



D3.2 - 5G Experimentation portal, tools and middleware

Editor:	Christos Tranoris, University of Patras	
Deliverable nature:	Report (R)	
Dissemination level:	Public (PU)	
Date: planned actual	30/06/2018	9/7/2018
Version No. of pages	1.0	82
Keywords:	5G Portal, OSM API, VxF/NSD management	

Abstract

This document contains, since D3.1, the latest updates, current design and implementation details of the 5GinFIRE service front-end, naming the 5GinFIRE portal, its underlying services and the support middleware. The work of this report is contributed by tasks: Task 3.1 Experimentation Portal and Application composer toolkit Integration; Task 3.2 - 5GinFIRE middleware, model transformations and code generation; Task 3.3 - FIRE Integration: AAI and RSPEC, Task 3.4 - VxF Open repository and Task 3.5 - Integration and support of open call tooling services

Disclaimer

This document contains material, which is the copyright of certain 5GINFIRE consortium parties, and may not be reproduced or copied without permission.

All 5GINFIRE consortium parties have agreed to full publication of this document.

Neither the 5GINFIRE consortium as a whole, nor a certain part of the 5GINFIRE consortium, warrant that the information contained in this document is capable of use, nor that use of the information is free from risk, accepting no liability for loss or damage suffered by any person using this information.

This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 732497. This publication reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains.



Impressum

Full project title: Evolving FIRE into a 5G-Oriented Experimental Playground for Vertical Industries

Short project title: 5GINFIRE

Number and title of work-package: WP3 Experimentation Architect Tooling

Number and title of task:

- Task 3.1 - Experimentation Portal and Application composer toolkit Integration
- Task 3.2 - 5GinFIRE middleware, model transformations and code generation
- Task 3.3 - FIRE Integration: AAI and RSPEC
- Task 3.4 - VxF Open repository
- Task 3.5 - Integration and support of open call tooling services

Document title: D3.2- 5G Experimentation portal, tools and middleware

Editor: Christos Tranoris, University of Patras

Work-package leader: Christos Tranoris, University of Patras

Copyright notice

2017-2018 University of Patras and members of the 5GINFIRE consortium

Executive summary

5GinFIRE adopts the ETSI architectural recommendation for services that are being considered to be provided on a NFV-enabled network infrastructure, and implemented under the NFV model, augmented for application/service composition and experimentation capabilities. In order to instantiate experimentation scenarios end-users will use the 5GinFIRE portal as well as tools to design/deploy their experiments. The 5GinFIRE middleware and portal are responsible for multiple services, like:

- Offer an entry point where experimentation requests will be accepted
- A web portal that allows users to subscribe, manage experiments, browse VxF repository, monitor experiment results, etc
- Access to the 5GinFIRE repository of VxFs metadata and templates, categorized in verticals
- Services that will allow admins and developers, using the DevOps paradigm, to manage the offered 5GinFIRE platform as well as to manage the repository.
- Support, not only for experimenters, but also for VxF developers: Users that want to maintain and offer their VxFs through our 5GinFIRE repository.
- Authentication Authorization Infrastructure (AAI), compatible with other FIRE testbeds via the Fed4FIRE AAI technology, thus accepting seamlessly FIRE users, allowing the creation of federated experiments, and facilitating integration of existing FIRE facilities.
- Visibility of the 5GinFIRE repository as an RSPEC [1], therefore to have all our infrastructure browsable by other FIRE catalogues like the Fed4FIRE portal.
- Model-to-model transformations that will provide the ability to automate the transformation of the experimentation and service requests into actions to be performed by the *Services and Management orchestrators* of the MANO layer.

This document contains the current design and implementation details of the 5GinFIRE services, naming the 5GinFIRE portal, its underlying services and the support middleware. It also provides designs for future implementations that will be reflected in a follow-up updates of this deliverable. The work here is reflected by the following Tasks of WP3:

- Task 3.1 Experimentation Portal and Application composer toolkit Integration: which has the sole purpose of creating the 5GinFIRE portal and the integration with other middleware services like OSM.
- Task 3.2 - 5GinFIRE middleware, model transformations and code generation: responsible of maintaining the underlying middleware services that accept VxF and Network Service Descriptor (NSD) packages and transforms them to orchestration artifacts.
- Task 3.3 - FIRE Integration: AAI and RSPEC: which will study and implement mechanisms for simple integrations with FIRE facilities.
- Task 3.4 - VxF Open repository: which studies and integrates repositories and catalogs.
- Task 3.5 - Integration and support of open call tooling services: which is focused on integrating, planning, collaborating and supporting new tooling services that are provided by open call proposals.

Section 2 of this report presents design and implementation details of the 5GinFIRE portal. Section 3 presents details about the Middleware tools and services utilized in 5GinFIRE. Section 4 presents various details about repositories and catalogues. Section 5 provides initial design details about the FIRE integration and section 6 presents the Automated Validation process. Section 7 provides the conclusions and the future work.

List of authors

Company	Author
UNIVERSITY OF PATRAS	Christos Tranoris, Apostolos Palladinos, Ioannis Chatzis
INSTITUTO DE TELECOMUNICAÇÕES	Diogo Gomes, Eduardo Sousa
UNIVERSIDAD CARLOS III DE MADRID	Borja Nogales, Iván Vidal
EASY GLOBAL MARKET	Mengxuan Zhao

Table of Contents

1	Introduction.....	13
1.1	Objective of this document.....	13
1.2	Structure of this report.....	13
2	The 5GinFIRE portal.....	14
2.1	Introduction.....	14
2.2	Requirements.....	14
2.2.1	Requirements and the supported actors	14
2.2.2	High level processes	18
2.3	Architecture.....	22
2.4	Design and implementation.....	24
2.4.1	The Portal API backend	24
2.4.2	The web frontend	31
2.5	Requirements Implementation.....	32
2.5.1	Public user Web interface/Landing page	32
2.5.2	VxF developer user interface description	34
2.5.3	Experimenter user interface description	37
2.5.4	Services administrator user interface description	41
2.6	Deployment details.....	48
2.7	Delivery plan.....	50
2.8	Code repositories.....	50
2.9	Licensing.....	51
3	Middleware tools.....	52
3.1	Admin tool and provisioning: Launchpad (OSM TWO/FOUR).....	52
3.2	YANG models and packaging for VxF/NSD.....	55
3.2.1	Overview of the YANG models	55
3.2.2	Requirements for VNF & NS descriptors	55
4	Repositories.....	58
4.1	Portal repository.....	58
4.2	The OSM Catalogues.....	58
4.3	Utilizing the OSM API.....	59
4.3.1	OSM TWO	59
4.3.2	OSM FOUR and NBI	60
4.4	Keystone repository.....	60
4.4.1	Introduction	60

4.4.2	Requirements	60
4.4.3	Keystone deployment	61
4.4.4	Use cases	61
4.4.5	Process flows	62
4.4.6	Integration with 5GinFIRE portal	72
4.4.7	Project status	72
5	FIRE Integration	73
5.1	integration with Fed4FIRE Identity provider	73
5.1.1	Listing resources	74
5.1.2	Slice request	75
6	Automated Validation process	77
6.1	Synergy with EU H2020 project 5GTANGO for Validation	78
7	Future enhancements/Next versions	80
8	References	81

List of figures and tables

Figure 1 upload a VxF archive. Onboarding and make published by administrator	19
Figure 2 Upload a VxF archive, Triggering automated validation.....	19
Figure 3 Uploading an Experiment/NSD archive and onboarding by administrator.....	20
Figure 4 Upload a NSD archive, Triggering automated validation.....	21
Figure 5 Request a new experiment deployment.....	21
Figure 6 Interaction between portal and OSM to automatically instantiate an NSD	22
Figure 7 5GinFIRE portal architecture	23
Figure 8 Web frontend architecture.....	24
Figure 9 Main package diagram	25
Figure 10 details for the definitions of VxF and Experiment metadata.....	26
Figure 11 Core model class diagram	27
Figure 12 Model of a deployment descriptor	27
Figure 13 Diagram of the class PortalRepositoryAPI which implements the RESTful API	28
Figure 14 Web Front end components.....	32
Figure 15 The 5GinFIRE portal landing page.....	33
Figure 16 Sign up to portal.....	33
Figure 17 Available experiments, VxF developer login page. (Use case #2010).....	34
Figure 18 Available VxFs. (Use case #2070 or #3030).....	35
Figure 19 VxFs Management. (Use case #3020)	35
Figure 20 VxF upload. (Use case #3010 or #2040)	36
Figure 21 VxF creation. (Use case #3010)	36
Figure 22 VxF details. (Use case #3030).....	37
Figure 23 Available experiments, experimenter login page. (Use case #2010).....	38
Figure 24 Available Deployed experiments (Use case #2010).....	38
Figure 25 Experiment deployment creation (Use case #2020).....	39
Figure 26 Experiments management (Use case #2050).....	40
Figure 27 Experiment creation (Use case #2060).....	40
Figure 28 Users management (Use case #5010).....	41
Figure 29 User details. (Use case #5010)	42
Figure 30 User account creation. (Use case #5010).....	42
Figure 31 Admin tab submenu	43
Figure 32 Extra fields for admin role about registered experiments (Use case #4010 and #5030).....	43
Figure 33 Extra fields for admin role about registered VxFs. (Use case #3020).....	43
Figure 34 Deploy a validated experiment (Use case #5030)	44
Figure 35 Available categories	45
Figure 36 Category creation.....	45
Figure 37 Available MANO platforms	46
Figure 38 MANO platform creation	46
Figure 39 Available MANO providers	47
Figure 40 MANO provider creation	47
Figure 41 Available target Infrastructures	48
Figure 42 Deployment diagram of 5GinFIRE	49
Figure 43 Internal mappings of deployed containers.....	49
Figure 44 OSM Release FOUR Northbound API.....	52
Figure 45 Launchpad: Dashboard	53
Figure 46 Launchpad: Compute Topology.....	53
Figure 47 Launchpad: Instantiate	54
Figure 48 Deployment status of a NS in Release FOUR.....	55
Figure 49 Composer	58
Figure 50 Component interconnection diagram	63
Figure 51 Create a user process flow	63
Figure 52 Create a project process flow.....	64
Figure 53 Create a role process flow	65
Figure 54 Add a role to a user in a project process flow	67

Figure 55 Retrieve user’s roles in a project process flow..... 69
Figure 56 Authenticate a user process flow 71
Figure 57 portal and Fed4FIRE+ integration 73

Table 1 5GinFIRE portal high level usage scenarios/requirements 14
Table 2 Detailed requirements for the Web front end component 15
Table 3 Public API (does not need authentication) 29
Table 4 Admin API (needs authentication)..... 30
Table 5 Deliver plan for Version 1 50
Table 6 Code repositories related with the portal..... 50
Table 7: requirements for NFV descriptors (complete list in [10]and [11]) 56
Table 7 Use cases supported by the Keystone repository 62
Table 8 User roles..... 72

Abbreviations

5G	5th Generation
AGPL	GNU Affero General Public License
AM	Aggregate Manager
API	Application Programming Interface
BSS	Business Support System
CI/CD	continuous integration/continuous deployment
DCU	Data Collection Unit
DSRC	Dedicated Short Range Communication
ETSI	European Telecommunications Standards Institute
EVI	Experimental Vertical Instance
GPL	GNU General Public License
HA	High Availability
HOT	Heat Orchestration Template
IF-MAP	Interface for Metadata Access Points
IFA	Interfaces and Architecture (IFA) Working Group of ETSI
ISG NFV	Industry Specification Group for Network Functions Virtualization
KVM	Kernel-based Virtual Machine
MAAS	Metal as A Service
MANO	Management and orchestration
NBI	Northbound Interface
NETCONF	Network Configuration Protocol
NF	Network Function
NS	Network Service
NFV	Network Function Virtualization
NFVI	NFV Infrastructure

NFVO	NFV Orchestrator
NoSQL	Not only SQL
NSD	Network Service Descriptor
NSO	Network Service Orchestrator
OBU	On Board Unit
OP-NFV	Open Platform for NFV
OS	Operating System
OSGi	Open Services Gateway initiative
OSM	Open Source Mano
OSS	Operations Support System
OVSDB	Open vSwitch Database Management Protocol
PNF	Physical Network Function
PXE	Preboot Execution Environment
REST	Representational state transfer
RO	Resource orchestrator
RSPEC	Resource Specification
RSU	Road Side Unit
SBI	Southbound interfaces
SDN	Software Defined Network
SFA	Statefull Forwarding Abstraction
SFC	Service Function Chaining
TOSCA	Topology and Orchestration Specification for Cloud Applications
VIM	Virtual Infrastructure Management
VL	Virtual Link
VLD	Virtual Link Descriptor
VNF	Virtual network function

VNFC	Virtual Network Function Component
VNFD	Virtual Network Function Descriptor
VNFFG	Virtual Network Function Forwarding Graph
VNFM	VNF Manager
VXLAN	Virtual Extensible LAN

Changes from D3.1

The following table contains changes from D3.1 to reflect the latest updates about Experimentation, portal, tools and 5GinFIRE middleware:

Section	Description
2.2.1.1	Added new requirements for automation
2.2.1.2	Added requirements for issue management and operations support
2.2.2.2	Added Uploading and validating a VxF (automated process)
2.2.2.4	Added Uploading an Experiment Descriptor/NSD (Automated)
2.2.2.5	Added details on how the portal will support automatically creating an Issue in our issue management system
2.2.2.6	Added Automated NSD Job Instantiation
2.3	Architecture is revised to support automation processes and continuous evolution of OSM and the infrastructure in terms of APIs
2.4	Design updated to reflect support automation processes and continuous evolution of the infrastructure
2.5	Updates to Requirements Implementation
2.8	Code repositories updated
3.1	Section updated to reflect OSM FOUR aspects like the NBI
3.2.3	Removed, since validation is implemented now
4.3	This is reflected in NBI of OSM FOUR section 4.3.2
4.4.3	Added details for Keystone deployment
5	Fully revised section 5 to reflect FIRE Integration via the SFA Wrapper
6	New section describing the Automated Validation process via a Continuous Integration service as well as synergy with 5GTANGO project regarding validation

1 Introduction

1.1 Objective of this document

This document contains the current design and implementation details of the 5GinFIRE service front-end, naming the 5GinFIRE portal, its underlying services and the support middleware. It also provides designs for future implementations that will be reflected in a follow-up updates (D3.3) of this deliverable. The work here is reflected by the following Tasks of WP3:

- Task 3.1 - Experimentation Portal and Application composer toolkit Integration: which has the sole purpose of creating the 5GinFIRE portal and the integration with other middleware services like OSM.
- Task 3.2 - 5GinFIRE middleware, model transformations and code generation: responsible of maintaining the underlying middleware services that accept VxF and NSD packages and transforms them to orchestration artefacts.
- Task 3.3 - FIRE Integration: AAI and RSPEC: which will study and implement mechanisms for simple integrations with FIRE facilities.
- Task 3.4 - VxF Open repository: which studies and integrates repositories and catalogues.
- Task 3.5 - Integration and support of open call tooling services: which is focused on integrating, planning, collaborating and supporting new tooling services that are provided by open call proposals.

1.2 Structure of this report

The report is organized as follows: Section 2 presents design and implementation details of the 5GinFIRE portal. Section 3 presents details about the Middleware tools and services utilized in 5GinFIRE. Section 4 presents various details about repositories and catalogues. Section 5 provides initial design details about the FIRE integration and section 6 presents the Automated Validation process. We finally conclude in Section 7.

2 The 5GinFIRE portal

2.1 Introduction

This section presents details about the 5GinFIRE portal. We start by briefly revisiting the requirements and the architecture already set-up in D2.1 with updates. We follow up with design details about the portal and implementation details of the requirements. Next, we provide details of how the portal is deployed, the delivery plan as well as the organization of the open source repositories and code licensing. We close this section about future enhancements and Next versions planning.

2.2 Requirements

2.2.1 Requirements and the supported actors

The requirements and the supported actors by the 5GinFIRE portal were presented in D2.1 Section 4. For the sake of completeness it is also reproduced here:

- **Experimenter:** Manages its Experiments in terms of NSDs and request the deployment of an experiment over the 5GinFIRE infrastructure.
- **VxF Developer:** Manages its VxF archives.
- **Testbed provider:** Manages the registration of its infrastructure.
- **Services administrator:** is the responsible for the portal management.

The portal, as well as the underlying 5GinFIRE services like the MANO stack, needs to support certain functions of the 5GinFIRE experimentation workflow presented in D2.1 Section 4.3. Here is a list of the high-level usage scenarios/requirements to be implemented by the portal. For details please see D2.1 Section 4.4. There is also a latest public available access list of these requirements at[2]:

Table 1 5GinFIRE portal high level usage scenarios/requirements

No	Actor	Title
#1001	Anonymous User	Signup, login to Portal
#2010	Experimenter	Browse available Vxfs, NSDs and experiments in portal
#2020	Experimenter	Define experiment
#2030	Experimenter / Testbed provider	Description and availability of experimentation resources
#2040	Experimenter	Upload an experiment description to portal
#2050	Experimenter	Management of network services
#2060	Experimenter	Submit an experiment for validation
#2070	Experimenter	Search for Vxfs
#3010	VxF developer	Register a VxF definition

#3020	VxF developer	Management of VxFs
#3030	Experimenter	Search for VxFs
#4010	System Administrators	Deployment support of a validated experiment
#4020	Experimenter / Testbed provider	Description and availability of experimentation resources
#5010	Service Administrator	Management of user accounts
#5020	Service Administrator	Validation of an experiment
#5030	Service Administrator	Deploy a validated experiment

The above high level requirements were broken down into detailed issues in our code repository (see details for code repositories at Section 2.8). As presented in detail in Section 2.3, there are two main components: The Web front-end and a backend portal API.

Here is a snapshot list of the issues for the Web front-end that track our progress. A latest snapshot can be found at [1].

Table 2 Detailed requirements for the Web front end component

No	Title	Description
1	A VxF should have a property Public	If true then the VxF will be public. Only Admin roles can manage this state. By default = false
2	A VxF should have a property Certified	A VxF should have a property Validated. Only Admin roles can manage this state. By default = false
3	A VxF should have a property list of supported MANO stacks	e.g. OSM TWO
4	A VxF should have a property of packaging format	Just a string to type e.g. OSM TWO model based packaging, TOSCA(CSAR), etc.
5	Admin should manage supported packaging formats	
6	Admin should manage supported MANO stacks	
7	Admin should manage target MANO endpoints	(Add, update, delete) name, API endpoint credentials etc.
8	Allow an experimenter to create a new	The experiment will contain a name, a description and optionally a logo. It can be also added to a

	experiment description	<p>category.</p> <p>The experiment will also contain a packaged file in tar.gz containing all relevant artifacts, e.g. NSD YAML descriptor, etc.</p> <p>It can also contain 1) indication of the network services that will be instantiated during the experiment; 2) description of the testbeds, facilities and resources required for the experiment; etc.</p> <p>The experimenter will select the uploaded artifact type (e.g. OSM YAML, TOSCA, etc.) The experiment descriptor will be submitted for validation</p>
9	An admin can validate or invalidate an experiment descriptor	Initially an experiment validation status is UNKNOWN but the admin can change it either to VALID or INVALID. There should be also a reason for being invalid
10	An admin can make an experiment descriptor status as PUBLIC or PRIVATE	<p>PRIVATE is default for an experiment descriptor</p> <p>If PUBLIC it means that it can be available in the repository to be used by any user</p> <p>If PRIVATE it can only be used by its owner (e.g. experimenter)</p>
11	User signup expressing role of interest	The user can express the interest in such a role or roles (experimenter, VxF developer, testbed provider, services administrator.)
12	User can submit an experiment request based on a validated experiment descriptor	<p>The user selects one of his private or public experiment descriptors in order to deploy it.</p> <p>A deployment request can include time requirements (e.g. list of proposed dates and duration of the experiment);</p>
13	The portal can contain a list of available experimentation resources (testbeds)	This implies also management of these entities
14	An admin can make an experiment request as valid and ready to be deployed	The response will contain also details about actual execution dates and any other needed details about the experiment
15	Allow VxF developers to create and upload a VxF description	<p>This allows to VxF developers to deploy a VxF descriptor package and define some metadata (name, logo, description, target infrastructures etc).</p> <p>The VxF initially is NOT Published and NOT CERTIFIED</p>
16	A user can be assigned with multiple roles	Blocked by

	by admin	https://github.com/5GinFIRE/eu.5ginfire.portal.api/issues/11
17	Allow an Admin to on-board a VNF package to a MANO provider	This is blocked by API issue: https://github.com/5GinFIRE/eu.5ginfire.portal.api/issues/4
18	VxF metadata will include vendor name	
19	Add a field for terms of use	
20	Display descriptor of VxF	
21	Display descriptor of NSD	
22	Allow a user to create a new VxF via uploading a package	
23	Allow a user to create a new NSD via uploading a package	
24	VxF, NSD name, version and fields after reading a package should be read only	
25	Allow authentication via keystone service	

2.2.1.1 Requirements for automation

Portal and its services can support various operations and processes driven by the supported roles. For example, users can register new VNFs and NSDs and administrators should go through the process of manually onboarding them on OSM MANO or Services administrators should trigger orchestrations on specific dates. Thus, there is a need to automate some processes in order to better support users and the service experience. The following requirements are considered:

No	Title	Description
1	Support for a VNF automated validation by a 3 rd service	When a new VNF is submitted to the portal, the archive is submitted to a Continuous Integration service for validation (Description Types, Compliance, etc)
2	Support for a VNF automated onboarding	If a VNF is automatically validated it can be also automatically Onboarded to OSM
3	Support for a NSD automated validation by a 3 rd service	When a new NSD is submitted to the portal, the archive is submitted to a Continuous Integration

		service for validation
4	Support for a NSD automated onboarding	If a NSD is automatically validated it can be also automatically Onboarded to OSM
5	Support for automated NSD instantiation	The automated orchestration will be triggered from the portal service via the OSM API at the requested deployment date

2.2.1.2 Requirements for issue management and operations support

Various process need to notify automatically the ticketing/issue management system, which is maintained in WP6 and will be also described in D6.1. The following requirements are considered:

No	Title	Description
1	Connectivity to Issue management system	Implement configuration for enabling service connectivity to 5GinFIRE Issue management system (Bugzilla)
2	Automated notification to issue management system when error exist in VNF onboarding	When an error occurred on VNF onboarding an issue should be raised
3	Automated notification to issue management system when error exist in NSD onboarding	When an error occurred on NSD onboarding an issue should be raised
4	A new deployment request should create an issue for tracking the deployment	When there is a new deployment request from an experimenter, an issue should be created in our issue tracking system to track the progress of the deployment

2.2.2 High level processes

We have identified the following high level processes that the portal should support. This also comes to support activities of the 5GinFIRE experimentation workflow described at D2.1 - Section 4.3.

2.2.2.1 Uploading and validating a VxF (manual process)

During this process (see Figure 1) the following occurs:

- A VxF developer submits a VxF archive.
- The administrator can manage the VxF (e.g. edit it).
- The administrator On-Boards the VxF to the target MANO.

- The administrator can optionally mark the VxF:
 - As public in order to be publicly visible by all portal users.
 - As certified which means this is certified by a certain entity.

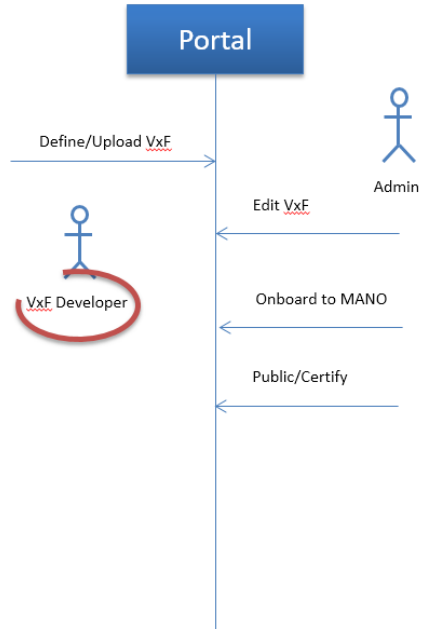


Figure 1 Uploading and validating a VxF (manual process)

2.2.2.2 Uploading and validating a VxF (automated process)

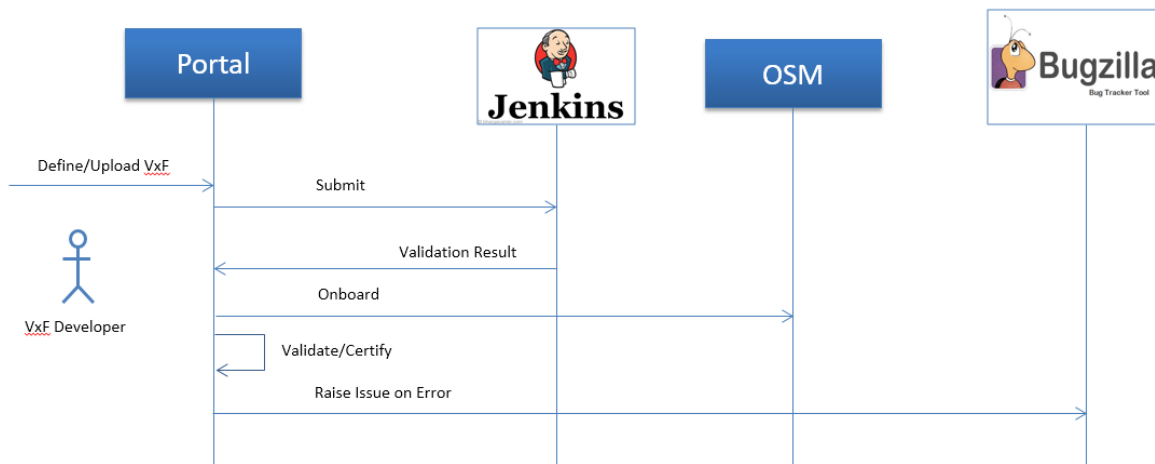


Figure 2 Uploading and validating a VxF (automated process)

As Figure 2 displays, when a VxF developer submits a VxF archive there is an automated process of

- Submitting the archive to our Continuous Integration service.
- Get the validation results.

- If validation is successful, the archive is automatically onboarded to OSM.
- If onboarding was successful, the VxF is automatically marked as Validated/Certified in the repository.
- If there was an error in the process an issue is automatically raised in our tracking system.

2.2.2.3 Uploading and Validating an Experiment Descriptor/NSD (manual process)

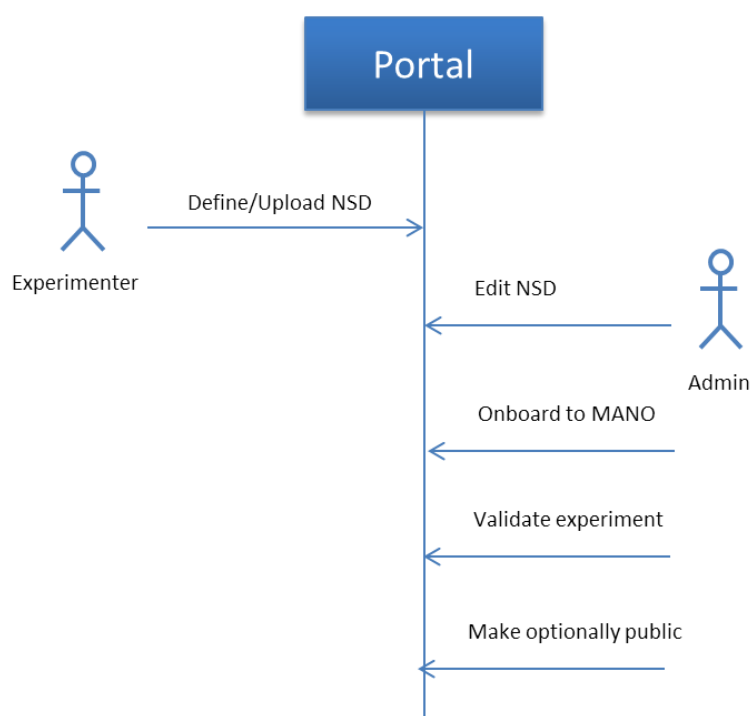


Figure 3 Uploading and Validating an Experiment Descriptor/NSD (manual process)

During this process (see Figure 3) the following occurs:

- An experimenter submits an experiment in terms of an NSD.
- The administrator can manage the NSD (e.g. edit it).
- The administrator on-boards the NSD to the target MANO.
- The administrator can optionally mark the VxF:
 - As valid, which means this NSD can be indeed deployed to VIMs.
 - As public in order to be publicly visible by all portal users.

2.2.2.4 Uploading an Experiment Descriptor/NSD (Automated)

During this process (see Figure 4) , when an Experimenter submits a NSD archive there is an automated process of:

- Submitting the archive to our Continuous Integration service
- Get the validation results
- If validation is successful, the NSD archive is automatically onboarded to OSM
- If onboarding was successful, the NSD is automatically marked as Validated/Certified in the repository
- If there was an error in the process an Issue is automatically raised in our tracking system

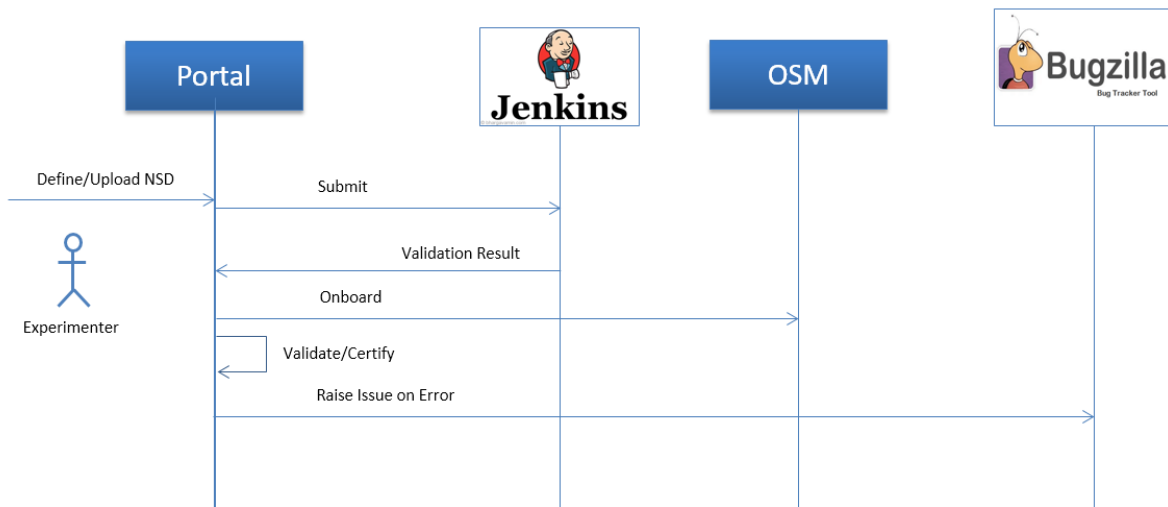


Figure 4 Uploading an Experiment Descriptor/NSD (Automated)

2.2.2.5 Request a new experiment deployment

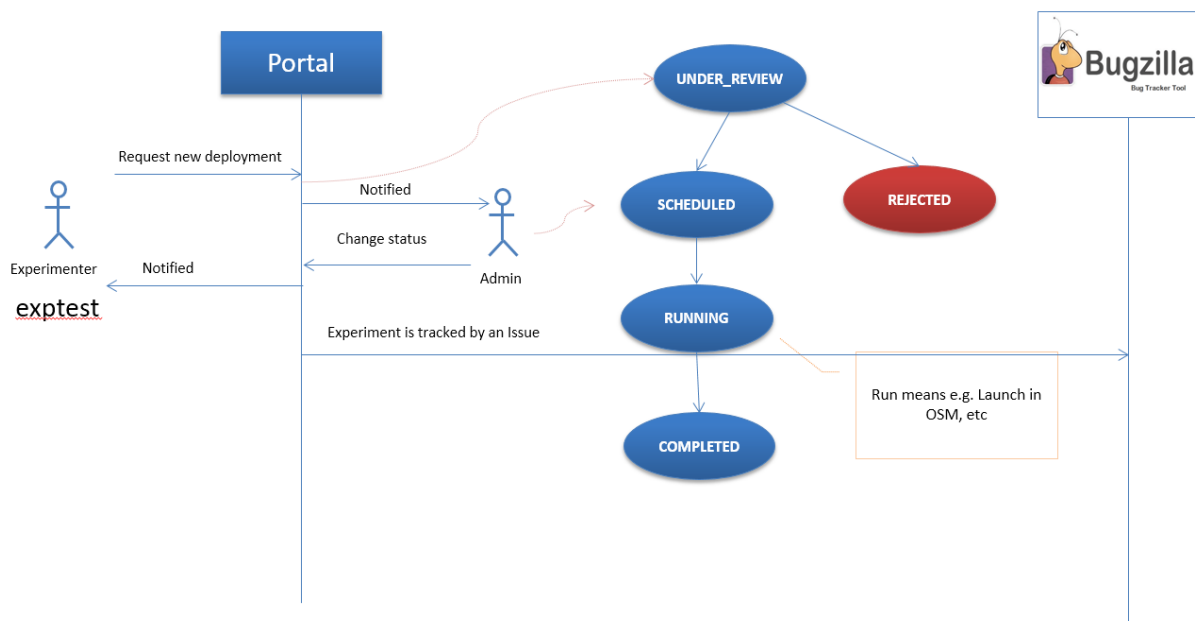


Figure 5 Request a new experiment deployment

During this process (see Figure 5) the following occurs:

- An experimenter requests a new experiment deployment (which NSD, tentative dates, target infrastructure, etc.). The request is marked as **UNDER_REVIEW**.
- The administrator is notified about the new request and he has the following options:

- Schedule the deployment for the requested dates or propose other dates. The request is marked as **SCHEDULED**.
- Reject the request for some reason. The Request is marked as **REJECTED**.
- Deploy the request to target VIM(s). The Request is marked as **RUNNING**.
- Finalize the deployment and release resources. The Request is marked as **COMPLETED**.

On every change of the request-lifecycle the experimenter is notified.

Moreover, the portal automatically creates an Issue in our issue management system to track the progress of the experiment.

2.2.2.6 Automated NSD Job Instantiation

Figure 6 displays the interaction during an automated NSD instantiation for an experiment on a specific date. This will be scheduled and triggered by the portal through the OSM API for a specific NSD. However, this will not be always possible unless certain conditions are met in order to be successful, such as:

- Proper VNF placement.
- Instantiation configuration parameters in VNFs.

The same process will be used for tearing down an NSD instance.

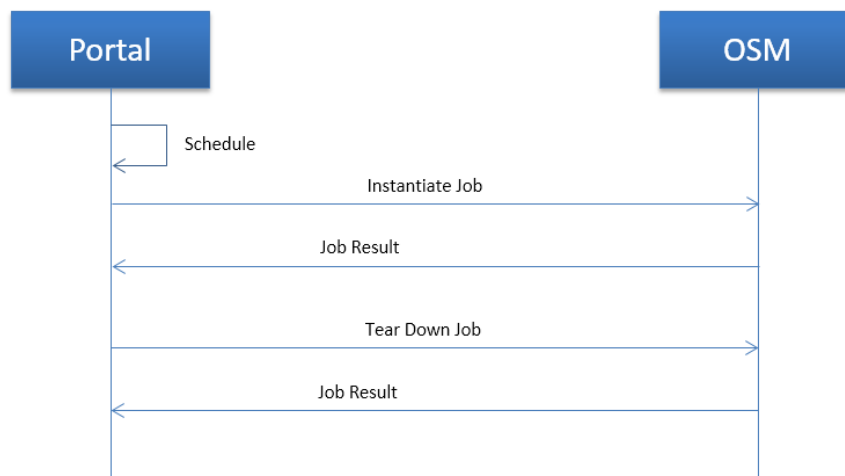


Figure 6 Interaction between portal and OSM to automatically instantiate an NSD

2.3 Architecture

In D2.1 Section 4.5 we presented a high level architecture of the portal. Figure 7 **Error! Reference source not found.** Displays a detailed architecture of the portal together with the related interfaces, services and components. In general, the portal offers a web frontend available for end-users, the portal API backend and a set of support services.

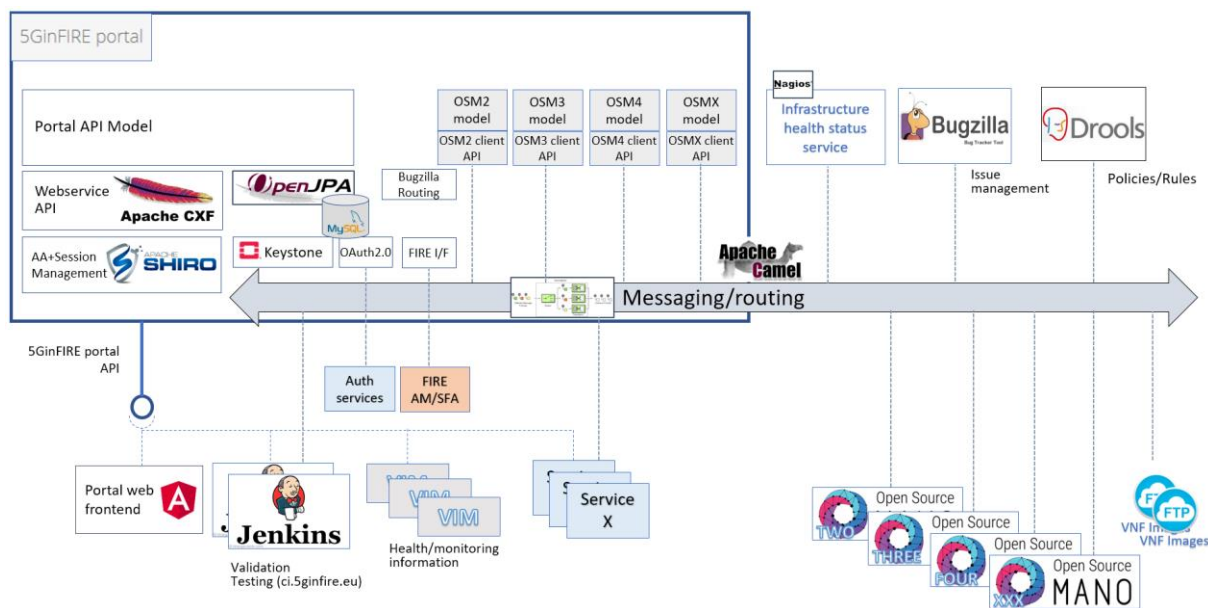


Figure 7 5GinFIRE portal architecture

The architecture in Figure 7 is an update of the related architecture described in D3.1, to support: i) automation and multiple services, ii) the evolution of components as well as iii) to transform the portal into a BSS/OSS system as defined in the ETSI reference architecture [3].

These are the services that the portal service needs to interact with:

- The Continuous Integration service that will perform the validation of submitted VNFs and NSDs
- OSM: There is a continuous evolution of OSM, which has a roadmap to provide new versions of OSM almost every 6 months. This evolution involves the North Bound API that the portal consumes which usually changes. The portal needs to support various version of OSM to ease the process between testing and production.
- Issue management system - Bugzilla: the portal needs to notify the 5GinFIRE operations automatically via the issue management system for various events, such as reporting
- VNF Images repositories: These repositories contain the VNF images that need to be deployed in target VIMs
- Policies/Rule engine: This service should keep various rules, such as which VNFs are allowed to be placed on target testbeds, roles permissions, etc
- Infrastructure health status services, central logging and alerting: the portal reports its health status as well as some information logging or generate alerts
- VIM monitoring information: VIMS might report information to the portal about their status
- OAuth 2.0 mechanisms: to support authentication from 3rd parties
- The FIRE Aggregation Manager service: to support access of FIRE users to 5GinFIRE testbed

Having the above needed interfacing, the portal service consists of the following components:

- Portal model: contains the model of entities, their definitions and associations of the portal entities like users, VxV/NSD experiment metadata, categories, etc.
- Persistence and DB: a persistence layer based on OpenJPA [1] to keep entities permanently available through the database system based on MySQL .
- User identity: This is based on Openstack Keystone service [2]
- AA: Authentication and authorization mechanism(s) to allow access to the portal API based on Apache SHIRO [3]

- Webservice REST API: implementation of the portal API server based on Apache CXF [4]
- OSM models and clients: implementation of a client and its model that communicates with OSM via the Northbound API, in order to on-board VxF and NSDs to the OSM repository and get related information, instantiate NSDs, etc. Each supported OSM version (e.g. TWO, THREE, FOUR) as well as future version might have multiple client connectors to support backward connectivity and future migration from testing to production systems
- OAuth 2.0 Client API AA: implementation of a client that can communicate with service(s) to authenticate/authorize users via OAuth 2.0
- A messaging/routing service bus: This is used to route messages to various services and is based on Apache Camel

The Web front end is implemented in Angular [5] and communicates with the backend via the portal RESTful API. See left lower part of Figure 7.

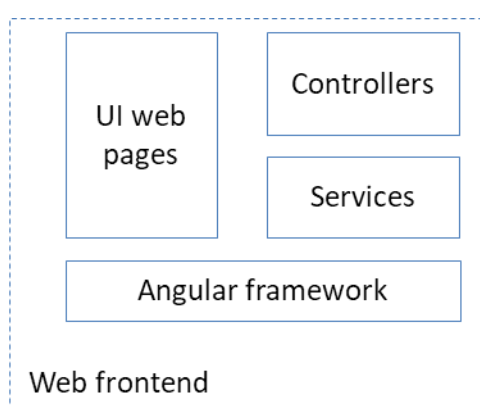


Figure 8 Web frontend architecture

Figure 8 displays the web frontend architecture. It consists of the following components:

- The Angular framework which supports the web implementation
- The UI web pages facing the end users
- The controllers for each page
- The services that correspond to model entities (VxF, User, Experiment, etc.) and provide communication means with the API backend

2.4 Design and implementation

2.4.1 The Portal API backend

The portal API backend is written in Java. The design described here is also reflected in the code

2.4.1.1 Class diagrams

This section presents the static view of the backend API portal design in terms of class diagrams.

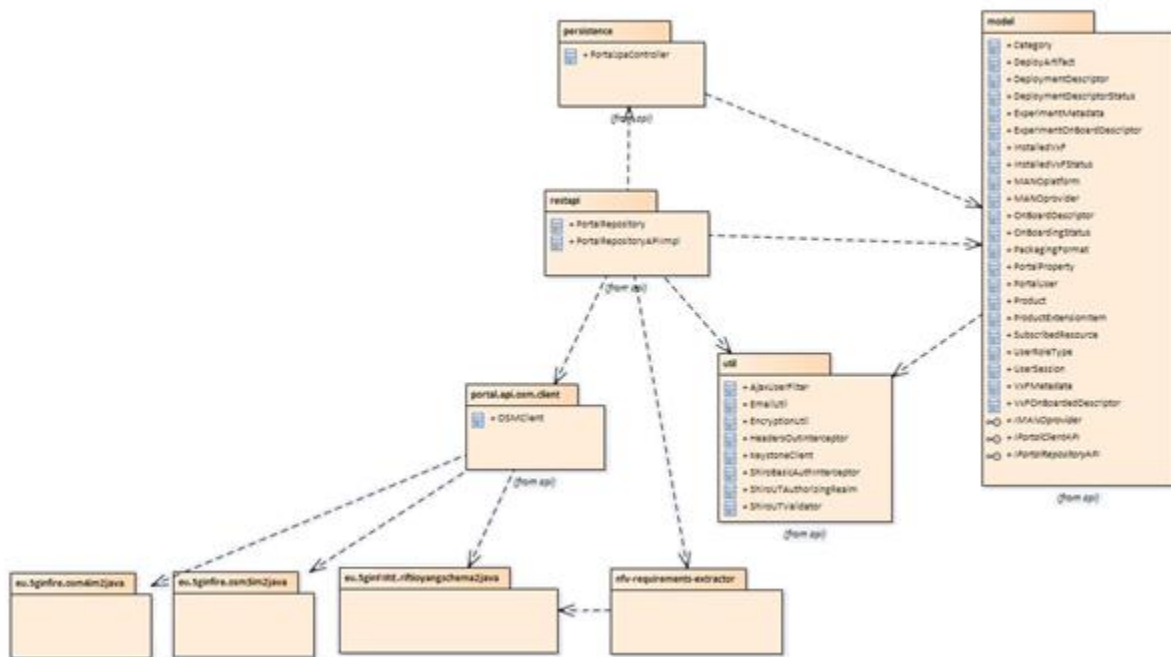


Figure 9 Main package diagram

Figure 9 displays the main package diagram of the backend API. There are the following packages:

- **model:** contains all the core model entities of the portal backed API as the next paragraph describes
- **persistence:** contains the implementation of the JPA persistence
- **util:** contains various utility classes
- **nfv-requirements-extractor:** utility package for extracting and reading the VxV/NSD archives
- **portal.api.osm.client:** contains implementation of a client that connects to OSM via the OSM API
- **restapi:** contains the Restful API implementation. Depends on all the packages
- **eu.5ginfire.riftioyangschema2java:** contains classes that implement the OSM TWO API model in Java, based on the OSM YANG model
- **eu.5ginfire.osm3im2java:** contains classes that implement the OSM THREE API model in Java, based on the OSM YANG model
- **eu.5ginfire.osm4im2java:** contains classes that implement the OSM FOUR API model in Java, based on the OSM NBI model

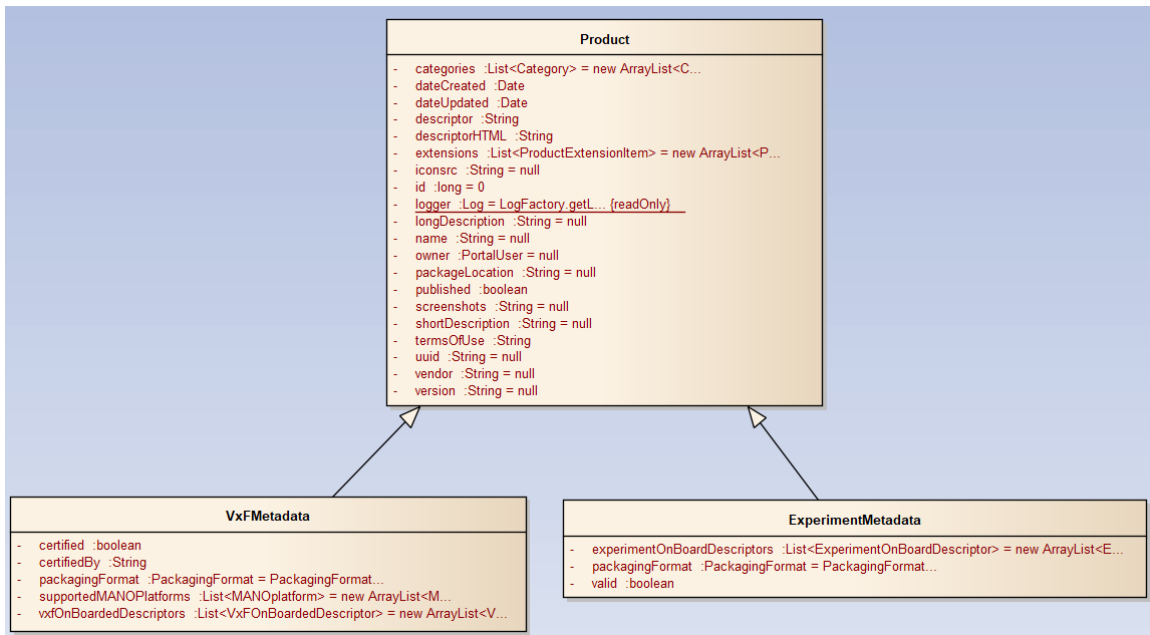


Figure 10 details for the definitions of VxF and Experiment metadata

Figure 10 displays the class diagram containing the core elements of the backend API portal. The VxFMetadata and ExperimentMetadata are both entities for describing a VxF and an experiment respectively; both inherit the abstract class Product. The metadata are common and useful to be displayed by the portal, e.g. name, owner, vendor, etc

Figure 11 displays details of the core elements of the model package. The portal products (VxFMetadata and ExperimentMetadata) are owned by a PortalUser and can belong to many categories.

The class OnBoardDescriptor contains details about the OnBoarding status of the VxF or Experiment on a target MANO provider. VxFOnBoardDescriptor and ExperimentOnBoardDescriptor both inherit OnBoardDescriptor. OnBoardDescriptor is useful to know if the VxF or the Experiment is already onboarded to the target MANO via the MANOProvider class. OnBoardDescriptor keeps also the OnBoardingStatus(OFFBOARDED,ONBOARDED)

Figure 12 displays the class diagram for the DeploymentDescriptor entity. A DeploymentDescriptor is created when a user requests to deploy an experiment (see e.g. Figure 3) and holds information about this deployment as well as its status (UNDER_REVIEW, SCHEDULED, RUNNING, REJECTED,COMPLETED)

Finally Figure 13 displays the relationships for PortalRepositoryAPIImpl class which implements the RESTful API.

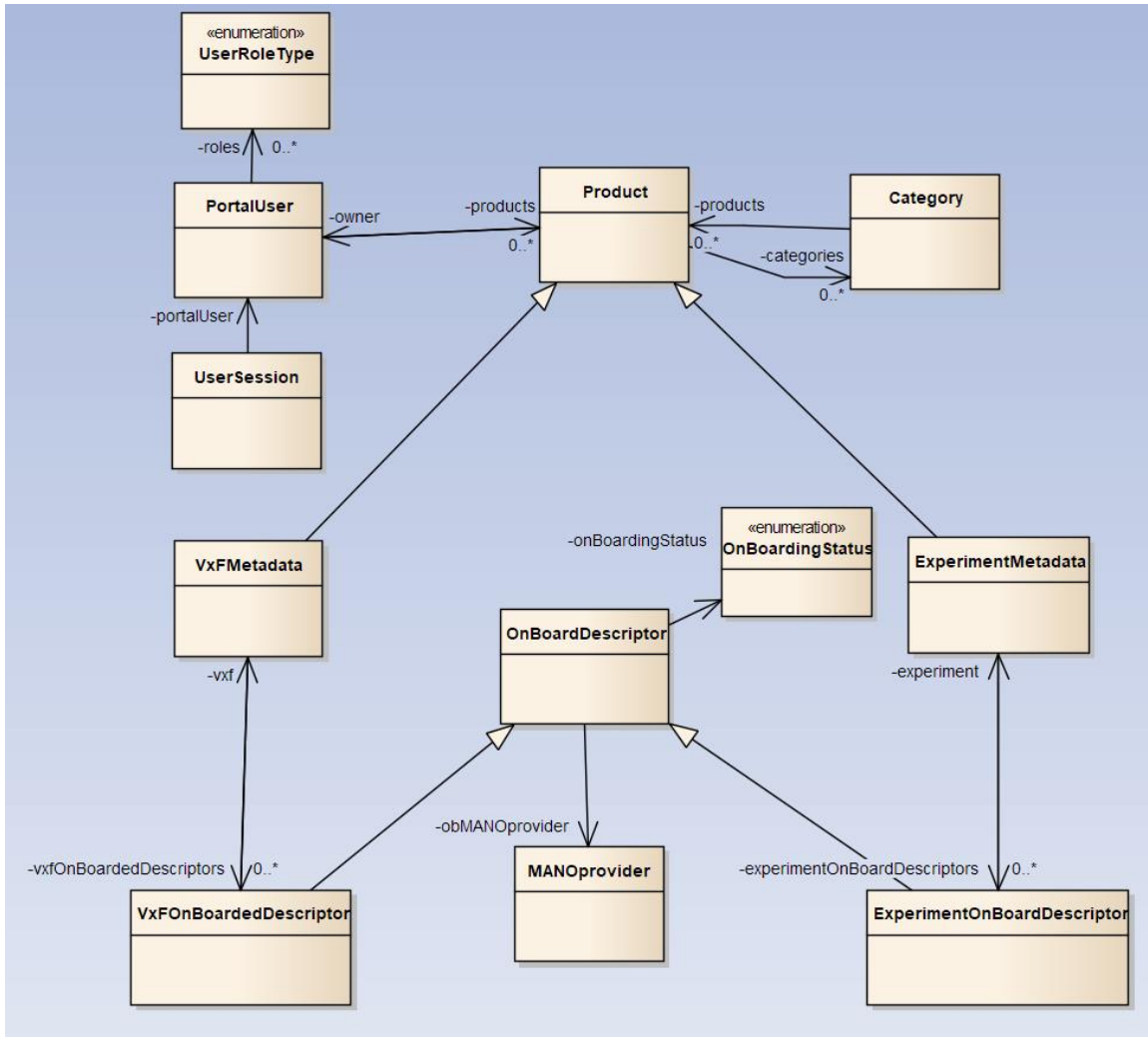


Figure 11 Core model class diagram

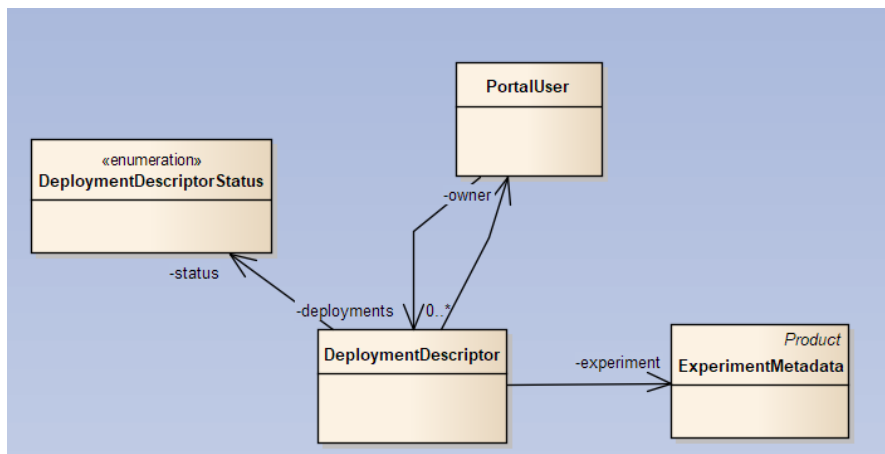


Figure 12 Model of a deployment descriptor

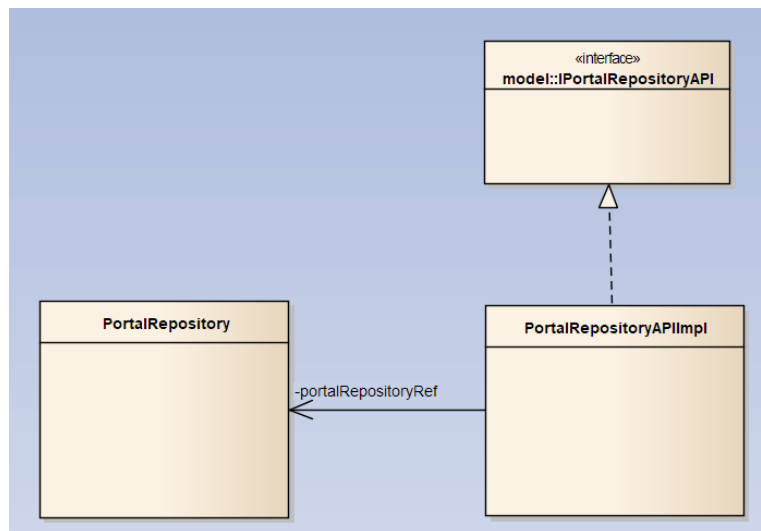


Figure 13 Diagram of the class PortalRepositoryAPI which implements the RESTful API

2.4.1.2 The REST API

The backend API is under `<serverURL>/5ginfireportal/services/api/repo/*` and `<serverURL>/5ginfireportal/services/api/repo/repo/admin/*` for authorized requests. For example, as the portal is under <https://portal.5ginfire.eu> requests can be madetowards: https://portal.5ginfire.eu/5ginfireportal/services/api/repo/*

The API, Produces("application/json") and Consumes("application/json") except some POSTs that Consume("multipart/form-data"). All requests should be to the /repo of the webservice.

The API endpoint is at:

https://portal.5ginfire.eu/5ginfireportal/services/api/repo/*

The API has an OpenAPI [3] specification under:

<https://portal.5ginfire.eu/5ginfireportal/services/api/swagger.json>

A complete API documentation can be found at:

<https://5ginfire.github.io/eu.5ginfire.portal.api/doc/html2-client/>

A login example:

```
curl -v -H "Content-Type: application/json" -X POST --data '{"username":"admin",
"password":"changeme"}' https://portal.5ginfire.eu/5ginfireportal/services/api/repo/sessions
{"username":"admin","password":"","portalUser":{"id":1,"organization":"5GinFIRE","name":"Portal
Administrator","email":"tranoris@ece.upatras.gr","username":"admin","password":"","active":true,"
currentSessionID":"5ec34075-1a12-46d8-97ec-b9e1ab064666","roles":["PORTALADMIN"]}}
```

The following table contains a list of endpoints of the public API that does not need authentication.

Table 3 Public API (does not need authentication)

API endpoint	description
GET/repo/categories GET/repo/categories/{catid}	View categories
GET/repo/vxfs GET/repo/vxfs/{vxfid} GET/repo/vxfs/uuid/{uuid}	View registered Vxfs or by vxfid
GET/repo/sessions POST/repo/sessions GET/repo/sessions/logout	Get user session
GET/repo/experiments GET/repo/experiments/{appid} GET/repo/experiments/uuid/{uuid}	View registered experiments or by id
GET/repo/manoplatforms GET/repo/manoplatforms/{mpid}	View registered supported MANO platforms
GET/repo/manoprovider/{mpid}/vnfds/{vxfid} GET/repo/manoprovider/{mpid}/vnfds GET/repo/manoprovider/{mpid}/nsds/{nsdid} GET/repo/manoprovider/{mpid}/nsds	View registered MANO providers
GET/repo/users/{userid}/vxfs GET/repo/users/{userid}/experiments GET/repo/users/{userid}/vxfs/{vxfid} GET/repo/users/{userid}/experiments/{appid}	Get Vxfs or experiments of specific user by userid
POST/repo/register POST/repo/register/verify	Register a new account to portal and verify it

The following table contains all API endpoints available for authenticated clients.

To make authentication request, after authentication the JSESSIONID cookie value is equal to the sessionId (and JSESSIONID given from server). The JSESSIONID cookie must be presented for authenticated requests

Table 4 Admin API (needs authentication)

API endpoint	Description
GET/repo/admin/vxfs POST/repo/admin/vxfs PUT/repo/admin/vxfs/{bid} GET/repo/admin/vxfs/{vxfid} DELETE/repo/admin/vxfs/{vxfid}	Manage registered Vxfs
GET/repo/admin/experiments POST/repo/admin/experiments GET/repo/admin/experiments/{appid} DELETE/repo/admin/experiments/{appid} PUT/repo/admin/experiments/{aid}	Manage registered experiments
GET/repo/admin/users/{userid} PUT/repo/admin/users/{userid} DELETE/repo/admin/users/{userid} GET/repo/admin/users POST/repo/admin/users	Manage registered users
GET/repo/admin/categories POST/repo/admin/categories GET/repo/admin/categories/{catid} PUT/repo/admin/categories/{catid} DELETE/repo/admin/categories/{catid}	Manage registered categories
GET/repo/admin/properties/{propid} PUT/repo/admin/properties/{propid} GET/repo/admin/properties	Manage portal properties
GET/repo/admin/deployments/{id} PUT/repo/admin/deployments/{id} DELETE/repo/admin/deployments/{id} GET/repo/admin/deployments POST/repo/admin/deployments	Manage requested deployments for experiments
GET/repo/admin/manoplatforms POST/repo/admin/manoplatforms GET/repo/admin/manoplatforms/{mpid}	Manage registered MANO platforms

PUT/repo/admin/manoplatforms/{mpid} DELETE/repo/admin/manoplatforms/{mpid}	
GET/repo/admin/manoproviders POST/repo/admin/manoproviders GET/repo/admin/manoproviders/{mpid} PUT/repo/admin/manoproviders/{mpid} DELETE/repo/admin/manoproviders/{mpid}	Manage registered MANO providers
GET/repo/admin/vxfobds POST/repo/admin/vxfobds GET/repo/admin/vxfobds/{mpid} PUT/repo/admin/vxfobds/{mpid} DELETE/repo/admin/vxfobds/{mpid} GET/repo/admin/vxfobds/{mpid}/status PUT/repo/admin/vxfobds/{mpid}/onboard PUT/repo/admin/vxfobds/{mpid}/offboard	Manage registered VxF onboard descriptors Onboard and offboard a descriptor from target MANO provider
GET/repo/admin/experimentobds POST/repo/admin/experimentobds GET/repo/admin/experimentobds/{mpid} PUT/repo/admin/experimentobds/{mpid} DELETE/repo/admin/experimentobds/{mpid} GET/repo/admin/experimentobds/{mpid}/status PUT/repo/admin/experimentobds/{mpid}/onboard PUT/repo/admin/experimentobds/{mpid}/offboard	Manage registered NSD onboard descriptors Onboard and offboard a NSD descriptor from target MANO provider

2.4.2 The web front-end

The Web front-end is written in HTML and AngularJS. The design described here is also reflected in the code.

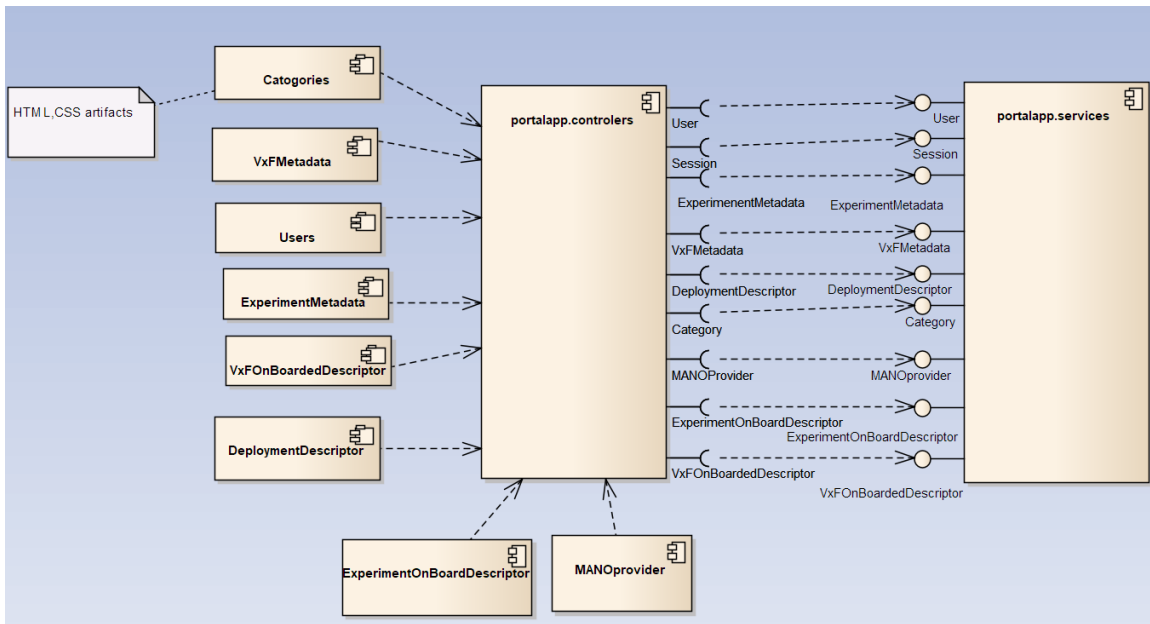


Figure 14 Web Front end components

Figure 14 displays the component diagram of the web front-end. On the left side there are defined services that communicate with the backend API to create/retrieve/update/delete (CRUD) remote objects - equivalents of PUT/GET/POST/DELETE on the REST API services.

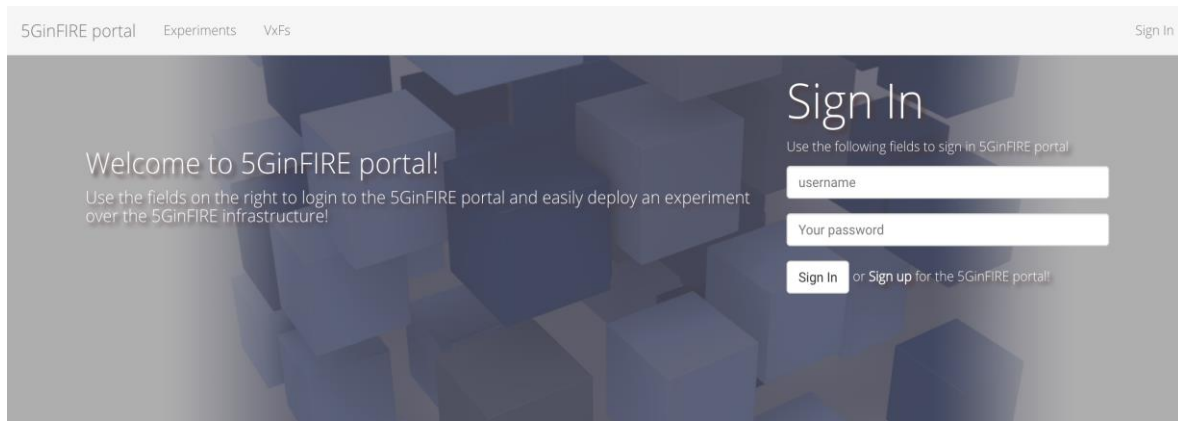
Controller functions at the back-end provide an interface for the services and implement the required control logic . All other HTML artifact components (Categories, Users, VxFMetadata, etc. - on the left side of Figure 14) depend on the controllers.

2.5 Requirements Implementation

This section describes how the implementation is performed around the requirements set in D2.1 and in Section 2.2 of this document. We will describe the portal menus and their capabilities for each and every user role of 5GinFIRE, namely experimenter, VxF developer, testbed provider, and service administrator.

2.5.1 Public user Web interface/Landing page

Initially (beside authorized user roles) there are anonymous/public users who can only see the published VxFs and experiments without needing a portal account. The main interface that all kind of users have access is the landing page that can be seen in figure below:



Deploy 5GinFIRE experiments!

Access, create and share experiments over the 5GinFIRE infrastructure!

Figure 15 The 5GinFIRE portal landing page

At the menu on the top experiments are presented and Vxfs tabs which redirect users to the pages in Figure 15 and Figure 16, accordingly. Additionally, an authorized user gets access to its account by inserting its username and password at the fields bellow the “Sign in” label. A user can sign up in order to get an authorized account by clicking on sign up text description next to sign in button and then it is redirected at the page which is shown up in Figure 18. After that the user inserts its details in the appropriate fields and then when is submitting its request administrator is responsible for its approval.

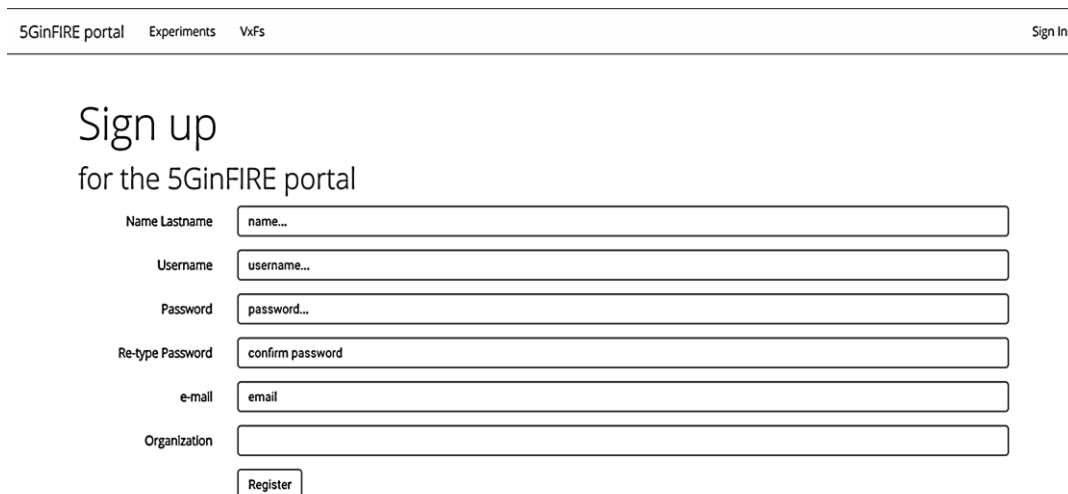


Figure 16 Sign up to portal

Next, the user interface for every authorized user role is described. The landing page is displayed when a user is logged in the portal. All the available menus provided by the user interface are described in the following sections.

2.5.2 VxF developer user interface description

The first user role is the VxF developer. In the next figure the landing page for this role is presented:

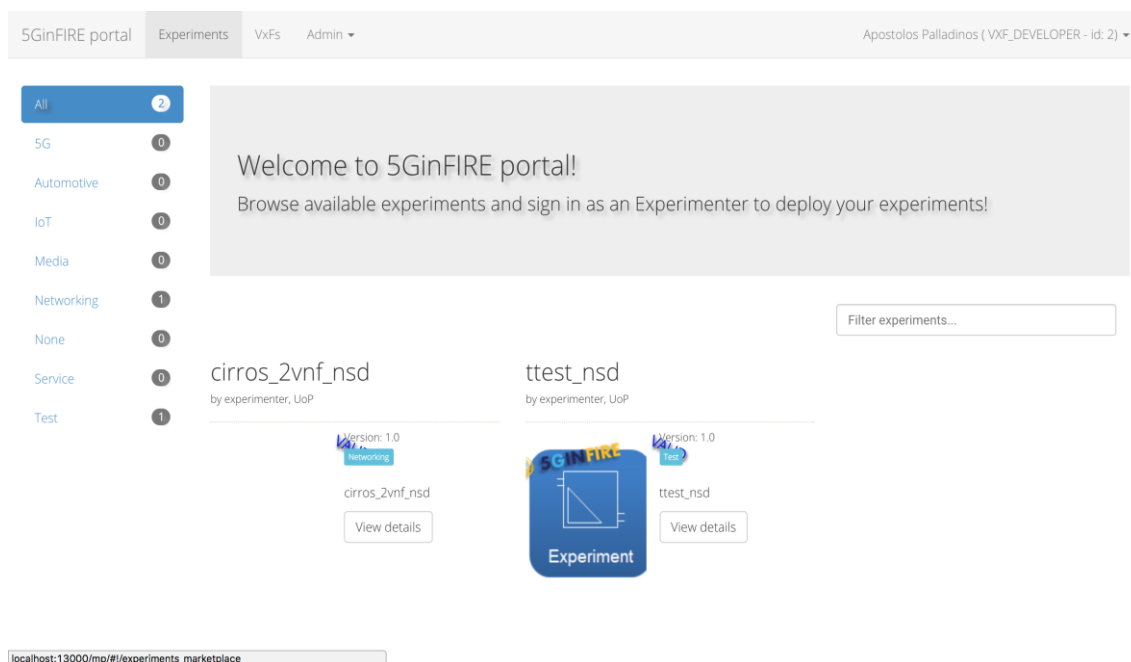


Figure 17 Available experiments, VxF developer login page. (Use case #2010)

On the top part of the interface the available options provided to the VxF developer role are presented. When a user of this role is logged in its account, the experiments option is selected by default. On the left part of the page, the available categories in terms of experiments that are to be uploaded by either the experimenter or the service administrator user role are shown. The categories mentioned previously are designated by service administrators, except for the “All” category, which includes all the published experiments by service administrators, and the pre-existing “None” category which is created by portal installation. Finally, the published experiments belonging to the opted category are present at the bottom of the interface.

Similarly with the previous description in the Figure 16 all the available published VxFs uploaded by either the VxF developers or the service administrators are available at VxF menu option:

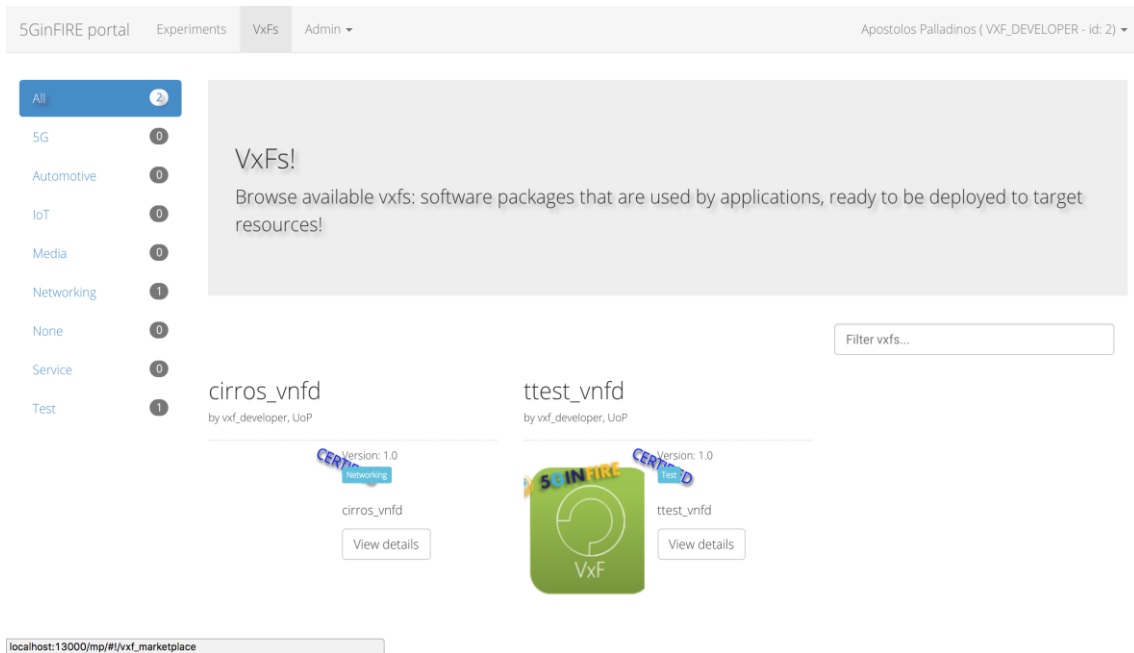


Figure 18 Available Vxfs. (Use case #2070 or #3030)

The options of the Admin tab of the menu can be seen in Figure 19 below:

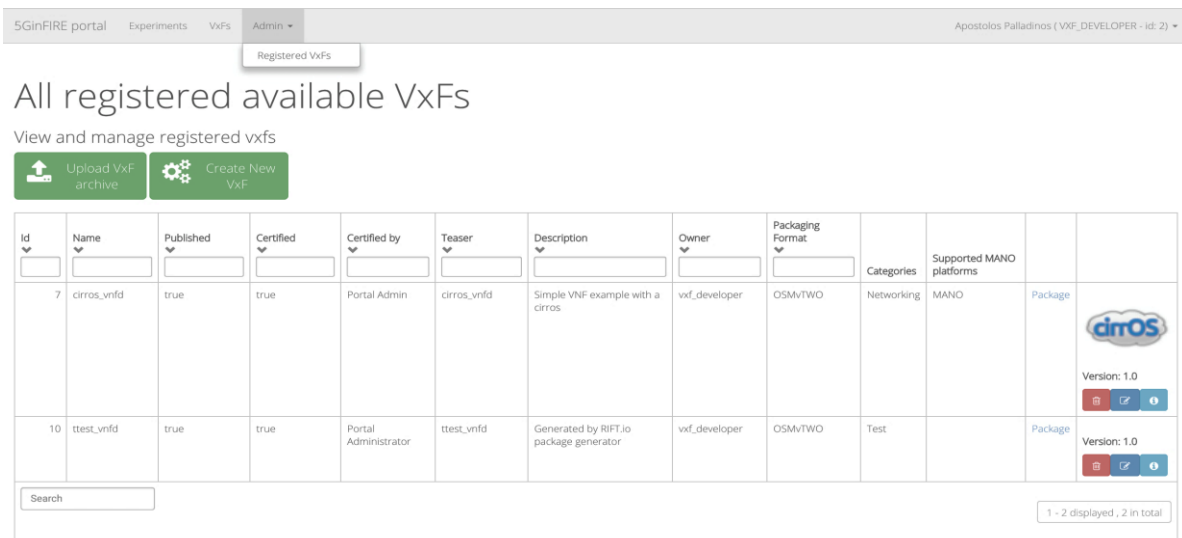


Figure 19 Vxfs Management. (Use case #3020)

The only submenu of this tab is the **Registered Vxfs** which are presented in the main body of the page in Figure 19. In this section a Vxfs developer can list the registered Vxfs in the Vxfs repository and additionally at the table provided by the interface the user can see some details about each Vxfs. Additionally, below each field descriptor a text box is provided in order to help the user to search for a specific Vxfs based on the corresponding search feature. Apart from this kind of search, a search text box can be found below the table where a general search decoupled from Vxfs's features can be performed. The Vxfs developer can also delete, edit and review some information on the fly for each record of the table by clicking accordingly the desired button at the last column of the provided

table. Finally, above the table there are two green buttons available where a user can upload a VxF archive or create a new one. In the first case, the user just uploads the VxF archive on the VxF repository by choosing also the category in which the VxF belongs and by writing some terms of use for this as well. You can see the user interface of this procedure at Figure 18.

Figure 20 VxF upload. (Use case #3010 or #2040)

In the second case a more refined procedure is provided by making the VxF developer able to insert some basic metadata of the uploaded VxF archive through the user interface in Figure 20.

Figure 21 VxF creation. (Use case #3010)

The most obvious metadata are the name, version, teaser, vendor, logo, description and terms of use of VxF. For the rest of the metadata there are some predefined values available. At packaging format field, the VxF developer provides the type of the VxF file. For instance the available formats could be OSM Release TWO or TOSCA. Regarding the last two metadata fields, the Category field refers to the category in which the VxF belongs to as we have also indicated in the first case, and the Supported MANO Platforms field contains a list of the supported MANO platforms like OSM TWO etc. Those platforms are declared by the services administrator role through its interface as we will see in the next section.

Finally, the description of VxF developer user role and the interfaces of edit and info button of each VxF are presented. In the case of the edit button, the interface is the same as Figure 21 besides that a descriptor metadata field is provided as well. In that field, we can overview the YAML description of the chosen VxF. The interface of the info button, as well as, the details button of the available VxFS in the Figure 18 can be seen in the figure below:

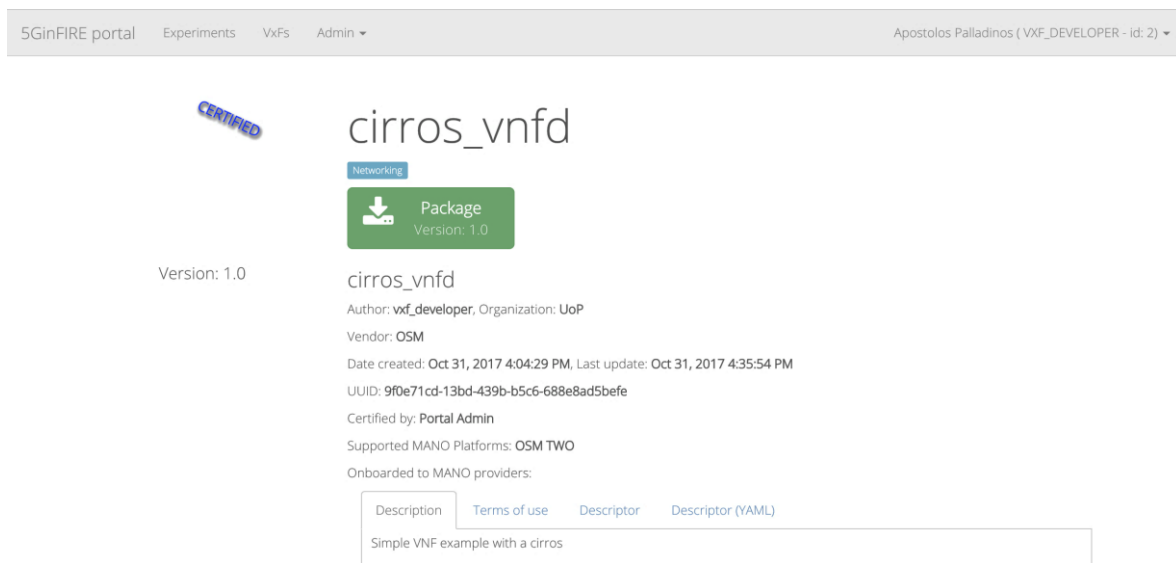


Figure 22 VxF details. (Use case #3030)

Here most of the information already provided by either the VxF package or the VxF developer is presented along with some additional fields

- Organization,
- Date created,
- Last Update,
- UUID,
- Certified by
- Onboarded to MANO providers.

Most of those fields are self-explainable. The Onboarded to MANO providers field, contains the MANO provider on which this VxF has been deployed. Finally, a VxF developer can download the stored VxF package just by clicking the green button on the top side of the page.

2.5.3 Experimenter user interface description

The second user role is named "Experimenter". Figure 23 shows the landing page for this role:

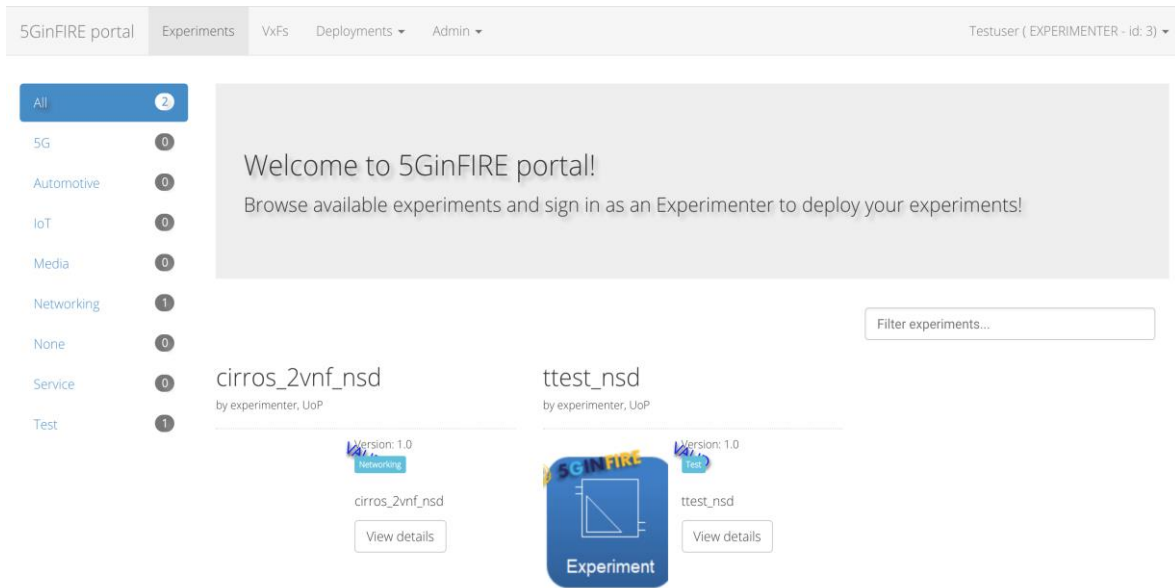


Figure 23 Available experiments, experimenter login page. (Use case #2010)

As we can note this page is almost identical to the page of VxF developer role but differs to the menu on the top side of the page. A new tab Deployments has been added comparing with the previous menu. In addition, the submenu of Admin tab has been renamed to registered experiment descriptors. The new functionalities available to the “Experimenter” role omitting the common already described at VxF developer section are presented in the following parts of this section. The first menu option is “Deployments” and the user selects the menu option “Deploy Experiments”. The user interface for this submenu is shown up in figure below:

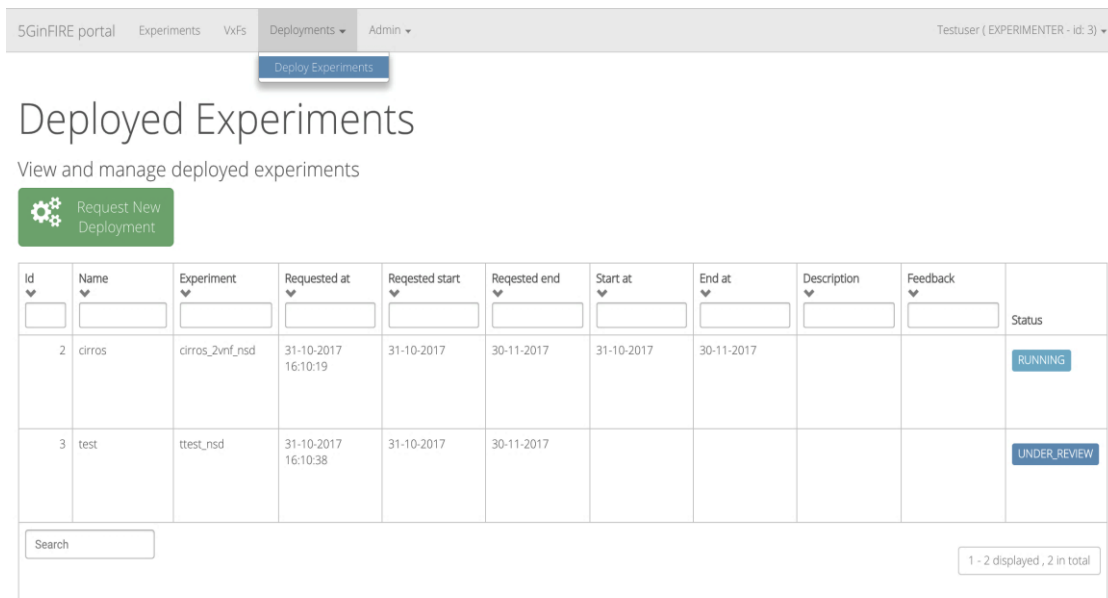


Figure 24 Available Deployed experiments (Use case #2010)

This screen lists the deployed experiments by the services administrator role. Similarly, with the available VxF screen for each field, a text box is provided in order to help the user to search for a specific deployed experiment based on the corresponding search feature. Also, a general search is provided under the table. All those fields are described at the interface shown after clicking on the green button on the top side of the page. Its role is to create a new deployment by selecting one of the available experiments. Finally, the status column denotes the current state of each deployment and is managed by the services administrator role. The available states will be described later on in services administrator role section. By pressing the green button, the layout in next figure is shown up:

Figure 25 Experiment deployment creation (Use case #2020)

In this figure all the necessary fields to define a new deployment are present. At the experiment drop down menu the desirable experiment from those provided to be deployed by a service administrator can be selected. The experimenter can select also the target infrastructure for all or for each individual constituent VxF. In the form fields the necessary information can be provided by the experimenter. The deployment can be submitted by clicking the “Request deployment” button on the bottom of the interface. Then the deployment request is sent to the service administrator in order to be proceeded or to be rejected.

In Figure 26, we can see the content of the interface when an experimenter selects the Admin tab of the menu, and specifically the registered experiment descriptors. This page is similar to the registered VxF page which appears in Figure 20 besides some metadata fields that are missing. Also this interface lists the available experiments instead of the available Vxfs.

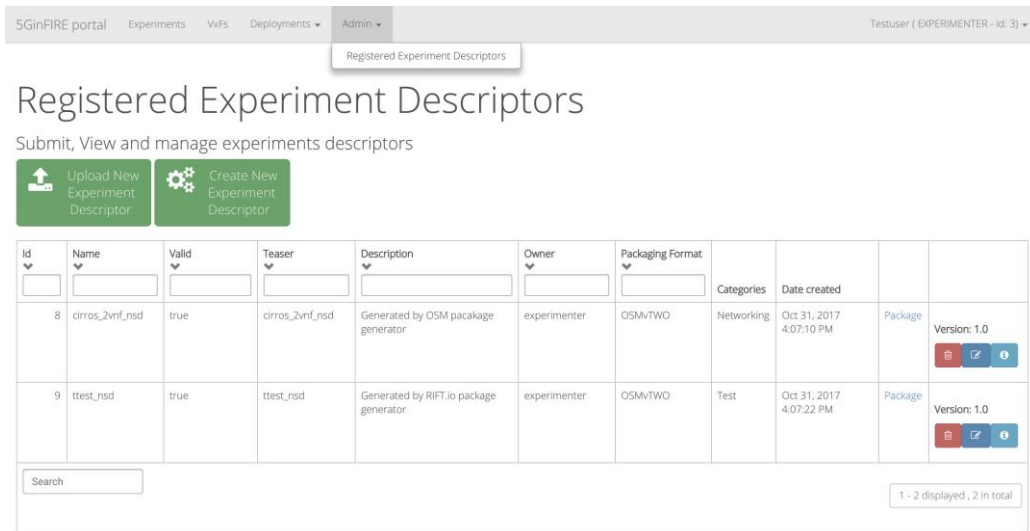


Figure 26 Experiments management (Use case #2050)

By clicking the first green button that is “Upload new Experiment Descriptor” the experimenter is redirected exactly to same interface as in Figure 20, but without the terms of use. This time the experimenter uploads an experiment package file instead of a VxF. The other green button follows similar logic as the corresponding button at the VxF interface but again instead of the creation of a new VxF, a new Experiment is defined by experimenter filling the appropriate metadata fields. The interface is also similar to the VxF interface in Figure 21 but some fields are missing comparing with it but the functionality for the rest of them remains the same. The interface metadata fields for a new experiment can be seen in figure below:

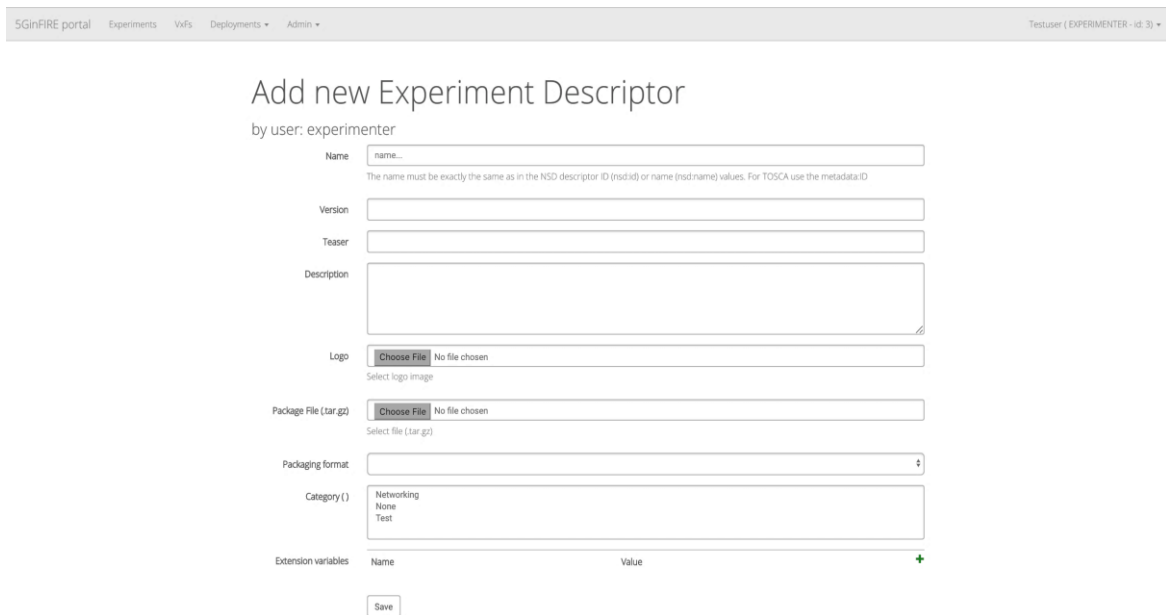


Figure 27 Experiment creation (Use case #2060)

Finally, in Figure 27 the interface where an experimenter lands on when clicking on the edit button of an experiment is shown in Figure 26 Alternatively the user can press the details button of an experiment seeing the screen shown in Figure 23. This page is also available for the VxF developer role as it is shown in Figure 18. This page has similar fields with the VxF details interface in Figure 21, but some fields have been subtracted and the status metadata field has been added to it.

2.5.4 Services administrator user interface description

The last menu we will describe is for the services administrator role. The landing page when an administrator logs in is similar to the experimenter one, and can be seen in Figure 28. The only difference is the submenu of admin tab which will be analysing in the following sections. The first submenu of admin tab is System Users and its functionality is similar with the previously described menus which list some sort of data in table form. The interface of this submenu can be seen in figure below:

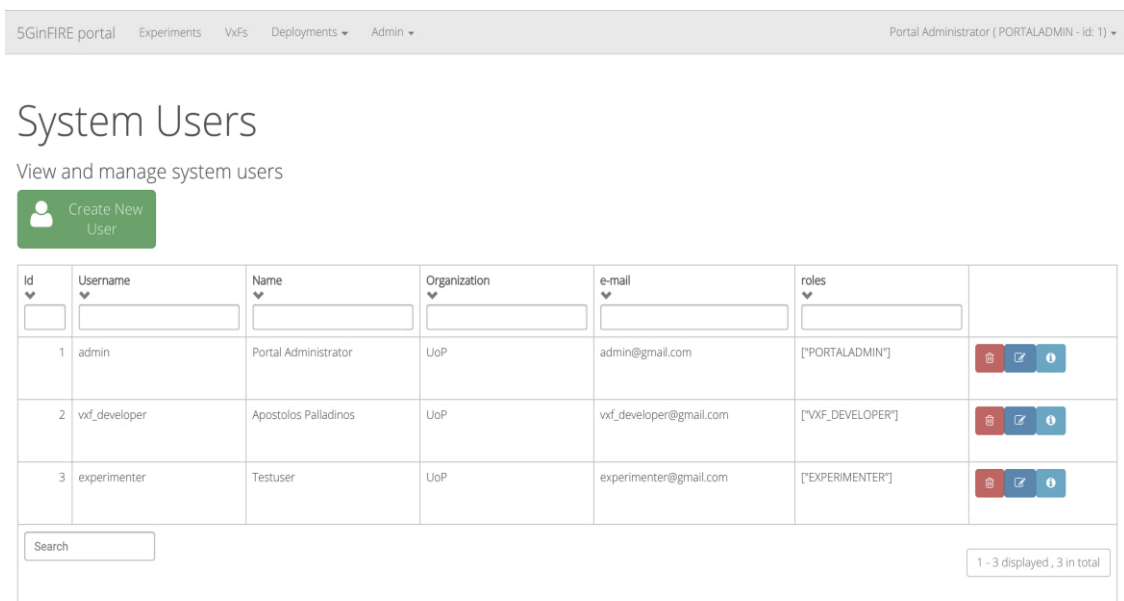


Figure 28 Users management (Use case #5010)

Additionally, by clicking the information icon of a system user, a message appears with user details as of Figure 28. The information interface can be seen in Figure 29 and the create new user interface in Figure 30:

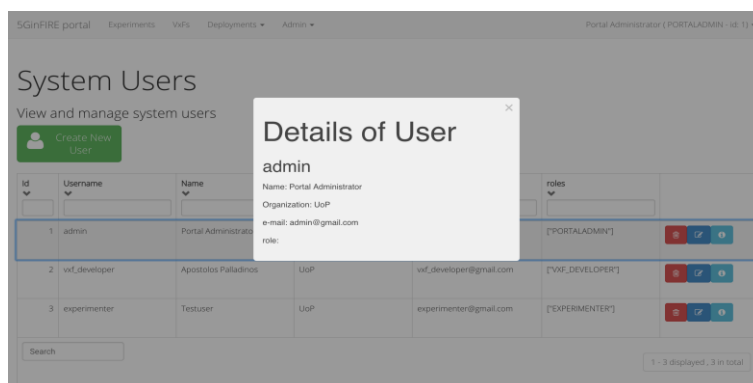


Figure 29 User details. (Use case #5010)

5GinFIRE portal Experiments VxFs Deployments Admin Portal Administrator (PORTALADMIN - id: 1)

Add new User

Name

Username

Password

e-mail

Organization

Role

Figure 30 User account creation. (Use case #5010)

In Figure 30 we see some common used information about a new user: its name, username, password, e-mail, organization and finally its role. Each and every role of this list has been described in detail in introduction section. When all those fields are filled up the new user is created when the administrator clicks on the save button on the bottom of the page.

Figure 31 shows the full submenu of the admin tab. The three submenus after System Users submenu that is Registered Experiment Descriptors, Registered VxFs and Registered Deployed Experiments have exactly the same interfaces as Figure 26, Figure 19, Figure 24 accordingly. The difference for the administrator role is when is clicking on the edit icon of an object where it redirects it to the same pages as the corresponding in Figure 26, Figure 19, Figure 24 but with some additional metadata fields. Each and every of those cases are analysed in the next sections.

5GinFIRE portal Experiments VxFs Deployments Admin Portal Administrator (PORTALADMIN - id: 1)

Registered Experiment Descriptors

Submit, View and manage experiments

- System Users
- Registered Experiment Descriptors
- Registered VxFs
- Registered Deployed Experiments
- System Categories
- System MANO Platforms
- System MANO Providers
- All pending requests
- System info

Id	Name	Valid	Teaser	Description	Owner	Packaging Format	Categories	Date created	Version
8	cirros_2vnf_nsd	true	cirros_2vnf_nsd	Generated by OSM package generator	experimenter	OSM/TWO	Networking	Oct 31, 2017 4:07:10 PM	Package Version: 1.0
9	ttest_nsd	true	ttest_nsd	Generated by RIFT.io package generator	experimenter	OSM/TWO	Test	Oct 31, 2017 4:07:22 PM	Package Version: 1.0

Search

1 - 2 displayed, 2 in total

Figure 31 Admin tab submenu

For the Registered Experiment Descriptors submenu edit button we get as response the page in next figure:

Figure 32 Extra fields for admin role about registered experiments (Use case #4010 and #5030)

Compared to Figure 27, there are two extra fields: published and valid. It is also possible to select experiments to on-board or off-board on the MANO platform. When the administrator checks the published checkbox, the current experiment becomes available to all system's users interfaces. Otherwise, only the user who created the VxF and the administrators can see it in their VxF listing submenus.

For the Registered Vxfs submenu edit button we get as response the page in next figure:

Figure 33 Extra fields for admin role about registered Vxfs. (Use case #3020)

With regard to Figure 21, the form proposes some extra fields: *published*, *certified* and *certified by*. It allows the capability of making the selected VxF on-board or off-board on the available MANO platform. When the administrator checks the published checkbox the current VxF becomes available to all system's users interfaces. Otherwise, only the user who created the VxF and the administrators can see it in their VxF listing submenus. Finally, an administrator can certify this VxF through the Certified and Certified by options.

Finally, regarding Registered Deployed Experiments submenu edit button we get as response the page in next figure:

Figure 34 Deploy a validated experiment (Use case #5030)

The two extra fields in that interface compared to Figure 25 are Status and Comments and Feedback. Status field contains the five states in which a deployed experiment can be found. Under review state denotes that the current deployable experiment is under reviewing by a service administrator who is checking its validity. Scheduled state firstly indicates that the deployable experiment is eligible to be deployable and secondly that it has been scheduled for a specific start date and that the end date has been acceptable as well. The Running and Completed states are self-explainable. Finally, the rejected state means that a service administrator has rejected the uploaded deployable experiment and the experimenter usually should follow the Comments and Feedback provided as a response by the administrator and should re-upload the experiment to get under review again.

System Categories submenu lists the Vxfs and experiments categories created by the services administrator role. The interface is similar to all other listing interfaces that have been presented previously. In Figure 35 and Figure 36 we can find the categories listing interface and the creation of a new category interface respectively.

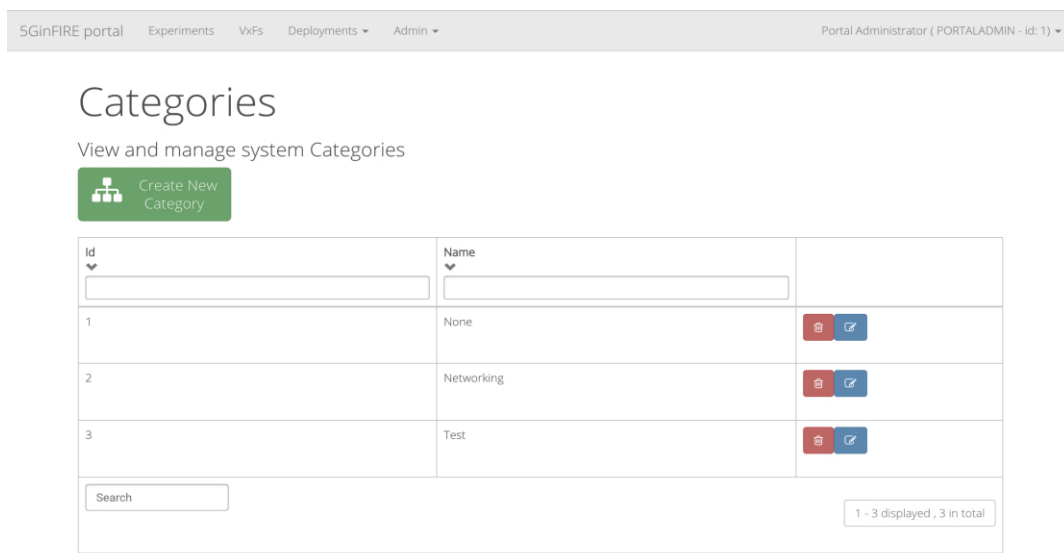


Figure 35 Available categories

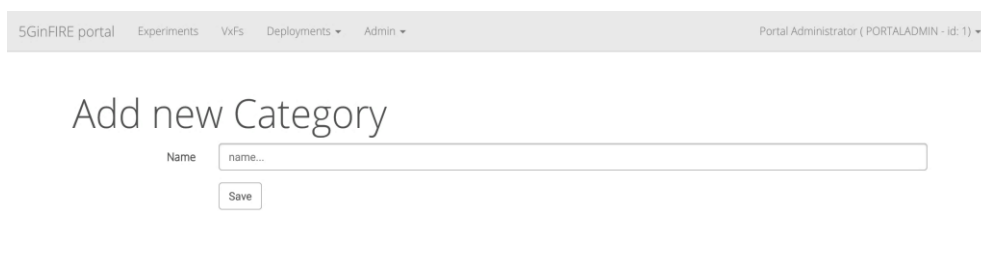


Figure 36 Category creation



As we can see in Figure 36 the field “name” is the only one that needs to be defined. the only field we need to define a new category it is name.

The System MANO Platforms submenu includes all the available MANO platforms defined by service administrators. Again, the interface is similar to all other listing interfaces we have already seen in terms of functionality. In Figure 37 and Figure 38 you can see the MANO platforms listing interface and the creation of a new MANO platform respectively.

MANO Platforms and Versions

View and manage available MANO Platforms list and their versions

[Add New MANO Platform](#)

Id	Name	Version	Description	
1	OSM TWO	2		 

Search 1 - 1 displayed, 1 in total

Figure 37 Available MANO platforms

Add new MANO Platform

Name

Version

Description

Figure 38 MANO platform creation

The only fields required to define a new MANO platform are its name, version and a short description, as Figure 38 indicates.

Through system MANO providers submenu, a services administrator essentially is able to connect the portal to a deployed MANO platform via the API URL field provided during the creation of a new MANO provider. The interface which lists all those MANO providers is approximately the same with the rest listing interfaces already presented. Figure 39 and Figure 40 show the MANO providers listing interface and the creation of a new MANO provider respectively.

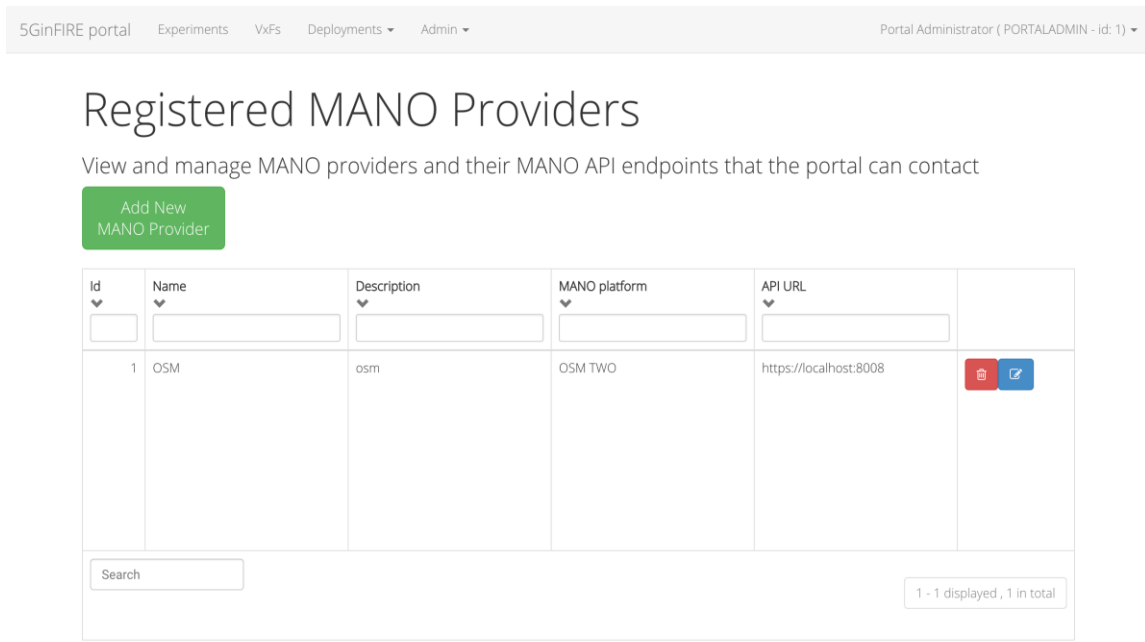


Figure 39 Available MANO providers

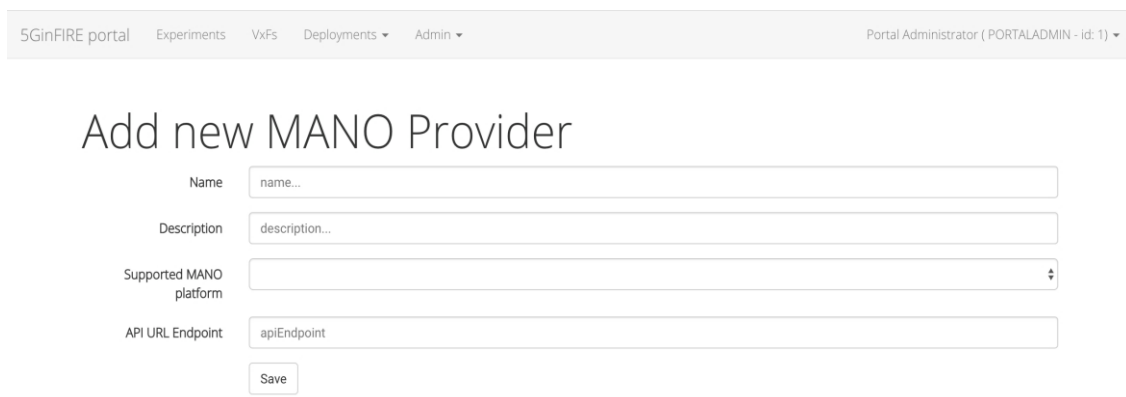


Figure 40 MANO provider creation

In Figure 40 the most notable fields are Supported MANO platform and API URL Endpoint. The former is associated with the MANO platform interface which we have previously described and includes all the available MANO platforms created by services administrators. The latter contains the URL to a deployed MANO platform and essentially is the place where the experiments and VxFS created by portal users are deployed. Finalizing this section, the submenu “All pending requests” is identical to the “deploy experiments” submenu presented in “deployments” tab of the main menu.

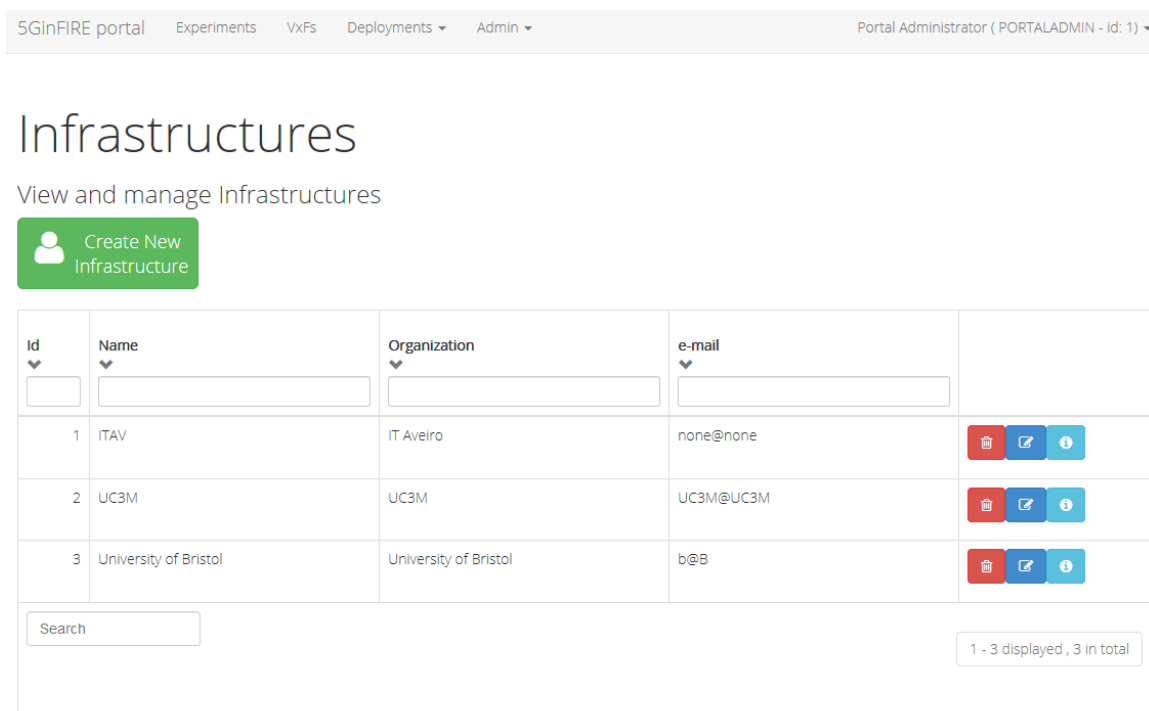


Figure 41 Available target Infrastructures

Finally, there is a page for managing the available target infrastructures for VxF deployments (see Figure 41). This is useful when Experimenters create a Deployment Request to select the target infrastructure.

2.6 Deployment details

5GinFIRE deployment is depicted in Figure 42 and it consists of 3 components:

- **nginx:** a popular web server which is used as a reverse proxy to the backend API application as well as it serves also the Web front end. It is configured also with certificates to ensure secure https communication with end users.
- **Jetty:** a popular web server that can host Java applications. It hosts the backend API
- **Mysql:** a popular database. It is used to persist all data.

5GinFIRE is deployed as a set of Docker containers. A docker compose script downloads the latest code artifacts and configures the deployment as depicted in Figure 43.

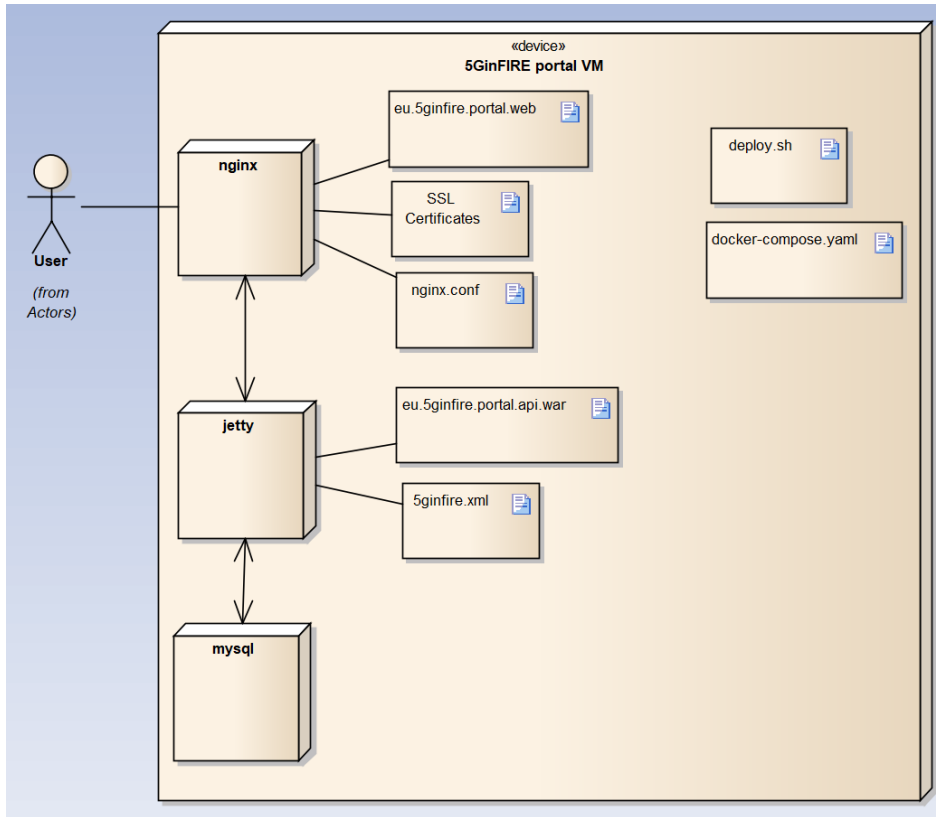


Figure 42 Component diagram of 5GinFIRE

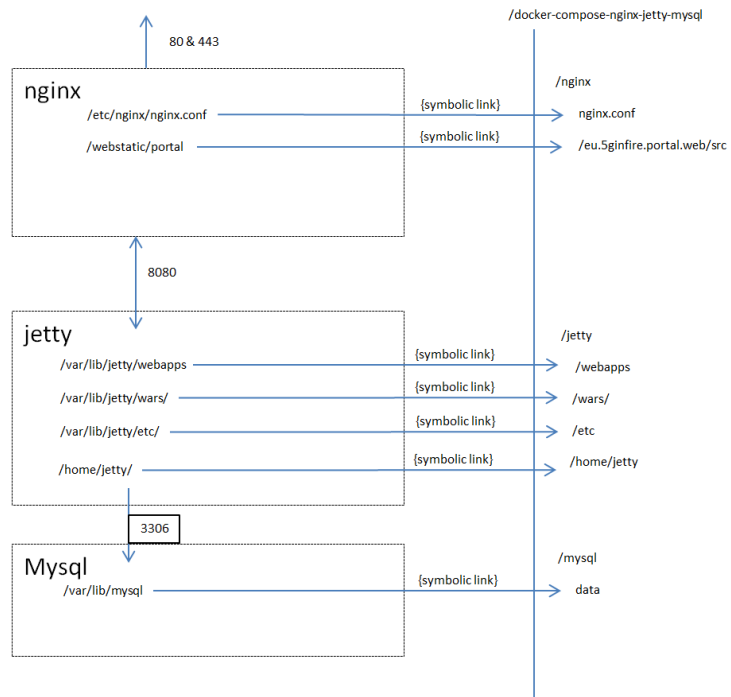


Figure 43 Internal mappings of deployed containers

2.7 Delivery plan

The delivery plan for the portal is depicted in Table 4. The plan is for the first version to be ready by end of February 2018 in order to accept the first Open Call deployments.

Table 5 Deliver plan for Version 1

Version	Expected date (end of)	Description
1.0.Beta 1	11/2017	Portal is online with secure certificates. All defined use cases implemented
1.0.Beta 2	12/2017	5GinFIRE OSM deployed communication established First Vxf/NSD deployments tests Allow authentication via Keystone API server
1.0.RC1	1/2018	First production release (with any issues previously reported finished)
1.0.RC2	2/2018	Second production release (with any issues previously reported finished)
2.0.RC1	6/2018	First production release of version 2 including the service message bus, keystone, FIRE integration and issue system integration
3.0.RC1	9/2018	First production release of version 3 including support of OSM FOUR and communications with OSM production services

The plan is to deliver weekly bug fixes and new features every three months. New features will be defined in upcoming months with 5GinFIRE consortium as well as new open call partners.

2.8 Code repositories

5GinFIRE provided Open Source code is hosted in the popular open source platform of GitHub. There is a specific organization <https://github.com/5GinFIRE/> that hosts all our delivered code artifacts.

The following table describes our repositories related with the portal

Table 6 Code repositories related with the portal

URL /Description
https://github.com/5GinFIRE/eu.5ginfire.portal.web Hosts the eu.5ginfire.portal.web Web front UI. The AngularJS based UI to be used with eu.5ginfire.portal.api Web Service.
https://github.com/5GinFIRE/eu.5ginfire.portal.web/wiki A Wiki that is used as a developer/user guide for the portal

<https://github.com/5GinFIRE/eu.5ginFIRE.riftioyangschema2java>

This project contains the OSM API model in Java, based on the OSM API YANG model.

<https://github.com/5GinFIRE/nfv-requirements-extractor>

Contains a utility to extract VNF/NS requirements from packages

<https://github.com/5GinFIRE/eu.5ginfire.portal.api>

Contains the source code for the Java backend API

<https://github.com/5GinFIRE/eu.5ginfire.osm3im2java>

[Contains the source code for API support in OSM THREE](#)

https://github.com/5GinFIRE/dockerized_keystone

[Contains script for instantiating keystone](#)

2.9 Licensing

All components are open source with Apache 2.0 License included as stated in each project

3 Middleware tools

3.1 Admin tool and provisioning: Launchpad (OSM TWO and OSM FOUR)

The 5GinFIRE consortium agreement regarding the orchestration service is to use OSM Release TWO in the first Open Call of the project, and evolve the software to Release FOUR as soon as this release has been verified in the pre-production environment of the 5GinFIRE MANO platform.

With respect to OSM Release TWO, the Launchpad is the point of contact in the OSM architecture (see Figure 44) to support the management and monitoring of the lifecycle of VNFs and network services. It is part of the User Interface (UI) plugin, within the plugin model framework supported by OSM.

Additionally, the UI is a part of the Service Orchestrator (SO) component. This means that some actions invoked by the UI, which indeed are Northbound API operations, required a direct access to other component of the OSM architecture (e.g., Resource Orchestrator or VNF Configuration and Abstraction) instead of using the Northbound REST API.

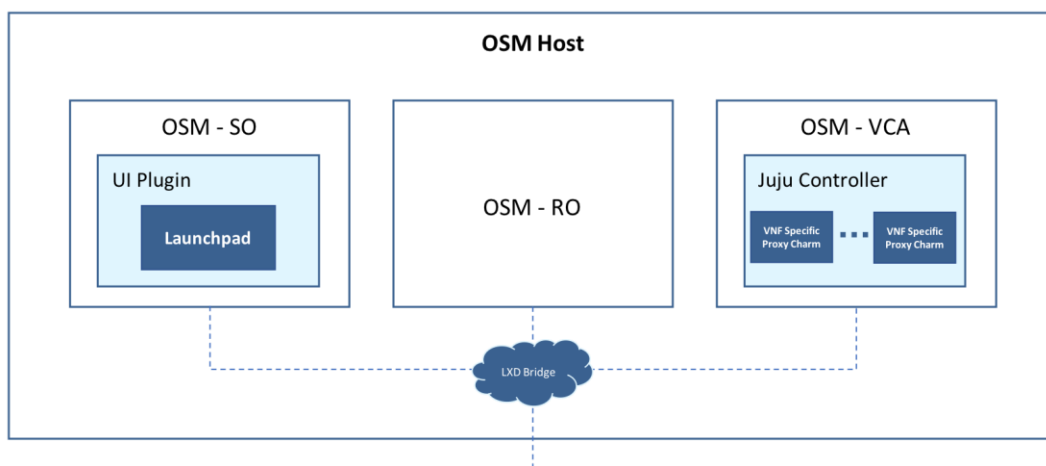


Figure 44 OSM Release FOUR new defined Northbound API

The Launchpad also provides an intuitive GUI which supplies the mechanisms to interact with the OSM run-time system (e.g., to deploy a network service), also providing real-time information through the Dashboard about the status process of the network services.

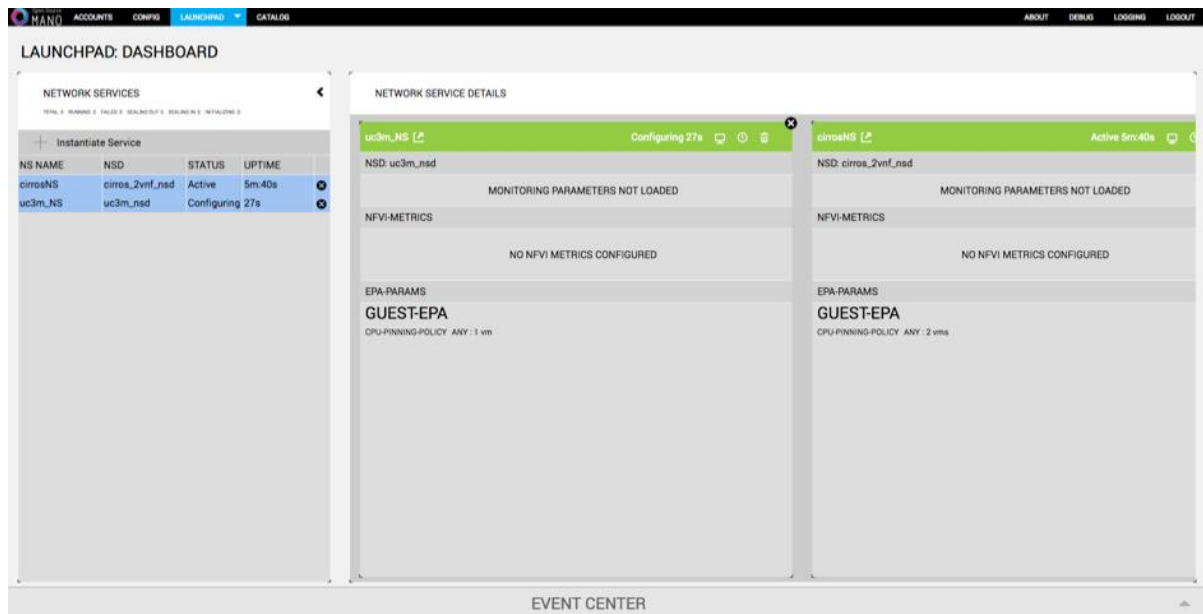


Figure 45 Launchpad: Dashboard

Figure 45 presents an example of the information provided by the Dashboard after the deployment of a network service composed of two VNFs. This web-based interface can also display a detailed view of the utilized compute resources and deployed network topologies, including IP addressing information (e.g., the management IP addresses of the VNFs) or the VDU links (see Figure 46).

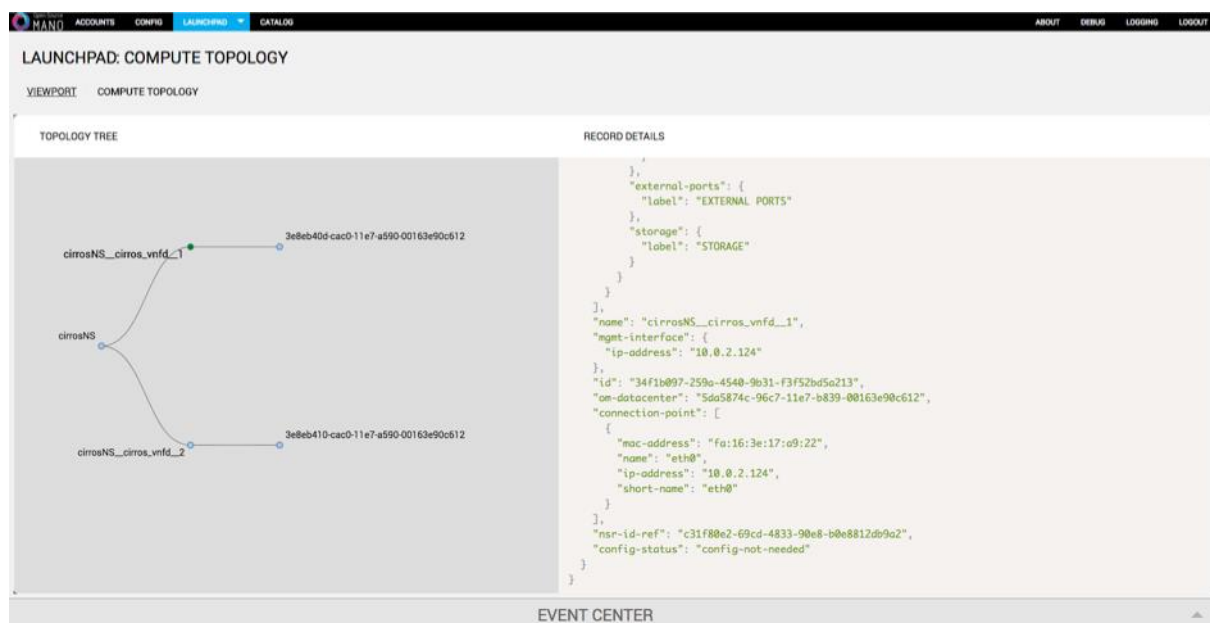


Figure 46 Launchpad: Compute Topology

Another main feature of the Launchpad is that it enables (via the *Instantiate* plugin) the configuration of the parameters that are needed for the instantiation of network services, for instance to support multi-site deployments in case that more than one datacenter has been configured. Figure 47 shows an example of how OSM allows the multi-site instantiation of a Network Service, enabling the selection of the desired datacenter for each of the constituent VNFs that composes the Network

Service. In fact, the Launchpad shows a drop-down list per VNF with the datacenters configured, allowing to select the one where the VNF should be deployed during the instantiation of the network service.

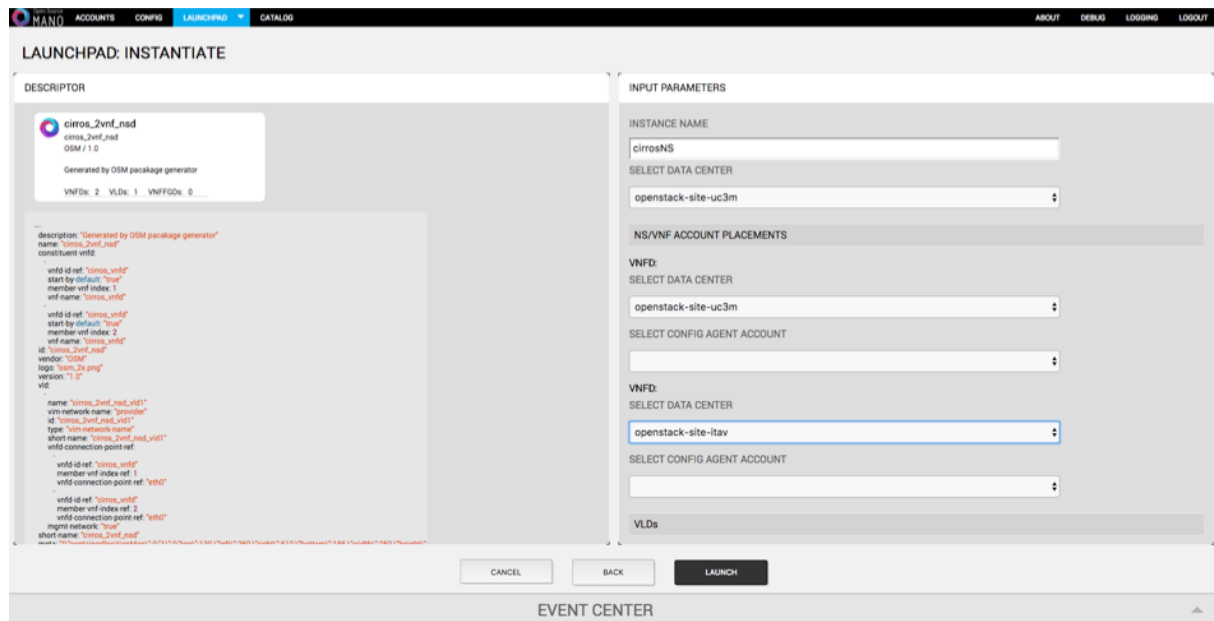


Figure 47 Launchpad: Instantiate

Finally, regarding the Release TWO, we want to mention that the source code of the Launchpad has been implemented with the JavaScript run-time environment Node.js. The code is available in the OSM-UI repository [9].

In reference to the OSM Release FOUR¹, its Northbound interface is aligned with the ETSI NFVI specification SOL005 [10] and provides an entry point for the invocations of the main MANO actions by the external systems such as the UI, OSS or the OSM client (i.e., a command line interface client to interact remotely with the OSM's Northbound API).

This release also includes a standalone light-UI that is capable of operating with the MANO system through the Northbound interface. Similarly, to previous releases, the light-UI offers a mechanism to interact with the MANO run-system, allowing the deployment of different network services and providing information about the status process of the network service deployment. Figure 48 represents an example of a network service deployment carried out using the light-UI.

¹ At the time of writing, ETSI Open Source MANO (ETSI OSM) has announced the availability of OSM Release FOUR (see <http://www.etsi.org/news-events/news/1306-2018-05-news-etsi-open-source-mano-announces-release-four-moving-faster-than-ever>)

Id	Name	Nsd name	Operational Status	Config Status	Detailed Status	Actions
351643b2-9209-4277-9a7c-ee4c5113e822	cirros	cirros_2vmf_ns	running	configured	done	i Actions

Figure 48 Deployment status of a NS in Release FOUR

3.2 YANG models and packaging for VxF/NSD

An NFV descriptor (VNFD or NSD) for OSM Release THREE and FOUR must be modelled using the YANG models defined in [6]. For OSM Release TWO, the information model is specified in [7].

3.2.1 Overview of the YANG models

YANG is a data modelling language addressing the definition of data managed by the Network Configuration Protocol (NETCONF) [8]. The YANG data modelling language was developed by the NETMOD working group of the Internet Engineering Task Force (IETF), and was published as [RFC 6020](#) [9] in October 2010.

The data modelling language can be used to model both configuration data as well as state data of network elements. Furthermore, YANG can be used to define the format of event notifications emitted by network elements, and it allows data modellers to define the signature of remote procedure calls that can be invoked on network elements via the NETCONF protocol. The language, being protocol independent, can then be converted into any encoding format, e.g. XML or JSON, that the network configuration protocol supports.

Further information and examples on the currently used YANG models in OSM Release TWO can be found in (last access in June 2018):

<https://open.riftio.com/documentation/riftware/4.4/a/descriptor/yang-models/mano-yang-models.htm>

3.2.2 Requirements for VNF & NS descriptors

The NFV specifications [10] and [11], from the Interfaces and Architecture (IFA) Working Group of ETSI, define the information models of VNF and NS descriptors, respectively. Table 7 indicates a relevant subset of the requirements for NFV descriptors extracted from the abovementioned documents.

Table 7: requirements for NFV descriptors (complete list in [10]and [11])

Req Number	Requirement Description
VNF_PACK.GEN.001	The VNF package contents, including the VNF descriptor, VNF binaries, configuration, scripts and software images, as well as manifest file, checksum, etc. as appropriate constitutes a single delivery unit from a distribution perspective. Any changes to the constituency of this unit shall be considered as a change to the whole and therefore shall be versioned, tracked and inventoried as one.
VNF_PACK.STRUCT.001	The VNF package shall be assembled in one file.
VNF_PACK.STRUCT.002	The VNF package shall be digitally signed by the VNF provider.
VNF_PACK.STRUCT.003	The VNF package should contain files for one VNF and its corresponding metadata
VNF_P ACK.DESC.004	VNF Package shall contain or reference one or more software images.
VNF_P ACK.DESC.005	The VNF Package may contain at most one software image per VNFC.
VNF_P ACK.DESC.007	The VNF Package shall contain VNFD metadata.
VNF_PACK.DESC 010	The VNF package shall enable including information supporting VNF testing.
VNF_P ACK.ID.001	There shall be a way to identify the version of the VNF Package Specification associated with a particular VNF.
VNF_P ACK.ID.002	VNF Package shall be globally uniquely identifiable. The globally unique identifier for the VNF Package shall be used to uniquely identify the VNFD and the VNF included in the package.
NST_NSD001	The NSD shall reference the VNFDs applicable to its constituent VNFs.
NST_NSD002	The NSD shall include the VLDs applicable to the VLs used by the NS to interconnect its constituent NFs.
NST_NSD005	The NSD shall include the descriptors of the VNFFGs applicable to the NS.
NST_NSD007	The NSD shall support the capability to provide monitoring parameters to be tracked during the lifetime of a NS instance.
NST_NSD012	The NSD shall include a globally unique identifier for identifying each descriptor instance.
NST_VLD001	A VLD shall enable specifying the type of connectivity provided by the link (e.g. Layer 2 E-Line, E-LAN or E-Tree, or Layer 3).

On the other hand, in [12] the OSM End User Advisory Group identifies a set of challenges related with the adoption of the abovementioned specifications. Nevertheless, the document highlights their relevance to describe the needs and requirements of NFV stakeholders, stating the commitment of the OSM community to keep their data models aligned with IFA specifications.

To help VxF/NSD developers to validate and use the YANG models, a set of tools are freely available:

- **Python YANG validator:** An extensible YANG validator and converter written in python, Available on Github under the ISC License: <https://github.com/mbj4668/pyang> (last access: June 2018).
- **Pyangbind:** A plugin for the python YANG validator that creates python bindings for a YANG model. Available at Github under Apache license: <https://github.com/robshakir/pyangbind> (last access: June 2018).
- **Python YANG model extraction tool:** Xym is a simple utility for extracting YANG modules from files. Available on Github under the BSD License: <https://github.com/xym-tool/xym> (last access: June 2018).
- **LibYang:** is a YANG data modelling language parser and toolkit written (and providing API) in C. Available on Github under the BSD License: <https://github.com/CESNET/libyang> (last access: June 2018).
- **YANG schema validator** is available online at <http://www.yangvalidator.com/> (last access: June 2018).

4 Repositories

4.1 Portal repository

The portal repository and its internal model have been extensively presented in Section 2.

The portal repository mainly:

- holds information about User data and roles later retrieved from other repositories like Keystone (see 4.4).
- holds information about VxF, experiment metadata.
- Utilizes the OSM repository/catalogues via the OSM API.
- Makes public and available for download VxF, NSD archives.
- Categorizes items.

Figure 9 displays the class model which contains all the core entities of the portal repository.

4.2 The OSM Catalogues

The UI of OSM differentiates two logical catalogues, supporting the management of NSs and VNFs. This management procedures can be addressed through a specific plugin, the Composer, which is an OSM tool that supports the design and development of VNF and network service descriptors. Analogously to the Launchpad (see section 3.1), the Composer is part of User Interface (UI) within the OSM plugin model framework.

The Composer module provides an easy and intuitive way to enable design-time operations, such as the creation and export of VNF/NS packages, the CRUD (create, read, update and delete) operations on the files which are contained within the packages, and rendering the visual display layout of VNFs and NSs, including the details of the descriptors written in YAML format.

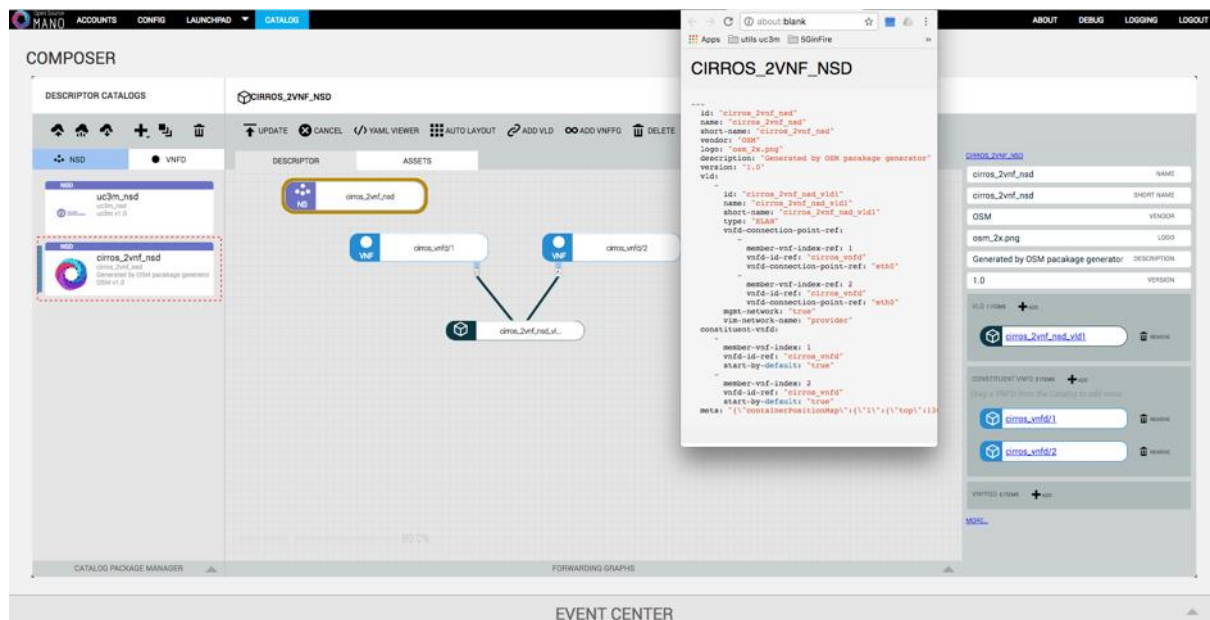


Figure 49 Composer

Figure 49 shows an example of the information displayed by the Composer, corresponding to a NS, through the graphical user interface. In this figure, we can also appreciate the division that the Composer makes between VNFD and NSD catalogues to manage the design of both separately. This screen also presents a number of buttons (below the “Descriptor catalogs” tag in Figure 46) that give access to the different tools that the composer provides for the management of packages, such as onboarding a catalogue package, export catalogue item(s), add NSs/VNFDs, among others. We want to highlight that the operation to onboard a NS/VNF package is provided to authorized 5GinFIRE entities through the portal, being the functionality provided by the OSM Composer at the disposal of the OSM site admin (e.g., to onboard packages without accessing the 5GinFIRE portal).

Finally, we want to highlight that all the code responsible for the Composer functionality is available in the OSM-UI repository².

4.3 Utilizing the OSM API

The OSM API is a REST service which provides all required methods for orchestration.

4.3.1 OSM TWO

This API provides access to 6 main categories of operations:

4.3.1.1 NS and VNF Package Management

REST wrapper for the NS and VNF package service. Provides methods for onboarding, updating and downloading service and virtual network function packages.

4.3.1.2 VNF Descriptor Management

REST wrapper for VNFD management provides methods for onboarding, updating, querying, and deleting a VNF descriptor through the `vnfd:vnfd-catalog`.

4.3.1.3 VNF Lifecycle Management

REST wrapper for the VNF lifecycle management service. Provides methods for querying a VNF and retrieving VNF records from the `vnfr:vnfr-catalog`.

4.3.1.4 Network Service Descriptor Management

REST wrapper for the NSD service (`nsd:nsd-catalog`). Provides methods for onboarding, updating, querying, and deleting a network service descriptor.

4.3.1.5 Network Service Lifecycle Management

REST wrapper for network service lifecycle management. Provides methods for instantiating, updating, finding, and terminating a network service (NS). Also provides methods for creating, updating, listing, and deleting or VNF forwarding graph (VNFFG).

Further information and documentation can be found here:

<https://open.riftio.com/documentation/riftware/4.4/a/api/orchestration/orchestration-api.html>

² Composer repository in OSM release TWO (last Access: Nov. 2017):

<https://osm.etsi.org/gitweb/?p=osm/UI.git;a=tree;f=skyquake/plugins/composer;h=6c50fe1162421f02ccb17f6017d1f86860954bda;hb=aed6500508197a8d64a41c795e25f5be05ddf930>

4.3.2 OSM FOUR and NBI

In OSM FOUR the API changed drastically, is called North Bound Interface and now is standardized via SOL005 [13]. The portal will implement the RESTful protocols specification for the Os-Ma-nfvo Reference Point

The portal will utilize:

4.3.2.1 NS and VNF Package Management

REST wrapper for the NS and VNF package service via resource Descriptors. Provides methods for onboarding, updating and downloading service and virtual network function packages.

4.3.2.2 VNF Descriptor Management

REST wrapper for VNFD descriptor management provides methods for onboarding, updating, querying, and deleting a VNF descriptor through the Individual resource Descriptors and Content

4.3.2.3 VNF Lifecycle Management

REST wrapper for the VNF lifecycle management service. Provides methods for querying a VNF and retrieving VNF records via resource Descriptors.

4.3.2.4 Network Service Descriptor Management

REST wrapper for the **NSD** service (via resource Descriptors). Provides methods for onboarding, updating, querying, and deleting a network service descriptor.

4.3.2.5 Network Service Lifecycle Management

REST wrapper for network service lifecycle management. Provides methods for instantiating, updating, finding, and terminating a network service (NS). Also provides methods for creating, updating, listing, and deleting or VNF forwarding graph (VNFFG). This will be implemented via the NS Lifecycle Management interface

4.4 Keystone repository

4.4.1 Introduction

This section presents information related to the Keystone Repository. At the beginning of the section, we explain the requirements for such repository, with use cases and process flows. After this brief explanation, we explain how the integration is possible with the 5GinFIRE portal. At the end of the section, we provide an overview of project status and the incorporation of the project into OSM.

Keystone is a component of the OpenStack project that we are reusing. The Keystone repository is a component of the security framework proposed in Deliverable D4.1. This repository acts as an Identity Provider for the 5GinFIRE portal and OSM. Using Keystone allows us to create isolation in the OSM, meaning that each user of OSM (except administrators) has sole access to its allocated resources.

4.4.2 Requirements

The main driver for the creation of this repository is to centralize user access control to the 5GinFIRE portal and OSM. This centralization enables the experimenters to use a single access credential to access both systems. The requirements are specified in the following list:

- Centralize user access control
- Centralize user access roles

- Provide user access control to 5GinFIRE portal
- Provide user access control to OSM
- Reuse the Keystone component from OpenStack
- User management
- Role management
- Project management

The centralization of user access control is necessary to provide a single identity provider that makes user management simpler and easier between portal and OSM, removing the burden of managing multiple accounts for the same entity.

The centralization of user access roles increases security, allowing both the 5GinFIRE portal and OSM to have access to the same information regarding an entity, enabling better decisions, avoiding duplication of information about an entity and disallowing conflicting roles.

It is necessary to allow both the 5GinFIRE portal and OSM to use the Keystone repository to authenticate users because the repository is the identity provider.

We propose the reuse of Keystone, a component already existing in the architecture of OpenStack. Keystone allows the use of other technologies widely used to store user information, such as LDAP, Kerberos or Microsoft's Active Directory.

User and role management is another requirement to make possible the creation, retrieval, update and deletion of that information.

OSM lacks the notion of users and projects, and this is important for user and project isolation. Isolation is needed when multiple users are working in the system, and one's work must not interfere with the next. This isolation should allow users to work on the same system without affecting one another and can be achieved using the notion of projects and project management at the level of OSM.

4.4.3 Keystone deployment

To be able to use Keystone as a component in the portal, there must be a way to deploy it. The solution is to deploy the Keystone component inside a Docker container and its database in a separate container, thus providing isolation and a finer grained control on deployed components. This approach also eases deployment since it makes deployments repeatable and automatic. To deploy both containers, it is used another tool called Docker Compose, which deploys the Keystone and all the other components needed.

The dockerized version of Keystone can be found at:

https://github.com/5GinFIRE/dockerized_keystone

4.4.4 Use cases

The systems in the use cases are:

- 5GinFIRE portal
- OSM

When an experimenter or an administrator interacts with these systems, they use Keystone to fulfil the process.

In the following table, it is possible to see all the use cases supported by Keystone system.

Table 8 Use cases supported by the Keystone repository

Number	Title	Actor(s)
01	Create user	5GinFIRE portal
02	Delete user	5GinFIRE portal
03	Update user information	5GinFIRE portal
04	Retrieve user information	5GinFIRE portal, OSM
05	Create project	5GinFIRE
06	Delete project	5GinFIRE
07	Update project information	5GinFIRE
08	Retrieve project information	5GinFIRE, OSM
09	Create role	5GinFIRE
10	Delete role	5GinFIRE
11	Update role information	5GinFIRE
12	Add role to user in project	5GinFIRE
13	Remove role from user in project	5GinFIRE
14	Retrieve user's roles in project	5GinFIRE, OSM
15	Authenticate user	5GinFIRE, OSM

These use cases support the requirements prior mentioned while adding the actors that perform the actions. In the next subsection, we go into further detail on how these use cases transform into process flows.

4.4.5 Process flows

In this section, we are going to observe the process flows of the most representative cases. The process flows detailed below belong to the following use cases:

- Create user (No. 01)
- Create project (No. 05)
- Create role (No. 09)
- Add role to user in project (No. 12)
- Retrieve user's roles in project (No. 14)
- Authenticate user (No.15)

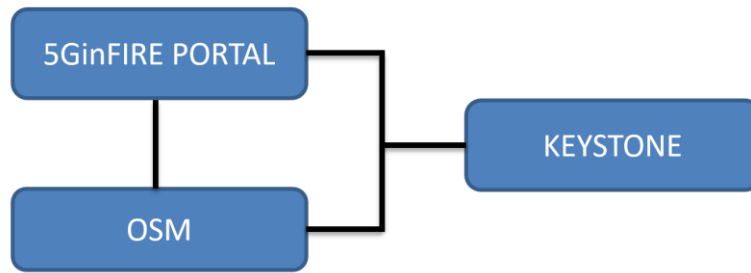


Figure 50 Component interconnection diagram

4.4.5.1 Create user process flow

In this process flow, we specify how the interaction between the actors and system occurs and which the possible outcomes of creating a user are.

The following ordered list enumerates the steps in the process flow.

1.	The 5GinFIRE portal sends a request to Keystone repository to create a user
2.	The Keystone repository receives the request to create a user
3.	The Keystone repository verifies if the user identifier exists
4a.	If the user identifier does not exist, the Keystone repository creates the user (go to step 5a)
4b.	If the user identifier exists, the process fails (go to step 5b)
4c.	If an error occurs during the previous verification (step 3), the process fails (go to step 5c)
5a.	The Keystone repository responds that the process execution succeeded
5b.	The Keystone repository responds that the process execution failed because a user with the same user identifier already exists
5c.	The Keystone repository responds that the process execution failed because an error occurred

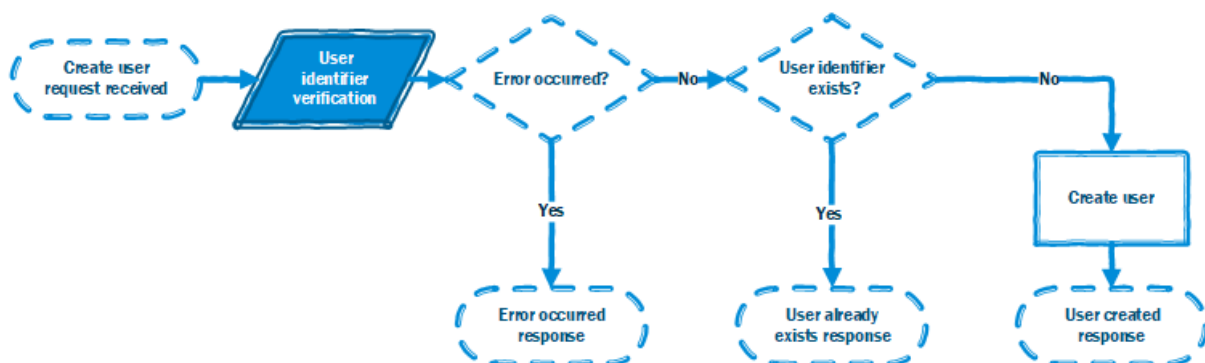


Figure 51 Create a user process flow

4.4.5.2 Create project process flow

In this process flow, we specify how the interaction between the actors and system occurs and what are the possible outcomes of creating a project.

The following ordered list enumerates the steps in the process flow.

1.	The 5GinFIRE portal sends a request to Keystone repository to create a project
2.	The Keystone repository receives the request to create a project
3.	The Keystone repository verifies if the project identifier exists
4a.	If the project identifier does not exist, the Keystone repository creates the project (go to step 5a)
4b.	If the project identifier exists, the process fails (go to step 5b)
4c.	If an error occurs during the previous verification (step 3), the process fails (go to step 5c)
5a.	The Keystone repository responds that the process execution succeeded
5b.	The Keystone repository responds that the process execution failed because a project with the same project identifier already exists
5c.	The Keystone repository responds that the process execution failed because an error occurred

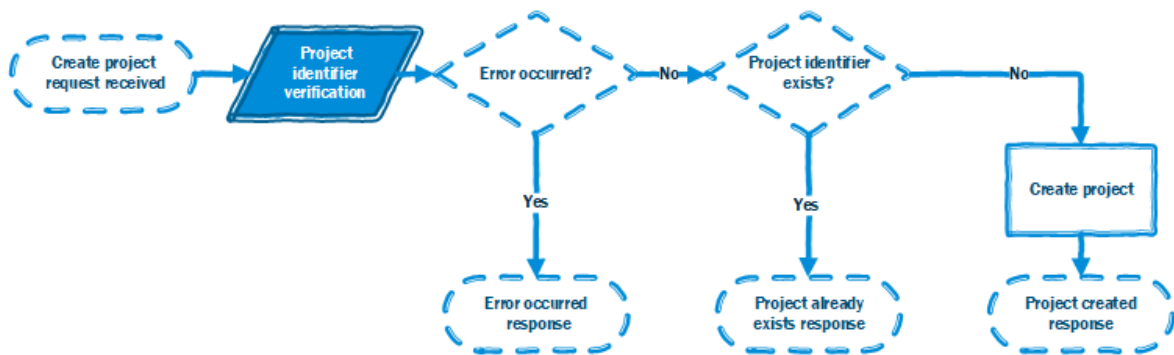


Figure 52 Create a project process flow

4.4.5.3 Create role process flow

In this process flow, we specify how the interaction between the actors and system occurs and what are the possible outcomes of creating a role.

The following ordered list enumerates the steps in the process flow.

1.	The 5GinFIRE portal sends a request to Keystone repository to create a role
2.	The Keystone repository receives the request to create a role

3.	The Keystone repository verifies if the role identifier already exists
4a.	If the role identifier does not exist, the Keystone repository creates the role (go to step 5a)
4b.	If the role identifier exists, the process fails (go to step 5b)
4c.	If an error occurs during the previous verification (step 3), the process fails (go to step 5c)
5a.	The Keystone repository responds that the process execution succeeded
5b.	The Keystone repository responds that the process execution failed because a role with the same role identifier already exists
5c.	The Keystone repository responds that the process execution failed because an error occurred

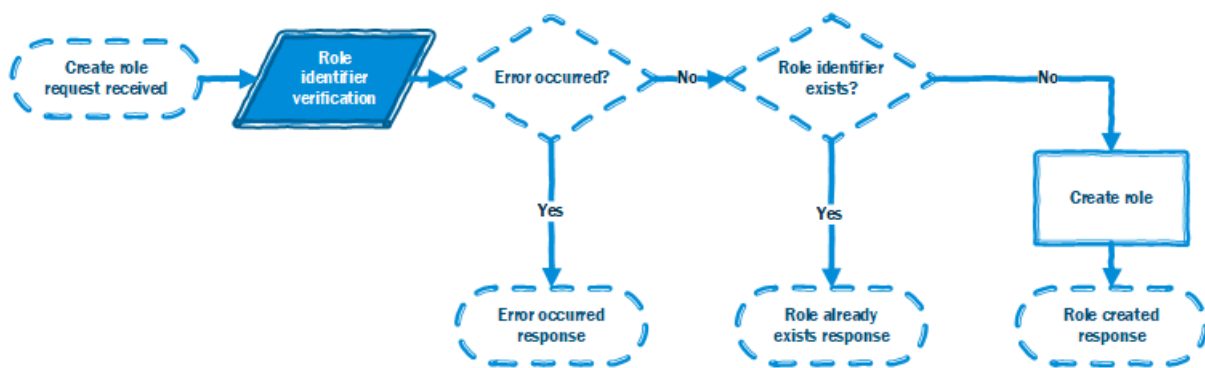


Figure 53 Create a role process flow

4.4.5.4 Add role to user in a project process flow

In this process flow, we specify how the interaction between the actors and system occurs and what are the possible outcomes of adding a role to a user in a project.

The following ordered list enumerates the steps in the process flow.

1.	The 5GinFIRE portal sends a request to Keystone repository to add a role to a user in a project
2.	The Keystone repository receives the request to add a role to a user in a project
3.	The Keystone repository verifies if the user identifier exists
4a.	If the user identifier exists, the process continues (go to step 5)
4b.	If the user identifier does not exist, the process fails (go to step 9b)
4c.	If an error occurs during the previous verification (step 3), the process fails (go to step 9e)
5.	The Keystone repository verifies if the project identifier exists

6a.	If the project identifier exists, the process continues (go to step 7)
6b.	If the project identifier does not exist, the process fails (go to step 9c)
6c.	If an error occurs during the previous verification (step 5), the process fails (go to step 9e)
7.	The Keystone repository verifies if the role identifier exists
8a.	If the role identifier exists, the Keystone repository adds the role to the user in that project (go to step 9a)
8b.	If the role identifier does not exist, the process fails (go to step 9d)
8c.	If an error occurs during the previous verification (step 7), the process fails (go to step 9e)
9a.	The Keystone repository responds that the process execution succeeded
9b.	The Keystone repository responds that the process execution failed because the user identifier does not exist
9c.	The Keystone repository responds that the process execution failed because the project identifier does not exist
9d.	The Keystone repository responds that the process execution failed because the role identifier does not exist
9e.	The Keystone repository responds that the process execution failed because an error occurred

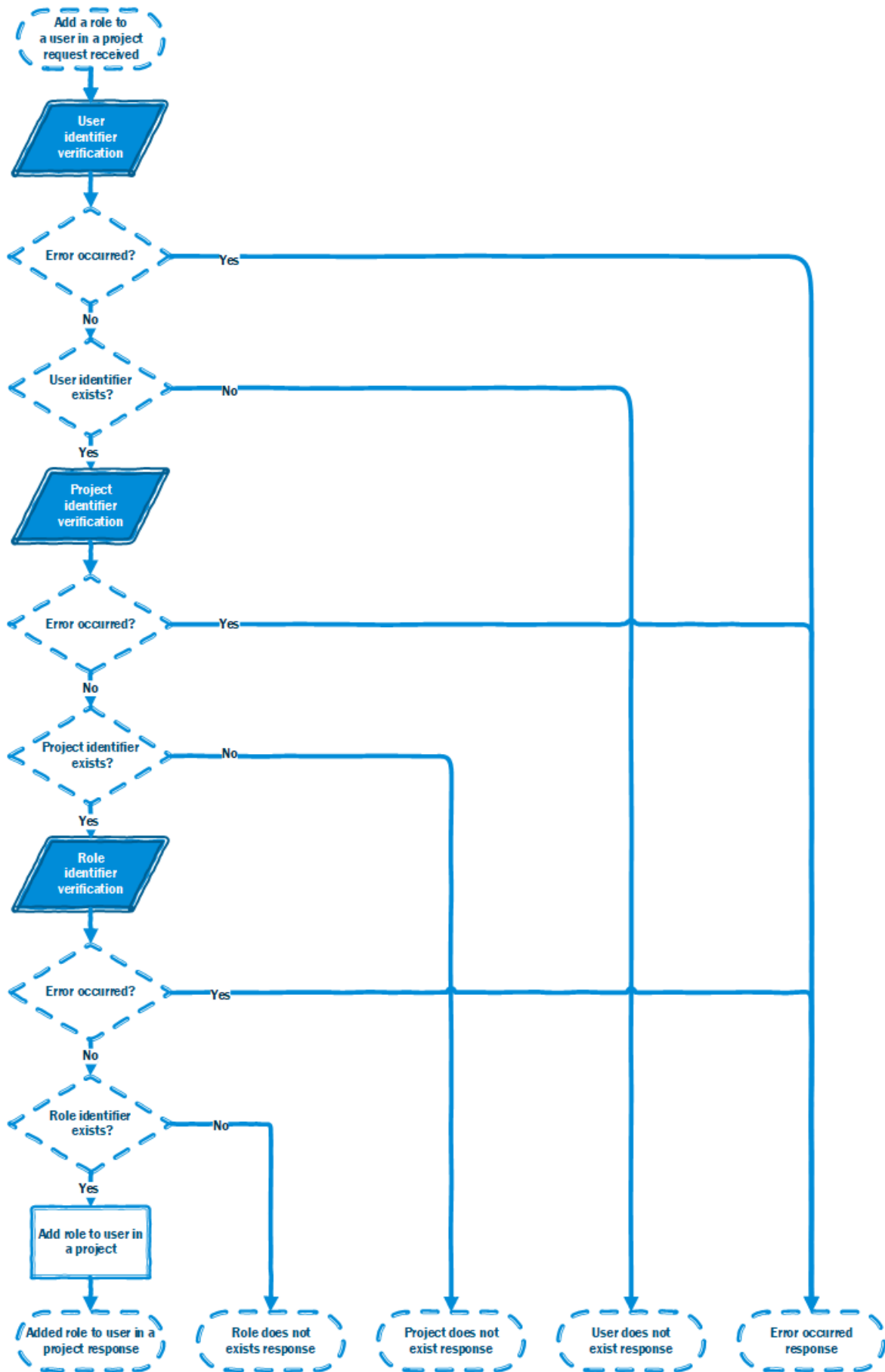


Figure 54 Add a role to a user in a project process flow

4.4.5.5 Retrieve user's roles in a project process flow

In this process flow, we specify how the interaction between the actors and system occurs and what are the possible outcomes of retrieving user's roles in a project.

The following ordered list enumerates the steps in the process flow.

1.	The 5GinFIRE portal sends a request to Keystone repository to retrieve the user's roles in a project
2.	The Keystone repository receives the request to retrieve the user's roles in a project
3.	The Keystone repository verifies if the user identifier exists
4a.	If the user identifier exists, the process continues (go to step 5)
4b.	If the user identifier does not exist, the process fails (go to step 7b)
4c.	If an error occurs during the previous verification (step 3), the process fails (go to step 7d)
5.	The Keystone repository verifies if the project identifier exists
6a.	If the project identifier exists, the Keystone repository retrieves the user's role in that project (go to step 7a)
6b.	If the project identifier does not exist, the process fails (go to step 7c)
6c.	If an error occurs during the previous verification (step 5), the process fails (go to step 7d)
7a.	The Keystone repository responds that the process execution succeeded and attaches to the response the user's roles in that project
7b.	The Keystone repository responds that the process execution failed because the user identifier does not exist
7c.	The Keystone repository responds that the process execution failed because the project identifier does not exist
7d.	The Keystone repository responds that the process execution failed because an error occurred

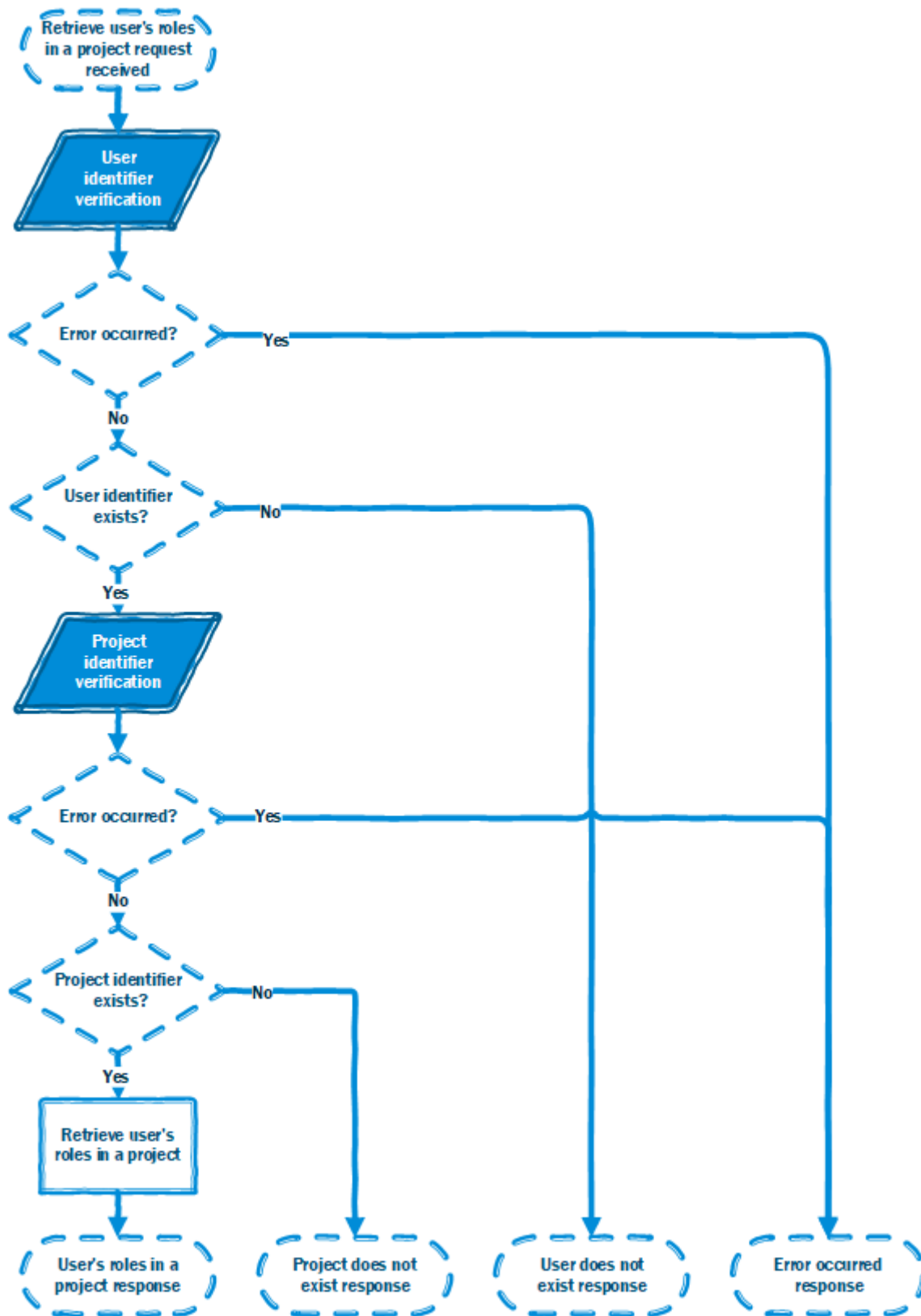


Figure 55 Retrieve user's roles in a project process flow

4.4.5.6 Authenticate user process flow

In this process flow, we specify how the interaction between the actors and system occurs and what the possible outcomes of creating a user are.

The following ordered list enumerates the steps in the process flow.

1.	The 5GinFIRE portal sends a request to Keystone repository to authenticate a user
2.	The Keystone repository receives the request to authenticate a user
3.	The Keystone repository verifies if the user identifier exists
4a.	If the user identifier exists, the process continues (go to step 5)
4b.	If the user identifier does not exist, the process fails (go to step 11b)
4c.	If an error occurs during the previous verification (step 3), the process fails (go to step 11c)
5.	The Keystone repository verifies if the user's credentials match with the ones stored
6a.	If the user's credentials match, the process continues (go to step 7)
6b.	If the user's credentials do not match, the process fails (go to step 11b)
6c.	If an error occurs during the previous verification (step 5), the process fails (go to step 11c)
7.	The Keystone repository creates an authentication token
8a.	If the creation of the authentication token succeeded, the process continues (go to step 9)
8b.	If an error occurs in the previous operation (step 7), the process fails (go to step 11c)
9.	The Keystone repository retrieves the projects that the user belongs to
10a.	If the projects retrieval succeeds, the process continues (go to step 11a)
10b.	If an error occurs in the previous operation (step 9), the process fails (go to step 11c)
11a.	The Keystone repository responds that the process execution succeeded and attaches to that message the user's authentication token and projects that he belongs to
11b.	The Keystone repository responds that the process execution failed because the user's credentials are wrong
11c.	The Keystone repository responds that the process execution failed because an error occurred

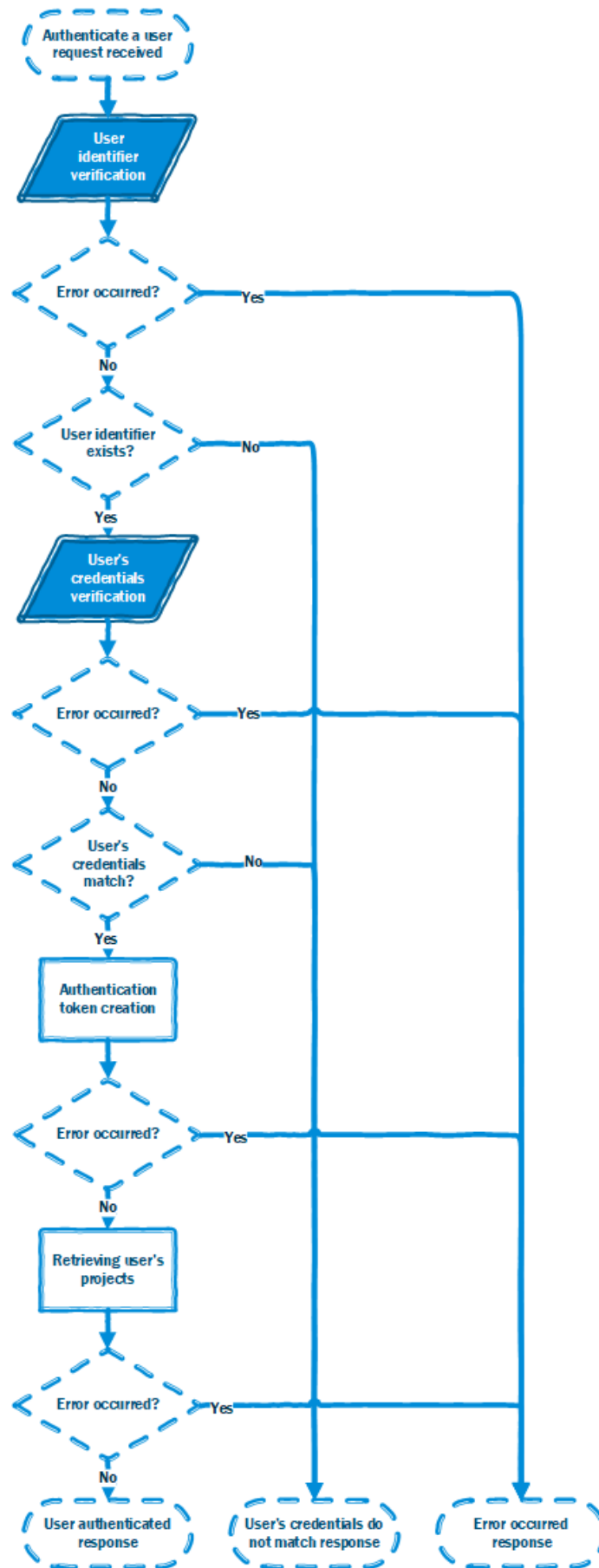


Figure 56 Authenticate a user process flow

4.4.6 Integration with 5GinFIRE portal

The 5GinFIRE portal connects to the Keystone Repository using its HTTP API. The 5GinFIRE portal using this API can fulfil all of the requirements presented above. The 5GinFIRE portal is responsible for managing the users, managing the projects and attributing the right roles to the users in the correct projects.

In the following table, we present the roles needed in the Keystone repository to manage the different users.

Table 9 User roles

No	Role	Role description
1	Owner	Is attributed to each user that creates an experiment in that project. An owner can manage its project
2	Administrator	Is attributed to each administrator so they can manage all the projects

We then separate users into projects that they own. Each project represents an experiment. Using this roles and projects system, we can isolate experiments. This isolation is translated to OSM, meaning that each experimenter has sole access to the resources allocated to him.

4.4.7 Project status

We are now in the phase of starting to integrate the Keystone repository into the 5GinFIRE portal. The OSM isolation using Keystone as an identity provider will be implemented in future versions of OSM. This work is part of the security framework for OSM described in Deliverable 4.1. The Keystone repository will be hosted with OSM, so in the current deployment, it is going to be located in the 5TONIC lab.

5 FIRE Integration

The goal of WP3 with FIRE will be to federate and accept users of the FED4FIRE+ project which maintains all the users of the FIRE federation. This integration will allow an existing user with FIRE credentials to request an experiment Deployment. For testbeds integration see WP5 D5.1.

5.1 integration with Fed4FIRE Identity provider



Figure 57 portal and Fed4FIRE+ integration

Figure 57 displays the high level integration with Fed4FIRE Identity provider via the Portal API Backend.

To allow Fed4FIRE users start using the 5GinFIRE facility we build a solution around the SFA Wrapper service provided by FIRE. Thus, we build a so called Aggregate Manager around our portal to allow to access resources of the portal via SFA client tools.

To invoke operations in the AM the user needs to present a credential. This credential is used for authentication and authorization. It contains information about the permissions that a certain user has, so the AM can determine if the user is allowed to perform the requested operation. (see <https://wiki.confine-project.eu/sfa:implementation>)

The credentials are issued by a SFA entity Slice Authority. The Community-Lab SFA wrapper does not provide a Slice Authority entity. Instead, it uses credentials issued by a federated authority. To get a valid credential you need to login at a slice authority federated with Community-Lab and retrieve the credential, which will be signed by this Slice Authority. If the Slice Authority is federated with Community-Lab, the the AM of Community-Lab will accept the credential issued by this Slice Authority.

Currently the AM of Community-Lab accepts credentials from the following authorities:

- iMinds Virtual Wall 2 (see Get a Fed4FIRE account)
- PlanetLab Europe (PlanetLab EU - <http://www.planet-lab.eu/>)
- Fed4FIRE Portal (Fed4FIRE - <https://portal.fed4fire.eu/>)

The wrapper is implemented as a software layer on top of the REST API of the testbed controller. It is based on the SFAWrap software, which helps testbeds to develop a SFA interface. The wrapper uses a Python library a predefined flavour called Federica and an XML RPC utility to interact with the 5GinFIRE Portal. Moreover, the XML RPC utility transforms the portal API to SFA XML.

SFAWrap (<http://svn.planet-lab.org/wiki/SFATutorial>) is a free software that allows to federate a testbed into the emerging SFA-based global federation of testbeds. The software package provides a set of general components to help any testbed to expose a SFA-compliant interface, which integrate

the generic part of the code. It also includes a set of skeleton classes to be implemented for each particular testbed, which integrate the testbed-specific part.

The generic part of the code implements all the standard processing related to SFA that does not depend on the testbed. It basically consists of modules that implement SFA interface and XML RPC servers, credential management and validation, generic SFA registry and Rspec management.

It also includes a generic command-line client tool for SFA: sfi.py.

The 5GinFIRE implemented SFA Wrapper allows users from the FIRE federation domain to use the 5GinFIRE facility, that is to request deployments. These users are external, they do not belong to the 5GinFIRE domain. Therefore, the users interacting with the Wrapper do not have an account in the 5GinFIRE portal. They have accounts from other FIRE trusted authorities.

To interact with the testbed and perform any action, a user has to be logged in with a 5GinFIRE account. This implies the need of mapping somehow the external users interacting with the Wrapper into the 5GinFIRE domain. To achieve this purpose the Wrapper uses a very simple solution: it uses a user service account "sfa5ginfire" already registered in the portal for all the external users. This way, the wrapper can log in with this generic user account and perform any requested action in behalf of the external users. However, when requested a deployment the FIRE user will need to provide some contact details, since through the SFA wrapper there is no way of identifying the user.

To use the SFAWrapper we use the FEDERICA flavour and thus we setup the url like this:

SFA_FEDERICA_URL = <http://150.140.184.212:13000/5ginfireportal/services/api/repo/sfawrap>

In the portal we created a specific access point only authorized to be used by the SFAWrapper. This endpoint responds to the commands of listing resources and requesting slices in terms of SFA. In our case and mapping to 5GinFIRE aspects and terminology we have the following:

5.1.1 Listing resources

When SFA tools request SFA listing resources, the 5GinFIRE portal responds with the list of available VNFs and public NSDs. Here is a response of the advertised RSPEC:

```
<rspec xmlns="http://www.protogeni.net/resources/rspec/2" type="advertisement" valid_until="2020-05-20T16:03:57+03:00" generated="2018-06-20T16:03:57+03:00">
  <statistics call="ListResources">
    <aggregate status="success" name="5ginfire" elapsed="0.1" />
  </statistics>
  <network name="5ginfire">
    <node
      component_manager_id="urn:publicid:IDN+5ginfire+authority+cm"
      component_id="urn:publicid:IDN+upatras:p2e+node+67ce4a0f-8c41-40e9-a2ad-bd2f8a783fe8"
      component_name="lab_vnfd" site_id="urn:publicid:IDN+5ginfire:p2e+authority+sa">
      <displayname>lab_vnfd</displayname>
      <package>https://portal.5ginfire.eu/5ginfireportal/services/api/repo/packages/67ce4a0f-8c41-40e9-a2ad-bd2f8a783fe8/lab_vnfd.tar.gz</package>
      <location country="unknown" longitude="21.7885" latitude="38.2845" />
      <description>A test lab_vnfd</description>
      <lease from="Fri Jun 15 12:49:32 EEST 2018" until="Fri Jun 15 12:49:32 EEST 2018">false</lease>
    </node>
    <node
      component_manager_id="urn:publicid:IDN+5ginfire+authority+cm"
      component_id="urn:publicid:IDN+upatras:p2' body: 'e+node+eb950234-20e9-4342-9dbb-4286450a80f9"
      component_name="ttest_vnfd" site_id="urn:publicid:IDN+5ginfire:p2e+authority+sa">
      <displayname>ttest_vnfd</displayname>
      <package>http://150.140.242.66:13000/5ginfireportal/services/api/repo/packages/eb950234-20e9-4342-9dbb-4286450a80f9/ttest_vnfd.tar.gz</package>
      <location country="unknown" longitude="21.7885" latitude="38.2845" />
      <description>my test vnf</description>
      <lease from="Fri Jun 15 12:49:32 EEST 2018" until="Fri Jun 15 12:49:32 EEST 2018">false</lease>
    </node>
```

```

    <node
      component_manager_id="urn:publicid:IDN+5ginfire+authority+cm"
      component_id="urn:publicid:IDN+upatras:p2e+node+aa7b52c7-2f24-46f9-bb3d-644a6ca8b2fa"
      component_name="rift_ping_vnf" site_id="urn:publicid:IDN+5ginfire:p2e+authority+sa">
        <displayname>rift_ping_vnf</displayname>
        <package>https://portal.5ginfire.eu/5ginfireportal/services/api/repo/packages/aa7b52c7-2f24-
46f9-bb3d-644a6ca8b2fa/ping_vnf.tar.gz</package>
        <location country="unknown" longitude="21.7885" latitude="38.2845" />
        <description>ping_vnf</description>
        <lease from="Fri Jun 15 12:49:32 EEST 2018" until="Fri Jun 15 12:49:32 EEST
2018">false</lease>
      </node>
    <node
      component_manager_id="urn:publicid:IDN+5ginfire+authority+cm"
      component_id="urn:publicid:IDN+upatras:p2e+node+a7805355-e9fc-48ce-b9b3-657d8bf5f241"
      component_name="rift_pong_vnf" site_id="urn:publicid:IDN+5ginfire:p2e+authority+sa">
        <displayname>rift_pong_vnf</displayname>
        <package>https://portal.5ginfire.eu/5ginfireportal/services/api/repo/packages/a7805355-e9fc-
48ce-b9b3-657d8bf5f241/pong_vnf.tar.gz</package>
        <location country="unknown" longitude="21.7885" latitude="38.2845" />
        <description>pong_vnf</description>
        <lease from="Fri Jun 15 12:49:32 EEST 2018" until="Fri Jun 15 12:49:32 EEST
2018">false</lease>
      </node>
    <node
      component_manager_id="urn:publicid:IDN+5ginfire+authority+cm"
      component_id="urn:publicid:IDN+upatras:p2e+node+3a96acd7-517f-4f7a-a0fe-2ecf9ac4d151"
      component_name="cirros_vnfd" site_id="urn:publicid:IDN+5ginfire:p2e+authority+sa">
        <displayname>cirros_vnfd</displayname>
        <package>https://portal.5ginfire.eu/5ginfireportal/services/api/repo/packages/3a96acd7-517f-
4f7a-a0fe-2ecf9ac4d151/cirros_vnf.tar.gz</package>
        <location country="unknown" longitude="21.7885" latitude="38.2845" />
        <description>cirros_vnfd</description>
        <lease from="Fri Jun 15 12:49:32 EEST 2018" until="Fri Jun 15 12:49:32 EEST
2018">false</lease>
      </node>
    <node
      component_manager_id="urn:publicid:IDN+5ginfire+authority+cm"
      component_id="urn:publicid:IDN+upatras:p2e+node+35cd7277-d8e4-42dd-8fdb-5329c3b8d70f"
      component_name="lab_vnfd" site_id="urn:publicid:IDN+5ginfire:p2e+authority+sa">
        <displayname>lab_vnfd</displayname>
        <package>http://192.168.2.237:13000/5ginfireportal/services/api/repo/packages/35cd7277-d8e4-
42dd-8fdb-5329c3b8d70f/lab_vnfd.tar.gz</package>
        <location country="unknown" longitude="21.7885" latitude="38.2845" />
        <description>lab_vnfd</description>
        <lease from="Fri Jun 15 12:49:32 EEST 2018" until="Fri Jun 15 12:49:32 EEST
2018">false</lease>
      </node>
  </network>
</rspec>

```

The provided RSpec has similarities with most published RSpecs of other FIRE testbeds. However, there are some extra elements that may help tools to display information to experimenters. For example, there are description elements, display name of the VNF, NSD resource and the package location. Within the RSpec there is a lease element. It has a slot of 1 day. That is the next available day that this resource is available.

5.1.2 Slice request

Slice request in 5GinFIRE is mapped to deploy an NSD on a specific schedule. For example, if an experimenter via a tool wants to reserve the rift_ping_pong_ns on the corresponding lease element one should write:

```
<lease from="2018-05-20T15:03:57+03:00" until="2018-05-28T15:03:57+03:00">true</lease>
```

This means that the node will be reserved for several days from 20/5/2018 15:00 UTC. Of course together with the lease, the requester should configure the requested NSD. As an example, assume that we want to reserve the rift_ping_pong_ns . The following request can be prepared:

```

<rspec type="request" >
  <network name="5ginfire">
    <node
      component_manager_id="urn:publicid:IDN+5ginfire+authority+cm"
      component_id="urn:publicid:IDN+upatras:p2e+node+35cd7277-d8e4-42dd-8fdb-5329c3b8d70f"
      component_name="rift_ping_pong_ns" site_id="urn:publicid:IDN+5ginfire:p2e+authority+sa">

```

```
<nsd>rift_ping_pong_ns</nsd>
<nsd_package>http://mywebsite.com/nsd.tar.gz</nsd_package>
<settings>
  <setting name="experimenter_name">Tranoris</setting>
  <setting name="experimenter_email">tranoris@ece.upatras.gr</setting>
  <setting name="experimenter_organization">University of Patras</setting>
  <setting name="experiment_name">A test deployment </setting>
  <setting name="experiment_description">A test NSD deployment request</setting>
  <setting name="rift_ping_vnf">ITaV</setting>
  <setting name="rift_pong_vnf ">Bristol</setting>
</settings>
<lease from="2018-05-20T15:03:57+03:00" until="2018-05-28T15:03:57+03:00">true</lease>
</node>
</network>
</rspec>
```

Since there is no way for 5GinFIRE through SFA to know the Identity of the experimenter, the experimenter needs to identify himself. He needs also to define the requested NSD as well as where each Constituent VNF will be placed. Within the `<nsd_package>` element the experimenter can define the location of his own NSD package to be deployed. The package though should contain VNFs that are available and valid in 5GinFIRE repository. AS soon as there is such a request the NSD should be checked for validity.

The above is subject to change and enhanced while the infrastructure moves to OSM FOUR and is enhanced with new testbeds, VNF, NSDs and features.

6 Automated Validation process

An important aspect when onboarding experiments is to validate VNF's submitted by experiments. Through the portal the experimenter is able to upload VNF descriptor files, but these are not validated by the portal itself. In order to validate the submitted files, whether they actually are VNF descriptors and whether they can actually be deployed to the testbeds using OSM, a Continuous Integration (CI) Tool (currently Jenkins) is integrated, that is able to validate the VNF descriptor (through Linting) and Charm (using JuJu build and test tools) and can go a step further and deploy to a staging testbed through OSM (as a test package). This process is illustrated in the following diagram.

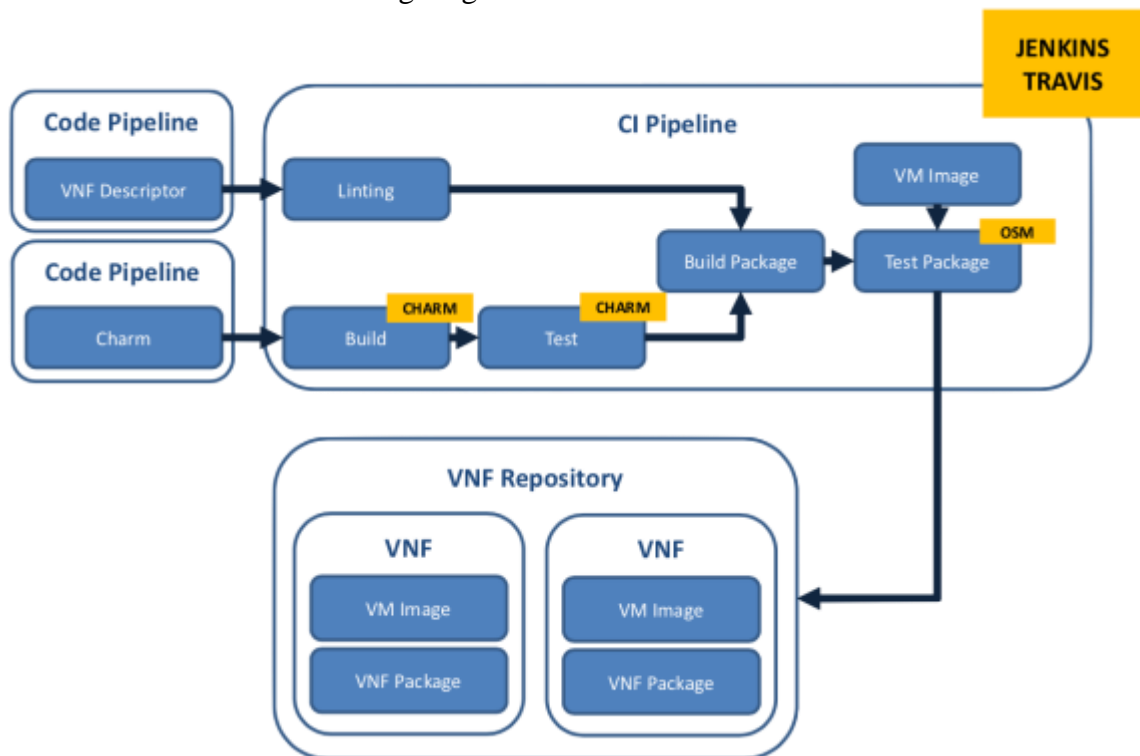


Figure 56 Automated Validation of VNFs

After a successful validation of the submitted VNF, the VNF can be stored in the VNF Repository.

The integration of this solution requires some new interfaces with the Portal and VNF Repository. The validation process is triggered by VNF Repository whenever a new VNF is received. Upon completion of the CI Pipeline the Portal must be notified of the results (Success/Failure) and potential feedback to the user (such as reason it failed).

Currently a Jenkins CI server has been installed and configured. The Jenkins CI server is available at <http://ci.5ginfire.eu>. Exposing the CI tool enables testers to directly address test failures and correctly addressing any bugs detected.

In the future we intend to add a functional test step and implement an ID system to keep track of the jobs through the portal.

6.1 Synergy with EU H2020 project 5GTANGO for Validation

5GTANGO (<https://5gtango.eu/>) project aims to enable the flexible programmability of 5G networks. The main objectives of the project are:

- Reduce time to market for networked services by shortening service development cycle and by qualifying network services to be adopted
- Reduce entry barrier to third party developers and support the creation and composition of VNFs and NSs

It provides the following software assets:

- An NFV-enabled SERVICE DEVELOPMENT KIT (SDK)
- **ASTORE PLATFORM** with advanced **VALIDATION AND VERIFICATION MECHANISMS** for VNFs/Network Services qualification (including 3rd party contributions), and
- A modular **SERVICE PLATFORM** with an innovative **ORCHESTRATOR** in order to bridge the gap between business needs and network operational management systems

The high-level architecture of the offered integrated platform is shown in Figure 56 displaying the 3 main assets as well as the way they are integrated together:

- **Service Development KIT** providing the developer with a powerful tool set to develop, test and evaluate NFV-based Network Services. It comes with an emulator providing a light-weight local rapid-prototyping environment for debugging purpose.
- **V&V platform** is in charge of managing verification and validation tests, test planning and execution, test result analysis and storage. It is independent from the service deployment platform and pluggable to the service platforms. It is also able to manage resource orchestration for performing multi tests.
- **Service platform** provides service and function orchestration features, plus all the needed complementary and supporting features, like slice management, policy and SLA management, user access management and infrastructure adapter.

A public catalogue where VNFs/NSs packages are stored interconnect the three above assets: a **developer** submits the VNF/NS developed and tested using the SDK to the public catalogue; the **V&V platform** takes the submitted package, plans and executes the tests upon the package deployed in a testing environment, collects the test results and associates them to the package to be stored back to the public catalogue; the **Service Platform** deploys only the VNF/NSs marked validated by the V&V on the production environment.

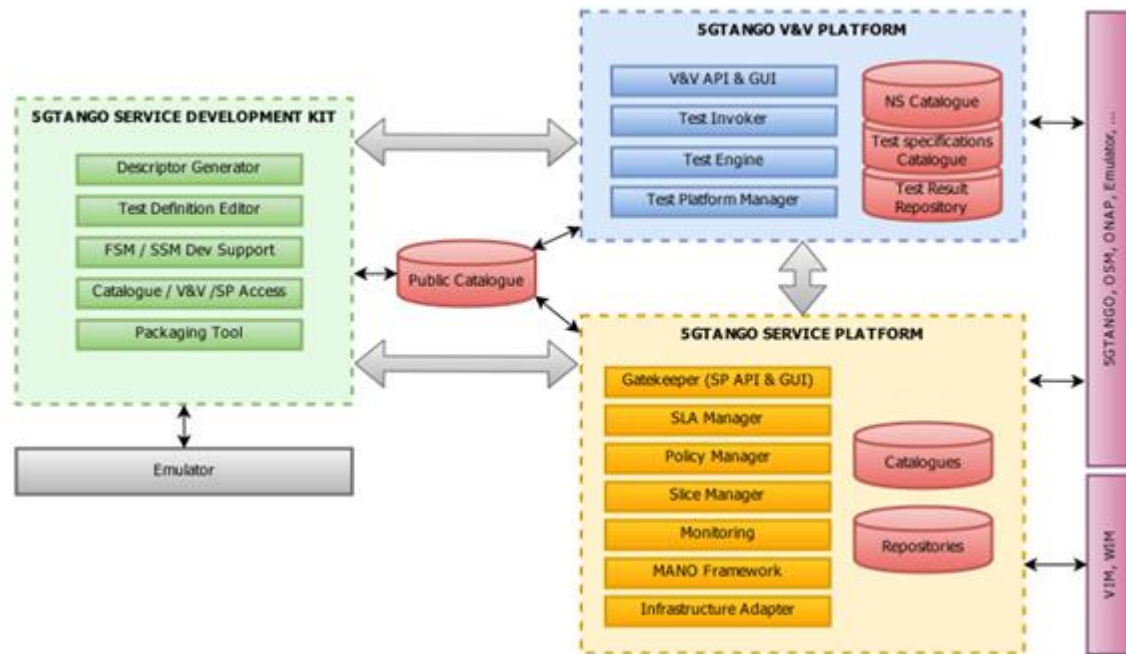


Figure 56 5GTANGO platform high-level architecture

The validation of VNF/NSs in 5GTANGO exists in all the 3 main modules with different emphasis:

- In the **SDK**, the tests and the validation have the objective of “**supporting the developer**”. The validation focuses on the correctness of the Descriptor (VNFD, NSD) syntax, as well as the composition of the package. Tools are available to let developer to create unit tests or other tests for debugging purpose.
- In the **V&V**, the tests and the validation and verification are for objective of “**functional and non-functional features verification**” from the service platform’s perspective to ensure a correct and coherent behaviour of the VNF/NSs to be deployed. The tests are developed independently from the developer therefore they have more credibility in terms of results.
- In the **Service Platform**, the tests and validation are more for objective of “**monitoring and resource orchestration**” of the running VNF/NSs. It collects the runtime parameters and regarding the pre-defined SLA (service level agreement) and policies, it allocates the appropriate amount of resources in order to ensure the whole service platform works correctly.

6.1.1.1 Potential of the 5GTANGO validation approach for 5GinFIRE platform

The validation and verification workflow of 5GTANGO can potentially serve the 5GinFIRE validation framework from several perspectives:

- For the descriptor and package validation, the two projects can check each other’s features to see if there are missing features in its own implementation and to decide whether to include them.
- For the validation and verification of the VNF/NS from the platform’s perspective, as 5GTANGO designs its V&V platform to be independent and pluggable to several service platform, it is adaptable to the 5GinFIRE platform with some configurations.
- For the runtime monitoring and enhanced orchestration features on the service platform, the 5GinFIRE platform can check the needed features that are missing for the moment, and how to include these features in the platform.
- The general workflow from 5GTANGO project can also inspire the 5GinFIRE project which is in the phase of finalizing the validation process.

7 Future enhancements/Next versions

D3.2 describes how the 5GinFIRE portal and middleware technologies continue to evolve. D3.1 described our planned integration with a Keystone service for authorization and the plan to implement authorization for Fed4Fire users with close collaboration to Fed4FIRE project, as well as Enhanced Deployment Management and Automated instantiation of the requested Experiment towards OSM via the OSM API as well as Integrations with future OSM versions.

All the above aspects have been considered and implemented since previous release as reflected in D3.2. Still there are open issues. Complementarily to the mechanisms implemented by the 5GinFIRE portal, defined in this deliverable, the evolution of the portal needs to consider the utilization of other relevant and well-established standards to support the definition and management of network and vertical-specific services. The project will carefully follow the progress on the work regarding the utilization of the Topology and Orchestration Specification for Cloud Applications (TOSCA) [14], standardized by OASIS, in NFV environments, and more importantly, the impact of this work in the normative specifications developed by ETSI NFV ISG and the availability of open source implementations.

At the time of writing, OASIS has already produced version 1.0 of a TOSCA simple profile for NFV [15]. On the other hand, the work on NFV Release 2 of the ETSI NFV ISG includes the specification of the structure and format of a VNF package as a TOSCA CSAR file. However, the work on the applicability of the TOSCA standard to NFV is still ongoing at ETSI, with the normative specification of a data model for NFV descriptors (e.g., NSDs and VNFDs) based on TOSCA still unavailable [16]. In parallel to this, an alternative direction has been taken by ETSI, currently addressing the standardization work of NFV descriptors based on YANG [9], aligned with the descriptors currently supported by OSM.

Leveraging the involvement of several 5GinFIRE partners at ETSI, the team in charge of portal development will closely follow the progress of the diverse and alternative standardization activities at ETSI NFV ISG, particularly on the specification of NFV descriptors and VNF packages, as well as the stability and maturity of related open-source implementations. This will serve to evaluate the possible adoption of TOSCA as an enabling technology of the 5GinFIRE portal.

8 References

- [1] "Resource Specification (RSpec) Documents in GENI," [Online]. Available: <http://groups.geni.net/geni/wiki/GENIExperimenter/RSpecs>.
- [2] "Portal Web Wiki," [Online]. Available: <https://github.com/5GinFIRE/eu.5ginfire.portal.web/wiki>.
- [3] "ETSI Group Specification NFV-SOL 004 V2.3.1; Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package specification;," [Online]. [Accessed 06 06 2018].
- [4] OpenJPA, "OpenJPA," [Online]. Available: <http://openjpa.apache.org/>. [Accessed 06 06 2018].
- [5] Keystone, "Keystone," [Online]. Available: <https://wiki.openstack.org/wiki/Keystone>. [Accessed 06 06 2018].
- [6] A. SHIRO, "Apache SHIRO," [Online]. Available: <https://shiro.apache.org/>. [Accessed 06 06 2018].
- [7] A. CXF, "Apache CXF," [Online]. Available: <http://cxf.apache.org/>. [Accessed 06 06 2018].
- [8] Angular, "Angular," [Online]. Available: <https://angular.io/>. [Accessed 06 06 2018].
- [9] "OSM-UI repository," [Online]. Available: Launchpad within the OSM-UI repository in OSM release TWO (last access Nov. 2017): .
- [10] ETSI, "SOL005," [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-SOL/001_099/005/02.04.01_60/gs_NFV-SOL005v020401p.pdf. [Accessed 20 06 2018].
- [11] O. I. Model, "OSM Information Model," [Online]. Available: https://osm.etsi.org/wikipub/index.php/OSM_Information_Model. [Accessed 06 06 2018].
- [12] OSM, "Open Source MANO Release TWO, OSM Information Model," [Online]. Available: https://osm.etsi.org/wikipub/images/2/26/OSM_R2_Information_Model.pdf. [Accessed 06 06 2018].
- [13] R. Enns, "NETCONF Configuration Protocol, RFC 4741, Internet Engineering Task Force (IETF)," 12 2018. [Online].
- [14] "M. Bjorklund, YANG-A Data Modeling Language for the Network Configuration Protocol (NETCONF), RFC 6020, Internet Engineering Task Force (IETF), October 2010.," [Online].
- [15] "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; VNF Descriptor and Packaging Specification, ETSI GS NFV-IFA 011 V2.4.1 (2018-02).," [Online]. Available: http://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/011/02.04.01_60/gs_NFV-IFA011v020401p.pdf. [Accessed 06 06 2018].
- [16] "Network Functions Virtualisation (NFV) Release 2; Management and Orchestration; Network Service Templates Specification, ETSI GS NFV-IFA 014 V2.4.1 (2018-02).," [Online]. Available:

http://www.etsi.org/deliver/etsi_gs/NFV-IFA/001_099/014/02.04.01_60/gs_NFV-IFA014v020401p.pdf. [Accessed 06 06 2018].

- [17] M. B. e. al., "Experience with NFV, architecture, interfaces, and information models, whitepaper prepared by the OSM End User Advisory Group, Issue 1, May 2018.," 05 2018. [Online]. [Accessed 06 06 2018].
- [18] T. S. P. f. N. F. V. (. V. 1.0., "Edited by Shitao Li and John Crandall. 11 May 2017. OASIS Committee Specification Draft 04," [Online]. Available: <http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/tosca-nfv-v1.0.html>.
- [19] "Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on TOSCA specification (status: standard not ready for download);," 2017. [Online]. Available: https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=49491. [Accessed 06 06 2018].