



Universidad Politécnica
de Madrid



**Escuela Técnica Superior de
Ingenieros Informáticos**

Grado en ingeniería informática

Trabajo Fin de Grado

**Aplicación web para gamificación
educativa basada en la resolución de
misterios**

Autor: Jaime Campo Martín

Tutor(a): Antonio Jesús Díaz Honrubia

Madrid, abril 2023

Este Trabajo Fin de Grado se ha depositado en la ETSI Informáticos de la Universidad Politécnica de Madrid para su defensa.

Trabajo Fin de Grado

Grado en Ingeniería Informática

Título: Aplicación web para gamificación educativa basada en la resolución de misterios

Abril 2023

Autor: Jaime Campo Martín

Tutor:

Antonio Jesús Díaz Honrubia

Lenguajes y sistemas informáticos e ingeniería del software

ETSI Informáticos

Universidad Politécnica de Madrid

Resumen

Este proyecto tiene como idea principal el desarrollo de una aplicación web para gamificación educativa. La gamificación educativa es una estrategia didáctica que está cogiendo importancia estos últimos años y que se basa en el empleo de recursos o técnicas propias de los juegos en la educación, favoreciendo la motivación y la implicación por parte del estudiante.

La asignatura en la que se pretende lanzar esta aplicación es en la asignatura de Bases de Datos de la Escuela Técnica Superior de Ingenieros Informáticos, ya que en esta asignatura ya existe una competición que utiliza elementos de la gamificación, lo que favorece la aceptación del alumno a una nueva competición sin muchas dificultades.

La aplicación se va a desarrollar haciendo uso del *framework* Symfony, que facilita el entendimiento y el manejo del código PHP, que es el lenguaje elegido para el desarrollo del código de la aplicación web debido a su popularidad y a su sencillez. Symfony cuenta con varias herramientas que son útiles para el proyecto, como Doctrine que proporciona una capa de abstracción para el uso de la base de datos sin necesidad de que el desarrollador tenga que realizar las consultas de SQL.

El enfoque elegido para esta aplicación ha sido la resolución de misterios o *escape rooms*, debido a que hace uso de diferentes recursos de la gamificación y a que para la participación en este tipo de juegos se requiere de mayor iniciativa e implicación por parte del alumno que en otros juegos.

La aplicación va a compartir página web y base de datos con la competición ya implantada en la asignatura, por lo que hay características, como el inicio de sesión, que ya están implementadas, permitiendo centrar el esfuerzo únicamente en el desarrollo de las funcionalidades necesarias para la realización de la resolución de misterios y la compatibilidad con lo ya desarrollado.

La resolución de misterios consiste en que los alumnos que participen van a tener que encontrar diferentes palabras, que están escondidas por el campus e introducirlas en la página web para canjearlas por problemas relacionados con el temario explicado en la asignatura. Una vez el estudiante haya resuelto un problema, tiene que responder, haciendo uso de la página web, a una serie de preguntas relacionadas con este. Si todas las preguntas son resueltas correctamente se le concederá una pista para resolver el misterio final. Para resolver este misterio se tiene que utilizar todas las pistas que se han conseguido a lo largo de la competición.

Palabras claves:

Educación, Gamificación, Aplicación Web, Resolución de misterios.

Abstract

The main idea of this project is the development of a web application for educational gamification. The educational gamification is a didactic strategy that has been gaining importance this last years and it is based on the use of techniques from the games in the education increasing the student's motivation and implication.

The subject where is pretended to be launched the application is in the subject *Base de Datos*, that is taught in the *Escuela Técnica Superior de Ingenieros Informáticos*, because in this subject it already exists a competition that uses different elements from the gamification, making it easier to accept a new competition for the student.

The application is going to be developed using the Symfony framework that facilitates the understanding and the management of the PHP code, which is the chosen language for the application development web code due to his popularity and simplicity. Symfony has several useful widgets for the project, like Doctrine that gives an abstraction layer to the database use without the need of making SQL consult by the developer.

The chosen approach for this application has been the mystery's resolution or scape rooms, because of the use of different resources from the gamification and because participating in this type of games needs implication and initiative of the student.

The application is going to share the web page and the database with the competition already implanted, so there is some characteristics as the login, that are already implemented, allowing to focus the effort only in the development of the necessary functionalities for the creation of the mystery resolution and the compatibility with the ones already developed.

To complete the scape room the students must find different key words that are hidden in the campus and introduce them in the web page to trade them for problems related to the topic of the subject. When the student has resolved the problem, he must answer in the web page a question string related to the problem. If all the question are solved correctly, he will be granted with a clue for the final mystery. To solve this mystery, all clues found during the competition must be used.

Key Words:

Gamification, Education, Web application, Mystery Resolution.

Tabla de contenidos

1	Introducción	1
1.1	Motivación.....	1
1.2	Objetivos	2
1.3	Lista de tareas y diagrama de Grantt	2
1.4	Estructura	3
2	La gamificación en la educación	4
2.1	Los elementos de la gamificación.....	4
2.2	Tipos de jugadores.....	4
2.3	Tipos de motivación	5
2.4	Estado del arte	5
2.4.1	Kahoot	5
2.4.2	The great quiz of databases.....	6
2.4.3	Diferencias con nuestro proyecto	8
3	Aplicación existente	9
3.1	Estado de la base de datos	9
3.1.1	Tablas influyentes	9
3.1.2	Tablas no influyentes.....	9
3.2	Estado de la aplicación	10
3.2.1	Inicio de Sesión	10
3.2.2	Registro	10
3.2.3	Pantalla principal	11
3.2.4	Preguntas del concurso.....	12
4	Tecnologías empleadas	13
4.1	Framework utilizado	13
4.2	Lenguajes utilizados	14
4.2.1	PHP	14
4.2.2	HTML.....	15
4.2.3	JavaScript.....	16
4.2.4	Twig.....	17
4.2.5	CSS	18
4.2.6	SQL	18
4.2.7	Doctrine.....	19
4.3	Herramientas utilizadas	20
4.3.1	PHPStorm	20
4.3.2	MySQL Workbench	20
5	Desarrollo de la aplicación web.....	22
5.1	Árbol de directorios.....	22
5.1.1	.Idea	22

5.1.2	Bin	23
5.1.3	Config.....	23
5.1.4	Migrations.....	24
5.1.5	Public	24
5.1.6	Src.....	25
5.1.7	Templates	27
5.1.8	Translations	27
5.1.9	Var	27
5.1.10	Vendor.....	28
5.2	Base de datos.....	28
5.2.1	Creación de entidades.....	28
5.2.2	Explicación de la base de datos.....	30
5.2.2.1	User.....	32
5.2.2.2	Edición	32
5.2.2.3	Misterio_problema.....	33
5.2.2.4	User_mystery_problem.....	33
5.2.2.5	Misterio_pregunta	34
5.2.2.6	Misterio_Pista.....	34
5.2.2.7	Mystery_answer.....	35
5.3	Explicación de las clases y las funciones usadas.....	35
5.3.1	Pantalla principal	35
5.3.2	Pantalla de inicio resolución de misterios	42
5.3.3	Pantalla de un problema.....	47
5.3.4	Pantalla de resolución de un problema.....	49
6	Evaluación	53
7	Resultados y conclusiones	55
8	Análisis de impacto	57
9	Bibliografía.....	58

1 Introducción

Este trabajo de fin de grado trata sobre el desarrollo de una aplicación web para la gamificación educativa en la asignatura de Base de Datos de la Escuela Técnica Superior de Ingenieros Informáticos, basada en la resolución de misterios. Se va a comenzar explicando la necesidad que existe en la actualidad de aplicaciones como esta.

1.1 Motivación

En estos últimos años la educación está sufriendo una evolución considerable en la forma de impartir las clases, puesto que existe una gran tendencia hacia la gamificación y al aprendizaje basado en juegos [1]. Estos métodos están teniendo un buen recibimiento debido a que generan en el alumno una motivación mayor que las clases tradicionales y crean un ambiente más lúdico, lo que va a fomentar la creatividad y la implicación de los estudiantes. Esta metodología normalmente va acompañada de una recompensa para los que logren completar los juegos o desafíos, esto va a permitir que estudiantes que necesitan un aliciente también se vean estimulados por estas actividades.

Cada vez es más habitual que se exija en el ámbito laboral la realización del trabajo en pequeños grupos que tienen que colaborar diariamente, esto puede llegar a ser un problema para mucha gente que tiene dificultades a la hora de cooperar o de relacionarse. Por ello, estas estrategias educativas pueden ser muy útiles, ya que normalmente se realizan mediante grupos de trabajo para que se aprenda a colaborar desde pequeño y para generar un mejor ambiente social, que habitualmente va acompañado de un mejor ambiente de trabajo.

Por todas estas ventajas estas técnicas están siendo muy utilizadas en colegios y guarderías, pero todavía no se han establecido como una de las metodologías habituales en universidades, a pesar de los buenos resultados que se están obteniendo. Es importante que se empiecen a introducir estas metodologías en estas instituciones debido a su gran tasa de abandono (siendo en informática del 42,9% según un estudio realizado por la fundación BBVA [2]).

Estos abandonos en su mayoría son debidos a la falta de motivación o la falta de claridad a la hora de entender las lecciones. En muchos casos no se llega al abandono, pero si a la falta de asistencia a las clases lo que es perjudicial para el alumno, ya que va a tener que prepararse las asignaturas acudiendo a academias o de manera autodidacta, lo que en muchas ocasiones va a conllevar al suspenso. También es perjudicial para el docente, que al tener menos alumnos la participación en su clase va a ser inferior y le va a dificultar a la hora de impartirla.

Por todo ello estas metodologías son tan necesarias actualmente.

1.2 Objetivos

A partir de la motivación recientemente comentada, el objetivo principal de este trabajo es el diseño de una aplicación web para la gamificación educativa. Esta aplicación está basada en la resolución de misterios, también conocidos como *escape rooms*. Esta resolución de misterios se divide en diferentes objetivos específicos:

1. Puesta en marcha del sistema que se va a emplear para el desarrollo.
2. Familiarización con el entorno de trabajo.
3. Implementación del módulo de canjeo de palabras por problemas.
4. Implementación del módulo de canjeo de problemas por pistas.
5. Implementación de los módulos desarrollados, en la aplicación ya existente.
6. Evaluación de los resultados obtenidos sobre el uso de la aplicación en un entorno real.

1.3 Lista de tareas y diagrama de Grantt

Para poder cumplir estos objetivos, se han definido una serie de tareas que se van a tener que realizar. Estas tareas son:

1. Familiarizarse con el *framework* de desarrollo web Symfony.
2. Familiarizarse con el código existente de la aplicación de gestión del *The Great Quiz of Databases*.
3. Desarrollo de un módulo para canjear palabras clave por ejercicios.
4. Desarrollo de un módulo para canjear ejercicios resueltos por pistas.
5. Realización de pruebas de la aplicación.
6. Análisis de los resultados obtenidos
7. Escritura de la memoria.
8. Preparación de la defensa.

Estas tareas han sido desarrolladas a lo largo del tiempo como se puede ver en el siguiente diagrama de Grantt:

	Semana 1	Semana 2	Semana 3	Semana 4	Semana 5	Semana 6	Semana 7	Semana 8	Semana 9	Semana 10	Semana 11	Semana 12	Semana 13	Semana 14
Tarea 1														
Tarea 2														
Tarea 3														
Tarea 4														
Tarea 5														
Tarea 6														
Tarea 7														
Tarea 8														

1.4 Estructura

El resto de este trabajo está dividido en los siguientes capítulos, considerando que el capítulo 1 ha sido el de introducción que se acaba de desarrollar:

- Capítulo 2: La gamificación en la educación. Se va a ampliar el concepto de la gamificación y se van a explicar diferentes aplicaciones ya desarrolladas e implementadas con una temática principal basada en la gamificación educativa.
- Capítulo 3: Aplicación existente. Se explica cuál es el estado de la aplicación en la que vamos a desarrollar nuestras funcionalidades en el momento previo al desarrollo de este proyecto.
- Capítulo 4: Tecnologías empleadas. Se desarrollan las tecnologías que se han utilizado para hacer posible el trabajo y el porqué de su elección.
- Capítulo 5: Desarrollo de aplicación web. Se profundiza en cómo se ha desarrollado la aplicación de este proyecto.
- Capítulo 6: Evaluación. Se va a realizar una apreciación del funcionamiento de la aplicación y un análisis de los resultados obtenidos.
- Capítulo 7: Resultados y conclusiones. Se genera un resultado final sobre el proyecto comprobando si se ha cumplido con los objetivos finales y el plan de futuro.
- Capítulo 8: Análisis de impacto. Se desarrolla el impacto que tiene nuestro trabajo sobre la sociedad.
- Capítulo 9: Bibliografía. Se incluyen todas las referencias que han sido citadas sobre el material utilizado durante el desarrollo de la memoria.

2 La gamificación en la educación

La gamificación es una estrategia que utiliza los elementos de los juegos en otros contextos, en el caso de este proyecto, en la enseñanza. El objetivo principal de la gamificación es incidir en la motivación del estudiante para diferentes metas, como pueden ser, aumentar la asistencia, mejorar el rendimiento y los resultados e incrementar la iniciativa, entre otros.

2.1 Los elementos de la gamificación

La gamificación incorpora diferentes elementos para poder completar las metas propuestas y con ellas el objetivo principal. Estos elementos fueron desarrollados por Werbach y Hunter [3] y son los siguientes:

- **La dinámica empleada:** es la forma en la que los estudiantes interactúan con el juego, y en caso de que fuese necesario, con otros jugadores. La dinámica es un concepto muy amplio en el que se encuentran las emociones del alumno, la narrativa del juego, las limitaciones existentes, las propias relaciones, etc.
- **Mecánicas:** son las reglas y acciones que los alumnos deben seguir, y que van a generar la atracción. Las mecánicas empleadas son iguales para todos los alumnos. Unos ejemplos de estas mecánicas son los retos, los turnos o las recompensas.
- **Componentes:** son las partes individuales que son usadas para el desarrollo del juego. Algunos ejemplos de componentes son: avatares, puntos, logros o insignias.

2.2 Tipos de jugadores

Los jugadores van a ser los propios alumnos que participen en la asignatura y más concretamente en la competición. Para ello es importante conocer los diferentes tipos de alumno que existen. Esta clasificación va a estar basada en la desarrollada por Richard Bartle en 1996 [4].

- **Killers:** estos jugadores disfrutan compitiendo y ganando al resto de participantes. Necesitan actividades rápidas y algún tipo de recompensa en la que puedan mostrar su superioridad.
- **Achievers:** disfrutan obteniendo recompensas o logros que les permitan demostrar su progreso y lo que han acumulado en el desarrollo del juego. Necesitan obtener todos los misterios o logros posibles, por lo que es necesario el uso de insignias o rankings.
- **Explorers:** les gusta explorar y descubrir nuevas cosas por su propia cuenta, sintiendo la libertad de crear diferentes caminos para resolver las cosas. Para estos jugadores es necesario permitir la personalización de las cosas, recompensas diferentes y actividades con libertad como los *escape rooms*.
- **Socializers:** estos jugadores no priorizan el propio juego, sino que se ven atraídos por la interacción con los otros alumnos participantes, permitiéndoles disfrutar de las experiencias y relaciones sociales. Necesitan poder comunicarse con el resto y realizar actividades que requieran de la formación de equipos.

Un alumno puede ser de varios tipos a la vez, por lo que es necesario conocer a que tipo de alumnos va destinado nuestro juego, para poder adaptarlo a ellos.

2.3 Tipos de motivación

Una vez explicados los tipos de alumnos que existen, se puede explicar la motivación. La motivación es el objetivo principal de la gamificación y es el motor que permite a los alumnos involucrarse y tener compromiso con las actividades, para ello es necesario entender los dos tipos de motivación que existen:

- **Intrínseca:** es la motivación interna del alumno y proviene del interés que este posea y de la satisfacción por lograr algo. Esta motivación no es igual para cada alumno y es más difícil de trabajar.
- **Extrínseca:** esta motivación es lo contrario de la anterior, no proviene del alumno si no que se genera a partir de recompensas externas. Esta motivación es eficaz a corto plazo, pero puede ser perjudicial con el tiempo, debido a que puede producir en el alumno una necesidad de recompensa para realizar cualquier actividad.

Es necesario trabajar los dos tipos de motivación para poder desarrollar actividades para la gamificación educativa.

2.4 Estado del arte

Una vez se ha desarrollado lo que es la gamificación se puede entender como las aplicaciones existentes hacen uso de ella para intentar mejorar en la enseñanza. Como se ha explicado, ya existen muchas aplicaciones que son usadas para este fin, pero vamos a desarrollar únicamente las que tienen mayor importancia indicando en que se diferencian de la nuestra y porque es necesaria.

Se va a diferenciar entre aplicaciones que no tenga relación con la asignatura de Base de Datos, y las que sí. Esto se debe a que en esta asignatura ya se ha implementado anteriormente alguna técnica de gamificación educativa.

2.4.1 Kahoot

Para poder hacer una buena comparación, es necesario que la aplicación de la que se va a hablar sea usada en universidades. Por ello la elegida es Kahoot [5].

Kahoot es una aplicación que cuenta con más de 50 millones de descargas en la *playstore* [6], por lo que es una herramienta totalmente consagrada en la educación. Este número de usuarios nos permite ver la importancia que tiene la gamificación y que hagan uso de ella en la actualidad. Es de las pocas aplicaciones que es usada tanto en el colegio como en la universidad.

Kahoot es una plataforma que puede ser usada directamente desde el navegador web o que puede ser descargada como aplicación, que permite al profesor crear cuestionarios personalizados para que luego sean respondidos por los alumnos. Estos cuestionarios son una buena forma de evaluación y fáciles de usar.

Para poder usarla, el profesor, una vez ha creado el cuestionario con las preguntas y las posibles repuestas, tiene que compartir un código que los alumnos tienen que introducir para poder participar. Una vez dentro se les irá haciendo las preguntas, suelen tener un tiempo límite para contestar y son preguntas en las que hay que seleccionar únicamente la respuesta correcta,

como se puede observar en la figura 1. Cada respuesta correcta sumará una serie de puntos según la velocidad en la que se haya respondido y al final del cuestionario aparece un ranking con los alumnos que más puntuación han sumado.

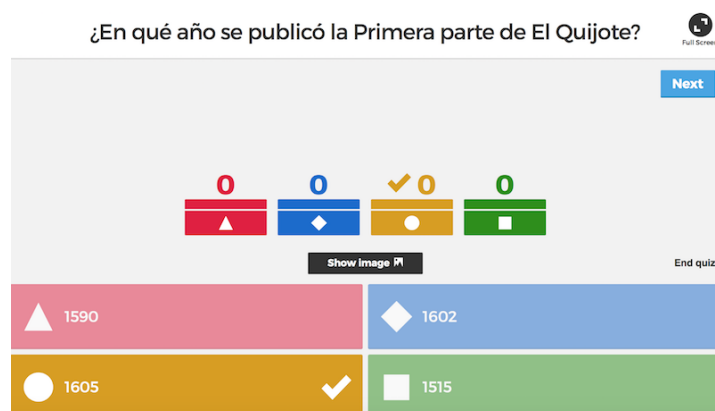


Figura 1. Ejemplo de pregunta de Kahoot

Esta aplicación tiene muchas similitudes con los concursos que se pueden ver a diario en la televisión, pero llevados al ámbito educativo lo que permite al estudiante aprender de una forma más entretenida. Igual que en estos concursos, como se ha comentado, existe una puntuación y un ranking que pueden ser empleados para cuando se haya acabado la competición recompensar a los ganadores con algún tipo de beneficio, como por ejemplo puntuación extra en la asignatura.

Esta aplicación intenta generar en el alumno una mayor motivación extrínseca, a través de las recompensas que se le dan al estudiante. Además, está enfocada en los alumnos *killers*, que necesitan de una recompensa con la que poder satisfacer su ambición y en los *achievers*, que necesitan una forma de poder mostrar su progreso mediante el ranking que ofrece esta aplicación. Por esto y por la facilidad de uso que tiene, esta plataforma tiene un buen recibimiento por parte de los docentes y del alumnado.

A pesar de todos estos aspectos positivos, existen una serie de aspectos que pueden ser mejorados y que son cubiertos en el desarrollo de este proyecto.

Estos cuestionarios no se suelen realizar con mucha frecuencia, lo que hace que la implicación del alumno sea menor que en otras competiciones constantes en el tiempo y que obliguen a un mayor esfuerzo. A su vez, el tipo de preguntas que se realizan suelen ser cortas y tienen que ser respondidas en el momento sin necesidad de realizar ningún tipo de cálculo o investigación profunda, lo que permite una evaluación de conceptos muy concretos y amplios, pero no permite entrar en detalle sobre ninguno de ellos, sumado al factor azar existente en la elección de la respuesta.

2.4.2 The great quiz of databases

En la asignatura de Base de Datos de la Escuela Técnica Superior de Ingenieros Informáticos, ya han existido diferentes competiciones que fomentaban la gamificación educativa, The Great Quiz Of Databases ha sido la más reciente y la que va a compartir plataforma con la desarrollada en este proyecto. Por esto, es la aplicación interna que vamos a utilizar como referencia.

The Great Quiz Of Databases [7] es una competición que consta de dos partes, una primera parte, que empezó estando basada en Kahoot para después hacer uso de Wooclap y una segunda parte basada en la antigua competición de SQL [8] que existía en esta universidad.

La primera parte está compuesta de una serie de cuestionarios semanales, creados mediante Wooclap y realizados mediante directos de Twitch, permitiendo que la participación sea mayor que si se hicieran de manera presencial. En un principio esta parte se realizaba en Kahoot, pero se acabó optando por el uso de Wooclap, ya que esta plataforma permite mayor personalización de las preguntas y se adapta mejor al formato de *streaming*, dado que Kahoot no permitía mostrar las preguntas en el dispositivo del alumno, sino que solo mostraba las posibles respuestas lo que se veía perjudicado por el retraso que existiese en el directo. Estos cuestionarios tienen un ranking global que cada semana se va actualizando según la puntuación conseguida esa semana. La temática de los cuestionarios realizados tiene que ver con el temario impartido en esa semana durante la asignatura, haciendo que los alumnos que estén involucrados en la competición lleven el temario al día.

La otra parte, como se ha mencionado, es la implementación de la antigua competición de SQL que existía, ésta se basa en una serie de preguntas que incrementan la dificultad según se va avanzando en su resolución. El objetivo principal de esta competición era facilitar el entendimiento de la parte de SQL de la asignatura, que por norma general era la más complicada de entender.

Igual que el Kahoot, esta competición está enfocada en la motivación extrínseca, que se ve favorecida debido a la recompensa que se le otorga al alumno. Y los alumnos a los que va dirigida son en su mayoría alumnos *killers* y *achievers*, que se ven motivados debido a la existencia de una puntuación y un ranking que puedan demostrar su progreso respecto al resto de los participantes.

Tras la experiencia de los alumnos en The Great Quiz Of Databases se ha podido observar el impacto positivo que tuvo en la calificación final, en la que en alguno de los grados llega a haber una diferencia de casi dos puntos, como se puede ver en la figura 2.

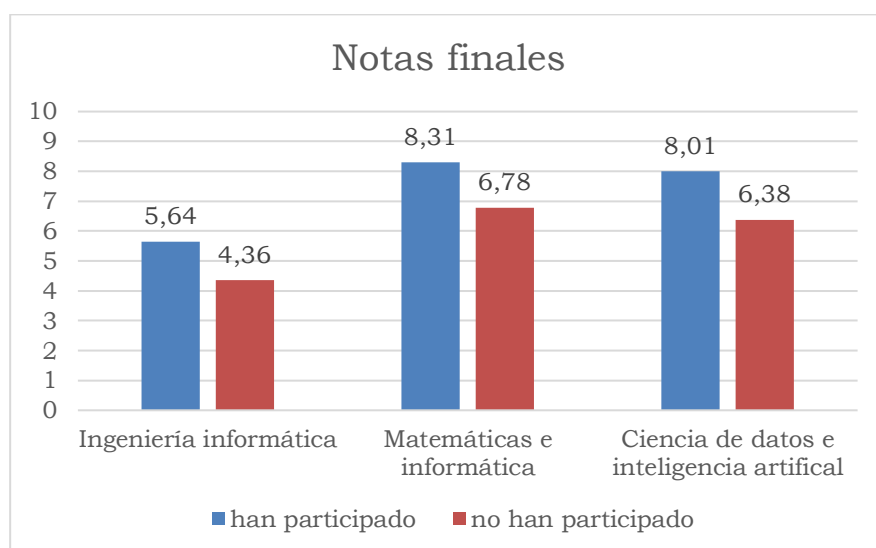


Figura 2. Notas finales de los estudiantes [7]

La participación en este concurso fue del 18% en el grado de Ingeniería Informática, del 21% en el grado de Matemáticas e Informática y del 46% en el grado de Ciencias de Datos e Inteligencia Artificial siendo una participación un poco escasa, pero con un impacto y un nivel de satisfacción notable.

2.4.3 Diferencias con nuestro proyecto

Como hemos podido observar estas aplicaciones son realmente útiles y tienen una buena acogida por parte del alumnado y de los docentes, pero tienen una serie de problemas que son los que se intentan corregir con nuestro proyecto.

Como se ha comentado, la implicación del Kahoot se veía afectada al no ser constante en el tiempo o al ser una actividad esporádica que no tenía realmente una importancia fija. Esto se contrarrestó en The Great Quiz Of Databases al ser una competición semanal, lo que permitía al alumno una constancia y una motivación marcada, pero esto a su vez implicó que la participación se viese disminuida debido a la incompatibilidad de horario, puesto que la competición se realizaba en un horario externo al de la asignatura, pero marcado, por lo que podían existir coincidencias con asuntos de cada persona. Para ello en esta nueva aplicación que se propone se ha buscado una solución, en vez de tener un horario fijo, la competición abre en una fecha y cierra en otra fecha asignada, por lo que el alumno es el que decide en qué momento realizar la actividad, pero teniendo unos plazos marcados y siendo actividades constantes no como en los cuestionarios de Kahoot habituales.

Por otro lado, otro problema existente en el Kahoot era la falta de preguntas con mayor profundidad en los temas y preguntas que pudiesen ser respondidas de diferente forma, no únicamente con selección de respuesta correcta. Esto también era corregido en la parte de SQL de The Great Quiz Of Databases, pero eran únicamente de esa parte de la asignatura. En la nueva aplicación que se ha desarrollado este problema se ha solucionado, una vez canjeas una palabra clave por un problema, tienes que descargar y resolver un problema muy parecido a los del examen, para después una vez resuelto responder a una serie de preguntas sobre él, estas preguntas pueden ser por ejemplo numéricas, de selección o de escribir, entre otras posibilidades.

El proyecto que se trata en este trabajo no solo se basa en corregir fallos anteriores, sino que intenta que la innovación sea la máxima posible, por ello la actividad que se propone es mediante la resolución de misterios. Es un ejercicio a los que los alumnos no están acostumbrados, sumado a la popularidad que tienen estos juegos, también llamados *escape rooms*, se espera que el interés sea mayor y que no solo los alumnos *killers* o *achievers* se vean beneficiados, sino que también va enfocada a los alumnos *explorers*, que necesitan de libertad creativa, en este caso mediante la forma de encontrar las palabras claves y de resolver el misterio final. Por otra parte, al ser un ejercicio que no solo se basa en la resolución de problemas, sino que también hay una parte de búsqueda de pistas y de resolución de un misterio, la participación necesaria es considerable.

3 Aplicación existente

En este apartado se va a explicar el estado de la aplicación existente en el momento previo al desarrollo de este proyecto.

Como se ha comentado anteriormente, el desarrollo de las nuevas funcionalidades va a ser realizado en la misma página web en la que se realiza la competición de SQL de The Great Quiz Of Databases.

3.1 Estado de la base de datos

La aplicación de resolución de misterios y la aplicación de la competición de SQL van a compartir la misma base de datos, por lo que el entendimiento de esta es importante para el desarrollo de la nueva aplicación. La base de datos previa a la incorporación de las nuevas funcionalidades se puede ver en la figura 3.

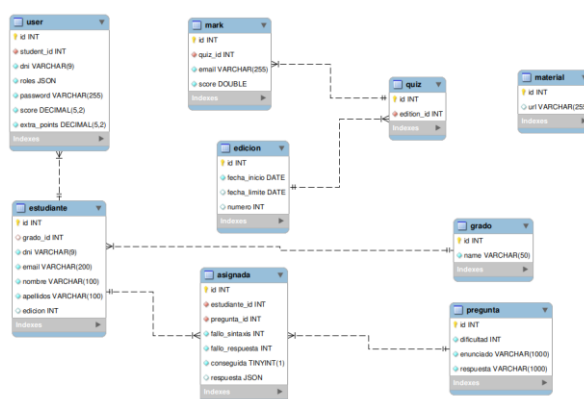


Figura 3. Diagrama de tablas de la aplicación previa

Para la explicación de cada una de las entidades vamos a clasificarlas en dos categorías, según si tienen influencia en nuestras funcionalidades o no.

3.1.1 Tablas influyentes

Estas tablas son las que, aunque no sea de forma activa, tienen importancia a la hora de desarrollar este trabajo:

- Estudiante: contiene toda la información del alumno participante en la competición. Esta información es obtenida a través del registro del usuario.
- User: contiene únicamente la información del estudiante que va a ser necesaria una vez ya ha iniciado sesión.
- Edición: cada año la competición es realizada en fechas distintas por lo que es importante un registro de estas y así diferenciar las propias ediciones entre sí.
- Grado: hace referencia al grado en el que está matriculado el alumno.

3.1.2 Tablas no influyentes

Estas tablas no tienen ningún tipo de relación con las funcionalidades que se van a desarrollar, es decir, son independientes de la resolución de misterios y solo influyen en el The Great Quiz Of Databases:

- Pregunta: contiene la información correspondiente a cada pregunta del *quiz*.
- Asignada: relaciona al estudiante con las preguntas que le han sido asignadas, almacenando las respuestas y si han sido conseguidas o no.
- Material: contiene un enlace que dirige al material de ayuda para la asignatura.
- Quiz: hace referencia a cada semana de la competición de *The Great Quiz Of Databases*.
- Mark: corresponde a la puntuación obtenida por los diferentes estudiantes en los cuestionarios semanales.

3.2 Estado de la aplicación

Tras la explicación de la base de datos que existía previa a nuestro desarrollo, se va a realizar una explicación del funcionamiento paso a paso de la aplicación existente.

3.2.1 Inicio de Sesión

La primera pantalla existente es la referente al inicio de sesión o *Log in* (Figura 4).

En esta pantalla podemos observar un formulario en el que se le pide al alumno que introduzca su DNI y su contraseña, estos datos han tenido que ser registrados previamente, por ello se puede observar en la parte inferior de la pantalla una opción de registrarse en caso de no estarlo ya, que redirige a la pantalla de registro.

También se puede observar la opción de recordar el inicio de sesión, que lo que permitirá es mantener la sesión iniciada una vez se haya cerrado la página, permitiendo acceder directamente a esta sesión la próxima vez que se abra.

Y, por último, mediante un botón podemos acceder a la pantalla principal de la página web y del concurso, previamente han tenido que ser rellenados correctamente los campos de DNI y contraseña.

Figura 4. Pantalla de Log In

3.2.2 Registro

La figura 5 hace referencia a la pantalla de registro, es una pantalla similar a la del inicio de sesión, pero esta vez el formulario está compuesto por más campos:

- DNI: el DNI del alumno sin letra.

- Email: el correo electrónico de la universidad del estudiante.
- Nombre: el nombre del alumno.
- Apellidos: los apellidos del alumno.
- Grado: el grado en el que está matriculado el alumno. Este campo es en forma del desplegable y las opciones son las siguientes: “Grado en Ingeniería Informática”, “Grado en Matemáticas e Informática”, “Grado en Ciencias de Datos e IA (Montegancedo)”, “Grado en Ciencias de Datos e IA (Campus Sur)” y “No matriculado en Bases de Datos”.
- Contraseña: la contraseña elegida por el estudiante para su futuro inicio de sesión.
- Repetir contraseña: en este campo se tiene que utilizar la misma contraseña utilizada en el campo anterior, para la comprobación de que los dos campos sean idénticos.

Figura 5. Pantalla de Registro

3.2.3 Pantalla principal

Una vez se ha completado el registro o el inicio de sesión, accedemos a la pantalla principal. Esta pantalla principal puede tomar 3 formas, según entremos antes de que empiece el concurso, durante la realización del concurso o una vez terminado el concurso. Para ello se utilizan las fechas de cada edición.

En la figura 6, se puede ver la pantalla principal antes del comienzo de la competición. En ella se observa una cuenta atrás del tiempo que falta para el inicio de dicha competición, la puntuación actual del alumno y un botón que permite acceder al material de la asignatura.



Figura 6. Pantalla principal previa al inicio de concurso

Por otro lado, en la figura 7 la competición ya ha comenzado y se puede observar una pequeña descripción sobre el funcionamiento de la competición, la explicación de cómo se puntúa, un enlace pulsando en “click aquí” que redirige a una imagen con la base de datos que se utiliza para la competición y dos botones, uno con el que se comienza a resolver la competición y el otro que se ha explicado previamente que redirige al material de la asignatura.



Figura 7. Pantalla principal durante el concurso

Por último, se puede observar en la figura 8 como se muestra la pantalla una vez se ha acabado la competición. Se puede ver el número de preguntas que se han respondido y la puntuación obtenida tanto de la parte de SQL como de la competición de Kahoot. Además, existe el mismo botón que en las pantallas anteriores que redirige al material de la asignatura.

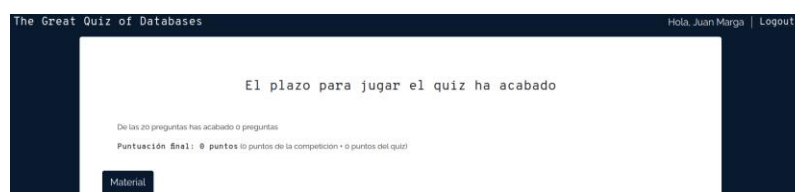


Figura 8. Pantalla principal una vez terminado el concurso

3.2.4 Preguntas del concurso

En la figura 9, se encuentra un ejemplo de cómo son las preguntas que se realizan durante este concurso.

Esta pantalla está formada por un formulario simple en el que se debe introducir la respuesta a la pregunta correspondiente, un botón para acceder a la siguiente pregunta, la puntuación obtenida hasta el momento y la posición respecto al resto de alumnos.



Figura 9. Ejemplo de pregunta

4 Tecnologías empleadas

4.1 Framework utilizado

En este apartado vamos a hablar del *framework* utilizado, que ha sido Symfony [9]. Pero antes de hablar de este, hay que explicar lo que es un *framework*.

Un *framework* o en castellano, entorno de trabajo, es una estructura previamente creada, que sirve como estructura base o referencia para facilitar el desarrollo de una tarea informática a los programadores, es decir, es una plantilla utilizada como guía para ciertas funciones, que va a simplificar el trabajo de tareas similares con origen común.

Actualmente son muy usados en el desarrollo software porque permiten la automatización de muchas tareas en las que no se debería emplear tanto tiempo, con lo que el trabajo se ve acelerado y simplificado. Además, permite la reutilización de código por parte de los desarrolladores, y una mejor colaboración entre éstos, lo que va a desencadenar en un mejor resultado con un menor número de errores, con un código más legible y fácil de mantener y siendo capaz de cumplir con los objetivos.

El elegido para este proyecto, como se ha comentado, ha sido Symfony. La primera versión desarrollada fue en 2005 por Fabien Potencier, coautor junto a Francois Zaninotto del libro sobre Symfony “The definitive guide to Symfony” [10], quien en el año 2003 realizó una investigación sobre las herramientas de *software* libre que existían para el desarrollo de aplicaciones web con PHP, pero no fue hasta el lanzamiento de PHP 5, cuando se pudo desarrollar el *framework* con todas las herramientas disponibles.

En un principio Symfony fue lanzado únicamente para proyectos de Sensio, la empresa cuyo presidente era el mismo Fabien Potencier, y su núcleo estaba basado en el modelo vista-controlador, lo que permitía separar el código del desarrollo en diferentes capas según la responsabilidad, y en el mapeo objeto-relacional (ORM), que permitía crear una capa de abstracción de la base de datos facilitando el uso de esta para los desarrolladores. Con el tiempo se lanzó como *open source*, lo que ha permitido que desarrolladores de todo el mundo puedan hacer uso de este framework y mejorarlo [11].

Actualmente es uno de los *frameworks* más utilizados para el desarrollo de aplicaciones web con PHP y ha sido usado para sitios con importancia como Yahoo! Answers o DailyMotion.

Las principales ventajas de Symfony son:

- **Sencillez:** es fácil la instalación y configuración del entorno de trabajo necesario para poder hacer uso de este *framework*.
- **Facilidad de uso:** el aprendizaje de este *framework* es sencillo, permitiendo a los nuevos usuarios poder adentrarse en el uso de este sin complicaciones.
- **Extensible:** permite la integración de librerías y paquetes desarrollados por terceros.
- **Legibilidad:** el código desarrollado hace uso de las estructuras y plantillas existentes permitiendo una mejor estructuración y legibilidad de este.
- **Amplia documentación:** la popularidad alcanzada por el *framework* ha permitido la creación de una comunidad extensa, que facilita el uso de este a través de la documentación que se publica. Además, la página

oficial del *framework* también contiene una amplia documentación sobre este.

- **Compatibilidad:** amplia compatibilidad con sistemas operativos como Windows y Linux, así como con cualquier gestor de base de datos (MySQL, PostgreSQL, Oracle...)
- **Personalizable:** no es necesaria la instalación de todas las partes existentes, sino que se puede personalizar e instalar las necesarias.
- **Independiente del gestor de bases de datos:** al estar basado en ORM concede una capa de abstracción, que hace posible cambiar de gestor de base de datos sin dificultades.

4.2 Lenguajes utilizados

Durante el desarrollo del proyecto se han utilizado diferentes lenguajes que se van a explicar a continuación.

4.2.1 PHP

PHP [12] es un lenguaje de programación para el desarrollo de aplicaciones y páginas web.

Fue creado originalmente en 1994 por el programador Rasmus Lerdorf, estaba compuesto por diferentes *scripts* en C y el nombre que recibía era Personal Home Page Tools.

Su creador decidió que PHP fuese *open source* lo que permitió que una gran cantidad de desarrolladores se viesen atraídos a participar en el uso y mejora de este lenguaje. Tras varias actualizaciones sin mucha importancia, apareció la versión PHP 3 lanzada en 1998, que es la que dio el nombre con el que se conoce actualmente a este lenguaje, Hypertext Preprocessor. Esta versión era compatible con la mayoría de los sistemas operativos de la época y vino marcada por la extensibilidad. Junto a esta mejora también se incluían una sintaxis de lenguaje más potente, soporte para la programación orientada a objetos y una interfaz más madura. Cuando se hizo oficial el lanzamiento de esta versión, estaba instalado en más de 70.000 dominios y aproximadamente en el 10% de servidores web.

La versión 4 llegó en el año 2000, en la que se reescribió todo el núcleo de PHP para mejorar su modularidad y mejorar el rendimiento de código. Esto se consiguió gracias a los avances que hubo en los motores de la época, que permitieron no solo aumentar el rendimiento, sino que también permitieron incluir soporte para la mayoría de los servidores web, buffers de salidas, controlar las entradas de usuarios, y otras características más que lo convirtieron en uno de los lenguajes más populares.

PHP 5 permitía un mejor soporte para la programación orientada a objetos, a XML y a MySQL, mejor rendimiento y una herramienta encargada de ser el gestor de dependencias conocida como Composer [13].

Actualmente PHP es utilizado aproximadamente por el 80% de los sitios web [14], entre las que se encuentran varios sitios destacados como WordPress o Drupal. La masificación del uso de este lenguaje puede acarrear futuros problemas, debido a que varios de estos sitios están usando versiones no soportadas activamente, es decir, versiones que ya no están actualizadas y que no están siendo revisadas.

Por otro lado, las versiones PHP 8.0 y posteriores todavía no están siendo usadas por la mayoría de la comunidad, sino que prefieren utilizar versiones pasadas ya establecidas.

PHP forma parte del grupo de tecnologías LAMP, que forman una plataforma potente y versátil para el desarrollo web. Este grupo está formado por Linux, Apache, MySQL y PHP, que dan nombre al acrónimo. Estas tecnologías tienen una gran importancia en el mundo del desarrollo de páginas web y son una pieza clave para los desarrolladores.

Las características principales de este lenguaje son:

- **Facilidad de aprendizaje:** es un lenguaje sencillo para la gente que no lo conoce, por lo que hace que gente no tan experta se inicie en él.
- **Gran comunidad:** al ser uno de los lenguajes más populares posee una cantidad notable de desarrolladores y de gente con conocimientos, que permite que haya un extenso número de documentos y ayudas de fácil acceso.
- **Gran cantidad de datos:** es capaz de soportar una notable cantidad de bases de datos y de peticiones, sin ver afectado su rendimiento.
- **Código abierto:** al ser *open source* permite que cualquier persona pueda desarrollar cualquier tipo de modificación que permite mejorarlo.
- **Versatilidad:** permite la conexión con la mayoría de las bases de datos.
- **Facilitadores:** posee gran variedad de *frameworks* y librerías que facilitan su uso y aumentan la productividad.
- **Multiplataforma:** puede ser utilizado en diferentes sistemas operativos como Windows, Linux o macOS.

En nuestro proyecto, ha sido empleado para el *backend*, es decir, para el desarrollo de todo el código referido a la lógica que se emplea para las funcionalidades y para el código que se ejecuta en el servidor.

4.2.2 HTML

HTML (*HyperText Markup Language*) [15] es un lenguaje de marcas, utilizado para la elaboración de páginas web. Un lenguaje de marcas hace referencia a un tipo de lenguaje que se utiliza para definir la estructura y el formato del contenido en un documento de texto, a través de marcas o etiquetas que indican como debe estructurarse y presentarse el código.

HTML se desarrolla en 1980, cuando Tim Berners-Lee desarrolla un sistema de hipertexto para compartir documentos. Los sistemas de hipertexto permitían acceder a la información relacionada con los documentos electrónicos. Tras finalizar este desarrollo, se presentó este sistema con el nombre de WorldWideWeb (W3).

En 1991 se encuentra el primer borrador relacionado con HTML bajo el nombre de HTML Tags [16], formado por un conjunto de etiquetas que permitían crear enlaces, párrafos básicos y encabezados.

La primera propuesta para convertir HTML en estándar se realizó en 1993 de parte de IETF (*Internet Engineering Task Force*), pero no fue hasta 1995 cuando se consiguió oficializar el estándar bajo el nombre de HTML 2.0, a pesar de ser la primera versión oficial.

La versión HTML 3.2 llegaría en 1997, siendo la primera versión publicada por W3C (*World Wide Web Consortium*), que a partir de este momento será la encargada de la publicación de estándares de HTML. En esta versión se añaden

los últimos avances existentes en diseño de páginas web. Además, también se incluyeron nuevas etiquetas.

En 1998 se desarrolla HTML 4.0, que incluye numerosas novedades, entre las que cabe destacar, las hojas de estilo en cascada o CSS y la posibilidad de incluir pequeños programas o *scripts* en las páginas web. Estas novedades producen una gran evolución de las versiones anteriores.

En 1999 se publica HTML 4.01 que solo servía como revisión y actualización de la anterior. A partir de esta versión se detuvo la estandarización de HTML, lo que llevó en 2004 a una preocupación por parte de empresas importantes como Apple o Mozilla. Por este motivo se organizó una nueva asociación conocida como WHATWG (*Web Hypertext Application Technology Working Group*) que obligó a W3C a retomar el desarrollo de HTML.

En la actualidad la versión más reciente es HTML 5. En esta versión se han incluido nuevas etiquetas que permiten el contenido multimedia, la capacidad de ajustar el tamaño de la página al tamaño del dispositivo y la capacidad para crear contenido dinámico. [17]

Las principales características de HTML son:

- **Compatibilidad:** es un lenguaje compatible con todos los navegadores Web importantes.
- **Integración:** permite la integración con otros lenguajes de programación, como JavaScript o CCS.
- **Tamaño:** los archivos generados por HTML no son de gran tamaño, lo que permite que la capacidad no sea un problema.
- **Gran comunidad:** posee una gran comunidad que va a permitir acceder a ayuda de diferentes maneras.
- **Aprendizaje sencillo:** al poseer tanta información sobre este lenguaje posee un aprendizaje sencillo.

En nuestro proyecto es utilizado en el *frontend* para definir la estructura y la organización del contenido en la página web.

4.2.3 JavaScript

JavaScript [18] es un lenguaje de programación intérprete, usado del lado del cliente, que permite mejoras en la interfaz del usuario y la creación de páginas web dinámicas.

Los inicios de JavaScript se retoman a 1995, por parte de Brendan Eich, un desarrollador de la compañía Netscape Communications Corporation. Surgió por la necesidad de interacción entre el navegador y el usuario y a su vez por la complejidad que estaban adquiriendo las páginas webs, que necesitaban ser ejecutadas en el propio navegador del usuario. El nombre que recibió este lenguaje fue LiveScript.

Netscape creó una alianza con Sun Microsystems y se cambió el nombre del lenguaje a JavaScript. Este cambio de nombre vino dado por la compatibilidad que fue agregada por parte de Netscape con la tecnología Java. Pero este cambio, fue más un elemento de *marketing* que técnico, debido a que Java era la tecnología de moda en ese momento.

El éxito de JavaScript era notable hasta tal punto de que Microsoft lanzó una copia para su navegador Internet Explorer 3, que tenía el nombre de JScript.

En 1997, se oficializó la estandarización del lenguaje por parte de ECMA International, lo que produjo la compatibilidad entre todas las implementaciones que existiesen de JavaScript, convirtiendo a este en un lenguaje multiplataforma e independiente de cualquier empresa, dando lugar al lenguaje ECMAScript.

A partir de ese momento JavaScript es la implementación desarrollada por Netscape del estándar ECMAScript. [19]

En los últimos años la popularidad de este lenguaje se ha visto incrementada, siendo uno de los lenguajes más usados e importantes para el desarrollo web.

Las principales características son:

- **Compatibilidad:** es compatible con la mayoría de los navegadores webs.
- **Integración:** permite una fácil integración con otras tecnologías como HTML, CSS y base de datos.
- **Actualizado:** está en constante evolución y mantenimiento, por lo que se va a encontrar actualizado en todo momento.
- **Buen rendimiento:** tiene un funcionamiento eficiente y con buen rendimiento.
- **Facilitadores:** existen numerosos *frameworks* o bibliotecas que facilitan el trabajo a los desarrolladores.

4.2.4 Twig

Twig [20] no es un lenguaje propiamente dicho, sino que es un motor de plantillas (*templates*) utilizado por el lenguaje de programación PHP, pero es importante conocer acerca de él, por las facilidades que aporta durante el desarrollo del proyecto.

Fue creado en 2009 por Fabien Potencier, fundador del *framework*, utilizado en este proyecto y ya explicado anteriormente, Symfony. El objetivo principal detrás de la creación de este motor fue mejorar la forma en que se manejaban las plantillas usadas por Symfony. Intentaba ser más seguro, rápido y fácil de usar.

Las principales ventajas de este motor son:

- **Conciso:** la sintaxis utilizada es más concisa que la usada por PHP lo que hace la lectura más fácil.
- **Orientado en la sintaxis:** al tener una sintaxis más sencilla se puede utilizar ésta como elemento clave para el desarrollo, por ejemplo, usando atajos o iteradores sobre estos.
- **Herencia:** permite la herencia de plantillas, es decir, una vez creada una plantilla se puede utilizar en otras plantillas sin necesidad de volver a escribir todo de nuevo.
- **Seguridad:** uno de los objetivos principales de Twig es la seguridad, para ello se ha reducido el riesgo de posibles vulnerabilidades.
- **Rápido:** se optimiza lo máximo posible todo el código de PHP para así tener un mejor rendimiento.
- **Documentado:** consta de una documentación completa que facilita el uso de este motor.

En nuestro proyecto ha facilitado el uso y desarrollo de las plantillas.

4.2.5 CSS

CSS (*Cascading Style Sheets*) [21] u hojas de estilo en cascada, es un lenguaje de diseño gráfico usado para establecer el diseño visual de documentos y para crear páginas web atractivas con una mejor interfaz gráfica.

Las hojas de estilos existían desde la creación del lenguaje de etiquetas SGML, con el que se observó que era necesario definir un sistema que permitiese aplicar diferentes estilos a los documentos electrónicos. Con el alcance que consiguió HTML, y la popularidad de las páginas web, era necesario facilitar la creación de los documentos y más concretamente la creación de documentos con el mismo estilo.

En 1994 Håkon Wium Lie y Bert Bos definieron un nuevo lenguaje llamado CSS, que unía lo mejor de las dos propuestas que habían sido lanzadas por parte de W3C para el lenguaje HTML. Fue en 1995 cuando W3C decidió estandarizar CSS y añadirlo a su grupo de trabajo de HTML.

La primera versión lanzada fue CSS1 en 1996, y permitía establecer la fuente, el color y la ubicación de los elementos en una página web de HTML. En 1998 es lanzada la segunda versión CSS2 donde se añadieron nuevas mejoras entre las que cabe destacar el soporte para impresión, es decir, el poder controlar la apariencia cuando se imprime un documento. [22]

La versión actual de CSS se podría decir que es la CSS3, aunque no es una versión como tal, sino que se trata de un conjunto de diferentes especificaciones lanzadas de manera individual. Cada especificación agrega nuevas características. A pesar de que CSS3 ha mejorado la compatibilidad con los dispositivos, existe todavía algunos problemas a la hora de utilizarlo en todos los navegadores, puesto que no todos soportan todas las especificaciones lanzadas.

Las principales ventajas de CSS son:

- **Accesibilidad:** permite mejorar la accesibilidad de las páginas web permitiendo incluir información adicional para ayudar al usuario.
- **Reusabilidad:** el código de CSS nos permite reutilizar el código sin necesidad de volver a escribirlo.
- **Limpieza de código:** el código desarrollado con CSS mantiene un orden y limpieza, evitando así problemas innecesarios.
- **Separación de la presentación del contenido:** CSS se utiliza principalmente para la presentación, lo que permite poder separar esta parte del contenido.
- **Menor almacenamiento:** el almacenamiento utilizado se ve reducido debido a que una vez se ha cargado una hoja de estilo no es necesario recargarla, aunque sea usada en otro elemento de la página.

En nuestro proyecto es utilizado para separar el contenido creado por HTML de su presentación, es decir, ha sido utilizado para la creación de los estilos requeridos.

4.2.6 SQL

SQL (*Structured Query Language*) [23] es un lenguaje diseñado para manejar y administrar bases de datos relacionales.

Su historia comienza en 1970 de la mano de IBM y del desarrollo, por parte de esta empresa, de base de datos relacionales. Fueron Edgar Frank Codd y Donald Chamberlin los principales encargados de proponer el modelo relacional. Este

modelo utiliza tablas para almacenar información y establece relaciones entre ellas.

Utilizando estas ideas, se desarrolló el lenguaje SEQUEL (Structured English Query Language), que sería implementado en el sistema de gestión de base de datos System R. Pero no fue hasta 1979, cuando Oracle introdujo este lenguaje en un producto comercial popularizando el uso de este, gracias al lanzamiento de su sistema de gestión de base de datos Oracle Database. SEQUEL fue el predecesor de SQL.

En 1986 ANSI (*American National Standards Insti*) estandarizó SQL. Esta versión se conoció como SQL 1 o SQL-86. Al año siguiente ISO estandarizó también este lenguaje lo que permitió su uso a nivel internacional.

En 1992 fue lanzado un nuevo estándar, que ampliaba las capacidades del ya existente y revisaba lo lanzado anteriormente, creando el SQL-92 o SQL-2. [24] [25]

Actualmente SQL se ha convertido en el lenguaje más usado en lo que a base de datos se refiere.

Las ventajas de este lenguaje son:

- **Gran capacidad:** SQL cuenta con la capacidad de manejar grandes cantidades de datos, lo que permite utilizar este lenguaje en proyectos de gran tamaño.
- **Eficiencia:** las consultas SQL son ejecutadas en poco tiempo, lo que genera gran eficiencia.
- **Lenguaje estandarizado:** cuenta con una trayectoria longeva y con una alta documentación, creando así una plataforma uniforme para todos los usuarios.
- **Facilidad de aprendizaje:** la existencia de tanta documentación y la sencillez de su lenguaje permite un aprendizaje fácil.
- **Flexibilidad:** SQL se puede utilizar en diferentes aplicaciones, permitiendo ser usado con las herramientas elegidas por el desarrollador.
- **Seguridad:** SQL está previsto con diferentes mecanismos que lo convierte en un lenguaje seguro, priorizando la protección de sus datos.

En este trabajo SQL ha sido elegido para manejar la base de datos con ayuda de Doctrine.

4.2.7 Doctrine

Doctrine [26] es un conjunto de herramientas usadas por diferentes softwares que proporcionan un ORM a PHP.

Un ORM como se ha comentado previamente, es capaz de trasladar los datos de una base de datos a un sistema de clases y objetos, permitiendo utilizar los métodos propios de objetos en vez de interactuar con la propia base de datos, que sería más complicado de hacer. Las clases son las tablas de la base de datos y los objetos son los registros.

La primera versión de Doctrine fue lanzada en 2008, por, el desarrollador y creador de SensioLabs, Fabien Potencier. Esta primera versión permitía interactuar con las bases de datos relacionales mediante el uso de objetos PHP, llamados entidades. [27]

En 2010 se lanzó Doctrine 2, que mejoraba el rendimiento y facilitaba el uso de Doctrine para los usuarios. Esta versión se estableció como la predeterminada para Symfony. [28]

En la actualidad ha seguido evolucionando y convirtiéndose en una de las soluciones ORM más populares.

Las ventajas principales de Doctrine son:

- **Facilidad de uso:** posee un aprendizaje muy sencillo, lo que permite la fácil iniciación en él.
- **Soporte para todo tipo de relaciones:** un problema existente con las ORM es la dificultad que existe de poder tratar todos los tipos de relaciones, pero con Doctrine es posible.
- **Escalabilidad:** Doctrine puede manejar gran cantidad de datos y aumentar su capacidad según sea necesario.
- **Soporte para plataformas:** como se ha comentado Doctrine es capaz de trabajar con diferentes bases de datos.
- **Eficiencia:** las consultas de Doctrine pueden ser ejecutadas rápidamente y sin necesidad de muchos recursos.

En este proyecto se utiliza como ORM para facilitar el manejo de datos.

4.3 Herramientas utilizadas

4.3.1 PHPStorm

PHPStorm [29] es un IDE (*integrated development environment*) para el desarrollo de código PHP.

El comienzo de este entorno de desarrollo es en 2010, cuando la compañía JetBrains decide desarrollar un IDE específico para PHP. En la actualidad es uno de los IDEs más utilizados por los desarrolladores de PHP.

Las ventajas de este entorno de desarrollo son:

- **Sistemas de control de versiones:** permite integrar sistemas de control de versiones como Git, facilitando la gestión de versiones sin aplicaciones extra.
- **Autocompletado de código:** incluye procesos de autocompletado de código que facilita y agiliza la escritura.
- **Depuración:** contiene diferentes herramientas que permiten el análisis y corrección de errores.
- **Personalización:** permite personalizar el entorno, adaptándose según las necesidades o gustos del desarrollador.
- **Refactorización:** ofrece herramientas que permiten reestructurar y optimizar el código sin cambios manuales.

En este trabajo PHPStorm ha sido el IDE elegido para la escritura del código PHP, HTML, JavaScript y CSS.

4.3.2 MySQL Workbench

MySQL Workbench [30] es una herramienta visual para el diseño de base de datos y que permite la gestión y mantenimiento de los datos de SQL. Esta herramienta proporciona una interfaz gráfica fácil de usar a los desarrolladores, para que el manejo de las bases de datos sea más intuitivo y natural.

Las ventajas de esta herramienta son:

- **Interfaz Gráfica de Usuario:** como se ha comentado previamente, MySQL Workbench proporciona una interfaz gráfica intuitiva y sencilla, facilitando al programador el desarrollo.
- **Diseño de base de datos:** incluye también las herramientas necesarias para el diseño y la creación de bases de datos, además de generar código SQL.
- **Integración:** permite la integración con otras herramientas mejorando así la eficiencia.
- **Accesible:** está disponible en la mayoría de los sistemas operativos.
- **Seguridad:** contiene diferentes herramientas que ofrecen seguridad a sus usuarios.
- **Almacenamiento:** posee un gran almacenamiento que le permite soportar gran cantidad de datos.

En este proyecto se ha utilizado para la visualización de la base de datos y la inserción de datos a las tablas creadas previamente con Doctrine.

5 Desarrollo de la aplicación web

Para poder realizar la competición de resolución de misterios la aplicación web tiene que cubrir una serie de requisitos:

- Una pantalla principal en la que se explica el funcionamiento y la puntuación de la competición y un botón que permita acceder a ésta.
- Un apartado donde poder introducir las palabras claves que el alumno haya encontrado. Este apartado tiene que verificar si la palabra introducida es una de las palabras clave y si no ha sido introducida previamente. Si cumple estos dos requisitos se le devuelve al estudiante el enunciado de un problema relacionado con la asignatura.
- Un apartado que permita ver todos los problemas que el alumno ha conseguido, permitiéndole acceder a cada uno de ellos.
- Una pantalla para cada uno de los problemas a los que tiene acceso el usuario, en la que poder volver a descargar el problema y acceder a resolver éste.
- Una pantalla en la que poder responder a cada problema al que se acceda, pudiendo observar las respuestas correctas dadas y en caso de haber resuelto el problema devolver al alumno una pista para el misterio final.
- Un apartado en el que poder ver todas las pistas conseguidas para el misterio final.

5.1 Árbol de directorios

En este apartado se va a explicar la estructura que tiene el árbol de directorios tras el desarrollo de la aplicación. Este árbol está formado por las carpetas y las clases que forman el proyecto tal y como se observa en la figura número 10.

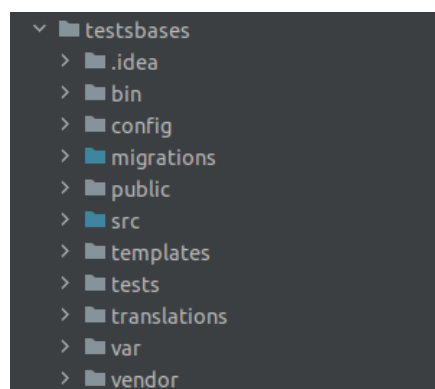


Figura 10. Estructura de las carpetas

5.1.1 .Idea

Como se puede observar en la figura 11, el nombre de esta carpeta comienza con un '.' indicando que es una carpeta oculta para el usuario. Esta carpeta se crea automáticamente al utilizar cualquier IDE de IntelliJ como es el caso de PhpStorm, y contiene la configuración específica del IDE para el proyecto y el desarrollador no suele modificarla, por lo que no se va a entrar en detalle sobre esta carpeta.

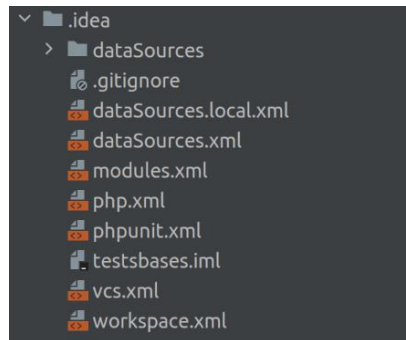


Figura 11. Carpeta .idea

5.1.2 Bin

Esta carpeta es generada en todos los proyectos Symfony y contiene los archivos ejecutables necesarios para el proyecto. Como podemos observar en la figura 12, en nuestro trabajo existen dos archivos:

- Console: este archivo es el ejecutable que permite interactuar con la consola de Symfony, permitiendo ejecutar comandos que son usados por ejemplo para el inicio de nuestro servidor, la creación de migraciones, etc.
- Phpunit: este archivo, es el ejecutable utilizado para lanzar las pruebas unitarias, es decir, el encargado de detectar errores o fallos en el código.



Figura 12. Carpeta bin

5.1.3 Config

Esta carpeta contiene los archivos de configuración del proyecto.

En la figura 13, podemos observar que esta carpeta está formada por otros dos directorios:

- Packages: contiene los archivos que almacenan la configuración de los paquetes instalados en el proyecto, habitualmente instalados mediante Composer. Es importante destacar que los nombres de los archivos deben coincidir con el nombre del paquete instalado.
- Routes: dentro de esta carpeta se encuentran los archivos que contienen la configuración para definir y manejar las rutas. Las rutas son las direcciones que nos permiten acceder a los recursos y a los sitios web.

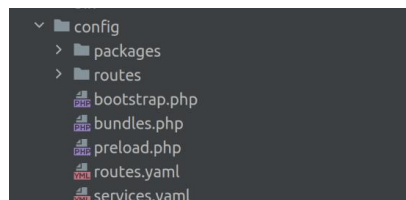


Figura 13. Carpeta Config

Dentro de estas carpetas podemos encontrar otras tres carpetas diferentes: dev, prod y test. Esto se debe a que se puede utilizar diferente configuración para el momento del desarrollo (dev), de la producción (prod) y de la realización de

pruebas (test), esto se consigue creando el mismo archivo especificando a qué momento hace referencia y con los cambios necesarios.

Algunos de los archivos que hay en la carpeta config son:

- Doctrine.yaml: contiene la configuración para el uso de Doctrine.
- Security.yaml: define las diferentes configuraciones de seguridad.
- Services.yaml: define los diferentes servicios y sus respectivas dependencias.
- Annotations.yaml: permite configurar las anotaciones de Symfony. Estas anotaciones añaden información adicional a las clases y a los métodos y aportan nuevas propiedades.

5.1.4 Migrations

Este directorio está presente en todos los proyectos que hagan uso de Doctrine, que como se ha comentado anteriormente, es la herramienta ORM utilizada en Symfony. En este directorio se almacenan las migraciones.

Las migraciones son los archivos utilizados para actualizar la base de datos, aplicando los cambios que sean necesarios en esta.

Para que los cambios sean efectivos se utilizan dos comandos:

1. `php bin/console make:migration` : que crea el archivo de la migración con los cambios que se hayan hecho.
2. `php bin/console doctrine:migrations:migrate` : que ejecuta la migración actualizando la base de datos con los cambios realizados.

5.1.5 Public

Este directorio es creado en todo proyecto Symfony y contiene todos los archivos públicos que van a ser accesibles desde la web. Estos archivos son de diferente tipo como se puede ver en la figura 14, por ejemplo, archivos JavaScript, imágenes, hojas de estilo y las fuentes utilizadas, entre otros.

En este directorio también se encuentra el archivo `index.php`, que es el encargado de cargar y ejecutar la aplicación una vez se accede a la URL correspondiente.

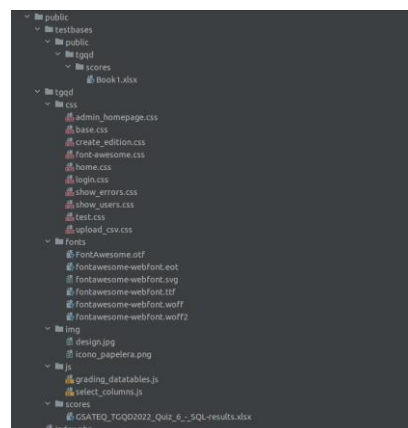


Figura 14. Carpeta Public

5.1.6 Src

La carpeta src es la carpeta que contiene todos los archivos PHP que controlan toda la lógica de la aplicación. Contiene el código fuente de la aplicación Symfony.



Figura 15. Carpeta src

Como podemos observar en la figura 15, dentro de la propia carpeta se pueden crear subdirectorios para mejorar la organización. En el caso de este proyecto estos subdirectorios son:

- **Controller:** almacena los controladores de la aplicación. Un controlador es el encargado de manejar diferentes acciones necesarias para la lógica del programa, para acceder a la base de datos o para devolver una respuesta. Este subdirectorio contiene las siguientes clases (las funciones que hayan sido desarrolladas serán explicadas más adelante):
 - **AdminController.php:** es el controlador encargado de las acciones específicas para los administradores. Todas las rutas que se utilizan en estas acciones contienen en su URL “/admin”.
 - **BaseController.php:** es una clase propia de Symfony y que se puede utilizar como base para el resto de los controladores.
 - **HomeController:** es la clase encargada de controlar la pantalla principal o *home page*.
 - **SecurityController:** se encarga de controlar las acciones de inicio de sesión, de registrar usuarios y de cerrar sesión.
 - **TestController:** maneja las acciones de la prueba de SQL de la competición The Great Quiz Of Databases.
 - **MysteryController:** maneja las acciones para la competición de resolución de misterios.
- **Entity:** este subdirectorio almacena las entidades de la aplicación. Como ha sido explicado anteriormente, en este proyecto se ha hecho uso de Doctrine para facilitar el manejo de base de datos a través de ORM. Con este modelo ORM se crean entidades que hacen referencia a cada una de las tablas existentes en las bases de datos y hacen uso de anotaciones PHP. Como podemos observar en la figura 16 hay una gran cantidad de entidades, que serán explicadas individualmente más adelante.

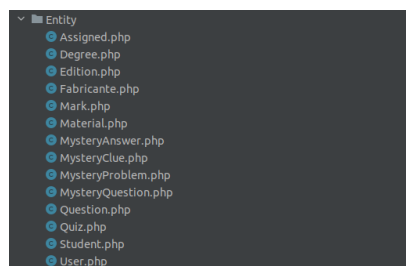


Figura 16. Subdirectorio Entity

- **Form:** es el encargado de almacenar todos los formularios creados para la aplicación. Los formularios son clases de PHP que utilizan la biblioteca Symfony/form [31] , que permiten crear formularios personalizados utilizados para almacenar información de los usuarios, ya sea datos del propio usuario o la respuesta a los ejercicios. Los formularios empleados son:
 - `EditionFormType.php`: formulario para la creación de ediciones por parte del administrador.
 - `KeyForm.php`: formulario que permite introducir al usuario una palabra clave.
 - `ScoreFormType.php`: se utiliza para poder insertar la puntuación obtenida por los alumnos en competiciones externas.
 - `StudentFormType.php`: se utiliza para el registro de estudiantes nuevos a la aplicación.
 - `TestForm`: permite la creación de las preguntas relacionadas con la parte de SQL de la competición existente.
 - `UserFormType.php`: es el formulario utilizado para el inicio de sesión del usuario en la aplicación.
- **Repository:** contiene todos los archivos conocidos como repositorios, estos archivos son los encargados de realizar las consultas a las bases de datos para recuperar entidades. Estas clases crean una capa de abstracción para el usuario y tiene que existir una por cada entidad que haya sido creada en la aplicación. En la figura 17 se pueden ver todos estos archivos.

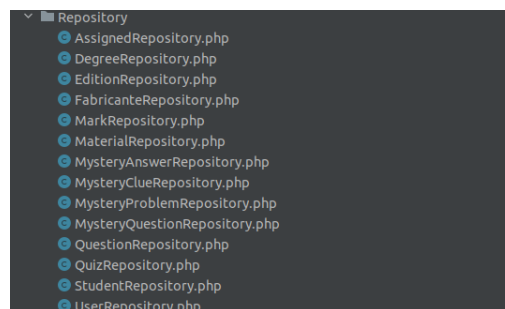


Figura 17. Subdirectorio Repository

- **Security:** contiene el archivo `LoginFormAuthenticator.php` que sirve para controlar si los datos introducidos en el inicio de sesión son correctos, y añade seguridad a esta acción.
- **Service:** se almacenan los servicios creados por el usuario para acciones específicas.
- **Validator:** es un subdirectorio creado por Symfony para las validaciones personalizadas de la aplicación. Una validación se utiliza para que los datos que sean introducidos por parte del usuario sigan las restricciones que se necesiten. En la figura 18 se puede observar este directorio.

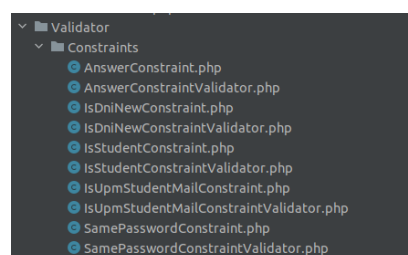


Figura 18. Subdirectorio Validator

5.1.7 Templates

Este directorio es el encargado de almacenar las plantillas o *templates*, estos archivos son los archivos HTML que van a servir para la organización del contenido de la página web. Se han creado tres subdirectorios para facilitar la estructura y que sea más sencillo entenderla, estos tres subdirectorios son:

- Admin: se encuentran todas las plantillas referidas a las pantallas utilizadas únicamente por los administradores.
- Main: se encuentran todas las plantillas que son usadas en el proyecto que no estén en los otros subdirectorios, y que hacen referencia a la mayoría de las funciones principales de la página.
- Security: se encuentran únicamente las plantillas usadas para el inicio de sesión o para el registro de usuarios y estudiantes.

Además de estos subdirectorios encontramos un archivo `base.html.twig` que van a utilizar el resto de plantillas como base, como se puede observar en la figura 19.



Figura 19. directorio templates

5.1.8 Translations

Es un directorio que está presente en todos los proyectos Symfony y que se encarga de almacenar los archivos de traducción de la aplicación. En el caso de este proyecto no ha sido utilizado.

5.1.9 Var

Este directorio no es modificado directamente por el usuario, sino que en él se almacenan los datos que se generan durante la ejecución de la aplicación sin formar parte del código.

Existen dos subdirectorios propios de este directorio:

- Cache: almacena los archivos de caché que se generan durante la ejecución.
- Logs: almacena los archivos de registro que se generan.

Este archivo es propio de las aplicaciones Symfony.

5.1.10 Vendor

Es un directorio común para las aplicaciones de Symfony, encargado de almacenar los archivos que administran las dependencias Composer. Estas dependencias son paquetes instalados para ser usados por la aplicación.

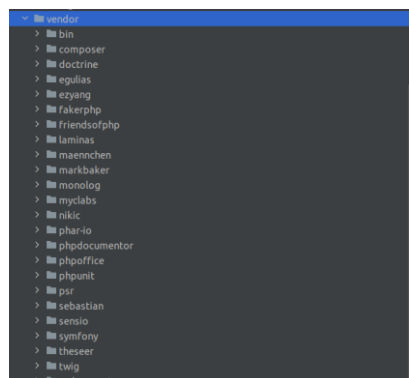


Figura 20. directorio vendor

5.2 Base de datos

5.2.1 Creación de entidades

Antes de comenzar con la explicación de la base de datos, es necesario explicar cómo se han creado las entidades que van a ser las tablas de la base de datos.

Como se ha mencionado anteriormente, se ha hecho uso de Doctrine como ORM, facilitando el manejo de la base de datos. Para empezar, se ha de configurar la base de datos, esto se hace modificando el archivo .env que se puede encontrar en el proyecto. Los parámetros que se han configurado y sus valores se pueden observar en la figura 21.

```
DB_USER=phpmyadmin
DB_PASSWORD=1234
DB_HOST=localhost
DB_PORT=3306
DB_NAME=bd_pie
DATABASE_URL=mysql://${DB_USER}:${DB_PASSWORD}@${DB_HOST}:${DB_PORT}/${DB_NAME}
###< doctrine/doctrine-bundle ###
```

Figura 21. parámetros de la base de datos

Tras la configuración de este archivo, se crea la base de datos mediante el uso del comando “php bin/console doctrine:database:create”. Como se ha explicado previamente, en el archivo doctrine.yaml, que encontramos en el directorio config, podemos configurar las diferentes opciones de esta base de datos como por ejemplo la versión.

Una vez creada la base de datos hay que crear las entidades, esto se realiza también mediante la consola con el uso de un comando. En la figura 22 vamos a observar el uso del comando “php bin/console make:entity” para la creación de una entidad Ejemplo.

```
campo@ubuntu:~/Desktop/testsbases/apps/testsbases$ php bin/console make:entity

Class name of the entity to create or update (e.g. FierceKangaroo):
> Ejemplo

created: src/Entity/Ejemplo.php
created: src/Repository/EjemploRepository.php

Entity generated! Now let's add some fields!
You can always add more fields later manually or by re-running this command.

New property name (press <return> to stop adding fields):
> nombre

Field type (enter ? to see all types) [string]:
> string

Field length [255]:
> 255

Can this field be null in the database (nullable) (yes/no) [no]:
> no

updated: src/Entity/Ejemplo.php

Add another property? Enter the property name (or press <return> to stop adding fields):
>

Success!

Next: When you're ready, create a migration with php bin/console make:migration
```

Figura 22. Ejemplo de creación de una entidad

Los pasos que se siguen para la creación son los siguientes:

1. Uso del comando “php bin/console make:entity”, creando el archivo php correspondiente a la entidad y su correspondiente repositorio.
2. Nombre de una de las propiedades de dicha entidad.
3. Tipo de la propiedad.
4. Según el tipo que se haya seleccionado este paso será diferente. En el caso de seleccionar el tipo *string*, se tiene que definir su longitud.
5. Indicar si la propiedad puede ser nula o no.
6. Si se desea añadir más propiedades se repetiría estos pasos, sino se puede dar por finalizada la creación de la entidad.

A pesar de que la entidad se haya creado, todavía no existe en la base de datos, para esto es necesario el uso de los comandos que se han mencionado en el apartado de migraciones “php bin/console make:migration” y “php bin/console doctrine:migrations:migrate”. El primero de los comandos va a crear la migración permitiendo ver las sentencias SQL que se van a ejecutar y el segundo de ellos ejecuta la migración.

Una vez la migración se ejecuta, se puede dar por finalizada la creación de la entidad y la actualización en la base de datos.

Para modificar una entidad que ya existe, se utiliza el mismo proceso, ya que Doctrine identifica que la entidad existe en la base de datos para su posterior modificación, esto se puede observar en la figura 23.

```
campo@ubuntu:~/Desktop/testsbases/apps/testsbases$ php bin/console make:entity

Class name of the entity to create or update (e.g. FierceKangaroo):
> User

Your entity already exists! So let's add some new fields!

New property name (press <return> to stop adding fields):
> 
```

Figura 23. Ejemplo de modificación de una entidad

5.2.2 Explicación de la base de datos

Ahora que se ha comentado como se crean las entidades, se va a explicar la base de datos usada en el proyecto, y la explicación de cada una de las entidades creadas o modificadas.

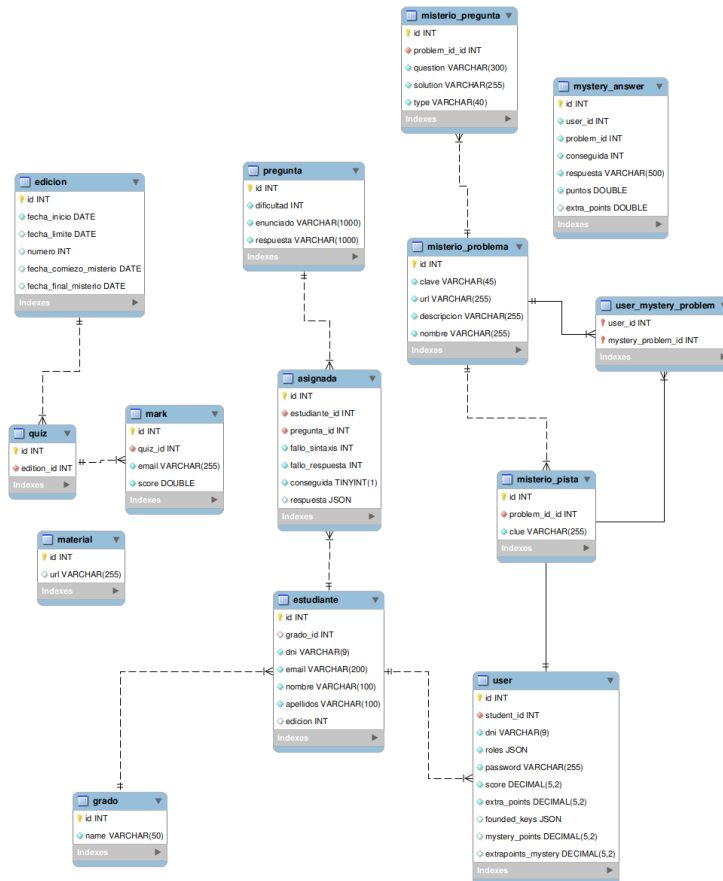


Figura 24. Diagrama de tablas de la base de datos

Como se ha mencionado anteriormente, el lenguaje utilizado ha sido SQL mediante la herramienta Doctrine como ORM para el manejo de las tablas o entidades y MySQL Workbench como la herramienta para la visualización de la base de datos y para la inserción de datos.

En la figura 24 se puede observar el diagrama de tablas de la base de datos, en este podemos observar todas las tablas que existen en la base de datos y como están relacionadas entre sí. Las entidades existentes son:

- Estudiante: contiene todos los datos del estudiante que se generan mediante el registro.
- User: almacena la información del usuario que va a ser utilizada para las competencias.
- Grado: hace referencia al grado en el que se encuentra matriculado el alumno.
- Asignada: contiene la información sobre las preguntas del Quiz de SQL que han sido asignadas a un estudiante, y las respuestas de este mismo.
- Pregunta: representa cada una de las preguntas del Quiz de SQL.
- Material: contiene la URL con el material de clase.

- Quiz: hace referencia a cada semana de la competición de SQL.
- Edición: contiene la información de cada una de las ediciones realizadas.
- Mark: almacena la puntuación conseguida por un usuario en el quiz semanal.
- Misterio_pista: contiene la información para cada una de las pistas de la competición.
- Misterio_problema: almacena la información de los problemas de la competición de resolución de misterios.
- Misterio_pregunta: hace referencia a una de las preguntas relacionadas con un problema concreto de la resolución de misterios.
- Mystery_answer: contiene todas las respuestas dadas por todos los usuarios para todos los problemas existentes.
- User_mystery_problem: una vez el usuario ha resuelto correctamente un problema se relaciona con él.

Algunas de estas entidades no son necesarias para el desarrollo de las nuevas funcionalidades que se han realizado en este proyecto. En la figura 25, se pueden observar cuales son las tablas que han sido creadas o modificadas para este trabajo, en esta figura no aparecen entidades como estudiante o grado que a pesar de ser usadas en las nuevas funcionalidades no han sido modificadas ni creadas para este trabajo.

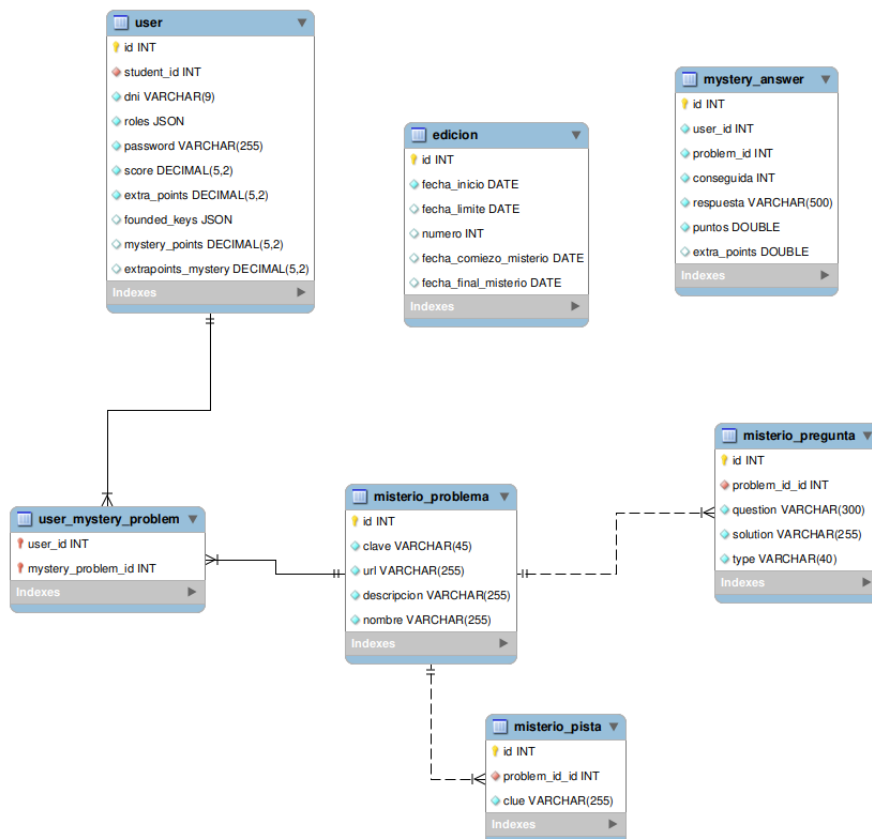


Figura 25. Diagrama de tablas de la base de datos de la resolución de misterios.

Están entidades son las que vamos a comentar a continuación en más detalle.

5.2.2.1 User

Es la entidad que hace referencia al usuario que utiliza la aplicación, por ello muchos de los datos que se completan en el registro no son almacenados en esta tabla, ya que no tienen importancia para la aplicación, sino que son relevantes para la asignatura o el profesor.

La entidad Doctrine que está relacionada con esta tabla es User.php y el repositorio es UserRepository.php.

Los parámetros de esta entidad son los siguientes:

- Id: es un número autogenerated que permite identificar al usuario frente al resto de usuarios. Se utiliza como clave primaria.
- Student_id: es el id que relaciona al usuario con un estudiante de dicha tabla, permitiendo acceder a los datos del registro en caso de ser necesario. Actúa como clave externa.
- DNI: es el DNI del usuario.
- Roles: es un *JSON* que almacena todos los roles que posee este usuario en la aplicación. Es importante este valor porque si tiene algún rol especial, como el rol de administrador, puede acceder a unas funciones que un usuario normal no.
- Password: es un *string* que almacena la contraseña. Esta contraseña no se almacena tal cual es escrita por el usuario, sino que se almacena encriptada para aumentar la seguridad.
- Score: almacena la puntuación que ha conseguido el usuario en la competición de SQL.
- Extra_points: son los puntos externos a la competición de SQL que haya conseguido el alumno en la asignatura.
- Founded_keys: es un *json* que almacena todas las palabras claves de la resolución de misterios que ha encontrado el alumno e introducido en la aplicación.
- Mystery_points: son los puntos que ha conseguido el alumno en toda la competición de resolución de misterios.
- Extrapoints_mystery: son los puntos extras que se consiguen en la resolución de misterios, dependiendo del momento en que se haya respondido la pregunta.

5.2.2.2 Edición

Esta entidad hace referencia a la edición de la competición. Cada curso la competición se desarrolla con unos parámetros diferentes que es importante almacenar.

La entidad Doctrine que se utiliza es Edition.php y el repositorio EditionRepository.php.

Los parámetros de esta tabla son:

- Id: es un número asignado por SQL para identificar cada entidad de esta tabla. Es la clave primaria
- Fecha_inicio: hace referencia a la fecha en la que comienza la competición de SQL.
- Fecha_limite: hace referencia a la fecha en la que finaliza la competición de SQL.
- Número: es el número asignado para esta edición.

- Fecha_comienzo_misterio: es la fecha de inicio para la competición de resolución de misterios.
- Fecha_final_misterio: es la fecha en la que finaliza la competición de resolución de misterios.

Las fechas de las competiciones no tienen que ser las mismas ya que pueden comenzar en momentos distintos, por eso es importante almacenarlas separadas.

5.2.2.3 Misterio_problema

Esta tabla almacena la información relacionada con los problemas de la resolución de misterios. Estos problemas son los que se van a tener que resolver para conseguir solucionar el misterio final. Para acceder a alguno de los problemas es necesario haber conseguido la palabra clave correspondiente al problema.

La entidad Doctrine que se relaciona con esta tabla es `MysteryProblem.php` y el repositorio es `MysteryProblemRepository.php`.

Los parámetros de esta entidad son:

- Id: número autogenerated por SQL para identificar cada uno de los problemas. Es la clave primaria
- Clave: es la palabra que hay que introducir en la plataforma para poder acceder a cada problema.
- Url: es la URL que nos permite acceder al enunciado del problema para descargarlo.
- Descripción: es una breve descripción del problema, que sirve como pequeña introducción para saber de qué trata el contenido de este sin necesidad de leer el enunciado.
- Nombre: es el nombre que se le asigna al problema para que sea más fácil su identificación.

5.2.2.4 User_mystery_problem

Esta tabla no es una entidad, sino que se trata de una relación N:M entre las tablas `user` y `misterio_problema`.

Esta relación es necesaria para poder relacionar al usuario con los problemas que ha conseguido completar. Al no ser una entidad como tal no tiene un archivo entidad de Doctrine ni un repositorio, sino que se encuentra como atributo de la entidad `User`, como podemos observar en la figura 26.

```
/**
 * @ORM\ManyToMany(targetEntity=MysteryProblem::class)
 */
5 usages
private $problem_mystery_id;
```

Figura 26. Atributo `problema_mystery_id` de la clase `User`

Como se puede ver el nombre que se le da es `problema_mystery_id` y se utiliza la anotación `ORM\ManyToOne` para referenciar que es una relación en la que cada una de las entidades va a tener múltiples instancias de la otra entidad. Es necesario indicar a que entidad va dirigida la relación, y para ello se utiliza el atributo `targetEntity`.

Las funciones que Doctrine añade para el manejo de esta relación son `getProblemMysteryId()` que devuelve una colección con los problemas que haya conseguido el usuario, `addProblemMysteryId()` que añade a la colección un nuevo problema, para ello esta función recibe como parámetro el `MysteryProblem` que se va a añadir, y `removeProblemMysteryId()` que elimina de la colección un problema que ha sido pasado como parámetro.

5.2.2.5 Misterio_pregunta

Esta tabla es la encargada de almacenar la información de cada una de las preguntas que van a tener los problemas que se han de solucionar para la competición de resolución de misterios.

La entidad Doctrine que hace referencia a esta tabla es `MysteryQuestion.php` y el repositorio es `MysteryQuestionRepository.php`.

Los parámetros de esta entidad son:

- `Id`: es un número autogenerado por SQL para poder identificar cada una de las preguntas. Se utiliza como clave primaria.
- `Problema_id_id`: es un número que se utiliza como clave externa. Esta clave proviene de la tabla `misterio_problema` y sirve para establecer la relación con esta. La relación que existe entre estas tablas es de un problema para varias preguntas.
- `Question`: contiene la pregunta tal y como va a ser mostrada al usuario.
- `Solution`: contiene la respuesta a la pregunta.
- `Type`: indica el tipo de respuesta que hay que dar en la solución. Los tipos existentes son:
 - `Choice`: hay que elegir entre las posibles respuestas que se dan.
 - `Numeric`: la respuesta es solo un número.
 - `Letters`: la respuesta son letras, sin importar el orden en el que se respondan o si son mayúsculas o minúsculas.
 - `Words`: la respuesta es una o varias palabras en las que sí importa el orden, los espacios y las mayúsculas, es decir, la respuesta tiene que estar escrita tal y como es.

5.2.2.6 Misterio_Pista

Esta tabla contiene la información de las pistas que los alumnos tienen que conseguir para poder seguir avanzando en la resolución del misterio final. Cada vez que un problema es resuelto correctamente, se le otorga al usuario una pista nueva.

La entidad Doctrine con la que se relaciona esta tabla es `MysteryClue.php` y el repositorio es `MysteryClueRepository.php`.

Los parámetros que tiene esta entidad son:

- `Id`: es un número autogenerado por SQL que permite identificar cada una de las pistas. Se utiliza como clave primaria.

- `Problem_id_id`: es la clave externa proveniente de la relación existente entre la tabla `misterio_pista` y `misterio_problema`. Esta relación es 1:1, es decir, un problema únicamente va a permitir acceder a una pista.
- `Clue`: es la propia pista que el alumno va a poder ver.

5.2.2.7 **Mystery_answer**

Esta tabla permite almacenar todas las respuestas que han dado todos los alumnos para todos los problemas de la competición. Esta tabla se utiliza para tener un registro de las respuestas y que en un futuro se pueda comprobar cada una de las respuestas.

La entidad Doctrine que se relaciona con esta tabla es `MysteryAnswer.php` y el repositorio es `MysteryAnswerRepository.php`

Los parámetros de esta clase son:

- `Id`: número autogenerado por SQL para identificar cada respuesta. Es utilizado como clave primaria.
- `User_id`: es el id del usuario que ha respondido el problema.
- `Problem_id`: es el id del problema que ha sido respondido.
- `Conseguida`: es un campo identificativo para saber si se ha respondido correctamente al problema o no. Si este valor es 1 es que se ha respondido correctamente, si es 0 es que no se ha respondido correctamente.
- `Respuesta`: almacena todas las respuestas de todas las preguntas del problema que ha sido respondido.
- `Puntos`: son los puntos obtenidos al resolver el problema.
- `Extra_points`: son los puntos extras obtenidos al resolver el problema. Estos puntos son debido a la fecha en la que se haya respondido.

5.3 **Explicación de las clases y las funciones usadas**

En este apartado se va a explicar las clases y las funciones que se han desarrollado para este proyecto, para ello se van a ir explicando las pantallas desarrolladas y las funcionalidades que ofrecen.

5.3.1 **Pantalla principal**

Esta pantalla es a la que se accede una vez se ha hecho el inicio de sesión de manera correcta. Es compartida tanto para la competición ya existente de SQL, como para la competición de misterios que se desarrolla en este proyecto.

Para esta pantalla, el controlador utilizado ha sido el `HomeController.php` y diferentes *templates* que se explicarán en el momento de su uso.

La clase `HomeController` extiende de la clase `BaseController`, como el resto de los controladores de la aplicación.

La ruta utilizada para esta página es `"/base-de-datos"` y la función a la que se llama cuando se accede a esta ruta, es la única función existente en la clase `HomeController`, que recibe el nombre de `homepage`. Como podemos ver en la figura 27 además de asignarle la ruta mediante la anotación de Symfony, también le asignamos un nombre `"app_homepage"`, que será el nombre con el que se hará referencia a esta ruta en el resto de los archivos del proyecto para facilitar la escritura. Esta anotación es usada para todas las funciones que se

van a ver en este proyecto, indicando en cada caso la ruta y nombre correspondiente.

```
/**
 * Función que renderiza el template de la pagina principal segun la fecha de inicio y la fecha limite de cada competición
 * @Route("/base-de-datos/", name="app_homepage")
 */
public function homepage()
```

Figura 27. definición de la ruta para la pantalla principal

Lo primero que se realiza en esta función, es la comprobación de si se está intentando acceder a esta pantalla sin haber iniciado sesión. Si fuese este el caso, el usuario no tendría ningún rol en la aplicación y sería redirigido a la pantalla de inicio de sesión.

Tras esto, se realiza otra comprobación para ver si el usuario tiene el rol de administrador y redirigirle a la pantalla de administrador ya existente.

Tras estas comprobaciones, se adquiere el registro de Doctrine a través de la función de Symfony `getDoctrine()`. Con este registro vamos a poder acceder al repositorio de la edición haciendo uso de la función `getRepository('App:Edition')` y lo mismo para el repositorio de Material `getRepository('App:Material')`.

A partir del usuario accedemos al estudiante para poder conseguir la edición del alumno. Con la edición y el repositorio, vamos a adquirir las fechas de inicio y de fin de las dos competiciones, para ello hacemos uso de las funciones del repositorio `getFinishDateByEdition($edition)`, `getStartDateByEdition($edition)`, `getFinishDateMysteryByEdition($edition)` y `getStartDateMysteryByEdition($edition)`.

En la función `getFinishDateMysteryByEdition($edition)` se realiza una consulta SQL para conseguir la fecha límite de la competición de misterios de la edición correspondiente, que es pasada como parámetro. Para el resto de las fechas, la función es la misma pero el parámetro por el que se realiza la consulta cambia, siendo en cada caso la fecha que se desea obtener.

A partir del repositorio de material, se obtiene la URL del OneDrive de la asignatura, que será utilizado en el *template* para redirigir al alumno, en caso de desearlo, a este sitio web.

Una vez se han obtenido todos estos valores hay que renderizar la plantilla de la pantalla principal, pero no siempre va a ser igual, sino que depende de las fechas de inicio y de fin de las dos competiciones, dando un total de 9 pantallas de inicio diferentes.

Si la fecha límite de la competición de SQL ya se ha pasado, pero la fecha de fin de la competición de resolución de misterios todavía no ha sido alcanzada pero la de inicio sí, quiere decir que la competición de SQL ya se ha finalizado pero la competición de misterios sigue abierta, por lo cual se puede seguir participando. El *template* relacionado con este caso es `'homepage_deadline.html.twig'` y necesita 4 parámetros relacionados con la competición de SQL : las preguntas asignadas al alumno, el total de preguntas, las preguntas respondidas por el estudiante y la URL del material que habíamos obtenido anteriormente.

Una vez renderizado el *template* la pantalla que se muestra al usuario es la que se puede observar en la figura 28. Como el quiz ha finalizado, se muestra el número de preguntas que ha respondido el usuario, la puntuación que ha obtenido en total en la competición de The Great Quiz Of Databases y un botón para redirigir al alumno al material de la asignatura. Por otro lado, en la parte

de la resolución de misterios, como el usuario se encuentra en el momento de realizar la competición, se puede observar la explicación de la propia competición y un botón para comenzarla.

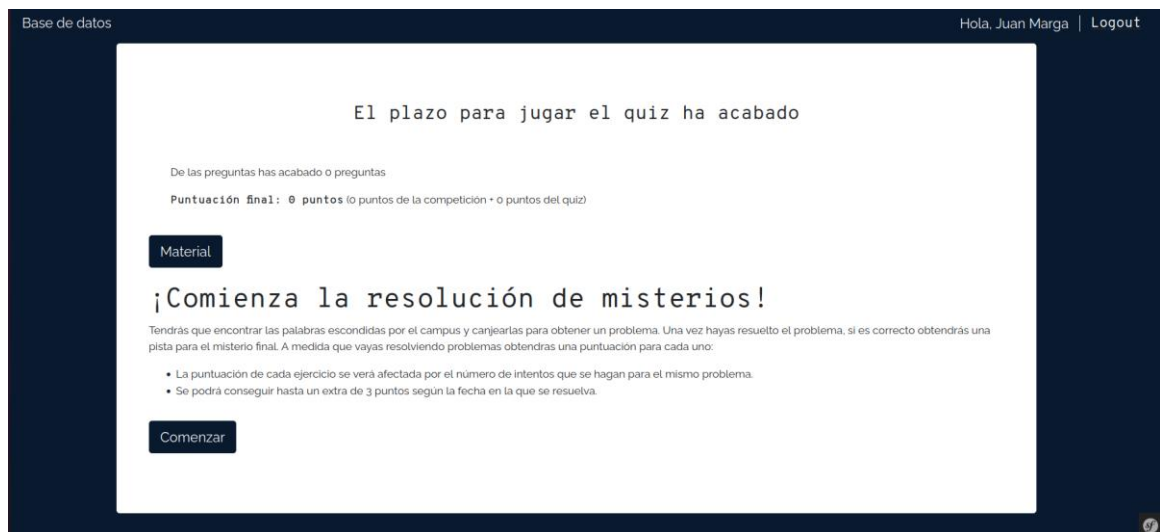


Figura 28. Pantalla principal con el quiz acabado y la resolución de misterios disponible

El segundo caso, hace referencia al momento en que las fechas de fin de las dos competiciones han sido sobrepasadas, por lo que se concluye que las dos competiciones han finalizado su periodo para ser realizadas. El *template* que se va a renderizar es "homepage_deadline_both.html.twig". Y los parámetros que se necesitan para este renderizado son los 4 que eran necesarios en el caso anterior y tres nuevos: los puntos obtenidos en la competición de resolución de misterios, los puntos extra de esta competición y la suma de ambos. Estos valores los obtenemos a partir de la entidad usuario que posee las funciones `getMysteryPoints()` que devuelve los puntos de la competición y `getExtraPointsMystery()` que devuelve los puntos extra de esta competición.

Una vez finalizado el renderizado de esta plantilla, al usuario se le muestra la pantalla que se puede observar en la figura 29. Como se puede ver, en lo que refiere a la competición de The Great Quiz Of Databases no ha cambiado nada respecto al caso anterior. Pero en la parte de la resolución de misterios se puede observar que se muestra la puntuación final obtenida en la competición, además de la explicación del origen esta misma, siendo la suma de los puntos obtenidos en la propia resolución y los puntos extra.

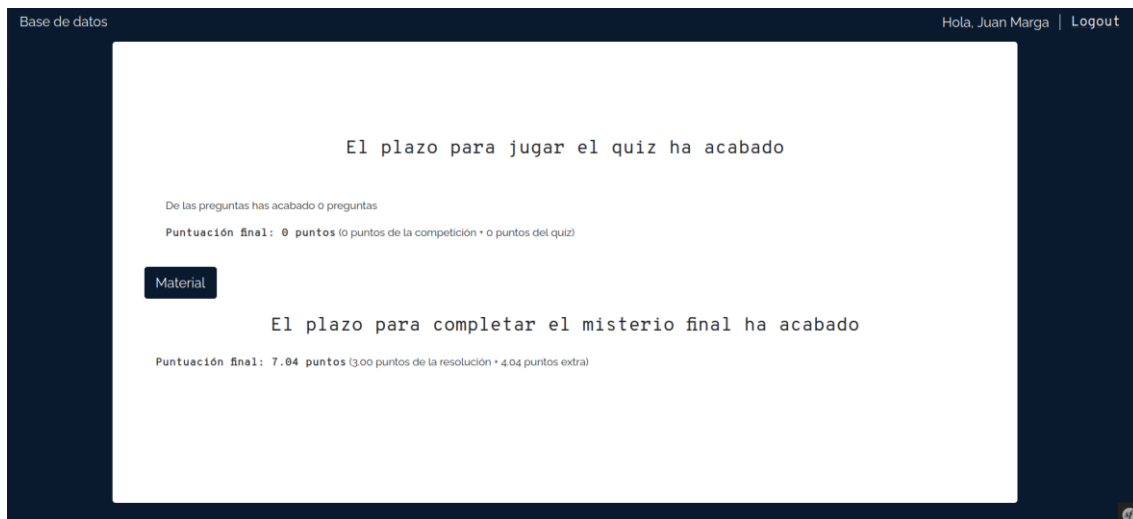


Figura 29. Pantalla principal con el quiz y la resolución de misterios finalizados

El tercer caso hace referencia a cuando la competición de resolución de misterios ha finalizado, es decir, cuando su fecha de fin ya se ha pasado, pero la competición de SQL se encuentra en el plazo para poder ser realizada. Para este caso el *template* utilizado es el de "homepage_deadline_mystery.html.twig" y los atributos que necesita este archivo son: la URL para acceder al material de la asignatura, los puntos totales obtenidos en la competición de resolución de misterios, los puntos únicamente de la resolución de misterios y los puntos extra de dicha competición.

En la figura 30, podemos observar la pantalla que se muestra al usuario una vez se finaliza el renderizado de la plantilla. En la parte de resolución de misterios se puede ver que es igual que en el caso anterior, en el que la competición también había finalizado. En la parte de la competición de SQL se puede observar la explicación de dicha competición, además de un enlace que redirige al diseño de la base de datos en la que hay que realizar las consultas, el botón para comenzar con la competición y el botón que había en las pantallas anteriores que redirige al material de la asignatura.

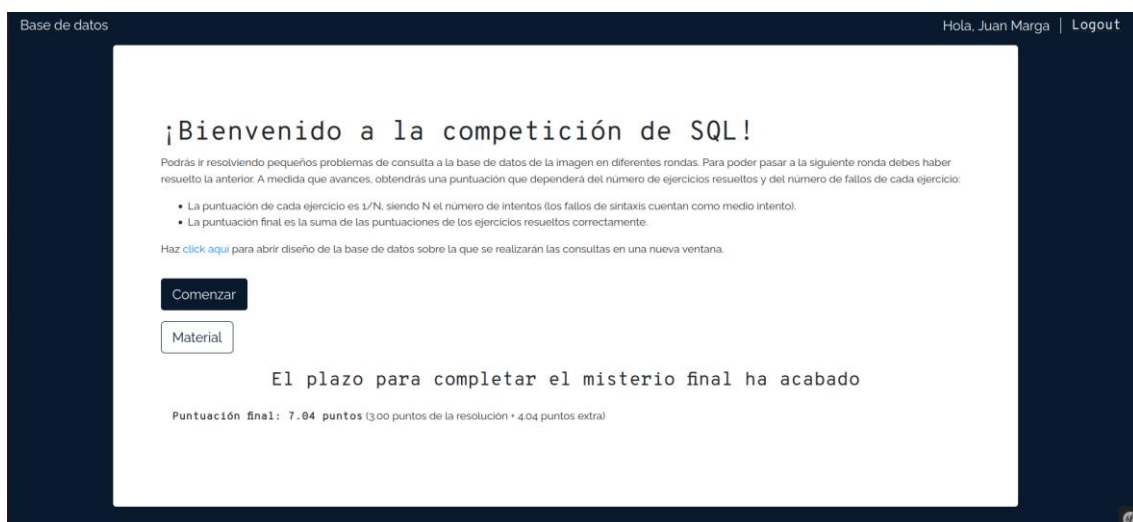


Figura 30. Pantalla principal con el quiz abierto y la resolución de misterios finalizada

El cuarto caso, es el caso en el que todavía las fechas de inicio de las dos competiciones no han sido alcanzadas, es decir, que ninguna de las dos competiciones ha empezado por lo que no se pueden acceder a ninguna de ellas. El *template* que se va a renderizar es “homepage_not_started_bothcompetitions.html.twig” que necesita cuatro atributos: la URL del material de la asignatura, los puntos extra que haya conseguido el usuario en la competición de The Great Quiz Of Databases, y las dos fechas de inicio de las dos competiciones.

La pantalla que se le muestra al usuario una vez renderizado el *template*, se puede observar en la figura 31. Se puede observar una cuenta atrás para el comienzo de ambas competiciones, en este caso son la misma fecha, pero no tiene por qué ser siempre así. También en la parte de la competición de The Great Quiz Of Databases, se incluye la puntuación actual que tiene el usuario, ya que, a pesar de no haber comenzado la competición de SQL, el usuario podría haber obtenido puntuación extra de la otra parte de la competición y un botón que redirige al material de la competición.



Figura 31. Pantalla principal con las dos competiciones sin haber empezado

En el siguiente caso, la fecha de la competición de resolución de misterios si ha sido alcanzada, pero la de la competición de SQL no, por lo que la resolución de misterios ha comenzado y la competición de SQL todavía no ha empezado. El *template* utilizado es “home_not_started.html.twig” y los parámetros que recibe este renderizado son: la URL del material de la asignatura, la fecha de comienzo de la competición de SQL y los puntos extra que haya obtenido.

La pantalla que se renderiza a partir de este *template* es el que se puede ver en la figura 32. La parte del *quiz* no ha cambiado respecto el caso anterior, mostrando la cuenta atrás, la puntuación extra y el botón del material. Por otro lado, la parte de la resolución de misterios ha cambiado, mostrándose la explicación de dicha competición y el botón para comenzarla.

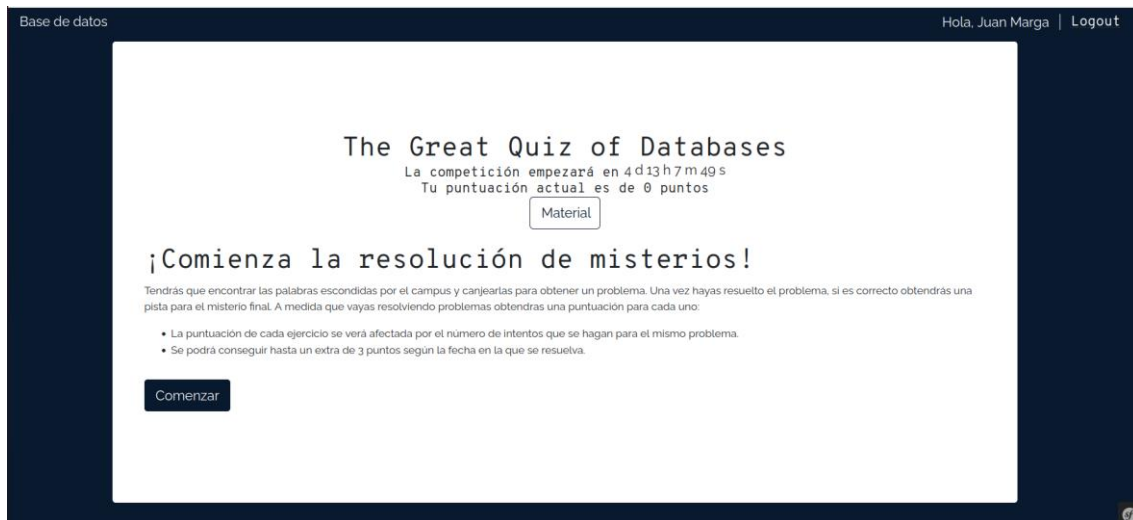


Figura 32. Pantalla principal con el quiz sin haber comenzado y la resolución de misterios sí

El sexto caso, se refiere al caso en el que la competición de SQL se encuentra en el rango de tiempo posible para la realización del *quiz*, pero la competición de misterios no ha comenzado todavía, es decir, no ha llegado la fecha de comienzo de dicha competición. El *template* que se renderiza en este caso es “homepage_not_started_mystery.html.twig” y los parámetros que se necesitan son: la fecha de inicio de la competición de misterios y la URL del material de la asignatura.

En la figura 33 se puede ver la pantalla que se muestra al usuario a partir de esta plantilla. En la competición de SQL se muestra, la explicación de dicha competición, un botón para acceder al material de la asignatura, un botón para comenzar el *quiz* y un enlace a la base de datos con la que hay que trabajar. En la parte referida a la competición de resolución de misterios se muestra la cuenta atrás hasta la fecha de comienzo de la competición.



Figura 33. Pantalla principal con el quiz comenzado y la resolución de misterios sin empezar

El séptimo caso hace referencia a cuando la fecha de la competición de SQL ha finalizado y la fecha de inicio de la competición de resolución de misterios no ha comenzado todavía. El *template* que se renderiza es “homepage_deadlinequiz_notstartedmystery.html.twig” y los parámetros que se necesitan son: el número de preguntas del *quiz* asignadas al alumno, el número de preguntas totales del *quiz*, el número de preguntas finalizadas por el alumno, la URL del material de la asignatura y la fecha de inicio de la competición de resolución de misterios.

En la figura 34 se puede observar la pantalla que se le muestra al usuario al renderizar este *template*. En la parte del *quiz* de SQL al haber terminado, se le muestra la puntuación obtenida y el número de preguntas totales respondidas. En la parte de la resolución de misterios se muestra una cuenta atrás hasta el comienzo de la competición.



Figura 34. Pantalla principal con el quiz terminado y la resolución de misterios sin empezar

El octavo caso, es el caso contrario al séptimo. La competición de resolución de misterios ha llegado a su fecha de fin, es decir, se ha terminado la competición y el *quiz* de SQL no ha comenzado todavía. El *template* que se renderiza es “homepage_notstartedquiz_deadlinemystery.html.twig” y los parámetros que se necesitan son: los puntos extras que haya conseguido el alumno en la competición de The Great Quiz Of Databases, la fecha de inicio del *quiz*, la URL del material de la asignatura, los puntos de la competición de resolución de misterios, los puntos extra de esta competición y la suma de ambos.

En la figura 35 se puede observar la pantalla que es mostrada al usuario al renderizar esta plantilla. Se puede ver una cuenta atrás para la fecha de inicio del *quiz* de SQL además de los puntos obtenidos en la competición The Great Quiz Of Databases y un botón que redirige al material de la asignatura. En la parte de la resolución de misterios se muestra únicamente la puntuación obtenida por el alumno en esta competición.

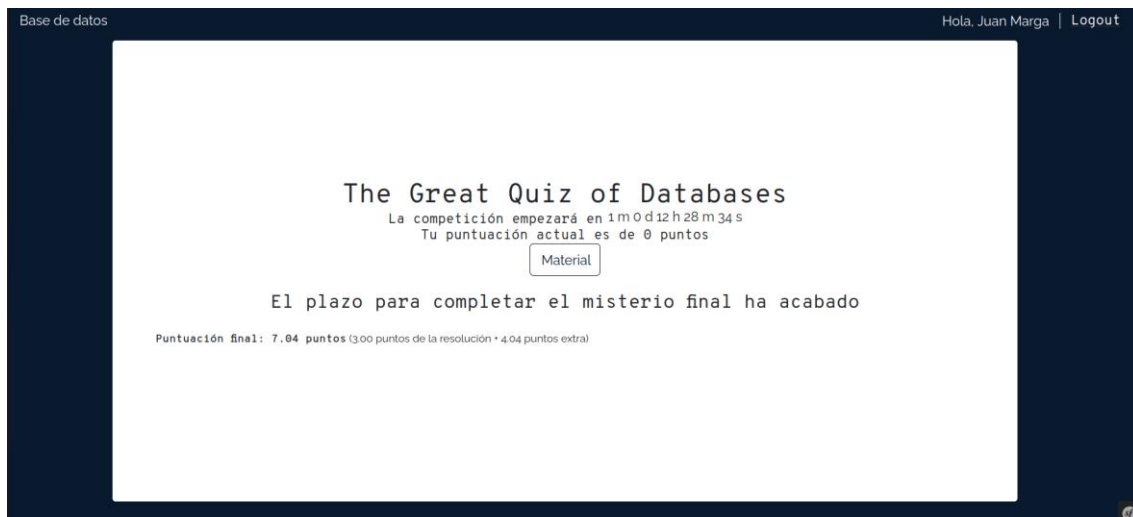


Figura 35. Pantalla principal con el quiz sin comenzar y la resolución de misterios finalizada

El último caso se da cuando las dos competiciones se encuentran en el rango de tiempo que les permite poder realizar las competiciones. El *template* utilizado es “homepage.html.twig” y solo se necesita la URL del material de la asignatura.

La pantalla que se le muestra al usuario es la que se puede observar en la figura 36. En la competición de SQL se muestra la explicación de esta, el enlace para acceder a la base de datos necesaria para la realización de las cuestiones, el botón para comenzar la competición y el botón para acceder al material de la asignatura. En la competición de resolución de misterios, se muestra la explicación de dicha competición y un botón para poder acceder a esta.



Figura 36. Pantalla principal con ambas competiciones disponibles

5.3.2 Pantalla de inicio resolución de misterios

Si la competición de resolución de misterios se encuentra en el rango de tiempo para poder realizarla, se puede acceder a la pantalla de la figura 37 a través del botón de comenzar, que se ha visto previamente. Esta pantalla es la página principal de esta competición. El controlador que maneja esta pantalla es el

“MysteryController.php” que como en el caso anterior también extiende el BaseController. El *template* utilizado es “mystery.html.twig”.



Figura 37. Pantalla principal de la competición de resolución de misterios

Se puede observar que esta pantalla está dividida en tres apartados, que hacen referencia a cada una de las nuevas funcionalidades que se iban a desarrollar.

El primero de los apartados está relacionado con los problemas a los que el usuario puede acceder. Para poder desbloquear más problemas el usuario tiene que introducir la palabra clave que haya encontrado y que está relacionada con un problema. Una vez el problema aparece en la pantalla se va a poder acceder a él pinchando sobre el nombre del problema, lo que redirigirá a otra pantalla que se verá más adelante, pero en la que se podrá observar la descripción, un botón para acceder al enunciado y un botón para resolver dicho problema.

El segundo de los apartados es una lista con las pistas para el misterio final que el usuario ha conseguido. Estas pistas se consiguen resolviendo correctamente uno de los problemas. Cada problema tiene asociado una pista necesaria para la resolución del misterio final.

El tercer de los apartados, es un formulario que permite al usuario introducir nuevas palabras claves. Estas palabras como se ha comentado están relacionadas con un problema, es decir, si la palabra clave introducida es correcta y no ha sido previamente introducida, desbloqueará en la pantalla principal de la competición, un nuevo problema.

La ruta para acceder a la pantalla principal de la competición de resolución de misterios es “base-de-datos/misterio” y el nombre que recibe para ser manejada durante el desarrollo de la aplicación es *mystery*. La función que se encarga de su renderizado es la función con el mismo nombre que la ruta, es decir, la función *mystery* que recibe los siguientes parámetros:

- Request: es un objeto propio de Symfony utilizado en el manejo de formularios para obtener y procesar los datos introducidos en dicho formulario por el usuario, como se verá más adelante.
- EntityManagerInterface: es el objeto que nos permite interactuar con la base de datos a través del ORM.

- `MysteryProblemRepository`: es el repositorio que se utiliza para interactuar con la tabla `misterio_problema` de la base de datos, haciendo uso de las funciones que se verán más adelante.
- `MysteryClueRepository`: es el repositorio que va a permitir interactuar con la tabla `misterio_pista` de la base de datos.

Estos dos últimos parámetros se podrían omitir y acceder a ellos a través del parámetro `EntiryManagerInterface`, pero para facilitar el entendimiento de la función se ha decidido estructurarlos como parámetros individuales.

Lo primero que se realiza en esta función, es la comprobación de que el usuario que está utilizando la aplicación tenga el rol de usuario, es decir, haya iniciado sesión correctamente. Si el usuario no tuviese este rol, significa que no ha realizado el *log in* por lo que no puede acceder a la aplicación y se le redirige a la pantalla de inicio de sesión. Esta comprobación se realiza mediante la función de `Symfony` `isGranted('ROLE_USER')`.

Lo siguiente que se realiza, es comprobar si las fechas de la competición se encuentran en el rango de tiempo que permite la participación en dicha competición. Para ello es necesario obtener la edición en la que se encuentra el usuario, esto se consigue a través del objeto `entityManager` que ha sido pasado como parámetro, con el que se consigue obtener el repositorio de la edición y a partir del usuario obtenemos el estudiante, que tiene como uno de sus valores la edición. Para posteriormente hacer uso de la función del repositorio de la edición, que se ha comentado en el apartado de la pantalla principal, que nos permite acceder a las diferentes fechas de dicha edición. Si la competición se encuentra fuera del rango que permite realizarla, se redirigirá al usuario a la pantalla principal.

Estas comprobaciones se realizan para aumentar la seguridad de la aplicación y que los usuarios no accedan a las funcionalidades sin permiso o en momentos en los que no se debería poder acceder.

Tras la realización de estas comprobaciones se continua con el apartado de problemas. Para poder conseguir la lista de problemas a los que el estudiante tiene acceso, es necesario conocer que palabras claves ha introducido. Esto se consigue a partir de la tabla `Usuario`, que almacena en uno de sus campos un *JSON* con las claves que ha conseguido. Como la tabla `Usuario` se relaciona con la entidad `User`, se accede a este *JSON* a través del objeto `user` que posee la función `getFoundedKeys()` que devuelve un *array* con todas las claves o un objeto nulo en caso de que no hubiese claves.

Una vez conseguido el *array* que almacena las claves se tiene que acceder a los problemas con los que están relacionadas. Esto se realiza mediante la función `findOneBy()` del `MysteryProblemRepository` que necesita como parámetro una colección de parámetros de búsqueda. Esta función realiza una consulta a la base de datos para encontrar el problema con los parámetros pasados, en este caso este parámetro va a ser únicamente una de las palabras claves. Esta función se repite tantas veces como claves contenga el *array* de claves obtenidas. Cada uno de los problemas obtenidos por esta función se almacena en un nuevo *array*.

El siguiente paso es acceder a las pistas que el alumno ha conseguido mediante la resolución de los problemas. Estos problemas que el estudiante ha resuelto se almacenan en la tabla `user_mystery_problem` de la base de datos, que como se ha comentado en el apartado de la base de datos, esta tabla surge de una relación `ManyToMany` entre el usuario y el problema. Para acceder a dicha tabla

se hace uso de la función `getProblemMysteryID()` que posee la entidad `Usuario` y que retorna una colección de los problemas que han sido resueltos por este.

Para cada uno de estos problemas se tiene que acceder a la pista con la que se relaciona. Esto se realiza mediante el `MysteryClueRepository` que tiene la función `findOneBy()`, a la que hay que pasarle como parámetro un *array* de parámetros de búsqueda. Esta función va a realizar una consulta a la tabla `misterio_pista` de la base de datos para devolver la pista que coincida con los parámetros pasados, en este caso solo va a ser un parámetro que es el id del problema. Una vez es retornada la pista relacionada con el problema, esta se almacena en un *array*.

Por último, es necesaria la creación del formulario en el que el alumno va a poder introducir las palabras claves que haya encontrado. Para ello lo primero es la creación del propio formulario, esto se realiza mediante la función `createForm()` en la que hay que indicar que tipo de formulario, pasando como parámetro la clase del formulario que se quiere, en este caso `KeyForm::class`. Este formulario está compuesto únicamente de un campo de tipo texto, en el que se va a introducir la clave. Con esta función el formulario ya estaría creado, pero es necesario controlar los datos que el alumno va a introducir para comprobar si la clave es correcta o no.

Para controlar el formulario es necesario utilizar la función propia de la clase `Form`, `handleRequest()` que recibe como parámetro el `Request` que se le pasa a la función `mystery` como parámetro. Esta función es la encargada de, una vez introducidos los datos, realizar el POST, este método todavía no ha hecho nada en la base de datos, sino que únicamente ha cambiado el estado del formulario estableciéndolo en `isSubmitted`, es decir, que han sido enviado los datos.

Si el alumno no ha introducido ningún dato la pantalla que se muestra es la que se ha mencionado en la figura 37, esto se realiza mediante el renderizado de la plantilla `"mystery.html.twig"` que necesita como parámetros: el formulario creado (para ello a partir del objeto `form` se llama a la función `createView()`), la lista de problemas que el usuario ha conseguido, y la lista de pistas obtenidas por el alumno.

En cambio, si el alumno sí que ha introducido algún dato en el formulario, hay que realizar una serie de comprobaciones. La primera de ellas es comprobar que el formulario haya sido enviado y que sea válido, a través de las funciones propias del objeto `form` `isValid()` e `isSubmitted`, si no es así estamos en el caso anterior y se renderiza el *template* `"mystery.html.twig"`. Si se cumple esto, es necesario ver si existe algún problema con dicha clave asignada, esto se realiza mediante la función `findOneBy()`, que se ha explicado antes y que posee el `MysteryProblemRepository`, a la que se le pasa como parámetro la clave que ha introducido el alumno, esta clave se obtiene a través del formulario y con la función de éste llamada `getData()`, que devuelve lo que haya sido introducido en éste. Si no se encuentra ningún problema con esta clave significa que la clave introducida no es correcta, y se le muestra al usuario el error que se puede observar en la figura 38.



CLAVE

ERROR clave no encontrada

Introducir clave

Figura 38. Formulario con clave incorrecta introducida

En caso de que la clave introducida sí esté relacionada con algún problema, hay que comprobar que no haya sido introducida previamente por el mismo alumno. Esta comprobación se realiza mediante la función `in_array()` propia de PHP, a la que se le tiene que pasar como parámetro la lista de claves que ha encontrado el usuario y que habíamos obtenido previamente y la propia clave introducida, esta función va a comprobar si la clave existe en la lista que le pasamos como parámetro. En caso de que sí exista, es debido a que el estudiante ya había introducido dicha clave, por lo que se genera el siguiente mensaje de error que se puede observar en la figura 39.



CLAVE

ERROR clave ya encontrada

Introducir clave

Figura 39. Formulario con clave ya introducida

En el caso de que no se encuentre en dicha lista es porque la clave introducida está relacionada con un problema y no ha sido encontrada previamente, por lo que hay que añadirla a lista de claves del usuario y actualizar la base de datos, haciendo uso de los métodos `persist()` y `flush()`, `persist` necesita como parámetro la entidad que ha sido actualizada y que queremos cambiar en la base de datos, mientras que `flush` no necesita ningún parámetro y es la función que va a actualizar la base de datos. Una vez se ha actualizado la base de datos se renderiza el `template` “`correct_key.html.twig`” que necesita como parámetro el

link del enunciado del problema y el nombre del propio problema. En esta nueva pantalla que se renderiza se muestra el nombre del problema y un botón que va a realizar la descarga del enunciado del problema como se puede ver en la figura 40.



Figura 40. Pantalla al introducir una clave correcta

5.3.3 Pantalla de un problema

La siguiente pantalla a la que se puede acceder, muestra al usuario uno de sus problemas en detalle. Para poder llegar a esta pantalla se tiene que seleccionar uno de los problemas de la lista que ha sido mostrada en la pantalla de inicio de la resolución de misterios.

La función que renderiza esta pantalla es la función `show()` del `MysteryController.php`, y recibe como parámetro el `MysteryProblem` que se va a mostrar y la `EntityManagerInterface`, que como se ha comentado en la pantalla anterior, va a permitir interactuar con la base de datos. El template que se va a renderizar es `"mystery_problem_show.html.twig"`.

La ruta utilizada para esta función es `"base-de-datos/misterio/problemas/{id}"` y el nombre con el que se hace referencia durante el desarrollo es `"problem_show"`. Esta ruta tiene gran importancia ya que nos va a evitar tener que realizar la consulta a la base de datos para obtener el problema. Como se ha explicado, a esta pantalla se accede mediante un *click* en uno de los problemas que aparecían en la lista, esto va a redirigir a la página gracias a que en el *template* `"mystery.html.twig"` cada problema es un elemento ancla, es decir, un elemento que crea enlaces a otras páginas, archivos o ubicaciones. La dirección de este enlace es el nombre de la ruta `"problem_show"` y la asignación del parámetro `id` al `id` del problema al que se quiere acceder. Una vez se redirige a la ruta, como en esta se hace uso del *wildcard* `{id}` y a la función se le pasa como parámetro un `MysteryProblem`, Symfony va a realizar la consulta a la base de datos para devolver un `MysteryProblem` con `id` igual al `id` pasado en el *template* anterior, esto lo realiza de manera automática debido a que el *wildcard* utilizado tiene el mismo nombre que uno de los parámetros del objeto `MysteryProblem`, evitándonos así tener que realizar nosotros la consulta. El uso del elemento ancla y de la dirección que se le asigna se puede observar en la figura 41.

```

<div class="">
  <h1 class="mt-5 text-center">
    PROBLEMAS
  </h1>
  {% for problem in problems %}
    <h5>
      <a class="" href="{{ path('problem_show', {id: problem.id}) }}">
        <li class="bases-name">{{ problem.nombre }}</li>
      </a>
    </h5>
  {% endfor %}
</div>

```

Figura 41. uso de elemento ancla en *mystery.html.twig*

Lo primero que se va a realizar en esta función son las comprobaciones que se realizaban en la función `mystery()` de la pantalla de inicio de resolución de misterios. Estas comprobaciones son para aumentar la seguridad de la aplicación, evitando que un usuario que no hubiese iniciado sesión pudiese acceder a esta pantalla y que un alumno no accediese a la competición en unas fechas que no estuviesen en el rango de tiempo permitido.

La siguiente comprobación que hay que realizar, es si el usuario ha desbloqueado el problema al que está intentando acceder. Para ello se comprueba si en la colección de claves que se almacenan en la entidad `Usuario` existe la clave del problema al que se está accediendo, si esta colección no la contiene quiere decir que el usuario no ha desbloqueado dicho problema y es devuelto a la página de inicio de la competición.

Una vez realizadas todas estas comprobaciones se renderiza la plantilla “`mystery_problem_show.html.twig`”, a la que se le pasa el problema como parámetro.

En la figura 42 se puede observar la pantalla que se muestra al usuario una vez renderizado el *template*. En esta pantalla se muestra, el nombre del problema, la breve descripción sobre este, un botón para descargar el enunciado y un botón para acceder a la página que permite resolver dicho problema.

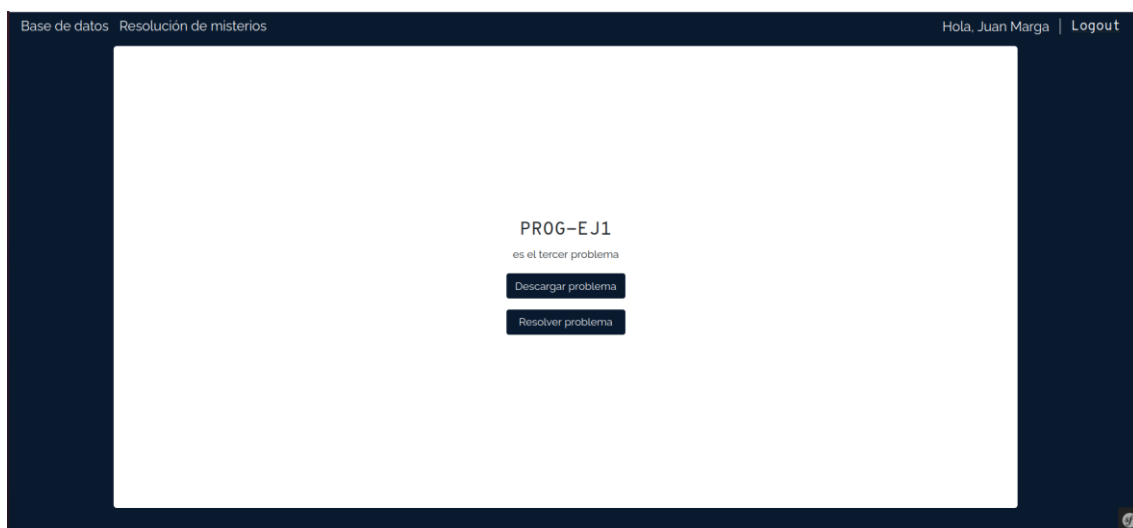


Figura 42. Pantalla de un problema en detalle

5.3.4 Pantalla de resolución de un problema

Una vez se pulsa en el botón de resolver problema, se accede a la pantalla que contiene las preguntas sobre este. Esta pantalla es controlada por la función `resolve()` de `MysteryController.php` que recibe como parámetros:

- `MysteryProblem`: el problema sobre el que se van a hacer las preguntas.
- `MysteryQuestionRepository`: el repositorio que permite realizar consultas SQL sobre una `MysteryQuestion`.
- `Request`: como se ha explicado anteriormente, es un objeto Symfony que permite trabajar con los datos de los formularios.
- `MysteryClueRepository`: el repositorio que permite realizar las consultas SQL sobre una `MysteryClue`.
- `EntityManagerInterface`: como se ha comentado previamente, es un objeto que facilita el uso de la base de datos.
- `MysteryAnswerRepository`: es el repositorio que permite realizar consultas SQL sobre una `MysteryAnswer`.

La URL utilizada para esta función es “base-de-datos/misterio/problemas/{id}/resolve” y el nombre con el que se le hace referencia durante el desarrollo es “`problem_resolve`”. Esta ruta, como en el caso anterior, facilita la obtención del problema, ya que no es necesario realizar la consulta SQL, sino que lo obtiene de la propia URL a través de Symfony y el uso de *wildcast*.

Lo primero que se hace en esta función son las comprobaciones que se han explicado en la función `show()`. Estas comprobaciones servían para asegurar que el usuario había iniciado sesión correctamente en la aplicación, para que no se pudiera acceder a la competición en fechas no permitidas y para que el usuario no pudiese acceder a esta pantalla si no había obtenido el problema previamente.

Tras asegurar que el usuario, que ha accedido a esta pantalla, cumplía con todas las restricciones previas comprobadas, lo siguiente que se realiza es comprobar si el alumno había resuelto el problema anteriormente. Esto se verifica comprobando si el problema está presente en la lista que se almacena en la tabla `user_mystery_problem`. Como esta tabla surge de una relación ManyToMany para acceder a esta lista se hace mediante la función `getProblemMysteryId()` presente en la entidad `User`. Si el problema está presente en esta lista se renderiza el *template* “`mystery_problem_resolved.html.twig`” que necesita como parámetro la pista relacionada con el problema, que se consigue a través del `MysteryClueRepository` y su función `findOneBy()` que recibe el id del problema como parámetro para realizar la consulta SQL, y retorna la pista. Y la pantalla que se muestra al usuario es la de la figura 43, en la que se puede observar la pista para el misterio final que se ha obtenido.

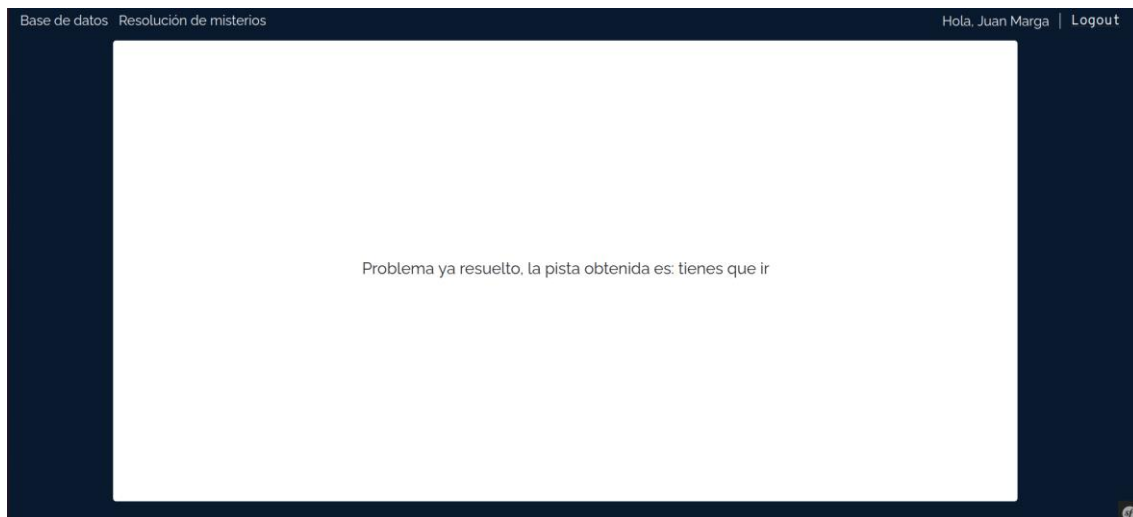


Figura 43. Pantalla mostrada cuando ya se ha resuelto un problema

En cambio, si el alumno no ha respondido previamente al problema, hace falta crear el formulario con las preguntas del problema. Para este formulario no se ha creado una clase independiente, sino que como cada problema va a tener un número y un tipo de preguntas diferentes, el formulario se crea dinámicamente según el problema que se vaya a responder. Para crear este formulario se necesita las preguntas que tiene este problema, estas preguntas se consiguen mediante la colección de ids que se almacena en la entidad del problema. Con cada id de la colección se va a llamar a la función `findOneBy()` de `MysteryQuestionRepository` que recibe un id como parámetro y devuelve la pregunta con ese id, para ser almacenada en un *array*. Las preguntas tienen diferentes tipos de respuesta que tienen un manejo diferente, estos tipos son:

- **Numeric:** este tipo solo acepta respuestas de tipo numérica por ello el tipo con el que se crea en el formulario es `"IntegerType::class"`.
- **Choice:** para este tipo hay que mostrar las respuestas posibles y permitir elegir una de ellas. Para esto se divide la solución que se almacena en la pregunta en las diferentes respuestas, estas respuestas están separadas por el símbolo `'`. El tipo para el formulario es `ChoiceType::class`, y como atributo *choices* se pasa el *array* generado, este atributo es el que va a contener las respuestas para su manejo interno y como *choices_label* se pasan las respuestas quitando la X inicial de la respuesta correcta, este atributo contiene las respuestas que se le muestran al usuario para seleccionar.
- **Letters y Words:** para estos tipos no hay ninguna restricción de respuesta y el tipo del formulario es `"TextType::class"`

Todos estos tipos que se crean en el formulario utilizan como atributo *label* la propia pregunta. En el formulario además de crearse cada pregunta se crea un botón para poder enviar la respuesta, por lo que es de tipo `"SubmitType::class"`.

En la figura 44, se puede ver el formulario creado con una pregunta de cada tipo. Se observa que en la primera pregunta solo se puede escribir números, pudiendo hacerlo de manera manual o aumentando o reduciendo con las flechas de la derecha, los dos siguientes son campos de texto vacío y la última de ellas es de tipo choice y se muestra el desplegable con las 3 respuestas posibles. Esta pantalla se obtiene de renderizar el *template* `"mystery_problem_resolve.html"` que recibe de parámetros el formulario creado

con las preguntas y el parámetro `correct` igualado a nulo, indicando que todavía no ha sido enviado el formulario.

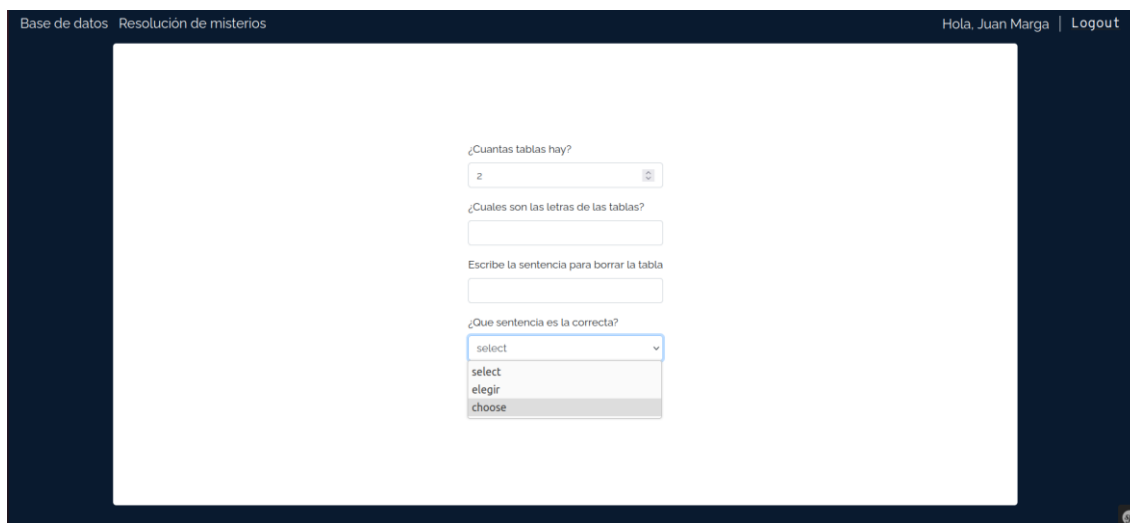


Figura 44. Pantalla con formulario para resolver un problema

Una vez creado el formulario y renderizado al usuario, hay que manejar los datos que éste introduzca. Para ello se utiliza la función `handleRequest()` propia de Form y se comprueba si el formulario ha sido subido por el alumno con las funciones `isValid()` e `isSubmitted()`. En este caso se necesita crear un objeto `MysteryAnswer`, que como ha sido explicado anteriormente se encarga de almacenar la respuesta del alumno. A este objeto se le asigna el id del usuario, el id del problema y se inicializa la puntuación en 0.

La competición cuenta con una penalización en la puntuación de cada problema por cada intento que se realice en este. La puntuación con la que se penaliza es de un 10 por ciento por cada intento, sin que pueda superar el 90 por ciento. Para saber cuántas veces el alumno ha intentado resolver el problema, se utiliza la función `FindOneBySomeField()` del `MysteryAnswerRepository` que recibe como parámetros el id del problema y del usuario, y que retorna directamente el número de veces que el alumno ha enviado una respuesta.

Para comprobar si la respuesta es correcta accedemos a los datos que ha introducido el usuario a través de la función `getData()`, propia del formulario, que devuelve una colección en la que cada elemento es una de las respuestas. Para cada elemento de esta colección hay que comprobar de que tipo es y tratarlo de una forma o de otra, además de almacenarlo en una variable para que luego se pueda insertar en la tabla de `MysteryAnswer`.

Si el tipo de la respuesta es `choice`, la respuesta correcta es la que internamente va precedida con X. Si es de tipo `letters` como no importa ni el orden ni las mayúsculas, se convierte tanto la solución como la respuesta a mayúsculas con la función de PHP `strtoupper()` y se ordena alfabéticamente mediante la función `sort`, quitando los espacios que pudiesen haber sido introducidos haciendo uso de la función `str_replace()`, una vez hecha esta transformación se comprueba si son iguales. Y por último en el caso de `numeric` y de `words` lo único que hay que hacer es comprobar si la respuesta y la solución son idénticas. Cada respuesta correcta es un punto para la puntuación final.

Una vez comprobada cuantas respuestas han sido correctas, se añade la solución y los puntos obtenidos afectados por la penalización al objeto `MysteryAnswer`. La puntuación extra que se puede obtener es de 3 puntos por

cada problema, y se calcula a partir de los días totales de la competición y los días que queden desde el momento en el que se realiza la respuesta. Como el máximo de la puntuación extra que se puede obtener es 3, se divide 3 entre el número de días totales de la competición y este resultado es el extra que se va a conseguir por cada día de competición que quede, es decir, este extra que se ha calculado se va a multiplicar por los días restantes hasta la fecha límite.

Por último, si el número de respuestas correctas coincide con el número de preguntas totales, quiere decir que el alumno ha respondido correctamente al problema, por lo que se va a almacenar en el usuario el id de éste y se va a actualizar tanto la puntuación extra como la puntuación de la competición. En el caso de la MysteryAnswer se va a añadir un 1 en el campo de conseguida. La base de datos se actualiza con estas dos entidades mediante los métodos de `persist()` y `flush()` y se va a renderizar el *template* “mystery_problem_resolve.html.twig” con los siguientes parámetros: el formulario, el número de respuestas correctas, el número de preguntas, la pista conseguida que ha sido obtenida a partir del ClueMysteryRepository, los puntos conseguidos y los puntos extra.

En caso de que no coincidiesen el número de respuestas correctas con el número de preguntas, es porque el problema no ha sido solucionado correctamente y se actualiza la base de datos con la MysteryAnswer con valor 0 en el campo de conseguida y se renderiza el *template* “mystery_problem_resolve.html.twig” con el formulario, el número de respuestas correctas, el número de preguntas y el campo pista como nulo indicando que no ha sido solucionado.

Las pantallas que se muestran al usuario en ambos casos son muy parecidas. En caso de haber resuelto correctamente el problema se muestra las preguntas acertadas, la pista y los puntos obtenidos, como se puede ver en la figura 45. En caso de no haberlo resuelto se muestra únicamente el número de respuestas acertadas.

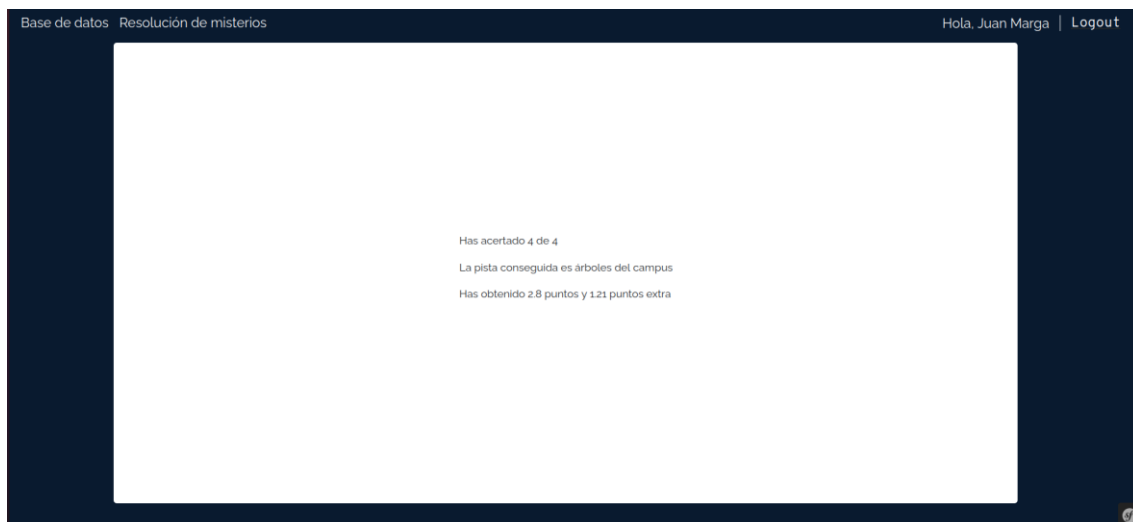


Figura 45. pantalla al resolver un problema

6 Evaluación

Como se ha mencionado a lo largo de este proyecto, la aplicación web desarrollada ha sido lanzada este curso en la asignatura de Bases de Datos de la Escuela Técnica Superior de Ingenieros Informáticos, como parte de la competición de The Great Quiz of Databases. En la figura 46 se puede observar el porcentaje de participación en alguna de las competiciones sobre los alumnos que se han registrado en el sistema que da acceso a las tres competiciones, siendo del 56% los alumnos que han participado en alguna de las competiciones, frente al 44% que no han participado pero que se habían registrado. Como se puede ver, un poco más de la mitad de los alumnos están dispuestos a participar en esta serie de actividades extras, a pesar de no haber podido asignarles una puntuación real en la asignatura.

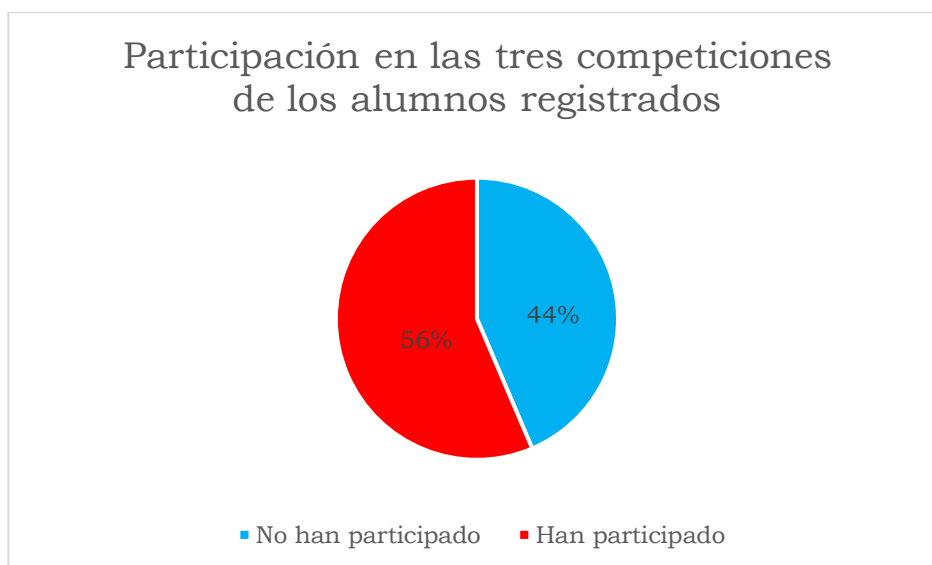


Figura 46. Gráfico con la participación de los alumnos registrados en alguna competición

En la figura 47, se puede observar la participación que ha habido en cada competición de manera separada, a pesar de que The Great Quiz of Databases es una única competición, se han evaluado por separado las partes que tiene. En esta figura se puede ver que las dos partes ya existentes de dicha competición tienen una participación similar mientras que la resolución de misterios tiene una participación menor. Esto es debido a una serie de factores que han influido negativamente en la participación.

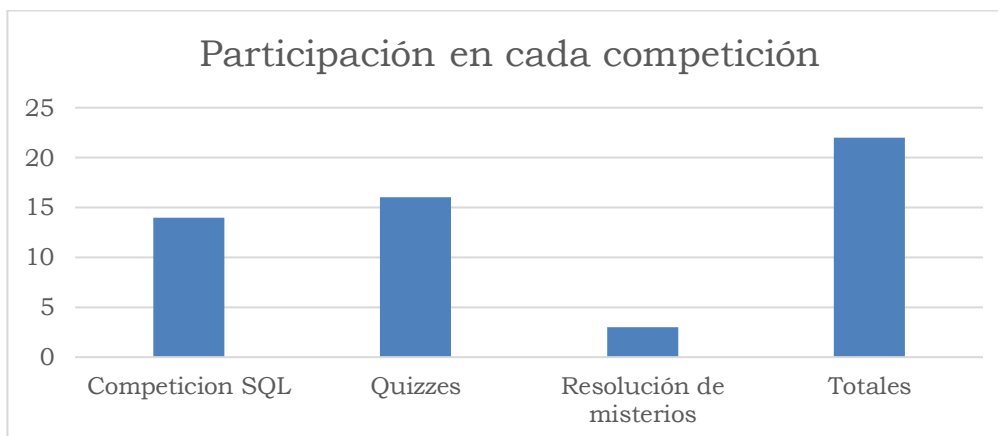


Figura 47. Participación de cada competición

El factor principal ha sido el momento en el que se ha comenzado la competición, siendo esta fecha un poco más tarde de la prevista y coincidiendo con el tramo final del cuatrimestre, donde los alumnos tienen una mayor carga de trabajo y de exámenes, impidiendo dedicar el tiempo necesario a actividades extras.

Otro de los factores importantes es el hecho de que la participación en esta competición se mide una vez el alumno haya resuelto uno de los problemas, por lo que al estar abierto todavía el plazo de participación en la competición, lo más seguro es que algún alumno más consiga resolver uno de estos problemas aumentando así la participación en dicha competición.

A pesar de la baja participación que ha habido en la competición hasta el momento, ésta está sirviendo para definir al ganador de las tres competiciones, ya que hasta el momento previo a esta competición el ganador era uno distinto al actual, por lo que se puede concluir que la participación en esta competición es importante para el desenlace de las competiciones.

7 Resultados y conclusiones

Para concluir este proyecto, cuyo objetivo principal era el desarrollo de una aplicación web para gamificación educativa en la asignatura de Bases de Datos de la Escuela Técnica Superior de Ingenieros Informáticos, se va a revisar si se ha cumplido cada uno de los objetivos específicos que se plantearon y el objetivo principal.

Los dos primeros objetivos específicos tenían muchas similitudes y una forma de realizarse común. El primero de ellos era la puesta en marcha del sistema que se iba a utilizar para el desarrollo del proyecto, este objetivo generó una serie de problemas debido a que muchas de las herramientas que se utilizaban estaban en una versión desactualizada y sin mantenimiento, por lo que en el momento que surgiese algún tipo de error llevaba más tiempo poder arreglarlo. Había que utilizar estas versiones para poder seguir la serie de videos que introducían el uso de Symfony y PHP entre otros, y para poder utilizar en un principio lo que ya había desarrollado para la competición anterior. Este objetivo se pudo completar sin más complicaciones y se pudo seguir con la lista de objetivos en el tiempo planificado.

El segundo de los objetivos, como se ha mencionado tenía una forma de hacerse similar al primero, este era la familiarización con el entorno de trabajo. Dentro de este objetivo entraba tanto la familiarización con las herramientas utilizadas como con el código desarrollado para la otra competición. Para la adaptación a las herramientas utilizadas sirvieron los videos que se siguieron para crear una base sobre estas, mientras que para el código ya desarrollado tuvo que realizarse un aprendizaje autodidacta. Este aprendizaje no tuvo ninguna complicación, debido a la claridad en el código que permitía un buen entendimiento, además el número de clases compartidas con la nueva competición que se iba a desarrollar no era muy amplio, por lo que no generaban muchos problemas. Por lo que podemos concluir que este objetivo fue desarrollado sin muchas complicaciones.

El tercer y el cuarto objetivo era el desarrollo de las dos nuevas funcionalidades que iban a estar disponibles para la página web, estas funcionalidades eran el canjeo de palabras por problemas y el canjeo de problemas por pistas. Estos objetivos eran los principales del proyecto ya que son donde iba a encontrarse la mayoría del desarrollo del código. El desarrollo relacionado con el código PHP no generó demasiadas dificultades debido a su parecido con Java, siendo este el lenguaje al que más familiarizado estaba, y por las facilidades que el *framework* y las demás herramientas daban. El mayor problema que supuso estos dos objetivos era el desarrollo del código HTML al haber sido un lenguaje totalmente nuevo y sin mucho parecido con ninguno de los vistos anteriormente. De todas maneras, estos objetivos fueron desarrollados correctamente y en el plazo estimado.

El quinto de los objetivos específicos era la implementación de los módulos desarrollados en la aplicación ya existente. Como se ha comentado, la familiarización con el código ya desarrollado fue buena por lo que este objetivo no ha generado ninguna complicación respecto a esto. El único cambio que se tuvo que hacer fue la reorganización de las pantallas debido a un cambio del diseño que hubo en una reunión con mi tutor. Por lo que este objetivo también ha sido completado.

Por último, el sexto de los objetivos específicos era la evaluación de los resultados obtenidos sobre el uso de la aplicación en un entorno real. Para ello la aplicación ha tenido que ser lanzada en la asignatura de Bases de Datos y

utilizada por los alumnos de dicha asignatura. La versión lanzada no ha sido la final por tema del calendario escolar, pero como se ha visto en el apartado de evaluación se ha podido realizar un análisis de los datos obtenidos.

Se puede concluir que el desarrollo de este proyecto ha sido satisfactorio y que el objetivo principal ha sido cumplido, habiendo sido desarrollada una aplicación web para gamificación educativa que ha podido ser probada en un entorno real, como es la asignatura de Bases de Datos.

A nivel personal, el desarrollo de este trabajo me ha hecho adquirir conocimientos sobre lenguajes o herramientas que no conocía y que me pueden ser útiles, además de poder participar en un proyecto que va a ser utilizado en un ámbito real que no todo el mundo puede.

Como plan de futuro se espera que todos los profesores de la asignatura hagan uso de esta aplicación o similares, pudiendo así establecer un porcentaje sobre la puntuación final de la asignatura, común para todos los grados que la cursen, además de generar una mayor participación y motivación en los alumnos. También se establece como plan de futuro que otras asignaturas tomen de ejemplo esta aplicación y utilicen metodologías parecidas en sus respectivas clases. Esta aplicación no es cerrada por lo que se pueden ir añadiendo diferentes funcionalidades o competiciones como se ha visto en este caso.

8 Análisis de impacto

Para explicar el análisis de impacto se va a analizar en diferentes contextos:

- Personal: en lo personal este proyecto me ha hecho adquirir conocimientos sobre el desarrollo web, que era un campo que no conocía y que alguna vez me había generado cierta curiosidad y que seguramente utilice en el ámbito profesional si se da la ocasión. Además, me ha permitido realizar un trabajo relacionado con el mundo de la educación, siendo este uno de los campos de trabajo que más interés me generan y que más me gustan.
- Empresarial: como se ha comentado, este trabajo ha sido en la asignatura de base de datos, por lo que tiene un efecto real en el mundo académico, ayudando en este caso a la escuela Politécnica de Madrid a aumentar el número de proyectos innovadores que utilizan. Al ser un proyecto con un uso real, me ha hecho adquirir una experiencia sobre cómo se tiene que trabajar y documentar un proyecto.
- Social: se espera que este trabajo aumente el interés y la participación de los alumnos en la asignatura de Base de Datos, además de generar que algún alumno quiera participar o proponer nuevas competiciones, como fue mi caso tras participar en la competición de The Great Quiz Of Databases. También se espera que el resto de los profesores viendo la acogida que tenga este proyecto, les surja las ganas de crear competiciones como ésta, que hagan uso de la gamificación.
- Económico: esta aplicación va a generar un ahorro para los alumnos que participen en la competición, debido a que supone una ayuda extra para la asignatura sin necesidad de tener que ir a academias u otros métodos de estudios. A la universidad Politécnica de Madrid le puede suponer beneficios en caso de que esta propuesta educativa sea premiada en un futuro.
- Medioambiental: el uso de internet se sabe que afecta negativamente al medioambiente, por lo que al ser una página web va a afectar de manera secundaria a este efecto dañino que se genera. Pero, por otro lado, el uso de esta aplicación va a evitar que se necesite papel para las preguntas o el enunciado de los problemas, por ello también se contrarresta este efecto negativo.
- Cultural: esta aplicación va a ser una de las primeras aplicaciones que hacen uso de la gamificación en la educación en las universidades, por lo que se espere que sea un pilar para la innovación en la educación y que se pueda usar para las diferentes asignaturas y universidades del mundo.

Para finalizar, es importante remarcar el impacto respecto al objetivo número 9 de la lista de objetivos de desarrollo sostenible de naciones unidas [32]. Este objetivo hace referencia a construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación, en esta última es donde este proyecto va a tener impacto. Desde el COVID, se ha visto la importancia y la necesidad de acabar con la brecha digital en sectores como la educación, por ello es necesario que se vayan desarrollando aplicaciones y herramientas que permitan acceder fácilmente a la educación de manera digital. Este proyecto forma parte de estas nuevas aplicaciones que se han ido desarrollando y que facilitan la enseñanza digital.

9 Bibliografía

- [1] Gobierno de Canarias, «Gamificación,» [En línea]. Available: <https://www3.gobiernodecanarias.org/medusa/ecoescuela/pedagogic/gamificacion/>.
- [2] Fundación BBVA, «Indicadores Sintéticos,» 2019. [En línea]. Available: <https://www.fbbva.es/wp-content/uploads/2019/04/Informe-U-Ranking-FBBVA-Ivie-2019.pdf>.
- [3] K. Werbach y D. Hunter, *The Gamification Toolkit: Dynamics, Mechanics, and Components for the Win*, 2015, p. 239.
- [4] R. Bartle, «Hearts, clubs, diamonds, spades: Players who suit MUDs,» *Journal of MUD Research*, p. 29, 1996.
- [5] «Kahoot,» [En línea]. Available: <https://kahoot.com/>.
- [6] Kahoot, «Kahoot!-Juega y crea quizzes,» [En línea]. Available: <https://play.google.com/store/search?q=kahoot&c=apps&gl=ES>.
- [7] A. J. Díaz Honrubia, L. Prieto Santamaría, A. Rodríguez González, E. Menasalvas Ruiz, L. Mengual Galán y C. Fernández Baizán, «The Great Quiz of Databases,» *INTED2023 Proceedings*, pp. 1395-1403, 2023.
- [8] A. J. Díaz Honrubia, A. Rodríguez González, E. Menasalvas Ruiz, L. Megual Galán y C. Fernández Baizán, «A competition to Strengthen the learning of the SQL programming language,» *INTED2021 Proceedings*, pp. 1669-1677, 2021.
- [9] «Symfony,» [En línea]. Available: <https://symfony.com/>.
- [10] F. Zaninotto y F. Potencier, *The Definitive Guide to Symfony*, Apress, 2007.
- [11] F. Potencier y F. Zaninotto, «Symfony en pocas palabras,» [En línea]. Available: <https://uniwebsidad.com/libros/symfony-1-4/capitulo-1/symfony-en-pocas-palabras>.
- [12] «¿Qué es PHP?,» [En línea]. Available: <https://www.php.net/manual/es/intro-what-is.php>.
- [13] PHP, «Historia de PHP,» [En línea]. Available: <https://www.php.net/manual/es/history.php>.
- [14] W3Techs, «Usage statistics of PHP for websites,» [En línea]. Available: <https://w3techs.com/technologies/details/pl-php>.
- [15] W3C, «HTML,» [En línea]. Available: <https://html.spec.whatwg.org/>.
- [16] T. Berners-Lee, «HTML TAGS,» [En línea]. Available: <https://www.w3.org/History/19921103-hypertext/hypertext/WWW/MarkUp/Tags.html>.
- [17] J. Eguiluz, «Breve historia de HTML,» de *Introducción a XHTML*, 2009, pp. 5-7.

- [18] MDN, «JavaScript,» [En línea]. Available: <https://developer.mozilla.org/es/docs/Web/JavaScript>.
- [19] J. Eguíluz, «Breve historia,» de *Introducción a JavaScript*, 2008, pp. 5-6.
- [20] Symfony, «Twig,» [En línea]. Available: <https://twig.symfony.com/>.
- [21] MDN, «CSS,» [En línea]. Available: <https://developer.mozilla.org/es/docs/Web/CSS>.
- [22] J. Eguiluz, «Breve historia de CSS,» de *Introducción a CSS*, pp. 5-6.
- [23] MySQL, «SQL,» [En línea]. Available: <https://dev.mysql.com/doc/refman/8.0/en/sql-statements.html>.
- [24] J. R. Groff, «A Brief History of SQL,» de *SQL: The Complete Reference*, 1999, pp. 22-26.
- [25] A. M. Chaparro, «Historia,» de *Oracle 11g SQL : curso práctico de formación*, 2011, pp. 93-94.
- [26] Doctrine, «Doctrine,» [En línea]. Available: <https://www.doctrine-project.org/>.
- [27] Doctrine, «Doctrine 1.0 Released,» 2008. [En línea]. Available: <https://www.doctrine-project.org/2008/09/01/doctrine-1-0-released.html>.
- [28] Doctrine, «Doctrine 2 First Stable Release,» 2010. [En línea]. Available: <https://www.doctrine-project.org/2010/12/21/doctrine2-released.html>.
- [29] JetBrains, «PHPStorm,» [En línea]. Available: <https://www.jetbrains.com/phpstorm/>.
- [30] MySQL, «MySQL Workbench,» [En línea]. Available: <https://www.mysql.com/products/workbench/>.
- [31] Symfony, «Symfony/form,» [En línea]. Available: <https://symfony.com/doc/current/forms.html>.
- [32] N. Unidas, «objetivos desarrollo sostenible,» [En línea]. Available: <https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/>.

Este documento esta firmado por



Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
Fecha/Hora	Sun May 28 13:39:00 CEST 2023
Emisor del Certificado	EMAILADDRESS=camanager@etsiinf.upm.es, CN=CA ETS Ingenieros Informaticos, O=ETS Ingenieros Informaticos - UPM, C=ES
Numero de Serie	561
Metodo	urn:adobe.com:Adobe.PPKLite:adbe.pkcs7.sha1 (Adobe Signature)