



## NETWORK FUNCTIONS AS-A-SERVICE OVER VIRTUALISED INFRASTRUCTURES

GRANT AGREEMENT NO.: 619520

Deliverable D6.3

# Users Dashboard

**Editor** Evangelos K. Markakis (TEIC)

**Contributors** Evangelos Markakis, Athina Burdema, George Alexiou, Evangelos Pallis, Anargyros Sideris (TEIC), Aurora Ramos, Javier Melián (ATOS), Thomas Pliakas (CLDST),

**Version** 1.0

**Date** December 22<sup>nd</sup>, 2015

## Executive Summary

---

The DoW describes this deliverable as: "*D6.3: User Dashboard (M24) – Report+Prototype – Design and implementation of the User Dashboard. Description of the implemented interfaces to the T-NOVA management modules. Presentation of the GUI and functionalities.*"

The dashboard constitutes T-NOVA's system front-end. It enables the Service Providers (SPs) to create and publish their services, the Function Providers (FPs) to create and publish their Virtual Network Functions (VNFs) and both of them to participate in auctions for buying and selling, respectively, the published VNFs. In addition, the Dashboard allows the customers to discover (exploiting various criteria) and consume the offered services. Having in mind terminal neutrality and seamless upgradeability, whenever a new version is available, a web-based implementation of the Dashboard has been selected. Finally and towards maximising the users' Quality of Experience, T-NOVA's Dashboard allows personalization for a variety of settings such as interface, appearance and content according to each user's profile configuration.

In this context, this document presents an overview of the current technologies related to the development of a Dashboard. This includes the fronted (e.g. BootStrap), middleware (e.g. Angular) and backend (e.g. Django) web development frameworks. In the same direction, the existing web development and deployment approaches are surveyed focussing in the ones adopting a client-side and modular logic, for reasons of better scalability and interoperability—towards this, the notion of containerised micro-services is proposed and adopted for the Dashboard's implementation and deployment. In addition, the document describes the interfacing between the Users' Dashboard and the rest T-NOVA management modules, namely the Billing, UMAA (User Management, Authentication and Access Control), SLA management, Brokerage, Business Service Catalogue, Function Store and Orchestrator. Finally a step by step presentation of how to use the realised Dashboard is given in a series of screenshots.

## Table of Contents

---

<b>1. INTRODUCTION .....</b>	<b>8</b>
1.1. DELIVERABLE'S STRUCTURE .....	10
<b>2. USERS DASHBOARD DESIGN.....</b>	<b>11</b>
2.1. AVAILABLE WEB DEVELOPMENT FRAMEWORKS .....	11
2.1.1. <i>Frontend</i> .....	11
2.1.2. <i>Middleware</i> .....	12
2.1.3. <i>Back-End</i> .....	13
2.1.4. <i>Summary</i> .....	15
2.2. DEVELOPMENT APPROACHES.....	15
2.2.1. <i>Server-Centric Web Application (SCWA)</i> .....	15
2.2.2. <i>Browser-Centric Web Application (BCWA)</i> .....	15
2.2.3. <i>Summary</i> .....	16
2.3. DEPLOYMENT ARCHITECTURES.....	16
2.3.1. <i>Monolithic</i> .....	16
2.3.2. <i>Micro-services</i> .....	17
2.3.3. <i>Containerized micro-services</i> .....	18
2.3.4. <i>Summary</i> .....	19
<b>3. USERS DASHBOARD IMPLEMENTATION .....</b>	<b>20</b>
3.1. ROUTING MODULE .....	20
3.2. BOOTSTRAP AUTHENTICATION FUNCTION.....	20
3.3. ROOT CONTROLLER.....	21
3.4. LOGIN CONTROLLER.....	22
3.5. REGISTER CONTROLLER.....	22
3.6. USER MANAGEMENT CONTROLLER.....	23
3.7. VNF MANAGEMENT CONTROLLER.....	24
3.8. NFSTORE IMAGES MANAGEMENT CONTROLLER.....	24
3.9. SERVICES MANAGEMENT CONTROLLER .....	25
3.10. BROKER CONTROLLER .....	25
<b>4. USERS DASHBOARD INTERFACING WITH REST T-NOVA MARKETPLACE MODULES.....</b>	<b>26</b>
4.1. APIs DEFINITION.....	26
4.1.1. <i>Billing</i> .....	26
4.1.2. <i>User Management, Authentication and Access control Module</i> .....	31
4.1.3. <i>SLA Management</i> .....	39
4.1.4. <i>Billing</i> .....	45
4.1.5. <i>Brokerage</i> .....	49
4.1.6. <i>Function Store</i> .....	51
4.1.7. <i>Business Service Catalogue</i> .....	54
4.1.8. <i>Service Selection</i> .....	58
4.1.9. <i>Orchestrator</i> .....	59
<b>5. GRAPHICAL USER INTERFACE AND FUNCTIONALITIES .....</b>	<b>61</b>

5.1. USER REGISTRATION.....	61
5.2. USER LOGIN .....	61
5.3. VNF PROVIDER .....	62
5.3.1. <i>VNF Creation</i> .....	62
5.3.2. <i>VNF Listing</i> .....	66
5.3.3. <i>VNF Tools</i> .....	67
5.3.4. <i>Images</i> .....	69
5.4. SERVICE PROVIDER.....	69
5.4.1. <i>Service/NSD Creation</i> .....	70
5.4.2. <i>NSD Listing</i> .....	74
5.4.3. <i>NSD Tools</i> .....	74
5.5. CUSTOMER.....	75
5.5.1. <i>Customer – Service Selection</i> .....	75
5.5.2. <i>Customer - Service Purchase</i> .....	76
5.5.3. <i>Customer Services</i> .....	76
5.6. ADMINISTRATOR .....	77
5.6.1. <i>User Management</i> .....	77
<b>6. VALIDATION .....</b>	<b>80</b>
6.1. FUNCTIONAL VERIFICATION .....	80
6.2. REQUIREMENTS FULFILLMENT .....	82
<b>7. CONCLUSIONS .....</b>	<b>84</b>
<b>8. REFERENCES .....</b>	<b>85</b>
<b>9. ANNEX A:.....</b>	<b>86</b>

## Index of Figures

Figure 1-1 Dashboard views.....	8
Figure 2-1 Single Page Application .....	16
Figure 2-2 Dashboard in monolithic deployment .....	17
Figure 2-3 Dashboard deployed as a micro-service .....	18
Figure 2-4 Dashboard deployed as a containerised micro-service .....	19
Figure 3-1 Routing module .....	20
Figure 3-2 Bootstrap authentication function.....	21
Figure 3-3 Root controller.....	21
Figure 3-4 Login controller.....	22
Figure 3-5 Register controller.....	23
Figure 3-6 User management controller .....	23
Figure 1-1 VNF management controller.....	24
Figure 1-1 NFStore Images Management controller.....	24
Figure 1-1 Service Management controller.....	25
Figure 1-1 Broker controller .....	25
Figure 1-1 User registration .....	61
Figure 1-1 Dashboard's login page.....	62
Figure 1-1 VNF Providers home page .....	62
Figure 1-1 VNF Creation - Step 1: VNF Basic Information .....	63
Figure 1-1 VNF Creation - Step 2: VNF Composition, SLA Flavors .....	64
Figure 1-1 VNF Creation - Step 2: VNF Composition, Virtual Machines.....	64
Figure 1-1 VNF Creation - Step 2: VNF Composition, Virtual Links.....	65
Figure 1-1 VNF Creation - Step 3: SLA .....	65
Figure 1-1 VNF Creation - Step 3: SLA, Assurance Parameters .....	66
Figure 1-1 VNF Creation - Step 4: Billing .....	66
Figure 1-1 VNF Listing .....	67
Figure 1-1 Generated VNFD Viewer.....	67
Figure 1-1 VNFD YAML Editor .....	68
Figure 1-1 VNF Diagram.....	68
Figure 1-1 Function Provider VNF Images.....	69
Figure 1-1 Service Provider.....	69
Figure 1-1 NSD Creation - Step 1: VNF Selection.....	70
Figure 1-1 NSD Creation - Step 1: VNF Selection, Trade Request .....	71
Figure 1-1 NSD Creation - Step 1: VNF Selection, Pending Trade Request .....	71
Figure 1-1 Trade Request FP View.....	71
Figure 1-1 Accepted Trade Offer.....	71
Figure 1-1 NSD Creation - Step 2: Basic Information.....	72
Figure 1-1 NSD Creation - Step 3: Service Composition and SLA.....	73
Figure 1-1 NSD Creation - Step 4: Assurance Parameters .....	73
Figure 1-1 NSD Creation - Step 4: Assurance Parameters .....	74
Figure 1-1 NSD Listing.....	74
Figure 1-1 Generated NSD Viewer.....	75
Figure 1-1 NSD YAML Editor.....	75
Figure 1-1 Customer – Service Selection.....	76
Figure 1-1 Customer - Service Purchase .....	76

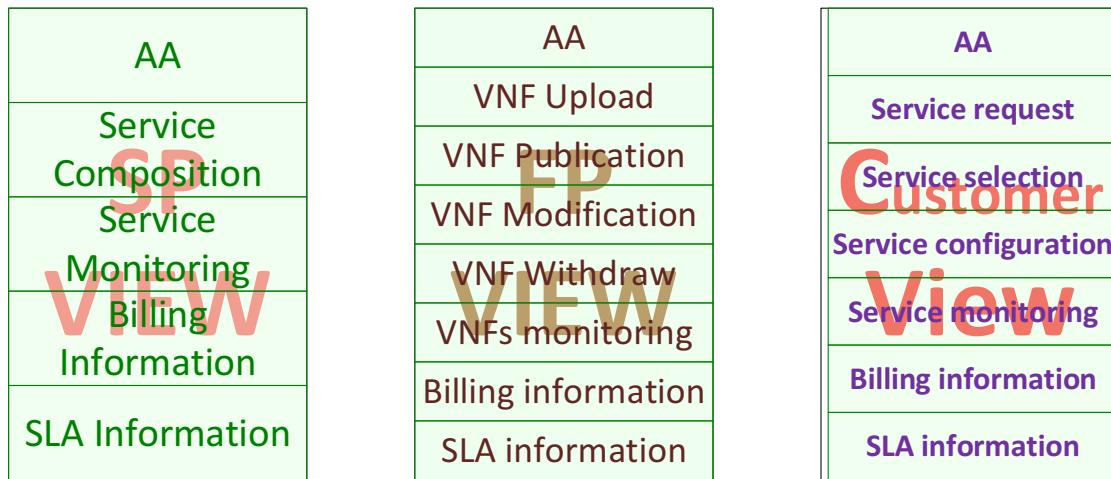
Figure 1-1 Customer Services .....	77
Figure 1-1 Administrator .....	77
Figure 1-1 User Management.....	78
Figure 1-1 User Creation.....	78
Figure 1-1 Edit User Profile.....	79

## Index of Tables

Table 1 SP dashboard view .....	9
Table 2 FP dashboard view .....	9
Table 3 Customer Dashboard view.....	9
Table 6-1 Dashboard functional verification.....	82
Table 6-2 Dashboard basic requirements.....	83

## 1. INTRODUCTION

As already stated at D2.42, the Dashboard constitutes the T-NOVA system's front-end and hosts the three views for the three basic stakeholders that will access the T-NOVA Marketplace: the Service Provider (SP), the Function Provider (FP) and the Customer. The main features of the Dashboard are presented in Figure 1-1.



**Figure 1-1 Dashboard views**

In summary, the SPs', FPs' and Customers' views of the dashboard will allow them to provide the functionalities shown in Table 1, Table 2 and Table 3 respectively.

Functionality	Short Explanation
AA	Authorization and Authentication of the respective role into the T-NOVA Dashboard.
Service composition	Graphical wizard that will help the SP to compose a new Network Service (NS) starting from the brokerage among the FPs owing the available VNFs.
Service monitoring	Graphical representation of all monitoring data for a selected or "consumed" Service.
Billing information	Graphical representation of the billing outcomes of selected or "consumed" service. There will be two types of billing information for the SP: <ul style="list-style-type: none"> <li>- Charges for the SP's customers (BSS functionality).</li> <li>- Invoices on behalf of its own suppliers, the FPs.</li> </ul>
SLA information	Details of the selected or "consumed" service based on how they respect the agreed SLA. The SP will have access to two different kinds of SLA contract and SLA monitoring information: <ul style="list-style-type: none"> <li>- SLA between SP and its customers (BSS)</li> <li>- SLA agreed with his its suppliers, the FPs</li> </ul>

**Table 1 SP dashboard view**

<b>Functionality</b>	<b>Short Explanation</b>
AA	Authorization and Authentication of the respective role into the T-NOVA dashboard.
VNF Upload	Graphical wizard that will help the FP to upload his VNF with the necessary parameters.
VNF Publication	Graphical representation for the FP to provide the last check in order to publish the uploaded VNF
VNF Modification	Small graphical wizard that provides the ability to the FP to modify the uploaded VNF.
VNF Withdraw	Graphical representation that gives to the FP the ability to remove an already published or uploaded VNF
VNFs monitoring	Graphical representation of all monitoring data for a selected or "consumed" NF.
Billing information	Graphical representation of the Billing outcomes for a selected or "consumed" NF.
SLA information	Information of the selected or "consumed" NFs based on the agreed SLA and its fulfillment.

**Table 2 FP dashboard view**

<b>Functionality</b>	<b>Short Explanation</b>
AA	Authorization and Authentication of the respective role into the T-NOVA Dashboard.
Service request	Graphical representation of the Services/Functions returned by the T-NOVA business service catalogue.
Service Selection	Graphical representation assisted by a check box providing the ability to the customer to select a service for consumption.
Service configuration	Small Graphical wizard providing to the customer predefined parameters for defining the selected service.
Service monitoring	Graphical representation of the data gathered from the monitoring modules.
Billing information	Graphical representation of the billing outcomes of selected or "consumed" service.
SLA information	Details of the selected or "consumed" Service based on how they respect the agreed SLA.

**Table 3 Customer Dashboard view**

## 1.1. Deliverable's structure

The rest of the document is composed by the following sections:

Section 1 Introduction: Here, we give a short overview of the Dashboard and describe its functionalities.

Section 2 Requirements Overview: Here, we summarise the requirements the Dashboard must meet.

Section 3 Users Dashboard Design: This section discusses in brief the available web development frameworks and the existing software development and deployment approaches focusing in the ones exhibiting better scalability and interoperability.

Section 4 Users Dashboard Implementation: The implementation of each distinct functional part of the Users Dashboard is presented here.

Section 5 Users Dashboard Interfacing with T-NOVA Management Modules: In this section, we describe the interfaces of the Users Dashboard with the T-NOVA's management modules.

Section 6 Graphical User Interface and Functionalities: This section gives a step by step guide on how to use the implemented Dashboard.

Section 7 Conclusions: The conclusions of this document are given in this section.

## 2. USERS DASHBOARD DESIGN

### 2.1. Available Web development frameworks

#### 2.1.1. Frontend

##### 2.1.1.1. Semantic-UI

Semantic-UI [1] is a new front-end development framework rich in features and with a robust API. According to its development team, "*Semantic is a UI component framework based around useful principles from natural language.*" To achieve this, Semantic-UI exploits a semantic, descriptive language for naming its classes, favoring the use of natural language words instead of abbreviations, as its counterparts mainly do.

This framework's uniqueness consists in its structured form, and in the provision of several components—namely the Feed, Comment, Shape and Sidebar—and a real-time debug feature not available in other frontend frameworks. Semantic-UI builds its structure around five descriptive categories (used for defining re-usable UI components) namely the UI Element, UI Collection, UI View, UI Module and UI Behavior. Additional advantages of Semantic UI are its minimal and neutral styling, leaving customization open to the developer, its components portability, its very good documentation and finally its website where many examples on how to use its different components can be found.

##### 2.1.1.2. Bootstrap

*"Bootstrap is the most popular HTML, CSS, and JS framework for developing responsive, mobile first projects on the web."* – Bootstrap [2]; a claim not far from truth. Bootstrap is a powerful, open-source front-end framework developed by a Twitter team. It combines HTML5/HTML, CSS3/CSS, and Javascript code towards enabling developers to easily build user interface components. It comes with a free set of tools and data APIs that allow the creation of flexible and responsive web layouts and common interface components (e.g. Scrollspy, Typeahead) without even writing a single line of JavaScript code. Bootstrap's popularity builds on the following advantages:

- Faster development cycle: There is a plethora of predefined design templates and classes available to the developer.
- Responsiveness: Bootstrap's responsive features enable for the proper display of the web pages on heterogeneous devices and screen resolutions without any change in markup.
- Consistency: The same design templates and styles are provided to all Bootstrap components through a central library.
- User-friendly: A basic working knowledge of HTML and CSS suffices for starting development with Bootstrap.
- Compatibility: Bootstrap is compatible with all modern browsers such as Mozilla Firefox, Google Chrome, Safari, Internet Explorer, and Opera.

- Open Source: It is completely free to download and use.

## 2.1.2. Middleware

### 2.1.2.1. AngularJS

AngularJS [3] is an open-source web application framework, adhering to the Model-View-Controller (MVC) architecture, designed for creating single page web applications. In this direction, it exploits declarative programming for creating user interfaces and for connecting the software components to each other. AngularJS most notable feature is the two-way data binding where changes in any model are automatically reflected to the related views and, vice versa, any modification in the views leads to the respective models update. This feature along with advantages like the inherently support of MVC, the use of conventional HTML as the declarative language for defining interfaces and the use of directives for bringing additional functionality to HTML have made AngularJS a very popular choice amongst web developers.

### 2.1.2.2. jQuery

jQuery [4] is a JavaScript library that simplifies operations like HTML document traversal and manipulation (Document Object Model elements manipulation), event handling, animation, and Ajax by offering an easy-to-use API that is compatible with a multitude of browsers. Among its most notable features are the event assignment and the event callback function definitions, both done once in the code. The main advantages of using this library are:

- JavaScript and HTML separation: Javascript is used for adding event handlers to DOM, rather than adding HTML event attributes to call JavaScript functions.
- Brevity and clarity: Features like chainable functions and shorthand function names enable for brevity and clarity.
- Cross-browser compatible: jQuery provides one consistent interface that handles the interoperability amongst the browsers' different JavaScript engines,
- Extensible: New events, elements, and methods can be easily added and then reused as a plugin.

### 2.1.2.3. Ember.js

Ember.js [5] is an open source, JavaScript application framework, adhering to the Model-View-Controller (MVC) model, used for, as its developers' state, "*creating ambitious web applications*". It follows the Convention over Configuration (CoC), and the Don't Repeat Yourself (DRY) principles. Ember.js provides a series of tools and incorporates common idioms and best practices into the framework for reducing the amount of written code. Although it is a web framework, it is also possible to exploit it for building desktop and mobile applications. Ember was designed around several key ideas:

- Focus on ambitious web applications: Ember supports full MVC in contrast with most of its counterparts focusing in the V (view) part of MVC.
- Production friendly: Enables the developer to be productive immediately. In this direction, Ember utilises a pluggable architecture and offers a Command Line Interface (CLI) that provides a standard application structure and build pipeline.
- Stability without stagnation: Assures backward compatibility while still innovating and evolving the framework.

Ember 2.0 was released August 13, 2015 and the most important changes occur in the view layer and include the one way data flow by default, the "Just refresh it" when something changes, the introduction of standard lifecycle hooks for components and the exploitation of Glimmer rendering engine aiming to improve re-render performance.

#### 2.1.2.4. React

React [6] is an open-source JavaScript library and is commonly exploited as the V in MVC. The library is highly interoperable as it makes no assumptions about the rest of the used technology stack. React abstracts the DOM, thus offering a simpler programming model and better performance. Contrary to its counterparts, React can also render on the server using Node. React aims to help developers building large applications that use time varying data. Its main features are:

- Use of vanilla JavaScript: React uses JavaScript's features for most of its operations (this explains React's lightweight API).
- One Way Data Flow: React utilises the simpler one way data flow than the more complex two way data flow. In this context, when the properties on a React component are updated, that component is re-rendered.
- Virtual DOM: React maintains its own virtual DOM, rather than using solely the browser's DOM. This allows the library to determine more efficient which parts of the browser's DOM need an update.
- Server-Side Rendering: Server-side rendering is a unique feature of React, especially important for high-traffic websites where the user experience (e.g. web page speed loading) must be excellent.

### 2.1.3. Back-End

#### 2.1.3.1. Django

Django [7] is a free, open source, MVC aware, high-level Python Web framework promoting rapid development and clean, pragmatic design. As its developer states "*Django makes it easier to build better Web apps more quickly and with less code.*" Django eases the creation of complex, database-driven websites and emphasizes reusability and "pluggability" of components, adhering to the principle of DRY (Don't Repeat Yourself). It uses an object-relational mapper (ORM) that mediates between data models (defined as Python classes) and a relational database ("Model"); a system

for processing HTTP requests with a web templating system ("View") and a regular-expression-based URL dispatcher ("Controller"). Django is highly extensible and rich in features, some of them are:

- a lightweight and standalone web server for development and testing
- a form serialization and validation system which can translate between HTML forms and values
- a template system that utilises the concept of inheritance
- a caching framework
- support for middleware classes
- an internal dispatcher system
- an internationalization system
- a XML/JSON serialization system
- a system for extending the capabilities of the template engine
- an interface to Python's built in unit test framework
- an extensible authentication system
- a dynamic administrative interface
- tools for generating RSS and Atom syndication feeds and Google Sitemaps
- built-in mitigation for cross-site request forgery, cross-site scripting, SQL injection, password cracking and other typical web attacks
- a framework for creating GIS applications

### 2.1.3.2. Flask

Flask [8] is a BSD licensed, Python micro web application framework built with a small core and easy-to-extend philosophy. Flask is based on the Werkzeug WSGI (Web Server Gateway Interface) toolkit and Jinja2 template engine but it does not because it does not presume the use of a particular tool or library—It has no database abstraction layer, form validation, or any other components where pre-existing third-party libraries provide common functions. However, it is easy to add extensions to Flask and there are already a number of them providing object-relational mappers, form validation, upload handling, various open authentication technologies and several common framework related tools. Some of its features include:

- Provision of a development server and debugger
- Integrated support for unit testing
- RESTful request dispatching
- Jinja2 templating
- Support for secure cookies
- 100% WSGI 1.0 compliant
- Unicode-based
- Extensive documentation
- Google App Engine compatibility
- Extensions available to enhance features desired

### 2.1.3.3. Symfony

According to its developers, "*Symfony is a set of PHP Components, a Web Application framework, a Philosophy, and a Community — all working together in harmony.*" Symfony [9], an open source project, relies on PHP, the largest web development and has a vibrant and growing community (e.g. eZ Platform uses Symfony's Full Stack, Drupal 8 uses many Symfony Components). The framework introduces a truly unique HTTP and HTTP cache handling system by being an HTTP-centric request/response framework. In addition, it supports the use of advance features like ESI (Edge Side Includes) for separating the different parts of the web application.

### 2.1.3.4. YII

Yii [10] is an open-source, MVC aware, PHP framework used for developing Web 2.0 applications, promoting clean, DRY (Don't Repeat Yourself) design, and supporting rapid development. It offers excellent documentation and has a supportive community. Features like Database Access Objects (DAO), Active Record, and programmatic Database migrations ease the effort for developing database-powered web applications. In addition, it has built in support for form input, validation, Ajax, and built-in authentication. Yii also provides a built-in code generation tool, named Gii, speeding application development. It also integrates well with other frameworks (e.g. Zend, PEAR) and supports I18N for providing localized versions of the developed applications. Finally, Yii supports PSR-4 class auto-loading, provides a RESTful API framework and a documentation generator.

## 2.1.4. Summary

Based on the factors, of maturity, community support, number of supported features, extensibility and interoperability, we selected Bootstrap, Angular and Django as frontend, middleware and backend Web development frameworks respectively.

## 2.2. Development Approaches

### 2.2.1. Server-Centric Web Application (SCWA)

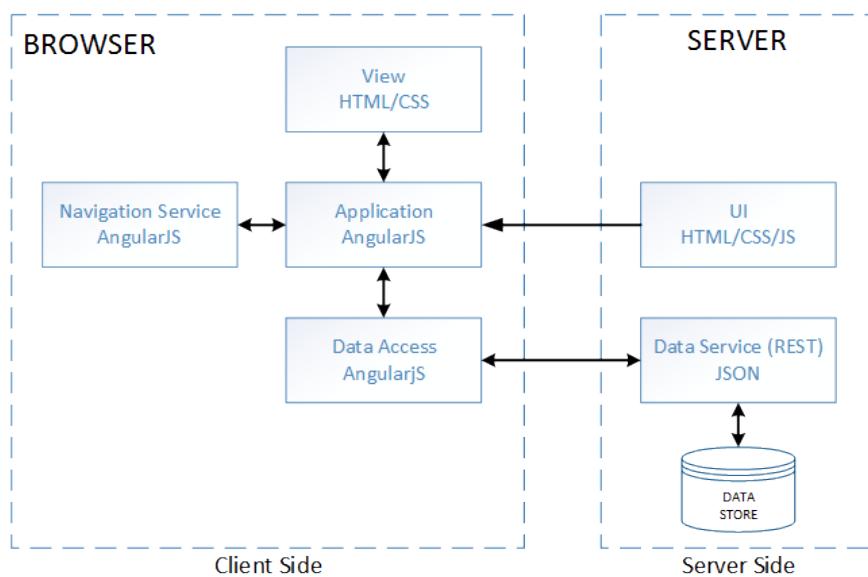
Server-Centric Web Application (SCWA) uses a Server to collect data, process them and serve the resulting HTML page to the client's browser. The disadvantage of this approach is that the page is posted back to the server; this introduces communication and processing overhead that can decrease the overall performance as it forces the user to wait for the page to be processed and recreated.

### 2.2.2. Browser-Centric Web Application (BCWA)

Browser-Centric Web Application (BCWA) approach embeds all the functional parts (e.g. Scripts) on the client's side and executes them on the client's Internet browser.

The advantage of BCWA architecture is the faster response time and less overhead (e.g. Data, Processing Power) on the web server. Additionally, BSWA is ideal when the page elements need to be changed without the need to contact the database.

A very popular technology belonging to the BCWA category is Single Page Application (SPA). Essentially SPA is a web application that fits in a single page and loads on the initial page request. SPA is suitable for web-centric applications that handle a large amount of data by exploiting, when needed, asynchronous download of features (HTML templates/JSON data) and by re-rendering locally any part of the interface without requiring from the server to re-render the complete HTML page. Finally, SPA allows web developers to give a "native-application" like experience to the end-user.



**Figure 2-1 Single Page Application**

### 2.2.3. Summary

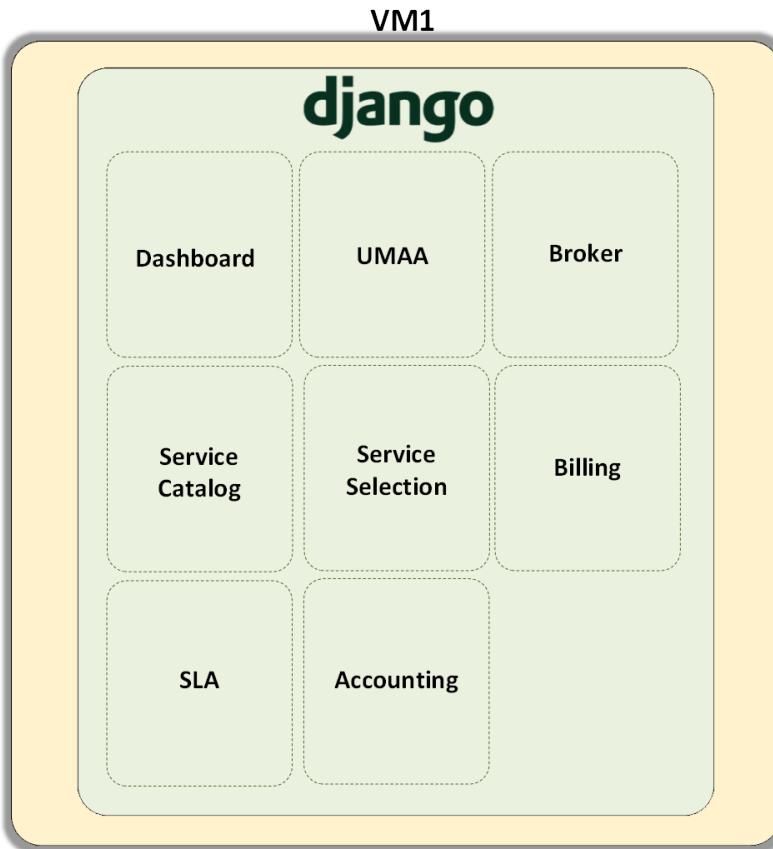
For the Dashboard's implementation, the Browser-Centric Web Application architecture was chosen for the following two reasons: scalability and faster response time (excluding the initial load time). For the former, BCWA is more suitable as it diffuses the processing load from the server to the clients; in other words all the code is downloaded, from the server, at the session's beginning and then it runs inside the clients' browsers. For the latter, as all (or almost all) of the data are inside the browser the time for getting, processing and displaying them is usually far less than the time needed to request and get the data from the server (SCWA paradigm).

## 2.3. Deployment Architectures

### 2.3.1. Monolithic

A monolithic architecture is defined in the traditional client-server model. In this architecture all the functionalities are contained into a single service, which is the most

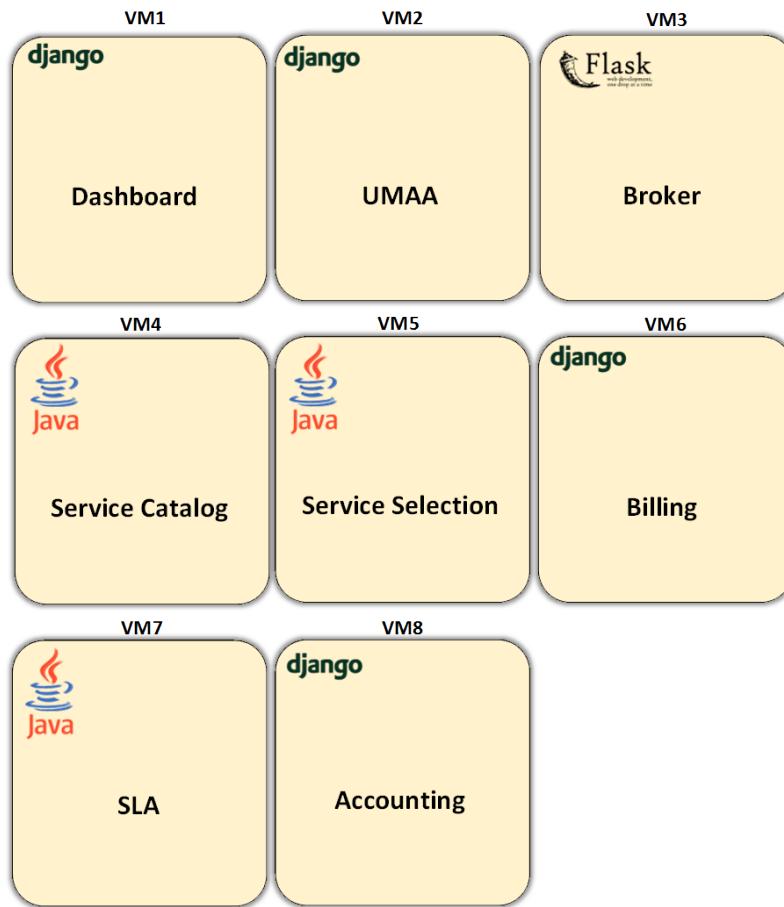
common way to develop and package web applications. A monolithic web application typically composed of the client, the server and the database. In this context, the Users' Dashboard along with the other modules are developed as a single application, exploiting one common web development framework (Django in Figure 2-2), and are deployed in one virtual machine.



**Figure 2-2 Dashboard in monolithic deployment**

### 2.3.2. Micro-services

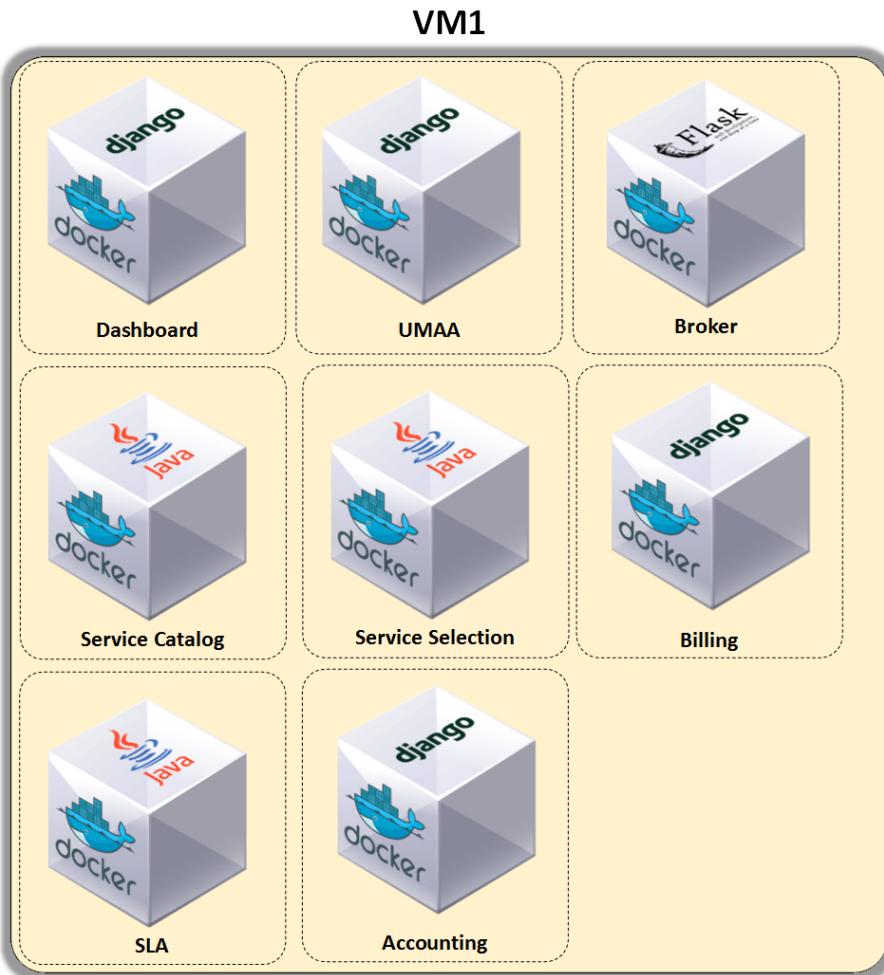
Micro-services architectural paradigm allows for the decomposition of one heavy application in several smaller components, named micro-services, which can be distributed across multiple locations. Each micro-service focuses in the execution of one or a small set of tasks, is independently deployable and uses, usually, an HTTP API for communicating with the rest micro-services. In this context, and as Figure 2-3 depicts, each T-NOVA module may be deployed as a distinct micro-service, exploiting a different development framework. Moreover, and for avoiding library versioning incompatibilities and conflicts, each micro-service can be deployed in a distinct virtual machine.



**Figure 2-3 Dashboard deployed as a micro-service**

### 2.3.3. Containerized micro-services

As Figure 2-4 depicts containerised micro-services refer to the deployment of each distinct micro-service inside a docker. Docker [11] is a container that packages all the software (e.g. code, libraries, third-party tools, etc.) needed to implement and run the micro-service. This allows the deployment of the micro-service, on any computer, on any infrastructure and in any cloud. In this context, each T-NOVA Marketplace module may be deployed as a distinct micro-service, exploiting a different development framework as in the “plain” micro-service case; however, with dockers there are no library versioning incompatibilities and conflicts and therefore all micro-services can be deployed in one virtual machine.



**Figure 2-4 Dashboard deployed as a containerised micro-service**

### 2.3.4. Summary

For the Dashboard's deployment, the containerised micro-service approach was selected. This will enable each developer assigned with the task of building a T-NOVA module to use its preferable development framework without having to take into account the selections of the other modules' developers (this may increase the development speed). Furthermore, the exploitation of docker technology enables for the deployment of all the modules in one virtual machine (VM) without having to worry about libraries incompatibilities and conflicts thus reducing the managerial and processing overhead needed to run a distinct VM for each micro-service.

## 3. USERS DASHBOARD IMPLEMENTATION

This section presents the implementation of the users' Dashboard in a series of code snippets. Each code snippet provides a snapshot of the code written to realise the Dashboard's functionalities.

### 3.1. Routing module

This module routes the users' requests (e.g. login, register) to the controller responsible for handling them.

```

1 // states definition
2 angular.module('dashboard').config(function ($stateProvider, $urlRouterProvider) {
3
4     $urlRouterProvider.otherwise("/index/dashboard");
5
6     $stateProvider
7         .state('login', {
8             url: "/login",
9             templateUrl: "/static/dashboard/templates/login.html"
10    })
11    .state('register', {
12        url: "/register",
13        templateUrl: "/static/dashboard/templates/register.html"
14    })
15    .state('index', {
16        url: "/index",
17        templateUrl: "/static/dashboard/templates/index.html"
18    })
19    .state('index.profile', {
20        url: "/profile",
21        templateUrl: "/static/dashboard/templates/index.profile.html"
22    })
23    .state('index.dashboard', {
24        url: "dashboard",
25        templateUrl: "/static/dashboard/templates/index.dashboard.html"
26    })
27    .state('index.users', {
28        url: "/users",
29        templateUrl: "/static/dashboard/templates/index.users.html"
30    })
31    .state('index.users.list', {
32        url: "/list",
33        templateUrl: "/static/dashboard/templates/index.users.list.html"
34    })
35    .state('index.users.new', {
36        url: "/new",
37        templateUrl: "/static/dashboard/templates/index.users.new.html"
38    })
39    .state('index.users.edit', {
40        url: "/{userId:int}/edit",
41        templateUrl: "/static/dashboard/templates/index.users.edit.html",
42        controller: function ($scope, $stateParams) {
43            $scope.userID = $stateParams.userID;
44        }
45    })
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
259
260
261
262
263
264
265
266
267
268
269
269
270
271
272
273
274
275
276
277
278
279
279
280
281
282
283
284
285
286
287
287
288
289
289
290
291
292
293
294
295
296
297
297
298
299
299
300
301
302
303
304
305
306
307
307
308
309
309
310
311
312
313
314
315
316
317
317
318
319
319
320
321
322
323
324
325
326
327
327
328
329
329
330
331
332
333
334
335
336
337
338
339
339
340
341
342
343
344
345
346
347
348
349
349
350
351
352
353
354
355
356
357
358
359
359
360
361
362
363
364
365
366
367
368
369
369
370
371
372
373
374
375
376
377
378
379
379
380
381
382
383
384
385
386
387
387
388
389
389
390
391
392
393
394
395
396
397
398
399
399
400
401
402
403
404
405
406
407
408
409
409
410
411
412
413
414
415
416
417
418
419
419
420
421
422
423
424
425
426
427
428
429
429
430
431
432
433
434
435
436
437
438
439
439
440
441
442
443
444
445
446
447
448
449
449
450
451
452
453
454
455
456
457
458
459
459
460
461
462
463
464
465
466
467
468
469
469
470
471
472
473
474
475
476
477
478
479
479
480
481
482
483
484
485
486
487
488
489
489
490
491
492
493
494
495
496
497
498
499
499
500
501
502
503
504
505
506
507
508
509
509
510
511
512
513
514
515
516
517
518
519
519
520
521
522
523
524
525
526
527
528
529
529
530
531
532
533
534
535
536
537
538
539
539
540
541
542
543
544
545
546
547
548
549
549
550
551
552
553
554
555
556
557
558
559
559
560
561
562
563
564
565
566
567
568
569
569
570
571
572
573
574
575
576
577
578
579
579
580
581
582
583
584
585
586
587
588
589
589
590
591
592
593
594
595
596
597
598
599
599
600
601
602
603
604
605
606
607
608
609
609
610
611
612
613
614
615
616
617
618
619
619
620
621
622
623
624
625
626
627
628
629
629
630
631
632
633
634
635
636
637
638
639
639
640
641
642
643
644
645
646
647
648
649
649
650
651
652
653
654
655
656
657
658
659
659
660
661
662
663
664
665
666
667
668
669
669
670
671
672
673
674
675
676
677
678
679
679
680
681
682
683
684
685
686
687
688
689
689
690
691
692
693
694
695
696
697
697
698
699
699
700
701
702
703
704
705
706
707
708
709
709
710
711
712
713
714
715
716
717
718
719
719
720
721
722
723
724
725
726
727
728
729
729
730
731
732
733
734
735
736
737
738
739
739
740
741
742
743
744
745
746
747
748
748
749
750
751
752
753
754
755
756
757
758
759
759
760
761
762
763
764
765
766
767
768
769
769
770
771
772
773
774
775
776
777
778
779
779
780
781
782
783
784
785
786
787
788
789
789
790
791
792
793
794
795
796
797
797
798
799
799
800
801
802
803
804
805
806
807
808
809
809
810
811
812
813
814
815
816
817
818
819
819
820
821
822
823
824
825
826
827
828
829
829
830
831
832
833
834
835
836
837
838
839
839
840
841
842
843
844
845
846
847
848
848
849
850
851
852
853
854
855
856
857
858
859
859
860
861
862
863
864
865
866
867
868
869
869
870
871
872
873
874
875
876
877
878
879
879
880
881
882
883
884
885
886
887
888
888
889
889
890
891
892
893
894
895
896
897
898
898
899
899
900
901
902
903
904
905
906
907
908
909
909
910
911
912
913
914
915
916
917
918
919
919
920
921
922
923
924
925
926
927
928
929
929
930
931
932
933
934
935
936
937
938
939
939
940
941
942
943
944
945
946
947
948
948
949
950
951
952
953
954
955
956
957
958
959
959
960
961
962
963
964
965
966
967
968
969
969
970
971
972
973
974
975
976
977
978
979
979
980
981
982
983
984
985
986
987
987
988
989
989
990
991
992
993
994
995
996
997
998
999
999
1000
1000
1001
1002
1003
1004
1005
1006
1007
1008
1009
1009
1010
1011
1012
1013
1014
1015
1016
1017
1018
1019
1019
1020
1021
1022
1023
1024
1025
1026
1027
1028
1029
1029
1030
1031
1032
1033
1034
1035
1036
1037
1038
1039
1039
1040
1041
1042
1043
1044
1045
1046
1047
1048
1048
1049
1050
1051
1052
1053
1054
1055
1056
1057
1058
1059
1059
1060
1061
1062
1063
1064
1065
1066
1067
1068
1069
1069
1070
1071
1072
1073
1074
1075
1076
1077
1078
1079
1079
1080
1081
1082
1083
1084
1085
1086
1087
1088
1088
1089
1090
1091
1092
1093
1094
1095
1096
1097
1097
1098
1099
1099
1100
1101
1102
1103
1104
1105
1106
1107
1108
1109
1109
1110
1111
1112
1113
1114
1115
1116
1117
1118
1119
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1129
1130
1131
1132
1133
1134
1135
1136
1137
1138
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187
1187
1188
1189
1190
1191
1192
1193
1194
1195
1195
1196
1197
1198
1199
1199
1200
1201
1202
1203
1204
1205
1206
1207
1208
1209
1209
1210
1211
1212
1213
1214
1215
1216
1217
1218
1218
1219
1220
1221
1222
1223
1224
1225
1226
1227
1228
1229
1229
1230
1231
1232
1233
1234
1235
1236
1237
1238
1238
1239
1240
1241
1242
1243
1244
1245
1246
1247
1248
1248
1249
1250
1251
1252
1253
1254
1255
1256
1257
1258
1259
1259
1260
1261
1262
1263
1264
1265
1266
1267
1268
1269
1269
1270
1271
1272
1273
1274
1275
1276
1277
1278
1278
1279
1280
1281
1282
1283
1284
1285
1286
1287
1287
1288
1289
1289
1290
1291
1292
1293
1294
1295
1296
1297
1297
1298
1299
1299
1300
1301
1302
1303
1304
1305
1306
1307
1308
1309
1309
1310
1311
1312
1313
1314
1315
1316
1317
1318
1318
1319
1320
1321
1322
1323
1324
1325
1326
1327
1328
1328
1329
1330
1331
1332
1333
1334
1335
1336
1337
1338
1338
1339
1340
1341
1342
1343
1344
1345
1346
1347
1348
1348
1349
1350
1351
1352
1353
1354
1355
1356
1357
1358
1359
1359
1360
1361
1362
1363
1364
1365
1366
1367
1368
1368
1369
1370
1371
1372
1373
1374
1375
1376
1377
1378
1378
1379
1380
1381
1382
1383
1384
1385
1386
1387
1387
1388
1389
1389
1390
1391
1392
1393
1394
1395
1396
1397
1397
1398
1399
1399
1400
1401
1402
1403
1404
1405
1406
1407
1408
1409
1409
1410
1411
1412
1413
1414
1415
1416
1417
1417
1418
1419
1420
1421
1422
1423
1424
1425
1426
1427
1427
1428
1429
1429
1430
1431
1432
1433
1434
1435
1436
1437
1437
1438
1439
1440
1441
1442
1443
1444
1445
1446
1447
1447
1448
1449
1449
1450
1451
1452
1453
1454
1455
1456
1457
1458
1458
1459
1460
1461
1462
1463
1464
1465
1466
1467
1467
1468
1469
1469
1470
1471
1472
1473
1474
1475
1476
1477
1477
1478
1479
1479
1480
1481
1482
1483
1484
1485
1486
1487
1487
1488
1489
1489
1490
1491
1492
1493
1494
1495
1496
1497
1497
1498
1499
1499
1500
1501
1502
1503
1504
1505
1506
1507
1508
1509
1509
1510
1511
1512
1513
1514
1515
1516
1517
1517
1518
1519
1519
1520
1521
1522
1523
1524
1525
1526
1527
1527
1528
1529
1529
1530
1531
1532
1533
1534
1535
1536
1537
1537
1538
1539
1539
1540
1541
1542
1543
1544
1545
1546
1547
1547
1548
1549
1549
1550
1551
1552
1553
1554
1555
1556
1557
1558
1558
1559
1560
1561
1562
1563
1564
1565
1566
1567
1567
1568
1569
1569
1570
1571
1572
1573
1574
1575
1576
1577
1577
1578
1579
1579
1580
1581
1582
1583
1584
1585
1586
1587
1587
1588
1589
1589
1590
1591
1592
1593
1594
1595
1596
1597
1597
1598
1599
1599
1600
1601
1602
1603
1604
1605
1606
1607
1608
1609
1609
1610
1611
1612
1613
1614
1615
1616
1617
1617
1618
1619
1619
1620
1621
1622
1623
1624
1625
1626
1627
1627
1628
1629
1629
1630
1631
1632
1633
1634
1635
1636
1637
1637
1638
1639
1639
1640
1641
1642
1643
1644
1645
1646
1647
1647
1648
1649
1649
1650
1651
1652
1653
1654
1655
1656
1657
1658
1658
1659
1660
1661
1662
1663
1664
1665
1666
1667
1667
1668
1669
1669
1670
1671
1672
1673
1674
1675
1676
1677
1677
1678
1679
1679
1680
1681
1682
1683
1684
1685
1686
1687
1687
1688
1689
1689
1690
1691
1692
1693
1694
1695
1696
1697
1697
1698
1699
1699
1700
1701
1702
1703
1704
1705
1706
1707
1708
1709
1709
1710
1711
1712
1713
1714
1715
1716
1717
1717
1718
1719
1719
1720
1721
1722
1723
1724
1725
1726
1727
1727
1728
1729
1729
1730
1731
1732
1733
1734
1735
1736
1737
1737
1738
1739
1739
1740
1741
1742
1743
1744
1745
1746
1747
1747
1748
1749
1749
1750
1751
1752
1753
1754
1755
1756
1757
1758
1758
1759
1760
1761
1762
1763
1764
1765
1766
1767
1767
1768
1769
1769
1770
1771
1772
1773
1774
1775
1776
1777
1777
1778
1779
1779
1780
1781
1782
1783
1784
1785
1786
1787
1787
1788
1789
1789
1790
1791
1792
1793
1794
1795
1796
1797
1797
1798
1799
1799
1800
1801
1802
1803
1804
1805
1806
1807
1808
1809
1809
1810
1811
1812
1813
1814
1815
1816
1817
1817
1818
1819
1819
1820
1821
1822
1823
1824
1825
1826
1827
1827
1828
1829
1829
1830
1831
1832
1833
1834
1835
1836
1837
1837
1838
1839
1839
1840
1841
1842
1843
1844
1845
1846
1847
1847
1848
1849
1849
1850
1851
1852
1853
1854
1855
1856
1857
1858
1858
1859
1860
1861
1862
1863
1864
1865
1866
1867
1867
1868
1869
1869
1870
1871
1872
1873
1874
1875
1876
1877
1877
1878
1879
1879
1880
1881
1882
1883
1884
1885
1886
1887
1887
1888
1889
1889
1890
1891
1892
1893
1894
1895
1896
1897
1897
1898
1899
1899
1900
1901
1902
1903
1904
1905
1906
1907
1908
1909
1909
1910
1911
1912
1913
1914
1915
1916
1917
1917
1918
1919
1919
1920
1921
1922
1923
1924
1925
1926
1927
1927
1928
1929
1929
1930
1931
1932
1933
1934
1935
1936
1937
1937
1938
1939
1939
1940
1941
1942
1943
1944
1945
1946
1947
1947
1948
1949
1949
1950
1951
1952
1953
1954
1955
1956
1957
1958
1958
1959
1960
1961
1962
1963
1964
1965
1966
1967
1967
1968
1969
1969
1970
1971
1972
1973
1974
1975
1976
1977
1977
1978
1979
1979
1980
1981
1982
1983
1984
1985
1986
1987
1987
1988
1989
1989
1990
1991
1992
1993
1994
1995
1996
1997
1997
1998
1999
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2009
2010
2011
2012
2013
2014
2015
2016
2017
2017
2018
2019
2019
2020
2021
2022
2023
2024
2025
2026
2027
2028
2029
2029
2030
2031
2032
2033
2034
2035
2036
2037
2038
2038
2039
2040
2041
2042
2043
2044
2045
2046
2047
2047
2048
2049
2049
2050
2051
2052
2053
2054
2055
2056
2057
2058
2058
2059
2060
2061
2062
2063
2064
2065
2066
2067
2067
2068
2069
2069
2070
2071
2072
2073
2074
2075
2076
2077
2077
2078
2079
2079
2080
2081
2082
2083
2084
2085
2086
2087
2087
2088
2089
2089
2090
2091
2092
2093
2094
2095
2096
2097
2097
2098
2099
2099
2100
2101
2102
2103
2104
2105
2106

```

```

1  angular.module('dashboard', ['ui.bootstrap', 'ngSanitize', 'angularModalService', 'ui.select', 'ui.checkbox', 'ui-rangeSlider']);
2
3  angular.module('dashboard').run(['$state', '$rootScope', '$timeout', '$cookies', 'Restangular', function ($state, $rootScope,
4    $rootScope.isAuthorized = false;
5    $rootScope.$state = $state;
6
7    var appStarted = 0; // flag to redirect only once when app is started
8    $rootScope.$on('$stateChangeStart',
9      function (event, toState, toParams, fromState, fromParams) {
10        if (appStarted) return;
11        appStarted = 1;
12        event.preventDefault(); // prevents from resolving requested url
13
14        var token = $cookies.get('token');
15        if (token) {
16          console.log("JWT token exists");
17          Restangular.setDefaultHeaders({Authorization: 'JWT ' + token});
18          $rootScope.isAuthorized = true;
19          $state.go('index.dashboard');
20        } else {
21          console.log("JWT not found");
22          $state.go('login');
23        }
24      });
25    ]);
26  ]);
27
28  // change [{} ] default template tags to [[ ] ] because conflict with django's template tags
29 angular.module('dashboard').config(function ($interpolateProvider) {
30   $interpolateProvider.startSymbol('[[');
31   $interpolateProvider.endSymbol(']]');
32 });
33
34 // setup restangular base url
35 angular.module('dashboard').config(function (RestangularProvider) {
36   RestangularProvider.setBaseUrl('http://marketplace.t-nova.eu');
37   RestangularProvider.setRequestSuffix('/');
38 });
39

```

**Figure 3-2 Bootstrap authentication function**

### 3.3. Root controller

This controller contacts the UMAA module for checking the requesting user's privileges and based on them renders only the authorised content.

```

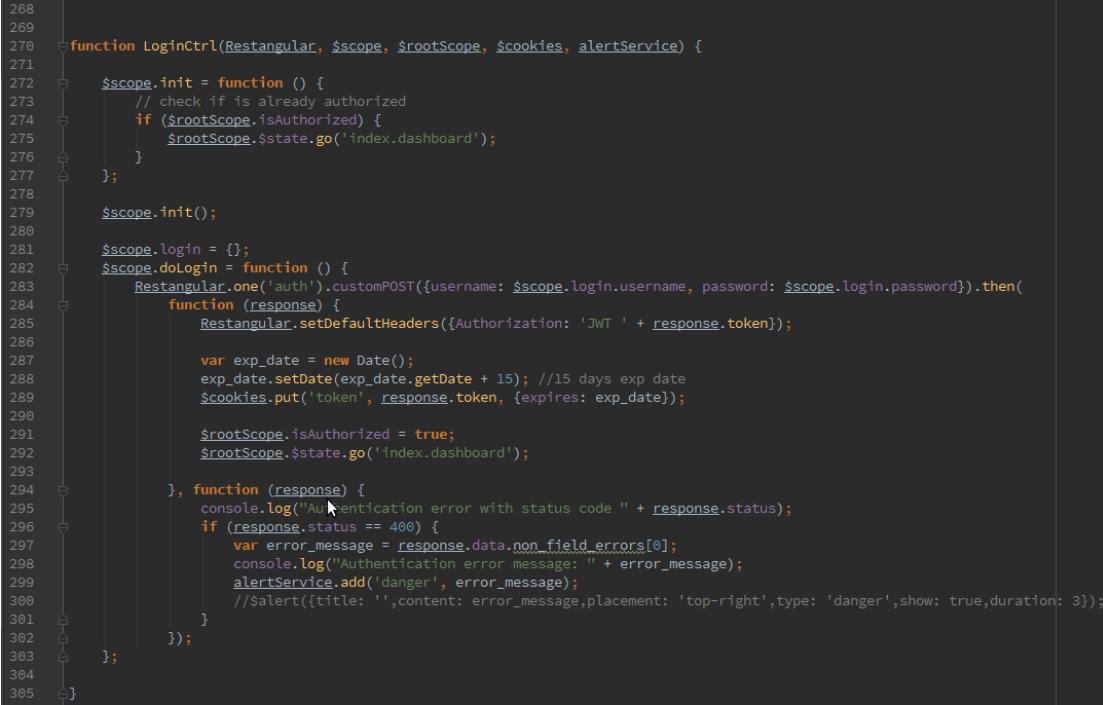
52  function RootCtrl(Restangular, $scope, $rootScope, $cookies, $state, $interval) {
53
54
55
56    // check if is not authorized
57    if (!$rootScope.isAuthorized) {
58      $rootScope.$state.go('login');
59    }
60
61    $scope.available_groups = [];
62    $scope.available_countries = [];
63    $scope.user_permissions = [];
64    $scope.user_profile = {};
65
66    $scope.getGroupName = function (group_id) {
67      var group_name = '';
68
69      angular.forEach($scope.available_groups, function (value, key) {
70        if (value.id == group_id) group_name = value.name;
71      });
72
73      return group_name;
74    };
75
76    $scope.getGroupID = function (group_name) {
77      var group_id = -1;
78
79      angular.forEach($scope.available_groups, function (value, key) {
80        if (value.name == group_name) group_id = value.id;
81      });
82
83      return group_id;
84    };
85
86    $scope.hasPerm = function (perm) {
87      var has_perm = false;
88
89      angular.forEach($scope.user_permissions, function (value, key) {
90        if (value == perm) has_perm = true;
91      });
92
93      return has_perm;
94    };

```

**Figure 3-3 Root controller**

### 3.4. Login controller

This controller contacts the UMAA module for authenticating the users.



```

268
269
270     function LoginCtrl($Restangular, $scope, $rootScope, $cookies, alertService) {
271
272         $scope.init = function () {
273             // check if is already authorized
274             if ($rootScope.isAuthorized) {
275                 $rootScope.$state.go('index.dashboard');
276             }
277         };
278
279         $scope.login = {};
280
281         $scope.doLogin = function () {
282             $Restangular.one('auth').customPOST({username: $scope.login.username, password: $scope.login.password}).then(
283                 function (response) {
284                     $Restangular.setDefaultHeaders({Authorization: 'JWT ' + response.token});
285
286                     var exp_date = new Date();
287                     exp_date.setDate(exp_date.getDate() + 15); //15 days exp date
288                     $cookies.put('token', response.token, {expires: exp_date});
289
290                     $rootScope.isAuthorized = true;
291                     $rootScope.$state.go('index.dashboard');
292
293                 }, function (response) {
294                     console.log("Authentication error with status code " + response.status);
295                     if (response.status == 400) {
296                         var error_message = response.data.non_field_errors[0];
297                         console.log("Authentication error message: " + error_message);
298                         alertService.add('danger', error_message);
299                         //alert({title: '', content: error_message, placement: 'top-right', type: 'danger', show: true, duration: 3});
300                     }
301                 });
302         };
303     };
304
305 }

```

**Figure 3-4 Login controller**

### 3.5. Register controller

This controller contacts the UMAA module for registering the new Dashboard's users, namely the new Customers, Service and Function Providers.

```

306
307     function RegisterCtrl($Restangular, $scope, $rootScope, alertService) {
308
309         $scope.init = function () {
310             // check if is already authorized
311             if ($rootScope.isAuthorized) {
312                 $rootScope.$state.go('index.dashboard');
313             }
314         };
315
316         $scope.init();
317
318         $scope.user = {};
319         $scope.user.group = '2';
320
321         $scope.doRegister = function () {
322
323             console.log($scope.user);
324             $scope.user.groups = [$scope.user.group];
325
326             $Restangular.one('/user-management/register/').customPOST($scope.user).then(
327                 function (response) {
328
329                     alertService.add('success', 'You have successfully registered!');
330                     $rootScope.$state.go('login');
331
332                 }, function (response) {
333                     console.log("Registration error with status code " + response.status);
334                     if (response.status == 400) {
335                         alertService.add('danger', 'Registration failed.');
336
337                         var html_error = '<ul>';
338                         angular.forEach(response.data, function (value, key) {
339                             html_error = html_error + '<li> Invalid ' + key + '</li>';
340                         });
341                         html_error = html_error + '</ul>';
342
343                 });
344
345             );
346
347         };
348
349     };
350
351     angular.module('app').controller('RegisterCtrl', RegisterCtrl);
352
353 
```

Figure 3-5 Register controller

### 3.6. User management controller

This controller allows the Dashboard's administrator to manage (create, modify, delete) the user accounts.

```

1  angular.module('dashboard').controller('UsersListCtrl', ['$Restangular', '$scope', '$rootScope', UsersListCtrl]);
2  angular.module('dashboard').controller('UserCreateCtrl', ['$Restangular', '$scope', '$rootScope', '$state', UserCreateCtrl]);
3  angular.module('dashboard').controller('UserEditCtrl', ['$Restangular', '$scope', '$rootScope', '$stateParams', UserEditCtrl]);
4  angular.module('dashboard').controller('UserProfileCtrl', ['$Restangular', '$scope', '$rootScope', UserProfileCtrl]);
5
6
7  function UsersListCtrl($Restangular, $scope, $rootScope) {
8
9      $scope.loading_users = false;
10     $scope.users = {};
11
12     $scope.loadUsers = function () {
13         $scope.loading_users = true;
14         $Restangular.all('user-management/users').getList().then(
15             function (response) {
16                 $scope.users = response;
17                 console.log("GetUserList " + response.length + " users found");
18                 $scope.loading_users = false;
19             }, function (response) {
20                 console.log("GetUserList error with status code " + response.status);
21                 console.log("GetUserList error message: " + response.data.detail);
22                 $scope.loading_users = false;
23             });
24     };
25
26     $scope.deleteUser = function (id, username) {
27         $scope.loading_users = true;
28         $Restangular.one("user-management/users", id).remove().then(
29             function () {
30                 $scope.loading_users = false;
31                 console.log("User " + username + " has been successfully deleted");
32                 $scope.loadUsers();
33             }, function (response) {
34                 console.log("DeleteUser error with status code " + response.status);
35                 console.log("DeleteUser error message: " + response.data.detail);
36                 $scope.loading_users = false;
37             });
38     };
39
40     $scope.userDeleteConfirm = function (id, username) {
41
42 
```

Figure 3-6 User management controller

### 3.7. VNF management controller

This controller interacts with the NFStore towards allowing Function Providers to create their functions.

```

1 angular.module('dashboard').controller('VNFCREATECtrl', ['$Restangular', '$scope', '$rootScope', '$state', 'ModalService']);
2 angular.module('dashboard').controller('VNFEditCtrl', ['$Restangular', '$scope', '$rootScope', '$state', '$stateParams', 'ModalService']);
3 angular.module('dashboard').controller('VNFListCtrl', ['$Restangular', '$scope', '$rootScope', 'ModalService', VNFLISTCtrl]);
4
5
6 function VNFCREATECtrl($Restangular, $scope, $rootScope, $state, ModalService){
7
8     $scope.vnf_types = [
9         {code:"TC", desc: "Traffic Classification"},  

10        {code:"FW", desc: "Firewall"},  

11        {code:"HG", desc: "Home Gateway"},  

12        {code:"SA", desc: "Security Appliance"}  

13    ];
14
15     $scope.memory_units = ['MB', 'GB'];
16     $scope.storage_units = ['MB', 'GB', 'TB'];
17     $scope.network_units = ['Kbps', 'Mbps', 'Gbps'];
18
19
20     $scope.available_bandwidths = [
21         "10Mbps",
22         "100Mbps",
23         "1Gbps",
24         "10Gbps",
25         "Unlimited"
26     ];
27
28
29     $scope.billing_model_types = [
30         {code: "PAYG", desc: "Pay-As-You-Go"},  

31         {code: "RS", desc: "Revenue Sharing"}  

32     ];
33
34     $scope.billing_currencies = [
35         {code: "EUR", desc: "Euro"},  

36         {code: "USD", desc: "US Dollars"}  

37     ];
38 }
```

Figure 3-7 VNF management controller

### 3.8. NFStore Images Management controller

This controller interacts with the NFStore towards allowing Function Providers to manage the images of their implemented functions.

```

1 angular.module('dashboard').controller('ImageListCtrl', ['$Restangular', '$scope', '$rootScope', 'ModalService', ImageListCtrl]);
2 angular.module('dashboard').controller('ImageUploadCtrl', ['$scope', '$cookies', 'Upload', ImageUploadCtrl]);
3
4 function ImageListCtrl($Restangular, $scope, $rootScope, ModalService) {
5
6     $scope.loading_images = false;
7     $scope.images = {};
8
9     $scope.loadImages = function () {
10         $scope.loading_images = true;
11         $Restangular.all('vnfs/images').getList().then(
12             function (response) {
13                 $scope.images = response;
14                 console.log("getImageList " + response.length + " Images found");
15                 $scope.loading_images = false;
16             },
17             function (response) {
18                 console.log("getImageList error with status code " + response.status);
19                 console.log("getImageList error message: " + response.data.detail);
20                 $scope.loading_images = false;
21             });
22
23     $scope.deleteImage = function (image_name) {
24         $scope.loading_images = true;
25         $Restangular.one('vnfs/images', image_name).remove().then(
26             function () {
27                 $scope.loading_images = false;
28                 console.log("Image " + image_name + " has been successfully deleted");
29                 $scope.loadImages();
30             },
31             function (response) {
32                 console.log("DeleteImage error with status code " + response.status);
33                 console.log("DeleteImage error message: " + response.data.detail);
34                 $scope.loading_images = false;
35             });
36     };
37 }
```

Figure 3-8 NFStore Images Management controller

### 3.9. Services Management controller

This controller interacts with the Service Catalogue towards allowing Service Providers to manage their services (e.g. composite a service from one or more network functions).

```

118  function ServiceListCtrl(Restangular, $scope, $rootScope, $state, ModalService) {
119    $scope.services = {};
120
121    $scope.loadNSDs = function () {
122      Restangular.all('nsds').getList().then(
123        function (response) {
124          $scope.services = response;
125          console.log("GetNSDList " + response.length + " NSDs found");
126        }, function (response) {
127          console.log("GetNSDList error with status code " + response.status);
128          console.log("GetNSDList error message: " + response.data.detail);
129        });
130    };
131
132    $scope.loadNSDs();
133
134
135
136    $scope.deleteNSD = function (id, nsd_name) {
137      Restangular.one('nsds', id).remove().then(
138        function () {
139          console.log("NSD " + nsd_name + " has been successfully deleted");
140          $scope.loadNSDs();
141          // $alert({title: '', content: "VNF " + vnf_name + " has been successfully deleted", placement: 'bot-right', type: 'success'});
142        }, function (response) {
143          console.log("DeleteNSD error with status code " + response.status);
144          console.log("DeleteNSD error message: " + response.data.detail);
145          // $alert({title: '', content: "Failed to delete vnf " + vnf_name + ".", placement: 'bot-right', type: 'danger'});
146          // $mdToast.show($mdToast.simple().content("Failed to delete vnf " + vnf_name).position('top right').hideDelay(3000));
147        });
148      );
149    };
150  };

```

Figure 3-9 Service Management controller

### 3.10. Broker Controller

This controller interacts with the Brokerage module towards allowing Service Providers and Customers to participate in the trading operation.

```

169  $scope.getPendingRequests = function(){
170
171    var pending_requests = 0;
172
173    angular.forEach($scope.trades, function (trade, trade_key) {
174
175      if (trade.status=='pending') {
176        pending_requests++;
177      }
178
179    });
180
181    return pending_requests;
182  };
183
184  $scope.acceptTradeRequest = function (id) {
185
186    Restangular.one('/broker/vnfs/trade/'+id+'/accept').get().then(
187      function (response) {
188        console.log("Trade Request " + response.id + " accepted");
189      }, function (response) {
190        console.log("Accept Trade Request error with status code " + response.status);
191        console.log("Accept Trade Request error message: " + response.data.detail);
192      });
193
194  };
195
196  $scope.rejectTradeRequest = function (id) {
197
198    Restangular.one('/broker/vnfs/trade/'+id+'/reject').get().then(
199      function (response) {
200        console.log("Trade Request " + response.id + " rejected");
201      }, function (response) {
202        console.log("Reject Trade Request error with status code " + response.status);
203        console.log("Reject Trade Request error message: " + response.data.detail);
204      });
205
206  };
207

```

Figure 3-10 Broker controller

## 4. USERS DASHBOARD INTERFACING WITH REST T-NOVA MARKETPLACE MODULES

This section presents the interface between the users' Dashboard and the rest t-NOVA marketplace modules, namely the Billing, UMAA (User Management, Authentication and Access Control), SLA management, Accounting, Brokerage, Business Service Catalogue, Function Store and Orchestrator.

### 4.1. APIs Definition

#### 4.1.1. Billing

The billing API for the dashboard manages the following information between the dashboard and the billing module:

- Bills charged per user and per service (SP and customer).
- Charges done to SP's customers (BSS functionality to the SP).
- Charges done to FP's customers, which is the SP.

#### Queries

Usage query API for getting user's data

URL	<a href="http://localhost:8080/udr/usage/users/{user_id}">http://localhost:8080/udr/usage/users/{user_id}</a>
Type	GET
Headers	x-auth-token : String
Parameters	from : Date to : Date
Response Code	200 : Success
Request	None

Response	<pre>{   "userid": "49588f5cea984040bc05d871eff67d2f",   "time": {     "to": "2015-01-12 01:10:00",     "from": "2015-01-12 01:01:00"   },   "usage": {     "openstack": [       {         "name": "cpu_util",         "columns": [           "time",           "sequence_number",           "avg"         ],         "points": [           [             1421024460734,             124666640001,             74.31932           ],           [             1421024460734,             124666550001,             0.7899716           ]<br ]<br="" pre="" }<="" }<br=""/></pre>
----------	---

	<pre>        }     }</pre>
--	----------------------------

#### Usage query API for particular resource / service id

URL	http://localhost:8080/udr/usage/resources/{resource_id}
Type	GET
Headers	x-auth-token : String
Parameters	from : Date to : Date
Response Code	200 : Success
Request	None
Response	<pre>{     "resourceid": "49588f5cea984040bc05d871eff67d2f",     "time": {         "to": "2015-01-12 01:10:00",         "from": "2015-01-12 01:01:00"     },     "column": [</pre>

	<pre>     "time",     "mean",     "userid" ], "usage": [ [     0,     0,     "46fe4a610a8b44948a5b61427b0b5ecd" ], [     0,     0,     "49588f5cea984040bc05d871eff67d2f" ], [     0,     2.950836399999999,     "99909daae8924e7a9b96cd964e9d64e3" ], ] } </pre>
--	---

#### Bill generation API for a particular customer

URL	<a href="http://localhost:8080/billing/invoice">http://localhost:8080/billing/invoice</a>
Type	GET
Headers	x-auth-token : String
Parameters	Customerid: String from : Date

	to : Date
Response Code	200 : Success
Request	None
Response	<pre>{   "time": {     "to": "2015-06-15 23:59",     "from": "2015-06-15 00:00"   },   "charge": {     "columns": [       "time",       "sequence_number",       "userid",       "usage",       "price",       "resource"     ],     "points": [       [         1434361731726,         413986240001,         "f83aa92bc3c64a3497b334cc712b0491",         5,         15.84,         "service-id-aaab-hg1562711-ahsbba"       ],       [         1434361731726,         413986230001,         "f83aa92bc3c64a3497b334cc712b0491",         5,         15.84,         "service-id-aaab-hg1562711-ahsbba"       ]     ]   } }</pre>

	<pre>     "f83aa92bc3c64a3497b334cc712b0491",     37,     124.4,     "service-id-aaac-hg1562711-ahsbbs"   ] } } </pre>
--	--

## 4.1.2. User Management, Authentication and Access control Module

UMMA's API collects all the information necessary to manage and authenticate the T-NOVA users or stakeholders. The root prefixes for this micro-service are **/user-management/** for the user/profile management and **/auth/** for the authentication. The micro-service uses these YAML files to initialize the permissions, groups and users:

- init\_users.yml
- init\_groups.yml
- init\_permissions.yml

### 4.1.2.1. Authentication

#### Methods

- POST /auth/

User authentication request

POST /auth/
-------------

**Request Body:**

{"username":"admin", "password":"123456"}

**Response Body:**

{"token":"eyJhbGciOiJIUzI1N....iIsInR5cCI6IkpXV "}

Request Example with JWT authentication

GET /user-management/users/
-----------------------------

Content-Type: application/json
--------------------------------

Authentication: JWT eyJhbGciOiJIUzI1N....iIsInR5cCI6IkpXV
---

### ***Token Metadata***

```
{
  'username': 'admin',
  'user_id': 1,
  'email': 'g.alexiou@pasiphae.eu',
  'company_name': 'TEIC',
  'exp': 1436620914
}
```

## 4.1.2.2. User Management

### **Methods**

- GET /user-management/users/
- POST /user-management/users/
- GET /user-management/users/{pk}/
- PUT /user-management/users/{pk}/
- PATCH /user-management/users/{pk}/
- DELETE /user-management/users/{pk}/
- GET /user-management/users/{pk}/groups/
- GET /user-management/users/{pk}/permissions/
- GET /user-management/profile/
- PUT /user-management/profile/
- PATCH /user-management/profile/
- GET /user-management/profile/groups/
- GET /user-management/profile/permissions/
- GET /user-management/groups/
- GET /user-management/countries/

### **Listing all Users**

This method returns the user's list.

GET /user-management/users/
-----------------------------

### ***Response Body:***

```
[
  {
    "id": 1,
    "username": "admin",
    "groups": [
      3
    ]
  }
]
```

```
[
    "company_name": "TEIC",
    "first_name": "",
    "last_name": "",
    "email": "g.alexiou@pasiphae.eu",
    "country": "GR",
    "city": "",
    "address": ""

},
{
    "id": 2,
    "username": "customer1",
    "groups": [
        1
    ],
    "company_name": "TEIC",
    "first_name": "",
    "last_name": "",
    "email": "customer1@t-nova.eu",
    "country": "GR",
    "city": "",
    "address": ""
}
]
```

## Create New User

This method creates a new user.

POST /user-management/users/
------------------------------

**Request Body:**

```
{
    "username": "new_user",
    "password": "123456",
    "groups": [
        3
    ],
    "company_name": "TEIC",
    "first_name": "",
    "last_name": "",
    "email": "g.alexiou@pasiphae.eu",
    "country": "GR",
    "city": "",
    "address": ""
}
```

```
}
```

**Response Body:**

```
{
  "id": 10,
  "username": "new_user",
  "groups": [
    3
  ],
  "company_name": "TEIC",
  "first_name": "",
  "last_name": "",
  "email": "g.alexiou@pasiphae.eu",
  "country": "GR",
  "city": "",
  "address": ""
}
```

## Get User's info

This method returns the user info.

```
GET /user-management/users/10/
```

**Response Body:**

```
{
  "id": 10,
  "username": "new_user",
  "groups": [
    3
  ],
  "company_name": "TEIC",
  "first_name": "",
  "last_name": "",
  "email": "g.alexiou@pasiphae.eu",
  "country": "GR",
  "city": "",
```

```
"address": ""  
}
```

## Edit User's info

This method changes user's info (PUT Method)

```
PUT /user-management/users/10/
```

### **Request Body:**

```
{  
    "id": 10,  
    "username": "new_user",  
    "groups": [  
        3  
    ],  
    "company_name": "TEIC",  
    "first_name": "George",  
    "last_name": "Alexiou",  
    "email": "g.alexio@pasiphae.eu",  
    "country": "GR",  
    "city": "Heraklion",  
    "address": "Estaurmenos"  
}
```

This method changes user's info partially (PATCH Method)

```
PATCH /user-management/users/10/
```

### **Request Body:**

```
{  
    "first_name": "George",  
    "last_name": "Alexiou",  
    "city": "Heraklion",  
    "address": "Estaurmenos"  
}
```

## Delete User

This method deletes a user

```
DELETE /user-management/users/10/
```

### Get User's groups

This method returns the user groups.

```
GET /user-management/users/10/groups/
```

#### **Response Body:**

```
[  
  {  
    "id": 10,  
    "name": "Customer",  
    "permissions": [  
      "umaa.edit_own_profile",  
      "umaa.view_own_profile"  
    ]  
  }  
]
```

### Get User's permissions

This method returns the user permissions.

```
GET /user-management/users/10/permissions/
```

#### **Response Body:**

```
[  
  "umaa.edit_own_profile",  
  "umaa.view_own_profile"  
]
```

### Get User's profile

This method returns the user's profile.

```
GET /user-management/profile/
```

#### **Response Body:**

```
{  
  "id": 10,  
  "username": "new_user",
```

```

"groups": [
  3
],
"company_name": "TEIC",
"first_name": "George",
"last_name": "Alexiou",
"email": "g.alexiou@pasiphae.eu",
"country": "GR",
"city": "Heraklion",
"address": "Estaurmenos"
}

```

### Edit User's profile

This method changes user's profile info (PUT Method)

PUT /user-management/profile/
-------------------------------

**Request Body:**

```

{
  "id": 10,
  "username": "new_user",
  "groups": [
    3
  ],
  "company_name": "TEIC",
  "first_name": "George",
  "last_name": "Alexiou",
  "email": "g.alexiou@pasiphae.eu",
  "country": "GR",
  "city": "Heraklion",
  "address": "Estaurmenos"
}

```

This method changes user's profile info partially (PATCH Method)

PATCH /user-management/profile/
---------------------------------

**Request Body:**

```
{  
  "first_name": "George",  
  "last_name": "Alexiou",  
  "city": "Heraklion",  
  "address": "Estaurmenos"  
}
```

**Get User's profile groups**

This method returns the user groups.

```
GET /user-management/profile/groups/
```

**Response Body:**

```
[  
  {  
    "id": 10,  
    "name": "Customer",  
    "permissions": [  
      "umaa.edit_own_profile",  
      "umaa.view_own_profile"  
    ]  
  }  
]
```

**Get User's profile permissions**

This method returns the user permissions.

```
GET /user-management/profile/permissions/
```

**Response Body:**

```
[  
  "umaa.edit_own_profile",  
  "umaa.view_own_profile"  
]
```

**Get Available Groups**

This method returns the available groups.

GET /user-management/groups/
------------------------------

**Response Body:**

```
[  
  {"id":3,"name":"Administrator"},  
  {"id":1,"name":"Customer"},  
  {"id":4,"name":"Function Provider"},  
  {"id":2,"name":"Service Provider"}  
]
```

### Get Available Countries

This method returns the available countries.

GET /user-management/countries/
---------------------------------

**Response Body:**

```
[  
  {"code":"AF","name":"Afghanistan"},  
  ...  
  {"code":"ZM","name":"Zambia"},  
  {"code":"ZW","name":"Zimbabwe"}  
]
```

### 4.1.3. SLA Management

This API's goal is to show the users the following information coming from the SLA management module:

- SLA template specification to be filled by the SP and FPs.
- SLA offering to the customer and associated to each service.
- SLA monitoring by all the stakeholders.
- SLA penalties for each network service and VNF.

▪ **Providers**

*POST* Creates a provider. The uuid is in the request

URL	/providers
Type	POST
Headers	Accept: application/json

	Content-type: application/json
Parameters	
Response code	409: The uuid or name already exists in the database. 201: Created.
Request example	POST /providers/ HTTP/1.1
POST item example	{           "uuid":"fc923960-03fe-41eb-8a21-a56709f9370f",           "name":"provider-prueba"         }

- **Templates:**

*POST* Creates a new template. The file might include a Template Id or not. In case of not being included, a uuid is assigned.

URL	/templates
Type	POST
Headers	Accept: application/json Content-type: application/json
Parameters	
Response code	<ul style="list-style-type: none"> <li>▪ 409: The uuid already exists in the database.</li> <li>▪ 409: The provider uuid specified in the template doesn't exist in the database.</li> <li>▪ 500: Incorrect data has been supplied.</li> <li>▪ 201: Created.</li> </ul>
Request example	POST /templates/ HTTP/1.1
POST item example	SLA template (see annex section 2).

*UPDATE* Updates the template identified by *TemplateId*. The body might include a *TemplateId* or not. In case of including a *TemplateId* in the file, it must match with the one from the url.

URL	/templates/{templateId}
Type	PUT
Headers	Accept: application/json

	Content-type: application/json
Parameters	<ul style="list-style-type: none"> <li>▪ <i>TemplateId</i>: Id of the template we want to modify.</li> </ul>
Response code	<ul style="list-style-type: none"> <li>▪ 409: The <i>templateId</i> from the url doesn't match with the one from the file.</li> <li>▪ 409: Template has agreements associated.</li> <li>▪ 409: Provider doesn't exist</li> <li>▪ 500: Incorrect data has been suplied</li> <li>▪ 200: OK</li> </ul>
Request example	PUT /templates/vnfvnf5gold HTTP/1.1
PUT item example	SLA template (see annex section 2).

*GET* Retrieves a template identified by *templateId*.

URL	/templates/{templateId}
Type	GET
Headers	Accept: application/json Content-type: application/json
Parameters	<ul style="list-style-type: none"> <li>▪ <i>templateId</i>: Id of the template we want to retrieve.</li> </ul>
Response code	<ul style="list-style-type: none"> <li>▪ 404: The <i>templateId</i> doesn't exist in the database.</li> <li>▪ 200: OK.</li> </ul>
Request example	GET /templates/vnfvnf5gold HTTP/1.1
Response example	SLA template in JSON form (see annex section 2)

#### ▪ **Agreements**

*GET* Retrieves an agreement identified by *agreementId*.

URL	/agreements/{agreementId}
Type	GET
Headers	Accept: application/json Content-type: application/json
Parameters	<ul style="list-style-type: none"> <li>▪ <i>agreementId</i>: Id of the agreement we want to retrieve.</li> </ul>
Response code	<ul style="list-style-type: none"> <li>▪ 404: The uuid doesn't exist in the database.</li> </ul>

	<ul style="list-style-type: none"> <li>▪ 200: OK.</li> </ul>
Request example	GET /agreements/vnfidf51 HTTP/1.1
Response example	SLA agreement in JSON form (see annex section 3)

The *AgreementId* matches the *AgreementId* attribute of *wsag:Agreement* element when the agreement is created.

#### ▪ SLA Information

Retrieve SLA related information to show in the dashboard given the userId and the wheter you want to retrieve VNFs or network services.

URL	/sla-info/?clientId={clientId}&kind={ns vnf}
Type	GET
Headers	Accept: application/json Content-type: application/json
Parameters	<i>clientId</i> : Id of the user that is using the network service or the VNF. <i>kind</i> : It can take 2 values: ns   vnf.
Response code	400 - Bad request: when the body in the request is not well formed. 500: There is a connection problem between the Accounting and the SLA modules. 200: OK.
Request example	GET /sla-info/?clientId=c1&kind=ns HTTP/1.1
Response example	[ { "productId": "service6", "productType": "ns", "clientId": "c1", "providerId": "p6", "SLAPenalties": 35, "agreementId": "serviceids101", "dateCreated": "2015-10-08T07:31:37Z", "dateTerminated": "2015-12-15T17:26:45.071444" }

	]
--	---

Retrieves the list of all running services the user (customer) is using.

URL	/servicelist/{userId}/
Type	GET
Headers	Accept: application/json Content-type: application/json
Parameters	<i>userId</i> : Id of the user for whom we want to retrieve the service list.
Response code	400 - Bad request: when the body in the request is not well formed. 200: OK.
Request example	GET /servicelist/c1 HTTP/1.1
Response example	[ { "id": 2, "instanceId": "id02", "productId": "s1", "agreementId": "s1vnf2_4", "relatives": "id01, id03", "productType": "ns", "flavour": null, "startDate": "2015-06-11T00:00:00Z", "lastBillDate": "2015-06-11T00:00:00Z", "providerId": "p1", "clientId": "c1", "status": "running", "billingModel": "PAYG", "period": "P1D", "priceUnit": "EUR", "periodCost": 1.0, "setupCost": 1.0, "renew": true, "dateCreated": "2015-06-11T13:29:16Z", }

	<pre>         "dateModified": "2015-12-10T09:29:41Z"       },     ]   </pre>
--	--

Retrieves the list of all running VNFs the user (service provider) is using.

URL	/vnflist/{userId}/
Type	GET
Headers	Accept: application/json Content-type: application/json
Parameters	<i>userId</i> : Id of the user for whom we want to retrieve the service list.
Response code	400 - Bad request: when the body in the request is not well formed. 200: OK.
Request example	GET /vnflist/p5 HTTP/1.1
Response example	[ { "id": 24, "instanceId": "idf50", "productId": "vnf5", "agreementId": "s1vnf2_4", "relatives": "ids100", "productType": "vnf", "flavour": null, "startDate": "2015-10-08T07:07:43Z", "lastBillDate": "2015-10-08T07:07:43Z", "providerId": "f5", "clientId": "p5", "status": "running", "billingModel": "PAYG", "period": "P1D", "priceUnit": "EUR", "periodCost": 1.0, "setupCost": 2.0, }]

	<pre>       "renew": false,       "dateCreated": "2015-10-08T07:07:43Z",       "dateModified": "2015-10-08T07:07:43Z"     } ] </pre>
--	--

#### 4.1.4. Billing

- **Bill Report**

Returns all the CDRs for a user within a specified date.

**GET** /billing/invoice?userId={userId}&from={from\_date}&to={to\_date}

**Response Body:**

```

{
  "name": "charge",
  "from": "2015-12-03 00:00:00",
  "to": "2015-12-09 23:59:59",
  "charges": [
    {
      "priceUnit": "EUR",
      "setupCost": 0,
      "resource": "id02",
      "providerId": "p1",
      "price": 0.003472222222222222,
      "usage": "300",
      "time": "2015-12-09T08:10:52Z",
      "userId": "c1",
      "periodCost": 1
    },
    {
      "priceUnit": "EUR",
      "setupCost": 0,
      "resource": "id02",

```

```

"providerId": "p1",
"price": 0.003472222222222222,
"usage": "300",
"time": "2015-12-09T08:10:52Z",
"userId": "c1",
"periodCost": 1
}
]
}

```

#### ▪ Revenue Sharing Report

Returns the revenues that a specified Service Provider owes to a VNF Provider

##### **GET**

/billing/revenue/report?vfpId={fpId}&spId={spId}&from={from\_date}&to={to\_date}

##### **Response Body:**

```

{
  "VNFProvider": "f1",
  "from": "2015-12-07 00:00:00",
  "to": "2015-12-09 23:59:59",
  "revenues": [
    {
      "priceUnit": "EUR",
      "instanceId": "id01",
      "price": 0.0034722222222222,
      "VNFProvider": "f1",
      "name": "tnova_revenue_sharing",
      "SPublisher": "p1",
      "time": "2015-12-09T08:10:52Z"
    },
    {
      "priceUnit": "EUR",
      "instanceId": "id03",
      "price": 0.0034722222222222,
    }
  ]
}

```

```

    "VNFProvider": "f1",
    "name": "tnova_revenue_sharing",
    "SProvider": "p1",
    "time": "2015-12-09T08:10:52Z"
}
]
}

```

Returns the revenues that a NON specified Service Provider owes to a VNF Provider

**GET** /billing/revenue/report?vfpId={fpId}&from={from\_date}&to={to\_date}

**Response Body:**

```

{
    "VNFProvider": "f1",
    "from": "2015-12-07 00:00:00",
    "to": "2015-12-09 23:59:59",
    "revenues": [
        {
            "priceUnit": "EUR",
            "instanceId": "id01",
            "price": 0.003472222222222222,
            "VNFProvider": "f1",
            "name": "tnova_revenue_sharing",
            "SProvider": "p2",
            "time": "2015-12-09T08:10:52Z"
        },
        {
            "priceUnit": "EUR",
            "instanceId": "id03",
            "price": 0.003472222222222222,
            "VNFProvider": "f1",
            "name": "tnova_revenue_sharing",
            "SProvider": "p1",

```

```
        "time": "2015-12-09T08:10:52Z"  
    }  
]  
}
```

#### ▪ **Usage Data Records**

Return usage data records

**GET** udr/usage/users/{customerId}?from={from\_date}&to={to\_date}

#### **Response Body:**

```
{  
    "name": "UDR",  
    "from": "2015-12-10 00:00",  
    "to": "2015-12-10 14:56",  
    "instanceId": "id02",  
    "usages": [  
        {  
            "time": "2015-12-10T07:36:27Z",  
            "usage": 300  
        },  
        {  
            "time": "2015-12-10T07:36:27Z",  
            "usage": 300  
        },  
        {  
            "time": "2015-12-10T07:36:27Z",  
            "usage": 300  
        }  
    ]  
}
```

#### 4.1.5. Brokerage

This interface is exploited for trading issues, among the T-NOVA users (i.e. Service Providers and Function Providers) and the brokerage module. The information that will go through this API will be related to:

- Service composition/VNF request: this functionality enables the SP to request network functions or compose a new service.
- Advertise VNF: this functionality is exploited for the communication between FP and the brokerage module, as the latter perform the intermediate communication, this is trading.

##### Get Available VNFs

Returns the available VNFs

**GET** /broker/vnfs/?<filters>

**Response Body:**

```
[  
{  
provider: "TEIC",  
release: "T-NOVA",  
type: "FW",  
id: 562,  
description: "PFSense is a firewall..."  
...  
},{  
provider: "TEIC",  
release: "T-NOVA",  
type: "FW",  
id: 563,  
description: "UNTagle is a firewall..."  
...  
}  
]
```

### New Trade Request

This method creates a new trade request

<b>POST</b> /broker/vnfs/trade/
---------------------------------

<b>Request Body:</b>
----------------------

{ provider_id: 6, vnf_id: 563, price_override: 12.0 }
---

### Get Trade Request

This method returns trade request

<b>GET</b> /broker/vnfs/trade/4
---------------------------------

<b>Response Body:</b>
-----------------------

{ created_at: %Y-%m-%dT%H:%M:%SZ, modified_at: %Y-%m-%dT%H:%M:%SZ, provider_id: 6, vnf_id: 563, price_override: 12.0, status:"pending", id:4 }
--

### Get Trades Requests

This method returns a list of trade requests

<b>GET</b> /broker/vnfs/trade/
--------------------------------

<b>Response Body:</b>
-----------------------

[ {
--------

```
created_at: %Y-%m-%dT%H:%M:%SZ,  
modified_at: %Y-%m-%dT%H:%M:%SZ,  
provider_id: 6,  
vnf_id: 563,  
price_override: 12.0,  
status:"pending",  
id:4  
}  
{  
created_at: %Y-%m-%dT%H:%M:%SZ,  
modified_at: %Y-%m-%dT%H:%M:%SZ,  
provider_id: 6,  
vnf_id: 564,  
price_override: 6.0,  
status:"accepted",  
id:5  
}
```

### Accept Trade Requests

This method accepts a trade request offer

```
GET /broker/vnfs/trade/5/accept/
```

### Reject Trade Requests

This method rejects a trade request offer

```
GET /broker/vnfs/trade/5/reject/
```

## 4.1.6. Function Store

This interface allows the FPs to publish and manage their VNFs into the NF Store. The publication consists in uploading the VNF image, registering the VNF and its metadata

into the function store. The VNFs are versioned allowing the FPs to provide further upgrades. Finally, the FPs can remove their VNFs. In summary, the information managed with this interface is:

- VNF image and VNF metadata descriptor.
- VNF version.
- Upload, upgrade and delete the VNF package.

## Methods

- GET /vnfs/
- POST /vnfs/
- GET /internal/vnfs/
- GET /vnfs/{pk}/
- PUT /vnfs/{pk}/
- DELETE /vnfs/{pk}/
- GET /vnfs/{pk}/yaml/
- PUT /vnfs/{pk}/yaml/

### **Listing all VNFs**

This method return VNF list.

<pre>GET /vnfs/ GET /internal/vnfs/ (for module to module communication only)</pre>
---

**Response Body:**

```
[  
{  
provider: "TEIC",  
release: "T-NOVA",  
type: "FW",  
id: 562,  
description: "PFSense is a firewall..."  
...  
}, {  
provider: "TEIC",  
release: "T-NOVA",  
type: "FW",  
id: 563,  
description: "UNTagle is a firewall..."  
...  
}  
]
```

### **Create New VNF**

This method creates a new VNF.

POST /vnfs/
-------------

**Request Body:**

```
{  
  provider: "TEIC",  
  release: "T-NOVA",  
  type: "FW",  
  id: 562,  
  description: "PFSense is a firewall..."  
  ...  
}
```

## Get VNFD

This method returns a vnfd

GET /vnfs/562/
----------------

**Response Body:**

```
{  
  provider: "TEIC",  
  release: "T-NOVA",  
  type: "FW",  
  id: 562,  
  description: "PFSense is a firewall..."  
  ...  
}
```

## Update VNFD

This method updates a vnfd

PUT /vnfs/562/
----------------

**Request Body:**

```
{  
  provider: "TEIC",  
  release: "T-NOVA",  
  type: "FW",  
  id: 562,  
  description: "PFSense is a firewall..."  
  ...  
}
```

## Delete VNFD

This method deletes a vnfd

DELETE /vnfs/562/
-------------------

### Get YAML VNFD

This method returns a vnfd file in YAML format

GET /vnfs/562/yaml/
---------------------

**Response Body:**

id: 562  
modified\_at: '2015-10-18T17:50:09Z'  
name: pfSense Firewall  
provider: TEIC  
provider\_id: 4  
release: T-NOVA  
....

### Update YAML VNFD

This method updates a vnfd file in YAML format

PUT /vnfs/562/yaml/
---------------------

**Request Body:**

id: 562  
modified\_at: '2015-10-18T17:50:09Z'  
name: pfSense Firewall  
provider: TEIC  
provider\_id: 4  
release: T-NOVA  
....

## 4.1.7. Business Service Catalogue

Get available services

<b>GET</b> /service/catalog
-----------------------------

**Response Body:**

[ {
--------

```
"nsd": {  
    "id": "56931f5de4b0c74fd56e890d",  
    "name": "network-service-0",  
    "vendor": "T-NOVA",  
    "version": "1.0",  
    "manifest_file_md5": "fa8773350c4c236268f0bd7807c8a3b2",  
    "vnfds": [  
        "52439e7c-c85c-4bae-88c4-8ee8da4c5485"  
    ],  
    ...  
},  
"auto_scale_policy": {  
    "criteria": [  
        {  
            "end-to-end bandwidth": "10Mbps"  
        },  
        {  
            "test": "test"  
        }  
    ],  
    "action": "upgrade"  
},  
"connection_points": [  
    {  
        "connection_point_id": "mgnt0",  
        "type": "ip"  
    },  
    {  
        "connection_point_id": "data0",  
        "type": "bridge"  
    },  
    {  
        "connection_point_id": "stor0",  
    }  
]
```

```
        "type": "bridge"  
    }  
]  
}  
}  
]
```

## Create new Service

**POST /service/catalog**

**Request Body:**

```
{  
    "nsd": {  
        "auto_scale_policy": {  
            "action": "string",  
            "criteria": [  
                {  
                    "end-to-end bandwidth": "string",  
                    "test": "string"  
                }  
            ]  
        },  
        "connection_points": [  
            {  
                "connection_point_id": "string",  
                "type": "string"  
            }  
        ],  
        "id": "string",  
        ...  
        "constituent_vnfs": [  
            "string"  
        ],  
        "graph": [  
    }
```

```

    "string"
],
"nfp_id": "string"
}
],
"number_of_endpoints": 0,
"number_of_virtual_links": 0,
"vnffg_id": "string"
}
]
}
}
}

```

### Get service information

**GET /service/catalog/{serviceId}**

**Response Body:**

```

{
  "nsd": {
    "auto_scale_policy": {
      "action": "string",
      "criteria": [
        {
          "end-to-end bandwidth": "string",
          "test": "string"
        }
      ]
    },
    ...
    "graph": [
      "string"
    ],
    "nfp_id": "string"
  }
}

```

```

        }
    ],
    "number_of_endpoints": 0,
    "number_of_virtual_links": 0,
    "vnffg_id": "string"
}
]
}
}
}
```

#### Delete service

**DELETE** /service/catalog/{serviceId}

### 4.1.8. Service Selection

#### Service instantiation

**POST** /service/selection

##### **Request Body:**

```
{
  "created_at": "string",
  "id": 0,
  "nsd_id": "string",
  "status": "string",
  "updated_at": "string",
  "vnfs": [
    {
      "id": 0,
      "vnfd_id": "string",
      "vnfi_id": "string"
    }
  ]
}
```

**Response Body:**

```
{
  "callbackUrl": "string",
  "customer_id": "string",
  "nap_id": "string",
  "ns_id": "string"
}
```

Service termination

<b>DELETE</b> /service/selection/{service_instance_id}

#### 4.1.9. Orchestrator

##### Get Monitoring Information

Dashboard use this interface to the orchestrator in order to retrieve monitoring information.

GET orchestrator/instances/{instanceId}/monitoring-data/?instance_type={instance_type}&metric={metric}

**Response Body:**

```
[
  {
    "metricname": "packets_per_second",
    "value": 100,
    "date": "2015-01-21T18:49:00CET",
  }
]
```

**Parameters:**

- instanceType: it can be "ns" or "vnf"
- instanceId: instance ID of the service or function
- metric: name of the metric we want to obtain the monitoring data



## 5. GRAPHICAL USER INTERFACE AND FUNCTIONALITIES

### 5.1. User Registration

The Customers or the Function Providers can register to the dashboard from this page.

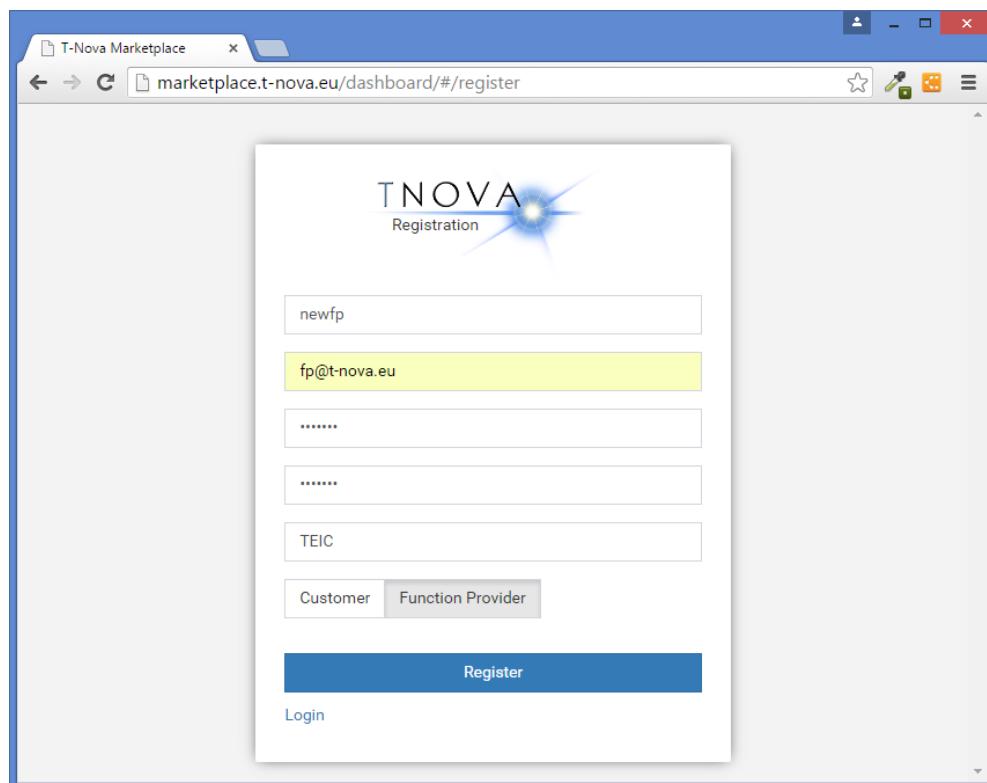


Figure 5-1 User registration

### 5.2. User Login

This is the Dashboard's login page.

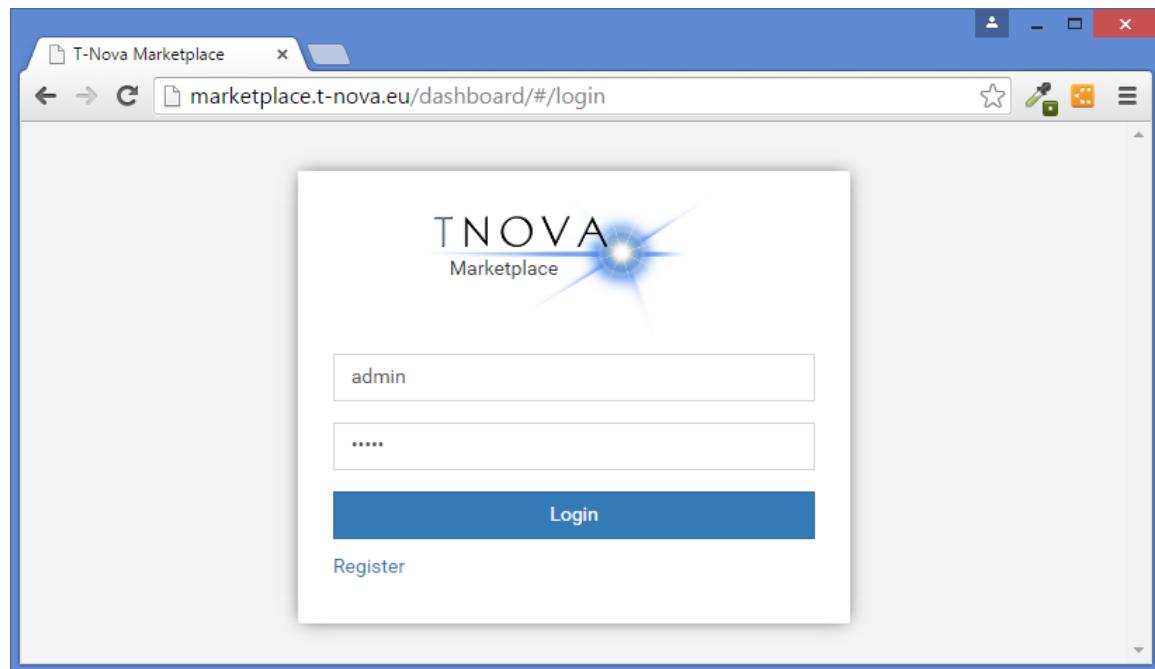


Figure 5-2 Dashboard's login page

### 5.3. VNF Provider

This is the VNF Providers' Home Page.

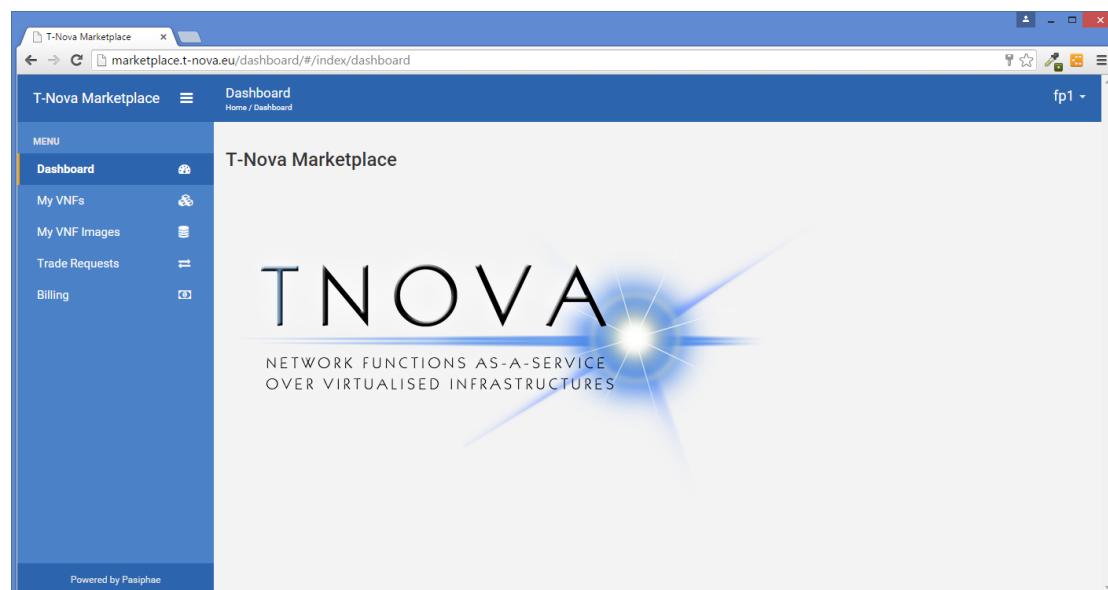


Figure 5-3 VNF Providers home page

#### 5.3.1. VNF Creation

Here the function provider can create new VNFs.

### 5.3.1.1. Step 1 – VNF Basic Information

The First step of the VNF creation is to give basic information about the Network Function

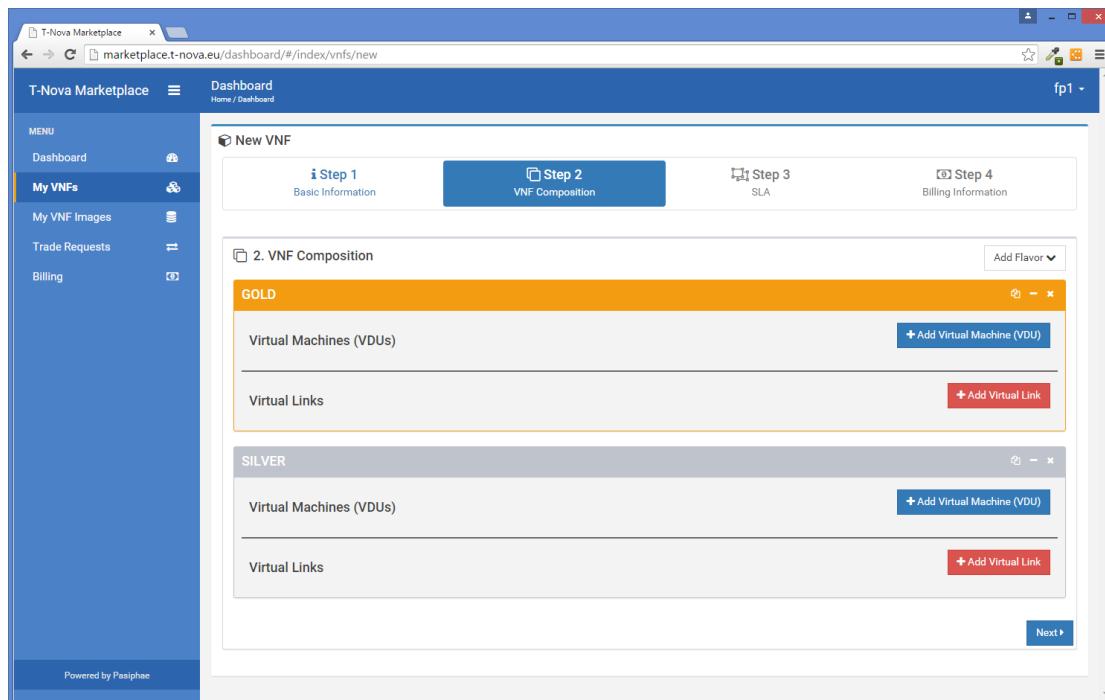
- The name of the VNF
- A Description
- VNF Version
- Descriptor Version
- VNF Type (Traffic Classifier, Firewall, etc.)

The screenshot shows a web-based application for creating a new Virtual Network Function (VNF). The interface is titled 'New VNF' and is divided into four steps: Step 1 (Basic Information), Step 2 (VNF Composition), Step 3 (SLA), and Step 4 (Billing Information). Step 1 is currently active and titled '1. Basic Information'. It contains fields for 'VNF Name' (set to 'Firewall'), 'VNF Description' (which includes a detailed description of pfSense as a firewall software), 'VNF Version' (set to '1'), 'VNF Descriptor Version' (set to '1'), and 'VNF Type' (set to 'Firewall'). A blue 'Next >' button is located at the bottom right of the form.

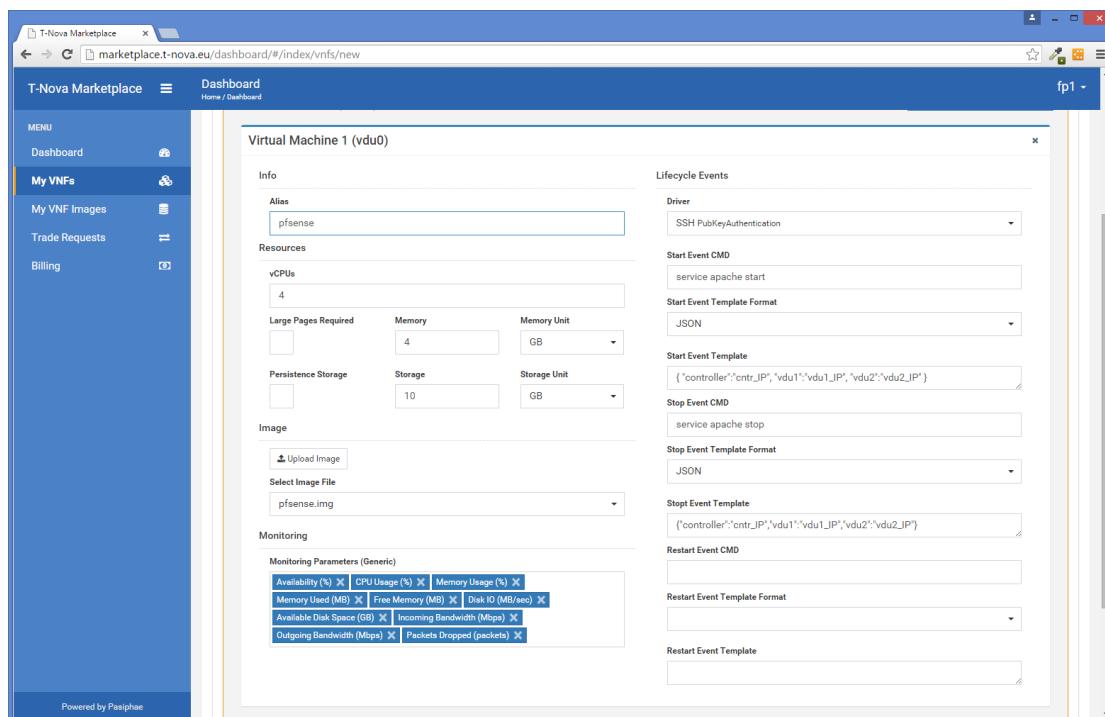
**Figure 5-4 VNF Creation - Step 1: VNF Basic Information**

### 5.3.1.2. Step 2 - VNF Composition

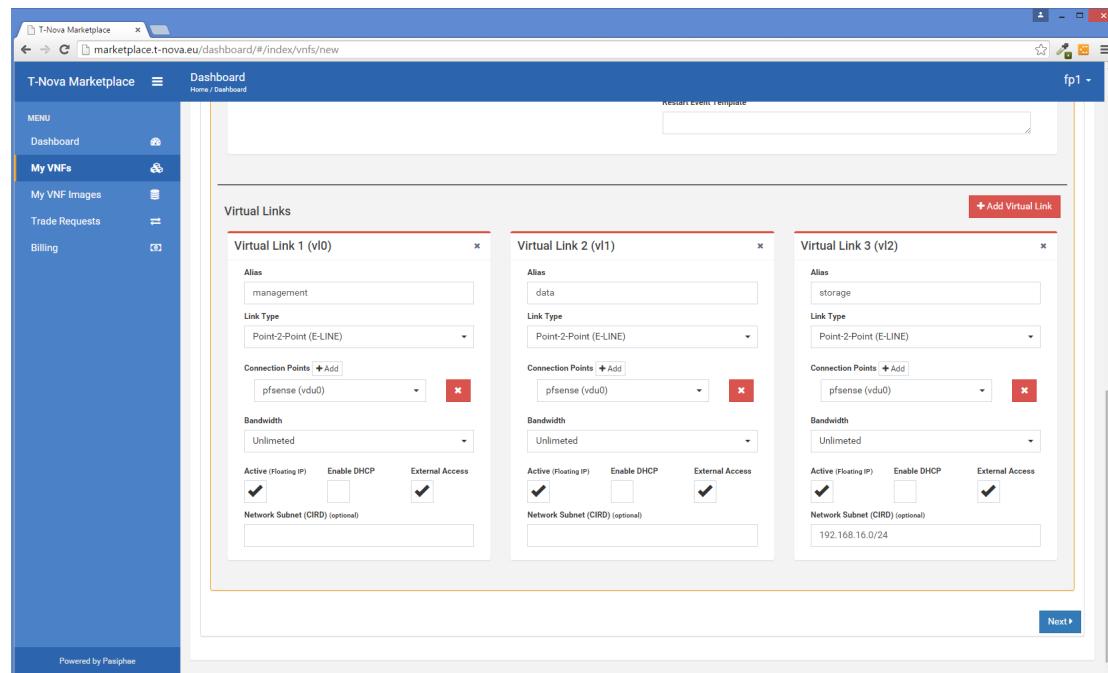
The second step of the VNF Creation is the VNF composition, here the provider can select the flavor that prefers (gold/silver/bronze) and create a composition of every flavor. Then the function provider can add a Virtual Machine (i.e. VDU) to the flavor, in this step the provider can define the required resources for the virtual machine as long as the image, the monitoring parameters and the lifecycle events. The last step is to define the virtual links that will be available in the composition in order to link individual virtual machines or give them external access.



**Figure 5-5 VNF Creation - Step 2: VNF Composition, SLA Flavors**



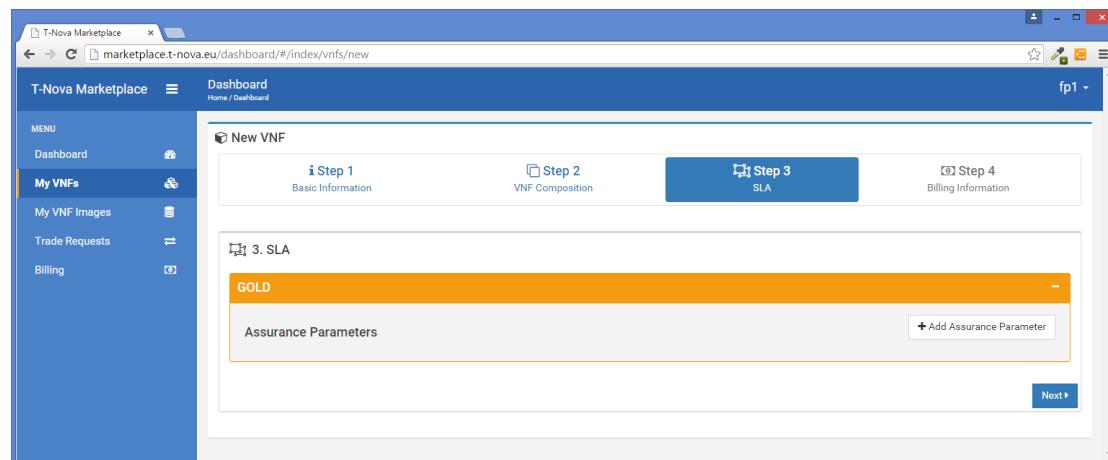
**Figure 5-6 VNF Creation - Step 2: VNF Composition, Virtual Machines**



**Figure 5-7 VNF Creation - Step 2: VNF Composition, Virtual Links**

### 5.3.1.3. Step 3 – SLA

Here the provider defines the assurance parameters for every selected flavor. For example, the provider can apply a discount, if there is an SLA violation.



**Figure 5-8 VNF Creation - Step 3: SLA**

The screenshot shows the 'My VNFs' section of the T-Nova Marketplace. A modal window titled 'GOLD' is open, displaying two assurance parameters. Assurance Parameter 1 (param0) has a monitoring parameter of 'Packets Dropped (packets)', an expression of 'GT (Greater Than)', a threshold value of 100, and a compiled formula of 'IF Packets Dropped IS/GRE Greater Than 100 packets'. Assurance Parameter 2 (param1) has a monitoring parameter of 'Availability (%)', an expression of 'LT (Less Than)', a threshold value of 99, and a compiled formula of 'IF Availability IS/ARE Less Than 99 % THEN ...'. Both parameters have sections for violations, penalty type (Discount), and validity period (2 days).

**Figure 5-9 VNF Creation - Step 3: SLA, Assurance Parameters**

#### 5.3.1.4. Step 4 – Billing

In this final step, the provider can define the billing model, price, currency and to enable the trading (for the broker module).

The screenshot shows the 'New VNF' creation process. Step 4, 'Billing Information', is selected. It includes fields for Billing Model (set to 'Pay-As-You-Go'), Currency (set to 'Euro'), and price ranges: Min Price/Period (10), Max Price/Period (20), and Setup Price (20). A 'Trade Enable' checkbox is checked. A 'Create +' button is located at the bottom right.

**Figure 5-10 VNF Creation - Step 4: Billing**

#### 5.3.2. VNF Listing

Here the providers are able to manage their VNFs, view the generated VNFD file, edit the NFD with the YAML editor, view the composed VNF (vnf diagram) and delete a VNF.

The screenshot shows the T-Nova Marketplace dashboard with a sidebar menu on the left containing 'Dashboard', 'My VNFs' (selected), 'My VNF Images', 'Trade Requests', and 'Billing'. The main area is titled 'VNFS' and lists six entries:

#	Name	Description	VNFD	YAML Editor	Logs	Action
#562	pfSense Firewall	PFsense is a firewall solution built upon the FreeBSD OS. It is very stable, very secure, and very polished.	<a href="#">VNFD</a>	<a href="#">YAML Editor</a>	<a href="#">Logs</a>	<a href="#">Delete</a>
#561	pfSense Firewall	pfSense is a firewall solution built upon the FreeBSD OS. It is very stable, very secure, and very polished.	<a href="#">VNFD</a>	<a href="#">YAML Editor</a>	<a href="#">Logs</a>	<a href="#">Delete</a>
#563	Untangle Firewall	Untangle is a firewall made by a company named Untangle. This Linux distro offers anti-virus, anti-spam, web caching, and many other protective features. A variety of software is available for Untangle including proprietary software.	<a href="#">VNFD</a>	<a href="#">YAML Editor</a>	<a href="#">Logs</a>	<a href="#">Delete</a>
#564	UNTAGLE	Untangle, a network software and appliance company, provides the most complete multi-function firewall and Internet management application suite available.	<a href="#">VNFD</a>	<a href="#">YAML Editor</a>	<a href="#">Logs</a>	<a href="#">Delete</a>
#565	pfSense Firewall	pfSense is an open source firewall/router computer software distribution based on FreeBSD	<a href="#">VNFD</a>	<a href="#">YAML Editor</a>	<a href="#">Logs</a>	<a href="#">Delete</a>
#568	dpi with dpdk	Intel based enabled DPI Function	<a href="#">VNFD</a>	<a href="#">YAML Editor</a>	<a href="#">Logs</a>	<a href="#">Delete</a>

6 VNFs

Powered by Pasiphae

**Figure 5-11 VNF Listing**

### 5.3.3. VNF Tools

#### 5.3.3.1. Generated VNFD Viewer

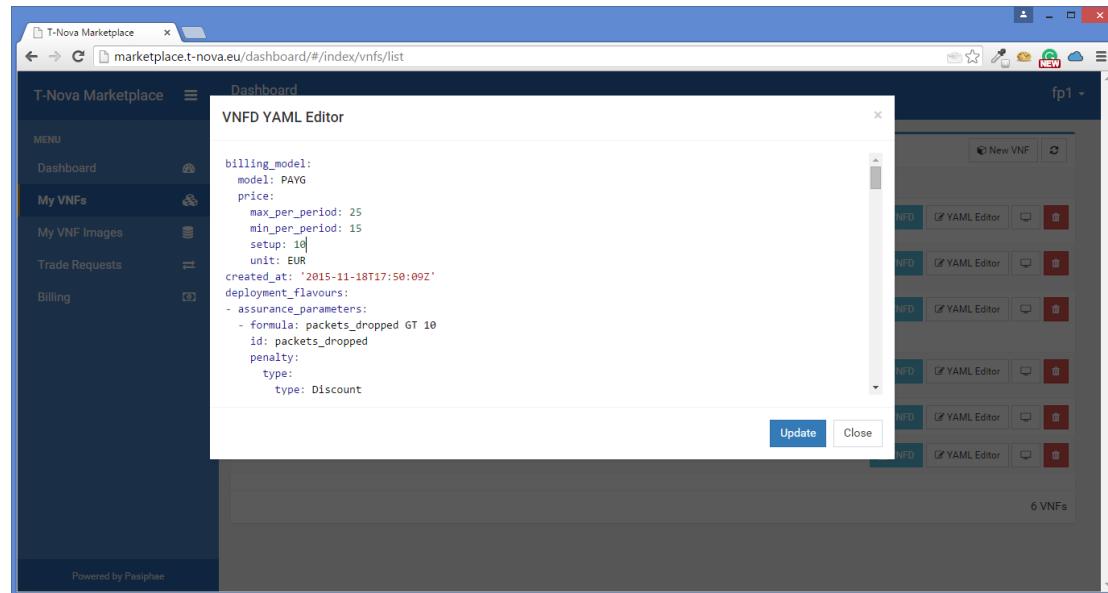
ETSI compliant Generated VNFD.

```
{
  provider_id: 4,
  - vdu: [
    - {
      - resource_requirements: {
        network_interface_bandwidth_unit: "",
        - hypervisor_parameters: {
          version: "10002|12001|2.6.32-358.el6.x86_64",
          type: "QEMU-KVM"
        },
        memory_unit: "GB",
        - network_interface_card_capabilities: {
          SR-IOV: true,
          mirroring: false
        },
        - storage: {
          size_unit: "GB",
          persistence: false,
          size: 0
        },
        network_interface_bandwidth: ""
      },
      - platform_pcie_parameters: {
        SR-IOV: true,
        device_pass_through: true
      },
      vcpus: 4,
      - vswitch_capabilities: {
        version: "2.0",
        type: "ovs",
        overlay_tunnel: "GRE"
      },
      data_processing_acceleration_library: "",
      memory: 2,
      - memory_parameters: {
        large_pages_required: false,
        numa_allocation_policy: ""
      }
    }
  ]
}
```

**Figure 5-12 Generated VNFD Viewer**

### 5.3.3.2. VNFD YAML Editor

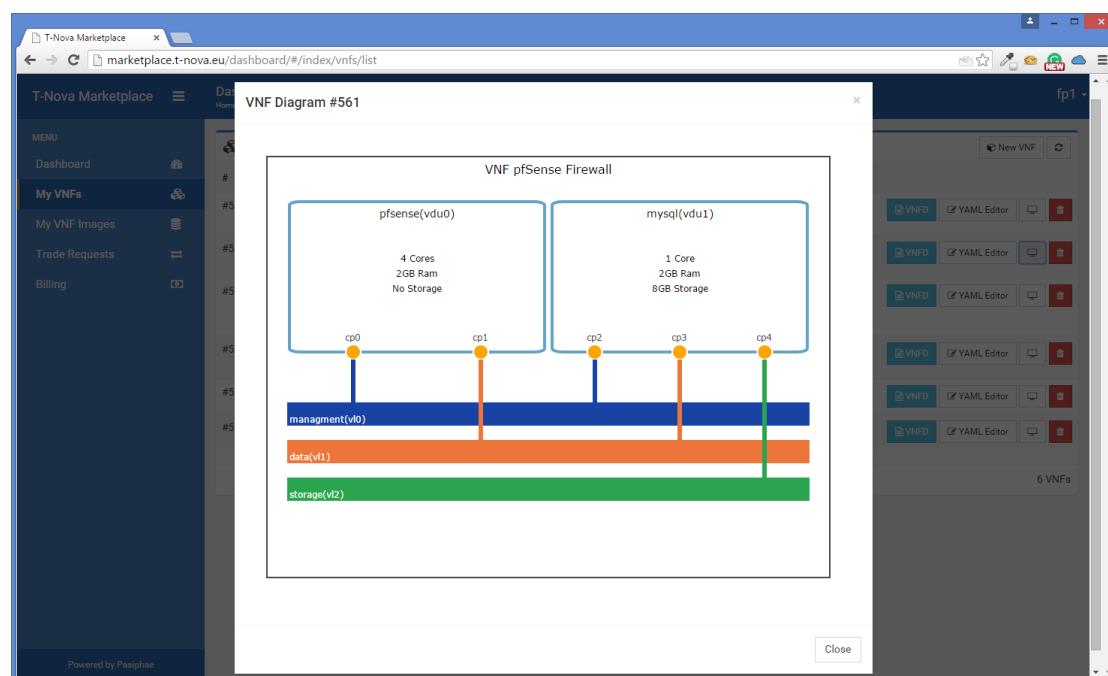
The provider can edit the generated VNFD with the YAML editor.



**Figure 5-13 VNFD YAML Editor**

### 5.3.3.3. VNFD Diagram

The provider can (visualize the VNFD compositions) view the diagram of the generated VNFD.



**Figure 5-14 VNFD Diagram**

### 5.3.4. Images

From this page the function provider can manage the VNF images.

Image Name	URL	Assigned To
apache.img	http://83.212.108.105:8080/NFS/files/4-apache.img	[564]
mysql.img	http://83.212.108.105:8080/NFS/files/4-mysql.img	[561,564]
pfsense.img	http://83.212.108.105:8080/NFS/files/4-pfsense.img	[561,562,565,565,576,577,613,614,615,616,617,618]
untangle.img	http://83.212.108.105:8080/NFS/files/4-untangle.img	[563,564]
Dpdk_vdu.img	http://83.212.108.105:8080/NFS/files/4-Dpdk_vdu.img	[568]
traffic_dpi.img	http://83.212.108.105:8080/NFS/files/4-traffic_dpi.img	[568]

6 Images

**Figure 5-15 Function Provider VNF Images**

## 5.4. Service Provider

This is the Service Provider's home page.

T-Nova Marketplace

T NOVA

NETWORK FUNCTIONS AS-A-SERVICE  
OVER VIRTUALISED INFRASTRUCTURES

Powered by Pasiphae

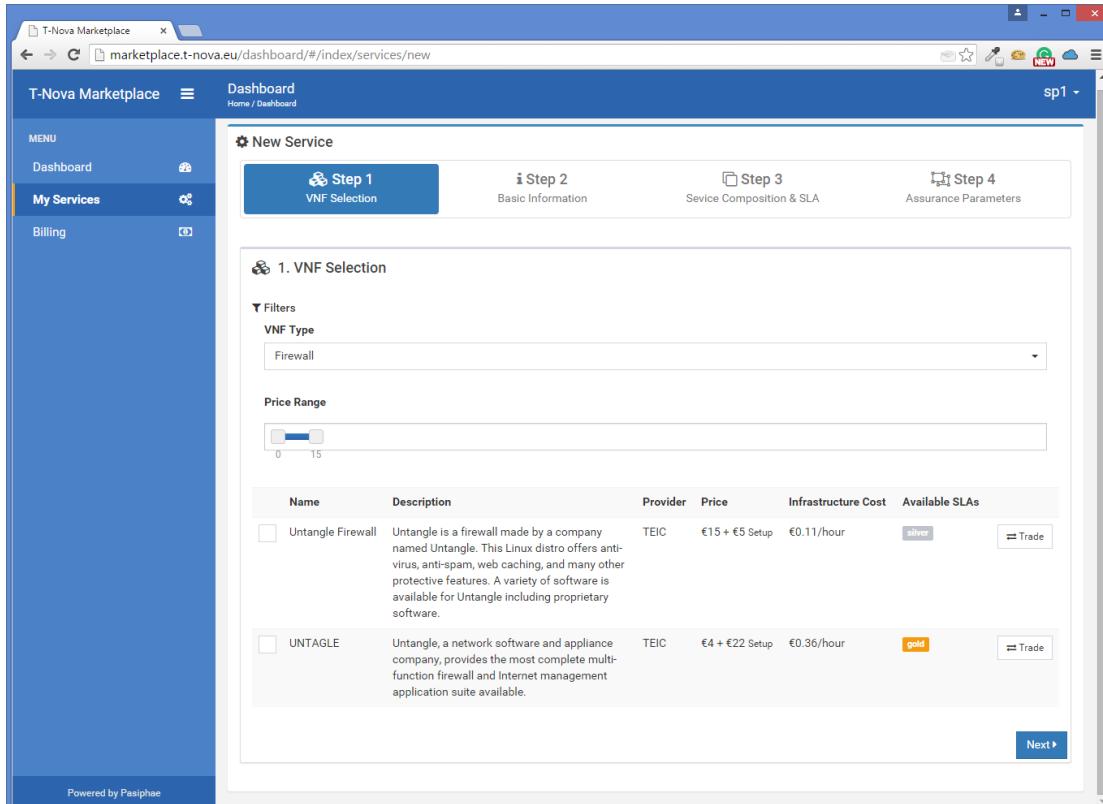
**Figure 5-16 Service Provider**

## 5.4.1. Service/NSD Creation

In the below pages, the service provider can compose/create a new service

### 5.4.1.1. Step 1 - VNF Selection

The first step of the NSD/service composition is the VNF Selection. The service provider selects the VNFs that would like to apply/add on this service.



**Figure 5-17 NSD Creation - Step 1: VNF Selection**

### 5.4.1.2. Step 1 - VNF Trading

The service provider is able to negotiate with the function provider requesting a better price (Figure 5-18 Figure 5-19). The function provider can accept this request/offer or to reject it Figure 5-20). In **Error! Reference source not found.**, the Function provider accepts the request, so the new price will override the old price.

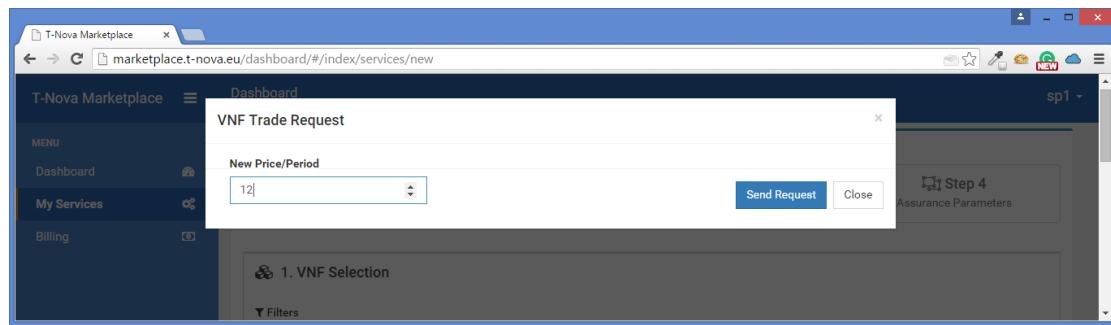


Figure 5-18 NSD Creation - Step 1: VNF Selection, Trade Request

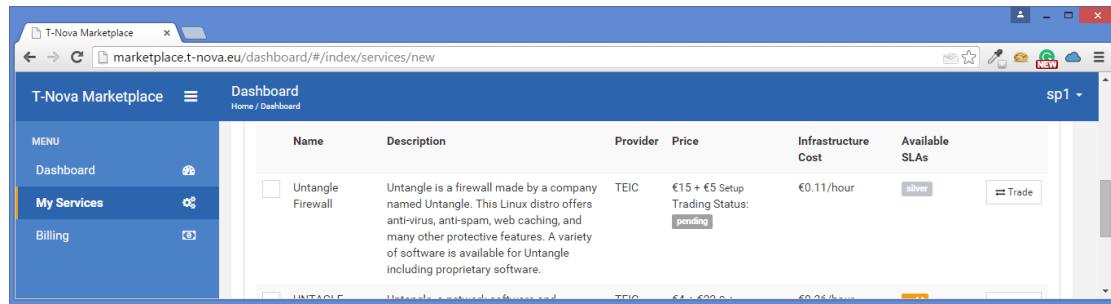


Figure 5-19 NSD Creation - Step 1: VNF Selection, Pending Trade Request

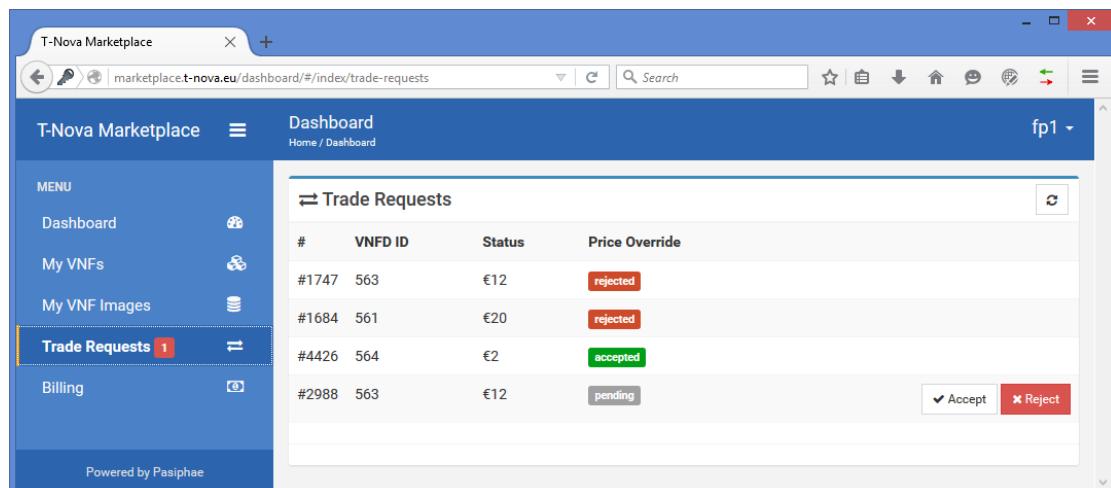


Figure 5-20 Trade Request FP View

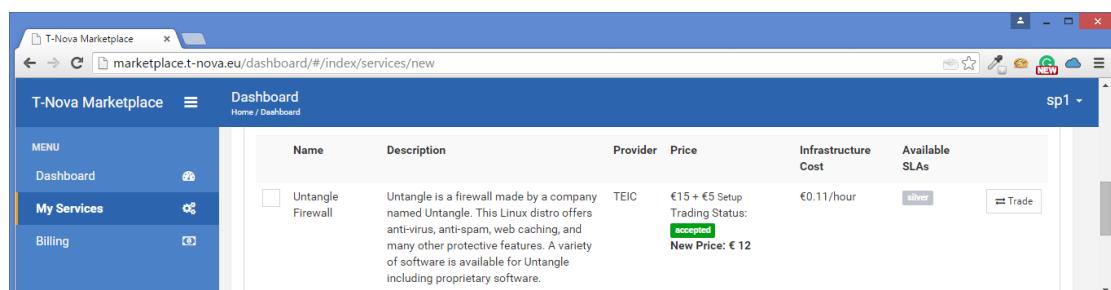


Figure 5-21 Accepted Trade Offer

### 5.4.1.3. Step 2 –Basic Information

The next step is the basic information of the Network service

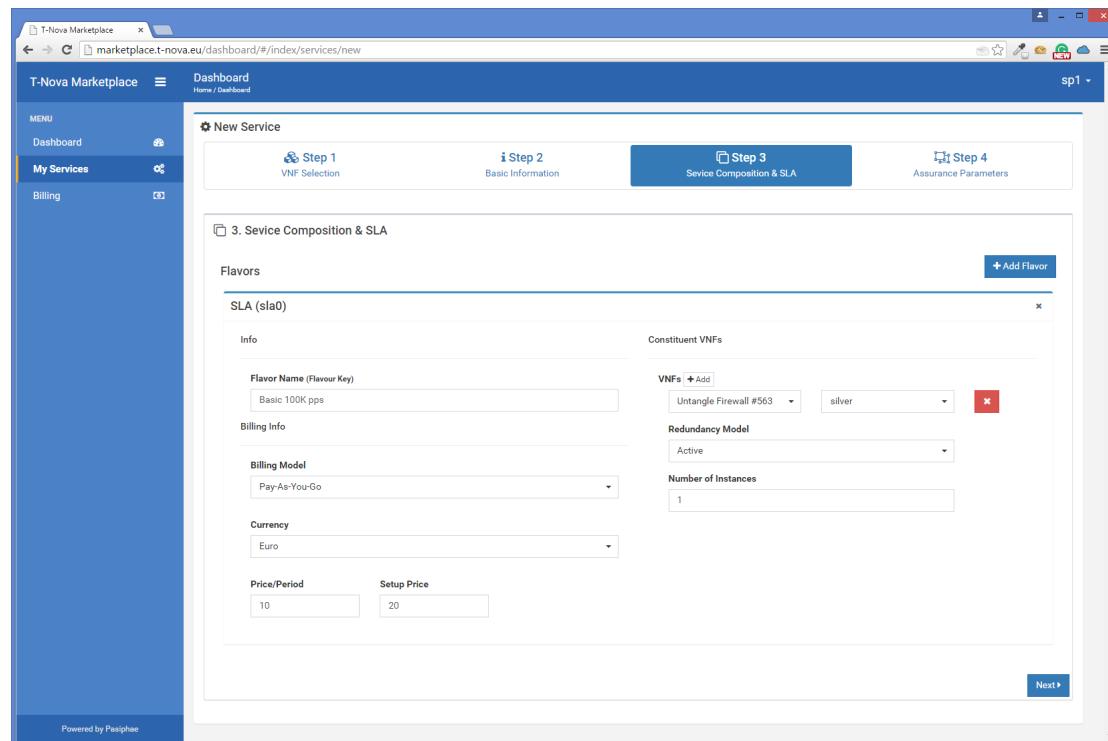
- Service name
- A description
- NSD version
- Descriptor Version

The screenshot shows a web browser window for the T-Nova Marketplace. The URL is marketplace.t-nova.eu/dashboard/#/index/services/new. The page title is 'T-Nova Marketplace' and the sub-page title is 'Dashboard'. On the left, there is a sidebar with 'MENU' and three options: 'Dashboard', 'My Services' (which is selected), and 'Billing'. The main content area has a header 'New Service' with four tabs: 'Step 1 VNF Selection' (disabled), 'Step 2 Basic Information' (selected and highlighted in blue), 'Step 3 Service Composition & SLA' (disabled), and 'Step 4 Assurance Parameters' (disabled). Below the tabs, the 'Step 2' section is titled '2. Basic Info'. It contains fields for 'NSD Name' (value: 'Managed Firewall'), 'NSD Description' (value: 'Managed Firewall services ensure network security and availability by preventing unauthorized visitors from accessing valuable business resources.'), 'NSD Version' (value: '1'), and 'NSD Descriptor Version' (value: '1'). At the bottom right of this section is a 'Next' button.

**Figure 5-22 NSD Creation - Step 2: Basic Information**

### 5.4.1.4. Step 3 – Service Composition and SLA

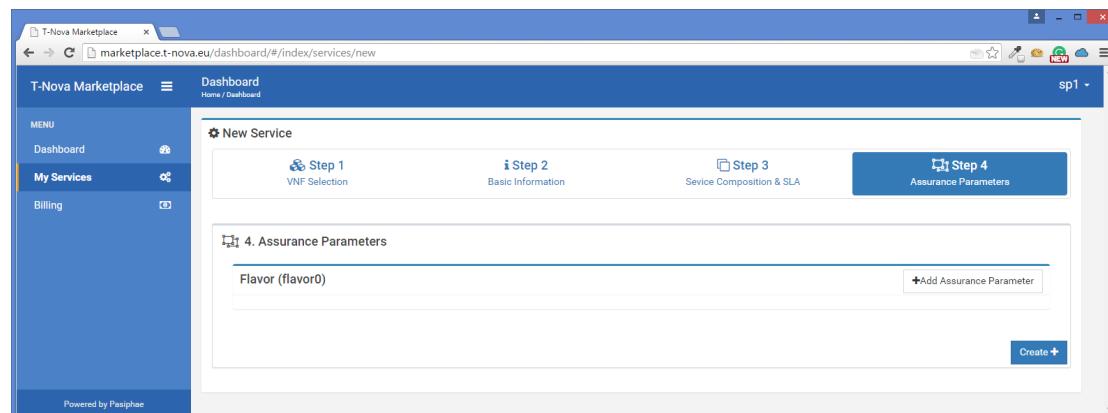
The next step is the service composition and the SLA specification. The Service Provider should specify SLA and add the constituent VNFs with the price and the billing model.



**Figure 5-23 NSD Creation - Step 3: Service Composition and SLA**

#### 5.4.1.5. Step 4 – Assurance Parameters

The final step is to define the assurance parameters for the service for every available flavour; the provider can apply discount policies, if there is any SLA violation.



**Figure 5-24 NSD Creation - Step 4: Assurance Parameters**

The screenshot shows the 'Assurance Parameter 1 (param0)' configuration. It includes a 'Monitoring Parameter' dropdown set to 'Availability (%)', an 'Expression' dropdown set to 'LT (Less Than)', a 'Threshold Value %' input field containing '50', a 'Constituent VNFs Expression' dropdown set to 'MIN (Minimum)', and a 'Constituent VNFs' section containing 'Untangle Firewall (VNF:563.availability)'. Below this are sections for 'Violations' (with 'Breaches Count' set to 2 and 'Interval (Seconds)' set to 360), 'Penalty Type' (set to 'Discount'), '% Discount' (set to 10), and 'Validity Value' (set to 2 with 'Validity Period' set to 'Day'). A '+Add Assurance Parameter' button is visible at the top right.

**Figure 5-25 NSD Creation - Step 4: Assurance Parameters**

## 5.4.2. NSD Listing

The Service provider can view and manage the created services, view the generated NSD or edit it with the NSD YAML editor or delete it.

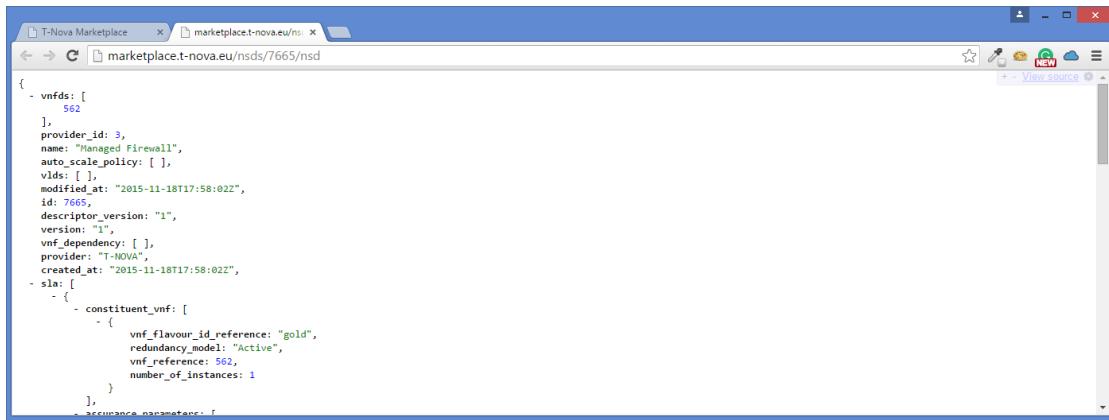
The screenshot displays a table of services under the 'Services' heading. It shows one service entry: '#7665 Managed Firewall'. The 'Description' column for this service states: 'Managed Firewall services ensure network security and availability by preventing unauthorized visitors from accessing valuable business resources.' To the right of the service entry are buttons for 'NSD' and 'YAML Editor'.

**Figure 5-26 NSD Listing**

## 5.4.3. NSD Tools

### 5.4.3.1. Generated NSD Viewer

The SP can view the generated NSD.

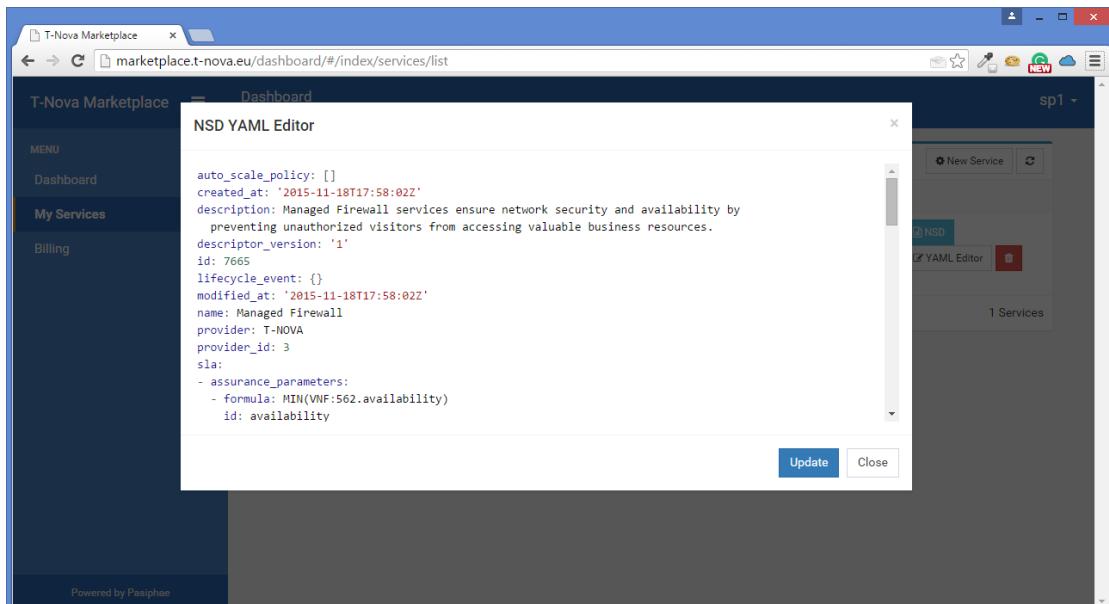


```
{
  - vnfds: [
    {
      provider_id: 3,
      name: "Managed Firewall",
      auto_scale_policy: [],
      vlds: [],
      modified_at: "2015-11-18T17:58:02Z",
      id: 7665,
      descriptor_version: "1",
      version: "1",
      vnf_dependency: [],
      provider: "T-NOVA",
      created_at: "2015-11-18T17:58:02Z",
      sla: [
        {
          - constituent_vnf: [
            {
              vnf_flavour_id_reference: "gold",
              redundancy_model: "Active",
              vnf_reference: 562,
              number_of_instances: 1
            }
          ],
          - assurance_parameters: [
            ...
          ]
        }
      ]
    }
  ]
}
```

**Figure 5-27 Generated NSD Viewer**

#### 5.4.3.2. NSD YAML Editor

The SP can edit the generated NSD with the NSD YAML Editor

**Figure 5-28 NSD YAML Editor**

## 5.5. Customer

### 5.5.1. Customer – Service Selection

The Customer can browse the available services.

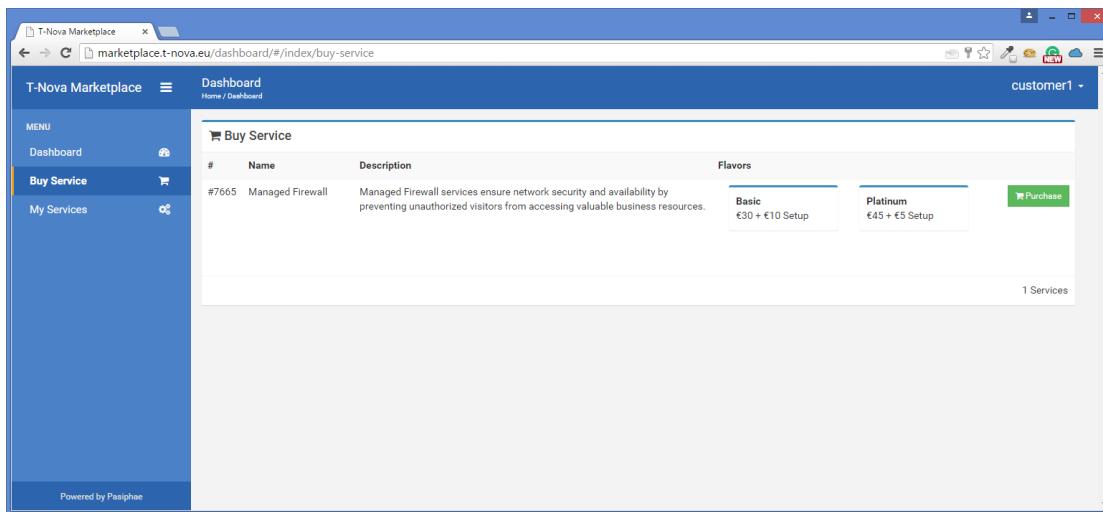


Figure 5-29 Customer – Service Selection

### 5.5.2. Customer - Service Purchase

The Customer can select the flavor of the service and purchase it.

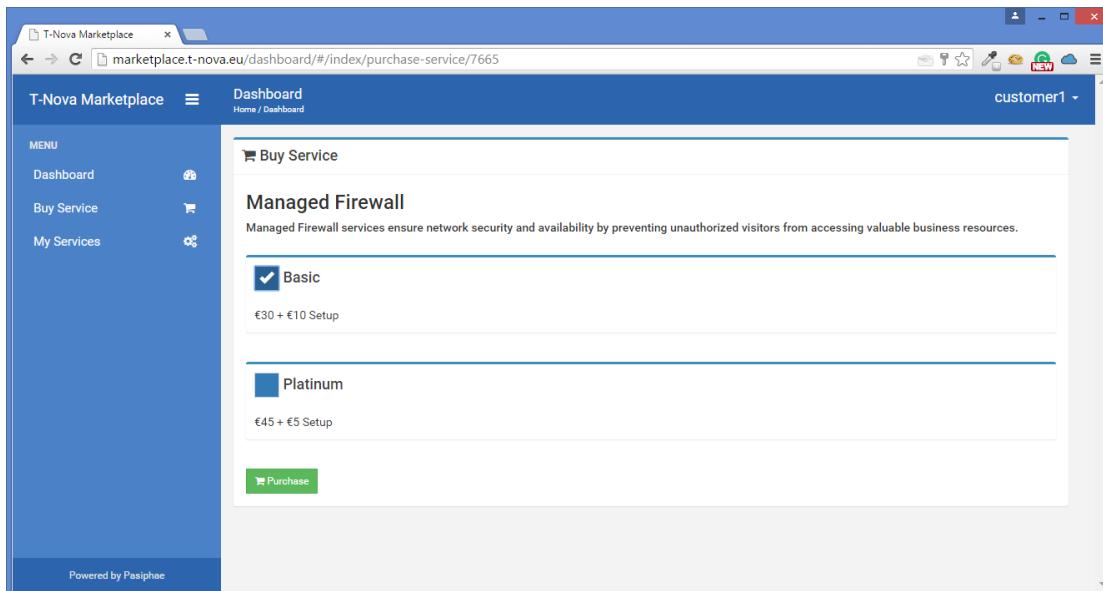


Figure 5-30 Customer - Service Purchase

### 5.5.3. Customer Services

The customer can view the running services.

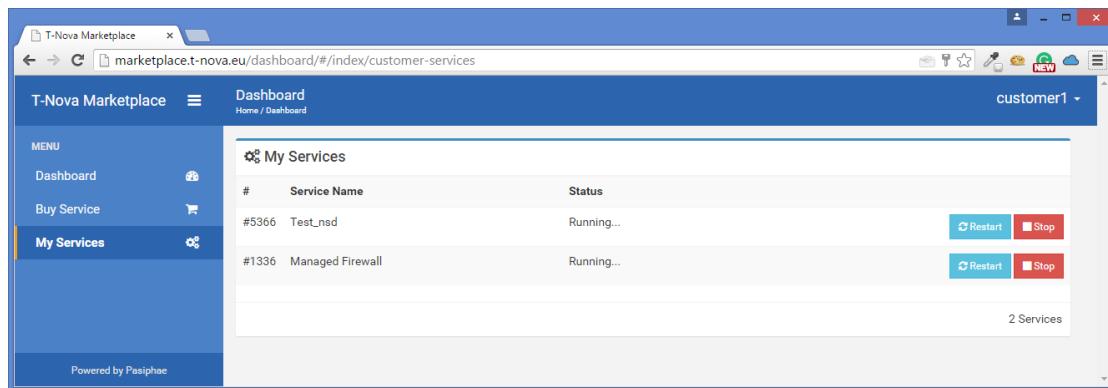


Figure 5-31 Customer Services

## 5.6. Administrator

This is the Dashboard's administrator home page.

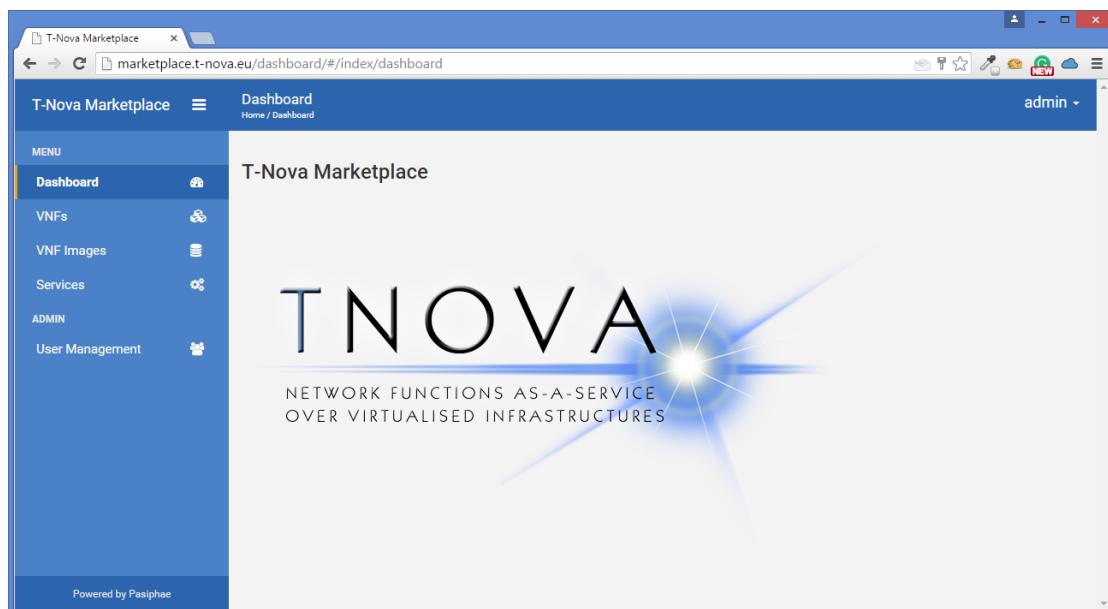


Figure 5-32 Administrator

### 5.6.1. User Management

The administrator can manage the user accounts.

The screenshot shows the T-Nova Marketplace dashboard with the 'User Management' section selected. The main area displays a table of users with columns for #, Username, Company, Email, and Roles. Each user entry includes edit and delete icons. A total of 5 users are listed.

#	Username	Company	Email	Roles
1	admin	TEIC	g.alexiou@pasiphae.eu	A
2	customer1	TEIC	customer1@t-nova.eu	C
3	sp1	T-NOVA	sp1@t-nova.eu	SP
4	fp1	TEIC	fp1@t-nova.eu	FP
5	fp2	NCSR Demokritos	fp2@t-nova.eu	FP

Powered by Pasiphae

**Figure 5-33 User Management**

#### 5.6.1.1. User Creation

Administrator can create a new user.

The screenshot shows the 'New User' creation form. It requires input for Username, Company, Email, Password, Password Confirm, Role, First name, Last name, Address, City, and Country. A '+ Create' button is at the bottom right.

**Figure 5-34 User Creation**

#### 5.6.1.2. Edit User Profile

The administrator can edit a user account.

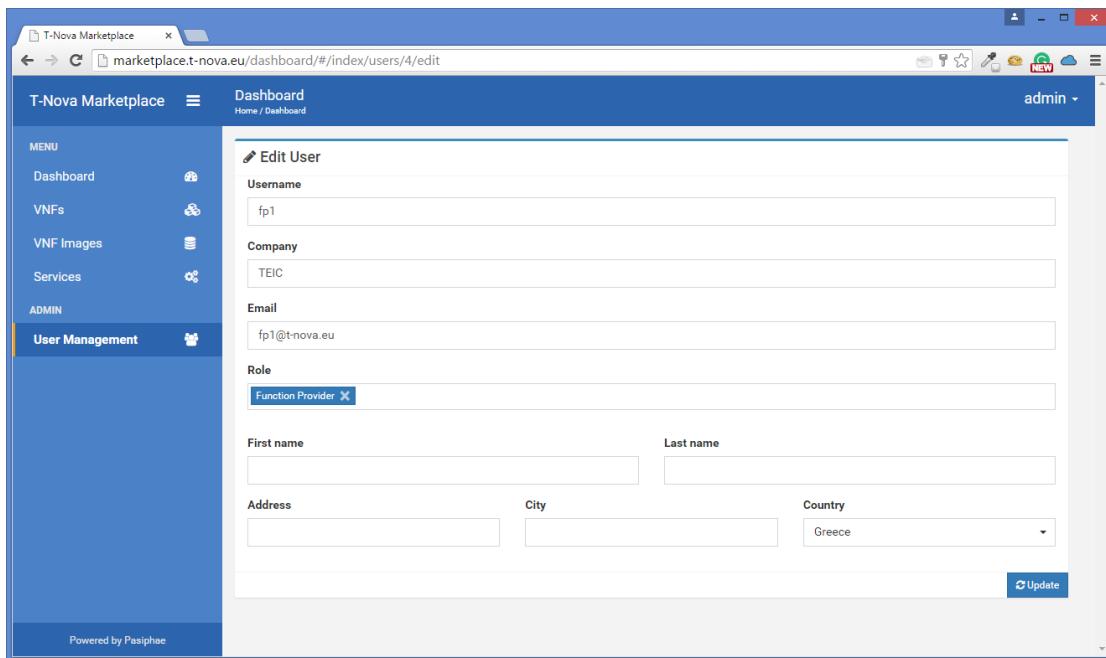


Figure 5-35 Edit User Profile

## 6. VALIDATION

### 6.1. Functional verification

To test the functionality of the Dashboard we have created some sample VNFs and Network Services with real data to make it as close to a real scenario as possible, trying to cover all the functionalities with one example.

#	Functionality	Action	Call from	Request to	Verification	Ok?
1	NSD storing	Store the recently created NSD in the BSC.	Dashboard	BSC	Verify the new NSD is present in the BSC database. Return code CREATED with the body including the autogenerated id after the call for NSD storage	Yes
2	Network services listing	List all available network services	Dashboard	BSC	Click on the NS catalogue and see the complete list on screen.	Yes
3	Filtered NS listing	List all available network services under a certain price	Dashboard	BSC	Set a maximum price, retrieve the filtered list of services and see all the network services in the list are below that price.	Yes
4	NSD modification	Select a NS from the BSC, modify any field	Dashboard	BSC	Verify the NSD has been updated in the BSC database.	Yes
5	NS deletion	Mark a NS for deletion from the BSC	Dashboard	BSC	Verify it has been deleted by retrieving again the complete list of services.	Yes
6	Providers registration	Register FP, SP and customer	Dashboard	SLA mgmt.	Verify the introduced providers are in the SLA management module database	Yes
7	VNF Templates introduction	Create VNF template based on the VNFD and send it to the SLA management module	Dashboard	SLA mgmt.	Check the SLA module database for the template and the logs to see there has not been errors in the process	Yes
8	Request bill	Request a bill generation for some user for a	Dashboard	Billing	The response contains the amount due for the customer.	Yes

		given frame	time		The charge data records can be verified in the InfluxDB series.	
9	Request revenues report	Make a call to the billing module with the function provider ID and the time frame	Dashboard	Billing	The response contains the revenue share report along with any penalties for SLA violations. The periodic reports can be verified in the InfluxDB series.	Yes
10	Show SLA statistics	Request SLA statistics for a given NS or VNF, running or stopped	Dashboard	Accounting	See the charts drawn on the screen	Yes

Table 6-1 collects the results for this verification tests.

#	Functionality	Action	Call from	Request to	Verification	Ok?
1	NSD storing	Store the recently created NSD in the BSC.	Dashboard	BSC	Verify the new NSD is present in the BSC database. Return code CREATED with the body including the autogenerated id after the call for NSD storage	Yes
2	Network services listing	List all available network services	Dashboard	BSC	Click on the NS catalogue and see the complete list on screen.	Yes
3	Filtered NS listing	List all available network services under a certain price	Dashboard	BSC	Set a maximum price, retrieve the filtered list of services and see all the network services in the list are below that price.	Yes
4	NSD modification	Select a NS from the BSC, modify any field	Dashboard	BSC	Verify the NSD has been updated in the BSC database.	Yes
5	NS deletion	Mark a NS for deletion from the BSC	Dashboard	BSC	Verify it has been deleted by retrieving again the complete list of services.	Yes
6	Providers registration	Register FP, SP and customer	Dashboard	SLA mgmt.	Verify the introduced providers are in the SLA management module database	Yes

7	VNF Templates introduction	Create VNF template based on the VNFD and send it to the SLA management module	Dashboard	SLA mgmt.	Check the SLA module database for the template and the logs to see there has not been errors in the process	Yes
8	Request bill	Request a bill generation for some user for a given time frame	Dashboard	Billing	The response contains the amount due for the customer. The charge data records can be verified in the InfluxDB series.	Yes
9	Request revenues report	Make a call to the billing module with the function provider ID and the time frame	Dashboard	Billing	The response contains the revenue share report along with any penalties for SLA violations. The periodic reports can be verified in the InfluxDB series.	Yes
10	Show SLA statistics	Request SLA statistics for a given NS or VNF, running or stopped	Dashboard	Accounting	See the charts drawn on the screen	Yes

**Table 6-1 Dashboard functional verification**

## 6.2. Requirements fulfillment

The main design decision, derived from the requirements in D2.42, has been to provide a common dashboard with different customized views based on different roles. In this context, Table 6-2 holds the Dashboard's requirements that are common for the Service Providers, the Function Providers and the Customers, as well as its implementation status.

Req. id	Requirement Description	Justification of Requirement	Implementation Status	Implementation Justification
D.1	The dashboard SHALL provide a "login in" page for the different stakeholders to be authenticated	Stakeholders interacting with the T-NOVA system should be authenticated and authorised in order to be able to browse the Business Service Catalog, issue SLA requests, or upload NFVs	YES	All three Roles identified are implemented
D.2	The "login" page in the SHALL offer to the different stakeholders means to use (username, password, OpenID, Google API) for authentication	Stakeholders interacting with the T-NOVA system should be able to use different authentication techniques to access T-NOVA	Partially	Google API and OpenID are simple to implement and not needed for the Proof of Concept Implementation

Req. id	Requirement Description	Justification of Requirement	Implementation Status	Implementation Justification
D.3	The "login" page in the SHALL offer to the different stakeholders means to remember credentials when logging on	Stakeholders interacting with the T-NOVA system should not be obliged to insert credentials when accessing again the system	Yes	This is done through the Browser
D.4	The Dashboard SHALL be accessible to authorized users via the Internet	The Dashboard will provide the necessary interface in order to be viewed over the Internet	YES	The Dashboard is available on a public IP
D.5	The Dashboard SHOULD provide multiple users login and no less than 10	The Parallel access will provide the necessary tools for every user to be able to provide his content	YES	The dashboard supports multiple login from multiple roles
D.6	The Dashboard SHOULD be available 24/7 365 days per year	The Dashboard must be always on in order to control every part of the T-NOVA infrastructure.	Partially	This is an implementation guideline for the real environment
D.7	The Dashboard MUST be as light way as possible in the Server side	The Dashboard must be able to take into account the processing power of the stakeholder accessing it.	YES	The use of the technologies identified in this deliverable constitutes the Dashboard a Ultra-Light Web page.
D.8	The Dashboard MUST be Open source	The Dashboard must be Open source	YES	Dashboard will be provided open source with a Open license that needs to be defined

**Table 6-2 Dashboard basic requirements**

## 7. CONCLUSIONS

This deliverable presented the design, implementation and operation of T-NOVA's users Dashboard. More specifically, it provided an overview of the available web development frameworks used in frontend, middleware and backend deployments of web applications. Based on factors such as community support, maturity and extensibility, Bootstrap, AngularJS and Django were chosen as frontend, middleware and backend web development frameworks respectively. In addition it, discussed the existing software development approaches for developing a web application, namely the Server-Centric Web Application (SCWA) and Browser-Centric Web Application (SCWA), focusing on the latter because of its advantages in scalability. In the same context, it examined the monolithic and micro-services deployment approaches and proposed the adoption of micro-services architecture enhanced with the Docker paradigm; this innovative approach allowed for faster development cycles, interoperability between different software development frameworks and, most important, seamless-transparent deployment and execution of the Dashboard in any physical or virtual OS, without having to worry about libraries and third party tools incompatibilities. Following this, the implementation of the Dashboard was presented in the form of code-snippets screenshots. Finally, a detailed description of the APIs between the users' Dashboard and the rest T-NOVA management modules was given followed by the demonstration of the Dashboard's operation.

### 7.1.

## 8. REFERENCES

- [1] Semantic-UI, <http://semantic-ui.com/introduction/new.html>, Last Access 10/2015
- [2] BootStrap, <http://getbootstrap.com/>, Last Access 10/2015
- [3] AngularJS, <https://angularjs.org/>, Last Access 10/2015
- [4] JQuery, <https://jquery.com/>, Last Access 10/2015
- [5] Ember, <http://emberjs.com/>, Last Access 10/2015
- [6] React, <http://facebook.github.io/react/>, Last Access 10/2015
- [7] Django: The Web framework for perfectionists with deadlines, <https://www.djangoproject.com/>, Last Access 10/2015
- [8] Flask (A Python Microframework), <http://flask.pocoo.org/>, Last Access 10/2015
- [9] Symfony, High Performance PHP Framework for Web Development, <https://symfony.com/>, Last Access 10/2015
- [10] Yii PHP Framework: Best for Web 2.0 Development, <http://www.yiiframework.com/>, Last Access 10/2015
- [11] Docker – Build, Ship, and Run Any App, Anywhere, <https://www.docker.com/>, Last Access 11/2015

## **9. ANNEX A:**