gateway4labs Documentation

Release 0.1

gateway4labs team

March 21, 2016

Contents

Welcome to the gateway4labs project documentation. Gateway4labs is a Go-Lab initiative on which relies its Smart Gateway for integrating external laboratories.

(Want to propose a nicer logo?)

Table of Contents

General

This section should be understandable by everybody. It should reflect the rationale, the features, but should not contains deployment, etc

1.1 What is it?

gateway4labs is a pragmatic approach targeting the integration of multiple remote laboratories in different digital learning environments (Learning Tools) -such as Learning Management Systems (LMSs), Content Management Systems (CMSs) or Personal Learning Environments (PLEs)-.

1.2 Vocabulary

This section just defines the terms used in this documentation. These terms should be updated so as to fill the GOLC terminology, and whenever they are changed, the whole document should change. So as to start with something, we'll define a basic set of concepts, sorted alphabetically:

Laboratory A laboratory represents the conceptual place where students will go to use an assigned physical equipment. If students wants to access a *Robot laboratory*, they'll request that laboratory and whenever they can access (after booking or queueing), they'll start using a particular copy of it (called Rig, and its definition is outside the scope of this document).

Learning Management System (LMS) It is a software application for the administration, documentation, tracking, and reporting of training programs, classroom and online events, e-learning programs, and training content. Extracted from the wikipedia. Examples: Moodle, .LRN, ilias, etc.

Remote Laboratory Management System (RLMS) Management software system that manages different remote laboratories. A RLMS can support multiple remote laboratories (such as a *Robot laboratory* and an *Electronics laboratory* at the same time). RLMSs provide authentication, authorization, user management, user tracking, and scheduling, as well as APIs to develop new laboratories on top of them. Examples: WebLab-Deusto, MIT iLabs, Labshare Sahara.

1.3 Design overview

1.3.1 Principles and decisions

There are few principles:

1. Supporting a Learning Tool must be simple and easy.
2. The solution must be secure. A student should never be able to open a lab session unless the Learning Tool grants him access to it or unless the lab provides open access to it.
3. It must work with Learning Tools in a variety of situations, from universities supporting Single Sign-On solutions -such as Shibboleth- or directory protocols -such as LDAP- to secondary schools with almost no IT infrastructure.
4. IT services of entities deploying gateway4labs must be able to audit quickly whatever involves the Learning Tools.
5. It must support multiple remote laboratories, so they can collaborate towards the same shared goal. While some code must be remote laboratory dependent, most of the code should be common.

Given that multiple Learning Tools are targeted, ideally there should be no Learning Tool dependent code. There are approaches that try to achieve this. However:

- Some of them do not guarantee that any malicious user can open a lab session without the Learning Tool supporting it. This is in conflict with principle number 2.
- Some of them rely on protocols not yet supported by existing Learning Tools, such as relying exclusively on Shibboleth or IMS LTI. This is in conflict with principles number 1 (if the Learning Tool does not support it, including it becomes complex), 2 and 3 (if the secondary school does not support that protocol, the solution will not work). Additionally, given that these protocols do not know what the Learning Tool itself states, certain common situations are not covered. For instance, instructors may want to establish which weeks lab sessions are available and which ones are not.

Therefore, we accept that there is some Learning Tool dependent code as the lesser of two evils, and only if it is required. However, so as to support the principles mentioned above, this code:

- Should be as small and simple as possible: IT services must be able to audit it quickly.
- Should not require upgrades too often: that will typically involve critical, slow processes by the IT services.

Most of the logic should therefore be located in other component. This component has been named *labmanager*, as detailed in the following section.

1.3.2 Roles

In gateway4labs there are the following roles:

1. Remote laboratory administrator
2. LabManager administrator
3. LMS/CMS/PLE administrator
4. Teacher
5. Student

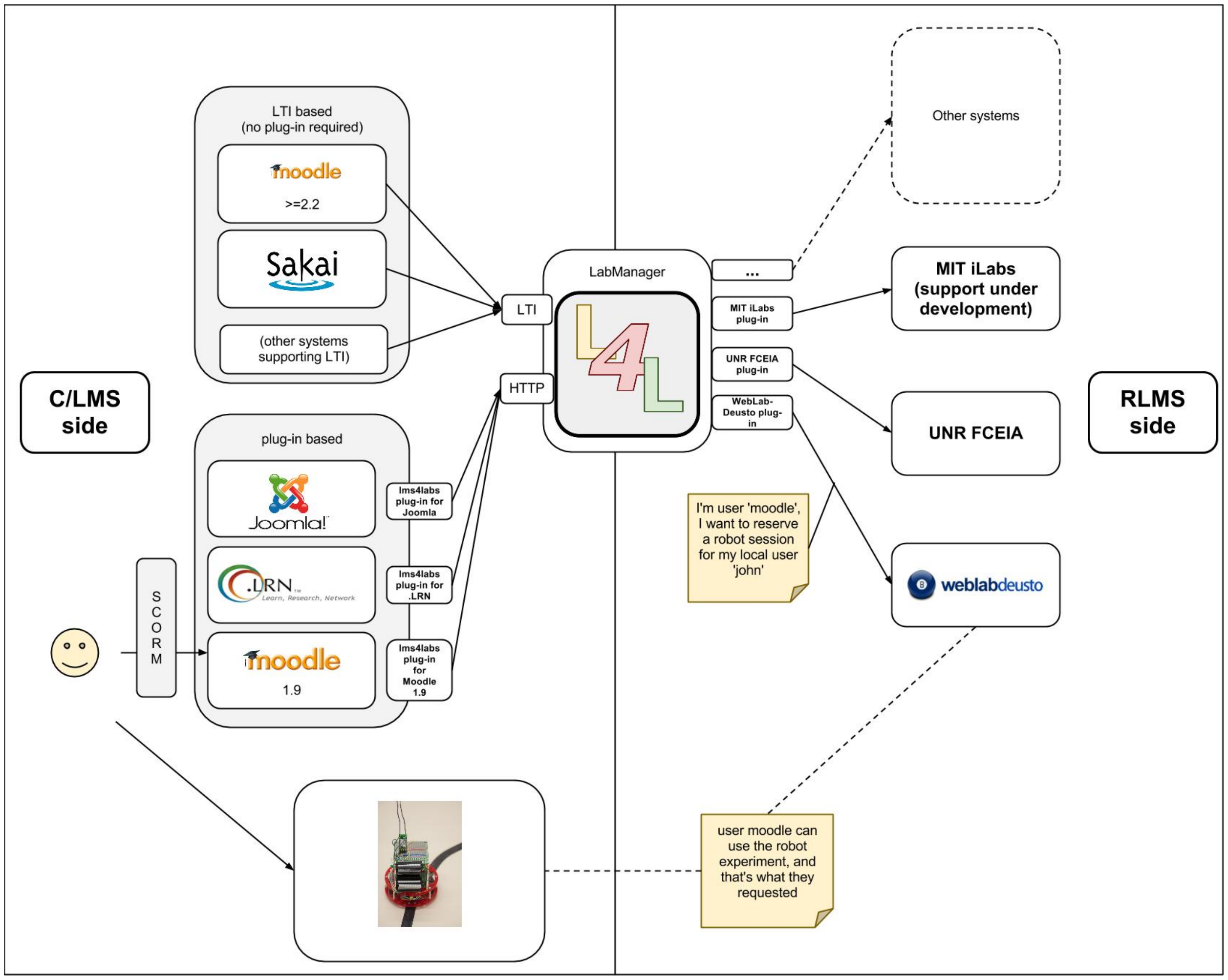
The remote laboratory administrator will create an account for each LabManager involved, and it will grant privileges for each LabManager.

The LabManager administrator will interact with the remote laboratory administrators to make sure that they are connected to the LabManager. He will also sign up different Learning Management Systems from different entities.

The LMS/CMS/PLE administrator will manage from the LMS/CMS/PLE to students. There are two versions at this moment: * When using LTI, the LMS/CMS/PLE administrator will create teacher users in the LabManager, and assign them permissions on laboratories. Teachers will use these permissions (which contain a key and a secret unique for each teacher and laboratory) to add those labs to their courses. Students of those courses will automatically be able to use these laboratories.

- When using Basic HTTP, the LMS administrator will query courses in the LMS/CMS/PLE, and they will assign permissions to those courses. From that point, teachers will be able to upload SCORM objects refering to those courses and upload them to the LMS/CMS/PLE. From that point, students of those courses will be able to use them.

1.3.3 Architecture overview


LTI based
(no plug-in required)
moodle
>=2.2
Sakai
(other systems supporting LTI)
C/LMS side
plug-in based
Joomla!
.LRN Learn, Research, Network
moodle
1.9
S C O R M
lms4labs plug-in for Joomla
lms4labs plug-in for .LRN
lms4labs plug-in for Moodle 1.9
LTI
HTTP
LabManager
...
MIT iLabs plug-in
UNR FCEIA plug-in
WebLab-Deusto plug-in
Other systems
MIT iLabs (support under development)
UNR FCEIA
weblabdeusto
RLMS side
I'm user 'moodle', I want to reserve a robot session for my local user 'john'
user moodle can use the robot experiment, and that's what they requested

As described in the figure above, there are three main components involved:

1. The LMS/CMS/PLE (left side). If it supports IMS LTI, it will use it and no code will be required in the LMS/CMS/PLE. If it does not support it, it will have a small plug-in that communicates with the LabManager.
2. The LabManager, which will receive requests from multiple, different LMSs and it will understand the protocols of different RLMSs. It does not have any LMS dependent code, but it has RLMS dependent plug-ins.
3. The RLMS, which will support a federation protocol to process requests from the LabManager. The federation protocol from one RLMS to other will be different. It should not require anything special for being supported by the LabManager.

This way, if a new LMS/CMS/PLE is aimed, if it supports IMS LTI the support is automatic. If it does not support it, a new plug-in for that LMS is required in the LMS, but it has no impact on the rest of the RLMSs neither on the LabManager. If a new RLMS is aimed, a new plug-in for that RLMS is required in the LabManager, but it has no impact on the LMSs/CMSs/PLEs.

1.3.4 LMS to LabManager protocol

This only applies when the non IMS LTI version is targeted. Sample reservation request:

```
POST /gateway4labs/labmanager/requests/ HTTP/1.0
Authorization: Basic ASDFASDF (LMS token)

{
   "user-id"    : "jsmith",
   "full-name"  : "John Smith",
   "is-admin"   : true,
   "user-agent" : "Mozilla/5.0 (X11; Ubuntu; Linux x86_64; rv:12.0) Gecko/20100101 Firefox/12.0",
   "origin-ip"  : "192.168.1.1",
   "referer"    : "http://.../",
   "courses"    : {
        "01"    : "s",
        "02"    : "s",
        "03"    : "t", // "t" = teacher, "s" = student
        "04"    : "s",
   },
   "request-payload" : "SOMETHING-THAT-SCORM-SENDS"
}
```

Sample authentication request:

```
GET /gateway4labs/lms/authenticate HTTP/1.0

POST /gateway4labs/labmanager/lms/admin/authenticate/ HTTP/1.0
Authorization: Basic ASDFASDF (LMS token)

{
    "full-name" : "John Smith"
}
```

Sample course listing request (q=text to filter, start=0 to go to the first page):

```
GET /gateway4labs/lms/list?q=elect&start=0 HTTP/1.1
Authorization: Basic ASDFASDF (LabManager token)
```

Sample course listing response:

```
{
   "start"    :   150,
   "number"   : 34000,
   "per-page" :    10,
   "courses" : [

     {
        "id"   : "3465",
        "name" : "Computers Architecture"
     },
     {
                "id"   : "2854",
                name"  : "Electronics Laboratory"
     },
     {
        "id"   : "2854",
        "name" : "IEEE Student Branch"
     },
   ],
}
```

CHAPTER 2

User's Guide

This section should contains how to deploy each system. For the LabManager, what options are there available, etc., and how to connect each component.

2.1 Labmanager administrators

2.1.1 Installation

Within the gateway4labs project, two main components need to be installed:

- The LabManager, which is the software system
- The LMS plug-in (only if a LMS that does not support LTI is installed)

Cardinality:

- There should be a LabManager representing each university or secondary school.
- While uncommon, there could be more than one LMS for each LabManager
- Each LabManager will support more than one RLMS
- Each RLMS must be prepared to support more than one LabManager

LabManager installation

The LabManager has been implemented in Python, so the first dependency to be installed is Python itself. In OS X and Linux, you usually have it already installed. In Windows, you should go to the Python website and download the latest 2.x version (e.g. 2.7).

The LabManager also uses the Flask microframework, and sqlalchemy to wrap the database, being able to use multiple systems but we have only tested MySQL and sqlite. If you want to use MySQL, you'll have to install it (in Windows, you may download it from the MySQL website or use the XAMPP package, which also comes with Apache; in Linux systems you can install it using your package manager -e.g. sudo apt-get install mysql-server -).

So at this point, the following software packages are assumed:

- Python 2.x (do not use Python 3.x; it is not yet supported)
- MySQL (unless you prefer using sqlite)

In order to deploy it, some Python packaging notions are required, explained in the first section. Then, the deployment itself is detailed for Microsoft Windows and Linux systems. Finally, notes on the development are described.

Notes on pip and virtualenv

Python open source packages are usually uploaded to PyPI (commonly refered to as the cheese shop), and tools such as easy_install and pip make it easy to query, search and install those packages. During this document we'll use pip, which is indeed a replacement for easy_install.

In order to install pip, we'll use our distribution package manager in Linux systems. For instance, in Ubuntu we can simply run:

```
$ sudo apt-get install python-pip
```

On Windows systems, the process is slightly longer since we have to install first setuptools. So download the distribute_setup.py file and run it, and then place the Python installation Script directory to the PATH environment variable. So append the following to the PATH variable:

```
;C:\Python27\Scripts
```

Once setuptools is installed, type the following on CMD:

```
easy_install pip
```

From this point, you'll have pip running in your Windows system.

When installing Python packages, by default they are all installed in a system-wide location. However, for different projects we might be interested in installing different versions of the same libraries. In order to avoid conflicts, and manage the installed libraries in an easy way, the virtualenv project was created.

With virtualenv, it is possible to create a virtual environment in a directory, where one can install a certain set of packages with particular versions. All those versions are managed in that particular directory, so you can later delete it, upgrade only that one, and especially, create other environments for other applications.

Warning: In some scenarios, using virtualenv causes certain problems (when trying to instantiate a virtual environment fails, etc.). While using virtualenvs is highly recommended, it is still optional, so if you're running into problems and you can't advance, you can go ahead and install the libraries in your system directly without any virtualenv.

The way to use it is very simple. First you need to install virtualenv, using pip:

```
$ pip install virtualenv
```

Or using your package manager in Linux systems:

```
$ sudo apt-get install python-virtualenv
```

And then, you can create an environment by running:

```
$ virtualenv --no-site-packages env1
New python executable in env1/bin/python
Installing distribute........done.
Installing pip...............done.
```

At this point, the environment has been created, but it is not yet being used. In order to start using this environment, we have to do the following on Linux and OS X:

```
$ . env1/bin/activate
```

Or the following on Windows:

```
> env1\scripts\activate
```

From this point, you'll see that in the prompt of your shell there is an indicator such as *(env1)*. At this point, we will be working with that environment. So if we install Flask:

```
$ pip install Flask
```

It will be installed in that isolated virtual environment. We can test it by running Python and checking that Flask is actually installed:

```
$ python
Python 2.7.2+ (default, Oct  4 2011, 20:06:09)
[GCC 4.6.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import flask
>>>
```

If we go out of the Python shell (Ctrl + D / Ctrl + Z), and we deactivate the environment:

```
$ deactivate
```

Or we simply open a new terminal, then we'll see that we are not using that environment anymore:

```
$ python
Python 2.7.2+ (default, Oct  4 2011, 20:06:09)
[GCC 4.6.1] on linux2
Type "help", "copyright", "credits" or "license" for more information.
>>> import flask
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ImportError: No module named flask
>>>
```

To start using it again, we only have to call or import the activate script again.

Notes on WSGI

WSGI stands for Web Server Gateway Interface, which is an interface that different Python web application providers will use and they can automatically be integrated in other web servers. For instance, there is a WSGI module for Apache or for nginx, so any application developed in a WSGI-compliant framework (such as Flask) can be deployed in those web servers. There is plenty of information and links about the support in the WSGI official site.

Lms4labs has been developed using Flask, which is WSGI-compliant microframework. Therefore, a WSGI-compliant server is required. There are two approaches:

1. Use Apache, nginx, IIS or any other well known web server. There is plenty of documentation on how to deploy Flask applications on those environments in the Flask documentation.
2. Use a Python WSGI-compliant web server such as cherrypy. The advantage of this is that it does not require you to deploy any additional plug-ins to the web server you are already using, and then you can use that server directly or the proxy module of the web server to manage the connections. This approach might be slower, but it is useful to test the system and even to use it in production with a small number of students.

This document covers both approaches, but it is important to understand the benefits and drawbacks of each one.

Deploying Lms4labs

In this section, it is assumed that you already have installed pip and virtualenv, and that you have notions of how you want to deploy the Lms4labs application.

First of all, download the source code of the gateway4labs project and go to the labmanager code:

```
$ git clone https://github.com/gateway4labs/labmanager/
$ cd labmanager
```

Then, create an environment called *env* in the same directory where the labmanager is installed, and activate it:

```
$ virtualenv --no-site-packages env
$ . env/bin/activate
(or, on Windows)
$ . env\scripts\activate
```

Install all the requirements. They are detailed requirements.txt file, so you can install them all by running:

```
$ pip install -r requirements.txt
```

Warning: In Microsoft Windows, some libraries are not installed automatically unless you have installed a proper development environment. If you're running Microsoft Windows, you need to download and install manually (if you've installed Python 2.7 for 32 bits, you'll need to install the file that is called *whatever-win32-python2.7.exe* or so):

- lxml
- PyCrypto
- PyYAML

Once installed, then run the command:

```
C:\...\> pip install -r requirements.txt
```

At this point, everything is ready to be deployed. First, we should add the configuration file. A sample one is distributed, so you can copy it:

```
$ cp config.py.dist config.py
```

And modify it so as to fit your local data. If the engine is sqlite, you don't need to worry about the connection DB configuration (username, password, hostname, etc.). If you are using MySQL, you don't need to create the user and the database by your own, since that is managed by the deployment script itself. Just check that you're fine with the credentials you're going to establish in the config.py file. Then you can create the database by running:

```
$ python deploy.py -cdu
```

Finally, you can test it by running:

```
$ python run.py
```

If you open http://localhost:5000/ with your web browser, you should see the system up and running in development mode. You'll be able to use the username *admin* and the password *password*.

Development

The development mode is a Flask mode used during the application development. By running:

```
$ python run.py
```

You are using that mode. It is a risky mode since users might be able to execute random code in the server, so use it only while developing or testing a particular condition.

While using the development mode, the application will be automatically reloaded every time you modify any code file, and if an exception is raised, you'll be able to see the complete trace and even evaluate conditions through the web browser by writing Python code in any stack level. To see further information, please refer to the official flask documentation.

Production

In order to run the system in production, there are two ways, as previously detailed. The easiest mode is to rely on a Python web server such as cherrypy. A very simple example is provided in the run_cherry.py script, which basically does the following:

```
from cherrypy import wsgiserver
from labmanager import app

PORT = 8080
server = wsgiserver.CherryPyWSGIServer(('0.0.0.0', PORT), app)
server.start()
```

This code is enough for deploying a threaded HTTP server. So as to run it, you must run:

```
$ python run_cherry.py
```

If you want to work with this server behind an Apache server, you can still use the Apache mod_proxy module, which comes by default with Apache. Refer to the Apache documentation for details, but this is an example of configuration (once the module has been enabled):

```
ProxyVia On
ProxyPass        /gateway4labs http://localhost:8080/gateway4labs
ProxyPassReverse /gateway4labs http://localhost:8080/gateway4labs
```

The other approach is using WSGI in the web server. Refer to the Flask documentation on how to deploy it. In the particular case of Apache, the documentation on how WSGI works on Apache is also very good.

As a summary for the deployment on Apache: first, download mod_wsgi. In Linux systems, it may be available in the package repositories (e.g. in Ubuntu, you may install the *libapache2-mod-wsgi* package). In Windows, the process is documented here. Once mod_wsgi is installed in Apache, the following configuration may work:

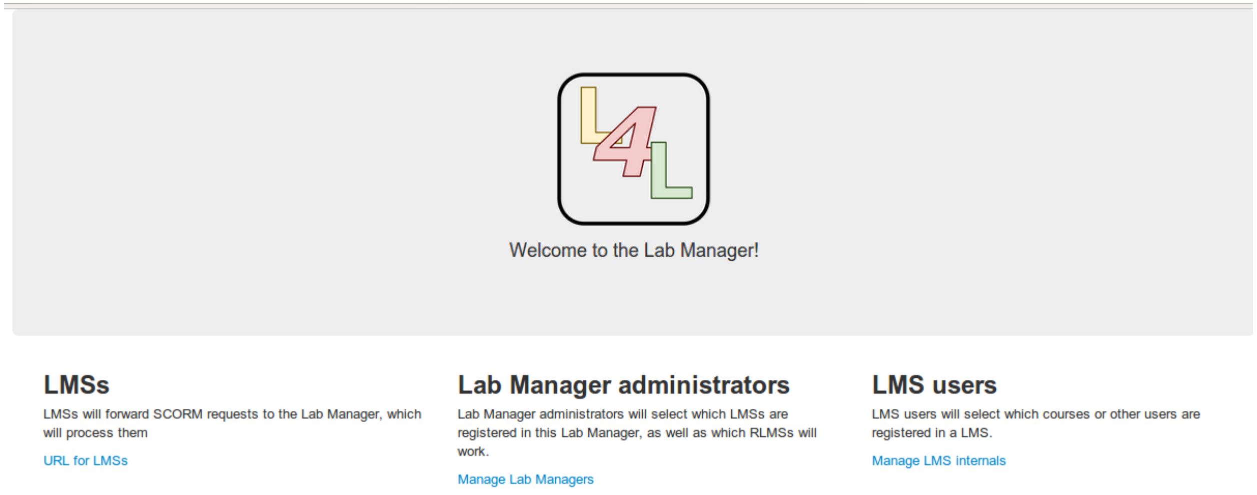
```
WSGIDaemonProcess labmanager user=weblab group=weblab threads=5 python-path=/PATH/TO/ENV/lib/pythonVE
WSGIScriptAlias /labmanager /PATH/TO/labmanager/run_wsgi.wsgi
WSGIRestrictStdout Off
WSGIPassAuthorization On

<Directory /PATH/TO/labmanager/>
    WSGIProcessGroup labmanager
    WSGIApplicationGroup %{GLOBAL}
    Order deny,allow
    Allow from all
</Directory>
```

Being /PATH/TO/labmanager/ the labmanager root project. Additionally, you will need to modify the *run_wsgi.wsgi* script to change the project directory.

2.1.2 Usage: LabManager administrator

The LabManager administrator can do three different things: administrating which remote laboratories are available from it, administrating which LMSs can access it, and assigning remote laboratories to each LMS.

So as to enter in the LabManager, click on the link in the middle in the front page:



Welcome to the Lab Manager!

LMSs

LMSs will forward SCORM requests to the Lab Manager, which will process them

URL for LMSs

Lab Manager administrators

Lab Manager administrators will select which LMSs are registered in this Lab Manager, as well as which RLMSs will work.

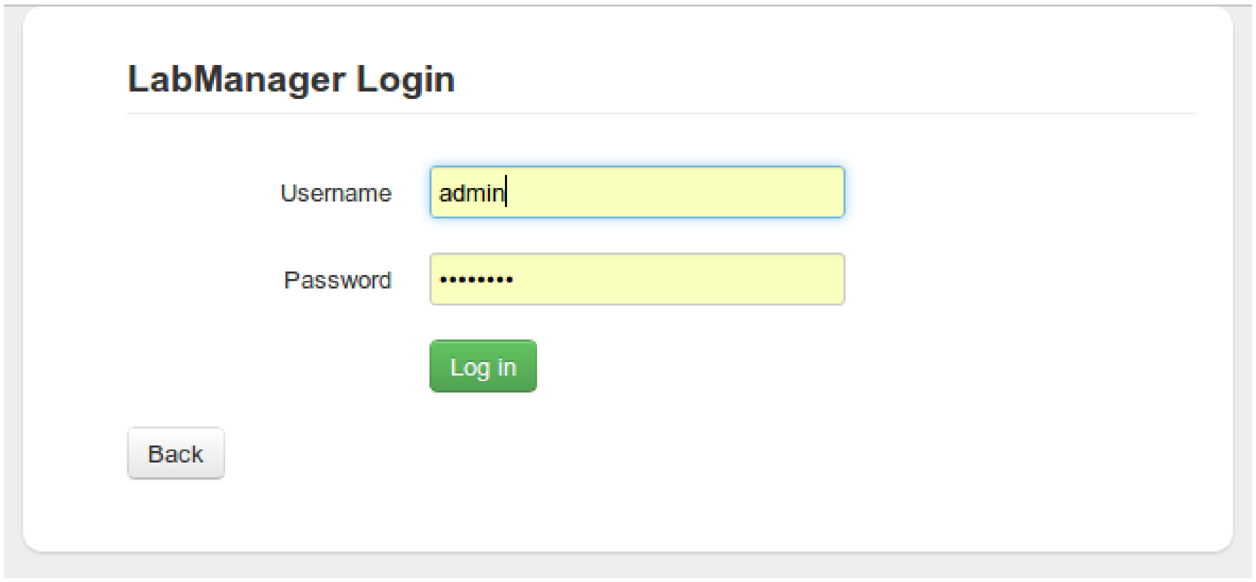
Manage Lab Managers

LMS users

LMS users will select which courses or other users are registered in a LMS.

Manage LMS internals

And then log in as a LabManager administrator (default credentials: *admin* as username and *password* as password).

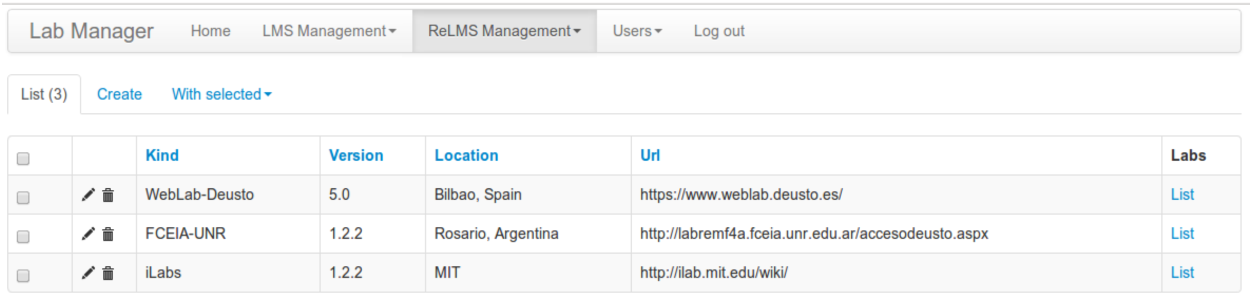


LabManager Login

Username	admin
Password	••••••••

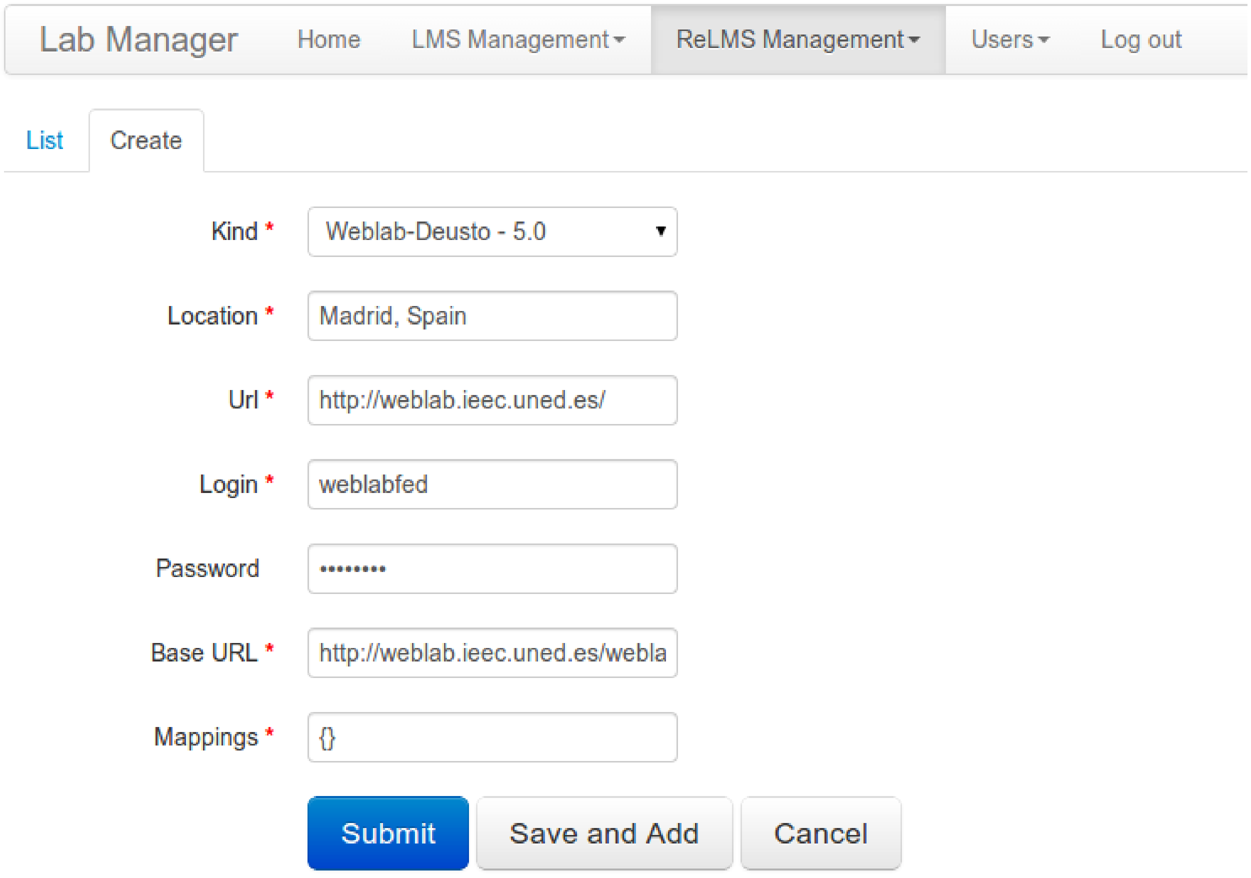
RLMS Administration

In the LabManager, you will configure the connection of the LabManager with the different RLMSs. In the following example, three RLMSs have been deployed: one WebLab-Deusto, one FCEIA-UNR and one MIT iLabs (support under development).



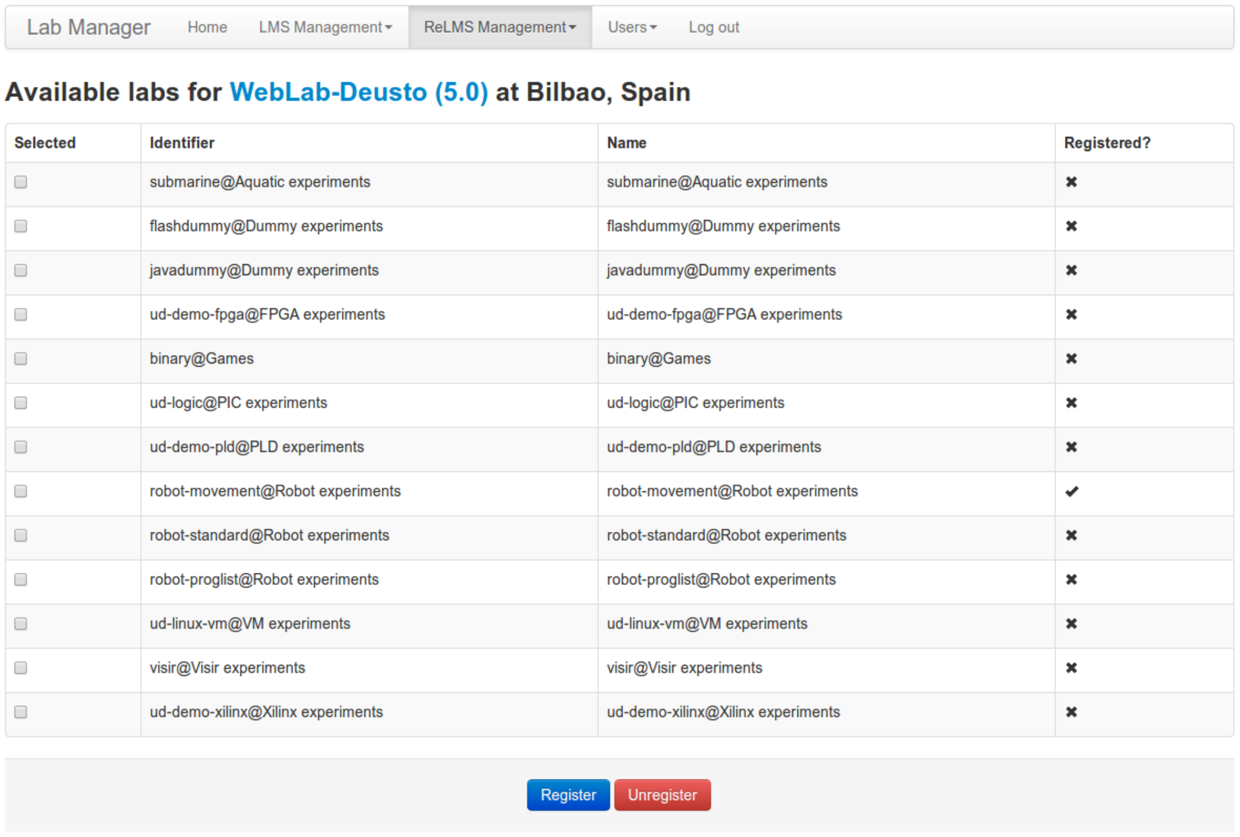
		Kind	Version	Location	Url	Labs
☐		WebLab-Deusto	5.0	Bilbao, Spain	https://www.weblab.deusto.es/	List
☐		FCEIA-UNR	1.2.2	Rosario, Argentina	http://labremf4a.fceia.unr.edu.ar/accesodeusto.aspx	List
☐		iLabs	1.2.2	MIT	http://ilab.mit.edu/wiki/	List

You may add more. For example, if you deploy a new WebLab-Deusto instance in your institution (or you have created one hosted in Deusto by the wCloud system), you may add it. So as to do this, complete the following form by clicking on *Create*:



Kind *	Weblab-Deusto - 5.0
Location *	Madrid, Spain
Url *	http://weblab.ieec.uned.es/
Login *	weblabfed
Password	••••••••
Base URL *	http://weblab.ieec.uned.es/webla
Mappings *	{}

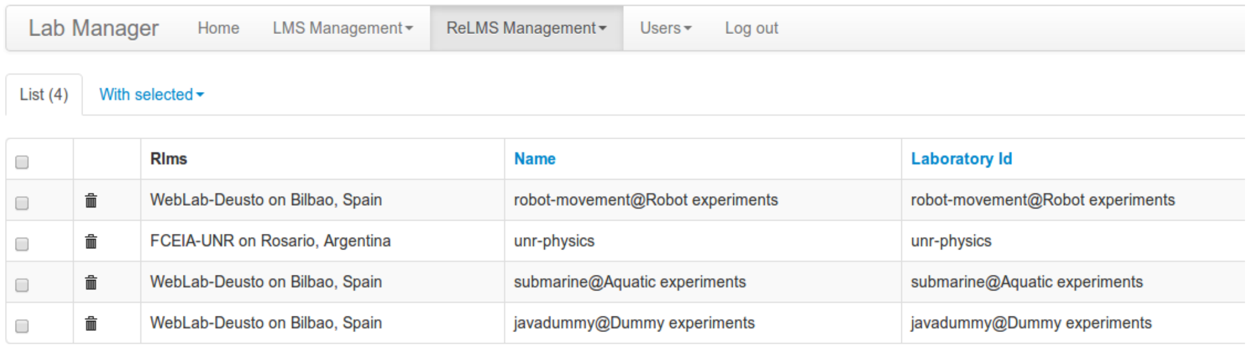
Once you've done this, you may want to register some laboratories. So as to do this, in the list of RLMSs click on `list`. You will see the following screen, listing the laboratories available in that RLMS and which ones are registered in the LabManager. You will be able to grant permission to LMS/CMS/PLEs only on those registered laboratories. So select the ones you want to use, and click on the register button.



Available labs for WebLab-Deusto (5.0) at Bilbao, Spain

Selected	Identifier	Name	Registered?
☐	submarine@Aquatic experiments	submarine@Aquatic experiments	✖
☐	flashdummy@Dummy experiments	flashdummy@Dummy experiments	✖
☐	javadummy@Dummy experiments	javadummy@Dummy experiments	✖
☐	ud-demo-fpga@FPGA experiments	ud-demo-fpga@FPGA experiments	✖
☐	binary@Games	binary@Games	✖
☐	ud-logic@PIC experiments	ud-logic@PIC experiments	✖
☐	ud-demo-pld@PLD experiments	ud-demo-pld@PLD experiments	✖
☐	robot-movement@Robot experiments	robot-movement@Robot experiments	✔
☐	robot-standard@Robot experiments	robot-standard@Robot experiments	✖
☐	robot-proglist@Robot experiments	robot-proglist@Robot experiments	✖
☐	ud-linux-vm@VM experiments	ud-linux-vm@VM experiments	✖
☐	visir@Visir experiments	visir@Visir experiments	✖
☐	ud-demo-xilinx@Xilinx experiments	ud-demo-xilinx@Xilinx experiments	✖

From this point, you will be able to see (or remove) the registered labs in the proper panel:



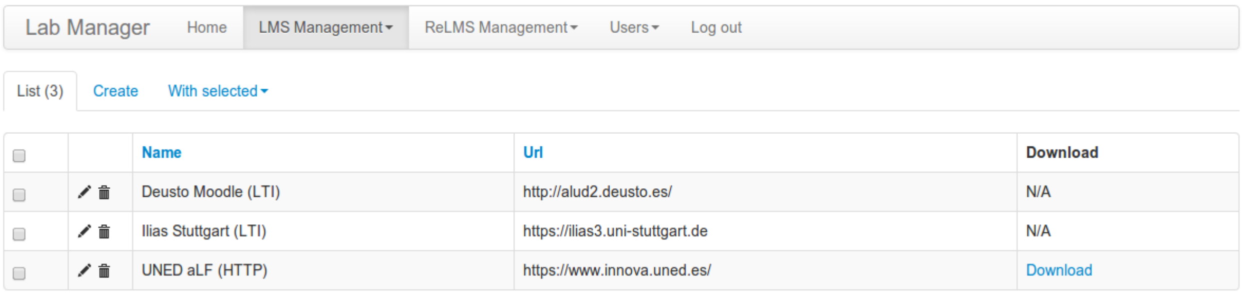
☐		Rlms	Name	Laboratory Id
☐	🗑	WebLab-Deusto on Bilbao, Spain	robot-movement@Robot experiments	robot-movement@Robot experiments
☐	🗑	FCEIA-UNR on Rosario, Argentina	unr-physics	unr-physics
☐	🗑	WebLab-Deusto on Bilbao, Spain	submarine@Aquatic experiments	submarine@Aquatic experiments
☐	🗑	WebLab-Deusto on Bilbao, Spain	javadummy@Dummy experiments	javadummy@Dummy experiments

LMS Administration

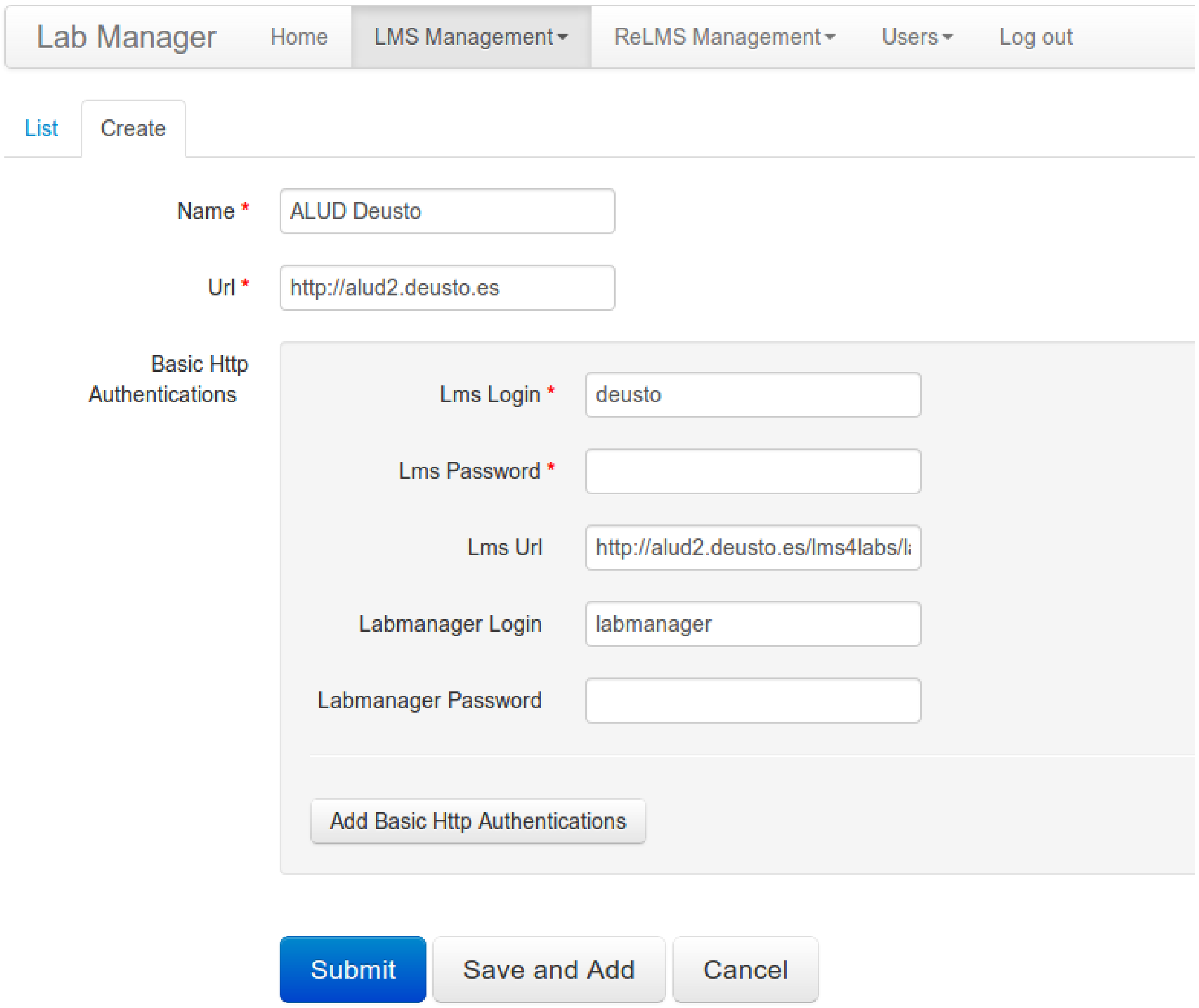
The LabManager administrator can also add LMSs/CMSs/PLEs so they can use this LabManager. The LabManager distinguishes among two types of LMSs/CMSs/PLEs: those supporting IMS LTI and those who do not support it. If IMS LTI is supported, this approach highly recommended. The rest will require a plug-in to be installed in the LMS/CMS/PLE.

So as to add a LMS, you have to go to the LMS management side and create a new LMS. When creating it, in the case of using the Basic HTTP approach (as opposed to the IMS LTI approach), you will need to add credentials. These include: LMS login (the username that the LMS will use in the LabManager), LMS password (the password used by the LMS in the LabManager), the LMS URL (pointing to the gateway4labs/list method), and the username and password of the LabManager in the LMS. The URL will be a URL pointing to the listing service. For instance, in Moodle, it will point to something like:

http://localhost/lms/moodle/2.3/blocks/gateway4labs/lms/list.php



		Name	Url	Download
		Deusto Moodle (LTI)	http://alud2.deusto.es/	N/A
		Ilias Stuttgart (LTI)	https://ilias3.uni-stuttgart.de	N/A
		UNED aLF (HTTP)	https://www.innova.uned.es/	Download

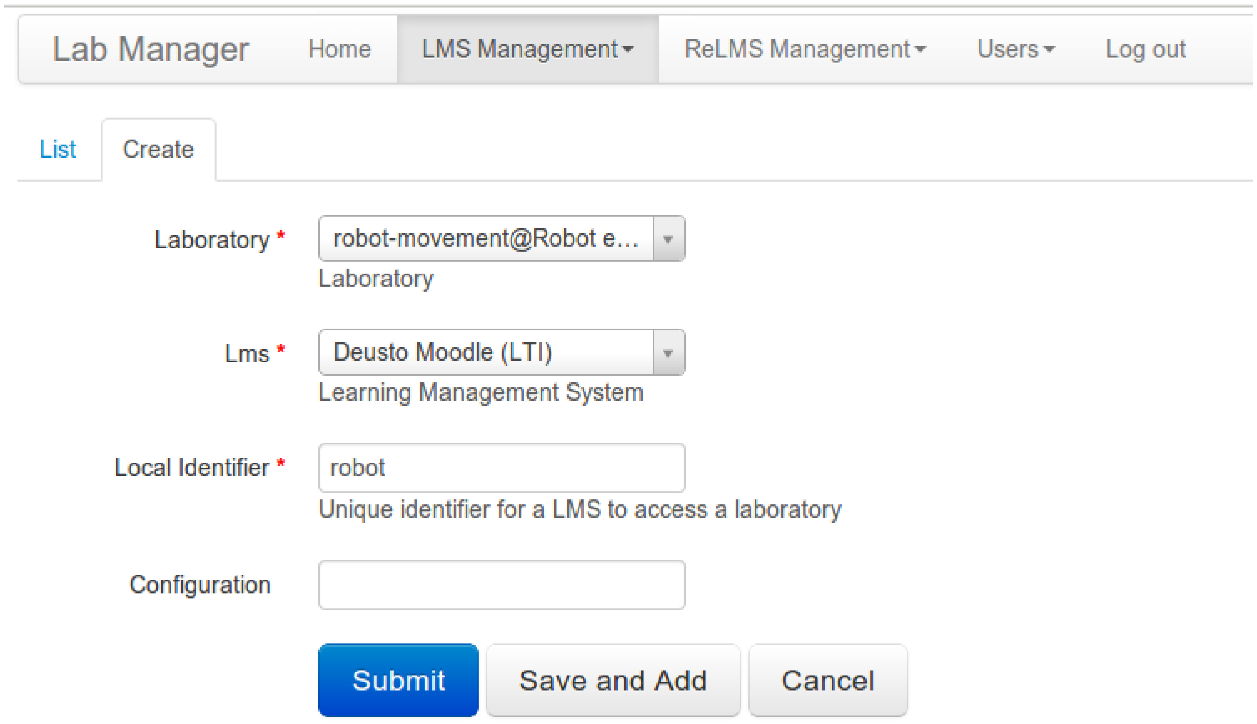


Name *	ALUD Deusto
Url *	http://alud2.deusto.es

Basic Http Authentications

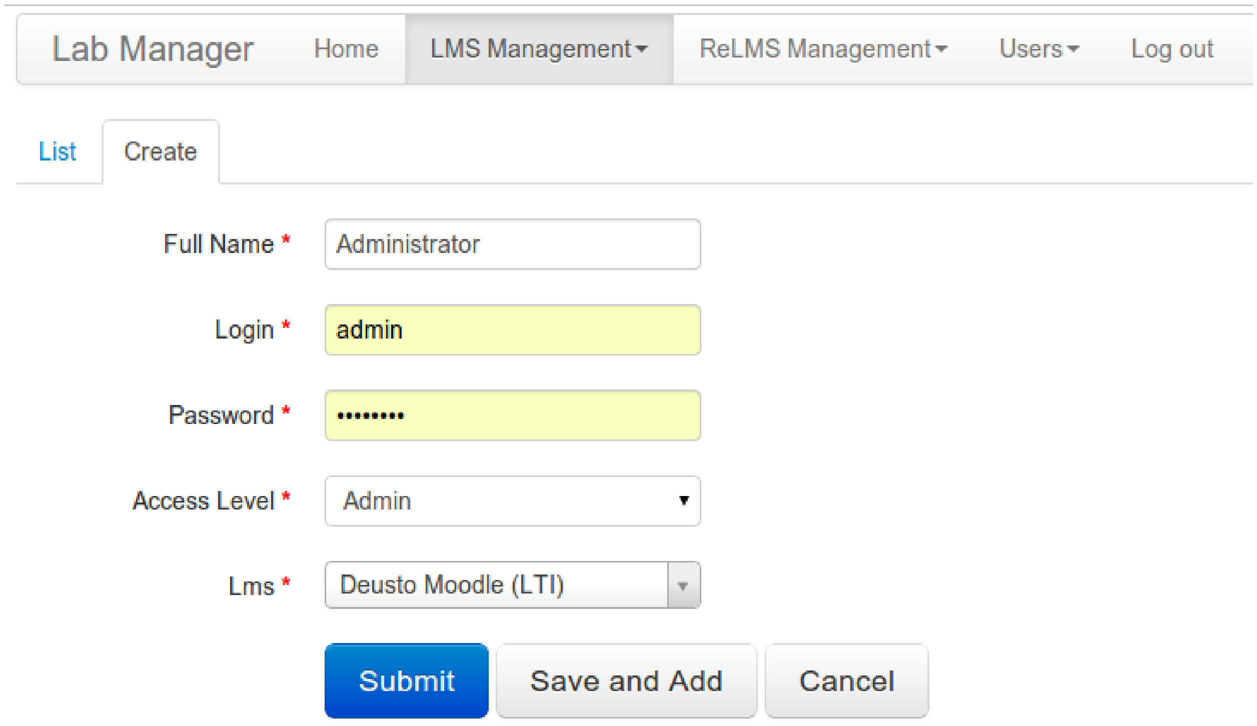
Lms Login *	deusto
Lms Password *	
Lms Url	http://alud2.deusto.es/lms4labs/l
Labmanager Login	labmanager
Labmanager Password	

After this, you can configure which permissions this LMS will have. For example, you may configure that it only has permission to a subset of the laboratories. When adding these permissions, you will define a unique identifier for that laboratory in that LMS/CMS/PLE.



Laboratory *	robot-movement@Robot e…
	Laboratory
Lms *	Deusto Moodle (LTI)
	Learning Management System
Local Identifier *	robot
	Unique identifier for a LMS to access a laboratory
Configuration	

Finally, the LabManager can create different LMS users, identifying who is administrating each LMS. From this point, you can contact the LMS administrator and give them these credentials.



Full Name *	Administrator
Login *	admin
Password *	••••••••
Access Level *	Admin
Lms *	Deusto Moodle (LTI)

2.1.3 Testing and debugging the Labmanager

Here we present details on how to debug new developments of the LabManager. First, we briefly present the way how to install the Labmanager for development purposes, followed by an explanation on how Eclipse IDE can be used to debug Labmanager.

Labmanager quick installation for testing

This document explains how to install the Labmanager from scratch over a Ubuntu 12.04 machine. The steps are the following:

1. Dependencies installation:

```
sudo apt-get install build-essential

sudo apt-get install libmysqlclient-dev

sudo apt-get install libsasl2-dev libldap2-dev

sudo apt-get install libxml2-dev libxslt1-dev

sudo apt-get install libfreetype6-dev libpng12-dev

sudo apt-get install build-essential python-dev

sudo apt-get install mysql-server
```

2. Download the code from git:

```
git clone https://github.com/gateway4labs/labmanager
```

3. Install and upgrade requirements:

```
cd labmanager

pip install -r requirements.txt

pip install --upgrade -r requirements.txt
```

4. Create the configuration file, based on the config.py.dist file that comes with the source code:

```
cp config.py.dist config.py
```

5. Open config.py with your favourite text editor and include the following information, replacing the words in CAPITALS with your information:

```
env_config = {'engine' : 'mysql', 'username' : 'YOUR_USERNAME', 'password' : 'YOUR_PASSWORD', 'd
```

6. Deploy the application (e.g. create the database, ...):

```
python deploy.py -cdu
```

7. Run the application in development mode:

```
python run.py
```

Debugging with Eclipse

By following the previous steps the application is running with the built-in Python debugger enabled, which reloads the application when it detects changes in the code. If you plan to debug the application using an external debugger (e.g. PyDev on Eclipse IDE), for instance, because you want to be able to perform step by step executions, you have to follow these steps:

1. Install Eclipse IDE:

```
sudo apt-get install eclipse
```

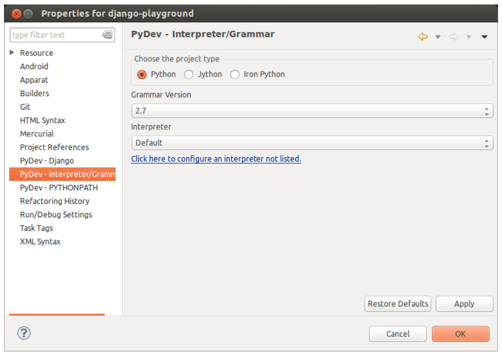
2. Install Pydev (Python plugin for Eclipse):

   Follow the installation guide here.

3. Make your Eclipse work with virtualenv (full instructions in Spanish here). NOTE: This assumes your are working in a virtualenv, skip this point otherwise.

   (a) Once you have a Eclipse project created based on PyDev, right-click on the project and select "Properties".

   (b) In the "PyDev – Interpreter/Grammar" entry, you will see the following window:



PyDev - Interpreter/Grammar

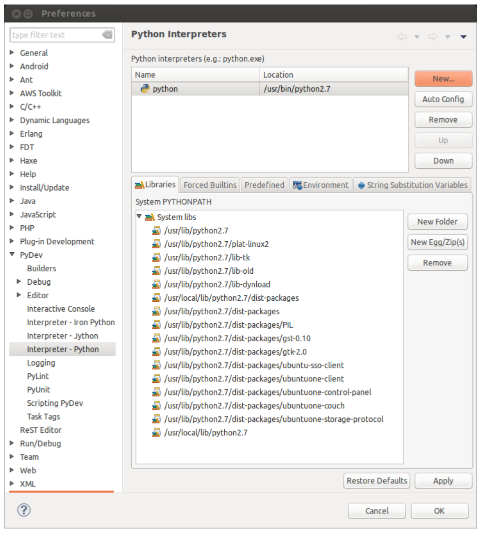
Choose the project type: [✓ Python] Jython Iron Python

Grammar Version: 2.7

Interpreter: Default

Click here to configure an interpreter not listed.

   (a) Click on "Click here to configure an interpreter not listed". Then we see the window shown in the following figure:



Python Interpreters

Python interpreters (e.g.: python.exe)

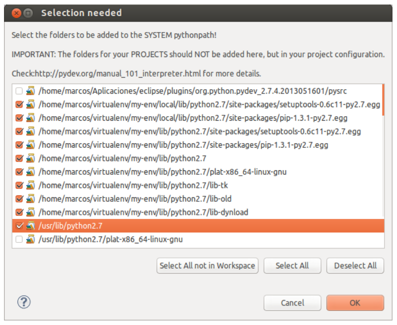
Name	Location
python	/usr/bin/python2.7

System PYTHONPATH

```
System libs
  /usr/lib/python2.7
  /usr/lib/python2.7/plat-linux2
  /usr/lib/python2.7/lib-tk
  /usr/lib/python2.7/lib-old
  /usr/lib/python2.7/lib-dynload
  /usr/local/lib/python2.7/dist-packages
  /usr/lib/python2.7/dist-packages
  /usr/lib/python2.7/dist-packages/PIL
  /usr/lib/python2.7/dist-packages/gst-0.10
  /usr/lib/python2.7/dist-packages/gtk-2.0
  /usr/lib/python2.7/dist-packages/ubuntu-sso-client
  /usr/lib/python2.7/dist-packages/ubuntuone-client
  /usr/lib/python2.7/dist-packages/ubuntuone-control-panel
  /usr/lib/python2.7/dist-packages/ubuntuone-couch
  /usr/lib/python2.7/dist-packages/ubuntuone-storage-protocol
  /usr/local/lib/python2.7
```

   (a) Click on "New". Then, we have to include the route to the Python executable in the virtualenv of interest.

(b) After clicking “Ok”, Eclipse will scan the route looking for libraries, and will offer the possibility to add more libraries. In this point, we will have to add “/usr/lib/python2.7” (or the Python version we have installed in our computer), as can be seen in this figure.



Select the folders to be added to the SYSTEM pythonpath!

IMPORTANT: The folders for your PROJECTS should NOT be added here, but in your project configuration.

Check:http://pydev.org/manual_101_interpreter.html for more details.

- [ ] /home/marcos/Aplicaciones/eclipse/plugins/org.python.pydev_2.7.4.2013051601/pysrc
- [✓] /home/marcos/virtualenv/my-env/local/lib/python2.7/site-packages/setuptools-0.6c11-py2.7.egg
- [✓] /home/marcos/virtualenv/my-env/local/lib/python2.7/site-packages/pip-1.3.1-py2.7.egg
- [✓] /home/marcos/virtualenv/my-env/lib/python2.7/site-packages/setuptools-0.6c11-py2.7.egg
- [✓] /home/marcos/virtualenv/my-env/lib/python2.7/site-packages/pip-1.3.1-py2.7.egg
- [✓] /home/marcos/virtualenv/my-env/lib/python2.7
- [✓] /home/marcos/virtualenv/my-env/lib/python2.7/plat-x86_64-linux-gnu
- [✓] /home/marcos/virtualenv/my-env/lib/python2.7/lib-tk
- [✓] /home/marcos/virtualenv/my-env/lib/python2.7/lib-old
- [✓] /home/marcos/virtualenv/my-env/lib/python2.7/lib-dynload
- [✓] /usr/lib/python2.7
- [ ] /usr/lib/python2.7/plat-x86_64-linux-gnu

(a) Click “Ok” several times until you get back to the first figure presented in this document. In this window, select the newly created interpreter and click “Ok”.

6. Open the config.py file in the labmanager installation and set:

```
DEBUG = False
```

7. Once you save the change to the config.py you are ready to debug your installation of LabManager, using breakpoints, watching variables, ...

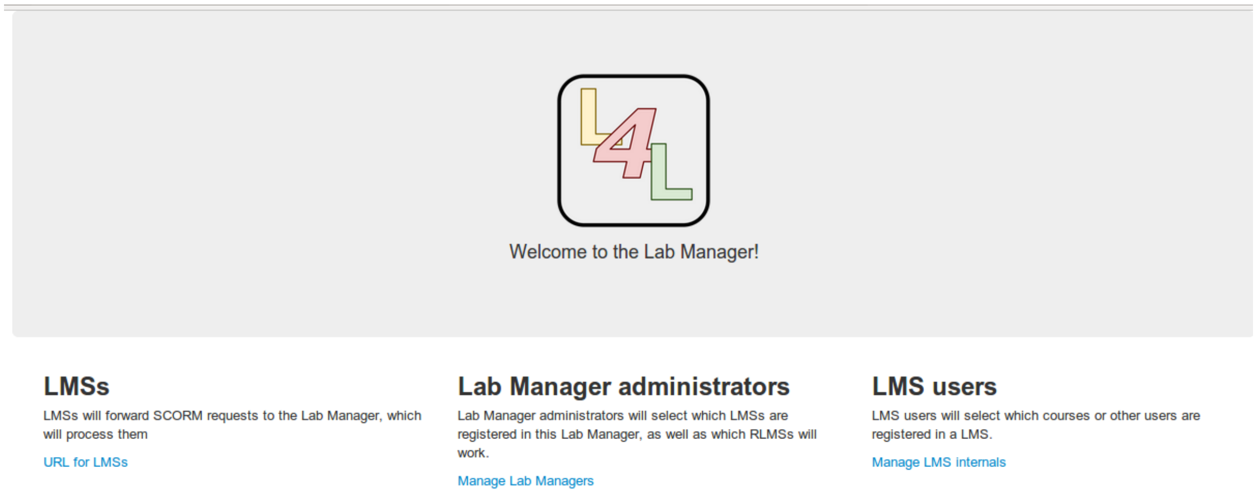
2.2 PLE/LMS/PLE administrators

A document for each supported LMS -e.g. any LTI, Moodle <= 2.1 through plug-in, Moodle 1.9 through plug-in, .LRN, etc. I would be fine with refering to the doc of each LMS/CMS/PLE plug-in repository

2.2.1 LMS/CMS/PLE administration

The Labmanager offers two ways of interaction with the LMS/CMS/PLE. One method uses the IMS LTI standard and the second is a custom communication schema called Basic HTTP and relies on a separate module/block developed for each specific LMS/CMS/PLE.

In any case, you need to log in as a LMS administrator. So in the root of the LabManager, click on the link in the right at the front page:



LMSs

LMSs will forward SCORM requests to the Lab Manager, which will process them

URL for LMSs

Lab Manager administrators

Lab Manager administrators will select which LMSs are registered in this Lab Manager, as well as which RLMSs will work.

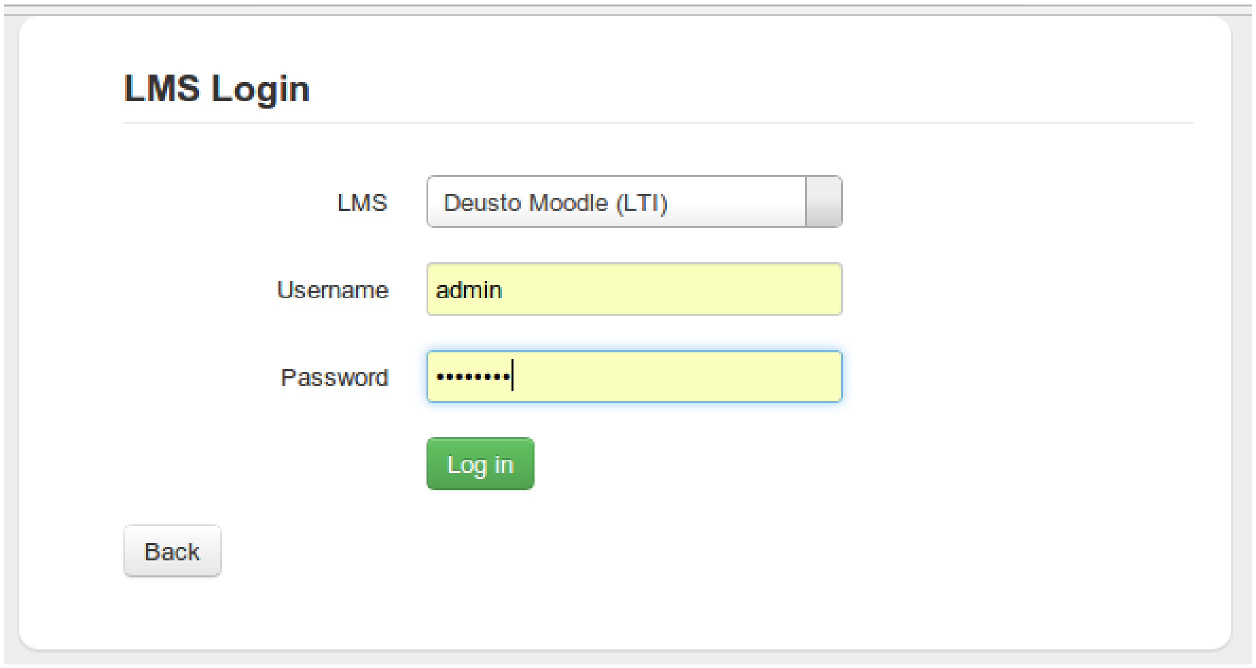
Manage Lab Managers

LMS users

LMS users will select which courses or other users are registered in a LMS.

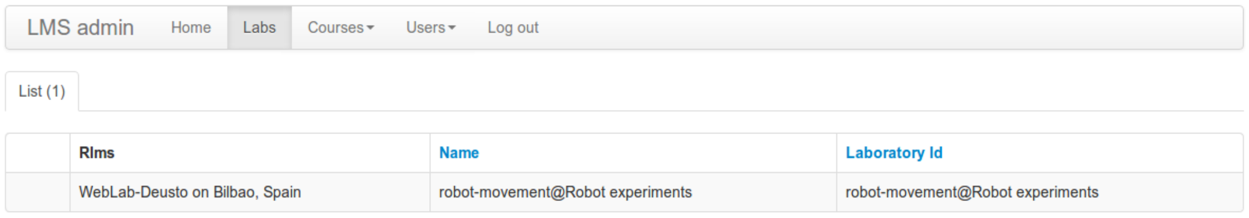
Manage LMS internals

And then, select which LMS you are administrating and use your credentials in that LMS.



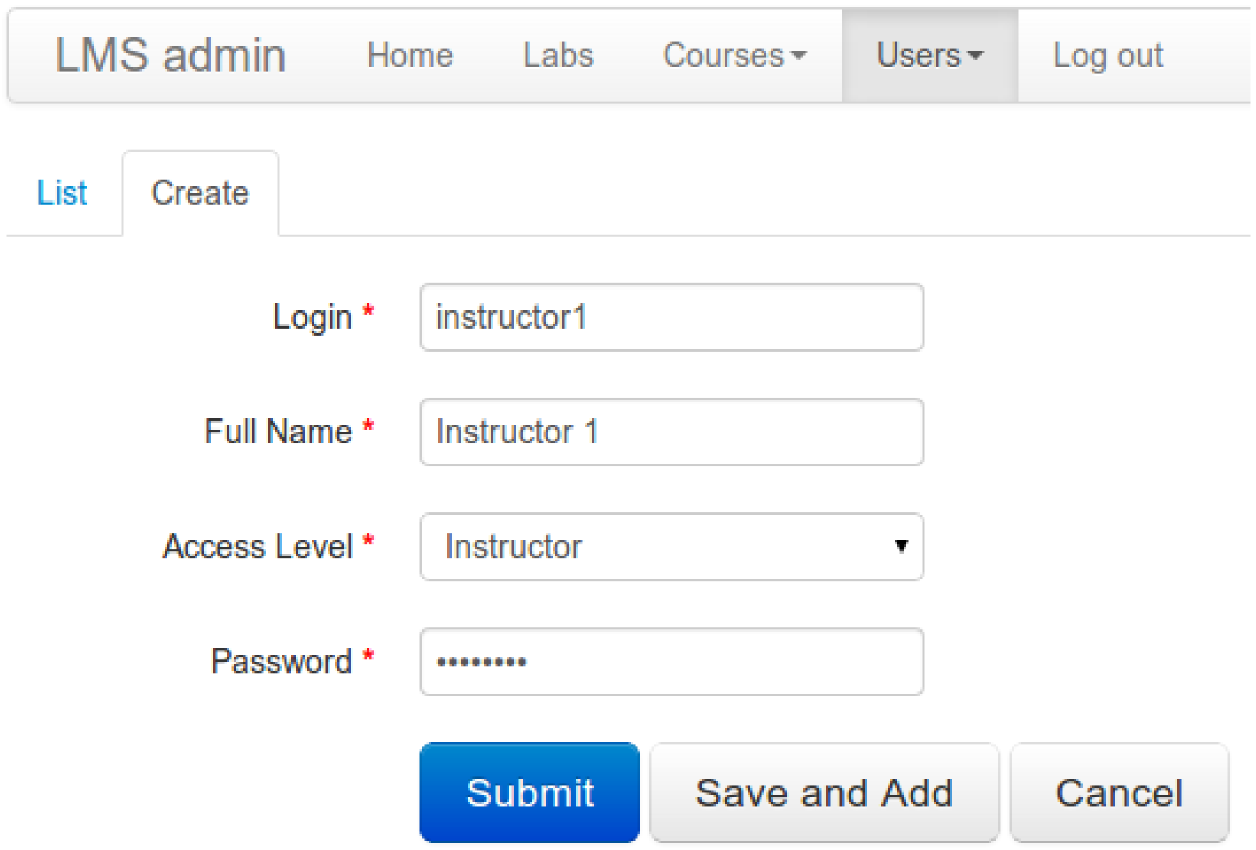
LMS	Deusto Moodle (LTI)
Username	admin
Password	••••••••

By clicking on Labs, you can see what permissions your LMS have. If you want to get more permissions, contact the LabManager administrator.



	Rlms	Name	Laboratory Id
	WebLab-Deusto on Bilbao, Spain	robot-movement@Robot experiments	robot-movement@Robot experiments

And in the Users panel, you may create other users (administrators or instructors) for this LMS:

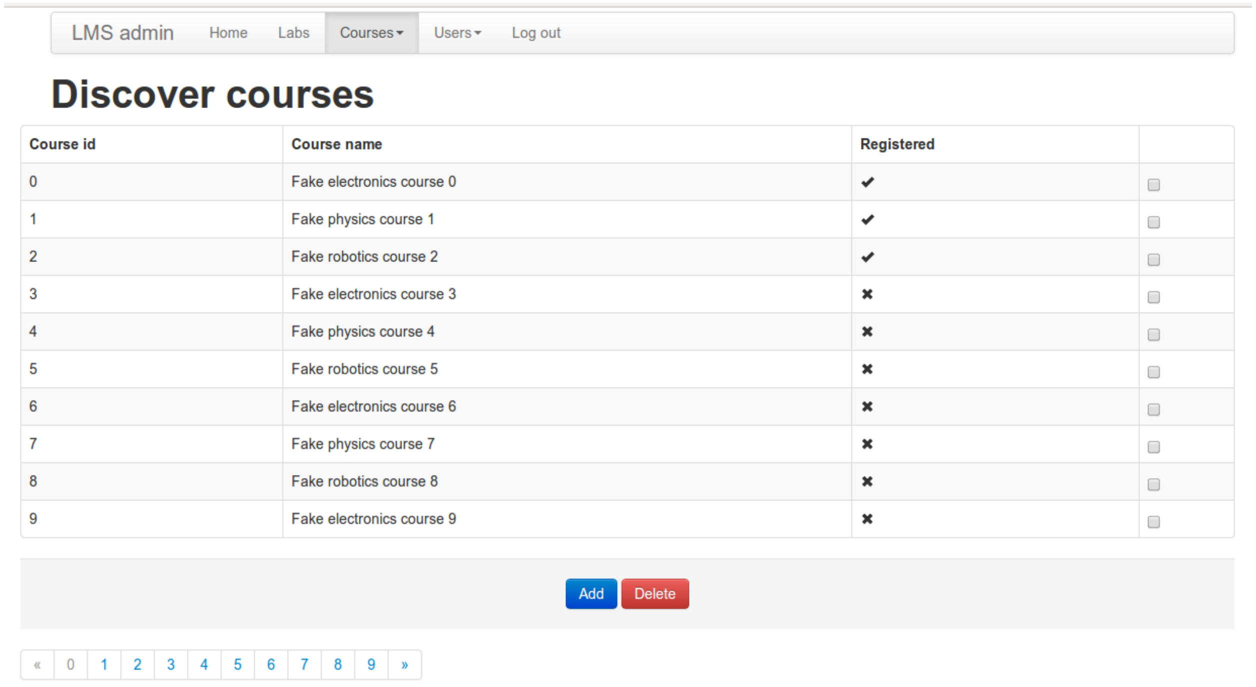


Login *	instructor1
Full Name *	Instructor 1
Access Level *	Instructor
Password *	•••••••••

1. Basic HTTP

In the case of Basic HTTP, we rely on **courses**: the LMS/CMS/PLE administrator will need to add the target courses to the LabManager and grant permissions to these courses. Then, when the student uses the plug-in in the LMS/CMS/PLE, this plug-in will inform us whether the student is enrolled in the course or not, and the LabManager will rely on this information to guarantee access or not.

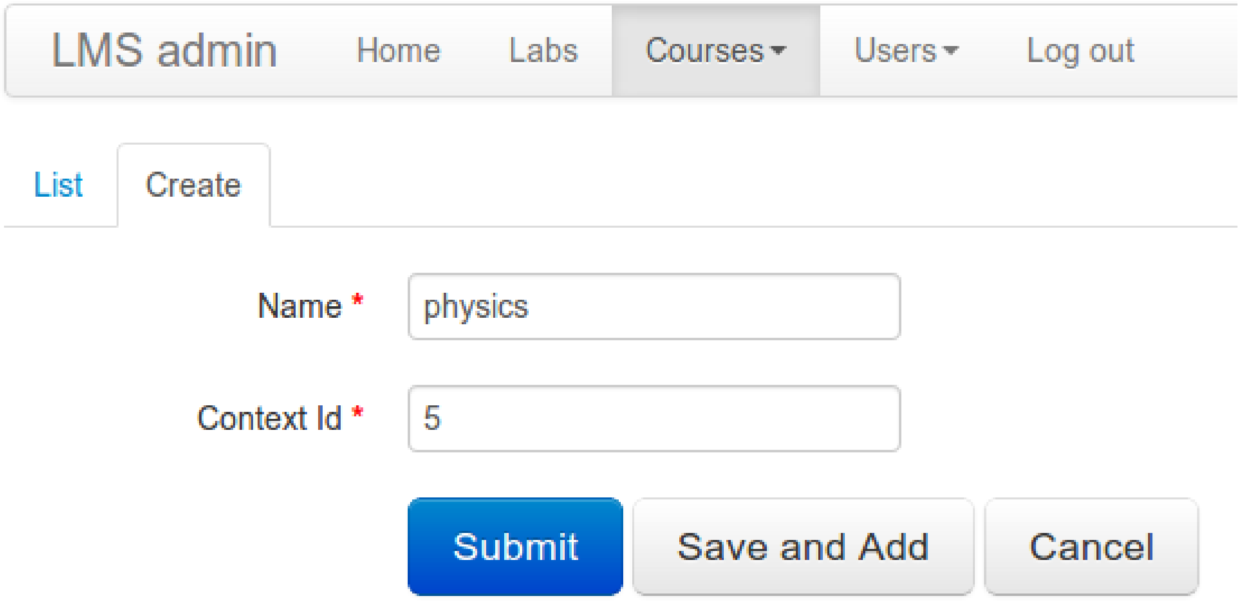
So as to add these courses, the administrator may do this manually, or he may use the discovery system (if implemented in the LMS/CMS/PLE). When clicking on it, a list of courses provided by the LMS/CMS/PLE is provided, and the administrator will register those targeted:



Discover courses

Course id	Course name	Registered	
0	Fake electronics course 0	✔	☐
1	Fake physics course 1	✔	☐
2	Fake robotics course 2	✔	☐
3	Fake electronics course 3	✖	☐
4	Fake physics course 4	✖	☐
5	Fake robotics course 5	✖	☐
6	Fake electronics course 6	✖	☐
7	Fake physics course 7	✖	☐
8	Fake robotics course 8	✖	☐
9	Fake electronics course 9	✖	☐

The other option is to do this manually:



Name *	physics
Context Id *	5

Once this is done, the LMS can already contact the LabManager requesting access to laboratories. If the LMS supports SCORM, a custom SCORM package can be downloaded from the Laboratory list.

2. IMS LTI

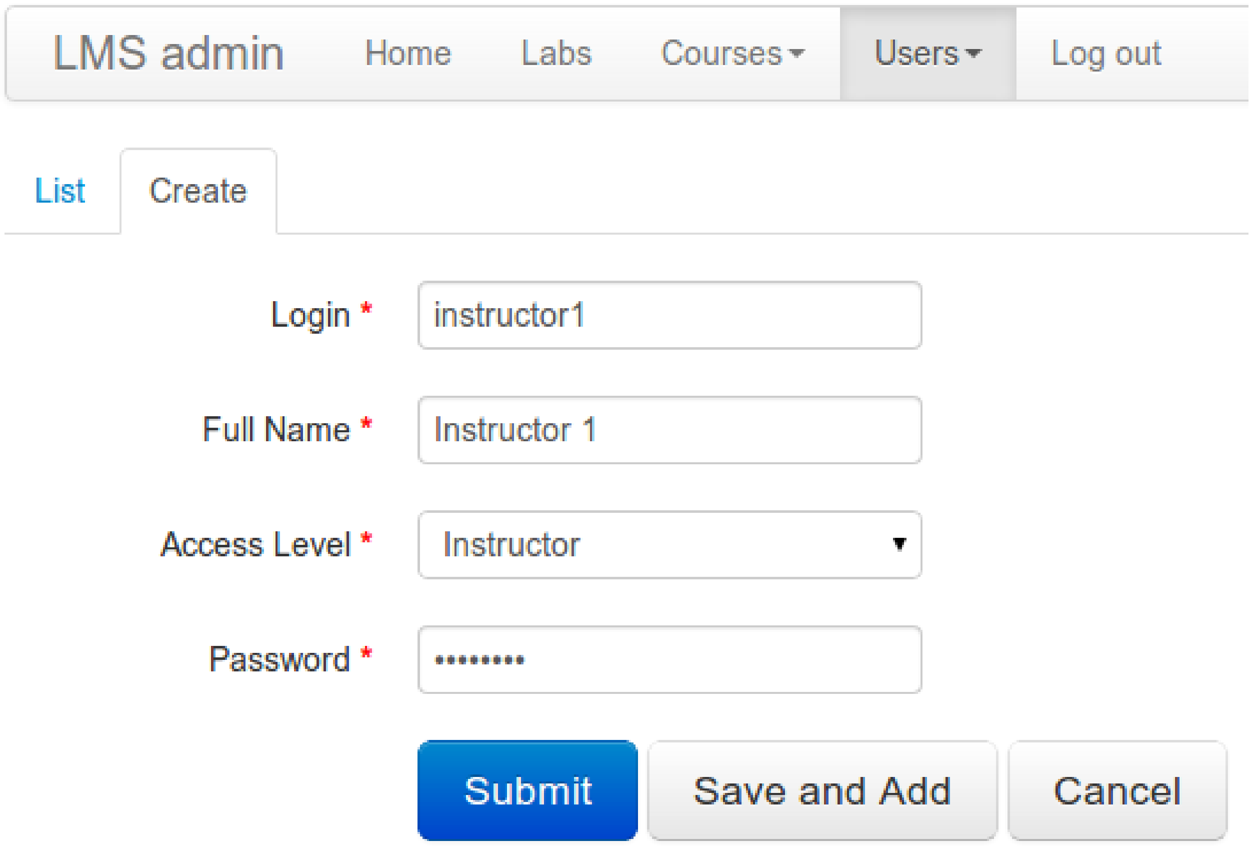
In an effort to promote tool interoperability, the IMS proposed an standard named Learning Tool Interoperability (LTI) that is now supported by different LMS such as Moodle (v2.2+), Sakai and Blackboard. Labmanager will act as an LTI tool provider and allow an LMS to request access to experiments registered in the Labmanager.

Some of the benefits that the LTI alternative offers are:

- Use of OAuth v1.0 for authentication.
- No need for extra developments to work.
- No need for requirements in the LMS side (and therefore, IT services will not need to install anything).

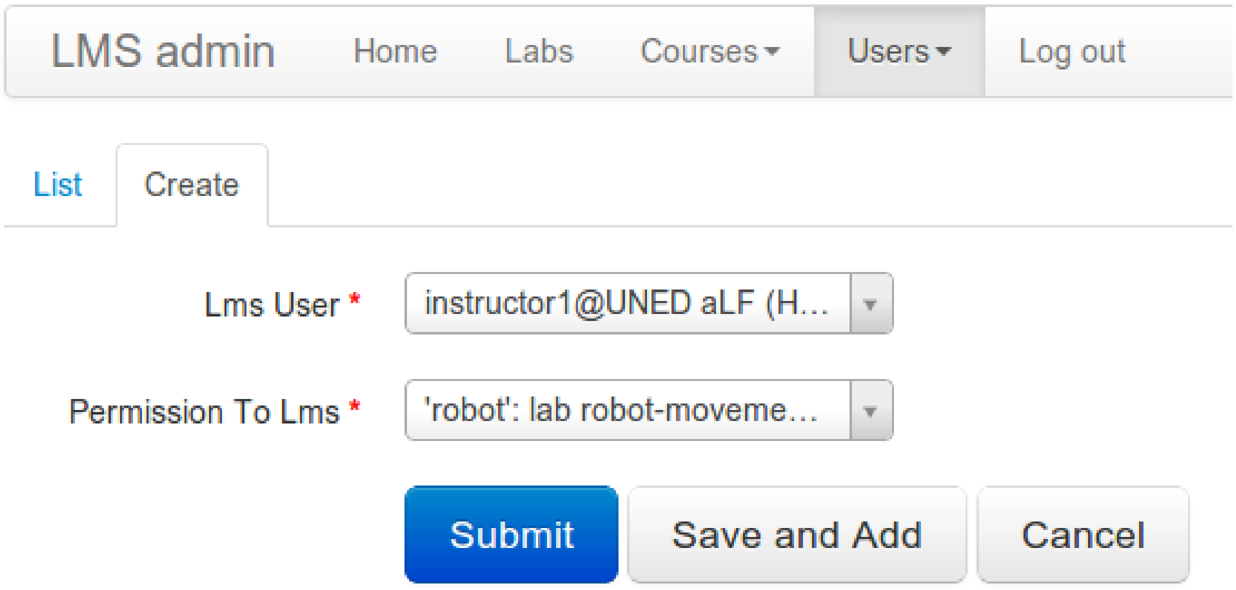
LTI work-flow

So as to use the LTI version, the **course** concept will not be used. Instead, the LMS/CMS/PLE administrator will create *Instructor* users, such as:



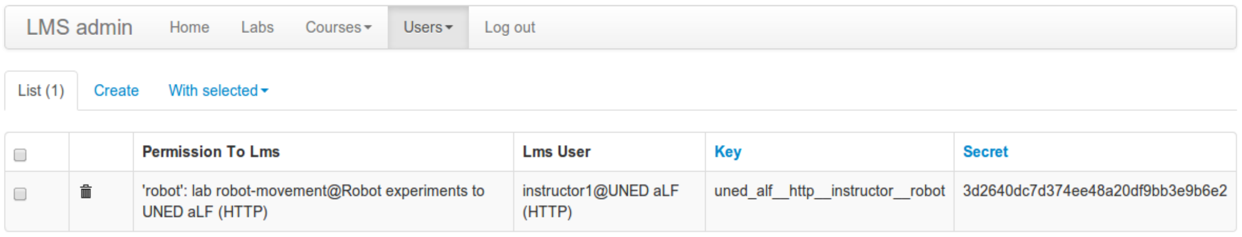
Login *	instructor1
Full Name *	Instructor 1
Access Level *	Instructor
Password *	•••••••••

Once the user is created, you can grant permissions on them:



Lms User *	instructor1@UNED aLF (H…
Permission To Lms *	'robot': lab robot-moveme…

From this point, the LMS administrator will be able to see two self-generated credentials for this user and laboratory:

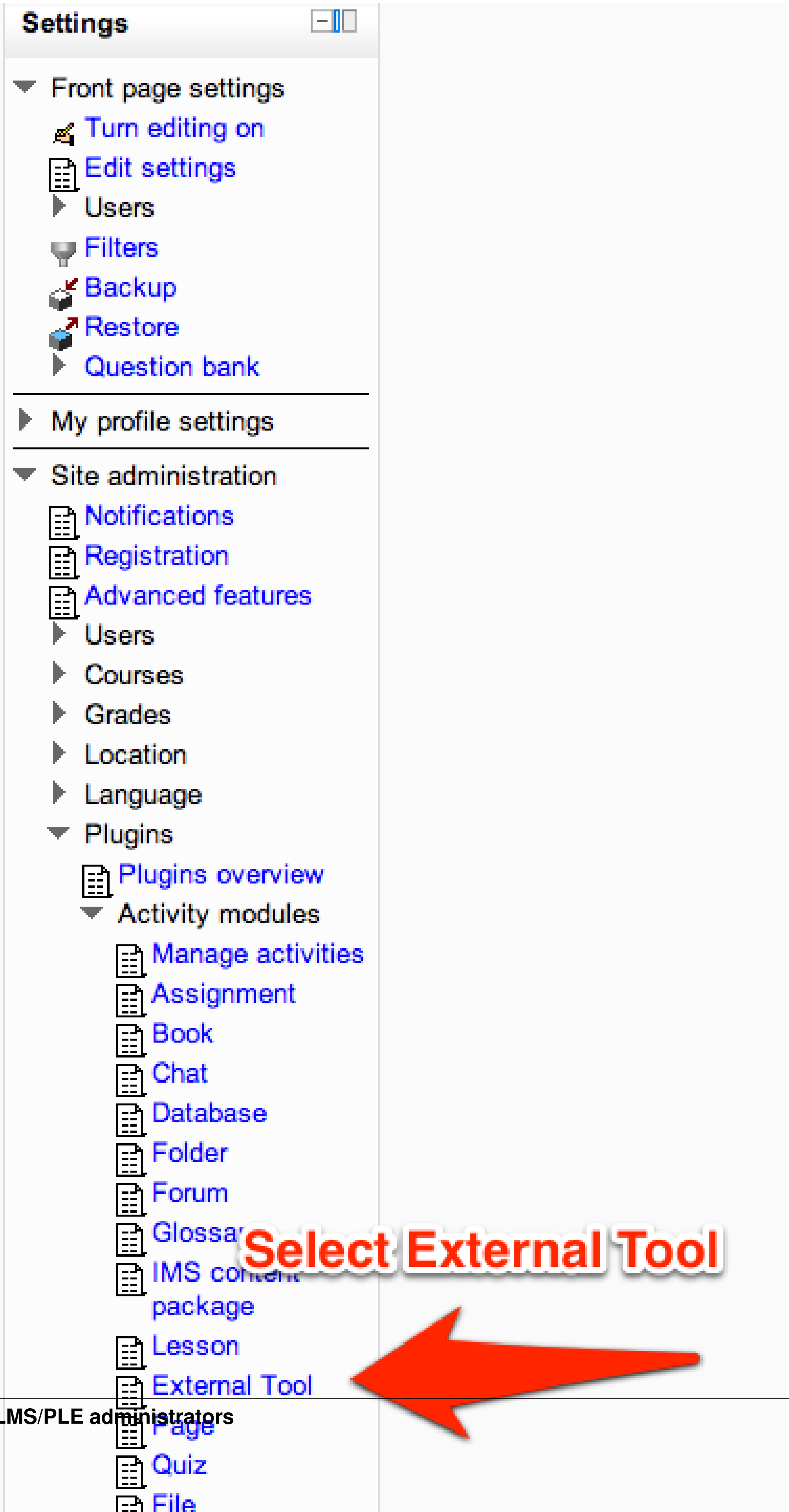


		Permission To Lms	Lms User	Key	Secret
		'robot': lab robot-movement@Robot experiments to UNED aLF (HTTP)	instructor1@UNED aLF (HTTP)	uned_alf__http__instructor__robot	3d2640dc7d374ee48a20df9bb3e9b6e2

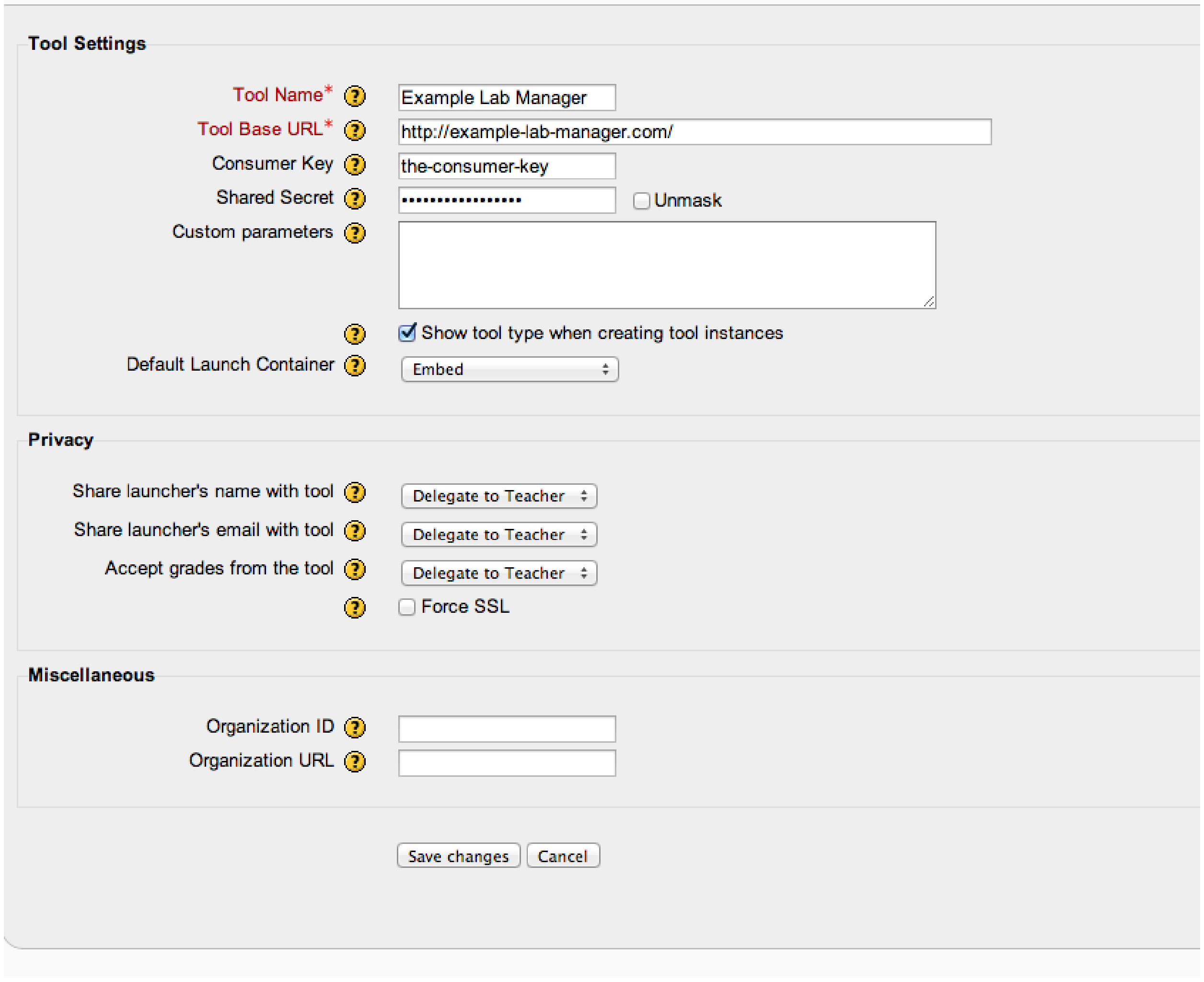
LMS administrator may now communicate to these instructors what are these credentials. Once done, these instructors will be able to use these pairs of credentials in their courses in the LMS/CMS/PLEs.

2.2.2 Usage: Moodle administrator

Since Moodle 2.x supports the IMS LTI 1.1.1 standard you just need to add an external tool:


Settings
Front page settings
Turn editing on
Edit settings
Users
Filters
Backup
Restore
Question bank
My profile settings
Site administration
Notifications
Registration
Advanced features
Users
Courses
Grades
Location
Language
Plugins
Plugins overview
Activity modules
Manage activities
Assignment
Book
Chat
Database
Folder
Forum
Glossa
IMS co
package
Lesson
External Tool
Page
Quiz
File
Select External Tool

Here, you can add a Lab Manager installation, you will need a **consumer key** and a **shared secret** from the Lab Manager administrator. Fill in the form and save the external tool.



Tool Settings

Tool Name*	Example Lab Manager
Tool Base URL*	http://example-lab-manager.com/
Consumer Key	the-consumer-key
Shared Secret	••••••••••••••••• ☐ Unmask
Custom parameters	
	☑ Show tool type when creating tool instances
Default Launch Container	Embed

Privacy

Share launcher's name with tool	Delegate to Teacher
Share launcher's email with tool	Delegate to Teacher
Accept grades from the tool	Delegate to Teacher
	☐ Force SSL

Miscellaneous

Organization ID	
Organization URL	

Note: You could add as many different Lab Manager installations as you want as long as you have the consumer and secret keys. This will allow you to access different laboratories on different universities each of them with a different Lab Manager installation.

2.3 Remote laboratory administrators

A document for each supported RLMS, refering to the doc available in each RLMS repository

2.3.1 REST API for RLMS

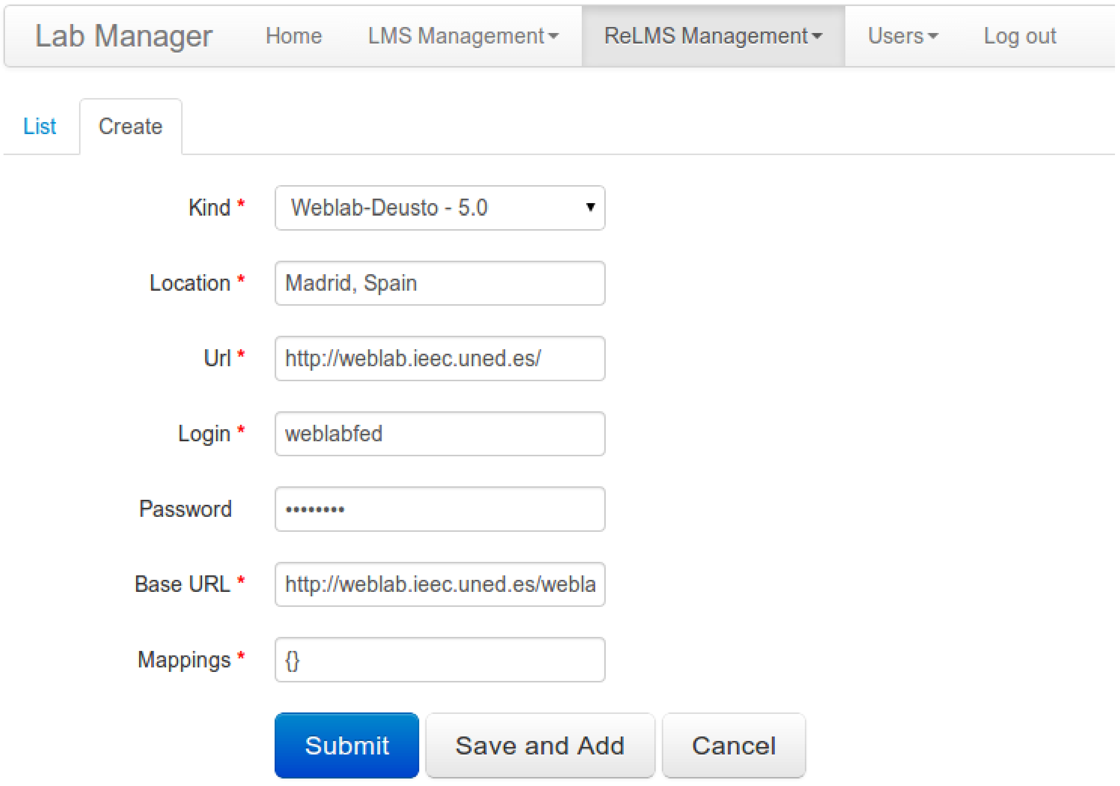
Right now, different RLMSs require a plugin written in Python so that the RLMS can be used by the LabManager. In this moment, plugins for three RLMS have been implemented, namely:

1. Weblab Deusto, available here.

2. FCEIA, Universidad Nacional del Rosario, available here.
3. PhET, available here.
4. MIT iLabs, available here.

Creation of a new RLMS in the LabManager

Right now, the creation of a new RLMS in the Labmanager involves the selection of the RLMS type (among those RLMSs whose plugins are installed). After that, several pieces of information must be inserted depending on the RLMS, and this is hard-coded in the LabManager code. As an example, the creation of a Weblab Deusto RLMS is depicted in the figure, which requires the insertion of the location, base URL, login, password, and mappings (other RLMSs, such iLabs, will require different parameters).



Kind *	Weblab-Deusto - 5.0
Location *	Madrid, Spain
Url *	http://weblab.ieec.uned.es/
Login *	weblabfed
Password	••••••••
Base URL *	http://weblab.ieec.uned.es/webla
Mappings *	{}

When the REST API is ready, the creation of RLMSs in the LabManager will be a three steps procedure, as follows:

1. **Selection of the RLMSs type.** Along with the existing RLMSs based on plugins, a new kind called *HTTP* will be added. A *Continue* button will be placed in this view.
2. **Initial insertion of information.** Based on the RLMS type selected in the previous step, a view requesting the necessary information for each RLMS type will be presented.
   - In the case of the plugins based RLMSs, this screen will ask for the same information as the current creation procedure, and will be the last step in this procedure. A *Finish* button will be placed in this view, once it is clicked, the RLMS will be inserted in the database.
   - In the case of the new *HTTP* RLMS kind, this second step will ask for the URL, user and password of the laboratory server. This information will be used on the next step to obtain the

list of parameters this RLMS requires to work properly. A *Continue* button will be placed in this view.

3. **Final insertion of information.** This step will only be executed in the case that the *HTTP* kind has been selected in the first step. For this step, the LabManager will ...

   (a) get the list of parameters needed by the RLMS, by querying the REST service implemented by the RLMS. For example, in the case of Weblab Deusto, this list of parameters would include location, base URL, login, password, and mappings.

   (b) create a view in which the LabManager admin is asked to provide values for these parameters.

A *Finish* button will be placed in this view. Once this button is clicked, the aforementioned parameters will be checked, and the LabManager admin will be informed whether they are correct or not. If they are correct, the new RLMS will be inserted in the database; otherwise, the Labmanager admin will be asked to fix them, or cancel the RLMS creation.

2.4 Instructors

A document for instructors: how to download the SCORM objects, where from, how to edit them, etc.

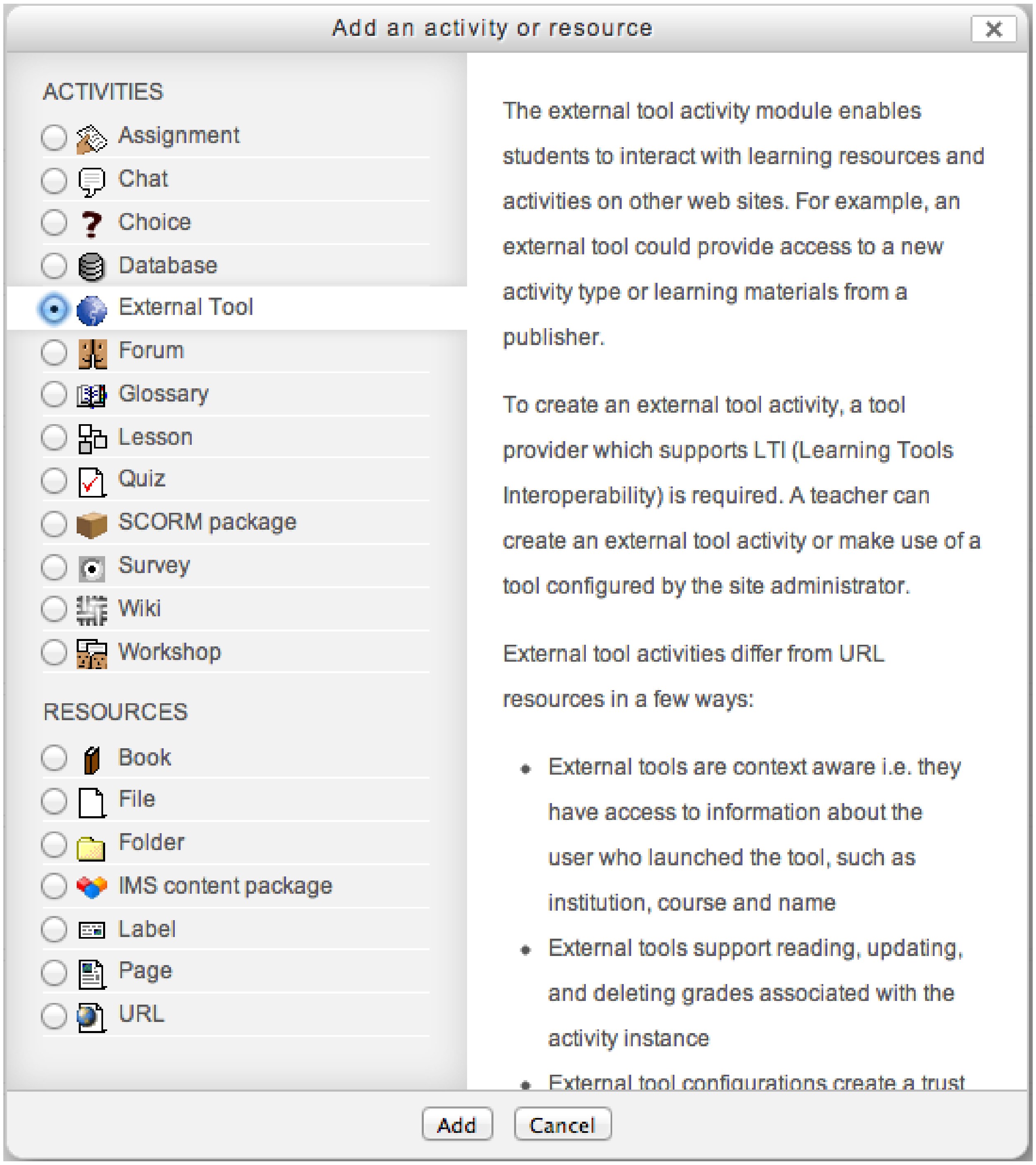
2.4.1 Usage: Moodle

1. IMS LTI

Once the Moodle Administrator added a Lab Manager as an external tool, it can be added to the courses.

You need to go into the course and click the "*Turn editing on*" button on the top right.

Then click on the "Add an activity or resource" on the section of the course where the laboratory will be used. Select the "External Tool" option and click the "Add" button.



The external tool activity module enables students to interact with learning resources and activities on other web sites. For example, an external tool could provide access to a new activity type or learning materials from a publisher.

To create an external tool activity, a tool provider which supports LTI (Learning Tools Interoperability) is required. A teacher can create an external tool activity or make use of a tool configured by the site administrator.

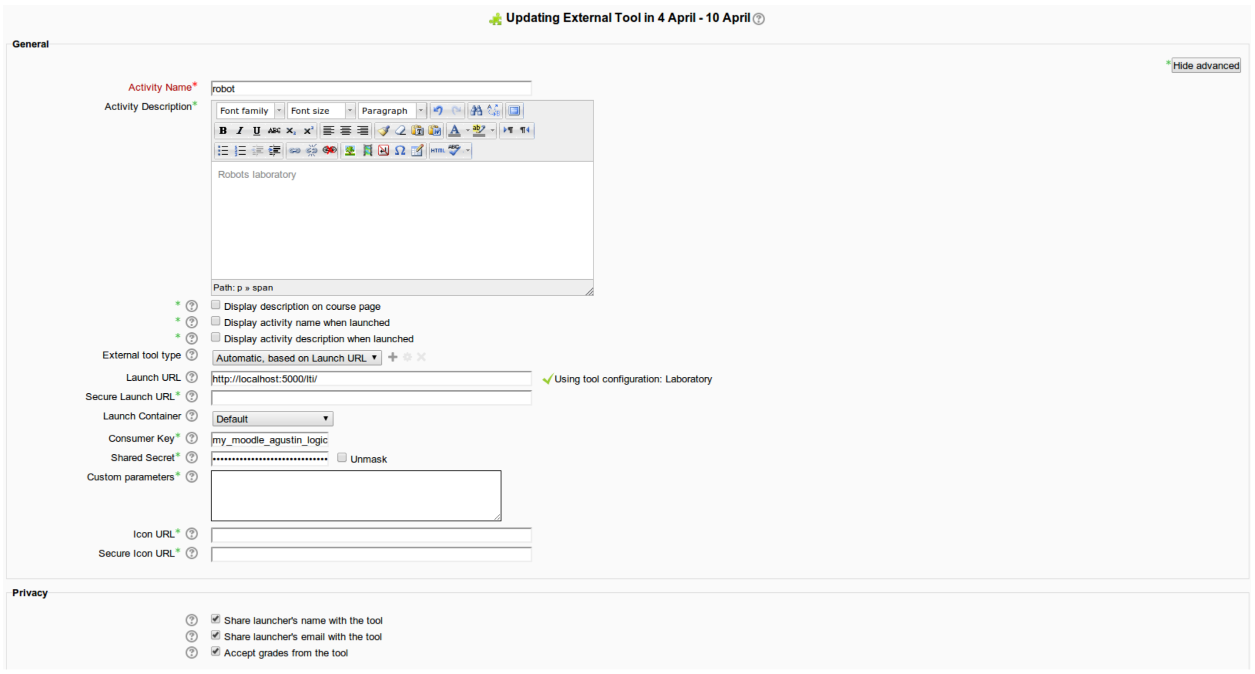
External tool activities differ from URL resources in a few ways:

- External tools are context aware i.e. they have access to information about the user who launched the tool, such as institution, course and name
- External tools support reading, updating, and deleting grades associated with the activity instance
- External tool configurations create a trust

Fill out the form and select the Lab Manager installation you want to add in the “External tool type” field. Notice that the options listed here are the names of the Lab Manager installations you added previously. So adding a it is always better to add a descriptive name when adding the Lab Manager tool.

You need to show the advanced features, and then fill the consumer key and the secret as the LMS Administrator has detailed (there is one consumer key and secret per instructor and laboratory).

You may leave the “Launch URL” blank, since the one from the selected tool will be used.



Updating External Tool in 4 April - 10 April

General

Hide advanced

Activity Name*	robot
Activity Description*	Robots laboratory
	[ ] Display description on course page
	[ ] Display activity name when launched
	[ ] Display activity description when launched
External tool type	Automatic, based on Launch URL
Launch URL	http://localhost:5000/lti/ ✓ Using tool configuration: Laboratory
Secure Launch URL*	
Launch Container	Default
Consumer Key*	my_moodle_agustin_logic
Shared Secret*	•••••••••••••••••••••••••••••• [ ] Unmask
Custom parameters*	
Icon URL*	
Secure Icon URL*	

Privacy

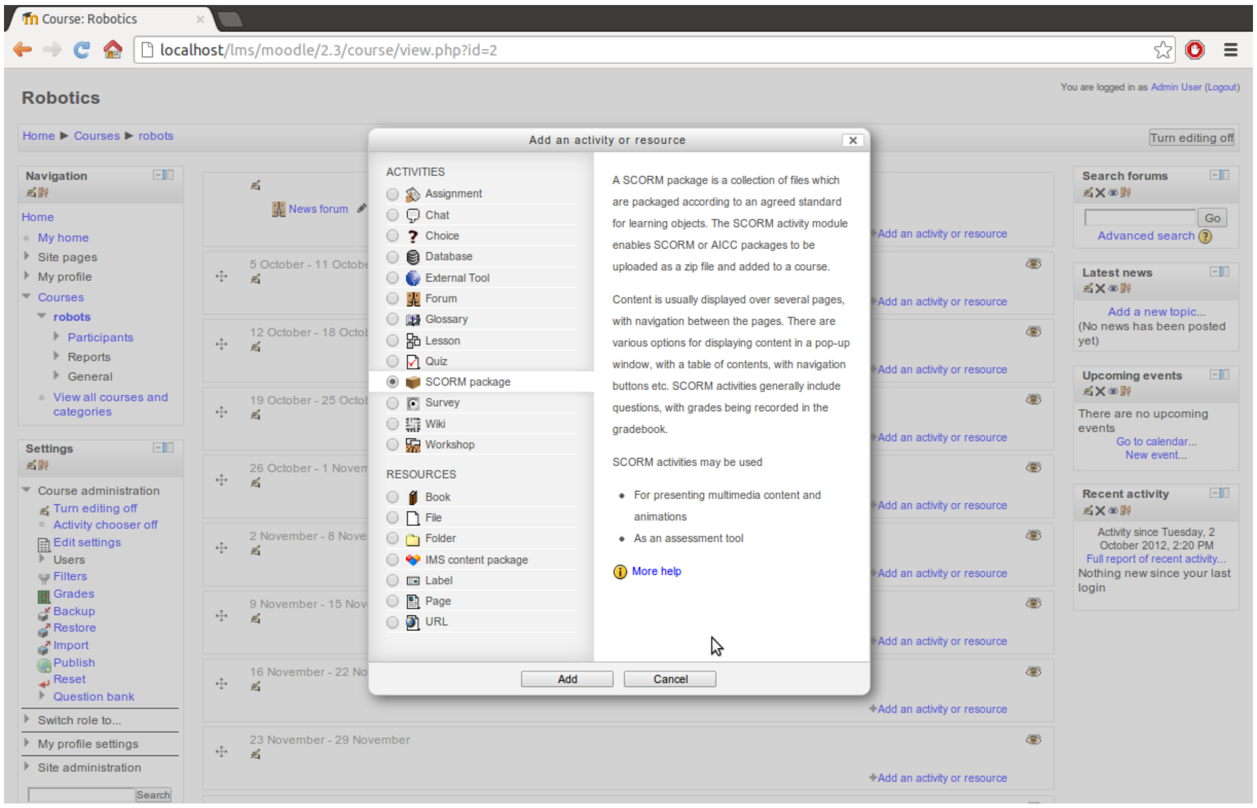
- [✓] Share launcher's name with the tool
- [✓] Share launcher's email with the tool
- [✓] Accept grades from the tool

Click the "Save and return to course" button and you should see the activity in the section of the course. If you click it, it will go to the Lab Manager and ask for the laboratory.

2. SCORM Package

A teacher needs a SCORM object per laboratory. So as to gather it, the LMS administrator needs to provide the SCORM object to the teacher. The teacher can optionally modify it (adding laboratory information, course information, etc.). Then, the teacher can upload it as a SCORM object to a particular week, make it visible for students, etc.

So, granted that the LMS administrator has sent a SCORM object to the teacher:



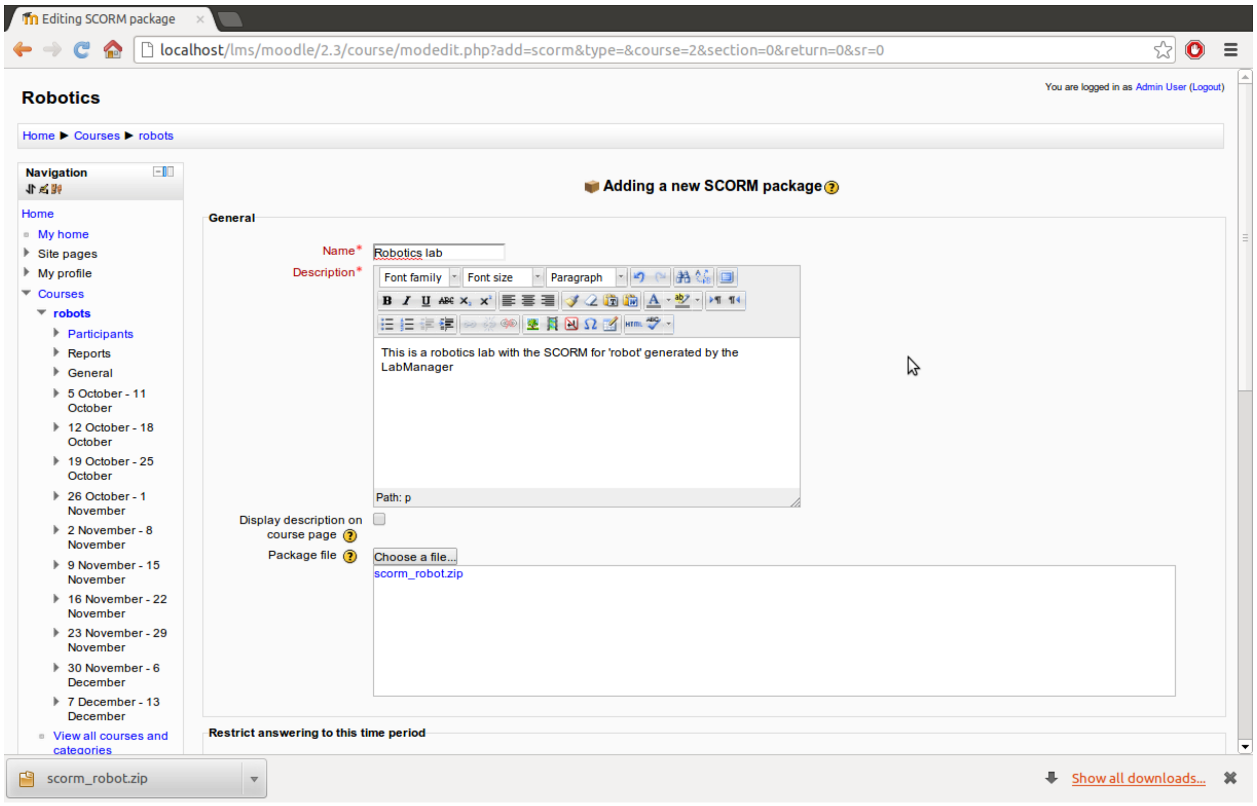
A SCORM package is a collection of files which are packaged according to an agreed standard for learning objects. The SCORM activity module enables SCORM or AICC packages to be uploaded as a zip file and added to a course.

Content is usually displayed over several pages, with navigation between the pages. There are various options for displaying content in a pop-up window, with a table of contents, with navigation buttons etc. SCORM activities generally include questions, with grades being recorded in the gradebook.

SCORM activities may be used

- For presenting multimedia content and animations
- As an assessment tool

More help



Name* Robotics lab

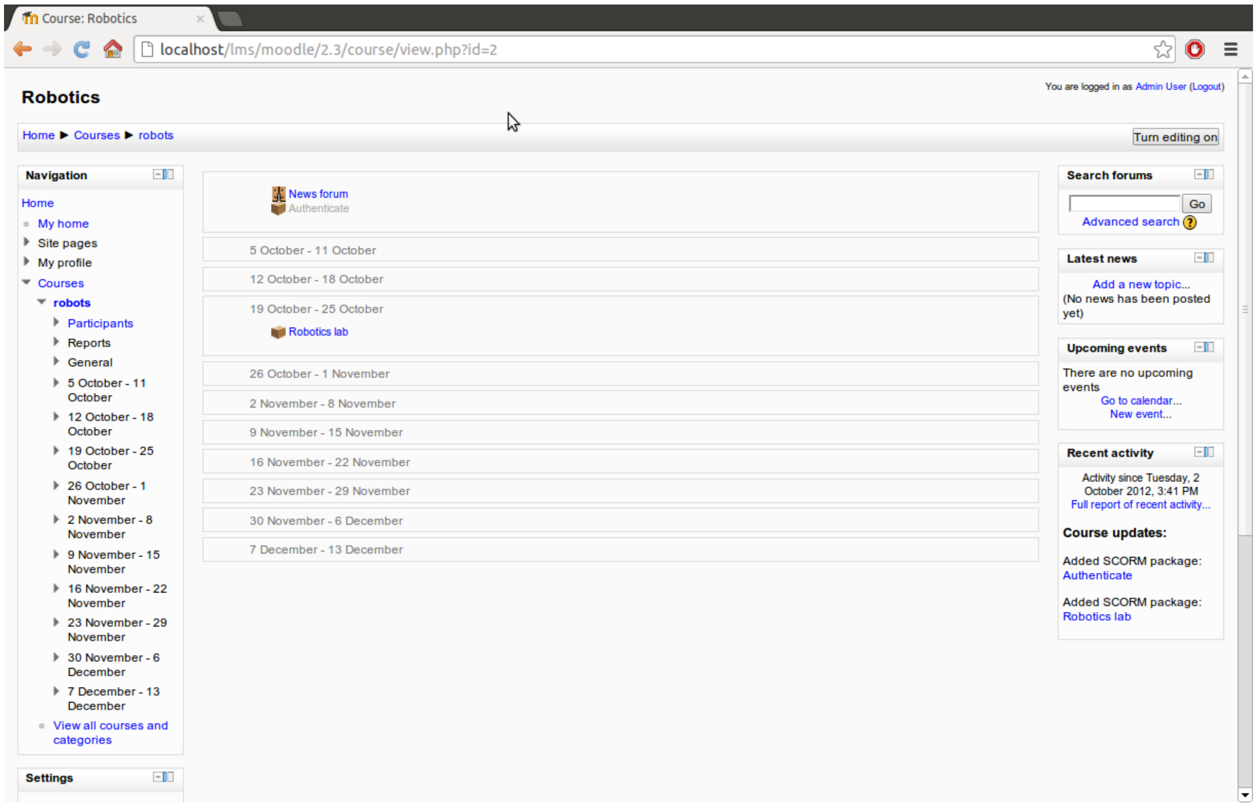
Description* This is a robotics lab with the SCORM for 'robot' generated by the LabManager

Path: p

Display description on course page ☐

Package file scorm_robot.zip

From that point, the SCORM object is just another SCORM object:


Course: Robotics

localhost/lms/moodle/2.3/course/view.php?id=2

Robotics

You are logged in as Admin User (Logout)

Home ▶ Courses ▶ robots

Turn editing on

Navigation
- Home
- My home
- Site pages
- My profile
- Courses
  - robots
    - Participants
    - Reports
    - General
    - 5 October - 11 October
    - 12 October - 18 October
    - 19 October - 25 October
    - 26 October - 1 November
    - 2 November - 8 November
    - 9 November - 15 November
    - 16 November - 22 November
    - 23 November - 29 November
    - 30 November - 6 December
    - 7 December - 13 December
  - View all courses and categories

Settings

News forum
Authenticate

5 October - 11 October

12 October - 18 October

19 October - 25 October
Robotics lab

26 October - 1 November

2 November - 8 November

9 November - 15 November

16 November - 22 November

23 November - 29 November

30 November - 6 December

7 December - 13 December

Search forums
Go
Advanced search

Latest news
Add a new topic...
(No news has been posted yet)

Upcoming events
There are no upcoming events
Go to calendar...
New event...

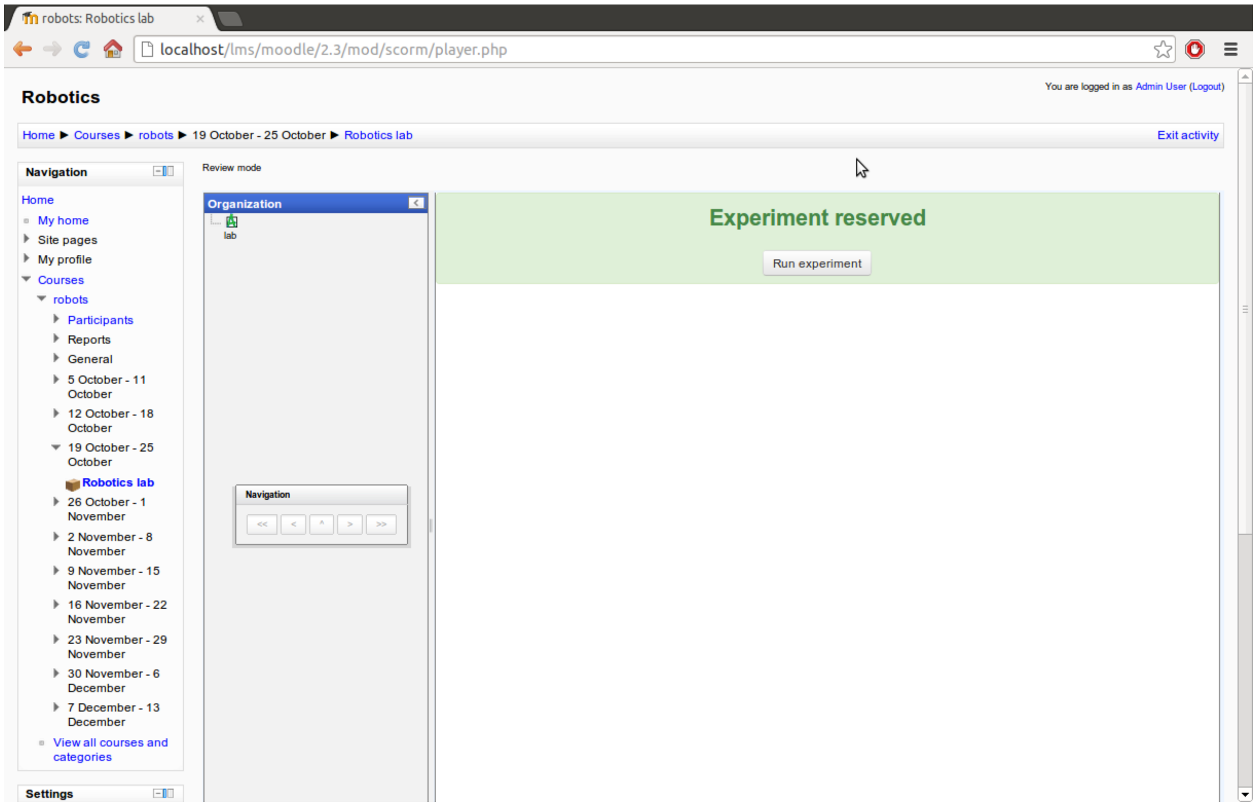
Recent activity
Activity since Tuesday, 2 October 2012, 3:41 PM
Full report of recent activity...

Course updates:

Added SCORM package: Authenticate

Added SCORM package: Robotics lab

And whenever a student enrolled in that course opens the SCORM object, a reservation will be performed, and the student can open the reservation:


robots: Robotics lab

localhost/lms/moodle/2.3/mod/scorm/player.php

Robotics

You are logged in as Admin User (Logout)

Home ▶ Courses ▶ robots ▶ 19 October - 25 October ▶ Robotics lab

Exit activity

Navigation
- Home
- My home
- Site pages
- My profile
- Courses
  - robots
    - Participants
    - Reports
    - General
    - 5 October - 11 October
    - 12 October - 18 October
    - 19 October - 25 October
      - Robotics lab
    - 26 October - 1 November
    - 2 November - 8 November
    - 9 November - 15 November
    - 16 November - 22 November
    - 23 November - 29 November
    - 30 November - 6 December
    - 7 December - 13 December
  - View all courses and categories

Settings

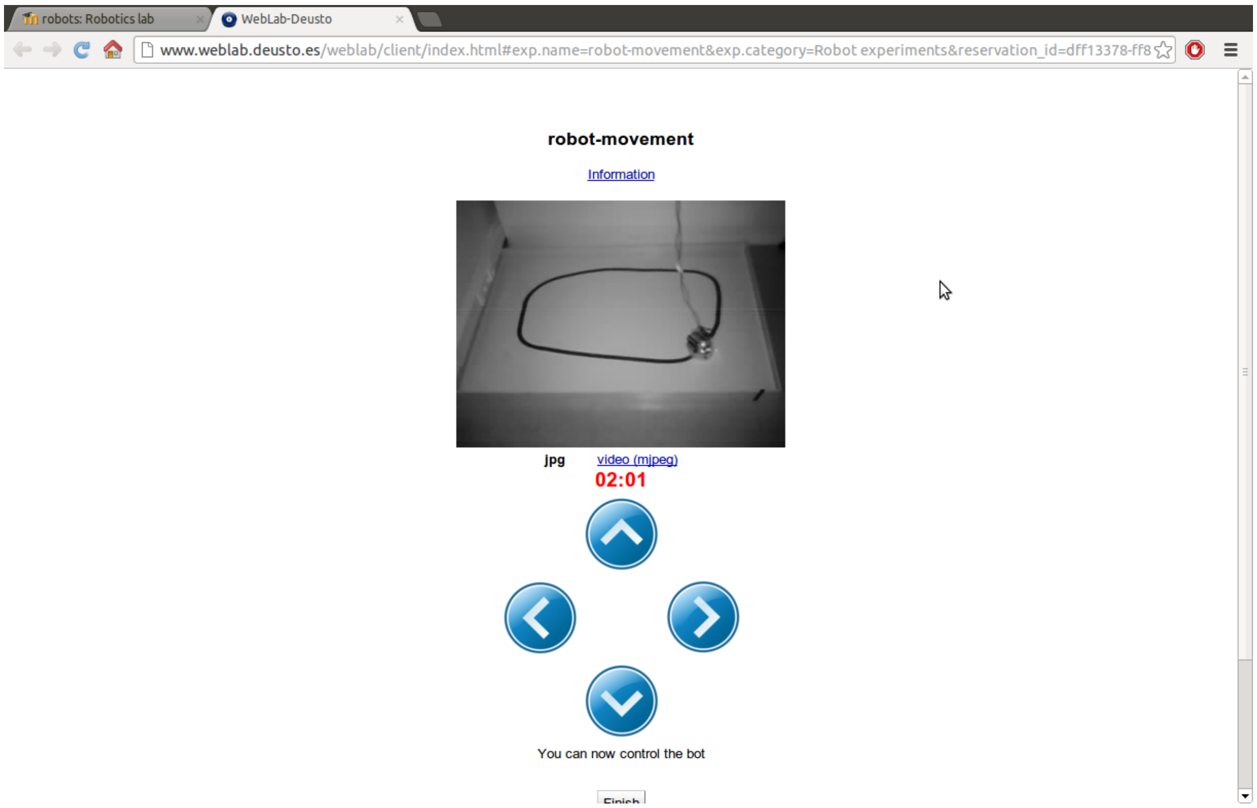
Review mode

Organization
lab

Navigation
<< < ^ > >>

Experiment reserved

Run experiment


robots: Robotics lab
WebLab-Deusto
www.weblab.deusto.es/weblab/client/index.html#exp.name=robot-movement&exp.category=Robot experiments&reservation_id=dff13378-ff8
robot-movement
Information
jpg
video (mjpeg)
02:01
You can now control the bot
Finish

CHAPTER 3

Developers

Generic documentation about how to create a new RLMS plug-in, how to support a new LMS which does not support LTI, and few little documentation about the internals of the LabManager

3.1 RLMS developers

3.2 LMS/CMS/PLE developers

3.3 LabManager developers

3.4 Professional services

LabsLand provides professional services on top of gateway4labs, such as consultancy services.

CHAPTER 4

Indices and tables

- genindex
- modindex
- search

Index