Integración de juegos educativos en edX

Autores:
Ricardo Javier Carrión Beltrán
María Díaz Calvo
Daniel López Carreras

SISTEMAS INFORMÁTICOS
FACULTAD DE INFORMÁTICA

UNIVERSIDAD COMPLUTENSE DE MADRID

PROYECTO DE INGENIERÍA INFORMÁTICA

Madrid, 20 de junio de 2014

Directores:
Dr. Baltasar Fernández Manjón
Dr. Manuel Freire Morán

Autorización de difusión y utilización

Se autoriza a la Universidad Complutense de Madrid a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria como el código, la documentación y/o el prototipo desarrollado.

Ricardo Javier Carrión Beltrán María Díaz Calvo Daniel López Carreras

Madrid, 20 de junio de 2014

Agradecimientos

Gracias a nuestros familiares y amigos por ayudarnos y animarnos en los momentos más difíciles, y agradecerles también su apoyo incondicional durante todo el año, ya que sin ellos hubiese sido mucho más difícil.

Agradecer profundamente al director de este proyecto, Dr. Baltasar Fernández Manjón, su orientación y numerosos consejos, ya que gracias a él hemos aprendido mucho y hemos sido capaces de sacar este proyecto adelante pese a las numerosas adversidades que nos han ido surgiendo a lo largo del año.

También mostrar un gran agradecimiento a nuestro codirector, Dr. Manuel Freire Morán, por su gran ayuda en el desarrollo de este proyecto. Hemos aprendido mucho de él, y sin sus conocimientos habría sido muy difícil alcanzar los objetivos.

Índice

Índice de imágenes

Resumen

En este proyecto se llevará a cabo una investigación de las diferentes alternativas para integrar juegos educativos en plataformas de aprendizaje (MOOCs), concretamente en edX. El objetivo principal es introducir juegos realizados en JavaScript para hacer más interactivo el aprendizaje de los alumnos. Esto permitirá que los juegos educativos realizados con la plataforma eAdventure (desarrollada en la Universidad Complutense de Madrid) puedan formar parte del bloque de ejercicios prácticos de un curso edX una vez se haya finalizado la versión 2.0, que tendrá la capacidad de exportar los juegos en JavaScript y HTML5.

Al ser juegos educativos, se puede obtener mucha más información de las interacciones del alumno, y se podrá realizar un análisis posterior. Para realizar este análisis se utilizará xAPI, ya que la herramienta de análisis de resultados de los alumnos integrada en edX (Insights), está aún en fase de desarrollo. De esta forma los juegos permitirían, por ejemplo, evaluar el conocimiento que han adquirido los estudiantes, o comprobar si en una situación concreta se aplican bien esos conocimientos teóricos. Así se conseguiría una interactividad mucho mayor y se podrían tener muchos más datos para mejorar la evaluación o el rendimiento de los alumnos.

Palabras clave:

eAdventure

edX

xModule

Lon-Capa

LTI

xAPI

LRS

Abstract

In this project, an investigation will be carried out to integrate educational games in learning platforms (MOOCs), concretely in edX. The main objective is to introduce games made in JavaScript to make pleasant the learning of the students. This will allow educational eAdventure games (developed at the Complutense University of Madrid) to be part of edX practical exercises in a course when eAdventure 2.0 has been finished, which it will be able to export games to JavaScript and HTML5.

Being educational games, you can get much more information from the students interactions, for a further analysis. XAPI is going to do this analysis because the result analysis tool integrated in edX (Insights) is still under development. This game would allow, for example, evaluate the knowledge that students have acquired, or check if in a particular situation students are using correctly theoretical knowledge. This could give to the teachers much more data to improve their scoring methods.

Keywords:

eAdventure

edX

xModule

Lon-Capa

LTI

xAPI

LRS

1. Introducción

En este proyecto, el objetivo es investigar y realizar la integración de juegos educativos en cursos on-line masivos en abierto (en inglés, Massive On-line Open Courses, MOOCs), y así poder conseguir una mayor interactividad con el usuario. Esto facilitaría, por ejemplo, la evaluación de los conocimientos de los estudiantes, ya que a veces este proceso es bastante complicado debido al elevado número de alumnos que se apuntan a estos cursos, de modo que en general no es posible evaluar a los mismos con ejercicios prácticos complejos y se suele hacer bien mediante preguntas (e.g. respuesta múltiple) o utilizando otros mecanismos como es la revisión por pares (peer review).

En la actualidad, hay varias iniciativas y empresas que proporcionan plataforma para la oferta de cursos como, por ejemplo, edX, Coursera o Udacity. Normalmente el contenido de dichos cursos está generado por profesores de prestigiosas universidades (e.g. Harvard, Standfor, Berkeley). Estas iniciativas cuentan con miles de usuarios y con un gran número de cursos de muy diversos temas (e.g. programación, química, medicina, derecho). El objetivo concreto de este proyecto es investigar la integración de juegos educativos en la plataforma edX. La plataforma edX parte de un sistema inicial desarrollado por el MIT al que posteriormente se unió la Universidad de Harvard y en el que actualmente colaboran algunas de las universidades más relevantes del mundo. Es una plataforma de código abierto y se ha considerado que es la más prometedora y que puede tener un mayor uso.

En los cursos actuales que se ofrecen en los MOOCs se pueden diferenciar principalmente dos bloques: el constituido por videos explicativos o teoría con los conceptos del curso, y otro bloque con preguntas y ejercicios más prácticos para afianzar mejor dichos conceptos. El objetivo de la investigación de este proyecto es ver si los juegos educativos realizados en JavaScript se pueden incluir como otra forma de ejercicios en estos cursos.

En concreto, la idea es que juegos educativos (o serious games) realizados con la plataforma eAdventure desarrollada en la Universidad Complutense de Madrid puedan formar parte del bloque de ejercicios prácticos de estos cursos. De esta forma los juegos permitirían, por ejemplo, evaluar el conocimiento que han adquirido los estudiantes, o comprobar si en una situación concreta se aplican bien esos conocimientos teóricos. Así se conseguiría una interactividad mucho mayor y se podrían tener muchos más datos para mejorar la evaluación o el rendimiento de los alumnos.

Como el campo de los juegos educativos es muy amplio y diverso en un principio se planteó solamente utilizar juegos creados con la plataforma eAdventure. Con eAdventure se facilita la creación de videojuegos altamente mantenibles y con un coste moderado ya que incluye una interfaz gráfica de modo que no es necesario programar y permite crear escenarios fotorrealistas basados en fotos y en los que se pueden incluir videos para reducir el coste de los recursos gráficos. eAdventure permite crear juegos de aventura gráfica y simulaciones con estrategia de juego con escenarios que representen alguna situación del contenido de estos cursos, simulando situaciones de casos prácticos que tengan relación con los mismos, en las que los alumnos tengan que aplicar los conocimientos adquiridos. Por ejemplo, en un curso de primeros auxilios se podría crear un juego en el que el estudiante, a través de una aventura interactiva, deba aplicar los conocimientos adquiridos para atender a una persona (o se puede reutilizar y adaptar el juego ya existente creado por el grupo e-UCM). Al final del juego, dependiendo de los errores cometidos por el estudiante y del tiempo utilizado, se podría calcular una nota que serviría como evaluación.

Actualmente se está desarrollando una nueva versión de eAdventure (versión 2.0) que proporcionará la opción de exportar juegos en JavaScript y HTML5, lo que facilitará su integración como ejercicio en edX. Debido a esta novedad que tendrá la nueva versión de eAdventure 2.0, se ha decidido poder integrar cualquier juego educativo realizado en JavaScript sin necesidad de ser un juego de eAdventure. Se facilitará así la integración de futuros juegos exportados por la nueva

versión de eAdventure 2.0.

En cuanto al aspecto técnico, para poder llevar a cabo este proyecto, hay que analizar cuál es la infraestructura técnica y la arquitectura software de edX. En concreto hay que identificar cuáles son los componentes software que permitirían la encapsulación o la inclusión de los juegos como contenido interactivo integrado en edX. Además hay que investigar cuales son los mecanismos de comunicación y los servicios de agregación de datos sobre interacción y usuarios que se proporcionan en edX (e.g. Learning Analytics) o de forma externa (e.g. xAPI), para permitir que la información generada durante el juego se pueda comunicar a la plataforma. De este análisis se obtendrá los requisitos que debe cumplir un juego para que se pueda desplegar en edX como un elemento activo.

De esta forma, una vez resuelto el despliegue y la comunicación, un juego JavaScript podría ser un elemento interactivo más en los cursos edX sin que haya conflicto con otros componentes del curso ni se tenga que hacer ningún otro desarrollo ad-hoc.

2. Descripción de los juegos eAdventure

La plataforma eAdventure es un proyecto de investigación, que aspira a facilitar la integración de juegos educativos y simulaciones basadas en juegos en procesos educativos en general, y entornos virtuales de aprendizaje (VLE) en particular ("e-Adventure ucm," 2009). Surge con la aspiración de introducir videojuegos con características específicamente educativas en el flujo de clase. Por ello, el objetivo principal de eAdventure es reducir los costes de producción de videojuegos y permitir su desarrollo sin necesidad de conocimientos de programación.

La plataforma consta de dos herramientas principales, ambas de código abierto, escritas en Java, y que se distribuyen de forma totalmente integrada. En primer lugar, el editor de aventuras permite crear los juegos de una manera sencilla. Crear un juego con el editor de eAdventure se asemeja a la creación de una obra de teatro. Primero se definen las escenas donde ocurrirán las acciones, los personajes que aparecerán, los objetos con los que se podrá interactuar, las conversaciones, etc. Una vez creados los elementos de juego se define cómo se unirán esas escenas y cómo transcurrirán las acciones, es decir, qué interacciones deberá desempeñar el jugador para avanzar en la historia, pudiendo definir multitud de caminos alternativos y bifurcaciones ("e-Adventure - Wikipedia," 2010).

Hay que destacar la posibilidad de extraer una gran cantidad de datos de la interacción del jugador de forma automática, sin que el examinador tenga que estar pendiente de las acciones del jugador. Estos datos son guardados, y podrán ser enviados por mail para su posterior evaluación. También existe la posibilidad de modificar el flujo de juego, para amoldarlo a las necesidades de cada alumno sin tener que crear un juego específico para cada uno de ellos. Otro punto a destacar, es la integración con Entornos Virtuales de Enseñanza como moodle, sakai, webCT, Blackboard o cualquier entorno virtual de enseñanza que se elija.

En cuanto a las características de eAdventure según ("e-Adventure ucm," 2009) se pueden dividir en tres grupos:

- Características de autoría:

Cuenta con soporte para todos los rasgos comunes de aventuras gráficas point & click y de ficción interactiva, además de la posibilidad de edición gráfica para autoría. También es posible la elección del estilo de interacción (menús de aventura tradicionales o menús contextuales). Por último cabe destacar la realización de interfaces personalizables y la notación XML para la descripción de las aventuras.

- Características educativas:

Cuenta con soporte para escenarios de aprendizaje adaptativos en tiempo real. Los juegos son empaquetados con metadatos estándares (IEEE Learning Object Metadata, LOM-ES). Es importante destacar la Integración con LMS a través de la implementación de diversos estándares educativos (SCORM 1.2 y 2004 e IMS Content Packaging), exportación especial para la integración con WebCT 4.0 y soporte para IMS Learning Design, además de la Integración en el repositorio de contenidos educativos digitales Agrega.

- Características del motor:

El lenguaje utilizado es Java y utiliza OpenSource (Código abierto). Además puede ser desplegado como una aplicación independiente, o como un applet para la educación online.

Las últimas líneas de investigación de la plataforma eAdventure están relacionadas con la mejora de la accesibilidad de los juegos y simulaciones, con la integración en nuevos Entornos Virtuales de Enseñanza, o con la portabilidad del motor a plataformas móviles como Android ("e-Adventure - Wikipedia," 2010).

Para una mejor comprensión, se ha realizado un juego con la herramienta eAdventure. El juego consiste en aprender a alimentar a los animales de un zoo. El jugador debe explorar el mapa y consultar los libros de los animales para aprender qué alimentos come cada uno y así alimentarlos correctamente después.

Imagen 2-1: Ejemplo de juego eAdventure.

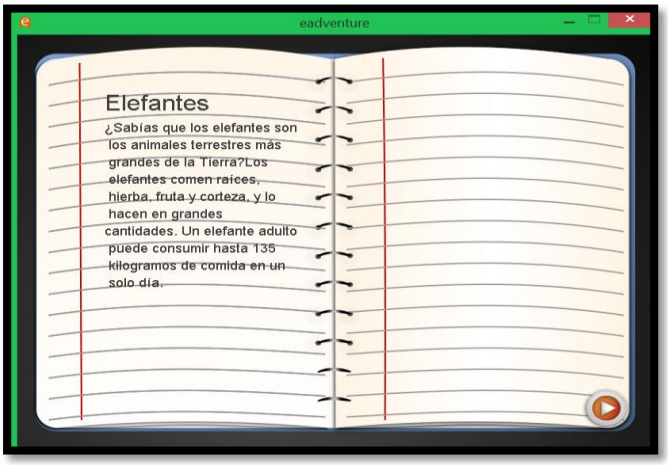


Imagen 2-2: Ejemplo de libro en eAdventure.

Una nueva versión de eAdventure (versión 2.0) está en fase de desarrollo. Esta nueva versión proporcionará la opción de exportar juegos en JavaScript y HTML5, lo que facilitará su integración como ejercicio en edX. El código de esta versión se encuentra disponible en ("Github eAdventure," 2014) y también se puede consultar información adicional en el blog de desarrollo ("eUCM | eAdventure 2.0," 2014).

3. edX

En este capítulo se hará una descripción de edX y se explicará cómo instalar una instancia del mismo. Una vez instalada la instancia, se mostrará cómo crear cursos dentro de ella.

También se describirá la arquitectura de edX y posteriormente la estructura de la base de datos que utiliza. Además se explicará cómo realizar cambios en ella.

3.1. Descripción de edx

En esta sección se explicará las características principales de la plataforma elegida así como los recursos ofrecidos por la misma para añadir nuevos ejercicios y analizar los resultados de éstos.

3.1.1. edX

edX es una plataforma de cursos abiertos, masivos, y en línea (MOOCs) fundada por el Instituto de Tecnología de Massachusetts (MIT) y la Universidad de Harvard en Mayo de 2012. Alberga cursos en línea de nivel universitario con más de 2,1 millones de usuarios en una amplia gama de disciplinas para todo el mundo, de forma gratuita, para llevar a cabo la investigación sobre aprendizaje. Las dos instituciones han contribuido con una cantidad de 30 millones de dólares cada una en los recursos para la organización no lucrativa del proyecto. El curso prototipo "Circuitos y Electrónica" se inició en diciembre de 2011 a través del programa online del MIT (MITx). Hay 47 escuelas que ofrecen, o tienen previsto ofrecer, cursos en la página web edX ("edx- Wikipedia," 2014).

Además de la oferta educativa, el proyecto también se utiliza para la investigación en la educación y el aprendizaje a distancia mediante la recopilación de las respuestas de los alumnos y el análisis de los datos, así como la obtención de datos demográficos de cada participante. Un equipo de investigadores de Harvard y del MIT, dirigido por David Pritchard y Lori Breslow, han publicado recientemente sus conclusiones iniciales en el siguiente documento (Pritchard, 2013). Las escuelas que forman parte del consorcio edX también están llevando a cabo su propia investigación a partir de datos recogidos de sus cursos. La investigación se centrará en los resultados del aprendizaje de los cursos universitarios tradicionales y en línea una vez finalizado el curso.

3.1.2. xBlock y xModule

xBlock es el SDK para la plataforma MOOC edX. Fue anunciado y lanzado públicamente el 14 de marzo de 2013 y está escrito en Python2. Su objetivo es permitir a la comunidad mundial de desarrollo de software participar en la construcción de la plataforma educativa edX y en la próxima generación de cursos combinados en línea ("xBlock Wikipedia," 2014).

xBlock es una arquitectura de componentes que permite a los desarrolladores crear componentes independientes del curso o xBlocks. Los xBlocks son capaces de trabajar de forma integrada con otros componentes en la construcción y presentación de un curso en línea.

Se construyen de manera similar a las aplicaciones web. Mantienen el estado en una capa de almacenamiento, es decir, procesan acciones de los usuarios a través de los controladores de procesos("Xblock Documentation," 2014).

xModule fue el primer componente diseñado para esta plataforma. Sus funcionalidades son las mismas que las de xBlock pero con ciertas limitaciones ya que no es tan independiente. xBlock está mejor diseñado para que los desarrolladores puedan extender edX, pero ya que el cambio de xModule a xBlock se encuentra en fase de transición, se ha tomado la decisión de trabajar con xModule.

3.1.3. Insight

Insight es una versión de desarrollo de un marco de análisis para la infraestructura edX. El objetivo de este marco es definir una arquitectura para módulos simples que nos permitan el análisis ("Github insights," 2014). La arquitectura debe tener las siguientes propiedades:

- Fácil de usar: Profesores, estudiantes de postgrado, etc. deben ser capaces de escribir plug-ins de forma rápida y sencilla. Además de trabajar en el sistema sin afectar a la estabilidad general. Los resultados deben ser presentados de forma automática a los clientes.

- La API debe ser compatible con las implementaciones robustas y escalables.

- Reutilizable: Los módulos de análisis individuales deben ser capaces de utilizar los resultados de otros módulos, y la gente debe ser capaz de aprovechar el trabajo de los demás.

- Interoperable: El marco debería ser lo suficientemente genérico para ser utilizable fuera de edX.

Cross-scope: Debe haber una conexión fluida entre el análisis offline y el análisis online para que sea posible el análisis en tiempo real.

Una vez que Insight esté terminado será la forma más completa y adecuada de realizar análisis en edX ya que éste se encuentra integrado dentro de la plataforma. Como método alternativo se ha escogido xAPI, que como se verá más adelante (4.3. El estándar xAPI) es un buen método para suplir las funcionalidades de Insight hasta que este esté completamente desarrollado.

3.2. Cómo instalar una instancia de prueba de la plataforma de edX

Existe un tutorial para instalar la instancia de edX en Windows("edx installation tutorial Windows," 2013), pero debido a los numerosos problemas existentes en este sistema operativo, se utilizará Linux, concretamente la distribución de Ubuntu 12.04 para instalarla.

El proyecto de edX-platform permite descargar e instalar una instancia de prueba en una máquina virtual local. Este es el procedimiento recomendado para familiarizarse con la plataforma y sus requisitos de hardware y software antes de intentar una instalación real en un servidor de producción.

3.2.1. Manual de instalación

Los pasos de instalación que se van a seguir según ("edX installation Manual," 2014), serán los siguientes:

1. Instalar la última versión de VirtualBox (≥ 4.3.12).
2. Instalar la última versión de Vagrant (≥1.5.3).Para las versiones Vagrant (≥ 1.6), pueden dar algunos problemas, a los que se dan soluciones en el siguiente enlace ("Vagrant issues," 2014)
3. Asegurar que nfsd se esté ejecutando.

4. Descargar el archivo Vagrant y crear la instancia de Vagrant:

- mkdir devstack
- cd devstack
- curl -L https://raw.github.com/edx/configuration/master/vagrant/release/devstack/Vagrantfile > Vagrantfile
- vagrant plugin install vagrant-vbguest
- vagrant up
    - Nota: La primera vez que se crea la instancia de la máquina virtual (VM), Vagrant descargará el paquete base, que ocupa unos 4 GB. Si se destruye y se vuelve a crear la máquina virtual, Vagrant volverá a utilizar el paquete que descargó.
    - El paquete base también se puede descargar mediante Torrent . Una vez descargado, se puede añadir el paquete con 'vagrant box add (box-name) (path-to-box-file)'.
    - Después de que la VM haya arrancado, hay que montar las carpetas compartidas para que puedan ser accesibles tanto por su sistema host como por el sistema virtual. Cuando se llegue a este punto, será necesario introducir la contraseña en el sistema host.
- [default] Exporting NFS shared folders...
- Preparing to edit /etc/exports. Administrator privileges will be required...
- Password:

3.2.2. Solución de posibles problemas

La firma de un nuevo usuario (problemas de mensajes de activación):

Al registrarse como nuevo usuario, se muestra este mensaje en el navegador:

- We need to verify your email address
- Almost there! In order to complete your sign up we need you to verify your email address.
- An activation message and next steps should be waiting for you there.

Cuando se ejecuta el stack a nivel local, no se envía ningún mensaje de correo electrónico. Sin embargo, el mensaje sale por consola, dando la clave de activación que hay que usar:

```
Thank you for signing up for edX Studio! To activate your account,
please copy and paste this address into your web browser's address
bar:

        http://localhost/activate/0699e17df6024fb382fadfabed100691

If you didn't request this, you don't need to do anything; you won't
receive any more email from us. Please do not reply to this e-mail; if
you require assistance, check the help section of the edX web site.
```

Problemas con la instalación en Ubuntu

- Si aparece este mensaje:

```
It appears your machine doesn't support NFS, or there is not an
adapter to enable NFS on this machine for Vagrant. Please verify that
`nfsd` is installed on your machine, and try again. If you'reon
Windows, NFS isn't supported. If the problem persists, please contact
Vagrant support.
```

Entonces, se necesita instalar nfs usando este comando:

```
sudo apt-get install nfs-common nfs-kernel-server
```

- Si se obtiene un error como NS_ERROR_FAILURE:

Se debe intentar actualizar VirtualBox.

3.2.3. Usar edx-platform

Para iniciar la máquina virtual (una vez instalada) se necesitará seguir unos determinados pasos ("edx installation tutorial Windows," 2013). Los pasos son los siguientes:

Introducir en consola el siguiente comando:

$ vagrant up –no-provision

Para acceder a la máquina virtual se utilizará:

$ vagrant ssh

Una vez dentro de la máquina virtual:

- Para arrancar el LMS (plataforma de cursos como tal):

  $ rake lms[cms.dev,0.0.0.0:8000]

Desde la máquina física local se puede usar con un navegador en: http://localhost:9000/

- Para arrancar Studio (herramienta de creación de cursos):

  $ rake cms[dev,0.0.0.0:8001]

Desde la máquina física local se puede usar con un navegador en: http://localhost:9001/

Esto es posible gracias a la redirección de puertos que fue configurada en la máquina virtual, la cual convierte las peticiones que se hagan a la máquina física por los puertos 9000/9001 en peticiones a la máquina virtual por los puertos 8000/8001.

3.3. Cómo crear un curso en edX

Para la creación de un curso en edX se utiliza la herramienta 'studio' proporcionada por la propia plataforma. El primer paso para poder crear cursos en esta plataforma es estar registrado. Para registrarse hay que aceptar un link de confirmación, pero el servidor por defecto no está configurado para enviar e-mails, por lo que el e-mail sale por consola. Una vez hecho el registro, será posible crear el nuevo curso.

La estructura de un curso edX es la siguiente:

Imagen 3-1: Estructura de una lección en edX.

Como se observa en la imagen hay tres tipos de bloques en un curso edX: las secciones, las subsecciones y las unidades.

Las secciones se corresponden a las lecciones que tiene el curso, o a las semanas que tiene el mismo. Para crear una sección sólo es necesario pulsar el botón "new section" que aparece arriba a la derecha.

Una vez creada la sección, se podrá indicar la fecha a partir de la cual estará disponible para los alumnos y se podrá añadir los diferentes componentes que la forman.

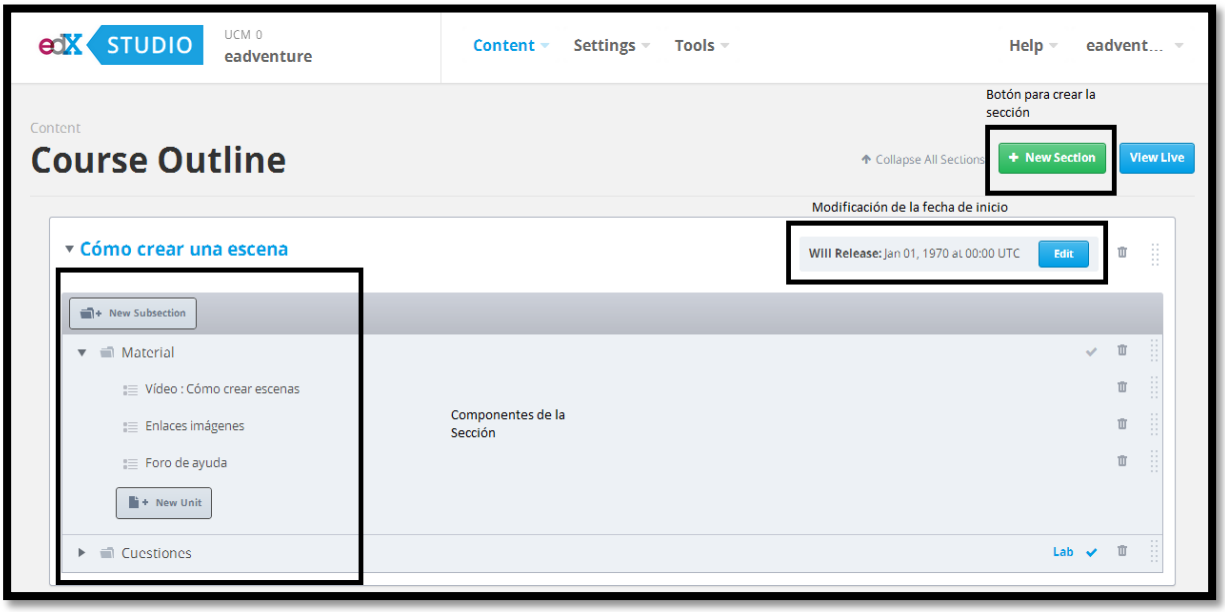


Imagen 3-2: Creación de una nueva sección.

Las subsecciones son el componente más alto de una sección. Se pueden crear a través del botón "New subsection" y su creación es importante debido a que las secciones se pueden asignar a diferentes etapas del curso (Laboratorio, ejercicios, examen parcial, examen final, etc.), haciendo

posible que exista una evaluación más completa del alumno.

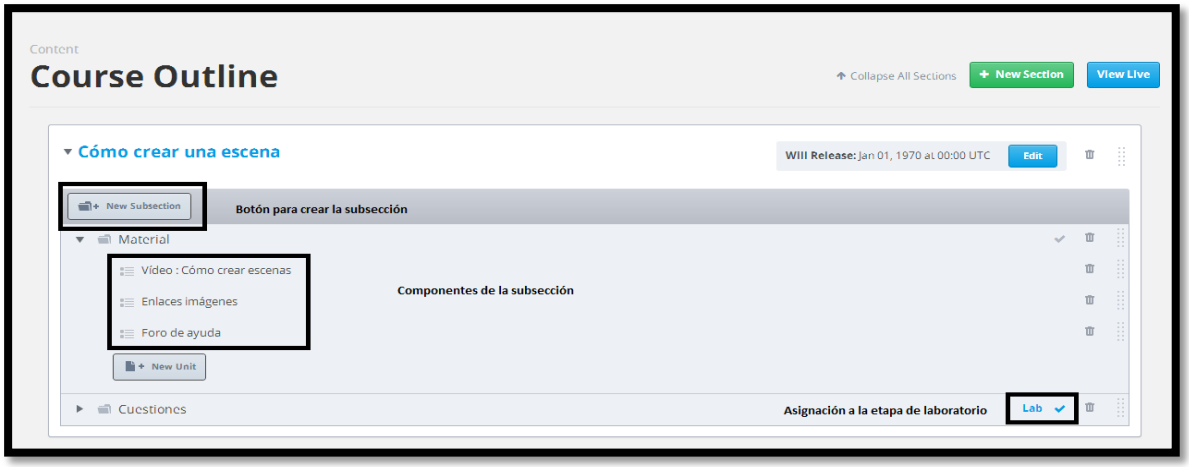


Imagen 3-3: Creación de una subsección.

Dentro de las subsecciones se encuentran las unidades, que son las categorías que contienen los componentes. Estas categorías son las más pequeñas que contiene la arquitectura de un curso edX. Se crean a través del botón "New unit" y pueden contener foros, textos, problemas o vídeos.

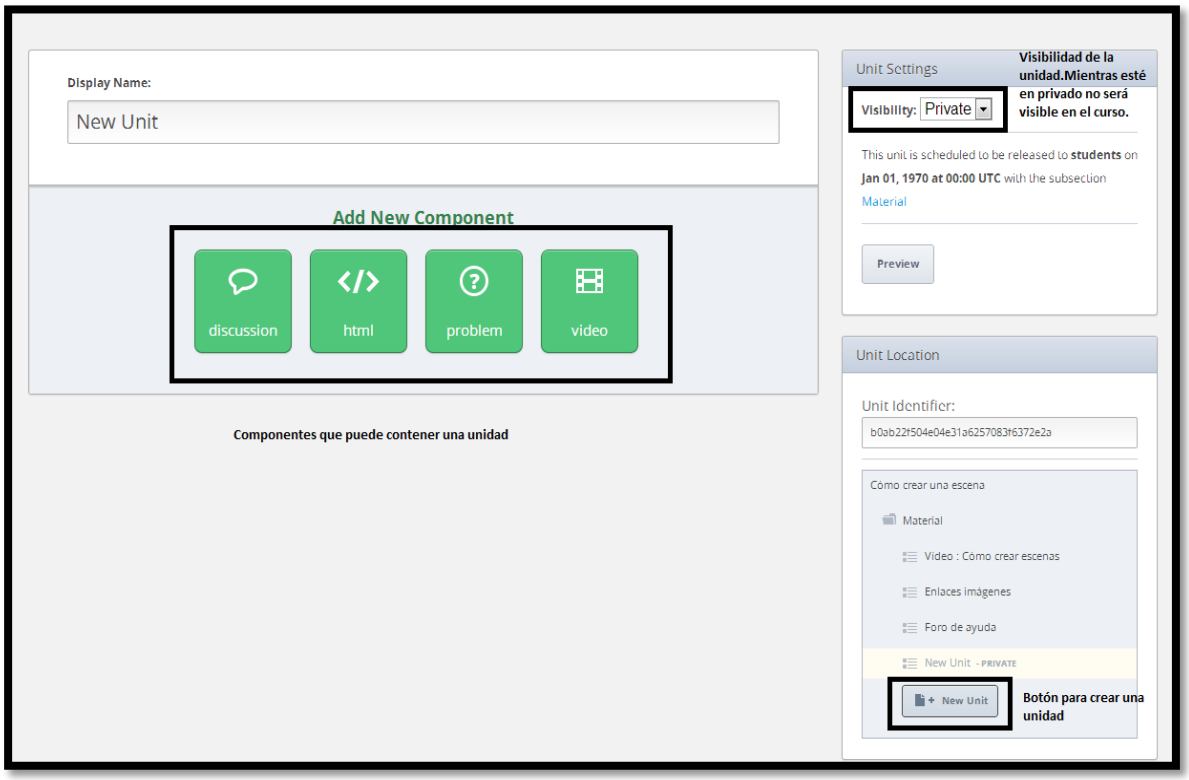


Imagen 3-4: Creación de una unidad.

Para crear un foro de discusión se utiliza el botón "discussion" y se añadirá un foro como el que aparece en la imagen.

Imagen 3-5: Creación de un foro.

Para crear un texto se utiliza el botón "HTML", una vez elegido el tipo de texto (puede ser un texto normal, una noticia o un texto escrito en LaTeX), se escribe el texto en el editor y se guarda el componente.

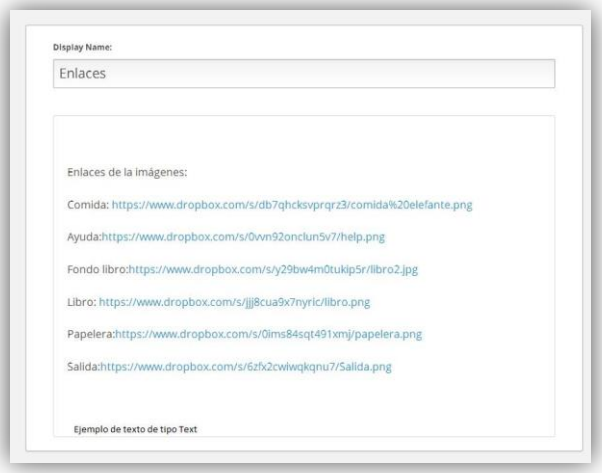


Imagen 3-6: Visualización de un HTML.

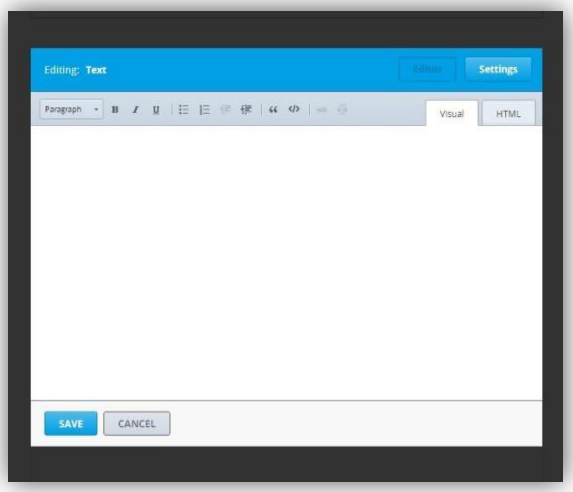


Imagen 3-7: Editor de texto.

Para crear problemas se utiliza el botón "Problem". Una vez elegido el tipo de problema (pregunta con respuesta simple, pregunta con respuesta múltiple, drag and drop, imágenes mapeadas, etc...), a través del botón "edit" se abrirá el editor.

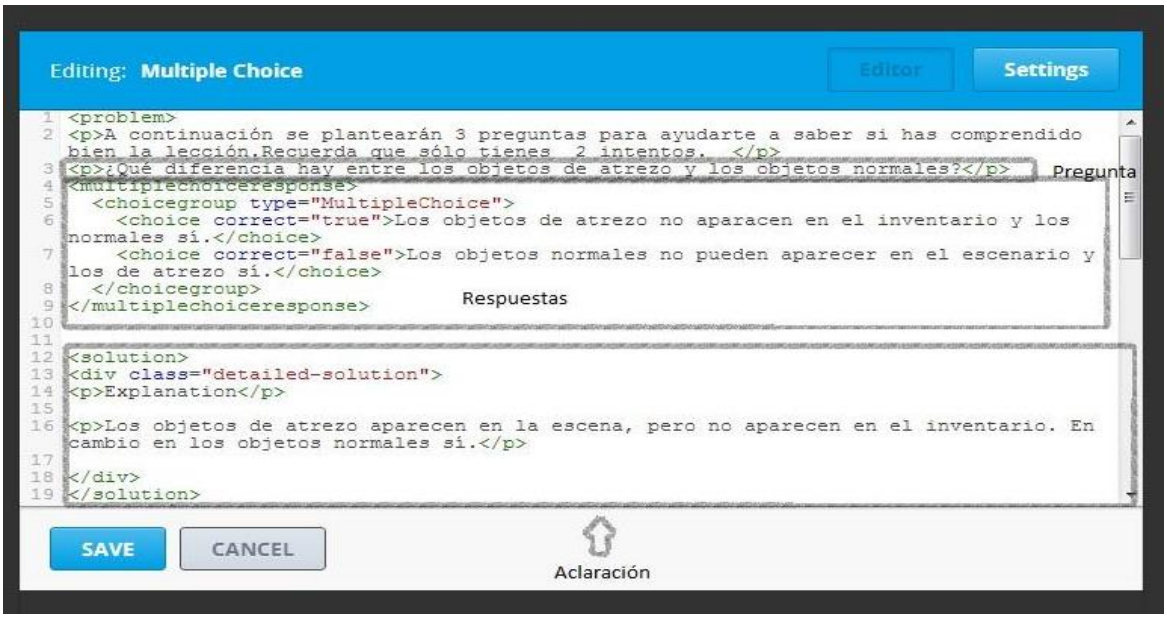


Imagen 3-8: Editor de problemas.

Imagen 3-9: Visualización de un problema.

El vídeo se crea a través del botón "video". Para añadirlo debe estar previamente subido en "Youtube" y, pulsando en el botón "edit" se abre el editor donde se podrá añadir en el campo "Youtube Id" el enlace del video creado.

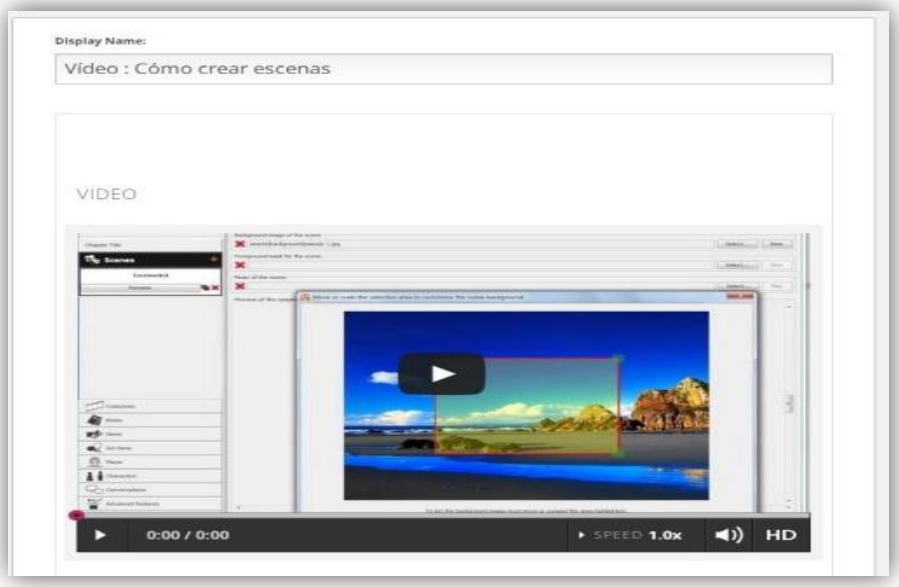


Imagen 3-10: Visualización de un vídeo.

Por último, podemos agregar evaluación al curso, en la pestaña "Settings→Grading", donde se seleccionarán las posibles notas del curso, el plazo para realizar el mismo, y el porcentaje de la nota final que corresponde a cada etapa.

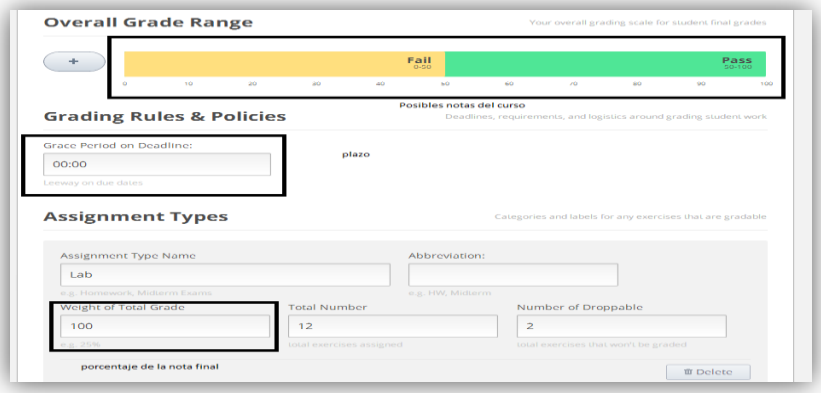


Imagen 3-11: Pantalla de elección de la evaluación.

Si es necesaria una información más detallada de cómo crear un curso en edX, existe un tutorial más completo ("Crear un curso en edx," 2014).

3.4. Arquitectura y base de datos de edX

En esta sección, se explicará con más detalle la arquitectura de la plataforma edX. También se dará a conocer la estructura de la base de datos explicando la funcionalidad de cada una de las tablas que la forman. Parte del contenido de esta sección pertenecen al código de la plataforma de edX.

3.4.1. Arquitectura

A alto nivel, y como una visión más global de la arquitectura de la plataforma, se puede considerar que edX tiene la siguiente estructura de módulos:

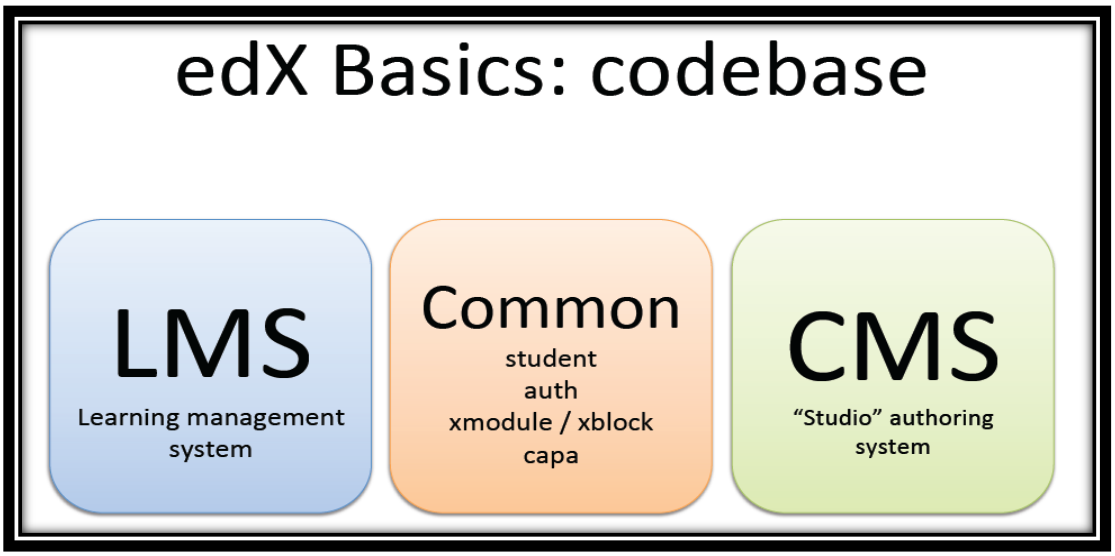


Imagen 3-12: Arquitectura de módulos de edX ("Arquitectura edX," 2013).

LMS (Learning management system)

Es la plataforma donde están todos los cursos y proporciona la interfaz con la que interactúan tanto los estudiantes como los profesores ("Arquitectura edX," 2013). Está formado por:

- **courseware:** Que contiene las vistas principales del curso, las llamadas ajax y la interacción con Xblocks.
- **instructor:** Contiene la vista del instructor.
- **django_comment_client:** Contiene la interfaz del foro.
- **django-wiki:** Una wiki es un sitio web cuyas páginas pueden ser editadas por múltiples voluntarios a través del navegador web. Los usuarios pueden crear, modificar o borrar un mismo texto compartido.
- Además contiene también los HTML de los diferentes tipos de ejercicios.

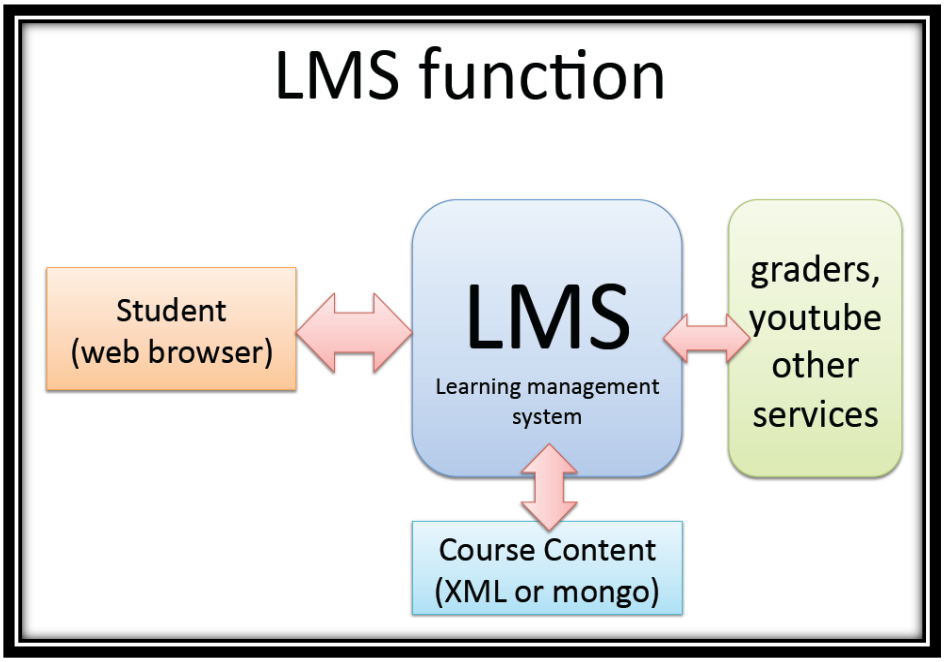


Imagen 3-13: Interacción con el LMS ("Arquitectura edX," 2013).

CMS ("Studio" authoring system)

Es la plataforma que se usa para crear los cursos, por lo que sólo tienen acceso los profesores. En esta plataforma se elige el número de lecciones, las diferentes partes de esas lecciones, los ejercicios que las forman y la calificación de los mismos ("Arquitectura edX," 2013). Se puede ver más detalladamente en el documento (3.3. Cómo crear un curso en edX).

Esta plataforma está formada por:

- **contentstore:** contiene las principales vistas del "studio".
- **auth:** grupos de usuarios del cms.

Common

Es la parte donde se guardan los datos de los cursos y los tipos de ejercicios, por lo tanto common es la conexión entre el LMS y el CMS.

Esta parte contiene los Xmodule de los distintos ejercicios que hay disponibles y los módulos JavaScript que se ejecutan.

3.4.2. Estructura de las tablas de la base de datos

Todos los diferentes tipos de tablas están en la ruta edX-platform/lms/djangoapp (tablas del LMS), o en la ruta edX-platform/common/djangoapp (tablas de COMMON). De estas carpetas se puede elegir el tipo de tabla que va a ser utilizado según las necesidades del problema.

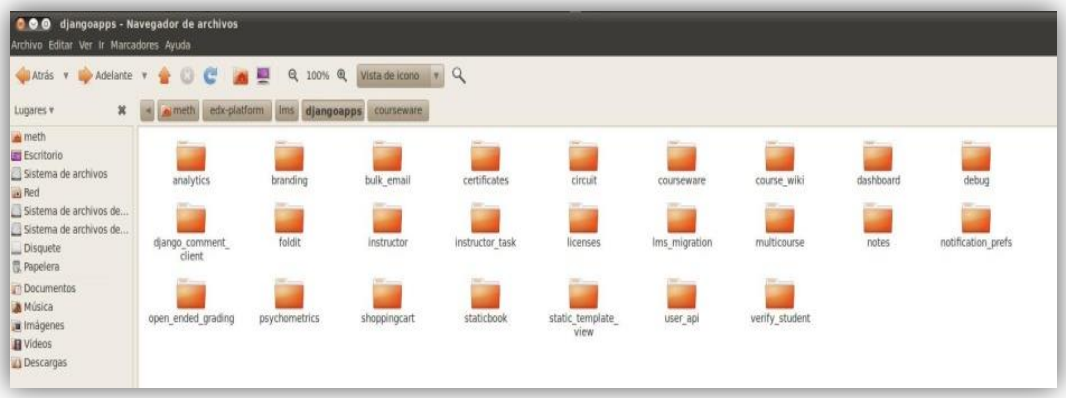


Imagen 3-14: Los diferentes tipos de tablas del LMS en edX.

Tablas del LMS:

- **Carpeta bulk_email** (guarda toda la información referente a los emails).
  - CourseEmail [bulk_email.courseemail] → Almacena información de un correo electrónico que ha sido enviado a un curso. Esta información se puede enviar de tres formas diferentes, haciéndolo siempre desde la vista del instructor:
    a. Únicamente mandar el email a sí mismo.
    b. Mandarlos al personal y a los instructores.
    c. Mandarlo a todas las personas que están inscritas en el curso.
  - Optout [bulk_email.optout] → Guarda los usuarios que no quieren recibir los correos.
  - CourseEmailTemplate [bulk_email.courseemailtemplate] → Guarda las plantillas de los emails usados en el curso.
- **Carpeta certificates** (guarda los certificados de los estudiantes que se generan en los diferentes cursos).
  - CertificateWhitelist [certificates.certificatewhitelist] → Guarda los estudiantes que están en la lista blanca. Todos los usuarios de esta tabla siempre tendrán derecho a un certificado independientemente de su grado a menos que se encuentran en la lista de países prohibidos.
  - GeneratedCertificate [certificates.generatedcertificate] → Guarda toda la información de los certificados.
- **Carpeta circuit** (guarda la información de los circuitos).
  - Servercircuit [circuit.servercircuit] → Guarda la información de los circuitos.

- **Carpeta courseware** (guarda la información relacionada con los cursos).
    - StudentModule [courseware.studentmodule] → Mantiene el estado del estudiante para un módulo en particular, en un curso particular.
    - StudentModuleHistory [courseware.studentmodulehistory] → Mantiene una historia completa de todos los cambios hechos en un xModule por un estudiante.
    - XModuleUserStateSummaryField [courseware.xmoduleuserstatesummary] → Guarda los datos asignados en "Scope.user_state_summary" en un campo de un xModule.
    - XModuleStudentPrefsField [courseware.xmodulestudentprefsfield]→ Guarda los datos asignados en "Scope.preferences" en un campo de un xModule.
    - XModuleStudentInfoField [courseware.xmodulestudentinfofield] → Guarda los datos asignados en "Scope.preferences" en un campo de un xModule.
    - OfflineComputedGrade [courseware.offlinecomputedgrade] → Tabla de calificaciones para un usuario y el curso dado.
    - OfflineComputedGradeLog [courseware.offlinecomputedgradelog] → Guarda los logs cuando se realizan nuevas calificaciones. Se usa para que el instructor sepa cuándo ha sido el último cambio en las calificaciones.
- **Carpeta Foldit** (guarda información del módulo Foldit[1]).
    - Score [foldit.score] → Almacena las puntuaciones de los diferentes usuarios en los problemas Foldit.
    - PuzzleComplete [foldit.puzzlecomplete] → Hace un seguimiento de los puzzles completados por cada usuario.
- **Carpeta instructor_task** (guarda los comentarios que han puesto los instructores en los trabajos subidos por los estudiantes).
    - InstructorTask [instructor_task.instructortask] → Almacena los comentarios que han puesto los instructores en los trabajos subidos por los estudiantes.
- **Carpeta psychometrics** (guarda los datos que vinculan al estudiante, módulo, y el rendimiento del módulo, incluyendo el número de intentos, la calificación, la calificación máxima, y el tiempo utilizado).
    - PsychometricData [psychometrics_psychometricdata] → Guarda los datos que vinculan al estudiante, módulo, y el rendimiento del módulo, incluyendo el número de intentos, la calificación, la calificación máxima, y el tiempo usado.

[1] Foldit: Es un ejercicio integrado en edX con LTI que consiste en una simulación en tres dimensiones donde los estudiantes pliegan proteínas.

- **Carpeta shoppingcart** (guarda información de las compras).
  - Order [shoppingcart.order] → Guarda la información de las compras.
  - OrderItem [shoppingcart.orderitem] → Guarda la información de los objetos comprados.
  - PaidCourseRegistration [shoppingcart.paidcourseregistration] → Guarda la información del pago de los cursos.
- **Carpeta user_api** (guarda las preferencias de los usuarios).
  - UserPreference [user_api.userpreference] → Guarda las preferencias de los usuarios.

Imagen 3-15: Los diferentes tipos de tablas del Common en edX.

Tablas de Common:

- **Carpeta course_groups** (guarda los grupos de usuarios que están en un curso).
  - CourseUserGroup [coursegroup.courseusergroup] → Guarda los grupos de usuarios que están en un curso.

- **Carpeta course_modes** (guarda el modo en el que es ofrecido el curso).
  - CourseMode [course_modes.coursemode] → Guarda los grupos de usuarios que están en un curso.

- **Carpeta django_comment_common** (guarda los roles de los miembros del curso).
  - Role [django_comment_common.role] → Guarda los roles de los miembros del curso.
  - Permission [django_comment_common.permission] → Guarda los permisos de cada uno de los miembros del curso.

- **Carpeta student** (guarda todos los campos demográficos de los usuarios. Tienen una tabla separada para esto, en vez de ampliar la incorporada en Django auth_user).
    - AnonymousUserId [student.anonymoususerid] → El propósito de esta tabla es poder obtener el usuario a través del id del usuario anónimo.
    - UserStanding [student.userstanding] → Contiene el estado de la cuenta de un estudiante.
    - UserProfile [student.userprofile] → Guarda todos los campos demográficos de los usuarios.
    - Registration [student.registration] → Permite esperar al e-mail de confirmación después de que el usuario haya sido registrado.
    - LoginFailures [student.loginfailures] → Guarda los intentos fallidos al iniciar sesión.
    - CourseEnrollment [student.courseenrollment] → Representa el registro de inscripción de un estudiante para un solo curso.
    - CourseEnrollmentAllowed [student.courseenrollmentallowed] → Tabla de usuarios (especificados por la dirección de correo electrónico) que están autorizados a inscribirse en un curso determinado.
- **Carpeta track** (define los campos que se almacenan en la base de datos de registro de seguimiento).
    - TrackingLog [track.trackinglog] → Almacena los campos de registro de seguimiento.

3.5. Cómo modificar la base de datos en edX

En esta sección se explicará la forma correcta de crear tablas en la base datos. Además se enseñará los métodos utilizados para poder leer y escribir. Para una mejor comprensión se pondrá un ejemplo práctico con imágenes.

3.5.1. Creación de una tabla en edX

Dentro de edX hay varios tipos de tablas, así que lo primero que hay que hacer es elegir el tipo de tabla que se necesita. Se pueden ver los tipos de tablas en el archivo Arquitectura (3.4. Arquitectura y base datos de edX).

En este ejemplo ha sido elegido el tipo courseware perteneciente a las tablas de LMS, por lo que se abrirá el documento de la ruta edX-platform/lms/djangoapp/courseware/models.py

En este archivo se debe añadir una nueva clase para crear la nueva tabla y se agregará tanto los campos como las claves primarias.

```python
class minigameField(models.Model):



    class Meta:
        unique_together = (('usage_id', 'field_name'),)

    # The name of the field
    field_name = models.CharField(max_length=64, db_index=True)

    # The definition id for the module
    usage_id = models.CharField(max_length=255, db_index=True)

    # The value of the field. Defaults to None dumped as json
    value = models.TextField(default='null')

    created = models.DateTimeField(auto_now_add=True, db_index=True)
    modified = models.DateTimeField(auto_now=True, db_index=True)

    def __repr__(self):
        return 'minigameField<%r>' % ({
            'field_name': self.field_name,
            'usage_id': self.usage_id,
            'value': self.value,
        },)

    def __unicode__(self):
        return unicode(repr(self))
```

Imagen 3-16: Ejemplo de la clase minigameField que será la nueva tabla.

Una vez añadida toda la información se abrirá la consola donde se está ejecutando el vagrant ssh y se ejecutará el siguiente comando:

./manage.py lms schemamigration courseware --auto + descripción del archivo

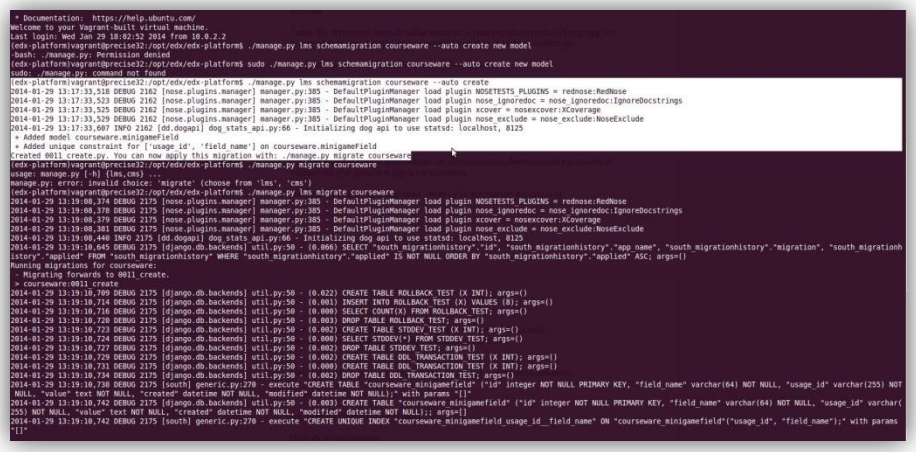


Imagen 3-17: Ejecución de los comandos en la consola.

Este comando creará un nuevo archivo que se encontrará en la ruta edX-platform/lms/djangoapp/[carpeta elegida]/migrations

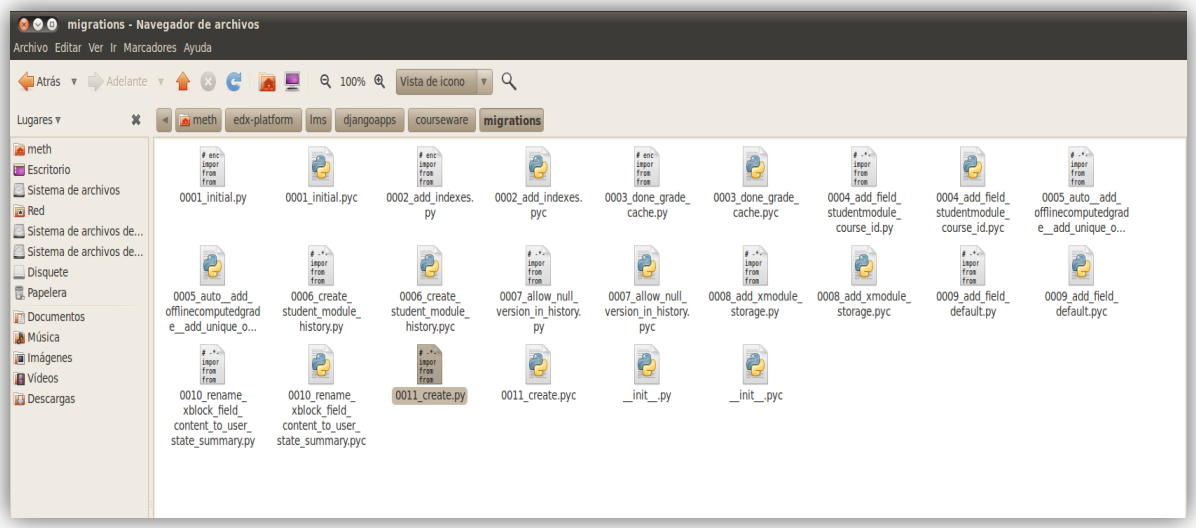


Imagen 3-18: Imagen del archivo creado en la carpeta migration.

Este archivo hay que copiarlo en otra ruta, así que hay que ejecutar el comando:

./manage.py lms migrate courseware

Así quedará creada la nueva tabla:

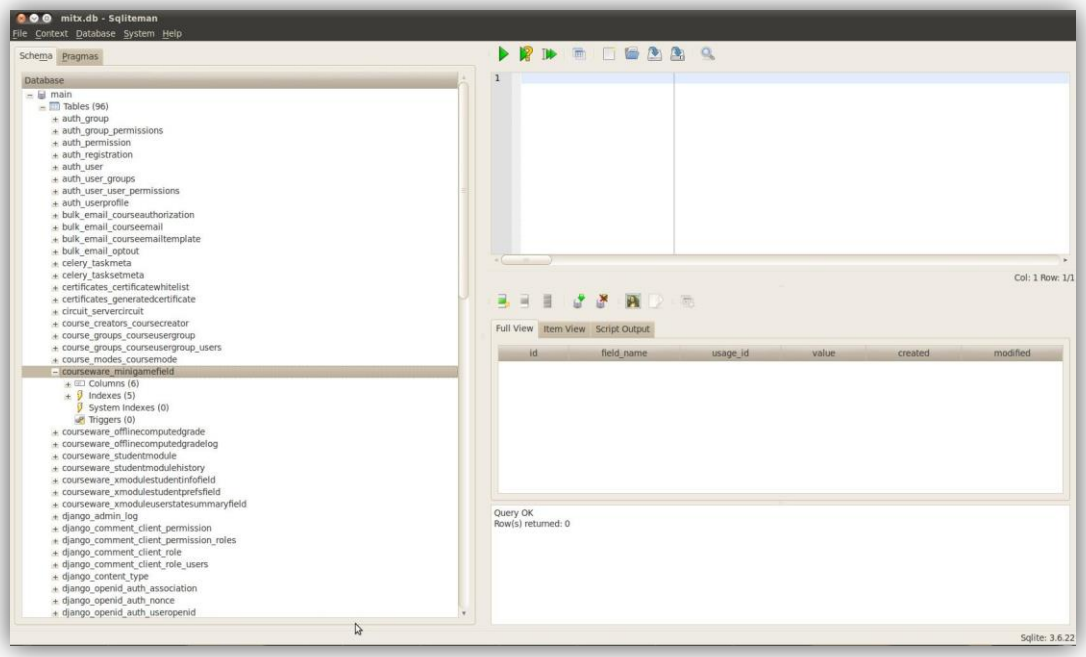


Imagen 3-19: Imagen de la nueva tabla en la base de datos.

3.5.2. Cómo escribir en la base de datos

Hay dos formas de escribir en la base de datos, dependiendo de la tabla en la que se va a escribir.

Si la tabla en la que se quiere escribir forma parte de las que se encuentran en "scope" (XModuleUserStateSummaryField,XModuleStudentPrefsFieldy XModuleStudentInfoField), primero hay que definir un atributo dentro del módulo e indicar en qué tabla y en qué campo se quiere guardar.

```
student_progress = List(
    help="Student progress.",
    scope=Scope.user_state,
    default=[]
)
```

Una vez obtenidos los datos que se quieren guardar, entonces se le asignan al atributo que se ha definido previamente, y éstos serán guardados en la base de datos.

```
student_progress = data.items()

self.student_progress = student_progress
```

En caso contrario, si se desea escribir en una tabla que no pertenece al "scope", hay que seguir el procedimiento que aparece en el siguiente código:

```
# FIXME: we must use raw JSON, not a post data (multipart/form-data)
from courseware.models import XModuleMinigame
m= XModuleMinigame(tagline= '2')
m.save()
```

Primero hay que importar la tabla donde se quiere insertar y después se creará una nueva fila con los datos que se quieren guardar. Por último, se guardará en la base de datos.

3.5.3. Cómo leer datos de la base de datos

Para leer de la base de datos hay que seguir los pasos que aparecen en el siguiente ejemplo:

```
from courseware.models import XModuleMinigame
pro=XModuleMinigame.objects.filter(id=19).values()
num=pro[0]
num2=num.get('tagline')
```

Primero se importa la tabla de la que se quiere leer, y después se obtiene la lista de diccionarios que cumplen la condición dada. En este caso la línea "XModuleMinigame.objects.filter(id=19).values()" devuelve la lista de diccionarios que contienen el id=19.

Una vez que se obtiene esa lista, ya se dispone de los datos deseados.

4. Estándares

En este capítulo se explicarán los diferentes estándares que se han investigado para hacer posible la integración de un juego en edX.

4.1. Lon-Capa

En esta sección se explicará cómo es la estructura de un ejercicio Lon-Capa de manera generalizada y cómo está definido este tipo de ejercicios en la plataforma edX.

4.1.1. Investigación de Lon-Capa

Características de Lon-Capa

Lon-Capa es un Ambiente Educativo Virtual, además de una plataforma de e-learning (Leyva García & Díaz García, 2013) y (Herrera Bautista, 2013), Learning Management System (LMS), Course Management System (CMS) o Virtual Learning Environment (VLE). Además Lon-Capa define un formato de ejercicios que es utilizado en algunos ejercicios de la plataforma edX ("LON-CAPA - Wikipedia," 2013).

El término Lon-Capa puede referirse también a la red de Lon-Capa, el sistema completo de servidores web de Lon-Capa y a la realización específica de un Protocolo de Internet (IP) que conecta estos servidores web.

Lon-Capa posee las características que existen en muchos otros sistemas similares como son las funciones de usuario, calendario, correo electrónico, chat, blogs, elaboración de recursos y corrección de pruebas.

Además Lon-Capa se distingue porque los servidores web (que se encuentran en varias partes del mundo) pueden comunicarse unos con otros. Otra de sus características importantes es la capacidad de diseñar recursos como tareas de exámenes, que usan números aleatorios (almacenados en variables de Perl). Esto permite que un solo recurso genere una variedad de ejercicios de prueba similares (pero diferentes), reduciendo así el riesgo de engaño por parte de los estudiantes.

En cuanto a las características propias de un sistema Lon-Capa para crear cursos online según ("Lon Capa. Elearning.," 2014), son las siguientes:

- Para permitir accesibilidad el software implementa las siguientes características: contenido disponible sin color, versiones con contenido de sólo texto, posibilidad de suprimir los applets de Java, posibilidad de incrementar el tamaño de la fuente.
- Posee capacidad para introducir recursos multimedia integrados en las unidades de aprendizaje.
- En cuanto a la apariencia se pueden modificar los colores de la fuente, de los enlaces visitados, de los no visitados, del fondo de la página, del borde de la cabecera, del fondo de la cabecera.
- No permite copias de seguridad y no es compatible con ningún estándar e-learning

- El sistema protege el acceso a los cursos mediante un usuario y una contraseña. El sistema puede autentificarlos además contra un servidor LDAP externo o usando el protocolo Kerberos.

- Los profesores pueden asignar diferentes niveles de acceso a los cursos basados en una serie de roles predefinidos: profesores, ayudantes, estudiantes, invitados y "staff". También se puede configurar los privilegios de los estudiantes.

- Los usuarios tienen disponible una herramienta de correo interno con características básicas. Además, se pueden enviar correo a cuentas externas.

- No tiene tablón de anuncios.

- Existe una sencilla herramienta de foro. Los estudiantes pueden enviar mensajes de forma anónima.

- Está disponible una página personal para el estudiante en el que se puede incluir también una foto aparte de diferentes referencias personales.

- No posee ni agenda, ni marcadores ni la opción de crear grupos de trabajo.

- Existen diferentes tipos de ejercicios que los estudiantes pueden realizar varias veces.

- Los contenidos del curso pueden ser enviados al servidor a través de un formulario.

- Se utiliza una plantilla para seguir paso a paso el proceso de composición de las características principales del curso.

- Los profesores pueden poner fechas de inicio y final a los materiales del curso. Pueden elegir tareas que sean imprescindibles de realizar para continuar con el resto del contenido.

- Los profesores pueden crear ejercicios de verdadero/falso, múltiple elección, múltiple respuesta, ordenación, rellenar huecos, seleccionar la parte correcta de una imagen, relacionar términos, etc. Los ejercicios pueden contener imágenes, video y otros tipos de archivos multimedia. También pueden crear una base de datos de cuestiones que el sistema elegirá aleatoriamente para crear un examen diferente para cada estudiante.

- Es posible el intercambio de ficheros con el servidor.

Formato de ejercicios Lon-Capa

Lon-Capa cuenta con varias plantillas para poder crear ejercicios a partir de las mismas, aunque también se pueden crear ejercicios personalizados sin la necesidad de utilizarlas (Abbott & September, 2012).

Hay tres representaciones diferentes en los problemas Lon-Capa conocidas como "Edit", "EditXML" y "view".

Representación "Edit"

Esta representación se basa en cajas de colores que representan cada elemento funcional o un grupo de elementos funcionales. Por ejemplo alguno de estos posibles elementos son:

- **Script:** Es el elemento donde se definen las variables para los cálculos.
- **Text Block:** Es el elemento donde se puede añadir directamente el texto del problema.
- **Response:** Es el elemento donde se puede pedir la evaluación de la respuesta del alumno.

Representación "EditXML"

Esta representación se basa simplemente en introducir código xml. Este código xml tiene que cumplir unas condiciones para que concuerde con la anterior representación. Estas condiciones son las siguientes:

- El código debe empezar con <problem> y terminar con </problem>
- Entre <script> y </script> se deben definir las variables que se utilizarán en los cálculos. (El lenguaje utilizado en esta área es una mezcla entre Lon-Capa y Perl).
- Entre <startouttext /> y <endouttext /> se puede añadir el texto del problema directamente.
- Entre <numericalresponse answer="$c"> y </numericalresponse> se puede pedir la evaluación de la respuesta del estudiante y también se puede añadir parámetros.

Representación "view"

En esta representación no aparece el código, sólo aparece el texto del problema y el cuadro donde debe ir la respuesta. Además si el usuario es el instructor, también podrá introducir información sobre la respuesta correcta y cuál es el rango de respuestas que se considerarán aceptadas.

Una parte muy importante del formato Lon-Capa es la evaluación de las respuestas.

La respuesta correcta simplemente debe encontrarse en el campo "answer" y puede ser tanto un número como una variable, en cambio la unidad física de la respuesta debe ir en el campo "unit".

Lon-Capa acepta cualquier respuesta de un estudiante que tenga un tamaño correcto y se encarga de hacer la conversión.

Por último existe un campo llamado "format" que es usado para que la respuesta se muestre correctamente en cualquier ordenador.

Por otro lado, otra parte importante del formato Lon-Capa es la utilización de los "hint", que son las ayudas o pistas que se le muestran al estudiante.

Para entender cómo funcionan hay que aclarar tres conceptos:

- Un "hint" es una pista que aparece en un cuadro de texto después de que el alumno haya intentado resolver el problema o cuando el instructor quiera enseñarlo.

- En "hint condition" indica cuándo debe aparecer la pista y Lon-Capa se encarga de evaluar esa condición.

- El "Text/HTML Block" es la respuesta que debe aparecer cuando se cumpla la condición anterior.

Ejercicios Lon-Capa en la plataforma edX

Los ejercicios de tipo Lon-Capa están definidos por la clase LoncapaProblem que se encuentra en el archivo edX-platform/common/lib/capa/capa/capa_problem.py. Parte del contenido d esta sección pertenece al código de la plataforma de edX.

Clase LoncapaProblem

Un ejercicio Lon-Capa tiene los siguientes atributos:

- **problem_text** (string): Contiene el xml del problema

- **id** (string): Es el identificador del problema, la mayoría de las veces es el nombre del archivo.

- **capa_system** (LoncapaSystem): Es una encapsulación de los recursos que se necesitan del exterior.

- **state** (dict): Es un diccionario que contiene las siguientes claves:

    - **seed** (int): Es un número aleatorio que es generado por el método seed.

    - **student_answers** (dict): Es un diccionario que guarda las respuestas dadas en cada entrada.

    - **correct_map** (CorrectMap): Indica si la respuesta es correcta, incorrecta, el número de puntos conseguidos con esa respuesta, etc.

    - **done** (bool): Indica si el problema es considerado como hecho o no.

    - **input_state** (dicts): Es un diccionario que contiene el estado de cada entrada.

- **seed** (int): Es el número aleatorio generado.

En esta clase aparecen algunos métodos importantes como:

- **do_reset():** Restablece el estado interno a inacabado, quitando todas las respuestas.

- **get_state():** Almacena los datos necesarios de cada estudiante para recuperar el estado del problema del estudiante.

- **get_max_score():** Devuelve la máxima puntuación del problema.

- **get_score():** Devuelve la puntuación obtenida en el problema.

- **update_score():** Devuelve un CorrectMap actualizado al objeto ResponseType que lo ha solicitado.

Clase CorrectMap

Esta clase se encuentra en la ruta edx-platform/lib/capa/capa/correctmap.py y guarda un objeto con la id de la pregunta (answer_id) y la evaluación de la respuesta para cada una de las preguntas de los problemas de Lon-Capa. Esta evaluación de la respuesta incluye los campos:

- **Correctness:** Indica si la respuesta es correcta o incorrecta
- **npoints:** Es un entero que indica el número de puntos conseguidos con esa respuesta.
- **msg** String (puede tener un HTML): Proporciona un mensaje extra para la respuesta. (Aparece debajo del texto y del cuadro de respuesta).
- **Hint**String (puede tener un HTML): Proporciona una pista opcional. (Aparece debajo del texto, del cuadro de respuesta y del msg).
- **hintmode:** Contiene el criterio para mostrar el hint. Los valores posibles son 'None' (nunca), 'on_request' (cuando se pida) o 'always' (siempre).
- **Queuestate:** Diccionario que contiene una "key" que es un string secreto y un 'time' que guarda el tiempo.

En cuanto a los métodos hay que destacar el método **set_dict(correct_map)** que se encarga de establecer el diccionario interno de la clase CorrectMap con el que entra por parámetro (correct_map).

El 'correct_map' es enviado por el LMS como un JSON dumps del CorrectMap. Esto significa que cuando se cambia la definición de CorrectMap (por ejemplo se añade nuevas propiedades), el CorrectMap existente no coincide con el formato de la nueva versión. Para solucionar esto, se comparan los contenidos en lugar de hacer una copia directa, de esta forma las claves comunes serán escritas y las que no lo estén serán ignoradas.

Hay una excepción cuando 'correct_map' es un diccionario de un solo nivel, ya que se convierte en el nuevo formato de diccionario.

Clase Inputtypes

Esta clase se encuentra en la ruta edX-platform/lib/capa/capa/inputtypes.py y contiene los elementos de los problemas que pueden ser utilizados como objetos de entrada en los mismos.

Clase LonCapaResponse

Esta clase se encuentra en la ruta edX-platform/lib/capa/capa/responsetypes.py y contiene los tipos de respuestas que pueden aparecer en un ejercicio Lon-Capa.

Cada tipo de respuesta hereda de la Clase LonCapaResponse, y debe redefinir los siguientes métodos:

- **get_score():** Evalúa las respuestas del estudiante y devuelve un CorrectMap con los resultados.

- **get_answers()**: Devuelve un diccionario con las respuestas esperadas para este problema.

Cada tipo de problema debe definir también el siguiente atributo:

- **Tags:** xhtml tags identificando la respuesta.

Además se pueden definir otros métodos de forma opcional:

- **setup_response():** Busca y anota los identificadores de la respuesta del campo de entrada. Es llamado por _init_
- **Check_hint_condition()**: Chequea si la respuesta del estudiante satisface la condición del hint para ser mostrada.
- **Render_html:** Representa la respuesta como un HTML.
- **_unicode_:** Representación unicode de la respuesta.

También se puede definir atributos como:

- **max_inputfields:** Guarda el máximo número de respuestas en un campo de entrada.
- **allowed_inputfields:** Lista de campos de entrada permitidos para cada respuesta.
- **Required_attributes:** Lista de atributos necesarios en la respuesta principal.
- **hint_tag:** xhtml tag que identifica la pista asociada con la respuesta del grupo de pistas.

4.1.2. Comunicación de Lon-Capa

En esta sección se describe la comunicación entre el JavaScript externo y la plataforma de edX, así como la evaluación que realiza Lon-Capa sobre el ejercicio. Además se describe las ventajas y las desventajas de utilizar este tipo de ejercicio.

Comunicación

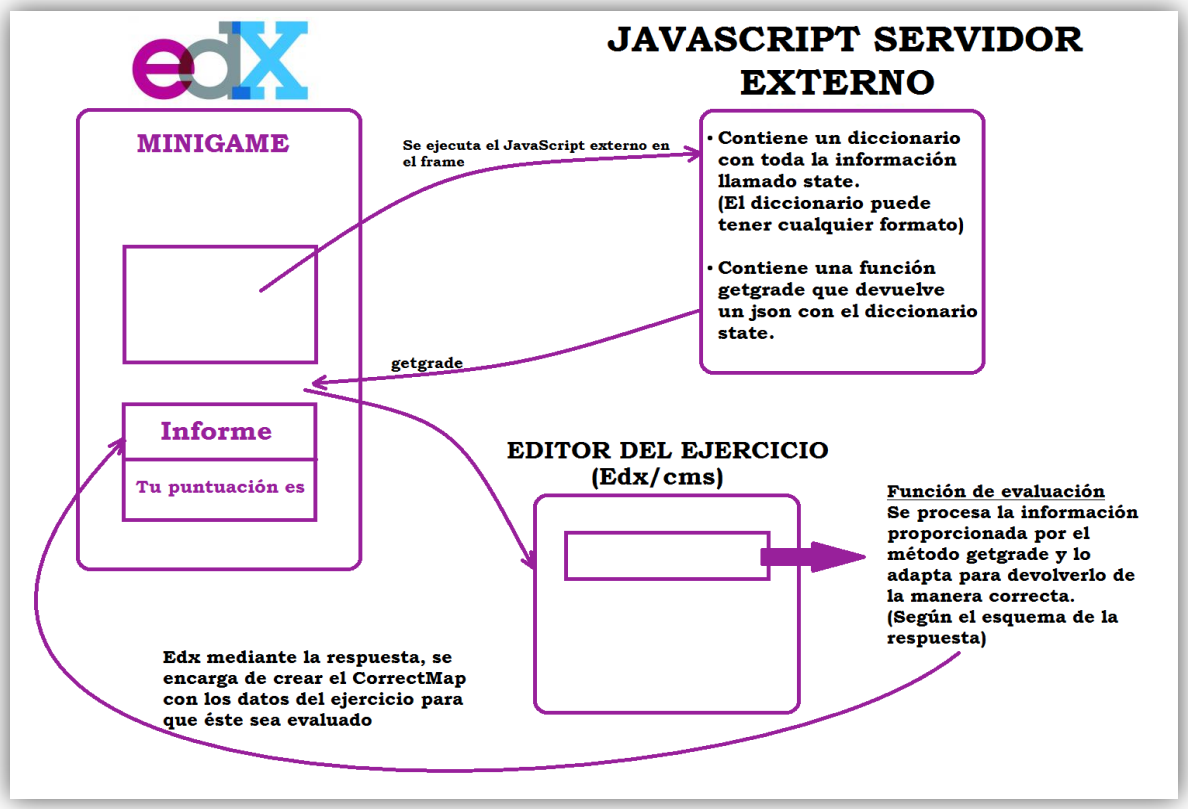


Imagen 4-1: Esquema de comunicación de Lon-Capa.

4.1.3. Ventajas y desventajas

Las ventajas y las desventajas de este tipo de ejercicio son:

Ventajas:

1. Se puede insertar cualquier JavaScript sin necesidad de que lo inserte el administrador.

2. No es necesario crear nuevas tablas en la base de datos, ya que toda la información se guarda en el CorrectMap. La información se guarda de forma automática en una tabla ya predefinida (courseware.studentmodule).

3. Para el autor es sencillo introducir este tipo de ejercicio sólo tiene que escribir el link del JavaScript a ejecutar y definir la función de evaluación.

Desventajas:

1. El JavaScript está en un servidor externo así que no tiene por qué tener la misma disponibilidad que edX.

2. El autor debe conocer la estructura del diccionario guardado en el JavaScript para poder hacer su función de evaluación.

3. Si el alumno no completa el ejercicio Lon-Capa no es capaz de recuperar el estado y se pierden todos los progresos. Este problema se solucionará con xAPI como se verá más adelante (4.3. El estándar xAPI).

4.2. Learning Tools Interoperability

En esta sección se explicará en qué consiste un módulo LTI ("Learnig Tools Interoperability"), que es la tercera forma de integrar un JavaScript en edX. Para ello se presentarán las bases conceptuales y los principios de la arquitectura en la que se basa la norma LTI.

4.2.1. Introducción

El concepto principal de LTI es establecer una forma estándar de integrar aplicaciones de aprendizaje con plataformas como learning management systems (LMS), portales u otros entornos educativos. En LTI estas aplicaciones de aprendizaje se llaman Tools (aplicaciones), y el LMS o plataformas se denominan Tool Consumers (consumidores de aplicaciones) ("IMS Global: LTI," 2004).

La función principal de LTI es permitir la conexión segura de aplicaciones web y contenidos alojados externamente, con las plataformas que se les presenta a los usuarios. En otras palabras, si se quiere incorporar una aplicación de evaluación interactiva o laboratorio de química virtual mediante LTI, se puede conectar de forma segura a una plataforma educativa de una manera estándar, sin tener que desarrollar ni mantener integraciones personalizadas para cada plataforma.

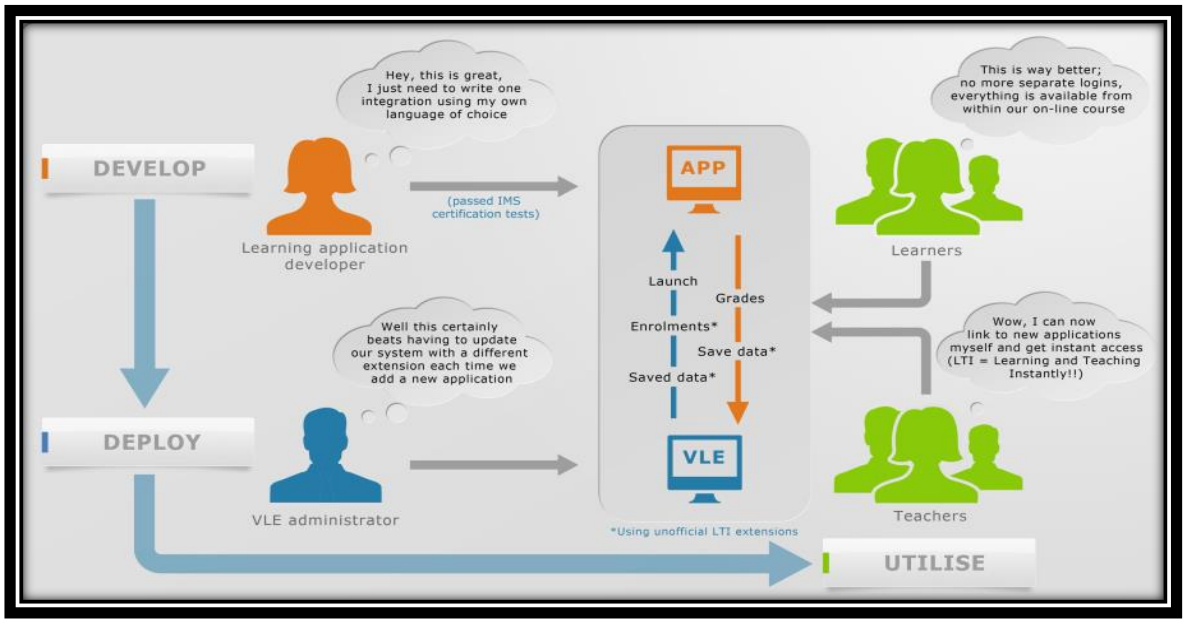


Imagen 4-2: Esquema de las ventajas que aporta LTI ("IMS Global: LTI," 2004).

4.2.2. La descripción básica del funcionamiento de LTI

El mercado de la educación de hoy en día incluye un número creciente de aplicaciones basadas en la web de alta calidad que mejoran la enseñanza y el aprendizaje ("IMS Global Developer," 2004).

Idealmente, cualquier Sistema de Gestión de Aprendizaje (LMS) debe facilitar el acceso a las aplicaciones de aprendizaje. Debería ser posible mezclar y hacer coincidir estas aplicaciones en el contexto de un curso dado.

El estándar IMS LTI tiene como objetivo ofrecer un marco único para la integración de cualquier producto LMS con cualquier aplicación de aprendizaje.

Los objetivos de la norma de LTI

Los objetivos son:

1. Proporcionar un modelo de implementación que consiste en una dirección URL, la clave y la contraseña secreta que el administrador del LMS o el instructor del curso ha de introducir en el LMS.

2. Definir un protocolo para el lanzamiento de una aplicación externa de un LMS en una manera que apoye y que conserve el contexto de aprendizaje y las funciones del usuario dentro de ese contexto.

3. Realizar enlaces a aplicaciones externas mediante la definición de los elementos de datos que se pueden introducir en un IMS común.

Conceptos clave

Desde el punto de vista del instructor, se suele añadir una aplicación LTI en su estructura del curso como un enlace de recursos mediante el panel de control de LMS. El instructor introduce la URL, la clave y la contraseña secreta como meta datos para el enlace de recursos. Cuando los estudiantes seleccionan la aplicación, el LMS utiliza la dirección URL, la clave y la contraseña secreta para poner en marcha, sin ningún tipo de problema, la aplicación remota mediante un iframe o una nueva ventana del navegador ("IMS Global Developer," 2004).

Desde el punto de vista del administrador, por lo general, se proporciona un "instrumento virtual" en el LMS para que sea posible escribir la dirección URL, la clave y la contraseña secreta. Una vez hecho esto, los profesores simplemente verán la herramienta LTI recién configurada como una aplicación o actividad para ser colocada como un enlace de recursos en su estructura del curso. Los profesores y los estudiantes pueden incluso no ser conscientes de que la aplicación que están utilizando se está ejecutando fuera del LMS. Ellos sólo tienen que seleccionar y utilizar la herramienta como cualquier otra herramienta que se construye en el LMS.

A continuación se expone una imagen explicando cómo y dónde se añade el módulo avanzado LTI al curso edX.

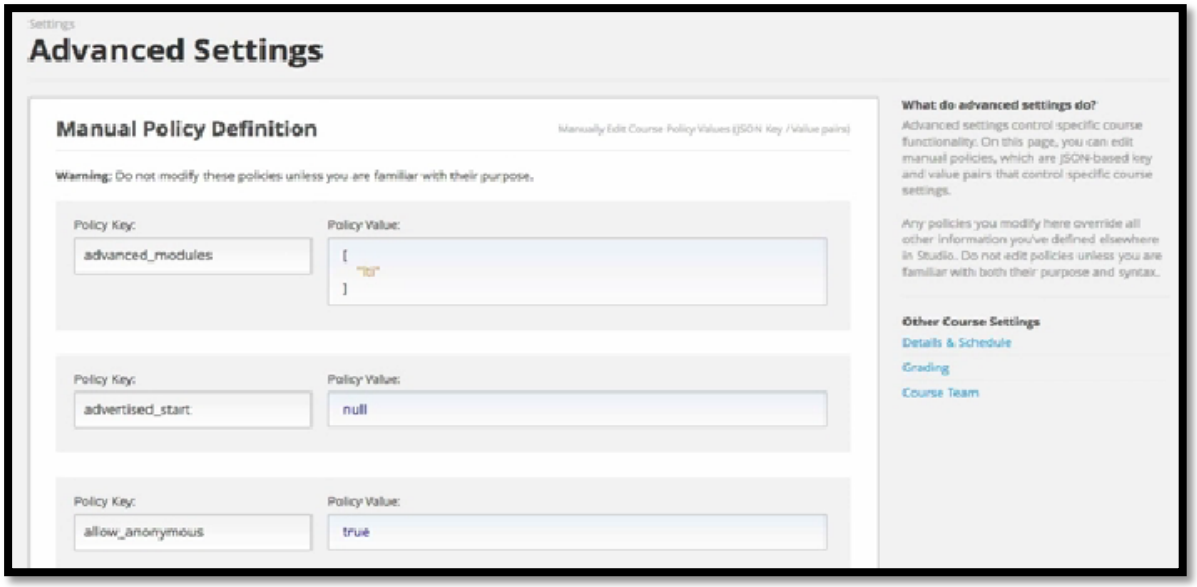


Imagen 4-3: Añadiendo el módulo avanzado LTI.

A continuación, en este mismo sitio se tiene que añadir la dirección URL, la clave y la contraseña secreta que se comentó anteriormente.

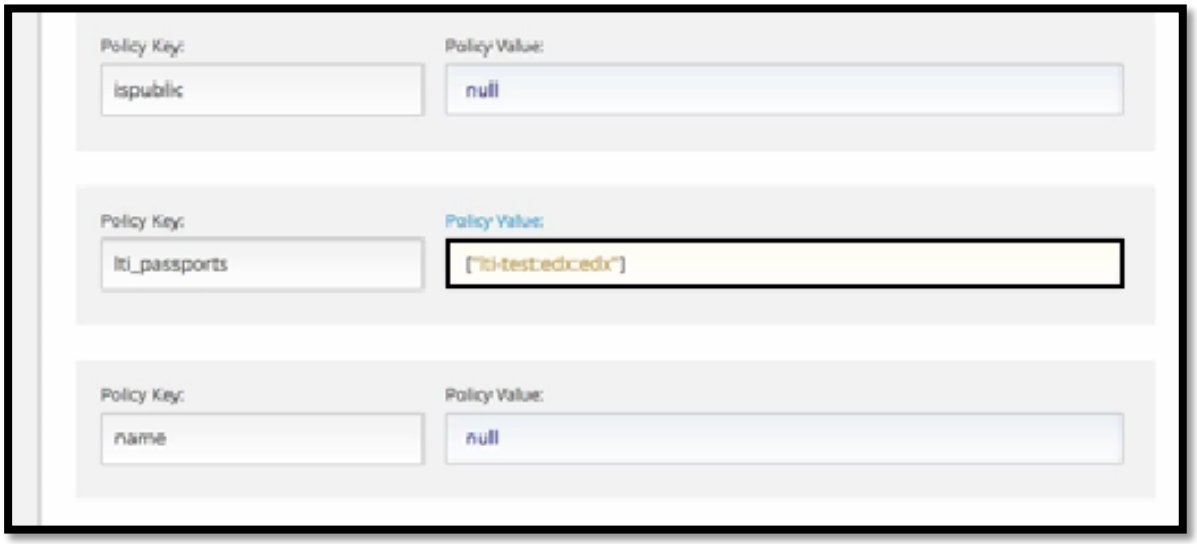


Imagen 4-4: Añadiendo la URL, la clave y la contraseña secreta, separados mediante dos puntos.

El modo de introducir los datos, como se puede observar en la imagen anterior, sigue el siguiente patrón: [URL:clave:contraseña_secreta]

Tanto para el instructor como para el administrador, la herramienta externa recibe una petición de lanzamiento que incluye la identidad del usuario, la información del curso, información de la función, la clave y la firma. La información se envía mediante un formulario HTTP generado en el navegador del usuario con los datos LTI en campos ocultos de formulario y son enviados automáticamente a la aplicación externa mediante JavaScript. Los datos en el formulario de HTTP se firma con el OAuth estándar de seguridad("OAuth Community Site," 2014), por lo que la

aplicación externa puede estar segura de que los datos de lanzamiento no se modifican hasta que la aplicación los recibe ("IMS Global Developer," 2004).

4.2.3. Conceptos

Los conceptos más importantes de LTI según ("IMS Global Developer," 2004) son los siguientes:

Tool Provider (Proveedor de aplicaciones):

Se trata de un proveedor de aplicaciones que expone interfaces para una o más aplicaciones. Como se ilustra en la imagen (Imagen 4-2), es útil pensar en el proveedor de aplicaciones como contenedor de una aplicación.

Consumer Tool (Consumidor de aplicaciones):

Los instructores y los estudiantes acceden a las herramientas mediante la activación de enlaces dentro de un LMS.

Enlace de Recursos:

El consumidor de aplicaciones utiliza entidades de enlace de recursos para generar enlaces clickeables dentro de su interfaz de usuario. Cada link de recursos tiene un título que define el texto que debe aparecer en el enlace clickeable y además una descripción opcional que aparecerá junto al enlace.

4.3. El estándar xAPI

En esta sección se explicará en qué consiste el estándar xAPI que será el utilizado para evaluar los resultados obtenidos por los estudiantes en la plataforma edX.

4.3.1. En qué consiste xAPI

The experience API (conocido como xAPI o The TinCan API) es un componente de TLA (*Training and Learning Architecture*). El propósito general de xAPI es almacenar y proporcionar acceso a los datos de las experiencias de aprendizaje (learning experiences). xAPI permite el seguimiento de las experiencias de aprendizaje, incluyendo los datos tradicionales, como las puntuaciones parciales o finales. No obstante es más amplio y también puede usarse para almacenar otros datos de las acciones de los alumnos, como pueden ser la lectura de un artículo, ver un vídeo de entrenamiento o haber realizado cualquier otra tarea ("xAPI," 2014)("Tin Can API | Jose Manuel Martín," 2012).

xAPI está diseñado para dar soporte a los casos de uso que ya se consideraban en SCORM. SCORM es el estándar más usado de e-learning para el empaquetamiento de contenidos, pero ahora se está cambiando a xAPI por dos motivos. El primero es que éste soporta otros casos de uso que son difíciles de implementar con SCORM, como la formación con dispositivos móviles (que pueden no estar siempre conectados), el contenido al que se accede fuera de un navegador web o la posibilidad de guardar cualquier experiencia de aprendizaje donde quiera y cuando quiera que pase. El segundo motivo es lo sencillo que resulta hacer que un contenido SCORM funcione en un sistema con xAPI. Para ello se puede utilizar el siguiente enlace ("SCORM to Tin Can Solution," 2014).

En la siguiente tabla obtenida de ("SCORM vs Tin Can," 2014), se puede ver un pequeño resumen de las diferencias entre SCORM y xAPI:

TIN CAN API	SCORM	ACCIÓN	TIN CAN API	SCORM	ACCIÓN
✓		Transición de Plataforma (Ordenador a móvil)	✓	✓	Finalización de la actividad
✓		Seguimiento de juegos serios	✓	✓	Tiempo de la actividad
✓		Seguimiento de simulaciones	✓	✓	Actividad Pass/Fail
✓		Seguimiento del aprendizaje informal	✓	✓	Informar de puntuación única
✓		Seguimiento del rendimiento del mundo-real	✓		Informar de puntuación múltiple
✓		Seguimiento offline del aprendizaje	✓		Detalles de los resultados
✓		Seguimiento del aprendizaje interactivo	✓		Seguridad sólida
✓		Seguimiento del aprendizaje adaptativo	✓		No requiere LMS

✓		Seguimiento del aprendizaje combinado	✓		No requiere navegador web
✓		Seguimiento del aprendizaje a largo plazo	✓		Mantener el completo control sobre el contenido
✓		Seguimiento del aprendizaje basado en equipo	✓		Sin limitación del dominio cruzado
✓		Uso de Apps móviles para el aprendizaje			

xAPI es una especificación que describe una interfaz de comunicación y unas reglas de almacenamiento y recuperación de datos que los desarrolladores pueden implementar para crear un seguimiento de las actividades de aprendizaje. Esta API permite capturar datos en un formato consistente sin imponer restricciones sobre la tecnología utilizada o sobre si se almacenan datos de una persona o sobre un grupo de actividades. El servicio funciona al permitir que las declaraciones de actividad (por lo general las experiencias de aprendizaje, pero podría ser cualquier otro tipo de experiencia) que se envían, se almacenen de forma segura en un LRS (*Learning Record Store*) ("xAPI," 2014)("Tin Can API | Jose Manuel Martín," 2012).

LRS es un sistema que nos permite guardar en la nube los registros de actividad. Este sistema puede estar integrado dentro de un LMS (e.g. Moodle) o por el contrario ser totalmente independiente. LRS nos permite identificar quién, cómo y qué ha realizado una actividad.

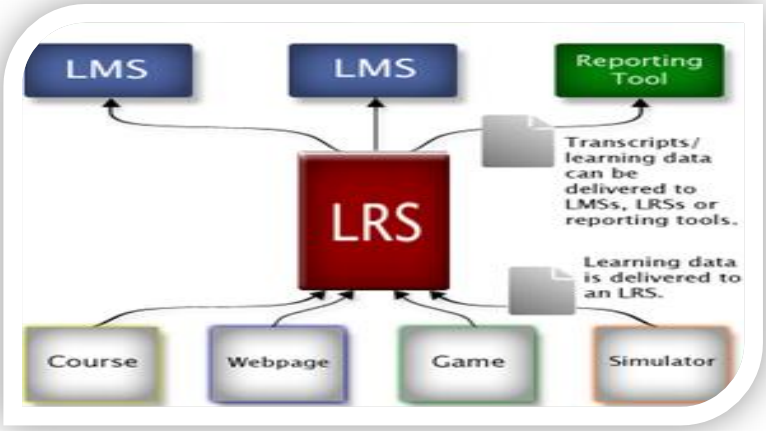


Imagen 4-5: Esquema de comunicación con el LRS.("Tin Can API | Jose Manuel Martín," 2012)

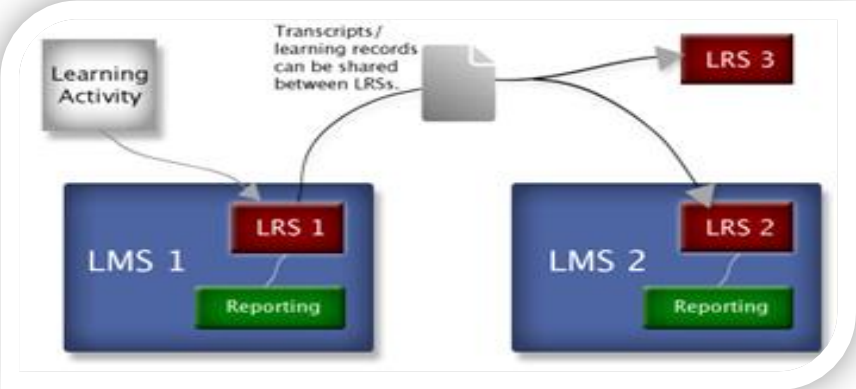


Imagen 4-6: Comunicación entre 2 LRS integrados en LMS y un LRS independiente.("Tin Can API | Jose Manuel Martín," 2012)

El formato de la información que se envía está basado en statements (declaraciones de estado). Es muy similar al concepto *Activity stream* ("Activity Streams," 2014) de las redes sociales. Los statements comunican los avances al LRS basándose en tres conceptos: Actor, Verbo, Objeto (por ejemplo, "Yo hice esto"). En este formato, el actor es el agente de la declaración, puede ser un alumno, un profesor o un grupo. El verbo describe la acción del estado, tal como lectura, pasado, o enseñado. Y el objeto es con lo que el actor interactúa, como un libro, una prueba, o una clase ("Tin Can API | Jose Manuel Martín," 2012).

Esta sintaxis que sirve para registrar el aprendizaje de un grupo de actividades se trata de en un JSON con una estructura definida en la especificación. En la sección (4.3.4. Tin Can Statement) se describe las partes de un statement.

Si se necesita un statement que no se encuentra en el registro se puede crear uno fácilmente con la herramienta que Tin Can API proporciona ("Statement Generator," 2014) o incluso se puede comprobar la validez de alguno que se haya creado por cuenta propia.

Existe un LRS público perteneciente a Tin Can API pero éste solo sirve para hacer pruebas, ya que los statements que se encuentran no son permanentes. Una de las maneras más rápida de construir un LRS para poder testear de forma privada y permanente es creando un *"SCORM Cloud account"* ("Scorm Cloud SCORM Cloud," 2014), aunque la versión gratuita solo permite un número limitado de registros al mes y un almacenamiento de 100Mb.

Por lo tanto, la mejor solución es que el propio desarrollador cree su propio LRS. Construirlo es complicado pero en este enlace la sección (4.4.2. LRS propio) se explican los pasos básicos para poder hacerlo.

El estándar Tin Can API según ("Tin Can Api ¿El futuro sustituto del SCORM?," 2012) se estructura en 4 capas:

Capa nivel 1: En esta capa del Tin Can API se ha desarrollado una nueva y optimizada versión del estándar SCORM. Permitiendo:

- Realizar el seguimiento en entornos desconectados o semiconectados, es decir, desde cualquier dispositivo (Smartphones, tablets, gps, etc...) a cualquier servidor tanto online como offline.

- Lanzar contenidos almacenados fuera del LMS.
- Mantener un control total sobre la distribución de contenidos y seguimiento del usuario.
- Guardar múltiples resultados para un curso (test inicial, test final, etc…)
- Permitir múltiples intentos.
- Resultados detallados de las pruebas.

Capa nivel 2: En esta capa se ha desarrollado la tecnología necesaria para poder grabar cualquier actividad de aprendizaje informal. Es decir, cualquier actividad relacionada con las redes sociales (Facebook, Twitter, linkedIn, Youtube, etc...), uso de recursos Web 2.0 (Blog, Wiki, etc...), actividades con dispositivos móviles (Smartphones, tablets, etc...) y actividades fuera de la red.

Capa nivel 3: En esta capa se ha desarrollado el acceso de lectura y escritura de los datos.

Tin Can exige que los LRS contengan datos accesibles, permitiendo tanto la grabación de datos, como su lectura. Los LRS pueden ser exportados a herramientas de informes, herramientas de análisis e incluso a otros LRS. Permitiendo el análisis de resultados y contenido.

Capa nivel 4: Permite analizar si la formación realizada es útil para el puesto de trabajo.

Los flujos de actividad están ganando fuerza como forma de controlar la actividad de una persona, tanto en las redes sociales como en la empresa. Es decir, los datos reales de rendimiento del trabajo y los datos de ocio están convergiendo. Esto permitirá en un futuro identificar los mejores itinerarios formativos y analizar la efectividad y ROI de la formación.

4.3.2. Comunicación en tiempo de ejecución

Transferencia de los statements entre el proveedor y el LRS

Según ("Github Experience API," 2014) los requisitos de la transferencia de statements entre el proveedor y el LRS son los siguientes:

Requisitos del LRS

- El LRS debe incluir la cabecera "X-Experiencia-API-Version" en cada respuesta.
- El LRS debe establecer este encabezado a "1.0.1 (Versión actual)".
- El LRS acepta solicitudes con un encabezado de versión "1.0", como si el encabezado fuese "1.0.0".
- El LRS debe rechazar las solicitudes que tienen la versión anterior a "1.0.0" a menos que tales peticiones se dirijan a una aplicación totalmente conforme a la versión anterior especificada en la cabecera.
- El LRS debe rechazar las peticiones con una versión "1.1.0" o mayor.
- El LRS debe rechazar con mensaje de error HTTP 400, incluyendo una breve descripción del problema.

Requisitos del cliente

- El cliente debe incluir la cabecera "X-Experiencia-API-Version" en cada petición.
- El cliente debe establecer este encabezado a "1.0.1 (Versión actual)".
- El cliente debe tolerar respuestas con una versión de "1.0.0" o posterior.
- El cliente debe tolerar las estructuras de datos con propiedades adicionales.
- El cliente debe ignorar cualquier propiedad que no se definen en la versión 1.0.0 de la especificación.

Requisitos de conversión

- Los sistemas no deben convertir las declaraciones de las nuevas versiones en un formato de la versión anterior, con el fin de manejar versiones diferentes.
- Los sistemas pueden convertir declaraciones de versiones anteriores en una nueva versión sólo siguiendo los métodos descritos en el Apéndice D: La conversión de los estados a 1.0.0. ("Github Converting Statements to 1.0.0," 2014)

Concurrencia

La concurrencia se encarga de que no se pongan datos de las versiones anteriores en un LRS ("Github Experience API," 2014).

xAPI usará etiquetas entidad "eTags"("HTTP/1.1: Header Field Definitions," 2014) HTTP 1.1 para implementar el control de concurrencia optimista en las partes de la API donde "PUT" puede sobrescribir los datos existentes:

- Estado API
- Perfil del agente API
- Actividad Perfil API

En este enlace ("GitHub Documents Apis," 2014) se obtiene más información acerca de Estado API, Perfil del agente API, Actividad Perfil API

Los siguientes requisitos se aplican solamente a agente Perfil de la API y Actividad Perfil API:

Requisitos del cliente

- Un cliente que utiliza Agente Perfil API o Actividad Perfil API debe incluir el encabezado If-Match o el encabezado If-None-Match ("HTTP/1.1: Header Field Definitions," 2014)

Requisitos LRS

- Un LRS respondiendo a una solicitud "GET" debe agregar un encabezado HTTP eTag a la respuesta. (La razón para especificar el formato LRS eTag es no permitir a los consumidores de la API leer la cabecera eTag para calcularla ellos mismos.)
- Un LRS respondiendo a una solicitud "GET" debe calcular el valor de esta

cabecera para ser una cadena hexadecimal del SHA-1.

- Un LRS respondiendo a una solicitud “GET” debe encerrar el encabezado entre comillas.

- Un LRS respondiendo a una solicitud “PUT” debe manejar la cabecera If-Match como se describe en RFC2616, HTTP 1.1 si contiene un eTag, con el fin de detectar las modificaciones.

- Un LRS respondiendo a una solicitud “PUT” debe manejar la cabecera If-None-Match como se describe en RFC2616, HTTP 1.1 si contiene "*", con el fin de detectar cuando hay un recurso presente del cual el consumidor no es consciente.

Si la condición de cabecera en cualquiera de los casos de petición “PUT” falla, el LRS:

- Debe devolver el estado HTTP 412 "Precondition Failed".

- No debe realizar una modificación en el recurso.

Si una solicitud “PUT” se recibe sin ninguna cabecera para un recurso que ya existe, el LRS:

- Debe devolver el estado HTTP 409 "Conflicto".

- Debe devolver un cuerpo de texto sin formato y explicar que el consumidor debe:

    - Comprobar el estado actual del recurso.

    - Establecer el encabezado "If-Match" con el eTag actual para resolver el conflicto.

- No debe realizar una modificación en el recurso.

Seguridad

Con el fin de equilibrar la interoperabilidad y los requisitos de seguridad de diferentes entornos se definen varias opciones de autenticación (“Github Experience API,” 2014).

Usuario desconocido	Conocido usuario	
El LRS permite a la aplicación acceder a xAPI sin credenciales de usuario adicionales.	Flujo de trabajo estándar para OAuth.	La aplicación se ha registrado
	El Agente de aplicación no se identifica como un agente registrado y el LRS no puede hacer suposiciones sobre su identidad.	La aplicación no se ha registrado
	Se utiliza autenticación básica HTTP en lugar de OAuth, ya que la aplicación no está involucrada.	Ninguna aplicación
	Podrán recibir apoyo de los LRS, posiblemente con fines de prueba.	Sin autenticación

Requisitos:

El LRS debe soportar la autenticación mediante al menos uno de los siguientes métodos:

- OAuth 1.0 (RFC 5849), con los métodos de firma de "HMAC-SHA1", "RSA-SHA1", y "PLAINTEXT"
- Autenticación básica HTTP
- Tarjetas de Acceso Común (detalles de implementación a seguir en una versión posterior)
- El LRS debe manejar o delegar decisiones sobre la validez de los estados y determinar qué operaciones se pueden realizar basándose en las credenciales utilizadas.

4.3.3. Transferencia de datos

xAPI consta de 4 sub-APIs: Statement (Declaración), State (Estado), Agent (Agente), Activity Profile (Perfil de Actividad). Estas 4 sub-APIs son manejadas vía métodos RESTful HTTP

Requisitos:

- El LRS debe rechazar cualquier solicitud que use parámetros que no reconozca con el estado "HTTP 400 Bad Request".
- El LRS debe rechazar con el estado "HTTP 400 Bad Request" cualquier solicitud de estas APIs que usen cualquier parámetro coincidente con los parámetros descritos en la especificación.
- El LRS debe rechazar un lote de instrucciones si cualquier statement dentro de ese lote es rechazado.
- EL LRS debe devolver el código de error más apropiado acorde a la lista de errores que se puede encontrar en la sección 7.1 de la especificación de ("Github Error Codes," 2014).
- El LRS debe devolver un mensaje explicando la causa del error.

Statement API

PUT Statments:

- Almacena el statement con el id dado.
- Devuelve "204 No content".

Parámetro	Tipo	Defecto	Descripción	Requisito
statementId	String		Id del Statement a guardar	Requisito

Requisitos:

- Un LRS no debe hacer ninguna modificación a su estado si ya tiene un statement aunque le llegue un statement con un statementID. En este caso devolverá 409 Conflict o 204 No Content.

- Si el LRS recibe un statement con un ID que ya tiene un statement asignado, debería verificar que el statement recibido encaja con el existente, y devolver 409 Conflict si no es así.

- Los LRS pueden responder ante statements que han sido almacenados estando disponibles para su recuperación.

POST Statements:

- Almacena un statement o un conjunto de ellos sin generar un statementID.

- Devuelve:200 OK.

Requisitos:

- Un LRS no debe hacer ninguna modificación a su estado si ya tiene un statement aunque le llegue un statement con un statementID. En este caso devolverá 409 Conflict o 204 No Content.

- Si el LRS recibe un statement con un ID que ya tiene un statement asignado, debería verificar que el statement recibido encaja con el existente, y devolver 409 Conflict si no es así.

- Los LRS pueden responder ante statements que han sido almacenados estando disponibles para su recuperación.

- Los statements GET deben ser llamados usando POST y los campos del formulario si fuera necesario ya que las cadenas usadas en las consultas tienen un límite.

- El LRS debe diferenciar entre un POST y añadir un statement o una lista de statements basados en los parámetros pasados.

Get Statements:

- Este método puede ser llamado para traer uno o múltiples statements. Si se especifica el parámetro statementId o voidedStatementId se devuelve un único statement. En caso contrario, devuelve un objeto StatementResult que es una lista de los statements en orden cronológico inverso basado en el tiempo de guardado, subject to permissions y la longitud máxima de la lista. Si los resultados adicionales están disponibles, se incluirá un IRL (Internationalized Resource Locator) para recuperarlos en el objeto *StatementResult*.

- Returns: 200 OK.

Parámetro	Tipo	Por defecto	Descripción	Necesario
statementId	String		ID del Statement que se quiere obtener	Opcional
voidedStatementId	String		ID del "void Statement" que se quiere obtener	Opcional
Agente	Agente u objeto de Identificador de grupo (JSON)		Es un filtro. Sólo devuelve los statements si el agente o grupo especificado son el actor o el objeto del Statement. • Los agentes o grupos identificados son iguales cuando tienen valores iguales. • Según este filtro, se considera una coincidencia a los grupos que tengan miembros que coincidan con el agente especificado en base a su identificador.	Opcional
verbo	Verbo id (IRI)		Es un filtro. Sólo devuelve los statements que coincidan con el ID del verbo especificado	Opcional
actividad	Actividad id (IRI)		Es un filtro. Sólo devuelve los statements para los cuales el objeto del statement es la actividad con el id especificado.	Opcional

registro	UUID		Es un filtro. Sólo devuelve los statements que coincidan con el id de registro especificado.	Opcional
“related_activities”	Boolean	Falso	Aplica el filtro de actividad general. Incluye los statements para las cuales el objeto, cualquiera de las actividades de contexto, o cualquiera de esas propiedades contenidas en una Sub-Statement coincida con el parámetro de actividad, en lugar de la conducta normal de ese parámetro. La concordancia se define de la misma manera que para el parámetro 'de actividad' ".	Opcional
“related_agents”	Boolean	Falso	Aplicación amplia del filtro. Incluye los statements para los cuales el actor, los objetos, la autoridad, el instructor, el equipo o cualquiera de las propiedades de un sub-Statement coincidan con el parámetro agente en lugar de la conducta normal de un parámetro. La concordancia se define de la misma manera que para el parámetro “agente”	Opcional
“since”	Timestamp		Sólo se devuelven los estados almacenados desde la marca de tiempo (exclusivo) especificado.	Opcional
“until”	Timestamp		Sólo se devuelven los estados almacenados antes o hasta la fecha y hora especificada.	Opcional
“limit”	Nonnegative Integer	0	Número máximo de declaraciones a devolver. 0 indica que devolverá el máximo que el servidor permite.	Opcional

“format”	String: ("ids", "exact", or "canonical")	exacto	• Si es "ids", sólo incluyen información mínima necesaria de agente, actividad y objetos de grupo para identificarlos. Para grupos anónimos esto significa que incluye la información mínima necesaria para identificar a cada miembro. • Si es "exact", devuelve agente, actividad y objetos de grupo de población tal y como estaban cuando se recibió el statement. Para un LRS que solicita statements para su importación debería que utilizar un formato de "exacta". • Si es "canonical", devuelve los objetos de actividad que satisfacen la definición canónica de la actividad de objetos según lo determinado por el LRS. Devolverá el objeto original de agente como en el modo "exacto".	Opcional
“Attachments”	Boolean	Falso	Si está el parámetro tiene el valor “cierto”, el LRS usa el formato de respuesta ‘multiparte’ e incluye todos los archivos descritos anteriormente. Si está el parámetro tiene el valor “falso”, el LRS envía la respuesta con “Content-Type application/json” y no puede usar los archivos adjuntos.	Opcional
“Ascending”	Boolean	Falso	Si está el parámetro tiene el valor “cierto”, devuelve los resultados en orden ascendente según el tiempo de almacenado.	Opcional

Requisitos:

- El LRS debe rechazar con un error "HTTP 400" cualquier solicitud para este recurso que contengan ambos parámetros (statementId y voidedStatementId).

- El LRS debe rechazar con un error "HTTP 400" cualquier solicitud para este recurso que contenga un "statementId" o un "voidedStatementId" y además cualquier otro parámetro de "attachments" o "format".

- El LRS debe incluir la cabecera X-Experience-API-Consistent-Through en ISO 8601" combinada con el formato de fecha y hora en todas las respuestas a las solicitudes de Statements, con un valor de la marca de tiempo para cada statement que tiene o tendrá guardado para su posible recuperación.

- Si la propiedad "attachment" de un statement de tipo "GET" es usada y establecida a cierto, el LRS debe usar el formato de respuesta multiparte e incluir todos los "attachments"

- Si la propiedad "attachment" de un statement de tipo "GET" es usada y establecida a falso, el LRS no deben incluir los datos adjuntos tal y como aparecen y deben reportar "application/json.

voided Statement:

- El LRS no debe devolver cualquier statement que ha sido anulado, a menos que el statement haya sido requerido por voidedStatementId

- El LRS debe devolver cualquier statement dirigido a "voided Statement" cuando recupera statements usando tiempo (explícito o implícito) o una secuencia basada en recuperación, a menos que ellos mismos hayan sido anulados. Esto incluye el "voided statement" que no puede ser anulado. La herramienta de informes puede identificar la presencia y el StatementId de cualquier "voided statement".

4.3.4. Tin Can Statement

En esta sección se nombrará y explicará los componentes de un statement incluyendo ejemplos para su mejor comprensión. La estructura más simple de un statement de Tin Can es "actor verbo objeto", pero se podrá añadir más campos para dar una información más completa.

Actor

Es el que realiza la acción. No tiene por qué ser solamente identificado por un sistema o una ID, también puede ser un nombre. Cada statement sólo contendrá una representación de una única persona, sin embargo se podrá referir a múltiples personas siempre que el sistema conozca de antemano quien pertenece a ese grupo ("Statements 101," 2014).

Un ejemplo de un objeto de actor sería:

```
{
    "name":"Sally Glider",
    "mbox":"mailto:sally@example.com"
}
```

Este actor tiene dos propiedades, "name" y "mbox". Puede haber muchas personas con ese nombre pero solamente una con esa dirección de email.

Verbo

Los verbos en Tin Can son URIs ("Uniform Resource Identifier," 2014) y son elementos fundamentales de los estados ya que describen lo que ha sucedido entre el actor y el objeto del statement ("Statements 101," 2014) ("Verbs," 2014) .

Un ejemplo podría ser el verbo "accepted":

```
{
  "name": {
     "en-US": "accepted"
  },
  "description": {
     "en-US": "Indicates that that the actor has accepted the object. For instance, a person accepting an
award, or accepting an assignment."
  }
}
```

Este verbo podría utilizarse para saber si una persona ha aceptado un premio o una asignación.

Objeto

Terminando con el núcleo de la estructura de un statement aparece el campo objeto. Normalmente el objeto es otra actividad (definido de forma única por un URI) de Tin Can, aunque podría ser otro actor u otro statement ("Statements 101," 2014).

Un ejemplo sería:

```
{
 "id" : "http://example.com/activities/solo-hang-gliding" ,
 "definition" : {
   "type" : "course" ,
   "name" : {
      "en-US" : "Solo Ala Delta " ,
      "es" : "Solo Ala Delta"
    },
    "description" : {
      "en-US" : "El curso 'Solo Ala Delta' proporcionado por el Club del ala delta" ,
      "ES" : "El Curso de 'Solo Ala Delta' Siempre Por El Club de Planeadores Hang "
    },
    "extensiones" : {
      "http://example.com/gliderClubId" : "Por supuesto-435"
    }
}
```

Esta actividad está definida de forma única por su ID. También hay un campo "definition" en el que se incluyen subcampos como "type", "name" o "description". Del subcampo extensiones se hablará más adelante.

Contexto

El campo "context" ofrece un lugar para añadir información sobre un statement. Se puede añadir la información sobre la actividad en sí, sobre el instructor de la actividad o sobre cómo una experiencia encaja en alguna actividad más amplia ("Statements 101," 2014).

Por ejemplo:

```
{
 "actor":{
    "name":"Sally Glider",
    "mbox":"mailto:sally@example.com"
  },
   "verb":{
      "id":"http://adlnet.gov/expapi/verbs/experienced",
      "display":{"en-US":"experienced"}
  },
  "object":{
     "type":"course",
     "id":"http://example.com/activities/solo-hang-gliding",
     "definition":{
        "name":{"en-US":"Solo Hang Gliding"}
     }
  },


  "context":{
     "instructor":{
        "name":"Irene Instructor",
        "mbox":"mailto:irene@example.com"
      },
      "contextActivities":{
         "parent":{"id":"http://example.com/activities/hang-gliding-class-a"},
         "grouping":{"id":"http://example.com/activities/hang-gliding-school"}
      }
  }
}
```

En este ejemplo se añade la información del nombre del instructor del curso. La API nativa de consulta de Tin Can permite ir hacia atrás y pedir todos los statements hechos por el mismo usuario. Esto es de muchísima utilidad para lograr una agrupación de los datos.

Resultado

Un statement también puede acabar en un resultado concreto. En un statement se puede guardar el tiempo utilizado hasta conseguir el resultado, la respuesta del actor y si estaba en lo cierto o no ("Statements 101," 2014). Por ejemplo:

```
{
 "actor":{
    "name":"Sally Glider",
    "mbox":"mailto:sally@example.com"
  },

   "verb":{
      "id":"http://adlnet.gov/expapi/verbs/completed",
      "display":{"en-US":"completed"}
  },
  "object":{
     "id":"http://example.com/activities/solo-hang-gliding",
     "definition":{
        "name":{"en-US":"Solo Hang Gliding"}
     }
  },
  "result":{
     "completion":true,
     "success":true,
     "score":{
        "scaled":.95
     }
  }
}
```


En este caso se guarda si se ha terminado el ejercicio, si lo ha realizado correctamente y la puntuación obtenida.

Extensiones

A lo largo de esta sección ha aparecido en repetidas ocasiones el campo "extensions". Este campo está disponible dentro de las definiciones de una actividad, como parte del contexto de una declaración o como parte de algún resultado obtenido. Dado que este campo puede contener datos arbitrarios, abre una gran flexibilidad en lo que se puede expresar en un statement ("Statements 101," 2014).

Este campo hay que utilizarlo con responsabilidad siguiendo los siguientes consejos:

1. En primer lugar, aunque sea posible empaquetar gran cantidad de datos específicos no significa que siempre se pueda usar. Un statement no está definido por este campo, ni al contrario. Hay que utilizar correctamente todos los campos.
2. En segundo lugar, las extensiones deben tener alguna relación con la parte de declaración a la que pertenecen. Las extensiones del campo contexto deben proporcionar elementos del campo contexto, mientras que las extensiones del campo resultado deberían proporcionar elementos relacionados con algún resultado.
3. Finalmente, la "key" en la extensión de pares "key/Value" debe ser un URI (un verbo o una actividad).

Otros campos

Existen otros campos como "timestamp" o "stored". El primero almacena cuando se produjo la experiencia y el segundo cuando se guardó la experiencia en el LRS ("Statements 101," 2014).

También existen los campos de "authority" y "voided" los cuales guardan quién o qué ha hecho el statement y si la información es válida o no respectivamente.

Por ejemplo:

"timestamp": "2012-07-05T18:30:32.360Z",
"stored": "2012-07-05T18:30:33.540Z",
"authority":

{
"name": "Irene

Instructor",
"mbox": mailto:irene@example.com

}

Datos proporcionados por Tin Can

Existe una lista de URIs guardadas en un registro proporcionado por Tin Can ("Partes de un 'Tin Can Statement,'" 2014). Este registro se ha creado por los siguientes motivos:

No todo el mundo tiene (o quiere mantener) su propio dominio en el que puedan resolver sus URI.

No todo el mundo se quiere comprometer a resolver los identificadores, así como la construcción y mantenimiento de sus herramientas.

Es importante construir sobre lo que ya se ha definido.

Es un lugar centrado en la comunidad donde cada uno puede agregar y referenciar verbos, actividades, tipos de actividades, las extensiones y los adjuntos.

Permite a un grupo que tiene un modelo de datos particular crear un lugar donde otras personas puedan encontrar todos los identificadores asociados con el modelo de datos de ese grupo.

En el caso de que no se encuentre un Uri específico se tiene la posibilidad de contactar con los desarrolladores de Tin Can para incluirlo en su registro.

4.3.5. Ventajas y desventajas

Ventajas

1. Guarda el progreso del juego en cualquier momento sin necesidad de completar el mismo.
2. Posibilidad de realizar un seguimiento en tiempo real y muy detallado de la actividad del alumno.
3. Permite realizar una comparación entre los resultados de todos los alumnos.
4. Es sencillo adaptar un juego JavaScript a éste estándar.

Desventajas

1. No está integrado en la plataforma de edX, por lo que se ejecuta de forma externa.
2. Los resultados no estarán integrados en edX ya que el formato de ejercicio Lon-Capa no permite guardar toda la información, por lo que se deberán visualizar de forma externa.
3. Para hacer los análisis es necesario saber programar.

4.4. LRS

4.4.1. Cómo integrar el LRS con edX

El LRS se puede integrar de dos formas. Por un lado se puede integrar dentro del LMS, o por el contrario se puede alojar en un servidor externo a edX. En esta sección se explicarán las ventajas y los inconvenientes de ambas opciones.

Ventajas

Las ventajas de incluir el LRS dentro del LMS frente a alojarlo en un servidor externo son:

- El LRS tiene la misma disponibilidad que edX, sin tener que depender de un servidor externo. Esto es beneficioso para el profesor ya que tener un servidor externo con la misma disponibilidad que edX puede ser muy costoso.

- El profesor no tiene que configurar su propio LRS, ni su propio visualizador de datos, ya que todo estaría hecho.

Desventajas

Las desventajas de incluir el LRS dentro del LMS frente a alojarlo en un servidor externo son:

- Los resultados que se visualizan no pueden ser personalizados, ya que estarían predefinidos. Ocurriría lo mismo con los verbos. Un profesor no podría incluir el verbo específico que requiere su juego, sino que tendría que utilizar los ya existentes.

- Para integrar el LRS dentro del LMS es necesario que lo haga el administrador.

4.4.2. LRS propio

Según lo explicado en la sección anterior, se ha elegido la opción de crear el LRS fuera del LMS y alojarlo en un servidor externo, ya que es un módulo más independiente y más personalizado. En esta sección se explicará cómo crear un LRS propio y cómo adaptar el juego JavaScript al estándar xAPI y enlazarlo con el LRS.

Cómo crear el LRS

En la actualidad, hay dos formas diferentes de crear un LRS propio. Se puede utilizar la herramienta Scorm Cloud que consiste en registrarse en la página, adaptar el juego al modelo xAPI y conectarlo. No es necesario hacer nada más y además cuenta con una interfaz.("SCORM," 2014)

Por otro lado existe un repositorio en github que permite crear un LRS propio en el ordenador como un servidor local.

Para instalar el LRS se deben seguir las instrucciones que aparecen en el github. ("Github ADL_LRS," 2014)

Una vez instalado, el repositorio cuenta con un juego ejemplo (JsTetris_XAPI) que permite adaptar el nuevo juego al modelo xAPI ("Github Experience API Client Examples," 2014)

Además de este juego ejemplo, ADL también ha desarrollado otros cuyo código es abierto, y que sirven

también como referencia. Todos estos ejemplos se explicarán en la próxima sección (4.4.3. Ejemplos ADL).

Una vez instalado, el aspecto de los statements en el LRS es el siguiente:

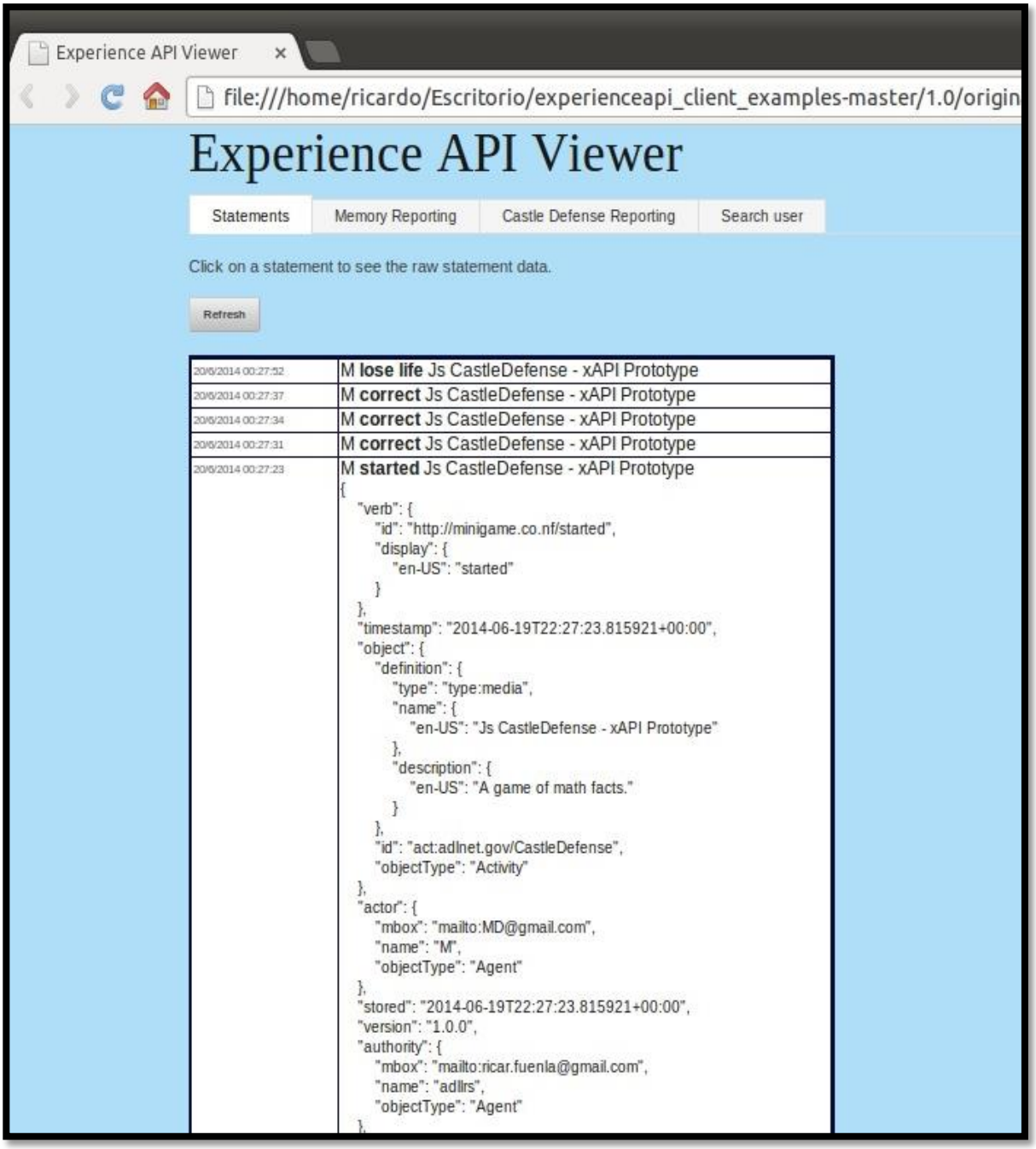


Imagen 4-7: Imagen de los statements del LRS.

Adaptación a xAPI

En primer lugar es necesario crear un JavaScript con los verbos que se van a utilizar y subir ese JavaScript a un servidor externo.

Los verbos han sido creados siguiendo el estándar de xAPI (4.4.1. Tin Can Statement) y son los siguientes:

```
(function(ADL){
   ADL.verbs = {
      "started" : {
        "id" : "http://minigame.co.nf/started",
        "display" : {"en-US" : "started"}
      },
      "finished" : {
         "id" : "http://minigame.co.nf/finished",
         "display" : {"en-US" : "finished"}
      },
      "levelcompleted" : {
         "id" : "http://minigame.co.nf/levelcompleted",
         "display" : {"en-US" : "levelcompleted"}
      },
      "exited" : {
        "id" : "http://minigame.co.nf/correct",
         "display" : {"en-US" : "correct"}
      },
      "failed" : {
         "id" : "http://minigame.co.nf/failed",
         "display" : {"en-US" : "failed"}
      },
      "clicked" : {
         "id" : "http://minigame.co.nf/clicked",
         "display" : {"en-US" : "clicked"}
      },

      "voided" : {
         "id" : "http://minigame.co.nf/voided",
         "display" : {"en-US" : "voided"}
      }
   };
}(window.ADL = window.ADL || {}));
```

Imagen 4-8: Formato de los verbos.

En segundo lugar es necesario adaptar el juego de JavaScript creado al formato xAPI y enlazarlo con el LRS, para ello se puede utilizar la estructura del juego prototipo y adaptarlo. Los pasos a seguir son los siguientes:

1. Hay que añadir a la carpeta donde se encuentra el JavaScript los siguientes archivos que se encuentra en la ruta:

experienceapi_client_examples-master/1.0/original_prototypes/JsTetris_XAPI

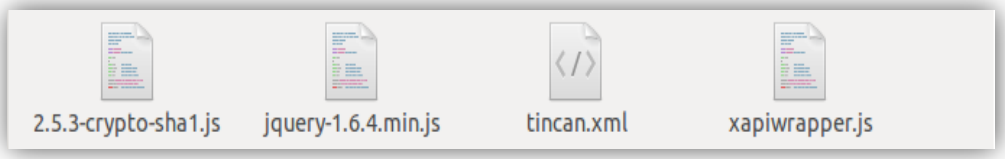


Imagen 4-9: Archivos que hay que insertar en la carpeta del JavaScript.

2. De estos archivos hay que modificar solo el archivo tincan.xml.

tincan.xml

```
<?xml version="1.0" encoding="utf-8" ?>
<tincan xmlns="http://projecttincan.com/tincan.xsd">
        <activities>
                <activity id="act:adlnet.gov/Memory" type="course">
                        <name>xAPI Memory game </name>
                        <description lang="en-US">A game of Memory</description>
                        <launch lang="en-us">index.html</launch>
                </activity>
        </activities>
</tincan>
```

Imagen 4-10: Aspecto del archivo tincan.xml una vez modificado.

3. Hay que modificar el archivo config.js que se encuentra en la ruta experienceapi_client_examples-master/1.0/original_prototypes. En este archivo hay que poner los datos del LRS para que el juego se pueda conectar con él.

tincan.xml | config.js

```
//globals: equal, responseText, statement, ok, deepEqual, QUnit, module, asyncTest, Util, start, golfStatements, console
/*jslint bitwise: true, browser: true, plusplus: true, maxerr: 50, indent: 4 */
function Config() {
        "use strict";
}
Config.endpoint = "http://localhost:8000/xapi/";
Config.user = "adllrs";
Config.password = "141090";
Config.actor = { "mbox":"ricar.fuenla@gmail.com", "name":"Ricar" };
```

Imagen 4-11: Aspecto del archivo config.js una vez modificado.

4. Por último hay que modificar el JavaScript para mandar los statements al LRS.

tincan.xml | config.js | TC_memory.js

```
                'id':GAME_ID,
                                "definition":{
                                        "type":"type:media",
                                        "name":{"en-US":"Js Memory - xAPI Prototype"},
                                        "description":{"en-US":"A game of Memory."}
                }
        };
        var stmt = {
                                "actor":act,
                                "verb":{"id":"http://minigame.co.nf/clicked",
                                        "display":{"en-US":"clicked"}},
                                "object":tcGameObj
        };

        tc_sendStatementWithContext(stmt);

}
function tc_sendStatment_Correct(){
var act={"name":"M","mbox":"mailto:MD@gmail.com"};
                var tcGameObj = {
                'id':GAME_ID,
                                "definition":{
                                        "type":"type:media",
                                        "name":{"en-US":"Js Memory - xAPI Prototype"},
                                        "description":{"en-US":"A game of Memory."}
                }
        };
        var stmt = {
                                "actor":act,
                                "verb":{"id":"http://minigame.co.nf/correct",
                                        "display":{"en-US":"correct"}},
                                "object":tcGameObj
        };

        tc_sendStatementWithContext(stmt);

}
function tc_sendStatment_Failed(){
```

Imagen 4-12: Ejemplo de un método que envía un statement al LRS.

Visualización de los resultados

Dentro de la carpeta del prototipo hay dos formas de visualizar los resultados: el StateViewer y el ReportSample.

El StateViewer muestra todos los statements y un buscador para poder encontrar el statement deseado.

Por otro lado está el ReportSample que además de todos los statements, también muestras las analíticas que se deseen. Para poder decidir las analíticas que se quieren ver hay que modificar el archivo XAPIViewer.js.

Para una generalización del LRS para cualquier juego, y así poder integrarlo con edX más tarde, se ha modificado el ReportSample para que aparezcan las siguientes analíticas:

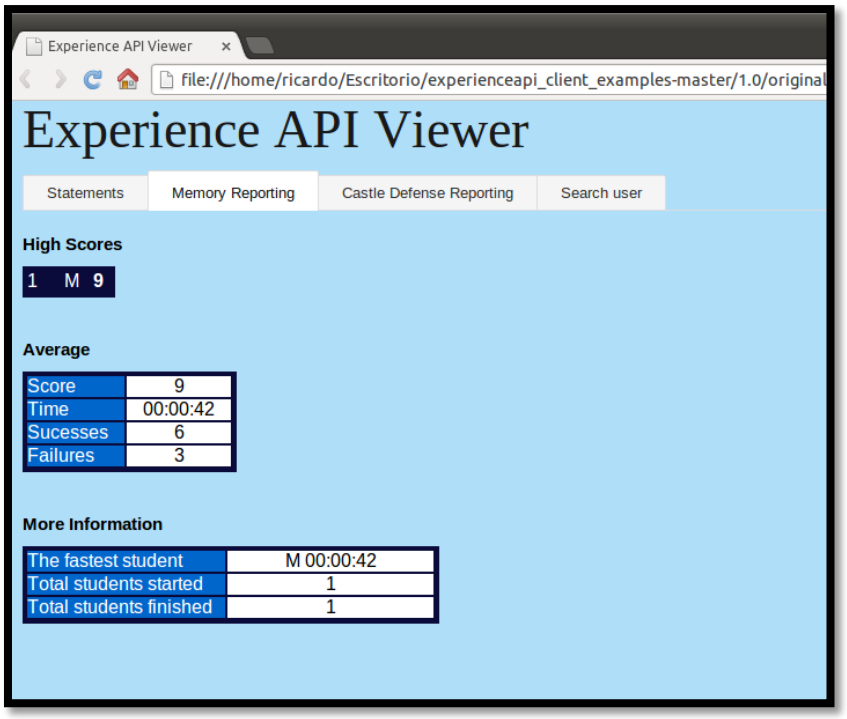


Imagen 4-13: Imagen de la pestaña Memory Reporting del Report Example.

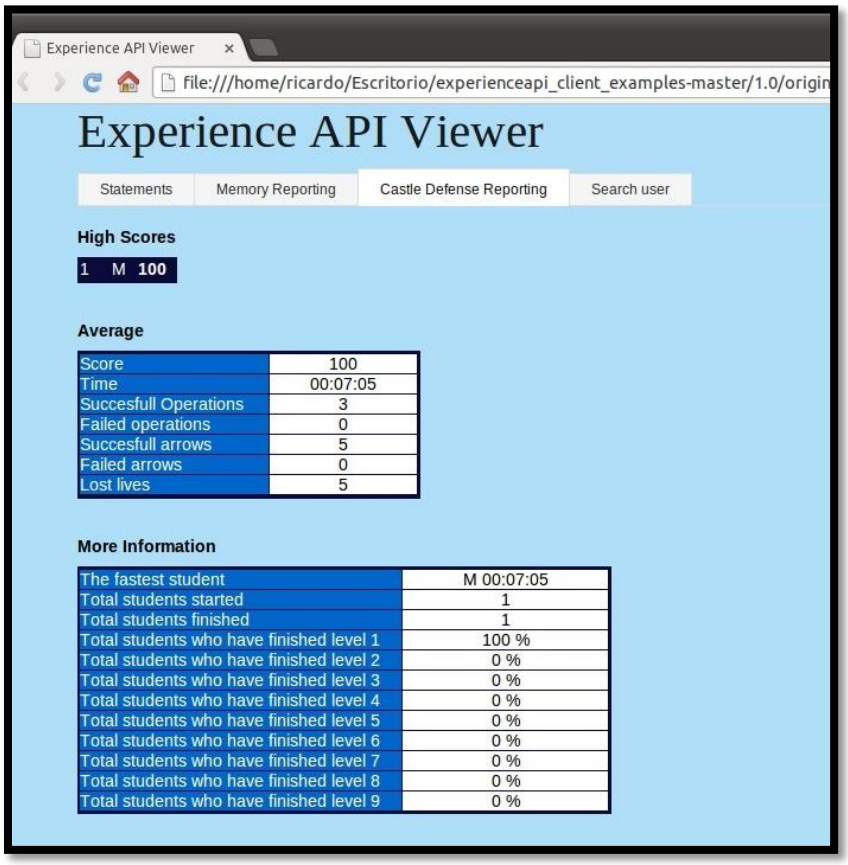


Imagen 4-14: Imagen de la pestaña Castle Defense Reporting del Report Example.

Esta pestaña "Memory Reporting" y en "Castle Defense Reporting" contiene un ranking de puntuaciones, las medias de todos los alumnos, el alumno más rápido y un pequeño seguimiento del progreso de los alumnos.

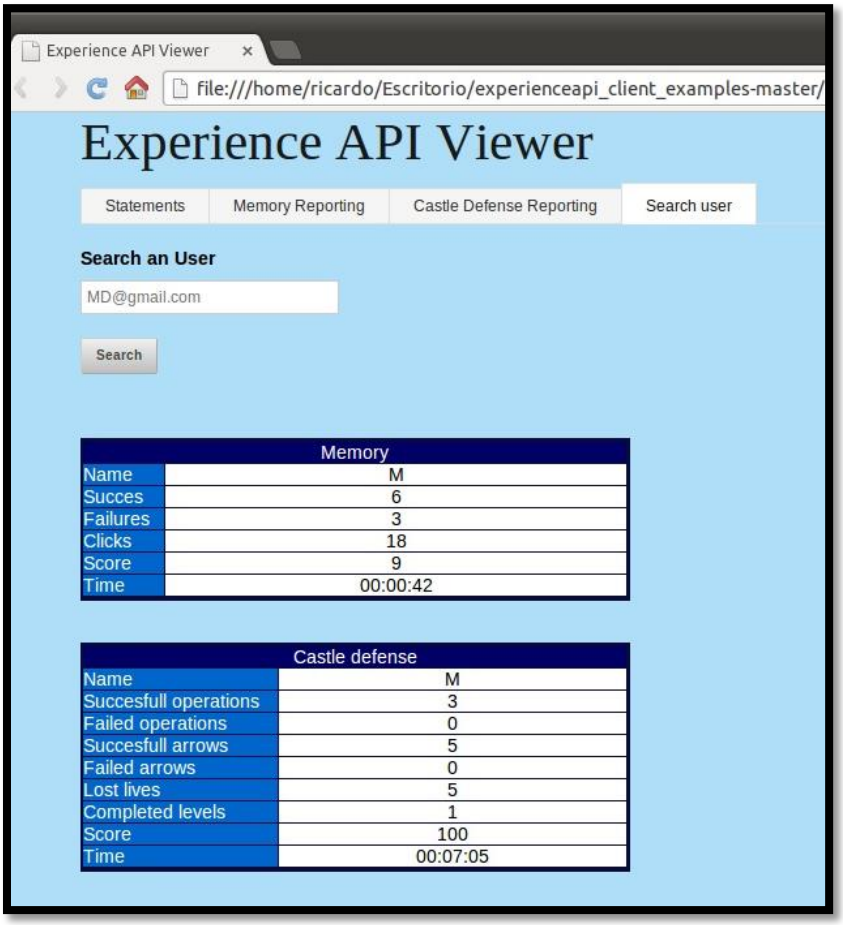


Imagen 4-15: Resultado búsqueda de un estudiante concreto (en este caso se busca por su email).

En la pestaña "Search user" aparece un buscador para poder encontrar a un estudiante y ver una descripción más detallada del mismo.

4.4.3. Ejemplos ADL

Golf Example

El ejemplo consiste en dar una información acerca del golf y después hacer una evaluación a través de un test con respuestas simples.

Cuando el test es completado aparece cada respuesta como correcta o incorrecta, mostrándose la respuesta correcta en el último caso.

Con la información que se recoge en el LRS, aparece cada jugador con su estado del juego (completo o incompleto), y cada pregunta con su respuesta correcta. Además aparece el nº de respuestas totales, fallos y aciertos.

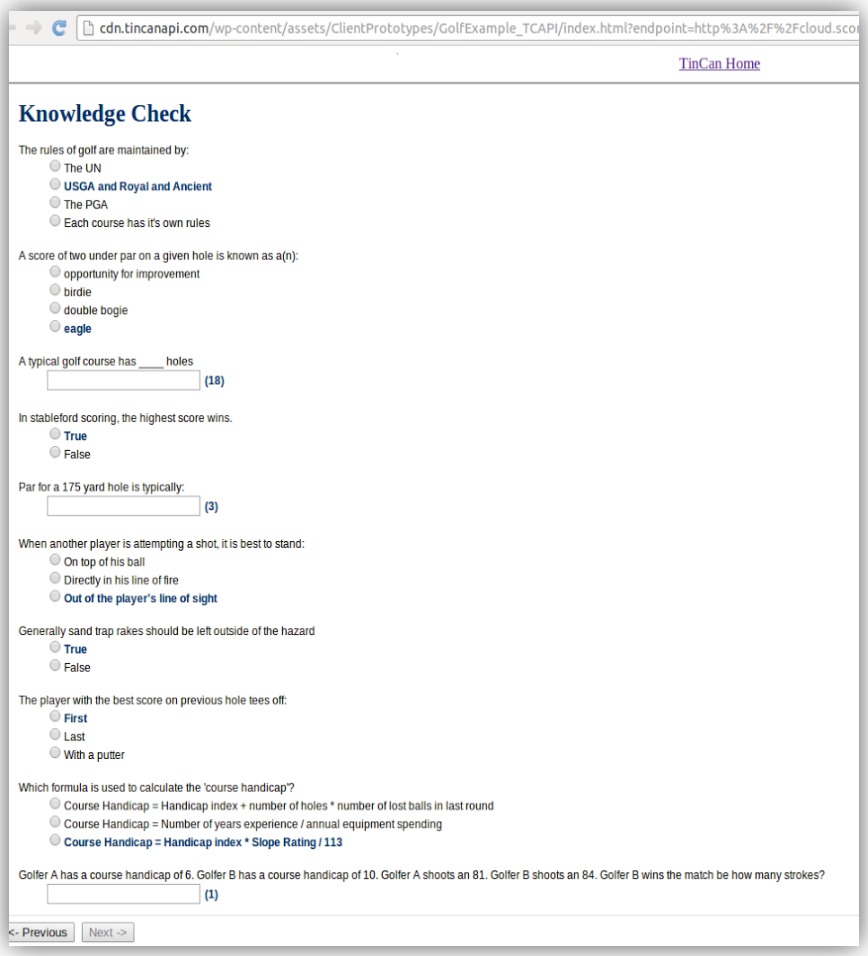


Imagen 4-16: Test de evaluación del Golf Example ("Prototypes Launcher," 2014).

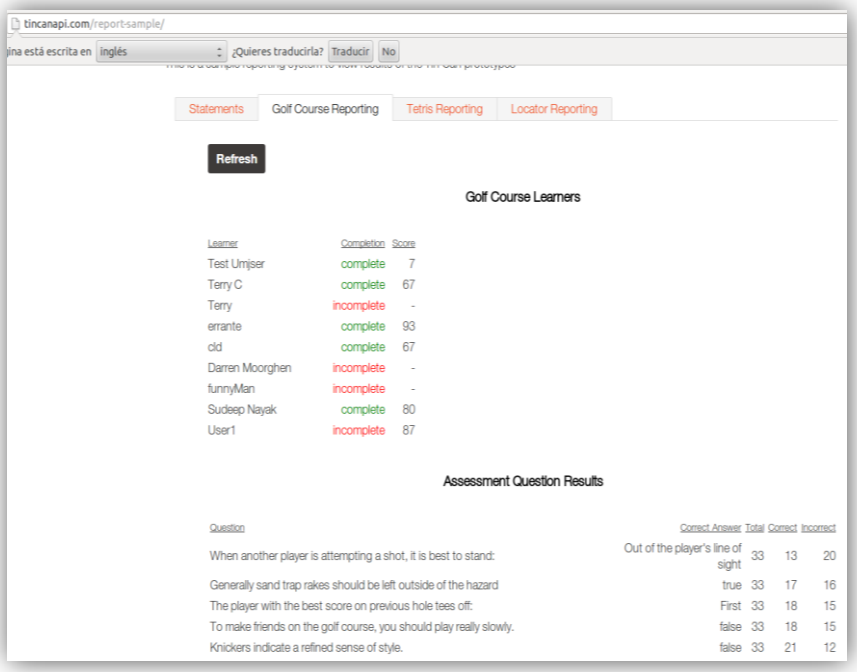


Imagen 4-17: Resultados del LRS del Golf Example ("Report Sample," 2014).

Js Tetris

El ejemplo consiste en una simulación del Tetris en JavaScript. El juego tiene varios niveles y una puntuación. Además también se puede pausar y reanudar.

Con la información que se recoge en el LRS, aparece un ranking de puntuaciones, un gráfico con los juegos necesarios para llegar a la máxima puntuación y todos los usuarios con su resultado.

A continuación se muestra una imagen de cómo es la interfaz del juego y del posterior análisis de los datos:

Imagen 4-18: Juego Js Tetris ("Prototypes Launcher," 2014).

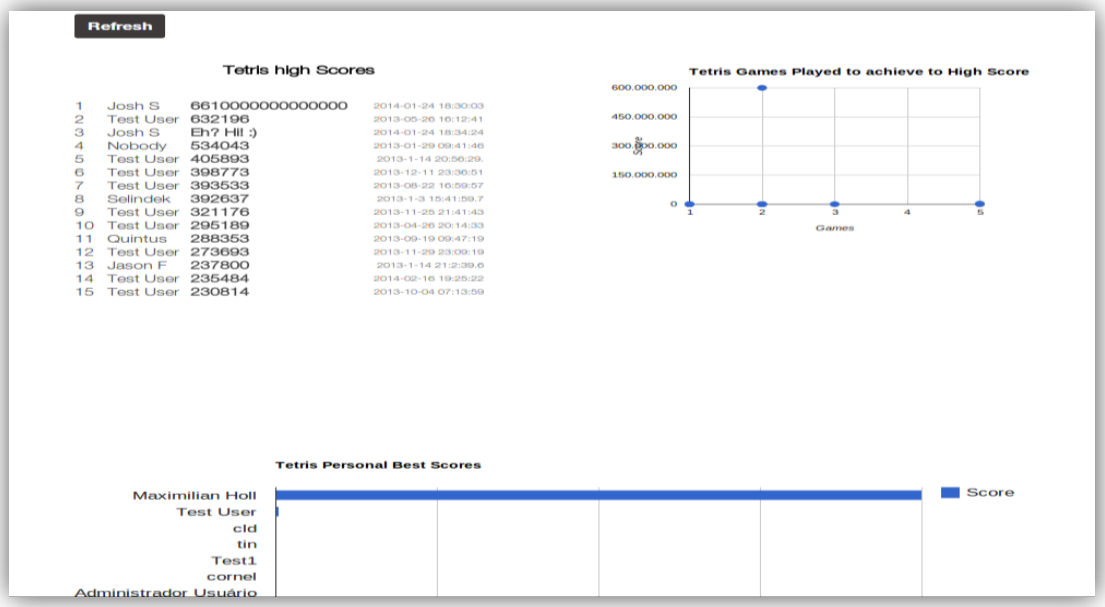


Imagen 4-19: Resultados del LRS del Js Tetris ("Report Sample," 2014).

Locator Example

El ejemplo tiene como objetivo verificar que el usuario ha visitado unas determinadas localizaciones. Cuando se visita una localización, ésta aparece como visitada, y una vez que se han visitado todas, el ejercicio está completo.

Con la información que se recoge en el LRS, aparecen todos los usuarios con su estado (completo o incompleto) y el nº de visitantes que ha tenido cada localización.

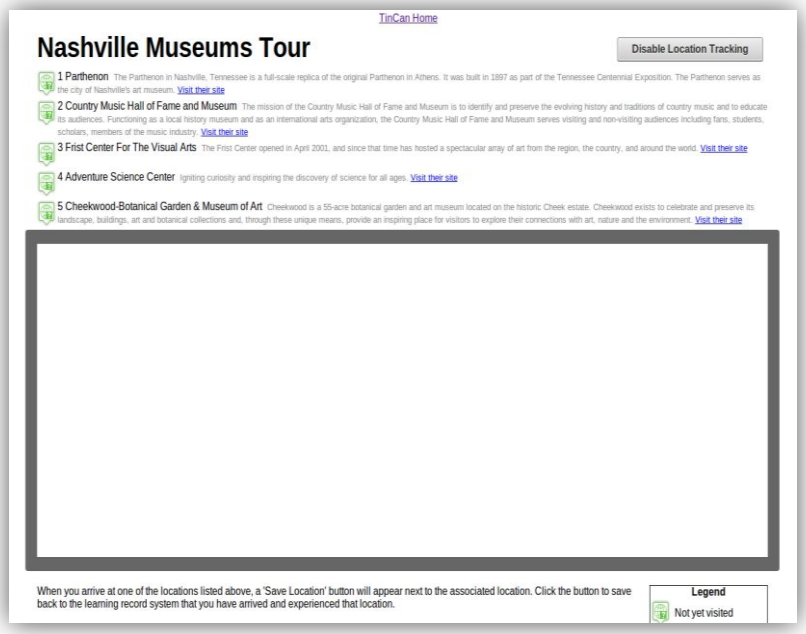
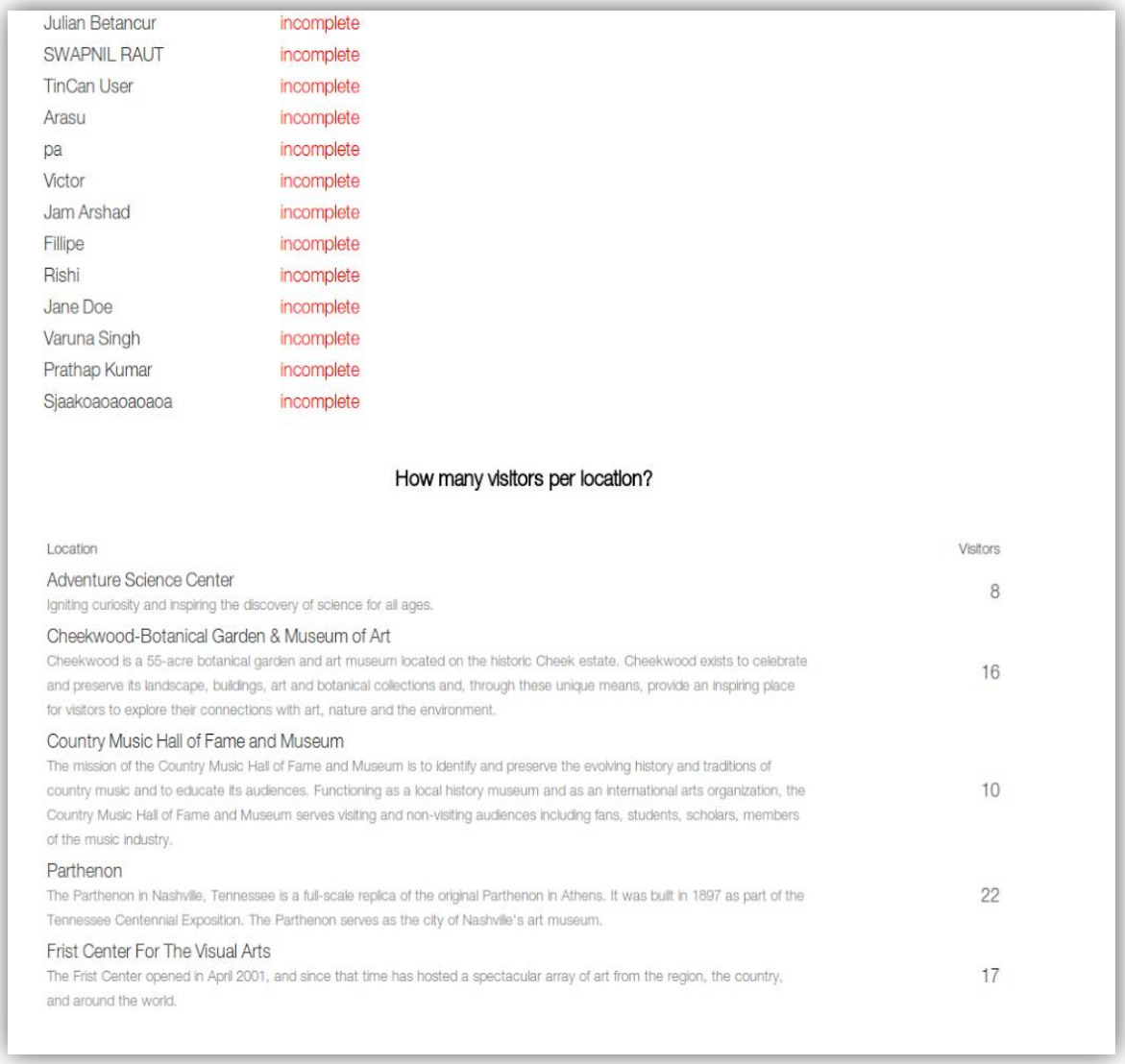


Imagen 4-20: Ejemplo ejecución Locator Example ("Prototypes Launcher," 2014).



Julian Betancur	incomplete
SWAPNIL RAUT	incomplete
TinCan User	incomplete
Arasu	incomplete
pa	incomplete
Victor	incomplete
Jam Arshad	incomplete
Fillipe	incomplete
Rishi	incomplete
Jane Doe	incomplete
Varuna Singh	incomplete
Prathap Kumar	incomplete
Sjaakoaoaoaoaoa	incomplete

How many visitors per location?

Location	Visitors
Adventure Science Center Igniting curiosity and inspiring the discovery of science for all ages.	8
Cheekwood-Botanical Garden & Museum of Art Cheekwood is a 55-acre botanical garden and art museum located on the historic Cheek estate. Cheekwood exists to celebrate and preserve its landscape, buildings, art and botanical collections and, through these unique means, provide an inspiring place for visitors to explore their connections with art, nature and the environment.	16
Country Music Hall of Fame and Museum The mission of the Country Music Hall of Fame and Museum is to identify and preserve the evolving history and traditions of country music and to educate its audiences. Functioning as a local history museum and as an international arts organization, the Country Music Hall of Fame and Museum serves visiting and non-visiting audiences including fans, students, scholars, members of the music industry.	10
Parthenon The Parthenon in Nashville, Tennessee is a full-scale replica of the original Parthenon in Athens. It was built in 1897 as part of the Tennessee Centennial Exposition. The Parthenon serves as the city of Nashville's art museum.	22
Frist Center For The Visual Arts The Frist Center opened in April 2001, and since that time has hosted a spectacular array of art from the region, the country, and around the world.	17

Imagen 4-21: Resultados del LRS del Locator Example ("Report Sample," 2014).

Book Scanner App

El ejemplo es una aplicación Android que permite escanear los códigos de los libros y mandarlos al LRS. No tienen publicado ningún informe de la información del LRS.

Bookmarklet

El ejemplo es un navegador que envía información al LRS sobre tu actividad.

Tampoco tiene publicado informe sobre la información del LRS.

Todos estos ejemplos se pueden descargar en("Download Prototypes - Tin Can API," 2014)

Los verbos utilizados por estos ejemplos son los siguientes:

www.adlnet.gov/wp-content/uploads/2013/10/verbs.js

```
(function(ADL){
   ADL.verbs = {
      "answered" : {
         "id" : "http://adlnet.gov/expapi/verbs/answered",
         "display" : {"en-US" : "answered"}
      },
      "asked" : {
         "id" : "http://adlnet.gov/expapi/verbs/asked",
         "display" : {"en-US" : "asked"}
      },
      "attempted" : {
         "id" : "http://adlnet.gov/expapi/verbs/attempted",
         "display" : {"en-US" : "attempted"}
      },
      "attended" : {
         "id" : "http://adlnet.gov/expapi/verbs/attended",
         "display" : {"en-US" : "attended"}
      },
      "commented" : {
         "id" : "http://adlnet.gov/expapi/verbs/commented",
         "display" : {"en-US" : "commented"}
      },
      "completed" : {
         "id" : "http://adlnet.gov/expapi/verbs/completed",
         "display" : {"en-US" : "completed"}
      },
      "exited" : {
         "id" : "http://adlnet.gov/expapi/verbs/exited",
         "display" : {"en-US" : "exited"}
      },
      "experienced" : {
         "id" : "http://adlnet.gov/expapi/verbs/experienced",
         "display" : {"en-US" : "experienced"}
      },
      "failed" : {
         "id" : "http://adlnet.gov/expapi/verbs/failed",
         "display" : {"en-US" : "failed"}
      },
      "imported" : {
         "id" : "http://adlnet.gov/expapi/verbs/imported",
         "display" : {"en-US" : "imported"}
      },
      "initialized" : {
         "id" : "http://adlnet.gov/expapi/verbs/initialized",
         "display" : {"en-US" : "initialized"}
      },
      "interacted" : {
         "id" : "http://adlnet.gov/expapi/verbs/interacted",
         "display" : {"en-US" : "interacted"}
      },
      "launched" : {
         "id" : "http://adlnet.gov/expapi/verbs/launched",
         "display" : {"en-US" : "launched"}
      },
      "mastered" : {
         "id" : "http://adlnet.gov/expapi/verbs/mastered",
         "display" : {"en-US" : "mastered"}
      },
      "passed" : {
         "id" : "http://adlnet.gov/expapi/verbs/passed",
         "display" : {"en-US" : "passed"}
      },
      "preferred" : {
         "id" : "http://adlnet.gov/expapi/verbs/preferred",
         "display" : {"en-US" : "preferred"}
      },
      "progressed" : {
         "id" : "http://adlnet.gov/expapi/verbs/progressed",
         "display" : {"en-US" : "progressed"}
      },
      "registered" : {
         "id" : "http://adlnet.gov/expapi/verbs/registered",
         "display" : {"en-US" : "registered"}
      },
      "responded" : {
         "id" : "http://adlnet.gov/expapi/verbs/responded",
         "display" : {"en-US" : "responded"}
      },
      "resumed" : {
         "id" : "http://adlnet.gov/expapi/verbs/resumed",
         "display" : {"en-US" : "resumed"}
      },
      "scored" : {
         "id" : "http://adlnet.gov/expapi/verbs/scored",
         "display" : {"en-US" : "scored"}
      },
      "shared" : {
         "id" : "http://adlnet.gov/expapi/verbs/shared",
         "display" : {"en-US" : "shared"}
      },
      "suspended" : {
         "id" : "http://adlnet.gov/expapi/verbs/suspended",
         "display" : {"en-US" : "suspended"}
      },
      "terminated" : {
         "id" : "http://adlnet.gov/expapi/verbs/terminated",
         "display" : {"en-US" : "terminated"}
      },
      "voided" : {
         "id" : "http://adlnet.gov/expapi/verbs/voided",
         "display" : {"en-US" : "voided"}
      }
   };
}(window.ADL = window.ADL || {}));
```

Imagen 4-22: Verbos utilizados en los prototipos.

5. Integración de un nuevo tipo de ejercicio en edX

En esta sección se explicarán las diferentes formas que hay de insertar un nuevo tipo de ejercicio en edX. Según lo investigado existen tres formas de incorporar un nuevo tipo de ejercicio en la plataforma, y son las siguientes:

Integración como ejercicio Lon-Capa

Es posible añadir problemas como ejercicios simples o como ejercicios avanzados. Este tipo de problema hereda de la Clase LonCapaResponse por lo que se trata como un LonCapaProblem (4.1.1. Investigación de Lon-Capa).

Para añadir un nuevo tipo de problema basta crear un archivo .yaml con la estructura principal del problema y los métodos necesarios para su ejecución. Estos problemas se basan en que el autor introduce un xml a través del editor siempre siguiendo una plantilla definida en el código previamente. Después ese xml es tratado por los diferentes métodos para recuperar la información del problema.

En cuanto a la evaluación, se utiliza el método getScore(), que es definido en todos los problemas de este tipo, y que genera un objeto de tipo CorrectMap donde se almacena el número de puntos, si la respuesta es correcta o no y las respuestas enviadas por el estudiante.

La explicación más detallada de cómo añadir un ejercicio de tipo Lon-Capa, se verá más adelante en la sección (5.1. Incorporación de JavaScript usando Lon-Capa).

Integración como un xModule

Otra forma de añadir un nuevo tipo de ejercicio es añadiéndolo como un módulo independiente, es decir, como un xModule. Este tipo de ejercicios no vienen añadidos por defecto cuando se crea un curso, sino que hay que añadirlo en las opciones avanzadas, y después aparece como un módulo avanzado en la herramienta "studio".

La explicación más detallada de cómo añadir un nuevo ejercicio como xModule y como incorporarlo como ejercicio avanzado a la herramienta studio se verá más adelante en la sección (5.2. Incorporación de un juego JavaScript como xModule)

Integración como un módulo LTI:

Por último, otra forma de incorporar nuevos ejercicios es a través del módulo LTI. El concepto principal de LTI es ejecutar un programa a través de un link externo a edX. Su funcionamiento se basa en que en la plataforma de edX aparece un link a una página externa con el programa, donde será ejecutado el mismo. Una vez que se termina, se devuelve a edX toda la información generada en el programa para calcular la puntuación final del usuario. Se puede ver más detalladamente en la siguiente sección (4.2. Learning Tools Interoperability).

5.1. Incorporación de módulos JavaScript usando Lon-Capa

En esta sección se explicarán los pasos necesarios para poder integrar un JavaScript como un ejercicio Lon-Capa en edX. El nuevo ejercicio se llama "Minigame" y se trata de un ejercicio avanzado de Lon-Capa.

5.1.1. MiniGameResponse

Este tipo de ejercicio utiliza un nuevo tipo de respuesta llamada "MiniGameResponse". Concretamente en este tipo de respuesta se cumple lo siguiente:

Solo admite objetos de tipo JSInput (5.1.3. Objeto JSInput).

Es necesario tener una función de evaluación que se guardará en el atributo "cfn" y que será especificada por el autor en el editor del problema.

```
<minigameresponse cfn="check">
  <jsinput gradefn="CatCha.getGrade"
    get_statefn="CatCha.getState"
    set_statefn="CatCha.setState"
    width="400"
    height="400"
    html_file="http://minigame.co.nf/catcha/catcha.html"
    sop="false"/>
</minigameresponse>
```

Imagen 5-1: Ejemplo de una minigame response.

5.1.2. Función de evaluación de MiniGameResponse

La función de evaluación tiene como parámetros la respuesta esperada y las respuestas dadas por los usuarios.

Debe devolver un diccionario de la forma:

'imagenes': [

{'ok': boolean},

{'ok': boolean}],

'mensajes': [

{ 'msg': String],

'tiempo': {'tiemp': tiempo}]

1. Puede haber tantos campos 'ok' o 'msg' como sean necesario. Además se añade un campo 'tiempo' donde se especificará el tiempo utilizado para realizar el ejercicio. Con esta respuesta se realiza la siguiente evaluación:

2. Por cada campo 'ok' que sea correcto se sumará un punto a la puntuación final, siendo la máxima puntuación el número de campos 'ok'.

3. El ejercicio será correcto si se ha llegado a la puntuación máxima, en caso contrario será evaluado como incorrecto.

4. Los mensajes de los campos 'msg' se utilizarán para crear un informe que será mostrado una vez se haya pulsado el botón "check" y se haya realizado la evaluación. Estos mensajes serán guardados en la base de datos en el objeto CorrectMap que se crea cuando se evalúa un ejercicio de Lon-Capa.

5. Por último se mostrará la puntuación obtenida, la puntuación máxima posible y el tiempo utilizado para realizar el ejercicio.

```
<script type="loncapa/python">
import json
def check(e, ans):

  par = json.loads(ans)
  # We can use either the value of the answer key to grade
  answer = json.loads(par["answer"])
  state = json.loads(par["state"])
  mensaje=""
  for k in range(len(answer["informe"])):
       mensaje+= answer["informe"][k] +" | "



  return {'imagenes': [
                  { 'ok': answer["img1"]},
                  { 'ok': not answer["img2"]},
                  { 'ok': not answer["img3"]},
                  { 'ok': not answer["img4"]},
                  { 'ok': answer["img5"]},
                  { 'ok': not answer["img6"]},],
              'mensajes': [
                  { 'msg': mensaje}], }

</script>
```

Imagen 5-2: Ejemplo función de evaluación.

5.1.3. Objecto JSInput

Es el único objeto que puede contener una MiniGameResponse y se utiliza para añadir cualquier JavaScript, siempre que éste cumpla unas normas:

- Debe estar en un servidor externo.

- Debe tener definido un método que se ejecutará cuando el usuario pulse el botón "Check" y que debe devolver toda la información que necesita la función de evaluación para dar la respuesta con el formato correcto. Este método será especificado en el atributo "gradefn".

- Además puede tener otros métodos para recuperar el estado "get_statefn" o para modificar el estado "set_statefn", pero estos son opcionales.

Los atributos que deben ser especificados en un JSInput son los siguientes:

- **gradefn:** Contiene el método del JavaScript que permite la evaluación.
- **get_statefn:** Contiene el método del JavaScript que permite recuperar el estado.(opcional)
- **set_statefn:** Contiene el método del JavaScript que permite cambiar el estado.(opcional)
- **width y height:** Indican el tamaño del frame donde será ejecutado el JavaScript.
- **html_file:** Contiene la dirección donde está el HTML del JavaScript que se va a ejecutar.
- **sop:** Que es un booleano que debe estar a "false" si el JavaScript se ejecuta desde un servidor distinto al local o a "true" en el caso contrario.

```
<jsinput gradefn="CatCha.getGrade"
  get_statefn="CatCha.getState"
  set_statefn="CatCha.setState"
  width="400"
  height="400"
  html_file="http://minigame.co.nf/catcha/catcha.html"
  sop="false"/>
```

Imagen 5-3: Ejemplo objeto JSInput.

Ejemplo ejercicio Minigame:

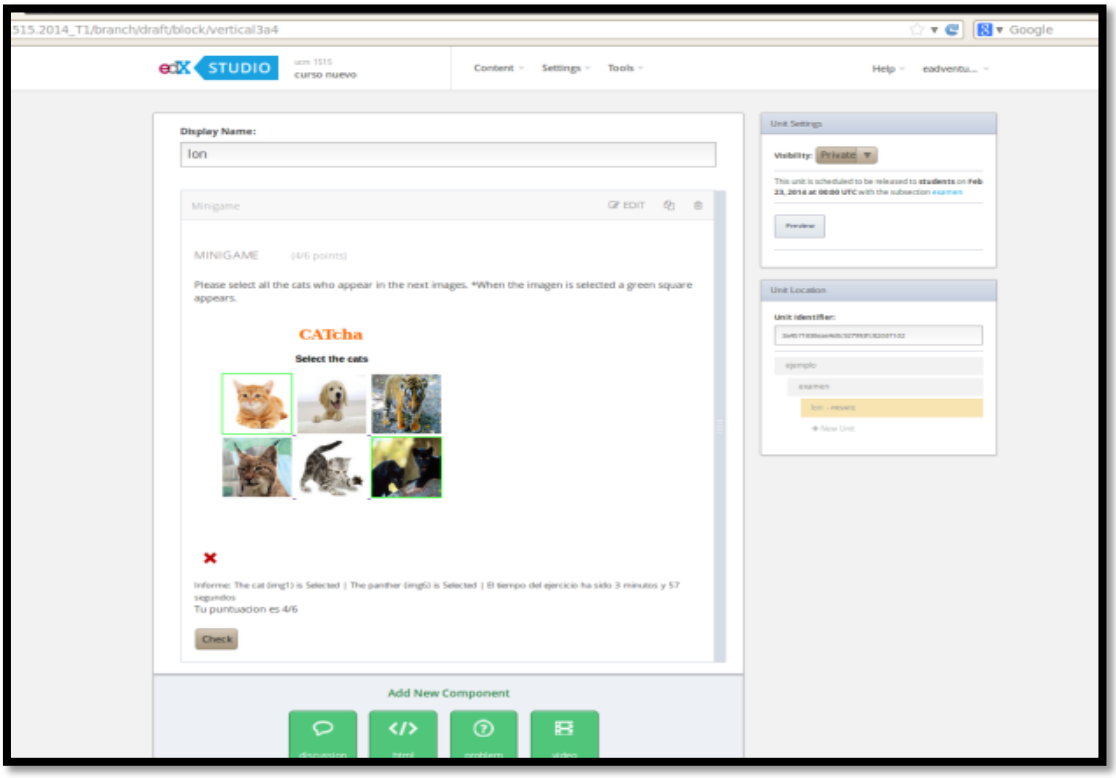


Imagen 5-4: Ejemplo de un ejercicio Minigame.

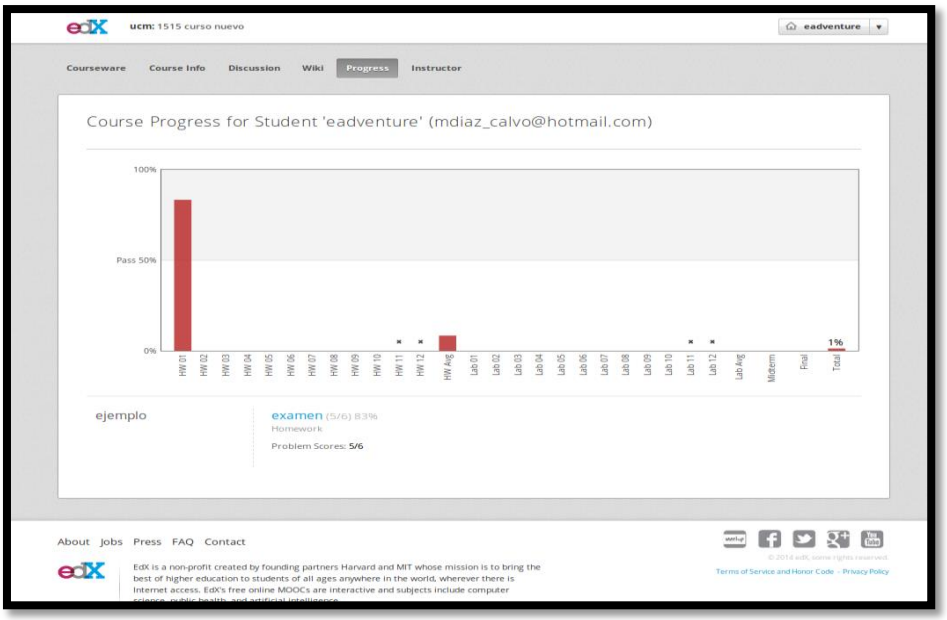


Imagen 5-5: Progreso del alumno con la puntuación del nuevo ejercicio.

5.2. Incorporación de un juego JavaScript como xModule

En esta sección se explicarán los pasos necesarios para integrar un juego JavaScript como un xModule, creando así un nuevo tipo de ejercicio avanzado.

En primer lugar se debe crear la carpeta donde estarán los módulos JavaScript necesarios para la simulación del juego. Después hay que configurar el HTML que el módulo va a referenciar al ejecutarse, y para terminar se debe crear el módulo con las inclusiones de los módulos JavaScript y la referencia a su HTML correspondiente.

Además, para que la simulación del juego esté disponible al crear el curso hay que incorporarlo, para lo que se usará la herramienta CMS ("studio") que proporciona edX.

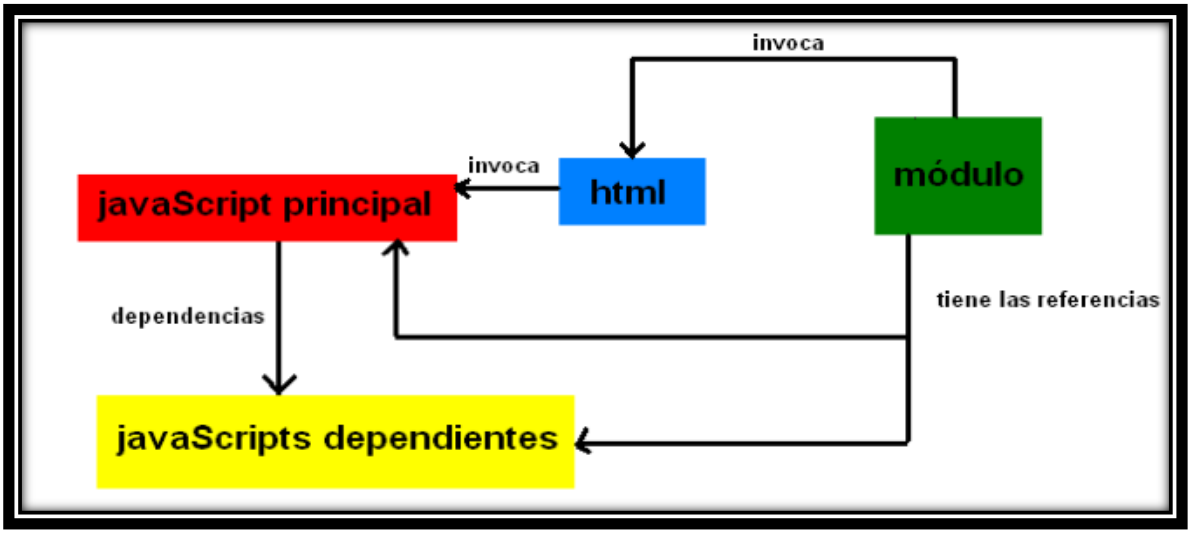


Imagen 5-6: Esquema de dependencias, referencias e invocaciones de la simulación de juego.

5.2.1. Creación de una carpeta con los módulos JavaScript

Se debe crear una carpeta con los módulos JavaScript necesarios para la simulación del juego. Siguiendo el protocolo de edX-platform, se guardarán en la ruta donde se encuentran todos los módulos JavaScript de los distintos problemas avanzados que edX-platform permite añadir por defecto:

edX-platform/common/lib/xmodule/xmodule/js/src/**minigame**

Donde **minigame** es la carpeta nueva que se ha creado anteriormente, y que contiene los módulos JavaScript de la nueva simulación que se quiere incorporar.

5.2.2. Creación del HTML

Los módulos en edX hacen referencia a un HTML, que es el que se instancia en primer lugar. edX permite añadir elementos directamente a través del HTML, pero como la simulación de juego está hecha completamente en JavaScript, únicamente habrá que referenciar al JavaScript principal de la simulación del juego desde dicho HTML:

```
<section
    id="word_cloud_${element_id}"
    class="${element_class} minigame"
    data-ajax-url="${ajax_url}"
>
</section>
```

Imagen 5-7: Código HTML invocando al JavaScript (minigame) de la simulación del juego.

La ruta que edX-platform usa para guardar los HTML de los problemas avanzados es: edX-platform/lms/templates. En esa ruta es donde hay que añadir el nuevo HTML creado.

5.2.3. Creación del módulo

Hay que crear un nuevo archivo Python en edX llamado **minigame.py**. Este fichero es el que crea el nuevo módulo y siguiendo el protocolo de edX, hay que guardarlo en la ruta que utiliza edX-platform para guardar los módulos de los ejercicios avanzados: edX-platform/common/lib/xmodule/xmodule/**minigame.py**

Éste módulo necesita las rutas y nombres del archivo HTML y de los módulos JavaScript que se añadieron anteriormente.

En primer lugar, hay que crear los elementos que la simulación del juego va a necesitar guardar en la base de datos, tal y como se puede ver en la imagen 12-2, en el apartado #Fields for descriptor.

También es necesario indicar en qué tabla de la base de datos van a ser guardados los resultados. Como se puede observar en la siguiente imagen, el atributo scope de cada elemento indica la tabla donde se guardará el mismo.

```python
class MiniGameFields(object):
    """XFields for Minigame."""
    display_name = String(
        display_name="Display Name",
        help="Display name for this module",
        scope=Scope.settings,
        default="minigame"
    )


    # Fields for descriptor.
    submitted = Boolean(
        help="submitted",
        scope=Scope.user_state,
        default=False
    )
    student_progress = List(
        help="Student progress.",
        scope=Scope.user_state,
        default=[]
    )
```

Imagen 5-8: Código con el que se guardan elementos en la base de datos.

A continuación se añaden las rutas de todos los módulos JavaScript que se han creado y añadido en el paso "Creación de una carpeta con los módulos JavaScript". Esto hay que hacerlo para que el módulo al ser ejecutado llame en primer lugar al HTML, que a su vez referencia al nombre del JavaScript principal de la simulación del juego creado (en este caso minigame).

A continuación, se añade la información que será guardada en el JSON. En la siguiente imagen se puede observar los campos que el JSON utiliza para una simulación de juego concreta, pero se pueden modificar los campos en función del juego que se haya creado en JavaScript.

```python
class MiniGameModule(MiniGameFields, XModule):
    """Minigame Xmodule"""
    js = {
        'coffee': [resource_string(__name__, 'js/src/javascript_loader.coffee')],
        'js': [resource_string(__name__, 'js/src/minigame/logme.js'),
        resource_string(__name__, 'js/src/minigame/minigame.js'),
        resource_string(__name__, 'js/src/minigame/minigame_main.js')]
    }
    css = {'scss': [resource_string(__name__, 'css/minigame/display.scss')]}
    js_module_name = "MiniGame"

    def get_state(self):
        """Return success json answer for client."""
        if self.submitted:
            return json.dumps({
                'levels': [
                    {
                        'name' : 'first',
                        'max' : 100,
                        'progress' : '0',
                        'finished' : False
                    },
                    {
                        'name' : 'second',
                        'max' : 80,
                        'progress' : '0',
                        'finished' : False
                    },
                    {
                        'name' : 'third',
                        'max' : 40,
                        'progress' : '0',
                        'finished' : False
                    }
                ],
                'status': 'success'
            })
```

Imagen 5-9: Código de un xModule edX.

Después se añade el nombre del HTML

```
def get_html(self):
    """Template rendering."""
    context = {
        'element_id': self.location.html_id(),
        'element_class': self.location.category,
        'ajax_url': self.system.ajax_url,
        'submitted': self.submitted
    }
    self.content = self.system.render_template('minigame.html', context)
    return self.content
```

Imagen 5-10: Código que invoca al HTML de la simulación del juego.

Para terminar, se creará el módulo.

```
class MiniGameDescriptor(MiniGameFields, MetadataOnlyEditingDescriptor, EmptyDataRawDescriptor):
    """Descriptor Minigame Xmodule."""
    module_class = MiniGameModule
    template_dir_name = 'minigame'
```

Imagen 5-11: Código de la creación del módulo.

Además hay que añadir el nuevo módulo al archivo setup.py que es el archivo donde están todos los módulos definidos. Este archivo se encuentra en la ruta:

edX-platform/common/lib/xmodule/**setup.py**

En la siguiente imagen se puede ver un fragmento del código del archivo setup.py ya mencionado. Se puede observar cómo se ha tenido que añadir manualmente el módulo "minigame" en la última línea del código.

```
from setuptools import setup, find_packages

XMODULES = [
    "abtest = xmodule.abtest_module:ABTestDescriptor",
    "book = xmodule.backcompat_module:TranslateCustomTagDescriptor",
    "chapter = xmodule.seq_module:SequenceDescriptor",
    "combinedopenended = xmodule.combined_open_ended_module:CombinedOpenEndedDescriptor",
    "conditional = xmodule.conditional_module:ConditionalDescriptor",
    "course = xmodule.course_module:CourseDescriptor",
     ...

    "crowdsource_hinter = xmodule.crowdsource_hinter:CrowdsourceHinterDescriptor",
    "lti = xmodule.lti_module:LTIDescriptor",
    "minigame = xmodule.minigame:MiniGameDescriptor",
]
```

Imagen 5-12: Código del setup.py, con la última línea añadida para la simulación del juego.

Por último, antes de incorporarlo al curso, hay que hacer un rake del CMS para que se añada el nuevo módulo a todos los archivos y se referencie correctamente.

5.2.4. Incorporación del módulo a un curso edX

Cuando se está creando un curso en el CMS, en advanced settings hay que añadir al conjunto el nombre del módulo (en este caso minigame).

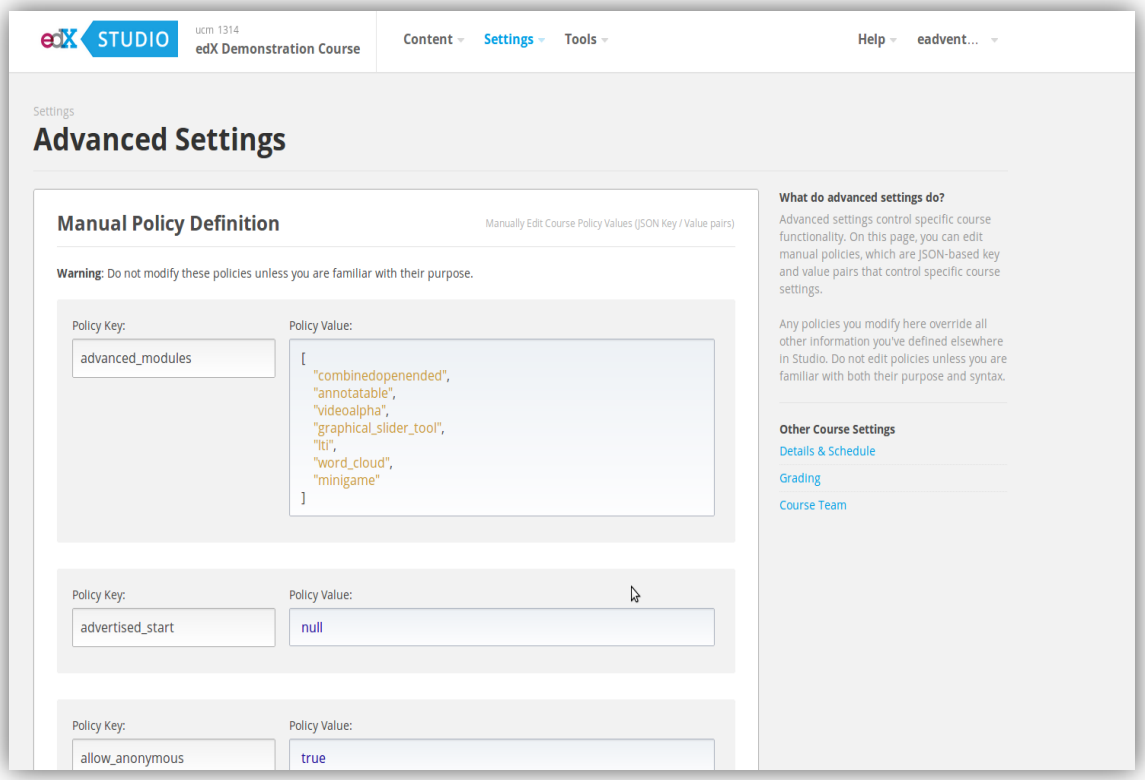


Imagen 5-13: Panel Advanced Settings al crear un curso dentro del CMS.

Además hay que añadirlo también al archivo component.py que se encuentra en la ruta: edX-platform/cms/djangoapp/contentstore/views/**component.py**

A continuación se muestra una imagen con el código de dicho fichero, que está escrito en Python.

```
OPEN_ENDED_COMPONENT_TYPES = ["combinedopenended", "peergrading"]
NOTE_COMPONENT_TYPES = ['notes']
ADVANCED_COMPONENT_TYPES = [
    'annotatable',
    'word_cloud',
    'graphical_slider_tool',
    'lti',
    'triviado',
    'minigame',

] + OPEN_ENDED_COMPONENT_TYPES + NOTE_COMPONENT_TYPES
ADVANCED_COMPONENT_CATEGORY = 'advanced'
ADVANCED_COMPONENT_POLICY_KEY = 'advanced_modules'
```

Imagen 5-14: Código de component.py

Después de esto, al crear una nueva unidad, aparecerá el módulo que se ha creado como un módulo avanzado.

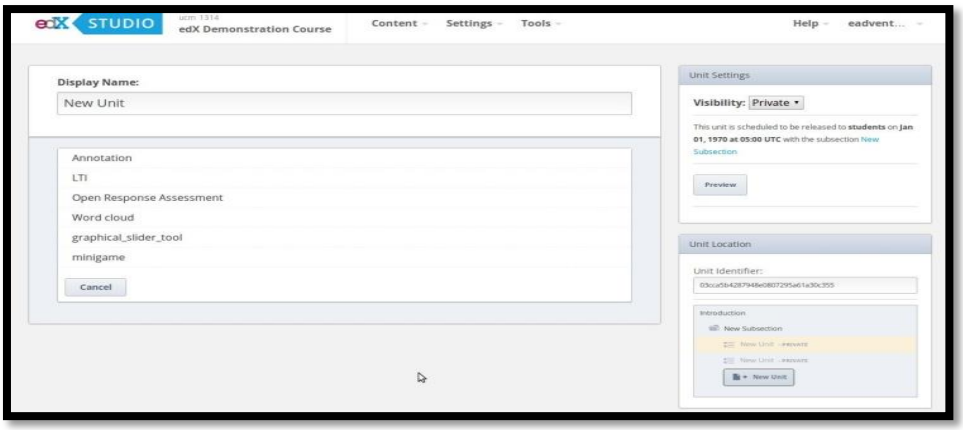


Imagen 5-15: Insertando un módulo avanzado desde el CMS.

Al elegir dicha opción se inserta el módulo automáticamente:

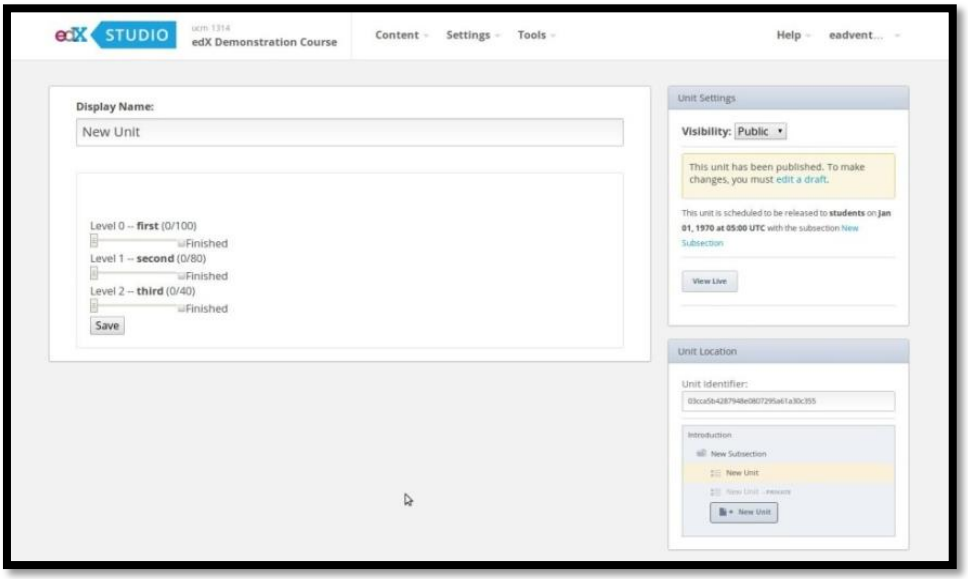


Imagen 5-16: Simulación del juego en un curso desde el CMS.

5.3. Juegos incorporados

Catcha

Este ejercicio simple consiste en un captcha en el que aparecen varias imágenes donde hay que identificar a los gatos. Este ejemplo sólo está incorporado en Lon-Capa.

El método de evaluación que se ha elegido es una calificación de 0 a 6 puntos ya que el ejercicio consiste en seleccionar de entre seis imágenes las que son correctas.

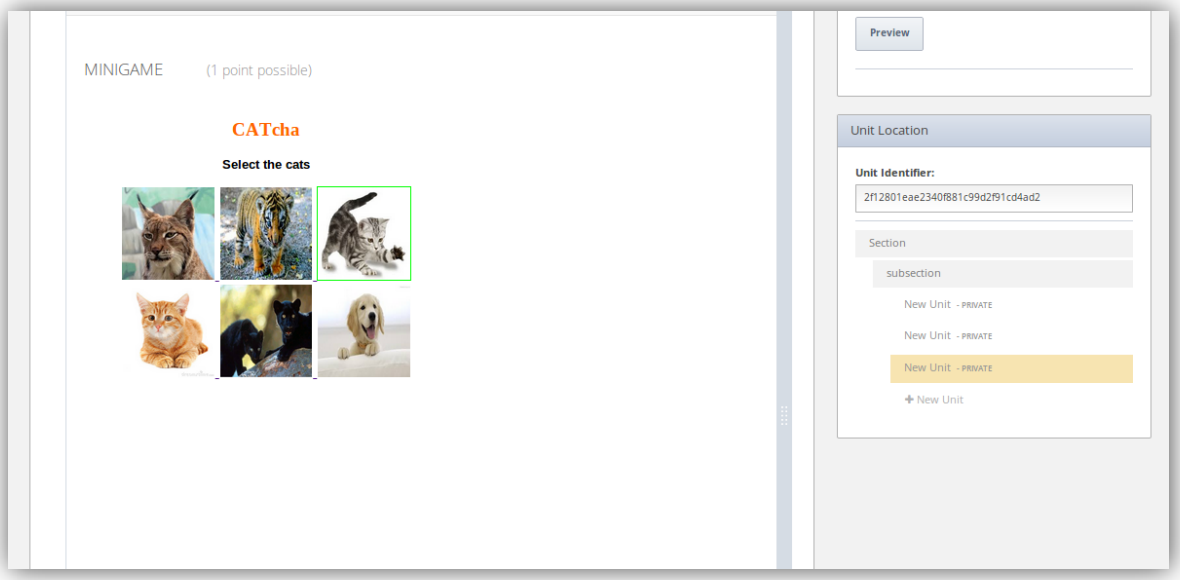


Imagen 5-17: Catcha en edX.

Se ha decidido guardar el tiempo de juego, la iteracciones del usuario para saber qué pasos han seguido para completar la prueba y el estado de las imágenes para saber si están seleccionadas las imágenes correctas.

Esa información se guarda en el JavaScript y sólo cuando se hace click en el botón "check", se manda a edX. Se trata de un ejercico de tipo Lon-Capa por lo que el resultado se guarda en un objeto de tipo CorrectMap. Como se ha visto en la sección (4.1.1 Investigación de Lon-Capa), los únicos datos que se pueden guardar son el número de puntos conseguidos (npoints) y un String (msg) donde se guardan las acciones y el tiempo utilizado a modo de informe.

Para este ejercicio, la información que permite guardar Lon-Capa podría ser suficiente ya que es un ejercicio simple. El mayor problema es que si se cierra el juego sin haber pulsado el botón "check" todo el progreso se perdería y no quedaría reflejada ninguna de las acciones del usuario.

Memory

Se trata de un juego educativo en el que el usuario debe emparejar las cartas de los elementos químicos con sus respectivas fórmulas. Actualmente sólo existe un nivel, pero este tipo de juego está pensado para superar niveles.

Este juego ha sido integrado tanto en Lon-Capa como en xAPI, siendo el análisis obtenido con el segundo mucho más detallado.

Los resultados obtenidos como ejercicio Lon-Capa sólo permitían guardar la puntuación obtenida en total y un pequeño informe de las acciones realizadas hasta llegar a la solución. Con este tipo de implementación, se perdía mucha información como la puntuación de cada nivel o la comparación del usuario respecto a los otros usuarios que han completado el juego.

Por otro lado, los resultados obtenidos con xAPI son mucho mayores ya que la información obtenida por los statements es más amplia. En el LRS se guardan todos los statements donde se encuentran los verbos, el usuario que ha realizado la acción y en algunos casos un contexto o una puntuación si la acción lo requiere. Posteriormente con todos estos datos, se podrá realizar un seguimiento individual del estudiante

analizando su puntuación, el tiempo que ha tardado, el número de aciertos, etc. También se podrá realizar un análisis colectivo de los estudiantes, obteniendo datos globales como pueden ser el número medio de aciertos, de fallos, de tiempo, un ranking de los mejores alumnos o el alumno más rápido en completarlo correctamente. Este último análisis es lo que diferencia principalmente Lon-Capa de xAPI, ya que en el primero no se pueden comparar los usuarios, sino que sólo se pueden obtener resultados individuales.

Un ejemplo del resultado detallado de la integración del juego en xAPI se puede observar en la imagen (Imagen 4-13).

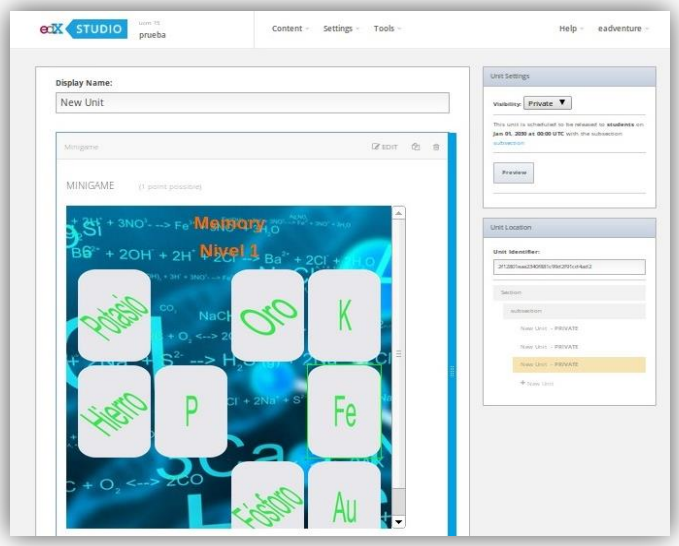


Imagen 5-18: Memory en edX.

Castle defense

Este juego se ha obtenido de ("Castle Defense | OpenGameArt.org," 2014) y consiste en realizar operaciones matemáticas para poder eliminar adversarios. Es un juego por niveles, en el que se va acumulando la puntuación.

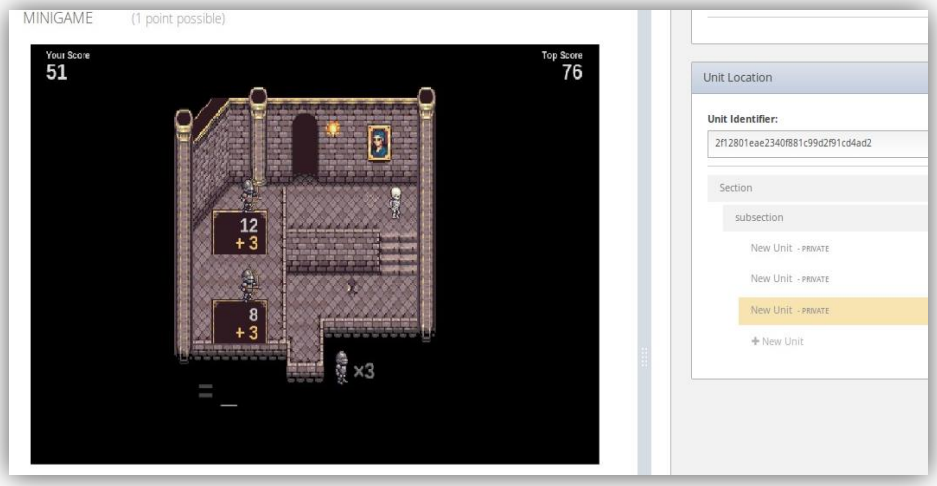


Imagen 5-19: Castle defense en edX.

Este juego se ha integrado en edX como ejercicio Lon-Capa y se ha decidido no utilizar la evaluación proporcionada este método ya que la información que se puede guardar es muy escasa y para este tipo de juegos en los que hay niveles y varios caminos para completarlo, se necesita mucho más detalle. Por lo tanto la evaluación se realizará únicamente utilizando xAPI.

Concretamente en este juego, se ha utilizado el contexto en los verbos para diferenciar los aciertos y los fallos de las operaciones respecto a los de la flecha disparada. Además en cada nivel se envía la puntuación por lo que se puede saber qué pruebas ha realizado con mayor éxito cada alumno. También es posible conocer el progreso de los alumnos en el juego, sabiendo el número de alumnos que han empezado, los que han terminado y los niveles que han superado.

Un ejemplo del resultado detallado de la integración del juego en xAPI se puede observar en la imagen (Imagen 4-14).

Ejemplo de la integración

Para la integración de todos estos juegos se ha utilizado el mismo tipo de ejercicio de Lon-Capa explicado en la sección (5.1 Incorporación de módulos JavaScript usando Lon-Capa).

En las imágenes siguientes se puede observar cuál es el código que es necesario modificar desde la pestaña "edit" para integrar los juegos.

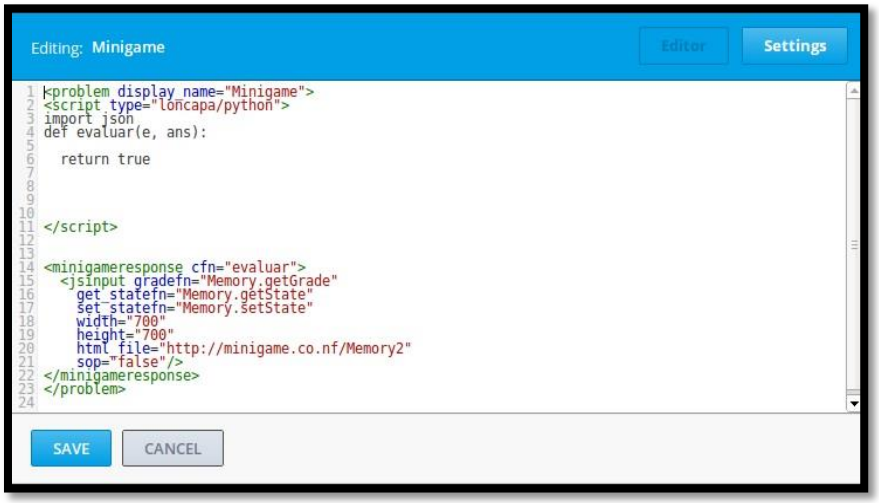


Imagen 5-20: Imagen del código para integrar el juego de Memory.

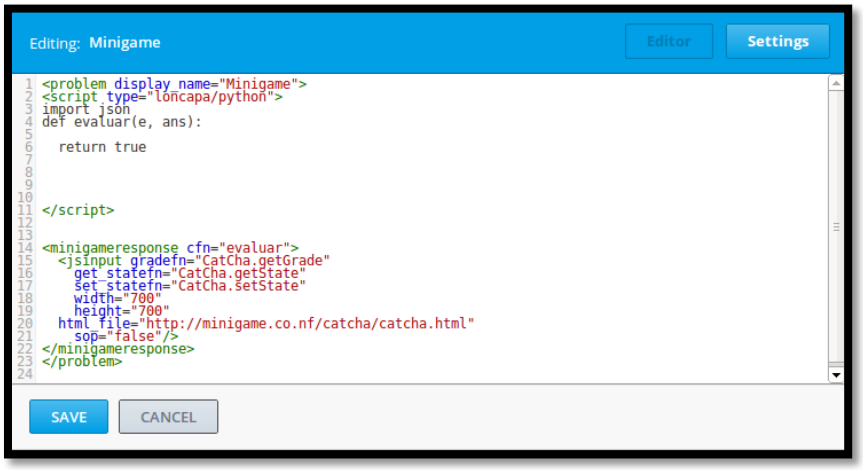


Imagen 5-21: Imagen del código para integrar el juego de Catcha.

6. Conclusiones y trabajo futuro

6.1. Conclusiones

Ya que la próxima versión de eAdventure (versión 2.0) va a permitir la exportación de juegos en JavaScript se decidió incorporar cualquier juego realizado en este lenguaje.

Durante el proceso de desarrollo de este proyecto se han visto varias formas de integración de los juegos en JavaScript dentro de edX con sus respectivas ventajas e inconvenientes.

En el inicio de esta investigación se empezó por estudiar la integración de los juegos como un xModule, utilizando un ejemplo en JavaScript para lograr incorporarlo de esta manera y averiguar qué información se podía guardar y qué información se podía transmitir. Se llegó a la conclusión de que esta forma de integración no era la ideal para este proyecto, ya que era necesario crear nuevas tablas en las base de datos y añadir ficheros que sólo podrían ser añadidos por el administrador de edX. Además no es posible integrar diferentes juegos con el mismo módulo, si no que se tendría que crear un módulo por cada juego.

Se investigó también la integración de los juegos a través de LTI, pero se decidió que no era factible, ya que no se podía transmitir información entre el módulo de LTI y edX.

Por último, se investigó la integración de juegos como ejercicio Lon-Capa y se llegó a la conclusión de que ésta es la mejor forma de integración, ya que da la posibilidad al profesor de elegir el juego que se quiere añadir como ejercicio sin necesidad de contactar con el administrador de la plataforma edX.

Dos de los juegos creados durante este proyecto, Catcha y Memory, han sido implementados específicamente para poder ir avanzando en la investigación e ir realizando el proceso de integración en Lon-Capa.

Sin embargo, Lon-Capa presenta varios problemas cuando se desea realizar un seguimiento de un alumno o un grupo de alumnos mediante e-learning analytics, ya que los datos proporcionados por Lon-Capa no serían suficientes para realizar este análisis y además no se guardan los progresos del estudiante si éste abandona el juego antes de completarlo. Para solucionarlo, edX está desarrollando una herramienta integrada en su plataforma cuyo nombre es insight que permite realizar esos análisis. Debido a que todavía no existe una versión para el desarrollador, se ha buscado una alternativa que pueda ofrecer este tipo de análisis aunque no esté integrada en edX. Esa alternativa es el estándar xAPI conjuntamente con un LRS que permite guardar toda la actividad de los usuarios y realizar un análisis posterior.

Por ello se ha adaptado otro juego (Castle defense) para poder obtener un análisis más detallado de las iteraciones de los alumnos y para realizar uso más completo de las prestaciones de xAPI.

Se ha llegado a la conclusión de que el análisis que se puede realizar con xAPI es muy amplio porque permite obtener numerosos datos tanto individuales como colectivos de los resultados de los alumnos. Los datos colectivos permiten observar si los ejercicios son demasiado fáciles o difíciles.

Tanto los juegos como el código necesario para realizar su análisis en xAPI, se encuentra disponible en los siguiente repositorio ("Github Integración juegos en edX," 2014), ("Github xAPI examples," 2014) y el código necesario para la integración de los mismos se encuentra en ("Github edx e-ucm," 2014).

6.2. Trabajo futuro

xBlock fue anunciado y lanzado públicamente el 14 de marzo de 2013. Se esperaba que estuviese desarrollado este año para poder así integrar juegos eAdventure en edX.

Debido a que xBlock está aún en desarrollo, se decidió usar xModule, que es el primer componente diseñado para esta plataforma. Sus funcionalidades son las mismas que las de xBlock pero con ciertas limitaciones ya que no es tan independiente. xBlock está mejor diseñado para que los desarrolladores puedan extender edX y por lo tanto sería interesante investigar y usar los xBlocks para integrar los juegos eAdventure en edX.

En cuanto al análisis de resultados en un principio el objetivo era realizarlo con Insight, pero debido a que aún no está terminado se ha decidido utilizar xAPI como alternativa.

Una vez que Insight esté disponible será la forma más completa y adecuada de realizar análisis de los juegos eAdventure integrados en edX ya que éste se encuentra integrado dentro de la plataforma. Como objetivo futuro sería interesante estudiar Insight para conseguir una mayor facilidad a la hora de la obtención de resultados de los alumnos y su posterior análisis.

También cabe mencionar que eAdventure lanzará próximamente la versión 2.0. Sería conveniente estudiar la integración de las Learning Analytics desarrolladas en este proyecto en los nuevos juegos de eAdventure2, independientemente de si se utiliza xAPI o Insight.

7. Bibliografía

Abbott, J., & September, G. K. (2012). Coding Problems in Authors / Co-Authors for.

Activity Streams. (2014). Retrieved June 08, 2014, from http://activitystrea.ms/

Arquitectura edX. (2013). Retrieved June 05, 2014, from https://docs.google.com/viewer?a=v&pid=forums&srcid=MDAxMjM3NDg1MDEzNjI0NjgwODYBMDc0MTQ0MjYwMzc1NTk5MDg2NzYBZy0zSDlKZW9aYkVKATQBAXYy&pli=1

Castle Defense | OpenGameArt.org. (2014). Retrieved June 19, 2014, from http://opengameart.org/content/castle-defense

Crear un curso en edx. (2014). Retrieved from http://files.edx.org/Getting_Started_with_Studio.pdf

Download Prototypes - Tin Can API. (2014). Retrieved June 08, 2014, from http://tincanapi.com/download-prototypes/

e-Adventure - Wikipedia. (2010). Retrieved from http://es.wikipedia.org/wiki/E-Adventure

e-Adventure ucm. (2009). Retrieved from http://e-adventure.e-ucm.es/lang.php?lang=es_es_utf8

edX installation Manual. (2014). Retrieved from https://github.com/edx/configuration/wiki/edX-Developer-Stack

edx installation tutorial Windows. (2013). Retrieved from http://arquitecturadesarrollosoftware.blogspot.com.es/2013/06/como-instalar-una-instancia-de-prueba.html#5

edx- Wikipedia. (2014). Retrieved from http://en.wikipedia.org/wiki/EdX

eUCM | eAdventure 2.0. (2014). Retrieved June 19, 2014, from http://www.e-ucm.es/ead2blog/

Github ADL_LRS. (2014). Retrieved from https://github.com/adlnet/ADL_LRS

Github Converting Statements to 1.0.0. (2014). Retrieved from https://github.com/adlnet/xAPI-Spec/blob/master/xAPI.md#AppendixD

GitHub Documents Apis. (2014). Retrieved from https://github.com/adlnet/xAPI-Spec/blob/master/xAPI.md#73-document-apis

Github eAdventure. (2014). Retrieved from https://github.com/e-ucm/ead

Github edx e-ucm. (2014). Retrieved from https://github.com/e-ucm/edx-platform

Github Error Codes. (2014). Retrieved from https://github.com/adlnet/xAPI-Spec/blob/master/xAPI.md#errorcodes

Github Experience API. (2014). Retrieved from https://github.com/adlnet/xAPI-Spec/blob/master/xAPI.md#rtcom

Github Experience API Client Examples. (2014). Retrieved from https://github.com/adlnet/experienceapi_client_examples

Github insights. (2014). Retrieved from https://github.com/edx/insights

Github Integración juegos en edX. (2014). Retrieved from https://github.com/danilc91/SI

Github xAPI examples. (2014). Retrieved from https://github.com/maria14/experienceapi_client_examples

Herrera Bautista, M. Á. (2013). Las nuevas tecnologías en el aprendizaje constructivo. *Revista Iberoamericana de Educación*. Retrieved from http://www.rieoei.org/deloslectores/821Herrera.PDF

HTTP/1.1: Header Field Definitions. (2014). Retrieved June 08, 2014, from http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.19

IMS Global Developer. (2004). Retrieved June 07, 2014, from http://developers.imsglobal.org/tutorials.html

IMS Global: LTI. (2004). Retrieved June 07, 2014, from http://www.imsglobal.org/toolsinteroperability2.cfm

Leyva García, V. H., & Díaz García, E. et. al. (2013). La función cognitiva del microcontrolador Arduino en la generación de aprendizaje. *UNAMente Robótica*, (5). Retrieved from http://dcb.fi-c.unam.mx/Publicaciones/UNAMenteRobotica/

Lon Capa. Elearning. (2014). Retrieved June 07, 2014, from http://www.uv.es/ticape/docs/sedelce/mem-sedelce.pdf

LON-CAPA - Wikipedia. (2013). Retrieved from http://es.wikipedia.org/wiki/LON-CAPA

OAuth Community Site. (2014). Retrieved June 07, 2014, from http://oauth.net/

Partes de un "Tin Can Statement." (2014). Retrieved April 30, 2014, from http://tincanapi.com/registry/

Pritchard. (2013). Bringing student backgrounds online: MOOC user demographics, site usage, and online learning. Retrieved April 29, 2014, from http://www.educationaldatamining.org/EDM2013/papers/rn_paper_57.pdf

Prototypes Launcher. (2014). Retrieved June 08, 2014, from http://tincanapi.com/prototypes/

Report Sample. (2014). Retrieved June 08, 2014, from http://tincanapi.com/report-sample/

SCORM. (2014). Retrieved June 07, 2014, from http://scorm.com/

Scorm Cloud SCORM Cloud. (2014). Retrieved June 08, 2014, from http://scorm.com/scorm-solved/scorm-cloud-features/

SCORM to Tin Can Solution. (2014). Retrieved June 08, 2014, from http://tincanapi.com/scorm-to-tin-can-solution/

SCORM vs Tin Can. (2014). Retrieved June 08, 2014, from http://tincanapi.com/scorm-vs-the-tin-can-api/

Statement Generator. (2014). Retrieved June 08, 2014, from http://tincanapi.com/statement-generator/

Statements 101. (2014). Retrieved June 07, 2014, from http://tincanapi.com/statements-101/

Tin Can API | Jose Manuel Martín. (2012). Retrieved June 07, 2014, from http://www.josemanuelmartin.com/2012/07/tin-can-api/

Tin Can Api ¿El futuro sustituto del SCORM? (2012). Retrieved June 08, 2014, from http://masquelearning.com/wordpress/tin-can-api-el-futuro-sustituto-del-scorm/

Uniform Resource Identifier. (2014). Retrieved June 07, 2014, from http://es.wikipedia.org/wiki/Uniform_Resource_Identifier

Vagrant issues. (2014). Retrieved from https://github.com/mitchellh/vagrant/issues/3769

Verbs. (2014). Retrieved June 07, 2014, from https://registry.tincanapi.com/#home/verbs

xAPI. (2014). Retrieved April 29, 2014, from http://tincanapi.com/

Xblock Documentation. (2014). Retrieved June 07, 2014, from https://media.readthedocs.org/pdf/xblock/latest/xblock.pdf

xBlock Wikipedia. (2014). Retrieved from http://en.wikipedia.org/wiki/XBlock