "type":"string"
                      },
                      "generate_ssh_keys":{
                         "type":"boolean"
                      },
                      "ssh_keys_id":{
                         "type":"string"
                      },
                      "role":{
                         "type":"string"
                      },
                      "baseimage":{
                         "type":"string"
                      },
                      "arch":{
                         "type":"string"
                      },
                      "os":{
                         "type":"string"
                      }
                   }
                }
             }
          }
       },
       "provision":{
          "type":"object",
          "properties":{
             "provisioner":{
                "type":"string"
             },
             "provisioner_src":{
                "type":"string"
             },
             "components":{
                "type":"array",
                "items":{
                   "type":"object",
                   "component_name":{
                      "type":"string"
                   }
                },
                "nodes":{
                   "type":"array",
                   "items":{
```

© Main editor and other members of the DITAS consortium

```
                    "type":"string"
                }
            },
            "conf_descriptor":{
                "type":"string"
            },
            "params":{
                "type":"object",
                "additionalProperties":{
                    "type":"string",
                    "description":"node_names"
                }
            }
        }
    }
  }
}
```

The following table details all the information introduced in the previous schema:

Name	JSON Example	Description
name and description	{ "name":"name of the deployment", "description":"description of the deployment" }	They define the name of the deployment and give the opportunity to the administrator to give a description. The name field will be used by the DITAS Deployment Engine the be able to search for this deployment in the future.
infrastructure	{ "infrastructure":[{ "name":"...", "description":"...", "type":"cloud", "on-line":true, "api_endpoint":"api url", "api_type":"cloudsigma", "keypair_id":"keypair_uuid", "resources":[{ "name":"unique_name_1", "type":"vm", "cpus":"8", "ram":"4096", "disk":"512000000", "generate_ssh_keys":false, "ssh_keys_id":"uuid", "role":"type of role, kubernate master, etc...", "baseimage":"optional parameter to indicate OS type" }, { "name":"...", "description":"...", "type":"edge", "on-line":false, "generate_ssh_keys":true, "role":"type of rolo...", "arch":"processor architecture", "os":"OS type" }] }]	This section of the document defines the resources to be created in a Cloud provider or Edge provider. In the case of the Cloud provider it allows to have several of them. In each Cloud provider it is possible to define several VMs and the role of each one of the VMs. For this version of DITAS the role will be related to its Kubernetes role, such as master or worker node, etcd role. A VM can combine etcd role with master or worker role. For the Cloud it is also necessary to indicate the type of Cloud provider so the Deployment Engine should know which API language talk. Also, endpoint and a pointer to the user credentials for that API

© Main editor and other members of the DITAS consortium

	} 	previously provided by other way it is also necessary. For the Edge side it is possible to have two scenarios. For one side the Edge provider can support remote access via an API. If that it is the case, the Deployment Engine will need a information quite similar to the Cloud side. But, if the Edge side of things it is not possible to automate the installation of DITAS via an API, it will be necessary to just prepare the packages of DITAS to be installed there for the hardware architecture and operating system that it is supported by the Edge provider. The end result should be a kubernetes cluster ready to be used by DITAS components that are going to installed using the information of the next section.

provision	{
	"provision": {
	"provisioner":"chef/ansible/puppet...",
	"provisioner_src":"http://recipes_manifests_playbooks.zip",
	"components":[]
	}
	}
		In this section it is necessary to indicate the provisioner to be used by the Deployment Engine. Although the example indicates several types such as Chef, Puppet or Ansible, for the first version of the Deployment Engine only Ansible will be supported.
		The cookbooks/manifests/playbooks will be provided via a git repository. This way this part can be easily updated without having to go again with the renegotiation of the whole blueprint,
		It is necessary to indicate which component it is going to be installed in which node defined in the previous section. It is also possible pass an array of parameters to this node.

Table 9. Cookbook Appendix parameters

© Main editor and other members of the DITAS consortium

3.5 Exposed API (Blueprint Section 5)

In the context of DITAS, the **Application Developer** knows only the Common Accessibility Framework (**CAF**) which hides the complexity behind the VDC. The programming model that **CAF** follows is **REST-oriented** and every VDC exposes a set of **predefined methods**, through which the **Data Administrator** makes available some of the data included in the data sources. Therefore, this section of the Abstract Blueprint contains the **API** that the VDC makes available to the **Application Developer**. It is a **technical** section that is used to enable the Developer to fully understand how the VDC methods work and then to conclude whether the VDC fits to the his/her DIA from a **developing point of view**.

This section is strongly **related** to the section regarding the **Internal Structure**, since the **VDC methods** and also the **data sources** are referred in both places and thus they need to be consistent with each other.

The JSON file below depicts the JSON schema of the "Exposed API" Blueprint section:

```
{
   "title":"CAF API",
   "type":"object",
   "properties":{
      "Methods":{
         "type":"array",
         "minItems":1,
         "uniqueItems":true
      },
      "RESTful_API":{
         "description":"The complete API of the VDC, written according to the OpenAPI
Specification or any other Specification"
      },
      "general_parameters":{
         "description":"Any other parameters that the Data Administrator would like to
add to the API"
      }
   },
   "additionalProperties":false,
   "required":[
      "Methods",
      "RESTful_API"
   ]
}
```

The "RESTful_API" field includes the complete API of the VDC, written according to the OpenAPI Specification or any other Specification. The APIs of the VDC Blueprint Examples, presented in the Appendix, are structured based on the OpenAPI Specification (formerly known as the Swagger Specification); Version 3.0.1. The "general_parameters" is an optional field, where the Data Administrator is free to add any other meaningful information about the exposed API. The "Methods" field is an array listing all the methods that are (can be) exposed by the VDC, while its nested fields (applied to every method) are analyzed in the following table:

Name	JSON Format	Description
name	{ "type":"string" }	The name of each method must be unique, meaning that different methods MUST have different names **(mandatory field)**

© Main editor and other members of the DITAS consortium

description	{ "type":"string" }	Free text that briefly describes the functionality of the method **(optional field)**
requestBody_ schema	{ "type":"object" }	The JSON schema of the body of an HTTP request to the method **(optional field)**
out-put_schema	{ "type":"object" }	The output JSON schema of the method **(mandatory field)**
data_sources	{ "type":"array", "minItems":1, "uniqueItems":true }	An array that contains all the data sources accessed by the method **(mandatory field)**
parameters	N/A (blank JSON object)	Any other parameters that the data administrator would like to add to the method **(optional field)**

Table 10. Exposed API parameters(Methods)

© Main editor and other members of the DITAS consortium

4 VDC Blueprint Repository & Resolution Engine

4.1 VDC Blueprint Repository Engine

The requirements and the structure of the data indicate the need for a document-oriented database. The VDC Blueprint Repository is based on MongoDB, a popular JSON-document database that supports high availability and horizontal scaling. The Repository communicates with other DITAS components or DITAS roles via an HTTP REST interface (API).

4.1.1 Blueprint Creation

Data Administrators interact with the VDC Blueprint Repository Engine, in order to define and store VDC Blueprints. Using the interface, they submit an Abstract VDC Blueprint that, after being evaluated by the VDC Blueprint Validator (see 4.1.1.1), is extended with the so-called auto-generated fields in order to be transformed to the Resolved VDC Blueprint and finally to be inserted into the VDC Blueprint Repository. The whole process is depicted in the figure below. The API also provides methods for updating and deleting existing blueprints.

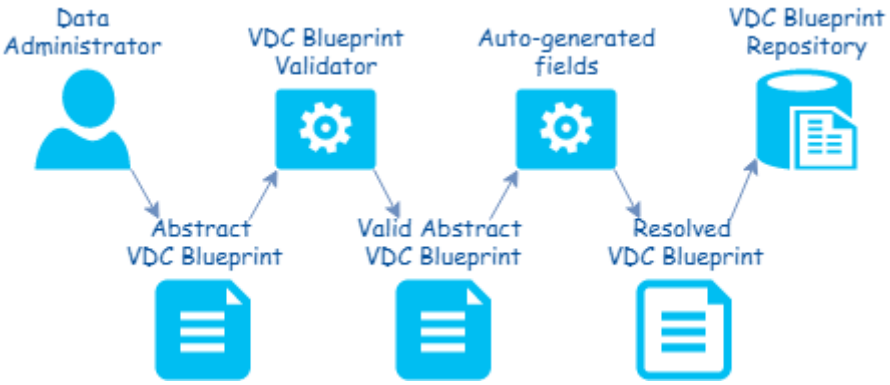


Figure 3: VDC Blueprint creation sequence

4.1.1.1 VDC Blueprint Validator

This subcomponent takes as input an abstract VDC Blueprint and verifies that it is valid against all limitations and requirements that are defined. A v4 JSON Schema is used to describe the format of an abstract VDC Blueprint along with other grammar standards and specifications. The Validator also checks about logic requirements, for example each "method_name" mentioned in the Internal Structure section must be also defined in the Exposed API section.

The goal of the validator is to enforce that the inserted or updated documents are valid before they end up in Blueprint Repository and in case of invalid write requests to provide descriptive and helpful error messages.

4.1.2 Blueprint Retrieval

VDC Resolution Engine component interacts with the VDC Blueprint Repository Engine, in order to retrieve the best VDC Blueprint candidates. The API enables the retrieval of many blueprints through read-methods based on queries as well as the retrieval of one specific blueprint identified by its unique id.

© Main editor and other members of the DITAS consortium

4.1.3 Compenet API

`GET /ditas/blueprints`

Purpose	Show all blueprints
Input	Query Parameters *Optional* *Name: section* Description: Section's name (case insensitive) or number to be projected. If not specified, all sections will be shown. Multiple section values can be provided as in the example below. Type: integer or string Examples: section=1, section=internal_structure, section=1&section=2, section=internal_structure&section=data_management *Optional* *Name: filter* Description: Allows to specify conditions on the documents to be returned. If multiple filter query parameters are specified, they are logically composed with the AND operator. Type: JSON document Examples: filter={'some_key':'some_value'} , filter={'INTERNAL_STRUCTURE.Overview.name':'My VDC'}
Output	*ResultSet* Array of blueprints Code: 200 OK

Table 11. Get Blueprint method documentation

`GET /ditas/blueprints/{id}`

Purpose	Get a blueprint by its id
Input	Path Parameters

© Main editor and other members of the DITAS consortium

	Required Name: id Description: Blueprint's unique id. Type: string Example: /ditas/blueprints/5aafc13872c68d14b9dabb36
	Query Parameters *Optional* *Name: section* Description: Section's name (case insensitive) or number to be projected. If not specified, all sections will be shown. Multiple section values can be provided as in the example below. Type: integer or string Examples: section=1, section=internal_structure, section=1&section=2, section=internal_structure&section=data_management
Output	*ResultSet* The selected blueprint Code: 200 OK
Error Responses	No matching document found Code: 404 Not Found

Table 12. Get Blueprint by id method documentation

`POST /ditas/blueprints`

Purpose	Create a new blueprint
Input	Request Body Description: Abstract VDC Blueprint to be stored in the VDC Blueprint Repository. In order to perform bulk write request pass an array

© Main editor and other members of the DITAS consortium

	of blueprints. Type: JSON document or array of documents

Output	*ResultSet*
	The id(s) of the created blueprint(s)
	
	Example:
	{
	"blueprint_id": ["5ad5f29372c68d14b9dabb3c"]
	}
	
	Code: 201 Created

Error Re-sponses	Unauthorized Code: 401 Unauthorized

	Schema Violation
	Code: 400 Bad Request
	Example:
	{
	"_warnings": [
	"#/INTERNAL_STRUCTURE: 2 schema violations found",
	"#/INTERNAL_STRUCTURE/Overview: required key [name] not found",
	"#/INTERNAL_STRUCTURE/Flow: required key [platform] not found"
	],
	"http status code": 400,
	"http status description": "Bad Request",
	"message": "request check failed"
	}

Table 13. Post Blueprint method documentation

`PATCH /ditas/blueprints/{id}`

Purpose	Update an existing blueprint
Input	Path Parameters *Required*

© Main editor and other members of the DITAS consortium

	Name: id Description: Blueprint's unique id. Type: string Example: /ditas/blueprints/5aafc13872c68d14b9dabb36

	Request Body
	
	Description:
	All the fields to be updated.
	
	Type: JSON document
	
	Example:
	{
	"INTERNAL_STRUCTURE.Overview":{
	"name":"VDC_1",
	"description":"Information about the patients in Milan",
	"tags":[{
	"method_name":"get_patient_details",
	"tags":["Milan","patients"]
	}]
	},
	"INTERNAL_STRUCTURE.Flow.platform":"Node-RED"
	}

Output	Empty Body Code: 200 OK
Error Responses	No matching document found Code: 404 Not Found
	Unauthorized Code: 401 Unauthorized
	Schema Violation Code: 400 Bad Request Example: {

© Main editor and other members of the DITAS consortium

	"_warnings": ["#: extraneous key [new_field] is not permitted"], "http status code": 400, "http status description": "Bad Request", "message": "request check failed" }

Table 14. Update Blueprint method documentation

`DELETE /ditas/blueprints/{id}`

Purpose	Delete a blueprint
Input	Path Parameters *Required* Name: id Description: Blueprint's unique id. Type: string Example: /ditas/blueprints/5aafc13872c68d14b9dabb36
Output	Empty Body Code: 200 OK
Error Responses	No matching document found Code: 404 Not Found
	Unauthorized Code: 401 Unauthorized

Table 15. Delete Blueprint method documentation

4.2 VDC Blueprint Resolution Engine

In this subchapter we describe all the steps and API's responsible for selecting the appropriate blueprint for deployment. All of the resolution processes are divided in several components, each and every one created to filter out specific user requirements.

As depicted in the image below (Figure 4) the resolution process takes as input the user requirements described by the Application Developer and provides as output the respected blueprint that fulfill these requirements.

© Main editor and other members of the DITAS consortium

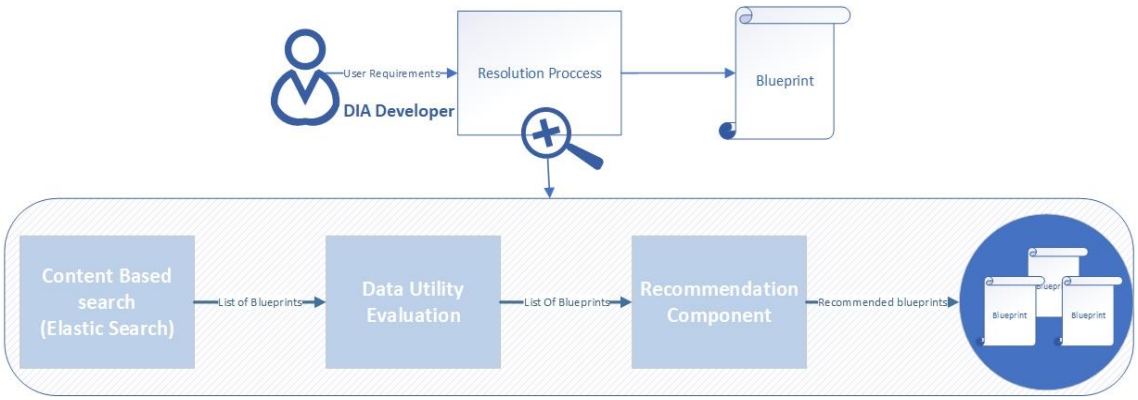


Figure 4: VDC Blueprint filtering processes

The resolution process is divided to three sub-processes, these are:

- **Content Based Search**: This component is responsible for finding out the most appropriate blueprints based on the type of data the VDC delivers. This process takes as input some free text that the user provides and delivers to the next component a list of blueprints that matches the content, based on the input of the user.
- **Data Utility Evaluator**: Filtering out based on the type of data is a crucial step, but it is not enough in order to fulfill the needs of an application. The quality of the data is an equally important factor that is taken care of by this component.
- **Recommendation:** Given that the requirements of the application are matched with the appropriate blueprints, it is important to be able to recommend and rank the best candidate blueprints in order to give the Application Developer a more sophisticated and personalized solution.

4.2.1 Content based resolution

4.2.1.1 Description

The first step towards Blueprint Selection is to find, using the UI of the Resolution Engine, a list of blueprints based on the type of data they provide (Figure 5).

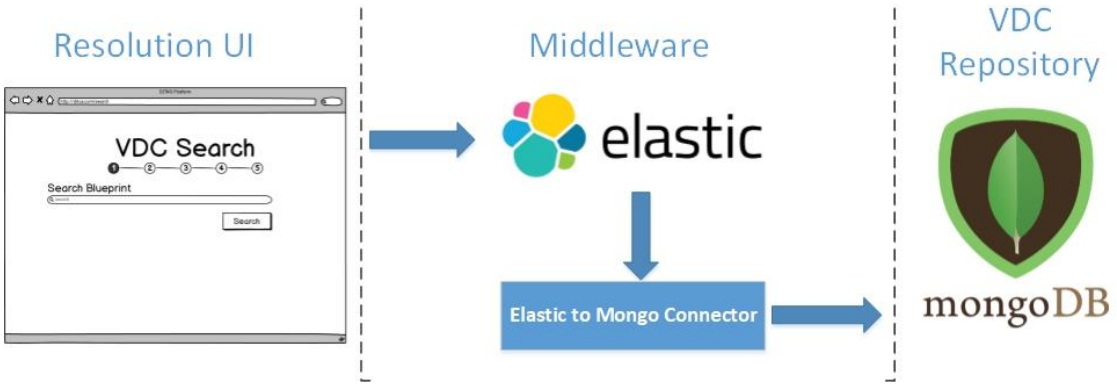


Figure 5: Content based resolution architecture

The frontend of the resolution engine is going to be a responsive UI developed with Angular. A middleware will also be embedded in order to create intelligent queries to optimize the search in the Blueprint Repository, which is based on MongoDB.

© Main editor and other members of the DITAS consortium

On the first step, users type in a search bar what kind of VDC they would like to use. More specifically, the search query should contain information about the type of data users need i.e. a search query can be: "weather forecast for europe". To process this query and return the best matching results the middleware is needed.

The reason a middleware is necessary is because MongoDB is a great general purpose database but one place where its limitations show is with its full text searching. MongoDB got a finalised full text search mechanism in MongoDB 2.6 but you can only have one field per collection indexed for that type of search. If one wants richer free text search capabilities than what MongoDB offers then an alternative is required. Elasticsearch gives much more extensive and powerful search capabilities, being built upon the Lucene text search library.

On top of that a connector between MongoDB and Elasticsearch is also needed. There are several tools created to fulfill this purpose, but we take into consideration the most popular and well established ones. More specifically, the three tools that are considered to work as a connector are:

- Mongoosastic, a Mongoose plugin which lets you select fields to be indexed in Elasticsearch.
- Logstash, an open source, server-side data processing pipeline that ingests data from a multitude of sources simultaneously, transforms it, and then sends it to your favorite "stash."
- Transporter, which allows the user to configure a number of data adaptors as sources or sinks. These can be databases, files or other resources.

4.2.1.2 Component API

`/api/elasticSearch`

Purpose	This method searches into specific fields on the blueprint, the tags and also the description of the blueprint.
Input	*Free text: "blood test from italy"*
	Candidates Array of Blueprint
Output	*ResultSet* Array of BlueprintUUID and scores of relevance: { "_index": "vdc_search", **"_id": "VyOReGAB1xWEy8e1njck",** **"_score": 2.463948,** }, { "_index": "vdc_search", **"_id": "OiOReGAB1xWEy8e1LDdq",** **"_score": 1.7260926,** }, {

© Main editor and other members of the DITAS consortium

	<pre>"_index": "vdc_search",
	"_id": "ViOReGAB1xWEy8e1lDfm",
	"_score": 1.1507283,
	},
	}</pre>

Table 16. Search Blueprint method documentation

4.2.2 Data utility Resolution Engine

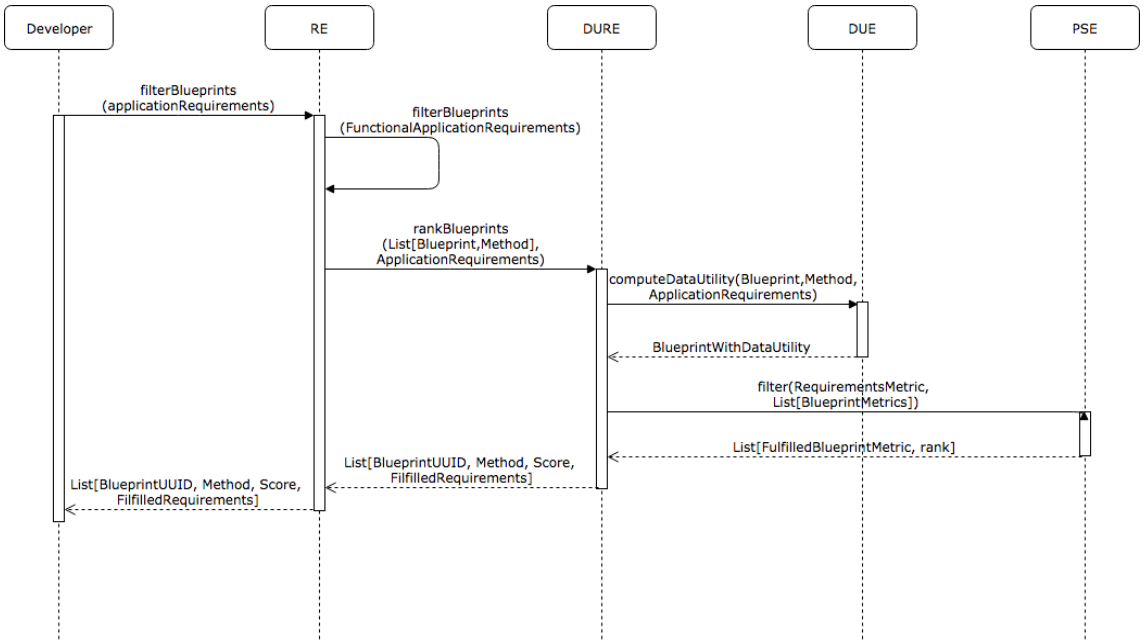


Figure 6: Data Utility resolution sequence diagram

4.2.2.1 Description

To ensure that only those blueprints that meet the non-functional requirements are retrieved, we verify that the goal models defined in the application requirements are -- at least partially -- fulfilled by the non-functional properties defined in the blueprints. In addition, we rank blueprints based on how well they fit the application requirements.

More in detail, given the application requirements and a blueprint, we perform the following steps:

1. Based on the application requirements, we analyze each selected method in order to understand if the user is interested in the whole output or just in a subset of attributes provided by the method. In the former situation, the module considers the Potential Data Utility values stored in the blueprint. In the latter case, we compute, instead, the actual data utility of the portion of data provided by the method to which the user is interested. This assessment is performed by invoking the Data Utility Evaluator module.
2. We explore the goal tree (see deliverable D2.2, subsection 4.2.1 [D2.2]) defined in the application requirements to filter out unsuited blueprints, and to assign a rank to each candidate blueprint. In particular, we trans-

© Main editor and other members of the DITAS consortium

form the goal tree into a boolean expression, whose variables are represented by the leaf goals. Then, for each leaf goal, we verify whether the goal is fulfilled or not by a blueprint:

   a. For goals that predicate on data utility metrics, we directly verify if the blueprint fulfills the goal. To do so, we consider three types of constraints: exact match, minimum, and maximum. In the first case (exact match), we verify if a metric indicating the same value as the one in the goal exists in the blueprint. In the other cases (minimum and maximum), we verify if a metric indicating a value or an interval that is within the threshold defined by the constraint exists in the blueprint.
   b. For goals that predicate on privacy and security metrics, we verify if the blueprint fulfills the goal by invoking the Privacy and Security Evaluator module [D2.2]. This module, given a constraint on a privacy or security metric, determines to which extent the blueprint supports such a constraint.

3. Finally, we evaluate the boolean expression. If the expression is false, we discard the blueprint. Otherwise, we assign a rank to the blueprint according to the following rules:
   a. For each leaf goal, we assign a rank corresponding to the goal's weight if the goal is fulfilled. Otherwise, we assign a rank of 0.
   b. For each OR goal, we assign a rank corresponding to the sum of the ranks of its child goals.
   c. For each AND goal, we assign a rank corresponding to the sum of the ranks of its child goals only if all child goals are fulfilled (i.e., their rank is greater than 0). Otherwise, we assign a rank of 0.
   d. Once a rank is assigned to every goal, we compute the rank of the blueprint as the ratio between the rank assigned to the root goal and the sum of all the weight defined for the child nodes.
4. If the blueprint is not discarded, we prune the goal tree, discarding the portions that are not fulfilled by the blueprint. To do so, we remove all goals whose rank is 0.

4.2.2.2 Component API

`/api/rankBlueprints`

Purpose	This method allows to filter and rank blueprints according to non-functional requirments
Input	*ApplicationRequirements* JSON document specifying the application requirements
	Candidates Array of pairs Blueprint, MethodName *Blueprint* JSON document describing the blueprint (see Chapter 4 of this document)

© Main editor and other members of the DITAS consortium

	MethodName String indicating the name of the method in the blueprint that fulfills the functional requirements)
Output	*ResultSet* Array of tuples BlueprintUUID, MethodName, Score, FulfilledRequirements *BlueprintUUID* String indicating the UUID of the blueprint *MethodName* String indicating the name of the method in the blueprint that fulfills the functional requirements) *Score* Double indicating the rank (in the 0-1 range) assigned to the blueprint *FulfilledRequirements* JSON document specifying the application requirements with the pruned goal tree

Table 17. Rank Blueprint method documentation

4.2.3 Product Based Recommendation

4.2.3.1 Description

In order to give the application developer an in depth and personalized solution, a recommendation component will be implemented. More specifically, after the filtering of the blueprints in the previous steps, a new component is introduced which takes as input the blueprints that derived from said filtering and ranks them in order and that way the user can choose which is the more suitable (figure 7).

© Main editor and other members of the DITAS consortium

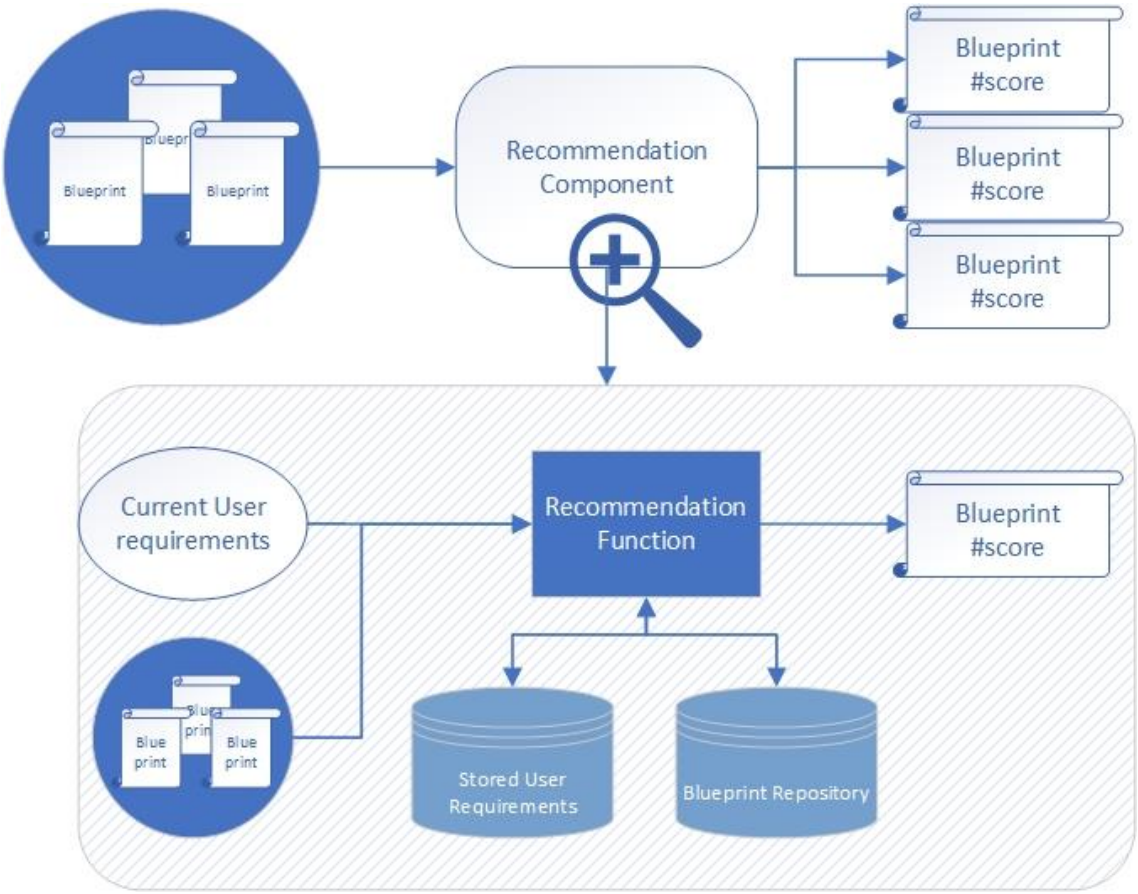


Figure 7: Product based recommendations architecture

This component has one major mission, to leave the final decision to the user and make the recommendation procedure transparent. More precisely, the recommendation system contains a function that gather parameters from different aspects of a blueprint (i.e. popularity, price, reputation), all of these values are taken under consideration and are added to a formula that produces a ranking between the most suitable candidate VDC Blueprints.

At first, the formula of the recommendation operation produces the basic score which is essential in order to create an unbiased result and then any extra product based parameters are added to the function. To form the basic score a combination of Elasticsearch and k-nearest neighbors (k-NN) algorithm is being used. Specifically, given the user requirements and the blueprint that is evaluated, the Recommendation Component finds different users that have consumed and rated this blueprint and happen to have similar requirements as current user. Finding similar users (in case of the k-NN the nearest neighbors) is a difficult task, given that the user requirements cannot be vectorized in order for the K-NN algorithm to work properly. This is where the Elasticsearch comes to place, by correlating the documents (in our case user requirements) and producing a score of relevance. The formula takes this score and combines it with the actual rating of the user to produce a final value which will be the basic score of the blueprint. After the production of this score any other parameters will be also considered. These parameters will be dynamically added or subtracted from the formula given the user preferences. i.e. a value to be considered in the formula is the price of the VDC but a user might not be interested in it and just wants to get the best one regardless of the price. In this case, the system takes this preference under consideration and eliminates the price parameter completely from the formula.

© Main editor and other members of the DITAS consortium

4.2.3.2 Component API

This components API is planned for the second iteration of DITAS project.

© Main editor and other members of the DITAS consortium

5 VDC Implementation

In case a VDC exposes massive amounts of data from multiple different data sources for data intensive applications, where performance is very important, it is advisable to implement the VDC using Apache Spark. The case of the e-Health sample use case, where patient data is stored in MySQL and in Minio, and is consumed using with JDBC API and S3 API respectively, is a good example of such a VDC.

The Spark DataSource API takes care of handling the different data access formats and protocols and the code inside the VDC operates on RDDs (Resilient Distributed Datasets). At the bottom of the diagram the different data sources are accessed using the DataSource API. At the top of the diagram the VDC exposes its functionality to the application using CAF, which is a REST API implemented as a simple Play Framework web application.

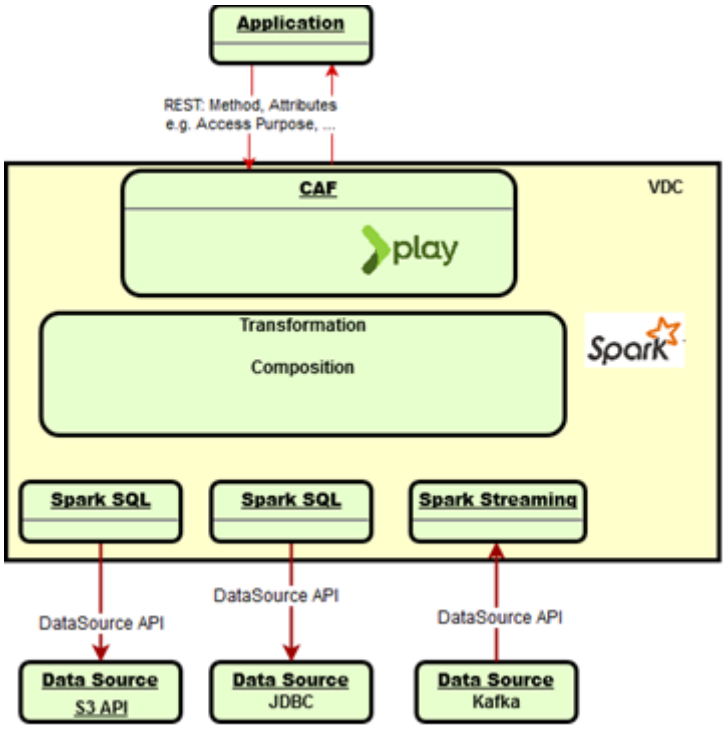


Figure 8: Virtual access to data in the federated store

This kind of VDC uses Spark both for accessing various types of data sources and for data-intensive processing of the data, where the application is performance-critical and the VDC operates on huge amounts of data. The VDC can be either configured with a Spark cluster per VDM, where the Spark cluster will be managed by Kubernetes. According to the Kubernetes blog: "Starting with Spark 2.3, users can run Spark workloads in an existing Kubernetes 1.7+ cluster and take advantage of Apache Spark's ability to manage distributed data processing tasks. Apache Spark workloads can make direct use of Kubernetes clusters for multi-tenancy and sharing through Namespaces and Quotas, as well as administrative features such as Pluggable Authorization and Logging." An alternative configuration would be with an external Spark cluster, and then the DITAS platform would not be responsible for managing the Spark cluster and the Data Administrator should make sure the Spark cluster matches the expected load of the VDC used by various applications.

© Main editor and other members of the DITAS consortium

For such a VDC to be instantiated, the Blueprint defined in chapter 3 should define the data sources with their names and all their connection parameters, the methods exposed by CAF, and the Spark configuration.

© Main editor and other members of the DITAS consortium

6 Conclusions

Creating a system to enable virtualization of data sources is a complicated task that requires a complete analysis and specification of methods, data sources and API's but at the same time it should be abstract enough in order to be able to describe heterogeneous resources. Given the landscape, on which this project is located, specific architectural and operational decisions were made in order to accomplish these requirements. One of the most important steps towards the data virtualization is the creation of the Blueprint which describes, in a detailed way and without being case specific, all the appropriate indicators that needed to be described for the deployment, storage and retrieval of VDC's. To achieve these goals the Blueprint is separated into five dedicated sections. By dividing the complete set of parameters needed, the task of creating a functional system can be done safely and methodically, pushing towards its goals and at the same time keep a certain level of abstraction. All these sections have thorough guidelines and schemas in order to unify the description of data sources, but in the meantime they try to be as abstract as possible in order to be able to describe the spectrum of data sources to a feasible extent.

Given the nature of the Blueprint it is crucial for the Application Developer to be able to select and use the best candidate blueprint which will instantiate the most appropriate VDC. The level of abstraction of the schema and the Blueprint description may help to generalize the VDC's but makes the process of selecting one rather difficult due to ambiguities. This is where the Resolution Engine comes to place, a component which takes as input the application requirements and filters the blueprint based on this information in order to find the most suitable one. That way the correct selection of a VDC is performed without adding extra effort to the user or without requiring from him/her knowledge of the VDC contents.

Lastly, the orchestration and deployment of a VDC is essential not only for the completion of the data virtualization task, but also for the refinement and continuous upgrade of the processes and the deployment mechanism through constant testing and experimentation.

© Main editor and other members of the DITAS consortium

References

[D2.2] Deliverable D2.1 of DITAS project: "DITAS Data Management – first release". © DITAS Consortium. September 2017.

[D2.2] Deliverable D2.2 of DITAS project: "DITAS Data Management – second release". © DITAS Consortium. April 2018.

[D3.1] Deliverable D3.1 of DITAS Project: "Data Virtualization Architecture Design. © DITAS Consortium. September 2017.

[D4.2] Deliverable D4.2 of DITAS Project: "Execution environment prototype (first release)". © DITAS Consortium. May 2018.

© Main editor and other members of the DITAS consortium

Annex A. VDC Blueprint Complete Schema

```
{
   "type":"object",
   "description":"This is a VDC Blueprint which consists of five sections",
   "properties":{
      "INTERNAL_STRUCTURE":{
         "type":"object",
         "description":"General information about the VDC Blueprint",
         "properties":{
            "Overview":{
               "type":"object",
               "properties":{
                  "name":{
                     "type":"string",
                     "description":"This field should contain the name of the VDC Blueprint"
                  },
                  "description":{
                     "type":"string",
                     "description":"This field should contain a short description of the VDC Blueprint"
                  },
                  "tags":{
                     "type":"array",
                     "description":"Each element of this array should contain some keywords that describe the
functionality of each one exposed VDC method",
                     "items":{
                        "type":"object",
                        "properties":{
                           "method_name":{
                              "type":"string"
                           },
                           "tags":{
                              "type":"array",
                              "items":{
                                 "type":"string"
                              },
                              "minItems":1,
                              "uniqueItems":true
                           }
                        },
                        "additionalProperties":false,
                        "required":[
                           "method_name",
                           "tags"
                        ]
                     },
                     "minItems":1,
                     "uniqueItems":true
                  }
               },
               "additionalProperties":false,
               "required":[
                  "name",
                  "description",
                  "tags"
               ]
            },
            "Data_Sources":{
               "type":"array",
               "items":{
                  "type":"object",
                  "properties":{
                     "name":{
                        "type":"string",
                        "description":"A name that uniquely identifies the data source in the VDC"
                     },
                     "type":{
                        "type":"string"
                     },
                     "parameters":{
                        "type":"object",
                        "description":"Connection parameters"
                     },
                     "schema":{
                        "type":"object"
                     }
                  },
                  "required":[
                     "name"
                  ]
               },
               "minItems":1,
               "uniqueItems":true
            },
            "Flow":{
               "type":"object",
               "description":"The data flow that implements the VDC",
               "properties":{
```

© Main editor and other members of the DITAS consortium

```
                "platform":{
                   "enum":[
                      "Spark",
                      "Node-RED"
                   ]
                },
                "parameters":{
                   "type":"object"
                },
                "source_code":{

                }
             },
             "additionalProperties":false,
             "required":[
                "platform"
             ]
          },
          "Testing_Output_Data":{
             "type":"array",
             "items":{
                "type":"object",
                "properties":{
                   "method_name":{
                      "type":"string",
                      "description":"name of the method, as indicated in EXPOSED_API.Methods section"
                   },
                   "zip_data":{
                      "type":"string",
                      "description":"The URI to the zip testing output data for each one exposed VDC method",
                      "format":"uri"
                   }
                },
                "additionalProperties":false,
                "required":[
                   "method_name",
                   "zip_data"
                ]
             },
             "minItems":1,
             "uniqueItems":true
          }
       },
       "additionalProperties":false,
       "required":[
          "Overview",
          "Data_Sources",
          "Flow",
          "Testing_Output_Data"
       ]
    },
    "DATA_MANAGEMENT":{
       "type":"object",
       "properties":{
          "functionalProperties":{
             "description":"functional properties specified by the Application Developer"
          },
          "methods":{
             "description":"list of methods",
             "type":"array",
             "items":{
                "type":"object",
                "properties":{
                   "name":{
                      "description":"name of the method, as indicated in EXPOSED_API.Methods section",
                      "type":"string"
                   },
                   "constraints":{
                      "type":"object",
                      "description":"goal trees",
                      "properties":{
                         "dataUtility":{
                            "type":"object",
                            "description":"data utility goal tree",
                            "properties":{
                               "goals":{
                                  "type":"array",
                                  "items":{
                                     "type":"object",
                                     "description":"goal definition",
                                     "properties":{
                                        "id":{
                                           "type":"string",
                                           "description":"id of the goal"
                                        },
                                        "name":{
                                           "type":"string",
                                           "description":"name of the goal"
                                        },
```


© Main editor and other members of the DITAS consortium

```
                                    "weight":{
                                       "type":"string",
                                       "description":"weight assigned to the goal"
                                    },
                                    "metrics":{
                                       "type":"array",
                                       "items":{
                                          "type":"object",
                                          "description":"definition of the metric",
                                          "properties":{
                                             "id":{
                                                "description":"id of the metric",
                                                "type":"string"
                                             },
                                             "name":{
                                                "description":"name of the metric",
                                                "type":"string"
                                             },
                                             "type":{
                                                "description":"type of the metric",
                                                "type":"string"
                                             },
                                             "properties":{
                                                "description":"properties related to the metric",
                                                "type":"array",
                                                "items":{
                                                   "type":"object",
                                                   "properties":{
                                                      "name":{
                                                         "description":"name of the property",
                                                         "type":"string"
                                                      },
                                                      "unit":{
                                                         "description":"unit of measure of the property",
                                                         "type":"string"
                                                      },
                                                      "maximum":{
                                                         "description":"lower limit of the offered prop-
erty",
                                                         "type":"number"
                                                      },
                                                      "minimum":{
                                                         "description":"upper limit of the offered prop-
erty",
                                                         "type":"number"
                                                      },
                                                      "value":{
                                                         "description":"value of the property",
                                                         "anyOf":[
                                                            {
                                                               "type":"string"
                                                            },
                                                            {
                                                               "type":"object"
                                                            },
                                                            {
                                                               "type":"array"
                                                            }
                                                         ]
                                                      }
                                                   }
                                                }
                                             }
                                          }
                                       }
                                    }
                                 }
                              },
                              "security":{
                                 "type":"object",
                                 "description":"security goal tree",
                                 "properties":{
                                    "goals":{
                                       "type":"array",
                                       "items":{
                                          "type":"object",
                                          "description":"goal definition",
                                          "properties":{
                                             "id":{
                                                "type":"string",
                                                "description":"id of the goal"
                                             },
                                             "name":{
                                                "type":"string",
                                                "description":"name of the goal"
                                             },
```


© Main editor and other members of the DITAS consortium

```
                                    "metrics":{
                                       "type":"array",
                                       "items":{
                                          "type":"object",
                                          "description":"definition of the property",
                                          "properties":{
                                             "id":{
                                                "description":"id of the metric",
                                                "type":"string"
                                             },
                                             "name":{
                                                "description":"name of the metric",
                                                "type":"string"
                                             },
                                             "type":{
                                                "description":"type of the metric",
                                                "type":"string"
                                             },
                                             "properties":{
                                                "description":"properties related to the metric",
                                                "type":"array",
                                                "items":{
                                                   "type":"object",
                                                   "properties":{
                                                      "name":{
                                                         "description":"name of the property",
                                                         "type":"string"
                                                      },
                                                      "unit":{
                                                         "description":"unit of measure of the property",
                                                         "type":"string"
                                                      },
                                                      "maximum":{
                                                         "description":"lower limit of the offered prop-
erty",
                                                         "type":"number"
                                                      },
                                                      "minimum":{
                                                         "description":"upper limit of the offered prop-
erty",
                                                         "type":"number"
                                                      },
                                                      "value":{
                                                         "description":"value of the property",
                                                         "anyOf":[
                                                            {
                                                               "type":"string"
                                                            },
                                                            {
                                                               "type":"object"
                                                            },
                                                            {
                                                               "type":"array"
                                                            }
                                                         ]
                                                      }
                                                   }
                                                }
                                             }
                                          }
                                       }
                                    },
                                    "privacy":{
                                       "type":"object",
                                       "description":"privacy goal tree",
                                       "properties":{
                                          "goals":{
                                             "type":"array",
                                             "items":{
                                                "type":"object",
                                                "description":"goal definition",
                                                "properties":{
                                                   "id":{
                                                      "type":"string",
                                                      "description":"id of the goal"
                                                   },
                                                   "name":{
                                                      "type":"string",
                                                      "description":"name of the goal"
                                                   },
                                                   "metrics":{
                                                      "type":"array",
                                                      "items":{
                                                         "type":"object",
```


© Main editor and other members of the DITAS consortium

```
                                        "description":"definition of the property",
                                        "properties":{
                                           "id":{
                                              "description":"id of the metric",
                                              "type":"string"
                                           },
                                           "name":{
                                              "description":"name of the metric",
                                              "type":"string"
                                           },
                                           "type":{
                                              "description":"type of the metric",
                                              "type":"string"
                                           },
                                           "properties":{
                                              "description":"properties related to the metric",
                                              "type":"array",
                                              "items":{
                                                 "type":"object",
                                                 "properties":{
                                                    "name":{
                                                       "description":"name of the property",
                                                       "type":"string"
                                                    },
                                                    "unit":{
                                                       "description":"unit of measure of the property",
                                                       "type":"string"
                                                    },
                                                    "maximum":{
                                                       "description":"lower limit of the offered prop-
erty",
                                                       "type":"number"
                                                    },
                                                    "minimum":{
                                                       "description":"upper limit of the offered prop-
erty",
                                                       "type":"number"
                                                    },
                                                    "value":{
                                                       "description":"value of the property",
                                                       "anyOf":[
                                                          {
                                                             "type":"string"
                                                          },
                                                          {
                                                             "type":"object"
                                                          },
                                                          {
                                                             "type":"array"
                                                          }
                                                       ]
                                                    }
                                                 }
                                              }
                                           }
                                        }
                                     }
                                  }
                               }
                            }
                         }
                      }
                   },
                   "required":[
                      "name",
                      "constraints"
                   ]
                }
             }
          },
          "required":[
             "functionalProperties",
             "methods"
          ]
       },
       "ABSTRACT_PROPERTIES":{
          "type":"object",
          "properties":{
             "methods":{
                "description":"list of methods",
                "type":"array",
                "items":{
                   "type":"object",
                   "properties":{
                      "name":{
                         "description":"name of the method, as indicated in EXPOSED_API.Methods section",
```


© Main editor and other members of the DITAS consortium

```
                        "type":"string"
                     },
                     "constraints":{
                        "type":"object",
                        "description":"goal trees",
                        "properties":{
                           "dataUtility":{
                              "type":"object",
                              "description":"data utility goal tree",
                              "properties":{
                                 "treeStructure":{
                                    "type":"object",
                                    "properties":{
                                       "type":{
                                          "description":"type of node (AND or OR)",
                                          "type":"string"
                                       },
                                       "children":{
                                          "type":"array",
                                          "items":{
                                             "$ref":"#/properties/DATA_MANAGEMENT/properties/methods/items/proper-
ties/constraints/properties/dataUtility/properties/goals/items"
                                          }
                                       },
                                       "leaves":{
                                          "type":"array",
                                          "items":{
                                             "type":"string"
                                          }
                                       }
                                    }
                                 }
                              }
                           },
                           "security":{
                              "type":"object",
                              "description":"security goal tree",
                              "properties":{
                                 "treeStructure":{
                                    "type":"object",
                                    "properties":{
                                       "type":{
                                          "description":"type of node (AND or OR)",
                                          "type":"string"
                                       },
                                       "children":{
                                          "type":"array",
                                          "items":{
                                             "$ref":"#/properties/DATA_MANAGEMENT/properties/methods/items/proper-
ties/constraints/properties/security/properties/goals/items"
                                          }
                                       },
                                       "leaves":{
                                          "type":"array",
                                          "items":{
                                             "type":"string"
                                          }
                                       }
                                    }
                                 }
                              }
                           },
                           "privacy":{
                              "type":"object",
                              "description":"privacy goal tree",
                              "properties":{
                                 "treeStructure":{
                                    "type":"object",
                                    "properties":{
                                       "type":{
                                          "description":"type of node (AND or OR)",
                                          "type":"string"
                                       },
                                       "children":{
                                          "type":"array",
                                          "items":{
                                             "$ref":"#/properties/DATA_MANAGEMENT/properties/methods/items/proper-
ties/constraints/properties/privacy/properties/goals/items"
                                          }
                                       },
                                       "leaves":{
                                          "type":"array",
                                          "items":{
                                             "type":"string"
                                          }
                                       }
                                    }
                                 }
                              }
```


© Main editor and other members of the DITAS consortium

```
                        }
                     }
                  }
               }
            }
         }
      }
   },
   "COOKBOOK_APPENDIX":{
      "type":"object",
      "description":"Deployment Information of the VDC Blueprint",
      "properties":{
         "name":{
            "type":"string"
         },
         "description":{
            "type":"string"
         },
         "Infrastructure":{
            "type":"array",
            "description":"an array with the description of the infrastructure",
            "items":{
               "title":"an infrastructure definition",
               "type":"object",
               "properties":{
                  "name":{
                     "type":"string"
                  },
                  "description":{
                     "type":"string"
                  },
                  "type":{
                     "type":"string"
                  },
                  "on-line":{
                     "type":"boolean"
                  },
                  "api_endpoint":{
                     "type":"string"
                  },
                  "api_type":{
                     "type":"string"
                  },
                  "keypair_id":{
                     "type":"string"
                  },
                  "resources":{
                     "type":"array",
                     "items":{
                        "type":"object",
                        "properties":{
                           "name":{
                              "type":"string"
                           },
                           "type":{
                              "type":"string"
                           },
                           "cpus":{
                              "type":"string"
                           },
                           "ram":{
                              "type":"string"
                           },
                           "disk":{
                              "type":"string"
                           },
                           "generate_ssh_keys":{
                              "type":"boolean"
                           },
                           "ssh_keys_id":{
                              "type":"string"
                           },
                           "role":{
                              "type":"string"
                           },
                           "baseimage":{
                              "type":"string"
                           },
                           "arch":{
                              "type":"string"
                           },
                           "os":{
                              "type":"string"
                           }
                        }
                     }
                  }
               }
            }
         }
      }
```


© Main editor and other members of the DITAS consortium

```
            },
            "provision":{
               "type":"object",
               "properties":{
                  "provisioner":{
                     "type":"string"
                  },
                  "provisioner_src":{
                     "type":"string"
                  },
                  "components":{
                     "type":"array",
                     "items":{
                        "type":"object",
                        "component_name":{
                           "type":"string"
                        }
                     },
                     "nodes":{
                        "type":"array",
                        "items":{
                           "type":"string"
                        }
                     },
                     "conf_descriptor":{
                        "type":"string"
                     },
                     "params":{
                        "type":"object",
                        "additionalProperties":{
                           "type":"string",
                           "description":"node_names"
                        }
                     }
                  }
               }
            }
         }
      },
      "EXPOSED_API":{
         "title":"CAF API",
         "type":"object",
         "properties":{
            "Methods":{
               "type":"array",
               "items":{
                  "title":"An exposed VDC method",
                  "type":"object",
                  "properties":{
                     "name":{
                        "type":"string",
                        "description":"The name of each method must be unique, meaning that different methods MUST
have different names",
                        "pattern":"^([a-zA-Z0-9]+-)*[a-zA-Z0-9]+$|^([a-zA-Z0-9]+_)*[a-zA-Z0-9]+$"
                     },
                     "description":{
                        "type":"string",
                        "description":"Free text that briefly describes the functionality of the method"
                     },
                     "requestBody_schema":{
                        "type":"object",
                        "description":"The JSON schema of the body of an HTTP request to the method"
                     },
                     "output_schema":{
                        "type":"object",
                        "description":"The output JSON schema of the method"
                     },
                     "data_sources":{
                        "type":"array",
                        "description":"An array that contains all the names of the data sources, as indicated in
INTERNAL_STRUCTURE.Data_Sources section, accessed by the method",
                        "items":{
                           "type":"string"
                        },
                        "minItems":1,
                        "uniqueItems":true
                     },
                     "parameters":{
                        "description":"Any other parameters that the Data Administrator would like to add to the
method"
                     }
                  },
                  "additionalProperties":false,
                  "required":[
                     "name",
                     "output_schema",
                     "data_sources"
                  ]
               },
```


© Main editor and other members of the DITAS consortium

```
               "minItems":1,
               "uniqueItems":true
            },
            "RESTful_API":{
               "description":"The complete API of the VDC, written according to the OpenAPI Specification or any
other Specification"
            },
            "general_parameters":{
               "description":"Any other parameters that the Data Administrator would like to add to the API"
            }
         },
         "additionalProperties":false,
         "required":[
            "Methods",
            "RESTful_API"
         ]
      }
   },
   "additionalProperties":false,
   "required":[
      "INTERNAL_STRUCTURE",
      "DATA_MANAGEMENT",
      "ABSTRACT_PROPERTIES",
      "COOKBOOK_APPENDIX",
      "EXPOSED_API"
   ]
}
```

© Main editor and other members of the DITAS consortium

Annex B. VDC Blueprint Example


```
{
   "INTERNAL_STRUCTURE":{
      "Overview":{
         "name":"VDC_1",
         "description":"This VDC provides information about the patients of the general hospital of Milan",
         "tags":[
            {
               "method_name":"patient-details",
               "tags":[
                  "Milan",
                  "patients"
               ]
            }
         ]
      },
      "Data_Sources":[
         {
            "name":"MySQLPatientDetails",
            "type":"JDBC",
            "parameters":{
               "jdbc_url":"jdbc:mysql://localhost:3306/sakila?profileSQL=true",
               "driver":"com.mysql.jdbc_5.1.5.jar"
            }
         }
      ],
      "Flow":{
         "platform":"Node-RED",
         "source_code":[
            {
               "id":"dcb2aaf0.9e2e48",
               "type":"tab",
               "label":"CAF",
               "disabled":false,
               "info":""
            },
            {
               "id":"b590daea.01c548",
               "type":"tab",
               "label":"patient-details",
               "disabled":false,
               "info":""
            },
            {
               "id":"94cf7fc3.f0bab",
               "type":"subflow",
               "name":"DITAS CAF",
               "info":"",
               "in":[

               ],
               "out":[
                  {
                     "x":854.9999923706055,
                     "y":1064.9998779296875,
                     "wires":[
                        {
                           "id":"d7802ea4.e7f43",
                           "port":0
                        }
                     ]
                  }
               ]
            },
            {
               "id":"b4332de4.7304e",
               "type":"MySQLdatabase",
               "z":"",
               "host":"",
               "port":"3306",
               "db":"ditas_dummy_example",
               "tz":""
            },
            {
               "id":"dbf13448.4be018",
               "type":"http in",
               "z":"94cf7fc3.f0bab",
               "name":"HTTP request",
               "url":"/CAF/:exposed_method",
               "method":"post",
               "upload":false,
               "swaggerDoc":"",
               "x":104.78118896484375,
               "y":46.164727210998535,
               "wires":[
                  [
                     "389ebc6d.6ca324"
```


© Main editor and other members of the DITAS consortium

```
                ]
            ]
        },
        {
            "id":"d7802ea4.e7f43",
            "type":"switch",
            "z":"94cf7fc3.f0bab",
            "name":"EXPOSED_API.Methods",
            "property":"req.params.exposed_method",
            "propertyType":"msg",
            "rules":[
                {
                    "t":"eq",
                    "v":"patient-details",
                    "vt":"str"
                },
                {
                    "t":"else"
                }
            ],
            "checkall":"false",
            "repair":false,
            "outputs":2,
            "x":332.7897033691406,
            "y":1258.6193180084229,
            "wires":[
                [

                ],
                [
                    "b099ac96.d4a2b"
                ]
            ]
        },
        {
            "id":"389ebc6d.6ca324",
            "type":"switch",
            "z":"94cf7fc3.f0bab",
            "name":"check that purpose exists",
            "property":"$exists(payload.purpose)",
            "propertyType":"jsonata",
            "rules":[
                {
                    "t":"false"
                },
                {
                    "t":"else"
                }
            ],
            "checkall":"false",
            "outputs":2,
            "x":262.01422119140625,
            "y":208.0056915283203,
            "wires":[
                [
                    "e6ee32cf.073dd"
                ],
                [
                    "cdd1da0e.446448"
                ]
            ]
        },
        {
            "id":"e6ee32cf.073dd",
            "type":"function",
            "z":"94cf7fc3.f0bab",
            "name":"error response",
            "func":"msg.payload = \"the body of every HTTP request should contain the purpose of the re-
quest\";\nreturn msg;",
            "outputs":1,
            "noerr":0,
            "x":642.2442855834961,
            "y":281.9261236190796,
            "wires":[
                [
                    "a83625f3.a841a8"
                ]
            ]
        },
        {
            "id":"a83625f3.a841a8",
            "type":"http response",
            "z":"94cf7fc3.f0bab",
            "name":"HTTP response",
            "statusCode":"",
            "headers":{

            },
            "x":713.241569519043,
```


© Main editor and other members of the DITAS consortium

```
            "y":169.75292491912842,
            "wires":[

            ]
        },
        {
            "id":"cdd1da0e.446448",
            "type":"function",
            "z":"94cf7fc3.f0bab",
            "name":"strore purpose details to msg",
            "func":"msg.purpose = {};\n\nmsg.purpose.value = msg.payload.purpose;\n\nmsg.purpose.type =
typeof(msg.payload.purpose);\n\nreturn msg;",
            "outputs":1,
            "noerr":0,
            "x":311.24436950683594,
            "y":306.9262409210205,
            "wires":[
                [
                    "cd2051d0.45e19"
                ]
            ]
        },
        {
            "id":"cd2051d0.45e19",
            "type":"switch",
            "z":"94cf7fc3.f0bab",
            "name":"check that purpose is string",
            "property":"purpose.type",
            "propertyType":"msg",
            "rules":[
                {
                    "t":"neq",
                    "v":"string",
                    "vt":"str"
                },
                {
                    "t":"else"
                }
            ],
            "checkall":"false",
            "outputs":2,
            "x":300.24436950683594,
            "y":429.65356159210205,
            "wires":[
                [
                    "f483345f.93a2b8"
                ],
                [
                    "f58284f2.3ea558"
                ]
            ]
        },
        {
            "id":"f483345f.93a2b8",
            "type":"function",
            "z":"94cf7fc3.f0bab",
            "name":"error response",
            "func":"msg.payload = \"the purpose of the HTTP request should be a string\";\nreturn msg;",
            "outputs":1,
            "noerr":0,
            "x":626.2443771362305,
            "y":428.4716901779175,
            "wires":[
                [
                    "23ee1013.9669e"
                ]
            ]
        },
        {
            "id":"23ee1013.9669e",
            "type":"http response",
            "z":"94cf7fc3.f0bab",
            "name":"HTTP response",
            "statusCode":"",
            "headers":{

            },
            "x":745.2415542602539,
            "y":335.2984094619751,
            "wires":[

            ]
        },
        {
            "id":"df38bb28.9b3118",
            "type":"function",
            "z":"94cf7fc3.f0bab",
            "name":"error response",
```


© Main editor and other members of the DITAS consortium

```
            "func":"msg.payload = \"the body of every HTTP request should contain the requester_id of the re-
quester\";\nreturn msg;",
            "outputs":1,
            "noerr":0,
            "x":647.2443695068359,
            "y":647.4715633392334,
            "wires":[
               [
                  "682582.1f479a8"
               ]
            ]
         },
         {
            "id":"682582.1f479a8",
            "type":"http response",
            "z":"94cf7fc3.f0bab",
            "name":"HTTP response",
            "statusCode":"",
            "headers":{

            },
            "x":870.2415313720703,
            "y":704.298433303833,
            "wires":[

            ]
         },
         {
            "id":"804c7937.51c3e8",
            "type":"function",
            "z":"94cf7fc3.f0bab",
            "name":"strore requester_id details to msg",
            "func":"msg.requester_id = {};\n\nmsg.requester_id.value = msg.payload.requester_id;\n\nmsg.re-
quester_id.type = typeof(msg.payload.requester_id);\n\nreturn msg;",
            "outputs":1,
            "noerr":0,
            "x":229.24436950683594,
            "y":898.471718788147,
            "wires":[
               [
                  "f4b8fb0f.ee5c88"
               ]
            ]
         },
         {
            "id":"f4b8fb0f.ee5c88",
            "type":"switch",
            "z":"94cf7fc3.f0bab",
            "name":"check that requester_id is string",
            "property":"requester_id.type",
            "propertyType":"msg",
            "rules":[
               {
                  "t":"neq",
                  "v":"string",
                  "vt":"str"
               },
               {
                  "t":"else"
               }
            ],
            "checkall":"false",
            "outputs":2,
            "x":274.244384765625,
            "y":1024.2899808883667,
            "wires":[
               [
                  "d72805c0.609648"
               ],
               [
                  "d7802ea4.e7f43"
               ]
            ]
         },
         {
            "id":"d72805c0.609648",
            "type":"function",
            "z":"94cf7fc3.f0bab",
            "name":"error response",
            "func":"msg.payload = \"the requester_id of the HTTP request should be a string\";\nreturn msg;",
            "outputs":1,
            "noerr":0,
            "x":622.244384765625,
            "y":1006.2899599075317,
            "wires":[
               [
                  "15db84f0.a13c5b"
               ]
            ]
```


© Main editor and other members of the DITAS consortium

```
        },
        {
            "id":"15db84f0.a13c5b",
            "type":"http response",
            "z":"94cf7fc3.f0bab",
            "name":"HTTP response",
            "statusCode":"",
            "headers":{

            },
            "x":705.2415771484375,
            "y":876.1166486740112,
            "wires":[

            ]
        },
        {
            "id":"f58284f2.3ea558",
            "type":"switch",
            "z":"94cf7fc3.f0bab",
            "name":"check that requester_id exists",
            "property":"$exists(payload.requester_id)",
            "propertyType":"jsonata",
            "rules":[
                {
                    "t":"false"
                },
                {
                    "t":"else"
                }
            ],
            "checkall":"false",
            "outputs":2,
            "x":305.01422119140625,
            "y":630.914794921875,
            "wires":[
                [
                    "df38bb28.9b3118"
                ],
                [
                    "804c7937.51c3e8"
                ]
            ]
        },
        {
            "id":"b099ac96.d4a2b",
            "type":"function",
            "z":"94cf7fc3.f0bab",
            "name":"error response",
            "func":"msg.payload = \"the exposed method that you selected is not included in the VDC Blue-
print\";\nreturn msg;",
            "outputs":1,
            "noerr":0,
            "x":769.0142211914062,
            "y":1386.1875,
            "wires":[
                [
                    "258a7eae.34e5d2"
                ]
            ]
        },
        {
            "id":"258a7eae.34e5d2",
            "type":"http response",
            "z":"94cf7fc3.f0bab",
            "name":"HTTP response",
            "statusCode":"",
            "headers":{

            },
            "x":966.0142211914062,
            "y":1491.1875,
            "wires":[

            ]
        },
        {
            "id":"c4eb7262.b81b1",
            "type":"link in",
            "z":"b590daea.01c548",
            "name":"patient-details",
            "links":[
                "ce536723.34952",
                "a825fc7c.12f2e"
            ],
            "x":215,
            "y":140,
            "wires":[
                [
```


© Main editor and other members of the DITAS consortium

```
                    "2d96815c.dc996e"
                ]
            ]
        },
        {
            "id":"61eb9802.2aba78",
            "type":"link out",
            "z":"b590daea.01c548",
            "name":"patient-details",
            "links":[
                "731361c1.56e26",
                "81c2d4d4.58b638"
            ],
            "x":1095,
            "y":140,
            "wires":[

            ]
        },
        {
            "id":"cc453a45.f98c38",
            "type":"subflow:94cf7fc3.f0bab",
            "z":"dcb2aaf0.9e2e48",
            "name":"",
            "x":390,
            "y":260,
            "wires":[
                [
                    "a825fc7c.12f2e"
                ]
            ]
        },
        {
            "id":"2d96815c.dc996e",
            "type":"function",
            "z":"b590daea.01c548",
            "name":"construct MySQL query",
            "func":"var SSN = msg.payload.attributes.SSN;\n\nmsg.topic = \"select * from patient where socialId
= \"+\"\\\"\"+SSN+\"\\\"\";\n\nreturn msg;",
            "outputs":1,
            "noerr":0,
            "x":226.2273178100586,
            "y":461.52301597595215,
            "wires":[
                [
                    "311b8ea9.2111b2"
                ]
            ]
        },
        {
            "id":"17bbf14f.87066f",
            "type":"function",
            "z":"b590daea.01c548",
            "name":"construct exposed data tuple",
            "func":"msg.payload = msg.payload[0];\n\nreturn msg;",
            "outputs":1,
            "noerr":0,
            "x":920,
            "y":460,
            "wires":[
                [
                    "61eb9802.2aba78"
                ]
            ]
        },
        {
            "id":"a825fc7c.12f2e",
            "type":"link out",
            "z":"dcb2aaf0.9e2e48",
            "name":"patient-details",
            "links":[
                "c4eb7262.b81b1"
            ],
            "x":695,
            "y":80,
            "wires":[

            ]
        },
        {
            "id":"f7d65dc8.9dc8",
            "type":"http response",
            "z":"dcb2aaf0.9e2e48",
            "name":"HTTP response",
            "statusCode":"",
            "headers":{

            },
            "x":600,
```


© Main editor and other members of the DITAS consortium

```
            "y":620,
            "wires":[

            ]
         },
         {
            "id":"81c2d4d4.58b638",
            "type":"link in",
            "z":"dcb2aaf0.9e2e48",
            "name":"HTTP response",
            "links":[
               "61eb9802.2aba78",
               "586d4d83.f57924",
               "9b6c786b.53f138",
               "c5a80e8.ea8ccf"
            ],
            "x":381.00565338134766,
            "y":617.8977880477905,
            "wires":[
               [
                  "f7d65dc8.9dc8"
               ]
            ]
         },
         {
            "id":"311b8ea9.2111b2",
            "type":"mysql",
            "z":"b590daea.01c548",
            "mydb":"b4332de4.7304e",
            "name":"query patient biographical data MySQL database",
            "x":590,
            "y":280,
            "wires":[
               [
                  "17bbf14f.87066f"
               ]
            ]
         }
      ]
   },
   "Testing_Output_Data":[
      {
         "method_name":"patient-details",
         "zip_data":"http://localhost:8080"
      }
   ]
},
"DATA_MANAGEMENT":{
   "functionalProperties":{
      "tbd":"tbd"
   },
   "methods":[
      {
         "name":"patient-details",
         "constraints":{
            "dataUtility":{
               "goals":[
                  {
                     "id":"1",
                     "name":"Service available",
                     "weight":"1",
                     "metrics":[
                        {
                           "id":"1",
                           "name":"Availability 95-99",
                           "type":"Availability",
                           "properties":[
                              {
                                 "name":"Availability",
                                 "unit":"percentage",
                                 "minimum":95,
                                 "maximum":99
                              }
                           ]
                        }
                     ]
                  },
                  {
                     "id":"2",
                     "name":"Fast data process",
                     "weight":"1",
                     "metrics":[
                        {
                           "id":"2",
                           "name":"ResponseTime 1",
                           "type":"ResponseTime",
                           "properties":[
                              {
                                 "name":"ResponseTime",
```


© Main editor and other members of the DITAS consortium

```
                        "maximum":1,
                        "unit":"second"
                     }
                  ]
               }
            ]
         },
         {
            "id":"3",
            "name":"Data volume",
            "weight":"1",
            "metrics":[
               {
                  "id":"3",
                  "name":"volume 10000",
                  "type":"volume",
                  "properties":[
                     {
                        "name":"volume",
                        "value":"10000",
                        "unit":"tuple"
                     }
                  ]
               }
            ]
         },
         {
            "id":"4",
            "name":"Temporal validity",
            "weight":"1",
            "metrics":[
               {
                  "id":"4",
                  "name":"Timeliness 0.6",
                  "type":"Timeliness",
                  "properties":[
                     {
                        "name":"Timeliness",
                        "maximum":0.6,
                        "unit":"NONE"
                     }
                  ]
               }
            ]
         },
         {
            "id":"5",
            "name":"Amount of Data",
            "weight":"1",
            "metrics":[
               {
                  "id":"5",
                  "name":"Process completeness 90",
                  "type":"Process completeness",
                  "properties":[
                     {
                        "name":"Process completeness",
                        "minimum":90,
                        "unit":"percentage"
                     }
                  ]
               }
            ]
         }
      ]
   },
   "security":{
      "goals":[
         {
            "id":"1",
            "name":"Encryption",
            "metrics":[
               {
                  "id":"1",
                  "name":"Encryption AES 128",
                  "type":"Encryption",
                  "properties":[
                     {
                        "name":"Algorithm",
                        "unit":"enum",
                        "value":"AES"
                     },
                     {
                        "name":"Keylength",
                        "unit":"number",
                        "minimum":128
                     }
                  ]
               }
```


© Main editor and other members of the DITAS consortium

```
                ]
             },
             {
                "id":"2",
                "name":"Tracing",
                "metrics":[
                   {
                      "id":"2",
                      "name":"Tracing",
                      "type":"Tracing",
                      "properties":[
                         {
                            "name":"Level",
                            "unit":"enum",
                            "value":"datasource"
                         },
                         {
                            "name":"SampleRate",
                            "unit":"percentage",
                            "minimum":99,
                            "maximum":100
                         }
                      ]
                   }
                ]
             },
             {
                "id":"3",
                "name":"ACL rolebased readOnly",
                "metrics":[
                   {
                      "id":"3",
                      "name":"ACL rolebased readOnly",
                      "type":"ACL",
                      "properties":[
                         {
                            "name":"Kind",
                            "unit":"enum",
                            "value":"rolebased"
                         },
                         {
                            "name":"Role",
                            "unit":"enum",
                            "value":"readOnly"
                         }
                      ]
                   }
                ]
             },
             {
                "id":"4",
                "name":"ACL rolebased readNonPersonal",
                "metrics":[
                   {
                      "id":"4",
                      "name":"ACL rolebased readNonPersonal",
                      "type":"ACL",
                      "properties":[
                         {
                            "name":"Kind",
                            "unit":"enum",
                            "value":"rolebased"
                         },
                         {
                            "name":"Role",
                            "unit":"enum",
                            "value":"readNonPersonal"
                         }
                      ]
                   }
                ]
             }
          ]
       },
       "privacy":{
          "goals":[
             {
                "id":"1",
                "name":"PurposeControl",
                "metrics":[
                   {
                      "id":"1",
                      "name":"PurposeControl NonCommercial Government",
                      "type":"PurposeControl",
                      "properties":[
                         {
                            "name":"AllowedPurpose",
                            "unit":"enum",
                            "value":"NonCommercial"
```


© Main editor and other members of the DITAS consortium

```
                                },
                                {
                                   "name":"AllowedGuarantor",
                                   "unit":"enum",
                                   "value":"Government"
                                }
                             ]
                          }
                       ]
                    }
                 ]
              }
           }
        }
     ]
  },
  "ABSTRACT_PROPERTIES":{
     "methods":[
        {
           "name":"patient-details",
           "constraints":{
              "dataUtility":{
                 "treeStructure":{
                    "type":"AND",
                    "children":[
                       {
                          "type":"OR",
                          "leaves":[
                             "1",
                             "4"
                          ]
                       },
                       {
                          "type":"AND",
                          "children":[
                             {
                                "type":"OR",
                                "leaves":[
                                   "3",
                                   "5"
                                ]
                             }
                          ],
                          "leaves":[
                             "2"
                          ]
                       }
                    ]
                 }
              },
              "security":{
                 "treeStructure":{
                    "type":"AND",
                    "children":[
                       {
                          "type":"OR",
                          "leaves":[
                             "1",
                             "2"
                          ]
                       }
                    ],
                    "leaves":[
                       "3",
                       "4"
                    ]
                 }
              },
              "privacy":{
                 "treeStructure":{
                    "type":"OR",
                    "leaves":[
                       "1"
                    ]
                 }
              }
           }
        }
     ]
  },
  "COOKBOOK_APPENDIX":{
     "name":"name of the deployment",
     "description":"description of the deployment",
     "infrastructure":[
        {
           "name":"...",
           "description":"...",
           "type":"cloud",
           "on-line":true,
```


© Main editor and other members of the DITAS consortium

```
            "api_endpoint":"api url",
            "api_type":"cloudsigma",
            "keypair_id":"keypair_uuid",
            "resources":[
               {
                  "name":"unique_name_1",
                  "type":"vm",
                  "cpus":"8",
                  "ram":"4096",
                  "disk":"512000000",
                  "generate_ssh_keys":false,
                  "ssh_keys_id":"uuid",
                  "role":"type of role, kubernate master, etc...",
                  "baseimage":"optional parameter to indicate OS type"
               },
               {
                  "name":"...",
                  "description":"...",
                  "type":"edge",
                  "on-line":false,
                  "generate_ssh_keys":true,
                  "role":"type of rolo...",
                  "arch":"processor architecture",
                  "os":"OS type"
               }
            ]
         }
      ],
      "provision":{
         "provisioner":"chef/ansible/puppet...",
         "provisioner_src":"http://recipes_manifests_playbooks.zip",
         "components":[

         ]
      }
   },
   "EXPOSED_API":{
      "Methods":[
         {
            "name":"patient-details",
            "description":"Given the SSN returns all the details of the patient",
            "requestBody_schema":{
               "type":"object",
               "properties":{
                  "purpose":{
                     "type":"string"
                  },
                  "requester_id":{
                     "type":"string"
                  },
                  "attributes":{
                     "type":"object",
                     "properties":{
                        "SSN":{
                           "type":"string"
                        }
                     },
                     "required":[
                        "SSN"
                     ],
                     "additionalProperties":false
                  }
               },
               "required":[
                  "purpose",
                  "requester_id",
                  "attributes"
               ],
               "additionalProperties":false
            },
            "output_schema":{
               "type":"object",
               "properties":{
                  "addressCity":{
                     "type":"string"
                  },
                  "addressRoad":{
                     "type":"string"
                  },
                  "addressRoadNumber":{
                     "type":"integer"
                  },
                  "birthCity":{
                     "type":"string"
                  },
                  "nationality":{
                     "type":"string"
                  },
                  "job":{
```


© Main editor and other members of the DITAS consortium

```
                  "type":"string"
               },
               "schoolYears":{
                  "type":"number"
               },
               "birthDate":{
                  "type":"string",
                  "format":"date-time"
               },
               "gender":{
                  "enum":[
                     "M",
                     "F"
                  ]
               },
               "name":{
                  "type":"string"
               },
               "patientId":{
                  "type":"string"
               },
               "socialId":{
                  "type":"string"
               },
               "surname":{
                  "type":"string"
               }
            },
            "required":[
               "birthDate",
               "gender",
               "name",
               "patientId",
               "socialId",
               "surname"
            ],
            "additionalProperties":false
         },
         "data_sources":[
            "MySQLPatientDetails"
         ]
      }
   ],
   "RESTful_API":{
      "openapi":"3.0.1",
      "info":{
         "title":"CAF API",
         "version":"0.0.1"
      },
      "servers":[
         {
            "description":"CAF server",
            "url":"http://localhost:1880/CAF"
         }
      ],
      "paths":{
         "/patient-details":{
            "post":{
               "tags":[
                  "patient biographical data"
               ],
               "requestBody":{
                  "content":{
                     "application/json":{

                     }
                  }
               },
               "responses":{
                  "200":{
                     "description":"successful HTTP request"
                  }
               }
            }
         }
      }
   }
}
```


© Main editor and other members of the DITAS consortium