TFC - Tin Can API

Memòria

TFC Technology Enhanced Learning -2014/2015 - P2

Marcel Muntaner i Raich
44022013-J

UOC
www.uoc.edu

Taula de continguts

1 – Introducció 3
1.1 - Descripció 3
1.2 - Objectius 3
1.3 - Consideracions 4
2 - SCORM i Tin Can API en detall 5
2.1 - SCORM 5
2.2 - Tin Can API 8
3 - El procés d'instal·lació i configuració 9
3.1 - Camins descartats 10
3.2 - Instal·lació de Learning Locker 11
3.3 - Instal·lació de Moodle i de l'extensió Tin Can API 17
4 - Proves del sistema 20
5 - Conclusions 21

1 – Introducció

1.1 - Descripció

SCORM (Sharable Content Object Reference Model), és un estàndard amb especificacions per a objectes pedagògics. Aquestes especificacions, estableixen uns protocols de comunicació entre els mòduls pedagògics i les plataformes que els allotgen.

El principal avantatge de comptar amb aquest model és que si fem un mòdul que el compleix, sabem que serà compatible amb pràcticament qualsevol plataforma.

La versió 1.2 d'SCORM ha gaudit d'una gran popularitat, però requeria una renovació. Hi va haver algun intent d'evolucionar el model amb la versió SCORM 2004 (o 1.3), però aquesta mai no va arribar a desbancar la 1.2.

Recentment, a partir d'una recollida d'opinions de comunitats d'usuaris, pedagogs, dissenyadors i desenvolupadors, s'ha creat un nou model anomenat Tin Can API.

La intenció d'aquest model és oferir una eina més potent, més flexible i més senzilla d'utilitzar. Per aconseguir aquests objectius, als elements ja existents (mòdul i plataforma) se n'hi ha afegit un tercer: LRS (Learning Record Store).

Al principi, per tenir accés a un LRS, calia contractar alguna solució de pagament, però recentment ha sortit un LRS de codi obert: Learning Locker.

1.2 - Objectius

L'objectiu principal d'aquest treball és crear un entorn complet per provar Tin Can API.

Amb aquest entorn, es podrà fer una comparació amb SCORM i un anàlisi de què pot oferir Tin Can API.

Personalment, fa anys que que utilitzo SCORM. Si trobeu a faltar referències bibliogràfiques, és perquè no em calen per explicar què és i com funciona. Però, tot i saber quins són els principis i la història de Tin Can API (vaig participar en la recollida de requisits que es va fer prèviament a la realització de l'estàndard), encara no havia tingut la ocasió d'experimentar-hi. És per això que tinc un gran interès en la realització d'aquest treball.

Un objectiu secundari és obtenir experiència amb les tecnologies requerides per crear aquest entorn. Learning Locker està construït amb tecnologies molt noves i cal familiaritzar-s'hi per tal de portar a terme la seva instal·lació, ja que encara està en fase beta i no ofereix un sistema d'instal·lació automàtic. Màquines virtuals, servidors, llenguatges de programació de costat de servidor o de costat de navegador, frameworks, editors, APIs, plugins, etc. Tot plegat forma un ecosistema que proveeix una visió molt completa de les aplicacions d'internet.

1.3 - Consideracions

He llegit amb interès els apunts sobre la presentació i la redacció.

Crec que són de gran utilitat i he intentat seguir-ne les indicacions, amb alguna excepció que no he seguit de manera expressa.

Destacar, a part de la manca de bibliografia (que ja he esmentat anteriorment), que no he adaptat la terminologia tècnica amb l'ús del Termcat, sinó que he utilitzat els termes que s'utilitzen a nivell professional, arrel de la meva experiència. Vull constatar, a més, que estic en particular desacord amb aquest tipus de traduccions de termes tècnics, que no veig que responguin a criteris d'harmonització o d'eficiència.

2 - SCORM i Tin Can API en detall

2.1 - SCORM

SCORM és un estàndard establert per ADL, acrònim de Advanced Distributed Learning.

El nom SCORM també és un acrònim, en aquest cas de Sharable Content Object Reference Model; és a dir Model de Referència per a Objectes de Contingut Compartible. Aquest acrònim és a l'hora una definició bastant completa, a manca, potser, d'una referència a que és per a objectes relacionats amb la formació.

Analitzem les dues parts principals d'aquesta 'definició':

- Model de referència: La característica principal d'SCORM és que estableix un model, que també podríem anomenar protocol o API. Aquest model és un conjunt d'operacions que tenen una sintaxi determinada i que obtenen uns resultats determinats. És el protocol o API (Application Programming Inerface) entre un mòdul i una plataforma.
- Objectes de Contingut Compartible: Amb SCORM podem crear objectes de contingut que seran compatibles amb qualsevol plataforma que segueixi les especificacions del model. La plataforma pot estar construïda amb qualsevol llenguatge de servidor (PHP, Java, .Net, etc), mentre el reproductor SCORM tingui els fitxers JavaScript que contenen les funcions de la API, que alhora són cridades per les funcions que hi ha a l'objecte de contingut, també en un fitxer Javascript. Els objectes han de fer servir HTML i JavaScript, però també poden utilitzar d'altres llenguatges de navegador com Flash.

Vegem aquest model amb una mica més de detall.:

- L'objecte de contingut o mòdul està composat per:
    - El contingut, que són una sèrie de fitxers HTML, que alhora fan servir JavaScript i CSS i que poden fer servir Flash o d'altres llenguatges de costat de navegador. Cada un d'aquests fitxers HTML és una unitat que s'anomena SCO.
    - L'API, que són una sèrie de funcions JavaScript que poden ser cridades pel contingut HTML i que alhora criden funcions de la API del reproductor SCORM de la plataforma.
    - Fitxers de configuració XML, entre els quals destaca el imsmanifest.xml, que conté informació específica del mòdul com ara:
        - La ubicació dels fitxers de contingut

        - L'ordre i condicions de seqüenciació
        - Notes màxima, mínima i de tall
        - Idioma
        - Nom del mòdul
        - Etc.

- La plataforma ha de tenir:
    - Un sistema per pujar i emmagatzemar els mòduls SCORM.
    - Un reproductor per obrir els mòduls i que comptarà amb una API JavaScript amb una sèrie de funcions que poden ser cridades pel mòdul i que emmagatzemaran dades a la base de dades de la plataforma o en recuperaran informació emmagatzemada.
    - Una base de dades, que tindrà una entrada per cada parella mòdul SCORM – Usuari, i també una entrada per cada SCO (cada una de les unitats del mòdul SCORM) – Usuari que hi ha accedit. A l'entrada de cada SCO hi ha una sèrie de variables, que és on es guarda les dades de seguiment de l'usuari en aquell SCO. Entre aquestes variables podem destacar:
        - cmi.core.score.raw: puntuació obtinguda.
        - cmi.lesson_status: ens diu si la unitat està començada, completada, aprovada...
        - cmi.core.lesson_location: generalment s'utilitza per precisar la ubicació on es troba l'usuari en aquesta unitat.
        - cmi.suspend_data: generalment s'utilitza per emmagatzemar informació que després es necessiti per recuperar l'estat en el que es deixa una unitat en sortir-ne.

  Una alta manera de veure i entendre què és SCORM i com es fa servir seria amb un cas d'us complet:

  1. Un equip de desenvolupament de continguts (format per un formador, un dissenyador i un desenvolupador) fa un mòdul SCORM, format per tres unitats.
  2. Un administrador crea un curs en una plataforma de formació i el formador hi afegeix continguts, entre ells el mòdul SCORM.
  3. Un usuari accedeix al mòdul SCORM. Completa la primera unitat i part de la segona.

   a. Obre el mòdul
   b. El mòdul demana a la plataforma l'estat de la primera unitat.
   c. La plataforma diu que està per començar.
   d. El mòdul inicialitza i es posa al principi de la unitat.
   e. L'usuari completa la unitat i treu una nota de 7.
   f. La plataforma emmagatzema la nota a cmi.core.score.raw i posa la variable cmi.lesson_status a l'estat 'passed'.
   g. El mòdul mostra la segona unitat.
   h. L'usuari surt quan ha completat 3 de les 10 preguntes de la unitat.
   i. El mòdul envia la informació a la plataforma i aquesta emmagatzema a cmi.lesson_status l'estat 'incomplete', a cmi.core.lesson_location la cadena '3#10' i a cmi.suspend_data la cadena 'c#a#b#', que són les respostes que seleccionat l'usuari. La codificació de les cadenes cmi.core.lesson_location i cmi.suspend_data, no segueix un model definit. És una codificació que utilitza el mateix mòdul per poder recuperar l'estat de la unitat. En el cas de cmi.core.lesson_location, de vegades hi ha plataformes que la utilitzen per fer informes més complets i els desenvolupadors, poden utilitzar el format establert per la plataforma per tal d'oferir una experiència més completa.

4. El Formador accedeix a la plataforma i genera un informe de seguiment. Aquest informe de seguiment accedeix a les dades:
   a. score_raw, que informa al que l'usuari ha tret un 7 de la unitat 1
   b. cmi.lesson_status, que informa que l'usuari ha aprovat la primera unitat 1 i començat la segona.
   c. cmi.core.lesson_location, que (com que en aquest cas està codificada en un format que la plataforma també pot desxifrar) informa que l'usuari ha completat el 100% de la primera unitat i el 30% de la segona.
5. L'usuari torna a accedir al mòdul SCORM, des d'un altre dispositiu:
   a. Obre el mòdul
   b. El mòdul demana a la plataforma l'estat de la primera unitat.
   c. La plataforma diu que està completada.

d. El mòdul demana a la plataforma l'estat de la segona unitat.

e. La plataforma proveeix les dades de les variables de la unitat.

f. El mòdul mostra la unitat tal com l'usuari l'havia deixat.

6. Multitud d'usuaris interaccionen amb la plataforma i el mòdul amb diferents dispositius. En els informes que veu el formador es correspon sempre les dades mostrades amb l'usuari que ha generat les dades. Quan els usuaris Tornen a accedir als continguts cadascú recupera l'estat en el que havia quedat.

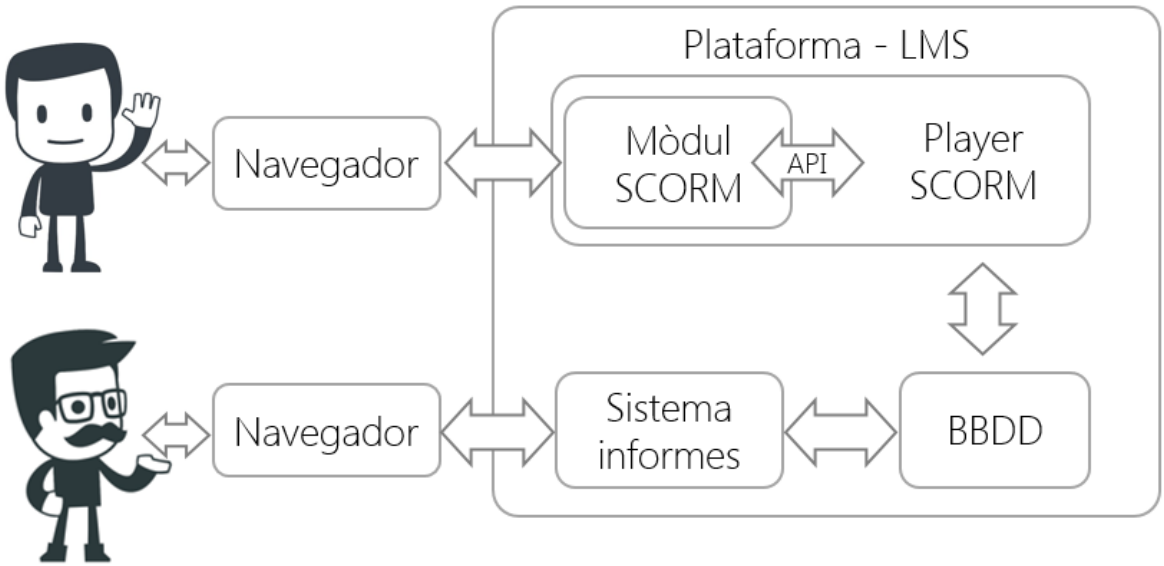


Figura: L'ecosistema SCORM

2.2 - Tin Can API

Tin Can API, va néixer amb la voluntat de millorar SCORM, amb la superació d'algunes limitacions i amb eines noves d'utilitat per desenvolupadors, formadors i alumnes.

Per tal que aquest nou model fos satisfactori per la comunitat d'usuaris, ADL, en associació amb Rustici Software, una de les empreses que més havia col·laborat en l'especificació i l'evolució d'SCORM, van fer una recollida de requeriments d'alt i baix nivell de tot tipus d'usuaris.

Algunes de les peticions més repetides van ser:

a) Més facilitat d'us per als desenvolupadors de continguts.

b) Possibilitat de fer visualitzar continguts amb el mòbil.

c) Possibilitat de realitzar continguts que no fossin necessàriament allotjats a la plataforma.

d) Possibilitat de realitzar continguts amb connexió intermitent.

e) Sistema de registre i d'informes més complet i flexible.

f) Poder emmagatzemar dades d'un mateix usuari a diferents plataformes.

Per tal d'aconseguir satisfer els requeriments la nova especificació compta amb les següents característiques:

- Sistema de transmissió REST, que emmagatzema la informació al dispositiu i l'envia quan hi ha connexió (b, c i d)
- Creació d'un nou element: LRS (Learning Record Store) que emmagatzemi les interaccions dels usuaris amb els mòduls que ens interessi (c, e, f)
- Possibilitat d'emmagatzemar amb frases naturals, sense limitacions (a, e)

La novetat més trencadora d'aquest nou model era la inclusió de l'LRS. Si bé és veritat que ens possibilita aquest sistema més flexible, menys encorsetat, més potent i fàcil d'utilitzar per formadors i desenvolupadors de continguts, afegeix molta complexitat pels desenvolupadors de plataformes.

Inicialment, només va sortir un LRS que era de pagament per ús, i que va realitzar i comercialitza Rustici.

Però com el model és obert, poc a poc han anat sortint d'altres alternatives, fins que ha sorgit Learning Locker, que és gratuït i de codi obert.

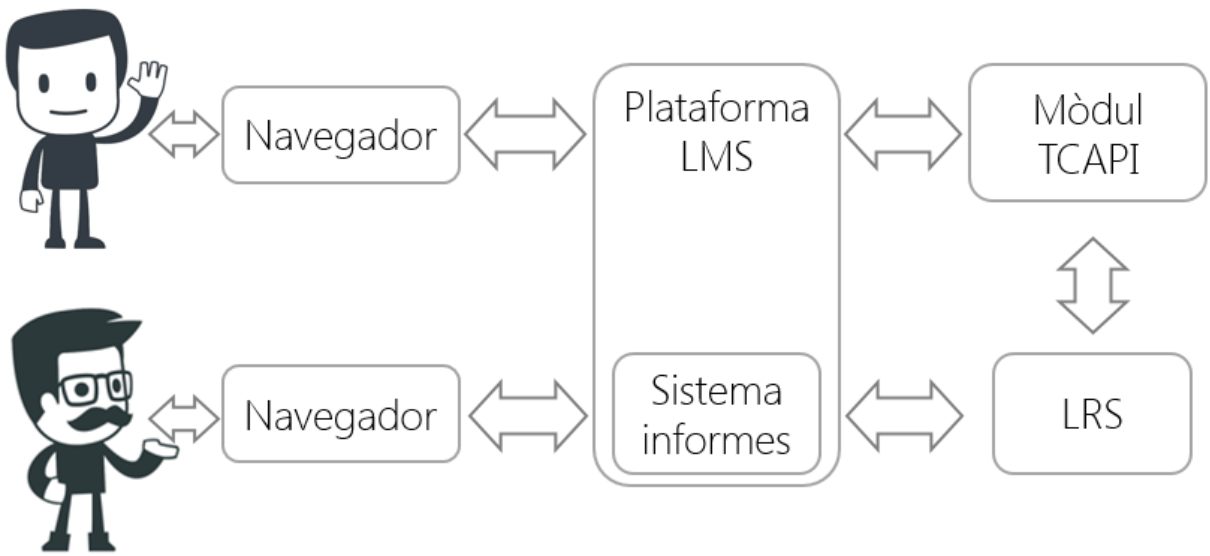


Figura: L'ecosistema Tin Can API

3 - El procés d'instal·lació i configuració

3.1 - Camins descartats

La instal·lació de l'entorn complet Tin Can API, ha estat un camí ple de reptes i dificultats que en més d'una ocasió ha suposat haver de fer canvis d'estratègia.

Estic content, però, de poder afirmar que els objectius s'han complert amb escreix, ja que finalment he pogut portar a terme la instal·lació de l'entorn complet; amb l'avantatge respecte al pla inicial que aquest no estarà en una màquina local sinó en un servidor accessible a internet. A més, les dificultats han fet necessari fer més investigació i obtenir un millor coneixement de les tecnologies que formen part d'aquest entorn.

El pla inicial era instal·lar Learning Locker en una màquina virtual, Homestead, feta pel desenvolupament amb Laravel, el Framework PHP amb què està fet Learning Locker.

Un cop aconseguida la instal·lació i funcionament de Homestead i Laravel, en intentar instal·lar-hi Learning Locker, no ho vaig aconseguir. Com que a la documentació oficial de la instal·lació de Learning Locker especifica que està construït sobre Laravel, però no ho enumera com a requisit, vaig decidir provar de fer la instal·lació en una màquina virtual diferent, seguint les instruccions a:
http://www.jpablo128.com/como_instalar_learning_locker/

De tota manera, he guardat el box Homestead i ara tinc aquest entorn preparat per quan vulgui experimentar, que serà aviat, perquè he pogut veure que és un Framework molt ben pensat i interessant.

També m'ha anat molt bé descobrir i experimentar la manera de treballar amb entorns de desenvolupament 'enllaunats' en un box o màquina virtual, utilitzant Vagrant. A partir d'ara faré tot el desenvolupament web així, enlloc d'eines com Wamp o Xampp.

Més endavant, amb Learning Locker funcionant i per un altre costat Moodle i l'extensió Tin Can API també funcionant, ambdós en una màquina virtual local, vaig veure que aquestes dues aplicacions no es comunicaven. Això em va fer decidir provar de pujar el sistema a internet i va funcionar.

Vaig contractar un VPS (Virtual Private Server), que és una màquina virtual (amb S.O. Ubuntu 14.04) connectada a internet.

Vaig fer la instal·lació seguint exactament els mateixos passos que vaig seguir per la instal·lació local, però connectant-me per IP amb SSH a la màquina virtual.

Com a resultat, es pot accedir a Learning Locker a:
http://176.31.171.10/learninglocker/public/

Amb usuari(email): marcelmuntaner@gmail.com i contrasenya: lrstest

I a Moodle a: http://176.31.171.10/moodle

Amb usuari: admin i contrasenya: lrstest

En la explicació dels passos per a la instal·lació de l'entorn complet, he omès explicar els camins que finalment s'han descartat. Algunes de les captures de pantalla contenen adreces i configuracions d'aquests camins, però la intenció d'aquestes captures és només il·lustrar el pas descrit.

3.2 - Instal·lació de Learning Locker

Un cop connectats al ssh de l'VPS, instal·lem mongodb:

```
sudo apt-key adv—keyserver hkp://keyserver.ubuntu.com:80 --recv
7F0CEB10

echo 'deb http://downloads-distro.mongodb.org/repo/ubuntu-
upstart dist 10gen' | sudo tee
/etc/apt/sources.list.d/mongodb.list

sudo apt-get update

sudo apt-get install -y mongodb-org
```

Tot seguit instal·lem una sèrie de requisits més amb una sola comanda:

```
sudo apt-get install curl nano apache2 php5 php5-mongo php5-
mcrypt git node npm mysql-server postfix
```

En ser preguntats, establim la contrassenya de root a mysql i la configuració de Postfix (internet site i deixar el nom per defecte).

Provem els serveis:

```
sudo service apache2 status

sudo service mongod status

sudo service mysql status

sudo service postfix status

php -v
```

Habilitem mycript:

```
sudo php5enmod mcrypt
```

instalem bower:

```
sudo npm install -g bower
```

instalem composar:

```
curl -sS https://getcomposer.org/installer | php

sudo mv composer.phar /usr/local/bin/composer
```

Ara ja tenim tots els requisits funcionant, podem procedir a la instal·lació de Learning Locker.

Clonem Learning Locker:

```
cd /var/www/

sudo git clone
https://github.com/LearningLocker/learninglocker.git
learninglocker
```

establim base de dades i usuari de mongodb:

```
mongo

use learninglocker

db.createUser( { user: "learnlockuser", pwd: "learnlockpwd",
roles: [ "readWrite" ] } );

exit

sudo service mongodb restart
```

Editem l'arxiu de configuració de la base de dades de learning Locker:

```
sudo vim /var/www/learninglocker/app/config/database.php
```

(a l'objecte mongodb hi posem l'usuari i password que hem establert a la línia de comandes de mongo)

Finalment, instal·lem learning Locker:

```
cd /var/www/learninglocker/

sudo composer install
```

Un error ens diu que hi ha un parèntesi inesperat a la línea 157 a app/locker/helpers/Helpers.php i s'aborta la instal·lació.

Tornem a clonar learning locker, editem el fitxer Helpers.php i li treiem un parèntesi a la línea indicada.

Tornem a editar el fitxer database.php

Tornem a instal·lar i ens dóna un error al final, però la instal·lació acaba i llavors fem la migració de la base de dades amb:

```
sudo php artisan migrate
```

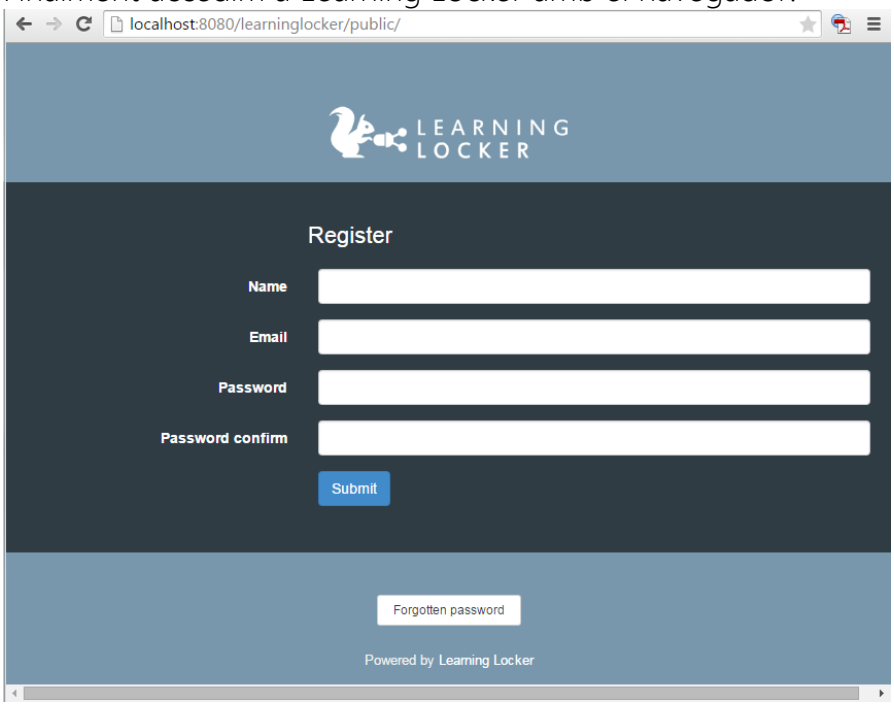
Editem la línia 57 de app/config/mail.php per posar una direcció de mail i nom correctes

Habilitem alguns mòduls d'apache i el reiniciem:

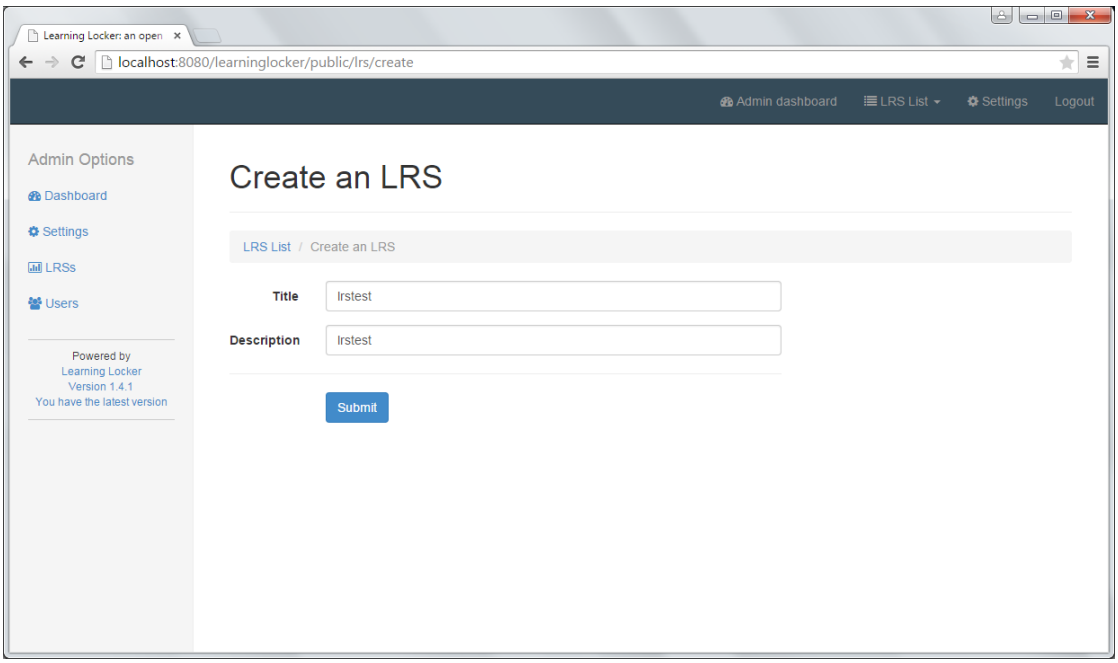
```
sudo a2enmod rewrite negotiation php5

sudo service apache2 restart
```

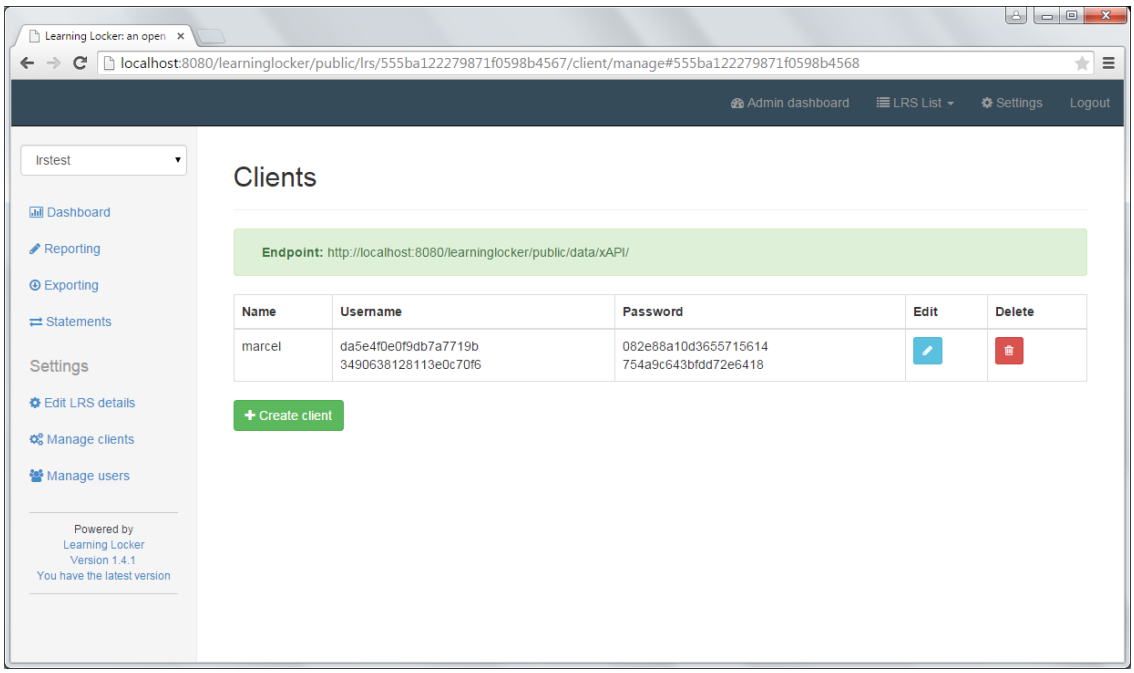
Finalment accedim a Learning Locker amb el navegador:



Tot seguit hem creat un LRS (Learning Record Store) i hem enviat un 'Statement' des d'una pàgina de prova per assegurar que funcionés Learning Locker.



Creem l'LRS

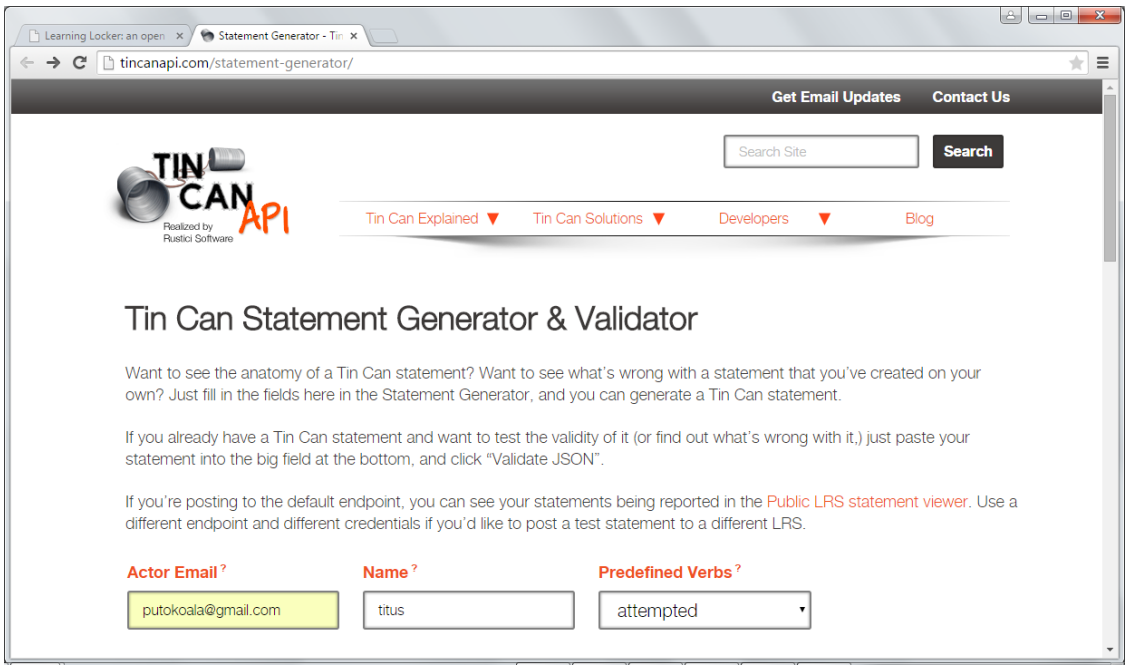


Clicant a 'Manage clients' podem veure la informació que necessitem per enviar 'statements' a l'LRS.

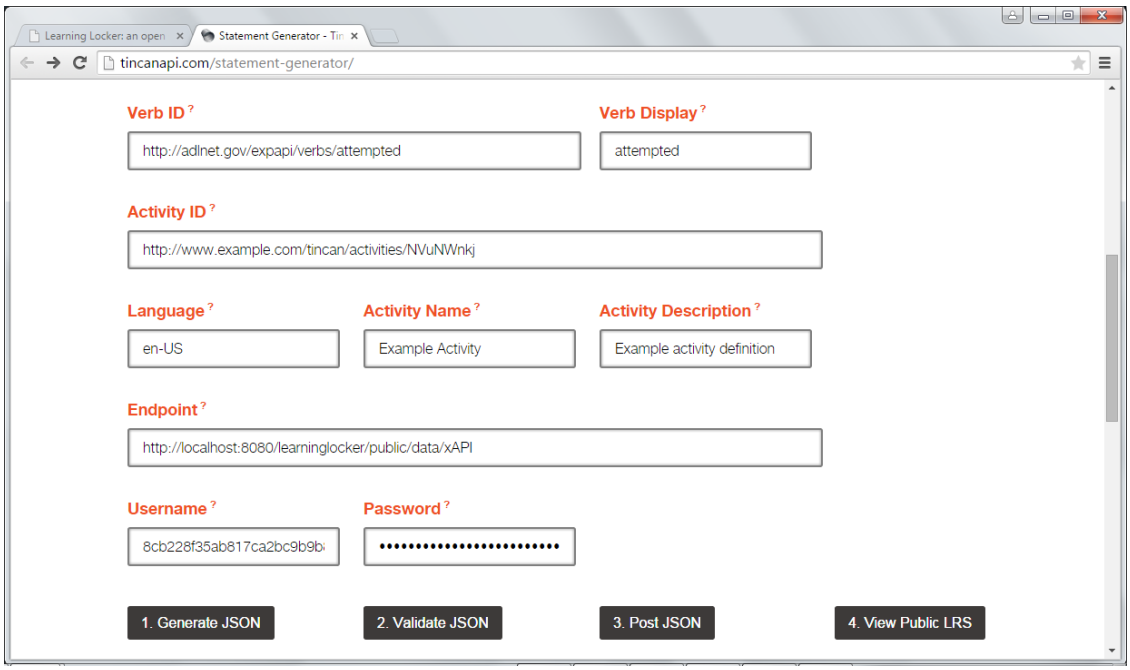
Ara podem anar a crear un statement.

Cal recordar, que un dels avantatges que té Tin Can API, és que podem enviar informació des de qualsevol lloc si tenim la informació del nostre LRS.

En aquest cas enviarem un statement des d'una eina que ens posa a disposició la pàgina oficial de Tin Can API.

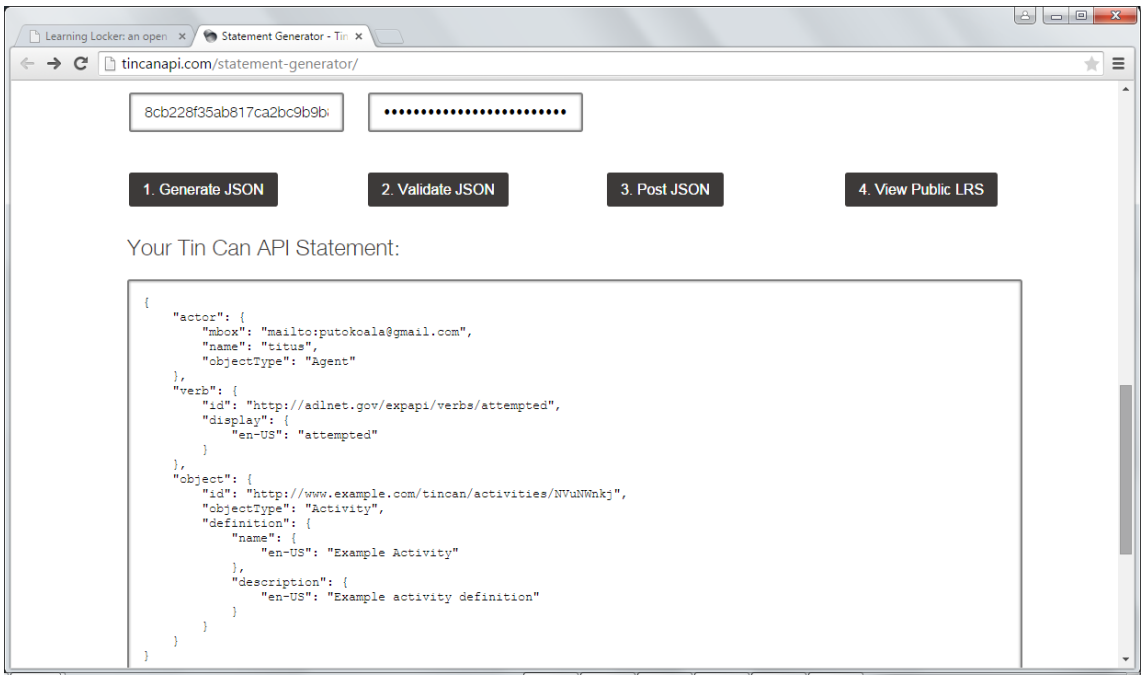


Posem email i nom de la persona que fa l'acció i escollim un verb.

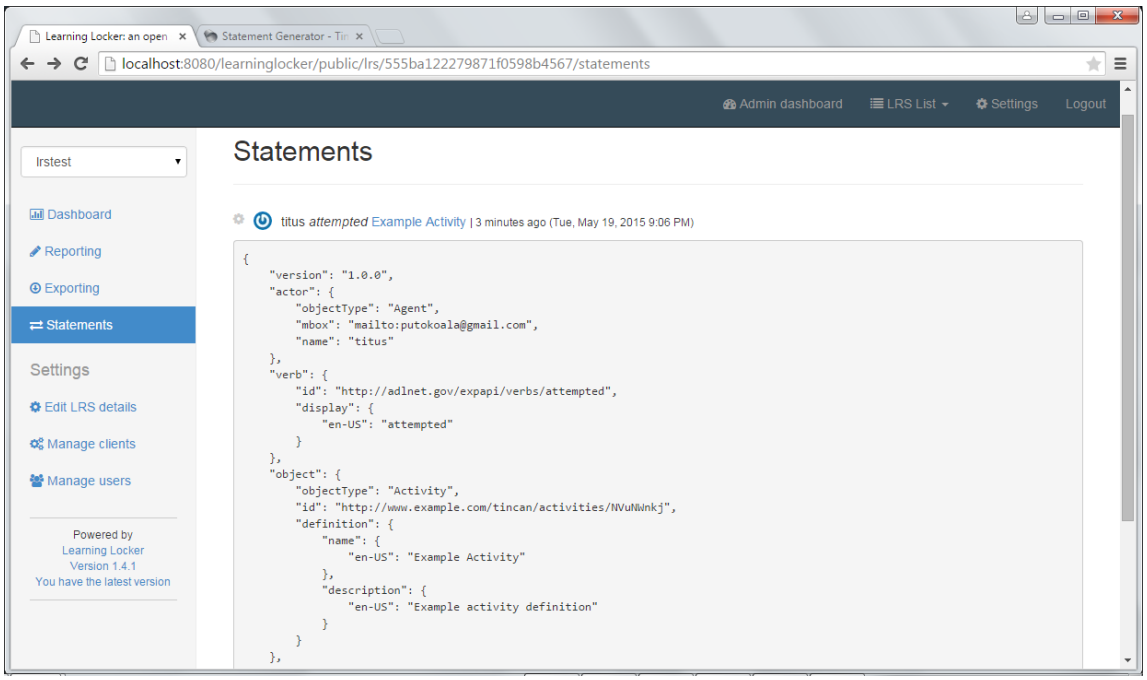


Aquí és molt important posar l'endpoint, username i password que hem trobat a l'apartat 'Manage Clients' del nostre LRS.

La resta d'informació és sobre l'acció que se suposa que volem guardar que s'ha fet, però com només és per propòsit de prova, ho deixem com està per defecte.



Generem el JSON (objecte JavaScript – JavaScript Object Notation), el validem i l'enviem



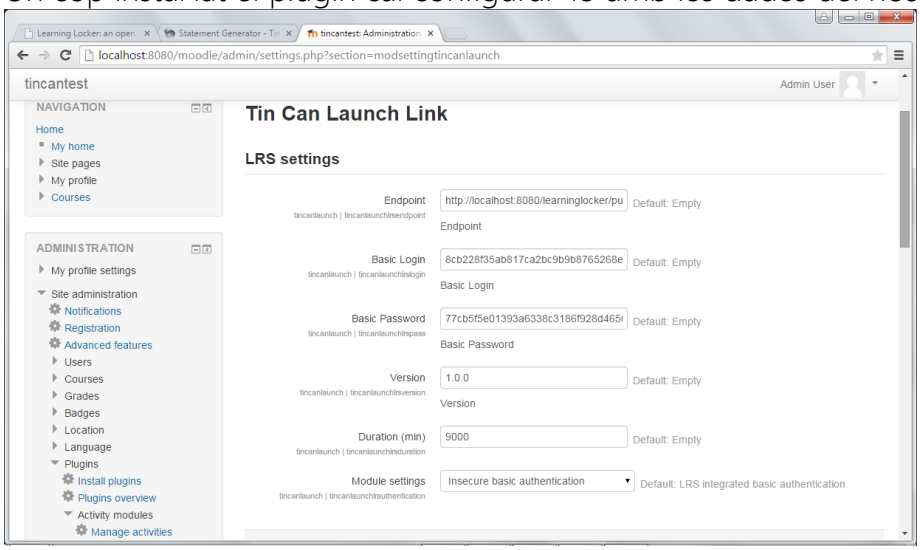
Tornant a Learning Locker i clicant a 'Statements' podem veure com la frase que acabem d'enviar, ha arribat correctament.

3.3 - Instal·lació de Moodle i de l'extensió Tin Can API

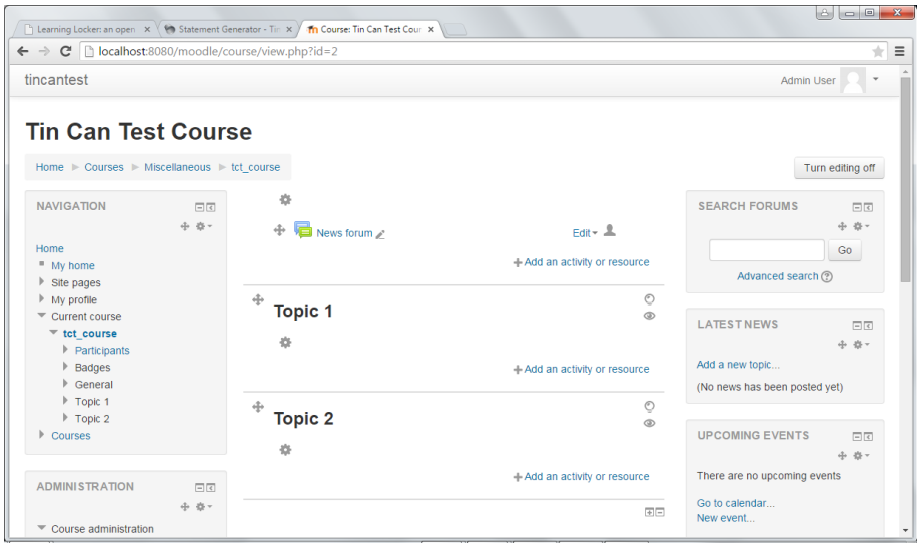
Per instal·lar Moodle he seguit al peu de la lletra les instruccions a:
https://docs.moodle.org/28/en/Step-by-step_Installation_Guide_for_Ubuntu

Amb Moodle instal·lat he descarregat i instal·lat el plugin:
https://moodle.org/plugins/view/mod_tincanlaunch

Un cop instal·lat el plugin cal configurar-lo amb les dades del nostre LRS

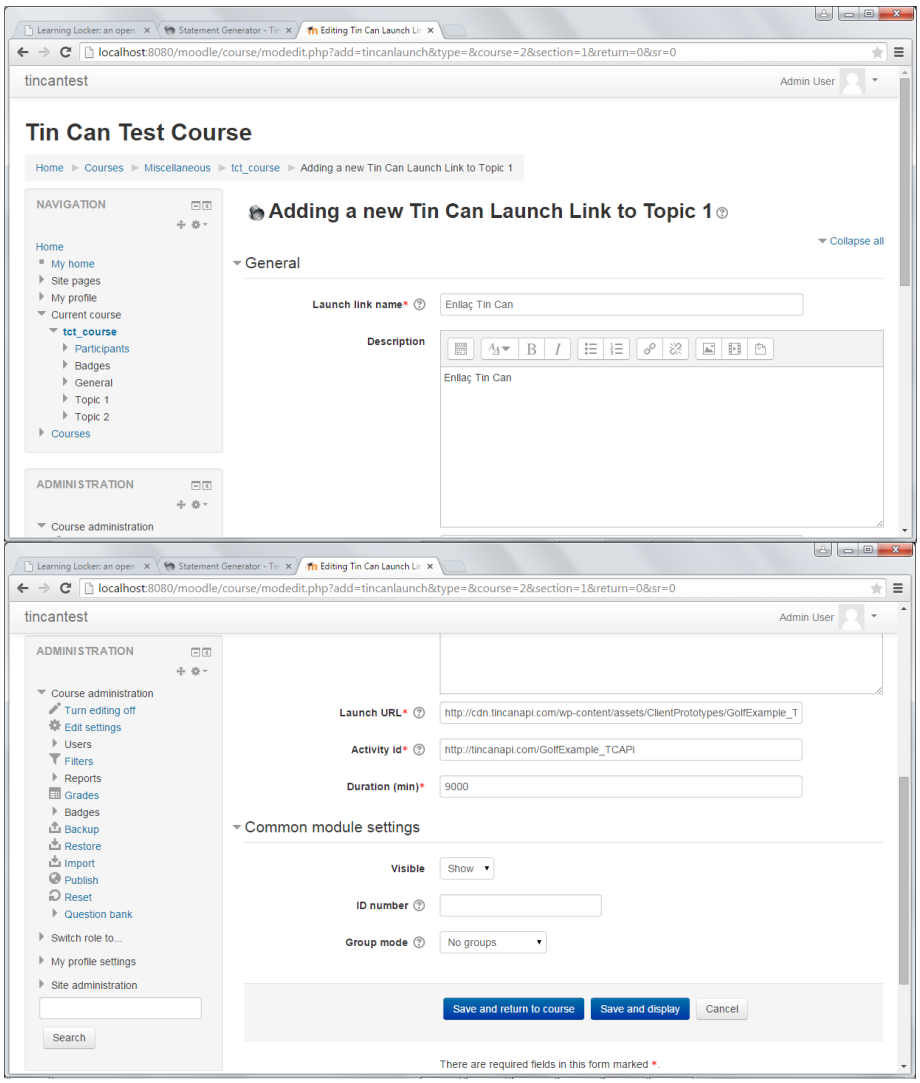


Un cop configurat el plugin, crearem un curs a Moodle per posar-hi l'activitat Tin Can API.

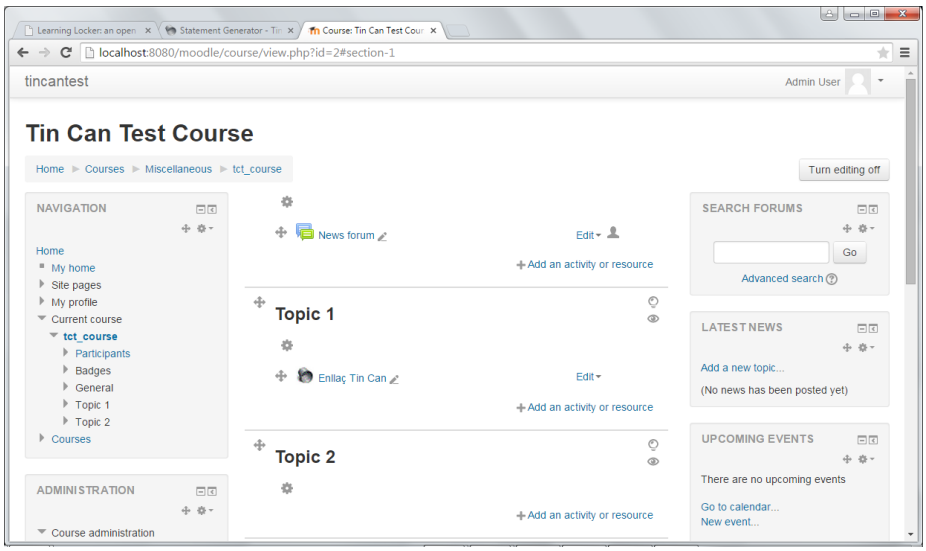


Aquest és el curs.

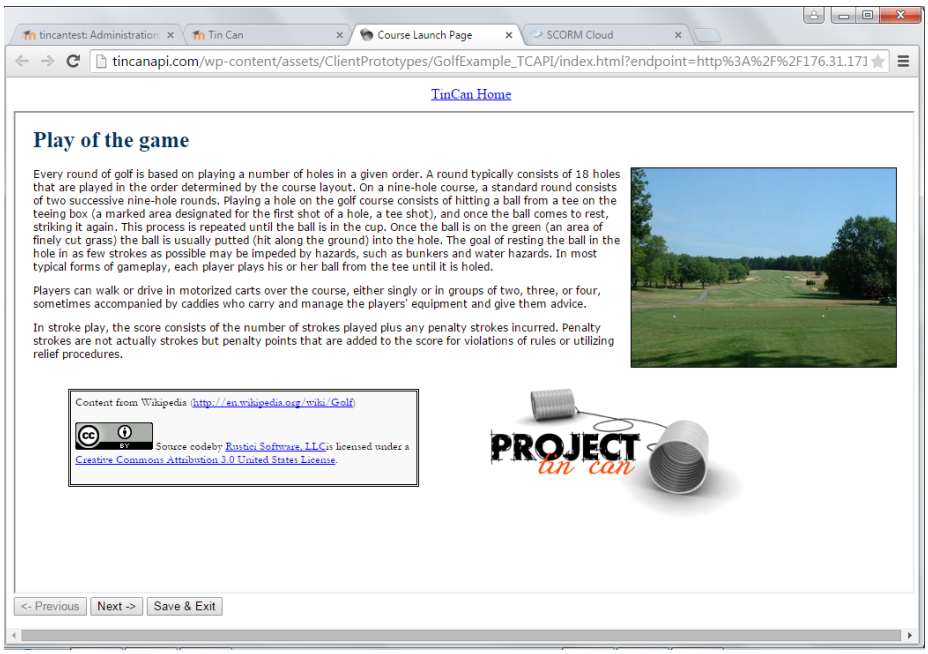
Cliquem 'Add an activity or resource' i seleccionem 'Tin Can Launch Link'.



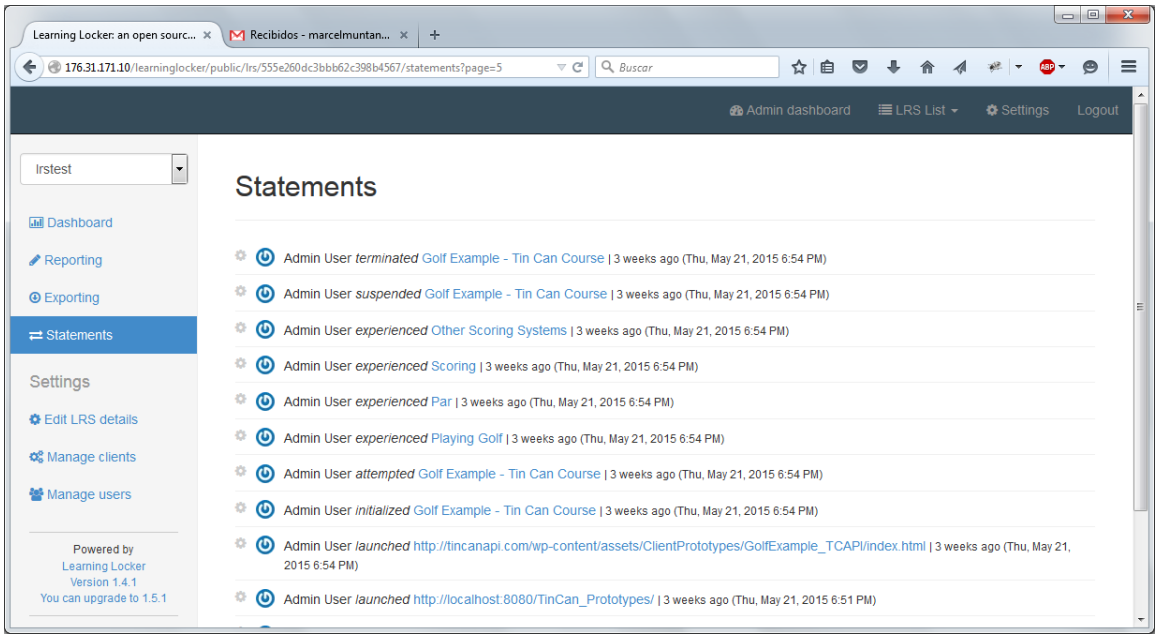
Configurem l'enllaç per tal que obri amb un mòdul de prova posat a disposició per la gent que han fet l'especificació de Tin Can API.



Ja tenim l'enllaç al mòdul Tin Can, el cliquem.



Veiem el contingut.



I veiem que a l'LRS s'emmagatzema informació.

4 - Proves del sistema

Un cop amb el sistema complet funcionant, he fet proves per assegurar que funciona al 100% però també per esbrinar quin és l'abast en termes de funcionalitat de Tin Can API.

Les proves han estat:

1. Crear un curs a Moodle amb un enllaç a una activitat Tin Can API i accedir a l'activitat: El sistema funciona correctament, s'accedeix correctament a l'activitat.

2. Mirar si s'emmagatzemen els registres d'activitat: L'LRS reflexa l'activitat de l'usuari amb diversos registres que queden correctament emmagatzemats. Aquests registres tenen un format molt més flexible que els registres del predecessor de Tin Can API (SCORM). Tin Can API, permet un format molt més lliure donant al desenvolupador de continguts un gran ventall de possibilitats.

3. Comprovar que l'activitat recupera l'estat en el que havia quedat: A més de generar registres per tal que es pugui veure l'activitat dels usuaris, una de les característiques que ja tenia SCORM, era que es podia configurar per recuperar l'estat en el que havies deixat una activitat. És a dir, com a usuari si feies un mòdul de 100 pàgines, podies sortir a la pàgina 30 i quan tornaves a obrir l'activitat et portava a la pàgina 30 (també podia recuperar més informació sobre l'estat en que havies deixat el curs). Aquest era un dels dubtes que tenia sobre Tin Can API, si mantindrien aquesta funcionalitat. He pogut comprovar que sí. He provat de sortir en diferents pàgines de l'activitat Tin Can API i quan hi torno a accedir em recupera la darrera posició on estava.

4. Comprovar que diferents usuaris recuperen les seves respectives posicions: Relacionat amb el punt anterior, de vegades hi ha activitats que recuperen la posició a través de la memòria cache del navegador, si aquest fos el cas, no seria satisfactori, ja que volem poder-nos connectar des de qualsevol dispositiu i amb més d'un dispositiu per part de cada usuari. També volem comprovar que el sistema d'usuaris sigui consistent i recuperi a cada usuari la seva posició. Per això hem creat dos usuaris diferents i ens hem connectat a la mateixa activitat amb cada un dels usuaris més d'una vegada i des de dispositius diferents. El resultat ha estat positiu, doncs a cada usuari li ha recuperat la posició que li correspon, independentment del dispositiu o navegador amb el que es connectés.

5 - Conclusions

- Tin Can API compleix amb els seus objectius: ofereix més flexibilitat, funcionalitats i facilitat pels usuaris, sense perdre cap capacitat respecte a SCORM.
- Per ara la configuració és molt més complexa, esperem que properament hi haurà solucions de codi obert fàcils d'instal·lar i integrades.
- El sistema d'informes encara està per madurar, però será un punt clau en l'evolució de Tin Can API i de l'eLearning en general. La possibilitat d'emmagatzemar la informació d'una manera més flexible ens obre un món nou en la interacció amb els usuaris. Un bon ús d'aquesta informació ens podrà permetre fer formació més individualitzada de manera més automàtica, dos conceptes que fins a dia d'avui sembla que siguin oposats.