

ESCUELA TÉCNICA SUPERIOR DE INGENIERÍA DE
TELECOMUNICACIÓN

UNIVERSIDAD POLITÉCNICA DE CARTAGENA



Trabajo Fin de Grado

Desarrollo de juegos para entornos docentes mediante tecnologías web

-

Game development educational environments through web technologies



AUTOR: David Martínez Gisbert

DIRECTOR: Juan Luis Pedreño Molina

CO-DIRECTOR: Daniel Pérez Berenguer

Septiembre / 2016

Índice

Resumen.....	4
Abstract.....	5
Capítulo 1: Introducción	7
1.1. Análisis del problema.....	8
1.2. Solución del problema.....	8
1.3. Estructura del proyecto.....	9
Capítulo 2: Objetivos del proyecto	11
2.1. Objetivos funcionales.....	12
2.2. Características del juego.....	12
Capítulo 3: Tecnologías empleadas.....	14
3.1. Tecnologías del lado del cliente (Front-end).....	15
3.1.1. HTML5 (HyperText Markup Language, versión 5)	15
3.1.2. CSS3	17
3.1.3. Javascript.....	18
3.1.4. AJAX (Asynchronous JavaScript And XML)	19
3.1.5. Canvas.....	19
3.2. Tecnologías del lado del servidor (Back-end).....	20
3.2.1. PHP (PHP Hypertext Preprocessor).....	20
3.2.2. MySQL	21
3.2.3. BASH.....	22
3.3. Frameworks auxiliares	22
3.3.1. jQuery	22
3.3.2. jQuery UI	23
3.3.3. CreateJS	24
3.3.4. Twitter Bootstrap	24
3.4. Otras tecnologías	25
3.4.1. Adobe Illustrator	25
Capítulo 4: Desarrollo del proyecto.....	28
4.1. Fase de elección del tipo de juego.....	29
4.2. Fase de diseño de la lógica de juego	30
4.3. Fase de diseño de la interfaz del juego.....	31
4.3.1. Pantalla de inicio.....	34
4.3.2. Pantalla principal del juego.....	34

4.3.3.	Pantalla de mostrar pregunta.....	37
4.3.4.	Pantalla de resultado de la pregunta.....	38
4.3.5.	Pantalla de resultado del reto	40
4.4.	Fase de desarrollo del juego	41
4.4.1.	Fase 1: Creación del esqueleto de la aplicación.....	41
4.4.1.1.	Cargador.js.....	42
4.4.1.2.	MostrarImagen.js.....	42
4.4.1.3.	ContenedorImágenes.js	45
4.4.1.4.	App.js	47
4.4.1.5.	global.js	49
4.4.1.6.	functions.js	49
4.4.2.	Fase 2: Comunicación con el servidor	51
4.4.2.1.	Cambios realizados a los ficheros “Javascript”	52
4.4.2.2.	dbconnect.php	52
4.4.2.3.	find_opponent.php.....	52
4.4.2.4.	retrieve_challenge.php	54
4.4.2.5.	update_challenge_state.....	56
4.4.2.6.	update_question.....	57
4.4.3.	Fase 3: Implementación de la comunicación mediante mensajería	58
4.4.3.1.	Servicio de mensajería instantánea: Whatsapp	58
4.4.3.2.	Servicio de mensajería instantánea: Telegram	58
4.4.3.2.1.	send_message.sh.....	58
4.4.3.2.2.	send_message.php	59
4.4.3.3.	Servicio de mensajería: Correo electrónico	60
4.4.3.3.1.	send_mail.php.....	61
4.4.3.4.	Cambios en los ficheros “PHP”	62
4.4.3.4.1.	find_opponent.php.....	62
4.4.3.4.2.	update_challenge_state.php.....	63
4.4.4.	Fase 4: Implementación del gestor de bancos de preguntas en la plataforma UPCTplay	65
4.4.4.1.	Pantalla de creación de preguntas	66
4.4.4.2.	Pantalla de visionado y modificación de preguntas.	69
4.4.4.3.	Pantalla de asignación de preguntas a una actividad.....	73
Capítulo 5: Conclusiones y líneas futuras.....		78
5.1.	Líneas futuras	80
Capítulo 6: Bibliografía		82

Resumen

La irrupción de las nuevas tecnologías en los entornos docentes, están haciendo necesaria una revisión de los principios en los que se apoya la educación, ya que las Tecnologías de la Información y Comunicación (TIC) proporcionan nuevos planteamientos para el aprendizaje. Tendencias como la *flipped classroom* o “aula invertida”, donde el alumno estudia la teoría en casa y resuelve sus dudas, ejercicios y trabajos en clase, o la *gamificación* o “ludificación”, que introduce la dinámica del juego en el aprendizaje, hacen que las clases tomen un mayor dinamismo y, por tanto, sean más atractivas para el alumno.

La Universidad Politécnica de Cartagena quiere dar el salto y empezar a implementar estas nuevas metodologías en sus enseñanzas regladas. A través del Centro de Producción de Contenidos Digitales se originan varios proyectos entre los que se encuentra la creación de un juego para aplicar la *gamificación* en las diferentes asignaturas de la universidad.

En esta memoria se expondrá la solución llevada a cabo, con sus diferentes fases y tecnologías usadas.

Palabras clave: gamificación, aprendizaje, innovación docente, tecnologías web.

Abstract

The emergence of new technologies in educational environments, has made necessary a review of the principles on which education is based, as the Information Technology and Communication (ICT) provide new approaches to learning they have become necessary. Trends such as the flipped classroom where students study the theory at home and answers their questions, exercises and assignments in class, or gamification which introduces the game dynamics in learning, make the classes more dynamic and therefore more attractive to the student.

The Polytechnic University of Cartagena wants to start implementing these new methodologies in their regulated education. Through Production Center Digital Content several projects among which is the creation of a game to implement gamification in different subjects of the university originate.

In this memory we expose the solution carried out, with their different phases and technologies used.

Keywords: gamification, learning, teaching innovation, web technologies.

Capítulo 1: Introducción

La irrupción de las nuevas tecnologías en los entornos docentes, están haciendo necesaria una revisión de los principios en los que se apoya la educación, ya que las Tecnologías de la Información y Comunicación (TIC) proporcionan nuevos planteamientos para el aprendizaje. Tendencias como la *flipped classroom* o “aula invertida”, donde el alumno estudia la teoría en casa y resuelve sus dudas, ejercicios y trabajos en clase, o la *gamificación* o “ludificación”, que introduce la dinámica del juego en el aprendizaje, hacen que las clases tomen un mayor dinamismo y, por tanto, sean más atractivas para el alumno.

1.1. Análisis del problema

La Universidad Politécnica de Cartagena quiere dar el salto y empezar a implementar nuevas metodologías docentes en sus enseñanzas regladas.

Este proyecto surge en el Centro de Producción de Contenidos Digitales (CPCD) de la Universidad Politécnica de Cartagena (UPCT) dentro del marco de innovación docente que se lleva a cabo en el mismo. Se pretende el desarrollo de un juego para ser aplicado en la enseñanza reglada de la Universidad Politécnica de Cartagena.

El juego deberá tener un diseño atractivo e intuitivo que anime a los alumnos a llevar a cabo una participación activa en el mismo. Además, el juego deberá incluir un sistema de notificaciones para que los alumnos sepan cuando han sido retados o cual ha sido el resultado de su partida.

Por otro lado, se presenta otro problema: no se dispone de un sistema gestor de bancos de preguntas para automatizar el juego, por lo que se deberá desarrollar uno donde se puedan crear y modificar las preguntas para, posteriormente, asignarlas a cada actividad de juego.

1.2. Solución del problema

Una vez reconocidos los objetivos y requisitos del juego, se decide desarrollarlo como una aplicación web, de forma que pueda accederse a ella desde cualquier dispositivo electrónico que cuente con un navegador web y conexión a internet. Para ello, se decide realizar un diseño adaptativo que permita visualizarse correctamente en todo tipo de dispositivos, independientemente del sistema operativo o navegador que se esté usando.

Por otro lado, el sistema de gestión de bancos de preguntas también se decide desarrollarlo como una aplicación web por el mismo motivo que el juego. El diseño deberá ser intuitivo para

el usuario.

1.3. Estructura del proyecto

El proyecto seguirá la siguiente estructura:

- Estudio y selección de las tecnologías necesarias para la realización del proyecto.
- Desarrollo del juego mediante las tecnologías seleccionadas.
- Simulación.
- Análisis de resultados.
- Conclusiones.

A partir de esta estructura se definen los siguientes capítulos de los que constará la memoria:

- **Capítulo 2, Objetivos del proyecto.** En este capítulo se detallan los objetivos funcionales y las características deseadas para el juego, así como las características del sistema gestor de bancos de preguntas.
- **Capítulo 3, Tecnologías empleadas.** En este capítulo se muestran las tecnologías que se van a usar, así como una descripción de las mismas.
- **Capítulo 4, Desarrollo del proyecto.** En este capítulo se detallan las distintas fases de desarrollo del proyecto, así como los problemas que han ido surgiendo en la elaboración del mismo y su correspondiente solución.
- **Capítulo 5, Conclusiones y líneas de trabajo futuras.** En este capítulo se analizará la experiencia de juego mediante un caso de uso real en la asignatura “Gestión de los Recursos Humanos” del Grado en Administración y Dirección de Empresas de la Universidad Politécnica de Cartagena, en el instituto CEFP Carlos III y en el servicio de CRAI Biblioteca de la Universidad Politécnica de Cartagena. Además, se mostrarán las líneas de trabajo que podrían seguirse en un futuro.

Capítulo 2: Objetivos del proyecto

A continuación, se exponen los objetivos definidos para el proyecto, así como las características que debe disponer el juego.

2.1. Objetivos funcionales

Los objetivos del proyecto serán los siguientes:

- **Gestión a través de una plataforma web.** El juego será enlazado desde la plataforma UPCTplay, que se enmarca en otro proyecto de innovación docente del Centro de Producción de Contenidos Digitales.
- **Parametrización según necesidades del docente.** El juego recogerá los diferentes parámetros que le lleguen desde la plataforma, permitiendo así al docente adaptar el juego a sus necesidades. Con esto, varios profesores podrían usar el juego a la vez, cada uno utilizando sus propias preguntas, cambiando el tiempo máximo de juego o el número de preguntas de cada reto.
- **Adaptación multidispositivo.** Debido a la gran variedad de dispositivos electrónicos, ya sean smartphones, tablets u ordenadores, con los que hoy en día cuenta gran parte de la población, se hace necesario que el juego se pueda usar en todos ellos, por lo tanto, se realizará un diseño *responsive*

2.2. Características del juego

Las características del juego serán las siguientes:

- El juego será enlazado desde la plataforma UPCTplay.
- El juego consistirá en retos entre jugadores que ganará aquel que conteste el mayor número de preguntas de forma correcta.
- El reto será lanzado por uno de los jugadores.
- El reto se le comunicará al oponente vía Telegram y/o correo electrónico.

Capítulo 3: Tecnologías empleadas

En esta sección se enumerarán las diferentes tecnologías empleadas durante el desarrollo del proyecto, así como una breve descripción de las mismas.

3.1. Tecnologías del lado del cliente (Front-end)

3.1.1. HTML5 (HyperText Markup Language, versión 5)

Para hablar de HTML5 primero debemos saber qué es HTML.

HTML (HyperText Markup Language) hace referencia al lenguaje de marcado para la elaboración de páginas web. Es un estándar a cargo del World Wide Web Consortium (W3C), organización dedicada a la estandarización de casi todas las tecnologías ligadas a la web, sobre todo en lo referente a su escritura e interpretación. Este estándar define una estructura básica y un código (denominado código HTML) para la definición del contenido de una página web.

Al ser un estándar, HTML busca ser un lenguaje que permita que cualquier página web escrita en una determinada versión, pueda ser interpretada de la misma forma (estándar) por cualquier navegador web actualizado.

A principios de 1990, Tim Berners-Lee define el HTML como un subconjunto de SGML (Standard Generalized Markup Language) con un conjunto de marcas muy reducido que permitiera su utilización en cualquier sistema operativo y la creación de enlaces de hipertexto. El éxito de la World Wide Web en los primeros años 90 hizo necesario la ampliación de las características del HTML, lo que requería nuevas marcas.

El W3C, creado a mediados de los 90, publicó en enero de 1997 la recomendación HTML 3.2. Esta versión incorporó muchas estructuras nuevas al HTML (tablas, imágenes flotantes, hojas de estilo).

En diciembre de 1998 se publicó la recomendación HTML 4.0. Los cambios más importantes fueron la incorporación de los frames y la mayor relevancia de las hojas de estilo.

En diciembre de 1999 se publicó la recomendación HTML 4.01 que modificaba ligeramente la versión anterior.

Con estas recomendaciones, uno de los objetivos del W3C había sido profundizar la separación entre contenido y su presentación, es decir, eliminar del HTML las etiquetas que hicieran referencia directa al formato visual (y almacenar esta información en una hoja de estilo).

En enero de 2000 se crea la recomendación XHTML 1.0 (eXtensible HyperText Markup Language). Esta no es más que el HTML 4.01 adaptado a las reglas del XML (eXtensible Markup Language).

Ante este bloqueo que impedía el desarrollo del HTML (el W3C había abandonado el desarrollo del HTML en favor del XHTML pero el mercado -por culpa de Microsoft, especialmente- rechazaba el XHTML), en 2004 se creó el WHATWG (Web Hypertext Application Technology Working Group), un grupo de trabajo informal formado por individuos y empresas (Mozilla, Opera, Apple, Google entre ellas) y dedicado al desarrollo del HTML, sobre todo para facilitar el desarrollo de las aplicaciones Web. Este grupo empezó a elaborar las especificaciones HTML 5, Web Forms 2.0 y Web Controls 1.0 con la intención de que en el futuro se publicaran como recomendaciones del W3C con el nombre de HTML 5.



Figura 3.1. Logo de HTML5.

En enero de 2011, el W3C anunció su compromiso definitivo con el HTML 5 y la intención de publicar la recomendación.

En HTML5 se añaden nuevos elementos (etiquetas), atributos y comportamientos, así como un conjunto más amplio de tecnologías que permiten a los sitios Web y a las aplicaciones ser más diversas y de gran alcance. Estas tecnologías se pueden clasificar en varios grupos según su función:

- **Semántica:** Permite describir con mayor precisión cuál es su contenido.
- **Conectividad:** Permite comunicarse con el servidor de formas nuevas e innovadoras.
- **Sin conexión y almacenamiento:** Permite a las páginas web almacenar datos localmente

en el lado del cliente y operar sin conexión de manera más eficiente.

- **Multimedia:** Nos otorga un excelente soporte para utilizar contenido multimedia como lo son audio y video nativamente.
- **Rendimiento e Integración:** Proporciona una mayor optimización de la velocidad y un mejor uso del hardware.
- **Acceso al dispositivo:** Proporciona APIs para el uso de varios componentes internos de entrada y salida de nuestro dispositivo.
- **CSS3:** Nos ofrece una nueva gran variedad de opciones para hacer diseños más sofisticados.
- **Gráficos y efectos 2D/3D:** Proporciona una amplia gama de nuevas características que se ocupan de los gráficos en la web como lo son canvas 2D, WebGL, SVG, etc.

3.1.2. CSS3

CSS (Cascading Style Sheets) es el lenguaje utilizado para describir la presentación de documentos HTML o XML, esto incluye varios lenguajes basados en XML como son XHTML o SVG. CSS describe como debe ser renderizado el elemento estructurado en pantalla, en papel, hablado o en otros medios.

CSS es uno de los lenguajes base de la Open Web y posee una especificación estandarizada por parte del W3C. Desarrollado en niveles, CSS1 es ahora obsoleto, CSS2.1 es una recomendación y CSS3, ahora dividido en módulos más pequeños, está progresando en camino al estándar.



Figura 3.2. Logo de CSS3.

CSS3 es la última evolución del lenguaje de las Hojas de Estilo en Cascada (Cascading Style Sheets), y pretende ampliar la versión CSS2.1. Trae consigo muchas novedades altamente

esperadas, como las esquinas redondeadas, sombras, gradientes, transiciones o animaciones, y nuevos layouts como multi-columnas, cajas flexibles o maquetas de diseño en cuadrícula (grid layouts).

3.1.3. Javascript

JavaScript® (a veces abreviado como JS) es un lenguaje ligero e interpretado, orientado a objetos con funciones de primera clase, más conocido como el lenguaje de script para páginas web. Es un lenguaje script multi-paradigma, basado en prototipos, dinámico, soporta estilos de programación funcional, orientada a objetos e imperativa.

Todos los navegadores modernos interpretan el código JavaScript integrado en las páginas web. Para interactuar con una página web se provee al lenguaje JavaScript de una implementación del DOM (Document Object Model).



Figura 3.3. Logo de Javascript.

Tradicionalmente se venía utilizando en páginas web HTML para realizar operaciones y únicamente en el marco de la aplicación cliente, sin acceso a funciones del servidor. Actualmente es ampliamente utilizado para enviar y recibir información del servidor junto con ayuda de otras tecnologías como AJAX (Asynchronous JavaScript And XML). JavaScript se interpreta en el agente de usuario al mismo tiempo que las sentencias van descargándose junto con el código HTML.

Desde el lanzamiento en junio de 1997 del estándar ECMAScript 1, han existido las versiones 2, 3 y 5, que es la más usada actualmente. En junio de 2015 se cerró y publicó la versión

ECMAScript 6.

3.1.4. AJAX (Asynchronous JavaScript And XML)

AJAX es una técnica de desarrollo web para crear aplicaciones interactivas o RIA (Rich Internet Applications). Estas aplicaciones se ejecutan en el cliente, es decir, en el navegador de los usuarios mientras se mantiene la comunicación asíncrona con el servidor en segundo plano. De esta forma es posible realizar cambios sobre las páginas sin necesidad de recargarlas, mejorando la interactividad, velocidad y usabilidad en las aplicaciones.



Figura 3.4. Logo de AJAX.

Ajax es una tecnología asíncrona, en el sentido de que los datos adicionales se solicitan al servidor y se cargan en segundo plano sin interferir con la visualización ni el comportamiento de la página, aunque existe la posibilidad de configurar las peticiones como síncronas de tal forma que la interactividad de la página se detiene hasta la espera de la respuesta por parte del servidor, como usaremos en muchos casos.

3.1.5. Canvas

Canvas es una de las nuevas tecnologías introducidas en la última versión de HTML. Sus usos son muy variados, pudiendo, por ejemplo, usarse para dibujar gráficos a través de scripting (normalmente con Javascript), hacer composición de fotos, crear animaciones o incluso procesamiento de video en tiempo real.

Para hacer más sencillo el trabajo con canvas, haremos uso del framework CreateJS.

3.2. Tecnologías del lado del servidor (Back-end)

3.2.1. PHP (PHP Hypertext Preprocessor)

PHP es un lenguaje de programación de uso general de código del lado del servidor originalmente diseñado para el desarrollo web de contenido dinámico. Fue uno de los primeros lenguajes de programación del lado del servidor que se podían incorporar directamente en el documento HTML en lugar de llamar a un archivo externo que procese los datos. El código es interpretado por un servidor web con un módulo de procesador de PHP que genera la página web resultante.



Figura 3.5. Logo de PHP.

Fue creado originalmente por Rasmus Lerdorf en 1995. Actualmente el lenguaje sigue siendo desarrollado con nuevas funciones por el grupo PHP. Este lenguaje forma parte del software libre publicado bajo la licencia PHP, que es incompatible con la Licencia Pública General (GPL) de GNU debido a las restricciones del uso del término PHP.

PHP puede ser desplegado en la mayoría de los servidores web y en casi todos los sistemas operativos y plataformas sin coste alguno.

El gran parecido que posee PHP con los lenguajes más comunes de programación estructurada, como C y Perl, permiten a la mayoría de los programadores crear aplicaciones complejas con una curva de aprendizaje muy corta.

La versión que usaremos de PHP será la 5.6.x, una de las últimas versiones estables.

Por último, destacaremos que permite la conexión a diferentes tipos de servidores de bases de datos tanto SQL (Structured Query Language) como NoSQL tales como MySQL, PostgreSQL, SQLite o MongoDB. En nuestro caso, usaremos bases de datos SQL, en particular MySQL.

3.2.2. MySQL

Antes de hablar de MySQL, explicaremos que es SQL.

SQL (Structured Query Language) es un lenguaje declarativo de acceso a bases de datos relacionales que permite especificar diversos tipos de operaciones en ellas. Este, explota la flexibilidad y potencia de los sistemas relacionales y permite así gran variedad de operaciones.



Figura 3.6. Logo de MySQL.

Es un lenguaje declarativo de "alto nivel" o "de no procedimiento" que, gracias a su fuerte base teórica y su orientación al manejo de conjuntos de registros - y no a registros individuales - permite una alta productividad en codificación y la orientación a objetos. De esta forma, una sola sentencia puede equivaler a uno o más programas que se utilizarían en un lenguaje de bajo nivel orientado a registros. SQL también tiene las siguientes características:

- Lenguaje de definición de datos: El LDD de SQL proporciona comandos para la definición de esquemas de relación, borrado de relaciones y modificaciones de los esquemas de relación.
- Lenguaje interactivo de manipulación de datos: El LMD de SQL incluye lenguajes de consultas basado tanto en álgebra relacional como en cálculo relacional de tuplas.
- Integridad: El LDD de SQL incluye comandos para especificar las restricciones de integridad que deben cumplir los datos almacenados en la base de datos.
- Definición de vistas: El LDD incluye comandos para definir las vistas.
- Control de transacciones: SQL tiene comandos para especificar el comienzo y el final de una transacción.
- SQL incorporado y dinámico: Esto quiere decir que se pueden incorporar instrucciones de SQL en lenguajes de programación como: C++, C, Java, PHP, Cobol, Pascal y Fortran.

- Autorización: El LDD incluye comandos para especificar los derechos de acceso a las relaciones y a las vistas.

MySQL es un sistema de gestión de bases de datos relacional desarrollado bajo licencia dual GPL/Licencia comercial por Oracle Corporation y está considerada como la base datos open source más popular del mundo, y una de las más populares en general junto a Oracle y Microsoft SQL Server, sobre todo para entornos de desarrollo web.

MySQL fue inicialmente desarrollado por MySQL AB (empresa fundada por David Axmark, Allan Larsson y Michael Widenius). MySQL A.B. fue adquirida por Sun Microsystems en 2008, y ésta a su vez fue comprada por Oracle Corporation en 2010.

Al contrario de proyectos como Apache, donde el software es desarrollado por una comunidad pública y los derechos de autor del código están en poder del autor individual, MySQL es patrocinado por una empresa privada, que posee el copyright de la mayor parte del código. Esto es lo que posibilita el esquema de doble licenciamiento anteriormente mencionado. La base de datos se distribuye en varias versiones, una Community, distribuida bajo la Licencia pública general de GNU, versión 2, y varias versiones Enterprise, para aquellas empresas que quieran incorporarlo en productos privativos.

Utilizaremos MySQL para guardar la base de datos de retos y las preguntas del juego.

3.2.3. BASH

BASH es un shell de Unix (intérprete de comandos de Unix) escrito para el proyecto GNU.

La sintaxis de órdenes de bash es un superconjunto de la sintaxis del shell Bourne.

Usaremos BASH para llamar al programa de envío de mensajes de Telegram desde PHP.

3.3. Frameworks auxiliares

3.3.1. jQuery

jQuery es una biblioteca de JavaScript, creada inicialmente por John Resig, que permite simplificar la manera de interactuar con los documentos HTML. Al igual que otras bibliotecas, ofrece una serie de funcionalidades basadas en JavaScript que de otra manera requerirían de mucho más código, es decir, con las funciones propias de esta biblioteca se logran grandes

resultados en menos tiempo y espacio.



Figura 3.7. Logo de jQuery.

Entre sus características se pueden destacar:

- Selección de elementos DOM.
- Interactividad y modificaciones del árbol DOM.
- Eventos.
- Efectos y animaciones.
- AJAX.

3.3.2. jQuery UI

jQuery UI es una biblioteca de componentes para el framework jQuery que le añaden un conjunto de plug-ins, widgets y efectos visuales para la creación de aplicaciones web. Cada componente o módulo se desarrolla de acuerdo a la filosofía de jQuery.



Figura 3.8. Logo de jQuery UI.

La biblioteca se divide en cuatro módulos:

- **Núcleo.** Contiene las funciones básicas para el resto de módulos.
- **Interacciones.** Añade comportamientos complejos a los elementos.
- **Widgets.** Es un conjunto completo de controles UI (*User Interface*). Cada control tiene un conjunto de opciones configurables y se les pueden aplicar estilos CSS.
- **Efectos.** Una API (*Application Programming Interface*) para añadir transiciones animadas y facilidades para interacciones.

De esta biblioteca usaremos los módulos núcleo e interacciones para crear el gestor de bancos de preguntas.

3.3.3. CreateJS

CreateJS es un conjunto de bibliotecas modulares y herramientas JavaScript que trabajan conjuntamente para permitir un rico contenido interactivo en tecnologías web abiertas a través de HTML5. Estas bibliotecas están diseñadas para funcionar de forma totalmente independiente, o mezclar y combinar para satisfacer sus necesidades.



Figura 3.9. Logo de CreateJS.

La suite CreateJS se compone de 4 bibliotecas:

- EaselJS: proporciona una lista completa jerárquica, un modelo de interacción, y las clases de ayuda para que pueda trabajar con el elemento HTML5 Canvas de una manera mucho más fácil.
- SoundJS: es una biblioteca de Javascript para trabajar con audio.
- PreloadJS: hace la precarga de activos y eventos de progreso agregados más fácil en JavaScript.
- TweenJS: es una simple pero potente biblioteca de interpolación / animación de Javascript.

En el desarrollo del proyecto se usarán únicamente las bibliotecas EaselJS, para la interacción con los elementos del canvas, y SoundJS, para la interacción con los sonidos del juego.

3.3.4. Twitter Bootstrap

Twitter Bootstrap (en adelante Bootstrap) es un framework de código abierto para diseño de sitios y aplicaciones web. Contiene plantillas de diseño con tipografía, formularios, botones, cuadros, menús de navegación y otros elementos de diseño basado en HTML y CSS, así como, extensiones de JavaScript opcionales adicionales.

Bootstrap fue desarrollado por Mark Otto y Jacob Thornton de Twitter, como un framework

para fomentar la consistencia a través de herramientas internas. Antes de Bootstrap, se usaban varias librerías para el desarrollo de interfaces de usuario, las cuales guiaban a inconsistencias y a una carga de trabajo alta en su mantenimiento.



Figura 3.10. Logo de Twitter Bootstrap.

Bootstrap proporciona un conjunto de hojas de estilo que proveen definiciones básicas de estilo para todos los componentes de HTML. Esto otorga una uniformidad al navegador y al sistema de anchura, da una apariencia moderna para el formateo de los elementos de texto, tablas y formularios.

En adición a los elementos regulares de HTML, Bootstrap contiene otra interfaz de elementos comúnmente usados. Ésta incluye botones con características avanzadas (e.g grupo de botones o botones con opción de menú desplegable, listas de navegación, etiquetas horizontales y verticales, ruta de navegación, paginación, etc.), etiquetas, capacidades avanzadas de miniaturas tipográficas, formatos para mensajes de alerta y barras de progreso.

Los componentes de JavaScript para Bootstrap están basados en la librería jQuery de JavaScript. Los plug-ins se encuentran en la herramienta de plug-in de jQuery. Proveen elementos adicionales de interfaz de usuario como diálogos, tooltips y carruseles.

Usaremos Bootstrap para la presentación del banco de preguntas.

3.4. Otras tecnologías

3.4.1. Adobe Illustrator

Adobe Illustrator (AI) es un editor de gráficos vectoriales en forma de taller de arte que trabaja sobre un tablero de dibujo y está destinado a la creación artística de dibujo y pintura para ilustración.

Está desarrollado y comercializado por Adobe Systems.



Figura 3.11. Logo de Adobe Illustrator.

Adobe Illustrator contiene opciones creativas, un acceso más sencillo a las herramientas y una gran versatilidad para producir rápidamente gráficos flexibles cuyos usos se dan en (maquetación-publicación) impresión, vídeo, publicación en la Web y dispositivos móviles.

Lo utilizaremos para realizar la composición de las imágenes del juego, ya que, al ser gráfico vectorial, no pierde calidad al reescalar las imágenes.

Capítulo 4: Desarrollo del proyecto

El proyecto se plantea en diciembre de 2015 como una actividad adicional a una experiencia piloto de innovación docente que va a comenzar a finales de febrero en la asignatura “Gestión de los Recursos Humanos” del Grado en Administración y Dirección de Empresas de la Universidad Politécnica de Cartagena. Se plantea desplegar una primera versión del juego antes de finalizar la asignatura, para utilizarlo como una evaluación del seguimiento del curso.

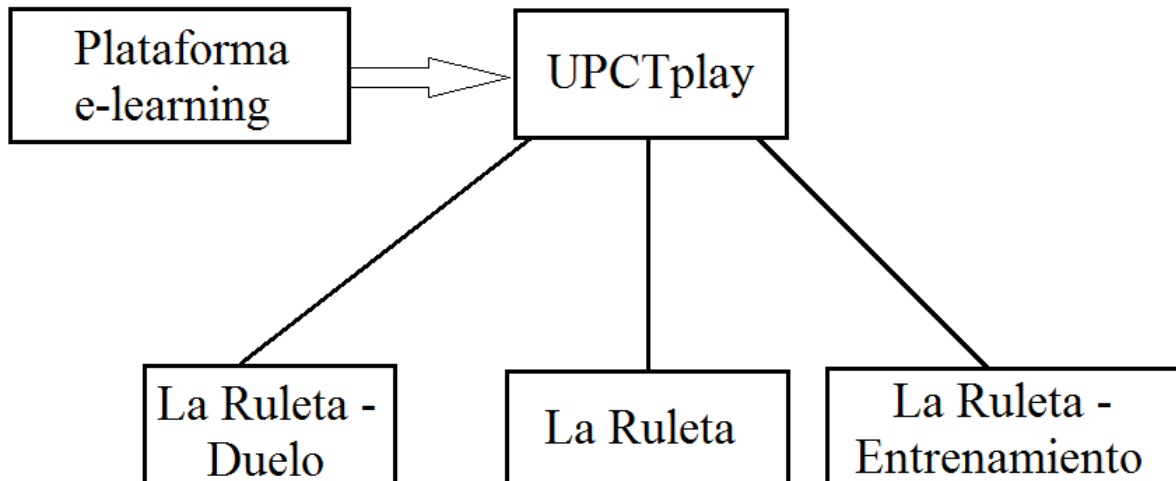


Figura 4.1. Estructura final de la aplicación

Al finalizar el proyecto se tendrá una aplicación totalmente funcional con todas las características propuestas en un principio. En el gráfico superior se muestra un diagrama de cómo estará enlazado el juego. Para llegar al juego, primero se debe crear una actividad en UPCTplay. Esta generará un enlace que el docente tiene que facilitar a los alumnos para registrarse en la actividad. La actividad se podrá crear para cualquiera de los juegos disponibles. Una vez registrados todos los alumnos participantes en el juego, este se habilitará para que los participantes comiencen a retarse. Al finalizar el periodo de juego, el docente podrá poner nota a los alumnos que hayan quedado en mejor posición, y esta se vinculará a la plataforma e-learning.

La realización del proyecto se dividió en varias fases.

4.1. Fase de elección del tipo de juego

La primera fase que se llevó a cabo fue la elección del tipo de juego a desarrollar. Se decidió elaborar un juego en el que los alumnos tuvieran que contestar preguntas relacionadas con la temática de la asignatura. Con el objetivo de conseguir una mayor motivación por parte del alumno, se decidió que los alumnos se retaran entre ellos en base a un conjunto de preguntas.

Existe una plataforma, UCPTplay, donde el docente crea las actividades de gamificación estableciendo el periodo de fechas en las que se llevará a cabo. En dicha plataforma, el docente tiene a su disposición un banco de preguntas donde puede grabar, eliminar o modificar preguntas y las respuestas asociadas a las mismas. Una vez creada la actividad, el docente le asigna las preguntas del banco de preguntas que quiere utilizar. Existen dos periodos de tiempo: periodo de inscripción y periodo de juego. En el periodo de inscripción los alumnos, a través del aula virtual, se inscriben en la actividad de gamificación. El periodo de juego comprende el desarrollo de la actividad.

Se decide crear el juego La Ruleta. Este juego se basa en las siguientes reglas:

- El juego se basa en un número de vidas establecidas como parámetro por el docente. Este número de vidas equivale al número de partidas o retos que puede jugar el alumno.
- En cada reto el alumno contesta un número de preguntas preestablecido por el docente. Las preguntas podrán tener entre dos y cuatro respuesta con solo una verdadera.
- Gana el jugador que mayor número de preguntas conteste de forma correcta.
- Cada partida ganada equivale a 3 puntos, empatada a 1 punto y 0 puntos en el caso de perder la partida
- El juego lo ganará el que mayor puntuación obtenga a lo largo de todas las partidas. En la plataforma UPCTplay, tanto el docente como el alumno, tienen a su disposición un ranking donde poder consultar esta información en tiempo real.

Tras esto se pasó a la fase de diseño de la lógica del juego.

4.2. Fase de diseño de la lógica de juego

En esta fase se definió cuál sería la lógica del juego.

El juego constaría de 5 fases:

- 1) **Inicio del juego.** En esta fase se cargará la aplicación al completo y se habilitará un botón para pasar a la siguiente fase.
- 2) **Elección de la pregunta.** En esta fase se elegirá una pregunta al azar sobre una de las categorías disponibles.
- 3) **Responder a la pregunta.** En esta fase aparecerá la pregunta en la pantalla con sus distintas respuestas.

- 4) **Resultado de la pregunta.** En esta fase se mostrará si se ha acertado o fallado la pregunta.
- 5) **Resultado del reto.** Tras finalizar la última pregunta, se mostrará el número de preguntas acertadas y falladas.

4.3. Fase de diseño de la interfaz del juego

En esta fase se propusieron varios diseños. Como el diseño debía ser amigable, atractivo e intuitivo, se decidió que el elemento principal del juego sería una ruleta y que el fondo del juego siempre sería el mismo.

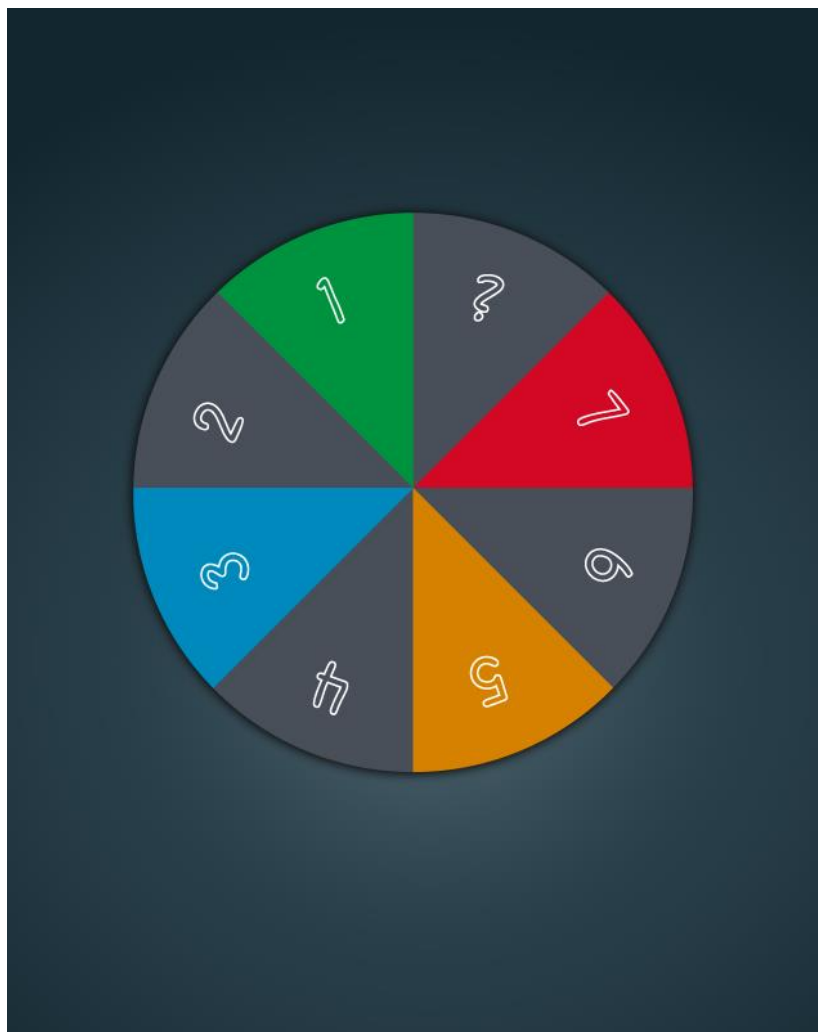


Figura 4.2. Diseño de la ruleta

Esta ruleta se dividiría en un total de 7 categorías más una categoría aleatoria de entre las 7 anteriores. Por tanto, la ruleta tendrá 8 secciones numeradas de 1 al 7 para las categorías fijadas, más un símbolo de interrogación para la categoría aleatoria.

El siguiente elemento que debía predominar serían los botones, ya que son más intuitivos que cualquier otro elemento, y se usan de la misma forma tanto en dispositivos que usan el ratón, como en dispositivos táctiles.

Existirán 2 tipos de botones:

- **De control de la aplicación.** Se usarán para pasar entre las diferentes pantallas.



Figura 4.3. Diseño de los botones de control de las pantallas.

- **De control de las respuestas.** Se usarán para saber cuál es la respuesta que ha dado el usuario.

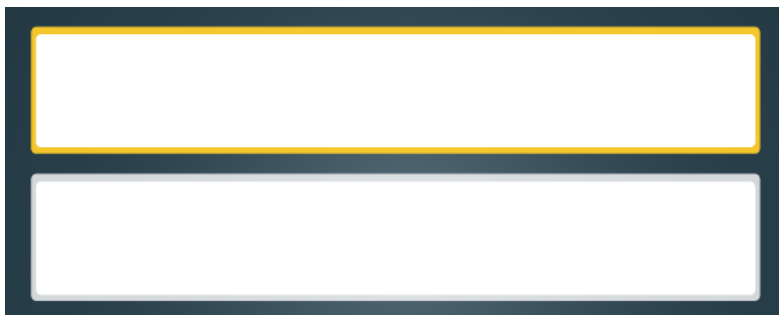


Figura 4.4. Diseño de los botones de control de las preguntas.

Como elementos añadidos para dar más atractivo al juego se proponen los siguientes:

- **Unas mascotas.** Se decide escoger unos monstruos con un diseño amigable para amenizar el juego.

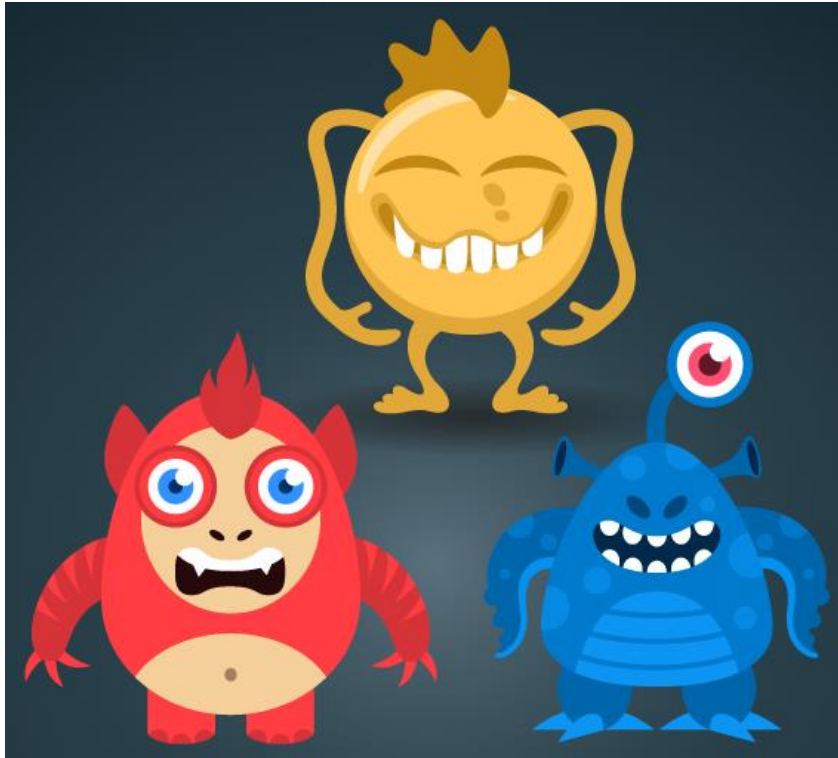


Figura 4.5. Diseño de las mascotas del juego.

- **Una peana.** Para que la pantalla no se quedara muy vacía y demasiado corta, se decide que la ruleta irá sobre una peana.



Figura 4.6. Diseño de la peana.

Además de estos elementos adicionales, se propone que haya elementos que le muestren al usuario el progreso del juego, es decir, que el usuario pueda saber cuántas preguntas lleva contestadas, cuántas ha acertado, cuántas ha fallado, y el tiempo que le queda para poder terminar de responder, así como quién es su oponente.

Ahora pasamos a definir las pantallas del juego, según la lógica especificada en el punto anterior.

4.3.1. Pantalla de inicio

La pantalla de inicio se divide en 3 secciones:

- **Zona superior.** En esta zona se muestran los logos de la Universidad Politécnica de Cartagena y del Centro de Producción de Contenidos Digitales.
- **Zona central.** En esta zona se muestra el logo de la plataforma UPCTplay, así como el nombre del juego y un botón con la leyenda “Comenzar” para comenzar el juego.
- **Zona inferior.** Esta zona servirá para mantener un *feedback* con el usuario en caso de algún problema.



Figura 4.7. Diseño de la pantalla de inicio del juego.

4.3.2. Pantalla principal del juego

La pantalla principal tendrá 2 fases:

- a) Cuando no ha girado aún la ruleta y no hay ninguna categoría elegida.
- b) Cuando ya se elegido una categoría.

Estas 2 fases únicamente se diferencian en la zona central de la pantalla.

Podemos dividir la pantalla en 3 zonas:

- **Zona superior.** En esta zona se mostrará el progreso del juego, así como la información de los jugadores.

Para mostrar el progreso del juego se decide hacer una banda dividida en 3 regiones:

- En la primera región se mostrará el número de preguntas contestadas, así como el número de preguntas totales de la partida.
- En la segunda región se mostrará el número de preguntas acertadas y el número de preguntas falladas.
- En la tercera región se mostrará el tiempo restante, en forma de cuenta atrás.



Figura 4.8. Diseño de la banda donde se muestra el progreso del juego.

Para la zona de información de los jugadores se decide poner unos avatares, que cambiarán en función de si los jugadores son chico o chica. Al lado de cada avatar aparecerá el nombre del jugador.



Figura 4.9. Diseño de los avatares de los jugadores.

- **Zona inferior.** En esta zona irá la imagen de la peana sobre la que se dispondrá la ruleta o la mascota, dependiendo de la fase en la que nos encontremos.
- **Zona central.** En esta zona vamos a describir cada una de las 2 fases diferentes:
 - **Fase 1, la ruleta aún no ha girado.** En esta fase aparecerá la ruleta, dividida en sus 8 categorías, 7 categorías dadas más la categoría al azar. En el centro de la ruleta encontraremos un botón con la leyenda “Pulsa” en el centro.



Figura 4.10. Diseño de la pantalla principal, antes de girar la ruleta.

- **Fase 2, se ha elegido una categoría.** En esta fase desaparecerá la ruleta y aparecerá una de las mascotas en su lugar. Encima de la mascota aparecerá una banda con el nombre de la categoría que se haya elegido y debajo de ella un botón con la leyenda “*Jugar*”.



Figura 4.11. Diseño de la pantalla principal, después de girar la ruleta.

4.3.3. Pantalla de mostrar pregunta

En esta pantalla seguirá apareciendo la banda de progreso del juego, pero desaparecerá la zona de información de los jugadores.

Debajo de la banda de progreso del juego podremos encontrar 2 zonas diferenciadas:

- **Zona de pregunta.** Esta zona estará dispuesta justo debajo de la banda de progreso del juego. En ella aparecerá un cuadro donde se mostrará la pregunta que deberá responder el jugador.
- **Zona de respuestas.** Esta zona estará dispuesta a continuación de la anterior. En ella aparecerán, según el número de respuestas, entre 2 y 4 botones de control de respuestas. Estos botones son los que controlan las respuestas que da el jugador.

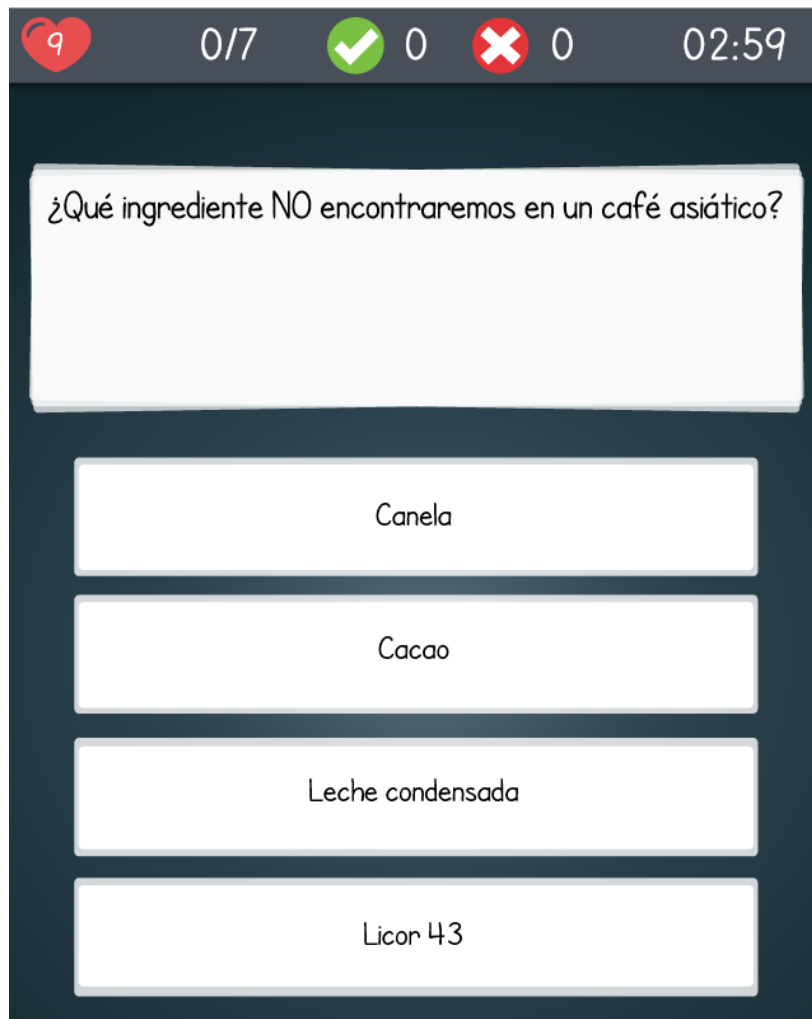


Figura 4.12. Diseño de la pantalla donde aparecen las preguntas y sus respuestas.

4.3.4. Pantalla de resultado de la pregunta

En esta pantalla se mostrará si hemos acertado o fallado la pregunta. Dependiendo del resultado, se verá una de las siguientes opciones, con un elemento en común, la banda de progreso del juego:

- Si se acertara la pregunta, aparecería el texto “*Correcto*” en la parte superior junto con un monstruo de cara alegre en la parte inferior. Debajo del texto encontraríamos un botón con la leyenda “*Continuar*”.



Figura 4.13. Diseño de la pantalla de respuesta correcta.

- Si se fallara, aparecería el texto “*Incorrecto*” en la parte superior, con un botón debajo con la leyenda “*Continuar*”. En la parte inferior encontraríamos un monstruo con cara triste y, debajo de él, la respuesta correcta.



Figura 4.14. Diseño de la pantalla de respuesta incorrecta.

4.3.5. Pantalla de resultado del reto

Esta pantalla aparecerá, a continuación de la anterior, cuando se hayan contestado todas las preguntas de la partida. En ella se mostrará una ventana modal en la que se hará un resumen de la partida, con el número de preguntas que hemos acertado y fallado.

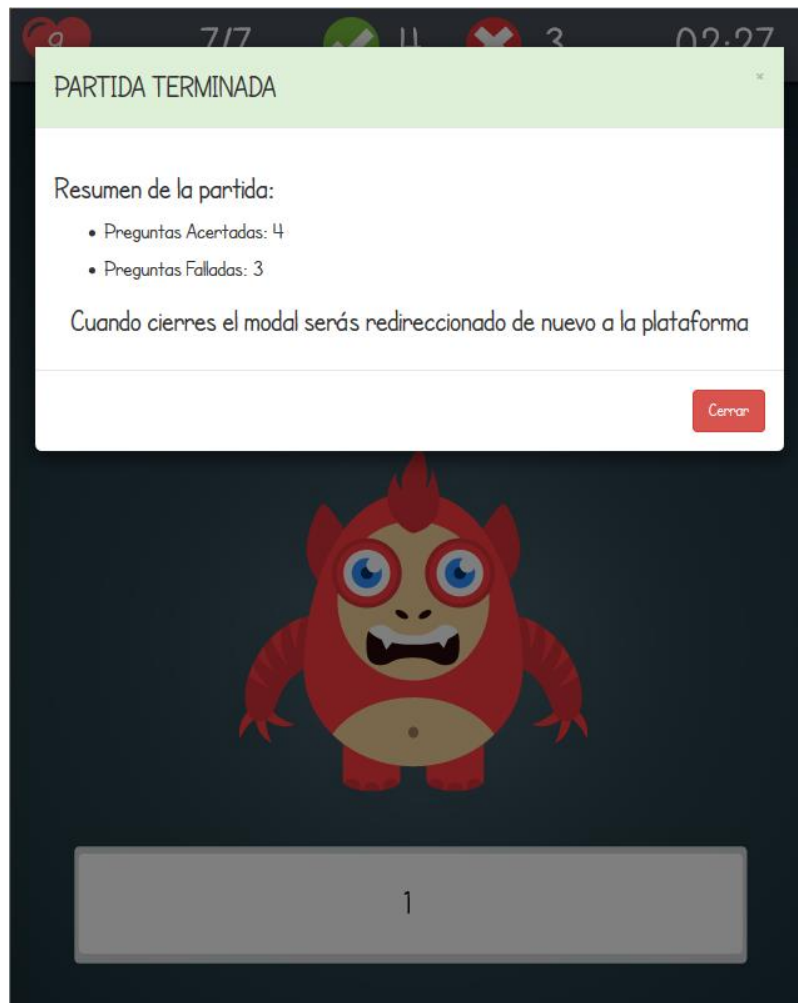


Figura 4.15. Diseño de la pantalla de resultado del juego.

4.4. Fase de desarrollo del juego

Terminado el diseño de las diferentes pantallas del juego, se comienza a desarrollar el juego.

Se pueden diferenciar distintas fases del desarrollo del juego:

4.4.1. Fase 1: Creación del esqueleto de la aplicación

En esta primera fase se creará el esqueleto de la aplicación, con los componentes básicos para que funcione, pero sin que haya comunicación con el exterior, es decir, no habrá conexiones a la base de datos y no se enviarán datos de los retos. En esta fase, lo que se tendrá será una demo funcional en la que podrá jugar una persona para probar el juego sin que se registre ningún dato. Se realizará en el lenguaje “*Javascript*”, ya que sólo tendremos interacciones del cliente con el navegador.

El objetivo de esta fase será que podamos interactuar con todos los elementos del juego. La

acción más importante que se deberá implementar será el giro de la ruleta, ya que es el elemento principal del juego. Gracias al *framework* “*CreateJS*”, que ofrece un conjunto de funciones para trabajar cómodamente sobre el elemento *canvas*, se podrá realizar esta acción mediante la rotación del objeto de la ruleta. Otras de las acciones que se deberán implementar serán la interacción con los botones, que tendrán una función distinta, dependiendo de la pantalla en la que nos encontremos.

Antes de comenzar el proyecto, se habían desarrollado 4 ficheros “*Javascript*” en los que se encuentran 3 clases para crear animaciones con “*CreateJS*”. Se utilizarán estas 3 clases para definir cada objeto de imagen, modificando algunas de sus funciones para adaptarlas al desarrollo, y se usarán las funciones nativas de “*CreateJS*” para realizar las animaciones de la ruleta y los cambios de pantalla.

4.4.1.1. [Cargador.js](#)

En este fichero se encuentra la clase “Cargador” que se encarga de cargar las imágenes en memoria, para poder acceder a ellas más tarde.

Las funciones que dispone son las siguientes:

- initialize()
 - Esta carga en memoria el objeto “Cargador” para poder usarlo de manera global.
- loadImagenes(lista)
 - Se encarga de recorrer la lista de rutas de imagen que se le pasa como argumento para pasarlas a la función “cargaImagen()”.
- cargaImagen(ruta)
 - Esta crea los objetos de imagen a partir de la ruta que se le pasa como parámetro y llama a la función “imagenCargada()” cuando ésta termina de cargarse.
- imagenCargada()
 - Esta asigna al evento “oncomplete” la función “onComplete()” que se ejecutará cada vez que una imagen termine de cargarse.

4.4.1.2. [MostrarImagen.js](#)

En este fichero se encuentra la clase “MostrarImagen” que se encarga de mostrar las imágenes, que se han cargado anteriormente, y de asignar eventos a los objetos de imagen. Además, se definen las funciones que ejecutarán dichos eventos. Los eventos que se asignarán en este caso

serán:

- mousedown
 - Se ejecutará cada vez que hagamos click con el ratón en un objeto imagen. La función asignada a este evento será “handleMouseDown(e)”.
- mouseover
 - Se ejecutará cada vez que pasemos el ratón por encima de un objeto imagen. La función asignada a este evento será “handleMouseOver(e)”.
- mouseout
 - Se ejecutará cada vez que el ratón salga del área de la imagen. La función asignada a este evento será “handleMouseOut(e)”
- tick
 - Se ejecutará cada que haya una actualización del “stage”, el área del canvas. La función asignada a este evento será “onTick()”.

Ahora se hablará de las funciones de esta clase:

- initialize(id, image, posX, posY)
 - Esta carga en memoria la imagen [image] en la posición que se le pasa como argumento [posX,posY] y le asigna los eventos citados anteriormente. Por último, realiza una actualización del “stage”.
- onTick()
 - Esta se encarga de hacer girar la ruleta durante un tiempo dado. En el siguiente fragmento de código podemos ver como se hace:

```

46  MostrarImagen.prototype.onTick = function() {
47
48      switch (this.id) {
49          case 0:
50              this.regX = 316;
51              this.regY = 377;
52              if ((contador >= 1) && (contador < (maxTime-2))) {
53                  if ((contador % 20) == 0) {
54                      vueltas = vueltas / 2
55                  }
56                  this.rotation += vueltas;
57                  rotacion = this.rotation;
58              }
59              break;
60          }
61      }
--

```

Figura 4.16. Código que hace girar la ruleta.

Cada imagen tiene asignado un identificador [id]. Como sólo se quiere que se mueva la ruleta, cuando el identificador asociado al evento “tick” sea el de la esta, se hará girar un número aleatorio de veces [vueltas] que vaya decreciendo conforme pase el tiempo [contador], para hacer que la ruleta gire más rápido al principio y se vaya parando. Cuando [contador] cumpla el tiempo máximo de giro [maxTime], se tendrá un valor de rotación que será el que se use para calcular en qué categoría se ha parado.

- handleMouseDown(e)
 - Esta se encarga de recoger el id del objeto que hemos pulsado y pasarlo a la función “Recargarapp(vid)”, de la que se hablará más adelante. En el siguiente fragmento de código se pueden ver las restricciones que se han impuesto a la función.

```

64  MostrarImagen.prototype.handleClick = function(e) {
65      if ((this.id == 1 && !empezado) || (this.id == 4) || (this.id == 7 &&
        !btncomenzar) || (this.id == 8 && !pregunta) || (this.id == 9 && !pregunta)
        || (this.id == 10 && !pregunta) || (this.id == 11 && !pregunta) || (this.id
        == 12)) {
66          Recargarapp(this.id);
67      }

```

Figura 4.17. Código que restringe la pulsación de elementos del juego.

Como se puede observar, sólo se ejecutará la función “Recargarapp(vid)” en los objetos de imagen con los siguientes identificadores: 1, 4, 7, 8, 9, 10, 11, 12. Además, existen otras restricciones como:

- El objeto con identificador 1 sólo se podrá pulsar si la ruleta no está girando.
 - Los objetos con identificadores: 8, 9, 10, 11, sólo se podrán pulsar cuando no se haya pulsado ninguno de los otros antes, es decir, si pulsamos el objeto con identificador 8 significa que hemos pulsado la respuesta 1, por lo tanto, hasta que no pasemos a la siguiente pregunta no podremos volver a pulsar ninguna de las respuestas.
- handleMouseOver(e)
 - Esta se encarga de cambiar el tipo de cursor del ratón al pasar sobre un objeto imagen. En el siguiente fragmento de código se pueden ver las restricciones que se han impuesto a la función.

```

70 | MostrarImagen.prototype.handleMouseOver = function(e) {
71 |     if ((this.id == 1 && !empezado) || (this.id == 4) || (this.id == 7 &&
    |     !btncomenzar) || (this.id == 8) || (this.id == 9) ||
72 |     (this.id == 10) || (this.id == 11) || (this.id == 12)) {
73 |         $("#juego").addClass('mano');
74 |     }
75 | }

```

Figura 4.18. Código que restringe el evento “mouseover” en los elementos del juego.

Como se puede observar, sólo se cambiará el tipo de cursor cuando se pase sobre los objetos con identificador: 1, 4, 7, 8, 9, 10, 11, 12. Además, en el objeto con identificador 1, sólo se cambiará si la ruleta no está girando.

- handleMouseOut(e)
 - Esta se encarga de volver a cambiar el cursor cuando se sale del área del objeto de imagen sobre el que se encontraba. En este caso no se impondrá ninguna restricción, ya que sólo se cambia el cursor a su aspecto por defecto.

```

76 | MostrarImagen.prototype.handleMouseOut = function(e) {
77 |     $("#juego").removeClass('mano');
78 | }

```

Figura 4.19. Código que revierte el cursor a su valor por defecto.

4.4.1.3. ContenedorImágenes.js

En este fichero se encuentra la clase “ContenedorImágenes” que se encarga de efectuar los cambios de objetos de imágenes que se muestran en pantalla. Dispone de las siguientes funciones:

- initialize()
 - Esta se encarga de crear el contenedor donde irán los objetos de imagen.
- init()
 - Esta se encarga de centrar el contenedor creado en la función anterior y llamar a la función que carga los objetos en el contenedor “cargarPiezas(e)”.
- cargarPiezas(e)
 - Esta, dependiendo del valor de la variable “enablePiezasapp”, cargará en el contenedor los diferentes objetos de imagen con las diferentes pantallas del juego. El único objeto de imagen que siempre se cargará será el fondo de las pantallas. Ahora se pasará a ver los diferentes valores de la variable “enablePiezasapp”:

- Valor 0
 - Cuando sea cero, quiere decir que se acaba de empezar el juego, por tanto, se mostrará la pantalla principal con el botón “Comenzar”.
- Valor distinto de 0
 - Cuando el valor sea distinto de 0, si es la primera vez que se va a la pantalla de la ruleta, se creará la banda en la que se encuentra el progreso del juego, y ya aparecerá durante toda la partida.
- Valor 1 o 7
 - Cuando el valor es 1 o 7 aparecerán los objetos que contienen la peana, donde descansará la ruleta o la mascota, y los avatares, con los nombres de los jugadores. Dependiendo de si la ruleta ha girado ya o no, se tendrán los siguientes 2 casos:
 - La ruleta no ha girado y no se tiene categoría elegida. Aparece el objeto de la ruleta con el botón que sirve para que empiece a jugar.
 - La ruleta ya ha girado. Aparece el objeto con la mascota. Encima de ella aparece el botón de jugar y más arriba el texto con la categoría escogida.
- Valor 4
 - Cuando el valor es 4 quiere decir que ya se ha pulsado el botón “Jugar”, por lo que en la pantalla aparecerán los objetos de pregunta y respuestas, con los correspondientes textos de la pregunta escogida. Además, se iniciará el cronómetro que controla el tiempo máximo de juego.
- Valores 8 a 11
 - Cuando el valor corresponde a un número entre 8 y 11 significa que se ha contestado a la pregunta. En ese caso, se para el cronómetro, para que no siga corriendo el tiempo, y se comprueba si se ha acertado o fallado la pregunta.
 - Si se ha acertado la pregunta, aparecerá el objeto de imagen correspondiente a la mascota con el texto “Correcto” y el botón continuar.

- Si se ha fallado la pregunta, aparecerá el objeto de imagen correspondiente a la mascota con el texto “Incorrecto” y el botón “Continuar”. Además, aparecerá la respuesta correcta debajo de la mascota.
- En cualquiera de los dos casos anteriores, se borrará la pregunta que ha tocado de la lista de pregunta, para que no vuelva a aparecer en la misma partida. Además, se reiniciará el valor de rotación de la ruleta.
 - Valor 12
 - Cuando el valor es 12 quiere decir que ya se ha pulsado el botón “Continuar” y pasará una de las siguientes opciones:
 - Si todavía no se ha llegado al límite de preguntas, se volverá a la pantalla de la ruleta.
 - Si ya se han contestado todas las preguntas del reto, aparecerá la pantalla modal con el resumen del reto.
- deleteImagenCargar()
 - Esta elimina del contenedor todos los objetos de imagen que haya y vuelve a llamar a la función “cargarPiezas()” para que lo rellene con los nuevos objetos de imagen.

4.4.1.4. App.js

En este fichero se encuentran las funciones básicas para crear una animación con las 3 clases que ya se han visto.

Las funciones disponibles son las siguientes:

- initializeapp()
 - En esta, lo primero que se hace es inicializar las variables necesarias para crear los objetos de las clases que ya se han visto. La más importante es “rutaPiezaIapp”, ya que ahí se pondrá la ruta de las imágenes que queremos que se carguen para luego cargarlas en memoria. La otra variable importante es “enablePiezasapp”, de donde se sacará en qué pantalla estamos.
 - Una vez hecho esto, se creará el “stage” de “CreateJS” donde irán los objetos de imagen.

- Luego, se creará el objeto del “Cargador”, que cargará en memoria todas las imágenes que hemos puesto en la variable “rutaPieza1app”.
- Por último, se cargarán las preguntas del juego. En esta primera versión, las preguntas se cargan desde varios ficheros JSON.
- ejecutaAppapp()
 - En esta se crea el contenedor sobre el que irán dispuestos los objetos de imagen.
- Recargarapp(vid)
 - Esta se ejecuta cada vez que se pulsa sobre un objeto que tenga asignado el evento “mousedown”.
 - En ella se pasa el identificador del objeto que se ha pulsado y se asigna a la variable “enablePiezasapp”.
 - Luego, se comprueba si la ruleta ha empezado a girar. Si no es así, se comprobará el identificador que se ha pasado como argumento y realizaremos una de las siguientes acciones:
 - Si se ha pulsado el objeto con identificador 1, significa que se va a hacer girar la ruleta, por tanto, se iniciará el contador que controla el tiempo de giro de la ruleta.
 - Si se ha pulsado alguno de los objetos con identificador del 8 al 11, significa que se ha seleccionado una de las respuestas y lo que se hará será remarcarla durante un segundo.
 - Si se ha pulsado cualquier otro objeto, lo que se hará será llamar a la función “deleteImagenCargar()” para borrar todos los objetos que haya en pantalla y pasar a la pantalla siguiente.
- tick()
 - Esta es la función asignada al evento “tick” del contador de vueltas de la ruleta. Lo primero que se hará será encender el sonido de la ruleta girando. Luego, si el valor del contador es menor que el tiempo máximo de giro, seguirá contando, si se ha llegado al tiempo máximo de giro, se desvinculará el evento “tick” del contador para que se pare. Por último, se calculará la categoría en la que se ha caído con la función “questionType(rotation)”.

Una vez descritos los ficheros principales, se describirán los ficheros auxiliares que se han creado.

4.4.1.5. `global.js`

En este fichero se encuentran todas las variables que se desea que sean globales, para poder usarse en cualquier función y en cualquier momento.

4.4.1.6. `functions.js`

En este fichero se encuentran todas las funciones necesarias para el correcto funcionamiento del juego. A continuación, se hará una breve descripción de las mismas.

- `questionType(rotation)`
 - Esta función tiene como entrada el valor de la rotación en la que se ha quedado la ruleta, y, a partir de esta, se calcula la categoría escogida.
- `randomRounds()`
 - Esta función devuelve un número aleatorio, que será el valor inicial de velocidad de giro de la ruleta.
- `createScreen(image, position)`
 - Esta función es una de las más importantes, ya que, a partir de una imagen y la posición que tomará en el canvas, como un vector $[x, y]$ (donde ‘x’ será el valor del eje horizontal, e ‘y’ será el valor del eje vertical), devuelve un objeto “CreateJS” que contendrá la imagen que se ha pasado como argumento y todas las funciones para manipular dicho objeto.
- `createTextContainer(params, question)`
 - Esta función se encarga de crear contenedores de texto, como los de los nombres de los participantes. Los argumentos de entrada son:
 - `params`: Se trata de un vector que contiene los datos del formato del texto, como puede ser el tipo de fuente, color de la fuente, tamaño de la fuente, tamaño del interlineado, etc.
 - `question`: Este parámetro es opcional, ya que sólo sirve si es para crear el contenedor que dice la categoría que nos ha tocado.
- `loadQuestionData(i)`
 - Esta función recoge los datos de las preguntas, respuestas y respuestas correctas del juego y los guarda en tres arrays diferentes (dependiendo del tipo de dato que sea). Los datos se recogen de un archivo en formato JSON con un formato específico. El parámetro de entrada se refiere al archivo del que debe coger las preguntas, respuestas y respuestas correctas.

- `loadQuestions()`
 - Esta función llama a la función anterior con los valores de los distintos archivos disponibles.
- `stringToInt(str)`
 - Esta función se usará para pasar el formato del cronómetro a un número entero que equivaldrá a los segundos. El parámetro de entrada será la cadena de texto que corresponde al valor del cronómetro.
- `toChrono(number)`
 - Esta función realiza la operación inversa a la de la anterior. El parámetro de entrada es el tiempo en segundos y devolverá una cadena de texto correspondiente al valor del cronómetro.
- `chronoEvent()`
 - Esta función se encarga de controlar el tiempo del cronómetro. Cuando se llegue al tiempo máximo, el juego terminará y aparecerá la ventana modal con el resumen del juego.
- `addChilds(childs)`
 - Esta función se usará para añadir varios objetos a la vez al contenedor de las imágenes. El parámetro de entrada será un array con todos los objetos que se van a añadir.
- `Header()`
 - Esta es la función que se encarga de crear la banda donde se encuentra el progreso del juego, con todos sus textos asociados.
- `removeHeader()`
 - Esta función elimina la banda de progreso del juego al terminar la partida, para que, si se vuelve a jugar, no se solape con la que se cree nueva.
- `updateAnswers(type)`
 - Esta función se encarga de cambiar el texto de las preguntas que se llevan contestadas, acertadas y falladas, en la banda de progreso del juego. El parámetro de entrada es si se ha acertado o fallado la pregunta.
- `calculateAnswerWidth(type, question, answer)`
 - Esta función sirve para controlar el tamaño de las respuestas. Según el tamaño que tenga, se pondrá en una posición o en otra en el botón de respuesta. Los parámetros de entrada son:

- type: la categoría de la pregunta.
- question: el número de la pregunta.
- answer: el número de la respuesta.
- getRandomQuestion(type)
 - Esta función devuelve un número de pregunta elegido de manera aleatoria de entre el número máximo de pregunta que haya en la categoría definida por el parámetro de entrada.
- resetVariables()
 - Esta función reinicia los valores de las variables críticas del juego al terminar la partida, para que, si se vuelve a jugar, no se solapen con los nuevos valores y tengamos datos erróneos del juego.
- getAvatars(left, right)
 - Esta función dispone en pantalla los avatares de los jugadores. Los parámetros de entrada definen si es el avatar de la derecha o de la izquierda, y si el avatar de un hombre o de una mujer.
- removeAvatars(scope)
 - Esta función elimina los objetos de los avatares de la pantalla.

Con esto, se termina la primera fase de desarrollo del juego, en la cual, ya se tiene una aplicación totalmente funcional, pero que sólo servirá para exhibirla como una demostración, ya que no realiza emparejamiento ni ninguna conexión con el servidor.

4.4.2. Fase 2: Comunicación con el servidor

En la fase anterior se decidió no realizar comunicaciones con el servidor para simplificar el problema. En la fase actual, se van a implementar con el fin de que el juego se convierta en lo que se quería desde el principio: un juego en el que se crean retos entre los participantes. Los retos se grabarán en la base de datos, para que el jugador retado pueda recuperarlo más tarde, siempre que no se haya pasado de la fecha máxima para responderlo, que en este caso son 2 días. Con los datos de los retos que hay en la base de datos se generará el ranking en tiempo real de la plataforma UPCTplay que podrán ver tanto los alumnos como el docente.

Esta fase se realizará en el lenguaje “*PHP*”, ya que el cliente hará interacciones con el servidor y la base de datos. Antes, se deberán hacer algunos cambios en los ficheros “*Javascript*” para poder llamar a los nuevos ficheros.

4.4.2.1. Cambios realizados a los ficheros “Javascript”

Los cambios que se realizarán serán los siguientes:

- App.js
 - En este fichero se deberán realizar 2 cambios a la función “Recargarapp(vid)”.
 - El primero será añadir el control de si somos la persona que va a retar. Si es así, mandaremos una petición “AJAX” al fichero *find_opponent.php* para crear un reto.
 - El segundo será añadir otro control, para saber si se es la persona retada. En este caso, se mandará una petición “AJAX” al fichero *retrieve_challenge.php* para recoger los datos del reto.
- ContenedorImágenes.js
 - En este fichero se deberá añadir el código para que se actualice el estado del reto al terminar de jugar. Esto se hará con una petición “AJAX” al fichero *update_challenge_state.php*
- functions.js
 - En este fichero, se cambiará la función “updateAnswers(type)” para que se registre la respuesta del usuario en la base de datos, para poder actualizar, posteriormente, el estado del mismo. Esto se hará con una petición “AJAX” al fichero *update_question.php*.

Ahora se describirán los ficheros “PHP”.

4.4.2.2. dbconnect.php

En este fichero se encuentra la función que permitirá conectar a la base de datos del servidor. Esta conexión se realiza con la extensión “mysqli” de PHP, ya que usa un entorno orientado a objetos más seguro que su predecesora “mysql”.

Si no se puede conectar a la base de datos, se mostrará un error en pantalla.

4.4.2.3. find_opponent.php

En este fichero se encuentran los pasos necesarios para realizar un reto.

En primer lugar, se comprobará que se tienen los datos necesarios para poder realizar un reto.

Los datos que se necesitan en este caso son:

- dni: El dni del usuario que va a realizar el reto.
- actividad: El identificador de la actividad en la que se va a realizar dicho reto.

Si no se tiene alguno de estos datos, se mostrará un mensaje en pantalla y no se dejará al usuario que juegue.

Una vez realizadas estas comprobaciones, se dispondrá la conexión con la base de datos, para poder empezar a realizar consultas.

La primera consulta que se hará será comprobar si el usuario tiene vidas para poder retar. Se realizará con la siguiente consulta:

```
SELECT `vidas` FROM `participantes` WHERE `dni`='$dni' AND
`actividad`='$actividad'
```

Si el usuario dispone de vidas para realizar retos, se dará el siguiente paso, que será recoger los datos del usuario que está retando, sino dispone de vidas, se mostrará un mensaje diciendo que no puede realizar más retos.

```
SELECT `nombre`, `apellidos`, `movil`, `genero` FROM alumnos WHERE dni='$dni'
```

Estos datos se guardarán en una variable para poder usarlos posteriormente.

El siguiente paso será escoger el oponente. Esta acción se realiza en varios pasos:

- Primero se seleccionarán las personas con las que todavía no ha jugado de entre todos los participantes.

```
SELECT `dni` FROM `participantes` WHERE `actividad`='$actividad' AND `vidas`>0
AND `dni` NOT IN (SELECT `dni2` FROM `retos` WHERE `actividad`='$actividad'
AND `dni1`='$dni') AND `dni` NOT IN (SELECT `dni1` FROM `retos` WHERE
`actividad`='$actividad' AND `dni2`='$dni') AND `dni` <> '$dni'
```

Como se puede observar, esta consulta es algo más complicada que las demás debido a que se tiene que comprobar a quien se ha retado ya, para descartarlo.

- El siguiente paso será guardar a los participantes que son candidatos a ser el oponente en un array.
- El último paso será escoger uno de los participantes del array para ser el oponente. Esto se hará escogiendo un número aleatorio con la función “*mt_srand(\$seed)*” de PHP, que

crea número aleatorios fuertes a partir de una semilla.

Tras escoger un oponente, se recogerán sus datos con otra consulta a la base de datos.

```
SELECT `dni`, `nombre`, `apellidos`, `email`, `movil`, `genero` FROM `alumnos`
WHERE `dni`='$challenge'
```

Luego, se insertarán los datos del reto en la base de datos y se restará una vida a cada oponente con las siguientes consultas:

```
INSERT INTO `retos`(`id`, `actividad`, `dni1`, `dni2`, `f_inicio`, `f_fin`) VALUES
(NULL,$actividad,$dni,$dni_challenged_person,NOW(),DATE_ADD(NOW(),INTERVAL 2 DAY))
```

- Para crear el reto con la fecha de inicio hoy, y la fecha de fin dentro de 2 días, para que pasados 2 días no se pueda contestar el reto.

```
UPDATE `participantes` SET `vidas`=`vidas`-1 WHERE `dni`='$dni' AND
`actividad`=$actividad
```

```
UPDATE `participantes` SET `vidas`=`vidas`-1 WHERE
`dni`='$dni_challenged_person' AND `actividad`=$actividad
```

- Para restar las vidas a ambos jugadores.

Por último, pasaremos los datos de ambos jugadores a la aplicación creando una cadena de texto en formato JSON.

```
echo '{"id_reto":'.$id.', "tipo":2, "vidas":'.$lives
['vidas'].',';
echo "persona1":{"dni":'.$dni.', "nombre":'.$person1
['nombre'].', "apellidos":'.$person1['apellidos'].', "movil": '.$person1
['code'].$person1['movil'].', "genero":'.$person1['genero'].'},';
echo "persona2":{"dni":'.$dni_challenged_person.',
"nombre":'.$person2['nombre'].', "apellidos":'.$person2
['apellidos'].', "movil": '.$person2['code'].$person2['movil'].',
"genero":'.$person2['genero'].', "mail":'.$person2
['email'].', "instituto":'.$tipo.' }';
echo '}';
```

Figura 4.20. Código para pasar el objeto JSON a la aplicación (1).

4.4.2.4. retrieve_challenge.php

En este fichero se encuentran los pasos necesarios para responder a un reto.

En primer lugar, se comprobará que tenemos los datos necesarios para poder recuperar el reto. Los datos que se necesitan en este caso son:

- dni: El dni del usuario que va a responder el reto.
- actividad: El identificador de la actividad de la que procede dicho reto.
- reto: El identificador del reto que realizó el oponente.

Si no se tiene alguno de estos datos, se mostrará un mensaje en pantalla y no se dejará al usuario que juegue, ya que no se podrán recuperar los datos del reto.

Una vez realizadas estas comprobaciones, se dispondrá la conexión con la base de datos, para poder empezar a realizar consultas.

La primera consulta que se realizará será comprobar si, efectivamente, el identificador del reto corresponde al usuario en cuestión, y si no se ha contestado aún, ya que, si ya se había contestado, no se podrá volver a jugar. Además, se recuperará el dni del oponente:

```
SELECT * FROM retos WHERE id=$id_reto AND dni2='$dni2' AND actividad=$actividad AND estado2=-1
```

Ahora que ya se tiene el dni del oponente, se recogerá a la información de ambos participantes, para que aparezca el nombre en pantalla.

```
SELECT * FROM alumnos WHERE dni='$dni1'
```

```
SELECT * FROM alumnos WHERE dni='$dni2'
```

También se recuperarán las vidas, para mostrarlas en pantalla:

```
SELECT vidas FROM participantes"." WHERE dni='$dni2' AND actividad=$actividad
```

Luego se actualizará el estado del reto, para señalar que el jugador retado ya ha contestado el reto una vez, y no pueda volver a hacerlo. Será con la siguiente consulta:

```
UPDATE retos SET estado2=0 WHERE id=$id_reto AND dni2='$dni2' AND actividad=$actividad
```

Por último, se pasarán los datos de ambos jugadores a la aplicación creando una cadena de texto en formato JSON.

```

echo '{"id_reto":' . $id_reto . ', "tipo": "1", "vidas": ' . $lives
['vidas'] . ', ' . '};
echo "persona1":{"dni":"' . $person1['dni'] . '",
"nombre":"' . $person1['nombre'] . '", "apellidos":"' . $person1
['apellidos'] . '", "movil": "' . $person1['movil'] . '", "genero":"' . $person1
['genero'] . '"}, ' . '};
echo "persona2":{"dni":"' . $dni2 . '", "nombre":"' . $person2
['nombre'] . '", "apellidos":"' . $person2['apellidos'] . '", "movil": "' . $person2
['movil'] . '", "genero":"' . $person2['genero'] . '"}';
echo '}}';

```

Figura 4.21. Código para pasar el objeto JSON a la aplicación (2).

4.4.2.5. update_challenge_state

Este fichero contiene la función para actualizar el estado del reto conforme los participantes están jugando.

Los datos que se necesitan en este caso, para que se ejecute, son:

- actividad: Identificador de la actividad en la que se quiere actualizar el resultado.
- id_reto: El identificador del reto en cuestión.

Una vez realizadas estas comprobaciones, se dispondrá la conexión con la base de datos, para poder empezar a realizar consultas.

La acción que realiza esta función es la siguiente:

- Primero, recoge los datos de las preguntas que ha acertado y fallado cada jugador con las siguientes consultas:


```

SELECT acertadas_1, falladas_1 FROM retos WHERE id=$id_reto AND
actividad=$actividad
SELECT acertadas_2, falladas_2 FROM retos WHERE id=$id_reto AND
actividad=$actividad

```
- Luego, decide qué estado poner según el siguiente criterio:
 - Si el jugador 2 no ha contestado ninguna pregunta aún, da como ganador al jugador 1, aunque no haya acertado ninguna pregunta.
 - Si el jugador 2 sí ha contestado alguna pregunta:
 - Si el jugador 1 ha acertado más preguntas que el jugador 2, da por ganador al jugador 1.
 - Si el jugador 2 ha acertado más preguntas que el jugador 1, da por

ganador al jugador 2.

- Si ambos jugadores han acertado las mismas preguntas, se produce un empate.
- Por último, se actualiza el estado del reto a una de las opciones ya mencionadas con la siguiente consulta.

```
UPDATE `retos`.`retos` SET `estado`=$estado WHERE id=$id_reto AND actividad=$actividad
```

4.4.2.6. update_question

Este fichero contiene la función para actualizar el número de preguntas acertadas y falladas de cada jugador.

Los datos que se necesitan en este caso, para que se ejecute, son:

- actividad: Identificador de la actividad en la que se quiere actualizar.
- id_reto: El identificador del reto en cuestión.
- user: El jugador que ha contestado la pregunta.
- type: Si el jugador ha acertado o fallado la pregunta.

Una vez realizadas estas comprobaciones, se dispondrá la conexión con la base de datos, para poder empezar a realizar consultas.

Con datos que se han recibido, se crean las variables necesarias para componer la consulta a la base de datos. En este caso se necesitarán crear las siguientes:

- \$user: En esta variable se pondrá 'dni1' o 'dni2' según qué jugador haya contestado la pregunta.
- \$response: En esta variable se pondrá 'acertadas_1' o 'falladas_1' si ha respondido el jugador 1, o 'acertadas_2' o 'falladas_2' si ha respondido el jugador 2.

Con esto, se realizará la consulta. En este caso será una actualización del valor de la columna que marque la variable \$response:

```
UPDATE `retos`.`retos` SET ".$response."=".$response."+1 WHERE id=$id_reto AND `actividad`=".$activity." AND ".$user."=".$dni."
```

Por último, se llamará a la función de actualizar el estado del reto. Esto se hace por si alguno

de los dos jugadores no llegara a terminar la partida, sea cual sea el motivo.

```
require_once 'update_challenge_state.php';
update_challenge_state($activity, $id_reto);
```

Figura 4.22. Llamada a una función de un archivo externo.

4.4.3. Fase 3: Implementación de la comunicación mediante mensajería

La implementación de la comunicación mediante mensajería era uno de los requisitos más importantes a la hora de plantear el proyecto, ya que, para enganchar a los participantes, se necesitaba un medio de comunicación con ellos.

4.4.3.1. Servicio de mensajería instantánea: Whatsapp

La primera opción que se barajó fue la implementación del servicio de mensajería instantánea “*Whatsapp*”. Tras diversos problemas que acarrea el uso de este servicio, se decidió descartar su uso.

4.4.3.2. Servicio de mensajería instantánea: Telegram

La siguiente opción fue el uso del servicio de mensajería instantánea “*Telegram*”. Esta opción fue la primera que se implementó, ya que “*Telegram*” dispone de diversos clientes para instalar en cualquier plataforma. En nuestro caso, se instaló el cliente para servidores “*Telegram-cli*” que permite el uso de todas las funciones de “*Telegram*” desde la línea de comandos de *Linux*.

Para su uso desde el juego se creó un fichero en el lenguaje “*BASH*”, que es el lenguaje de la consola de *Linux*, y otro fichero “*PHP*”, desde el que se llamará al anterior.

4.4.3.2.1. send_message.sh

Este fichero contiene la llamada a la función que permite mandar mensajes de “*Telegram*” a un contacto de la agenda. El contenido del archivo es el siguiente:

```
1  #!/bin/bash
2  telegram-cli -W -e "msg $1 $2"
```

Figura 4.23. Llamada a la función que manda mensajes de “*Telegram*”.

En la primera línea, se define que el fichero es un script de “*BASH*”.

En la segunda línea, se llama al cliente *telegram-cli* con las siguientes opciones:

- -W: Ejecuta el cliente en modo administrador.
- -e: Ejecuta el cliente, realiza la acción y vuelve a cerrar el cliente.

En esta última línea, vemos también la función del cliente que se usa: “*msg*”. Esta función admite 2 parámetros, que son el nombre del contacto al que vamos a mandar el mensaje, y el mensaje en cuestión que vamos a enviar.

Por tanto, se tiene una función con 2 parámetros de entrada:

- \$1: Será el nombre del contacto al que se va a mandar el mensaje.
- \$2: Será el mensaje que se va a enviar al contacto.

4.4.3.2.2. `send_message.php`

En este fichero nos encontramos con 2 funciones:

1. `sendTelegram($cod_pais, $num_retado, $retado, $retador, $ffin)`

- Esta función sirve para enviar al contrincante un mensaje de que ha sido retado por un jugador, y que tiene dos días para contestar al reto. Los parámetros de entrada son los siguientes:
 - a) `$cod_pais`: El código de país del número de teléfono de la persona que se ha retado.
 - b) `$num_retado`: El número de teléfono de la persona que se ha retado.
 - c) `$retado`: El nombre de la persona que se ha retado.
 - d) `$retador`: El nombre de la persona que ha realizado el reto.
 - e) `$ffin`: La fecha en la que terminará el reto.
- El nombre de contacto se compondrá con el siguiente formato: `codpais_numtelefono`. Un ejemplo sería `+34_600000000`.
- El mensaje será el siguiente:


```
¡".$retado.", has sido retad@! Tu compañer@ ".$retador." te ha desafiado,
¿serás capaz de superarlo? Entra a play.upct.es para responder al reto. Tienes
hasta ".$ffin." para responder. Mucha suerte.
```
- Por último, se ejecutará la siguiente función de “*PHP*”:
 - a) `exec('sudo -u ubuntu telegram/bash-script/send_message.sh "$num_retado." "$message."');`

- Esta función permite que se ejecute un script que se tiene en el servidor. Como se está haciendo desde un servidor web, se debe dar privilegios de administrador. Esto se hará con la instrucción: `sudo -u ubuntu`.

2. `sendTelegramWin($cod_pais, $num_retado, $retado, $retador, $ganador)`

- Esta función es parecida a la anterior. Se diferencia de ella únicamente en el mensaje que se envía.
- En este caso, se mandará el mensaje a la persona que haya contestado el reto en primer lugar, para informarle del resultado.
- La variable `$ganador` nos dirá quién es el ganador del reto y se generará uno de los siguientes textos según el valor de la misma:
 - ❖ Si el valor es 0, significa que han empatado el reto.
 - ❖ Si el valor es 1, significa que la persona a la que se le va a enviar el mensaje es la ganadora.
 - ❖ Si el valor es 2, significa que la persona a la que se le va a enviar el mensaje ha perdido.
- El mensaje será el siguiente:
`¡".$retado.", has ".$men." contra tu compañer@ ".$retador."!`
- Por último, se ejecutará la misma función que en la función anterior.

4.4.3.3. Servicio de mensajería: Correo electrónico

Tras realizar una prueba del juego en la asignatura “Gestión de los Recursos Humanos” del Grado en Administración y Dirección de Empresas, se comprobó que no todos los participantes sabían lo que era “*Telegram*” y otros no querían instalar otra aplicación de mensajería instantánea, por lo tanto, se decidió que se debería habilitar otra vía de comunicación con los participantes.

Cómo todos los participantes tenían correo electrónico, se decidió que ésta sería la opción más asequible de implementar.

La implementación del servicio de envío de correos se realizará en el lenguaje “*PHP*” a través de una clase de código abierto llamada “*PHPMailer*”. Esta clase permite el uso del protocolo *SMTP* (Simple Mail Transfer Protocol) sobre un protocolo seguro (en ese caso *TLS*, Transport Layer Security), y también permite el envío del texto en formato *HTML*, por lo que se podrá enviar el texto en un formato más estructurado.

Para usar esta clase se seguirán los siguientes pasos:

1. Se creará un objeto PHPMailer.
2. Se modificará la configuración de las variables, para usar el protocolo seguro de envío, y para usar la cuenta de correo que se quiera.
3. Se añadirá el destinatario del correo, el idioma en el que se está enviando, y se dirá que irá en formato *HTML*.
4. Se añadirá el contenido del mensaje en el formato *HTML* y, además, se enviará sin formato, por si el destinatario no puede leerlo en ese formato.
5. Por último, se enviará el mensaje.

Para el envío de correos a través del juego, se creará un archivo “*PHP*”.

4.4.3.3.1. `send_mail.php`

En este fichero se encontrarán 2 funciones. Estas funciones son equivalentes a las del envío del mensaje a través de “*Telegram*”:

1. `sendMail($mail_retado, $retado, $retador, $ffin)`
 - Esta función sirve para enviar al contrincante un mensaje de que ha sido retado por un jugador, y que tiene dos días para contestar al reto. Los parámetros de entrada son los siguientes:
 - ❖ `$mail_retado`: El correo electrónico de la persona que se ha retado.
 - ❖ `$retado`: El nombre de la persona que se ha retado.
 - ❖ `$retador`: El nombre de la persona que ha realizado el reto.
 - ❖ `$ffin`: La fecha en la que terminará el reto.
 - El mensaje será el siguiente:

```
<h1>Hola, '.$retado.'</h1><p>¡Has sido retad@ por tu compañer@
'.$retador.'! ¿Serás capaz de superarlo?</p><p>Entra a <a
href="http://play.upct.es">play.upct.es</a> para responder al reto. Tienes hasta
<strong>'.$ffin.'</strong> para responder. Mucha suerte.</p>
```
2. `sendMailWin($mail_retado, $retado, $retador, $ganador)`
 - Esta función se diferencia de la anterior en el mensaje que se envía, al igual que su análoga de “*Telegram*”.
 - En este caso, se mandará el mensaje a la persona que haya contestado el reto en

primer lugar, para informarle del resultado.

- La variable \$ganador nos dirá quién es el ganador del reto y se generará uno de los siguientes textos según el valor de la misma:
 - ❖ Si el valor es 0, significa que han empatado el reto.
 - ❖ Si el valor es 1, significa que la persona a la que se le va a enviar el mensaje es la ganadora.
 - ❖ Si el valor es 2, significa que la persona a la que se le va a enviar el mensaje ha perdido.
- El mensaje será el siguiente:


```
<h1>Hola, '.$retado.'</h1><p>¡Has '.$men.' contra tu compañer@
'.$retador.'!</p>
```

4.4.3.4. Cambios en los ficheros “PHP”

Al crear estas nuevas funciones, se deben realizar varios cambios en los ficheros que ya se tenían. Los cambios son los siguientes:

4.4.3.4.1. find_opponent.php

En este fichero, se deberá llamar a las nuevas funciones para poder mensajes al oponente.

Se hará de la siguiente manera:

```
if ($person2['telegram'] == 1) {
    require_once 'telegram/send_message.php';
    sendTelegram($person2['code'],$person2['movil'],
    $person2['nombre'], $person1['nombre'], $ffin);
}
```

Figura 4.24. Llamada a la función externa para mandar un mensaje de “Telegram”.

En la imagen se puede observar que primero se comprueba que el participante quiere recibir mensajes a través de “Telegram”, y si es así, se llamará a la función para enviarle el mensaje de que ha sido retado.

Lo mismo ocurre con el mensaje de correo electrónico:


```

if ($person2['correo'] == 1) {
    require_once 'mail/send_mail.php';
    sendMail($person2['email'], $person2['nombre'], $person1
    ['nombre'], $ffin);
}

```

Figura 4.25. Llamada a la función externa para mandar un correo electrónico.

En la imagen se puede observar cómo se realiza la comprobación de si quiere recibir correos electrónicos, y si es así, se llamará a la función para informarle de que ha sido retado.

4.4.3.4.2. update_challenge_state.php

En este fichero, también se deberá llamar a las nuevas funciones para poder mensajes, pero en este caso, se enviará al jugador que haya terminado de jugar en primer lugar.

Lo que se hará será lo siguiente:

1. En primer lugar, se comprobará que los 2 jugadores han contestado todas las preguntas.
2. Después, se recogerán los datos de los 2 participantes, para enviar el mensaje (“Telegram” o correo) al que haya terminado en primer lugar. Los datos se recogerán con las siguientes consultas:

```

SELECT `nombre`, `code`, `movil`, `email`, `telegram`, `correo` FROM `alumnos`.`"
WHERE `dni`=(SELECT `dni1` FROM retos"." WHERE id=$id_reto AND
actividad=$actividad)

```

```

SELECT `nombre`, `code`, `movil`, `email`, `telegram`, `correo` FROM `alumnos`.`"
WHERE `dni`=(SELECT `dni2` FROM retos"." WHERE id=$id_reto AND
actividad=$actividad)

```

3. Después, se harán las comprobaciones de quién es la primera persona que ha terminado, y si quiere que se le envíe “Telegram” y/o correo. Si no tiene las opciones de envío activadas, no se hará nada más. Si sí las tiene activadas, se comprobará quién ha ganado, para mandarlo a través de las funciones. Los códigos serán los siguientes:

```

if ($opponent == 2 && $info2['telegram'] == 1) {
    require_once 'telegram/send_message.php';
    if ($estado == 1) {
        $ganador = 2;
    } elseif ($estado == 2) {
        $ganador = 1;
    } else {
        $ganador = 0;
    }
    sendTelegramWin($info2['code'],$info2['movil'], $info2['nombre']
, $info1['nombre'], $ganador);
} elseif ($opponent == 1 && $info1['telegram'] == 1) {
    require_once 'telegram/send_message.php';
    if ($estado == 1) {
        $ganador = 1;
    } elseif ($estado == 2) {
        $ganador = 2;
    } else {
        $ganador = 0;
    }
    sendTelegramWin($info1['code'],$info1['movil'], $info1['nombre']
, $info2['nombre'], $ganador);
}

```

Figura 4.26. Llamada a la función externa para mandar un “Telegram” al jugador que terminó primero el reto, con el cálculo del resultado.

```

if ($opponent == 2 && $info2['correo'] == 1) {
    require_once 'mail/send_mail.php';
    if ($estado == 1) {
        $ganador = 2;
    } elseif ($estado == 1) {
        $ganador = 1;
    } else {
        $ganador = 0;
    }
    sendMailWin($info2['email'], $info2['nombre'], $info1['nombre'],
        $ganador);
} elseif ($opponent == 1 && $info1['correo'] == 1) {
    require_once 'mail/send_mail.php';
    if ($estado == 1) {
        $ganador = 1;
    } elseif ($estado == 2) {
        $ganador = 2;
    } else {
        $ganador = 0;
    }
    sendMailWin($info1['email'], $info1['nombre'], $info2['nombre'],
        $ganador);
}

```

Figura 4.27. Llamada a la función externa para mandar un correo electrónico al jugador que terminó primero el reto, con el cálculo del resultado.

4.4.4. Fase 4: Implementación del gestor de bancos de preguntas en la plataforma UPCTplay

Durante el trascurso de la prueba del juego, en la asignatura “Gestión de los Recursos Humanos” del Grado en Administración y Dirección de Empresas, se comprobó que se tenía un problema: al tener las preguntas en archivos “JSON”, si se quería crear una actividad diferente a la que se estaba cursando, se tenía que crear una copia completa del juego para poder tener preguntas distintas.

La solución que más efectiva fue el desarrollo de un gestor de bancos de preguntas en la plataforma UPCTplay. Cada profesor podría entrar en dicha plataforma para grabar las preguntas que posteriormente fuese a utilizar en sus actividades.

El funcionamiento debía ser muy sencillo y se llegó a los siguientes requisitos:

- Debería existir una pantalla en la que se pudieran escribir las preguntas con sus respuestas asociadas, y la respuesta correcta de la misma, además de la categoría de la

pregunta. Como elemento adicional, se decidió añadir un campo de palabras clave, para que luego se pudieran encontrar de una manera más rápida.

- Debería existir una pantalla en la que aparecieran todas las preguntas asociadas al profesor, y que se pudieran modificar.
- Debería existir una pantalla en la que se pudieran asignar las preguntas a una actividad que no hubiera comenzado aún.

Con estas directrices se va a explicar cada una de las funciones del gestor de bancos de preguntas.

4.4.4.1. Pantalla de creación de preguntas

Esta pantalla tiene el siguiente aspecto:

The screenshot displays a web form for creating questions. It includes the following elements:

- Categoría:** A text input field containing "Asignatura, temática, etc".
- Enunciado de la pregunta:** A large text area for the question text.
- Respuestas:** Three separate text areas for "Respuesta 1", "Respuesta 2", and "Respuesta 3".
- Buttons:** Two buttons below the answers: a green "➕ AÑADIR RESPUESTA" and a red "➖ ELIMINAR RESPUESTA".
- Número de la respuesta correcta:** A dropdown menu currently set to "1".
- Palabras clave:** A text area with the placeholder "Añade aquí algunas palabras clave para la indexación, separadas por comas".
- Footer Buttons:** A row of four buttons: "AÑADIR MÁS PREGUNTAS" (green), "BORRAR ÚLTIMA PREGUNTA" (red), "ENVIAR PREGUNTAS" (blue), and "LIMPIAR FORMULARIO" (orange).

Figura 4.28. Diseño de la pantalla de creación de preguntas.

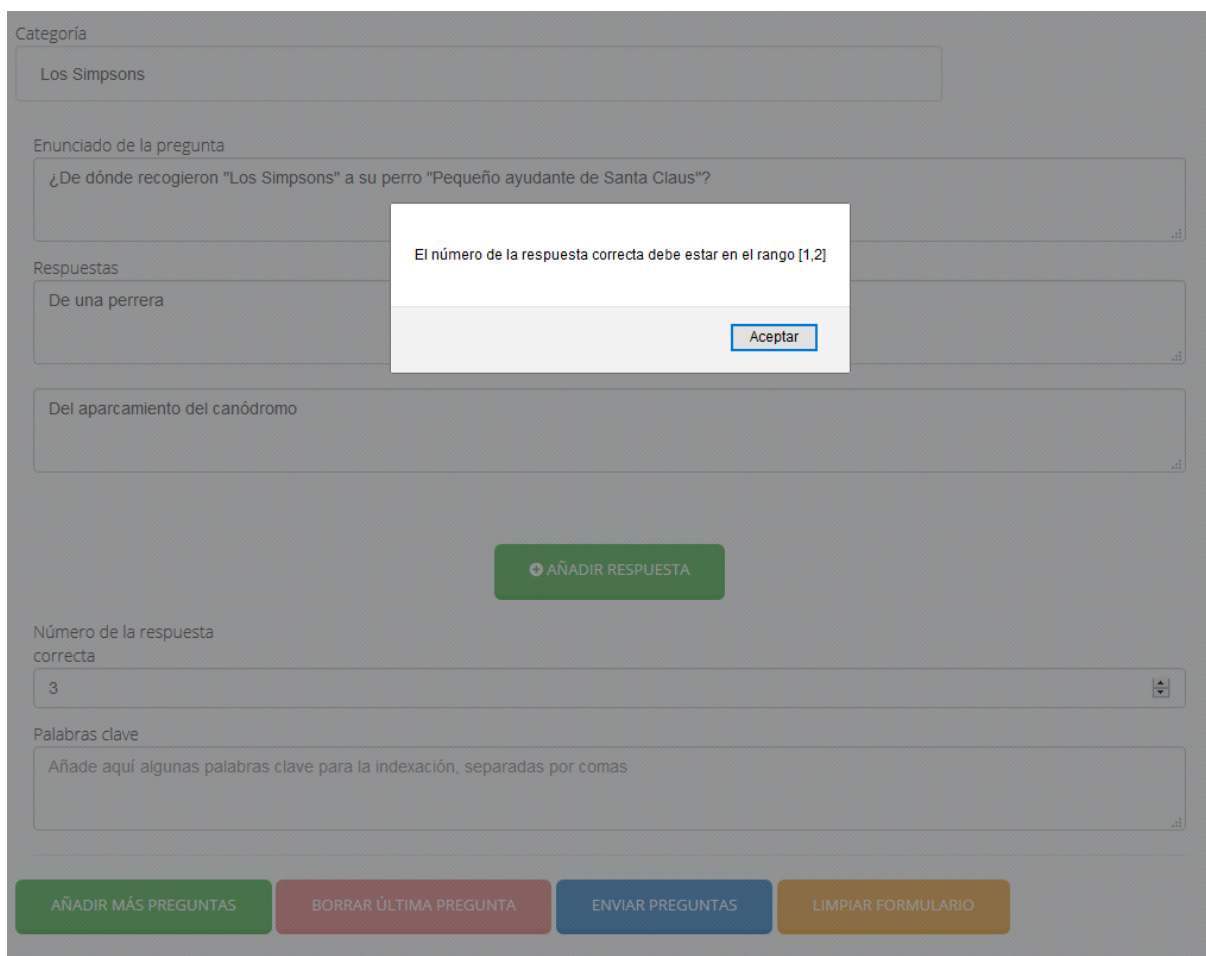
En ella podemos observar que se trata de un formulario con varios campos y varios botones.

Los campos son los siguientes:

- Categoría: Se trata de la categoría a la que se asignará la pregunta.
- Enunciado: Se trata del enunciado de la pregunta.
- Respuestas: Se trata de las respuestas que se quiera que tenga nuestra pregunta. Se podrá crear entre 2 y 4 respuestas.
- Número de la respuesta correcta: Se trata del número de la respuesta correcta.
- Palabras clave: Se trata de un campo de texto en el que se podrán poner palabras clave para que sea más fácil buscarlas más tarde.

En los campos: enunciado y respuestas, se tendrá un límite de 140 caracteres, para que la pregunta quepa correctamente en el juego.

En el campo “Número de la respuesta correcta” no se podrá poner un número mayor que el número de campos de respuesta que tengamos. Si se hace, saldrá un aviso como el siguiente:



The image shows a web form with several fields: 'Categoría' (Los Simpsons), 'Enunciado de la pregunta' (¿De dónde recogieron "Los Simpsons" a su perro "Pequeño ayudante de Santa Claus"?), 'Respuestas' (De una perrera, Del aparcamiento del canódromo), 'Número de la respuesta correcta' (3), and 'Palabras clave' (Añade aquí algunas palabras clave para la indexación, separadas por comas). A modal alert box is displayed in the center, containing the text 'El número de la respuesta correcta debe estar en el rango [1,2]' and an 'Aceptar' button. At the bottom of the form, there are four buttons: 'AÑADIR MÁS PREGUNTAS', 'BORRAR ÚLTIMA PREGUNTA', 'ENVIAR PREGUNTAS', and 'LIMPIAR FORMULARIO'.

Figura 4.29. Mensaje de alerta al pasarnos en el número de la respuesta correcta.

Los botones son los siguientes:

- Añadir respuesta: Crea un nuevo campo, para que se pueda añadir una respuesta adicional. Sólo aparece si se tienen menos de 4 respuestas.
- Eliminar respuesta: Elimina el último campo de respuesta. Sólo aparece si se tienen más de dos respuestas.
- Añadir más preguntas: Crea un contenedor nuevo con los campos: enunciado, respuestas, número de la respuesta correcta y palabras clave, para que se añadan más preguntas a la misma categoría.
- Borrar última pregunta: Elimina el último contenedor de preguntas creado. Sólo aparece si se tiene más de una pregunta.
- Enviar preguntas: Envía el formulario para ser procesado.
- Limpiar formulario: Vacía todos los campos del formulario.

Todos estos botones tienen asignados eventos “*Javascript*” para controlar su funcionamiento. En el archivo *funciones.js* se tienen las funciones que controlan los eventos de dichos botones. El control de los eventos se hará a través de *jQuery*.

Al pulsar el botón “Enviar preguntas”, todos los datos de los campos de texto se mandarán al archivo *procesa.php*, en el cual se realizarán las siguientes acciones:

1. Primero se comprobará que han llegado datos.
2. Luego se dispondrá la conexión con la base de datos, para poder empezar a realizar las consultas necesarias.
3. Lo siguiente será crear una cadena de texto que servirá como patrón para realizar las consultas: `INSERT INTO `bancopreguntas`(`profesor`, `categoria`, `enunciado`, `respuesta1`, `respuesta2`, `respuesta3`, `respuesta4`, `respuestacorrecta`, `comentarios`) VALUES(%s, %s, %s, %s, %s, %s, %s, %u, %s)`. En esta cadena se puede observar que los parámetros que insertaremos se cambiarán por “%s” y “%u”, que equivalen a decir que ahí irá una cadena de texto y un valor entero respectivamente.
4. Por último, se irán insertando las preguntas que haya grabado el profesor. Este proceso se hará a través de un bucle, por si hay más de una pregunta para guardar. Al guardar las preguntas en la base de datos, se tendrán que codificar algunos de los caracteres como pueden ser las comillas simples o dobles, para que no haya ningún error en la consulta. Esto se realizará con la función “*real_escape_string(\$string)*” de la extensión

de “PHP” “mysql”.

Cuando la pregunta se envíe, se mostrará una ventana modal que dirá si se han subido correctamente o si ha habido algún error. La pantalla será como la siguiente:

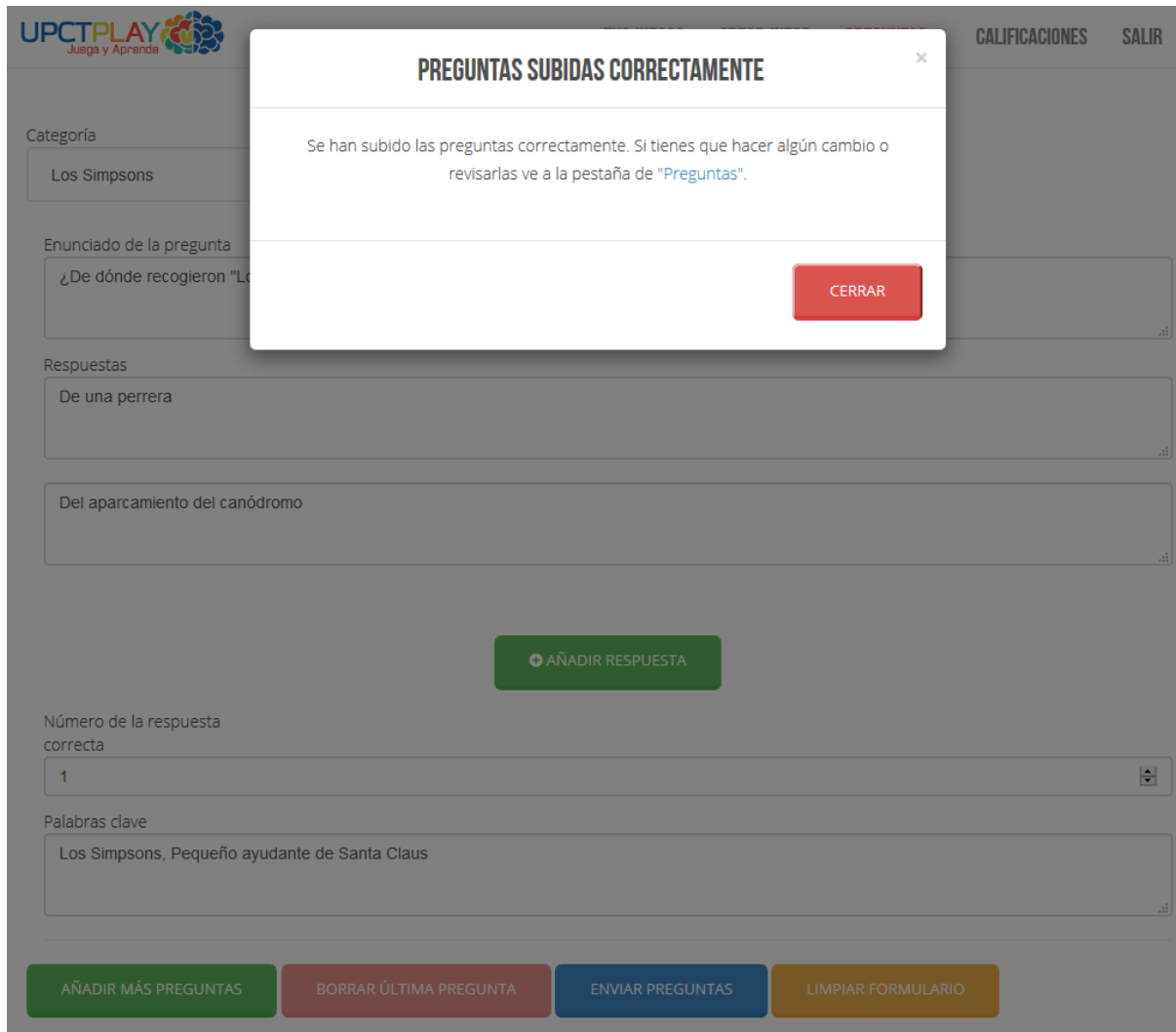


Figura 4.30. Pantalla modal indicando que todas las preguntas se han subido correctamente.

4.4.4.2. Pantalla de visionado y modificación de preguntas.

Esta pantalla tiene el siguiente aspecto:

The screenshot shows a question management interface with the following elements:

- Categoría:** A text input field containing "Los simpsons".
- Enunciado:** A text input field containing "¿De dónde recogieron "Los Simpsons" a su perro "Pequeño ayudante de Santa Claus"?"
- Respuesta 1:** A text input field containing "De una perrera".
- Respuesta 2:** A text input field containing "Del aparcamiento del canódromo".
- Botón:** A green button labeled "AÑADIR RESPUESTA" with a plus icon.
- Número de la respuesta correcta:** A dropdown menu showing "1".
- Palabras clave:** A text input field containing "Los Simpsons, Pequeño ayudante de Santa Claus".
- Botones de acción:** Two red buttons labeled "EDITAR" and "ELIMINAR PREGUNTA".
- Navegación:** Blue left and right arrow icons flanking the question card.
- Paginación:** A "Slide 2" indicator with a dropdown arrow and "- 2" next to it.

Figura 4.31. Diseño de la pantalla para visualizar y modificar preguntas.

Como se puede observar, las preguntas se muestran en un carrusel. Hemos elegido este elemento para no sobrecargar la página de contenido y hacer que el profesor sólo vea una pregunta en cada momento. Para pasar de una pregunta a otra se podrá hacer con las flechas o a través del paginado implementado en la zona inferior.

En la pantalla se puede ver que están todos los campos desactivados. Para activarlos deberemos pulsar el botón "Editar".

Categoría
Los simpsons

Enunciado
¿De dónde recogieron "Los Simpsons" a su perro "Pequeño ayudante de Santa Claus"?

Respuesta 1
De una perrera

Respuesta 2
Del aparcamiento del canódromo

+ AÑADIR RESPUESTA

Número de la respuesta correcta
1

Palabras clave
Los Simpsons, Pequeño ayudante de Santa Claus

DEJAR DE EDITAR ELIMINAR PREGUNTA

Slide 2 - 2

Figura 4.32. Vista de la pantalla cuando nos disponemos a editar una pregunta.

En la figura se puede observar que todos los campos se han activado para poder editarlos, y el botón “Editar” ha cambiado su nombre a “Dejar de editar”. Cuando se pulse este botón, se guardarán todos los cambios que se hayan hecho.

Por último, se tiene el botón “Eliminar pregunta”. Como su mismo nombre indica, sirve para borrar la pregunta de la base de datos. Si se elimina una pregunta, no podremos recuperarla y tendremos que escribirla de nuevo. Antes de borrarla se nos mostrará un mensaje de alerta como el siguiente:

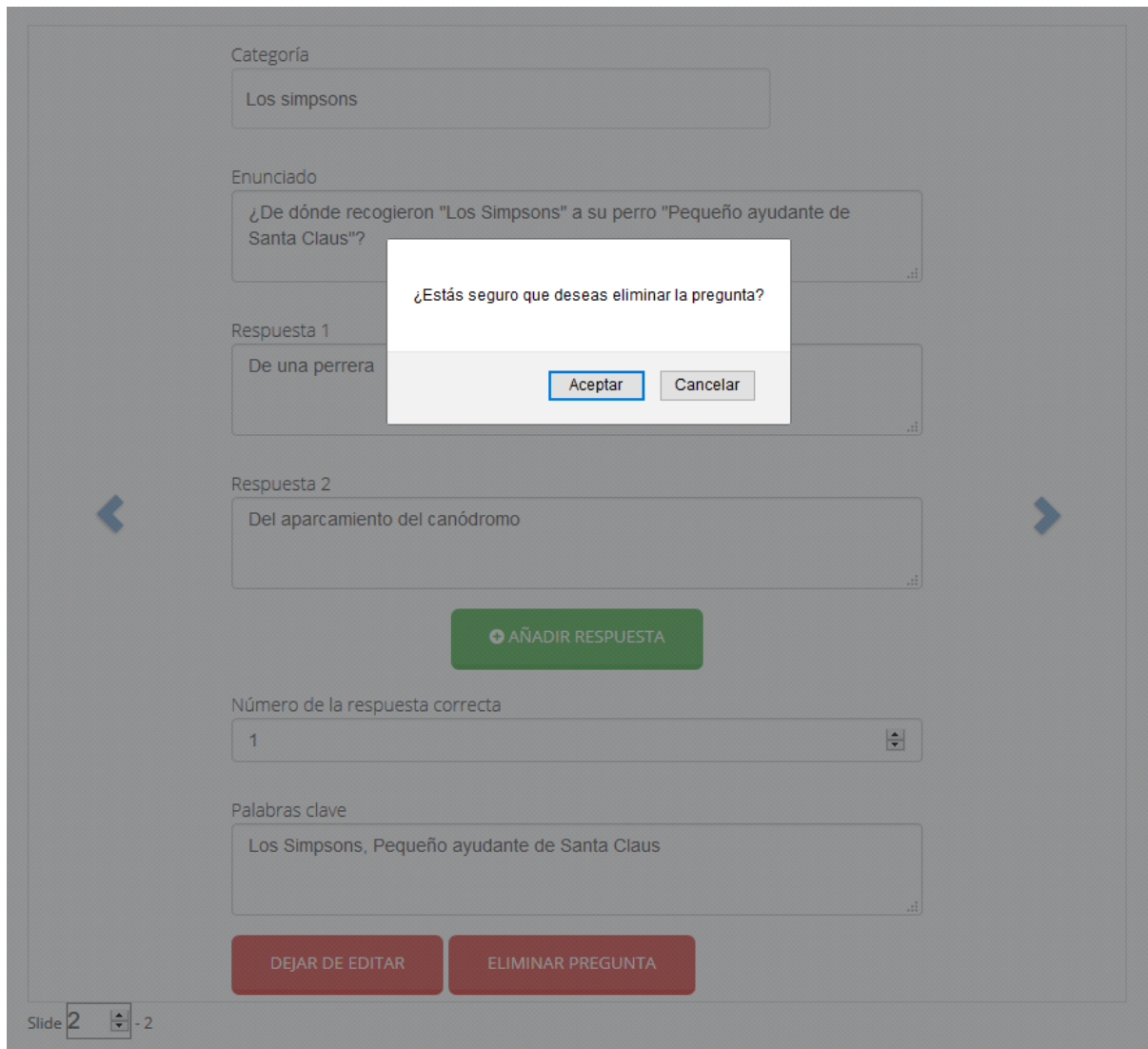


Figura 4.33. Mensaje de alerta antes de eliminar una pregunta.

En esta pantalla, se controlarán los eventos de los campos y los botones con el archivo *funciones_editar.js*. En él están descritos todos los eventos que generan los botones y las funciones asociadas a dichos eventos.

Por otro lado, usaremos 3 scripts de “PHP”:

- `getPreguntasDocente.php`: Este script recoge de la base de datos todas las preguntas de un profesor. Si este no tuviera ninguna, aparecería un mensaje en pantalla diciendo que primero tiene que crearlas para poder verlas o modificarlas. Se hará con la siguiente consulta: `SELECT * FROM 'bancopreguntas' WHERE 'profesor'='$dni'`. Se usa el dni del profesor, ya que es un campo único.
- `updatePreguntaDocente.php`: Este script es el encargado de actualizar los datos de la pregunta que se ha modificado. Funciona de una manera parecida al script de grabar

todas las preguntas, pero en este caso sólo de una en una, y en vez de creando un campo en la base de datos, actualizando uno ya existente. La consulta asociada es la siguiente:

```
UPDATE `bancopreguntas` SET `categoria`=%s, `enunciado`=%s, `respuesta1`=%s, `respuesta2`=%s, `respuesta3`=%s, `respuesta4`=%s, `respuestacorrecta`=%u, `comentarios`=%s WHERE `id`=%u.
```

- deletePreguntaActual.php: Este script hace que se elimine la pregunta escogida de la base de datos. La consulta que se realizará será la siguiente: **DELETE FROM `bancopreguntas` WHERE `id`=\$id.**

4.4.4.3. Pantalla de asignación de preguntas a una actividad.

En esta pantalla, primero se tendrá que elegir la actividad en la que se quieren asignar preguntas. Una vez elegida, se tendrá que escoger la categoría en la que se asignarán las preguntas. Por último, se verá la siguiente pantalla:

The screenshot shows a web interface for assigning questions. At the top, there are two dropdown menus: the first is set to 'Los Simpsons' and the second to 'Categoría 1'. Below these are two buttons: 'Buscar' (Search) and 'RESTABLECER' (Reset). The main content area is divided into three parts: a list of questions on the left, a central navigation area with arrows and a 'PREVISUALIZAR' (Preview) button, and an empty box on the right for the preview.

Figura 4.34. Diseño de la pantalla de asignación de preguntas.

Para asignar una pregunta a una categoría basta con seleccionarla y pulsar el botón “>”. Al pulsarlo, si la pregunta se ha asignado correctamente se mostrará un mensaje en la esquina superior derecha:

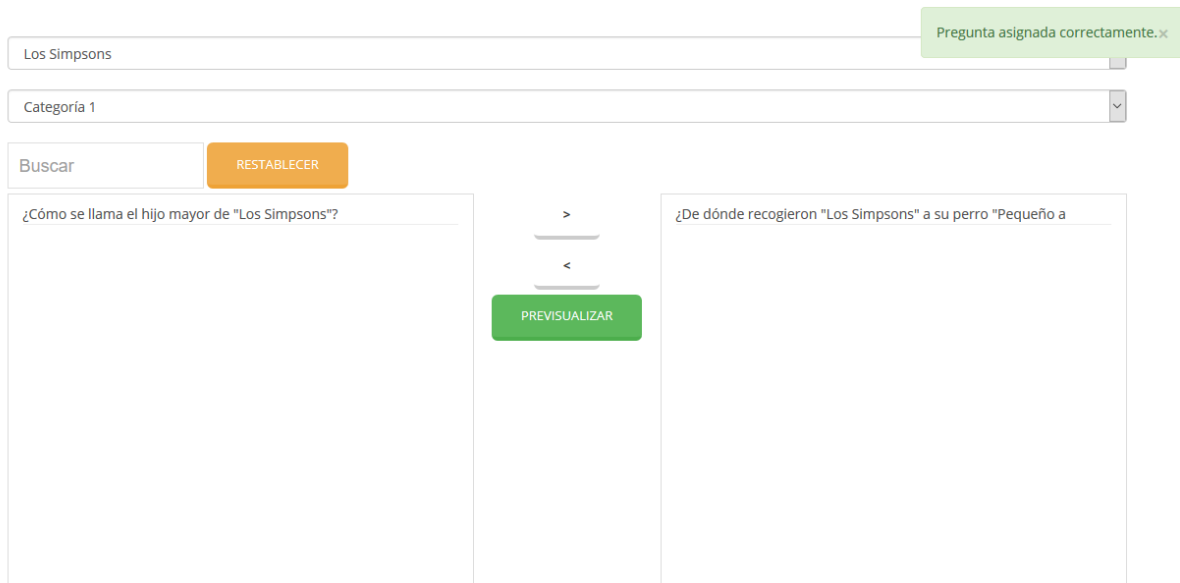


Figura 4.35. Vista del mensaje que aparece al asignar preguntas.

Gracias a que se usa *jQuery UI*, también se puede seleccionar varias preguntas a la vez, y al pulsar el botón “>” se asignarán todas a la categoría.

Al asignar una pregunta a una categoría, ya no volverá a aparecer en las demás categorías. Se muestra un ejemplo. En la captura anterior, se puede ver que se ha asignado una pregunta a la categoría 1. Si ahora se quieren asignar preguntas a la categoría 2, sólo aparecerá la otra pregunta que queda:

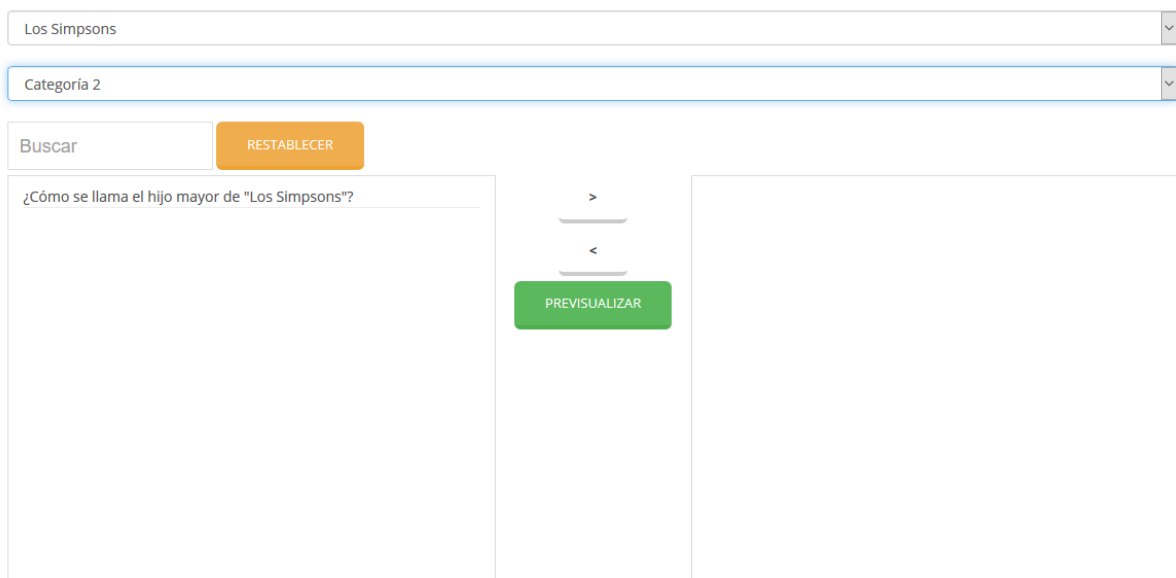


Figura 4.36. Ejemplo de asignar preguntas a otra categoría.

Para eliminar una pregunta de una categoría, basta con seleccionar una pregunta y pulsar el

botón “<”. En este caso no aparecerá ningún mensaje. Como en el caso anterior, para eliminarlas de la actividad, también se pueden seleccionar varias preguntas a la vez y pulsar el botón “<”.

Por último, se puede ver que se tiene el botón “Previsualizar”. Si se selecciona una pregunta y se pulsa este botón, se abrirá una pantalla con la previsualización de la pregunta, como se vería en el juego. Además, también se podrá contestar para asegurar que se ha creado bien.

La pantalla que aparecerá será como la siguiente:

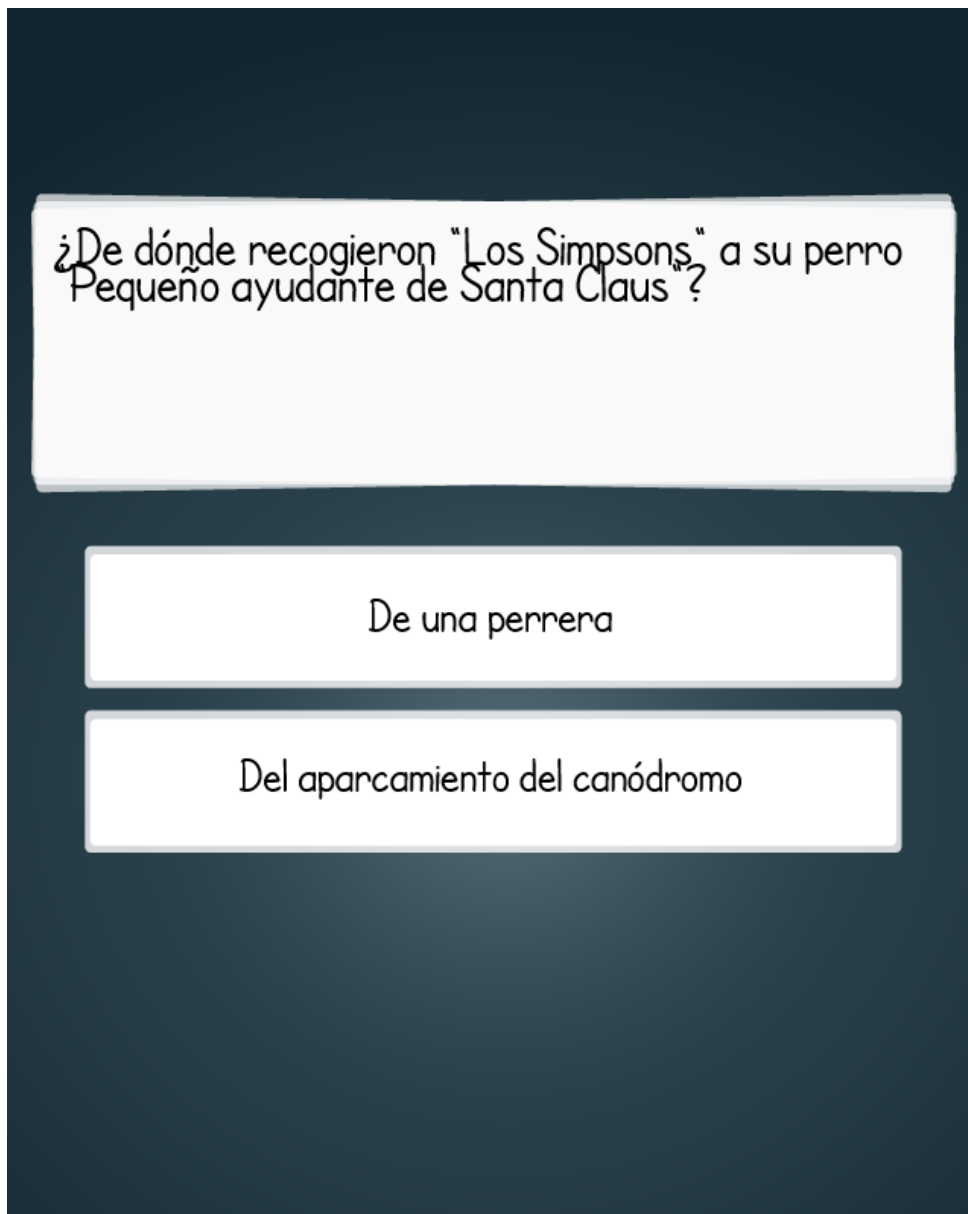


Figura 4.37. Pantalla de previsualización de preguntas.

Para mostrar la pregunta y poder responderla, se usará una porción del código del juego.

Como apunte final, se dispone de un buscador que busca las preguntas por categoría y por palabras clave, ya que, si se tienen muchas preguntas podría resultar un trabajo tedioso el tener que buscarlas a mano.

En esta pantalla, todos los eventos y funciones “*Javascript*” estarán controlados en el fichero *interfaz-asignacion-preguntas.js*. En él se definen los eventos de los botones y de los campos de selección, así como las funciones asociadas a los mismos.

Por otro lado, usaremos 4 scripts de “*PHP*”:

- `getAsignaPreguntas.php`: Se encarga de mostrar los dos primeros campos de selección de actividad y categoría. Éste recoge de la base de datos todas las actividades que tiene el docente asignadas que aún no han empezado. Se hará con la siguiente consulta:

```
SELECT `id`, `nombre` FROM `actividades` WHERE `dni_docente`='$dni' AND `f_inicio`>CURDATE().
```
- `getPreguntasActividad.php`: Se encarga de recoger todas las preguntas de un docente que no están asignadas a la actividad en la que se están asignando preguntas para mostrarlas en el cuadro de la izquierda, y de recoger las preguntas que ya se han asignado a alguna categoría, para mostrarlas en el cuadro de la derecha de la respectiva categoría. Las consultas que se realizan en este caso son:

```
SELECT * FROM `bancopreguntas` WHERE `profesor`='$dni' AND `id` NOT IN (SELECT `idpregunta` FROM `asigna-pregunta-actividad` WHERE `idactividad`=$actividad)
SELECT A.`idpregunta`, B.`id` AS id, B.`categoria` AS categoria, B.`comentarios` AS comentarios, B.`enunciado` AS enunciado FROM `asigna-pregunta-actividad` AS A, `bancopreguntas` AS B WHERE A.`categoria`=$categoria AND A.`idactividad`=$actividad AND A.`idpregunta`=B.`id`
```
- `setPreguntasActividad.php`: Se encarga de guardar en la base de datos la relación entre una pregunta, la categoría y la actividad a la que se va a asociar. Se hará mediante la siguiente consulta: `INSERT INTO `asigna-pregunta-actividad`(`idpregunta`, `categoria`, `idactividad`) VALUES(%u, $categoria, $actividad).`
- `deletePreguntasActividad.php`: Se encarga de borrar la relación entre una pregunta, la categoría y la actividad a la que se había asignado. Se ejecutará la siguiente consulta:

```
DELETE FROM `asigna-pregunta-actividad` WHERE `idpregunta`=$id AND `categoria`=$categoria AND `idactividad`=$actividad.
```


Capítulo 5: Conclusiones y líneas futuras

Al finalizar el proyecto, se comprueba que se ha realizado una aplicación muy atractiva y fácil de manejar, tanto para el alumno como para el docente:

- Al alumno, se le ofrece un juego que se controla de manera muy intuitiva, con el uso de botones como elemento principal del juego.
- Al docente, se le ofrece un completo gestor de bancos de preguntas, que le permite crear nuevas preguntas e incluso reutilizar las que ya dispone en las diferentes actividades que vaya creando. Además, se le permite controlar diferentes variables del juego, como son: el número de preguntas o el tiempo máximo de cada reto.

En la actualidad, el juego se ha desplegado en 3 actividades: la primera con alumnos de la asignatura de “Gestión de los Recursos Humanos” del Grado en Administración y Dirección de empresas, la segunda con alumnos y profesores del instituto CIFP Carlos III, y la tercera con el personal de “CRAI Biblioteca” de la UPCT.

En la primera actividad se realizaron un total de 157 retos entre los 30 alumnos que participaron. Los alumnos contaron con un total de 12 vidas cada uno. Además, en la mesa redonda del curso “Curso de Guionización, Diseño y Maquetación Web”, impartido en el Centro de Producción de Contenidos Digitales, en el que participaron 2 de los alumnos de la asignatura, se comentó que el juego había sido una muy buena herramienta para afianzar los conocimientos de la asignatura, ya que les obligaba a estudiar para poder ganar a sus compañeros. En la encuesta de satisfacción que se pasó al finalizar la asignatura, el juego obtuvo una media de 4.5 sobre 5 puntos.

En la segunda actividad hubo una total de 22 participantes entre alumnos y profesores del centro y un total de 105 retos. En esta actividad también se pasó una encuesta de satisfacción al finalizarla y también se obtuvo una muy buena nota: 4.3 sobre 5 puntos.

En la última actividad participó todo el servicio “CRAI Biblioteca”, un total de 11 participantes. En esta actividad también hubo una gran aceptación y casi terminaron el juego el primer día. Se realizaron un total de 53 retos.

Gracias al despliegue de estas actividades se tuvo un continuo *feedback* que permitió mejorar hasta tener la última versión, la actual, del juego.

Debido a la gran aceptación en estas 3 actividades, se propuso a la Consejería de Educación de

la Región de Murcia la realización de una actividad con todos los institutos de la Región que quisieran participar. Esta propuesta se aceptó y se realizará próximamente el concurso “Rétame y Aprendo”.

Además, a partir de este juego se han desarrollado dos variantes más: “La Ruleta – Duelo” y la “La Ruleta – Entrenamiento”. “La Ruleta – Duelo” es una variante en la que los participantes de un reto contestan las mismas preguntas. “La Ruleta – Entrenamiento” se trata de una variante del juego para un solo jugador

Como apunte final, decir que se ha llevado a cabo una adaptación para personas ciegas del juego, en colaboración con la Fundación ONCE. Esta versión, en la actualidad, no ha sido aún desplegada, pero sí ha sido probada por personas de dicha fundación.

5.1. Líneas futuras

Debido a que el juego se ha realizado aplicando tecnologías web, una de las posibles líneas futuras sería la creación de aplicaciones para Android e iOS. De esta forma tendríamos un mayor control de las notificaciones, ya que vienen integradas en los propios sistemas operativos.

Respecto a la versión adaptada a personas ciegas, también podría ser integrada en la versión móvil, ya que los sistemas operativos Android e iOS cuentan con funciones de accesibilidad nativas.

Respecto al sistema gestor de bancos de preguntas, uno de los desarrollos futuros, sería la adaptación con los diferentes juegos que se vayan creando en un futuro.


Capítulo 6: Bibliografía

 HTML:

- http://www.mclibre.org/consultar/htmlcss/otros/otros_historia.html
- <https://developer.mozilla.org/es/docs/HTML/HTML5>

 CSS:

- <https://developer.mozilla.org/es/docs/Web/CSS>
- <https://developer.mozilla.org/es/docs/Web/CSS/CSS3>

 Javascript:

- <https://es.wikipedia.org/wiki/JavaScript>
- <https://developer.mozilla.org/es/docs/Web/JavaScript/Guide/Introducci%C3%B3n>
- <https://developer.mozilla.org/es/docs/Web/JavaScript>

 Canvas:

- <https://developer.mozilla.org/es/docs/Web/HTML/Canvas>

 AJAX:

- <https://es.wikipedia.org/wiki/AJAX>
- <http://api.jquery.com/jquery.ajax/>

 jQuery:

- <https://es.wikipedia.org/wiki/JQuery>
- <http://api.jquery.com/>

 jQuery UI:

- https://es.wikipedia.org/wiki/JQuery_UI
- <http://jqueryui.com/selectable/>

 CreateJS:

- <http://createjs.com/>
- <http://createjs.com/docs/easeljs/modules/EaselJS.html>
- <http://createjs.com/docs/soundjs/modules/SoundJS.html>

 Twitter Bootstrap:

- https://es.wikipedia.org/wiki/Twitter_Bootstrap
- <http://getbootstrap.com/>

 PHP:

- <https://es.wikipedia.org/wiki/PHP>
- <http://php.net/docs.php>

🚧 SQL:

- <https://es.wikipedia.org/wiki/SQL>
- <https://es.wikipedia.org/wiki/MySQL>
- <https://dev.mysql.com/doc/refman/5.7/en/>

🚧 BASH:

- <https://es.wikipedia.org/wiki/Bash>

🚧 Adobe Illustrator:

- https://es.wikipedia.org/wiki/Adobe_Illustrator
- <https://helpx.adobe.com/illustrator/tutorials.html>