Bases del Concurso de Gamificación

Anexo 1

ULPGC Universidad de Las Palmas de Gran Canaria | Escuela de Ingeniería Informática

Fundación MAPFRE guanarteme

Tabla de contenido

1 Introducción

En el presente documento se detallará los aspectos técnicos que ha de cumplir en la implementación del juego online.

2 Funcionalidades y requisitos mínimos

2.1 Características del desarrollo

El desarrollo debe de ser realizado utilizando **HTML5** / **JavaScript** con objeto de utilizarlo por medio de un navegador web. Quedará excluido cualquier otro tipo de implementación relacionados con desarrollo de aplicaciones de escritorio, Android, IOS, etc.

Será un punto imprescindible que sea **compatible** con cualquier **dispositivo** (PC, Tablet, Móvil) así como en las versiones más actuales de los **navegadores** más demandados en el mercado (Firefox, Chrome, Safari, Opera, Internet Explorer).

El desarrollo a realizar debe de ser compatible con la plataforma de formación de la **Fundación MAPFRE Guanarteme**, basada esta última en **Moodle**, y la cual será instalada como una actividad de tipo **Scorm 1.2** en futuras acciones formativas de la fundación.

Este es un requisito prioritario, dado que los resultados del juego deben almacenarse en las hojas de calificaciones de las distintas acciones formativas donde se hará uso del juego.

El estándar Scorm 1.2 es muy extendido en la industria de la formación online, compatible entre las principales plataformas de formación online (entre ellas Moodle), y se encarga tanto de la definición del empaquetado de los contenidos online así como de la comunicación de información entre el contenido online y la plataforma de formación.

Para cumplir con estas características **se proveerá a los participantes del concurso de una API** que permitirá almacenar y recuperar datos (puntuación, estado del juego, tiempos), la cual será descrita en el apartado “Empaquetado Scorm & API”.

Por otro lado, este desarrollo será utilizado en distintas acciones formativas con diferentes temáticas, por lo tanto, otro de los requisitos del entregable final será la **reusabilidad** del juego, de forma que este pueda ser configurado y adaptado con facilidad a otras temáticas distintas a la aportada en las bases del concurso: cambios de texto, baterías de preguntas, apartado gráfico, modalidad de juego, etc. Este tipo de configuraciones se realizará mediante la definición de archivos XML los cuales deben de ser cargados de forma dinámica.

Junto al desarrollo se debe de aportar la siguiente documentación técnica:

- **Descripción funcional del juego para usuarios**, el cual será aportado en las acciones formativas a modo de ayuda.
- **Descripción técnica** del juego describiendo distintos tipos de configuraciones: textos, baterías de preguntas, parámetros para modificar comportamientos en el juego.

2.2 Lenguajes y librerías permitidas

El desarrollo del juego busca el uso de lenguajes de código abierto y de libre uso compatibles con navegadores, dado que este ha de ser ejecutado como una actividad Scorm dentro de una plataforma de formación Moodle, así como cualquier otro tipo de plataforma de formación compatible con el estándar SCORM 1.2.

Por ello, los lenguajes que se permitirán durante el desarrollo serán los siguientes:

- **Html5**
- **JavaScript**
- **CSS**
- **Xml**

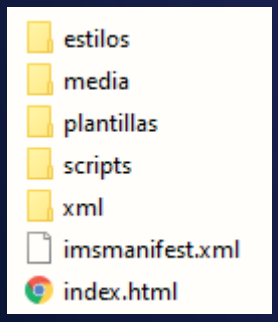
En lo referente a las APIS a utilizar, se proveerá de una API especifica (**CEXT**) de desarrollo especifico que facilite las labores de comunicación con Moodle para el reporte de puntuaciones, estado del juego, así como cualquier otro tipo de valores predefinidos en la misma.
Los participantes no podrán incluir librerías de terceros más allá de la librería jQuery y en caso de necesitarlo la librería jQuery UI.

2.3 Estructura de carpetas

Con objeto de hacer entregables homogéneos se define unas pautas en lo relacionado con la estructura de carpetas de los entregables, facilitando así la localización de los recursos para el equipo técnico y organizativo.

En la fase II se hará entrega a los participantes de una **carpeta de trabajo** inicial la cual incluirá un desarrollo vacío, pero con las librerías y Apis permitidas en el desarrollo siguiendo la estructura comentada.

A continuación, se describen las carpetas y archivos mínimos de esta estructura:



imsmanifest.xml Archivo manifiesto para reconocimiento de SCORM por parte de la plataforma e-learning. Este archivo tiene como objetivo indicar a la plataforma LMS cuál es el archivo que iniciará la actividad Scorm, en este caso el juego a desarrollar.
Este archivo no ha de modificarse, ya que esta actualmente vinculado con el archivo index.html que será el archivo principal de la aplicación.

index.html Archivo principal de la aplicación, donde se incluirán todas las librerías y hojas de estilos necesarias y desde donde se dará inicio el juego.

El archivo incluido contiene un código mínimo de inclusión de librerías e inicialización a partir del cual los participantes deben de continuar con su desarrollo.

Por motivos de implementación / sincronización con LMS, el inicio se realizará por medio de una función del api CEXT.

estilos Directorio para incluir las hojas de estilo del juego. Su contenido es el siguiente:

- **estilos.css**: Archivo base para la definición de estilos del juego, este archivo ya está enlazado en index.html
- **fuentes**: Directorio para incluir fuentes incrustadas en caso que se deseen usar.

media Directorio para los recursos multimedia del juego, imágenes, audios, etc. Su contenido debe de estar organizado por carpetas distintas para cada tipo de recurso:

- **Audio**: Directorio para audios
- **Imágenes**: Directorio para imágenes

plantillas Directorio para incluir (en el caso que se usen) archivos HTML con partes de código HTML a cargar.

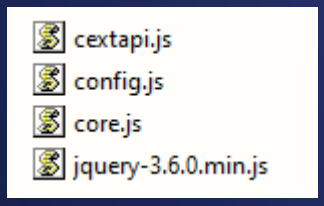
xml Directorio donde incluir todos los archivos XML para almacenamiento de datos que se deseen.

Se ha incluido un archivo XML de ejemplo para definir la estructura del cuestionario del juego, no obstante, la definición de este archivo queda totalmente abierta por los desarrolladores.

Esta carpeta debe de contener todo tipo de archivo xml que permita parametrizar de alguna forma la entrada de información del juego (preguntas, textos, configuraciones).

scripts Directorio donde incluir todos los archivos js que se necesiten crear, donde previamente se han incluido 4 iniciales que describimos a continuación

De todas estas carpetas, por su importancia, se describirá el contenido de la carpeta scripts:



Jquery-3.6.0.min.js Potente librería de uso extendido en la comunidad de desarrollo online para la creación de elementos interactivos y manejo del DOM del navegador.
Se podrá hacer libre uso de esta librería durante el desarrollo del juego.
Para ejemplos y más documentación: https://jquery.com/.
No se incluye jQuery UI pero se puede hacer si se desea.
Dado que es una librería de terceros, no se dará soporte de esta librería por medio del canal de comunicación con el equipo técnico del concurso.

cextapi.js Librería aportada para el concurso de uso especifico con la plataforma de formación Moodle de Fundación Mapfre. Esta librería se describe en detalle en los siguientes apartados y dará acceso al modelo de comunicación con el LMS para almacenar y recuperar datos de puntuación y estado del juego.

config.js Archivo destinado a la inclusión de variables globales de configuración internas del juego.

core.js Este es el archivo donde los participantes deben añadir la programación principal del juego.
En caso de necesidad se pueden crear otros archivos JS con objeto de organizar y estructurar mejor la programación.

2.4 Configuración del juego

Uno de los principales requisitos es la adaptabilidad del juego para su reutilización en distintas temáticas y contextos formativos.

Para ello, se valorarán positivamente todas las propuestas por parte de los participantes en pro de obtener cambios funcionales del juego, modificación de textos, preguntas, etc. los

cuales hagan del desarrollo una herramienta más versátil y adaptable a diferentes necesidades.

Este tipo de configuraciones se debe de implementar mediante la carga dinámica de información almacenada en distintos archivos en formato XML.

A continuación, se define el mínimo modelo de configuración esperado por la aplicación.

2.4.1 Preguntas

La información principal que ha de suministrarse a la aplicación son baterías de preguntas, estas clasificadas según temáticas (al estilo de los juegos trivial). El número de temáticas de las preguntas y las preguntas por cada temática ha de ser variable, ajustándose el juego a las que han sido definidas.

Sera un requisito que estas preguntas se definan mediante un archivo XML con una estructura de información capas de contener la batería de preguntas completa, así como modificarla a futuro para establecer otras temáticas distintas.

También se valorará que el juego pueda mostrar preguntas de otro tipo distinto (no solo preguntas de tipo test), como por ejemplo rellenar, emparejar, agrupar, etc..

2.4.2 Textuales

El juego mostrará, a parte de las preguntas, distintos textos en botones, pantallas, mensajes. Se valorará positivamente que estos textos estén localizados únicamente en un archivo xml donde pueda localizarse fácilmente cualquier texto, permitiendo traducciones o ajustes personalizados sin necesidad de revisar el código fuente.

Este archivo debe de ser un listado de identificadores y textos, de forma que cada vez que se añada un texto en el juego se haga uso del identificador ubicado en el archivo de configuración.

2.4.3 Configuración del juego

La mecánica de un juego puede dar pie a distintos mecanismos configurables, tales como aleatoriedad de las preguntas, activación o desactivación de funcionalidades, etc.

Este tipo de configuraciones quedará de mano del desarrollador, y se valorará positivamente los posibles ajustes configurables en el juego que permita que la herramienta sea más versátil.

No obstante, se requerirán como mínimo los siguientes ajustes:

- Control de aleatoriedad: Si se muestran las preguntas aleatoriamente, y que número de preguntas aleatorias se obtiene de cada temática.

Estos ajustes, para facilidad de comprensión, deben de estar en un archivo de configuración fácilmente localizable.

2.5 Responsive

El **diseño responsive** del juego hace referencia a la adaptabilidad de la disposición de los elementos de la interfaz del juego a las distintas resoluciones que admite la pantalla del dispositivo donde se va a realizar la experiencia de usuario.

No es la misma resolución (pixeles ancho x pixeles alto) la que pueden soportar los dispositivos hoy en día. Por ejemplo, una pantalla de ordenador tiene una resolución de pantalla mucho mayor que la de otros dispositivos como tablets o móviles los cuales a parte de tener menor resolución la pantalla real tiene muchas menos pulgadas, lo cual permite mucha menos información en pantalla.

Es requisito indispensable hacer el juego online compatible con los distintos dispositivos del mercado, y a su vez, se valorará la elegancia con la cual se adapta al medio.

Tener en cuenta que un diseño responsive dista de redimensionar y reescalar los contenidos al tamaño de la pantalla. El diseño para cada dispositivo debe de tener en cuenta muchos factores que faciliten el uso por el participante final tales como la cantidad de información a mostrar, la reubicación de elementos, mostrar el contenido con un tamaño que el usuario pueda ver fácilmente, etc.

2.6 Partidas y estado del juego

Uno de los puntos a tener en cuenta en la ejecución del juego es que este permitirá al usuario realizar una o **varias partidas**, salir de la partida en cualquier momento, iniciar una nueva partida.

La ejecución de cada partida generará una **puntuación final** y un logro, tanto si completó como si no.

A parte, como **puntuación global** se tendrá una puntuación general que se reflejará en el informe de calificaciones de Moodle (donde solo recibe una puntuación). Esta puede ser por ejemplo la puntuación mas alta, la media de todas las puntuaciones, la ultima puntuación (estas funciones podrían ser parámetros de configuración comentados en anteriores apartados).

Otro punto a tener en cuenta es la secuencia de pantallas que puedan aparecer para conformar un inicio de juego y continuidad de partida.

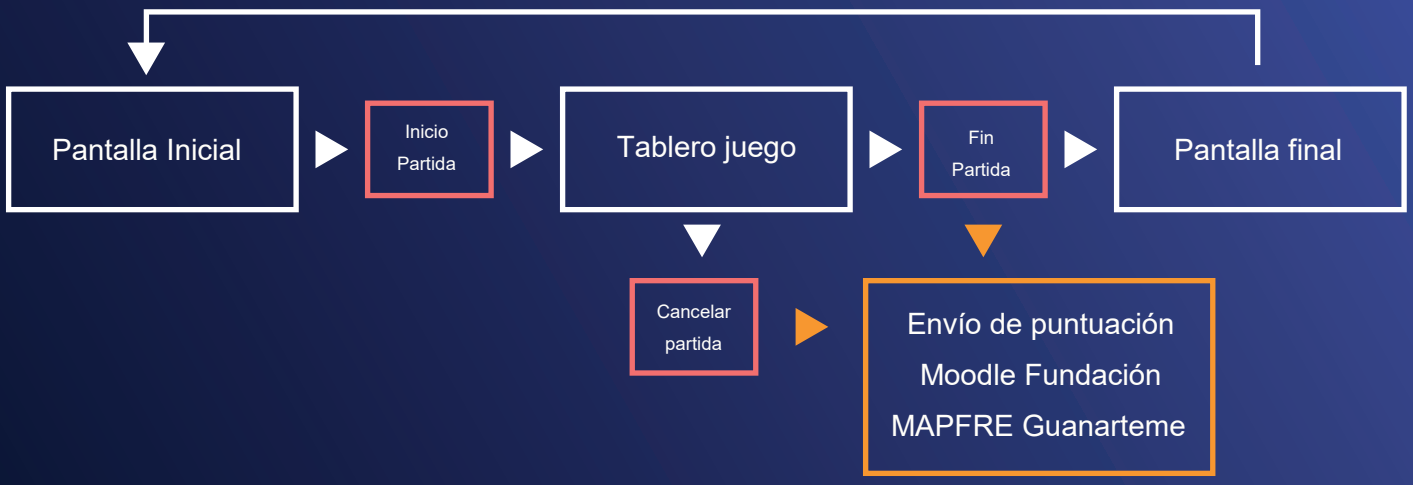
Mínimo se espera obtener las siguientes pantallas:

- **Pantalla presentación juego**: En esta pantalla se inicia la partida y se valorará positivamente la posibilidad de contextualizar el tipo de juego, instrucciones y reglas del juego, temática de las preguntas.

- **Tablero del juego**: Es la pantalla principal donde se mostrará el tablero de juego y donde se realizará la partida, siendo esta donde se centralizará la mayor parte del desarrollo.

- **Pantalla final**: Donde se mostrará al usuario el resultado de la partida, estadísticas del juego, puntuación obtenida, así como la posibilidad de volver a jugar.

El flujo de estados del juego y el envió de información a la plataforma queda representado con el siguiente esquema:



Cada vez que se realiza una partida se hará un envío de información a la plataforma Moodle:

En caso de finalizar:

- **Estado de la partida**: Completado / incompleto dependiendo si cancelo la partida o la finalizó. Si se define una puntuación mínima de superación de juego también podría basarse en este parámetro para indicar el completado / incompleto.
- **Puntuación obtenida en la partida**: Puntuación / logro de la partida recién finalizada
- **Puntuación general del juego:** Una única puntuación se traslada a la hoja de calificaciones, podría ser por ejemplo la última puntuación de la ultima partida, o una media de puntuaciones, o la máxima puntuación alcanzada. Este puede ser un parámetro configurable en el juego.
- **Estado general del juego**: Un único estado general representando todas las partidas. Este estado puede ser por ejemplo un estado “completo” cuando se ha finalizado alguna de las partidas, o el estado de finalización del último intento. Este puede ser un parámetro configurable en el juego.

2.7 Sistema de puntuación

El cálculo de la puntuación dependerá de la mecánica del juego, pero debe estar basado en un criterio lógico, (por ejemplo, % de preguntas acertadas con respecto del total) y debe definirse inequívocamente en el MVP.

No obstante, hay una restricción respecto a la puntuación, ya que el valor a devolver a la plataforma ha de estar en un rango de 0 a 100.

Es por ello que sea cual visualmente la puntuación que obtiene el usuario en el juego, como puntuación final, debe de disponer de un valor en este rango que represente el resultado del juego.

2.8 Empaquetado Scorm & API

Tal como se ha indicado previamente, el juego debe de funcionar siguiendo las indicaciones definidas en el estándar Scorm 1.2 es muy extendido en la industria de la formación online, compatible entre las principales plataformas de formación online (entre ellas Moodle), y se encarga tanto de la definición del empaquetado de los contenidos online así como de la comunicación de información entre el contenido online y la plataforma de formación.

De esta forma se logra:

- Que cualquier plataforma, al instalar el juego como paquete Scorm, sea capaz de inicializarlo correctamente (empaquetado, definido en el archivo imsmanifest.xml).
- Que el juego pueda trasladar puntuaciones con un modelo estándar que conozca cualquier plataforma del mercado que soporte Scorm 1.2.

Para abstraer a los participantes del concurso de la necesidad de conocer los entresijos del estándar Scorm 1.2, se ha optado por facilitar un API que permita hacer uso del envio de

información y sincronización de plataforma sin necesidad de conocimientos salvo las llamadas definidas en la API.

Haciendo uso del paquete base junto con el api CEXT, y haciendo las llamadas correctas al api, lo único a tener en cuenta es la forma de empaquetar el contenido. Un paquete Scorm es un **archivo zip** de los archivos online comprimiendo desde la ruta base, es decir, dejando el archivo imsmanifest.xml en la raíz de la estructura comprimida. De esta forma, las plataformas LMS, al instalar el zip y descomprimir su contenido saben exactamente donde encontrar el archivo imsmanifes.xml utilizado para la instalación.

2.8.1 Api

El api en cuestión se denomina **CEXT** y esta ubicado en el archivo **/scripts/cextapi.js** ya vinculado en el archivo **index.html** suministrado en el paquete base de desarrollo.

Respecto a este API conocer que realiza las siguientes actuaciones:

- Inicializa la comunicación con la plataforma LMS.
- Recupera información previamente almacenada mediante llamadas a la API.
- Almacena información mediante llamadas a la API.
- Finaliza la comunicación en la plataforma LMS, estableciendo automáticamente valores tales como el tiempo dedicado.

El API se basa en el objeto CEXT, ya instanciado por la inclusión de las librerías. Para inicializar el juego se ha de lanzar con el siguiente evento:

```
<script>

    CEXT.onInit=function()
    {
        // codigo de inicio del juego
    }

</script>
```

De esta forma nos aseguramos que el juego se inicializa sincronizado al momento en el cual se ha inicializado la comunicación con la plataforma LMS, momento en el cual se pueden hacer uso del resto de métodos del API.

Los métodos del objeto son los siguientes:

Método	Descripción
setGlobalPoints (int puntos // valor de 0 a 100)	Establece la puntuación global del juego. Debe de ser siempre un valor de 0 a 100. Ejemplo: *CEXT.setGlobalPoints(100);*
return null	
setGlobalStatus (bool estado // true completo false incompleto)	Establece el estado "Finalizado" o "No finalizado" global del juego por medio de un valor booleano. Ejemplo: *CEXT. setGlobalStatus (true); // Establece estado finalizado* *CEXT. setGlobalStatus (false); // Establece estado no finalizado*
return null	
getGlobalPoints()	Recupera la puntuación global del juego. Debe de ser siempre un valor de 0 a 100. Ejemplo: *// Devuelve puntos almacenados* *puntosGlobales=CEXT.getGlobalPoints();*
return int (puntos de 0 a 100)	
getGlobalStatus()	Recupera el estado global del juego. Ejemplo: *// Valor true o false dependiendo del estado finalizado o no finalizado* *estado=CEXT. getGlobalStatus ();*
return boolean (estado)	
setAttempPoints(int n, // número de partida int points // puntos de 0 a 100)	Establece la puntuación de la partida nº "N" del juego. Debe de ser siempre un valor de 0 a 100. Ejemplo: *// Establece la puntuación 50 para la partida n° 1* *CEXT. setAttempPoints (1,50);*
return null	

setAttempStatus (int n, // número de partida bool estado // true completo false incompleto) **return** null	Establece el estado de la partida nº "N" del juego. Debe de ser un valor true en caso de finalizar correctamente la partida o false en caso de no finalizarla. Ejemplo: *// Indica que la partida 1 ha finalizado correctamente (no ha abandonado).* *CEXT. setAttempStatus (1,true);*
getAttempPoints (int n // número de partida) **return** int (puntos de 0 a 100)	Recupera la puntuación de la partida nº "N". Ejemplo: *// Devuelve los puntos que almacenamos para la partida 1* *puntos=CEXT.getAttempPoints(1);*
getAttempStatus(int n // número de partida) **return** estado // true completo // false incompleto	Recupera el estado de la partida nº "N". Ejemplo: *// Devuelve los puntos que almacenamos para la partida 1* *estado=CEXT.getAttempStatus(1);*
starNewAttemp() **return** null	Se ha de llamar antes de iniciar una nueva partida, incluida cuando se inicia la primera partida. Permite llevar una cuenta del número de intentos. Establece automáticamente el valor de puntuación 0 / incompleto en la partida actual. Este método se ha de llamar inmediatamente después de pulsar el botón "Start" que inicie la partida. *partidas=CEXT.startNewAttemp();*
getNumAttemps() **return** int; // numero de partidas realizadas	Devuelve el número de partidas realizadas contando la actual. Ejemplo: *// Devuelve el número de partidas* *partidas=CEXT.getNumAttemps();*

<table>
<tr><td>getCurrentAttemp()</td><td rowspan="2">Devuelve el número de partida actual. Ejemplo:

// Devuelve cual es la partida actual
numPartidaActual=CEXT. getCurrentAttemp (100);

// Establece el valor 70 en la partida actual
CEXT. setAttempPoints (CEXT. getCurrentAttemp (),70);</td></tr>
<tr><td>return int; // número intento actual</td></tr>
<tr><td>setTmpData
(string data; // datos temporales)</td><td rowspan="2">Almacena una cadena de texto de tamaño máximo 1000 caracteres, la cual permite almacenar datos temporales.

Este valor puede ser recuperado tras cerrar el navegador y volver a iniciar el juego.

Es un valor útil cuando se quiere recordar información de partidas de anteriores inicios del juego.

Ejemplo:

CEXT.setTmpData(“TotalA=100;
TotalB=30;TotalAciertos=2;TotalFallos=33”);</td></tr>
<tr><td>return null</td></tr>
<tr><td>getTmpData()</td><td rowspan="2">Recupera el valor almacenado como tmpData.

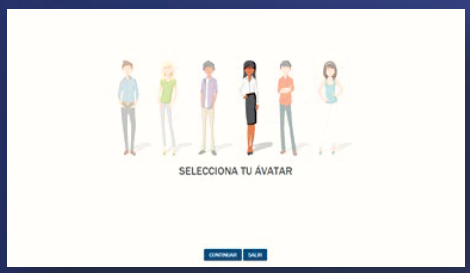
Ejemplo:

// Según ejemplo anterior devolvería
// “TotalA=100; TotalB=30;TotalAciertos=2;TotalFallos=33”

tmpData=CEXT.getTmpData();</td></tr>
<tr><td>return string</td></tr>
</table>

3 Posibles mejoras del entregable

Este documento recoge las especificaciones técnicas mínimas que debe cumplir el juego, no obstante, se deja la libertad a los participantes para incluir cualquier mejora que consideren una aportación valiosa al mismo, las cuales serán valoradas positivamente.



Ejemplos de posibles mejoras:

- Posible selección del alumno (usuario final del juego) de las categorías con las que desea jugar, añadiendo una pantalla de selección de categorías.

- Recuperación de una partida abandonada, para continuarla por donde se dejo al cerrar el navegador.

- Inclusión de un personaje conductor que nos realice cada pregunta.

- Posibilidad de incluir imagen/texto tanto en los enunciados de las preguntas como en las diferentes opciones.

- Posibilidad de incluir locuciones sincronizadas con los diferentes elementos del juego (enunciado de una pregunta, de las opciones, etc.)

- Posibilidad de elegir tu propio avatar en la partida, recordando la elección en los distintos inicios de sesión.

- Posibilidad de confeccionar la apariencia gráfica de tu avatar (ojos, boca, pelo, color de piel…).

- Añadir múltiples tipos de preguntas (relacionar, rellenar huecos, etc..)
- Definir preguntas con distinto valor de puntuación (pesos)