



D3.1 - 5G Experimentation portal, tools and middleware

Editor:	Christos Tranoris, University of Patras	
Deliverable nature:	Report (R)	
Dissemination level:	Public (PU)	
Date: planned actual	30/11/2017	8 December 2017
Version No. of pages	1.1	69
Keywords:	5G Portal, OSM API, VxF/NSD management	

Abstract

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 of this deliverable. 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 and Task 3.4 - VxF Open repository.

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

Document title: D3.1 - 5G Experimentation portal, tools and middleware

Editor: Christos Tranoris, University of Patras

Work-package leader: Christos Tranoris, University of Patras

Copyright notice

2017 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 endpoint where experimentation requests will be accepted
- A portal in terms of web application that end users can subscribe, manage experiments, browse our 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, therefore to have all our infrastructure browsable by other FIRE catalogs 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 the 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 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

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. Finally Section 5 provides initial design details about the FIRE integration.

List of authors

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

Table of Contents

1	Introduction.....	11
1.1	Objective of this document.....	11
1.2	Structure of this report.....	11
2	The 5GinFIRE portal.....	12
2.1	Introduction.....	12
2.2	Requirements.....	12
2.2.1	Requirements and the supported actors	12
2.2.2	High level processes	15
2.3	Architecture.....	17
2.4	Design and implementation.....	19
2.4.1	The Portal API backend	19
2.4.2	The web frontend	26
2.5	Requirements Implementation.....	27
2.5.1	Public user Web interface/Landing page	27
2.5.2	VxF developer user interface description	29
2.5.3	Experimenter user interface description	32
2.5.4	Services administrator user interface description	36
2.6	Deployment details.....	43
2.7	Delivery plan.....	45
2.8	Code repositories.....	45
2.9	Licensing.....	46
3	Middleware tools.....	47
3.1	Admin tool and provisioning: Launchpad (OSM TWO/THREE).....	47
3.2	YANG models and packaging for VxF/NSD.....	49
3.2.1	Overview of the YANG models	49
3.2.2	Developer tools for YANG modelling	50
3.2.3	Functional requirements for VNF & NSD Packaging specification	50
3.2.4	Package pre-validation in the 5GinFire portal	51
4	Repositories.....	52
4.1	Portal repository.....	52
4.2	The OSM Catalogues.....	52
4.3	Utilizing the OSM API.....	53
4.3.1	NS and VNF Package Management	53
4.3.2	VNF Descriptor Management	53
4.3.3	VNF Lifecycle Management	53

4.3.4	Network Service Descriptor Management	53
4.3.5	Network Service Lifecycle Management	53
4.4	Keystone repository	53
4.4.1	Introduction	53
4.4.2	Requirements	54
4.4.3	Use cases	54
4.4.4	Process flows	55
4.4.5	Integration with 5GinFIRE portal	65
4.4.6	Project status	65
5	FIRE Integration	66
6	Future enhancements/Next versions	68
7	References	69

List of figures and tables

Figure 1 upload a VxF archive. Onboarding and make published by administrator	16
Figure 2 Uploading an Experiment/NSD archive and onboarding by administrator.....	16
Figure 3 Request a new experiment deployment.....	17
Figure 4 5GinFIRE portal architecture	18
Figure 5 The architecture of the Portal API.....	18
Figure 6 Web frontend architecture.....	19
Figure 7 Main package diagram	20
Figure 8 details for the definitions of VxF and Experiment metadata.....	21
Figure 9 Core model class diagram	22
Figure 10 Model of a deployment descriptor	22
Figure 11 Diagram of the class PortalRepositoryAPI which implements the RESTful API	23
Figure 12 Web Front end components.....	27
Figure 13 The 5GinFIRE portal landing page.....	28
Figure 14 Sign up to portal.....	28
Figure 15 Available experiments, VxF developer login page. (Use case #2010).....	29
Figure 16 Available VxFs. (Use case #2070 or #3030).....	30
Figure 17 VxFs Management. (Use case #3020)	30
Figure 18 VxF upload. (Use case #3010 or #2040)	31
Figure 19 VxF creation. (Use case #3010)	31
Figure 20 VxF details. (Use case #3030).....	32
Figure 21 Available experiments, experimenter login page. (Use case #2010).....	33
Figure 22 Available Deployed experiments (Use case #2010).....	33
Figure 23 Experiment deployment creation (Use case #2020).....	34
Figure 24 Experiments management (Use case #2050).....	35
Figure 25 Experiment creation (Use case #2060).....	35
Figure 26 Users management (Use case #5010).....	36
Figure 27 User details. (Use case #5010)	37
Figure 28 User account creation. (Use case #5010).....	37
Figure 29 Admin tab submenu	38
Figure 30 Extra fields for admin role about registered experiments (Use case #4010 and #5030).....	38
Figure 31 Extra fields for admin role about registered VxFs. (Use case #3020).....	39
Figure 32 Deploy a validated experiment (Use case #5030).....	39
Figure 33 Available categories.....	40
Figure 34 Category creation.....	40
Figure 35 Available MANO platforms	41
Figure 36 MANO platform creation	41
Figure 37 Available MANO providers.....	42
Figure 38 MANO provider creation	42
Figure 39 Available target Infrastructures	43
Figure 40 Deployment diagram of 5GinFIRE	44
Figure 41 Internal mappings of deployed containers.....	44
Figure 42 OSM Release THREE Northbound API	47
Figure 43 Launchpad: Dashboard	48
Figure 44 Launchpad: Compute Topology.....	48
Figure 45 Launchpad: Instantiate	49
Figure 46 Composer	52
Figure 47 Component interconnection diagram	56
Figure 48 Create a user process flow	56
Figure 49 Create a project process flow.....	57
Figure 50 Create a role process flow.....	58
Figure 51 Add a role to a user in a project process flow	60
Figure 52 Retrieve user's roles in a project process flow.....	62
Figure 53 Authenticate a user process flow	64
Figure 54 portal and Fed4FIRE+ integration	66
Figure 55 Interaction diagram between the portal and Fed4FIRE IDP.....	67

Table 1 5GinFIRE portal high level usage scenarios/requirements 12

Table 2 Detailed requirements for the Web front end component 13

Table 3 Public API (does not need authentication) 24

Table 4 Admin API (needs authentication)..... 25

Table 5 Deliver plan for Version 1 45

Table 6 Code repositories related with the portal 45

Table 7 Use cases supported by the Keystone repository 55

Table 8 User roles..... 65

Abbreviations

5G	5th Generation
AGPL	GNU Affero General Public License
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
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
NFV	Network Function Virtualization
NFVI	NFV Infrastructure
NFVO	NFV Orchestrator
NoSQL	Not only SQL
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
RSU	Road Side Unit
SBI	Southbound interfaces
SDN	Software Defined Network
SFC	Service Function Chaining
TOSCA	Topology and Orchestration Specification for Cloud Applications
VIM	Virtual Infrastructure Management
VNF	Virtual network function
VNFM	VNF Manager
VXLAN	Virtual Extensible LAN

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 (D3.2, 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

1.2 Structure of this report

This 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. Finally Section 5 provides initial design details about the FIRE integration.

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 clarity we also present them here with less details:

- Experimenter: can upload Experiments in terms of NSDs and request the deployment of an experiment over the 5GinFIRE infrastructure
- VxF Developer: can upload VxF archives
- Testbed provider: can register a target infrastructure
- Services administrator: 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 <https://github.com/5GinFIRE/eu.5ginfire.portal.web/wiki> :

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 frontend 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 experiment description	The experiment will contain a name, a description and optionally a logo. It can be also added to a category.

		<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	PRIVATE is default for an experiment descriptor If PUBLIC it means that it can be available in the repository to be used by any user If PRIVATE it can only be used by its owner (e.g. experimenter)
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	The user selects one of his private or public experiment descriptors in order to deploy it. A deployment request can include time requirements (e.g. list of proposed dates and duration of the experiment);
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	This allows to VxF developers to deploy a VxF descriptor package and define some metadata (name, logo, description, target infrastructures etc). The VxF initially is NOT Published and NOT CERTIFIED

16	A user can be assigned with multiple roles by admin	Blocked by 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	

2.2.2 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 a VxF

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

- A VxF developer submits a VxF archive (he can later manage if needed some metadata)
- 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

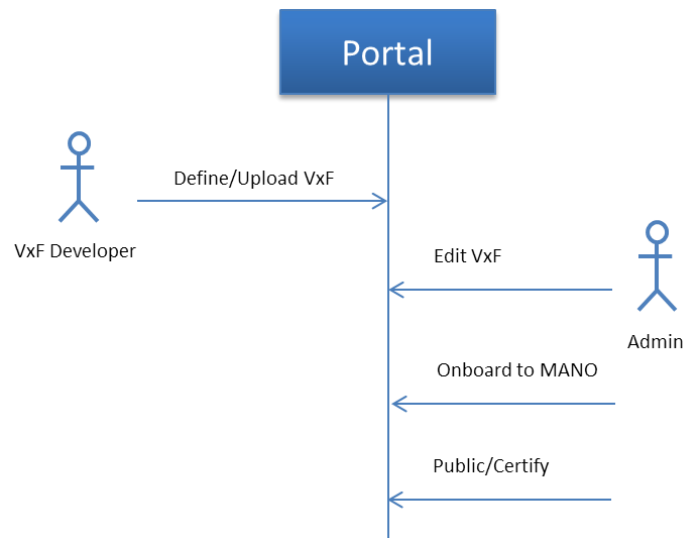


Figure 1 upload a VxF archive. Onboarding and make published by administrator

2.2.2.2 Uploading an Experiment Descriptor/NSD

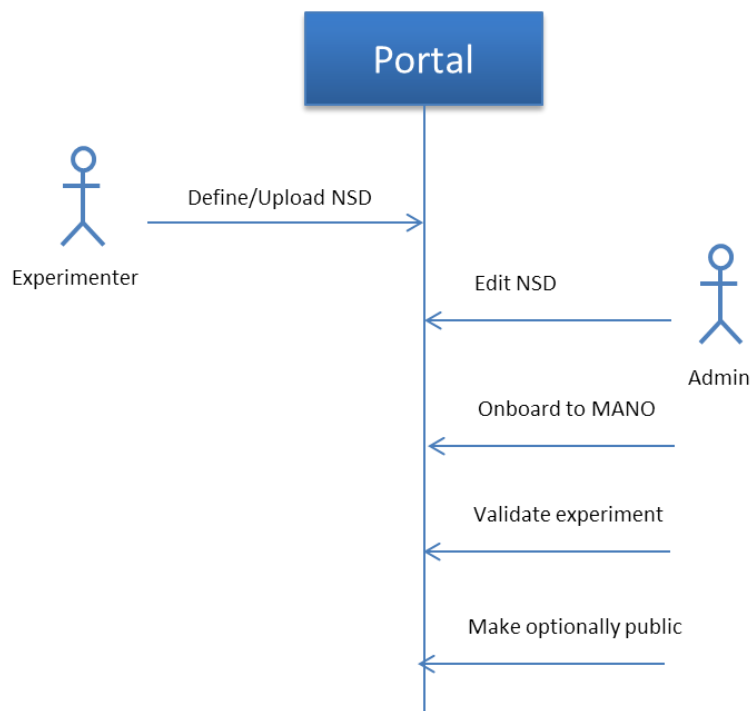


Figure 2 Uploading an Experiment/NSD archive and onboarding by administrator

During this process (see Figure 2) the following occurs:

- An experimenter submits an experiment in terms of an NSD archive (he can later manage if needed some metadata)
- 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

2.2.2.3 Request a new experiment deployment

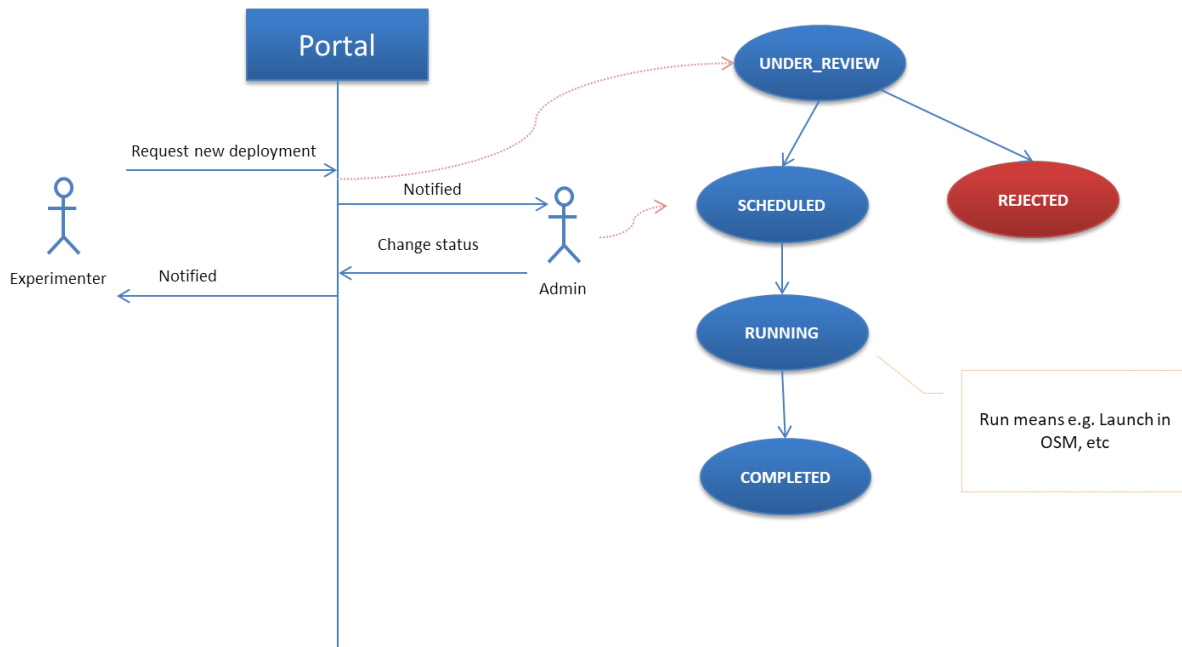


Figure 3 Request a new experiment deployment

During this process (see Figure 3) 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.

2.3 Architecture

In D2.1 Section 4.5 we presented a high level architecture of the portal. Figure 4 displays a detailed architecture of the portal together with the related interfaces. The portal consists of two main components: The portal web frontend and the portal API backend. The Web front end communicates with the backend via a RESTful API.

Moreover, the portal backend implements: An interface towards the OSM API in order to on-board VxF and NSDs to the OSM repository and get related information, and an interface towards an identity provider and a portal repository that reflects the artefacts of the OSM repository

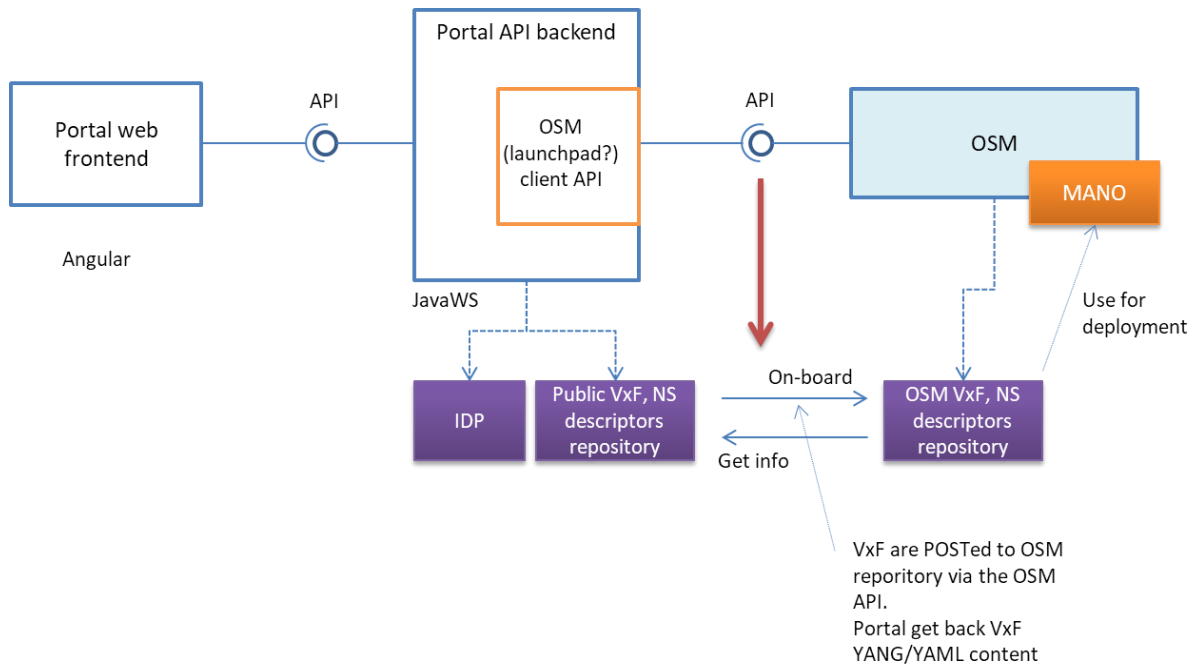


Figure 4 5GinFIRE portal architecture

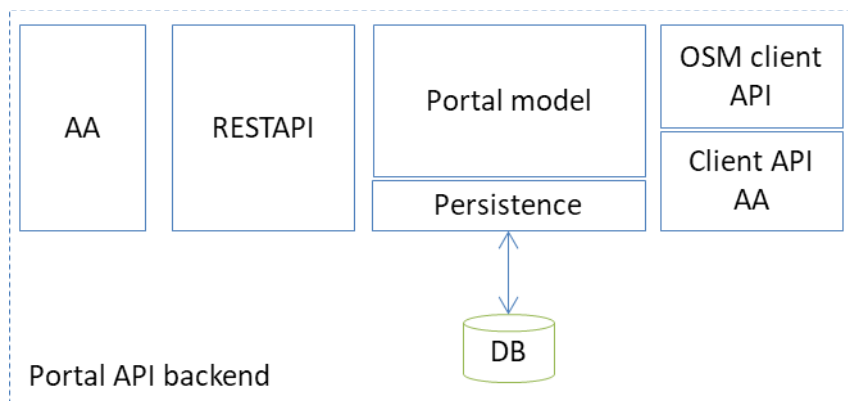


Figure 5 The architecture of the Portal API

Figure 5 displays the architecture of the portal API. It consists of the following components:

- Portal model: contains the model of entities, their definitions and associations of the portal entities like users, VxF/experiment metadata, categories, etc.
- Persistence and DB: a persistence layer to keep entities permanently available through a database system.
- AA: Authentication and authorization mechanism(s) to allow access to the portal API
- RESTAPI: implementation of the portal API server
- OSM client API: implementation of a client that communicates with OSM via the OSM API
- Client API AA: implementation of a client that is capable of communicating with another AA service(s) to authenticate/authorize users

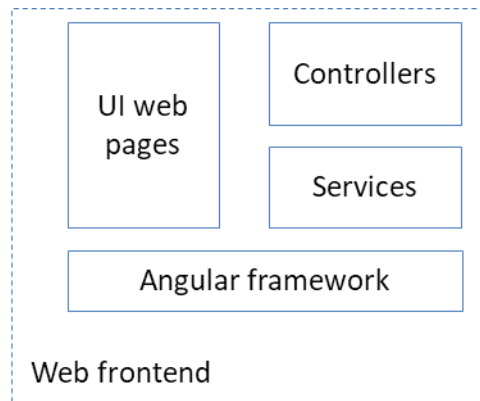


Figure 6 Web frontend architecture

Figure 6 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 present the static view of the backend API portal design in terms of class diagrams.

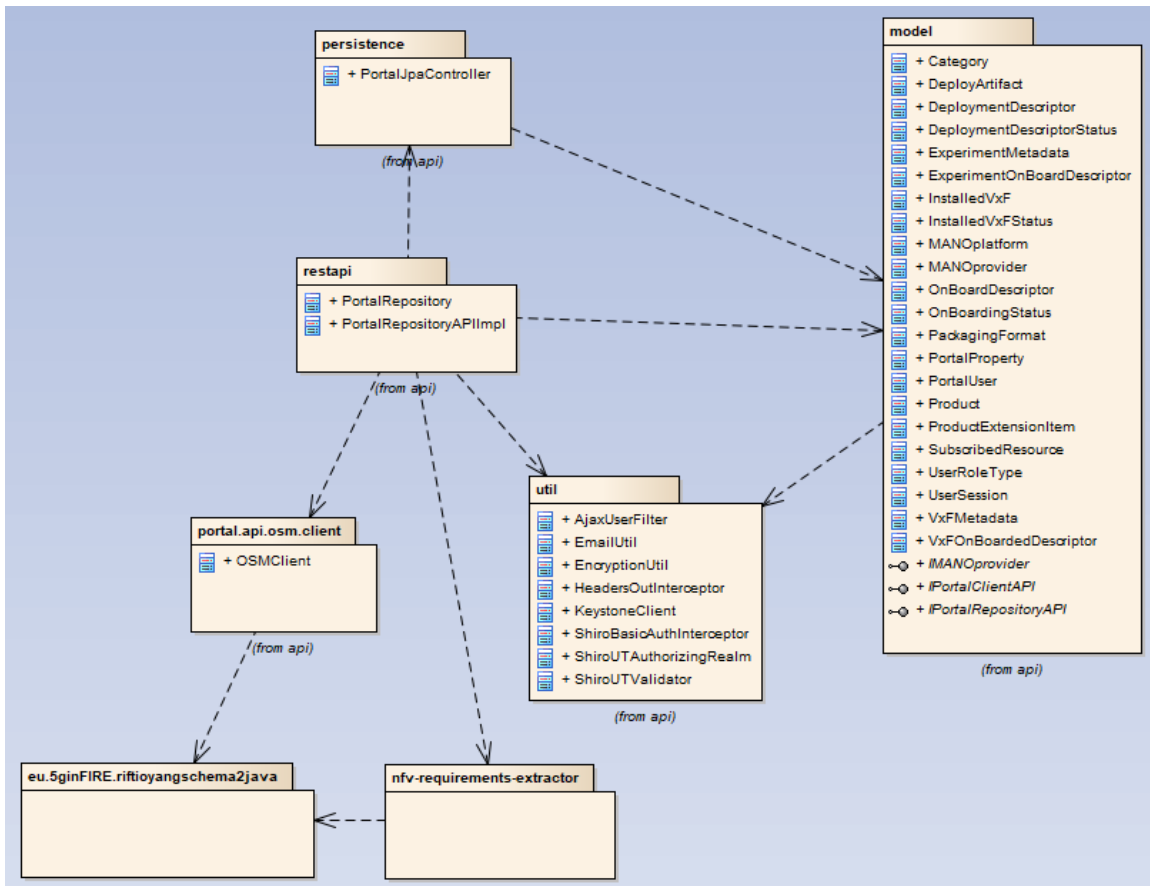


Figure 7 Main package diagram

Figure 7 displays the main package diagram of the backend API. There are four 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
- **eu.5ginFIRE.riftioyangschema2java**: contains classes that implement the OSM API model in Java, based on the OSM YANG model
- **nfv-requirements-extractor**: utility package for extracting and reading the VxF/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

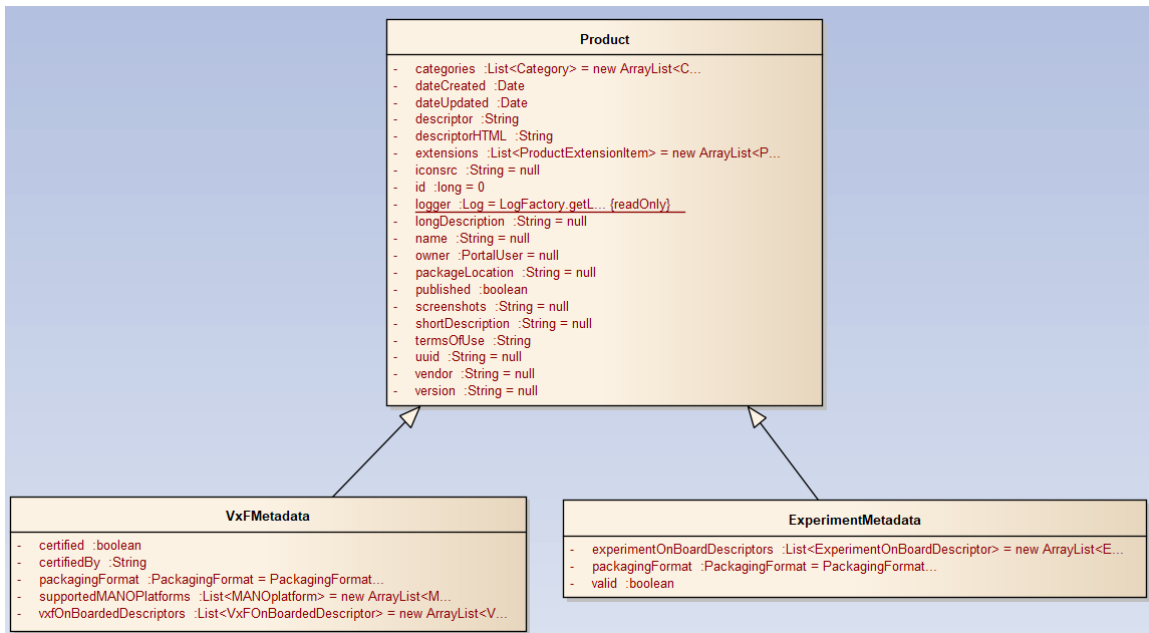


Figure 8 details for the definitions of VxF and Experiment metadata

Figure 8 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 9 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 10 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 11 displays the relationships for PortalRepositoryAPIImpl class which implements the RESTful API.

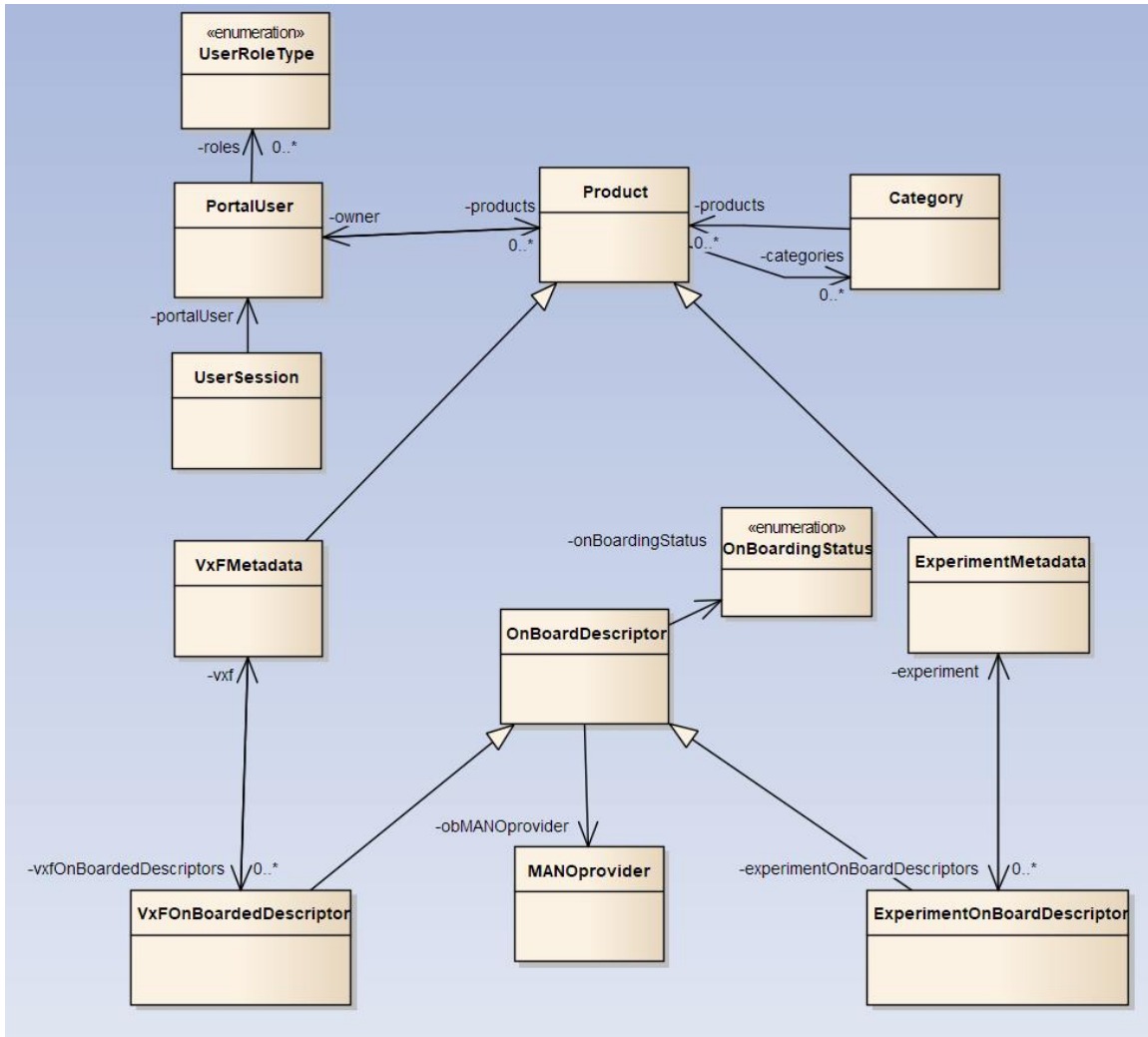


Figure 9 Core model class diagram

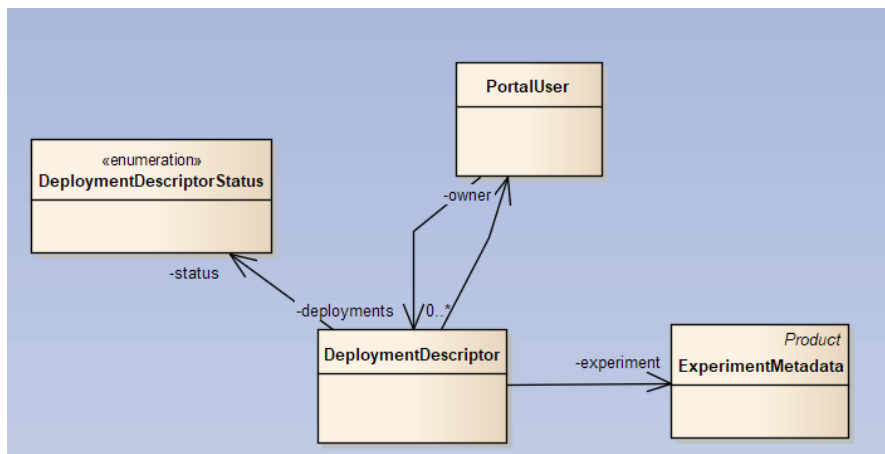


Figure 10 Model of a deployment descriptor

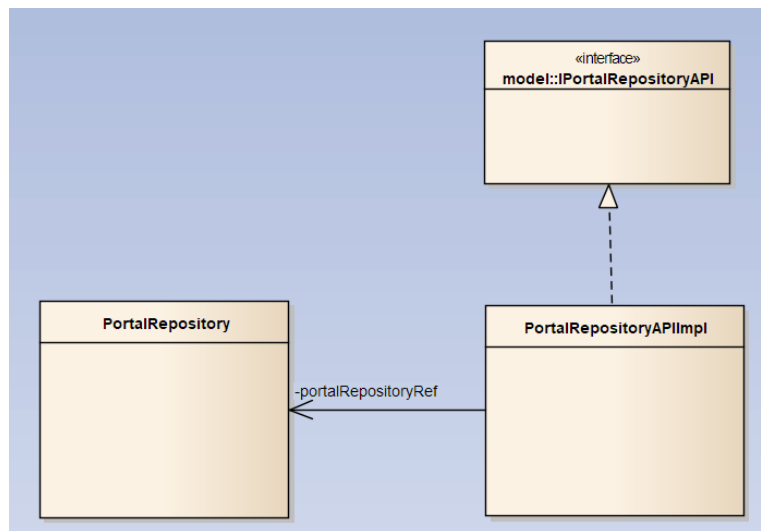


Figure 11 Diagram of the class PortalRepositoryAPI which implements the RESTful API

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, since our portal will be under <https://portal.5ginfire.eu> you can request towards: 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/>

Here is 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 the endpoints of the API that in order to be called the client must be authenticated.

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 present 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 frontend

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

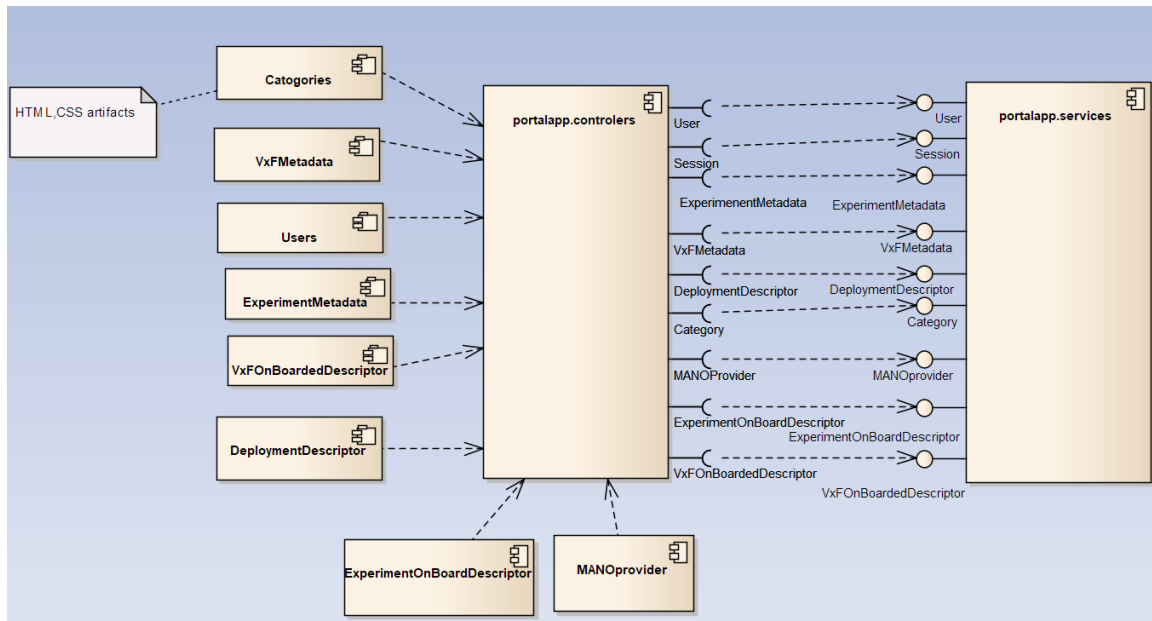


Figure 12 Web Front end components

Figure 12 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 remote (CRUD) objects - equivalents of PUT/GET/POST/DELETE on the REST API services.

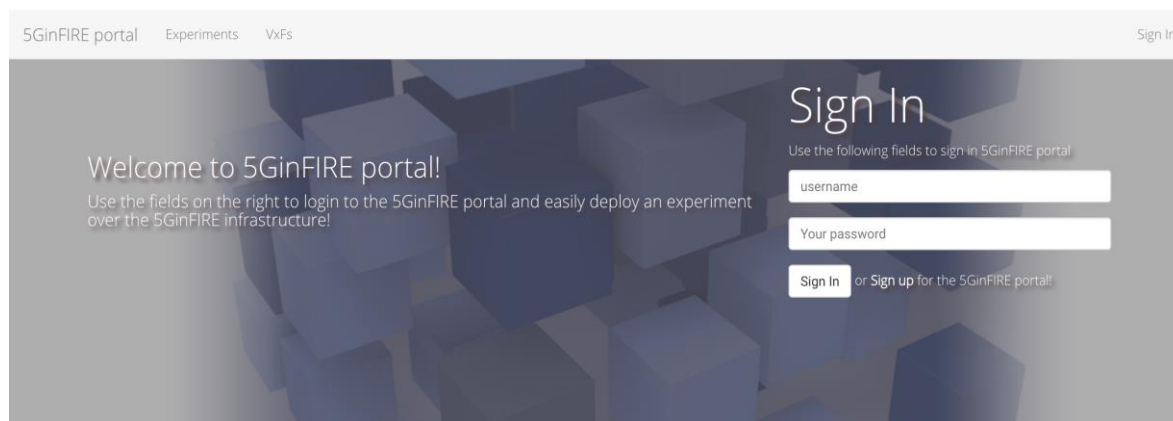
Controllers interface the services and implement any logic of the front end. All other HTML artifact components (Categories, Users, VxFMetadata, etc. - on the left side of Figure 12) depend on the controllers.

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 13 The 5GinFIRE portal landing page

At the menu on the top we can see experiments and Vxfs tabs which redirect users to the pages in Figure 14 and Figure 15, 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 16. After that the user inserts its details in the appropriate fields and then when is submitting its request administrator is responsible for its approval.

The image shows the sign-up page of the 5GinFIRE portal. At the top, there is a navigation bar with '5GinFIRE portal', 'Experiments', and 'Vxfs' on the left, and a 'Sign In' link on the right. The main content area has a light gray background. It features a large 'Sign up' heading followed by 'for the 5GinFIRE portal'. Below this are several input fields: 'Name Lastname' (placeholder: name...), 'Username' (placeholder: username...), 'Password' (placeholder: password...), 'Re-type Password' (placeholder: confirm password), 'e-mail' (placeholder: email), and 'Organization'. At the bottom of the form is a 'Register' button.

Figure 14 Sign up to portal

Next, we continue by describing the user interface for every authorized user role separately. At the start we review the landing page which is shown up when a user is logged in the portal and then we walk through all the available menus provided by the user interface.

2.5.2 VxF developer user interface description

The first user role we are going to describe is the VxF developer. In the next figure you can see the landing page for this role:

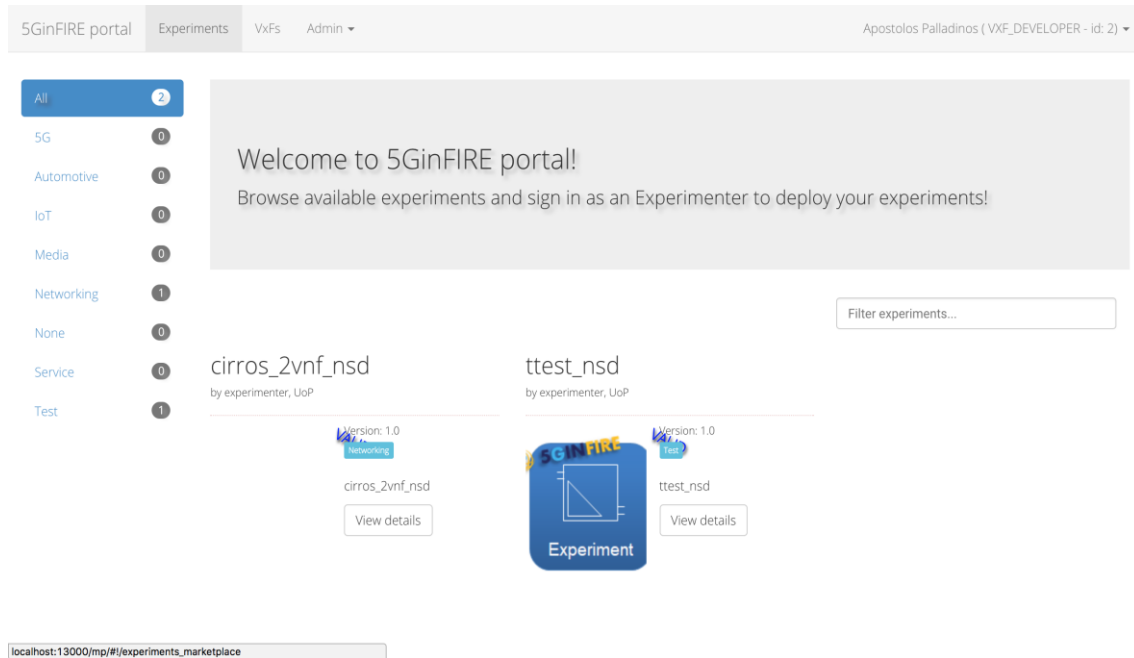


Figure 15 Available experiments, VxF developer login page. (Use case #2010)

On the top part of the interface we can distinguish the options that are provided to the VxF developer role. 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 we can see all the available categories in terms of experiments that are to be uploaded by either the experimenter or the service administrator user role. 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, at the bottom of the interface we can find the published experiments belonging to the opted category.

Similarly with the previous description in the Figure 16 at VxF menu option we can list all the available published Vxfs uploaded by either the VxF developers or the service administrators:

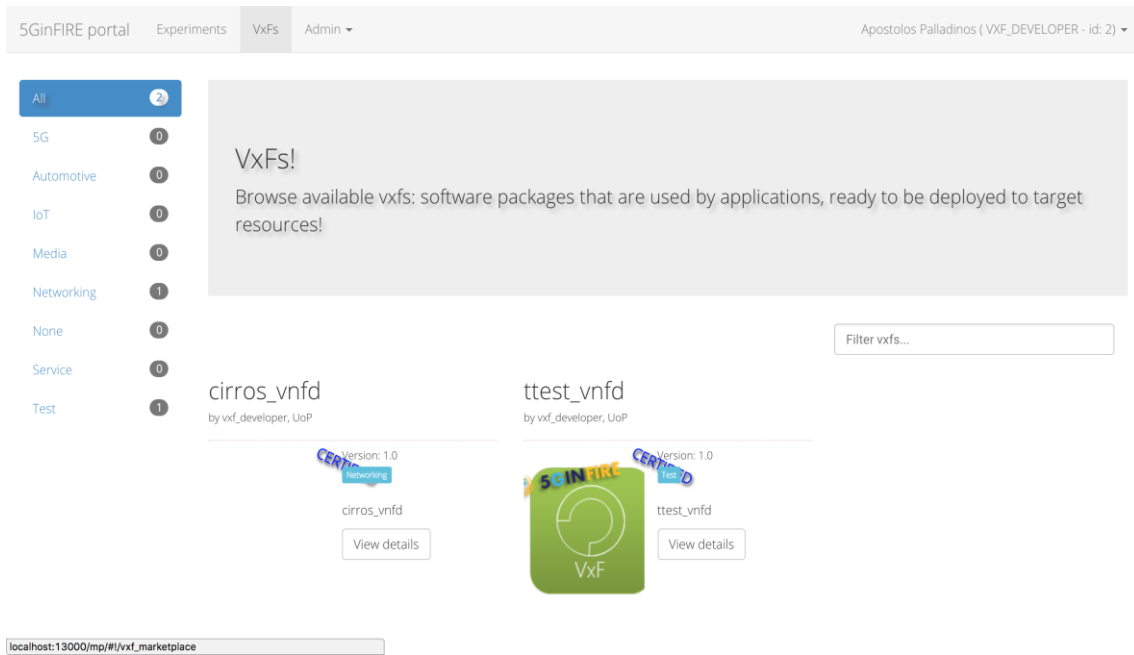


Figure 16 Available Vxfs. (Use case #2070 or #3030)

The options of the Admin tab of the menu can be seen in Figure 17 below:

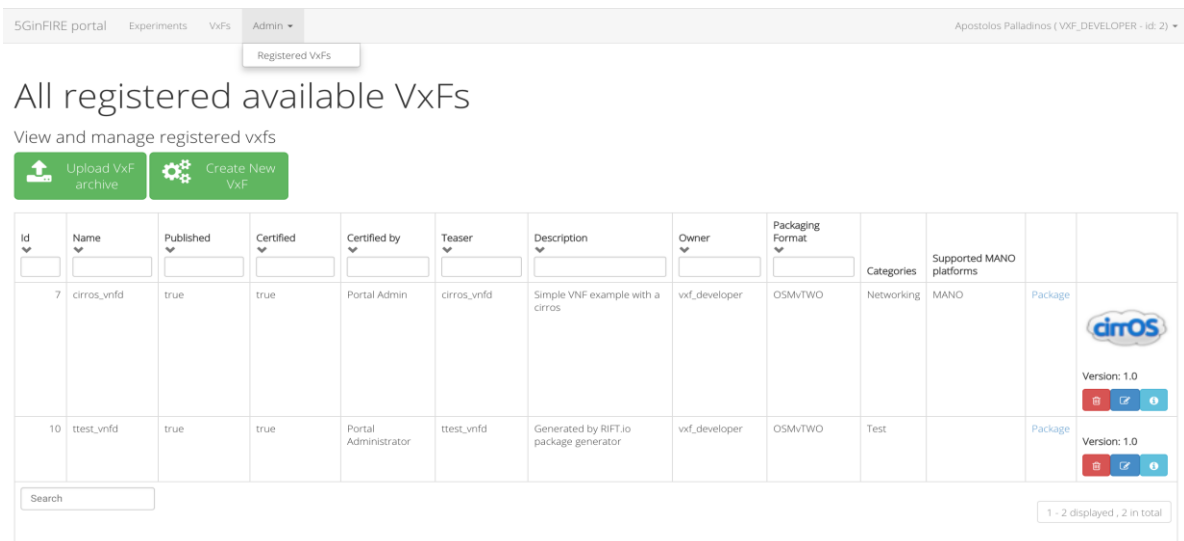


Figure 17 Vxfs Management. (Use case #3020)

The only submenu of this tab is the Registered Vxfs which are presented in the main body of the page in Figure 17. 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 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 18 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 19.

Figure 19 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.

Finalizing, the description of VxF developer user role we present the interfaces of edit and info button of each VxF. In the case of the edit button the interface is the same as Figure 19 but the only difference is 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 20 can be seen in the figure below:

The screenshot shows the 5GinFIRE portal interface. At the top, there is a navigation bar with '5GinFIRE portal', 'Experiments', 'Vxfs', and 'Admin'. On the right, the user is identified as 'Apostolos Palladinos (VXF_DEVELOPER - id: 2)'. The main content area displays details for a VxF package named 'cirros_vnfd'. A blue 'CERTIFIED' badge is visible. Below it, a green 'Package' button indicates 'Version: 1.0'. The package name 'cirros_vnfd' is prominently displayed. Metadata includes: Author: vxf_developer, Organization: UoP, Vendor: OSM, Date created: Oct 31, 2017 4:04:29 PM, Last update: Oct 31, 2017 4:35:54 PM, UUID: 9f0e71cd-13bd-439b-b5c6-688e8ad5befe, Certified by: Portal Admin, and Supported MANO Platforms: OSM TWO. A section for 'Onboarded to MANO providers:' is present. At the bottom, there are tabs for 'Description', 'Terms of use', 'Descriptor', and 'Descriptor (YAML)'. The 'Description' tab is selected, showing the text 'Simple VNF example with a cirros'.

Figure 20 VxF details. (Use case #3030)

As we can see here the most information have already been provided by either the VxF package or the VxF developer as we have described earlier in the previous parts but also some new fields are visible that is Organization, Date created, Last Update, UUID, Certified by and Onboarded to MANO providers. Most of those fields are self-explainable apart from perhaps the Onboarded to MANO providers field, which 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 we are going to describe is the experimenter. In the next figure you can see the landing page for this role:

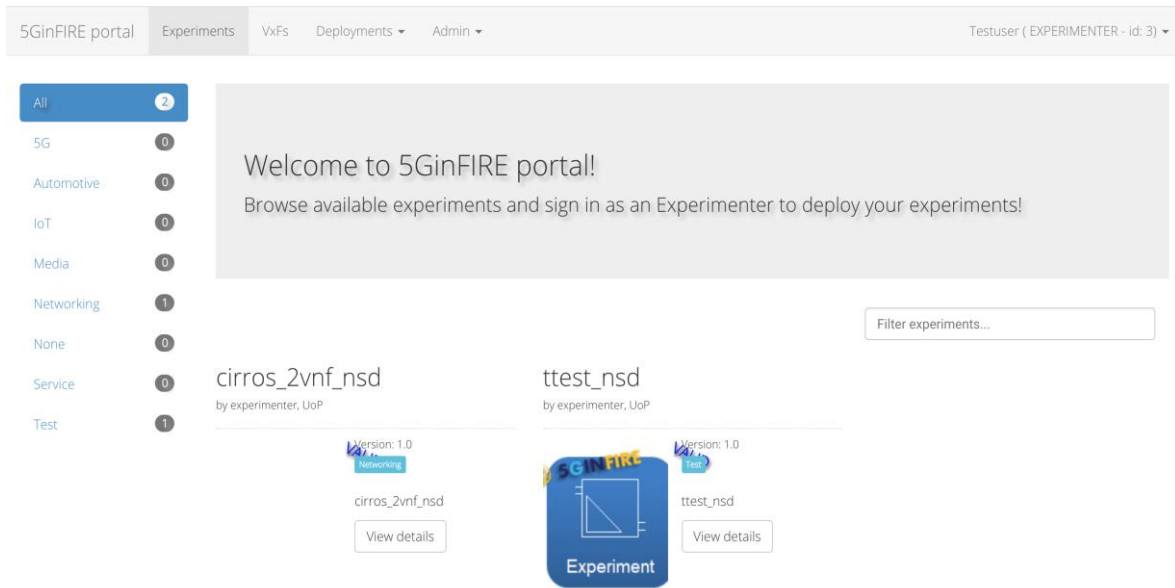


Figure 21 Available experiments, experimenter login page. (Use case #2010)

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 changed to registered experiment descriptors as we will see later on. In the next parts of this section we will describe only the new functionalities available to the experimenter role omitting the common already described at VxF developer section. The first menu we are going to describe is Deployments and especially its only submenu that is deploy experiments. The user interface for this submenu is shown up in figure below:

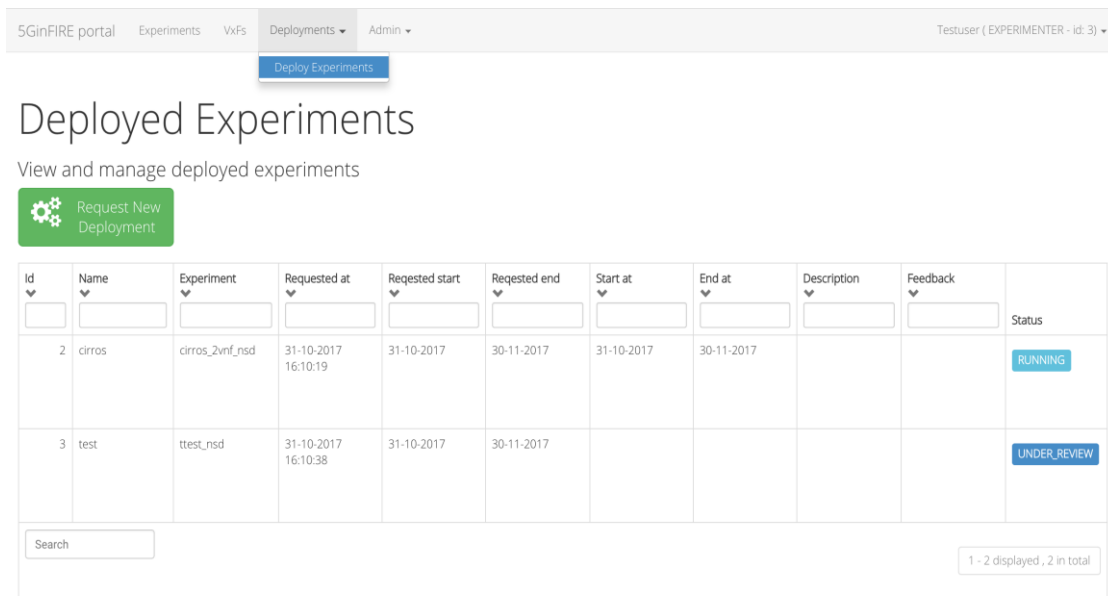


Figure 22 Available Deployed experiments (Use case #2010)

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 which is shown up after clicking on the green button on the top side of the page whose 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:

The screenshot shows the 'Request new deployment' interface. At the top, there is a navigation bar with '5GinFIRE portal', 'Experiments', 'Vxfs', 'Deployments', and 'Admin'. The user is identified as 'Portal Administrator (PORTALADMIN - Id: 1)'. The main heading is 'Request new deployment' with the user 'admin'. The form contains the following fields:

- Experiment (NSD):** A dropdown menu with 'cirros_2vnf_nsd' selected.
- Infrastructure:** A dropdown menu with 'UC3M' selected. Below it, a note says 'Select Infrastructure to place all constituent Vxfs'.
- Constituent VxF Placement:** A table with two columns: 'constituent VxF' and 'Infrastructure'.

constituent VxF	Infrastructure
cirros_vnfd [membervnfdindex:1]	UC3M
cirros_vnfd [membervnfdindex:2]	University of Bristol

 Below the table, a note says 'You optionally can select separate Infrastructure to place all each constituent Vxfs'.
- Name:** A text input field with the placeholder 'enter an alias for your requested deployment'.
- Description:** A large text area for providing details.
- Tentative Start Date:** A date picker set to '05-11-2017'.
- Tentative End Date:** A date picker set to '05-11-2017'.
- Request deployment:** A green button at the bottom of the form.

Figure 23 Experiment deployment creation (Use case #2020)

In this figure we can see all the necessary fields to define a new deployment. At the experiment drop down menu we can select the desirable experiment from those provided to be deployed by a service administrator. The experimenter can select also the target infrastructure for all or for each individual constituent VxF. At the next fields we also provide the necessary information be indicated by them and when all those fields are filled up the experimenter can submit this specific deployment by clicking the request deployment button on the bottom of the interface. Then this deployment request is sent to the service administrator in order to be proceeded or to be rejected.

In Figure 24 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 again is similar to the registered VxF page which appears in Figure 17 but the difference is that some metadata fields are missing and also this interface lists the available experiments instead of the available Vxfs.

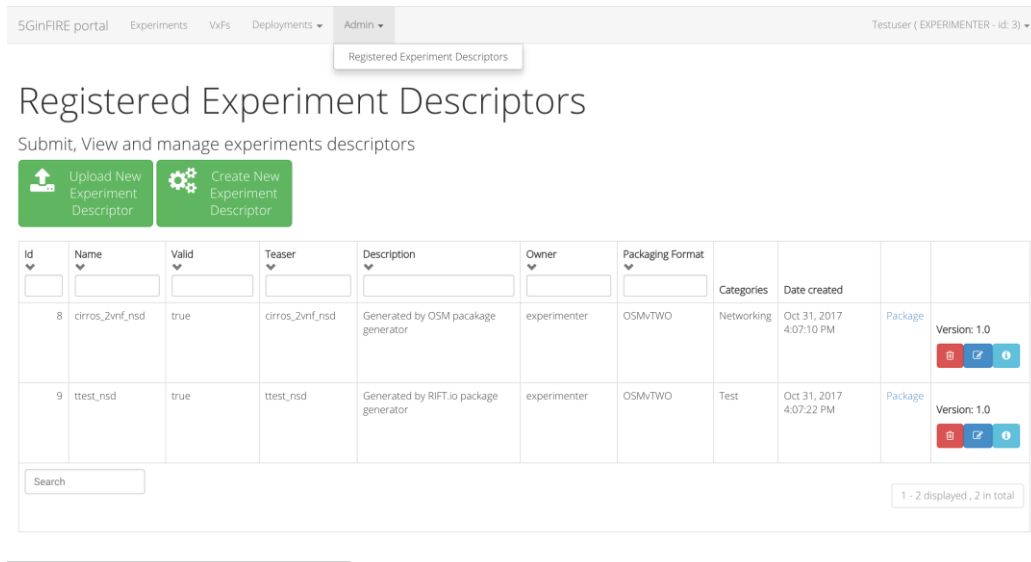


Figure 24 Experiments management (Use case #2050)

By clicking the first green button that is Upload new Experiment Descriptor the experimenter is redirected exactly to same interface as in Figure 18 but without the terms of use field and apart from that 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 19 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:

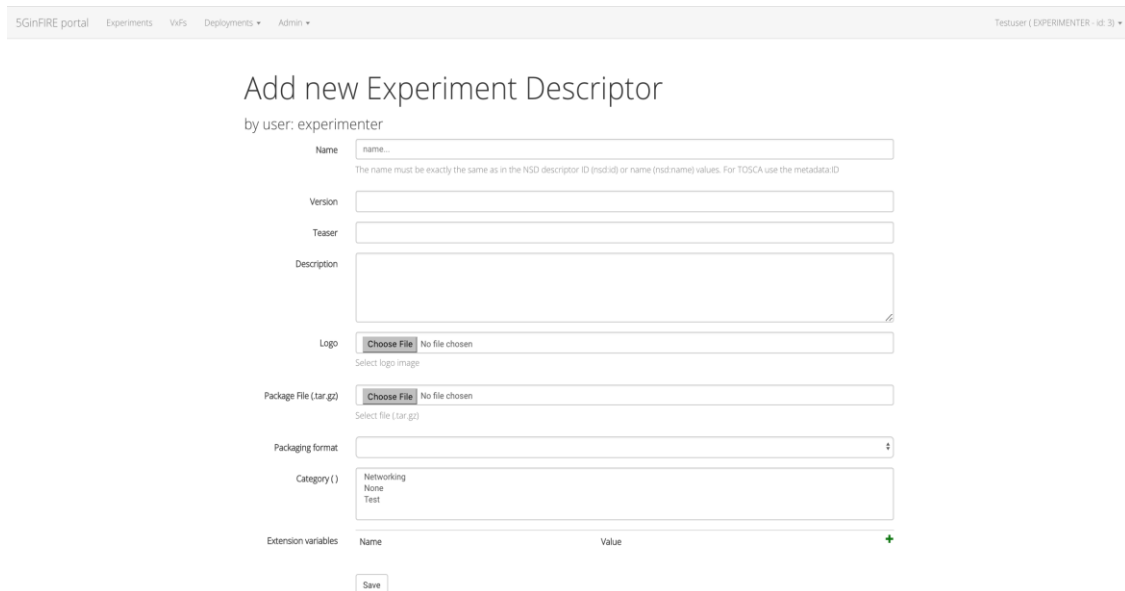


Figure 25 Experiment creation (Use case #2060)

Finally, in Figure 25 we can see the interface where an experimenter lands on when clicking on the edit button of an experiment in Figure 24 or on the details button of an experiment in Figure 21. This page

is also available for the VxF developer role as we can see in Figure 15. This page has similar fields with the VxF details interface in Figure 20 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 logging in is similar with the experimenter role and can be seen in Figure 26. 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:

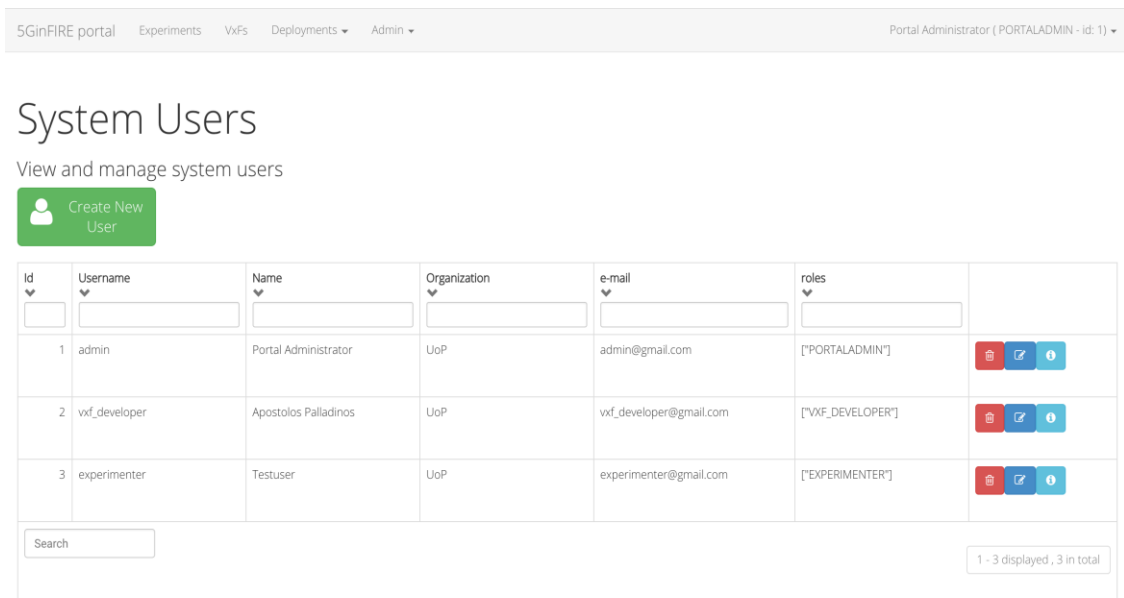


Figure 26 Users management (Use case #5010)

Additionally, the information icon of a system user pops up the details defined at the create new user interface which can be shown up when the user admin is clicking on the green button Create new user of Figure 26. The information interface can be seen in Figure 27 and the create new user interface in Figure 28:

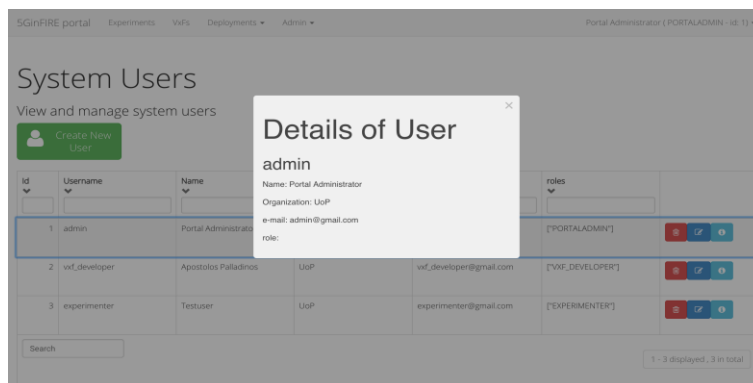


Figure 27 User details. (Use case #5010)

Figure 28 User account creation. (Use case #5010)

In Figure 28 we see some common used information about a new user that is 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 is clicking on the save button on the bottom of the page.

Next we can see in Figure 29 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 24, Figure 17, Figure 22 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 24, Figure 17, Figure 22 but with some additional metadata fields. Each and every of those cases are analysed in the next sections.

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	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	Version: 1.0

Figure 29 Admin tab submenu

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

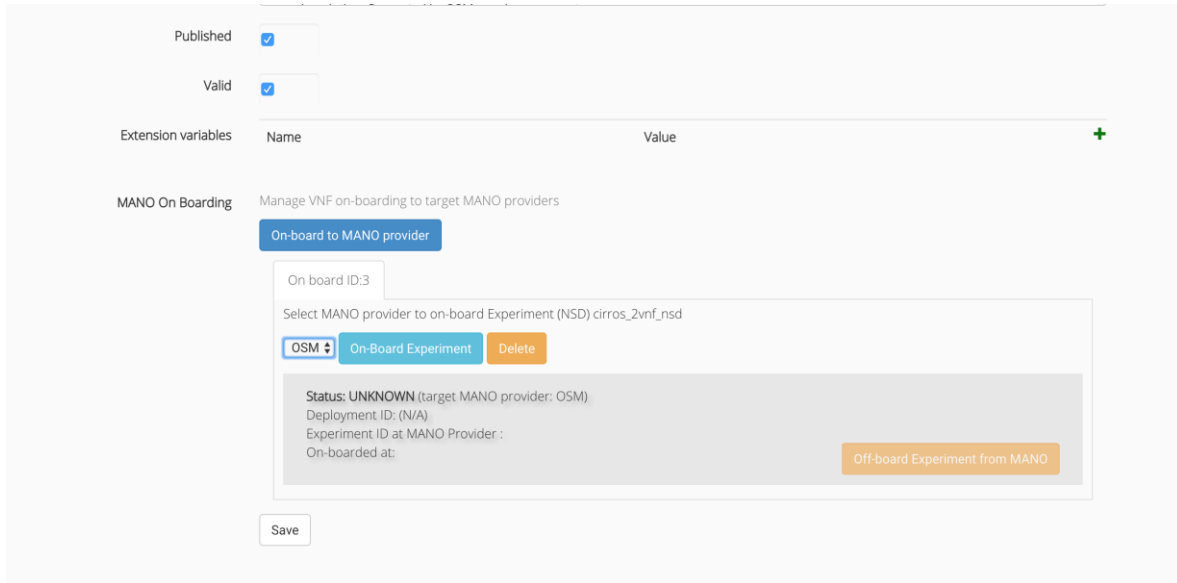


Figure 30 Extra fields for admin role about registered experiments (Use case #4010 and #5030)

We note that comparing with the Figure 25 we have two extra fields published and valid and also the capability of making the selected experiment on-board or off-board on the provided by the administrator 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:

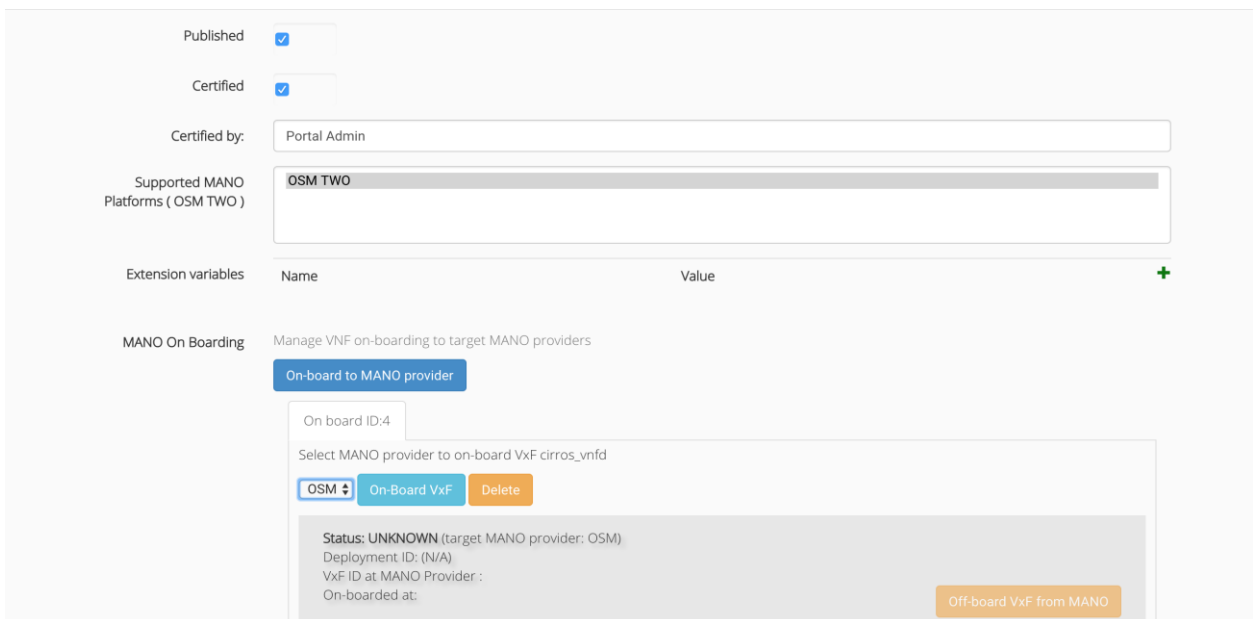


Figure 31 Extra fields for admin role about registered VxFs. (Use case #3020)

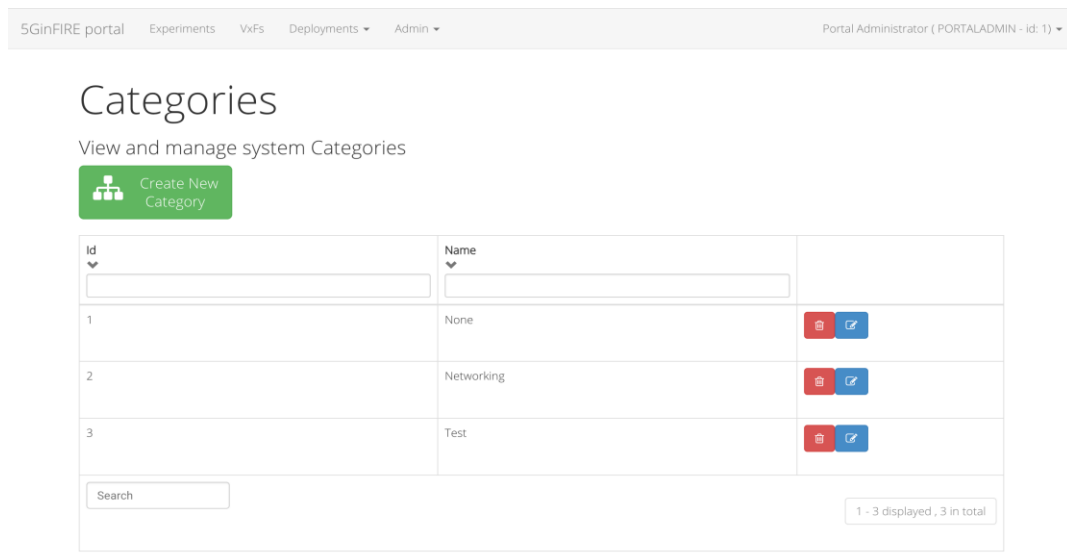
We note that comparing with the Figure 19 we have some extra fields that is published, certified and certified by and also the capability of making the selected VxF on-board or off-board on the provided by the administrator MANO platform. When the administrator checks the published checkbox the current 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 32 Deploy a validated experiment (Use case #5030)

The two extra fields in that interface comparing to Figure 23 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 33 and Figure 34 we can find the categories listing interface and the creation of a new category interface respectively.









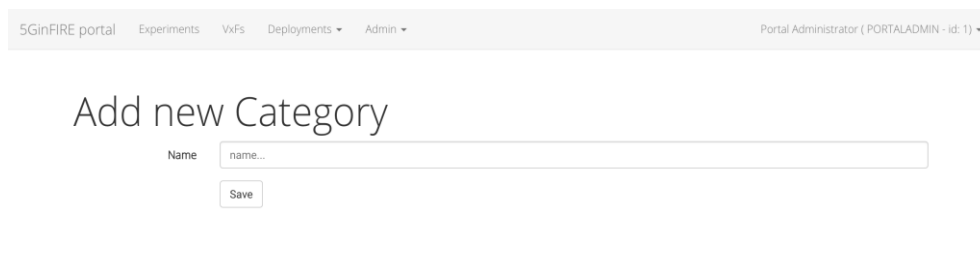
Id	Name	
1	None	 
2	Networking	 
3	Test	 

Figure 33 Available categories



Name

Figure 34 Category creation

As we can note in Figure 34 the only field we need to define a new category 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 35, Figure 36 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		✖ ✎

1 - 1 displayed , 1 in total

Figure 35 Available MANO platforms

Add new MANO Platform

Name

Version

Description

Figure 36 MANO platform creation

The only fields required to define a new MANO platform are its name, version and a small description, as Figure 36 indicates.

Through system MANO providers submenu a services administrator essentially is able to connect portal to a deployed MANO platform via the API URL field provided during the creation of a new MANO provider. Interface which lists all those MANO providers is more or less the same with the rest listing interfaces that we have seen before. In Figure 37 and Figure 38 we can see the MANO providers listing interface and the creation of a new MANO provider respectively.

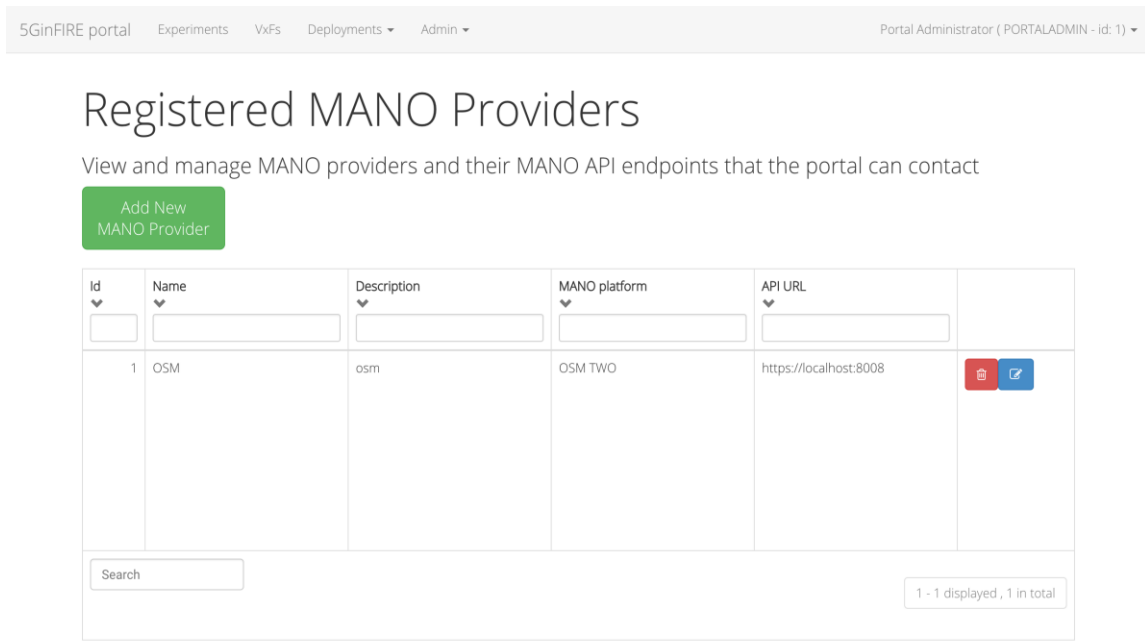


Figure 37 Available MANO providers

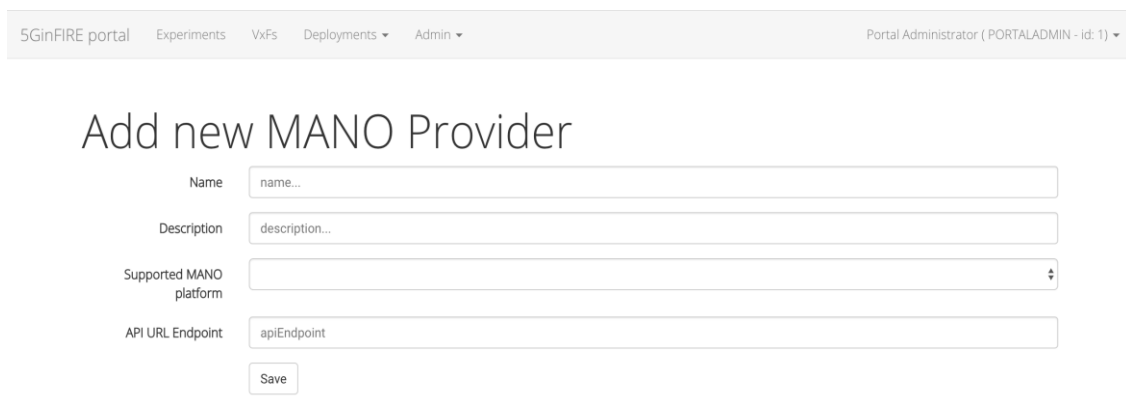


Figure 38 MANO provider creation

In Figure 38 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 and perhaps the most interesting 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.

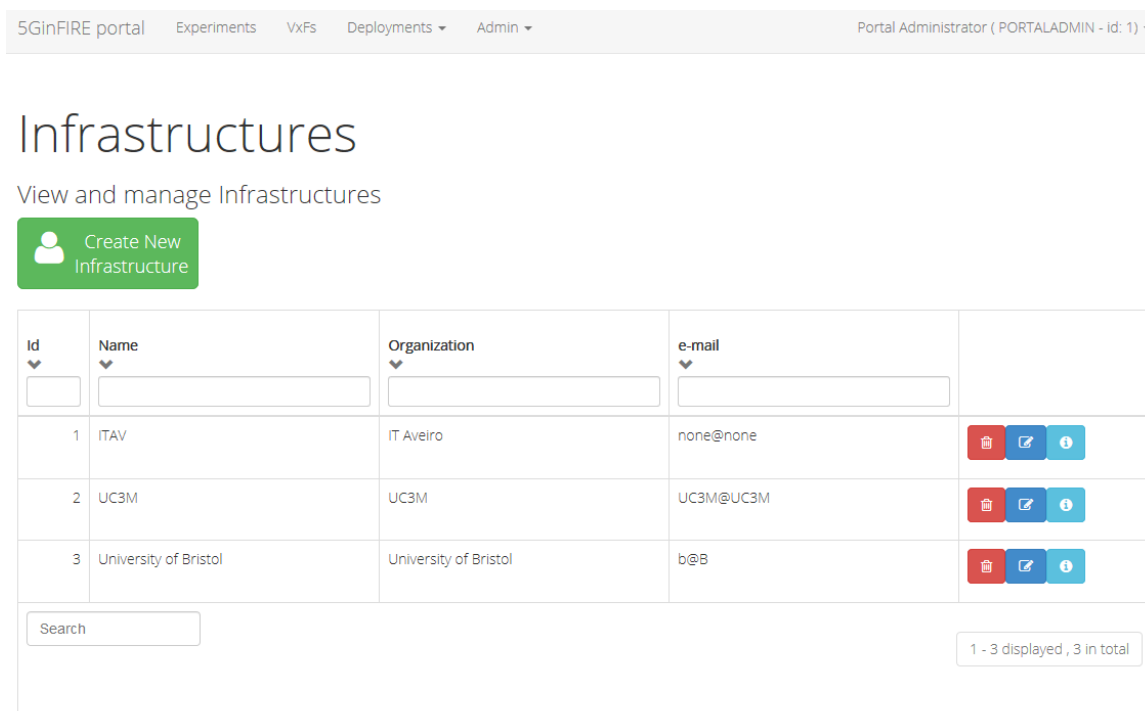


Figure 39 Available target Infrastructures

Finally there is page for managing the available target infrastructures (see Figure 39). This is useful when Experimenters create the Deployment Request to select target infrastructure

2.6 Deployment details

5GinFIRE deployment is depicted in Figure 40 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 41.

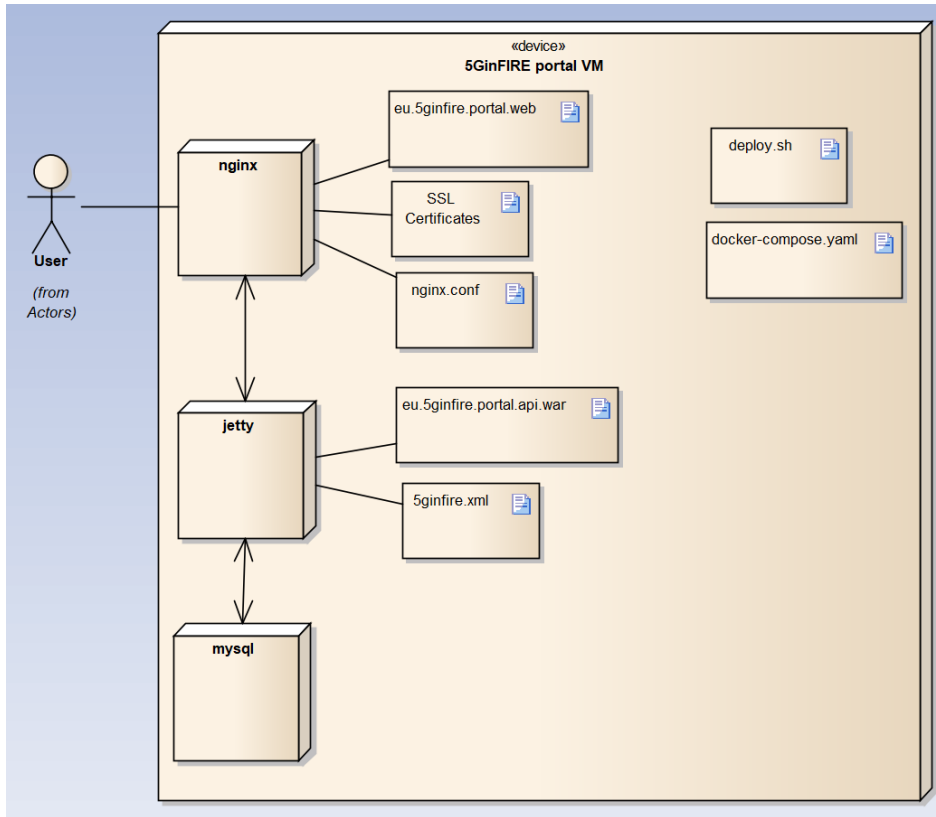


Figure 40 Deployment diagram of 5GinFIRE

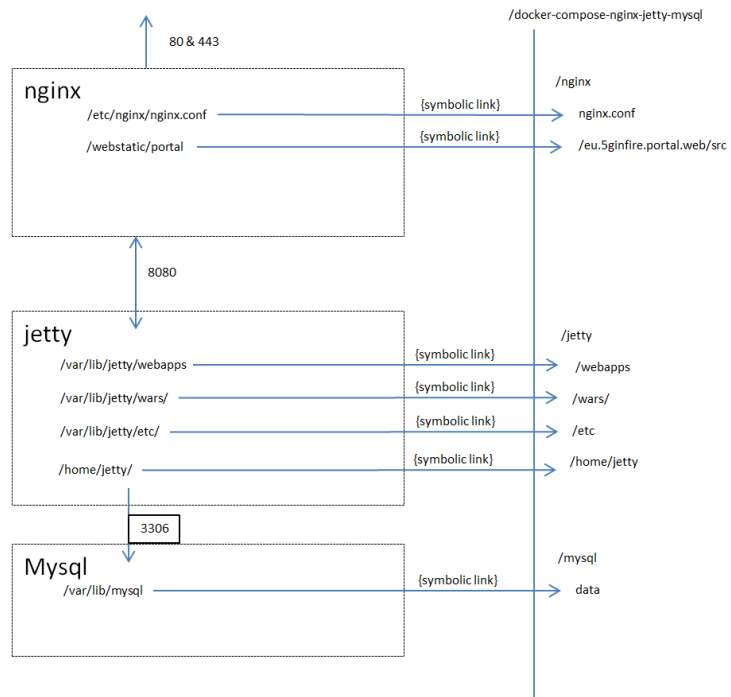


Figure 41 Internal mappings of deployed containers

2.7 Delivery plan

The delivery plan for the first version of the portal is depicted in Table 4. The plan is 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)

The next 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 nakedn API

2.9 Licensing

All components are open source with Apache 2.0 License

3 Middleware tools

3.1 Admin tool and provisioning: Launchpad (OSM TWO/THREE)

The Launchpad is the point of contact in the OSM architecture 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.

In OSM Release TWO, which is the version of OSM that has been chosen to build the reference MANO stack within the project, 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.

This Northbound API, in future OSM releases (OSM Release THREE has been announced, at time of writing) aims at supporting all the invocations of the main MANO actions by external systems (see Figure 42), such as the UI, OSS or the OSM client (i.e. a command line interface client to interact remotely with the OSM’s Northbound REST API), being the Service Orchestrator the responsible entity of triggering all the subsequent actions and requests to the other OSM components.

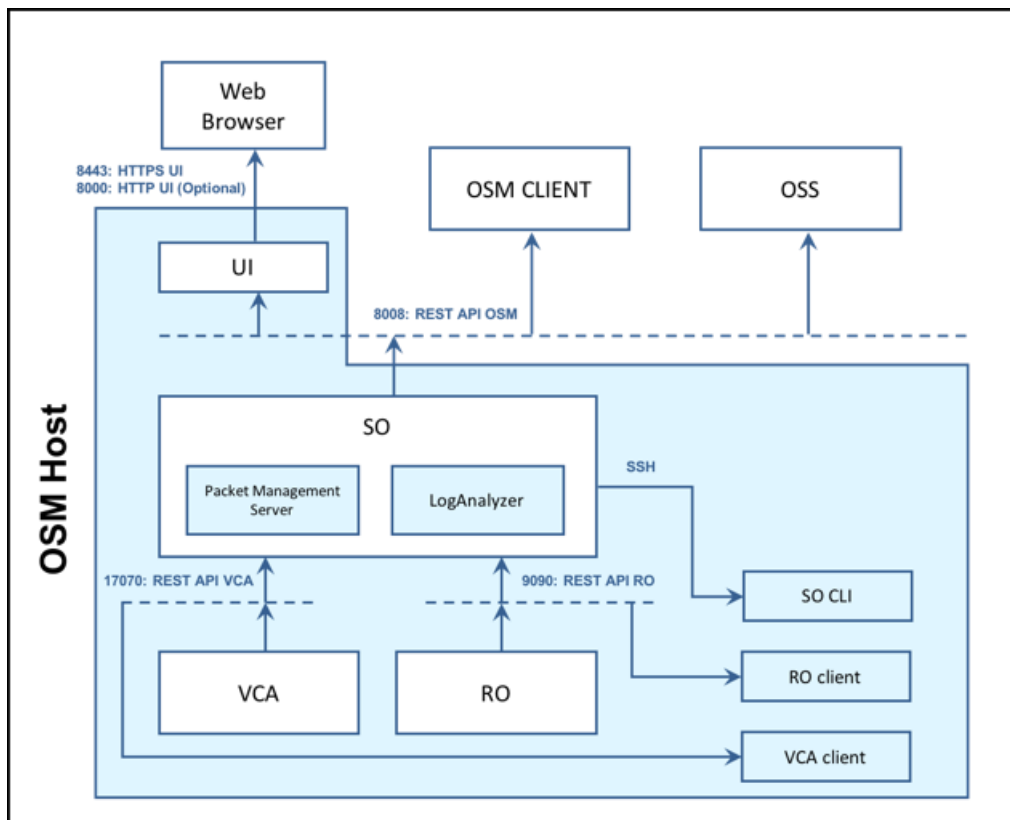


Figure 42 OSM Release THREE Northbound API

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. The Launchpad in OSM Release THREE will provide additional real-time statistics for VNFs and Network Services.

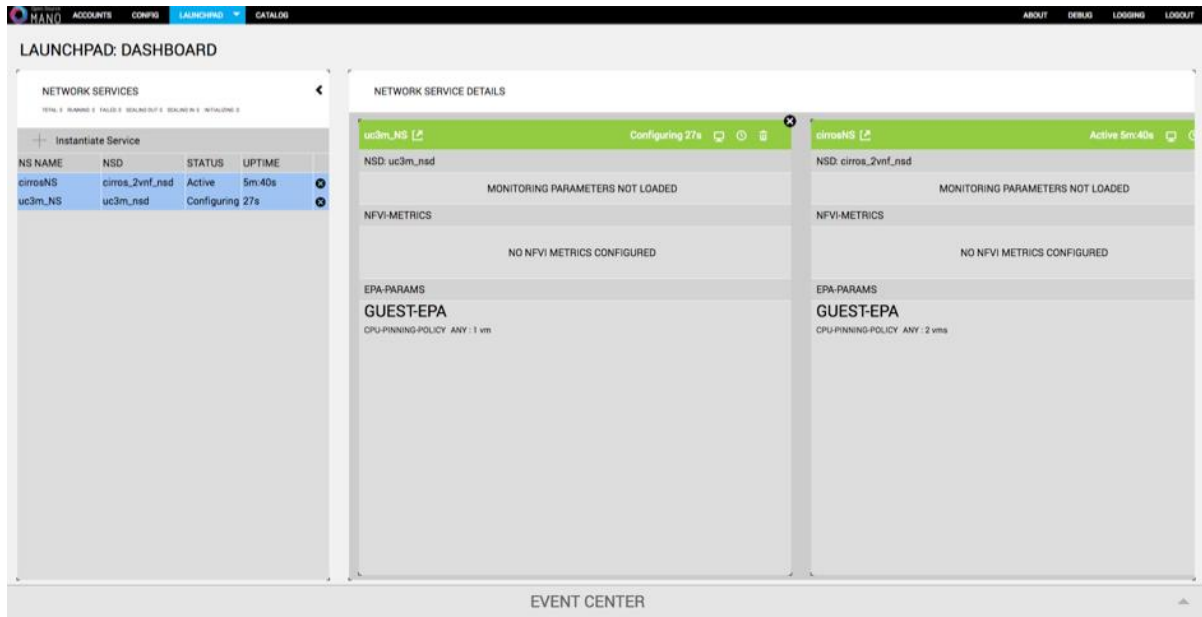


Figure 43 Launchpad: Dashboard

Figure 43 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 44).

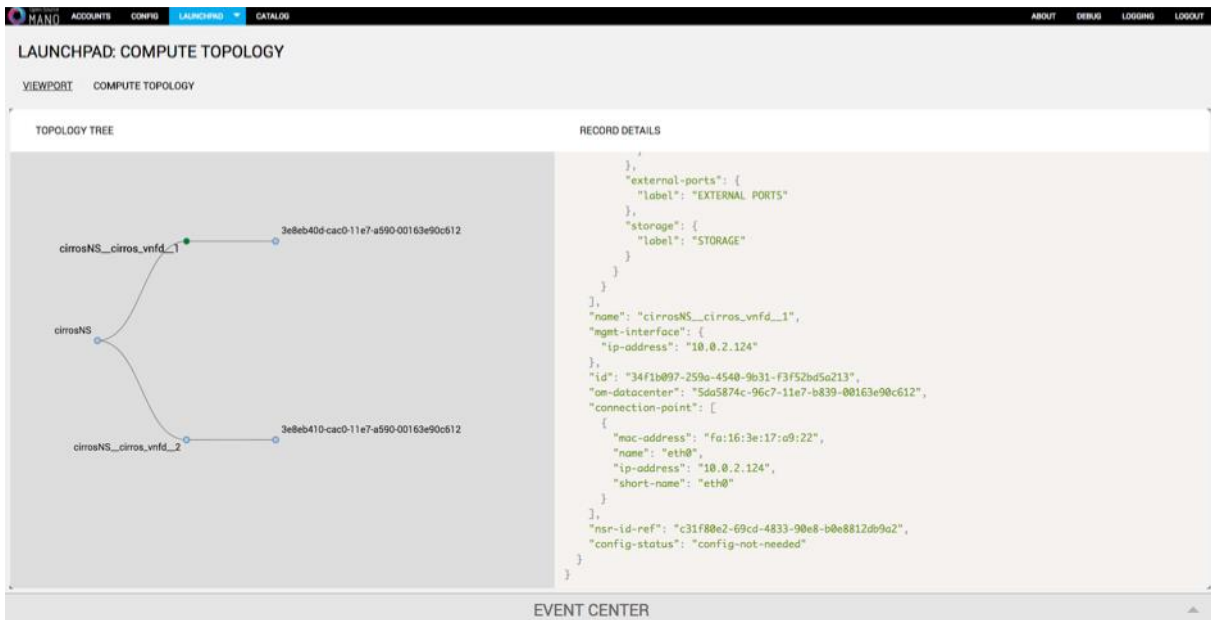


Figure 44 Launchpad: Compute Topology

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 45 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 to select the datacenter where the VNF should be deployed in the instantiation of the network service.

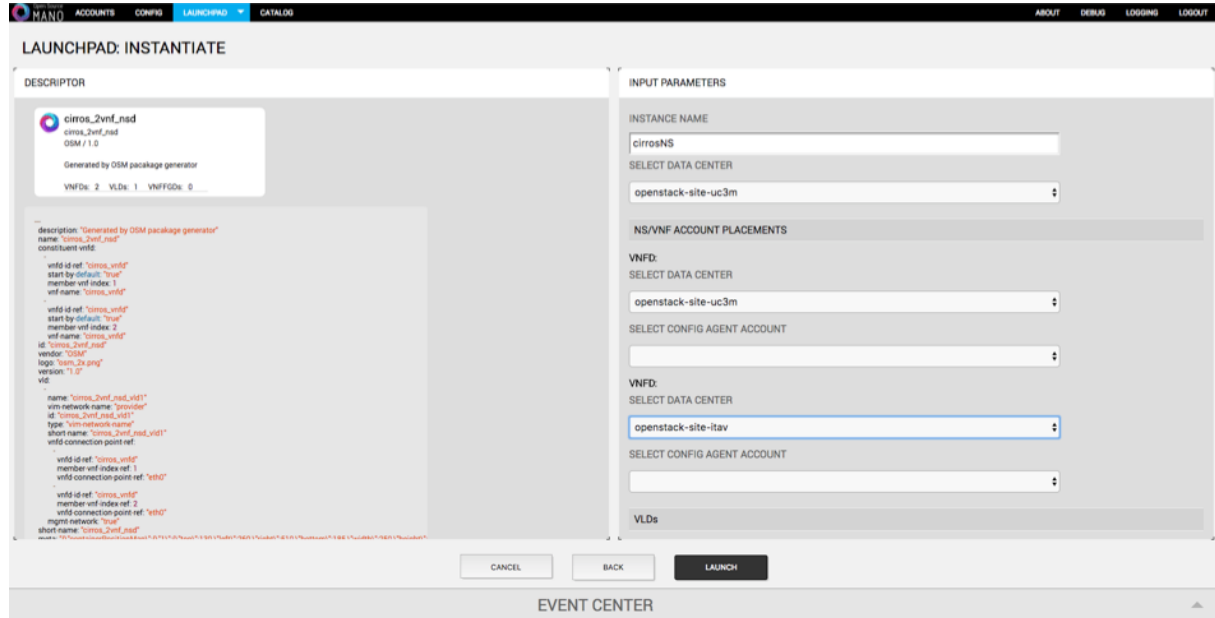


Figure 45 Launchpad: Instantiate

Finally, we want to mention that the source code of the Launchpad has been implemented with the JavaScript run-time environment Node.js¹, being the code is available in the OSM-UI repository².

3.2 YANG models and packaging for VxNF/NSD

A NFV descriptor (VNFD or NSD) must be modelled using the YANG models defined in this specification and then encoded as JSON text using the using the procedures specified in [RFC 7951](#).

3.2.1 Overview of the YANG models

YANG is a data modelling language for the definition of data sent over the NETCONF network configuration protocol. The YANG data modelling language was developed by the NETMOD working group in the Internet Engineering Task Force (IETF) and was published as [RFC6020](#) 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

¹ Node.js website (last access: Nov. 2017): <https://nodejs.org>

² Launchpad within the OSM-UI repository in OSM release TWO (last access Nov. 2017): <https://osm.etsi.org/gitweb/?p=osm/UI.git;a=tree;f=skyquake/plugins/launchpad;h=a6705ba25b1413e4f35b01a341e0149c07844432;hb=aed6500508197a8d64a41c795e25f5be05ddf930>

independent, can then be converted into any encoding format, e.g. XML or JSON, that the network configuration protocol supports.

You can also find further information and examples on the currently used YANG models on:

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

There is also an ongoing documentation of YANG based NFVD specification by ETSI is available here:

https://docbox.etsi.org/ISG/NFV/Open/Drafts/SOL006_YANG_based_NFV_Descriptors_spec

3.2.2 Developer tools for YANG modelling

To help the VxNF/NSD developer to validate and use the YANG model, several sets of tools are available for free:

- **Python YANG validator:** An extensible YANG validator and converter written in python, Available on Github under the ISC License <https://github.com/mbj4668/pyang>
- **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>
- **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>
- **YANG schema validator** is available online at: <http://www.yangvalidator.com/>

3.2.3 Functional requirements for VNF & NSD Packaging specification

In order to be compliant with OSM, the package must strictly follow the requirements precisely defined by ETSI in the following documents:

- *“NFV, management and orchestration, VNF Packaging Specification”* Part. 6. [ETSI GS NFV-IFA 011](#)
- *“NFV, Management and Orchestration; Network Service Templates Specification”* Part 5. [ETSI GS NFV-IFA 014](#)

Here are the general and packaging requirements extracted from the above documents.

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_PACK.STRUCT.004	The VNF package shall enable including VNF specific files organized according to the design of the VNF, or referencing these files if they are external to the package.
VNF_PACK.STRUCT.005	The VNF package shall provide means to address individually the files which it contains and/or which it references.
VNF_PACK.STRUCT.006	If an external reference (e.g. URL) is used, file integrity information (such as checksum/signature) shall be specified to guarantee the integrity of the referenced file, so it cannot be substituted with a different file by the same name.

You can find some examples of VnF/NSD package here:

<https://osm-download.etsi.org/ftp/osm-3.0-three/examples/>

3.2.4 Package pre-validation in the 5GinFire portal

Currently, the 5GinFIRE portal is not automatically pre-validating a package with the basic rules defined above. The administrator will be validating each package manually.

An interesting feature would be to implement an automatic basic validation in the API, preventing users to submit invalid VxNF/NSD package, and so, he will get instant feedback on what's wrong.

The validation process would be in two steps:

- First validate the package with the requirements specified by the ETSI. (Requirements needs to be transposed with code).
- Validate the Yang model (use available tools to validate the model as listed above).

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)
- VxF, experiment metadata
- Utilizes the OSM repository/catalogues via the OSM API
- Makes public and available for download VxF, NSD archives
- Categorizes items

Figure 7 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 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.

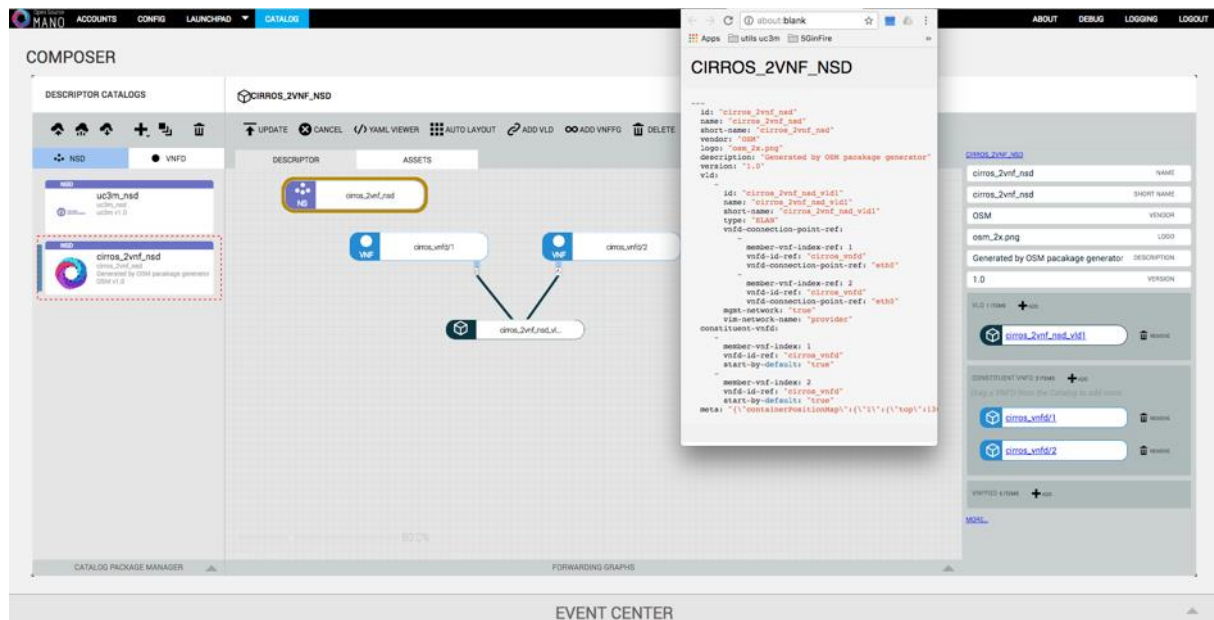


Figure 46 Composer

Figure 46 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 provide 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/VNFs, 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 5GiFIRE portal).

Finally, we want to highlight that all the code responsible for the Composer functionality is available in the OSM-UI repository³.

4.3 Utilizing the OSM API

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

This API provides access to 5 main categories of operations:

4.3.1 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.

4.3.2 VNF Descriptor Management

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

4.3.3 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`.

4.3.4 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.

4.3.5 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.htm>

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

³ 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>

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) have sole access to the resources allocated to them.

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. For simplicity, we specify the requirements 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, 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 Use cases

The actors in the use cases are:

- 5GinFIRE portal
- OSM

These actors are the systems involved because when an experimenter or an administrator interacts with these actors, they use this system to fulfil the process. So this system is a support system for those actors.

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

Table 7 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.

4.4.4 Process flows

In this section, we are going to observe the process flows of some of the use 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)

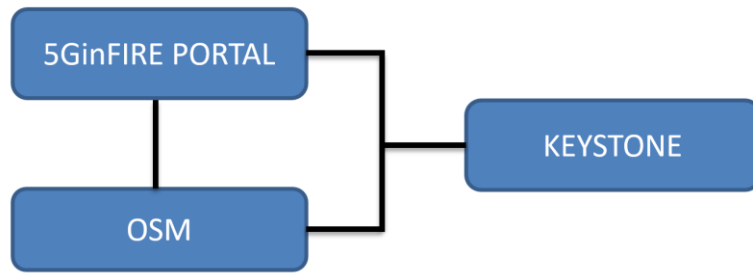


Figure 47 Component interconnection diagram

4.4.4.1 Create user 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 user.

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

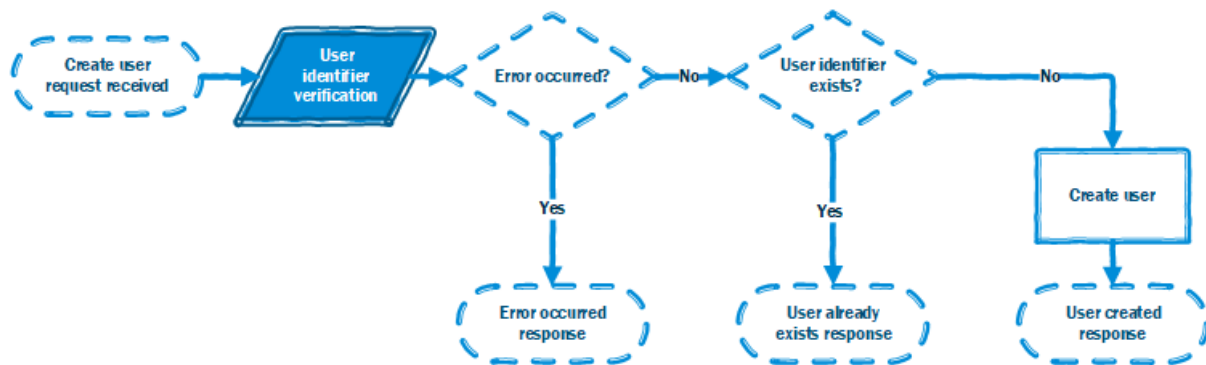


Figure 48 Create a user process flow

4.4.4.2 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

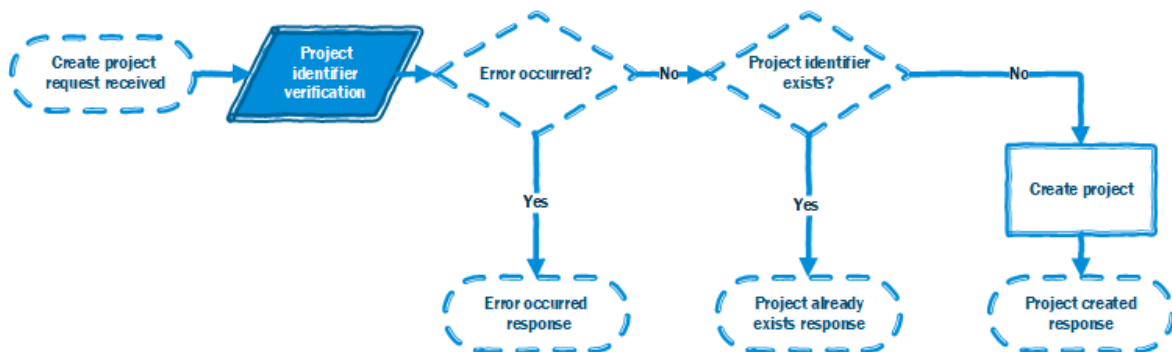


Figure 49 Create a project process flow

4.4.4.3 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

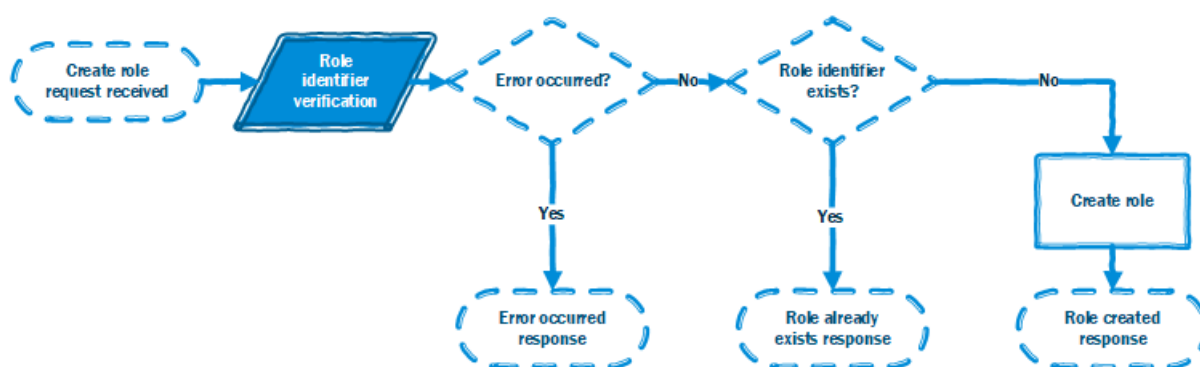


Figure 50 Create a role process flow

4.4.4.4 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

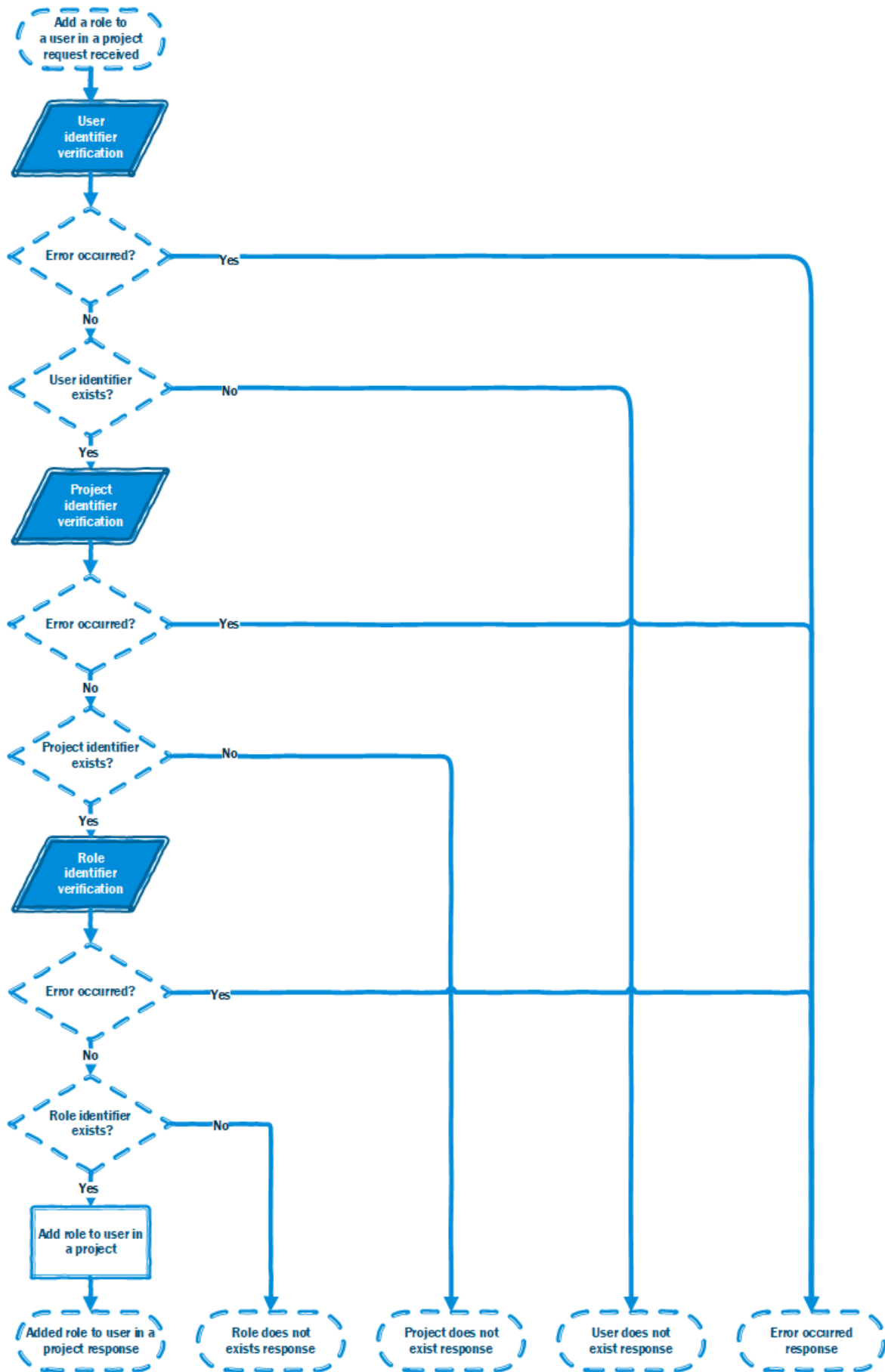


Figure 51 Add a role to a user in a project process flow

4.4.4.5 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

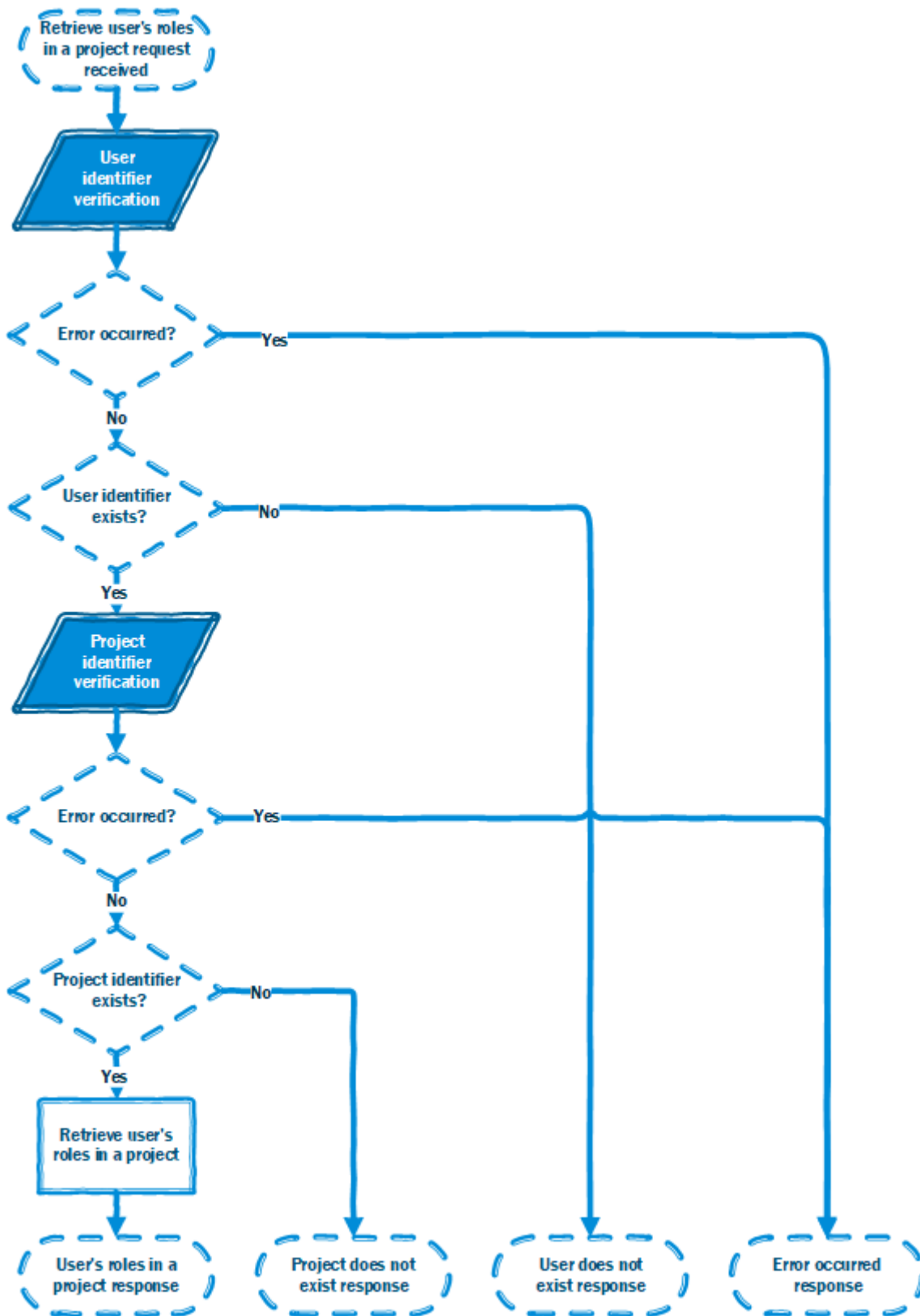


Figure 52 Retrieve user's roles in a project process flow

4.4.4.6 Authenticate user 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 user.

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

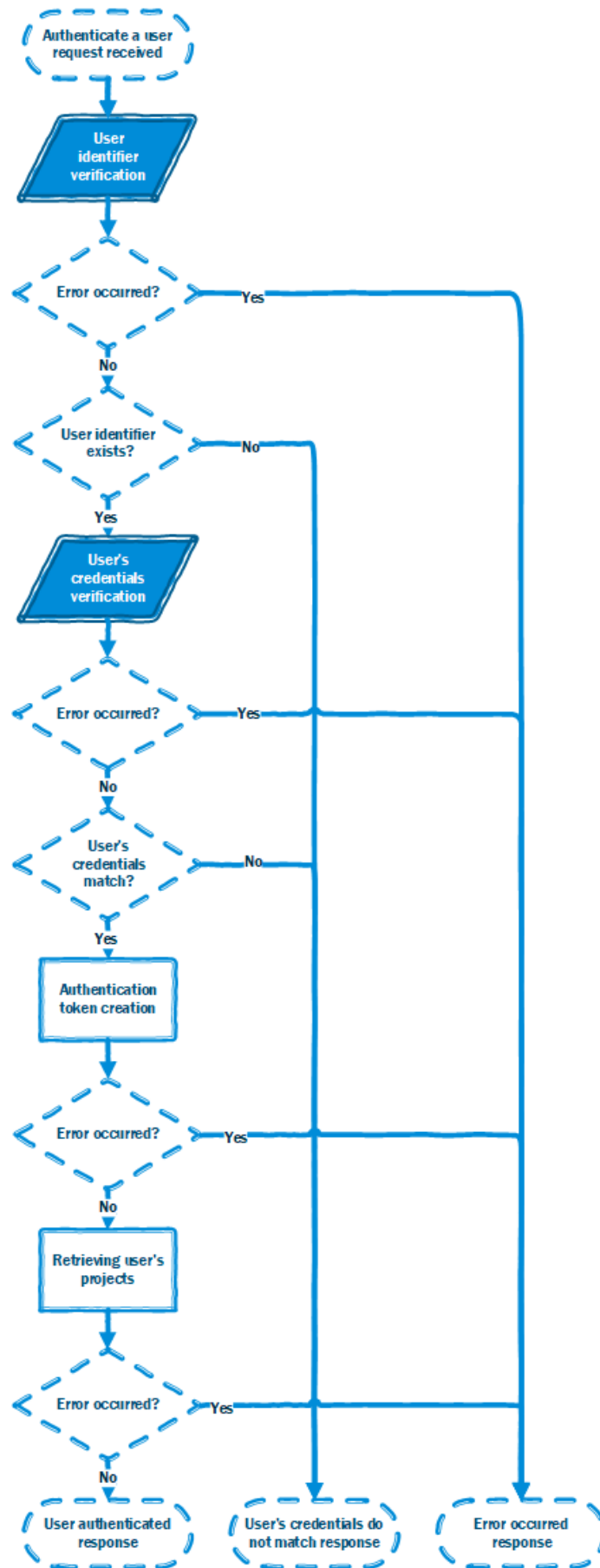


Figure 53 Authenticate a user process flow

4.4.5 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 8 User roles

No	Role	Role description
1	Owner	Is attributed to each user that creates an experience 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.

4.4.6 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.

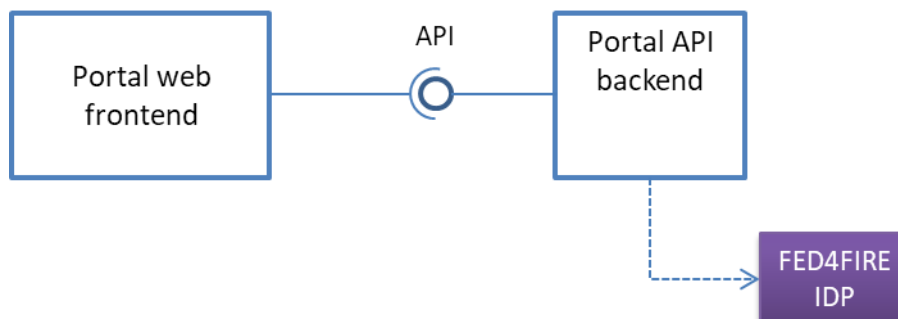


Figure 54 portal and Fed4FIRE+ integration

Figure 54 displays the high level integration with Fed4FIRE Identity provider via the Portal API Backend. Figure 55 displays the designed interactions between the user, the 5GinFIRE portal and the Fed4FIRE IDP. The scenario is as follows:

- The user requests to login with Fed4FIRE authentications via the Web interface
- The request goes to the backend API which redirects the client to the Fed4FIRE IDP.
- The user is authenticated to Fed4FIRE IDP
- Upon success the IDP redirects the Web browser to the return backend API URL with a token
- The Backend API uses the token to retrieve user information (name, lastname, email, etc) from Fed4FIRE
- The Backend API process user details and assigns a user role (by default will be EXPERIMENTER/VxF DEVELOPER)
- The Backend API redirects the end user to the admin page of the portal.

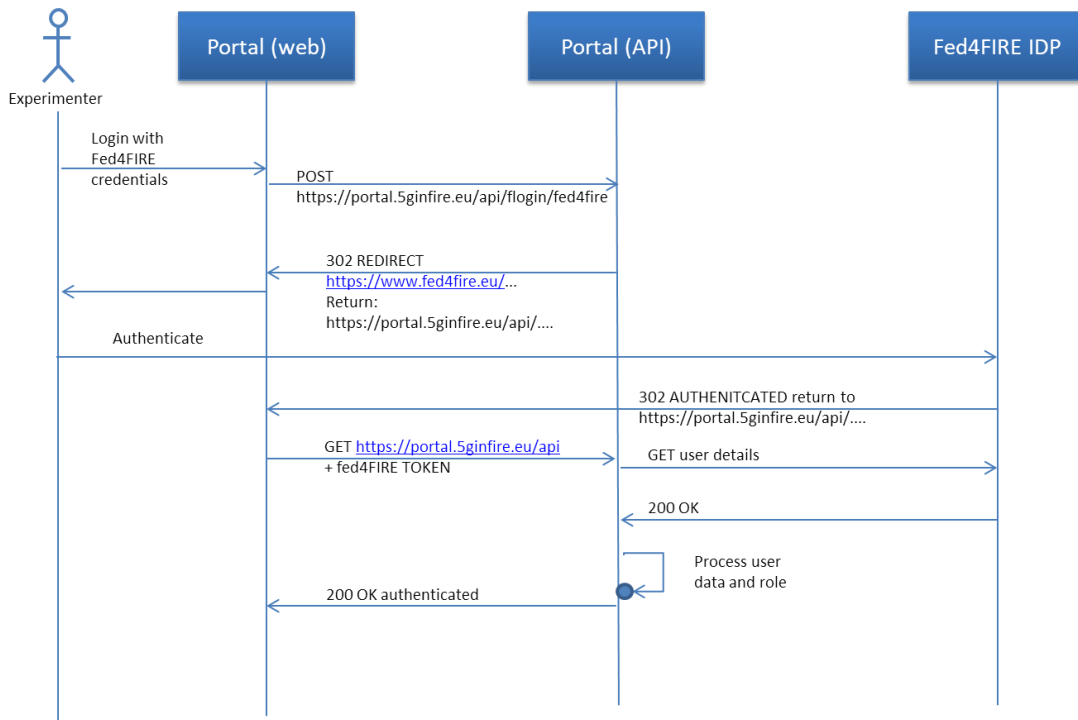


Figure 55 Interaction diagram between the portal and Fed4FIRE IDP

6 Future enhancements/Next versions

The 5GinFIRE portal and middleware technologies will continue to evolve. As presented in 4.4 there is a planned integration with a Keystone service for authorization. It is also planned to implement authorization for Fed4Fire users with close collaboration to Fed4FIRE project. Other enhancements will include: i) Scheduler/Calendars, ii) Enhanced Deployment Management, iii) Automated instantiation of the requested Experiment towards OSM via the OSM API (this depends on future testbed policies) and iv) Integrations with future OSM versions.

Complementarily to the mechanisms implemented by the 5GinFIRE portal, defined in this deliverable, the evolution of the portal will also 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) [4], 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 [5]. 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 [6]. 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 [7]. In parallel to this, an alternative direction has been taken by ETSI, currently addressing the standardization work of NFV descriptors based on YANG [8], 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.

7 References

- [1] [Online]. Available: <https://github.com/5GinFIRE/eu.5ginfire.portal.web/issues>.
- [2] [Online]. Available: <https://github.com/5GinFIRE/eu.5ginfire.portal.api/issues>.
- [3] [Online]. Available: <https://www.openapis.org/>.
- [4] Topology and Orchestration Specification for Cloud Applications Version 1.0. 25 November 2013. OASIS Standard. <http://docs.oasis-open.org/tosca/TOSCA/v1.0/os/TOSCA-v1.0-os.html>.
- [5] TOSCA Simple Profile for Network Functions Virtualization (NFV) Version 1.0. Edited by Shitao Li and John Crandall. 11 May 2017. OASIS Committee Specification Draft 04. <http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/csd04/tosca-nfv-v1.0-csd04.html>. Latest version: <http://docs.oasis-open.org/tosca/tosca-nfv/v1.0/tosca-nfv-v1.0.html>.
- [6] ETSI Group Specification NFV-SOL 004 V2.3.1; Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; VNF Package specification; July 2017
- [7] Network Functions Virtualisation (NFV) Release 2; Protocols and Data Models; NFV descriptors based on TOSCA specification (status: standard not ready for download); https://portal.etsi.org/webapp/WorkProgram/Report_WorkItem.asp?WKI_ID=49491
- [8] M. Bjorklund, YANG-A Data Modeling Language for the Network Configuration Protocol (NETCONF), RFC 6020, Internet Engineering Task Force (IETF), October 2010.