H2020-EINFRA-2015-1

VI-SEEM

VRE for regional Interdisciplinary communities in Southeast Europe and the Eastern Mediterranean

Deliverable D3.2

Service registry, operational and service level monitoring

Author(s): A. Mishev (editor)

Status –Version: Final - e

Date: September 30, 2016

Distribution - Type: Public

Abstract: Deliverable D3.2 – "Service registry, operational and service level monitoring" – gives a detailed description of the VRE service registry, the operational methodology and the operational and service level monitoring for each type of service to be deployed within the VRE.

The VI-SEEM Consortium consists of:

GRNET	Coordinating Contractor	Greece
CYI	Contractor	Cyprus
IICT-BAS	Contractor	Bulgaria
IPB	Contractor	Serbia
NIIF	Contractor	Hungary
UVT	Contractor	Romania
UPT	Contractor	Albania

UNI BL	Contractor	Bosnia-Herzegovina
UKIM	Contractor	FYR of Macedonia
UOM	Contractor	Montenegro
RENAM	Contractor	Moldova (Republic of)
IIAP-NAS-RA	Contractor	Armenia
GRENA	Contractor	Georgia
BA	Contractor	Egypt
IUCC	Contractor	Israel
SESAME	Contractor	Jordan

The VI-SEEM project is funded by the European Commission under the Horizon 2020 e-Infrastructures grant agreement no. 675121.

Document Revision History

Date	Issue	Author/Editor/Contributor	Summary of main changes
September 06th, 2016	a	Anastas Mishev, Ioannis Liabotis	Initial version of ToC.
September 13th, 2016	b	Anastas Mishev, Ioannis Liabotis	Final ToC
September 26th, 2016	c	Anastas Mishev, Ioannis Liabotis, Dusan Vudragovic, Gjorgji Strezoski, Dario Stojanovski, Mihalo Savic, Artemis Zamani, Sonja Filiposka	Draft version with the input from the partners
September 29th, 2016	d	Ioannis Liabotis, Anastas Misev, Ognjen Prnjat	Final version with additions and corrections
September 30th, 2016	e	Ioannis Liabotis, Anastas Misev, Ognjen Prnjat, Evangelia Athanasaki	Final editing addressing the quality assurance comments

Preface

In the last decade, a number of initiatives were crucial for enabling high-quality research - by providing e-Infrastructure resources, application support and training - in both South East Europe (SEE) and Eastern Mediterranean (EM). They helped reduce the digital divide and brain drain in Europe, by ensuring access to regional e-Infrastructures to new member states, states on path to ascension, and states in European Neighbourhood Policy area – in total 14 countries in SEE and 6 in EM.

This VI-SEEM project brings together these e-Infrastructures to build capacity and better utilize synergies, for an improved service provision within a unified Virtual Research Environment (VRE) for the inter-disciplinary scientific user communities in the combined SEE and EM regions (SEEM). The overall objective is to provide user-friendly integrated e-Infrastructure platform for regional cross-border Scientific Communities in Climatology, Life Sciences, and Cultural Heritage for the SEEM region; by linking compute, data, and visualization resources, as well as services, models, software and tools. This VRE aspires to provide the scientists and researchers with the support in full lifecycle of collaborative research: accessing and sharing relevant research data, using it with provided codes and tools to carry out new experiments and simulations on large-scale e-Infrastructures, and producing new knowledge and data - which can be stored and shared in the same VRE. Climatology and Life Science communities are directly relevant for Societal Challenges.

The driving ambition of this proposal is to maintain leadership in enabling e-Infrastructure based research and innovation in the region for the 3 strategic regional user communities: supporting multidisciplinary solutions, advancing their research, and bridging the development gap with the rest of Europe. The VI-SEEM consortium brings together e-Infrastructure operators and Scientific Communities in a common endeavor.

The overall objective is to provide user-friendly integrated e-Infrastructure platform for Scientific Communities in Climatology, Life Sciences, and Cultural Heritage for the SEEM region; by linking compute, data, and visualization resources, as well as services, software and tools.

The detailed objectives of the VI-SEEM project are:

1. Provide scientists with access to state of the art e-Infrastructure - computing, storage and connectivity resources - available in the region; and promote additional resources across the region.
2. Integrate the underlying e-Infrastructure layers with generic/standardised as well as domain-specific services for the region. The latter are leveraging on existing tools (including visualization) with additional features being co-developed and co-operated by the Scientific Communities and the e-Infrastructure providers, thus proving integrated VRE environments.
3. Promote capacity building in the region and foster interdisciplinary approaches.
4. Provide functions allowing for data management for the selected Scientific Communities, engage the full data management lifecycle, link data across the region, provide data interoperability across disciplines.

5. Provide adequate user support and training programmes for the user communities in the SEEM region.
6. Bring high level expertise in e-Infrastructure utilization to enable research activities of international standing in the selected fields of Climatology, Life Sciences and Cultural Heritage.

The VI-SEEM project kicked-off in October 2015 and is planned to be completed by September 2018. It is coordinated by GRNET with 15 contractors from Cyprus, Bulgaria, Serbia, Hungary, Romania, Albania, Bosnia-Herzegovina, FYR of Macedonia, Montenegro, Moldova (Republic of), Armenia, Georgia, Egypt, Israel, Jordan. The total budget is 3.300.000 €. The project is funded by the European Commission's Horizon 2020 Programme for Excellence in Science, e-Infrastructure.

The project plans to issue the following deliverables:

Del. no.	Deliverable name	Nature	Security	Planned Delivery
D1.1	Project management information system and “grant agreement” relationships	R	CO	M01
D1.2	3-Monthly progress report	R	CO	M03n *
D1.3a	First period progress reports	R	CO	M18
D1.3b	Final period progress reports	R	CO	M36
D2.1	Internal and external communication platform, docs repository and mailing lists	DEC	PU	M02
D2.2	Promotional package	DEC	PU	M04
D2.3	Dissemination and marketing plan	R	PU	M05
D2.4	Training plan	R	PU	M06
D2.5	Promotional package with updates	R	PU	M16
D2.6	1st Dissemination, training and marketing report	DEC	PU	M18
D2.7	2nd Dissemination, training and marketing report	R	PU	M35
D3.1	Infrastructure and services deployment plan	R	PU	M04
D3.2	Service registry, operational and service level monitoring	R	PU	M12
D3.3	Infrastructure overview, assessment and refinement plan	R	PU	M18
D3.4	VRE AAI Model and compatibility with other	R	PU	M27

	eInfrastructures			
D3.5	Final infrastructure overview and assessment report	R	PU	M36
D4.1	Data sources and services deployment plan	R	PU	M06
D4.2	Description of the initial deployed data services	R	PU	M11
D4.3	Description of the final data platform available to VRE users	R	PU	M23
D4.4	Final report on data, services, availability and usage	R	PU	M35
D5.1	Detailed technical implementation plan for VRE services and tools	R	PU	M04
D5.2	Data management plans	R	PU	M06
D5.3	User-oriented documentation and training material for VRE services	R	PU	M13
D5.4	Report on integrated services and the VRE platform	R	PU	M14
D5.5	Final report on integrated services and the VRE platform	R	PU	M36
D6.1	Framework for VRE resource and service provision	R	PU	M09
D6.2	1st Report of open calls and integration support	R	PU	M20
D6.3	Sustainability and business model	R	PU	M24
D6.4	2nd Report of open calls and integration support	R	PU	M36

Legend: R = Document, report, DEC = Websites, patent fillings, videos, etc., PU = Public, CO = Confidential, only for members of the consortium (including the Commission Services).

*n=1,2,3,…12

Table of contents

References

[1] The FitSM Standard
http://fitsm.itemo.org/

[2] Jack Probst, “ANATOMY OF A SERVICE, A Practical Guide To Defining IT Services”, Whitepaper, September 2009

[3] Thomas Erl, "Service-Oriented Architecture: Concepts, Technology, and Design", ISBN: 0131858580, Prentice Hall/Pearson PTR, 2006

[4] IT Service Management
http://www.itsm.info/ITSM.htm

[5] ITIL – IT Service Management books
http://www.itil.org.uk/

[6] VI-SEEM Deliverable D3.1.: Infrastructure and services deployment plan

[7] EUDAT2020 Research Data Services, Expertise & Technology Solutions
https://www.eudat.eu/

[8] PRACE Fourth Implementation Phase (PRACE-4IP) project
http://www.prace-ri.eu/

[9] Django framework
https://www.djangoproject.com

[10] Bootstrap library
http://getbootstrap.com

[11] jQuery
https://jquery.com

[12] Jenkins
https://jenkins.io

[13] Swagger
http://swagger.io

[14] VI-SEEM code repository
https://code.vi-seem.eu

[15] VI-SEEM official web site
https://vi-seem.eu

[16] VI-SEEM GOCDB instance
https://gocdb.vi-seem.eu/portal/

[17] GLUE 2
http://www.ogf.org/documents/GFD.147.pdf

[18] CLARIN-EL
http://www.clarin.gr/en

[19] Nagios
http://www.clarin.gr/en

[20] ChemBioServer
http://bioserver-3.bioacademy.gr/Bioserver/ChemBioServer/

[21] VI-SEEM helpdesk
http://support.vi-seem.eu

[22] VI-SEEM deliverable D1.1: Project management information system and “grant agreement” relationships

[23] VI-SEEM Operational team
http://wiki.vi-seem.eu/index.php/Access_to_resources

[24] VI-SEEM Service enablers
http://wiki.vi-seem.eu/index.php/VI-SEEM_Services_Integration

List of Figures

List of Tables

Glossary

API	Application Programming Interface
CMDB	Configuration Management Database
FitSM	Authentication and Authorization Infrastructure
GOCDB	Grid Configuration Database
IDE	Integrated Development Environment
IdP	Identity Provider
ITIL	Information Technology Infrastructure Library
ITSM	IT Service Management
JSON	JavaScript Object Notation
OLA	Operation Level Agreement
OOD	Operator On Duty
SAML	Security Assertion Markup Language
SLA	Service Level Agreement
SLM	Service Level Management
SP	Service Provider
VI-SEEM	VRE for regional Interdisciplinary communities in Southeast and the Eastern Mediterranean
VRE	Virtual Research Environment

Executive summary

What is the focus of this Deliverable?

The focus of this deliverable is to describe the elements of the project needed to establish the service orientation. It covers the service management view of the project, the systems for service catalogue and portfolio management, monitoring infrastructure, and provides the framework for the desired service and operational levels.

What is next in the process to deliver the VI-SEEM results?

The contents of this deliverable will form a basis for the other work packages to make efficient use of the e-Infrastructure services. The service portfolio and catalogue management system, whose design and implementation is presented in this deliverable, includes both the common services (WP3 for e-Infrastructure and WP4 for data services) that are used by all three scientific communities, as well as the application-specific services that WP5 will deploy. The monitoring system will support the full service lifecycle and enable efficient implementation of the operational and service level definitions.

In particular, the content of this deliverable will be used in the following VI-SEEM activities:

- WP4.1 – Data services design
- WP4.3 – Data collection and provisioning
- WP4.4 – Data analysis
- WP5.1 – Refinement of service requirements & tech assessment for integration
- WP5.3 – Development of the VRE platform
- WP5.4 – Overall integration of services

What are the deliverable contents?

The deliverable contents include an overview of the necessity for IT service management in order to enable a service-oriented view of the underlying e-Infrastructure. It also provides information about the developed tools necessary to publish the current and future services that will be developed within the project, including the monitoring procedures, service level framework and monitoring infrastructure all in place with the aim to support the successful delivery of the services. In detail, the deliverable includes:

- Overview of the service management process and tools used to support the service management, created using the FitSM standard [1] as guideline (Section 2).
- Monitoring methodology and infrastructure supporting the service orientation and service management (Section 3).
- Operational and service level definitions needed to be maintained for continuous delivery of the VI-SEEM services to the scientific communities (Section 4).

Conclusions and recommendations

The necessity for solid but flexible IT service management becomes one of the keystones of the foundation for the service-oriented design. The specifics of the federated environment, such as the one found in the VI-SEEM consortium, impose requirements for service

management tools that cannot be met using common off-the-shelf solutions. Hence, special care must be taken for the design and the implementation of easy to use, custom solutions that are tailor made for scientific communities. The main points regarding the service management infrastructure development and deployment can be summarized in the following:

- Cleary defined service management processes using a standard that supports federated environment must be used as the foundation for the IT service management solution design.
- The implemented service management solution will enable various categories of users to access the set of services offered by the project VRE platform. While the end-users from all of the scientific communities can browse and search through the service catalogue, the management interface provides full control over the complete service portfolio for the project.
- Both the catalogue and the portfolio are accessible via RESTful API, enabling easy integration with other components of the Virtual Research Environment.
- The service registry, implemented as a GOCDB instance, enables the discovery of the endpoints for the monitoring system.
- The proposed community-based monitoring, specifically designed for the VI-SEEM project, enables each of the three scientific communities to have the in-depth view of availability of the community-specific services.
- The proposed approach will enable the project to identify and define service and operational level agreements specific to the needs of the users of the three scientific communities.

The efforts presented in this deliverable are strongly aligned with the service orientation approach of the overall VI-SEEM service provisioning architecture, maintaining compatibility with other EU e-Infrastructure projects and external resource providers.

1 Introduction

A service is a means of delivering value to customers by facilitating outcomes customers want to achieve without the ownership of specific costs and risks [2]. As long as the task or function being provided is well defined and can be relatively isolated from other associated tasks it can be distinctly classified as a service.

Service orientation is based on several principles; among which these are the most important [3]:

- Services share a formal contract - services express their purpose and capabilities via a service contract. The Standardized Service Contract design principle is perhaps the most fundamental part of service-orientation in that it essentially requires that specific considerations be taken into account when designing a service's public technical interface and assessing the nature and quantity of content that will be published as part of a service's official contract.
- Services are loosely coupled - this principle advocates the creation of a specific type of relationship within and outside of service boundaries, with a constant emphasis on reducing ("loosening") dependencies between the service contract, its implementation, and its service consumers.
- Services abstract underlying logic - on a fundamental level, this principle emphasizes the need to hide as much of the underlying details of a service as possible. Doing so directly enables and preserves the previously described loosely coupled relationship.
- Services can be composed - the ability to effectively compose services is a critical requirement for achieving some of the most fundamental goals of service-oriented computing. Services are expected to be capable of participating as effective composition members, regardless of whether they need to be immediately enlisted in a composition.
- Services are reusable - emphasizes the positioning of services as enterprise resources with agnostic functional contexts.
- Services are autonomous - for services to carry out their capabilities consistently and reliably, their underlying solution logic needs to have a significant degree of control over its environment and resources. The principle of Service Autonomy supports the extent to which other design principles can be effectively realized in real world production environments by fostering design characteristics that increase a service's reliability and behavioral predictability.
- Services are stateless - management of excessive state information can compromise the availability of a service and undermine its scalability potential. Services are therefore ideally designed to remain stateful only when required.
- Services are discoverable - for services to be positioned as IT assets with repeatable ROI, they need to be easily identified and understood when opportunities for reuse present themselves. The service design therefore needs to take the "communications quality" of the service and its individual capabilities into account, regardless of

whether a discovery mechanism (such as a service registry) is an immediate part of the environment.

The service orientation has its own requirements that have to be met. When talking about IT services, the key element of the successful service orientation is the IT service management [4]. IT Service Management provides the necessary guidance for an IT organization to plan, design, develop, deploy and support business aligned IT Services. These services include the hardware, software and other IT assets necessary as well as the overall guidance for the IT organization in the provision of these services. Therefore, ITSM provides the necessary underpinning processes required to realize an infrastructure delivery platform that would support a Service-oriented Architecture.

There are many standards and best practices that cover the area of IT service management. ITIL [5] is one of the most frequently used, especially in the large enterprises. The IT Infrastructure Library (ITIL®) as we all know provides us with a descriptive framework of best practices for the delivery of the components of the IT infrastructure as a set of services to the enterprise. But ITIL cannot be as successfully implemented in a case of federated environment, similar to the one built by most of the e-Infrastructure international projects. The current efforts toward addressing these issues led to the establishment of a new, lightweight standard, with special focus on the federated environment, the FitSM [1].

FitSM is a free and lightweight standards family aimed at facilitating service management in IT service provision, including federated scenarios. The main goals of FitSM are:

- Create a clear, pragmatic, lightweight and achievable standard that allows for effective IT service management (ITSM).
- Offer a version of ITSM that can cope with federated environments, which often lack the hierarchy and level of control seen in other situations.
- Provide a baseline level of ITSM than can act to support ‘management interoperability’ in federated environments where disparate or competing organisations must cooperate to manage services.

To successfully implement the service management process into an e-Infrastructure, there are several systems and processes that have to be put in place. The key elements are:

- Service management system, consisting of service catalogue and portfolio - the Service Portfolio is used to manage the entire Lifecycle of all Services, and includes three Categories: Service Pipeline (proposed or in Development); Service Catalogue (live or available for Deployment); and Retired Services.
- Service registry - The service registry is a database populated with information on how to dispatch requests to service instances.
- Monitoring system - real-time observation of and alerting about health conditions (characteristics that indicate success or failure) in an IT environment. It helps to ensure that deployed services are operated, maintained, and supported in line with the service level agreement (SLA) targets agreed to between the business and IT.

- Operational and service level definitions – SLAs define what the organization as a whole is promising to the customer, while OLAs define what the functional internal groups promise to each other.

In the rest of the deliverable the implementation and usage of each of these key elements is described in detail.

2 Service management

IT Service Management is a set of specialized organizational capabilities for providing value to customers in the form of services [4]. It aims at providing high quality IT services meeting customers' and users' expectations by defining, establishing and maintaining service management processes. The process itself can be defined as a set of activities that bring about a specific objective or set of results from a set of defined inputs.

The most important role of IT service management processes is to support the delivery of IT services. In many cases, the provisioning of one IT service requires several processes. All of these processes need to be successfully operating and interacting to deliver an IT service.

2.1 Service management process

An IT service management process, as any other process has a number of standard elements, including:

- Goals and objectives
- Clearly defined inputs, triggers and outputs
- Set of interrelated activities
- Roles and responsibilities

Having in mind the federated nature of the VI-SEEM consortium, the FitSM was the clear choice for governing the IT service management of the offered services.

2.1.1 FitSM

The goals and activities of the FitSM standard series are aimed at supporting effective, lightweight IT service management (ITSM) processes in an organization (or part of an organization delivering IT services to customers, and harmonizing ITSM across federated computing infrastructures. FitSM is designed to be compatible with the International Standard ISO/IEC 20000-1 (requirements for a service management system) and the IT Infrastructure Library (ITIL).

The FitSM standard was analyzed in depth within the D3.1 deliverable [6]: please refer to this document for more information on the standard.

2.2 Service management system

A service management system is an overall management system that controls and supports management of services within an organization or federation.

According to FitSM, the first requirement for implementing the service management system is PR1: Service portfolio management. The rest of the section 2 refers to the developed system for IT service catalogue and portfolio management for the needs of VI-SEEM project.

2.2.1 Requirements

The VI-SEEM service portfolio management system has been developed to support the service portfolio management process within VI-SEEM as well as being usable for other infrastructures if required. The main requirements for the creation of this tool have been collected from the service management process design within VI-SEEM work packages WP3 (infrastructure services), WP4 (storage services) and WP5 (application level services). The service management system has been designed to be compatible with the FitSM service portfolio management. Requirements gathered in the context of EUDAT2020 [7] project have also been considered for compatibility and completeness. . Following is a list of the main functional requirements that drove the development of the service portfolio management system.

- There is a set of different roles that should have access to the functionalities of the tool. These are:
    - The **potential customers** or **end users** of the services that are listed in the **service catalogue**. Such users should be able to see the list and details of the services that are currently into production or beta stages and are for offer to them. Such users should be also able to order or use the services via the service catalogue, interact with the helpdesk or the dedicated support channel for that service and see features and use cases of each service.
    - The **service managers** within the VI-SEEM environment. Such users should be able to see all the details for the services that are in the **service portfolio** which contains a superset of the service and the information found in the service catalogue.
    - The **service owners** that are the persons responsible for each service listed in the service portfolio. Members with this role have the full responsibility of the content that is provided within the service catalogue and portfolio for the services under their responsibility.
- The service catalogue contains only information about services that are to be provided to the potential customers and end users of the services. A subset of all the service information is provided for a subset of the services that are registered in the service portfolio.
- Each service can have multiple versions in the service portfolio. Each version can have a different readiness status i.e. “concept”, “under development”, “beta”, “production”, “retired” etc.
- The service owners and the service managers should be able to state which combinations of service and service versions should appear in the service catalogue.
- The service can be either customer facing or internal to the organization

- The dependencies between services should be depicted in the service portfolio in order to facilitate better the SLA management process.
- The components that are required for deploying each service should be detailed in the service portfolio providing information needed for the operations stuff to deploy such services.
- The service portfolio should be accessible via a RESTful API to accommodate 3rd party application building and different views of the service catalogue depending on the needs of the organization.
- The service portfolio/catalogue should have a default web UI to accommodate the main service catalogue view for the VI-SEEM project as well as to facilitate adding and editing of information from service managers and service owners.

2.2.2 Design

The design of the system was detailed in the D3.1: Infrastructure and services deployment plan [6]. The selected model takes into consideration similar approaches on service portfolio management and service catalogue that are being performed in projects such as EUDAT2020 [7] and PRACE-4IP [8]. The class diagram of the system is given in Figure 1.

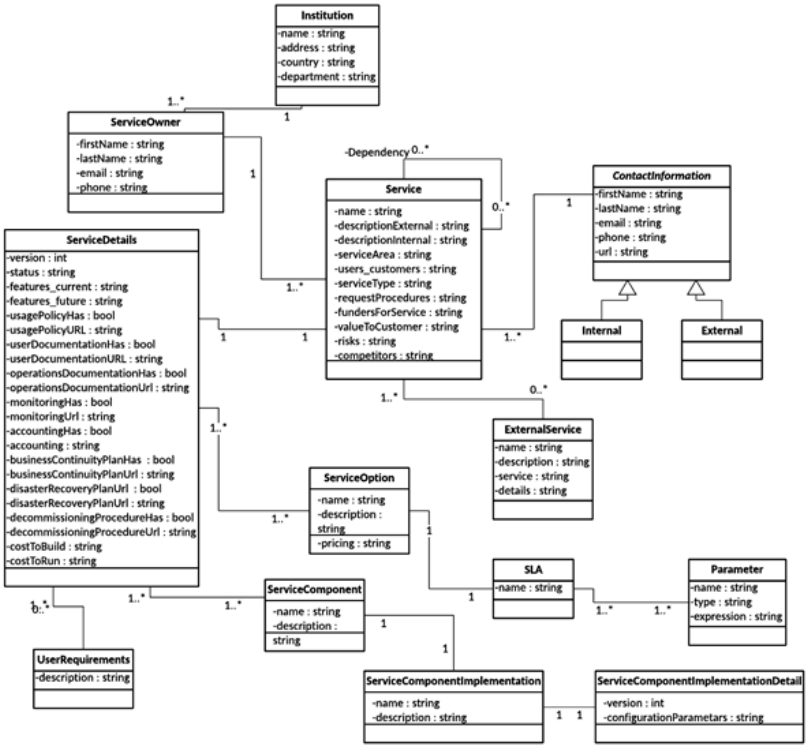


Figure 1 - Class diagram of the system

2.2.3 Technologies and environment for development

The main technology used for developing the project is Python, specifically the Django framework for web development [9]. This framework is used to provide the RESTful interface to the service catalogue and portfolio information. Additionally, it is also used for providing the appropriate User Interface for viewing and managing information in the database. Django is a high-level Python Web framework that encourages rapid development and clean, pragmatic design. Built by experienced developers, it takes care of much of the hassle of Web development, so you can focus on writing your app without needing to reinvent the wheel. It’s free and open source. It is compliant with modern standards and with an active development community is one of the best frameworks for rapid web development. The backend of the application is implemented using MySQL.

The web UI is implemented using standard web technologies such as HTML, CSS and JavaScript and common libraries such as Bootstrap [10] and JQuery [11]. In addition, for better compositionality, we use the React library. React was used for defining the views and core JavaScript and JQuery for managing the data and communicating with the server side. These technologies were chosen in front of AngularJS, which was initially chosen, because of their ease of use. React makes it easy to create interactive UIs. It enables to design simple or complex views for each state in the application and renders only the necessary components when the data changes. React works by enabling to define encapsulated components that manage their own state and composing them to create more complex UIs.

A good deal of the UI components were acquired from the BeyondAdmin template. This template provides a variety of different components, ranging from sidebar menus, tables, buttons etc. Each of the components is provided as a React class which enables for a quick integration in the project. Several test cases were implemented to check the validity of the implemented methods. This was achieved using the built-in test suite for Django.

2.2.3.1 DjagoSAML2 – SAML2 authentication package for Django

Djangosaml2 is a Django application that integrates the PySAML2 library into your project. This means that we can protect the Django based project with a service provider based on PySAML. This way we can use a SAML Identity Provider to control the authentication process and return a user ready to be processed by our application.

2.2.3.2 XMLSec – XML security library for XML security standards in Django

XML Security Library provides support for XML Digital Signature and XML Encryption. It is based on LibXML/LibXSLT and can use practicaly any crypto library (currently there is "out of the box" support for OpenSSL, MSCrypto, GnuTLS, GCrypt and NSS).

2.2.3.3 Django Social Auth - Package that provides user registration and login using social sites credentials

Registration and Login using social sites using the following providers:

- Google OpenID
- Google OAuth
- Google OAuth2
- Yahoo OpenID
- OpenId like myOpenID
- Twitter OAuth
- Facebook OAuth
- Amazon OAuth2

2.2.3.4 Django Jenkins

Django - Jenkins [12] is a package that enables continuous integration of a Django project by running the standard unit testing engine that is already present in the Django framework itself. It uses the manage.py method of running tests and integrates easily with projects that have not planned continuous integration as part of their development process.

2.2.3.5 Django custom loggers

Custom logging engine for handling multiple file logs on the server side.

2.2.3.6 Django REST Swagger

Swagger [13] is used to provide for a nice and clear way of presenting all of the implemented API endpoints. It creates a user interfaces where a description of the API call is presented alongside with the necessary request parameters and the expected response. Swagger/OpenAPI Documentation Generator for the Django REST Framework performing automatic comment parsing and endpoint detection.

2.2.3.7 Django filter

Django-filter is a generic, reusable application to alleviate writing some of the more mundane bits of view code. Specifically, it allows users to filter down a queryset based on a model’s fields, displaying the form to let them do this.

2.2.3.8 Django suit

As part of the application, there is an Admin panel for managing existing services and all other models. This was implemented using the built-in Django admin panel. In order to provide for better usability and because of its modern design, the admin panel is also implemented with the Django Suit library.

2.2.3.9 Django reversion

Django Reversion is an extension to the Django web framework that provides version control for model instances. It enables to roll back to any point in a model instance’s history and recover deleted model instances. In order to increase ease of use, it is seamlessly integrated with the Django admin panel.

2.2.3.10 Development environment

The project is developed and deployed on an Apache server. In order for the Python based framework to be compatible with the Apache server, we use a wsgi script that enables seamless integration. The application is developed locally on Ubuntu 14.04 operating system, while for internal checking while in development, it was deployed on a Debian server.

As a development environment, the PyCharm IDE is used. The MySQL database was managed through the phpMyAdmin in addition to its built-in terminal environment. phpMyAdmin is a free software tool written in PHP, intended to handle the administration of MySQL over the Web. phpMyAdmin supports a wide range of operations on MySQL. Frequently used operations (managing databases, tables, columns, relations, indexes, users, permissions, etc) can be performed via the user interface, while you still have the ability to directly execute any SQL statement.

Git was used as a version control mechanism, while the code is setup on a VI-SEEM GitLab repository [14].

2.2.4 Core system elements

2.2.4.1 Agora module

The Agora module is the core module of the application where all the application level settings are stored and loaded. It also contains the first level of url routing and the generic and shared views for all the other modules.

2.2.4.2 Service module

The service module is the central component of the whole system where the main object type Service**,** is created, modified and managed. As every Django module in this project it consists of model definitions, controller and view functions, tests, URL mappings and admin preferences.

Models defined in this module are:

- Service
- ServiceDetails
- ExternalService
- DependsOnService
- UserCustomer
- ServiceArea

2.2.4.3 Accounts module

The accounts module manages the user accounts and contains the SAML2 integration methods for creating new unknown users, the user manager for managing existing users and the admin preferences and settings for the User objects.

2.2.4.4 Owner module

This Owner module is managing the following models:

- Institution
- ServiceOwner
- ContactInformation
- InternalContactInformation
- ExternalContactInformation
- AditionalUsernames

There are also controller actions for creating, querying and managing contact information, service owners and institutions.

2.2.4.5 Options module

This module maintains all the options for the object created from the other modules like the Service one. It houses the following models:

- ServiceDetailsOption
- SLAParameter
- Parameter

- SLA
- ServiceOption

2.2.4.6 Components module

This module houses the Service Components models and functions. The models are:

- ServiceComponent
- ServiceDetailsComponent
- ServiceComponentImplementationDetails
- ServiceComponentImplementation

There are also controller actions for creating, querying and managing service components and their implementation details.

2.2.5 Basic functionality

The aim of the system is to provide service management functionality. It provides access to a service’s catalogue or portfolio and all the corresponding information. Moreover, it enables creating, editing and deleting existing services.

The service catalogue is a customer facing list of services that are in production and provide value to the customers of the service provider. Among others, it provides also information on service options including the various SLAs available for each service.

At a high level the service catalogue is a subset of the service portfolio, both in terms of the number of services that they contain and also in terms of the number of fields or attributes each one holds.

The core system architecture is structured into several models which all encompass a given service. All of these distinct models are also accessible using the application and beside the basic CRUD operations can be linked to its appropriate linked models. Consequently, one can manage service components and their respective implementations and their details, service options and related parameters and SLAs, service owners and the necessary contact information. Also, it provides management of services and different service versions, external services etc. Users can also add foreign keys to appropriate models when adding or editing given objects.

The RESTful API provides the service information by returning JSON objects. The Figure 2 depicts a basic JSON response from the API. All responses provide a status message which is a standard response code, an info field explaining the content of the actual response and the data itself. The data field varies based on the type of object it returns.


```
{
    status: "200 OK",
    info: "service information",
  - data: {
        uuid: "995f2227-aafc-499c-8908-7fc7d30a6710",
        name: "VI-SEEM Login",
        description_external: "<p>The VI-SEEM Login service enables research communities to access VI-SEEM e-
        Infrastructure resources ...",
        service_area: "Authentication and Authorisation",
        value_to_customer: "VI-SEEM Login enables individual researchers to seamlessly and securely access VI-SEEM e-
        Infrastructure resources using federated authentication mechanisms, for example using credentials provided by
        the Identity Provider of their Home Organisation through eduGAIN ...",
        request_procedures: "<a class="btn btn-high btn-info" style="background-color:transparent;border-
        color:#2980b9;color:#2980b9;" href="https://vi-seem.eu" target="_blank">Click to Sign Up / Login to VI-SEEM
        </a>",
        service_type: "Service provider proxy",
      - service: {
            name: "VI-SEEM Login",
          - links: {
                self: "http://agora-dev.vi-seem.eu/api/v1/catalogue/services/VI-SEEM_Login"
            }
        },
      - service_details_list: {
            count: 1,
          + service_details: [...]
        },
    }
}
```


Figure 2 - JSON example response from the API

Apart from the RESTful service and admin panel, the system also provides a UI for viewing and managing catalogue and portfolio data. The system UI is generally separated based on whether it is used for viewing or managing the data. The viewing only UI part can be used as standalone, but can also be used as embedded component in the VI-SEEM website [15].

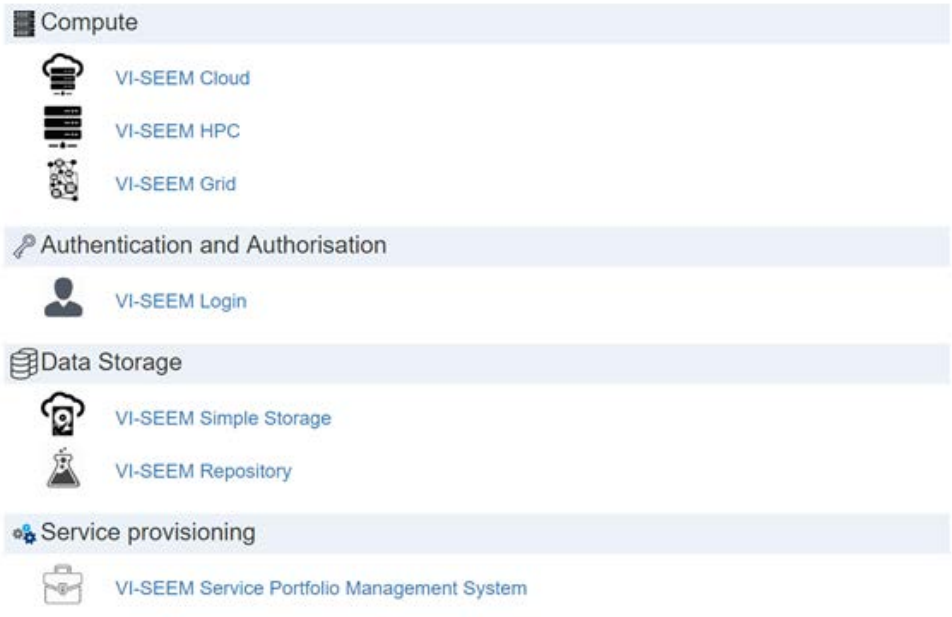


Figure 3 - Service picker, a list of service

The Figure 3 presents a list of services in the abovementioned UI. The difference between the catalogue and portfolio view is the level of detail each view presents. The details for each service can displayed, as shown in Figure 4.

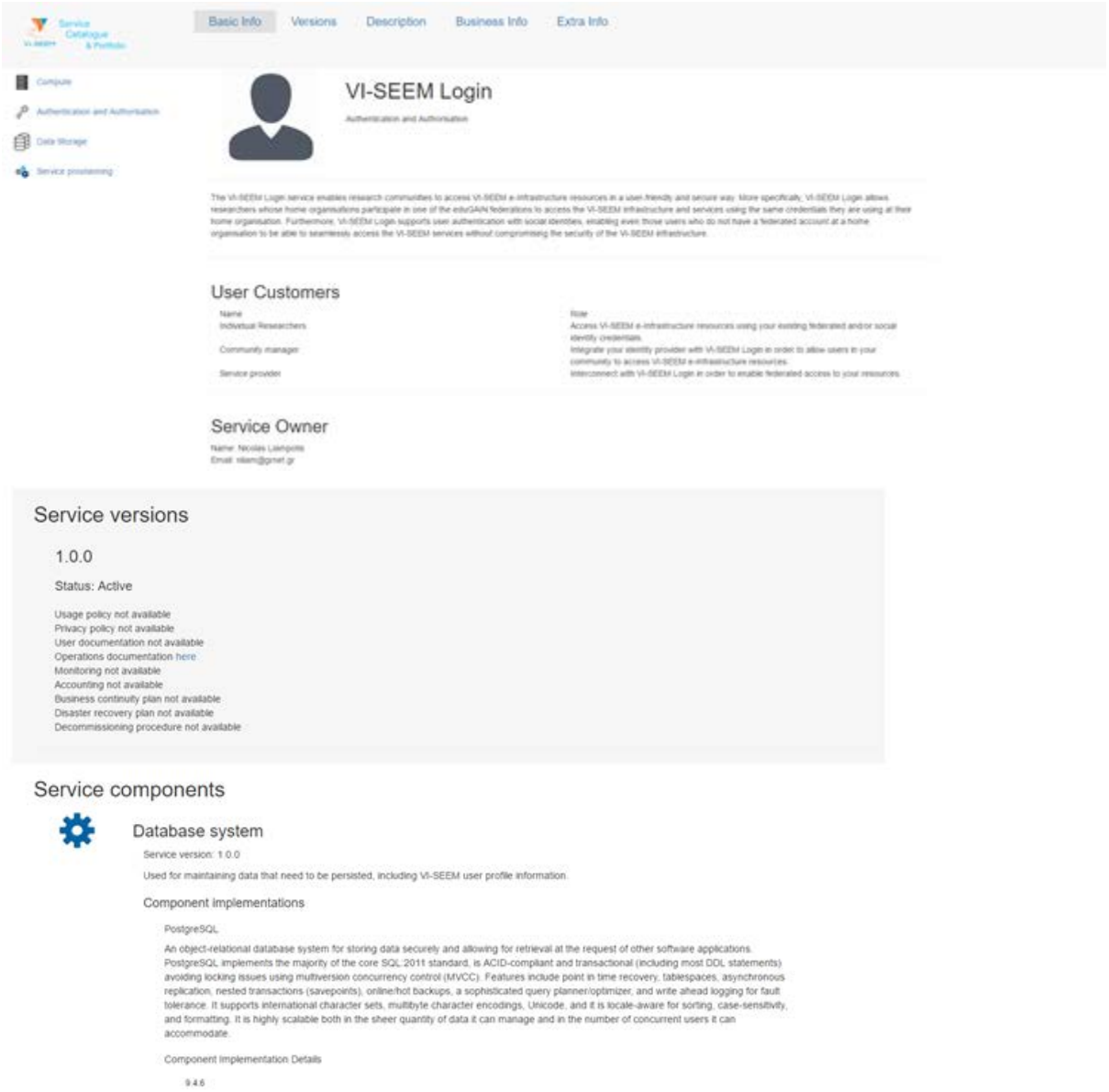
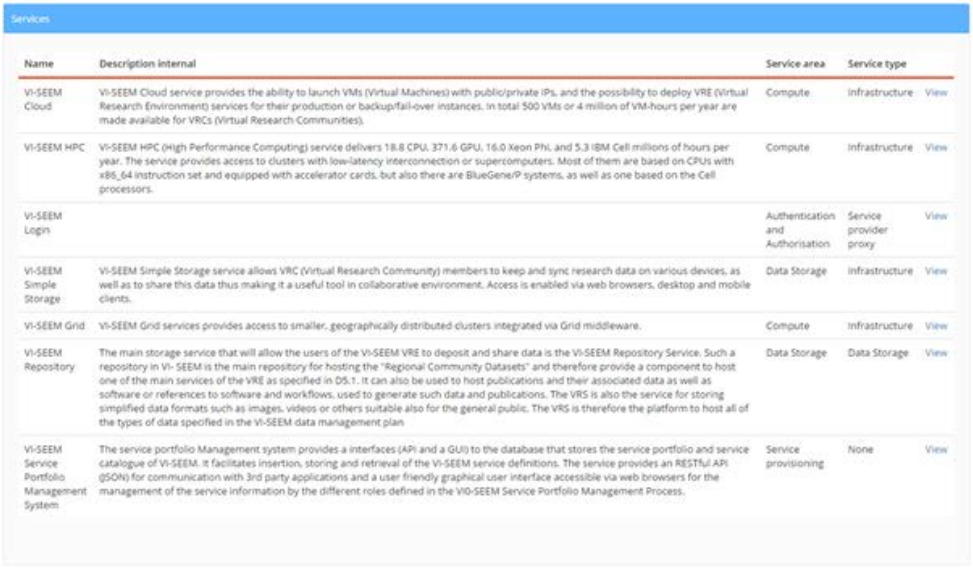


Figure 4 - Service details

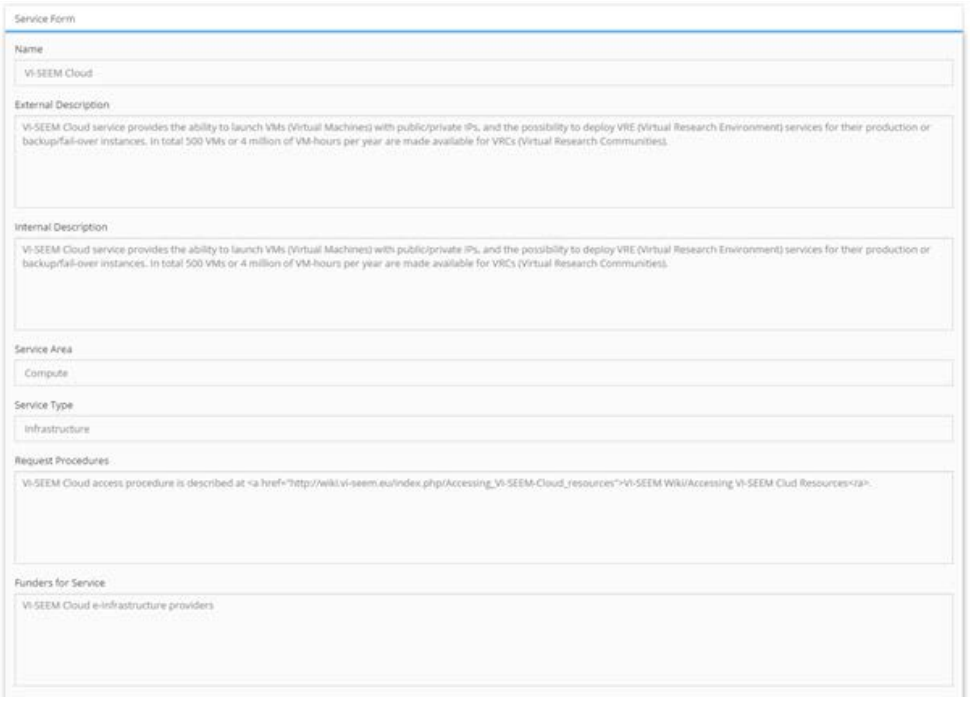
Using the write part of the user interface, authorized users can edit the services information in the system. In this part of the system, the user can view a list of all services, as shown in Figure 5, service components and service owners. He/she can also create and edit each of the existing service, Figure 6, or service components, Figure 7, for example.



Services

Name	Description internal	Service area	Service type	
VI-SEEM Cloud	VI-SEEM Cloud service provides the ability to launch VMs (Virtual Machines) with public/private IPs, and the possibility to deploy VRE (Virtual Research Environment) services for their production or backup/fail-over instances. In total 500 VMs or 4 million of VM-hours per year are made available for VRCs (Virtual Research Communities).	Compute	Infrastructure	View
VI-SEEM HPC	VI-SEEM HPC (High Performance Computing) service delivers 18.8 CPU, 371.6 GPU, 16.0 Xeon Phi, and 5.3 IBM Cell millions of hours per year. The service provides access to clusters with low-latency interconnection or supercomputers. Most of them are based on CPUs with x86_64 instruction set and equipped with accelerator cards, but also there are BlueGene/P systems, as well as one based on the Cell processors.	Compute	Infrastructure	View
VI-SEEM Login		Authentication and Authorisation	Service provider proxy	View
VI-SEEM Simple Storage	VI-SEEM Simple Storage service allows VRC (Virtual Research Community) members to keep and sync research data on various devices, as well as to share this data thus making it a useful tool in collaborative environment. Access is enabled via web browsers, desktop and mobile clients.	Data Storage	Infrastructure	View
VI-SEEM Grid	VI-SEEM Grid services provides access to smaller, geographically distributed clusters integrated via Grid middleware.	Compute	Infrastructure	View
VI-SEEM Repository	The main storage service that will allow the users of the VI-SEEM VRE to deposit and share data is the VI-SEEM Repository Service. Such a repository in VI- SEEM is the main repository for hosting the "Regional Community Datasets" and therefore provide a component to host one of the main services of the VRE as specified in D5.1. It can also be used to host publications and their associated data as well as software or references to software and workflows, used to generate such data and publications. The VRS is also the service for storing simplified data formats such as images, videos or others suitable also for the general public. The VRS is therefore the platform to host all of the types of data specified in the VI-SEEM data management plan	Data Storage	Data Storage	View
VI-SEEM Service Portfolio Management System	The service portfolio Management system provides a interfaces (API and a GUI) to the database that stores the service portfolio and service catalogue of VI-SEEM. It facilitates insertion, storing and retrieval of the VI-SEEM service definitions. The service provides an RESTful API (JSON) for communication with 3rd party applications and a user friendly graphical user interface accessible via web browsers for the management of the service information by the different roles defined in the VI0-SEEM Service Portfolio Management Process.	Service provisioning	None	View

Figure 5 - List of services for editing



Service Form

Name

VI-SEEM Cloud

External Description

VI-SEEM Cloud service provides the ability to launch VMs (Virtual Machines) with public/private IPs, and the possibility to deploy VRE (Virtual Research Environment) services for their production or backup/fail-over instances. In total 500 VMs or 4 million of VM-hours per year are made available for VRCs (Virtual Research Communities).

Internal Description

VI-SEEM Cloud service provides the ability to launch VMs (Virtual Machines) with public/private IPs, and the possibility to deploy VRE (Virtual Research Environment) services for their production or backup/fail-over instances. In total 500 VMs or 4 million of VM-hours per year are made available for VRCs (Virtual Research Communities).

Service Area

Compute

Service Type

Infrastructure

Request Procedures

VI-SEEM Cloud access procedure is described at VI-SEEM Wiki/Accessing VI-SEEM Clud Resources.

Funders for Service

VI-SEEM Cloud e-Infrastructure providers

Figure 6 - Editing a service

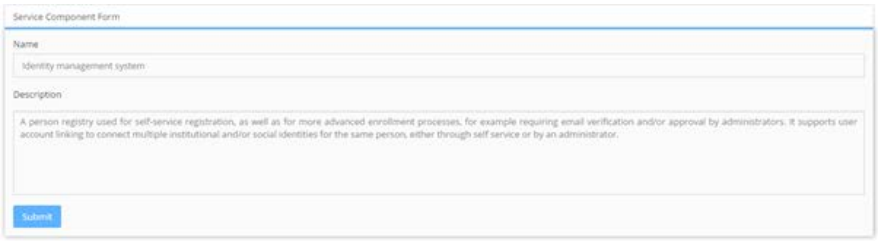


Figure 7 - Editing a service component

2.2.6 Usage and ways of access

The system can be used through the provided RESTful service. As mentioned the application is separated into 4 main components or modules. Consequently, the application’s endpoints are structured into 4 main groups. Each group provides access to the corresponding module of the system and its related models.

In order to access the service catalogue information, the user only needs to generate a GET request to the appropriate URL. Accessing the service portfolio information requires the user to be logged in with the appropriate account credentials.

External users are also enabled to generate POST requests in order to enter new or edit existing data. Both functionality require a logged in user in order for the operation to be completed.

Some sample ways of accessing the API are presented in the table below.

GET	**/api/v1/{var}/services**
Retrieves a JSON list of all services in the system. The {var} is a variable part of the URL which can take either catalogue or portfolio as values and provides the corresponding level of detail. If neither of the both values is supplied, an error message is returned.	
GET	**/api/v1/{var}/services/{service_name_or_uuid}**
Retrieves a specific service. The service can be retrieved by either supplying its name or UUID.	
GET	**/api/v1/{var}/services/{search_type}/service_details/{version}/ service_components/{comp_uuid}/service_component_implementations/ {imp_uuid}/service_component_implementation_detail**
Retrieves the list of component implementation details for the selected service component implementation.	

POST	/api/v1/services/add
Inserts a service object.	

Table 1 - Example API access

The system can also be used through the Admin panel. In this scenario, the user has full access to all aspects of the system and is additionally enabled to delete existing information in the database.

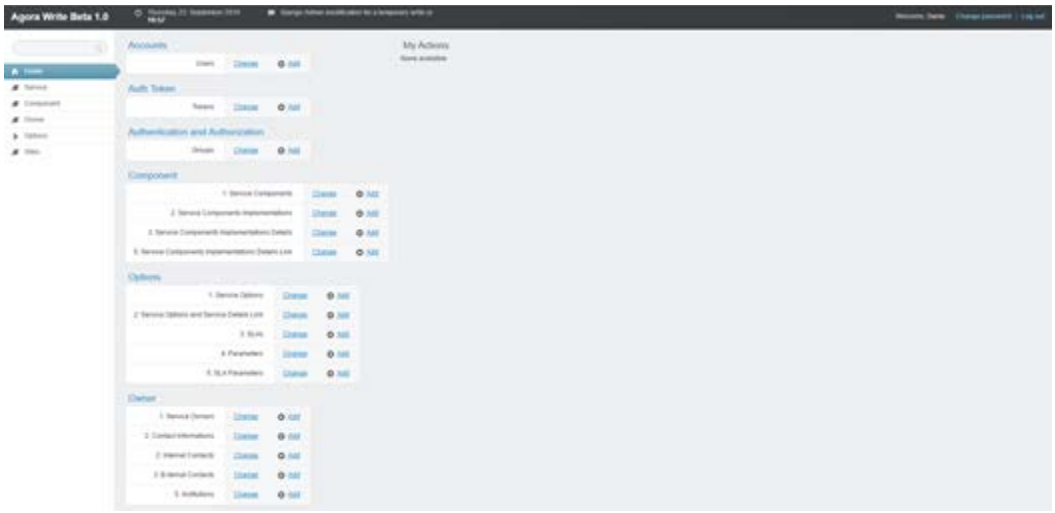


Figure 8 - Accessing the admin panel

In the admin panel, all models in the database can be listed and specific instances of the models can listed, Figure 9, added, edited and deleted, Figure 10.

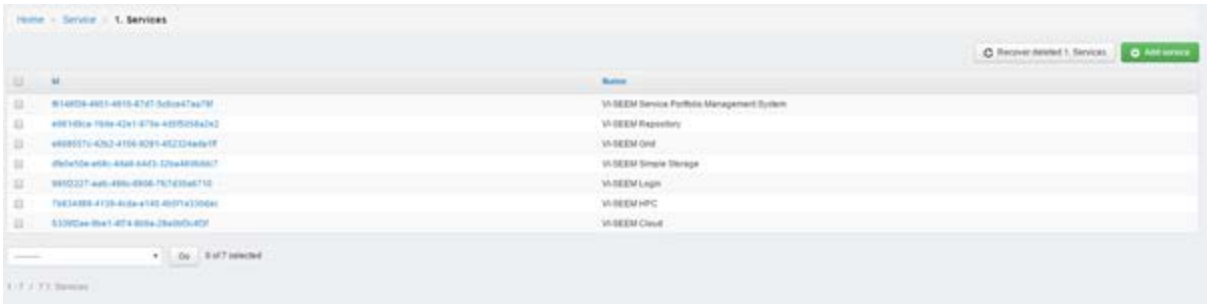


Figure 9 - Listing the database models

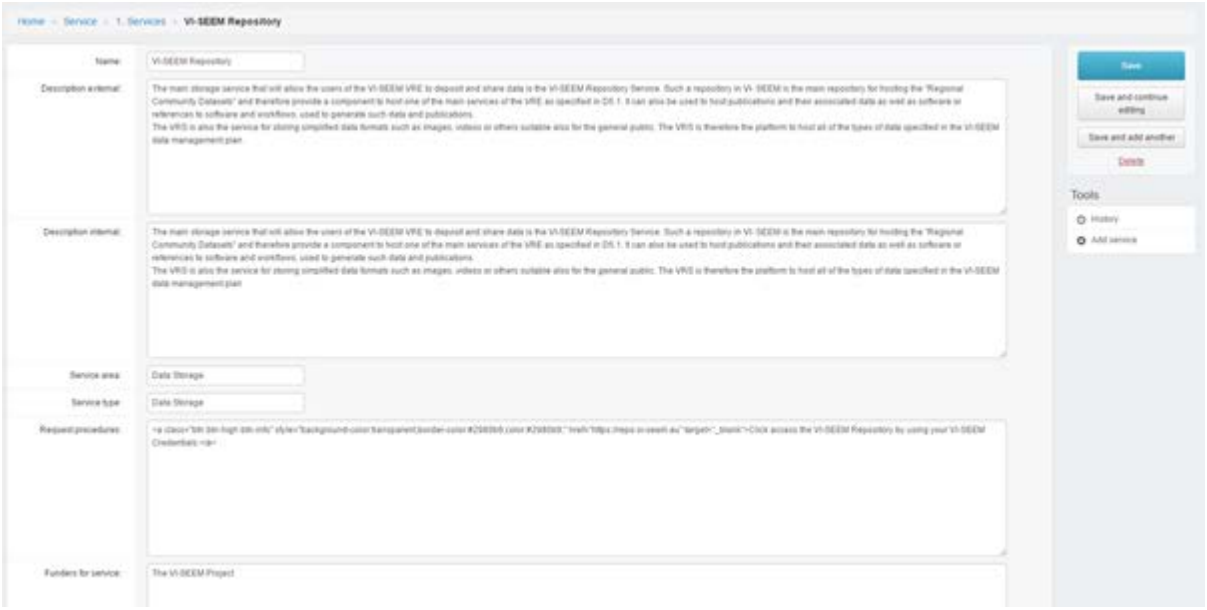


Figure 10 - Editing the database models

2.2.7 Integration

2.2.7.1 Google accounts

The system is integrated with Google Login API so users can login with Google accounts. The Google Accounts Authentication pipeline is primarily intended for the graphical user interface and not the API.

2.2.7.2 VI-SEEM AAI

The system is also integrated with the VI-SEEM AAI system implemented through SAML authentication. Every user that can authenticate through the VI-SEEM Identity Provider can also use and manage the whole application.

2.2.7.3 User interface integration

The user interface is created with ReactJs and is embeddable in other existing web pages through including the rendering script or just placing that iframe which renders the content.

3 Monitoring

In ITIL the service support area implicitly addresses “monitoring and operations” through Incident and Problem Management. Incidents in ITIL can be automated tool-generated events or they can be service calls from users experiencing service quality deterioration. Monitoring in VI-SEEM has the following main aims:

- To identify problems and malfunctions with the operation of the offered services and proactively inform the administrators of such services to provide a solution to them.
- To provide a tool to the VI-SEEM service desk operators to identify the root cause of incidents reported in the helpdesk by customers or users and work with the service administrators in resolving such issues.
- To provide to VI-SEEM service managers, scientific community leaders and service operators with appropriate service availability and reliability reports in order to assess the quality of the offered services and take corrective measures to improve it when necessary.
- To provide the end users with a quick service status report so that they can identify if a problem identified in the scientific workflow is caused by the VI-SEEM service or any other components they might be using. It also assists users to identify issues that are known to the VI-SEEM operators, thus increasing the level of trust of the user towards the infrastructure.

Monitoring of services listed in the service portfolio of an organization is an important action that facilitates both day to day operations as well as help the service provided understand the quality of service that they can offer to the customer / users.

3.1 Monitoring methodology

VI-SEEM supports multiple communities and each community will use services and service instances specific to their research. Each service end point in the VI-SEEM service portfolio is registered in the service instance registry (GOCDB). This process outlines one of the main input of the Monitoring service which is the topology of the infrastructure in VI-SEEM.

Once the services with their details (endpoints, service types) are defined in the GOCDB, the monitoring engine can identify the targets for sending test (probes) for execution. A probe is mainly service specific and when it is executed performs one or more tests (metrics) and returns a status. The monitoring system understands the following statuses: OK, WARNING, CRITICAL, UNKNOWN, MISSING, DOWNTIME.

A profile management is necessary to organize the existing metrics into groups and define actions that configure the way the monitoring engine executes service tests (checks) for specific service types (called POEM profiles). A POEM profile includes a list of Service Types and the existing metrics that are relevant for each Service Type. These metrics are being tested by the monitoring probes which are executed by the monitoring engine for the known instances of the relevant Service Types.

The monitoring engine collects the results and delivers the metric data (probe check results) to a Message Broker network (messaging). A consumer collects the metric data (probe check results) from the Message Broker Network and delivers them to the compute engine (analytics engine) in a predefined format.

Gathering information on the status of the services, in predefined time intervals, - in combination with data from sources (topology, profiles, downtimes) - the the availability and reliability of the service can be calculated on a monthly, 3 monthly or 6 monthly bases. In principle the information on status, availability and reliability and the parameters for calculating those, can be provided by multiple reports that utilise customer based profiles for the different roles within the service management framework i.e. management, operations etc.

The following are definitions of service availability and reliability:

Service availability is calculated all the time by subtracting from 100% the percentage of time within a reporting period (i.e. on a monthly basis) the service tests return results that lead to critical error results.

Service reliability is the same as service availability, with the exception that it is calculated only during service hours and predefined service breaks are excluded.

The monitoring system exposes all information about status, availability and reliability of the services over time via a RESTful API. 3rd party clients can use such information to display results in a meaningful way depending on the user of such information and his role.

3.1.1 Monitoring portal requirements

VI-SEEM supports multiple communities (the Life Sciences, the Climate and the Digital Cultural Heritage communities) and each community will use services and service instances specific to their research. Some services might be relevant for more than one community, while other services will be specific to just one community. Therefore, the monitoring system should offer views about the status, availability and reliability of the services from the point of view of a specific Scientific Community. Each scientific community should be able to monitor the services it uses on the VI-SEEM platform.

The monitoring portal will offer a set of views about the status, availability and reliability of the services from an infrastructure point of view. Typically, services are grouped in Sites (the administrative domain within which the service is operated). Access to this information should be limited to authorised VI-SEEM operations personnel.

3.2 Monitoring infrastructure

3.2.1 GOCDB as service instance registry

The configuration management database used within the VI-SEEM project is GOCDB. GOCDB is a Configuration Management Database (CMDB) for recording and managing

assets in federated environments. It defines a number of topology objects including admin-domains, sites, services, service-groups, service-endpoints, service-downtimes, users and roles. GOCDB is the central CMDB for the EGI-Engage and EUDAT2020 projects. The tool provides a web portal for editing information and a REST style programmatic interface (PI) for querying data in XML. Relationships between different objects are defined using a well constrained relational schema that closely resembles a subset of the GLUE 22 information model. A comprehensive role-based permissions model controls user permissions. Importantly, the project can self-manage their users; users make requests for roles over target objects and users that already hold the necessary role(s) can accept or reject those role requests.

An instance of the GOCDB [16] is deployed by the partner UKIM for the needs of the project - Figure 11. The instance is customized to reflect the actual needs and objects of the VI-SEEM project infrastructure and services.

GOCDB data model closely resembles GLUE 2 [17] specification in terms of the entities and their relationships. Therefore, it is expected that the structure of the data model remains largely static. The structure supports multiple projects. Service groups can also be used to group Services belonging to different Sites. Downtimes are declared over services, and registered users have roles over target objects.

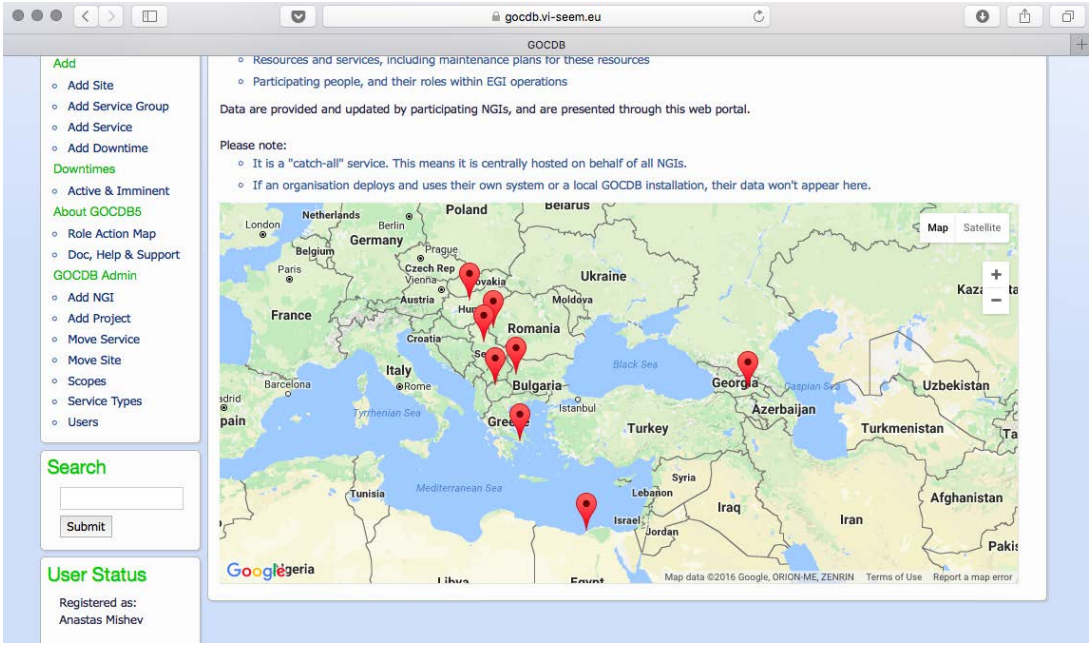


Figure 11 - VI-SEEM GOCDB

VI-SEEM instance of the GOCDB reuses this structure. VI-SEEM project is created within the Project section as a GOCDB NGI object, service and resource providers (VI-SEEM consortium partners) are listed as separate sites (partner short name is used for this), and

each provider (partner) lists deployed services. Service has endpoint object that defines a network location/address for a service, and can be linked to downtime object.

Each user account is represented as a user object. This object holds roles that defines users' permissions. Scope entity defines tag/label that can be associated to service, and in VI-SEEM GOCDB instance this feature is used to group user community specific services.

3.2.2 ARGO monitoring system

ARGO is the monitoring system being used for monitoring the VI-SEEM services status, availability and reliability. ARGO is being developed by GRNET (Greece) in collaboration with SRCE (Croatia) and CNRS (France) and is currently in use as the monitoring system of the EGI infrastructure and services, the EUDAT2020 project services and GRNET HPC Tier-1 system and CLARIN-EL [18], with specific customizations for each.

ARGO is based on a multitenant architecture where each infrastructure or project can set up their own monitoring project. VI-SEEM is being set up as a new tenant in the ARGO system owning the project “VI-SEEM Monitoring”. The ARGO monitoring system supports flexible deployment models and has a modular design enabling the integration with external systems such as the GOCDB. ARGO is based on open source components.

For status monitoring, ARGO relies on Nagios [19]. All probes developed follow the Nagios conventions and can run on any stock Nagios box. ARGO provides optional set of add-ons for the stock Nagios that provide features such as auto-configuration from external information sources, publishing results to an external messaging service etc.

For Availability & Reliability monitoring ARGO introduces a modular architecture, which relies on Nagios for service endpoint monitoring and which can ingest in the Nagios monitoring results in order to track a vast number of monitoring metrics, provide real-time notifications and status reports and monitor SLAs/OLAs.

ARGO comes in two flavors: a standalone version for deployment in low density e-Infrastructures with a limited number of services and a cluster version for deployment in high density e-Infrastructures with a large number of services.

ARGO is built following a modular architecture. At its core, ARGO uses a flexible monitoring engine (Nagios), a powerful analytics engine and a high performance web API.

Embracing a modular, pluggable architecture, ARGO can easily support a wide range of e-Infrastructures. Through the use of custom connectors, ARGO can connect to multiple external Configuration Management Databases and Service Catalogues.

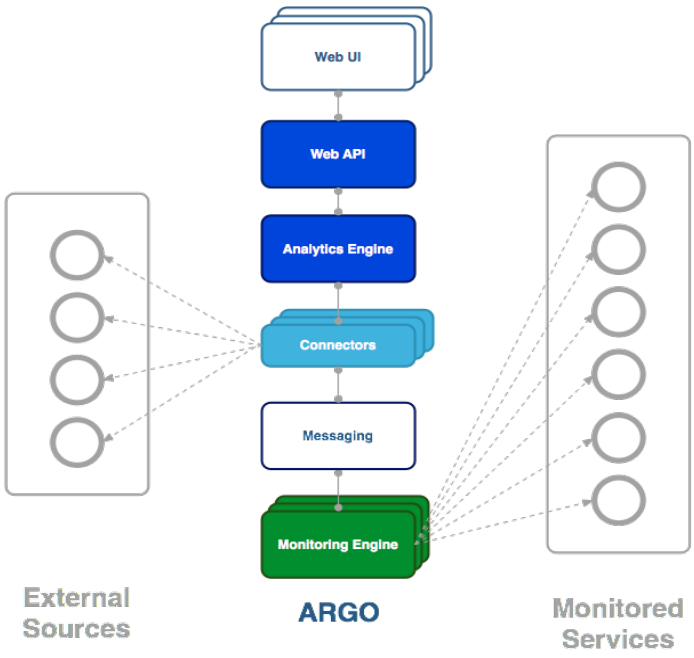


Figure 12 - ARGO components and architecture

As shown Figure 13 the ARGO service comprises of the following building blocks:

- The POEM service: This service is used in order to define checks (probes) and associate them to service types. Each grouping of checks and service types forms a profile.
- The monitoring engine: The engine executes the service checks against the infrastructure and delivers the metric data (probe check results) to a Message Broker or a Message Broker network (messaging).
- The consumer: This service collects the metric data (probe check results) from the Message Broker or the Message Broker Network and delivers them to the compute engine in a predefined format (avro encoded format).
- The connectors: This is a collection of libraries that periodically (usually once per day) connect to sources of truth (such as GOCDB for topology or downtimes, or POEM services for metric profiles etc) and deliver the information to the compute engine in a predefined format (avro encoded format).
- The compute engine: The compute engine is also known as the analytics engine. Using the metric data collected the engine is responsible for flattening out the metric results and for computing the services availability and reliability metrics. Results (status and A/R) are passed onto a fast, reliable and distributed datastore. . (Described in next section)
- The Web API: This component delivers all computed status and A/R results via a programmatic interface. (Described in next section)

The Web UI: This is the main interface for the users of VI-SEEM. It is used in order to represent the status and A/R results graphically and gives the ability to any given user to

drill down from the availability of a given resource down to the actual metric results that were recorded and contributed to the computed figures. (Described in next section)

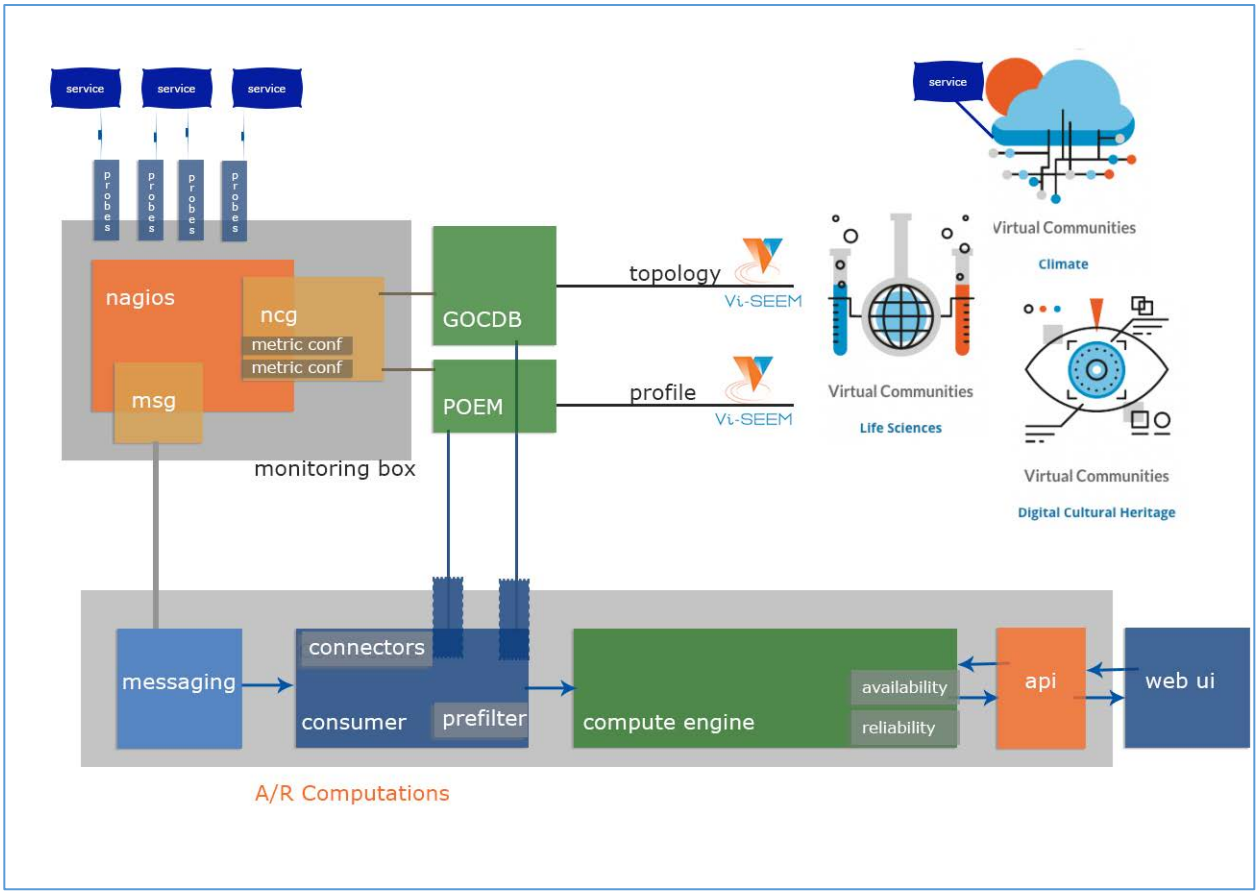


Figure 13 - The building blocks of ARGO

3.2.2.1 The ARGO Compute Engine

The Compute Engine is responsible for computing the status, availability and reliability of the monitored services.

Status computations

The Compute Engine expects to receive a stream metric results produced by a monitoring engine. A metric result is the output of a specific test that was run at a specific time against a specific service endpoint. A metric result includes at least:

- a timestamp showing when the given monitoring probe was executed
- the name of the service type (e.g. HTTPS Web Server)
- the name of the hostname on which the service is running (e.g. www.example.com)
- the name of the metric that was tested (e.g. TCP_CHECK)

- the status result that was produced by the monitoring probe (e.g. OK)

A default ARGO Compute Engine installation comes with seven statuses predefined, in alignment with the statuses produced by a Nagios compatible monitoring engine: "OK", "WARNING", "UNKNOWN", "MISSING", "CRITICAL" and "DOWNTIME". The computed service statuses generate status timelines for each service endpoint.

Availability and reliability computations

In order to understand how service availability and reliability is computed, it is important to understand what availability and reliability mean:

- Service Availability is the fraction of time a service was in the UP Period during the known interval in a given period
- Service Reliability is the ratio of the time interval a service was UP over the time interval it was supposed (scheduled) to be UP in the given period.

Before computing the availability and reliability of a given service or group, the Compute Engine computes the status timelines for that service or group by aggregating the individual status results of the lower level groups or service endpoints. How these aggregations are taking place at each group level, is defined by user defined algorithms called metric profiles 1for the status computations and availability profiles2 for the availability and reliability computations.

The Compute Engine computes hourly availability and reliability for Service Endpoints based on the individual statuses of each Service Endpoint for a specific time period. The computation is taking into account:

- the metric results that are collected from monitoring engine
- information about the topology of the infrastructure from GOCDB
- information about scheduled downtimes from GOCDB
- information about the importance of each entity in the infrastructure

3.2.2.2 ARGO Web API

ARGO exposes all information about status, availability and reliability of the services over time via a RESTful API. Follow some examples of such calls.

Get reports

[1] A metric profile defines which metrics are to be considered when computing the status of a service of a particular flavor. This defined in the POEM service.

2 Availability profiles are used to define logical rules on how to aggregate individual status computations into groups. (for example two service endpoints within the same site and of the same service flavor can be combined with an “OR” or an “AND” statement. If it is an OR statement one of them must be OK, if it is an AND both must be OK).

Provides the list of reports that a user has access to:

```
curl -k -X GET -H "Accept: application/json" -H "x-api-key: hidden"
'https://web-api-devel.argo.grnet.gr/api/v2/reports'
```

Get one report

Provides the details of the selected report.

```
curl -k -X GET -H "Accept: application/json" -H "x-api-key: hidden"
'https://web-api-devel.argo.grnet.gr/api/v2/reports/e50e7252-ca8d-43d6-9645-
63de1301e838'
```

Get project results

Lists the Availability/Reliability results for a given type of supergroup)

GET /results/{report_name}/{group_type}

GET results/Critical/PROJECT

```
curl -k -X GET -H "Accept: application/json" -H "x-api-key:
cd098725facf50c17c82216abf9d372d" 'https://web-api-
devel.argo.grnet.gr/api/v2/results/Critical/PROJECT?start_time=2016-02-
01T00:00:00Z&end_time=2016-02-02T23:59:59Z&granularity=daily'
```

3.2.2.3 VI-SEEM monitoring web UI

UoBL is currently developing the VI-SEEM monitoring web UI. One of the main requirements that such a UI tries to address is monitoring of service per scientific community. The ARGO monitoring system addresses this requirement via grouping the services per discipline via well-designed reports and integration with the GOCDB. The results of such reports are provided to the web UI in the form of JSON data. The monitoring UI then presents such status, availability and reliability data in the forms of separate pages or tabs for each scientific community.

Main components of web UI are:

- Dashboard – overview of current status and availability and reliability of services for the current month per community
- Service Status – detailed view of latest available data and status of each service and service component per community
- Availability and Reliability – detailed view of availability and reliability figures for every service and service component, including historical data and advanced search capabilities

The system consists of web server serving static assets and dynamically generated web page that represents single page JavaScript application and ARGO API Proxy service on the same domain. Monitoring UI is defined as SAML Service Provider (SP) and has support for

federated authentication via project defined Identity Provider (IdP). Users can access the system in two modes – guest and login. Guest mode provides for information overview and is publically accessible, while login mode requires successful authentication via SAML and enables users to access detailed monitoring data, depending on the attributes provided by SAML IdP. ARGO API Proxy has knowledge of ARGO API access key that is kept secret from end users so no sensitive information is transmitted. The web UI is developed as a single page JavaScript application and uses React.js library for its user interface. This architecture is presented in Figure 14.

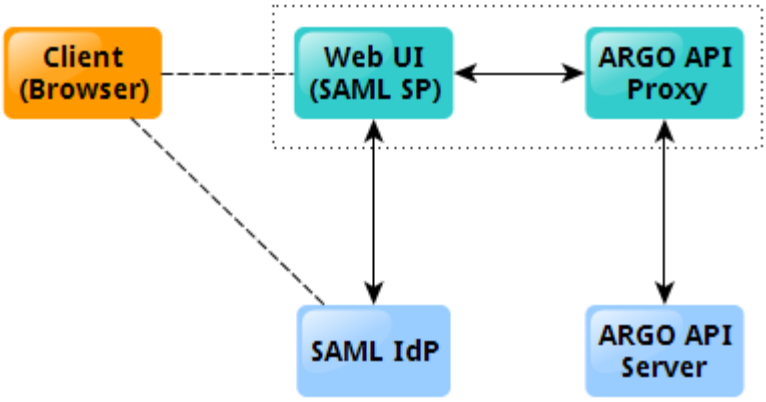


Figure 14 - VI-SEEM monitoring UI architecture

User are greeted by “Dashboard” view of monitoring UI that gives simple overview of current statuses as well as availabilities and reliabilities of services for a chosen scientific community. Every community has a separate tab in the UI. Each tab contains tabular and graphical data for each of the services tested within the community and provides data on availability and reliability as well as latest test status for a chosen service as shown in Figure 15. Upon selecting a specific service user is provided with more detailed view as described later on. Users also have a choice to access overview pages for current service status, availability and reliability figures, to log in to restricted are of UI and access various project related links.

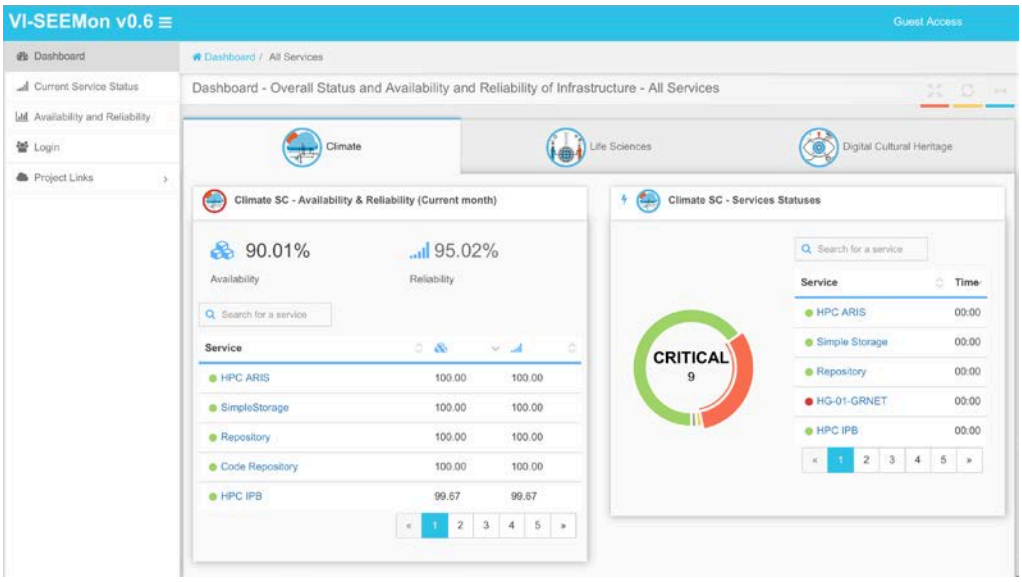
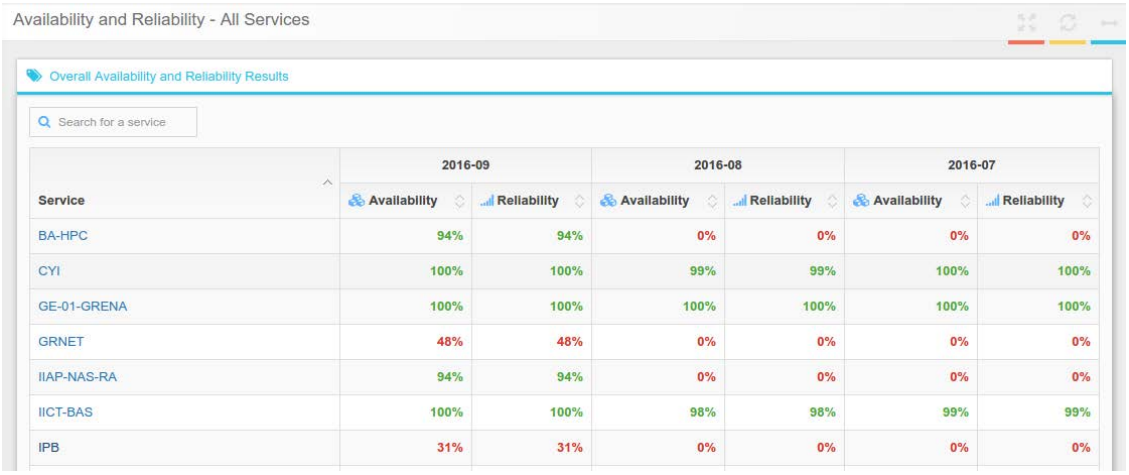


Figure 15 - VI-SEEM monitoring UI – dashboard

Users that are interested in service availability and reliability can access the overview page that provides quick access to figures for all services for last three months, as presented in Figure 16.



Service	2016-09		2016-08		2016-07	
	Availability	Reliability	Availability	Reliability	Availability	Reliability
BA-HPC	94%	94%	0%	0%	0%	0%
CYI	100%	100%	99%	99%	100%	100%
GE-01-GRENA	100%	100%	100%	100%	100%	100%
GRNET	48%	48%	0%	0%	0%	0%
IIAP-NAS-RA	94%	94%	0%	0%	0%	0%
IICT-BAS	100%	100%	98%	98%	99%	99%
IPB	31%	31%	0%	0%	0%	0%

Figure 16 - VI-SEEM monitoring UI - availability and reliability overview

Upon selecting the desired service, the user is presented with more detailed data for given service, including its service components as shown in Figure 17. This mode allows users to quickly see the data for availability, reliability, downtime and unknown statuses for the current month for both service and its components. It also provides daily breakdown for individual days of the current month. If more specific data is required, users can either move through time month at a time from the same tab or switch to “Search Over Time” tab

and specify exact time period and choose if they want monthly or daily granularity of obtained data.

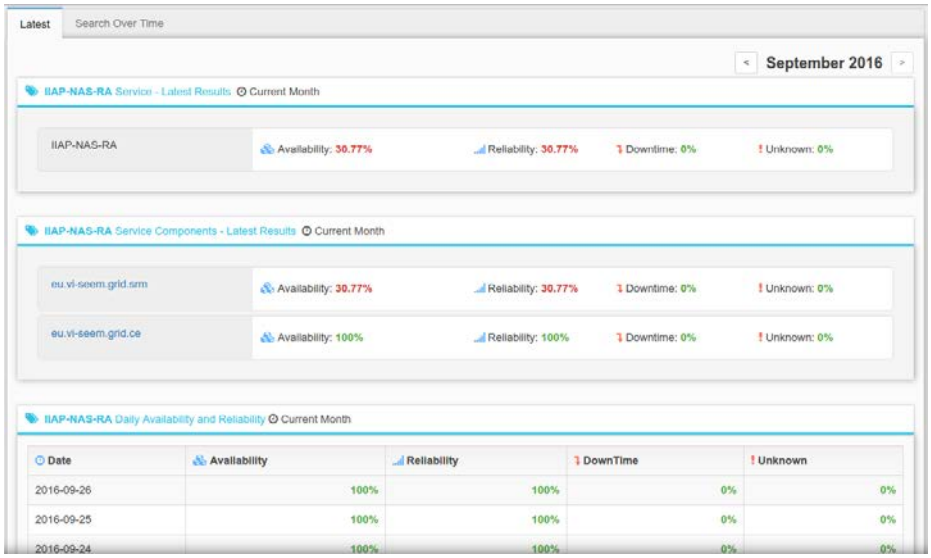


Figure 17 - VI-SEEM monitoring UI – service availability and reliability

If the users are interested in individual test statuses of each service and its component, they are provided under the section “Current Service Status”. Main screen presents a grid of services with indicator representing service’s current test status. Color-coded time line for services and components is under development and will be included in the final version.

4 Operational and service level definitions

The second FitSM requirement regarding the service management system is the PR2: Service Level Management. In order to implement it, there must be clear definitions of the operational and service levels.

4.1 Operational procedures

Aim of the VI-SEEM operational level monitoring procedure is to describe step-by-step processes for service validation, registration, certification and monitoring, as well as to define related workflow. The procedure is approved by the Project Steering Committee and it is periodically reviewed.

Services provided by the project at the higher level could be grouped into infrastructure services (access to HPC/Cloud/Grid infrastructure resources and operational tools), storage services (such as simple storage, repository, archival, data analytics, etc.), and application level services (generic, and specific such as LAS, Clowder, ChemBioServer [20]). Key players, as defined in the deliverable D1.1 [22], are: operational team [23], service enablers [24] and Scientific Community leader.

4.1.1 Service validation

Validation of infrastructure and storage services is performed by VI-SEEM operational team. This team has one representative per country, and based on geographical location of the service, corresponding representative performs validation.

Application level services are produced during the six-months integration phase jointly: by scientific community members on particular infrastructure resource with support of team that operates particular resource. Development and service provision is closely monitored by service enablers (one representative per country), and the same person performs technical validation of the service. Scientific community leader is responsible for functional validation of the application level service.

4.1.2 Service registration and certification

Successful validation of a service ends with service registration. Registration is performed by service enablers in the case of application level services, or by operational team country representative in the case of infrastructure or storage services. For service registration VI-SEEM GOCDB database is used. Beside numerous GOCDB service attributes, service endpoint and service type are crucial for the monitoring system. The endpoint gives a network location/address of the service, and service type defines set of probes that will be checking the service. Service types are predefined in GOCDB, and in the monitoring system

are linked to the real probes. Service ‘Production’ flag has to be ‘N’, and ‘Monitored’ flag ‘Y’ at this stage.

Once the service is registered, it goes through another technical validation – certification. This action is performed by Service Enabler, and it has to ensure that service is passing monitoring probes, and that the service support unit is established within the VI-SEEM ticketing support system (helpdesk). Once it is ensured, service ‘Production’ flag can be switched to ‘Y’.

4.1.3 Service monitoring

Service monitoring, performed by the project monitoring system – ARGO, is automatically triggered once the monitoring flag is set to ‘Y’ in the GOCDB. ARGO is based on Nagios monitoring system, and its probes are compatible with the Nagios probes. Therefore, infrastructure and storage specific probes are coming with the monitoring system and additional development of probes is not required. On the other hand, beside generic application level probes (these probes are already available in the ARGO system), some specific probes will be provided by the developers as well. Procedure that describes how these probes go from the developers to the monitoring system will be provided in addition to this procedure.

Stable infrastructure is necessary for the successful work of VI-SEEM service developers, and its quality is measured by the monitoring system (ARGO). In addition to this, a pro-active monitoring of VI-SEEM services will be organized in rotating shifts taken by VI-SEEM operational team. During the shift, VI-SEEM operational team member will be designated as Operator-On-Duty (OOD).

Basically, the idea is that each VI-SEEM operational team member is on shift during one week, and opens tickets in VI-SEEM Helpdesk to services from all countries which are failing ARGO tests, or have other problems manually identified. Of course, all operational team members are expected to continually monitor and provide support related to services deployed in their countries - this is their day-to-day duty, not related to OOD shifts.

Details of the organization of OOD shifts are given at the VI-SEEM wiki, and will be updated according to the available information. In order to have a written record of each shift and OOD actions taken during them, an OOD shift ticket will be created for each OOD shift in the VI-SEEM Helpdesk. Shift ticket should be created by the previous OOD for the next one at the end of each shift.

When creating a shift ticket to the next OOD, the previous one should enter brief hand-over report, i.e. list of major problems that remain to be solved, observations about some hard cases, number of newly created OOD tickets during the last week, overall number of OOD tickets still open, number of OOD tickets closed last week, etc. During the shift, OOD should update the shift ticket with all relevant information about newly created tickets, updates on operational documents on the Wiki etc. The ticket should be closed when the shift is finished, and new ticket opened for the next OOD, containing hand-over report.

4.2 Service levels

The process responsible for negotiating achievable service level agreements and ensuring that these are met, in ITIL, is called Service Level Management (SLM). SLM is responsible for ensuring that all services, processes as well as operational level agreements and agreements with external providers are appropriate to meet the agreed service level standards. Service level management monitors and reports on service levels, holds regular service reviews with customers, and identifies required improvements. The VI-SEEM project aims at defining service levels, specify the monitoring methodology towards the creation of Operational and Service Level Definitions for selected services, ensure that the data are provided in the desired registry form, and that the content remain up-to-date.

As described in the previous sections VI-SEEM monitoring based on the ARGO framework and reporting on “status”, “availability” and “reliability” of services fulfils the requirements for providing a monitoring methodology that is suitable for monitoring operational and service level agreements. The creation of probes for each type of service, monitoring profiles (POEM profiles) and the development of custom availability and reliability reports provides all the necessary technical infrastructure for monitoring service targets. This, coupled with usage of VI-SEEM helpdesk provides framework for monitoring the following service targets, suitable for providing the SLA for VI-SEEM services when necessary.

Table below shows the service targets VI-SEEM is planning to follow up for its services.

Service	Service level target	(%)
	Service availability	
	Service reliability	

Table 2 - VI-SEEM service targets

Table below shows the service level targets for service requests originating from users.

Service Desk	Service level target	Time to respond / resolve
	Maximum response time	
	Service Desk Resolution	

Table 3 - Service level targets for customer created requests

Table below gives the service level targets for incident handling.

Service Desk	Service level target	Time to respond
	Maximum response time	

Table 4 - Service level targets for incident handling

VI-SEEM will follow up those service targets from the time the service monitoring system will be fully in production and work on a recommendation for the definition of appropriate service levels and the definition of future Service Level Agreements.

5 Conclusions

The necessity for solid but flexible IT service management is one of the keystones for the service-oriented design. The specifics of the federated environment, such as the one found in VI-SEEM, imposes requirements for service management tools that cannot be met using common off-the-shelf solutions. Hence, special care must be taken for the design and the implementation of easy to use, custom solution that is tailor made for scientific communities. The main crucial points regarding the service management infrastructure development and deployment can be summarized in the following:

- Cleary defined service management processes using a standard that supports the federated environment must be used as the foundation for the IT service management solution design.
- The implemented service management solution will enable the various categories of users to access the set of services offered by VI-SEEM. While the end-users from all of the scientific communities can browse and search through the service catalogue, the management interface provides full control over the complete service portfolio of VI-SEEM.
- Both the catalogue and the portfolio are accessible via RESTful API, enabling the easy integration with other components of the Virtual Research Environment.
- The service registry, implemented as a GOCDB instance, enables the discovery of the endpoints for the monitoring system.
- The proposed community-based monitoring, specifically designed for the VI-SEEM project, enables each of the three scientific communities to have in-depth view of the community-specific services’ availability.
- The proposed infrastructure will enable VI-SEEM to identify and define service and operational level agreements specific to the needs of the users from the three scientific communities.

The efforts presented in this deliverable are strongly aligned with the service orientation approach of the overall VI-SEEM service provisioning architecture, maintaining compatibility with other EU e-Infrastructure projects and external resource providers.

