



Universidad Politécnica de Madrid
**Escuela Técnica Superior de
Ingenieros Informáticos**



Grado en Ingeniería Informática

Trabajo Fin de Grado

**Revisión y mejora de un sistema auto-
corrector de Procesadores de Lenguajes
con Gamification, extracción de listados
resumen y mantenimiento**

Autora: Luna Asegurado Sastre

Tutor: José Luis Fuertes Castro

Madrid, Abril 2025

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: Revisión y mejora de un sistema auto-corrector de Procesadores de Lenguajes con Gamification, extracción de listados y mantenimiento.

Mes Año: Abril 2025

Autora: Luna Asegurado Sastre

Tutor:

José Luis Fuertes Castro

Departamento de Lenguajes y Sistemas Informáticos e Ingeniería de Software

ETSI Informáticos

Universidad Politécnica de Madrid

Resumen

Este Trabajo de Fin de Grado enmarca la ampliación y mejora del sistema web DRACO, una plataforma gamificada de refuerzo del aprendizaje utilizada en las asignaturas de Procesadores de Lenguajes y Traductores de Lenguajes impartidas en la Escuela Técnica Superior de Ingenieros Informáticos de la Universidad Politécnica de Madrid. El objetivo principal del trabajo ha sido la mejora y corrección de funcionalidades ya existentes y la implementación de nuevas funcionalidades.

El desarrollo se ha centrado en tres bloques distintos: la mejora de los comprobadores de las prácticas, el diseño de un nuevo módulo de estadísticas de actividades y la implementación de distintas tareas de mantenimiento.

En primer lugar, se ha trabajado en la corrección de errores dentro de los sistemas de comprobación de las prácticas centrándonos en los errores que presentaba el sistema a la hora de seleccionar una plantilla aleatoriamente para el alumno y los errores que mostraba en la comprobación de dichas plantillas, detectando como erróneas plantillas correctas.

En segundo lugar, se ha trabajado en un nuevo módulo de estadísticas que amplía las capacidades analíticas del sistema. Este módulo permite al profesor consultar y visualizar el desempeño de los estudiantes al realizar las actividades mediante tablas estadísticas, ayudando al profesorado a identificar patrones de respuestas, detectar dificultades comunes y tomar decisiones pedagógicas fundamentales.

Finalmente, se han realizado diversas tareas de mantenimiento dentro del sistema desde mejoras visuales hasta funcionales, que han contribuido a refinar el sistema y lo hacen más coherente, usable y robusto de cara a su utilización tanto por parte del alumnado como del profesorado.

Este documento incluye todas las fases del proceso de desarrollo de software: desde el análisis de requisitos hasta el diseño, implementación, pruebas y presentación de resultados. Las mejoras introducidas refuerzan el potencial de DRACO como herramienta educativa, automatizando procesos clave, mejorando la accesibilidad a la información y facilitando la gestión pedagógica en un entorno gamificado orientado al aprendizaje continuo.

Abstract

This Final Degree Project focuses on the expansion and improvement of the DRACO web system, a gamified learning reinforcement platform used in the subjects of Language Processors and Language Translators taught at the Escuela Técnica Superior de Ingenieros Informáticos of the Universidad Politécnica de Madrid. The main objective of the work has been to enhance and correct existing functionalities and to implement new features.

The development has been structured around three distinct blocks: the improvement of the practice checkers, the design of a new activity statistics module, and the implementation of various maintenance tasks.

First, work has been carried out to fix errors within the practice checking system, focusing on issues related to the random selection of templates for students and the incorrect validation of those templates, where correct templates were being erroneously marked as incorrect.

Second, a new statistics module has been developed to expand the system's analytical capabilities. This module allows instructors to consult and visualize student performance during activities through statistical tables, helping teachers to identify response patterns, detect common difficulties, and make informed pedagogical decisions.

Finally, various maintenance tasks have been carried out in the system, ranging from visual improvements to functional enhancements, which have contributed to refining the platform and making it more consistent, usable, and robust for both students and teachers.

This document covers all phases of the software development process: from requirements analysis to design, implementation, testing, and presentation of results. The improvements introduced strengthen DRACO's potential as an educational tool by automating key processes, improving information accessibility, and facilitating pedagogical management in a gamified environment focused on continuous learning.

Tabla de contenidos

1	Introducción.....	1
2	Estado de la cuestión.....	4
2.1	HTML.....	4
2.2	CSS.....	6
2.3	PHP.....	8
2.4	SQL.....	11
2.4.1	Comandos para definir datos	11
2.4.2	Comandos para la manipulación de datos	12
2.4.3	Comandos para la consulta de datos	12
2.4.4	Comandos para el control de acceso.	12
2.5	Compiladores	12
2.5.1	Análisis léxico.....	13
2.5.2	Análisis sintáctico.....	14
2.5.3	Análisis semántico.....	14
2.5.4	Generación de código intermedio	15
2.5.5	Optimización de código	15
2.5.6	Generación de código.....	15
2.5.7	Tabla de símbolos.....	16
2.5.8	Gestor de errores	16
2.6	Gamificación	16
3	Planteamiento del problema	18
3.1	Introducción.....	18
3.1.1	Propósito	18
3.1.2	Ámbito del sistema	18
3.1.3	Definiciones, Acrónimos y Abreviaturas	18
3.1.3.1	Definiciones	19
3.1.3.2	Acrónimos y abreviaturas	19
3.2	Descripción general.....	20
3.2.1	Perspectiva del producto.....	20
3.2.2	Funciones del sistema.	20
3.3	Requisitos	21
3.3.1	Requisitos funcionales.....	21
3.3.1.1	Prácticas.....	21
3.3.1.2	Estadísticas de actividades.	22
3.3.1.3	Mantenimiento.....	23
3.3.2	Requisitos no funcionales.	24
4	Solución	26
4.1	Diseño.....	26

4.1.1	Diseño de la base de datos.....	26
4.1.1.1	Base de datos del módulo de actividades.	26
4.1.1.2	Base de datos del módulo de prácticas.....	29
4.1.2	Diseño de los requisitos funcionales.	34
4.1.2.1	Prácticas.....	34
4.1.2.2	Estadísticas de actividades	36
4.1.2.3	Mantenimiento.....	38
4.2	Implementación	42
4.2.1	Prácticas.....	42
4.2.2	Estadísticas de actividades.	43
4.2.3	Mantenimiento	47
4.3	Pruebas.....	49
4.3.1	Entorno de pruebas	49
4.3.2	Tipos de pruebas	50
5	Conclusiones	54
6	Futuras líneas de trabajo	57
7	Análisis de Impacto	59
8	Bibliografía	62

1 Introducción

Este Trabajo de Fin de Grado tiene como objetivo ampliar y mejorar las funcionalidades de DRACO (Dinámica de Refuerzo del Aprendizaje de Compiladores). DRACO es la página web que sirve de apoyo para las asignaturas de PdL (Procesadores de Lenguajes) y TdL (Traductores de Lenguajes) impartidas en la Escuela Técnica Superior de Ingenieros Informáticos de la Universidad Politécnica de Madrid. Este sitio web busca fomentar la participación y la motivación de los estudiantes utilizando técnicas de gamificación para ello, es decir contiene elementos propios de los juegos en un entorno no lúdico.



Ilustración 1: Interfaz de DRACO del profesor

El sistema consta de dos interfaces, una para los profesores (Ilustración 1) y otra para los alumnos. En la primera, los profesores pueden encontrar los distintos módulos de configuración del sistema. En la configuración de actividades, los profesores pueden crear nuevas actividades y gestionar las actividades existentes pudiendo hacerlas visibles para los alumnos. Hay varios tipos de actividades diferentes: cuestionarios de respuesta única o múltiple, preguntas de respuesta corta, preguntas con respuesta larga, actividades en las que se deben ordenar las respuestas, actividades en las que se deben formar parejas entre las respuestas ofrecidas... También se ofrece al profesor la posibilidad de ver las estadísticas individuales de cada alumno por cada actividad.

En la configuración de las prácticas los profesores encuentran todas las herramientas relacionadas con la práctica de la asignatura que consiste en la realización de un procesador por parte del alumno. Los profesores pueden definir el lenguaje fuente que aceptará el compilador y gestionar los diferentes módulos de la práctica: análisis léxico, tabla de símbolos, análisis sintáctico y análisis semántico. Dentro de cada módulo los profesores pueden gestionar el sistema de plantillas y ficheros o fragmentos fuente que generarán la prueba al alumno. Los fragmentos fuente son ficheros de texto que contienen fragmentos de un programa, en los que se pueden encontrar zonas aleatorias en las que se encuentran diferentes versiones para una parte del código. Estas zonas se delimitan mediante símbolos de euro y cada opción está separada por símbolos de apertura de interrogación, en la Ilustración 2 hay un ejemplo de estas zonas (en la que las zonas aleatorias permiten asignar distintos operadores). Al

generarse el fichero de prueba para el alumno el sistema selecciona una de las opciones que lo contiene y solamente es mostrada esta.

```
a = b € + ? - ? * ? / € c;  
print (a);  
a € = ? += ? -= ? *= ? /= € d;  
print (a);
```

Ilustración 2: Ejemplo de zona aleatoria en un fragmento

Las plantillas están formadas por un conjunto de fragmentos fuente e indicaciones sobre cómo se deben combinar, como la posición que ocupa cada uno, para formar los programas finales que se les entregarán a los alumnos como prueba. Además de por la posición, los ficheros son cribados por las opciones que se presentan en ellos, que los alumnos pueden tener o no ya que cada grupo de prácticas tiene asignado un subconjunto distinto del lenguaje. Por tanto, hay ciertos ficheros u opciones de las zonas aleatorias que no se pueden mostrar según el grupo que está realizando la prueba.

Estas plantillas, antes de poder ser seleccionadas por el sistema para ser mostradas deben pasar un sistema de comprobación, el cual realiza todas las combinaciones posibles de los ficheros que contiene la plantilla y analiza estas combinaciones y su corrección dentro del lenguaje definido para ser aceptado por el procesador a realizar por los alumnos.

Finalmente, el profesor puede ver diferentes estadísticas de los grupos que realizan las pruebas en cada módulo y realizar consultas sobre las pruebas que se han realizado, pudiendo ver el fichero que se ha generado al alumno y la respuesta introducida por este.

El profesor tiene también disponible las estadísticas generales, la configuración de las insignias que se entregarán a los alumnos, una configuración del sistema...



Ilustración 3: Interfaz del alumno

En la interfaz disponible para el alumno (Ilustración 3), este puede encontrar diferentes actividades que realizar diariamente para obtener puntos, un sistema de *ranking* global respecto a sus compañeros, los puntos o monedas obtenidas, o un sistema de insignias. Todos ellos son elementos clásicos de un sistema de gamificación.

Además, cuenta con el módulo de prácticas donde los alumnos pueden probar su trabajo de realización de un procesador definido por los profesores de la asignatura. Este módulo permite probar la práctica en sus cuatro etapas de desarrollo. Previamente, deben registrar las decisiones de diseño que han tomado durante el desarrollo, realizando las tareas de definición de los *tokens*, la gramática sintáctica y la definición de tipos.

Los alumnos tienen disponible una página en la que encuentran el *ranking* global de la asignatura según los puntos obtenidos por cada alumno. Además, encuentran también otra página con todas las insignias que les han sido otorgadas por sus avances y las que todavía no ha obtenido. Estas insignias las obtienen realizando actividades, pruebas de práctica y distintas exploraciones del sistema.

Finalmente, DRACO consta de pequeñas secciones como un tablón de noticias en las que los profesores pueden subir anuncios con el objetivo de comunicarse con los alumnos, una sección en la que el alumno puede configurar su perfil y una tienda con un sistema monetario en la que pueden comprar puntos por monedas, o utilizar estas para comprar más pruebas, la reducción de tiempo hasta la siguiente prueba o la repetición de una actividad.

En este Trabajo de Fin de Grado se van a desarrollar distintas mejoras y nuevas implementaciones trabajando sobre varios de los módulos descritos.

Primero se va a trabajar en la reducción del tiempo de espera entre las entregas de corrección de la práctica, generando un sistema en el que, dependiendo de la puntuación obtenida por el alumno al realizar su prueba de la práctica, se reduzca el tiempo hasta la siguiente prueba. Además, de cara a los profesores, en la configuración de cada módulo se incorporará una funcionalidad para reducir el tiempo de espera hasta la siguiente prueba de un grupo concreto.

En el módulo de actividades se va a trabajar en la interfaz del profesor, la cual actualmente solo muestra las estadísticas colectivas en cada actividad. Se va a introducir una tabla resumen, diferente para cada tipo de actividad, la cual mostrará las estadísticas generales con la información detallada de todos los alumnos que han realizado dicha actividad para que el profesor pueda obtener la información de cada actividad de manera sencilla.

Finalmente, se va a trabajar en la mejora de los módulos de comprobación de las prácticas. Llevando a cabo un análisis exhaustivo de estos en búsqueda de posibles fallos centrándose en la selección de ficheros y plantillas por parte del sistema. Por un lado, se analizará cómo se selecciona la plantilla según las opciones del alumno y si esta selección se realiza de forma aleatoria. Por otro lado, se analizará el sistema que comprueba las plantillas antes de ponerlas a disposición del sistema para ser seleccionable de cara a los alumnos por parte del profesor, debido a que se han encontrado plantillas correctas en las que a la hora de realizar la comprobación el sistema no las valida correctamente.

2 Estado de la cuestión

2.1 HTML

HTML es un lenguaje de programación que da estructura y significado a las páginas web.

La historia de HTML, el Lenguaje de Marcado de Hipertexto o *HyperText Markup Language* en inglés, se remonta a 1990 cuando el físico investigador Tim Berner-Lee del CERN (Organización Europea para la Investigación Nuclear) propuso un nuevo sistema de “hipertexto” para compartir documentos [1].

HTML no fue el primer sistema de hipertexto que hubo; antes de la aparición de este existían otros sistemas de vinculación en documentos electrónicos como los hipervínculos y el SGML [2]. Sin embargo, estos no habían logrado una implementación completa debido a numerosos problemas entre los que destaca la falta de estandarización.

En 1991 se publica el primer documento HTML llamado HTML Tags, cuyo contenido contaba con la descripción de este lenguaje. Durante la década de los 90 se sucedieron numerosas propuestas para convertir HTML en un estándar oficial, mientras se iban definiendo distintas etiquetas del lenguaje. En 1995 fue publicado el primer estándar oficial, HTML 2.0, por el organismo IETF (*Internet Engineering Task Force*).

A partir de 1996 es el W3C (*World Wide Web Consortium*) quien publica los estándares de HTML. En 1997 se publican dos versiones nuevas, HTML 3.2 que fue la primera recomendación oficial publicada por dicha organización y que destaca por la incorporación de textos “enmarcados” y que se disponen alrededor de las imágenes; y HTML 4.0 en la que se produce un gran avance respecto a versiones anteriores, como la incorporación de las hojas de estilo CSS o la posibilidad de incorporar *scripts* a las páginas web [3].

En 2004, ante las preocupaciones de distintas empresas por el desinterés del W3C en el HTML, se funda una nueva asociación llamada WHATWG (*Web Hypertext Application Technology Working Group*), la cual se centró en el desarrollo del HTML 5.0, publicado en 2008. Este cumplió el objetivo de crear un lenguaje más flexible y compatible con los nuevos avances multimedia incluyendo funcionalidades como audio, video, Canvas y API para aplicaciones web [4].

Finalmente, en octubre de 2014 el consorcio W3C junto con Tim Berner-Lee publicaron la versión definitiva y actual de HTML5 [5].

HTML es un lenguaje de etiquetas también llamado lenguaje de marcas. Habitualmente cada página web está formada por pares de etiquetas, las cuales siguen normalmente la estructura mostrada a continuación:

`<Nombre de etiqueta> Contenido </Nombre de etiqueta>`

La principal ventaja de este tipo de lenguaje es su sencillez tanto en la escritura como en la lectura por parte tanto de las personas como de los sistemas electrónicos. Sin embargo, su principal desventaja es el gran tamaño del

documento que se genera [6]. En la Tabla 1 se muestran algunas de las etiquetas más utilizadas junto con su tipo y una breve descripción [7].

Etiqueta	Descripción
<!DOCTYPE>	Utilizada para especificar la versión HTML utilizada en la página.
<html>	Define el inicio y el final de la página web
<head>	Define la sección de encabezado de página
<body>	Define la sección del cuerpo de la página.
<h1> a <h6>	Define los títulos de diferentes niveles de jerarquía en la web.
<a>	Define un enlace en el que el usuario puede hacer clic para ir a otra página web o sección.
	Define una imagen que se mostrará en la página web.
<table>	Define una tabla en la página web.
<p>	Define un párrafo de texto.
 	Etiqueta de salto de línea sencillo.
<section>	Define una sección.
<label>	Utilizada para añadir nombre o título a otros elementos.
	Para crear listas ordenadas.
	Para crear listas sin orden.

Tabla 1: Etiquetas HTML

Las páginas HTML se dividen en dos partes: la cabecera, identificada con la etiqueta <head>, y el cuerpo, identificado con la etiqueta <body>, mostrados en la Ilustración 4. La cabecera incluye información sobre la propia página, tales como el título o el idioma, la cual no se visualiza, aunque son de importancia para los navegadores y motores de búsqueda, a excepción del título que se muestra como nombre de la ventana en muchos navegadores. Por otro lado, el cuerpo incluye todo lo que el usuario visualiza en su pantalla, como los párrafos de texto y las imágenes, y con el cual pueden también interactuar.

```
<html lang="es">

<head>
  <title>Mi primer documento HTML</title>
</head>

<body>
  <p>El lenguaje HTML es <strong>tan sencillo</strong> que
    prácticamente se entiende sin estudiar el significado
    de sus etiquetas principales.</p>
</body>

</html>
```

Ilustración 4: Ejemplo de la estructura de un documento HTML

2.2 CSS

CSS o *Cascading Style Sheets*, cuyo significado es “hojas de estilo en cascada” es un lenguaje que se utiliza para dar estilo a elementos escritos en un lenguaje de marcado como HTML.

Las hojas de estilo aparecen hacia 1970 tras el lenguaje de etiquetas SGML, cuando se observó la necesidad de definir un mecanismo que permitiera aplicar de forma consistente diferentes estilos a los elementos electrónicos.

Con el *boom* de internet y el crecimiento de HTML, los lenguajes de estilo tuvieron un gran impulso. Sin embargo, debido a la falta de un estándar para la definición de estilo, se encontraron problemas con la creación de documentos con la misma apariencia en distintos navegadores.

Por esto el organismo W3C, encargado de crear las especificaciones relacionadas con la web, propuso la creación de un lenguaje de hojas de estilo específico para HTML y nueve propuestas fueron presentadas. Las dos propuestas que tomaron más peso fueron la CHSS (*Cascading HTML Style Sheets*) de Håkon Wium Lie y SSP (*Stream-based Style Sheet Proposal*) propuesto por Bert Bos. Finalmente, a comienzos de 1995 ambos creadores se unieron para combinar lo mejor de cada propuesta y definir así un nuevo lenguaje que denominaron CSS. A partir de este momento, en 1996, CSS se convirtió en la especificación de estilos para brindar diseño a los sitios web [8].

Esta primera versión de CSS, conocida como CSS1, presentaba características básicas como la capacidad de aplicar estilos a elementos HTML, definir colores y fondos o establecer márgenes y espaciados. También introdujo la noción de selectores, permitiendo a los diseñadores seleccionar y aplicar estilos a elementos específicos de una página.

En 1998, es lanzada la segunda versión de este lenguaje, CSS2, la cual incluye una amplia gama de nuevas características entre las que destacan la posibilidad de establecer estilos para tablas, formularios y otros elementos de las páginas web. Fue también incluida la capacidad de posicionar elementos de manera más precisa mediante el uso de propiedades como “position” o “z-index”. CSS2 introdujo también nuevos tipos de selectores como los selectores de atributo, pseudo-clases y pseudo-elementos.

Un año más tarde, en 1999, se produjo el lanzamiento de CSS3, versión en la cual se introdujeron muchas características nuevas y útiles como la posibilidad de aplicar sombras, bordes redondeados, gradientes o transparencias. Se añadieron también nuevas propiedades de texto como “text-shadow” o “text-overflow”. Esta versión también introdujo módulos nuevos para la animación y las transiciones permitiendo a los diseñadores crear efectos más dinámicos y atractivos [9].

Finalmente, a esta versión de CSS3 se la han añadido dos características muy importantes, *CSS Flexbox* y *CSS Grid*. *CSS Flexbox*, fue introducida en 2009 como parte de la especificación de CSS3; este módulo permite crear diseños flexibles y adaptativos, alineando y distribuyendo elementos en un contenedor de manera flexible. *CSS Grid* ha sido uno de los desarrollos más importantes en los últimos años, permitiendo crear diseños complejos y flexibles utilizando una cuadrícula de filas y columnas.

Actualmente, están siendo desarrollados los módulos de la siguiente versión, CSS4. Se espera que incluyan algunas características como la capacidad de aplicar estilos a elementos de la página web basados en la hora del día o la ubicación, así como la posibilidad de aplicar estilos en función de la dirección en la que se lee el texto.

Existen tres formas diferentes de introducir CSS en un documento HTML. La primera es CSS en línea, en la cual las instrucciones de diseño se integran directamente en el código fuente mediante un atributo de estilo como se muestra en la Ilustración 5. Estas propiedades se asignan solamente a un elemento, por lo que es posible realizar diseños diferentes por todo el documento. Sin embargo, con este tipo de integración se pierden muchas ventajas de CSS como la posibilidad de aplicar un estilo a todos los elementos h1, por ejemplo, con un solo comando.

```
<h1 style="color:blue; font-size:14px;">Esto es un encabezado</h1>
```

Ilustración 5: Estilo en línea

En segundo lugar, se puede introducir CSS añadiendo la etiqueta <style> en la cabecera del HTML tal y como se muestra en la Ilustración 6. De esta manera la etiqueta de estilo se convierte en parte del elemento de cabecera aplicándose a todo el archivo. Por tanto, las instrucciones están contenidas en el documento, pero separadas del código HTML.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      h1 {color:blue; font-size:2em;}
    </style>
  </head>
  <body>
    <h1> Esto es un encabezado </h1>
    <p>Esto es un párrafo<p>
  </body>
</html>
```

Ilustración 6: CSS en la cabecera

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="stylesheet.css">
  </head>
  <body>
    <h1> Esto es un encabezado</h1>
    <p> Esto es un párrafo <p>
  </body>
</html>
```

Ilustración 7: HTML con CSS en documento externo

La tercera y última opción para integrar el CSS es mediante el uso de un archivo externo, tal y como se muestra en la Ilustración 7 y la Ilustración 8, en las que en la primera se muestra el documento HTML al que se le aplica el estilo y el

cual hace referencia mediante una etiqueta de enlace al archivo mostrado en la segunda de las ilustraciones, el cual contiene el estilo en CSS. Esta es la forma más utilizada ya que permite una separación limpia entre el contenido y el diseño facilitando así el mantenimiento [10].

```
h1 {  
    color: blue;  
    font-size:14px;  
}
```

Ilustración 8: Documento CSS externo

El código CSS de un documento se basa en los siguientes conceptos:

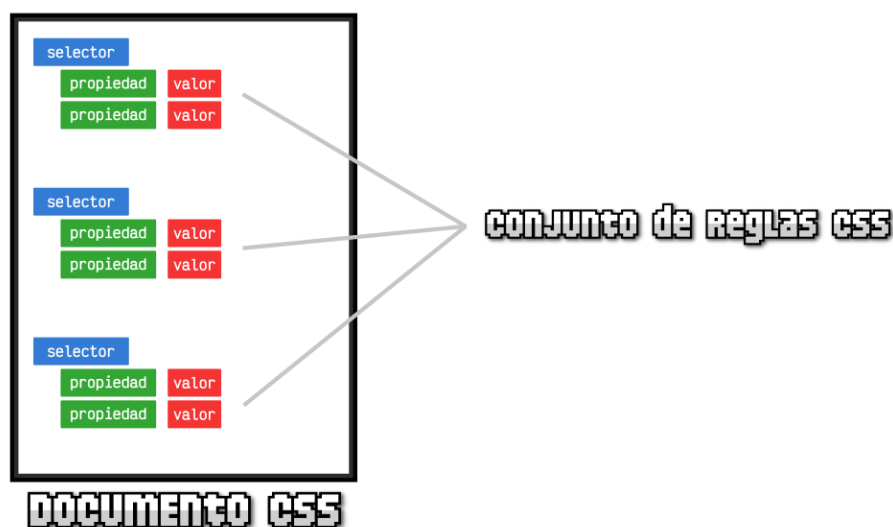


Ilustración 9: Esquema documento CSS

- El selector, que define el elemento o elementos del documento que se va a seleccionar para aplicar un estilo concreto.
- La regla, es un par formado por la propiedad, que es la característica principal que vamos a definir y a continuación el valor, que define el valor a tomar por la propiedad, con el que tendrá uno u otro comportamiento.

Tal y como se muestra en la Ilustración 9, un documento CSS es una combinación de varias reglas con distintos selectores, propiedades y valores [11].

2.3 PHP

PHP (acrónimo recursivo de *PHP: Hypertext Preprocessor*) es un lenguaje de programación de código abierto muy popular especialmente adecuado para el desarrollo web y que puede ser incrustado dentro del código HTML.

PHP, tal y como se conoce hoy en día, es el sucesor de un producto conocido como PHP/FI. Su desarrollo comienza en 1994 cuando Rasmus Lerdoof creó un conjunto de *scripts* en el lenguaje de programación C para rastrear visitas en su currículum *online*. Este tomó el nombre de *Personal Home Page Tools*, comúnmente conocido como *PHP Tools* y durante los siguientes meses continuó con su desarrollo añadiendo más funcionalidades como la capacidad de

interactuar con bases de datos, lo que permitió a los usuarios desarrollar aplicaciones web dinámicas sencillas [12].

En junio de 1995, Rasmus publicó el código fuente de *PHP Tools*, alentando a otros desarrolladores a contribuir con mejoras. En septiembre de ese mismo año, amplió las funcionalidades del lenguaje publicando la nueva versión bajo el nombre de FI (*Forms Interpreter*). Esta versión incluía variables al estilo *Pearl*, procesamiento automático de formularios y una sintaxis incrustada en HTML en forma de comentarios, lo que no fue bien recibido. En octubre de 1995, recuperó el nombre de PHP siendo renombrado a *Personal Home Page Construction Kit*, con una estructura de código similar a C buscando facilitar su adopción por parte de programadores con experiencia en *Pearl* y C [13].

En abril de 1996, Rasmus presentó PHP/FI (*Personal Home Page / Forms Interpreter*), una versión más avanzada que consolidó PHP como un lenguaje de programación. Esta versión soportaba bases de datos como DBM, MySQL y Postgres95, además de permitir *cookies* y funciones definidas por el usuario. En junio de 1996, se lanzó PHP/FI 2.0, y aunque solo hubo una versión estable, en noviembre de 1997, PHP/FI ya contaba con una creciente comunidad de usuarios.

A pesar del éxito de PHP/FI 2.0, Andi Gutmans y Zeev Suraski, programadores de Israel, encontraron que era insuficiente para desarrollar una aplicación de comercio electrónico en la que estaban trabajando. En 1997, rediseñaron el motor de PHP y se unieron a Rasmus para crear una versión completamente nueva, más potente y estructurada. Para evitar la percepción de que PHP era solo para uso personal, renombraron el lenguaje como *PHP: Hypertext Preprocessor*, estableciendo la base del PHP moderno [14].

PHP 3 fue lanzado en junio de 1998. Esta versión destacó por su extensibilidad, permitiendo integrar nuevas bases de datos, protocolos y API con facilidad. Su comunidad creció rápidamente, atrayendo a desarrolladores que contribuyeron con módulos y extensiones. También introdujo soporte para programación orientada a objetos (OOP) y una sintaxis más consistente.

Tras el éxito de PHP 3, en 1998, Gutmans y Suraski iniciaron el desarrollo de un nuevo núcleo para PHP, buscando mejorar su rendimiento y modularidad. Así nació el Motor *Zend*, que fue la base de PHP 4, lanzado en mayo de 2000. Esta versión introdujo mejoras clave como sesiones HTTP, *buffers* de salida, mayor seguridad en la entrada de datos y compatibilidad con múltiples servidores web. Gracias a su optimización, PHP 4 permitió el desarrollo de aplicaciones más complejas y dinámicas, consolidándose como una de las principales tecnologías para la web [13].

En julio de 2004, después de un largo desarrollo, se lanzó PHP 5, impulsado por *Zend Engine 2.0*. Esta versión trajo un nuevo modelo de objetos con soporte mejorado para herencia, visibilidad, excepciones y destructores, acercando PHP a lenguajes como Java y C++. También se incluyeron mejoras en la gestión de bases de datos con PDO (*PHP Data Objects*) y nuevas extensiones como SimpleXML y SOAP.

Con el paso del tiempo, PHP ha seguido evolucionando con versiones como PHP 7 y PHP 8, mejorando su rendimiento y seguridad. Hoy en día, PHP sigue siendo uno de los lenguajes más utilizados en el desarrollo web, con una comunidad

activa y una infraestructura de servidores que lo respaldan a nivel global. Se estima que está instalado en millones de dominios en todo el mundo, siendo la base de plataformas populares como WordPress, Drupal y Joomla.

PHP es un lenguaje de *scripting* del lado del servidor: el programa es ejecutado en el servidor web y posteriormente el contenido es enviado al navegador web utilizado por el usuario tal y como se ilustra en la Ilustración 10. Además, es de código abierto, por lo que cuenta con una gran comunidad de desarrolladores lo que permite que tenga numerosas bibliotecas que facilitan el trabajo con este lenguaje. Debido a todo esto, PHP es muy utilizado en el desarrollo web. Según datos de *Web Technology Surveys*, este lenguaje es utilizado por un 77,6% de todos los sitios web, incluyendo algunos muy populares como Facebook o Wikipedia [15].



Ilustración 10: Diagrama del funcionamiento de PHP

El código PHP puede integrarse con contenido HTML; para esto se debe encerrar el código PHP con la etiqueta de inicio `<?php` y la etiqueta de cierre `?>` tal y como se muestra en el código de la Ilustración 11.

```
<!DOCTYPE html>
<html lang="en">
  <head>
    <title>Embedded PHP in a .html File</title>
  </head>
  <body>
    <h1><?php echo "Hello world"; ?></h1>
  </body>
</html>
```

Ilustración 11: Ejemplo de código PHP incrustado en HTML

PHP también permite la comunicación con casi cualquier tipo de base de datos. Por ejemplo, para trabajar con una base de datos SQL ofrece las funciones `mysqli_connect` para la conexión con esta, `mysqli_query` para la ejecución de consultas y `mysqli_fetch` para la obtención de los datos. También tiene funciones para evitar la inyección SQL.

2.4 SQL

SQL (*Structured Query Language*) es un lenguaje de programación que permite manipular los sistemas de bases de datos relacionales. El lenguaje se utiliza principalmente para comunicarse con las bases de datos con el objetivo de gestionar los datos que contienen. En particular, permite almacenar, manipular y encontrar estos datos mediante consultas.

La historia de SQL comienza en 1969, con la definición por parte del investigador de IBM, Edgar F. Codd, de un modelo de bases de datos relacionales, basado en la asociación de claves con varios datos. Unos años más tarde, IBM creó un lenguaje para los sistemas de gestión de bases de datos relacionales basado en el trabajo de Codd. Este lenguaje primero se llamó SEQUEL (*Structured English Query Language*) y tras varias implementaciones y revisiones pasó a llamarse SQL [16].

A partir de 1986 se popularizó su uso gracias a que el ANSI (*American National Standards Institute*) lo adoptó como estándar. En 1990 se crea MySQL, el primer sistema de gestión de bases de datos de código abierto.

Las bases de datos SQL organizan los datos en un formato tabular utilizando sus filas y columnas para estructurar la información. Estas tablas, están vinculadas a través de relaciones, a menudo mediante el uso de un identificador específico o clave primaria para facilitar la recuperación y gestión eficaz de los datos [17].

Sublenguaje SQL	Comandos	Ejemplos
Data Definition Language (DDL)	Comandos para definir el esquema de la base de datos: crear, modificar y eliminar tablas de la base de datos; definir claves primarias, claves externas y limitaciones (<i>constraints</i>).	CREATE TABLE, DROP TABLE
Data Manipulation Language (DML)	Comandos de manipulación de datos: modificar, insertar y eliminar registros.	INSERT, UPDATE
Data Query Language (DQL)	Comandos para consultar y acondicionar los datos	SELECT
Data Control Language (DCL)	Comandos para la gestión de derechos.	GRANT, REVOKE

Tabla 2: Sublenguajes de los comandos SQL

Los comandos SQL ordenan a un sistema de gestión de bases de datos que realice determinadas acciones como definir tablas y sus estructuras, introducir, modificar y eliminar datos o ejecutar consultas. Los comandos más importantes se pueden dividir, en cuatro sublenguajes mostrados en la Tabla 2.

2.4.1 Comandos para definir datos

Estos comandos se utilizan para definir estructuras de datos. Todos ellos operan sobre objetos agregados como las bases de datos, las tablas o los índices.

Con estos se pueden crear nuevos objetos agregados o eliminar los ya existentes. Además, estos comandos permiten establecer las claves foráneas y las primarias [18]. Algunos ejemplos de estos comandos son:

- **CREATE DATABASE:** Crea una nueva base de datos.
- **DROP DATABASE:** Elimina una base de datos existente.
- **USE:** Selecciona la base de datos para el siguiente comando
- **CREATE TABLE:** Crea una nueva tabla dentro de una base de datos, además del nombre de la base de datos se definen los nombres de las columnas y sus tipos.
- **DROP TABLE:** Elimina una tabla existente por completo.
- **ALTER TABLE:** Modifica una tabla existente. Añade o elimina columnas, cambia el tipo de una columna o su nombre.
- **ALTER COLUMN:** Cambia el tipo de datos de una columna existente.

2.4.2 Comandos para la manipulación de datos

Con las tablas de datos ya definidas, estos comandos realizan operaciones sobre los registros de datos individuales. Se pueden insertar, modificar y eliminar datos. Estos comandos son:

- **INSERT INTO:** Inserta un registro en una tabla.
- **UPDATE:** Modifica los campos de uno o varios registros.
- **DELETE FROM:** Elimina los registros de una tabla.

2.4.3 Comandos para la consulta de datos

Permite acceder a los datos de forma controlada y recuperarlos mediante filtros [18]. Estos comandos son:

- **SELECT:** Selecciona la columna de la que recuperar datos.
- **WHERE:** Limita la consulta a los registros que cumplen un determinado criterio.
- **LIMIT:** Restringe el conjunto de resultados a los primeros resultados.
- **DISTINCT:** Elimina los duplicados.
- **GROUP BY:** Agrupa el conjunto de resultados según una característica común.
- **ORDER BY:** Ordena el conjunto de resultados según una característica.
- **JOIN:** Filtra dos conjuntos de resultados según un criterio común.

2.4.4 Comandos para el control de acceso.

Se utilizan para gestionar las bases de datos de forma independiente, pudiendo definir para cada usuario los derechos con los que puede acceder a las bases de datos y a las tablas dentro de estas. Estos comandos son:

- **GRANT:** Asigna derechos.
- **REVOKE:** Quita derechos.

2.5 Compiladores

Un compilador es un software que traduce un programa escrito en un lenguaje fuente a un lenguaje objeto. Generalmente, el lenguaje fuente es de alto nivel y el objeto de bajo nivel, ensamblador o código máquina. Además, durante el proceso de compilación se detecta cualquier error en el programa fuente.

El término compilador fue acuñado a principios de 1950 por Grace Murray Hopper. Entonces, el término fue visto como la compilación de una secuencia de rutinas seleccionadas. Ella popularizó la idea de los lenguajes de programación independientes de la máquina, los cuales necesitarían la traducción por un compilador a código máquina. Entre 1954 y 1957 en IBM se desarrolló FORTRAN, el considerado primer compilador de alto de nivel.

Tradicionalmente, los compiladores generaban código máquina de peor calidad que el que programadores humanos podían escribir. Sin embargo, actualmente los compiladores generan un código máquina de alta calidad pequeño y rápido [19].

Los compiladores son procesos complejos debido a que tiene varias fases, la fase de análisis y la fase de síntesis (Ilustración 12). En la fase de análisis, también conocida como *frontend* del compilador, el código fuente es leído y analizado léxica, sintáctica y semánticamente. Además, la tabla de símbolos es generada para almacenar los identificadores. Por tanto, esta fase es dividida en tres módulos: análisis léxico, análisis sintáctico y análisis semántico. En la fase de síntesis, también llamada *backend* del compilador, se genera el programa objeto con ayuda del código fuente, su representación intermedia y la tabla de símbolos. Esta fase consta de tres subfases, la generación de código intermedio, optimización de código y generación de código objeto [20].

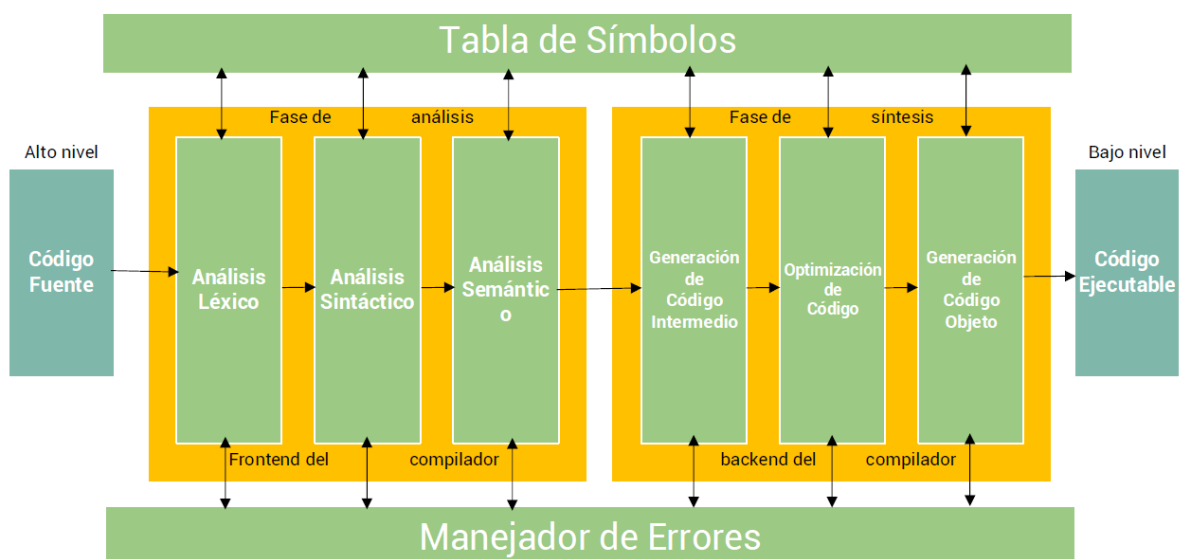


Ilustración 12: Módulos de un compilador

2.5.1 Análisis léxico

El análisis léxico es la primera fase del análisis. Su misión principal es la transformación de la secuencia de caracteres que componen el código fuente en una secuencia de unidades léxicas conocidas como *tokens*. Esta transformación permite al compilador identificar de manera estructurada los elementos del texto fuente, facilitando su procesamiento en fases posteriores. Para ello, el analizador léxico actúa como un filtro que lee el código carácter a carácter, agrupando conjuntos significativos de símbolos según los patrones definidos por el lenguaje de programación.

Cada *token* consta de dos elementos fundamentales: un código que identifica su clase léxica (como identificadores, palabras clave, operadores, constantes...)

y un atributo que contiene información adicional, como un puntero a la tabla de símbolos o un valor numérico. Durante esta fase, los identificadores nuevos son introducidos como entradas de la tabla de símbolos [20].

2.5.2 Análisis sintáctico

La segunda fase del análisis es el análisis sintáctico. En esta fase el compilador comprueba si la secuencia de *tokens* se ajusta a las reglas de la gramática del lenguaje fuente.

Para ello es imprescindible especificar la gramática del lenguaje fuente, la cual define cómo se pueden combinar los *tokens* para formar construcciones válidas en el lenguaje. El analizador sintáctico construye teóricamente un árbol de análisis para representar la relación jerárquica entre los *tokens* y sus categorías sintácticas. El resultado de esta fase es un árbol de análisis, como el de la Ilustración 13, que representa el significado y la lógica del código fuente [21] [22], que puede representarse por el *parse* (la relación de reglas usadas para construir el árbol).

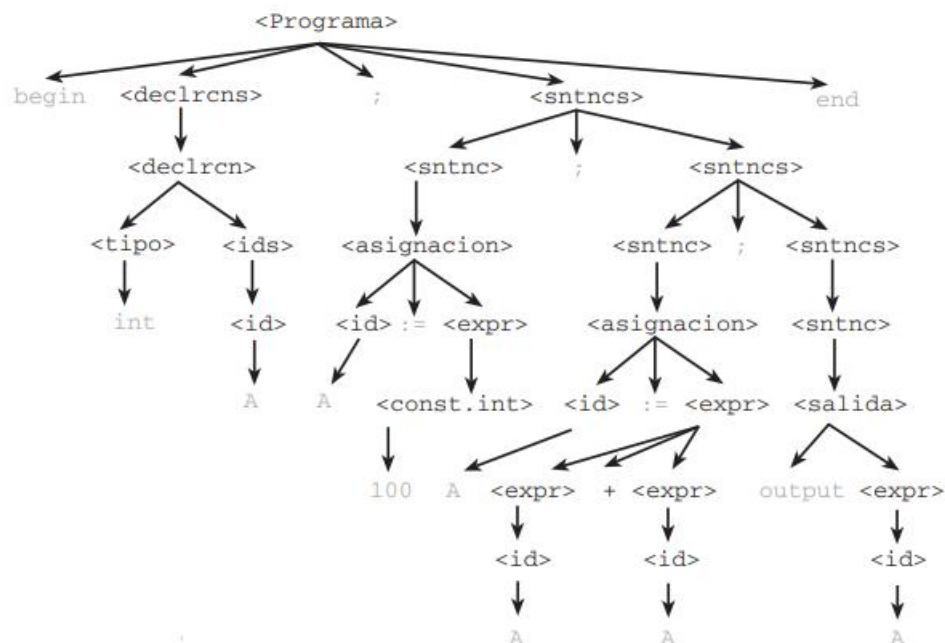


Ilustración 13: Árbol de análisis

2.5.3 Análisis semántico

La tercera etapa en el proceso de compilación es el análisis semántico. Durante esta fase, el compilador se encarga de validar que el árbol sintáctico generado previamente cumple con las normas semánticas del lenguaje fuente. El objetivo del análisis semántico es detectar errores como el uso de variables sin declarar, incompatibilidades entre los tipos, operaciones no permitidas o problemas relacionados con el alcance de las variables.

Además, esta etapa incluye funciones como la deducción de tipos o la conversión entre distintos tipos de datos. El resultado final del análisis semántico es un árbol sintáctico anotado, similar al que se muestra en la Ilustración 14, el cual incorpora información adicional sobre los tipos, valores y propiedades de los distintos elementos del código fuente [22].

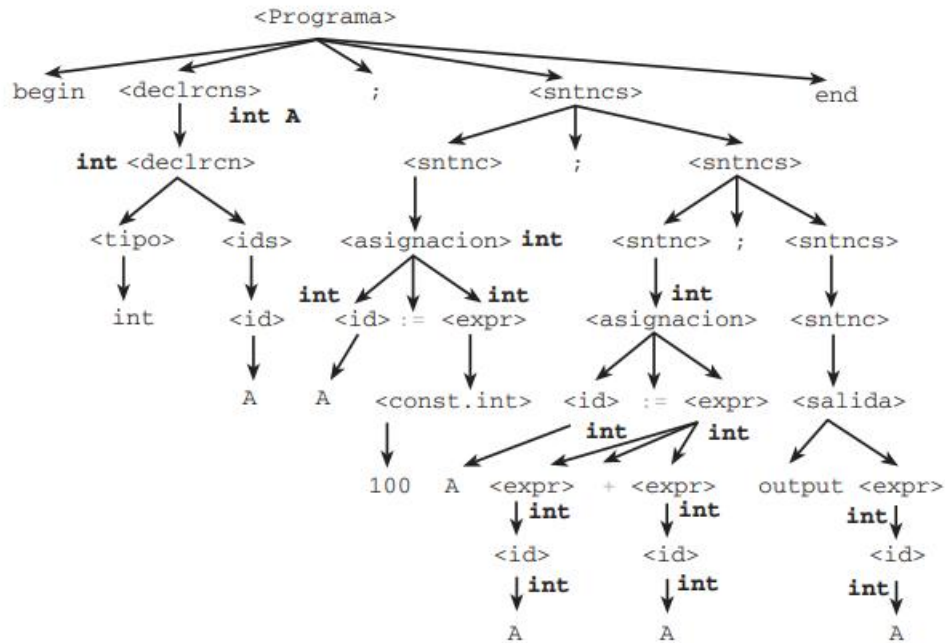


Ilustración 14: Árbol de análisis anotado.

2.5.4 Generación de código intermedio

Es la primera de las fases de síntesis; en esta fase se genera un código intermedio para una máquina abstracta. Esta representación debe ser fácil de producir y de traducir al programa objeto. Puede tener diversas formas, una de ellas es el llamado código de tres direcciones, el cual consiste en una secuencia de instrucciones cada una de las cuales involucra un máximo de una operación y un máximo de tres direcciones, las de los dos operandos y la del resultado. Además, deberá generar nombres temporales, introducidos en la tabla de símbolos, para almacenar los resultados intermedios [20] [22].

```
temp1 := InToReal(60)
temp2 := id26 * temp1
tem3 := id24 + temp2
id25 := temp3
```

Ilustración 15: Ejemplo código de tres direcciones

2.5.5 Optimización de código

Esta es la segunda fase de síntesis (aunque también puede aparecer al final del proceso); en ella se busca la mejora del código en el sentido de reducir la cantidad de recursos, tiempo y memoria, necesarios. Algunas optimizaciones son triviales, como hacer transformaciones, mientras que otras pueden requerir un trabajo mucho mayor, pero mejoran significativamente la eficiencia, como eliminar código inaccesible o eliminar variables intermedias. Muchos compiladores permiten elegir la cantidad de optimización a realizar.

2.5.6 Generación de código

La generación de código es la fase final de un compilador; en esta se genera el código objeto, normalmente código máquina o ensamblador. Se seleccionan en ella posiciones de memoria relativas o registros para el almacenamiento de las variables y cada sentencia de código intermedio se traduce a una secuencia de instrucciones que ejecutan la tarea [21].

2.5.7 Tabla de símbolos

La tabla de símbolos es una estructura de datos utilizada desde todas las fases de un compilador; en esta se almacenan todos los atributos asociados a los símbolos que se utilizan en el lenguaje de programación. Estos atributos pueden ser, por ejemplo, el tipo o la dirección de memoria.

Todas las fases del compilador se apoyan en ella para almacenar o para obtener información. El analizador léxico, sintáctico y semántico introducen la información, mientras que el generador de código intermedio, la fase de optimización y la de generación de código se apoyan en ella para generar el código necesario [22].

Por todo esto, la tabla de símbolos debe poder realizar la tarea de introducción de nuevas entradas, modificación de entradas ya presentes, detección de identificadores ya declarados y entrega de la información almacenada.

2.5.8 Gestor de errores

El gestor de errores se encarga de reaccionar a los errores detectados por cada una de las fases. Para ello debe ser capaz de informar al usuario de los errores detectados de forma clara y relacionándolos con el código fuente, indicando la línea en la que estos se encuentran y la fase en la que se ha detectado el error [21].

2.6 Gamificación

La gamificación es una técnica que se define como la aplicación de elementos y técnicas de diseño de juegos en contextos no lúdicos. Este enfoque busca trasladar la concentración y emociones vividas por el jugador al mundo real transformando cualquier experiencia en una experiencia lúdica [23].

La gamificación nace como un concepto novedoso en el mundo digital y tecnológico como respuesta a la creciente necesidad de hacer más atractivas y motivadoras las interfaces de usuario y experiencias digitales. En sus inicios, la gamificación se enfocaba principalmente en el sector empresarial, buscando incrementar la participación del cliente y mejorar la experiencia del usuario en sitios web y aplicaciones.

El término gamificación fue acuñado en 2002 por el británico Nick Pelling, el cual propuso la idea de integrar componentes del juego en contextos que tradicionalmente no están relacionados con el entretenimiento con el objetivo de fomentar ciertas conductas y optimizar las interacciones en una variedad de aplicaciones.

Inevitablemente, se produjo una transición de la gamificación hacia la educación mediante un proceso gradual. Educadores y desarrolladores reconocieron el potencial de los elementos del juego para mantener la concentración y mejorar el compromiso y la motivación en el aula, lo que se tradujo en la incorporación de sistemas de puntos, medallas o narrativas lúdicas a las actividades educativas tradicionales. Con el paso del tiempo, la gamificación en la educación ha experimentado una notable expansión y evolución. No se limita solo a la incorporación de puntos y recompensas, sino que también abarca la creación de entornos de aprendizaje inmersivos, simulaciones interactivas y desafíos basados en la resolución de problemas [24].

Algunos de los elementos principales de la gamificación son:

- **Mecánicas:** Son los componentes básicos de los juegos, como las reglas. Dentro de las mecánicas se encuentran herramientas que ayudan a involucrar al usuario en la experiencia, como el lugar donde ocurre el juego. Las mecánicas de la gamificación incluyen también elementos como los sistemas de puntos y niveles que mantienen al estudiante comprometido y motivado. Por todo esto, son consideradas la columna vertebral de la gamificación.
- **Dinámicas:** Las dinámicas están relacionadas con la motivación de los participantes y los involucra en la historia o la narrativa teniendo como objetivo despertar el interés de los estudiantes e impulsarlos a participar en la actividad. Algunas de las principales dinámicas son los retos y las recompensas.
- **Estética:** El tercer y último elemento de la gamificación es la estética, un elemento vital para esta. Un diseño atractivo y amigable para el usuario, como unos gráficos atractivos, una navegación intuitiva y una interfaz agradable, contribuye en una gran medida a la experiencia del aprendizaje haciendo que los estudiantes se sientan cómodos y comprometidos con el contenido, mejorando así la efectividad de la formación [25].

3 Planteamiento del problema

3.1 Introducción

Esta sección presenta la Especificación de Requisitos Software (ERS) de las modificaciones y ampliaciones realizadas en el sistema web DRACO durante el desarrollo del Trabajo de Fin de Grado (TFG).

3.1.1 Propósito

El objetivo de la especificación de requisitos es la definición de las modificaciones y ampliaciones que hay que realizar durante el desarrollo del TFG. En esta definición se reflejan las necesidades del sistema y los objetivos decididos para cumplimentar las mismas.

3.1.2 Ámbito del sistema

El trabajo parte de la disposición del sistema web DRACO, el cual está ya en funcionamiento y sirve de apoyo para las asignaturas de Procesadores de Lenguajes y Traductores de Lenguajes.

A través de este sistema, los alumnos pueden poner a prueba sus conocimientos de las asignaturas mediante la realización de distintas actividades, cuestionarios o test. Estas actividades pueden ser de 9 tipos distintos: Test de selección única, test de selección múltiple, ordenar, emparejar, respuesta corta, respuesta larga, secuencial, examen o encuesta.

Además, los alumnos tienen a su disposición en el sistema los módulos de corrección de la práctica de la asignatura. Estos permiten a los alumnos poder probar cada una de las fases del analizador que deben desarrollar. Las cuatro fases para las que disponen de pruebas son: léxico, sintáctico, semántico y tabla de símbolos.

Para que un alumno pueda realizar una comprobación con uno de estos módulos el sistema debe generar un fichero de prueba acorde a las opciones del alumno, el cual se genera semi-automáticamente a través de un sistema de plantillas y fragmentos de ficheros. Previamente, el alumno deberá haber definido las decisiones tomadas en el diseño de la práctica mediante la definición de *tokens*, la definición de terminales y la definición de tipos.

DRACO contiene elementos de la gamificación, es decir, elementos propios de los juegos. Algunos ejemplos de esto son su sistema de puntos y clasificación a través de un ranking global, el uso de medallas como recompensas al interactuar con el sistema o su moneda propia, el DRACoin, que permite comprar diferentes ventajas en la tienda.

Finalmente, existe una interfaz centrada en los administradores del sistema la cual permite a los profesores definir el lenguaje de la práctica, configurar los distintos módulos de esta y generar y crear actividades. Además, los profesores a través de esta interfaz pueden ver datos del desempeño de los alumnos en las distintas funcionalidades.

3.1.3 Definiciones, Acrónimos y Abreviaturas

En este apartado se presentan las definiciones y abreviaturas necesarias para comprender la ERS.

3.1.3.1 Definiciones

Término	Definición
Analizador o Procesador	El programa encargado de la realización de la primera fase de la compilación, el cual está formado por tres módulos: análisis léxico, análisis semántico y análisis sintáctico y se apoya en la tabla de símbolos.
Gamificación	Uso de elementos y técnicas pertenecientes al juego en entornos no lúdicos.
Actividad	Elemento para la realización por el usuario que permite evaluar sus conocimientos. Hay de distintos tipos.
Actividad de selección única	Actividad que presenta una pregunta y múltiples respuestas de las cuales solo se puede escoger una.
Actividad de selección múltiple	Actividad que presenta una pregunta y múltiples respuestas de las cuales se pueden escoger varias.
Actividad de respuesta corta	Actividad en la que se presenta una pregunta y el alumno tiene libertad para escribir una respuesta la cual puede ser clasificada como correcta o incorrecta a través de unas palabras clave.
Actividad de tipo ordenar	Actividad en la que se presentan varias respuestas que deben ser ordenadas correctamente.
Actividad de tipo emparejar	Actividad en la que se le presentan al alumno dos columnas de respuestas y este debe emparejar respuestas de ambas columnas según el enunciado.
Actividad de tipo secuencial	Actividad en la que el alumno puede realizar varias actividades de manera consecutiva ganando más puntos por cada actividad y pudiendo apostar puntos. Se puede finalizar la actividad sin haberla terminado.
Actividad de tipo examen	Actividad en la que se le presentan al alumno varias actividades de manera consecutiva y de las que no puede salir hasta terminarlas.
Fragmento de fichero	Fichero que contiene código fuente. Las pruebas de los comprobadores son generadas combinando varios de estos según las normas establecidas por una plantilla.
Plantilla	Conjunto de normas que indican cómo se combinan los fragmentos de ficheros fuente siguiendo restricciones para la generación de un fichero de prueba.
Zona aleatoria	Parte de un fragmento de fichero en la que se presentan múltiples opciones de código para añadir variedad a los ficheros de las pruebas.

Tabla 3: Tabla definiciones

3.1.3.2 Acrónimos y abreviaturas

Abreviatura	Significado
DRACO	Dinámica de Refuerzo para el Aprendizaje de COmpiladores
BBDD	Bases de datos.
PdL	Procesadores de Lenguajes.
TdL	Traductores de Lenguajes.
TFG	Trabajo de Fin de Grado
ERS	Especificación de Requisitos Software

Tabla 4: Tabla abreviaturas y acrónimos

3.2 Descripción general

En esta sección se va a exponer una visión general del sistema que se va a desarrollar. Se describirán las principales funcionalidades que deberá cumplir el sistema, la información necesaria para su funcionamiento y las limitaciones que podrían influir en su diseño e implementación. Además, se tendrán en cuenta otros aspectos que puedan afectar al desarrollo de este, como los requisitos no funcionales o posibles restricciones externas. Esta descripción servirá para contextualizar el proyecto y guiar las decisiones técnicas posteriores.

3.2.1 Perspectiva del producto.

El sistema cuenta con un módulo de corrección de prácticas el cuál será revisado en sus distintas partes. Este módulo cuenta en el sistema del administrador con un apartado para la gestión de ficheros fuente y plantillas. Una vez se han subido correctamente todos los ficheros de una plantilla y esta se ha definido, se lleva a cabo una comprobación de la plantilla en la que el sistema realiza todas las combinaciones posibles de las zonas aleatorias de un fragmento fuente y de todos los fragmentos fuentes disponibles para cada posición. En esta comprobación, actualmente se producen algunos errores que deben ser revisados. Por otro lado, en este módulo de corrección de prácticas se revisará el funcionamiento de la selección por parte del sistema del fichero para realizar la prueba a un grupo determinado, ya que se ha detectado la repetición de los mismos ficheros y la no selección de otros.

Además, se va a desarrollar un módulo de estadísticas de las actividades realizadas por los alumnos que proporcionará al profesorado información general sobre las respuestas proporcionadas. Este módulo se encontrará al revisar cada actividad y mostrará información diferente dependiendo del tipo de actividad que es. Además de esto, el módulo permitirá la opción de ordenar los datos de mayor a menor y viceversa en cada una de sus columnas para permitir al profesor ver detalladamente la información.

Además del trabajo en estos dos módulos, se van a llevar a cabo tareas de mantenimiento en otras áreas como la reducción del tiempo entre pruebas atendiendo a la puntuación obtenida por el alumno al realizarla, trabajos en el módulo de obtención de medallas en la información almacenada en las BBDD o la mejora del corrector del analizador léxico permitiendo detectar errores más precisos ante la falta de un *token* por parte del alumno.

3.2.2 Funciones del sistema.

A continuación, se describen las distintas funcionalidades del sistema relacionadas con este trabajo.

Prácticas:

- **Validación avanzada de plantillas:** El sistema realiza una comprobación de todas las combinaciones posibles generadas por una plantilla, variando los distintos ficheros para una misma posición y las combinaciones de zonas aleatorias dentro de estos, asegurando que se ajustan a la gramática y características definidas por el profesor. Se mejora el sistema de validación para evitar falsos errores en plantillas correctas.
- **Selección de ficheros y plantillas:** Cuando un grupo realiza una prueba, el sistema determina, tras seleccionar una plantilla, una combinación de fragmentos fuente y zonas aleatorias que sean acordes a las opciones previamente seleccionadas por el grupo. Esta selección de plantillas

deberá ser totalmente aleatoria acorde a la dificultad alcanzada por el grupo y su configuración de la práctica.

Estadísticas:

- **Tabla resumen por actividad:** En el módulo de actividades del administrador mostrará, para cada tipo de actividad, una tabla con datos agregados de los alumnos que la han completado. Esta información deberá incluir medidas como el número de aciertos, tiempo medio de resolución o porcentaje de éxito, facilitando una visión general del rendimiento del grupo.

Mantenimiento:

- **Reducción dinámica del tiempo entre pruebas:** El sistema reducirá el tiempo de espera entre la realización de pruebas en el módulo de prácticas proporcionalmente a la puntuación obtenida por el alumno. Esta reducción alcanzará un 75% del tiempo si el alumno obtiene la puntuación máxima, buscando premiar el buen desempeño.
- **Reducción manual del tiempo de espera:** El administrador, desde el panel de configuración del módulo de prácticas, podrá reducir manualmente el tiempo de espera entre pruebas para un grupo en concreto, indicando el porcentaje de esta reducción.
- **Filtro de medallas:** El sistema de configuración de actividades y el sistema de importación de actividades deben ofrecer filtros correctos y funcionales con todos los tipos de actividades y las medallas posibles para la signatura activa en el sistema.
- **Corrección detallada de errores léxicos:** Durante la evaluación de prácticas, el sistema identifica y clasifica con mayor precisión los errores léxicos producidos por la ausencia de *tokens* esperados en el análisis. Esta funcionalidad permite asignar puntuaciones más ajustadas según el tipo de fallo y mejora la retroalimentación ofrecida al alumno.

3.3 Requisitos

En esta sección se presentan todos los requisitos que deben ser cumplidos por el sistema a la finalización de este TFG.

3.3.1 Requisitos funcionales.

A continuación, se describen aquellos requisitos que detallan las funciones y características que explican lo que el sistema debe hacer. Estos han sido clasificados en tres grupos, identificados en la descripción general: Mantenimiento, Prácticas y Estadísticas.

3.3.1.1 Prácticas.

RQF 1.1 Comprobación de plantillas: El sistema debe permitir al administrador, desde el módulo de comprobación de prácticas, la realización de la comprobación de las plantillas correctamente subidas en el sistema.

RQF 1.2 Generación ficheros prueba: La comprobación de una planilla llevada a cabo por el sistema, forma los ficheros de prueba posibles con las restricciones establecidas

RQF 1.3 Validación normas lenguaje: El sistema, al realizar las comprobaciones, verifica si cada combinación de ficheros cumple con las normas del lenguaje definido para la realización de la práctica.

RQF 1.4 Plantillas disponibles: Las plantillas que el sistema detecte como correctas estarán disponibles para su selección de cara a la realización de una prueba por parte de los alumnos.

RQF 1.5 Ejecución de pruebas por el alumno: El alumno desde el módulo de prácticas de su sistema debe poder realizar una prueba de su procesador.

RQF 1.6 Selección aleatoria de plantillas: El sistema seleccionará, según la dificultad asignada al grupo de prácticas del alumno, una plantilla de acorde a esta dificultad aleatoriamente dentro de las disponibles en el sistema.

RQF 1.7 Selección ficheros compatibles: El sistema seleccionará, dentro de los ficheros disponibles para la plantilla seleccionada, aquellos que para cada posición sean compatibles con las opciones del lenguaje seleccionadas previamente por el grupo de prácticas del alumno.

RQF 1.8 Selección zonas aleatorias: El sistema seleccionará para cada zona aleatoria de cada fragmento de fichero fuente una opción de código que vaya de acuerdo con las opciones del grupo de prácticas del alumno.

RQF 1.9 Descarte de plantilla: Si el sistema no encuentra para una posición de la plantilla un fichero que cumpla con las opciones del alumno dicha plantilla será descartada y se seleccionará aleatoriamente otra disponible en el sistema.

RQF 1.10 Entrega del fichero al alumno: Si el sistema encuentra un fichero de acuerdo con las opciones del grupo de prácticas para cada posición de la plantilla seleccionada, este fichero de prueba generado es entregado al alumno para su descarga.

RQF 1.11 Cambio de dificultad: Si el sistema no encuentra ninguna plantilla de acuerdo con las opciones del grupo de prácticas en las plantillas de la dificultad seleccionada se procederá a la selección de plantillas de otra dificultad menor.

RQF 1.12 Error si no hay plantilla válida: Si el sistema no encuentra ninguna plantilla de acuerdo con las opciones del grupo dentro de todas las disponibles en el sistema, generará un error que informará al alumno de la situación.

3.3.1.2 Estadísticas de actividades.

RQF 2.1 Tablas resumen actividades: El módulo de revisión de una actividad dispondrá de una o varias tablas resumen que contenga estadísticas de la actividad adaptada al tipo de actividad.

RQF 2.1.1 Estadísticas actividades de selección: Para las actividades de tipo respuesta única, respuesta múltiple, encuesta con selección única y encuesta con selección múltiple mostrará la respuesta, las veces que ha sido seleccionada, las veces que ha sido enseñada y el porcentaje de veces seleccionada sobre las veces enseñadas.

RQF 2.1.2 Estadísticas actividades respuesta corta: Para las actividades de tipo respuesta corta el sistema mostrará la respuesta y las veces que ha sido respondida. Las respuestas correctas serán agrupadas como una única respuesta, mostrando la más repetida.

RQF 2.1.3 Estadísticas actividades ordenar: Para las actividades de tipo ordenar, el sistema mostrará la respuesta, las veces que ha sido seleccionada en el orden correcto, las veces que ha sido mostrada para ordenar y el porcentaje de veces ordenada en la posición correcta sobre las veces mostrada.

RQF 2.1.4 Estadísticas actividades emparejar: Para las actividades de tipo emparejar el sistema mostrará dos tablas diferentes. La primera contendrá las parejas correctas, mostrando sus dos componentes, las veces que han sido seleccionadas correctamente, las veces que han sido mostradas entre las respuestas y el porcentaje de veces respondida correctamente sobre veces enseñada. En la segunda tabla se mostrarán todas las parejas incorrectas realizadas por los alumnos, mostrando ambos pares de estas y las veces que se ha emparejado de esta manera.

RQF 2.1.5 Estadísticas actividades secuencial: Para las actividades de tipo secuencial el sistema mostrará el número de actividades que ha realizado el alumno, cuántos han llegado a realizar este número y la puntuación media en forma de porcentaje obtenida por estos alumnos.

RQF 2.1.6 Estadísticas actividades examen: Para las actividades de tipo examen el sistema mostrará dos tablas. En la primera se analizan los resultados por intervalo de nota mostrando cuántos alumnos han obtenido una puntuación dentro de cada intervalo y el porcentaje que representan estos sobre el total. En la segunda tabla se mostrarán el nombre de las actividades mostradas a los alumnos, las veces que ha sido respondida y la puntuación media obtenida en dicha actividad.

RQF 2.2 Reordenación de columnas: La tabla de revisión debe permitir al usuario reordenar descendente o ascendentemente las columnas mediante botones asociados a cada encabezado. Por defecto las filas deben estar ordenadas descendentemente según el porcentaje calculado en cada caso.

RQF 2.3 Colores para respuestas: El sistema debe aplicar un estilo visual en color verde a las respuestas consideradas correctas, y un estilo en color rojo a las respuestas incorrectas, en la tabla de revisión de actividades. Esta información irá acompañada de un símbolo de *check* o un aspa.

RQF 2.4 Soporte visual respuestas con imágenes: Las actividades cuyas respuestas contienen una imagen deben mostrar esta y adaptar la tabla a la altura y ancho correspondiente.

3.3.1.3 Mantenimiento.

RQF 3.1 Actualización automática tiempo entre pruebas: El sistema actualizará automáticamente, tras la realización de una prueba por parte del alumno, el tiempo personalizado entre pruebas si este obtiene una puntuación mayor a 0.5 sobre 1.

RQF 3.2 Cálculo reducción de tiempo: El cálculo del porcentaje de reducción de tiempo se calculará de forma que, con una puntuación máxima, es decir de 1, se reduce en un 75% y con una puntuación menor a 0.5 la reducción es de 0. Las puntuaciones intermedias son proporcionales.

RQF 3.3 Visualización próxima prueba: Al consultar los datos de un grupo para un módulo de pruebas el sistema debe mostrar la fecha y hora a la que tendrá disponible la realización de la siguiente prueba.

RQF 3.4 Reducción manual tiempo entre pruebas: El sistema permite al administrador, al consultar los datos de un grupo, introducir un porcentaje para la reducción manual del tiempo entre pruebas para dicho grupo en dicho módulo.

RQF 3.5 Notificación de reducción aplicada: Tras pulsar el botón de la reducción de tiempo, el sistema debe informar de la reducción aplicada y la nueva fecha y hora de entrega del grupo para dicho módulo de prácticas.

RQF 3.6 Reducción acumulativa: Si el alumno ya ha reducido su tiempo entre pruebas mediante otros métodos, la reducción del tiempo debe ser aplicada sobre dicho tiempo ya reducido.

RQF 3.7 Filtro por tipo de actividad: El filtro de búsqueda de la sección de configuración de actividades por tipo de actividad debe mostrar para seleccionar los 12 tipos de actividades disponibles en el sistema por el momento.

RQF 3.8 Filtro por medalla: El filtro de búsqueda en la sección de configuración por medalla debe mostrar para seleccionar las medallas que se pueden obtener gracias a la realización de una actividad correspondiente a la asignatura (PdL o TdL) que está activa en el sistema.

RQF 3.9 Detección de error por token: El sistema detectará como un nuevo error la diferencia de un único *token* entre la lista entregada por el alumno y la esperada por el sistema. Esta diferencia puede ser dada por la falta de un *token*, el exceso de un *token* o por la aparición de un *token* erróneo.

RQF 3.10 Corrección continua: El sistema debe continuar con la corrección de los *tokens* tras la detección de un *token* erróneo.

RQF 3.11 Penalización por tokens incorrectos: El sistema reducirá la puntuación obtenida por el alumno según el número de *tokens* incorrectos encontrados durante la corrección del fichero, la penalización por cada fallo estará indicada en el fichero de configuración de datos.

RQF 3.12: El sistema debe mostrar en el módulo de configuración de datos del comprobador la penalización actual por tener un *token* Incorrecto.

RQF 3.13 Configuración nueva penalización: El administrador debe poder introducir en el módulo de configuración de datos del comprobador léxico una nueva puntuación de penalización por tener un *token* incorrecto.

3.3.2 Requisitos no funcionales.

RQNF 1 Tecnologías utilizadas: El sistema deberá ser desarrollado utilizando PHP 7.4 garantizando compatibilidad con navegadores actuales. Se emplearán HTML5 y CSS3 para la interfaz, y el código deberá estar adecuadamente documentado según los estándares del lenguaje utilizado.

RQNF 2 Compatibilidad con entorno: El sistema debe ser compatible con el entorno de despliegue actual, basado en un servidor Apache sobre Linux y una base de datos relacional MySQL. Todo desarrollo deberá asegurar el correcto funcionamiento dentro de este entorno y respetar sus restricciones.

RQNF 3 Seguridad del sistema: El sistema debe implementar mecanismos de seguridad para evitar accesos indebidos y ataques como inyección SQL, XSS o manipulación de roles. Toda acción crítica deberá validarse mediante autenticación y control de permisos.

RQNF 4 Accesibilidad: El sistema debe cumplir con las pautas de accesibilidad WCAG 2.2 en nivel AA, al menos en las vistas orientadas al alumnado. Se prestará especial atención al contraste visual, navegación con teclado y uso de etiquetas semánticas.

RQNF 5 Robustez: El sistema debe ser robusto ante errores y fallos inesperados. En caso de error, se debe mantener la operatividad general del sistema, mostrar mensajes adecuados al usuario y registrar internamente el fallo para su análisis posterior.

RQNF 6 Diseño flexible: El diseño del sistema deberá ser *responsive*, adaptándose a distintos tamaños de pantalla y dispositivos, incluyendo ordenadores, tabletas y teléfonos móviles, sin pérdida de información ni funcionalidad.

4 Solución

4.1 Diseño

4.1.1 Diseño de la base de datos.

Para dar una visión global del sistema a continuación, se presentan en la Ilustración 16 y en la Ilustración 17 el diagrama entidad relación de las tablas usadas durante el desarrollo de este TFG. Además, se proporcionará una descripción de las tablas que incluyen y los campos que contienen junto con una descripción.

4.1.1.1 Base de datos del módulo de actividades.

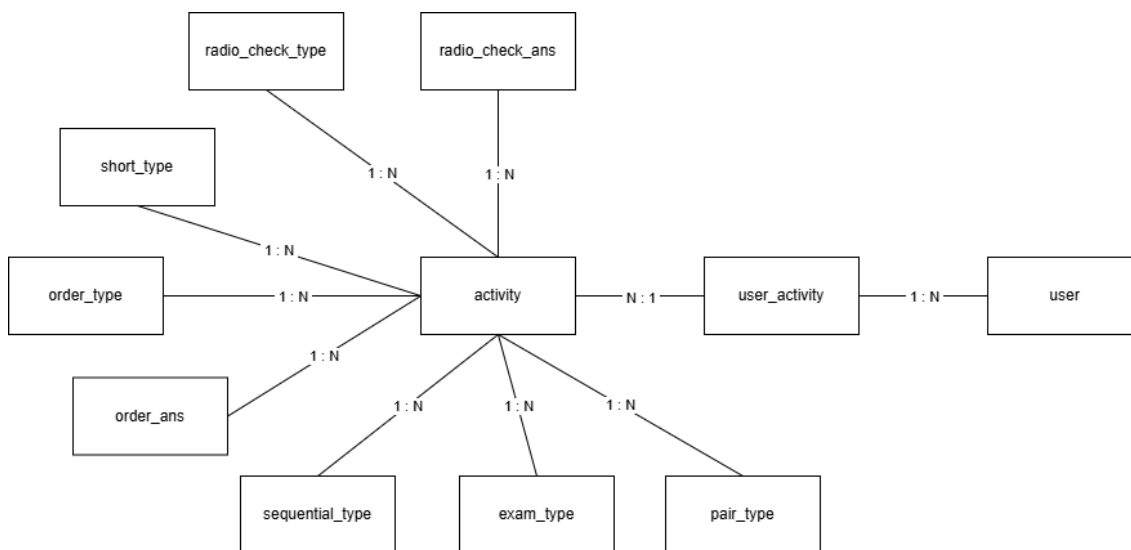


Ilustración 16: Diagrama base de datos draco

A continuación, se realiza una descripción de las tablas pertenecientes a la base de datos draco, que contiene información sobre las actividades, junto con los campos más relevantes de estas. Los campos que actúan como clave en cada tabla están destacados.

Tabla user_activity: Almacena la información obtenida cuando un usuario realiza una actividad.

Campo	Descripción
User_enrolment_number	Número de matrícula de usuario
Activity_id	Identificador de la actividad
Datedone	Fecha y hora de realización de la actividad
Done	1 si la actividad ha sido completada y 0 en caso contrario.
Score	Puntuación obtenida del 0 al 1.
Points_award	Recompensa de puntos del sistema entregados al alumno tras la realización de la actividad.
Time	Tiempo que el usuario ha tardado en completar la actividad
Answer	Identificadores de las respuestas escogidas por el alumno.
Offered	Identificadores de las respuestas mostradas al alumno.

Tabla 5: Tabla user_activity

Tabla activity: Contiene la información general de las actividades existentes en el sistema.

Campo	Descripción
Id	Identificador de la actividad.
Type	Tipo de la actividad
Status	Estado de la actividad. Sus valores pueden ser activa, inactiva, editando, incompleta y auto.
Title	Título de la actividad.
Req_level	Nivel requerido para realizar la actividad.
Idate	Fecha a partir de la que se permite la realización de la actividad.
Ddate	Fecha hasta la que se permite la realización de la actividad.
Time_limit	Tiempo disponible para realizar la actividad.
Question	Pregunta.
Archived	0 si no está archivada y 1 en caso contrario.
Course	Curso al que pertenece la actividad.
Badge_id	Identificador de la insignia asociada.
Photo_question	Ruta absoluta dentro del servidor de la imagen asociada a la actividad
Height_image	Altura de la imagen asociada a la pregunta.
Image_desc	Texto alternativo de la imagen asociada a la pregunta.

Tabla 6: Tabla activity

Tabla radio_check_type: Contiene los datos específicos de las actividades de tipo selección única y selección múltiple.

Campo	Descripción
Activity_id	Identificador de la actividad asociada.
Type	Tipo de la actividad. Toma los valores radio o checkbox.
Num_choices	Número de respuestas ofrecidas al alumno.
Min_correct	Número de mínimas respuestas correctas para ser ofrecidas.
Max_correct	Número máximo de respuestas correctas para ser ofrecidas.
Penalty	Penalización por cada respuesta incorrecta.
Images	1 si contiene imágenes y 0 en caso contrario
HeightImg	Altura por defecto de las imágenes de las respuestas.

Tabla 7: Tabla radio_check_type

Tabla radio_check_ans: Contiene la información de las respuestas de las actividades de tipo selección única y selección múltiple.

Campo	Descripción
Id	Identificador de la respuesta.
Radio_check_type_activity_id	Identificador de la actividad.
Answer	Texto ofrecido en la respuesta.
Explanation	Explicación de la respuesta.
Correct	1 si es una respuesta correcta y 0 en caso contrario.
Answer_image	Ruta en el servidor de la imagen de la respuesta.
Height_image	Tamaño de la imagen.
Image_desc	Texto alternativo de la imagen.

Tabla 8: Tabla radio_check_ans

Tabla short_type: Contiene los datos específicos de las actividades de tipo respuesta corta.

Campo	Descripción
Activity_id	Identificador de la actividad.
Keywords	Lista de las palabras claves que se dan por correctas en una respuesta.
Explanation	Explicación de la respuesta de la actividad.

Tabla 9: Tabla short_type

Tabla order_type: Contiene los datos específicos de las actividades de tipo ordenar.

Campo	Descripción
Activity_id	Identificador de la actividad.
Num_choices	Número de respuestas ofrecidas al alumno.
Num_ans2order	Número de respuestas a ordenar
Penalty	Penalización por cada respuesta incorrecta.
Images	1 si las respuestas tienen una imagen asociada y 0 en caso contrario.
HeightImg	Tamaño de las imágenes de las respuestas.

Tabla 10: Tabla order_type

Tabla order_ans: Contiene los datos específicos de las respuestas de las actividades de tipo ordenar.

Campo	Descripción
OAnswer_id	Identificador de la respuesta.
Order_type_activity_id	Identificador de la actividad.
Answer	Texto ofrecido en la respuesta.
Explanation	Explicación de la respuesta.
Order	Orden correcto.
Answer_image	Ruta en el servidor de la imagen de la respuesta.
Height_image	Tamaño de la imagen.
Image_desc	Texto alternativo de la imagen.

Tabla 11: Tabla order_ans

Tabla sequential_type: Contiene los datos específicos de las actividades de tipo secuencial.

Campo	Descripción
Activity_id	Identificador de la actividad
Num_activities	Número máximo de actividades que se pueden realizar por intento.

Tabla 12: Tabla sequential_type

Tabla exam_type: Contiene los datos específicos de las actividades de tipo examen.

Campo	Descripción
Activity_id	Identificador de la actividad
Num_activities	Número máximo de actividades que se realizan por intento.

Tabla 13: Tabla exam_type

Tabla pair_type: Contiene los datos específicos de las actividades de tipo emparejar.

Campo	Descripción
Activity_id	Identificador de la actividad.
Num_pairs	Numero de parejas mostradas en la actividad.
Images	1 si contiene imágenes y 0 en caso contrario.

Tabla 14: Tabla pair_type

Tabla pair_ans: Contiene los datos específicos de las respuestas de las actividades de tipo emparejar.

Campo	Descripción
pair_id	Identificador de la pareja de respuestas
Pair_type_activity_id	Identificador de la actividad.
Answer_pos	Posición de la respuesta dentro de la pareja. 1 si corresponde a la parte izquierda y 2 si corresponde a la derecha.
PAnswer_id	Identificador de la respuesta
Answer	Texto ofrecido en la respuesta.
Explanation	Explicación de la respuesta.
Answer_image	Ruta en el servidor de la imagen de la respuesta.
Height_image	Tamaño de la imagen.
Image_desc	Texto alternativo de la imagen.

Tabla 15: Tabla pair_ans

4.1.1.2 Base de datos del módulo de prácticas.

A continuación, se realiza una descripción de las tablas pertenecientes a la base de datos que contienen información sobre los módulos de prácticas y sus comprobadores, junto con los campos más relevantes de estas. La descripción se realiza de la base de datos del comprobador léxico para ilustrarlo, pero las bases de datos de los otros módulos siguen una estructura similar. Los campos que actúan como clave en cada tabla están destacados.

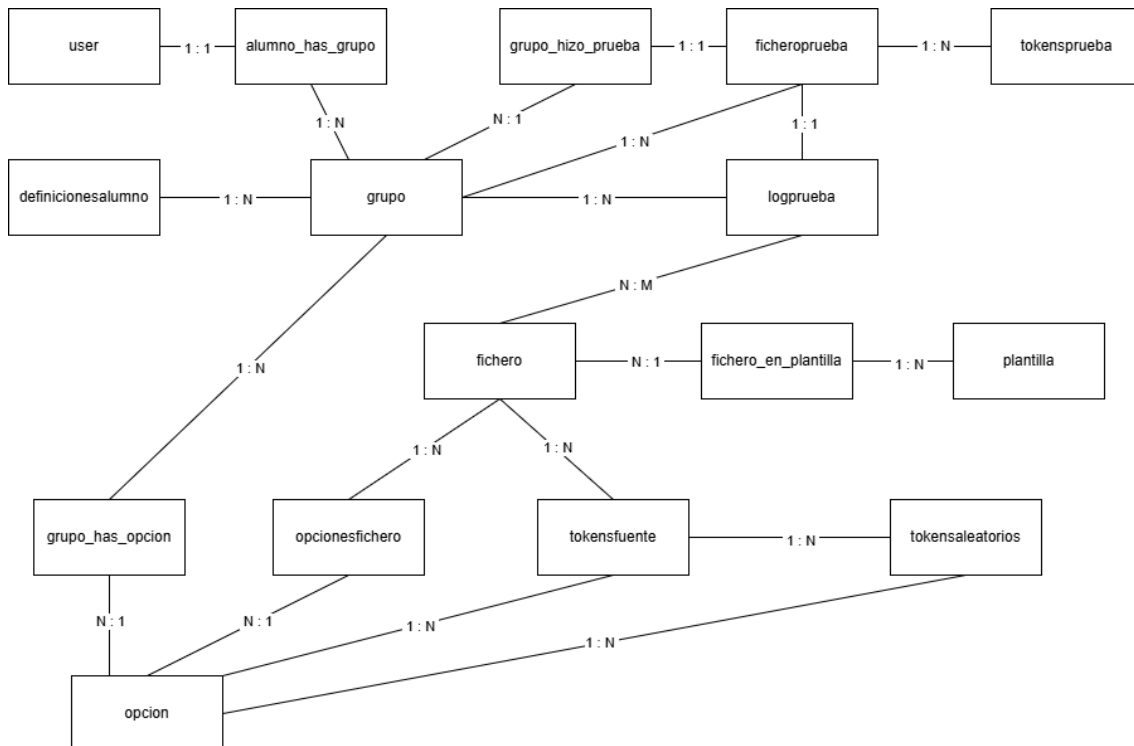


Ilustración 17: Diagrama dese de datos draco_clexico

Tabla user: Contiene los datos de un usuario.

Campo	Descripción
IdAlumno	Identificación del alumno
nombreAlumno	Nombre del alumno
DNI	DNI del alumno
Matrícula	Número de matrícula del alumno
Apellidos	Apellidos del alumno
Correo	Correo electrónico oficial del alumno
Clase	Identificador del grupo de clase en el que el alumno pertenece al cursar la asignatura.

Tabla 16: Tabla user

Tabla grupo: Contiene los datos relativos al grupo de prácticas.

Campo	Descripción
IdGrupo	Número del grupo
Dificultad	Nivel máximo de dificultad de las pruebas a las que se puede enfrentar el grupo.
Intentos	Número de pruebas que puede realizar el grupo.
Superados	Número de pruebas que el grupo ha realizado dentro del tiempo disponible.
IntentosGastados	Número de pruebas consumidas totales.
tiempoCustomEntrePruebas	Tiempo en minutos entre pruebas asignado al grupo.

Tabla 17: Tabla grupo

Tabla alumno_has_grupo: Relaciona a un usuario (alumno) con un grupo de prácticas.

Campo	Descripción
idAlumno_ag	Identificador del alumno.
idGrupo_as	Número del grupo.

Tabla 18: Tabla alumno_has_grupo

Tabla grupo_hizo_prueba: Contiene los datos relativos a las pruebas realizadas por los grupos.

Campo	Descripción
IdGrupo	Número del grupo.
IdPrueba	Identificador del fichero entregado
Fecha	Fecha en la que se realizó la prueba.
FormaFichero	0 si se trataba de un fichero correcto y 1 en caso contrario.
ComprobacionAtributos	Indica si los ficheros <i>tokens</i> eran correctos
Puntuación	Puntos otorgados a cada miembro del grupo tras realizar una prueba.
Score	Nota obtenida al realizar la prueba entre 0 y 1.

Tabla 19: Tabla grupo_hizo_prueba

Tabla opción: Contiene las posibles opciones del lenguaje.

Campo	Descripción
IdOpción	Identificador de la opción
nombreOpcion	Descripción de la opción.

Tabla 20: Tabla opción

Tabla grupo_has_opcion: Relaciona los grupos con sus opciones obligatorias del lenguaje.

Campo	Descripción
idGrupo_go	Número del grupo.
idOpcion_go	Identificador de la opción.

Tabla 21: Tabla grupo_has_opcion

Tabla fichero: Contiene los datos de los fragmentos de ficheros subidos por el administrador al sistema.

Campo	Descripción
Id	Identificador del fichero.
Nombre	Dirección de la localización del fichero en el sistema.
Tipo	Posición del fichero dentro de una plantilla.
VersionActual	1 si este fichero es la versión más reciente en el sistema y 0 en caso contrario.
IdLenguaje	Identificador del lenguaje al que pertenece el fichero.

Tabla 22: Tabla fichero

Tabla ficheropruueba: Contiene la información relativa a los ficheros entregados a los alumnos al realizar una prueba.

Campo	Descripción
IdPrueba	Identificador del fichero.
IdGrupo	Número del grupo.
FechaLimite	Fecha hasta la cual los alumnos pueden realizar la prueba.
Ruta	Localización del fichero en el sistema.
IdLog	Identificador de la composición del fichero.
Dificultad	Dificultad de la plantilla usada.
Actual	1 si es el último fichero generado para el grupo y 0 en caso contrario.

Tabla 23: Tabla ficheropruueba

Tabla opcionesfichero: Contiene la información relativa a las opciones que necesitan los grupos para poder obtener un fichero.

Campo	Descripción
Id_opcion_fichero	Identificador de cada opción asignada al fichero.
Fichero	Identificador del fichero
Aleatorio	1 si la opción pertenece a una zona aleatoria y 0 en caso contrario.
IdAleatorio	Identificador de la zona aleatoria a la que pertenece la opción.
IdAleatorioNumInterno	Identificador de la opción dentro de la zona aleatoria a la que pertenece la opción.
Opcion	Nombre de la opción.

Tabla 24: Tabla opcionesfichero

Tabla plantilla: Contiene la información relativa a las plantillas que indican cómo generar las pruebas.

Campo	Descripción
IdPlantilla	Identificador de la plantilla.
Nombre	Nombre de la plantilla.
NumCampos	Número mínimo de fragmentos fuente que debe tener el fichero generado.
NumFicherosLibres	Número máximo de fragmentos sin posición que se pueden añadir al fichero generado.
Dificultad	Dificultad de la plantilla.
FechaInicial	Fecha a partir de la cual estará disponible la plantilla.
FechaFinal	Fecha hasta la que estará disponible la plantilla.
Descripción	Descripción breve de la plantilla.
Comprobada	1 si la plantilla ha sido comprobada correctamente y 0 en caso contrario.
IdLenguaje	Identificador del lenguaje al que pertenece

Tabla 25: Tabla plantilla

Tabla log_prueba: Contiene información relativa a los ficheros generados para los alumnos.

Campo	Descripción
IdLog	Identificador del fichero de prueba.
IdPrueba	Identificador de la prueba a la que corresponde.
IdGrupo	Identificador del grupo de prácticas.
IdPlantilla	Identificador de la plantilla seleccionada para generar la prueba.
IdFicheros	Identificadores de los ficheros seleccionados para la generación de la prueba.
OpcionesElegidas	Identificadores de las opciones de las zonas aleatorias seleccionadas al generar la prueba

Tabla 26: Tabla log_prueba

Tabla fichero_en_plantilla: Contiene la información necesaria para relacionar los ficheros con las plantillas a las que pertenecen.

Campo	Descripción
IdFichero	Identificador del fichero.
IdPlantilla	Identificador de la plantilla.
Posicion	Posición que ocupa el fichero dentro de la plantilla.

Tabla 27: Tabla fichero_en_plantilla

Tabla definicionesalumno: Contiene las decisiones de los alumnos relativas a la información del formato de los *tokens*.

Campo	Fichero
idDefinicionAlumno	Identificador de la definición.
IdGrupo	Número del grupo.
Lexema	Patrón que siguen los lexemas de un <i>token</i> .
codigoToken	Nombre dado al <i>token</i> por el alumno en la definición de <i>tokens</i> .
codigoTokenAdministrador	Nombre dado por el administrador al <i>token</i> .
atributo	Tipo de atributo del <i>token</i> según el alumno.

Campo	Fichero
atributoAdministrador	1 si el <i>token</i> puede tener atributo y 0 en caso contrario.

Tabla 28: Tabla definicionesalumno

Tabla *tokens_fuente*: Contiene la información de los *tokens* encontrados en los fragmentos de fichero.

Campo	Descripción
Fichero	Identificador del fichero al que pertenecen los <i>tokens</i> .
NIdToken	Identificador del <i>token</i> dentro del fichero.
<i>Token</i>	Nombre del <i>token</i> .
Simbolo	Lexema del <i>token</i>
Linea	Línea en la que se posiciona el <i>token</i> dentro del fragmento.
Opcion	Restricción necesaria para que el <i>token</i> pueda ser mostrado al alumno.
Tipo	Indica si es un elemento, un <i>token</i> de fin de sentencia o de fin de fichero.
Bloque	1 si el <i>token</i> forma un bloque y 0 en caso contrario.
Estructura	Si el <i>token</i> forma un bloque indica a qué estructura pertenece este.
LugarEstructura	Indica el lugar de la estructura en la que debe estar. Dentro o a continuación.
Prioridad	Si el <i>token</i> es un operando indica la prioridad de este.
Asociatividad	Si el <i>token</i> es un operando indica su asociatividad
Cardinalidad	Si el <i>token</i> es un operando indica su cardinalidad.
Pre	Si el <i>token</i> es un operando indica su precedencia.

Tabla 29: Tabla *tokens_fuente*

Tabla *tokens_aleatorios*: Contiene la información e los *tokens* encontrados en las zonas aleatorias.

Campo	Descripción
Fichero	Identificador del fichero al que pertenecen los <i>tokens</i> .
NIdToken	Identificador del <i>token</i> dentro del fichero.
NOpcion	Identificador de la zona aleatoria en la que se encuentra el <i>token</i>
NumToken	Identificador del <i>token</i> dentro de la zona aleatoria en la que se encuentra.
<i>Token</i>	Nombre del <i>token</i> .
Simbolo	Lexema del <i>token</i>
Linea	Línea en la que se posiciona el <i>token</i> dentro del fragmento.
Opcion	Restricción necesaria para que el <i>token</i> pueda ser mostrado al alumno.
Tipo	Indica si es un elemento, un <i>token</i> de fin de sentencia o de fin de fichero.
Bloque	1 si el <i>token</i> forma un bloque y 0 en caso contrario.
Estructura	Si el <i>token</i> forma un bloque indica a qué estructura pertenece este.
LugarEstructura	Indica el lugar de la estructura en la que debe estar. Dentro o a continuación.
Prioridad	Si el <i>token</i> es un operando indica la prioridad de este.

Campo	Descripción
Asociatividad	Si el <i>token</i> es un operando indica su asociatividad
Cardinalidad	Si el <i>token</i> es un operando indica su cardinalidad.
Pre	Si el <i>token</i> es un operando indica su precedencia.

Tabla 30: Tabla tokens_aleatorios

Tabla tokens_prueba: Contiene la información de los *tokens* pertenecientes a una prueba generada.

Campo	Descripción
IdPrueba	Identificador de la prueba a la que corresponde
NidToken	Identificador del <i>token</i> dentro del fichero.
<i>Token</i>	Nombre del <i>token</i> .
Simbolo	Lexema del <i>token</i>
Linea	Línea en la que se posiciona el <i>token</i> dentro del fragmento.
Tipo	Indica si es un elemento, un <i>token</i> de fin de sentencia o de fin de fichero.
Prioridad	Si el <i>token</i> es un operando indica la prioridad de este.
Asociatividad	Si el <i>token</i> es un operando indica su asociatividad
Cardinalidad	Si el <i>token</i> es un operando indica su cardinalidad.

Tabla 31: Tabla tokens_prueba

4.1.2 Diseño de los requisitos funcionales.

4.1.2.1 Prácticas.

En esta sección se describe el diseño de los requisitos funcionales que se engloban el apartado de prácticas del sistema.

RQF 1.1 a RQF 1.4: Validación de plantillas de pruebas.

A continuación, se detalla el diseño de los requisitos comprendidos entre el 1.1 y el 1.4, centrados en la funcionalidad de validación de plantillas, la cual es una funcionalidad ya existente en el sistema, pero cuyo funcionamiento no es el esperado en algunas situaciones. Para cumplimentar dichos requisitos desde el módulo de comprobación de prácticas el sistema permitirá al administrador realizar la comprobación de una plantilla que podrá seleccionar si desea realizar una comprobación rápida o extendida. Para la comprobación rápida el sistema realizará un análisis estructural sintáctico de los ficheros pertenecientes a la plantilla, formando según las posiciones en esta todas las combinaciones posibles de estos y sus zonas aleatorias. Para cada combinación se comprobará que sigue con las normas y sintaxis del lenguaje definido, comprobando para las palabras reservadas que forman bloques o que deben estar dentro o a continuación de uno que estén en la posición correcta. Finalmente, una plantilla cuyas comprobaciones han sido todas validadas se marca como comprobada en el sistema y pasará a estar disponible a la hora de realizar una prueba.

RQF 1.5 a RQF 1.12: Generación de fichero de prueba a alumno.

Para la implementación de estos requisitos cuando un alumno accede al módulo de prácticas y pulsa el botón de realizar una práctica el sistema comienza con el proceso de generación de un fichero de prueba adaptado a las características del lenguaje seleccionadas previamente por su grupo, el proceso que sigue se muestra en el diagrama de la Ilustración 18.

Para ello, el sistema seccionará automáticamente una plantilla dentro de las disponibles en el sistema que concuerde con la dificultad asignada al grupo. Esta funcionalidad está actualmente implementada en el sistema, sin embargo, se duda de la aleatoriedad de la selección por lo que esto se pretende comprobar y arreglar.

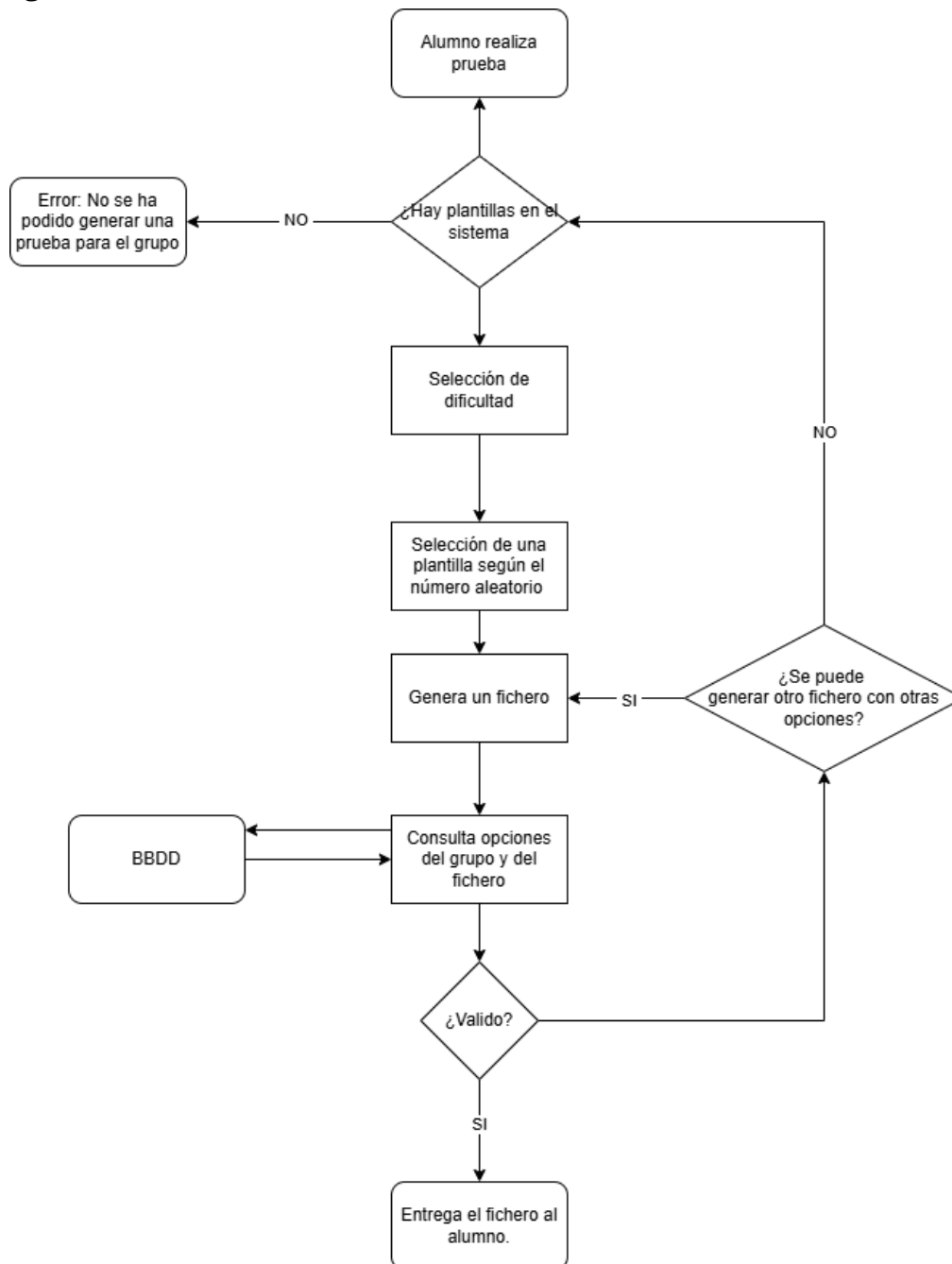


Ilustración 18: Diagrama generación fichero prueba.

El sistema que selecciona la plantilla actualmente funciona de la siguiente forma: El sistema genera las listas de las plantillas disponibles en el sistema por dificultad, selecciona la dificultad asignada al grupo, en caso de que esta lista estuviera vacía bajaría 1 nivel la dificultad. Una vez seleccionada la lista genera un número aleatorio entre 0 y el tamaño de la lista menos 1 y selecciona la plantilla ubicada en dicha posición. Para esta plantilla comprueba para cada una de las posiciones dentro de esta la existencia de un fichero que pueda ser asignado al grupo, es decir cuya combinación de opciones sea compatible con

las opciones seleccionadas por el grupo. Esto se realiza comparando los textos de las opciones asignadas al fichero como las asignadas al grupo.

En caso de encontrar una posición para la que no se pudiera seleccionar un fichero esta plantilla sería descartada, siendo eliminada de las listas mencionadas previamente y se reiniciaría el proceso de selección siempre y cuando las listas de plantillas no estuvieran vacías, ya que en este caso la generación de una prueba finalizaría mostrando un error al alumno.

En caso de encontrar un fichero posible para cada posición, el sistema generaría el fichero de prueba final juntando los fragmentos seleccionados para cada posición y este fichero de prueba será entregado al alumno para su prueba.

4.1.2.2 Estadísticas de actividades

En esta sección se describe el diseño de los requisitos funcionales que se engloban el apartado de estadísticas de actividades del sistema.

RQF 2.1 Tablas resumen actividades

Para ofrecer al administrador un resumen de los resultados de las actividades realizadas por los alumnos, el sistema genera una o varias tablas estadísticas que varían en función del tipo de actividad (Ilustración 19).

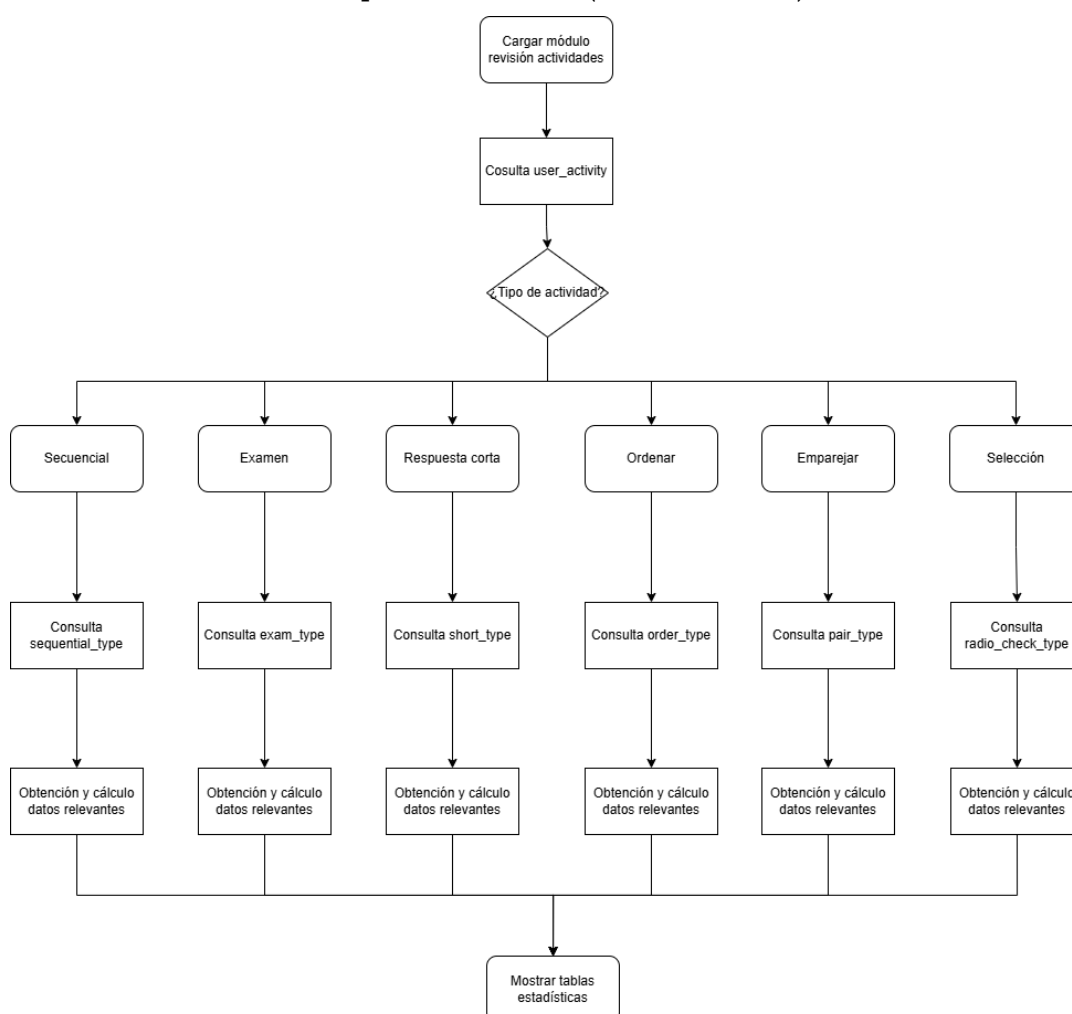


Ilustración 19: Diagrama mostrar tablas actividad

Por ejemplo, en actividades de respuesta única o múltiple, se cuentan las veces que cada opción ha sido seleccionada y mostrada, calculando el porcentaje relativo para cada una.

En actividades de respuesta corta, las respuestas correctas se agrupan, mostrando la respuesta más común con la frecuencia de acierto total y todas las respuestas incorrectas cada una con su frecuencia. Para actividades de ordenar, se cuentan las veces que la respuesta ha sido posicionada correctamente y el porcentaje de acierto. En actividades de emparejar, se distinguen las parejas correctas e incorrectas con sus frecuencias y porcentajes respectivos en dos tablas distintas. Para actividades secuenciales, se muestran las estadísticas de los jugadores según el número de preguntas que han llegado a contestar, mostrando cuántos han llegado a ese número y la media de puntuación obtenida. En exámenes, se generan tablas con rangos de notas y frecuencia, además de un resumen por actividad con frecuencias y medias. Las consultas de las respuestas dadas por los alumnos para dicha actividad se obtienen de la base de datos `user_activity` (Tabla 5) posteriormente para obtener datos relevantes para el análisis como, por ejemplo, el texto de cada respuesta se consultarán las tablas específicas de cada tipo de actividad.

RQF 2.2 Reordenación de columnas.

Para cumplir este requisito se añaden al diseño de la tabla dos botones en cada una de las columnas reutilizando la imagen de otras tablas del sistema que tiene reordenación. Estos botones, al ser pulsados, llevarán a la recarga de la página actual y los datos tras recolectarlos en un *array* serán reordenados utilizando la función “`uasort`” de PHP (Ilustración 20). Para la reordenación de las respuestas según su texto será necesario acceder previamente a la base de datos de respuesta de dicha actividad asociando el *Id* de la respuesta con su texto.

```
uasort($responses, function ($a, $b) {  
    return $a['responded'] <=> $b['responded'];  
});
```

Ilustración 20: Función uasort

RQF 2.3 Colores para respuestas

Para el cumplimiento de este requisito se aplicará mediante el uso de los estilos de CSS un estilo en color verde a la respuesta si esta es correcta y un estilo en rojo si esta es incorrecta. Se añadirá también un *check* para indicar que es correcta y un aspa para indicar que es incorrecta, asegurando así que la información no se transmite utilizando únicamente el color, lo cual, podría plantear problemas de accesibilidad a ciertas personas.

RQF 2.4 Soporte visual respuestas con imágenes

Para el diseño de este requisito (Ilustración 21) se deben tener en cuenta los tres tipos de actividades que pueden tener imágenes en sus respuestas y las bases de datos de las que obtener está información en cada una. Estos tipos de actividades son las de tipo selección cuyos datos de respuesta están en la base de datos `radio_check_ans` (Tabla 8), las de tipo emparejar cuyos datos de respuesta se encuentran en la base de datos `pair_ans` (Tabla 15) y las de tipo ordenar cuyos datos de repuesta se encuentran en la base de datos `order_ans` (Tabla 11). Primero se comprueba si las respuestas de la actividad tienen imágenes; en caso afirmativo, se realizará la consulta de los datos de esta en la

base de datos correspondiente al tipo de respuesta; en caso contrario se mostrará solo el texto de la respuesta. Además, se debe tener en cuenta la altura de las imágenes y el texto alternativo de estas para formar correctamente el código HTML de la tabla con toda la información.

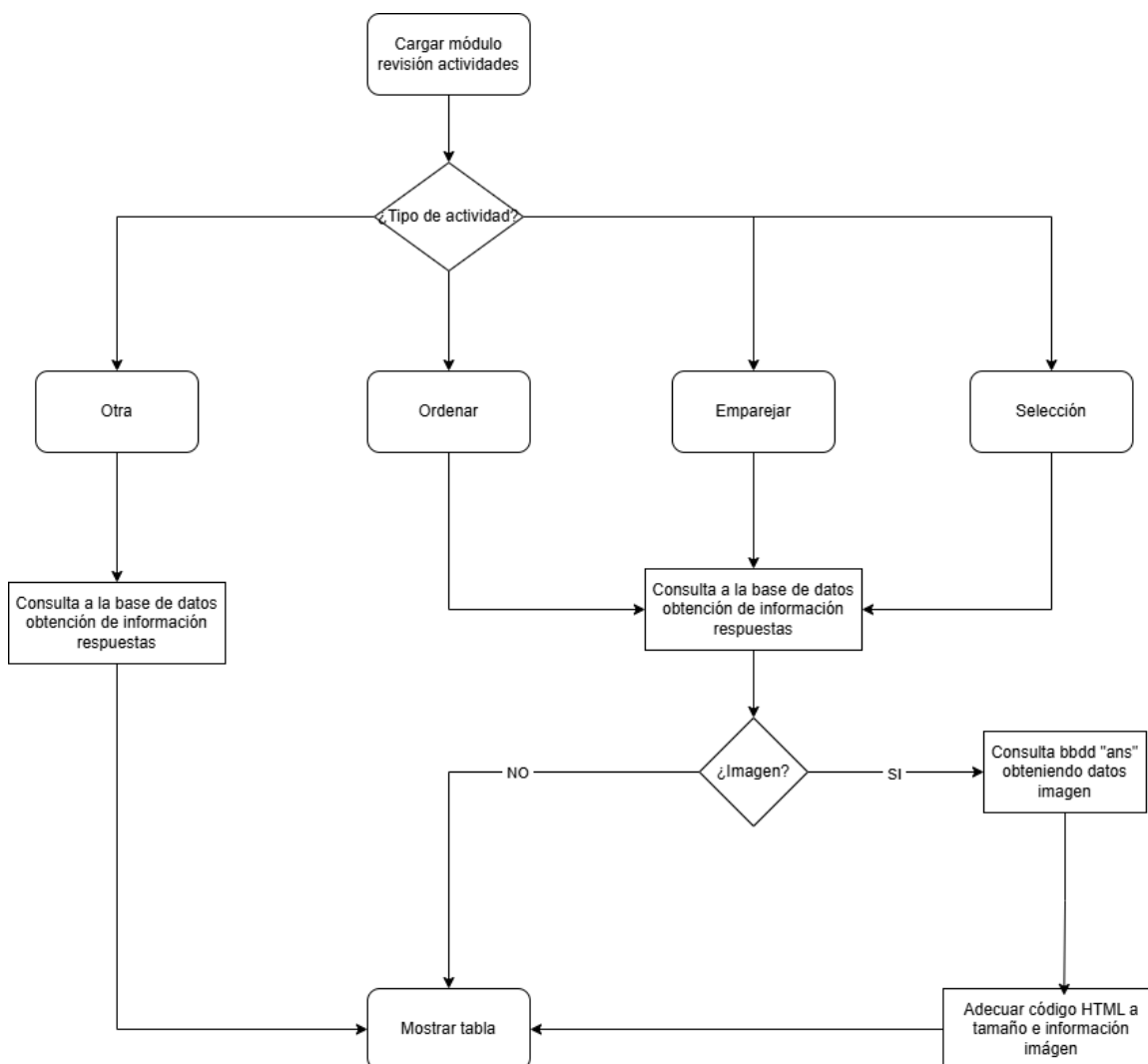


Ilustración 21: Diagrama muestra imágenes

4.1.2.3 Mantenimiento.

En esta sección se describe el diseño de los requisitos funcionales que se engloban en las tareas de mantenimiento.

RQF 3.1 Actualización automática tiempo entre pruebas y REQ 3.6 Reducción acumulativa.

Para cumplir con estos requisitos tras la realización de una prueba en cualquiera de los módulos por parte del alumno, el sistema debe automáticamente evaluar la puntuación obtenida por este, campo *score* de la tabla *grupo_hizo_prueba* (Tabla 19). En los casos en los que esta puntuación sea mayor a 0.5 el sistema comprobará si este grupo de alumnos ya tiene un tiempo reducido, consultando el campo *tiempoCustomEntrePruebas* de la tabla *grupo* (Tabla 17); si este campo fuera nulo obtendría del archivo de configuración del módulo actual el tiempo entre pruebas por defecto y la reducción se aplicaría sobre este. Finalmente, con el tiempo actualizado en

minutos se actualizará en la base de datos grupo (Tabla 17) el campo tiempoCustomEntrePruebas con el nuevo dato.

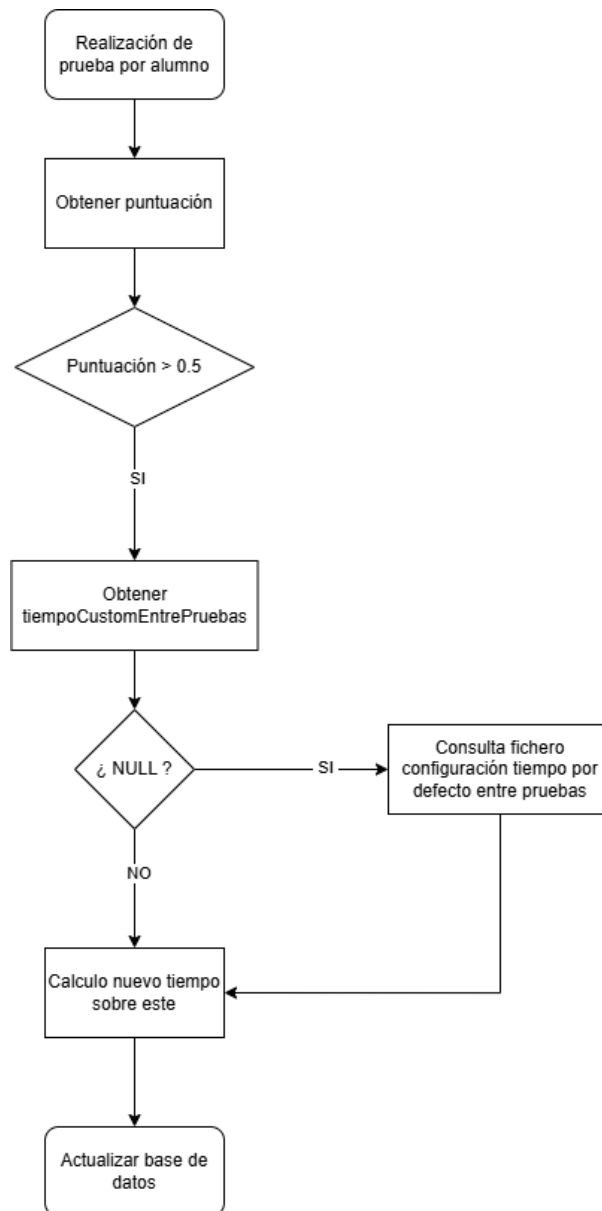


Ilustración 22: Diagrama calculo score automático

RQF 3.2 Cálculo reducción de tiempo.

Para la cumplimentación de este requisito se describe el diseño de la función de cálculo del tiempo que deberá esperar el grupo hasta la siguiente prueba tras aplicar la reducción según el score obtenido. Para un score de 1 la reducción será en un 75% del tiempo de espera anterior. Para un score de 0.5 la reducción es de un 0%. Para el cálculo de los tiempos intermedios se ha seguido la función de la Ilustración 23.

```

tiempoCustom= tiempoEntrePruebas* (1 - (1.5 * (Puntuación - 0.5)));
  
```

Ilustración 23: Función tiempo entre pruebas

RQF 3.3 Visualización próxima prueba.

Para cumplir con este requisito se debe añadir al módulo de consultas de datos de un grupo de un comprobador la fecha y hora a la que dicho grupo tendrá disponible la realización de la siguiente prueba. Para ello se debe obtener de la tabla grupo_hizo_prueba (Tabla 19) la fecha de la última prueba realizada por el grupo. De la tabla grupo (Tabla 17) se debe obtener el campo tiempoCustomEntrePruebas, el cual si es nulo se deberá obtener del fichero de configuración del módulo de prácticas actual el tiempo entre pruebas por defecto. Con esto se calcula la fecha y hora de la siguiente prueba disponible para el grupo sumando el tiempo.

RQF 3.4 Reducción manual tiempo entre pruebas y RQF 3.5 Notificación de reducción aplicada.

Para la cumplimentación de este requisito se va a añadir en el módulo de consultas de datos una opción para que el administrador pueda introducir un porcentaje entre 0 y 100 que aplicar al tiempo entre pruebas del grupo. A su lado se encontrará un botón para aplicar la reducción.

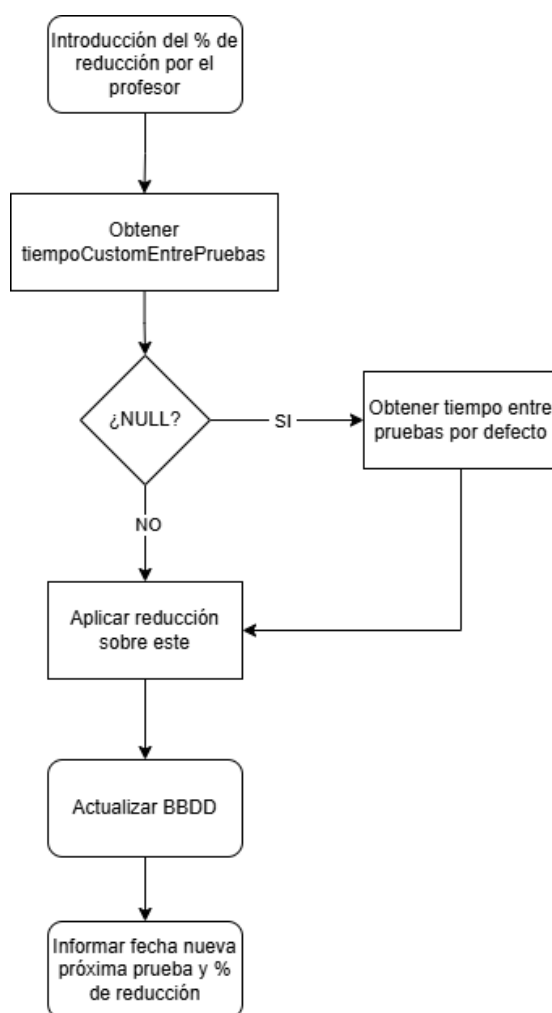


Ilustración 24: Diagrama reducción manual tiempo entre pruebas

Al pulsarlo el sistema consultará la tabla grupo (Tabla 17) y su campo tiempoCustomEntrePruebas para obtener el tiempo entre pruebas del grupo. Si este campo fuera nulo se consultará el tiempo entre pruebas por defecto del módulo en el archivo de configuración. Tras obtener el tiempo entre pruebas actual del grupo se le aplicará la reducción indicada por el profesor y se

actualizará el campo tiempoCustomEntrePruebas. Finalmente, se le mostrará una página al usuario en la que se le indicará de cuánto ha sido la reducción aplicada y la fecha a la que el grupo tendrá la siguiente prueba disponible.

RQF 3.7 Filtro por tipo de actividad.

Para cumplir con este requisito se revisará la funcionalidad dentro del módulo de configuración de actividades que permite filtrar el contenido mostrado según el tipo de actividad. Esta funcionalidad deberá consistir en un desplegable que se cargará con los tipos de actividad registrados en el sistema, los cuales están definidos actualmente como 12 tipos distintos. Al seleccionarse un valor en el filtro, el sistema actualizará automáticamente los datos visibles en la sección de configuración, limitando los resultados únicamente a las actividades que coincidan con el tipo elegido.

RQF 3.8 Filtro por medalla.

Para dar cumplimiento a este requisito se revisará el filtro actual en la sección de configuración de actividades, el cual permite seleccionar entre las diferentes medallas disponibles en el sistema. La lista de medallas que se debe ofrecer en el filtro se obtendrá mediante una consulta a la base de datos de las medallas, en las cuales se obtendrán solamente las que se obtienen al realizar una actividad y serán filtradas solo las de la asignatura activa en el sistema en el momento. Al aplicar el filtro, el sistema limitará la visualización de actividades a aquellas que permitan obtener la medalla seleccionada.

RQF 3.9, RQF 3.10 y RQF 3.11: Detección y penalización de errores por token

El sistema de corrección léxica realizará una comparación secuencial entre los *tokens* entregados por el alumno y la secuencia esperada de *tokens* creada por el sistema.

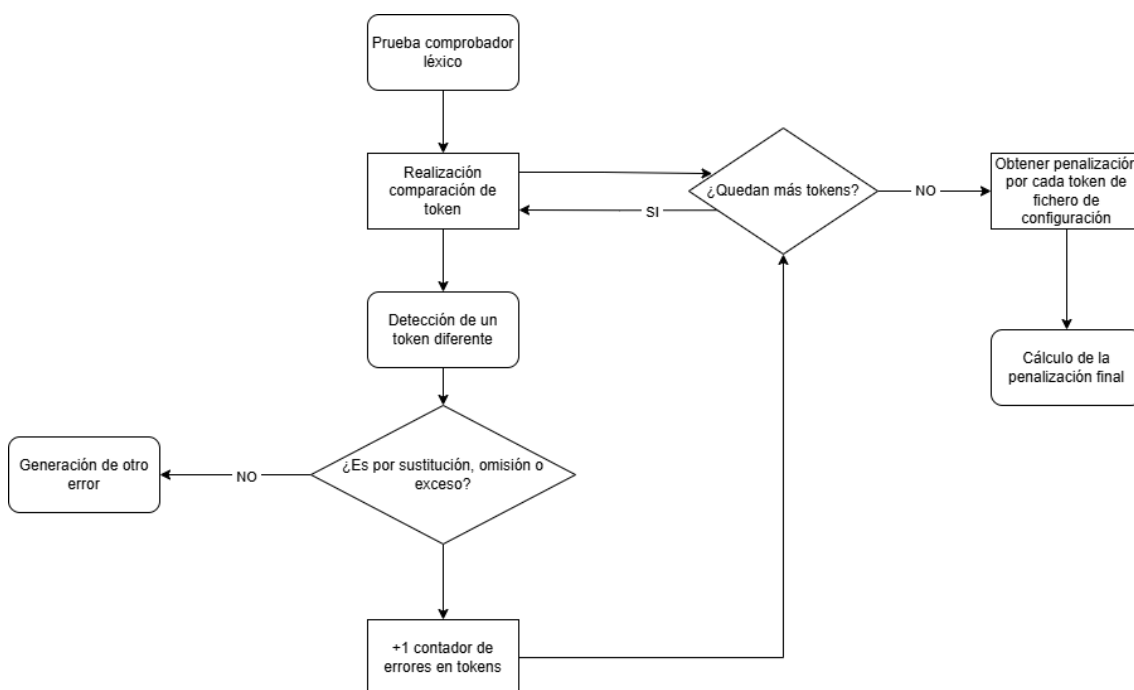


Ilustración 25: Diagrama detección y penalización por tokens

Cuando se detecte una diferencia exacta de un *token*, ya sea por omisión, exceso o sustitución, se contabilizará automáticamente como un nuevo error de tipo falta un *token*. Esta detección no detendrá el proceso de corrección, ya que el

algoritmo continuará evaluando el resto de los *tokens*. Esta estrategia de corrección continua asegura una evaluación más detallada del trabajo entregado, permitiendo registrar múltiples errores en un solo análisis. Cada *token* incorrecto reducirá la puntuación del alumno en una cantidad definida, la cual será obtenida del fichero de configuración del módulo actual. Al finalizar el proceso, la penalización total se calculará como la suma de las penalizaciones por cada error identificado, y se restará de la puntuación máxima posible.

RQF 3.12 Visualización de penalización por *token* faltante

Para cumplir este requisito, dentro del módulo de configuración de datos del comprobador se mostrará una sección específica en la que se visualizará el valor actual de la penalización aplicada por cada *token* faltante. Este valor se extraerá automáticamente del fichero de configuración asociado al módulo léxico y se cargará al acceder a dicha sección.

RQF 3.13 Configuración nueva penalización

Para atender este requisito se habilitará, dentro del mismo módulo de configuración de datos del comprobador léxico, un campo de entrada donde el administrador podrá introducir un nuevo valor numérico correspondiente a la penalización por la falta de un *token*. Una vez introducido el nuevo valor y confirmado mediante un botón de guardado del cuestionario de modificación de datos, el sistema actualizará el fichero de configuración del módulo con este nuevo dato.

4.2 Implementación

Para la codificación se han utilizado las tecnologías ya presentes en el sistema DRACO, las cuales son PHP para la generación de la lógica del sistema, HTML5 y CSS3 para dar contenido y estructura a las páginas y dar estilo y MySQL para las bases de datos.

Para el desarrollo se han hecho uso de WampServer [26], una plataforma que genera un entorno local en Windows que proporciona un servidor web Apache, un gestor de bases de datos MySQL y PHP. Para la edición y depuración del código se ha utilizado Visual Studio Code como IDE con soporte para múltiples tecnologías. Finalmente, para visualizar el sistema web se ha utilizado el explorador Opera principalmente.

En esta sección se describen algunos de los aspectos más destacados de la implementación de los requisitos especificados.

4.2.1 Prácticas

Para la selección automática y aleatoria de una plantilla se ha modificado el método de comprobación de opciones y el almacenamiento en las bases de datos de estas. Las opciones correspondientes a un fichero se almacenaban en la tabla *opcionesfichero* utilizando la función *htmlentities* [27] la cual devuelve el texto codificando todos los caracteres aplicables a entidades HTML, tal y como se muestra en el ejemplo de la Ilustración 26. Las opciones correspondientes a un alumno y las opciones de los *tokens* se almacenaban en UTF-8 en sus respectivas bases de datos, igual que el texto mostrado en la Ilustración 26. Se decidió modificar la introducción de los nombres de las opciones en la base de

HTML: Con comillas dobles (" ") UTF-8: Con comillas dobles (" ")

Ilustración 26: Texto en HTML y UTF8

datos opcionesfichero para que se realice de una manera similar a las otras siendo los datos introducidos en UTF-8.

Tras esta modificación se debió adaptar el algoritmo de comprobación de plantillas ya que la búsqueda de las opciones de los ficheros estaba hecha con los nombres de las opciones codificadas en HTML por lo que se eliminó las funciones `htmlentities` antes de comparar y se cambió los caracteres de separación de opciones, que estaban codificados por su versión sin codificar.

Por otro lado, para el algoritmo de comprobación de plantillas se detectó un fallo en la comprobación al aparecer un salto de línea al generar un bloque. El lenguaje fuente sobre el que se está trabajando no consideraba el salto de línea como un *token*, sin embargo, el algoritmo de comprobación de plantillas sí lo hacía siempre. Se ha modificado el algoritmo para que consulte en la base de datos, en la tabla *tokens*, si el lenguaje sobre el que se está trabajando genera el salto de línea como un *token*. En caso de ser así, el comprobador genera el *token* de salto de línea. En caso contrario, se avanza el índice para apuntar al siguiente *token* y se continúa con la comprobación, no generándose el salto de línea como un *token*.

4.2.2 Estadísticas de actividades.

Se han implementado todas las tablas estadísticas descritas en la sección de diseño. A continuación, se presentan algunos ejemplos de cómo ha quedado su interfaz:

- Tabla de una actividad de tipo selección: Se muestran todas las respuestas posibles para dicha actividad (Ilustración 27), las estadísticas mostradas son las veces que ha sido seleccionada, las veces que ha sido enseñada al alumno y el porcentaje de estas. Estos datos son tomados de la tabla *radio_check_type* mientras que si esta actividad contiene una imagen como respuesta esta se toma de la base de datos *radio_check_ans* para ser mostradas.

INFORMACIÓN DETALLADA DE LA ACTIVIDAD			
Respuesta ▲ ▼	Veces respondida ▲ ▼	Veces enseñada ▲ ▼	Porcentaje ▲ ▼
✓ Si, quiero recibir los puntos de regalo	160	165	96.97%
✓ Por supuesto que quiero recibir el regalo	158	165	95.76%
✗ No, prefiero ganarme los puntos con mi esfuerzo y mis conocimientos	10	231	4.33%
✗ No, gracias	1	206	0.49%
✗ No quiero los puntos	1	223	0.45%

Ilustración 27: Tabla actividad de tipo selección

- Tabla de una actividad de tipo ordenar: Se muestran todas las respuestas posibles para dicha actividad (Ilustración 28), las estadísticas mostradas son las veces que ha sido ordenada correctamente, las veces que ha sido enseñada al alumno y el porcentaje de estas. Estos datos son tomados de la tabla *order_type* mientras que si esta actividad contiene una imagen como respuesta esta se toma de la base de datos *radio_order_ans* para ser mostradas.
- Tablas de una actividad de tipo emparejar: Para este tipo de actividad se mostrarán dos tablas, la primera (Ilustración 29) en la que se muestran las parejas correctas, las veces que han sido mostradas y las que han sido emparejadas correctamente. Mientras que en la segunda (Ilustración 30) se muestran las parejas incorrectas que han realizado

los alumnos y cuántas veces se ha realizado dicha pareja. Los datos de ambas tablas son tomados de la tabla pair_ans de la base de datos.

INFORMACIÓN DETALLADA DE LA ACTIVIDAD			
Respuesta ▲ ▼	Veces ordenada correctamente ▲ ▼	Veces mostrada ▲ ▼	Porcentaje ▲ ▼
✓ Análisis Léxico	250	259	96.53%
✓ Análisis Sintáctico	228	252	90.48%
✓ Generación de Código Intermedio	220	251	87.65%
✓ Análisis Semántico	215	249	86.35%
✓ Optimizador de Código Objeto	111	248	44.76%
✓ Optimizador de Código Intermedio	59	244	24.18%
✓ Generación de Código Final	24	237	10.13%

Ilustración 28: Tabla actividad tipo ordenar

INFORMACIÓN DETALLADA DE LA ACTIVIDAD				
Respuestas correctas:				
Par1 ▲ ▼	Par2 ▲ ▼	Veces respondida ▲ ▼	Veces enseñada ▲ ▼	Porcentaje ▲ ▼
✓ <	<MENOR, ->	58	58	100.00%
✓ >	<MAYOR, ->	58	58	100.00%
✓ if	<IF, >	59	59	100.00%
✓ ++	<AUTOINCREMENTO, ->	52	52	100.00%
✓ "Hola"	<CADENA, 'Hola'>	59	59	100.00%
✓ ==	<ES_IGUAL, ->	66	66	100.00%
✓ 55.5	<REAL, 55.5>	68	69	98.55%
✓ function	<FUNCTION, ->	67	68	98.53%
✓ (	<PARÉNT_IZQ, ->	64	65	98.46%
✓ +=	<ASIGNA_SUMA, ->	61	62	98.39%
✓ +	<MÁS, ->	61	62	98.39%
✓ !=	<DISTINTO, ->	57	58	98.28%
✓ !	<NOT, ->	56	57	98.25%

Ilustración 29: Tabla respuestas correctas actividad emparejar

Respuestas incorrectas dadas:		
Par1 ▲ ▼	Par2 ▲ ▼	Veces respondida ▲ ▼
X {	<CORCHETE_DER, ->	2
X int	<ID, 55>	2
X]	<LLAVE_IZQ, ->	2
X }	<LLAVE_IZQ, ->	2
X }	<CORCHETE_IZQ, ->	2
X contador	<INT, >	2
X {	<LLAVE_DER, ->	2
X ;	No genera ningún token	1
X ;	<ID, 55>	1
X <=	<LLAVE_DER, ->	1
X 55.5	<ID, 55>	1
X [	No genera ningún token	1

Ilustración 30: Tabla respuestas incorrectas actividad emparejar

- Tabla de una actividad de tipo secuencial: Para este tipo de actividad se muestra cuántos alumnos han llegado a responder cada número de

respuestas posibles y la nota media de estos en forma de porcentaje (Ilustración 31). Estos datos son tomados de la tabla sequential_type.

INFORMACION DETALLADA DE LA ACTIVIDAD		
Respuestas correctas:		
Número de respuestas ▲ ▼	Número de personas ▲ ▼	Puntuación media ▲ ▼
✓ 5	84	93.13%
✓ 4	29	79.31%
✓ 3	25	70.69%
✓ 2	36	54.80%
✓ 1	72	38.36%

Ilustración 31: Tabla actividad de tipo secuencial

- Tablas de una actividad de tipo examen: En este tipo de actividad de muestran dos tablas, la primera (Ilustración 32) se muestra el número de personas cuya nota ha quedado dentro de cada intervalo y cuántas representan en porcentaje respecto al total. En la segunda (Ilustración 33) se muestra el nombre de cada pregunta, cuántas veces ha sido respondida esta y la nota media obtenida por los alumnos en dicha pregunta.

INFORMACIÓN DETALLADA DE LA ACTIVIDAD		
Nota media de los alumnos:		
Intervalo	Número de personas	Porcentaje
0 - 3	7	4.76%
3 - 5	19	12.93%
5 - 7	48	32.65%
7 - 9	53	36.05%
9 - 10	20	13.61%

Ilustración 32: Tabla general actividad tipo examen

Estadísticas por actividad		
Nombre ▲ ▼	Número de veces respondida ▲ ▼	Nota media ▲ ▼
Comentarios	12	100.00%
Termina la frase	12	100.00%
Lexema-token	16	100.00%
Diseño de tokens	15	100.00%
Orden de tokens	14	100.00%
Token	17	94.12%
Tokens empresa - 1	14	92.86%
Tokens empresa - 7	20	90.00%
Constante real	9	88.89%
Autómata del Léxico	18	88.89%
Gramática léxica	17	88.24%
Tokens empresa - 4	14	85.71%
Reconocimiento-1	17	85.59%

Ilustración 33: Tabla por pregunta, actividad tipo examen

- Tabla de una actividad de tipo respuesta corta (Ilustración 34): En esta tabla se muestra la respuesta correcta que más veces ha sido respondida junto con el total de veces que se ha respondido correctamente la actividad y cada una de las respuestas erróneas que han introducido los alumnos junto con cuántas veces esta respuesta ha sido dada.

INFORMACIÓN DETALLADA DE LA ACTIVIDAD	
Respuesta ▲ ▼	Veces respondida ▲ ▼
✓ lexico	280
✗ sintactico	2
✗ asdasd'"><s>asd</s>	1
✗ semantico	1

Ilustración 34: Tabla actividad de tipo respuesta corta

Para el diseño de las tablas se ha definido al comienzo del archivo PHP un estilo propio para ellas en CSS, mostrado en la Ilustración 35, el cual establece aspectos como el tamaño de la fuente, la distancia de esta al borde de la tabla o el color y tamaño de los bordes de las tablas entre otros.

```
<style>
    .activity-table {
        border-collapse: collapse;
        width: 100%;
    }
    .activity-table th,
    .activity-table td {
        border: 1px solid #ccc;
        padding: 0.5em 1em;
        text-align: left;
    }
    .activity-table th {
        background-color: #f8f8f8;
        font-weight: bold;
    }
    .activity-table tbody tr:nth-child(even) {
        background-color: #f9f9f9;
    }
    .activity-table tbody tr:nth-child(odd) {
        background-color: #f3f3f3;
    }
</style>
```

Ilustración 35: Estilo CSS

Para el código HTML de las tablas, como el ejemplo de la Ilustración 36, se construye utilizando la etiqueta <table>, que define el contenedor principal. Dentro de ella, se utiliza la etiqueta <thead> para la cabecera, <tbody> para el cuerpo y <tr> para definir cada fila. Las celdas de encabezado se declaran con <th>, mientras que las celdas de datos usan <td>.

Además, cada columna de la cabecera incluye botones con iconos en forma de flechas que permiten ordenar los datos por diferentes criterios, como respuesta, número de veces respondida o porcentaje, mediante formularios que se activan al hacer clic en dichos botones.

```

<table class="activity-table">
  <thead>
    <tr>
      <th>
        <label for="respuesta">Respuesta </label>
      </th>
      <th>
        <label for="vecesRespondida">Veces respondida
        </label>
      </th>
    </tr>
  </thead>
  <tbody>
    <tr>
      <td style='color: green;'>✓ léxico </td>
      <td> 280 </td>
    </tr>
    <tr>
      <td style='color: red;'>X sintactico </td>
      <td> 2 </td>
    </tr>
  </tbody>
</table>

```

Ilustración 36: Código tabla HTML

4.2.3 Mantenimiento

Para la automatización de la reducción de tiempo tras la realización de una prueba en cualquiera de los cuatro módulos de prácticas se ha implementado una función PHP en un fichero común para así evitar el código repetido. Esta función recibe por parámetros el Id del grupo, la puntuación que ha obtenido, la base de datos activa y el nombre del módulo activo. Con estos datos obtiene el tiempo entre pruebas por defecto del fichero de configuración correspondiente y cuando el score es mayor que 0.5 utilizando la función diseñada en el apartado de diseño calcula el tiempo hasta la siguiente prueba y lo actualiza en la base de datos.

Para la actualización manual por parte de los administradores se implementó en el módulo de datos del grupo las estadísticas y formulario descritos el apartado de diseño, tal y como se muestra en la Ilustración 37.

DATOS GRUPO

Grupo número: 333

Intentos restantes: 32

Fecha de la última prueba: 07/05/2025 23:22

Fecha de la siguiente prueba: 07/05/2025 23:23

Añadir el siguiente número de intentos

Reducir el tiempo en el siguiente porcentaje

Ilustración 37: Módulo datos grupo

Además, tras la introducción de un porcentaje de reducción por parte del administrador, tal y como se describe en el apartado de diseño, el sistema informa de la fecha de la próxima prueba para el grupo y porcentaje de reducción que se ha aplicado tal y como se muestra en la Ilustración 38.

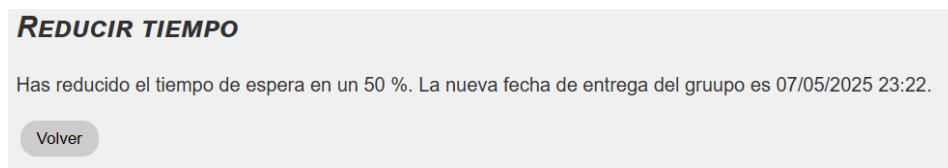


Ilustración 38: Información tras una reducción de tiempo manual

Para los filtros por medalla y tipo de actividad se han corregido los errores que había para que sigan las especificaciones de los requisitos y lo diseñado en el apartado correspondiente. Los resultados son mostrados en la Ilustración 39 y la Ilustración 40.

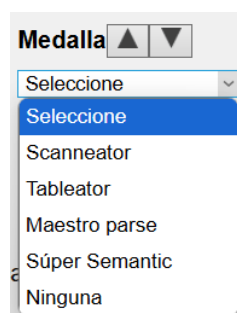


Ilustración 39: Filtro por medallas.

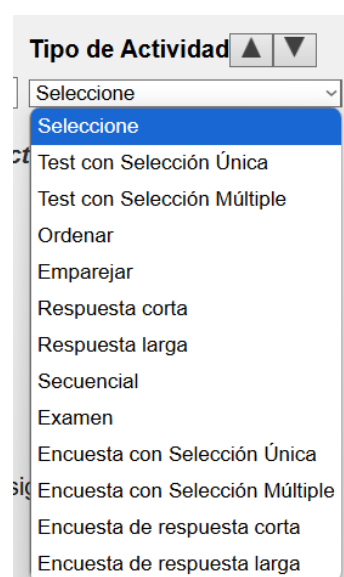


Ilustración 40: Filtro por tipo de actividad

Para la detección de errores en la corrección de *tokens* se ha implementado una función que detecta los tres tipos de errores que pueden existir tras encontrar un *token* diferente: que se haya omitido un *token* por el alumno, que haya un *token* extra en sus *tokens* o que haya un *token* diferente. Esta función se ha implementado de la siguiente manera: Cuando encuentra un *token* diferente lo primero que hace es comparar este *token* con el próximo de la lista de *tokens* correctos. Si estos son iguales, se considera que está en el caso de error de omisión de un *token*, se añade 1 al contador de errores de *tokens*, se avanza la lista de *tokens* correctos para igualarlo y se continúa con la ejecución.

En caso contrario, se activa una variable lógica que indica que el anterior *token* ha sido erróneo y se procede a avanzar el puntero de la lista de *tokens* de los alumnos y el de la lista del profesor. Ahora, dado que la variable lógica está activada, la comprobación que se hace es la siguiente: si el *token* actual del alumno es igual al *token* actual de la lista, será el caso en el que solamente los *tokens* anteriores eran diferentes y por tanto en la especificación del alumno hay un único *token* erróneo, se suma 1 al contador de *token* erróneos y se continúa con la comprobación. Si estos *tokens* no son iguales, se comprueba si el *token* actual del alumno es igual al *token* anterior en la lista de *tokens* correctos, si esto es así se está en el caso de que el alumno a incluido un *token*

extra en su definición de *tokens*, se sumaría 1 al contador de *tokens* erróneos, avanzaría en 1 el puntero de la lista de *tokens* del alumno, pero no se avanzaría en la lista de los profesores para así igualarlos. En caso de que ninguna de esta situación se cumpliera el sistema determina que hay más de un *token* erróneo consecutivo y finaliza con la ejecución de la prueba.

Además, para cumplir con el diseño se ha añadido al módulo de configuración de datos el campo penalización por falta de un *token*, para permitir al administrador modificar la puntuación de penalización por cada *token* erróneo tal y como se muestra en la Ilustración 41.

CONFIGURAR DATOS DEL COMPROBADOR

Configuración de los tiempos correspondientes a pruebas.

Tiempo para plantilla fácil 777

Tiempo para plantilla intermedia 888

Tiempo para plantilla difícil 999

Tiempo límite de cálculo 555555

Tiempo entre pruebas 1

Puntuaciones y penalizaciones otorgadas a los alumnos por conseguir determinadas metas en cada prueba.

Puntos por análisis correcto 60

Puntos por participar 3

Penalización por atributo incorrecto 3

Penalización por no generar adecuadamente tokens de saltos de línea 0,5

Penalización por no generar el token de fin de fichero 0,11

Penalización por la falta de un token 10

Puntos por detectar un error 10

Puntos por indicar la línea del error 20

Porcentaje de los puntos por indicar el tipo del error si hay parámetros 10 %

Ilustración 41: Configuración datos del comprobador

4.3 Pruebas.

Durante el desarrollo del trabajo se han realizado pruebas con diferentes objetivos, el primero encontrar el porqué de fallos detectados en los funcionamientos actuales del sistema y el segundo, asegurar que se cumplen los requisitos y el correcto funcionamiento de lo implementado.

4.3.1 Entorno de pruebas

Para la realización de las pruebas se ha utilizado la opción de depuración de la extensión PHP Debug de Xdebug (Ilustración 42). Esta herramienta permite establecer puntos de ruptura y condiciones para estos, ejecutar las instrucciones paso a paso, la posibilidad de ver las variables siendo estas accesibles en tiempo de ejecución junto con los valores almacenador o la posibilidad de entrar y salir de una función. Esta herramienta ha sido de gran ayuda para la comprensión del código en primera instancia y la realización de análisis específicos y la realización de las pruebas de regresión, ayudando en la comprensión de lo que estaba sucediendo en las funciones.

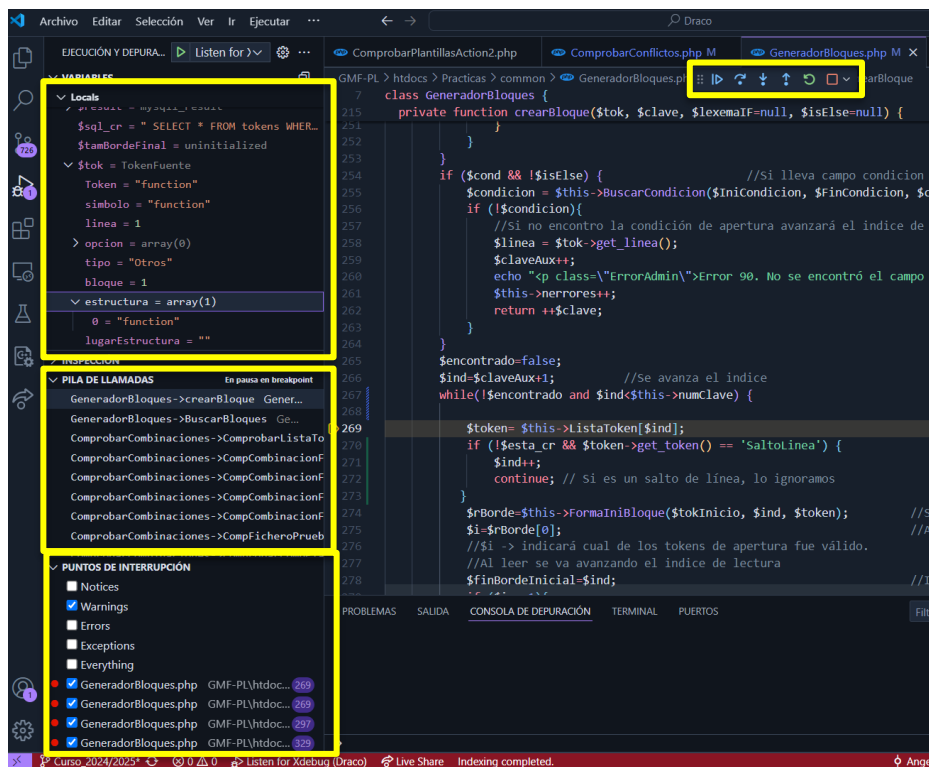


Ilustración 42: Herramienta Xdebug en depuración.

Además, se ha tenido en consideración el fichero de errores de la herramienta WampServer “PHP error log” (Ilustración 43), en el cual se recogen los errores y advertencias que aparecen durante la ejecución del sistema. Gracias a esto se podían detectar errores que de forma visual no se puede.

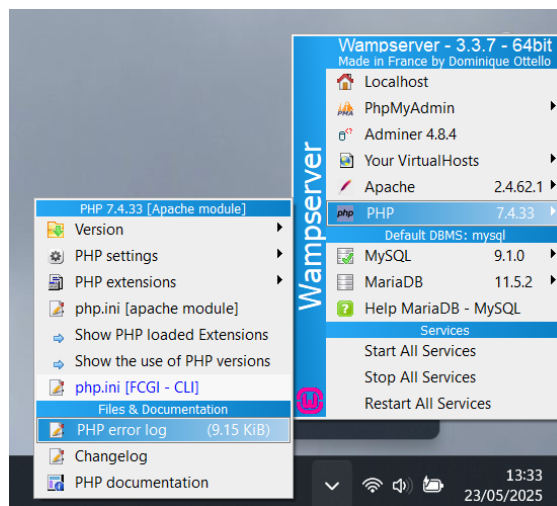


Ilustración 43: PHP error log

4.3.2 Tipos de pruebas

Durante el desarrollo del trabajo se llevaron a cabo distintos tipos de pruebas con el objetivo de garantizar el correcto funcionamiento del software, detectar errores en fases tempranas del desarrollo y asegurar la estabilidad del sistema tras la incorporación de nuevas funcionalidades, además de detectar errores en los funcionamientos actuales. A continuación, se detallan los tipos de pruebas más relevantes aplicadas durante el proceso de desarrollo.

Pruebas unitarias:

Este tipo de pruebas consisten en la evaluación de unidades particulares de código como funciones, métodos o clases, de forma aislada para verificar su correcto funcionamiento.

Este tipo de pruebas se han utilizado principalmente para comprobar que cada una de las nuevas funciones introducidas, como la de reducción del tiempo según la puntuación obtenida por el alumno, funcionan correctamente. Primero, se calculaba manualmente según la puntuación obtenida el tiempo en minutos que debería tener que esperar el alumno hasta la realización de la siguiente comprobación de su práctica. Tras esto, se comprobaba que el valor del campo `tiempoCuustomEntrePruebas` de la tabla grupo coincidiera con esto. Gracias a estas pruebas se detectó que al realizar una prueba en la cual el fichero contenía un error esta funcionalidad no se aplicaba ya que la traza de ejecución iba por otros ficheros. Para resolver este error se añadió la reducción del tiempo tras la comprobación del error detectado por el alumno.

Por otro lado, se han realizado este tipo de pruebas para comprobar el correcto funcionamiento del comprobador léxico tras añadir los nuevos errores por falta de un *token*. Se ha probado esta funcionalidad con ficheros de *tokens* totalmente correctos, con otros errores que detectaba el comprobador para asegurarse de que todo sigue funcionando como debe y se han realizado múltiples pruebas con los 3 tipos de errores a los que se puede deber el encontrar un único *token* diferente (la omisión de un *token*, la introducción de un *token* extra o el error en un *token*), combinando varios de estos fallos en un mismo fichero y comprobando que la reducción de los puntos obtenidos por el alumno fuera acorde al número de errores encontrados.

Para el módulo de estadísticas se realizaron también este tipo de pruebas, comprobando manualmente los datos desde la base de datos y calculando las estadísticas y posteriormente obteniendo las tablas de dicha actividad y comprobando la veracidad de los datos. En estas pruebas, gracias al fichero de errores de WamppServer se han podido detectar distintos errores relacionados con el trato de las listas y la consulta de índices sin inicializar en estas o índices fuera de límites. Estos errores se fueron corrigiendo durante el desarrollo del código. Además, se ha realizado para cada tipo de actividad una prueba con una actividad con solo una respuesta para comprobar la correcta inicialización de los datos y que ninguna celda apareciese vacía sino con un 0.

Pruebas de regresión:

Este tipo de pruebas tienen como objetivo asegurar que las correcciones y cambios realizados en el sistema no han introducido nuevos errores.

Este tipo de pruebas han tenido una gran importancia en el desarrollo de las tareas de mantenimiento y en las modificaciones de la selección de fichero de prueba para el alumno y la comprobación de plantillas. Esto es debido a que se tenía que evaluar que los cambios realizados no supusieran ninguna incompatibilidad en el sistema que generase errores nuevos.

Mediante la utilización de este tipo de pruebas se pudo encontrar un error en el código implementado para resolver el requisito 1.5. Tras la modificación de la base de datos `opcionesfichero` y su campo `opción` que contiene el nombre de la opción disponible del fichero el fallo encontrado fue que la comprobación de las plantillas por el sistema en el módulo del administrador detectaba errores no existentes debido a la modificación del texto del nombre de la opción que antes estaba codificado mediante la función `htmlentities` y ahora estaba en UTF8. Este

error se solventó cambiando la codificación en la comprobación para que esta se realizase con ambos textos en UTF8 y se ajustó la búsqueda de caracteres a la nueva codificación del texto.

Tras la modificación del campo tiempoCustomEntrePruebas por parte de la función de reducción del tiempo entre pruebas de tuvieron que realizar pruebas con la opción del sistema que permite intercambiar monedas del sistema (DraCOINS) por una reducción del tiempo de espera entre prácticas y que ambas reducciones fueran aplicadas correctamente. Todas estas pruebas fueron exitosas concluyendo en el buen funcionamiento del sistema.

Por otro lado, tras la modificación del algoritmo de comprobación de las plantillas se han realizado pruebas tanto con plantillas que previamente pasaban las pruebas como con plantillas que anteriormente eran detectadas como erróneas, cuando no era así para garantizar que la comprobación se sigue realizando correctamente. Además, debido a la cercanía de la comprobación de plantillas con la comprobación del fichero introducido por el alumno se han realizado diferentes pruebas en todos los comprobadores para garantizar su correcto funcionamiento tanto a la hora de seleccionar las plantillas como de corregir y evaluar el fichero resultado introducido por el alumno al sistema.

Pruebas de depuración:

Este tipo de pruebas se aplicaban en funcionalidades del sistema que no se comportan de manera esperada. A diferencia de otras pruebas planificadas, las pruebas de depuración no parten de una hipótesis de correcto funcionamiento, sino del conocimiento previo de que algo está fallando. El objetivo principal es analizar el código y el flujo de ejecución para identificar el origen del error. Estas pruebas implicaron, en muchos casos, ejecutar la funcionalidad sospechosa bajo distintas condiciones, revisar los datos obtenidos desde la base de datos y emplear las herramientas de depuración para observar el estado interno del programa en tiempo real, los valores de las variables y la traza de ejecución de estos.

Durante el desarrollo del trabajo este tipo de pruebas resultaron especialmente útiles para la resolución de los requisitos del apartado 1.

En los primeros, la validación de plantillas se había detectado que tras la introducción de una plantilla aparentemente correcta el sistema detectaba errores con los *tokens* que forman bloques y los que deben aparecer a continuación o dentro de estos ya que en algunas de estas plantillas se detectaban errores no existentes. Tras la realización de un grupo de pruebas se detectó que el fallo tenía que ver con el cierre temprano del bloque, al encontrarse una estructura de bloque que contiene otra, como por ejemplo un bucle *while* dentro de una función, el *token* que cerraba el bloque interno lo detectaba como cierre del bloque externo, llevando esto a los errores que detectaba. Tras la realización de un segundo grupo de pruebas más específico dado que se conocía ya que el error tenía relación con la generación de bloques se encontró que en el primer bloque el sistema detectaba su inicio y su final en el salto de línea que existía entre la condición y el *token* de apertura del bloque (“{“). Esto era debido a que el salto de línea era detectado y generaba un *token*, cuando en las especificaciones del lenguaje se establecía que el salto de línea no debía generar un *token*. Para la confirmación de este fallo, se realizó una prueba del mismo fichero sin el salto de línea entre la condición y la apertura del bloque, la cual finalizó con la comprobación correcta de la plantilla, corroborando así la hipótesis de que el fallo era producido por el *token* de salto de línea. Para la corrección de este error, se introdujo en el algoritmo de

comprobación de plantillas, primero una consulta a la base de datos para conocer si el lenguaje en uso genera un *token* con el salto de línea o no. Y después, en la generación de *tokens*, en caso de que el salto de línea genere *token* sí será tratado como esto; en caso contrario, el salto de línea será obviado y se generará el siguiente *token*.

En el segundo grupo de requisitos, los que comprobaban la aleatoriedad de la selección de plantillas por parte del sistema para la realización de las pruebas de los alumnos se partía de la hipótesis de que se entregaban siempre las mismas plantillas, aunque con combinaciones diferentes y había otras que nunca eran seleccionadas por el sistema. Tras la realización de pruebas unitarias y la extracción de estadísticas se llegó a la conclusión de que esta hipótesis era correcta y se procedió al análisis de la funcionalidad. Se detectó que en la comparación entre las opciones del fichero y las opciones del alumno los nombres de las opciones de ambos casos estaban escritos de diferente forma, es decir con diferente codificación, lo que llevaba a que dos opciones que debían ser detectadas como la misma no lo fueran. Esto se arregló modificando la inserción de los nombres de las opciones de los ficheros en la base de datos, igualando la codificación de ambas listas.

5 Conclusiones

Durante el desarrollo de este TFG, se han realizado modificaciones y mejoras en las funcionalidades ya existentes del sistema web DRACO. Estas modificaciones están divididas entre el módulo de estadísticas de las actividades, las mejoras en los sistemas de selección y comprobación de las plantillas y distintas implementaciones que tienen que ver con el mantenimiento del sistema.

La implementación realizada para la inclusión de un módulo de estadísticas generales asociadas a cada actividad permitirá a los profesores de la asignatura conocer el desempeño general de los alumnos al realizar las actividades.

Este nuevo módulo proporciona a los profesores una herramienta para evaluar de manera más objetiva y detallada el desempeño global del alumnado en las actividades. Gracias a la recopilación y análisis de los datos, los docentes podrán conocer aspectos clave como la frecuencia con la que los alumnos aciertan o fallan en determinadas actividades o la frecuencia con la que una respuesta es seleccionada. Además, la capacidad para reorganizar los datos obtenidos permitirá un análisis más fino posibilitando encontrar patrones comunes, detectar errores recurrentes o identificar las dificultades que experimentan los alumnos.

Esta información, no solo contribuye a una evaluación más completa por parte del profesorado, sino que también puede resultar fundamental para una toma de decisiones pedagógicas más fundamentada. Basándose en estas estadísticas, los docentes pueden adaptar sus estrategias de enseñanza, dedicar más tiempo a aquellos contenidos que hayan generado una mayor confusión y reforzar conceptos clave en los que los alumnos hayan presentado más dificultades. Todo esto supone una ventaja importante para los alumnos. Los cuales se beneficiarán de una enseñanza más personalizada, ajustada a sus necesidades reales y centrada en áreas en las que requieran un refuerzo.

En consecuencia, se favorece un aprendizaje más efectivo y centrado en los conceptos en los que se presentan más dificultades por parte de los alumnos. En conclusión, este módulo no solo mejora el control y seguimiento del desempeño de los alumnos en las actividades, sino que también actúa como un facilitador para la mejora del proceso enseñanza-aprendizaje.

Las correcciones realizadas en el sistema de selección y comprobación de plantillas han supuesto una mejora significativa en la fiabilidad y funcionalidad del entorno de pruebas disponible para los alumnos. En concreto, se ha solucionado un error presente en el módulo de validación de plantillas que provocaba la detección errónea de determinadas plantillas como inválidas debido a un tratamiento incorrecto de los *tokens* que generan bloques. Este problema impedía el uso de plantillas perfectamente válidas, limitando así las posibilidades del profesor a la hora de generar casos de prueba y del alumno a la hora de realizar comprobaciones con distintos casos.

Asimismo, se ha corregido un fallo relacionado con la lógica de la selección de plantillas durante de la fase de preparación de una prueba por parte del alumno. Dicho fallo tenía su origen en una comparación defectuosa entre cadenas de texto lo que impedía al sistema detectar correctamente si la plantilla cumplía los criterios de opciones de acuerdo con las del grupo del alumno. Como

resultado, el sistema no ofrecía al alumno todas las opciones viables, reduciendo el abanico de plantillas accesibles para la realización de sus pruebas.

La resolución de estos errores aporta muchos beneficios al proceso de comprobación de su práctica por parte del alumno. En primer lugar, se incrementa considerablemente el número de plantillas a las que puede acceder, lo que favorece a una mayor diversidad en los casos de prueba. Esta variedad amplía el espectro de situaciones que el alumno puede simular, permitiéndole reconocer errores menos evidentes y llevar a cabo una validación más profunda de su comprobador. En consecuencia, el alumno se enfrenta a un conjunto más amplio y realista de posibles entradas, lo que mejora tanto su capacidad de análisis como la robustez de sus soluciones.

Por otro lado, las mejoras introducidas en el ámbito del mantenimiento del sistema también representan un avance significativo tanto en términos de usabilidad como de eficiencia. En primer lugar, la implementación de un mecanismo de reducción del tiempo de espera entre pruebas, en función de la puntuación obtenida por el alumno en la prueba anterior constituye una optimización relevante. Este enfoque permite adaptar la experiencia de uso a la progresión del estudiante: aquellos cuyo comprobador funciona de manera más eficaz obteniendo mayores puntuaciones pueden acceder a nuevas pruebas con mayor rapidez, lo que fomenta un aprendizaje continuo y evita interrupciones innecesarias, mientras que los alumnos cuyo comprobador funcione de peor manera, disponen de un margen de tiempo más amplio para implementar los cambios y mejoras necesarios en este antes de intentar otra comprobación.

En segundo lugar, se ha mejorado sustancialmente el comportamiento del comprobador léxico ante la detección de *tokens* erróneos durante la comprobación de la prueba introducida por el alumno. Anteriormente, la aparición de un *token* erróneo suponía la interrupción inmediata del proceso de corrección y la asignación automática de la puntuación mínima al alumno. Con la nueva implementación, el sistema es capaz de identificar y manejar de forma más inteligente tres casos concretos de errores comunes: la presencia de un único *token* incorrecto, la existencia de un *token* adicional o la omisión de un *token* necesario. En lugar de detenerse, el sistema continúa la ejecución del análisis y aplica una penalización proporcional al número y tipo de errores encontrados. Esta mejora no solo permite una evaluación más justa y precisa del trabajo del alumno, sino que también facilita la detección de errores específicos, proporcionando una retroalimentación mucho más útil para el proceso de aprendizaje.

Finalmente, cabe destacar la incorporación de una serie de pequeñas mejoras individuales de menor escala, pero de gran valor acumulativo. Estas modificaciones, centradas en aspectos como la mejora de los filtros de las actividades o mejoras visuales en la interfaz contribuyen de manera notable a que el sistema resulte más accesible y cómodo de utilizar tanto para alumno como para los docentes. En conjunto, todas estas mejoras técnicas y funcionales refuerzan la mantenibilidad del sistema a largo plazo y garantizan una experiencia de usuario más fluida y satisfactoria.

La realización de este TFG personalmente ha supuesto una experiencia muy enriquecedora tanto a nivel académico como personal. Este proyecto me ha permitido aplicar y consolidar muchos de los conocimientos adquiridos a lo largo de la carrera, enfrentándome por primera vez a un desarrollo completo

desde el análisis de requisitos hasta la implementación y mejora de funcionalidades en un sistema real. Además, ha sido una oportunidad para enfrentarme a situaciones similares a las que pueden surgir en un entorno profesional, lo cual me ha ayudado a desarrollar una mayor autonomía, capacidad de análisis y toma de decisiones técnicas fundamentadas.

A lo largo del desarrollo del TFG también he tenido la oportunidad de aprender y trabajar con tecnologías que hasta ahora no había utilizado en profundidad, como PHP para la lógica del lado del servidor, y HTML y CSS para el diseño de interfaces web. Este aprendizaje ha ampliado mis competencias técnicas y me ha permitido comprender mejor la integración entre el *frontend* y el *backend* de una aplicación. En definitiva, este trabajo no solo ha sido un reto, sino también una experiencia motivadora que me ha permitido crecer como desarrolladora y reforzar mi interés por seguir aprendiendo y mejorando profesionalmente en el ámbito del desarrollo de software.

6 Futuras líneas de trabajo

DRACO es un sistema web que lleva en funcionamiento muchos años y está constantemente en desarrollo por lo que tiene muchas posibles nuevas líneas de trabajo.

Una posible línea de trabajo es la creación de dos nuevos comprobadores destinados a la parte de la práctica de la asignatura Traductores de Lenguajes que se corresponde con la fase de síntesis del compilador, sobre todo para el generador de código intermedio y el generador de código objeto.

Para el primero, tras generarle un fichero de prueba al alumno, debe introducir el fichero de texto con el código en formato código de tres direcciones correspondiente a la prueba. De manera similar a cómo funciona el comprobador léxico, el comprobador generaría el código de 3 direcciones que corresponde a la prueba e iría comprobando las similitudes con el código del fichero introducido por el alumno para detectar los errores. Previamente a esto el alumno debería de haber definido en una actividad aparte la secuencia que seguirán los nombres que recibirán las variables temporales en las que se almacenan los resultados necesarios.

Para el segundo, tras la generación del fichero de prueba al alumno este debe introducir el archivo de texto con el código en ensamblador correspondiente. Esta vez se necesitaría poder probar el funcionamiento del código ensamblador, ejecutándolo y poder comprobar que este genera las salidas esperadas según el código de prueba.

Otra posible línea de trabajo es la introducción de nuevos tipos de actividades. Por ejemplo, una actividad en la que aparezcan frases incompletas con unos recuadros para terminar de rellenarlas. Este tipo de actividad puede tener dos opciones, una en la que el alumno puede escribir libremente en el recuadro y otra en la que tiene distintas opciones de recuadro para elegir y por ejemplo arrastrando una opción al hueco libre la selecciona. También pueden aparecer en la pantalla varias frases y varias respuestas y el alumno debe completar cada frase con la posibilidad que cree que queda mejor.

Otro tipo de actividad que se puede implementar es la de corrección de errores en frases, al alumno se le muestra una definición incorrecta por pantalla y este debe reescribir la definición de manera correcta. Otra opción para esta clase de actividad es que el error se encuentre en una única palabra y el alumno deba identificar cuál es y seleccionarla.

Siguiendo con la línea de las actividades se podrían crear actividades más grandes para ayudar al repaso al final del curso como juegos clásicos como crucigramas o sopas de letras. Los crucigramas serían de definiciones relacionadas con el temario y podrían generarse en el sistema usando algún sistema en el que teniendo un banco de palabras y sus definiciones genere el tablero conectando las palabras por letras. Para las sopas de letras se podría usar el mismo banco de palabras y definiciones teniendo plantillas de letras aleatorias para completar los huecos restantes. La idea es que mediante la definición el alumno sepa qué palabra debe buscar y la seleccione.

Terminando con las líneas de las actividades y buscando trabajar en la línea de la mejora continua y fidelización del estudiante, se puede implementar un mismo juego o actividad, de tipo reto diario, que no lleve más de 2 minutos y

que tenga niveles nuevos cada día para fomentar que los alumnos entren diariamente a pasarse el nuevo nivel y conseguir los puntos diarios.

Siguiendo con la línea de la fidelización del estudiante, se podría implementar una rueda de la fortuna en la que el alumno tenga una tirada gratuita al día y pueda comprar más tiradas cuyo precio se incrementaría con cada compra mediante puntos o DRACoins. Esta ruleta tendría premios como un intento extra en alguna actividad, distintas cantidades de puntos, distintas cantidades de monedas o una reducción de tiempo de espera del grupo de prácticas entre otras. Esta implementación daría un motivo a los alumnos para entrar a la web diariamente y le daría otro uso a las monedas y los puntos.

Otra línea en la que se podría trabajar puede ser en una biblioteca de DRACodes. Los DRACodes son recompensas en forma de código que al ser introducidos en el sistema se transforman en puntos para el alumno. Estos pueden ser encontrados en distintos puntos de la universidad para su escaneo o pueden ser entregados por el profesor en clase. En la vista del alumno podría tener una biblioteca dónde le aparezcan en blanco y negro todos los DRACodes que no ha conseguido y se marquen en color cuando haya introducido el código de este. En la vista del profesor este podía ver todos los DRACodes en una página y estadísticas sobre cuántos alumnos han introducido el código de dicho DRACode y ya pinchando sobre él podría ver nombre del alumno y fecha en la que ha introducido el DRACode.

Otra posible mejora que se puede realizar en el sistema es en el módulo del tablón de avisos. La idea es que cuando un nuevo aviso es subido por los administradores del sistema un correo electrónico sea enviado a todos los alumnos matriculados en el sistema en ese momento. Esto sería útil ya que este tablón de anuncios es el modo de comunicación más rápido que tienen los profesores con los alumnos, sin embargo, los alumnos no suelen revisar este diariamente y pueden perderse noticias importantes; así, mediante esta notificación al correo, pueden estar enterados.

7 Análisis de Impacto

El desarrollo de este Trabajo de Fin de Grado en el sistema web DRACO tiene un significativo impacto en distintos aspectos como el personal, el social, el universitario o el económico. A continuación, se desarrolla cada uno de ellos:

Ámbito personal:

La realización de este Trabajo de Fin de Grado ha supuesto un importante avance en el desarrollo de competencias técnicas, como el aprendizaje de nuevos lenguajes como PHP, HTML o CSS y organizativas a nivel individual. La exploración profunda del sistema DRACO y el análisis de este no solo ha reforzado los conocimientos adquiridos a lo largo de los cuatro años de estudio del Grado sobre todo en el ámbito de los compiladores y procesadores de lenguajes, sino que también ha favorecido la consolidación de habilidades transversales como la capacidad de análisis crítico, la planificación estructurada de tareas o la adaptación a tecnologías ya existentes. Además, este proyecto ha servido como una oportunidad para integrar conocimientos de distintas asignaturas de forma práctica y aplicada, consolidando una base técnica sólida que podrá ser útil en otros contextos profesionales, aunque no estén directamente relacionados con el desarrollo de compiladores.

Ámbito social:

Desde el punto de vista social, el trabajo puede contribuir a mejorar la calidad del aprendizaje en asignaturas clave dentro de los estudios del Grado, como Procesadores de Lenguajes o Traductores de Lenguajes, que suelen implicar una alta carga conceptual y técnica. La utilización de un entorno interactivo como DRACO promueve un aprendizaje más dinámico, en el que el alumno refuerza de forma práctica los conocimientos teóricos adquiridos en clase. Al incorporar elementos de gamificación, el sistema consigue aumentar la implicación del estudiante, que se siente motivado a avanzar y superar retos progresivos. Esto, a su vez, proporciona al profesorado una herramienta eficaz para realizar un seguimiento detallado del progreso individual, detectando patrones de error o conceptos que requieren refuerzo adicional. Esta retroalimentación bidireccional entre alumno y docente favorece una experiencia educativa más participativa, que impacta positivamente en el rendimiento académico y en la percepción general de la asignatura por parte del alumnado.

Ámbito económico:

Desde la perspectiva económica, la implementación de este sistema puede representar una inversión rentable a largo plazo. Al incorporar técnicas de gamificación, es probable que se incremente el compromiso de los alumnos con la asignatura, lo que contribuirá a reducir el abandono y probablemente a elevar el rendimiento académico. Estas mejoras podrían repercutir favorablemente en la imagen institucional de la universidad, haciendo que esta resulte más atractiva para futuros estudiantes o profesores que valoren métodos pedagógicos modernos y dinámicos en el ámbito de la informática. Asimismo, el sistema desarrollado podría adaptarse para ser reutilizado en otras materias del plan de estudios o incluso exportarse a distintas facultades o instituciones educativas.

El Trabajo de Fin de Grado desarrollado también está relacionado con varias de las metas establecidas por los Objetivos de Desarrollo Sostenible (ODS) de la

Agenda 2030 establecidos por las Naciones Unidas. Estos se describen a continuación:

ODS 4: Educación de calidad

El sistema desarrollado tiene como uno de sus pilares fundamentales la mejora de la calidad educativa mediante el uso de estrategias activas de aprendizaje. A través de la gamificación y de un entorno interactivo que fomenta la participación continua, se estimula la implicación del alumnado, lo cual contribuye a una mejor comprensión de los contenidos, un mayor interés por la asignatura y una reducción significativa de los índices de abandono. Esta iniciativa se vincula estrechamente con la meta 4.3, al promover un acceso equitativo a la educación superior y técnica, eliminando barreras tradicionales de motivación o metodología, y con la meta 4.4, al proporcionar a los estudiantes competencias clave para el entorno laboral actual, como el pensamiento lógico, la resolución de problemas, la autonomía en el aprendizaje y el dominio de herramientas digitales que favorecen la empleabilidad y el desarrollo profesional a largo plazo.

ODS 8: Trabajo decente y crecimiento económico

Este proyecto promueve la adquisición de capacidades y conocimientos directamente aplicables al mundo laboral, lo que refuerza la preparación de los estudiantes para afrontar futuros desafíos profesionales en el ámbito tecnológico. A través de un modelo formativo más práctico y centrado en el aprendizaje activo, se cultivan habilidades blandas y técnicas que resultan valiosas para empresas e instituciones. Esto contribuye especialmente a la meta 8.6, que busca reducir el porcentaje de jóvenes que no están empleados ni reciben educación o formación, reforzando la transición entre el entorno académico y el profesional, y también a la meta 8.9, fomentando un modelo de enseñanza que impulsa la innovación, la productividad y el crecimiento económico sostenible desde las bases del sistema educativo. Además, la experiencia adquirida puede fomentar el espíritu emprendedor entre los estudiantes, animándolos a generar sus propios proyectos tecnológicos en el futuro.

ODS 9: Industria, innovación e infraestructura

La propuesta integra el desarrollo y la aplicación de herramientas tecnológicas novedosas dentro del contexto educativo, lo que supone una apuesta clara por la innovación pedagógica y por la actualización del entorno de enseñanza. El diseño del sistema implica el uso de infraestructuras digitales modernas y promueve una evolución en la forma de enseñar y aprender, haciendo uso de plataformas accesibles, entornos gamificados y herramientas de seguimiento del progreso estudiantil. De este modo, se contribuye a la meta 9.5, al incentivar la investigación aplicada en el ámbito de la educación tecnológica y el desarrollo de nuevos métodos didácticos basados en tecnologías emergentes, y a la meta 9.c, facilitando el acceso a soluciones digitales adaptables y escalables que pueden beneficiar a otros centros educativos con recursos limitados, contribuyendo a cerrar la brecha tecnológica entre instituciones con diferentes capacidades económicas.

ODS 10: Reducción de las desigualdades

La plataforma busca ofrecer una experiencia accesible, flexible y abierta, eliminando barreras que tradicionalmente afectan al acceso equitativo al

conocimiento. Gracias a su diseño adaptable y a su compatibilidad con múltiples dispositivos, desde ordenadores hasta teléfonos móviles, se facilita la participación de estudiantes con distintos perfiles, condiciones y capacidades, reduciendo así la dependencia de recursos físicos o tecnológicos avanzados. Esto responde a la meta 10.2, que plantea la inclusión social y académica de todos los colectivos, promoviendo entornos educativos que respeten la diversidad y fomenten la participación activa sin discriminación, y a la meta 10.3, que aboga por asegurar la igualdad de oportunidades mediante sistemas que no discriminen por motivos socioeconómicos, físicos o culturales, impulsando una educación más justa, inclusiva y universal.

8 Bibliografía

- [1] A. Casero, «Historia de HTML: Desde su origen hasta 2025,» 2025 Febrero 2025. [En línea]. Available: <https://keepcoding.io/blog/historia-de-html/>.
- [2] T. School, «Historia del lenguaje HTML,» 2025 Febrero 22. [En línea]. Available: <https://www.tokioschool.com/noticias/historia-lenguaje-html/>. [Último acceso: 19 Marzo 2025].
- [3] A. M. y. Villena, «HTML, Historia y características generales,» [En línea]. Available: http://www.aeemt.com/contenidos_socios/Informatica/Informac_Informat_Tecnolog/AMV_AGI_AEEMT_HTML_Historia.pdf. [Último acceso: 18 Marzo 2025].
- [4] Whatwg, «HTML,» [En línea]. Available: <https://html.spec.whatwg.org>. [Último acceso: 19 Marzo 2025].
- [5] P. Vinuesa, «HTML5: Todo sobre el último estándar de HTML,» Orange, 25 noviembre 2022. [En línea]. Available: <https://blog.orange.es/consejos-y-trucos/html5/>. [Último acceso: 19 marzo 2025].
- [6] U. d. Murcia, «Introducción a HTML y CSS. Desarrollo de Aplicaciones Web.,» 11 02 2019. [En línea]. Available: <https://www.um.es/docencia/barzana/DAWEB/2017-18/daweb-tema-1-introduccion-html-css.html>. [Último acceso: 21 Marzo 2025].
- [7] Semrush, «Etiquetas HTML: Qué son y para que sirven, lista completa.,» 20 Enero 2023. [En línea]. Available: <https://es.semrush.com/blog/lista-de-html-tags/>. [Último acceso: 22 Marzo 2025].
- [8] R. M. Zermeño, «CSS, historia de los estilos y el diseño web.,» Azul School, 26 Agosto 2022. [En línea]. Available: <https://azulschool.net/css-historia-de-los-estilos-y-el-diseno-web/>. [Último acceso: 22 Marzo 2025].
- [9] I. Larios, «La evolución de CSS en la web.,» Designcode, 26 Marzo 2023. [En línea]. Available: <https://designicode.com/2023/03/26/la-evolucion-de-css-en-la-web/>. [Último acceso: 22 Marzo 2025].
- [10] E. e. d. IONOS, «CSS en HTML: Una guía con ejemplos,» IONOS, 24 Octubre 2022. [En línea]. Available: <https://www.ionos.es/digitalguide/paginas-web/disenio-web/css-en-html/>. [Último acceso: 26 Marzo 2025].
- [11] ManzDev, «Estructura de CSS,» [En línea]. Available: <https://lenguajecss.com/css/introduccion/estructura-de-css/>. [Último acceso: 26 Marzo 2025].
- [12] E. Virtual, «Historia y evolución de PHP,» [En línea]. Available: <https://einavirtual.com/php/historia-es>. [Último acceso: 8 Abril 2025].

- [13] PHP, «Historia de PHP - Manual,» PHP, [En línea]. Available: <https://www.php.net/manual/es/history.php.php>. [Último acceso: 26 Marzo 2025].
- [14] JetBrains, «25 años de historia de PHP,» [En línea]. Available: <https://www.jetbrains.com/es-es/lp/php-25/>. [Último acceso: 7 Abril 2025].
- [15] Hostinger, «¿Qué es PHP? Aprende todo sobre el lenguaje de scripting,» 3 Septiembre 2024. [En línea]. Available: <https://www.hostinger.com/es/tutoriales/que-es-php>. [Último acceso: 4 Abril 2025].
- [16] DataScientest, «SQL: todo lo que necesitas saber sobre el lenguaje de programación de bases de datos,» 3 Diciembre 2021. [En línea]. Available: <https://datascientest.com/es/sql-todo-lo-que-necesitas-saber-sobre-el-lenguaje-de-programacion-de-bases-de-datos>. [Último acceso: 5 Abril 2025].
- [17] IBM, «¿Qué es el lenguaje de consulta estructurado (SQL)?,» 31 Mayo 2024. [En línea]. Available: <https://www.ibm.com/es-es/think/topics/structured-query-language>. [Último acceso: 6 Abril 2025].
- [18] IONOS, «SQL Commands,» 29 Noviembre 2022. [En línea]. Available: <https://www.ionos.es/digitalguide/servidores/configuracion/comandos-sql/>. [Último acceso: 6 Abril 2025].
- [19] Europeanvalley, «¿Que es un Compilador en programación?,» 17 Abril 2018. [En línea]. Available: <https://europeanvalley.es/noticias/que-es-un-compilador-en-programacion/>. [Último acceso: 6 Abril 2025].
- [20] J. R. Catalán, «COMPILADORES Teoría e implementación,» [En línea]. Available: https://www.rclibros.es/wp-content/uploads/pdf/Cap.1_Compiladores.pdf?srsId=AfmBOooN3SPJIQepJW_t2ZmlAKjIVn0paRwqss_0LCPSFO7X8X9R8kfn. [Último acceso: 10 Abril 2025].
- [21] M. L. G. Díaz, «UVA,» [En línea]. Available: <https://www.infor.uva.es/~mluisa/talf/docs/aula/A7.pdf>. [Último acceso: 7 Abril 2025].
- [22] M. A. Moreno, M. d. l. C. Echeandía, A. O. d. l. Punete y E. P. Cañabate, Compiladores e interpretes: teoría y práctica, Prearson Educacion, 2006.
- [23] G. d. canarias, «Gamificación,» [En línea]. Available: <https://www3.gobiernodecanarias.org/medusa/ecoescuela/pedagogic/gamificacion/#:~:text=¿Qué%20es%3F,la%20participación%20de%20los%20estudiantes..> [Último acceso: 11 Abril 2025].
- [24] Euroinnova, «Origen de la gamificación: historia y evolución en educación,» [En línea]. Available: <https://www.euroinnova.com/docencia-y-formacion-para-el-profesorado/articulos/origen-gamificacion>. [Último acceso: 6 Abril 2025].

- [25] iSeazy, «Elementos de la gamificación: Componentes clave para enriquecer la experiencia de e-learning,» 16 Octubre 2023. [En línea]. Available: <https://www.iseazy.com/es/blog/elementos-de-la-gamificacion/>. [Último acceso: 7 Abril 2025].
- [26] WampServer, «WampServer,» [En línea]. Available: <https://www.wampserver.com/en/>. [Último acceso: Abril 2025].
- [27] PHP, «PHP - Manual,» [En línea]. Available: <https://www.php.net/manual/es/function.htmlentities.php>. [Último acceso: 18 Abril 2025].

Este documento esta firmado por



Firmante	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
Fecha/Hora	Wed Jun 04 12:52:23 CEST 2025
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)