wstore Documentation

Release v0.5.1

January 12, 2016

WStore is the reference implementation of the Store GE, and enables selling digital assets (i.e. applications, services and data) for consumers as well as developers of future Internet applications, and is responsible for managing offerings and sales

WStore supports:

- Registration and publication of new offerings by application/service and data providers
- Contracting of applications/services and data
- Gathering application/services (including data services) usage accounting info
- Charging for the acquisition and usage of application/services, on the basis of the predefined price model.

This project is part of FIWARE.

CHAPTER 1

Index

1.1 Installation and Administration Guide

1.1.1 Introduction

This Installation and Administration Guide covers WStore versions since 0.3 (0.3.0, 0.3.1, 0.4 and 0.5) corresponding to FIWARE releases 3.3.1, 3.3.2, 4.1.1 and 4.4.3). Any feedback on this document is highly welcomed, including bugs, typos or things you think should be included but aren't. Please send it to the "Contact Person" email that appears in the Catalogue page for this GEi.

1.1.2 Installation

Requirements

In order to have WStore up and running the following software is required. This section describes all the requirements of a basic WStore installation. However, these dependencies are not meant to be installed manually in this step, as they will be installed throughout the documentation

- A Web Server (i.e Apache)
- MongoDB
- Python 2.7. Python 3 and other versions are not supported.
- Django nonrel 1.3 or 1.4
- djangotoolbox
- django_mongodb_engine
- lxml
- rdflib 3.2.0+
- rdflib-jsonld
- Pymongo

- Whoosh
- paypalpy
- django-crontab
- django-social-auth
- wkhtmltopdf

Installing basic dependencies

The Web Server, MongoDB, wkhtmltopdf, python and pip itself can be installed using the package management tools provided by your operating system or using the available installers.

These packages are available for Linux and Mac OS so WStore should work in those systems. However, the current version of WStore and its installer / installation guide have been tested under Ubuntu 12.04, Ubuntu 13.10, Ubuntu 14.04, CentOS 6.3, CentOS 6.5 and CentOS 7. THESE ARE THEREFORE CONSIDERED AS THE SUPPORTED OPERATING SYSTEMS.

Note: WStore needs Python 2.7 to work; however, CentOS 6 uses Python 2.6 in the system. Although it is possible to install WStore in CentOS 6 (as explained before), it is strongly recommended to use an Ubuntu/Debian distribution.

Installing basic dependecies using the script

In order to facilitate the installation of the basic dependencies the script *resolve-basic-dep.sh* has been provided. This script will install the needed packages for both Ubuntu/Debian and CentOS systems. For CentOS 6 systems, this script will install Python 2.7 and its tools, without replacing the system Python, making them avalailable as python2.7, pip2.7 and vitualenv2.7.

Note: The script *resolve-basic-dep.sh* may replace some of your system packages, so if you have software with common dependencies you may want to manually resolve WStore basic dependencies.

To execute the script run the following command

```
$ sudo ./resolve-basic-dep.sh
```

Manually resolving basic dependencies

Following, you can find how to resolve WStore basic dependencies if you do not want to use the script. Be aware that some commands require to be executed as root.

Debian/Ubuntu

To install Python and pip

```
# apt-get install python python-pip
```

To install MongoDB

```
# apt-get install mongodb
```

To install wkhtmltopdf

```
# apt-get install wkhtmltopdf
```

CentOS/RedHat

As mentioned above, CentOS 6 systems include Python 2.6. Replacing this Python version with Python 2.7 may break the system, so it should be installed separately.

To install Python 2.7, you need to resolve some development dependencies

```
# yum groupinstall "Development tools"
# yum install zlib-devel
# yum install bzip2-devel
# yum install openssl-devel
# yum install ncurses-devel
```

The next step is to download and compile Python 2.7

```
# cd /opt
# wget --no-check-certificate https://www.python.org/ftp/python/2.7.6/Python-2.7.6.tar.xz
# tar xf Python-2.7.6.tar.xz
# cd Python-2.7.6
# ./configure --prefix=/usr/local --enable-shared
# make && make altinstall
```

Then, include the line */usr/local/lib* at the end of the file */etc/ld.so.conf*, In a CentOS 6.5 it shoud be similar to:

```
include ld.so.conf.d/*.conf
/usr/local/lib
```

To finish with Python 2.7 installation execute the following command:

```
# /sbin/ldconfig
```

Finally, install Python 2.7 setup tools

```
# cd /opt
# wget https://bitbucket.org/pypa/setuptools/raw/bootstrap/ez_setup.py
# /usr/local/bin/python2.7 ez_setup.py
# /usr/local/bin/easy_install-2.7 pip

# ln -s /usr/local/bin/python2.7 /usr/bin/python2.7
# ln -s /usr/local/bin/pip2.7 /usr/bin/pip2.7
```

Now, Python 2.7 and its pip are available as python2.7 and pip2.7

In CentOS 7, python 2.7 is included with the system. To install pip execute the following commands:

```
# rpm -iUvh http://dl.fedoraproject.org/pub/epel/7/x86_64/e/epel-release-7-5.noarch.rpm
# yum -y update
# yum install -y python-pip
```

MongoDB is included in the official MongoDB downloads repositories. Once the related repositories has been included (see http://docs.mongodb.org/manual/tutorial/install-mongodb-on-red-hat-centos-or-fedora-linux/) install MongoDB with the command

```
# yum install -y mongodb-org
```

To install wkhtmltopdf get the related rpm for your system from http://wkhtmltopdf.org/downloads.html and install the package. For example, version 0.12.1 for a 64 bits architecture:

```
# wget http://download.gna.org/wkhtmltopdf/0.12/0.12.1/wkhtmltox-0.12.1_linux-centos6-amd64.rpm
# rpm -ivh wkhtmltox-0.12.1_linux-centos6-amd64.rpm
```

Resolving extra dependencies

Once basic dependencies have been resolved, it is possible to install python and Django dependencies using the provided scripts (As explained in the next section). However, before launching the installation script you should be aware of some aspects:

Note: If you have used the script *resolve-basic-dep.sh* to resolve the basic dependencies you do not need to install the following packages, since they are already installed.

- The script used to resolve python dependencies will create a virtual environment for the project with the corresponding packages, so to use this script you need virtualenv2.7 and python 2.7.

```
    # Ubuntu/Debian, CentOS 7
    $ pip install virtualenv

    #CentOS 6 (Suposing you have installed Python 2.7 following the previous instructions)
    $ pip2.7 install virtualenv
```

- WStore uses wkhtmltopdf for creating invoices. This software requires an X Server to work. If you do not have one, WStore will try to run Xvfb on the display :98. To install Xvfb use the following command.

```
    # Ubuntu/Debian
    $ apt-get install xvfb

    #CentOS/RedHat
    $ yum install xorg-x11-server-Xvfb
```

- It is possible that the setup.sh script fails while installing lxml. See http://lxml.de/installation.html#installation if in trouble installing lxml. You probably have to install the following packages.

```
    # Ubuntu/Debian
    $ apt-get install libxml2-dev libxslt1-dev zlib1g-dev python-dev

    #CentOS/RedHat
    $ yum install libxml2-devel libxslt-devel zlib-devel python-devel
```

Installing WStore

Installing WStore using scripts

To install WStore the script *setup.sh* has been provided. This script resolve all needed python and django dependencies (This script does not install the basic dependencies such as MongoDB, python, etc), and execute a complete test in order to ensure that WStore is correctly installed.

Be aware os having MongoDB up and running before executing the script. If MongoDB fails when starting you may need to configure the smallfiles option (see http://docs.mongodb.org/manual/reference/configuration-options/).

```
# Ubuntu/Debian
$ service mongodb start

# CentOS/RedHat
$ service mongod start
```

You can execute the script *setup.sh* to perform the complete installation. **Please note that this script should be run as an user without using sudo (no root permissions are needed, although root user is allowed).** Executing the script using sudo will cause Python and Django packages to be installed in the system, not in the virtualenv, which can cause WStore not working properly or even break your system if using CentOS.

```
$ ./setup.sh
```

The setup.sh script will also offers you a wizard to ease the configuration process. This wizard will generate the settings.py file for you, so if you follow the wizard, you can avoid following the Configuration section (unless you want to introduce some specific configuration). However, it is highly recommended to read the Configuration section for a better understanding of the parameters. To use this wizard, just type ‘y’ when asked:

```
Do you want to create an initial configuration? [y/n]:
y
```

First, you will be required a database name. You can introduce the name that you want:

```
Include a database name:
wstore_db
```

Then, you should include a site name. This value is up to you:

```
Include a site name:
store
```

After that, the script will ask you the domain where the Store is to going run. You must introduce a valid domain because otherwise the Store won’t run.

```
Include a site domain:
http://host:port
```

Later, you will be required to introduce the name of your store instance. You are free to introduce any name that you want. This will be the name used to register your WStore instance in external components such as the Marketplace:

```
Include a name for your instance:
FIWARE
```

Then, the script will ask you for a basic e-mail configuration. If you don’t want to provide a mail configuration, just type ‘n’ when asked.

```
Do you want to include email configuration? [y/n]:
<y/n>
```

If you choose to include the mail configuration, you will be asked for a SMTP server, a mail address, a mail user, and the password associated to that user. This mail configuration will be used as the source address for notifications sent by email. You will be also asked for a requests mail that will be used as the destination mail for user requests asking for the provider role:

```
Include email smtp server endpoint:
{YOUR_SMPT_SERVER}
Include WStore email:
{YOUR_EMAIL_ADDRESS}
Include WStore email user:
{USER_NAME}
Include WStore email password:
{PASSWORD}
Include WStore provider requests email:
{REQUEST_MAIL}
```

Finally, you must choose the authentication method. You have two possible options: use (1) an identity manager or (2) the Django Authentication System.

```
Select authentication method:
1) Identity manager
2) WStore
```

If you choose the identity manager option, you will be asked for the identity manager endpoint, and the basic OAuth2 configuration (Client ID and Client Secret). You can avoid to introduce the basic OAuth2 configuration if you don't have the credentials at that moment. However, in order to start the Store, you need to introduce this information in the settings.py file as explained in the Configuration section. Note that for using this authentication method you must have registered your WStore instance in the identity Manager using the Callback URL explained in the configuration section of this document.

```
Include Identity manager endpoint:
{IDM_END_POINT}
Do you want to include OAuth2 configuration? [y/n]:
y
Include Client id:
{CLIENT_ID}
Include client secret:
{CLIENT_SECRET}
```

If you are installing WStore version 0.5 (FIWARE 4.4.3), it includes support for KeyRock new version, so this step requires more information. First you will be asked to provide the identity manager endpoint. In this case, you can leave the default one (The one in the FIWARE Lab). If you choose a different identity manager, you will need to provide the API version and the endpoint of the KeyStone instance of the concrete cloud environment

```
Include Identity manager endpoint: (default https://account.lab.fiware.org/)
{IDM_END_POINT}
Include KeyRock API version [1/2]:
{API_VERSION}
Include KeyStone endpoint:
{KEYSTONE_ENDPOINT}
```

In you choose the Django Authentication System and you don't have a superuser in the selected database, you will be asked to create a new superuser in order to be able to manage the Store.

```
Would you like to create one now? (yes/no): yes
Username (leave blank to use 'basic'): {USERNAME}
E-mail address: {MAIL_ADDR}
Password: {PASS}
Password (again): {PASS}
```

If you don't want the wizard to start when the script is executed, you must run the script as follow:

```
$ ./setup.sh --noinput
```

Manually resolving python dependencies

In case you do not want to use the script *setup.sh*, Python and Django dependencies can be easily installed pip. Note that if you do not use the provided script, you will need to configure WStore manually as explained in the following section.

It is sugested to create a virtualenv where install Python and Django dependencies.

```
$ virtualenv-2.7 src/virtenv
```

or, if virtualenv-2.7 is not available

```
$ virtualenv src/virtenv
```

Then it is needed to activate the virtual env

```
$ source src/virtenv/bin/activate
```

To install *rdflib*, *lxml*, *pymongo*, Whoosh, Stemming, requests, and regex

```
$ pip install "lxml==3.4.4" "rdflib==4.2.0" "pymongo==2.8" "Whoosh==2.7.0" "Stemming==1.0.1" requests
```

Note: See http://lxml.de/installation.html#installation if in trouble installing lxml. You probably have to install the following packages:

```
# Ubuntu/Debian
$ apt-get install libxml2-dev libxslt1-dev zlib1g-dev python-dev

#CentOS/RedHat
$ yum install libxml2-devel libxslt-devel zlib-devel python-devel
```

WStore requires the *Django nonrel* framework ready to work with *MongoDB*. To install this framework in its version 1.4 as well as *djangotoolbox* and *django_mongodb_engine* for this version use the following commands:

```
$ pip install https://github.com/django-nonrel/django/archive/nonrel-1.4.zip
```

```
$ pip install https://github.com/django-nonrel/djangotoolbox/archive/toolbox-1.4.zip
```

```
$ pip install https://github.com/django-nonrel/mongodb-engine/archive/mongodb-engine-1.4-beta.zip
```

To install the *rdflib* plugin for json-ld format use the following command:

```
$ pip install https://github.com/RDFLib/rdflib-jsonld/archive/master.zip
```

To install the PayPal module *paypalpy* use the following command:

```
$ pip install https://github.com/conwetlab/paypalpy/archive/master.zip
```

WStore uses some plugins for django, to install them use the following commands:

```
$ pip install "nose==1.3.6" "django-nose==1.4"
```

```
$ pip install "django-social-auth==0.7.28"
```

```
$ pip install "django-crontab==0.6.0"
```

1.1.3 Configuration

Note that if the script has been used to resolve WStore python dependencies, they have been installed in a virtual environment that must be activated before running any configuration command (*python manage.py {command}*). To activate the virtualenv execute the following command from the installation directory.

```
$ source src/virtenv/bin/activate
```

Moreover, if you have followed the configuration wizard of the *setup.sh* script you can skip this section. However, it is highly recomended to read it in order to understand the different configuration settings.

Database Configuration

The preliminary configuration of the database connection is included in *settings.py* and is ready to work using MongoDB in the default host and port, with a database called wstore_db, and without security. To modify the database connection configuration edit the *DATABASES* setting:

```
DATABASES = {
    'default': {
        'ENGINE': 'django_mongodb_engine',
        'NAME': 'wstore_db',
        'USER': '',
        'PASSWORD': '',
        'HOST': '',
        'PORT': '',
        'TEST_NAME': 'test_database',
    }
}
```

Using this setting is possible to change the database name and the test database name, include an user and password, and specify the host and port of MongoDB.

Note: The engine field cannot be changed, since WStore only works with MongoDB.

The name of the instance is included in the *STORE_NAME* setting:

```
STORE_NAME = 'WStore'
```

Creating the deafult site

WStore (and any software using django_mongodb_engine and django sites framework) requires the creation of a default ‘’Site” model. To create the default site execute the following command including a site name and the site domain where your instance is going to run:

```
$ python manage.py createsite site_name http://host:port
```

Get the default site id:

```
$ python manage.py tellsiteid
```

Include the site id in ‘’settings.py” updating the ‘’SITE_ID” setting

```
SITE_ID = u'515ab0738e05ac20b622888b'
```

PayPal Credentials Configuration

WStore can use PayPal to perform chargings. To activate this fuctionality change PAYMENT_METHOD setting:

```
PAYMENT_METHOD = 'paypal'
```

In order to receive the payments, it is necessary to include the credentials of a Business PayPal account in the *src/wstore/charging_engine/payment_client/paypal_client.py* file. In this file is also possible to configure the endpoints used by PayPal, this settings contain by default the testing sandbox endpoints.

```
# Paypal creadetials
PAYPAL_USER = '<PayPal_user_name>'
PAYPAL_PASSWD = '<PayPal_password>'
```

```
PAYPAL_SIGNATURE = '<PayPal_signature>'
PAYPAL_URL = 'https://api-3t.sandbox.paypal.com/nvp'
PAYPAL_CHECKOUT_URL='https://www.sandbox.paypal.com/webscr?cmd=_express-checkout'
```

Pay-Per-Use Cron Configuration

WStore uses a Cron task to perform the aggregation and charging of Pay-per-use information. The periodicity of this task can be configured using the CRONJOBS setting of settings.py using the standard Cron format.

```
CRONJOBS = [
    ('0 5 * * *', 'django.core.management.call_command', ['resolve_use_charging']),
]
```

Once the Cron task has been configured, it is necessary to include it in the Cron tasks using the command:

```
$ python manage.py crontab add
```

It is also possible to show current jobs or remove jobs using the commands:

```
$ python manage.py crontab show

$ python manage.py crontab remove
```

Email configuration

WStore uses some email configuration for sending notifications. To configure the source email used by WStore for sending notifications include the following settings:

```
WSTOREMAILUSER = 'email_user'
WSTOREMAIL = 'wstore_email'
WSTOREMAILPASS = 'wstore_email_passwd'
SMTPSERVER = 'email_smtp_server'
```

It is also possible to configure a provider notification email. This email will be used by WStore as the destination email when an user requests the provider role. To set this email, include it in the *WSTOREPROVIDERREQUEST* setting:

```
WSTOREPROVIDERREQUEST = 'provider_requ_email'
```

Authentication method Configuration

WStore allows two different methods for the authentication of users. The method for users management should be selected in the initial configurantion of the WStore instance. Note that WStore does not store exactly the same info for the two methods, so, changing between authentication methods when the system has started to be used may cause unexpected behaviours.

FI-WARE Identity management

It is possible to delegate the authentication of users to the FI-WARE Identity Management system on a FI-WARE instance. View FI-LAB info in:

- http://help.lab.fi-ware.org

To do that, the first step is setting up the OILAUTH setting to True (Note that this is the default value).

```
OILAUTH=True
```

Then configure the authentication endpoint in filling the setting:

```
FIWARE_IDM_ENDPOINT='https://fiware_endpoint'
```

Next, register WStore as an application in the identity management portal, to do that WStore uses the following URL as as callback URL for OAuth2 authentication:

```
<host_wstore>/complete/fiware/
```

Once you have registered your WStore instance, get OAuth2 credentials needed for the authenticacion of your application. You will need to create some roles in your application, one for offering provider, other for offering customer, and a role for developers. This roles will be used in the organizations with access to your WStore instance in order to grant organization users the corresponding rights for purchasing and creating offerings for a complete organization. To include the name you have specified for that roles, you have to fill the following settings in social_auth_backend.py:

```
FIWARE_PROVIDER_ROLE='Name of the role'
FIWARE_CUSTOMER_ROLE='Name of the role'
FIWARE_DEVELOPER_ROLE='Name of the role'
```

Finally, include OAuth2 credentials in your WStore instance by filling the settings:

```
FIWARE_APP_ID = client_id_number
FIWARE_API_SECRET = client_secret
```

If you are using WStore version 0.5 (FIWARE 4.4.3), you will need to include some aditional fields. First, it is needed to include the Idm API version. If the API has version 2, it is also required to include the KeyStone endpoint :

```
FIWARE_IDM_API_VERSION = 2
FIWARE_KEYSTONE_ENDPOINT = '{ KEYSTONE_ENDPOINT }'
```

WStore Identity Management

WStore has its own authentication mechanism based on django auth. To enable WStore authentication, set up the OILAUTH setting to False:

```
OILAUTH=False
```

For API accesses, WStore has an OAuth2 server that can be enabled by including the oauth2provider in the INSTALLED_APPS setting.

Applications can be registered in WStore using the django admin view.

Database Population

Before running WStore, it is necessary to populate the database. This can be achieved by using this command:

```
$ python manage.py syncdb
```

This command creates indexes for the different models of the database and ask if you want to create a Django superuser. In case you are using WStore authentication, this superuser is required in order to perform administrative tasks. If you are using FI-WARE authentication, users are taken from the identity management system, so do not create the user. Users with corresponding role (Provider) will be able to perform the administrative tasks.

An example of the output of this command follows:

```
...

You just installed Django's auth system, which means you don't have any superusers defined.
Would you like to create one now? (yes/no): yes
Username (leave blank to use 'francisco'): admin
E-mail address: admin@email.com
Password: ***** (admin)
Password (again): ***** (admin)
Superuser created successfully.

...
```

1.1.4 Final Steps

Make sure that the directories wstore_path/src/media, wstore_path/src/media/resources, wstore_path/src/media/bills, wstore_path/src/wstore/search/indexes exist, and that the server has sufficient permissions to write on them. For example, the following commands give permissions to apache user in a Debian/Ubuntu system:

```
# chgrp -R www-data  <wstore_path>/src/media <wstore_path>/src/wstore/search/indexes <wstore_path>/s

# chmod g+wrX -R <wstore_path>/src/media <wstore_path>/src/wstore/search/indexes <wstore_path>/src/w
```

Note: In a CentOS system the commands are similar but using *apache* instead of *www-data* as group.

it is possible to collect all static files in WStore in a single directory using the following command and answering yes when asked. Be aware of activating the virtualenv if needed as explained in the previous sections.

```
$ python manage.py collectstatic
```

1.1.5 Running WStore

Running WStore using the Django internal web server

Be aware that this way of running WStore should be used for evaluation purposes. Do not use it in a production environment.

Note: Since the installation scripts create a virtualenv to install the dependencies, you must activate virtualenv before running the runserver command if you have installed and configured the Store using these scripts. To do so, you must run the following command (in the src folder):

```
$ source virtenv/bin/activate
```

To start WStore, type the following command:

```
$ python manage.py runserver 0.0.0.0:8000
```

Then, go to http://computer_name_or_IP_address:8000/ where computer_name_or_IP_address is the name or IP address of the computer on which WStore is installed.

Integrating WStore with Apache

If you choose to deploy WStore in Apache, the *libapache2-mod-wsgi* module must be installed (and so does Apache!). To do so, type the following command in Ubuntu/Debian:

```
# apt-get install apache2 libapache2-mod-wsgi
```

In CentOS 6 systems apache can be installed as

```
# yum install -y httpd
```

In the case of *mod_wsgi* in CentOS 6, it is not possible to directly use the existing package. As explained in previous sections CentOS 6 relies in Python 2.6 to work, while WStore uses Python 2.7. For this reason when mod_wsgi is installed using yum, it uses Python 2.6, causing WStore not working properly over Apache.

To install mod_wsgi using python 2.7 (It suposes that you have installed Python 2.7 as explained in the *Installing basic dependencies* section) use the following commands (For version 4.3.0 of mod_wsgi):

```
# yum install -y httpd-devel
# wget https://github.com/GrahamDumpleton/mod_wsgi/archive/4.3.0.zip
# unzip 4.3.0.zip
# cd mod_wsgi-4.3.0/
# ./configure --with-python=/usr/local/bin/python2.7
# make install
# chmod 755 /usr/lib64/httpd/modules/mod_wsgi.so
```

Finally, turn on mod_wsgi in apache by creating the file */etc/httpd/conf.d/wsgi.conf* and including:

```
LoadModule wsgi_module modules/mod_wsgi.so
```

Then you have to populate the wsgi.py file:

```
import os
import sys
path = 'path_to_wstore/src'
if path not in sys.path:
    sys.path.insert(0, path)
os.environ['DJANGO_SETTINGS_MODULE'] = 'settings'
import django.core.handlers.wsgi
application = django.core.handlers.wsgi.WSGIHandler()
```

If you are running WStore using a virtualenv environment (for example if you have installed the dependencies using the provided script) your wsgi.py file sholud have the following structure:

```
import os
import sys
import site

site.addsitedir('vitualenv_path/local/lib/python2.7/site-packages')
path = 'path_to_wstore/src'
if path not in sys.path:
    sys.path.insert(0, path)

os.environ['DJANGO_SETTINGS_MODULE'] = 'settings'

# Activate your virtual env
activate_env=os.path.expanduser("vitualenv_path/bin/activate_this.py")
execfile(activate_env, dict(__file__=activate_env))

import django.core.handlers.wsgi
application = django.core.handlers.wsgi.WSGIHandler()
```

Please, pay attention that you set the right path to the wtore/src directory.

The next step consist on creating the virtualhost for WStore. To do that, it is possible to modify the default site configuration file (located in */etc/apache2/sites-available/* in an Ubuntu/Debian system or in */etc/httpd/sites-available* in a CentOS/RedHat system) or create a new site configuration file (i.e wstore.conf).

In a CentOS system you may need to create the *sites-enabled* and *sites-available* directories and include them in the apache configuration. To do that follow the next steps:

```
# cd /etc/httpd/
# mkdir sites-available
# mkdir sites-enabled
```

Then edit */etc/httpd/conf/httpd.conf* file and include the following lines at the end of the file

```
NameVirtualHost *:80
Include /etc/httpd/sites-enabled/
```

Once you have the site enabled, restart Apache

```
# Ubuntu/Debian
# service apache2 restart

# CentOS/RedHat
# service httpd restart
```

To configure WStore virtualhost add the following lines to the site configuration file:

```
<VirtualHost *:80>
        ...
        ### WStore ###
        WSGIScriptAlias / <path_to_django_wsgi>
        WSGIPassAuthorization On
        Alias /static <path_to_wstore>/src/static
        <Location "/static">
                SetHandler None
                <IfModule mod_expires.c>
                        ExpiresActive On
                        ExpiresDefault "access plus 1 week"
                </IfModule>
                <IfModule mod_headers.c>
                        Header append Cache-Control "public"
                </IfModule>
        </Location>
        <Location "/static/cache">
                <IfModule mod_expires.c>
                        ExpiresDefault "access plus 3 years"
                </IfModule>
        </Location>
        ...
</VirtualHost>
```

Again, pay special attention to the paths to the django wsgi file and the path_to_wstore/src/static directory.

Moreover, it is important that the apache user (www-data in Ubuntu/Debian, apache in CentOS/RedHat) could access the directory where WStore is deployed. Be aware of configuring the directory permissions so this user can access wstore directory and go through the previous directories in the path (x permission).

Finally, depending on the version of apache you are using, you may need to explicitly allow the access to the directory where WStore is deployed in the configuration of the virtualhost. To do that, add the following lines to your virtualhost:

Apache version < 2.4

```
<Directory /path/to/wstore/src>
    Order deny,allow
    Allow from all
</Directory>
```

Apache version 2.4+

```
<Directory /path/to/wstore/src>
    Require all granted
</Directory>
```

1.1.6 Sanity check Procedures

The Sanity Check Procedures are the steps that a System Administrator will take to verify that an installation is ready to be tested. This is therefore a preliminary set of tests to ensure that obvious or basic malfunctioning is fixed before proceeding to unit tests, integration tests and user validation.

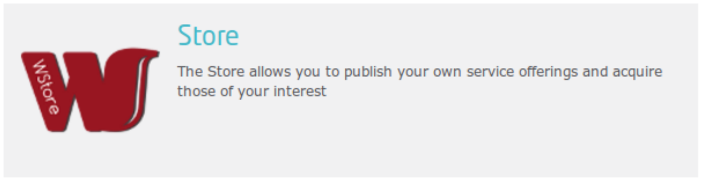
End to End Testing

Please note that the following information is required before performing this process.

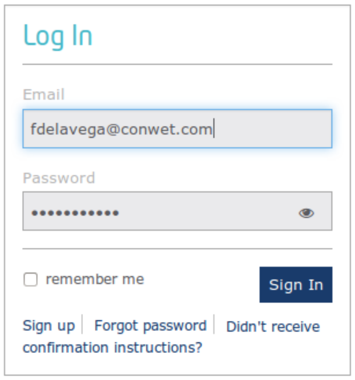
- The computer name or the IP address where WStore is running.
- Valid credentials for WStore (i.e credentials created during the syncdb command or an Identity Manager user).
- A logo image.

To check if WStore is running follow these steps:

1. Open a browser and enter WStore.
2. The login window should appear (WStore or idM depending on the configuration).

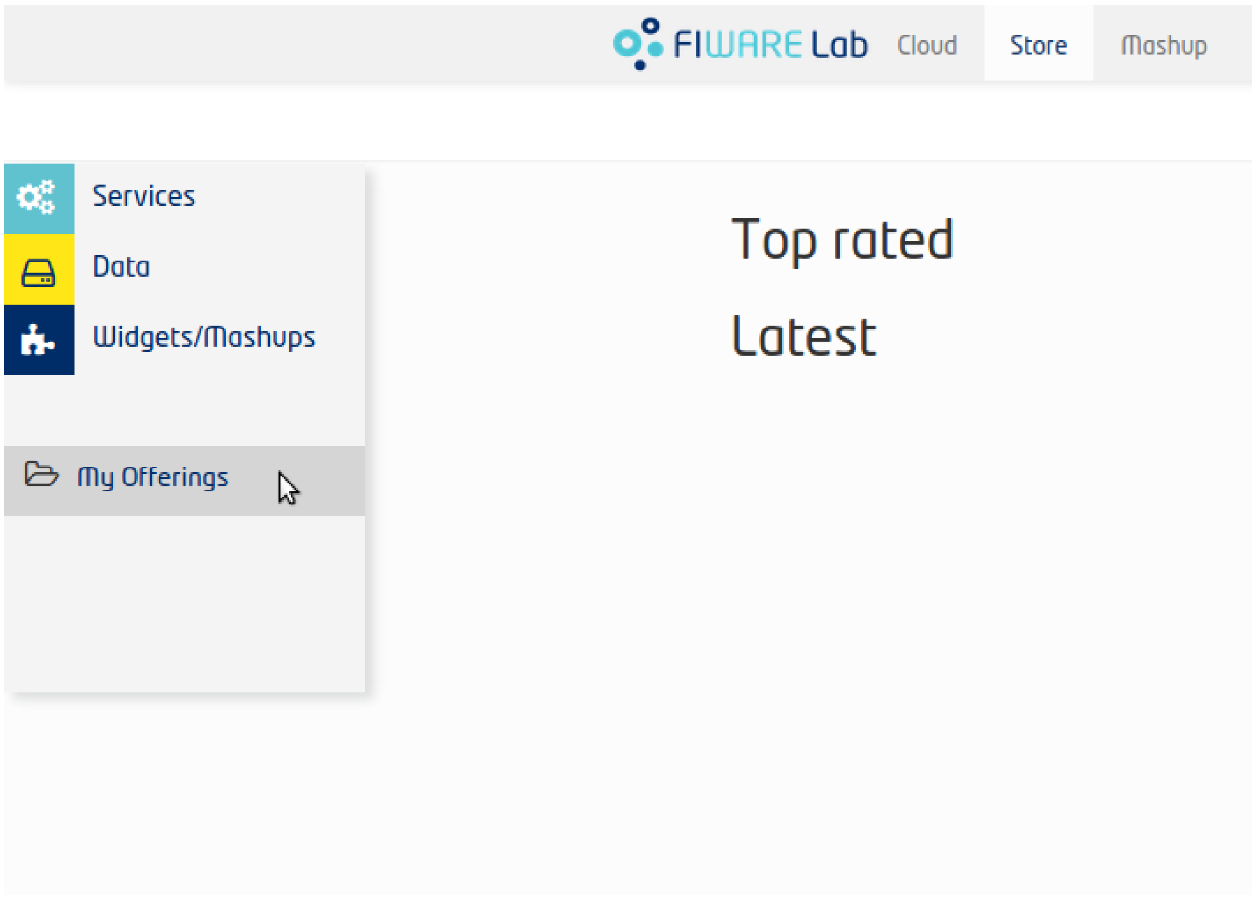


The Store allows you to publish your own service offerings and acquire those of your interest

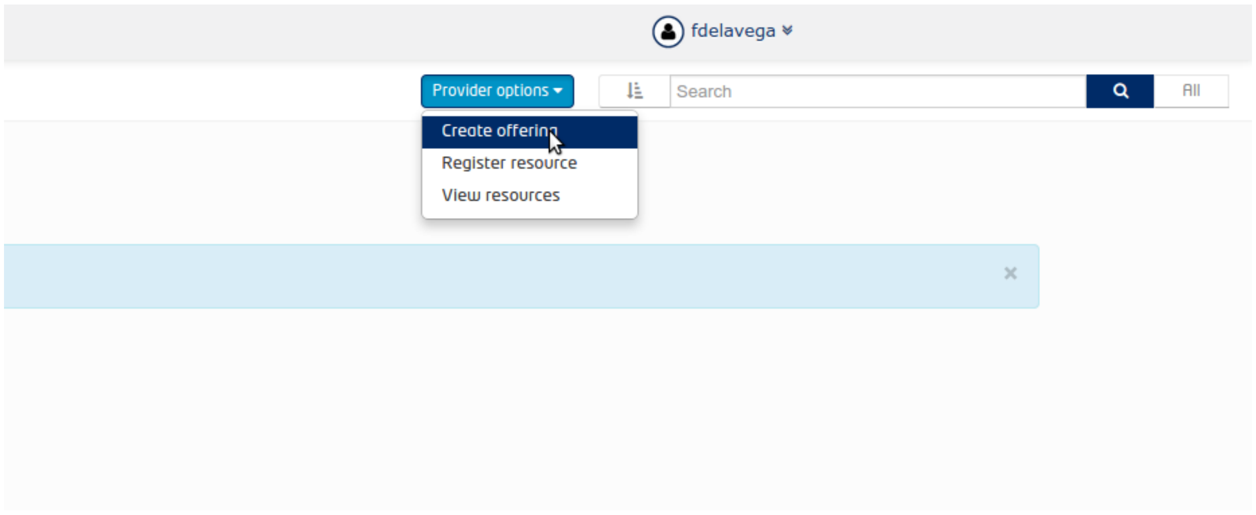


Email: fdelavega@conwet.com
Password: •••••••••••
☐ remember me

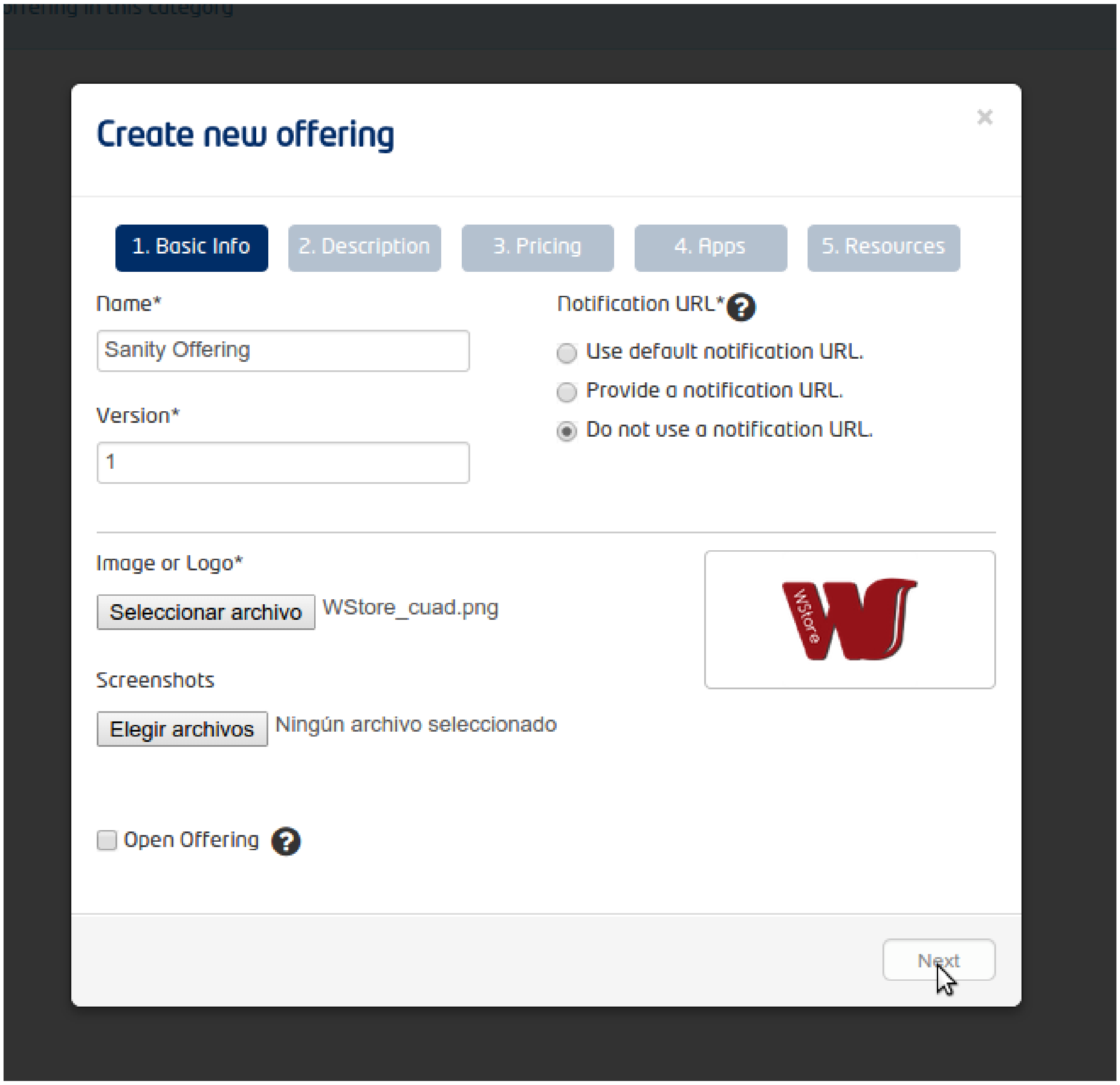
3. Introduce your credentials and click login
4. Go to *My Offerings*



5. Select the *Provider Options* dropdown and choose *Create offering*



6. Fill the name, and the version
7. Include the logo and the screenshots
8. Select the option for not providing a notification URL
9. Press *Next*



Name*: Sanity Offering

Version*: 1

Notification URL*

( ) Use default notification URL.

( ) Provide a notification URL.

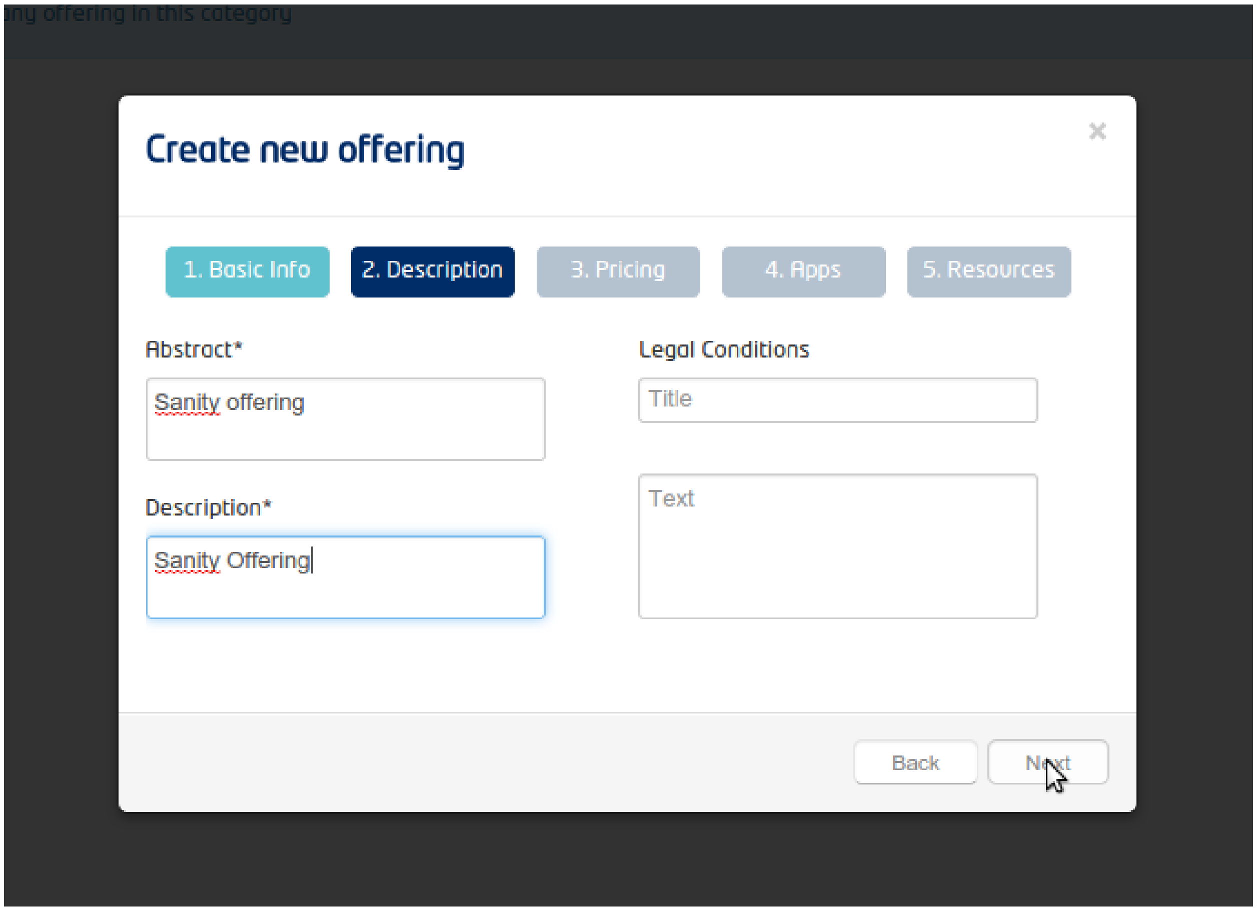
[✓ Do not use a notification URL.]

Image or Logo*: WStore_cuad.png

Screenshots: Ningún archivo seleccionado

☐ Open Offering

10. Fill description Info and press *Next*



Abstract*

Sanity offering

Description*

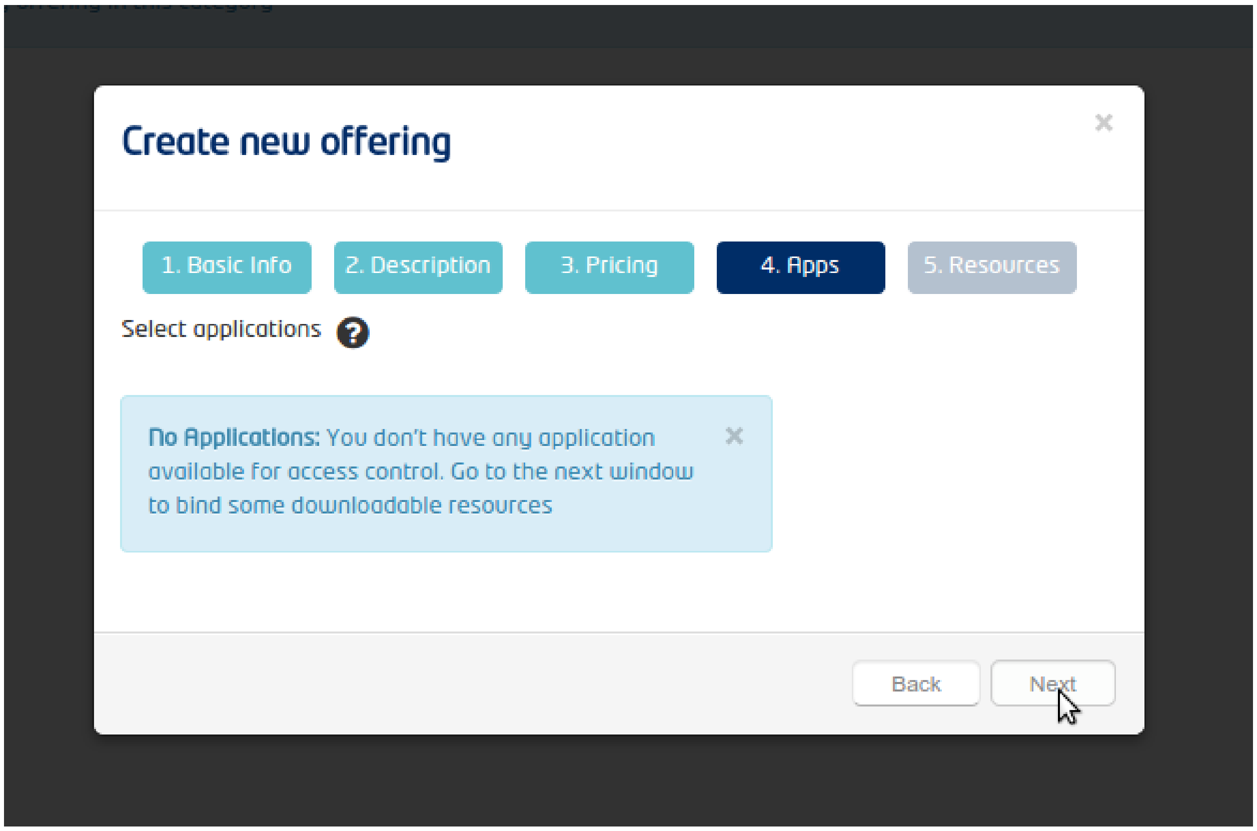
Sanity Offering

Legal Conditions

Title

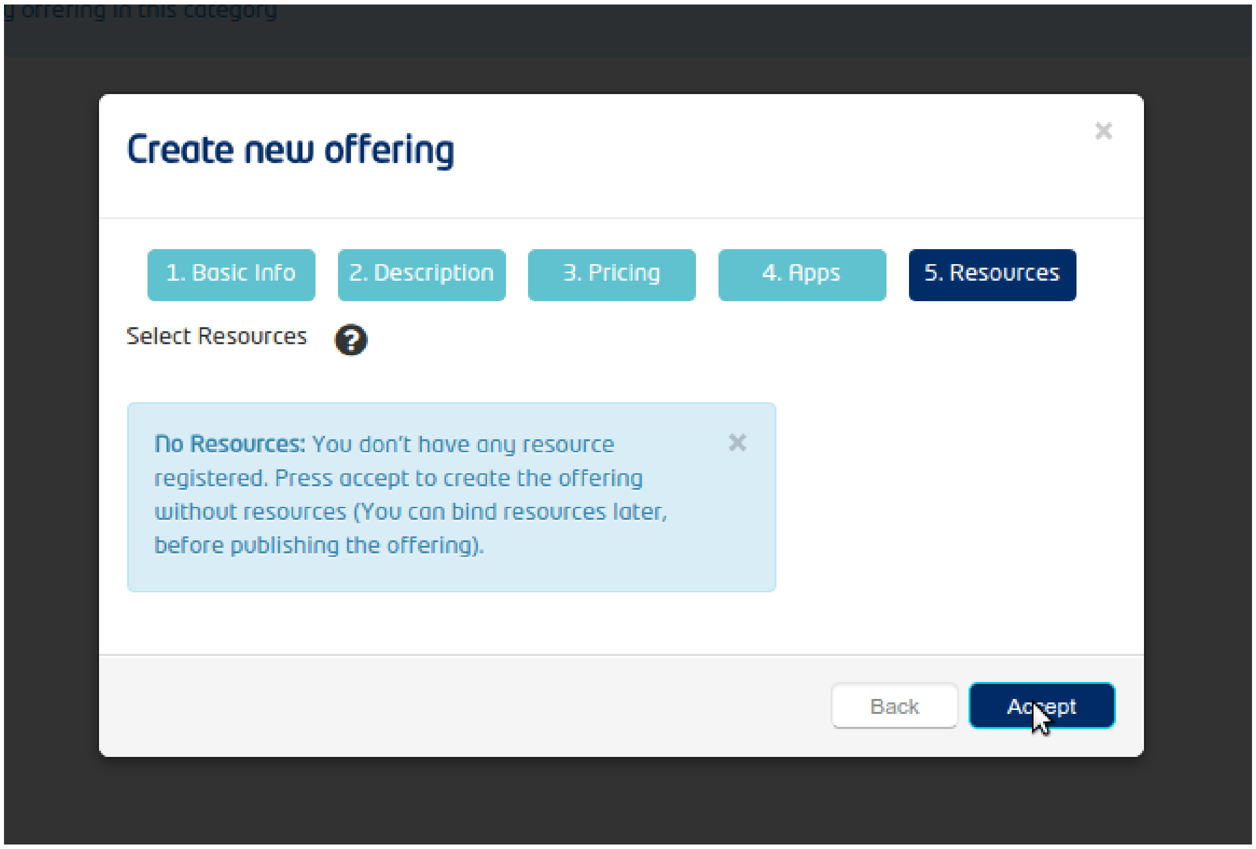
Text

10. Click *Next* in the pricing form.
14. Click *Next* in the Application selection form (This form only appears if using idM for authentication).



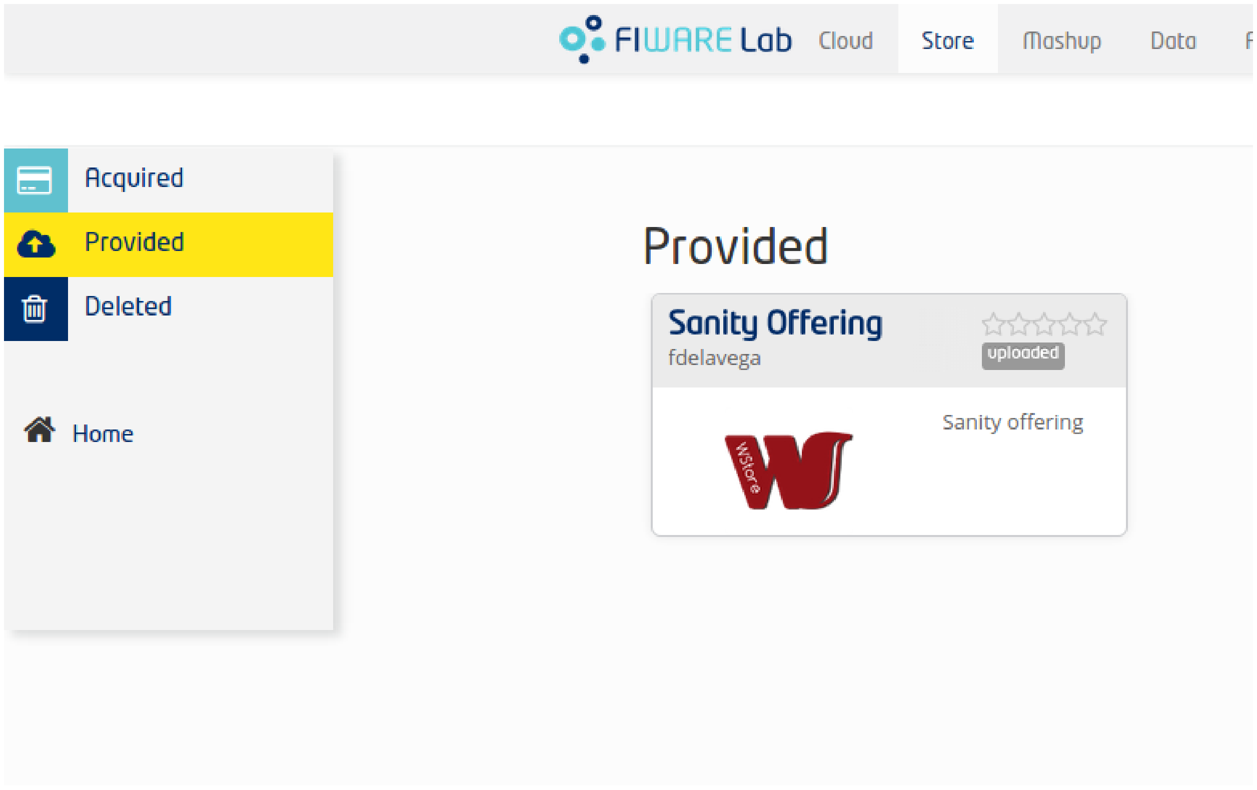
No Applications: You don't have any application available for access control. Go to the next window to bind some downloadable resources

15. Press *Accept*



No Resources: You don't have any resource registered. Press accept to create the offering without resources (You can bind resources later, before publishing the offering).

16. The created offering should appear in the Provided section



List of Running Processes

We need to check that the Apache web server and the MongoDB database are running. WStore uses a python interpreter, but it will not be listed as it runs embedded into apache2. If we execute the following command:

```
ps -ewF | grep 'apache2\|mongodb' | grep -v grep
```

It should show something similar to the following:

```
$ ps -ewF | grep 'apache2\|mongodb' | grep -v grep
root       1154     1  0 22744  3584   1 11:07 ?        00:00:00 /usr/sbin/apache2 -k start
www-data   1157  1154  0 22677  2620   2 11:07 ?        00:00:00 /usr/sbin/apache2 -k start
www-data   1178  1154  0 111374 6672   0 11:07 ?        00:00:00 /usr/sbin/apache2 -k start
www-data   1179  1154  0 111374 6672   2 11:07 ?        00:00:00 /usr/sbin/apache2 -k start
mongodb    4879     1  0 176281 16016  2 12:28 ?        00:00:01 /usr/bin/mongod --config /etc/mongodb
```

Network interfaces Up & Open

To check the ports in use and listening, execute the command:

```
$ sudo netstat -ltp
```

The expected results must be something similar to the following:

```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State       PID/Program name
tcp        0      0 localhost:27017         *:*                     LISTEN      4879/mongod
tcp        0      0 localhost:28017         *:*                     LISTEN      4879/mongod
tcp6       0      0 [::]:http               [::]:*                  LISTEN      1154/apache2
```

Databases

The last step in the sanity check, once that we have identified the processes and ports, is to check the MongoDB database that have to be up and accepting queries. If we execute the following command:

```
$ mongo wstore_db -u wstore -p wstore
```

It should show a message text similar to the following:

```
MongoDB shell version: 2.0.4
connecting to: wstore_db
>
```

1.1.7 Diagnosis Procedures

The Diagnosis Procedures are the first steps that a System Administrator will take to locate the source of an error in a GE. Once the nature of the error is identified with these tests, the system admin will very often have to resort to more concrete and specific testing to pinpoint the exact point of error and a possible solution. Such specific testing is out of the scope of this section.

Resource availability

Memory use depends on the number of concurrent users as well as the free memory available and the hard disk. WStore requires a minimum of 512 MB of available RAM memory, but 1024 MB of free memory are recomended.

Moreover, WStore requires at least 10 GB of hard disk space.

Remote Service Access

N/A

Resource consumption

Resource consumption strongly depends on the load, especially on the number of concurrent users logged in.

- The main memory consumption of the Apache Web server should be between 64 MB and 1024 MB.
- MongoDB main memory consumption should be between 30 MB and 500 MB.

I/O flows

The only expected I/O flow is of type HTTP, on port defined in Apache Web Server configuration files.

1.2 User and Programmer Guide

1.2.1 Introduction

This page contains the user and programmer guide for WStore, a reference implementation of the Store Generic Enabler.

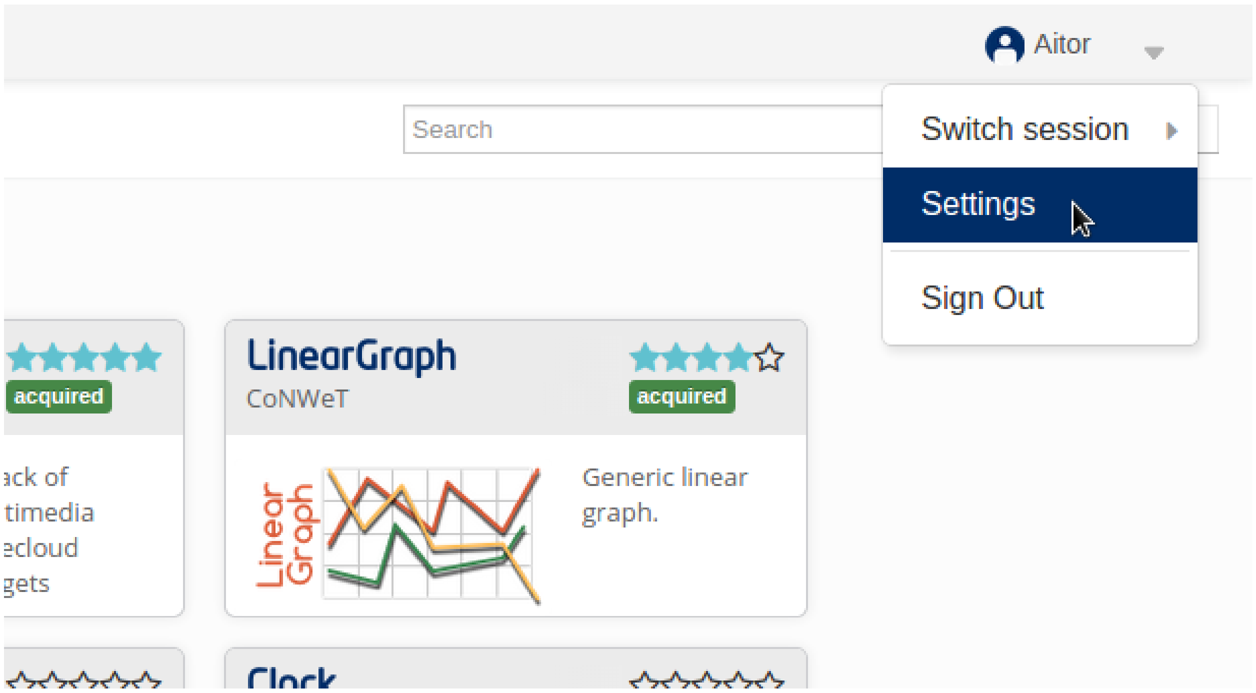
1.2.2 User Guide

This user guide contains a description of the different tasks that can be performed in WStore using its web interface from different points of view, depending on the roles of the corresponding user.

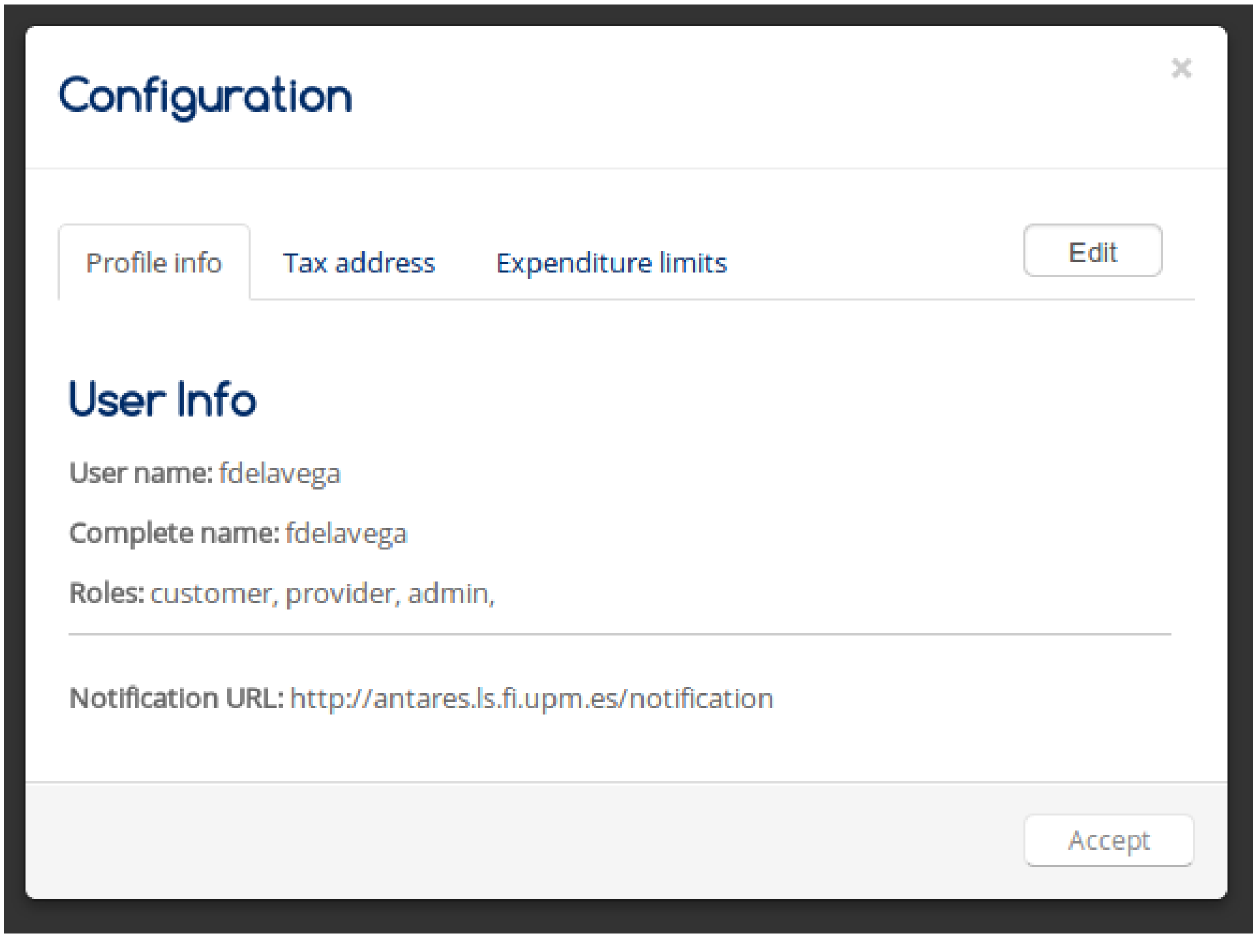
Profile configuration

All users can configure their profile including some information such as a default tax address.

To configure the user profile, move your mouse to username on the top right of the page and choose settings.



In the displayed modal, it is possible to view user profile info. To change this info or provide new one, select the Edit button.



User Info

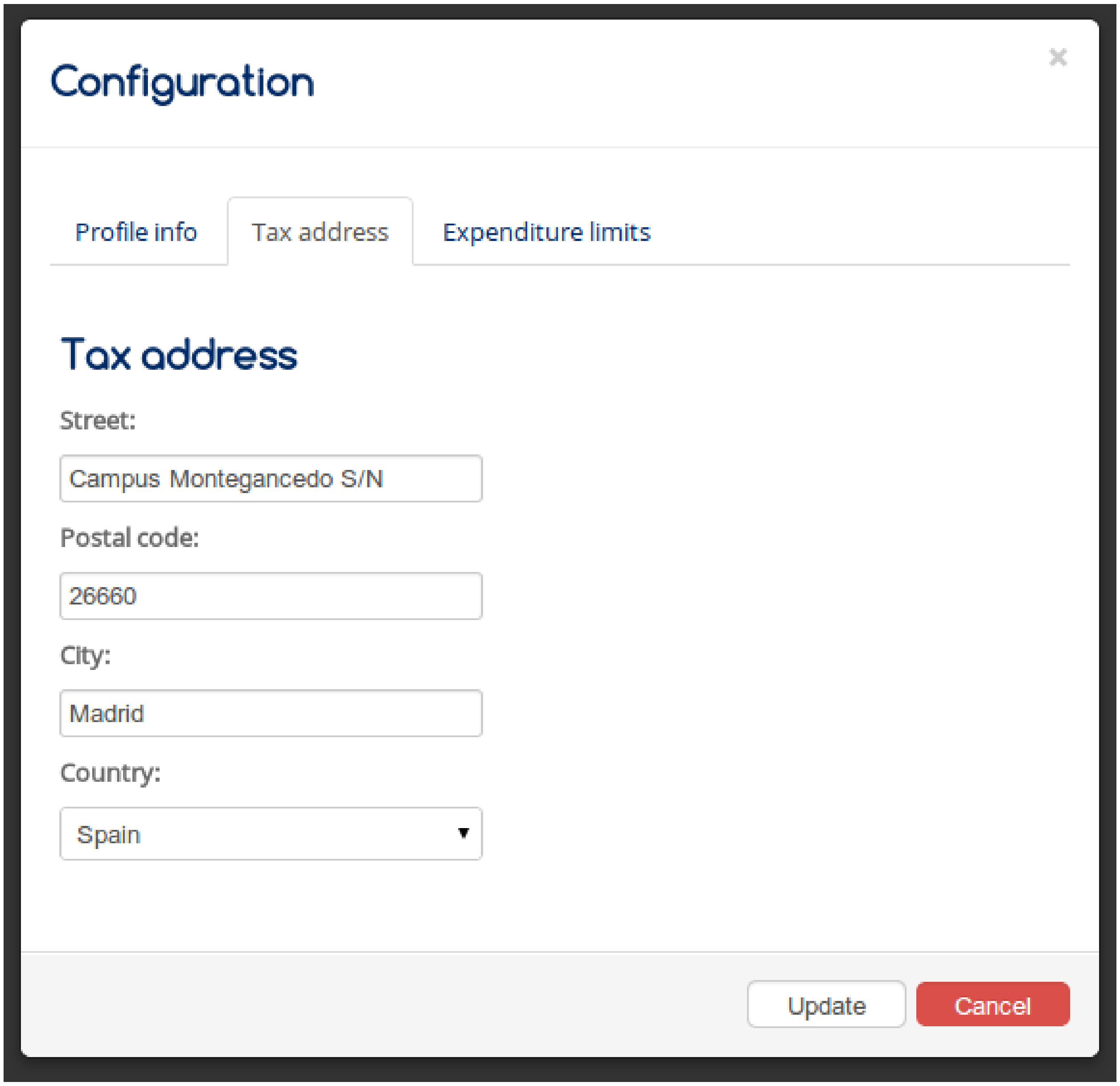
User name: fdelavega

Complete name: fdelavega

Roles: customer, provider, admin,

Notification URL: http://antares.ls.fi.upm.es/notification

Fill the different fields through the Tabs and press Update.



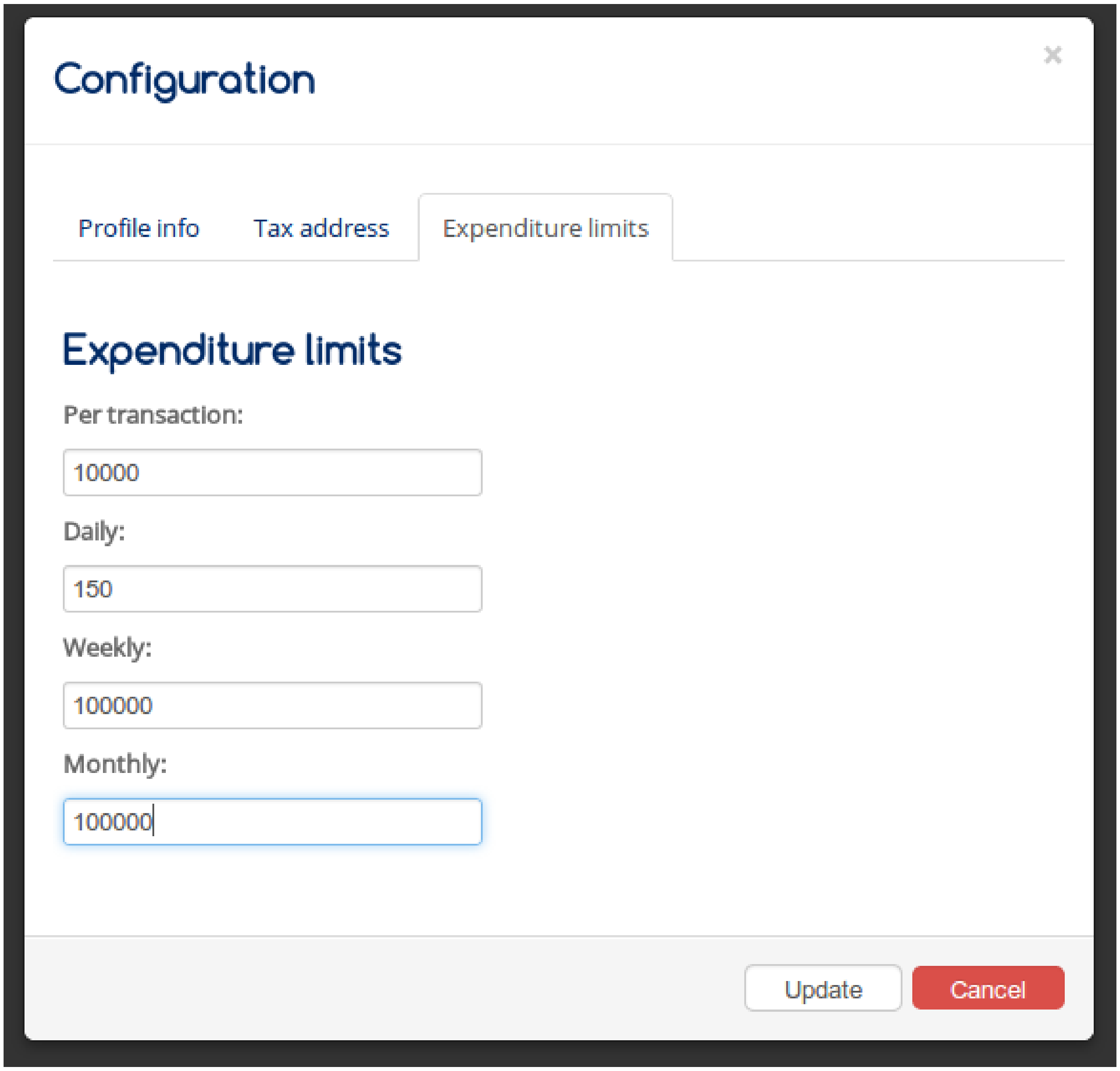
Tax address

Street: Campus Montegancedo S/N

Postal code: 26660

City: Madrid

Country: Spain



Expenditure limits

Per transaction: 10000

Daily: 150

Weekly: 100000

Monthly: 100000

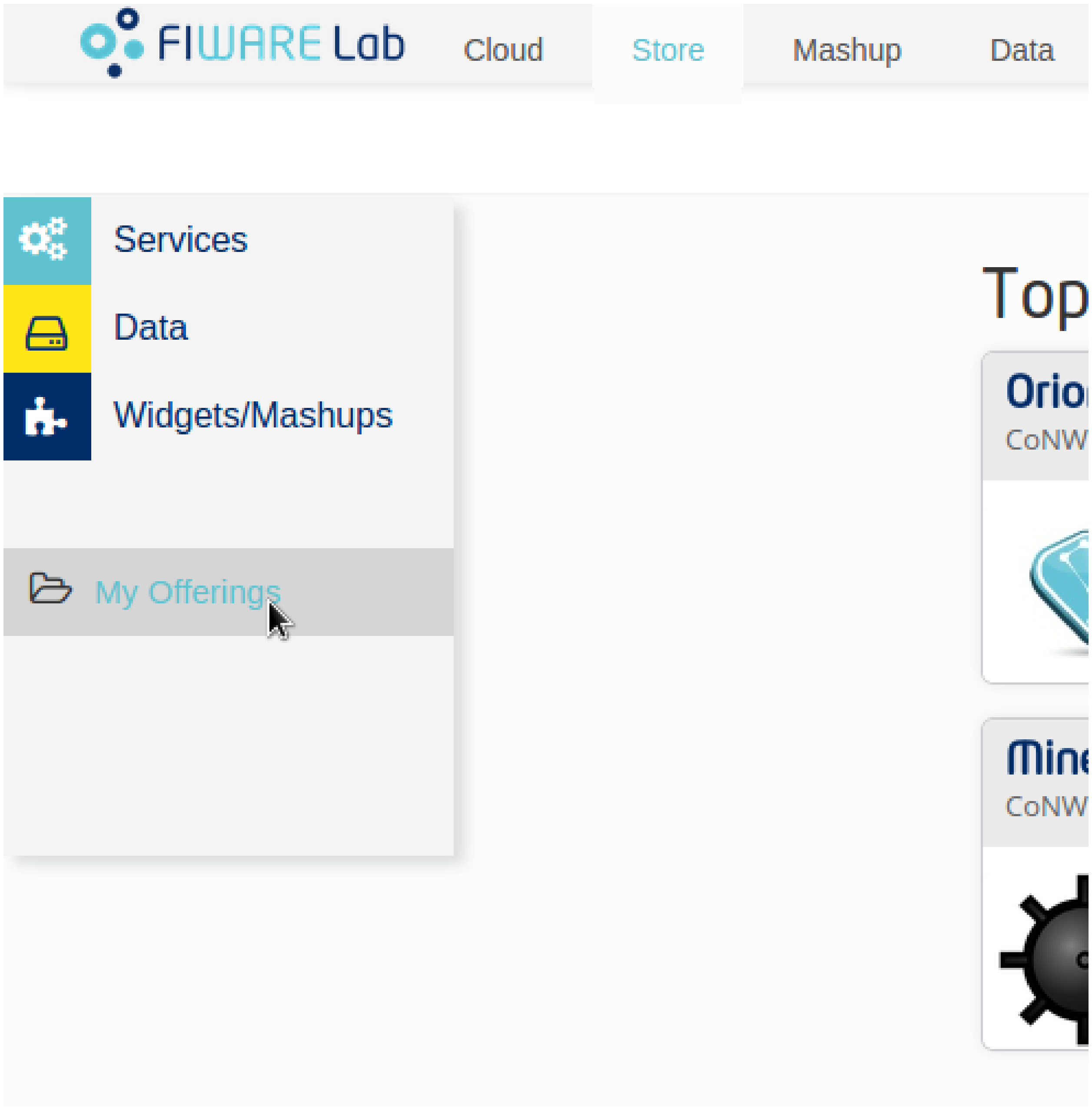
Service provider

This section explains how a service provider can create and monetize her service offerings using WStore GUI.

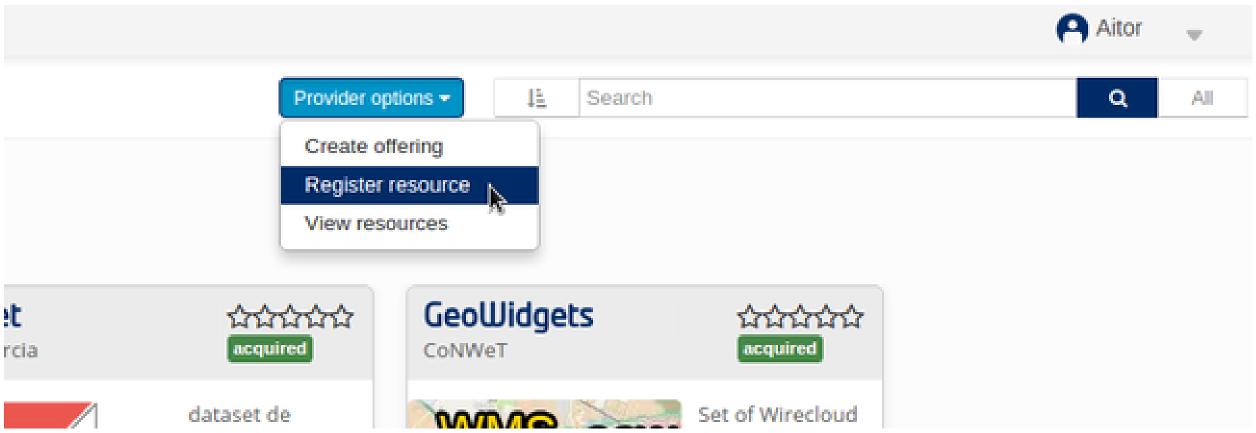
Registering a resource

Resources are used in the offerings in order to allow to include downloadable digital assets, such as applications, widgets, etc.

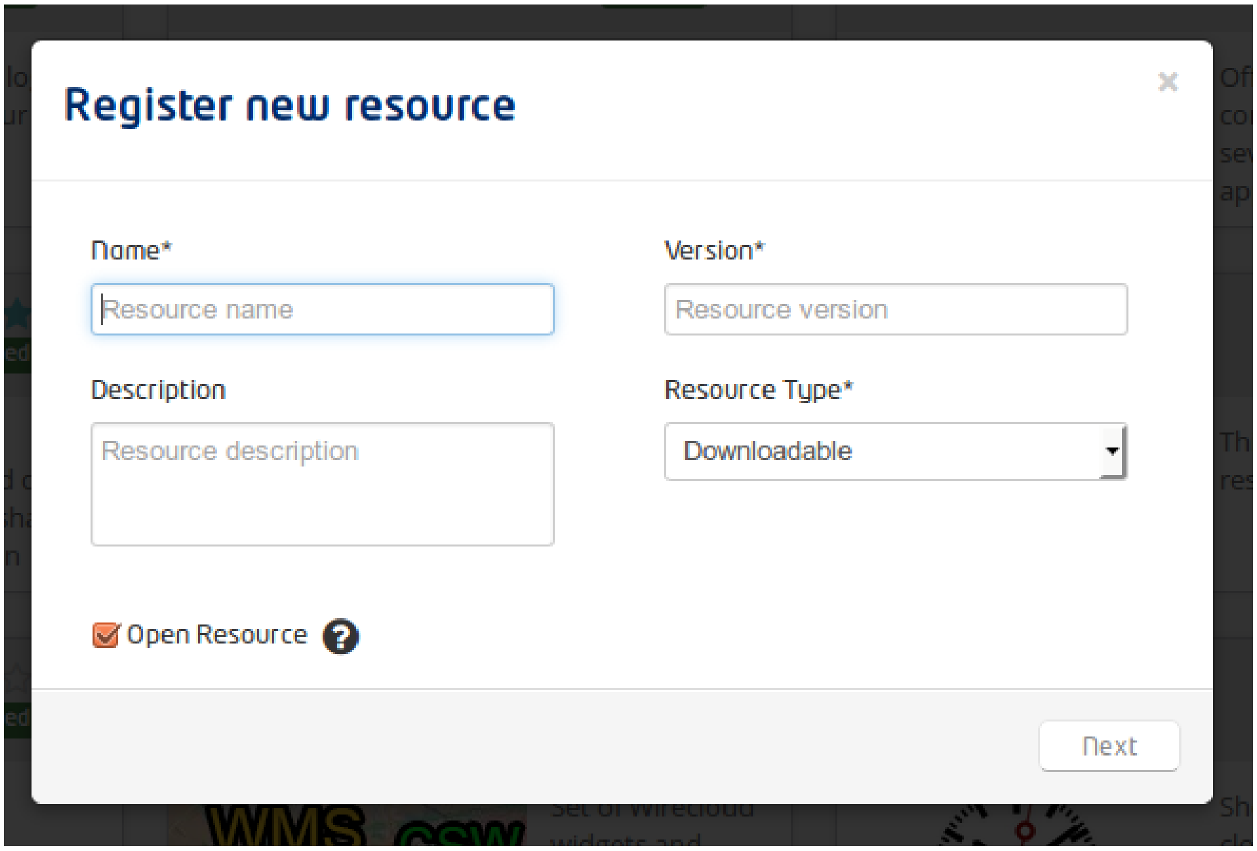
To register a resource, select the My Offerings view.



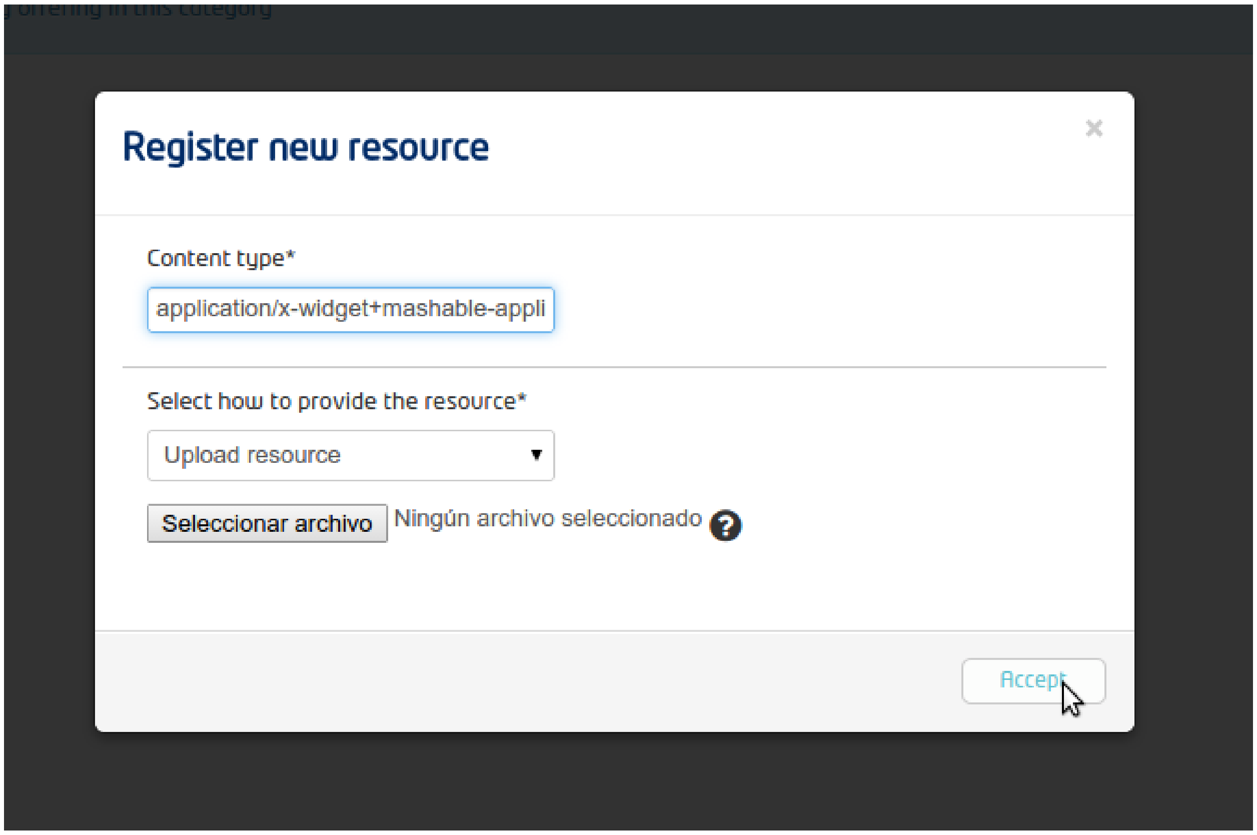
Select the options dropdown and choose Register resource



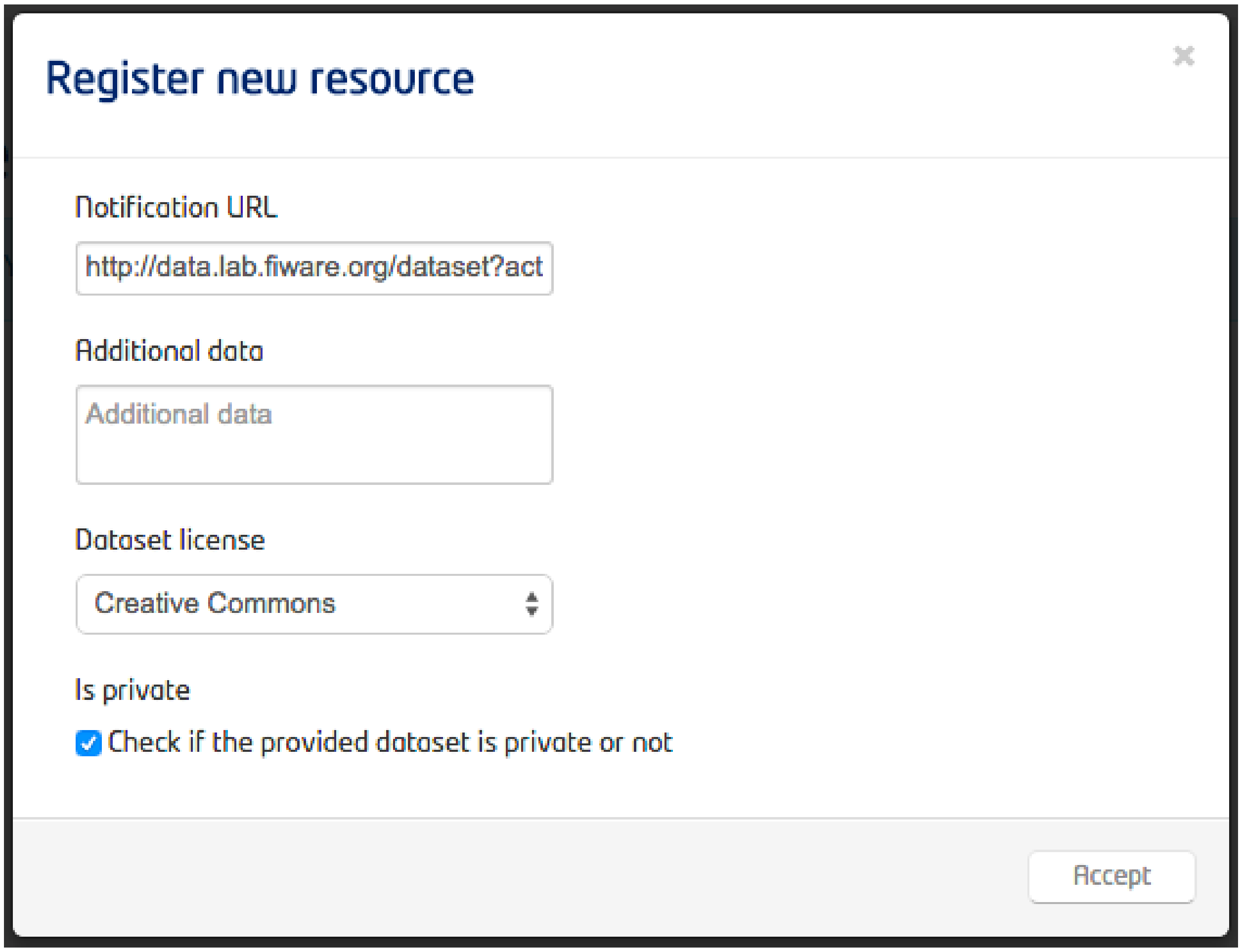
In the displayed modal, the first step is filling the resource name, the resource version and the resource description. Then, it is necessary to choose the resource type and to choose whether the resource is open or not. This flag is used to specify if the resource can contain an authorization mechanism that does not depend on WStore, for example an API resource that requires authenticated users.



The next step consists on providing a mime type for the resource and uploading the resource or providing a link from where the resource can be accessed, depending on the selected type of resource.


Register new resource
×
Content type*
application/x-widget+mashable-appli
Select how to provide the resource*
Upload resource
Seleccionar archivo
Ningún archivo seleccionado
Accept

Finally, depending on the type of resource a final form may appear for retrieving meta information.


Register new resource

Notification URL

http://data.lab.fiware.org/dataset?act

Additional data

Additional data

Dataset license

Creative Commons

Is private

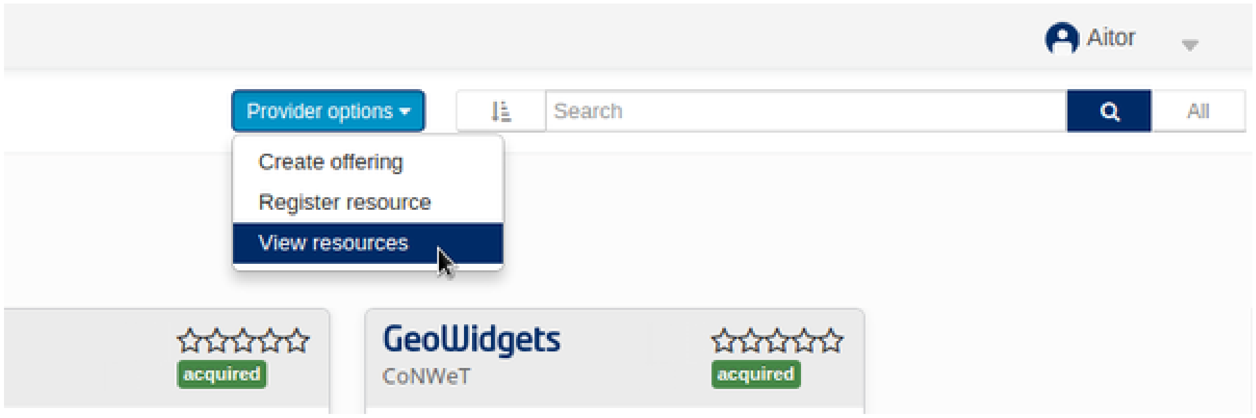
☑ Check if the provided dataset is private or not

Accept

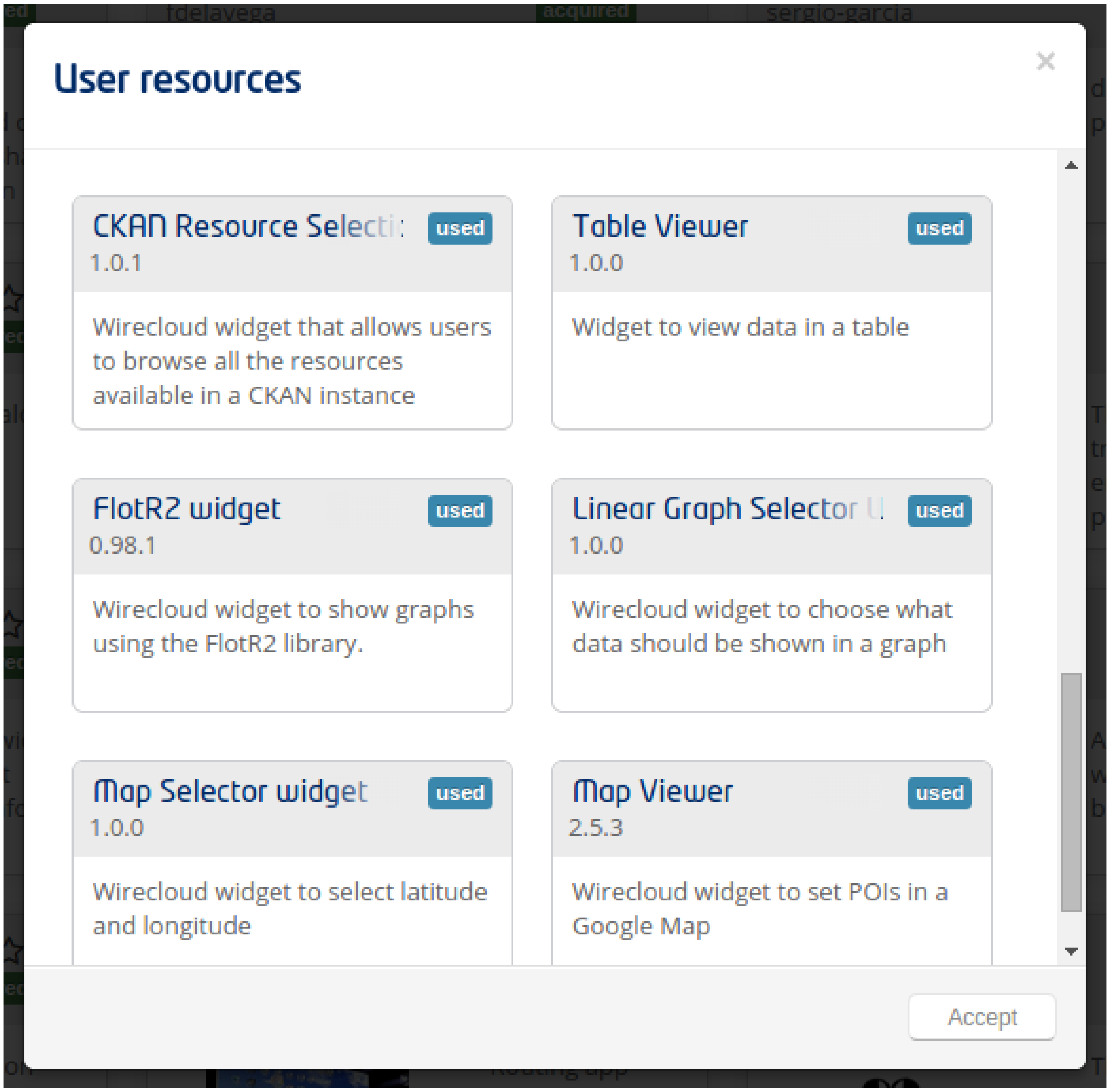
Note: The concrete types of resources available will depend of the concrete instance of WStore and the resource plugins installed on it. By default WStore has the Downloadable and API types.

Viewing resources

A service provider is able to view the resources which are already registered. To view the resources, select the catalogue view and choose View resources in the dropdown menu.



In the displayed modal, you can see the different resources you have registered.



User resources

CKAN Resource Selecti: used
1.0.1
Wirecloud widget that allows users to browse all the resources available in a CKAN instance

Table Viewer used
1.0.0
Widget to view data in a table

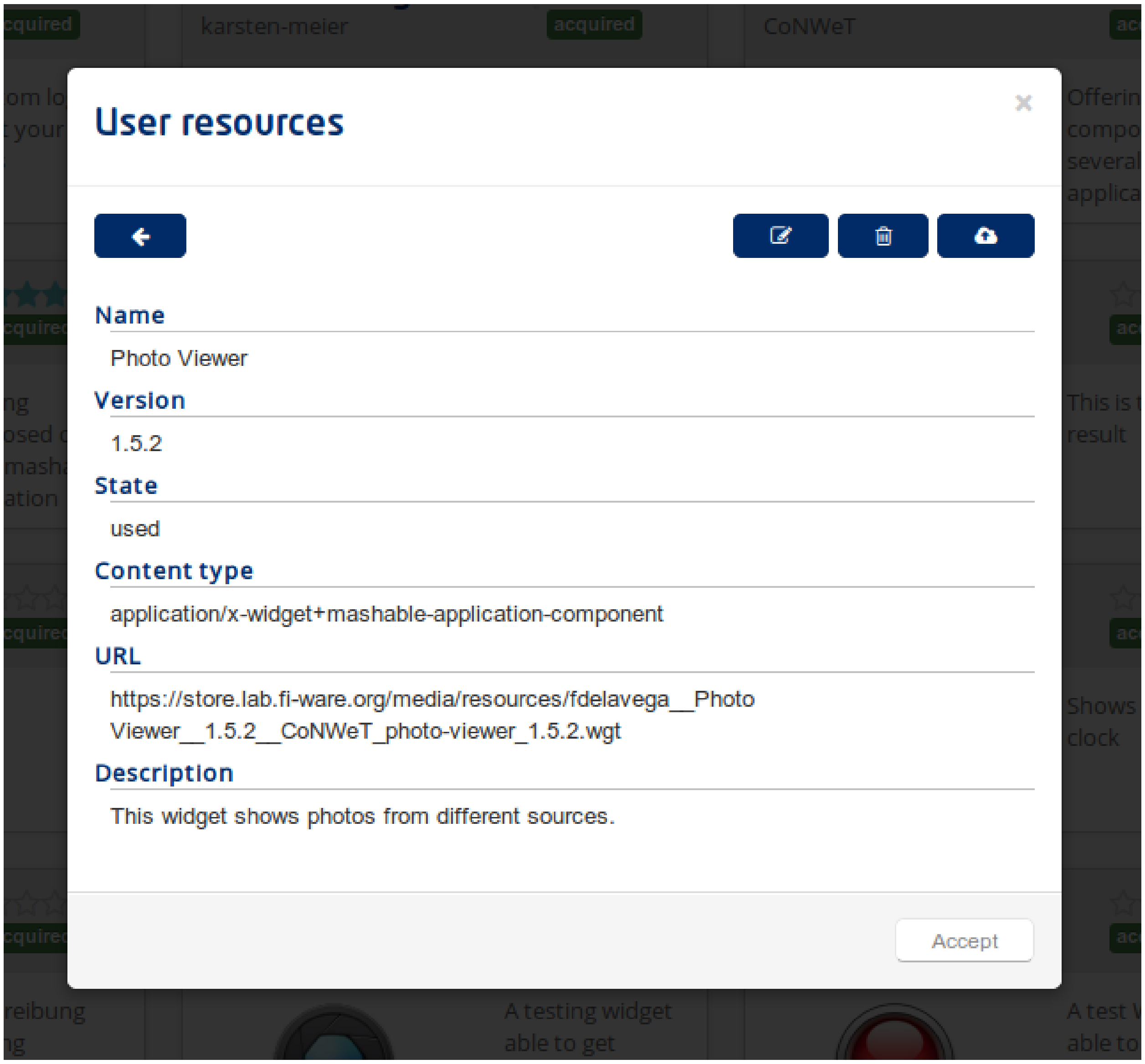
FlotR2 widget used
0.98.1
Wirecloud widget to show graphs using the FlotR2 library.

Linear Graph Selector U used
1.0.0
Wirecloud widget to choose what data should be shown in a graph

Map Selector widget used
1.0.0
Wirecloud widget to select latitude and longitude

Map Viewer used
2.5.3
Wirecloud widget to set POIs in a Google Map

It is possible to view the details of a registered resource clicking on it.



Name

Photo Viewer

Version

1.5.2

State

used

Content type

application/x-widget+mashable-application-component

URL

https://store.lab.fi-ware.org/media/resources/fdelavega__Photo Viewer__1.5.2__CoNWeT_photo-viewer_1.5.2.wgt

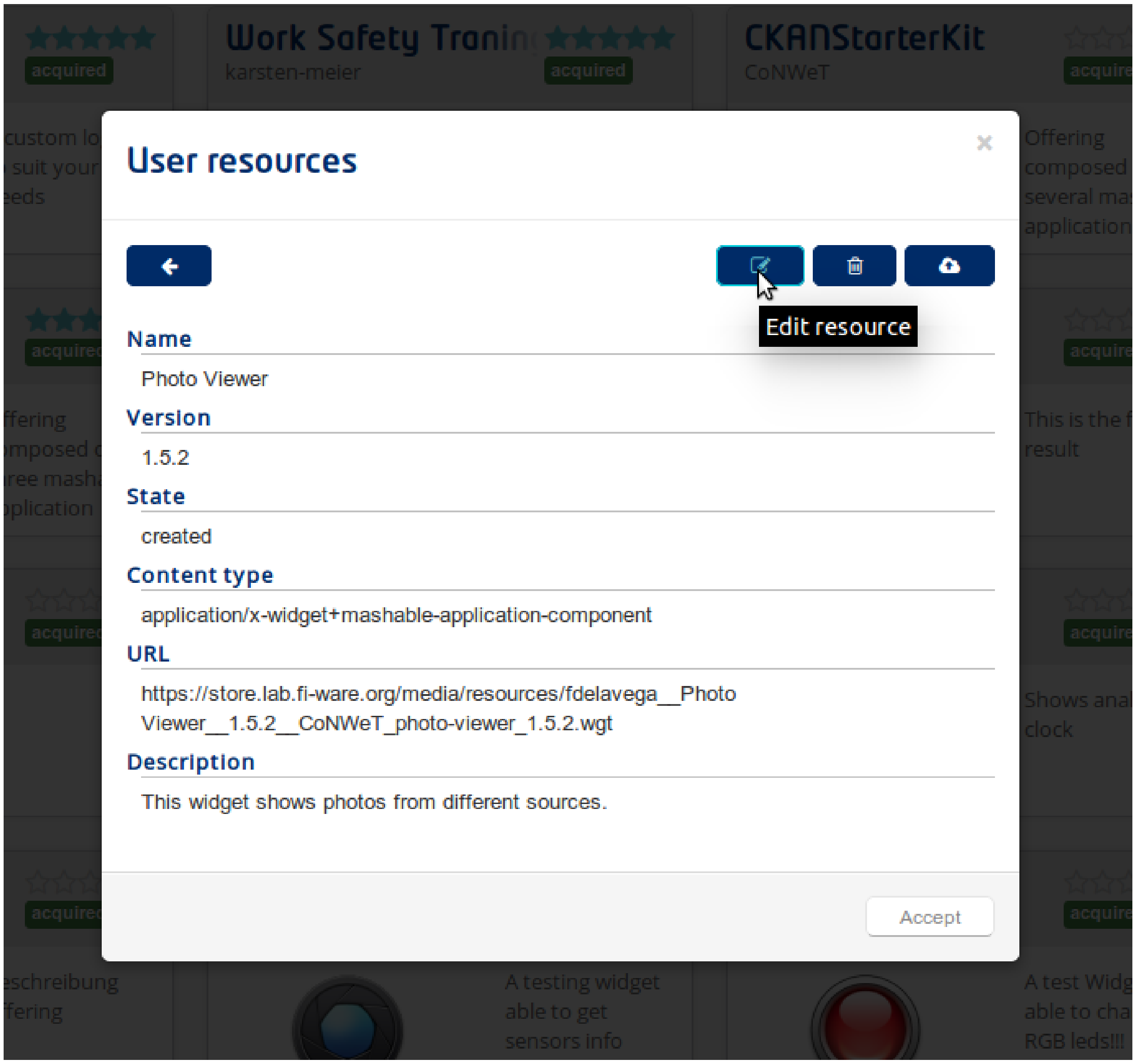
Description

This widget shows photos from different sources.

Editing a resource

Depending on the state of the resource (created or used) a service provider can edit the information of a resource.

To edit a resource, click on the edit button inside the resource details view.



User resources

Name
Photo Viewer

Version
1.5.2

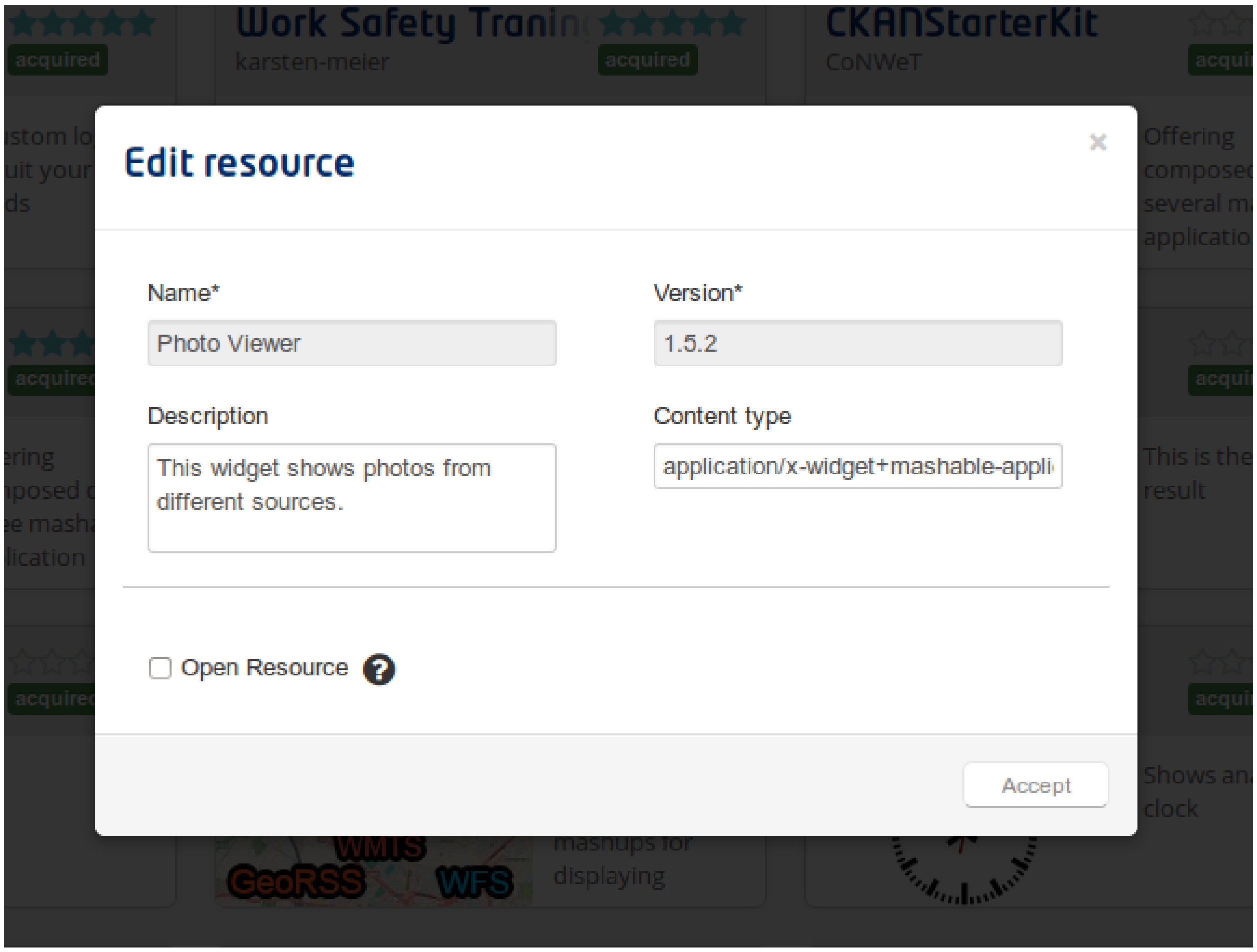
State
created

Content type
application/x-widget+mashable-application-component

URL
https://store.lab.fi-ware.org/media/resources/fdelavega__Photo Viewer__1.5.2__CoNWeT_photo-viewer_1.5.2.wgt

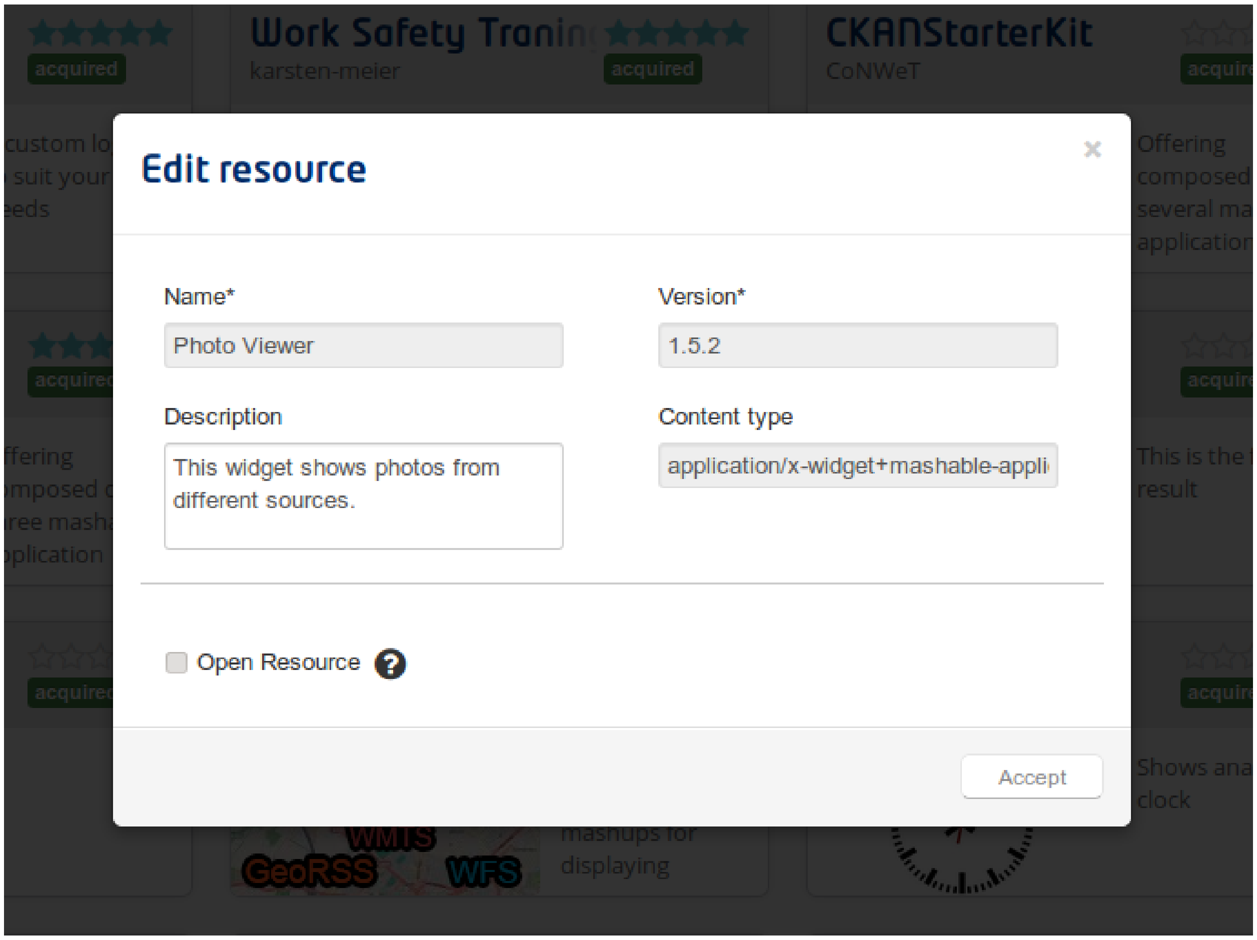
Description
This widget shows photos from different sources.

If the resource is in created state, that is, the resource has not been included in any offering, it is possible to modify the description, the content type and whether the resource is open or not.



Name*	Version*
Photo Viewer	1.5.2
Description	Content type
This widget shows photos from different sources.	application/x-widget+mashable-appli

If the resource is in used state, only the description can be modified.



Name*: Photo Viewer

Version*: 1.5.2

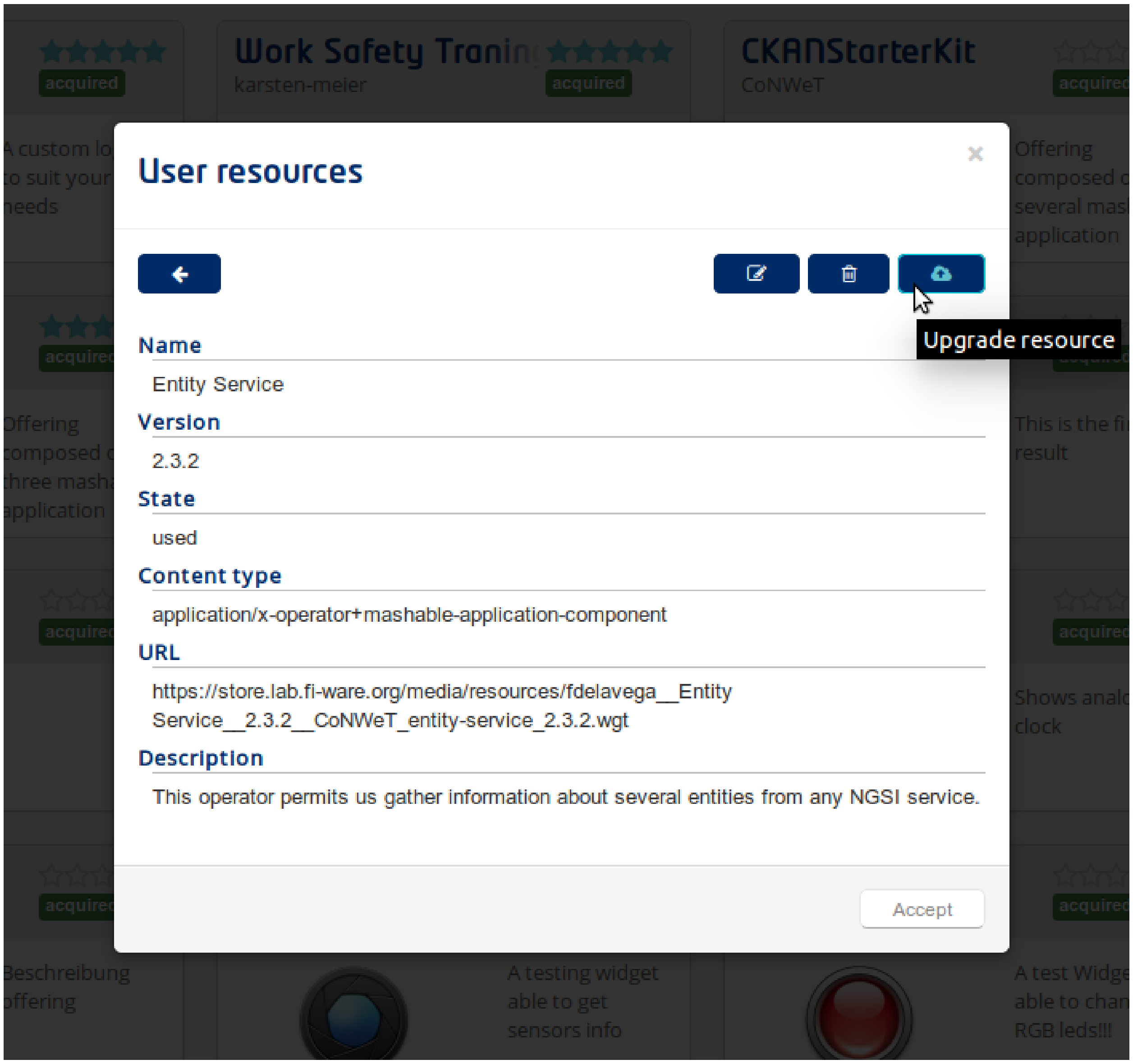
Description: This widget shows photos from different sources.

Content type: application/x-widget+mashable-appli

Upgrading a resource

It is possible to upgrade resources providing a new version of them. Note that when a new version of a resource is provided all the offerings that include this resource are automatically updated, so this is the mechanism that allows providers to perform automatic updates on their published apps, for example for including a patch.

To upgade a resource, click on the upgrade button in the resource details view.



Name

Entity Service

Version

2.3.2

State

used

Content type

application/x-operator+mashable-application-component

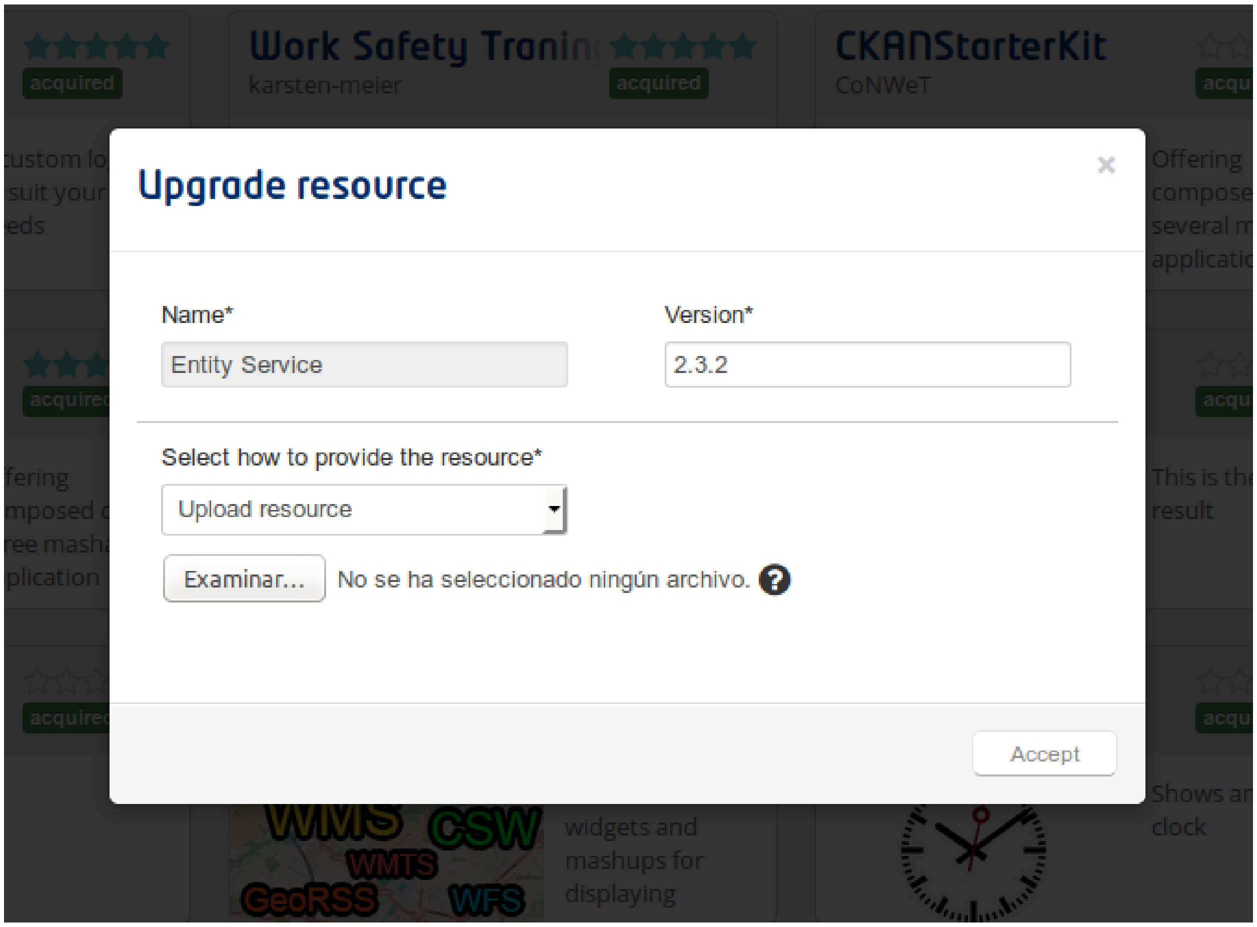
URL

https://store.lab.fi-ware.org/media/resources/fdelavega__Entity Service__2.3.2__CoNWeT_entity-service_2.3.2.wgt

Description

This operator permits us gather information about several entities from any NGSI service.

Then, it is necesary to provide the new version number and the new resource itself.


Upgrade resource

Name*: Entity Service

Version*: 2.3.2

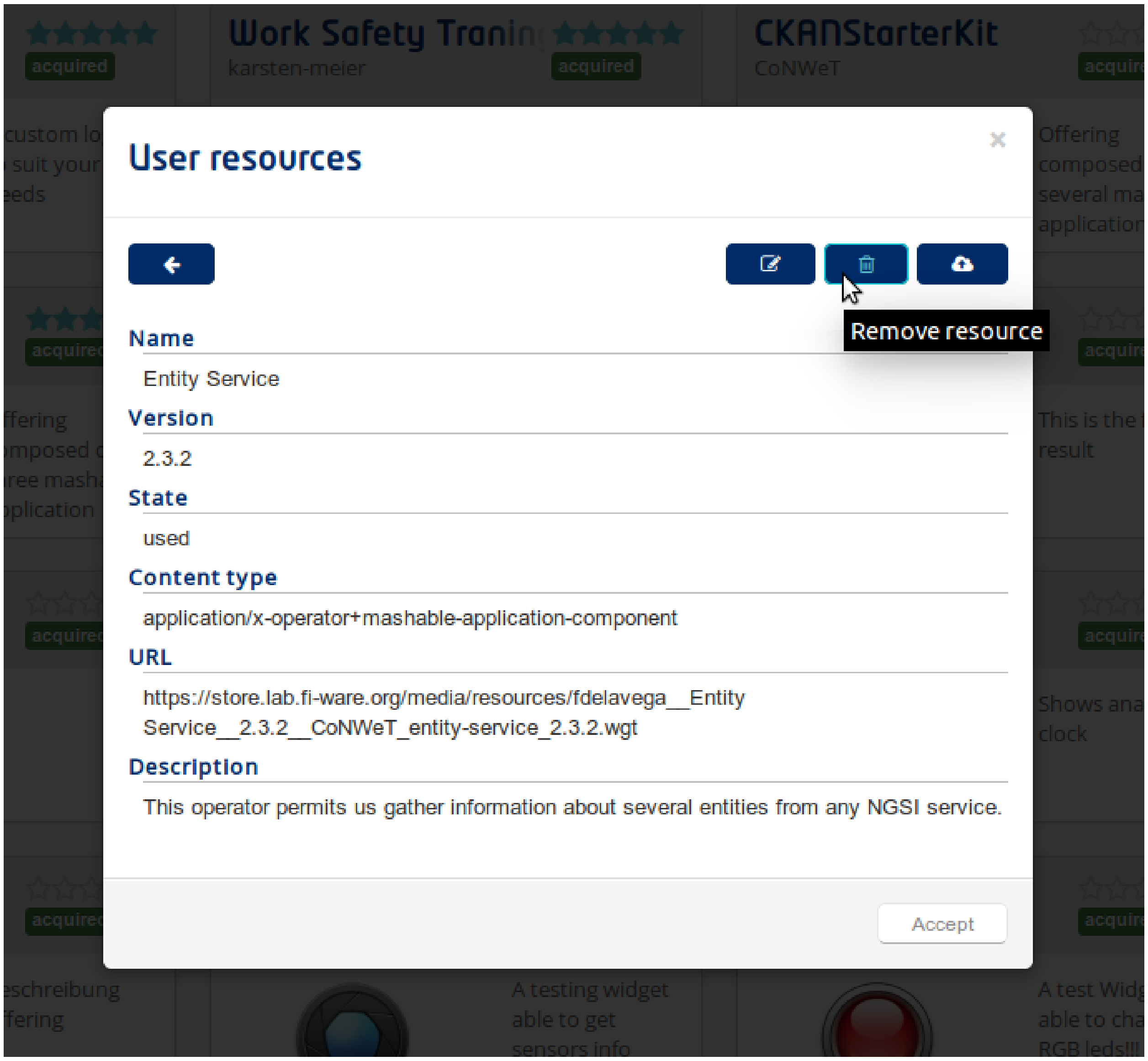
Select how to provide the resource*: Upload resource

Examinar... No se ha seleccionado ningún archivo.

Deleting a resource

A provider can delete their resources, depending on the state of the resource this action has different behaviours. If the resource is in created state, so it is not being used in any existing offering, the resource is completely removed from WStore. On the other hand, if the resource is in used state, it is set as deleted and cannot be included in more offerings.

To remove a resource, click on the remove button in the resource details view.



User resources

Name
Entity Service

Version
2.3.2

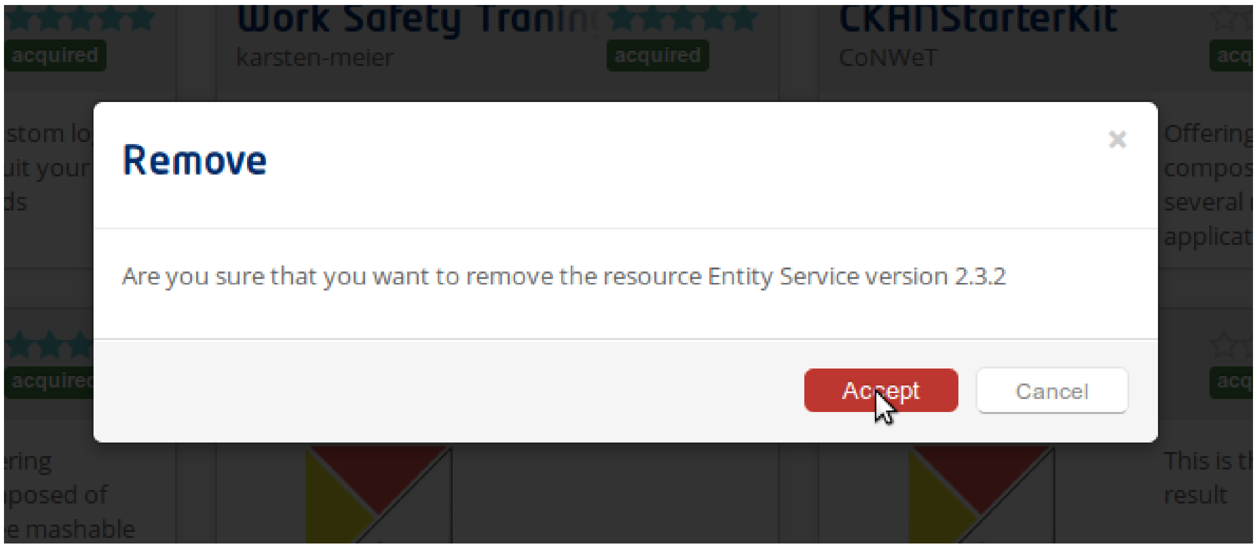
State
used

Content type
application/x-operator+mashable-application-component

URL
https://store.lab.fi-ware.org/media/resources/fdelavega__Entity Service__2.3.2__CoNWeT_entity-service_2.3.2.wgt

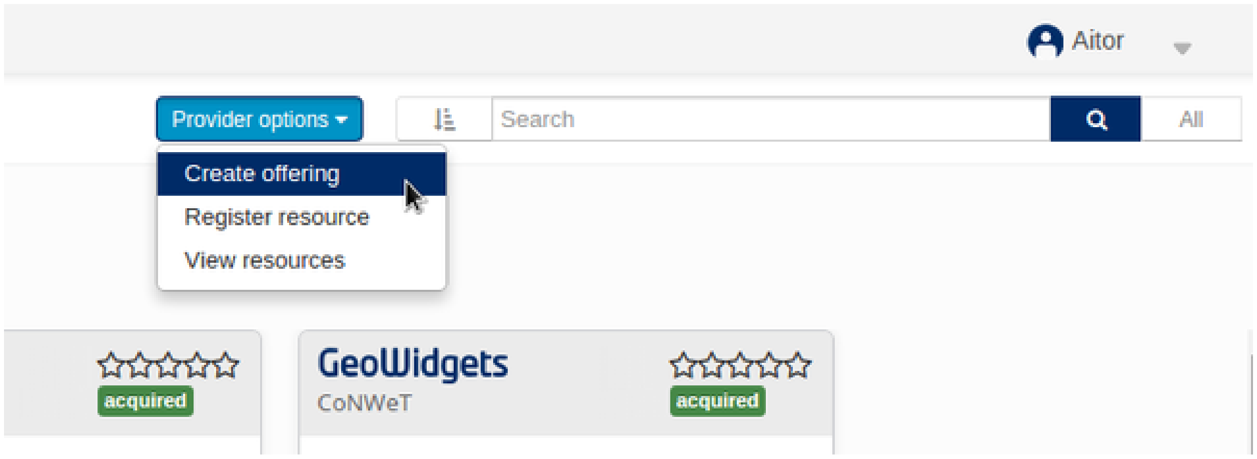
Description
This operator permits us gather information about several entities from any NGSI service.

Then, press Accept when asked.


Remove
Are you sure that you want to remove the resource Entity Service version 2.3.2
Accept
Cancel

Creating an offering

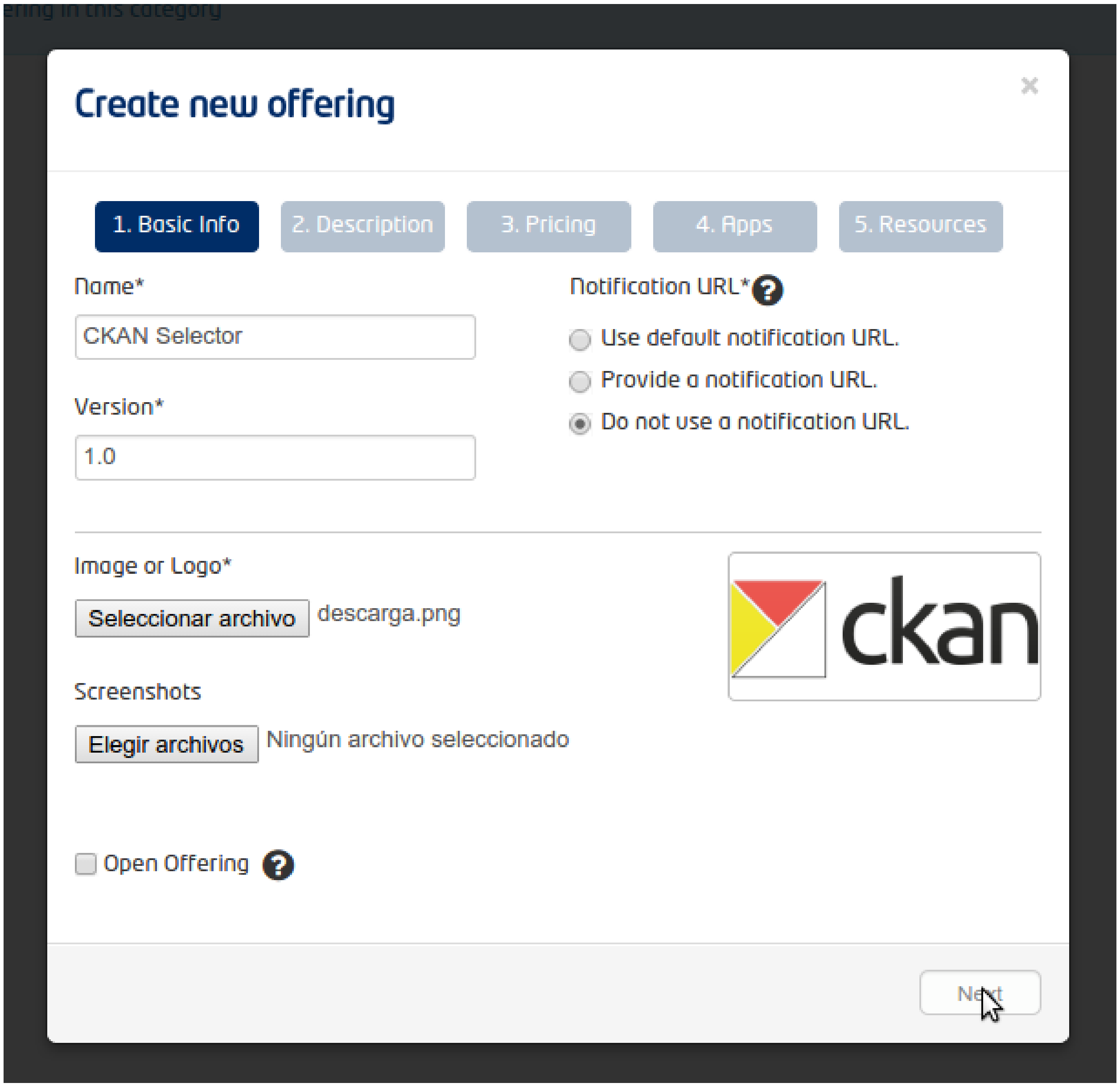
Offerings are the main entity managed by WStore and include all the relevant information such as the pricing model, legal conditions, interactions, service level agreement, etc. To create an offering, go to My Offerings. and choose Create offering from the Provider Options dropdown.



In the displayed modal, fill the name, version and description fields. Next, provide an image and an optional set of screenshots

The next step consits on selecting how to provide the notification URL. This field is used by WStore to notify the service provider when its offering has been purchased. There are three different options: (a) Provide a new notification URL for this offering. (b) Use the default notification URL of the provider that can be configured in the user profile configuration form. (c) Not using a notification URL for this offering.

Finally, choose whether the offering is open or not, that is, all the resources will be direcly accessible by WStore customers whitout the need of acquiring the offering. Note that if the offering is created as open, only open resources (resources that do not contain an external authetication mechanism that do not depend on WStore) can be bound to it.



Name*: CKAN Selector

Version*: 1.0

Notification URL*

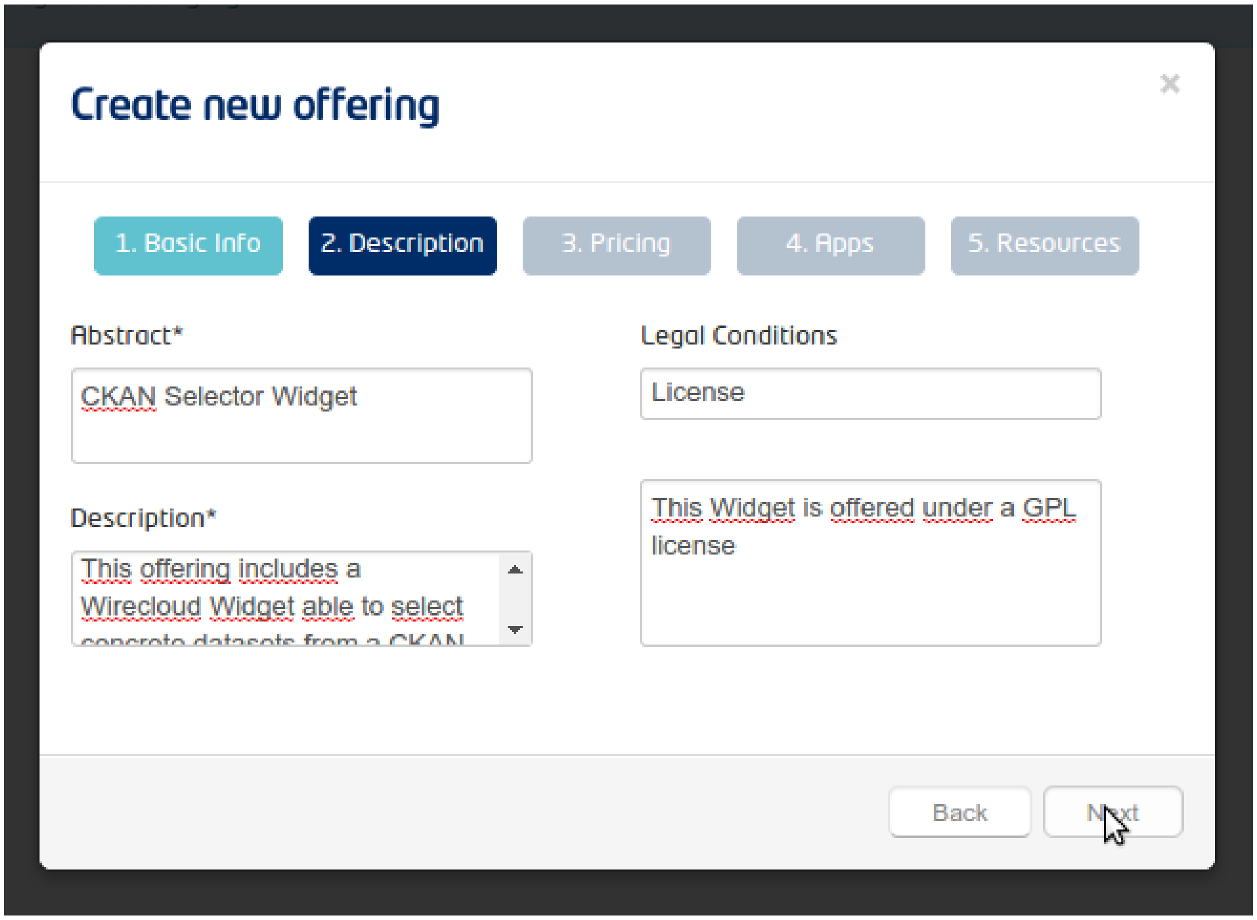
( ) Use default notification URL.
( ) Provide a notification URL.
[✓ Do not use a notification URL.]

Image or Logo*: descarga.png

Screenshots: Ningún archivo seleccionado

[ ] Open Offering

The next step consist on providing description information for the offering, including an abstract and a long description. Additionally, it is also possible to provide a legal description including the terms and conditions of the offering.



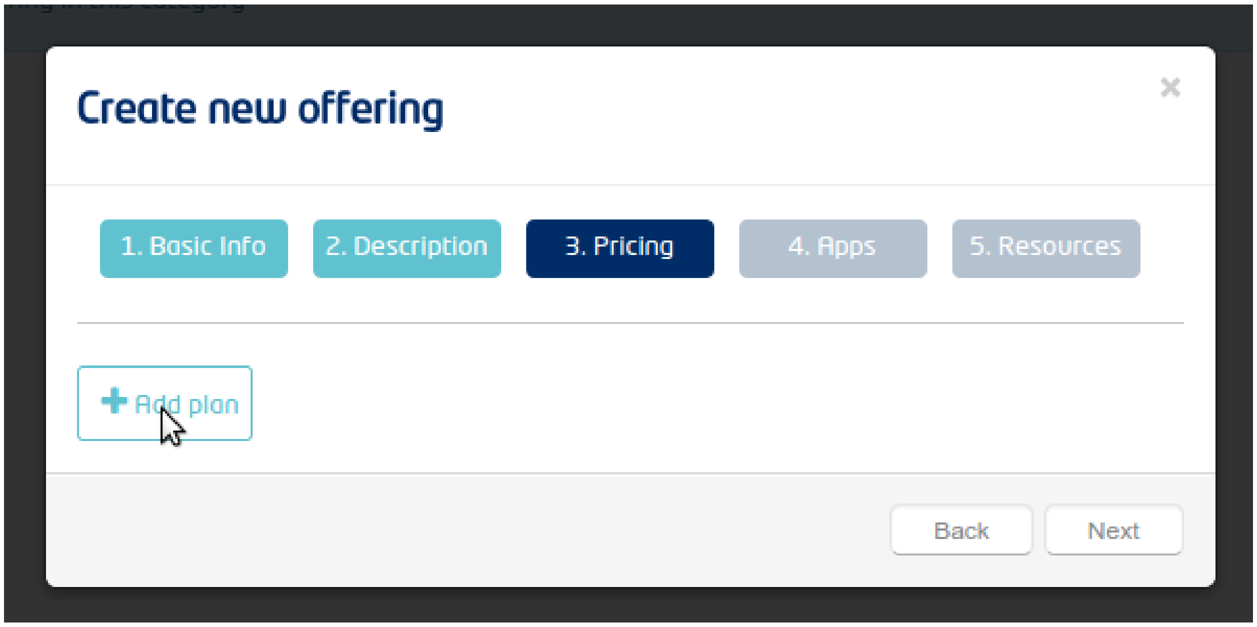
Abstract*: CKAN Selector Widget

Description*: This offering includes a Wirecloud Widget able to select concrete datasets from a CKAN

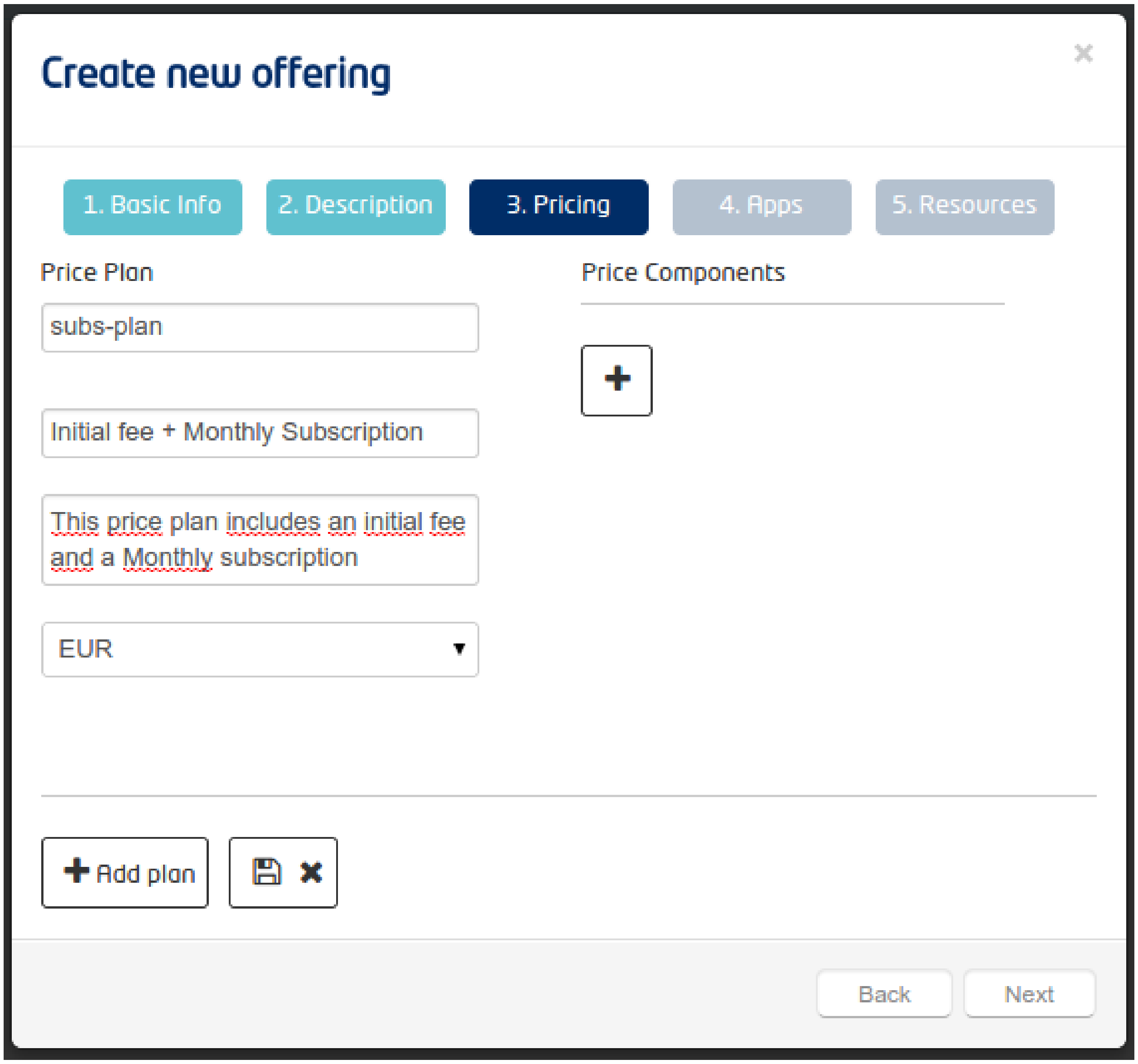
Legal Conditions: License

This Widget is offered under a GPL license

Once offering description has been provided, the next step is providing pricing information. The pricing models in WStore are based different price plans that can be chosen by the customers when acquiring the offering. To create a price plan click on *Add Plan*



Then, it is required to include a label that identifies the price plan, a display name, and a description. Additionally, it is needed to select the currency of the price plan.



Create new offering

Price Plan

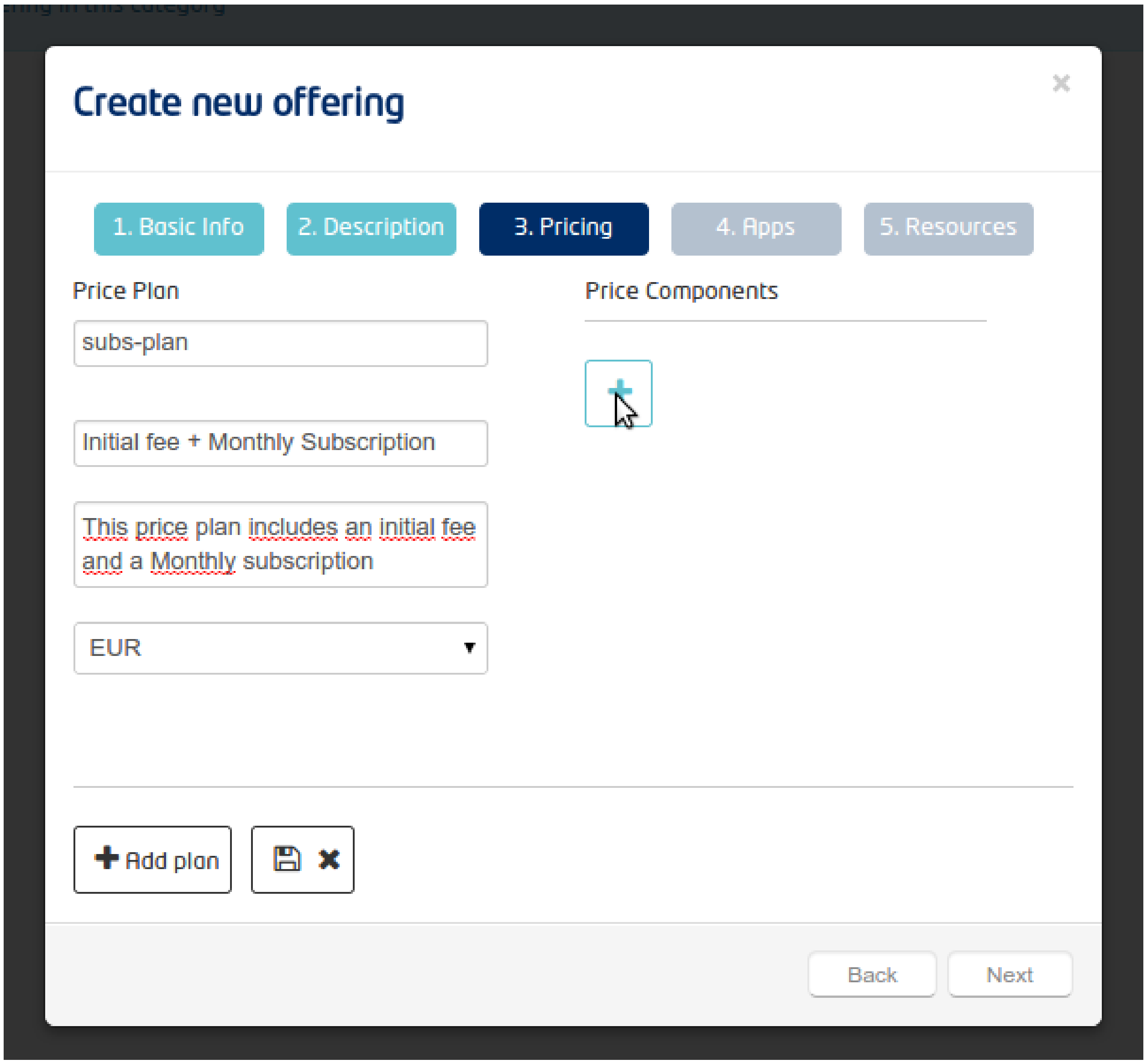
subs-plan

Initial fee + Monthly Subscription

This price plan includes an initial fee and a Monthly subscription

EUR

For including the pricing information, it is needed to create price components. To create a price component click on the *add* button in the price components section.



Price Plan

subs-plan

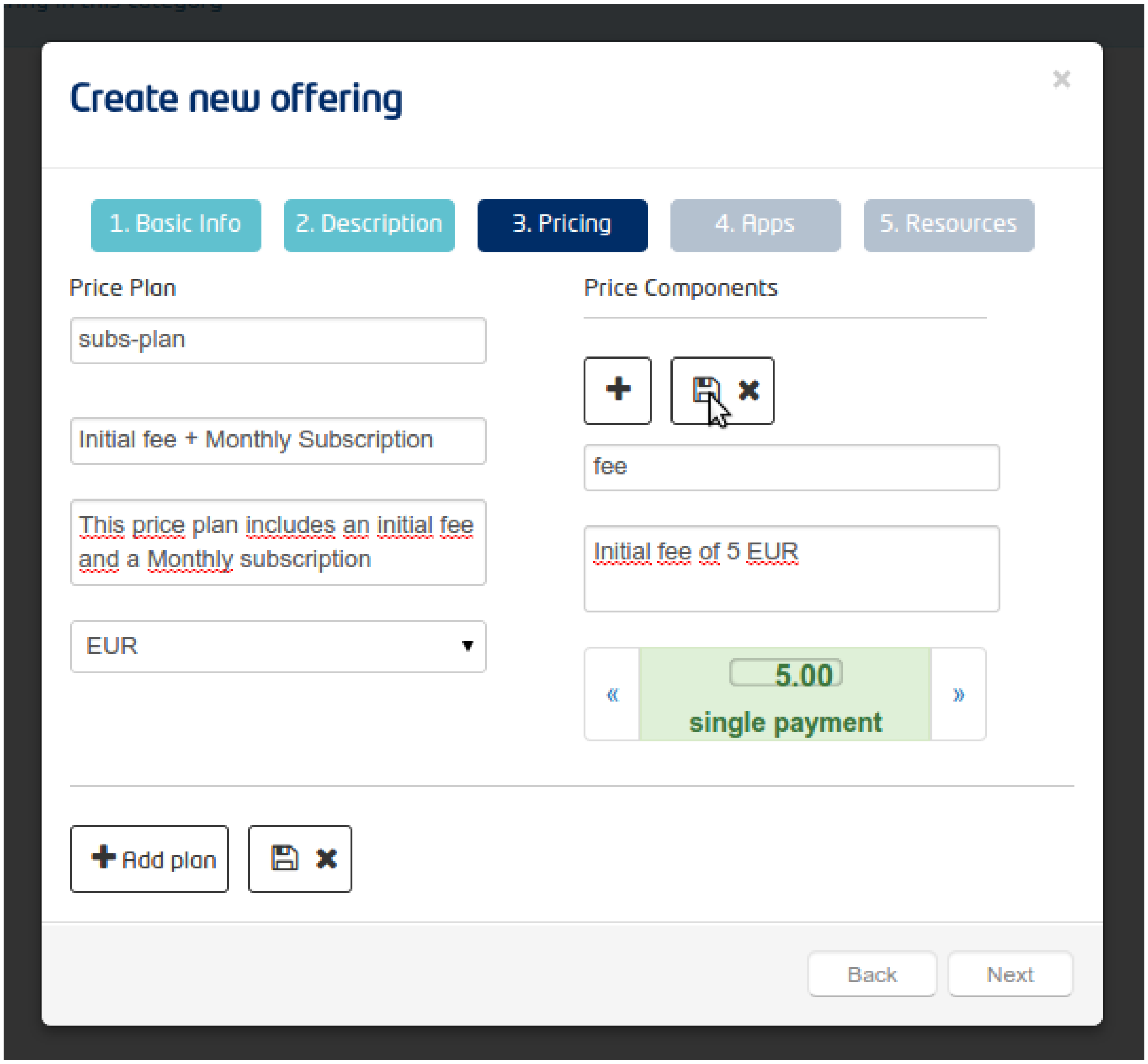
Initial fee + Monthly Subscription

This price plan includes an initial fee and a Monthly subscription

EUR

Price Components

Next, provide a label for identifying the component and a description. Next, provide the value of the component chosing the unit from the ones available. Finally, save the price component clickin on the *save* button.



Price Plan

subs-plan

Initial fee + Monthly Subscription

This price plan includes an initial fee and a Monthly subscription

EUR

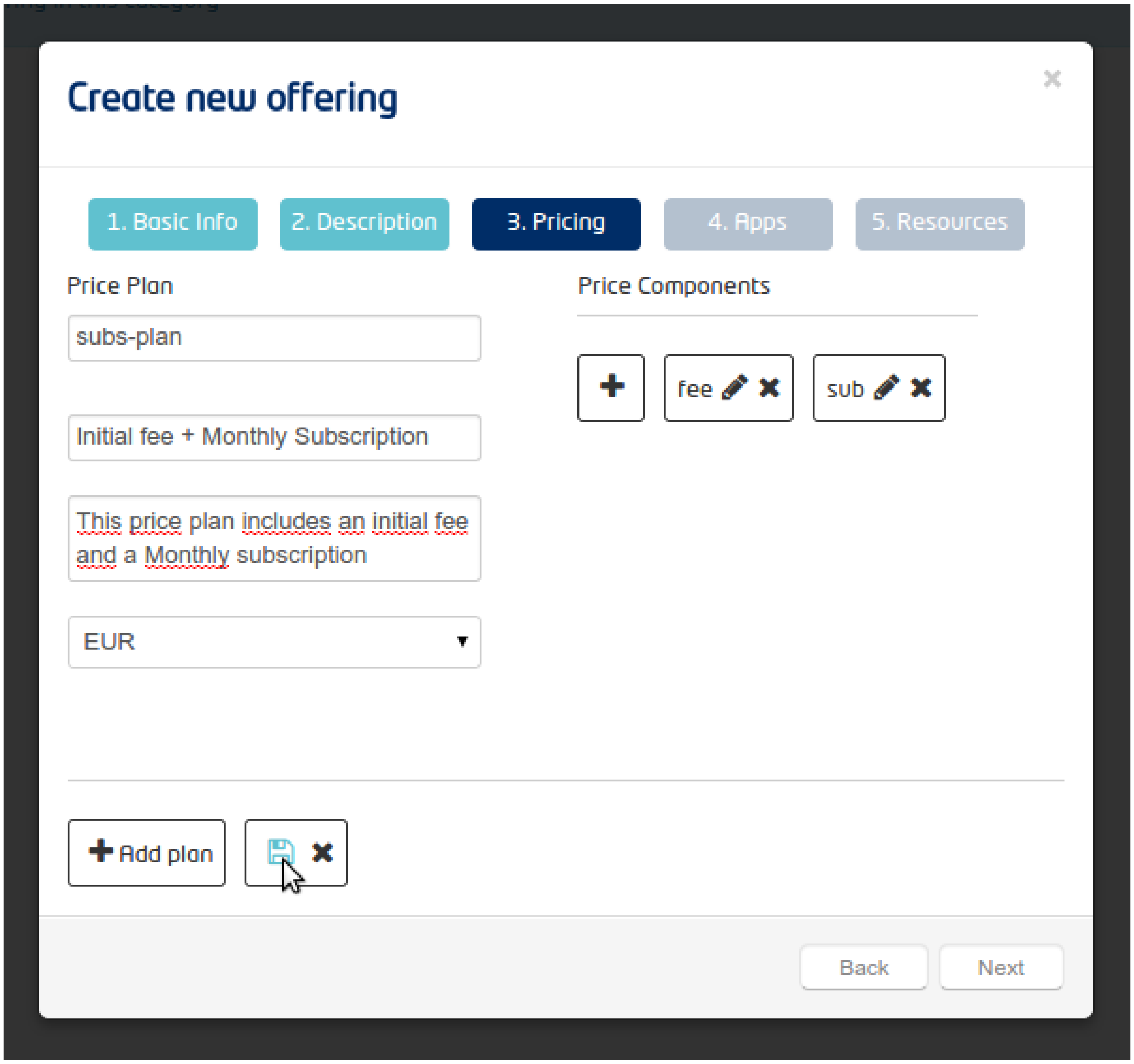
Price Components

fee

Initial fee of 5 EUR

5.00
single payment

The last step for the creation of a price plan is saving it by clicking on the *save* button.



subs-plan

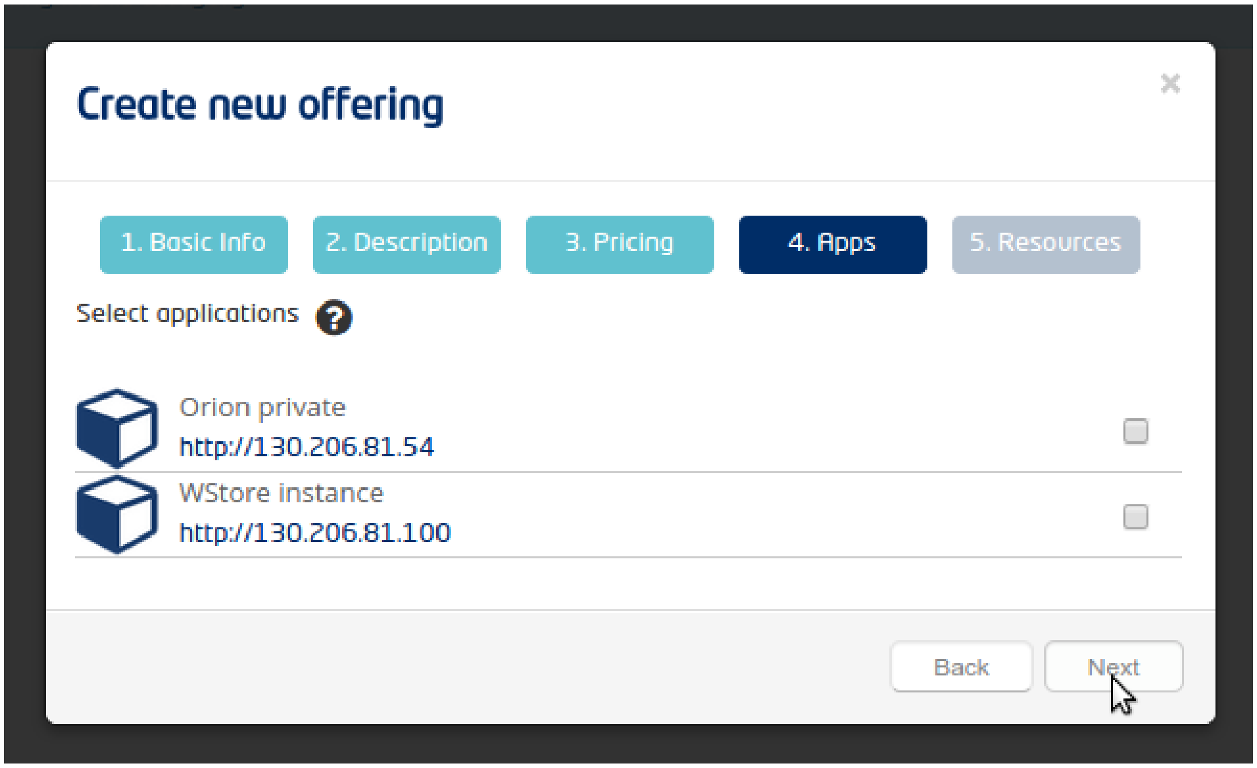
Initial fee + Monthly Subscription

This price plan includes an initial fee and a Monthly subscription

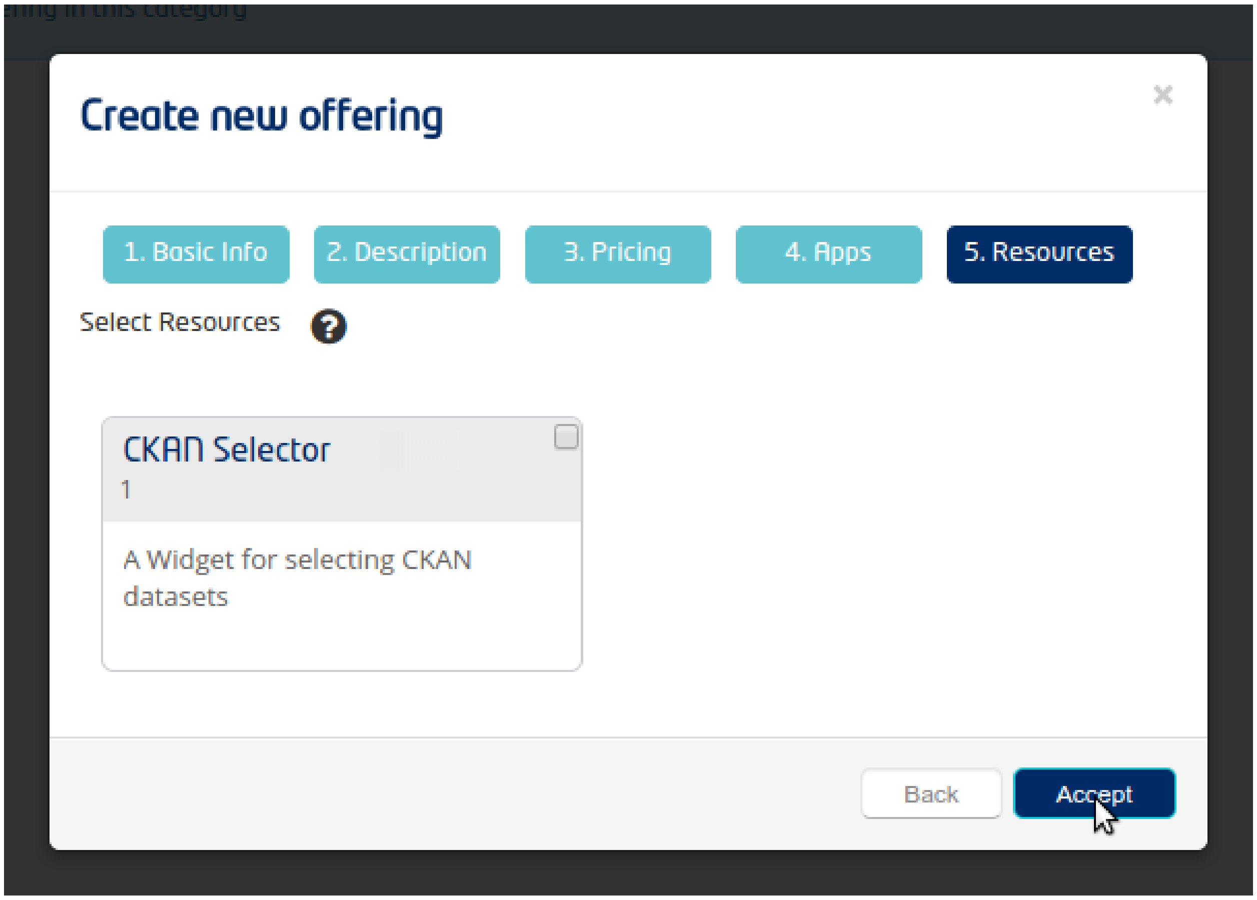
EUR

Note: For creating a free offering you can just skip this step.

Once the pricing info has been provided, the next step consist of including Applications. This applications need to be uderstood as OAuth2 Applications and are those registered in the Identity Manger by the provider. Including Applications in an offering allows to grant real access to the related services via OAuth2 to the customers that acquire the offering.

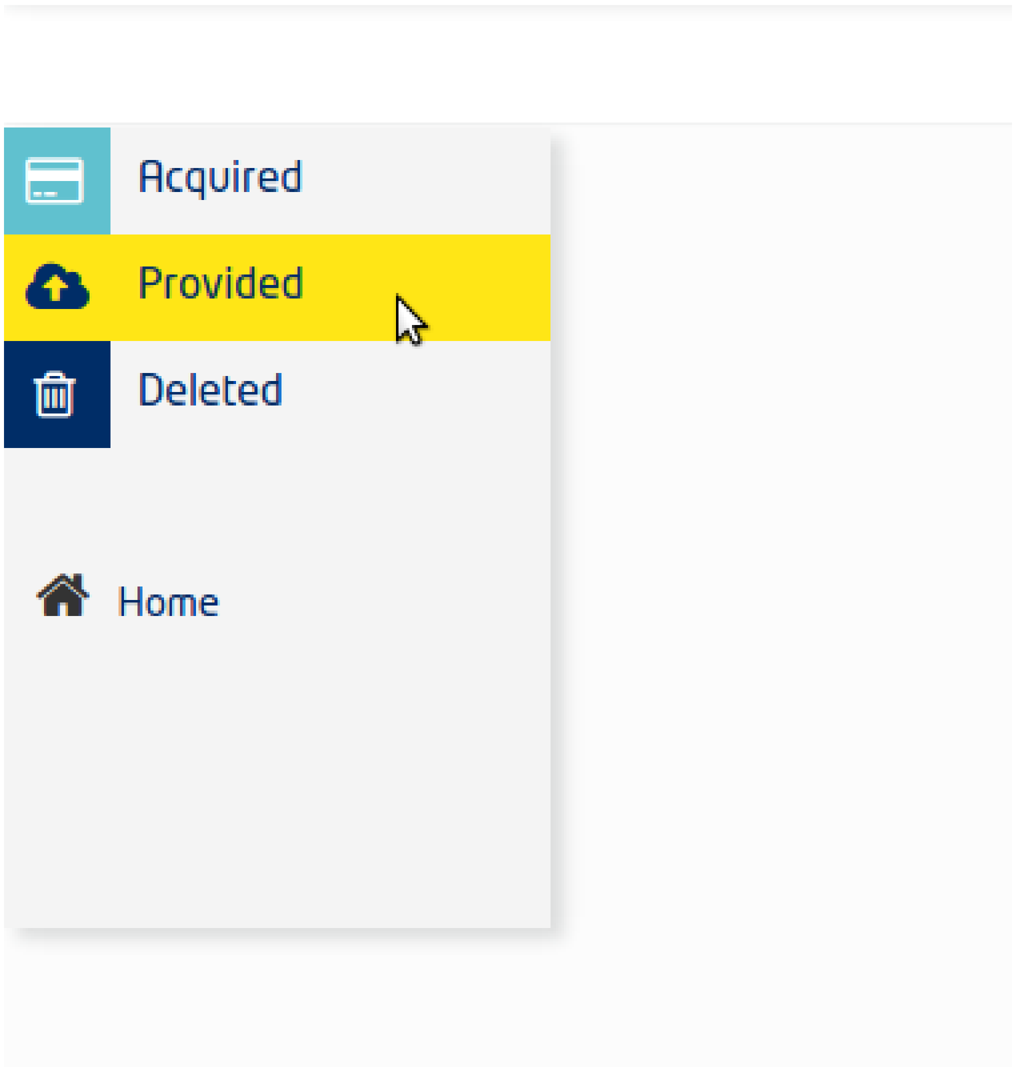


The final step consist of selecting resources previously registered by the provider.

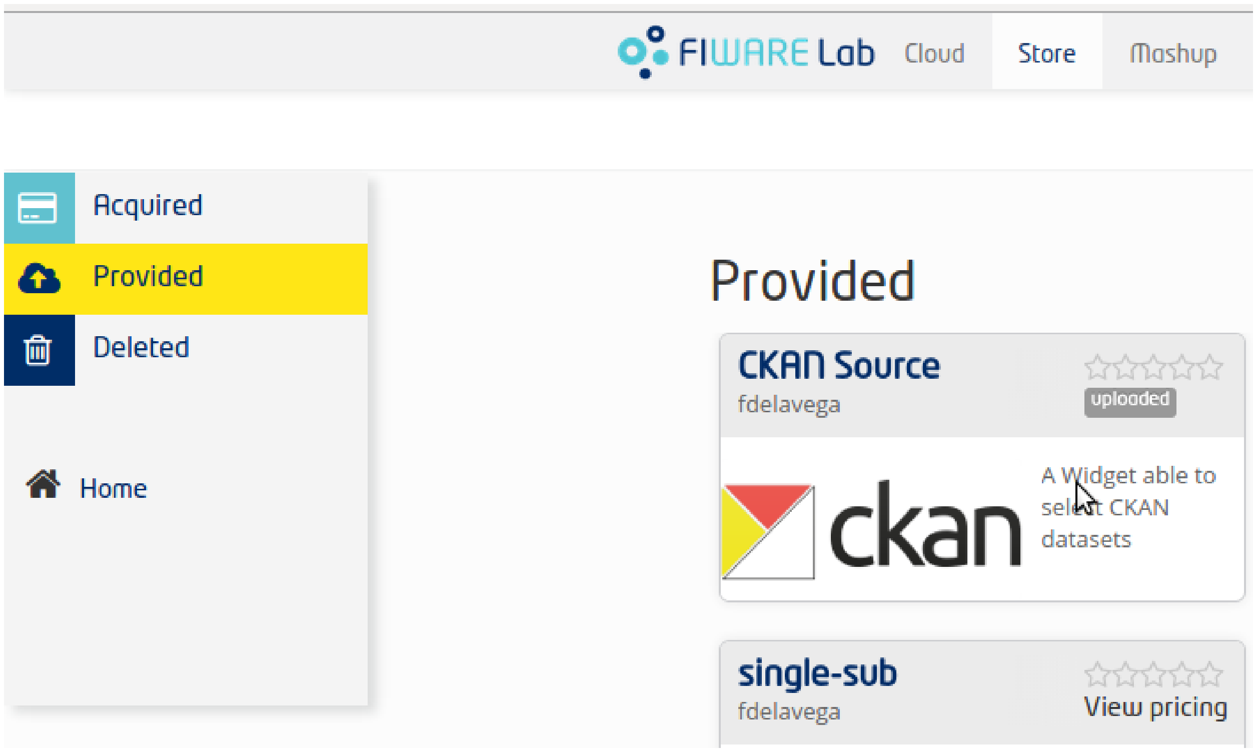


Updating an offering

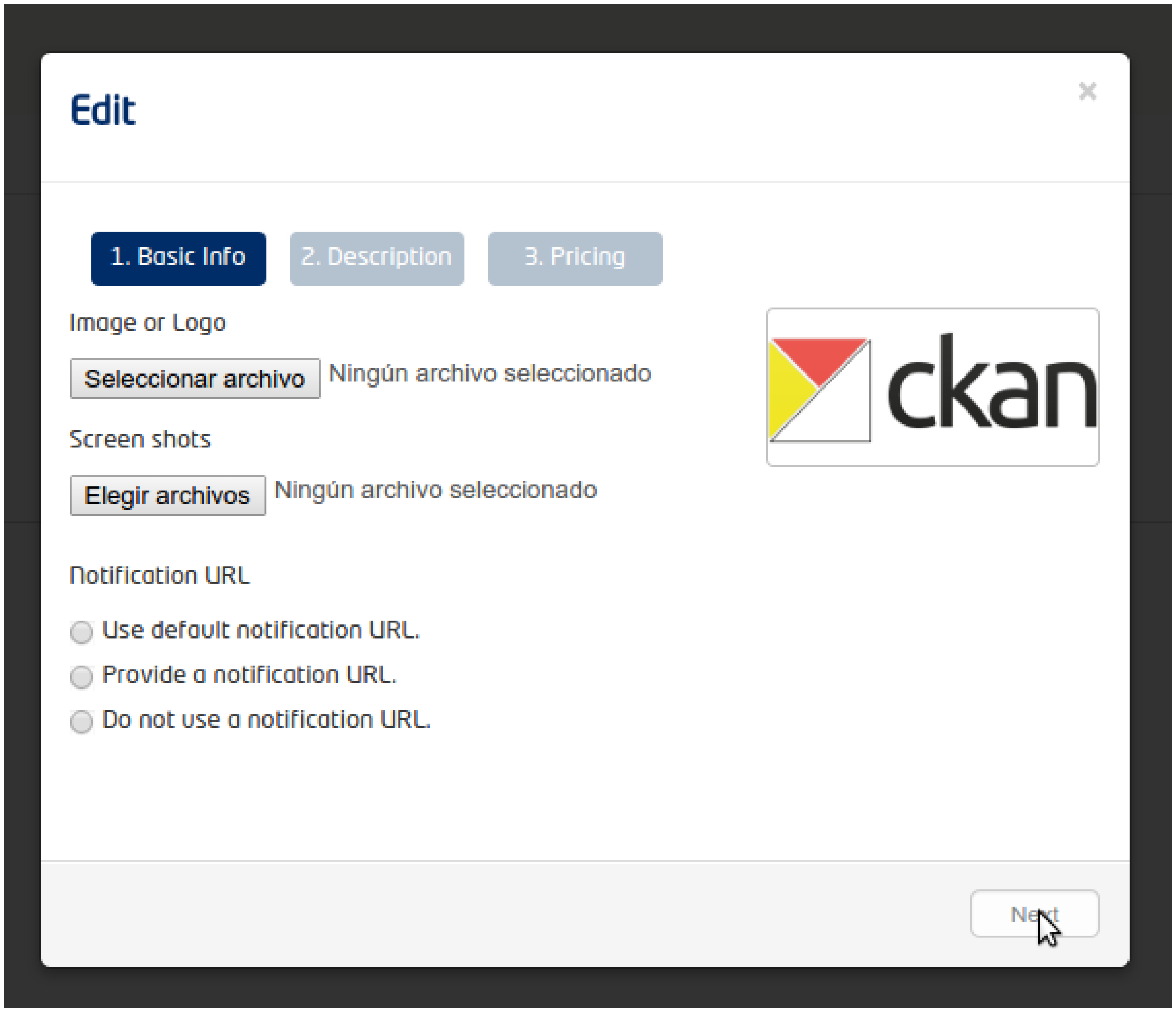
To update an offering, go to My Offerings view and the Provided section. This tab contains the offerings provided by the service provider.



Select the offering to be updated. Note that only offerings with uploaded state (Offerings that have not been published yet) can be updated.



In the advanced operations, select Edit.


Edit
×
1. Basic Info
2. Description
3. Pricing
Image or Logo
Seleccionar archivo
Ningún archivo seleccionado
Screen shots
Elegir archivos
Ningún archivo seleccionado
ckan
Notification URL
Use default notification URL.
Provide a notification URL.
Do not use a notification URL.
Next

In the displayed modal, it possible to provide some screenshots or a new logo. Additionally, it allows to modify the notification URL of the offering.

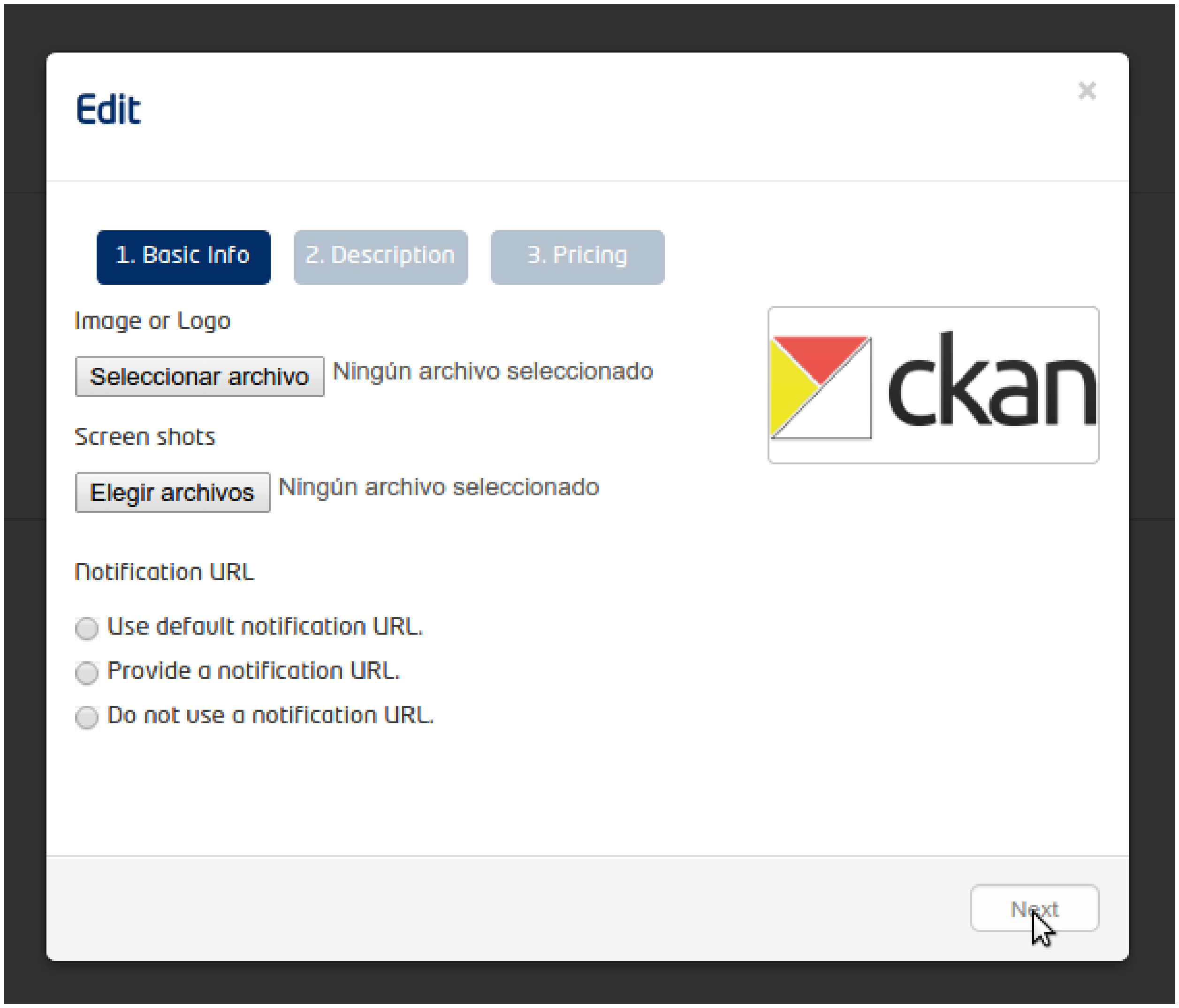


Image or Logo

Ningún archivo seleccionado

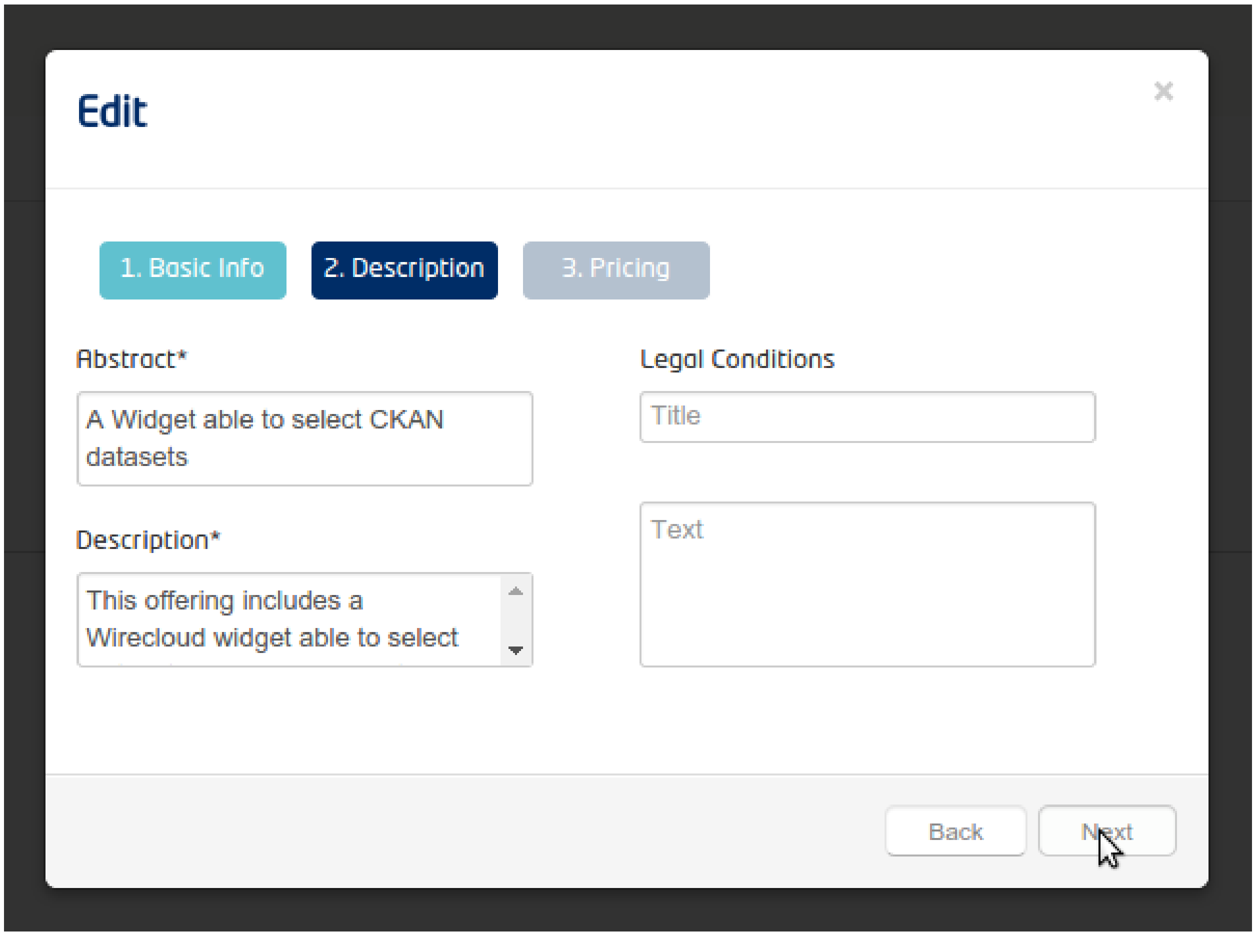
Screen shots

Ningún archivo seleccionado

Notification URL

- Use default notification URL.
- Provide a notification URL.
- Do not use a notification URL.

The next step, allows to modify the description information of the offering.



Abstract*

A Widget able to select CKAN datasets

Description*

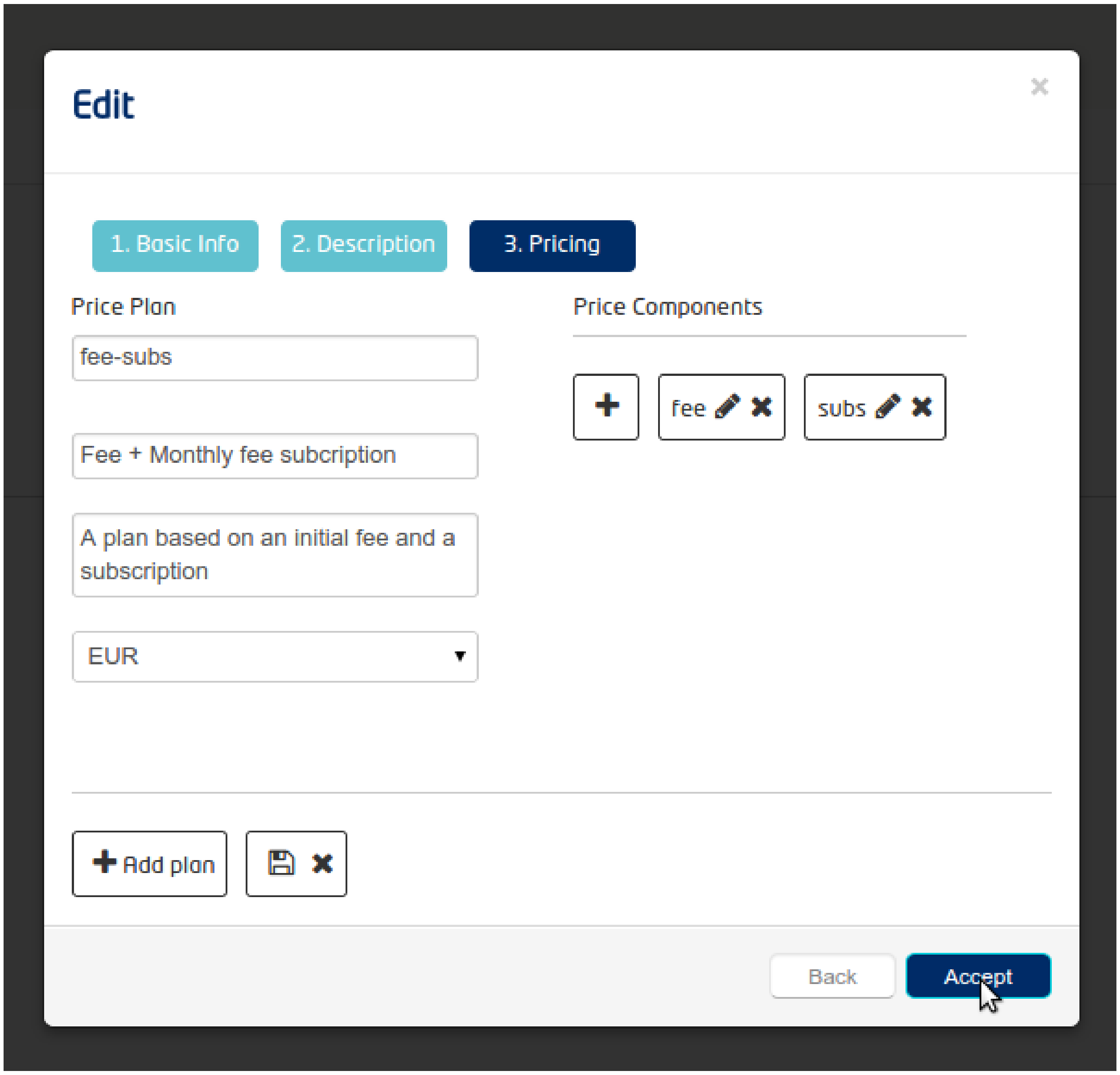
This offering includes a Wirecloud widget able to select

Legal Conditions

Title

Text

Finally, the last step allows to modify the pricing model of the offering.



fee-subs

Fee + Monthly fee subcription

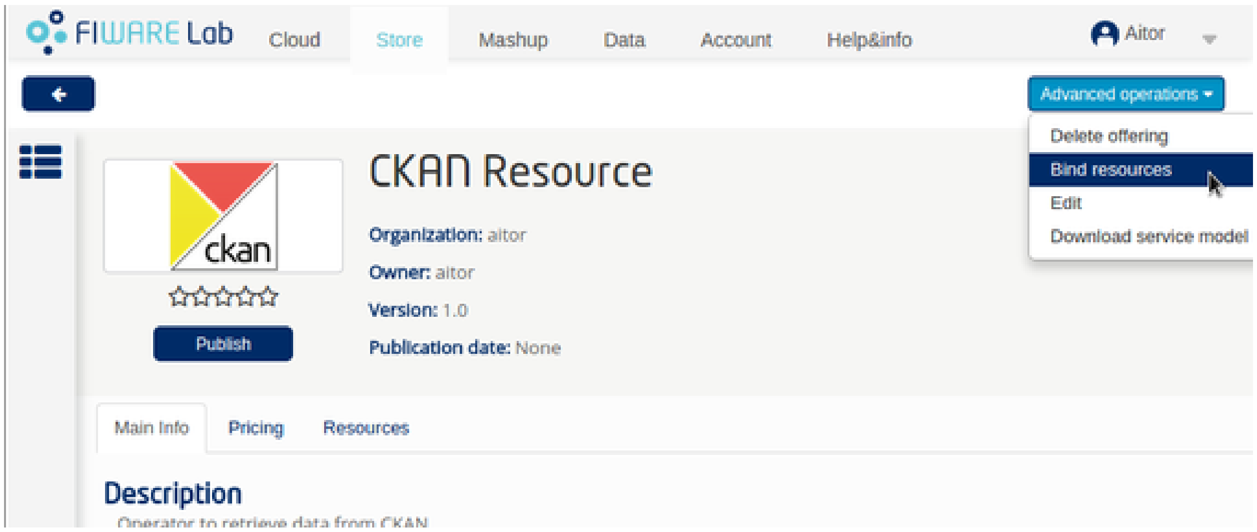
A plan based on an initial fee and a subscription

EUR

Binding resources

Once an offering has been created it is still possible to manage the included resources. To bind resources, select the My Offerings view and the Provided section. Then select the offering to be bound. Note that only offerings with uploaded state can be updated.

Select the “Bind resources” option.



CKAN Resource

Organization: aitor

Owner: aitor

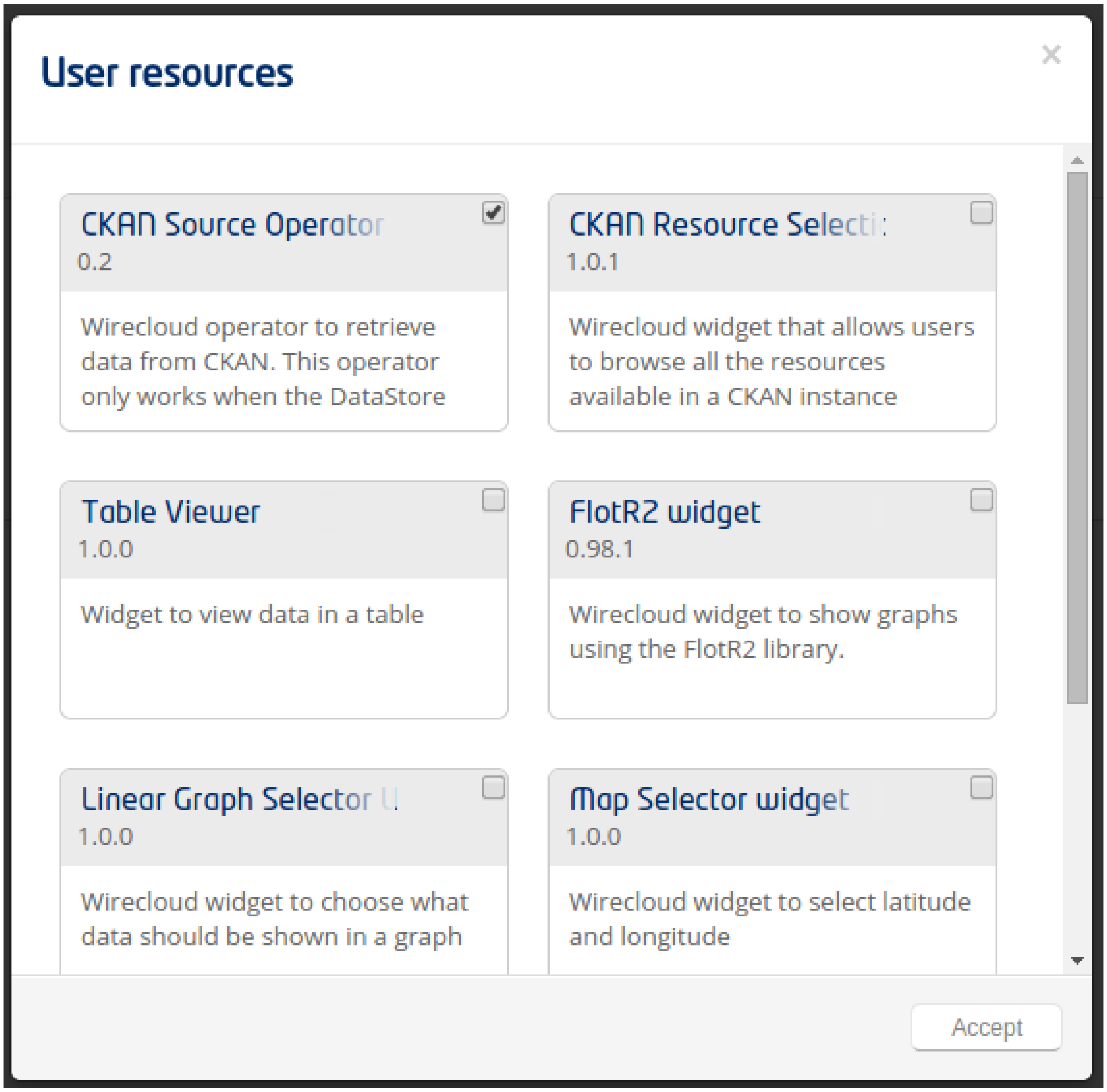
Version: 1.0

Publication date: None

Description

Operator to retrieve data from CKAN

In the displayed modal, select the resources to be bound and press Accept

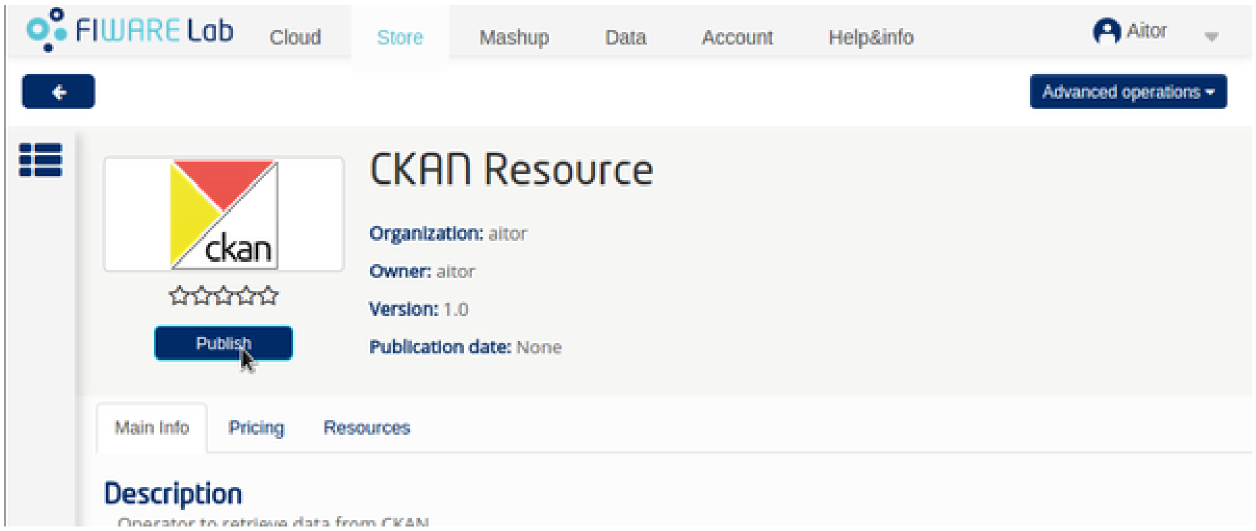

User resources
CKAN Source Operator
0.2
Wirecloud operator to retrieve data from CKAN. This operator only works when the DataStore
CKAN Resource Selecti
1.0.1
Wirecloud widget that allows users to browse all the resources available in a CKAN instance
Table Viewer
1.0.0
Widget to view data in a table
FlotR2 widget
0.98.1
Wirecloud widget to show graphs using the FlotR2 library.
Linear Graph Selector
1.0.0
Wirecloud widget to choose what data should be shown in a graph
Map Selector widget
1.0.0
Wirecloud widget to select latitude and longitude
Accept

Note that this operation is an absolute update, that is, the selected resources are the bound resources. Therefore, it is possible to bind and unbind resources in the same action.

Publishing an offering

Publishing an offering means start selling it. To publish an offering select the My Offerings view and the Provided section. Then select the offering to be published.

In the offering details view select the Publish option.



CKAN Resource

Organization: aitor

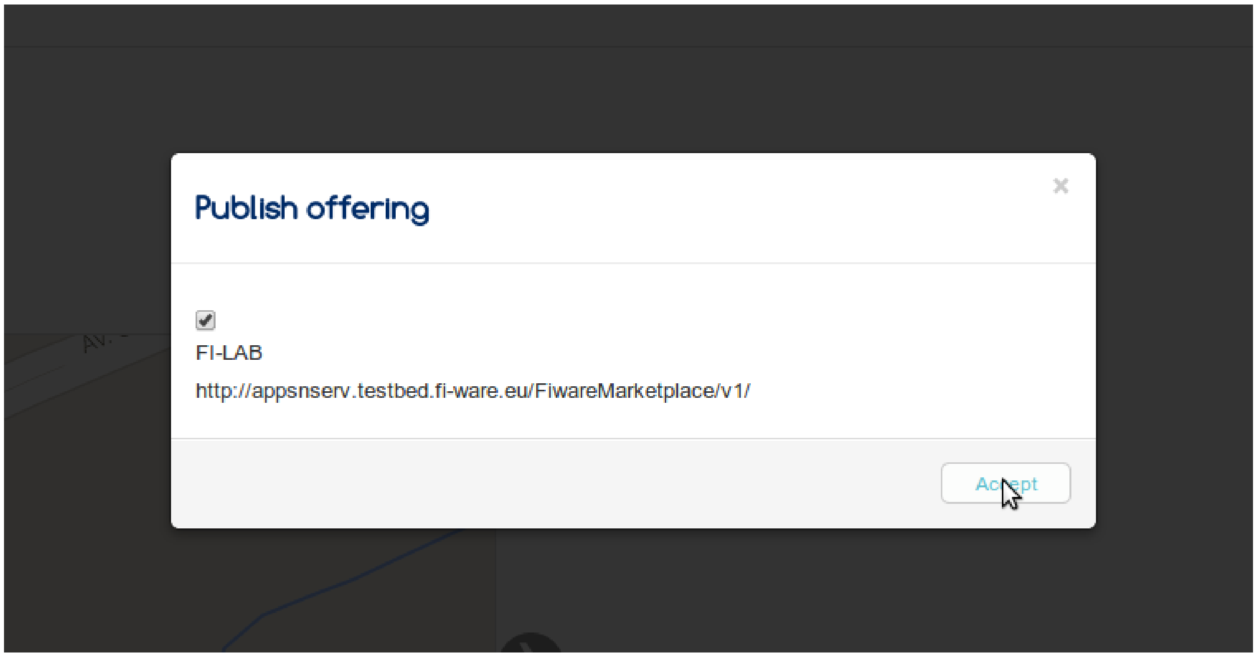
Owner: aitor

Version: 1.0

Publication date: None

Description

In the displayed modal, select the Marketplaces where publish the offerings. Note that selecting a Marketplace is not mandatory.



FI-LAB

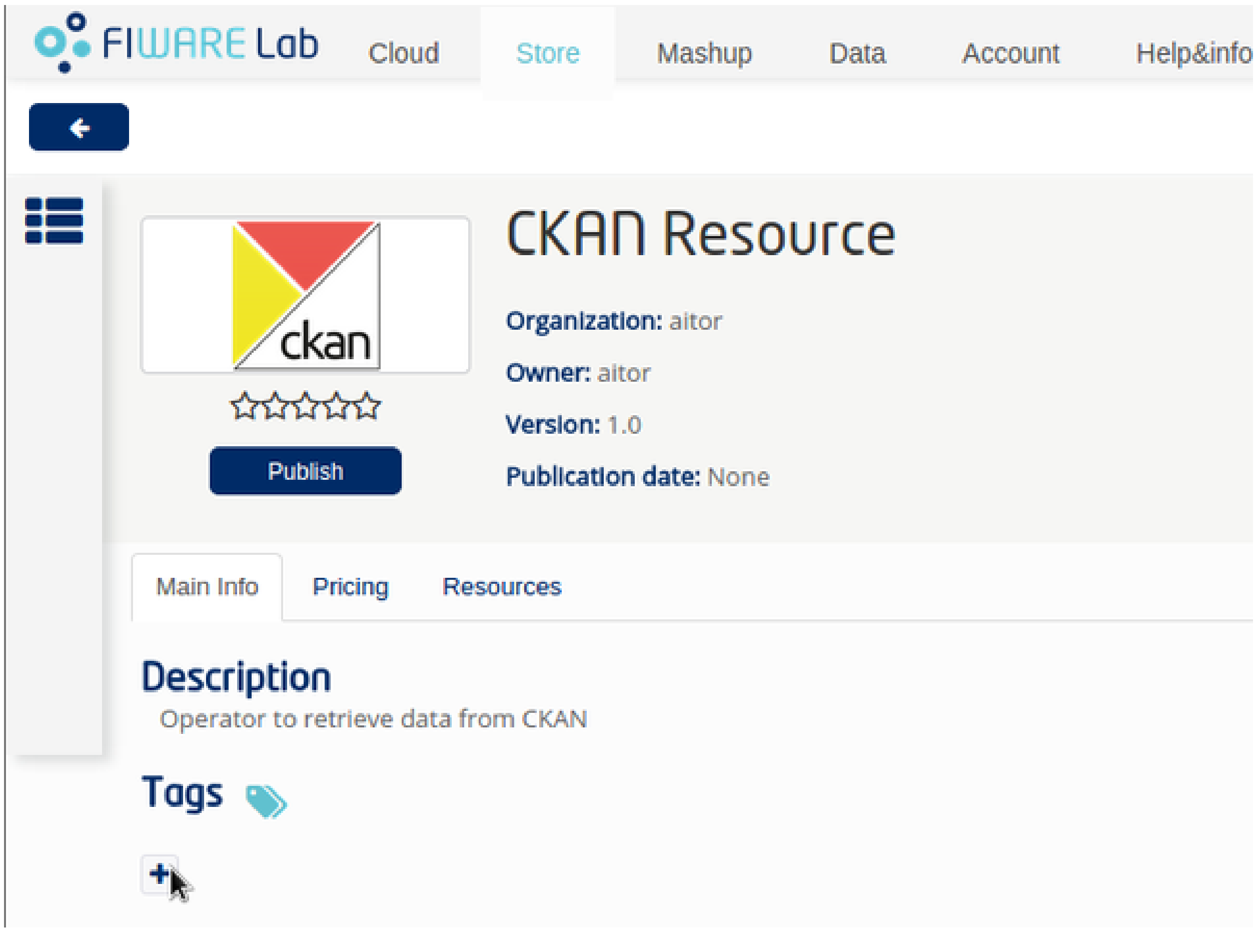
http://appsnserv.testbed.fi-ware.eu/FiwareMarketplace/v1/

The offering is now Published and cannot be updated.

Tagging an offering

It is possible for an offering provider to tag their offerings. To tag an offering, select the My Offerings view and the Provided section. Then select the offering to be tagged.

Select the Update tags option



CKAN Resource

Organization: aitor

Owner: aitor

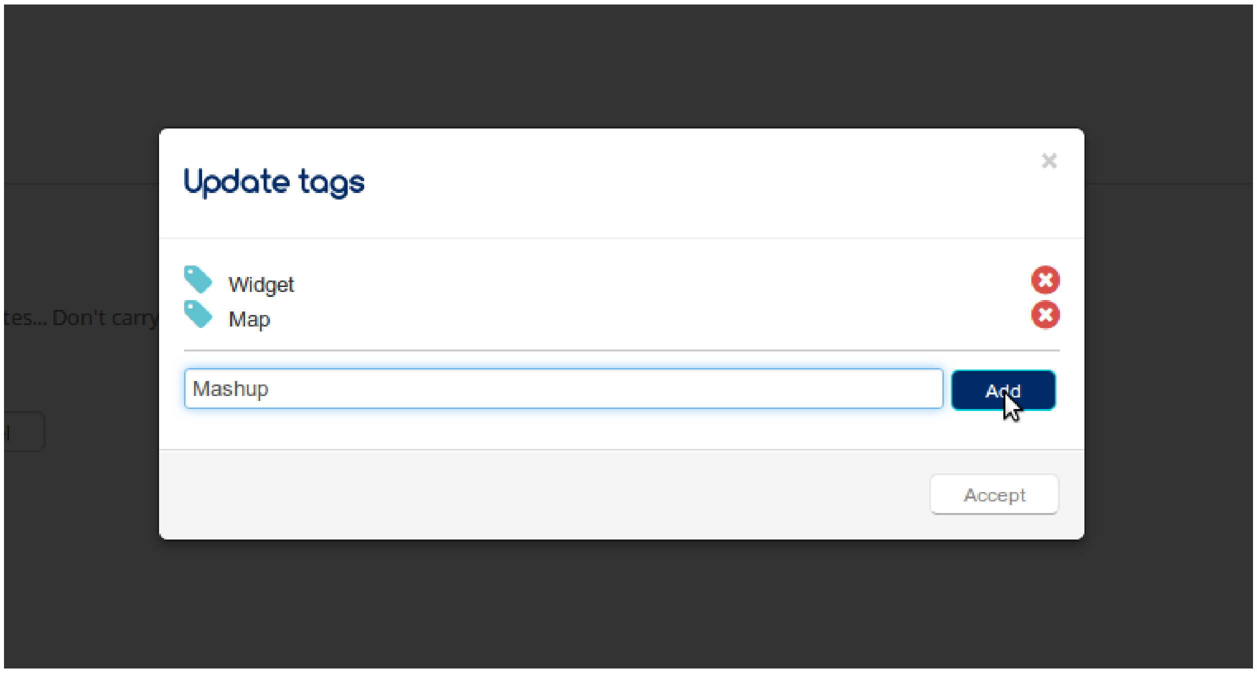
Version: 1.0

Publication date: None

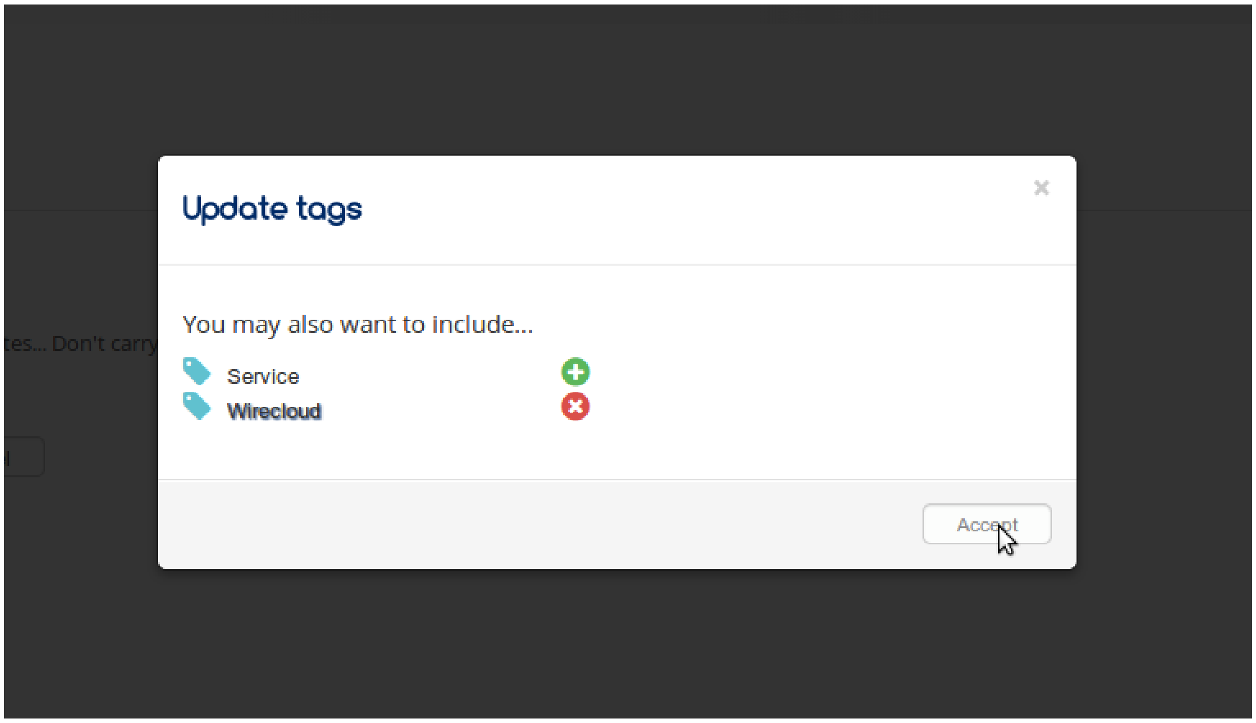
Description

Operator to retrieve data from CKAN

Include the different tags and press Accept



You may also be suggested some tags that may fit your offering.

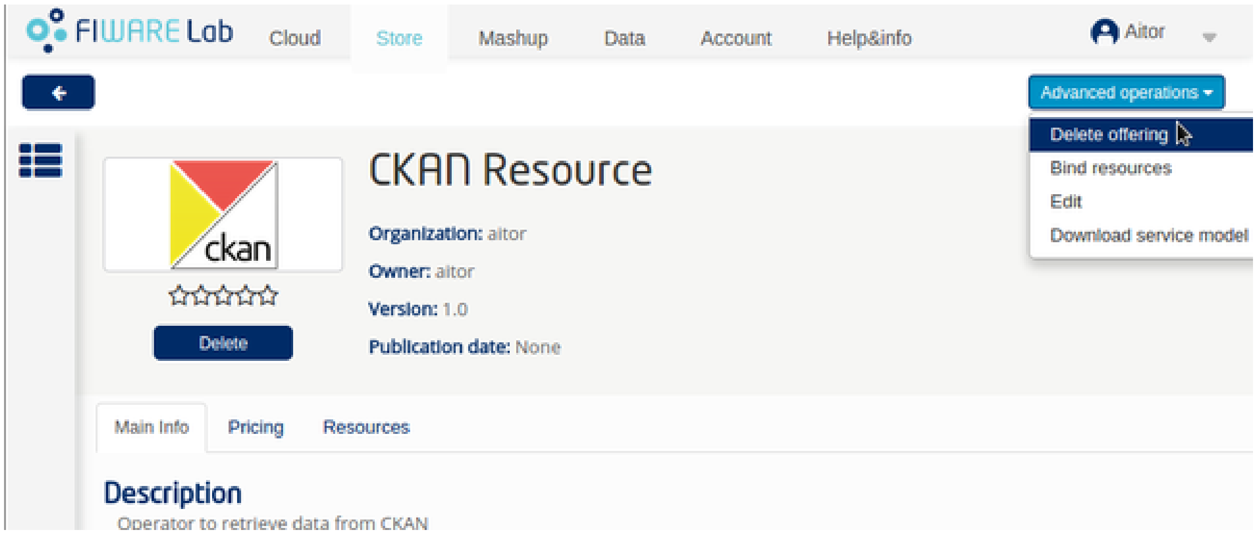


Deleting an offering

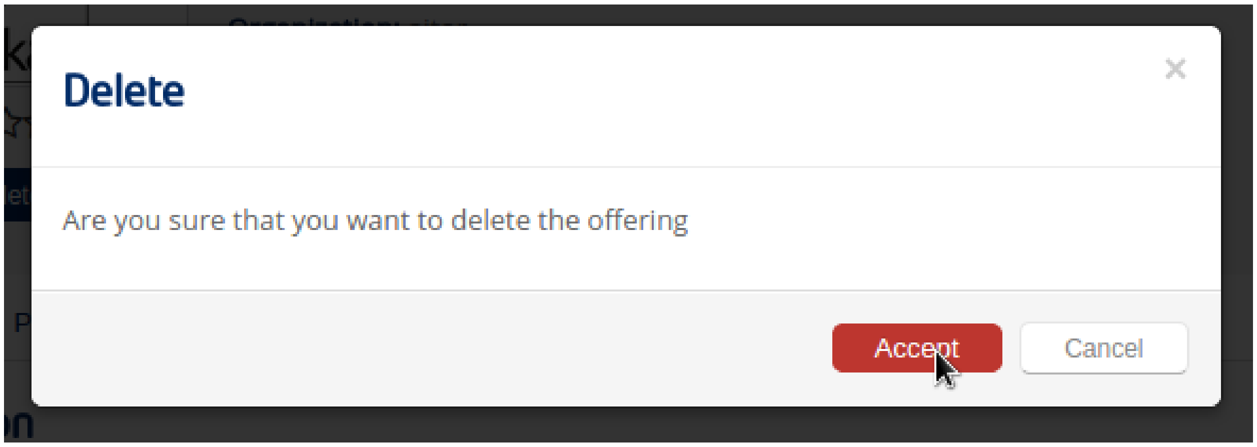
The action of deleting an offering has different effects depending on its state. If the offering has not been published, it is completely deleted from WStore. However, If the offering has been published, its state changes to deleted and cannot be acquired anymore, but customers that has already acquired it still has access to the offering and its resources.

To delete an offering, select the My Offerings view and the Provided section. Then select the offering to be deleted.

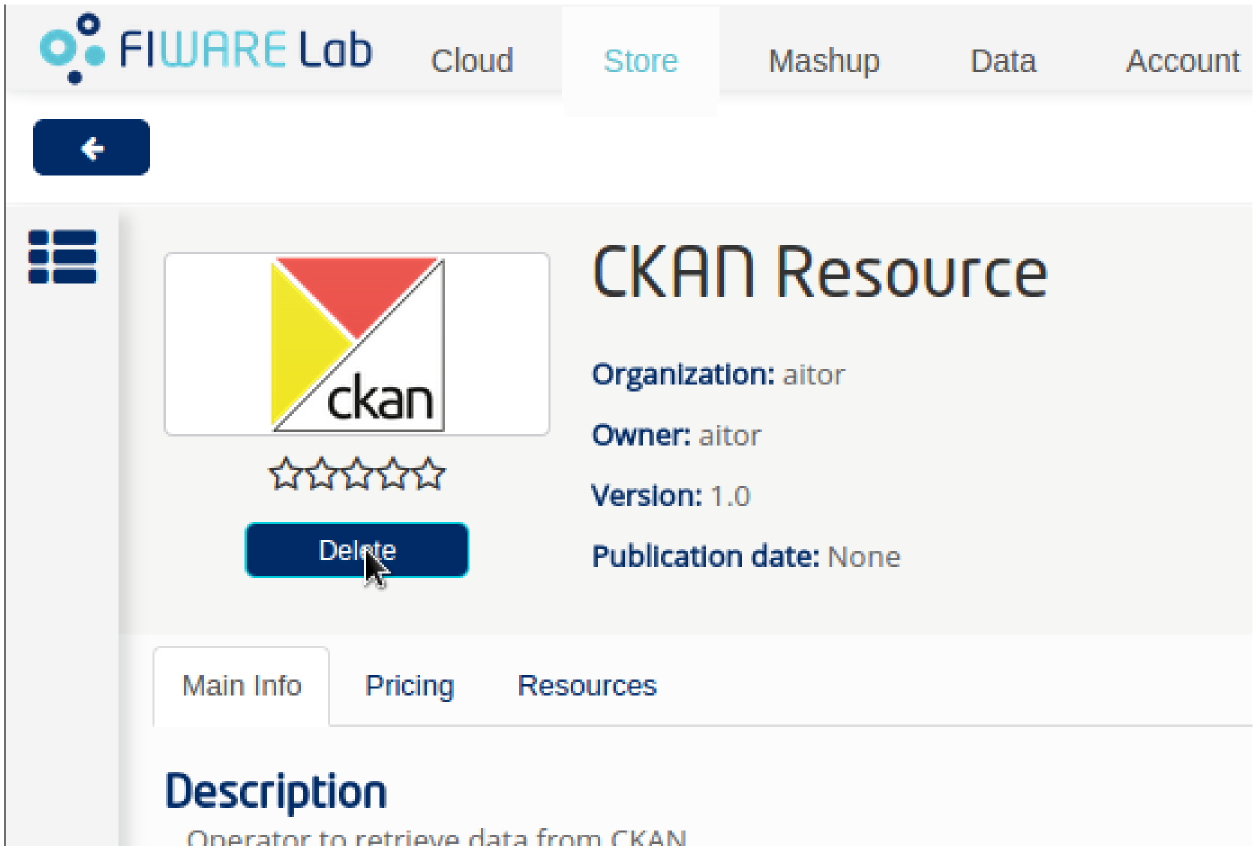
In the offering details view, select the Delete offering option



Select accept in the displayed window.



If the offering has been published the option Delete replaces Publish as main action.

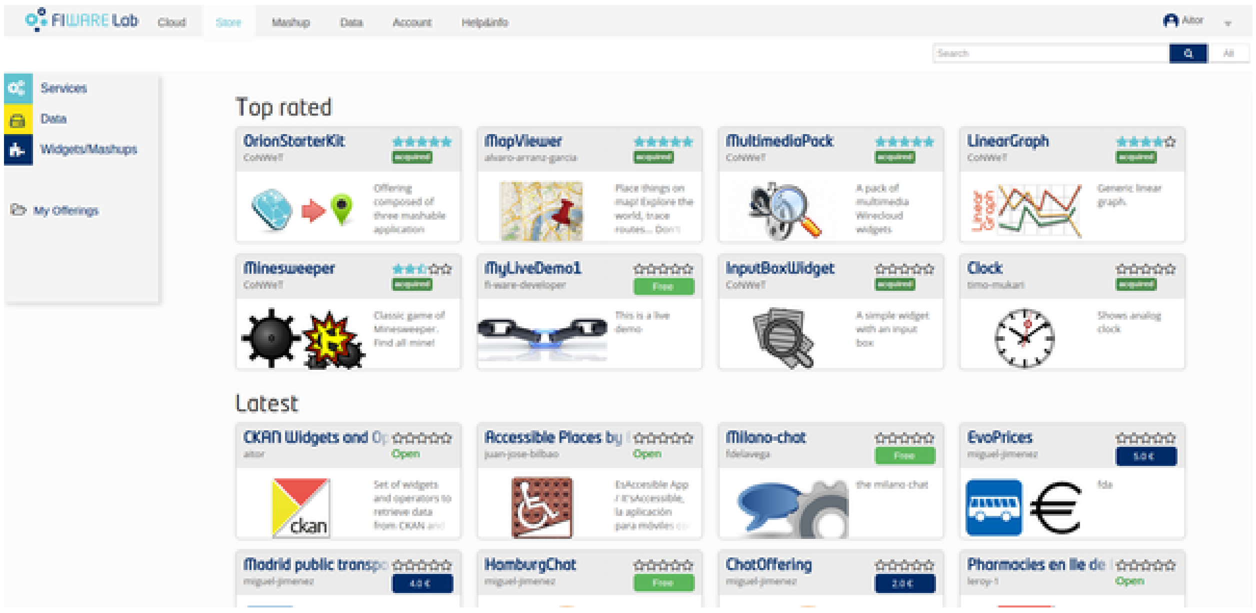


Customer

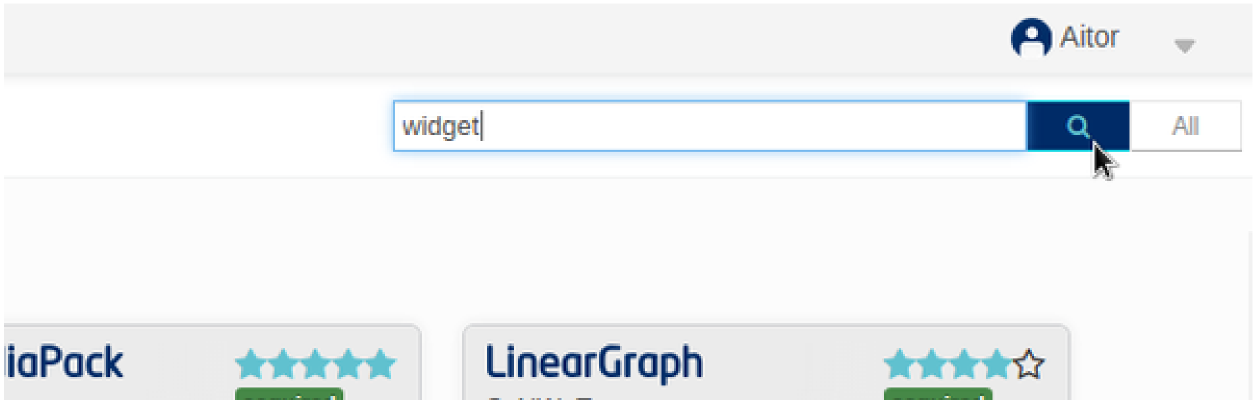
This section explains how a customer can search and acquire offerings using WStore GUI.

Searching for offerings

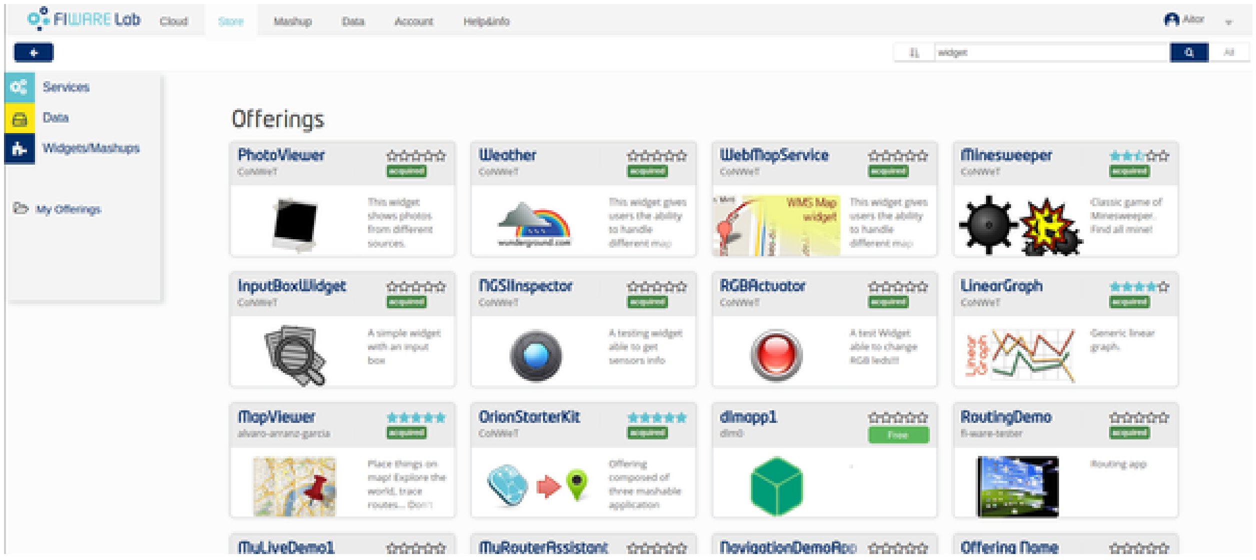
There are some options for searching offerings in WStore, As it can be seen in the following image, the main page contains the Top rated and the latest offerings.



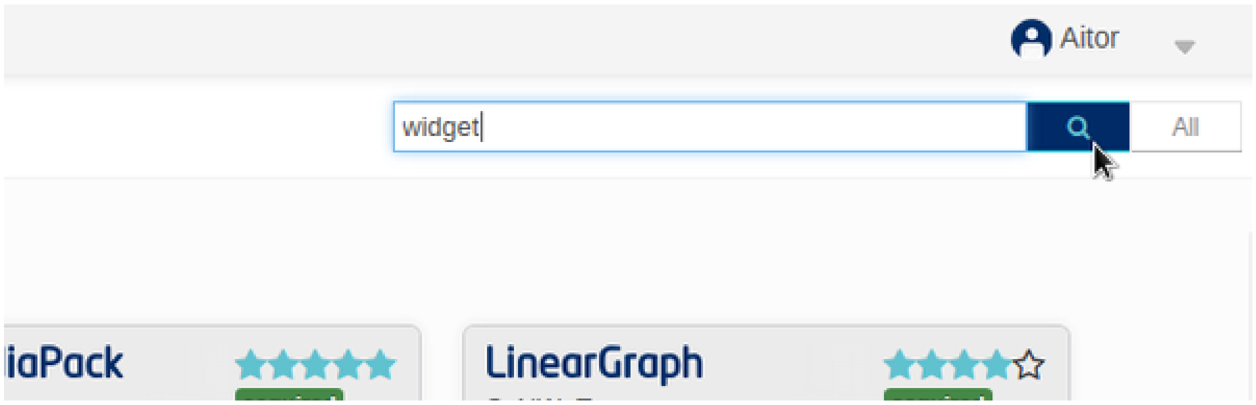
To search using a keyword type it in the textbox and press Search.



The offerings that match the search are shown.

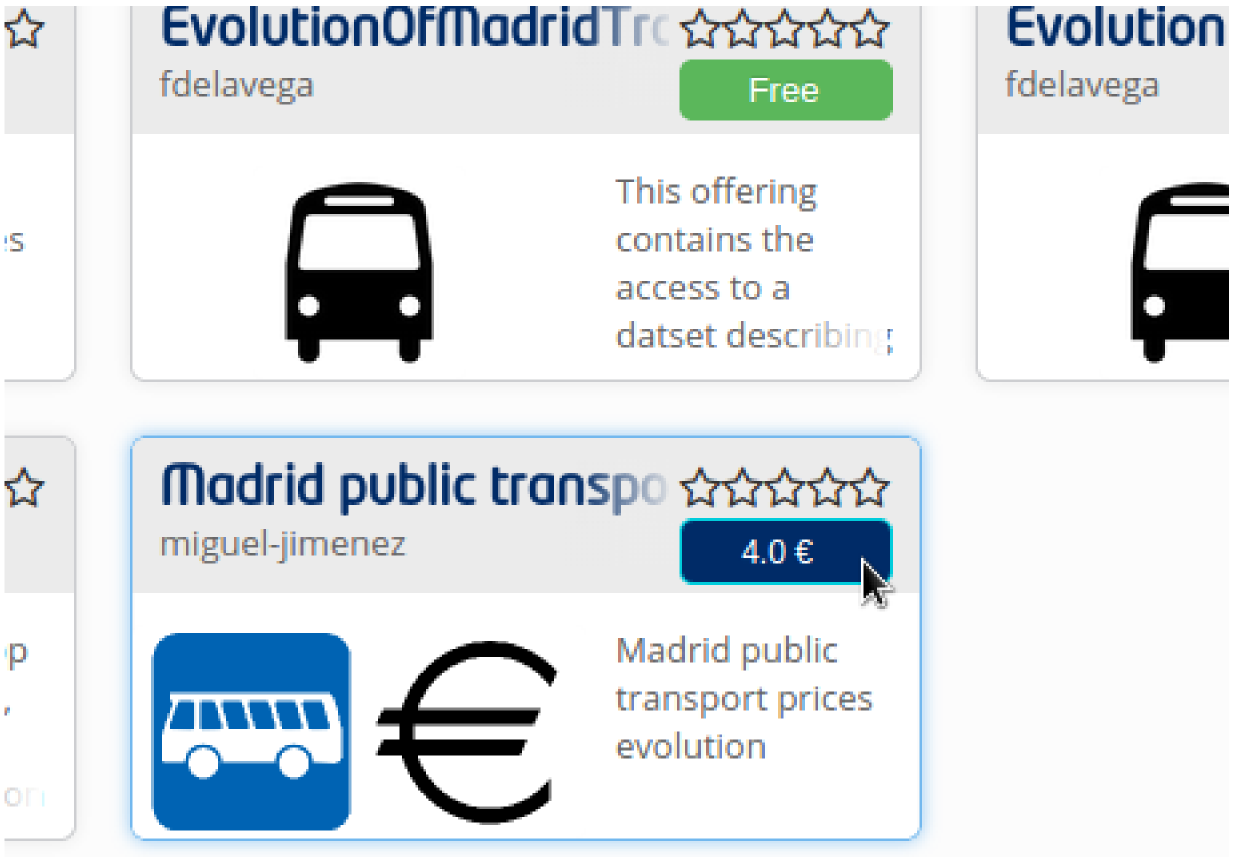


It is also possible to view all the offerings selecting the *All* button.

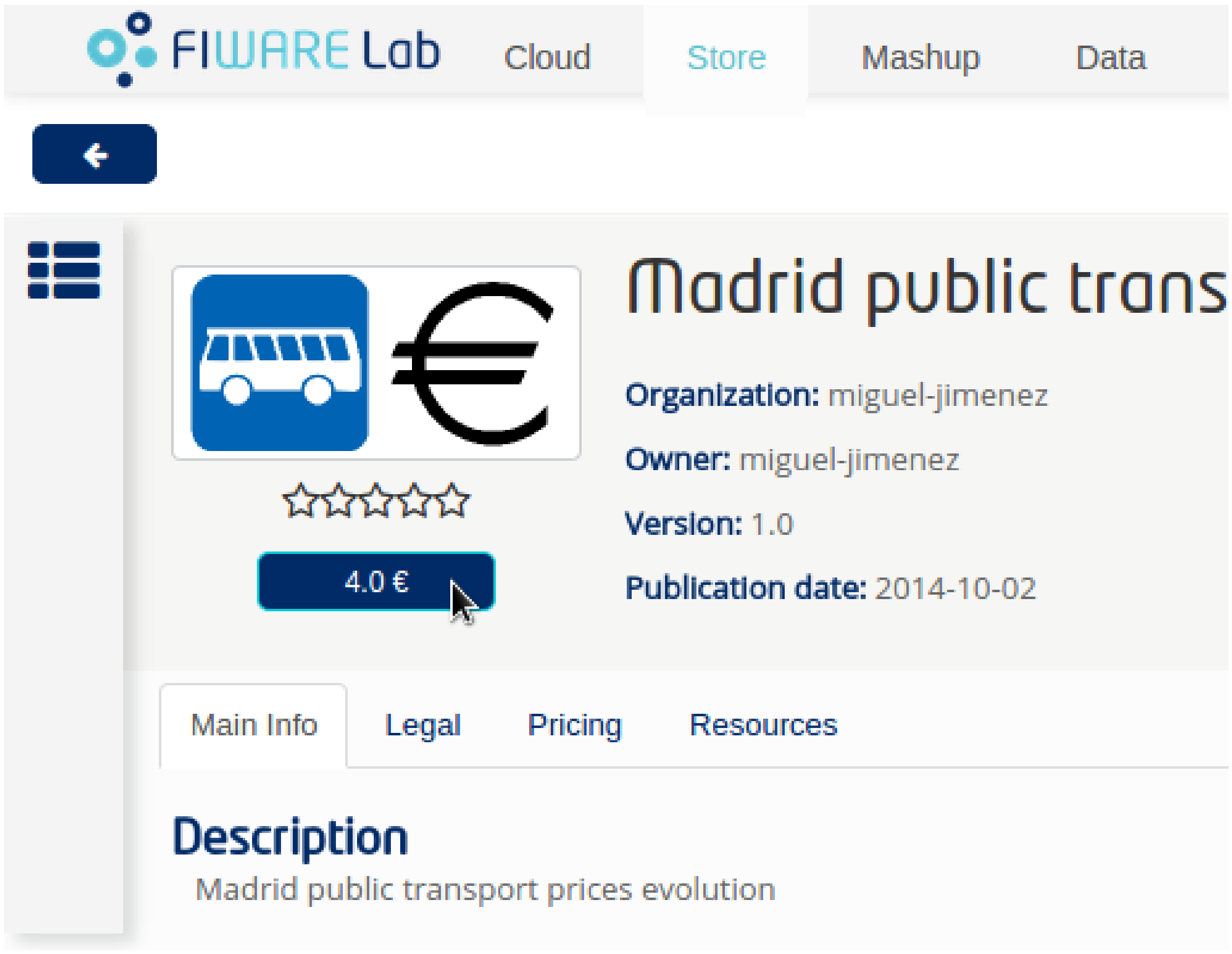


Acquiring an offering

The first step to acquire a published offering is selecting it after searching. To start with the purchasing process click on the button included in the offering.


EvolutionOfMadridTr
fdelavega
Free
This offering contains the access to a datset describing
Evolution
fdelavega
Madrid public transpo
miguel-jimenez
4.0 €
Madrid public transport prices evolution

Arternatively, it is possible to select the Acquire button in the offering details view.



Madrid public trans

Organization: miguel-jimenez

Owner: miguel-jimenez

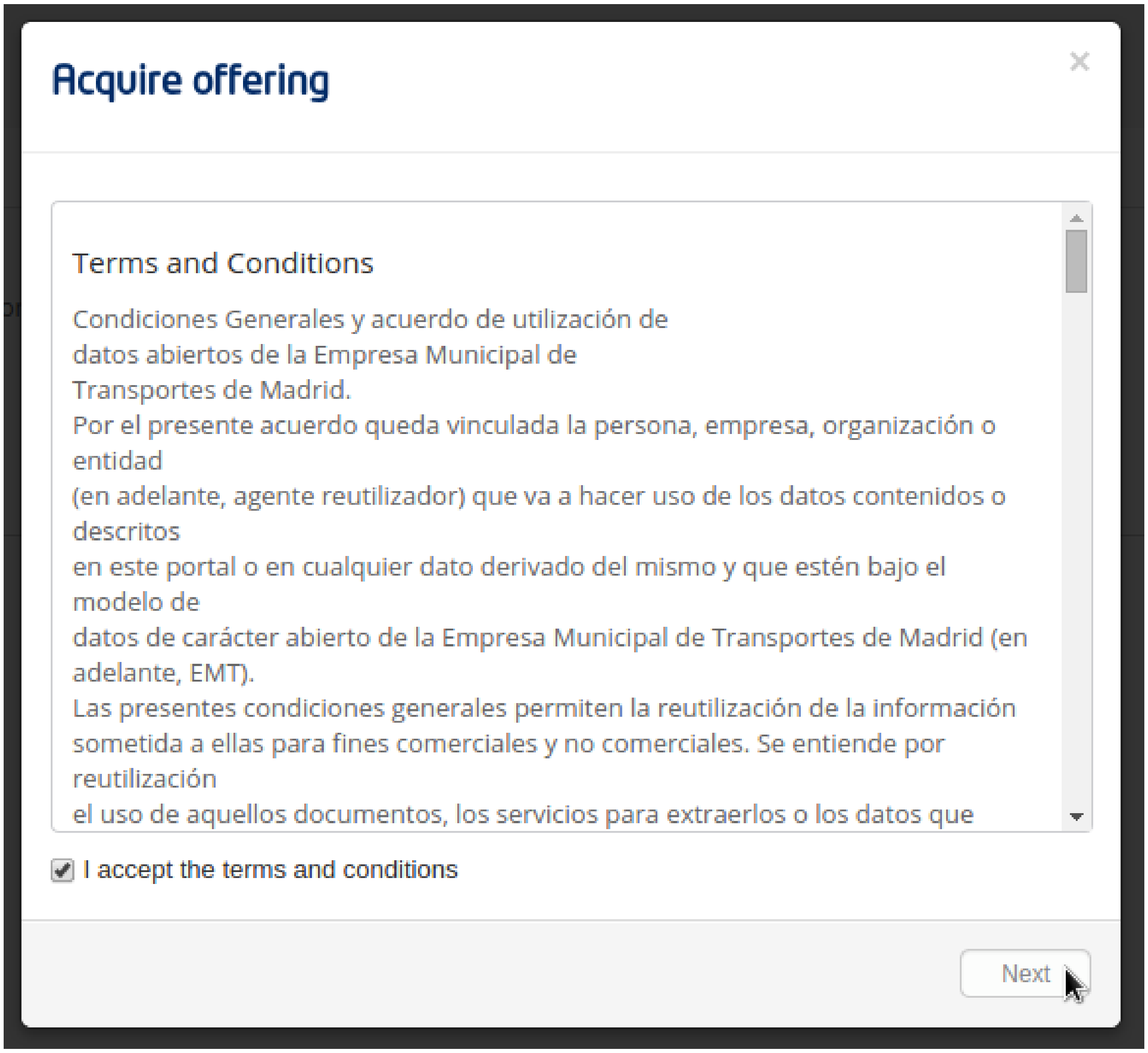
Version: 1.0

Publication date: 2014-10-02

Description

Madrid public transport prices evolution

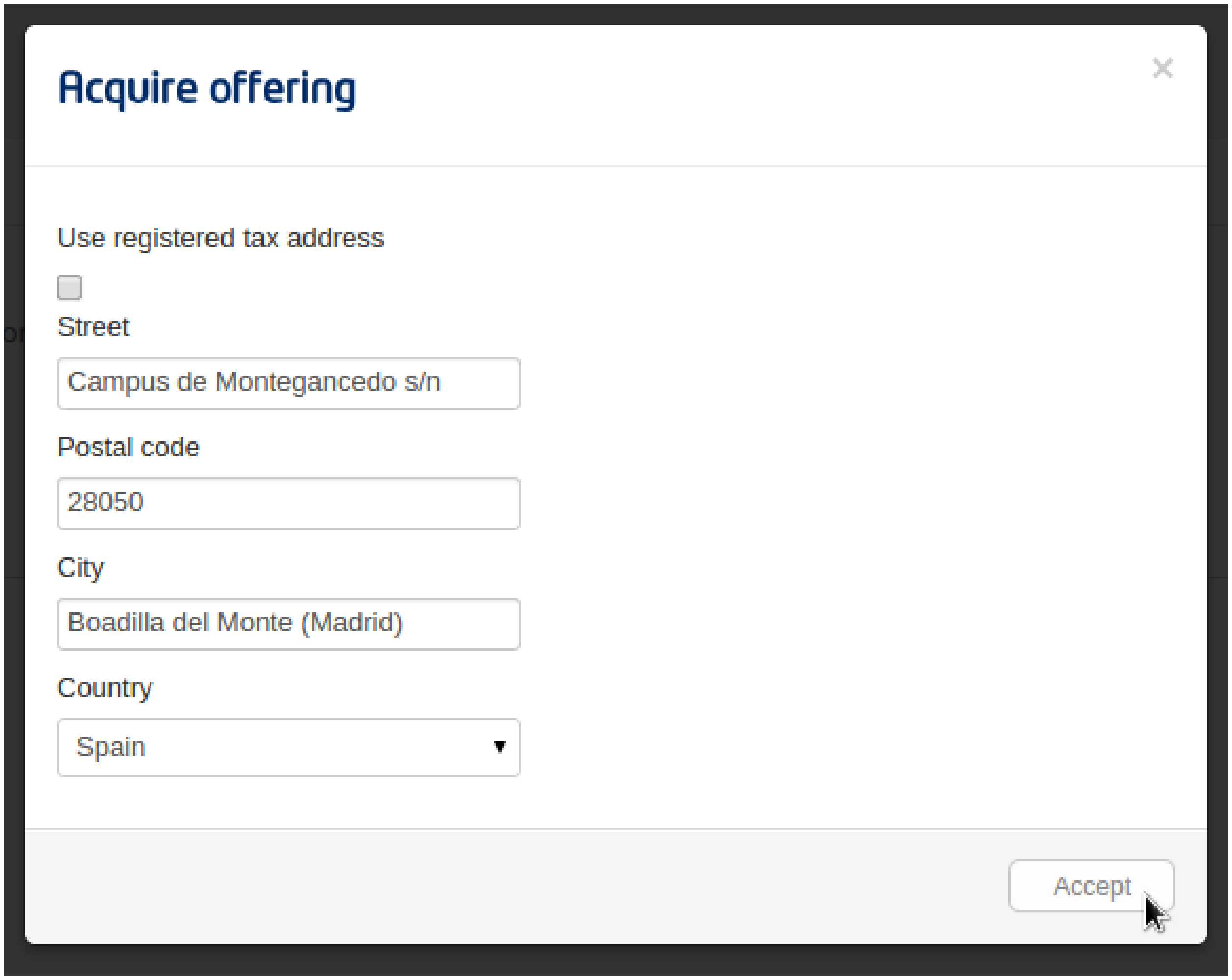
If the offering has some legal terms, you will be forced to accept them in order to be able to acquire it.



Terms and Conditions

Condiciones Generales y acuerdo de utilización de
datos abiertos de la Empresa Municipal de
Transportes de Madrid.
Por el presente acuerdo queda vinculada la persona, empresa, organización o
entidad
(en adelante, agente reutilizador) que va a hacer uso de los datos contenidos o
descritos
en este portal o en cualquier dato derivado del mismo y que estén bajo el
modelo de
datos de carácter abierto de la Empresa Municipal de Transportes de Madrid (en
adelante, EMT).
Las presentes condiciones generales permiten la reutilización de la información
sometida a ellas para fines comerciales y no comerciales. Se entiende por
reutilización
el uso de aquellos documentos, los servicios para extraerlos o los datos que

Once that you have accepted the terms, you will have to provide a tax address for the purchase. Is possible to use the default tax address from the user profile by clicking the checkbox Use user profile tax address. Then, select Accept.



Acquire offering

Use registered tax address

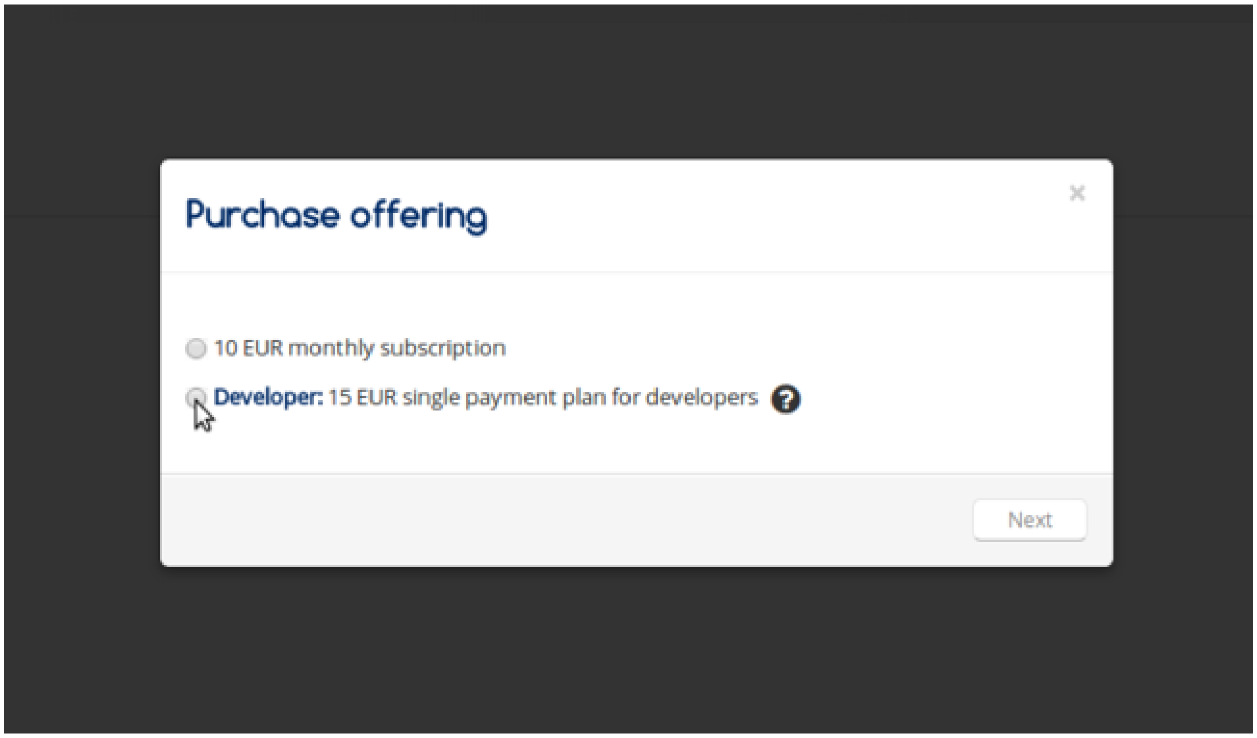
Street: Campus de Montegancedo s/n

Postal code: 28050

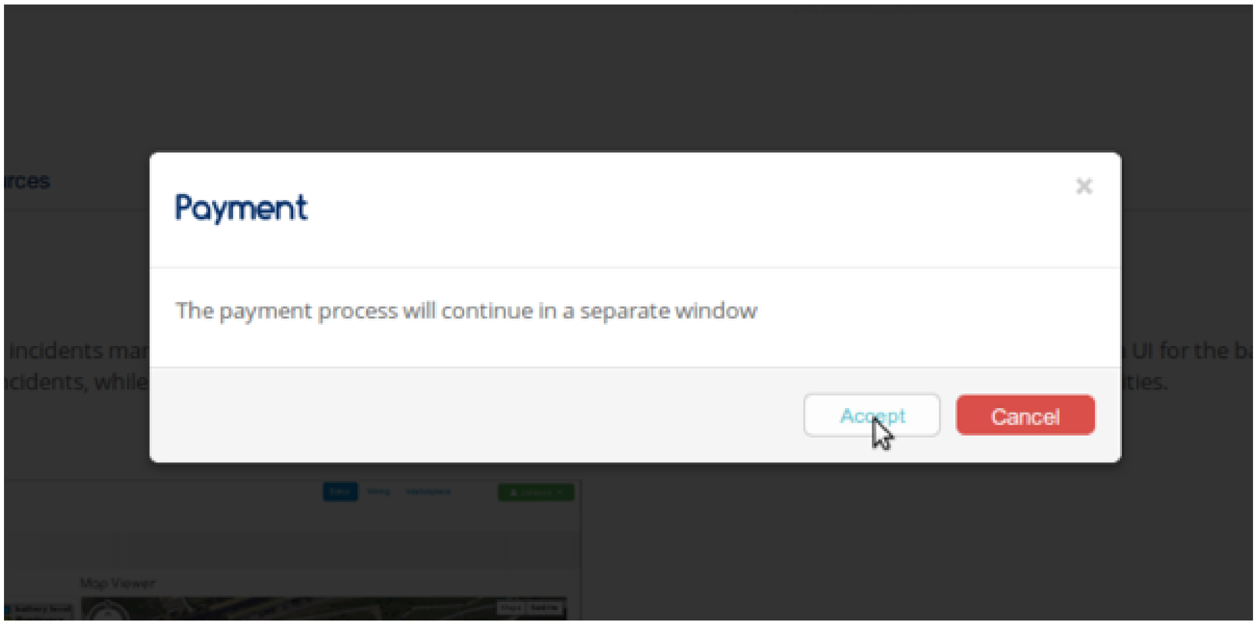
City: Boadilla del Monte (Madrid)

Country: Spain

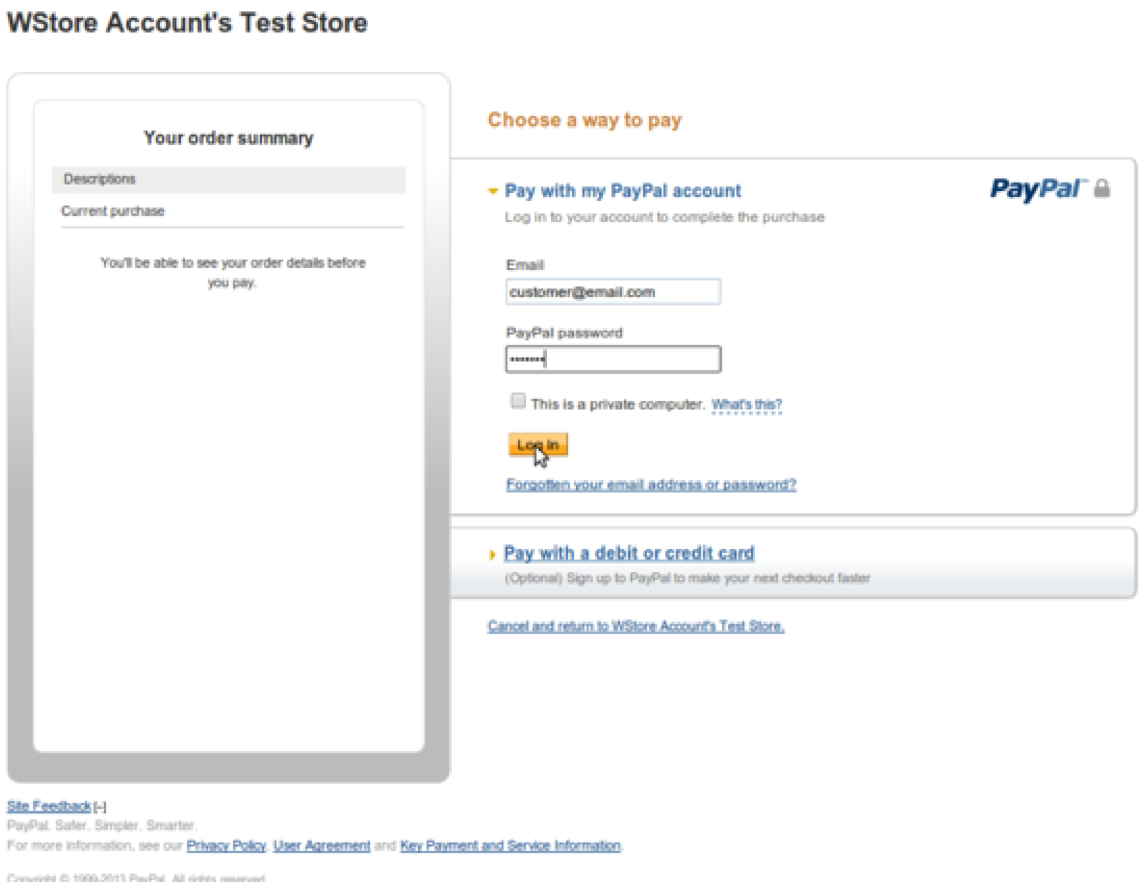
In case the offering can be acquired under different pricing models, the first step is selecting the plan.


Purchase offering
10 EUR monthly subscription
Developer: 15 EUR single payment plan for developers
Next

WStore informs that the payment process will continue in a separate window.



WStore redirects the browser to the PayPal confirmation page.



WStore Account's Test Store

Your order summary

Descriptions

Current purchase

You'll be able to see your order details before you pay.

Choose a way to pay

Pay with my PayPal account

Log in to your account to complete the purchase

Email

customer@email.com

PayPal password

This is a private computer. What's this?

Log In

Forgotten your email address or password?

Pay with a debit or credit card

(Optional) Sign up to PayPal to make your next checkout faster

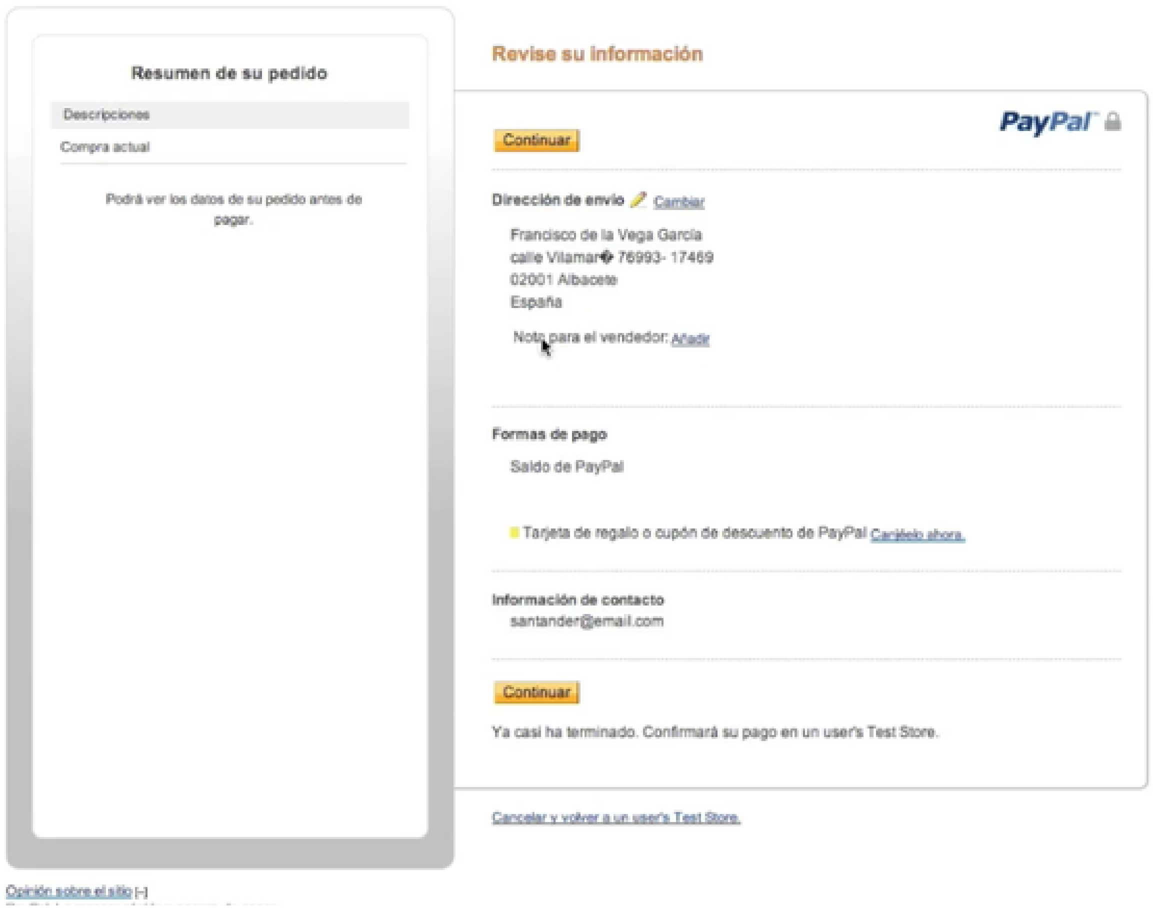
Cancel and return to WStore Account's Test Store.

Site Feedback [-]

PayPal. Safer. Simpler. Smarter.

For more information, see our Privacy Policy, User Agreement and Key Payment and Service Information.

Introduce your PayPal credentials and confirm the payment.


Resumen de su pedido

Descripciones

Compra actual

Podrá ver los datos de su pedido antes de pagar.

Revise su información

Continuar

Dirección de envío Cambiar

Francisco de la Vega García
calle Vilamar� 76993- 17469
02001 Albacete
España

Nota para el vendedor: Añadir

Formas de pago

Saldo de PayPal

Tarjeta de regalo o cupón de descuento de PayPal Canjéelo ahora.

Información de contacto
santander@email.com

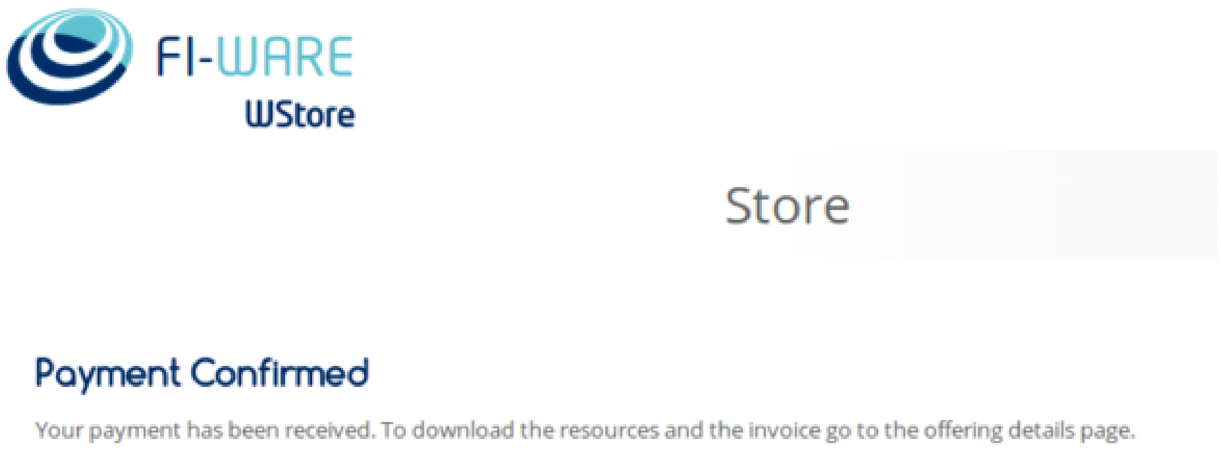
Continuar

Ya casi ha terminado. Confirmará su pago en un user's Test Store.

Cancelar y volver a un user's Test Store.

Opinión sobre el sitio [-]

Return to WStore page and end the process by closing the displayed window.


FI-WARE WStore

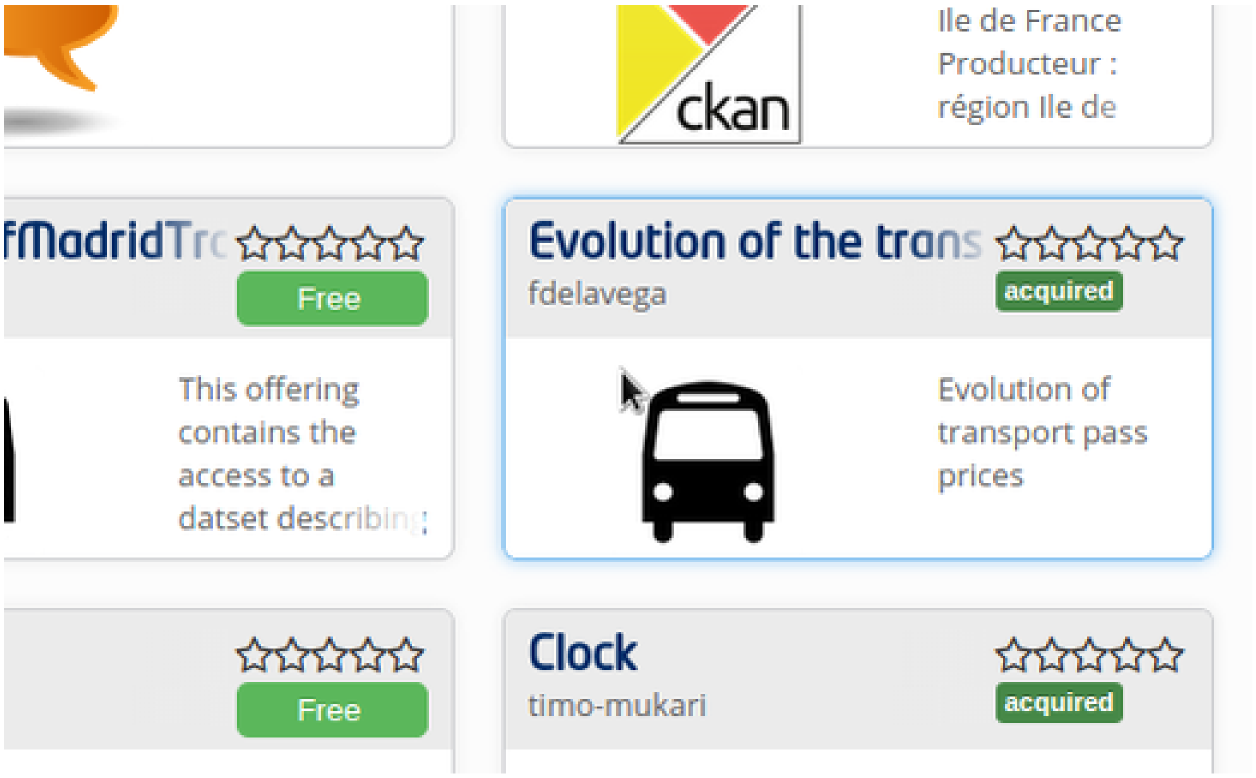
Store

Payment Confirmed

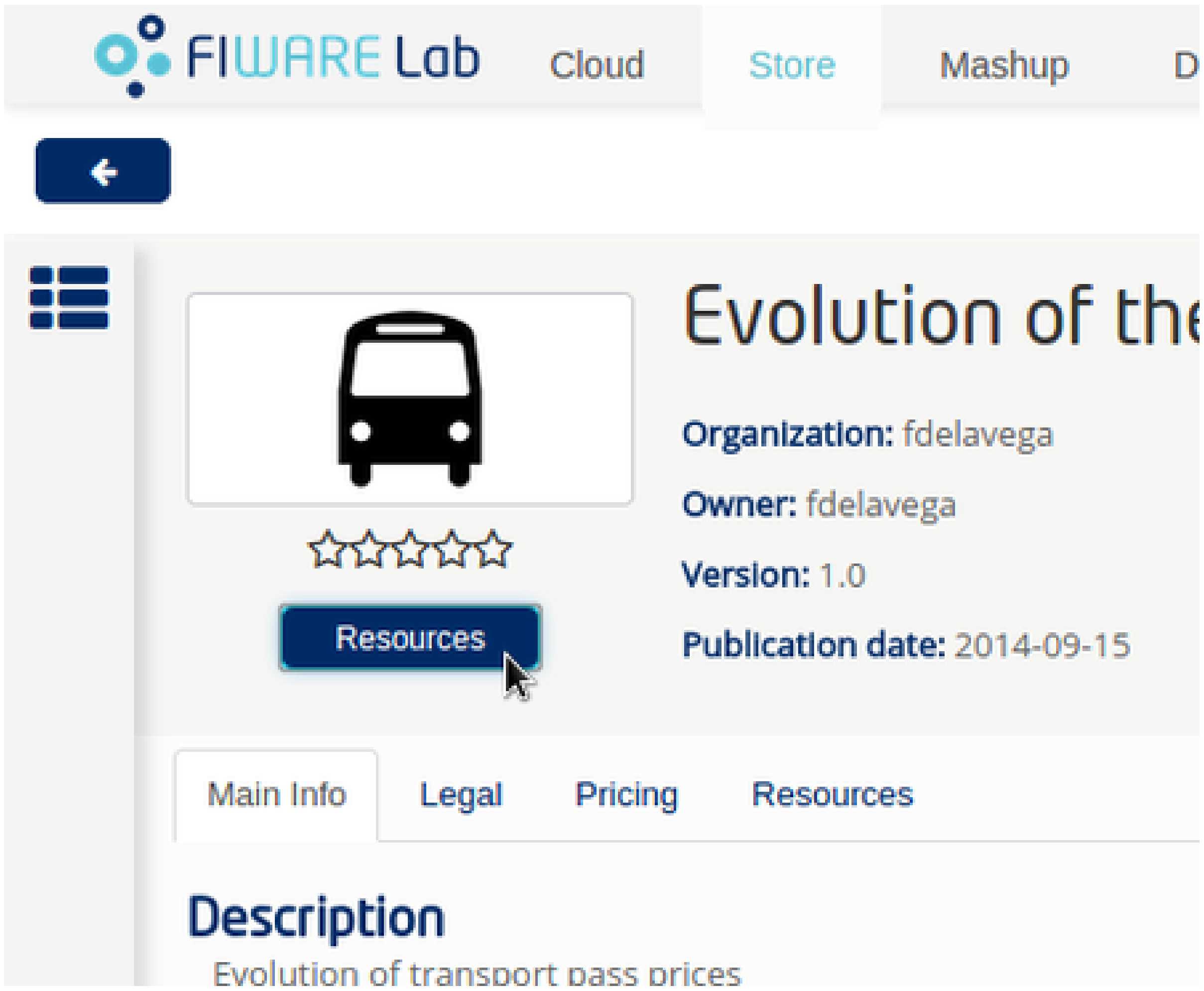
Your payment has been received. To download the resources and the invoice go to the offering details page.

Downloading resources and invoices

To download the resources and the invoices of a purchased offering, select the My Offerings view and the Acquired section . Then, select the offering.



Select the Resources button.



Evolution of the

Organization: fdelavega

Owner: fdelavega

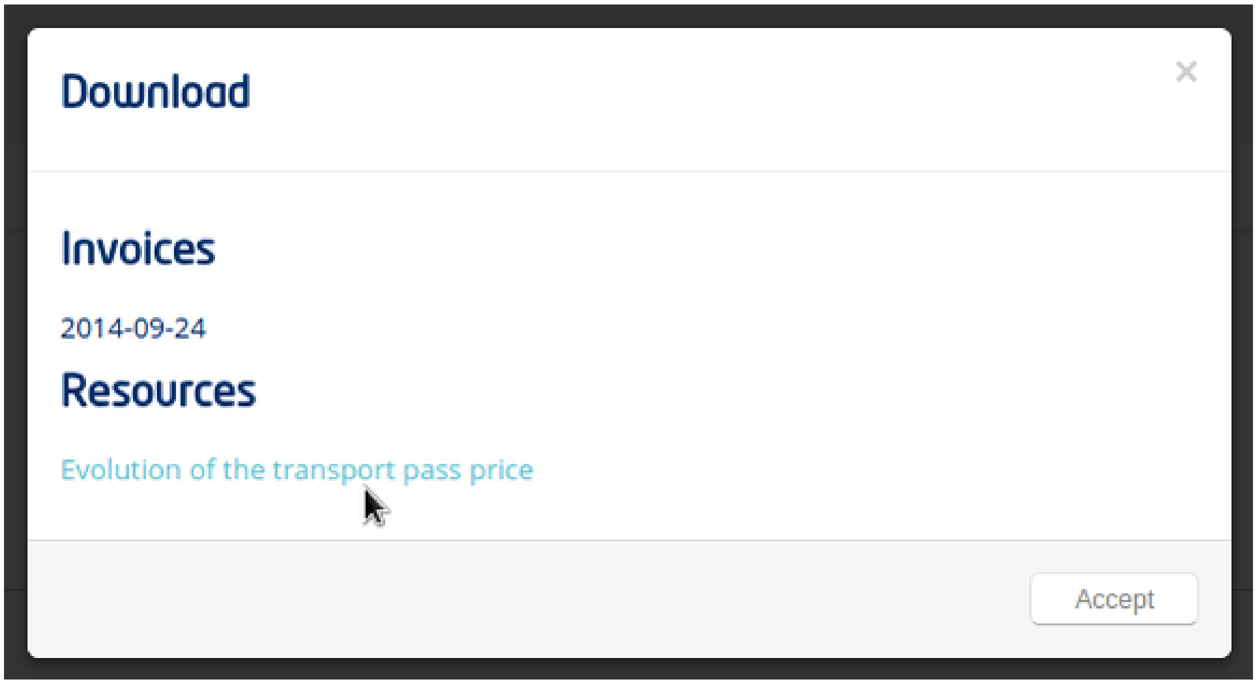
Version: 1.0

Publication date: 2014-09-15

Description

Evolution of transport pass prices

In the displayed modal, is possible to download invoices and resources by clicking on the link.


Download

Invoices

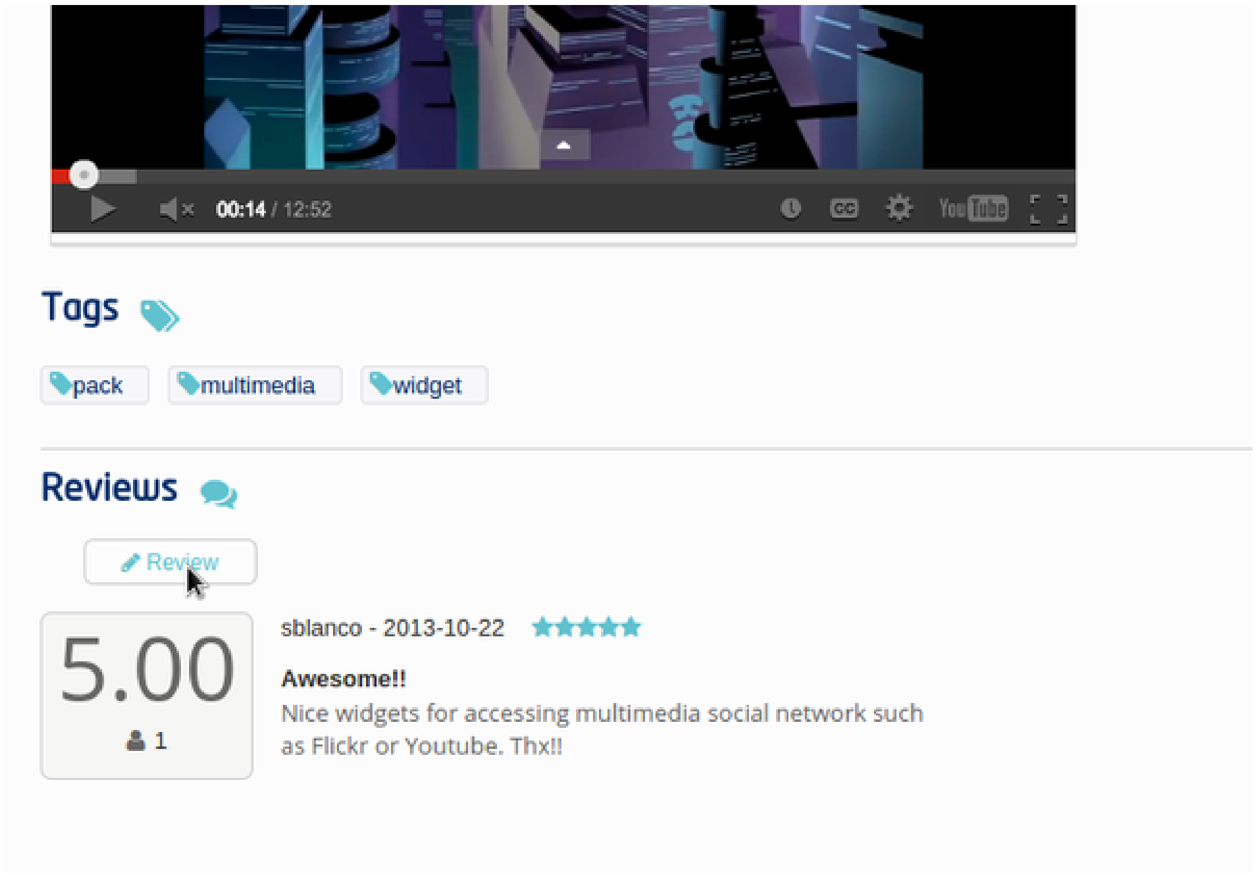
2014-09-24

Resources

Evolution of the transport pass price

Reviewing an offering

To review and rate an offering, select an acquired or an open offering and click on the Review button situated in the Reviews section.

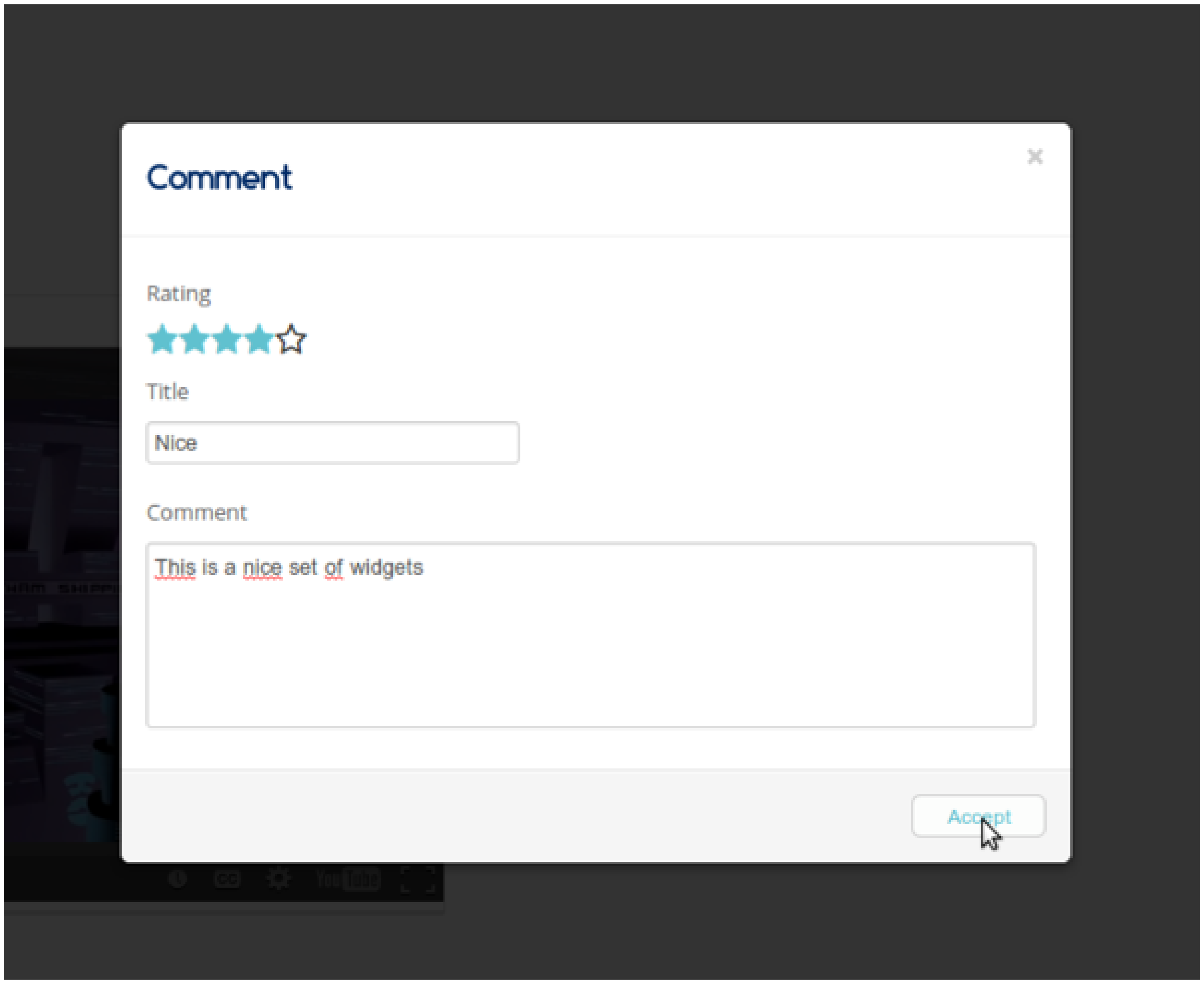


sblanco - 2013-10-22

Awesome!!

Nice widgets for accessing multimedia social network such as Flickr or Youtube. Thx!!

Fill the number of stars, give a title and a comment for your review.



Rating

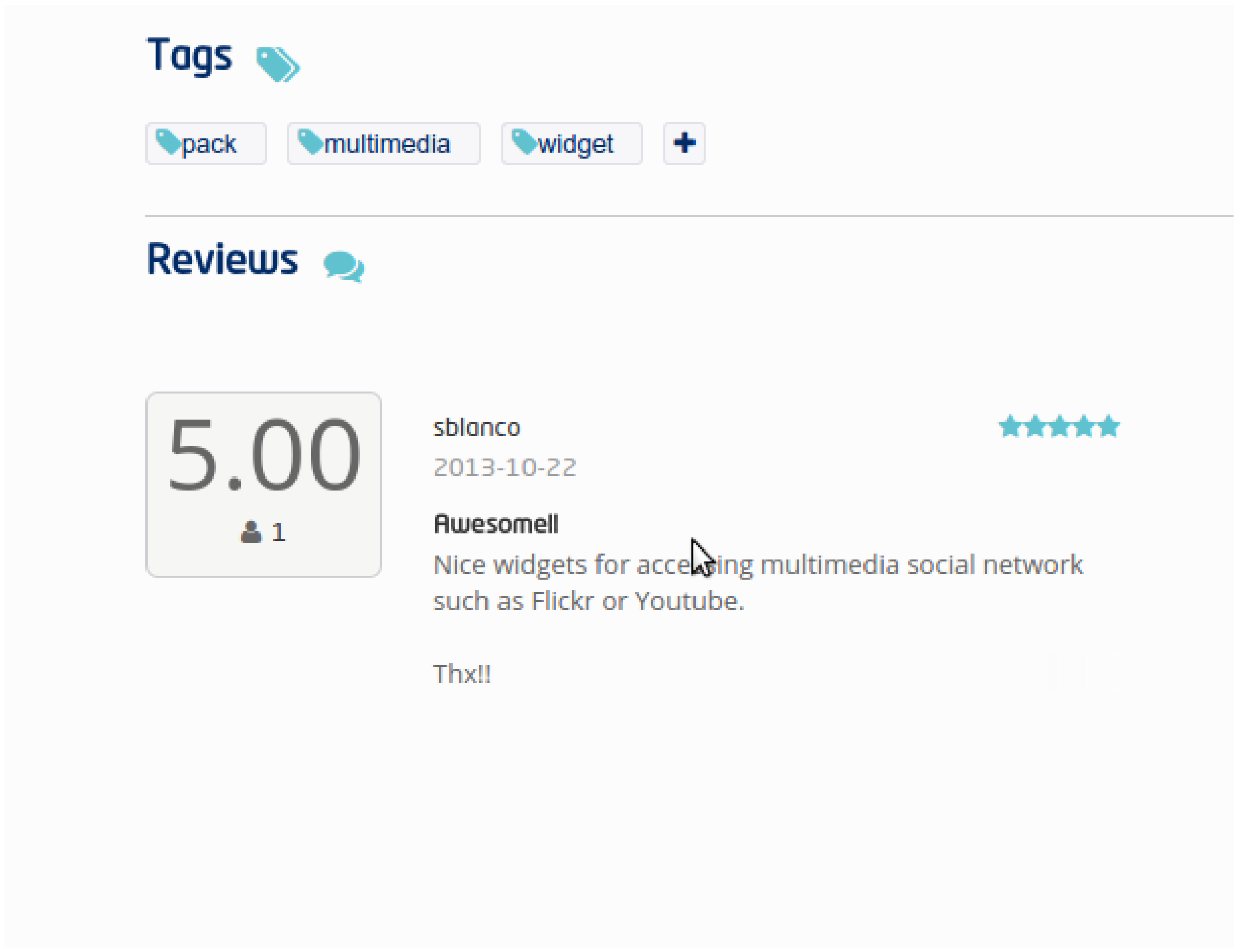
Title

Nice

Comment

This is a nice set of widgets

Additionally, the owner of the offering can reply to the existing reviews. To reply a review, the first step is clicking on the review to display its whole information.



sblanco

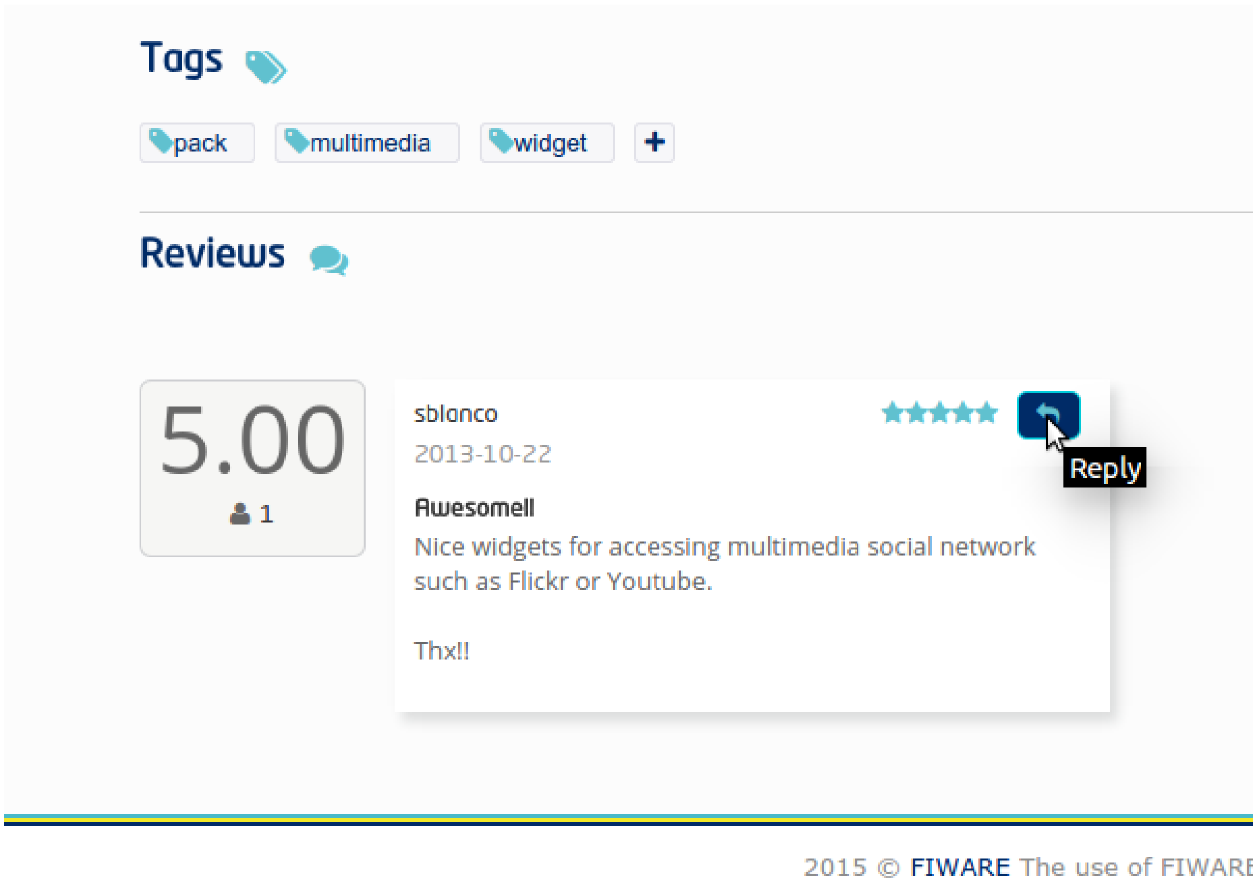
2013-10-22

Awesome!!

Nice widgets for accessing multimedia social network such as Flickr or Youtube.

Thx!!

Then, click on the *reply* button.



sblanco
2013-10-22

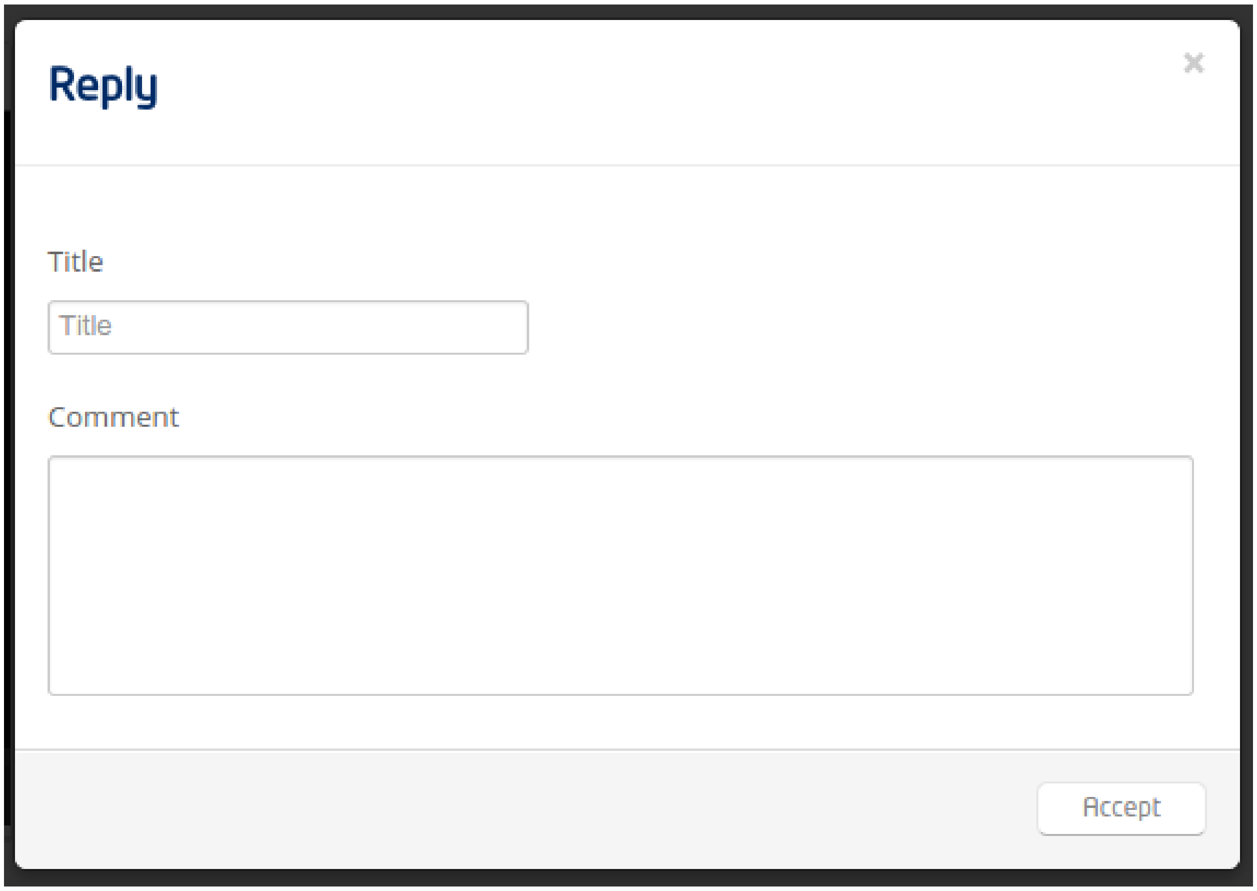
Awesome!!

Nice widgets for accessing multimedia social network such as Flickr or Youtube.

Thx!!

2015 © FIWARE The use of FIWARE

Finally, provide a title and a comment for your reply.



Admin

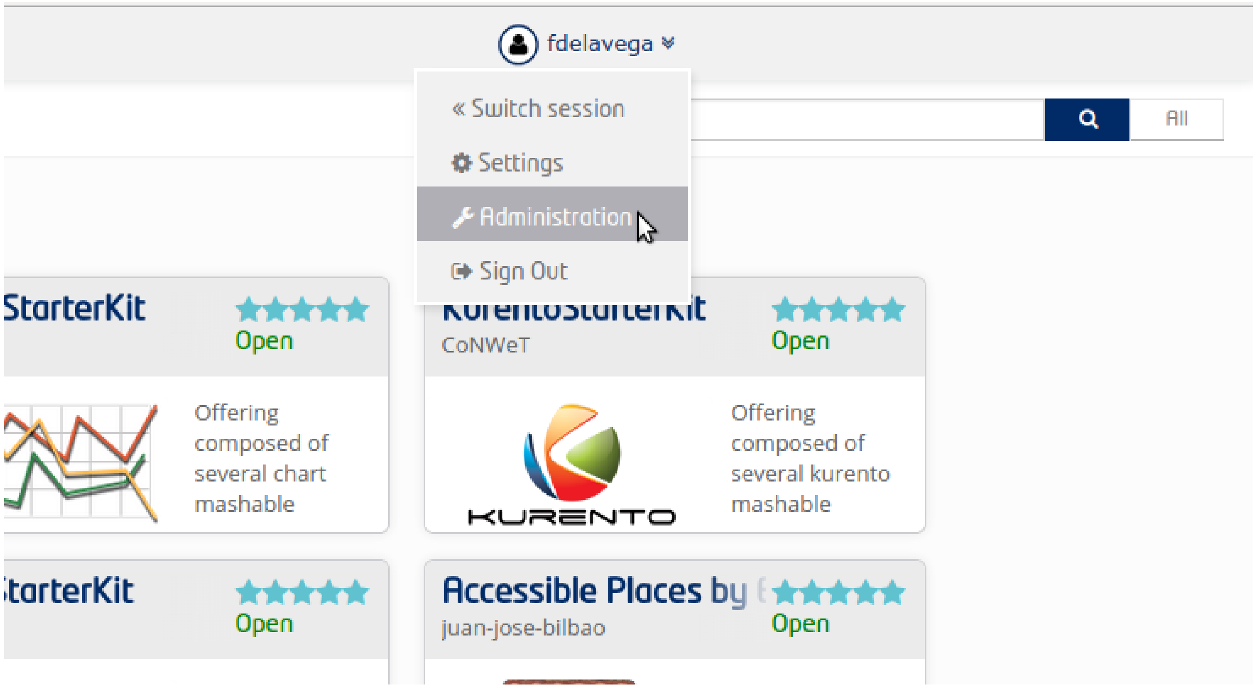
This section describes the different tasks that can be performed by an admin user using WStore GUI.

Registering WStore on a Marketplace

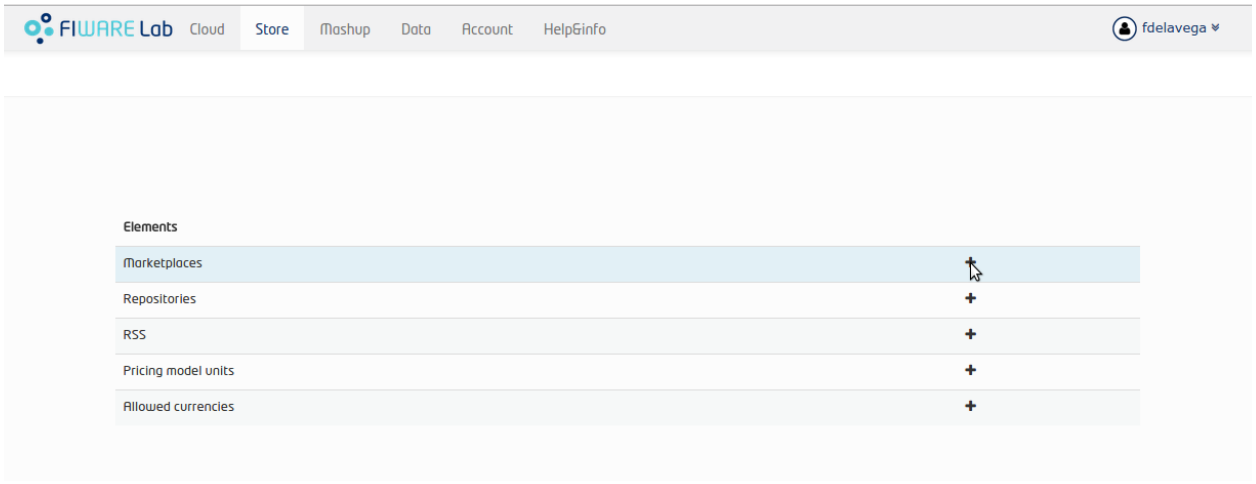
WStore can be registered on a Marketplace in order to allow service providers to publish their offerings on them, making their offerings available to potential customers that search for offerings in the Marketplace.

Note that this process is made from WStore GUI, since WStore needs to have information about in what Marketplaces is registered on.

To register WStore on a Marketplace, select the Administration view.

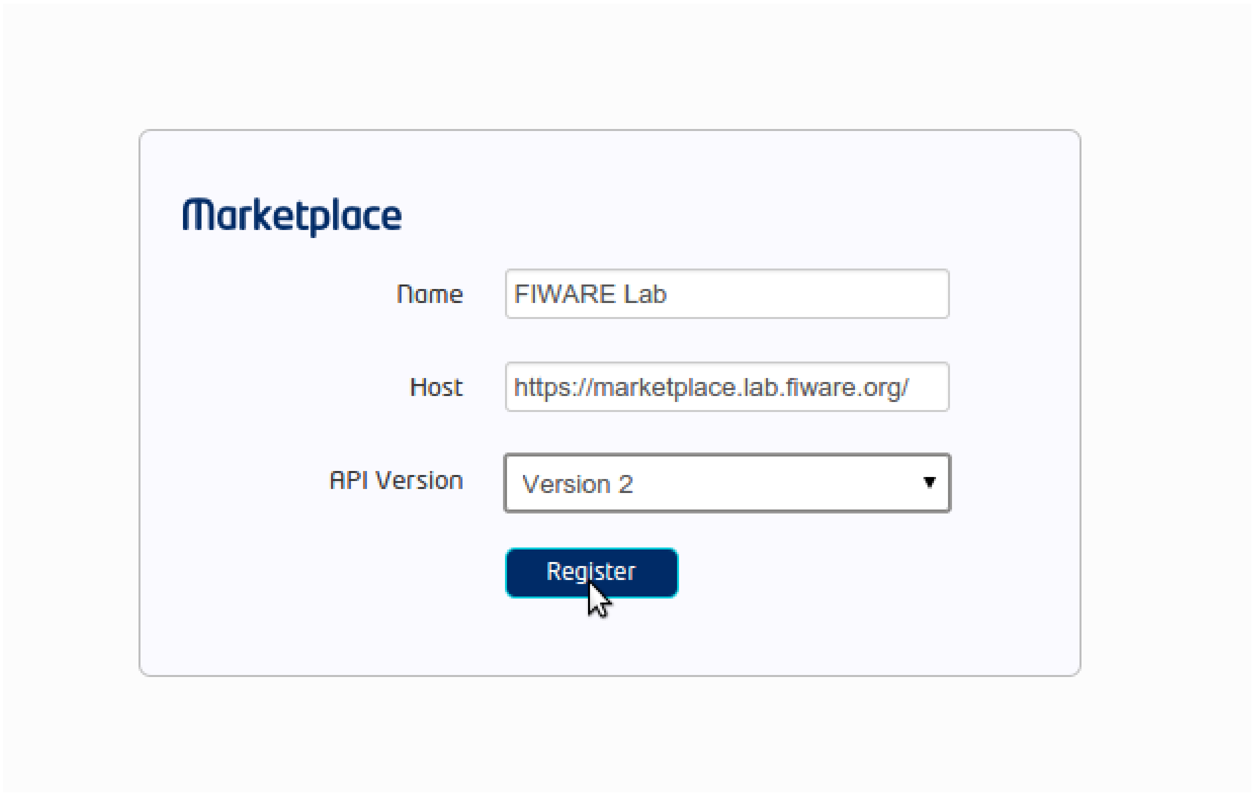


Press the Add symbol of the Marketplaces row.



Elements	
Marketplaces	+
Repositories	+
RSS	+
Pricing model units	+
Allowed currencies	+

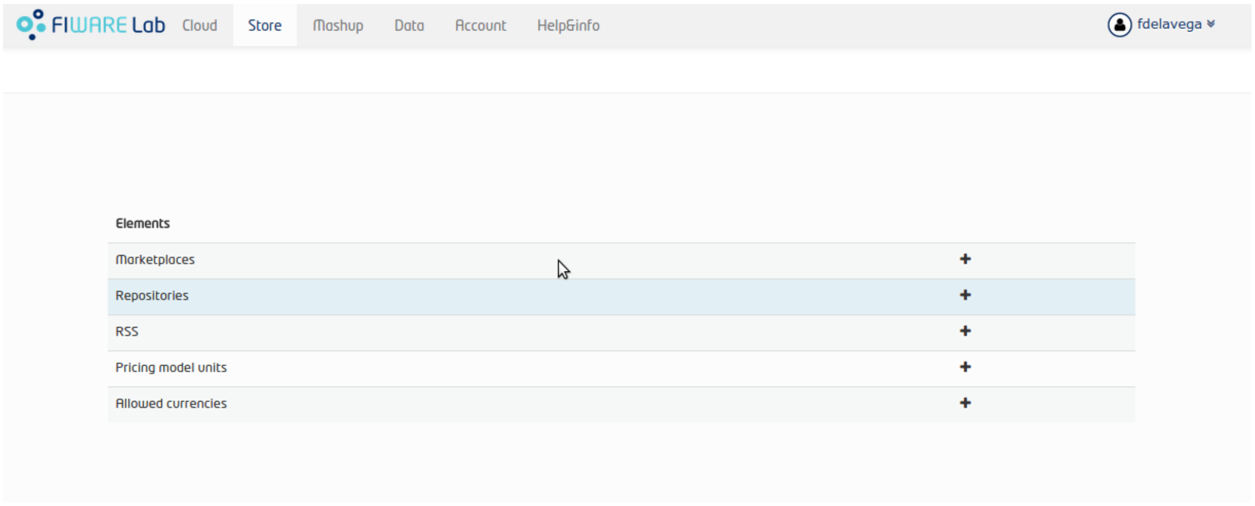
Fill the internal name, the host and the API version (1 or 2) of the Marketplace.



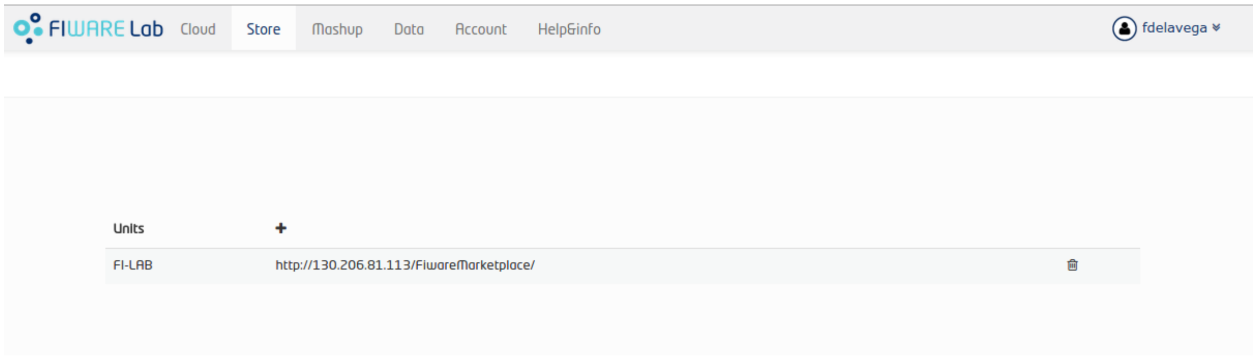
Name	FIWARE Lab
Host	https://marketplace.lab.fiware.org/
API Version	Version 2

Note: Marketplace API version 1 is deprecated

Pressing on the Marketplaces row is possible to view in what Marketplaces WStore is registered on.



Elements	
Marketplaces	+
Repositories	+
RSS	+
Pricing model units	+
Allowed currencies	+

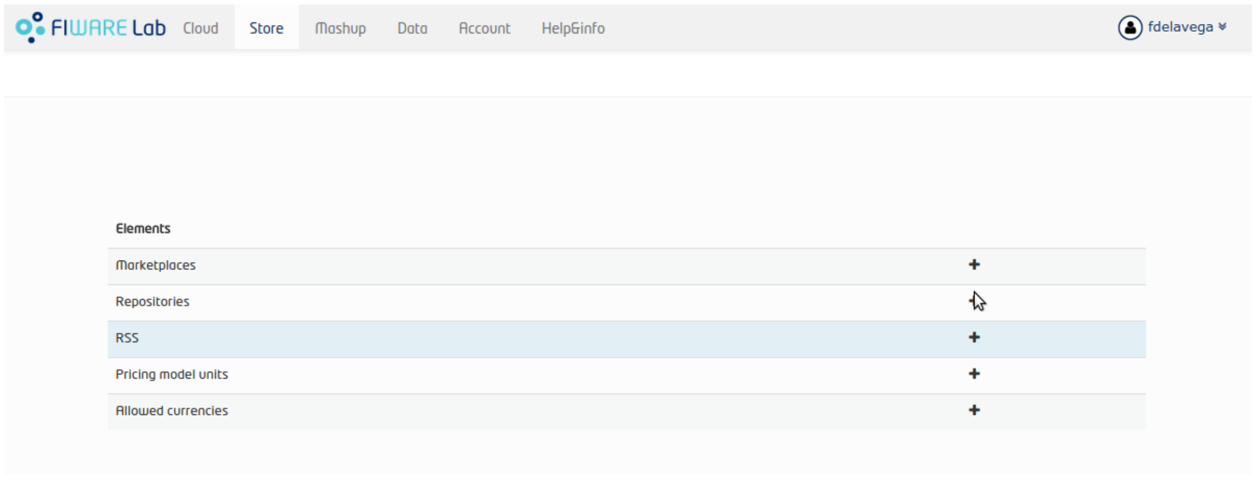


Units	+
FI-LAB	http://130.206.81.113/FiwareMarketplace/

Registering a Repository on WStore

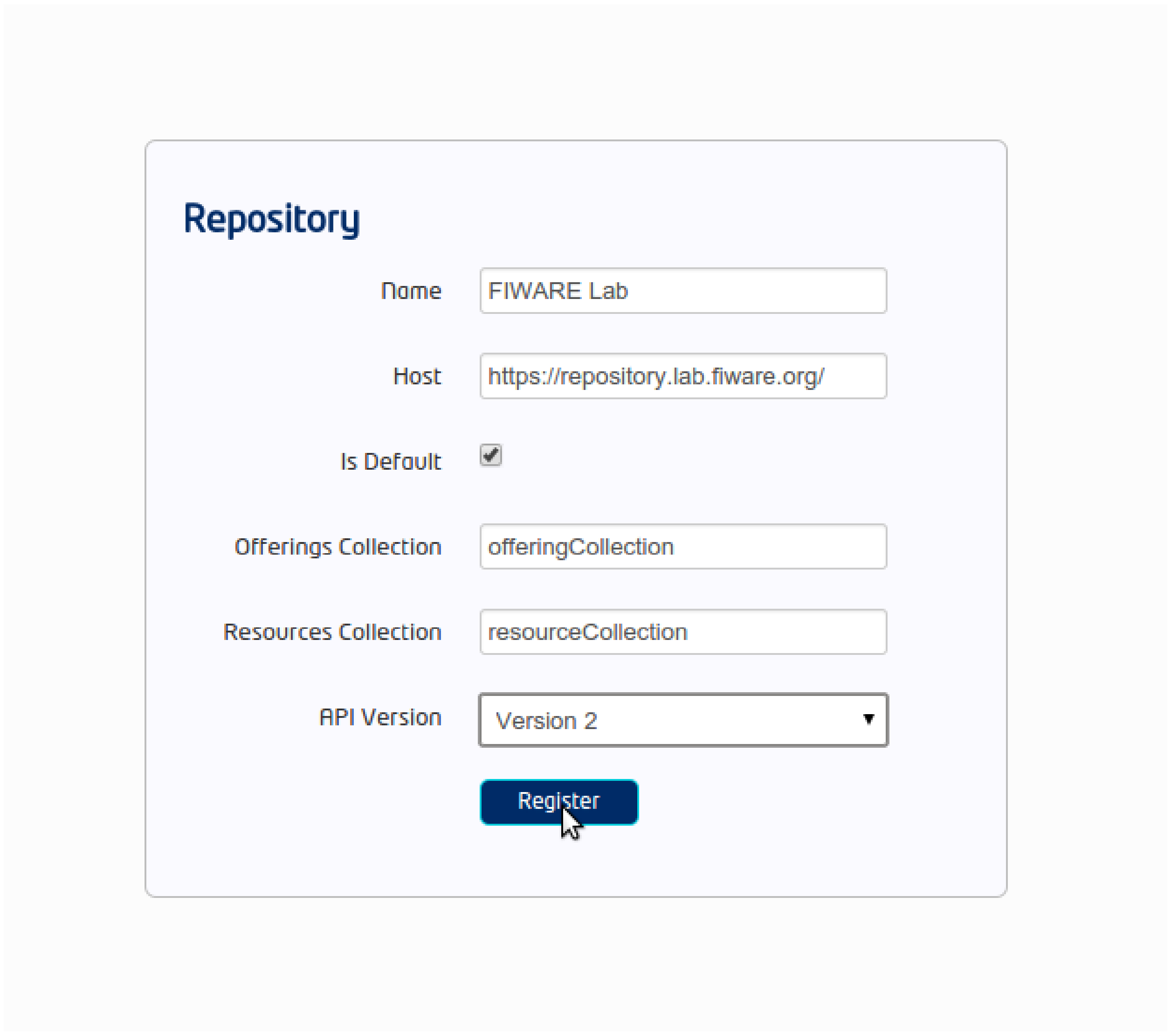
It is possible to register some instances of the Repository GE in order to allow service providers to Upload USDL documents directly when creating an offering.

To register a Repository on WStore select the Administration view and press the Add symbol of the Repositories row.



Elements	
Marketplaces	+
Repositories	+
RSS	+
Pricing model units	+
Allowed currencies	+

Fill the internal name and the host of the Repository. Addtionally, it is possible to specify that the current repository is the default one in WStore by selecting the flag *is default*. Moreover, it is necessary to provide the collections that will be used for storing offering and resource USDL description. Finally, it is required to choose the API version of the repository (1 or 2).

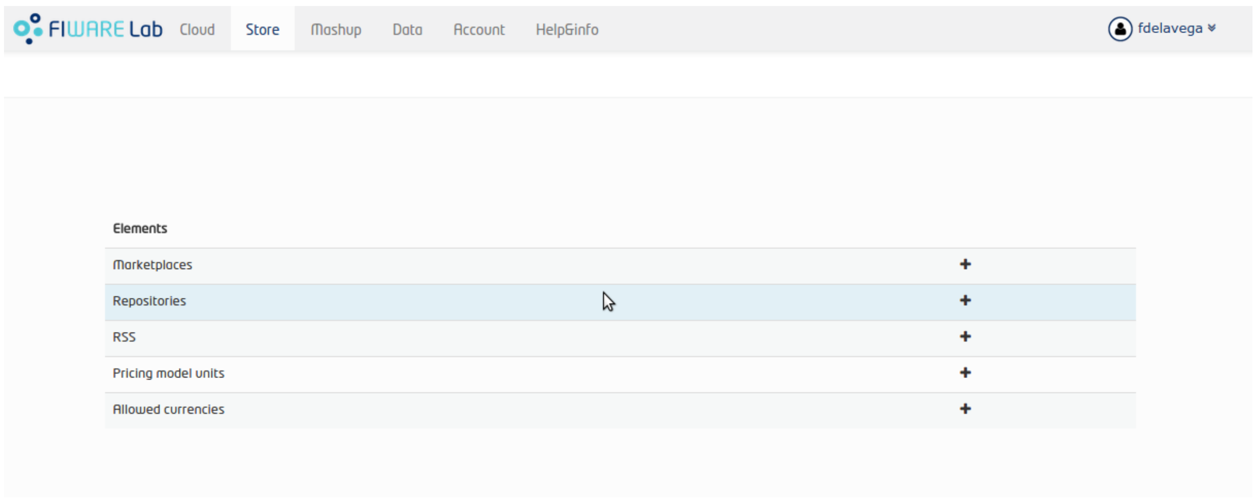


Repository

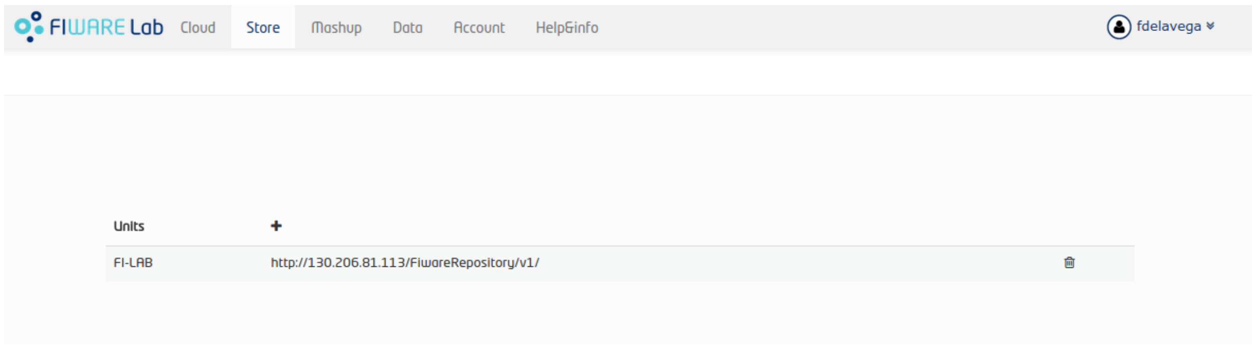
Name	FIWARE Lab
Host	https://repository.lab.fiware.org/
Is Default	✓
Offerings Collection	offeringCollection
Resources Collection	resourceCollection
API Version	Version 2

Note: Repository API version 1 is deprecated

Pressing on the Repositories row is possible to view what Repositories are registered on WStore.



Elements	
Marketplaces	+
Repositories	+
RSS	+
Pricing model units	+
Allowed currencies	+

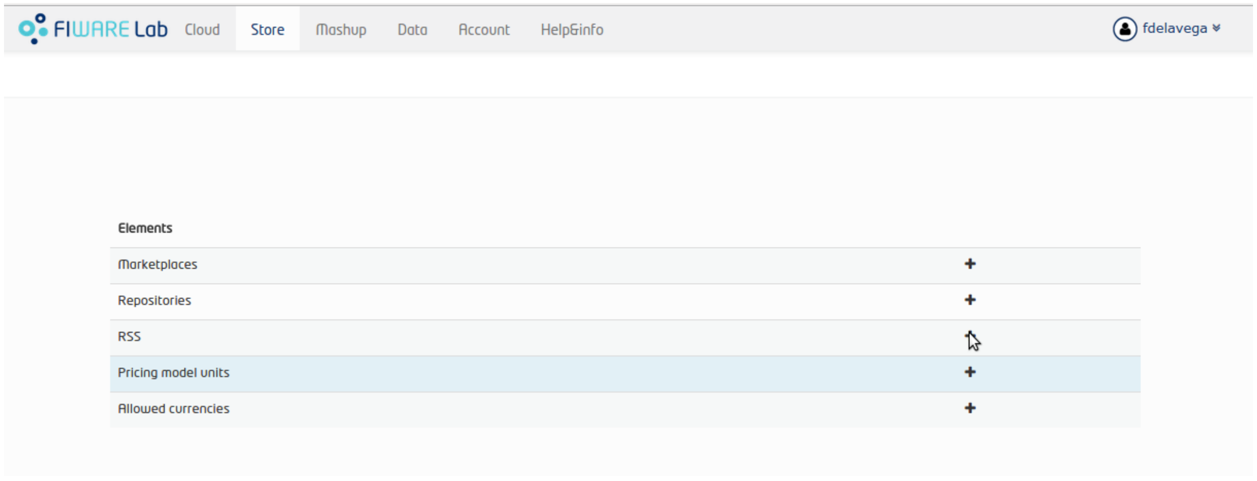


Units	+
FI-LAB	http://130.206.81.113/FiwareRepository/v1/

Registering a RSS on WStore

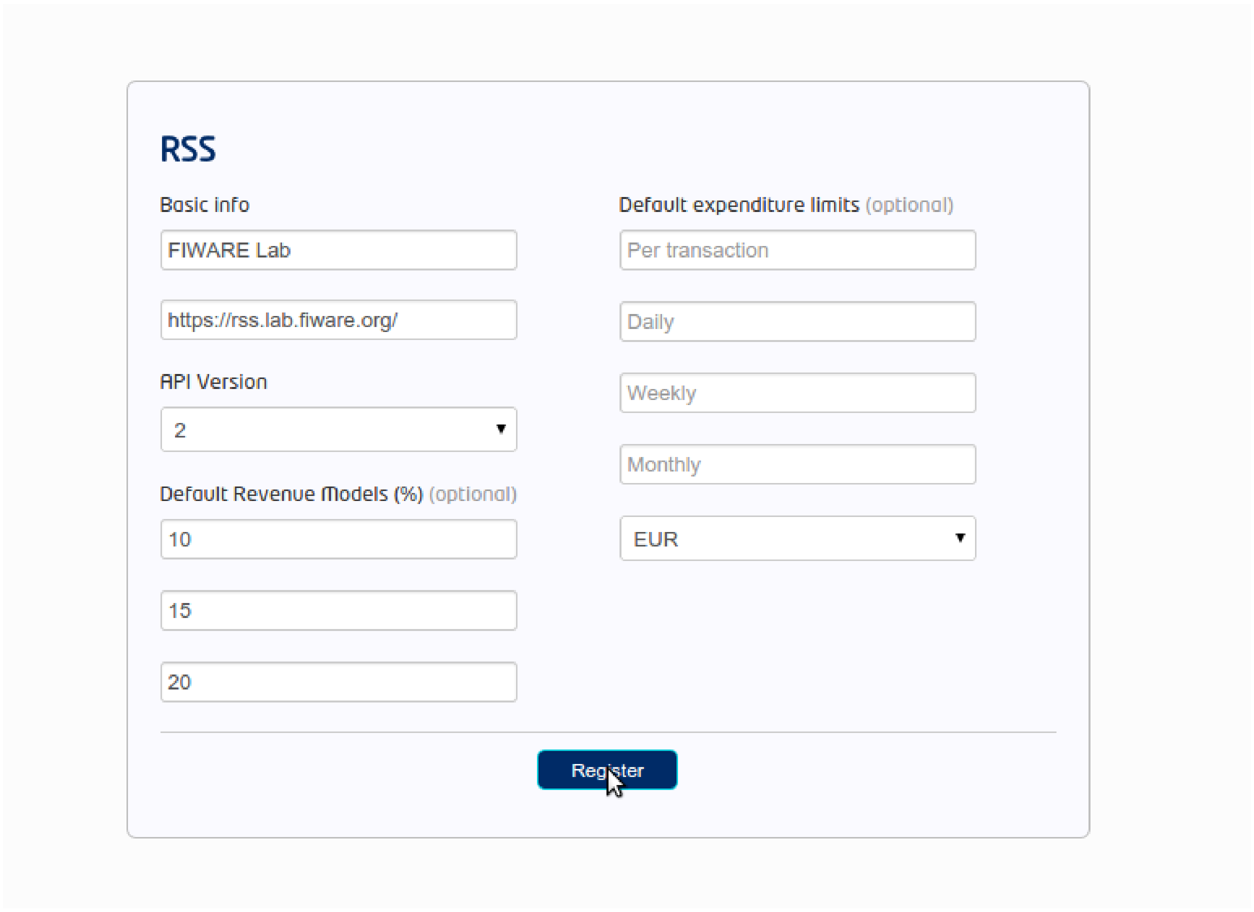
It is possible to register RSS instances on WStore in order to perform the revenue sharing of the purchased offerings.

To register a RSS on WStore select the Administration view and press the Add symbol of the RSS row.



Elements	
Marketplaces	+
Repositories	+
RSS	+
Pricing model units	+
Allowed currencies	+

Fill the internal name, the host and the API version (1 or 2) of the RSS. Then provide the basic revenue models for WStore. This fields are used to specify the percentage of the revenues generated in WStore that belongs to WStore owners as platform providers. It allows to specify a different percentage for offerings containing single payments, subscriptions and pay-per-use respectively. Finally, it is also possible to provide the default expenditure limits for WStore.



RSS

Basic info

FIWARE Lab

https://rss.lab.fiware.org/

API Version

2

Default Revenue Models (%) (optional)

10

15

20

Default expenditure limits (optional)

Per transaction

Daily

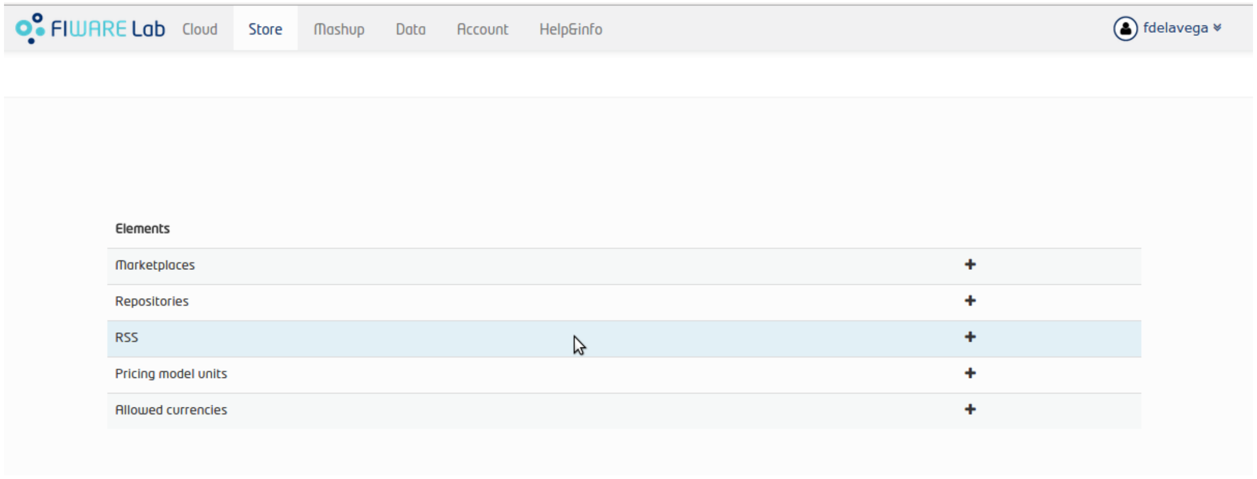
Weekly

Monthly

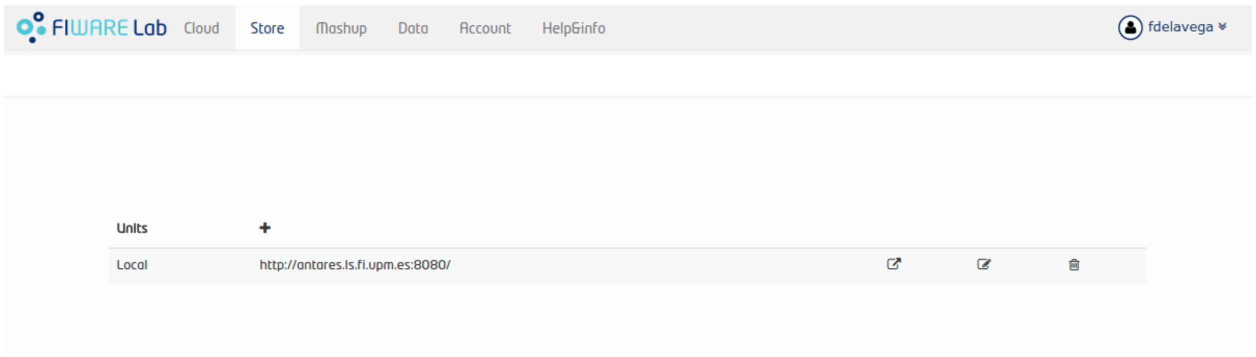
EUR

Note: RSS API version 1 is deprecated

Pressing on the RSS row is possible to view what RSSs are registered on WStore.



Elements	
Marketplaces	+
Repositories	+
RSS	+
Pricing model units	+
Allowed currencies	+

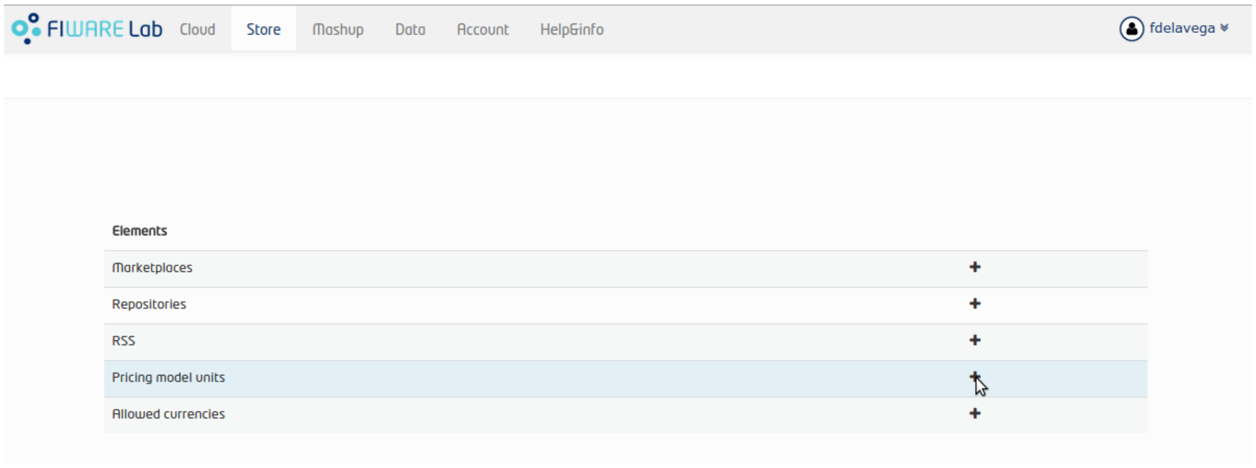


Units	+
Local	http://antares.ls.fi.upm.es:8080/

Registering a Price Unit

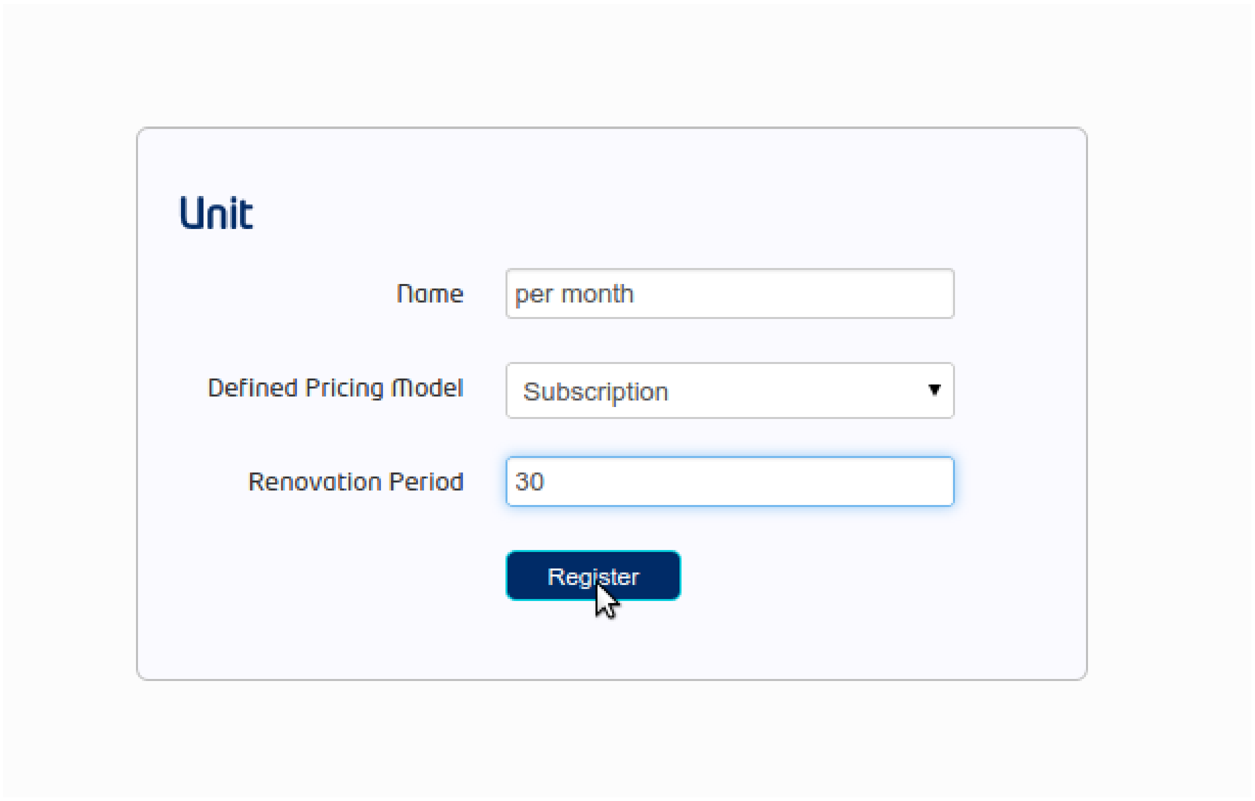
Price Units are used in order to determine the concrete pricing model that applies to an offering.

To include a new supported price unit select the Administration view and click the add symbol in the Pricing model units row.



Elements	
Marketplaces	+
Repositories	+
RSS	+
Pricing model units	+
Allowed currencies	+

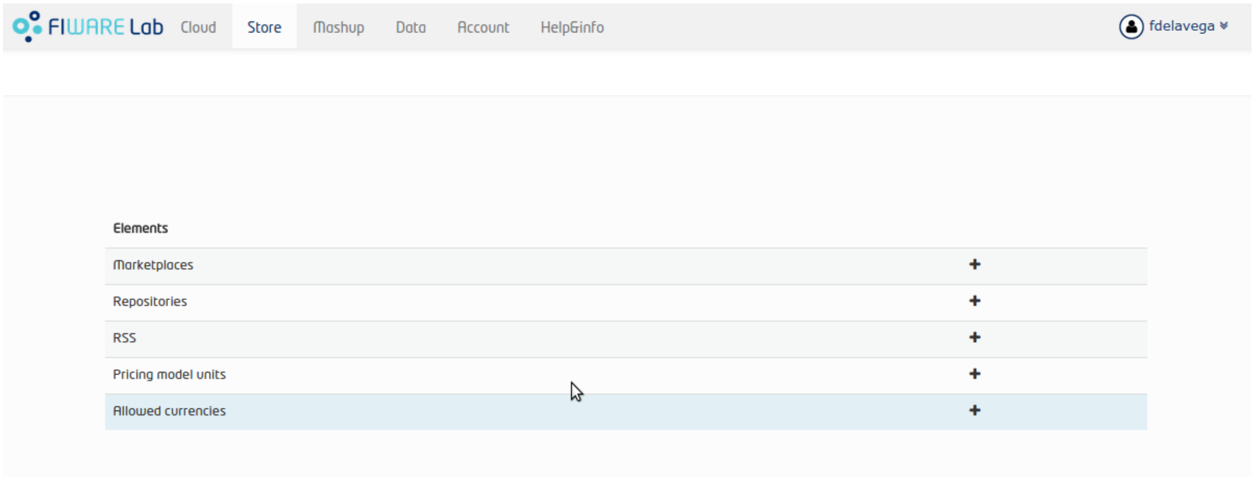
Fill the name and the defined model of the unit. If the defined model is Subscription it is also necessary to specify the renovation period.



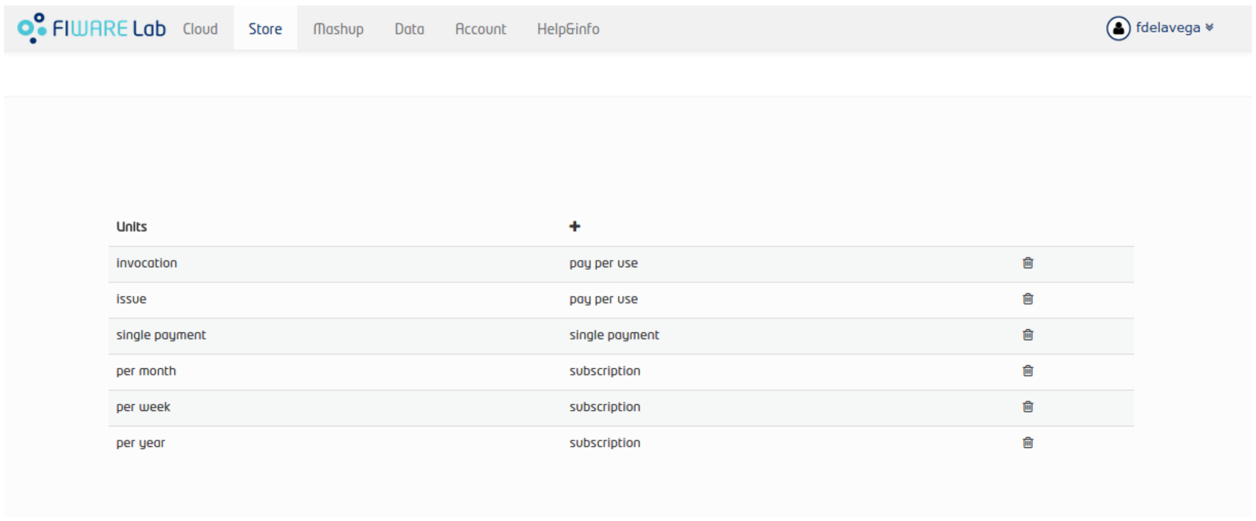
Unit

Name	per month
Defined Pricing Model	Subscription
Renovation Period	30

It is possible to view existing units by click on the Pricing model units row.



Elements	
Marketplaces	+
Repositories	+
RSS	+
Pricing model units	+
Allowed currencies	+

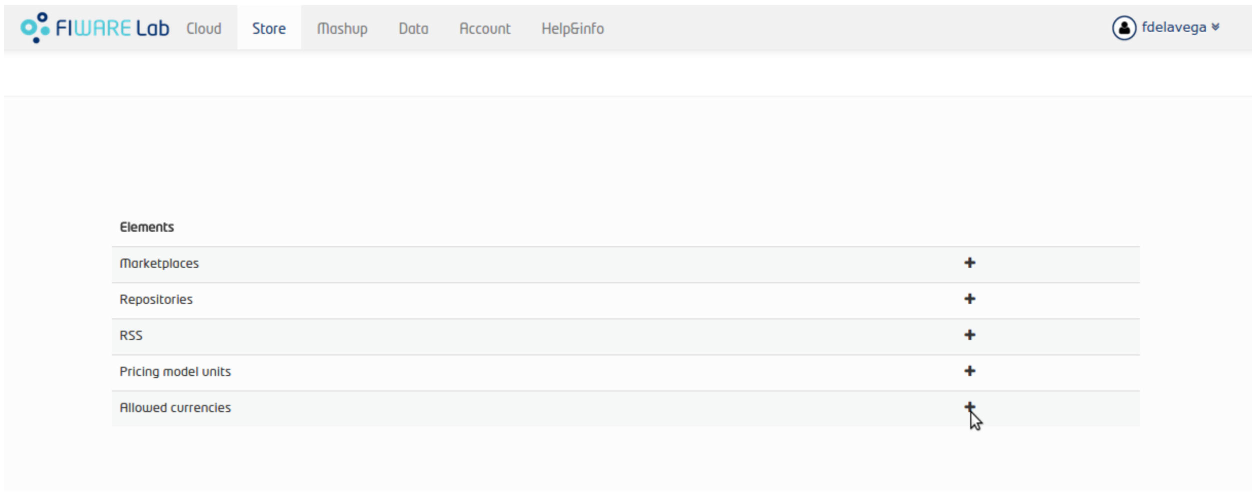


Units	+	
invocation	pay per use	🗑
issue	pay per use	🗑
single payment	single payment	🗑
per month	subscription	🗑
per week	subscription	🗑
per year	subscription	🗑

Registering a Currency

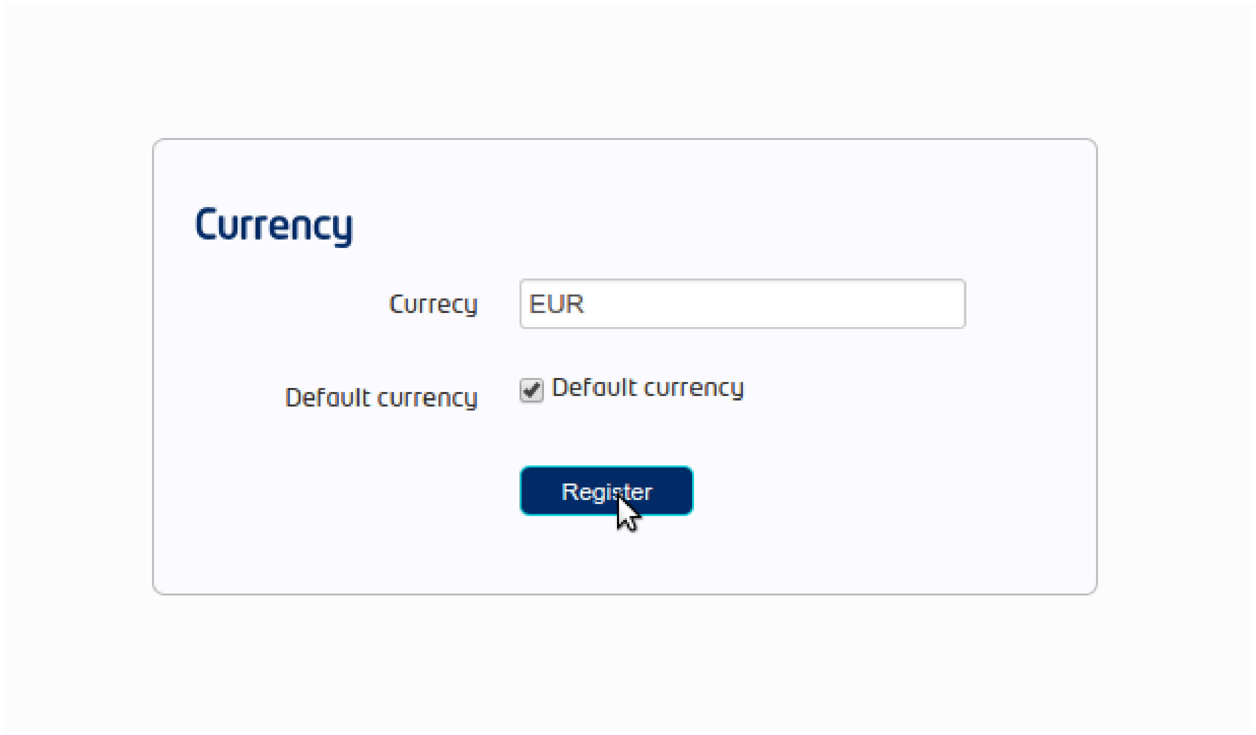
Currencies are used in order to determine what currencies can be used in the pricing model of an offering.

To include a new supported currency select the Administration view and click the add symbol in the Allowed currencies units row.



Elements	
Marketplaces	+
Repositories	+
RSS	+
Pricing model units	+
Allowed currencies	+

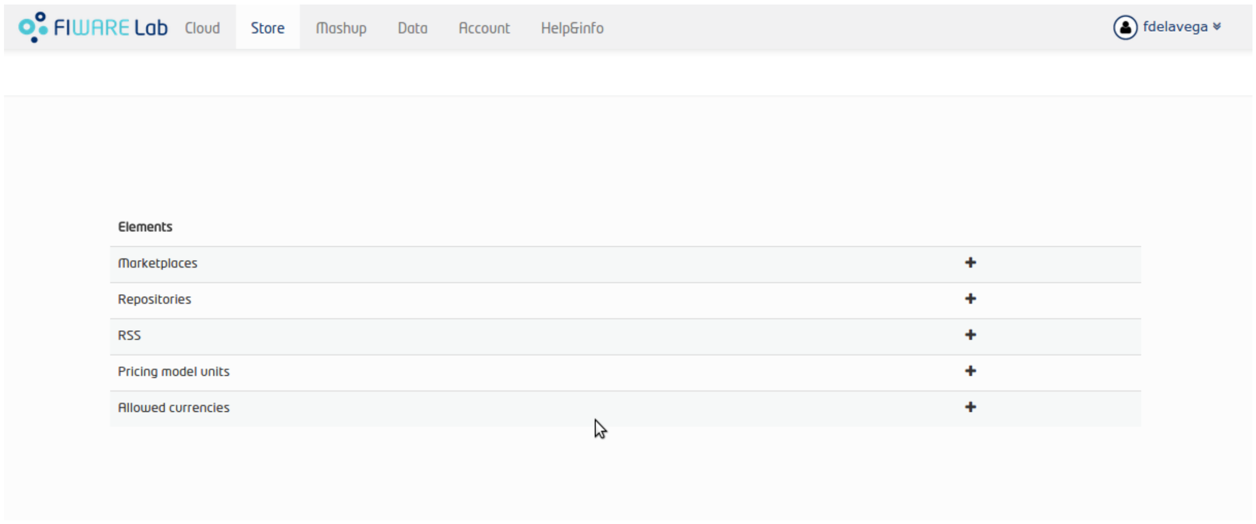
Fill the name and choose whether the currency is the default one or not.



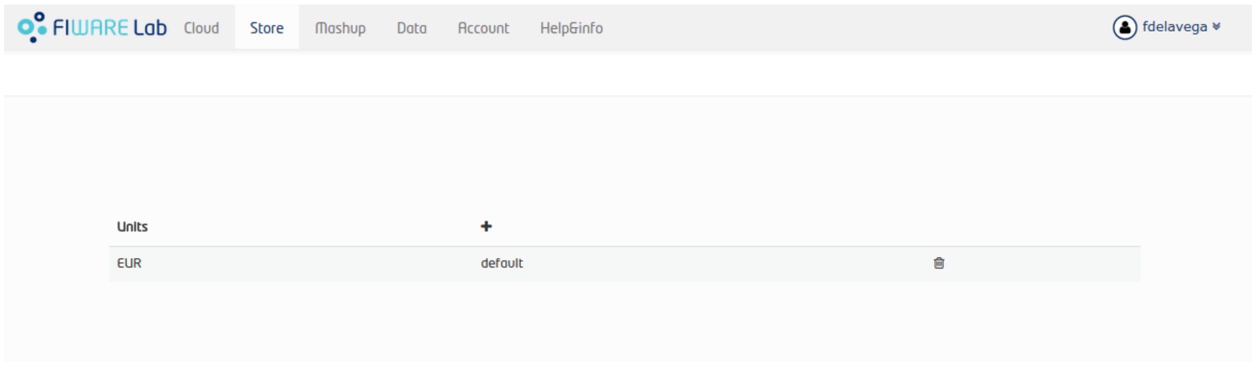
Currency

Currecy	EUR
Default currency	✓ Default currency

It is possible to view existing currencies by click on the Allowed currencies row.



Elements	
Marketplaces	+
Repositories	+
RSS	+
Pricing model units	+
Allowed currencies	+



Units	+	
EUR	default	🗑

1.2.3 Programmer Guide

WStore allows to offer any kind of digital asset. In this regard, some kind of digital assets may require to perform specific actions and validations that require to know the format of the asset. To deal with this problem WStore allows to register types of resources by creating plugins. This section explains how these plugins are created.

Additionally, WStore exposes an API that can be used by developers in order to integrate the monetization features offered with their own solutions. The complete description of this API can be found in:

- Apiary
- GitHub Pages

Plugin Package

WStore plugins must be packaged in a zip. This file will contain all the sources of the plugin and a configuration file called *package.json* in the root of the zip. This configuration file allows to specify some aspects of the behaviuor of the plugin and contains the following fields:

- name: Name given to the resource type. This is the field that will be shown to providers
- author: Author of the plugin.
- formats: List that specify the different allowed formats for providing a resource of the given type. This list can contain the values "URL" and "FILE".
- module: This field is used to specify the main class of the Plugin.
- version: Current version of the plugin.
- overrides: List that specify a set of fields of the resource that will be overriden by the plugin code when creating a resource of the given type. This list can contain the values "NAME", "VERSION" and "OPEN".
- media_types: List of allowed media types that can be selected when creating a resource of the given type
- form: Optional field that can be used to define a form that is displayed to providers in order to retrieve meta information that is required by this specific resource type.

Following you can find an example of a *package.json* file:

```
{
    "name": "Test Resource",
    "author": "fdelavega",
    "formats": ["FILE"],
    "module": "plugin.TestPlugin",
    "version": "1.0",
    "overrides": ["NAME", "VERSION"],
    "form": {}
}
```

The form field allows to specify a concrete form that is rendered and diaplyed to providers when registering a resource of the given type in order to retrieve metadata. In this regard, this field allows to specify text inputs, selects, checkboxes and textareas. The following example shows a configuration file where this form has been filled:

```
{
    "defined_type": "CKAN Dataset",
    "author": "fdelavega",
    "version": "1.0",
    "module": "ckan_dataset.CKANDataset",
    "media_types": [],
    "formats": ["FILE", "URL"],
    "overrides": [],
    "form": {
        "notif": {
            "type": "text",
            "placeholder": "Notification URL",
            "default": "http://data.lab.fiware.org/notify_creation",
            "label": "Notification URL",
            "mandatory": true
        },
        "license" : {
            "type": "select",
            "label": "Dataset license",
            "options": [{
                "text": "Creative Commons",
                "value": "opt1"
            }, {
                "text": "BSD",
                "value": "opt2"
            }]
        },
        "is_private": {
            "type": "checkbox",
            "label": "Is private",
            "text": "Check if the provided dataset is private or not",
            "default": true
        },
        "add_data": {
            "type": "textarea",
            "label": "Additional data",
            "placeholder": "Additional data"
        }
    }
}
```

The source code of the plugin must be written in Python and must contain a main class that must be a child class of the Plugin class defined in WStore. Following you can find an example of a plugin main class.

```
from wstore.offerings.resource_plugins.plugin import Plugin

class TestPlugin(Plugin):
    def on_pre_create_validation(self, provider, data, file_=None):
        return data

    def on_post_create_validation(self, provider, data, file_=None):
        pass

    def on_pre_create(self, provider, data):
        pass

    def on_post_create(self, resource):
        pass

    def on_pre_update(self, resource):
        pass

    def on_post_update(self, resource):
        pass
    def on_pre_upgrade_validation(self, resource, data, file_=None):
        return data

    def on_post_upgrade_validation(self, resource, data, file_=None):
        pass

    def on_pre_upgrade(self, resource):
        pass

    def on_post_upgrade(self, resource):
        pass

    def on_pre_delete(self, resource):
        pass

    def on_post_delete(self, resource):
        pass
```

Implementing Event Handlers

It can be seen in the previous section that the main class of a plugin can implement some methods that are inherited from WStore Plugin class. This methods can be used to implement handlers of the different events of the life cycle of a resource. Concretely, WStore defines the following events:

- **on pre create validation**: This event is raised before WStore validates the information of the resource given by the provider. The main objective of this event is allowing the concrete plug-in to override some information of the resource, which is then validated by WStore. The handler of this event receives the provider, and the raw information given by the provider for the creation of the resource, that may include a file for "FILE" formats. Additionally, the handler of this event must return the processed data of the resource.
- **on post create validation**: This event is raised after the information given by the provider for creating the resource has been validated by WStore. The main objective of this event is allowing plug-ins to override resource information that is not intended to be validated by the Store. The event handler of this event receives the provider and validated information given by the provider.
- **on pre create**: This event is raised before a new resource is saved to the database. This event can be used to modify the resource object before saving it or for executing some tasks required by the concrete type of resource, such as generating some meta data to be saved with resource or performing specific validations. The handler of this event is called passing the provider and the validated data of the resource as parameters.
- **on post create**: This event is raised after a new resource has been saved to the database. The intention of this event is allowing the plug-in to perform some tasks that depend on the complete creation of the resource, for example notifying a server that a resource has been created in case it might be necessary. The handler of this event receives the saved resource object.
- **on pre update**: This event is raised before WStore saves the result of updating the basic info of a resource. This event is intended to allow plug-ins to perform specific validations of the data or override some fields. The handler of this event receives the modified resource object before saving it.
- **on post update**: This event is raised after WStore has saved the result of updating the basic info of a resource. The main objective of this event is allowing plug-ins to execute specific tasks that require the updated resource to have been saved in the database (e.g sending notifications). The handler of this event receives the modified resource object already saved in the database.
- **on pre upgrade validation**: This event is raised before WStore validates the information of a new version of a resource given by the provider. The main objective of this event is allowing the concrete plug-in to override

some information of the new version of the resource, which is then validated by WStore. The handler of this event receives the raw information given by the provider for upgrading of the resource and the resource object.

- **on post upgrade validation**: This event is raised after the information given by the provider for upgrading the resource has been validated by WStore. The main objective of this event is allowing plug-ins to override some information of the new version of the resource that is not intended to be validated by WStore. The event handler of this event receives the validated information given by the provider and the resource object.
- **on pre upgrade**: This event is raised before a new version of a resource is saved to the database. This event can be used to modify the resource object before saving it or for executing some tasks required by the concrete type of resource, such as generating some meta data to be saved with resource or performing specific validations. The handler of this event is called passing the resource object as a parameter.
- **on post upgrade**: This event is raised after a new version of a resource has been saved to the database. The intention of this event is allowing the plug-in to perform some tasks that depend on the upgrade of the resource, for example notifying a server that a resource has been upgraded. The handler of this event receives the saved resource object.
- **on pre delete**: This event is raised before WStore deletes a resource. This event is intended to allow plug-ins to perform specific tasks and validations before a resource is removed. In this regard, a concrete resource type might require some actions to have been tackled by the provider of the resource before allowing a deletion. The handler of this event receives the resource object to be deleted.
- **on post delete**: This event is raised after WStore has deleted a resource. The main objective of this event is allowing plug-ins to execute specific tasks that require the resource to have been deleted from the database (e.g sending notifications). The handler of this event receives a copy of the deleted resource.

Managing Plugins

Once the plugin has been packaged in a zip file, WStore offers some management command that can be used to manage the plugins.

When a new plugin is registered WStore automatically generates an id for the plugin that is used for managing it. To register a new plugin the following command is used:

```
python manage.py loadplugin TestPlugin.zip
```

It is also possible to list the existing plugins in order to retrieve the generated ids:

```
python manage.py listplugins
```

To remove a plugin it is needed to provide the plugin id. This can be done using the following command:

```
python manage.py removeplugin test-plugin
```