Universidad Politécnica
de Madrid

Escuela Técnica Superior de Ingenieros Informáticos

Grado en Ingeniería Informática

Trabajo Fin de Grado

Diseño de la Configuración y Evaluación de Comprobadores Automáticos y Mantenimiento de un Sistema de Gamificación para la Educación

Autor: Sergiu Paul Petrila

Tutor: José Luis Fuertes Castro

Madrid, junio 2024

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: Diseño de la Configuración y Evaluación de Comprobadores Automáticos y Mantenimiento de un Sistema de Gamificación para la Educación
Junio 2024

Autor: Sergiu Paul Petrila

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 se ha realizado con el objetivo de implementar nuevas funcionalidades e interfaces al sistema DRACO (Dinámica de Refuerzo del Aprendizaje de Compiladores). Estas nuevas funcionalidades e interfaces amplían el alcance de DRACO para poder poner el sistema en uso también para la asignatura de Traductores de Lenguajes. Además de estas nuevas implementaciones, se han realizado mejoras dentro del sistema mediante tareas de mantenimiento.

Las nuevas funcionalidades que se han añadido son las definiciones del Generador de Código Intermedio y Generador de Código Objeto previas a las comprobaciones de dichos módulos, además de sus respectivas interfaces. Este Trabajo de Fin de Grado también ha dejado preparado las interfaces de los comprobadores para que, cuando se desarrollen dichos comprobadores en el sistema, se puedan implementar junto dichas interfaces.

Las definiciones previas a los comprobadores darán la posibilidad a los alumnos a poder probar sus Traductores, ya sea al generar el fichero de código intermedio o el fichero de código objeto.

Por otro lado, las tareas de mantenimiento que se han realizado han servido para corregir errores o incluir mejoras en el sistema, tanto para la parte del administrador, como para la parte del alumno.

En esta memoria se explica todo el proceso que se ha seguido para la realización del trabajo, mediante un planteamiento del problema, las soluciones, las implementaciones y las pruebas necesarias.

Abstract

This Final Degree Project has been carried out with the objective of implementing new functionalities and interfaces to the DRACO system (Dynamics of Reinforcement of Compiler Learning). These new functionalities and interfaces expand the scope of DRACO to be able to put the system into use also for the Language Translators subject. In addition to these new implementations, improvements have been made within the system through maintenance tasks.

The new functionalities that have been added are the definitions of the Intermediate Code Generator and Object Code Generator prior to the checks of said modules, in addition to their respective interfaces. This Final Degree Project has also prepared the interfaces of the checkers so that, when these checkers are developed in the system, these interfaces can be implemented together.

The definitions prior to the checkers will give students the possibility to test their Translators, whether to generate the intermediate code file or the object code file.

On the other hand, the maintenance tasks that have been carried out have served to correct errors or include improvements in the system, both for the administrator and for the student.

This report explains the entire process that has been followed to carry out the work, through a statement of the problem, the solutions, the implementations, and the necessary tests.

Tabla de Contenidos

Contenido

1 Introducción

Draco es una plataforma *online* de apoyo a la enseñanza desarrollada con los lenguajes PHP, HTML, CSS y SQL, que usa técnicas de gamificación y está diseñada para alumnos de la Universidad Politécnica de Madrid que estén cursando Procesadores de Lenguajes o Traductores de Lenguajes. El sistema tiene implementado un sistema de niveles, monedas y puntos que proporciona un entorno interactivo en el que los alumnos pueden competir y aprender en un entorno similar a un juego.

Draco está construido en dos partes principales: la vista del estudiante, donde el alumno puede realizar actividades, entregar prácticas y completar tests, entre muchas otras cosas, y la vista del administrador, en la que el profesor puede configurar el sistema, tanto las prácticas como las actividades.

Por un lado, en la parte del alumno, mostrada en la Figura 1, existen varios campos importantes a destacar. En primer lugar, está el apartado de actividades, donde los alumnos pueden realizar tareas que publican los profesores a medida que avanza el curso. El otro módulo importante es el de las prácticas, donde están los comprobadores de la práctica de Procesadores de Lenguajes, incluyendo, la definición de *tokens*, el análisis léxico, el análisis de la tabla de símbolos, la definición de terminales, el análisis sintáctico, la definición de tipos y el análisis semántico.

Figura 1. Página principal de DRACO por parte del alumno

El sistema de niveles con el que cuenta Draco permite al alumno avanzar en la clasificación a medida que completa ejercicios dentro de la plataforma. Además, cada nivel desbloquea nuevos desafíos y recompensas.

Los alumnos también pueden hacer uso de Dracodes, que pueden ser tanto códigos alfanuméricos que los alumnos pueden introducir en el sistema como códigos QR que pueden ser escaneados con un dispositivo móvil para la obtención adicional de puntos o monedas. Se obtienen tanto en clase por mérito específico o en algún tablón de anuncios por la Escuela y sirve para conectar el mundo real con DRACO.

Adicionalmente, la plataforma dispone de un sistema de monedas virtuales llamadas Dracoins que los alumnos pueden obtener por participaciones en el sistema o realizando actividades o entregas de los módulos de las prácticas dentro del sistema. Las Dracoins se pueden utilizar para obtener otros beneficios como, por ejemplo, comprar intentos para los comprobadores o pistas al resolver actividades.

Draco registra el progreso de cada alumno a través de puntos. Estos puntos se otorgan por participar en ejercicios, ya sean, actividades propuestas por el profesor o haciendo entregas mediante los comprobadores de las prácticas.

Además, hay varios apartados de información para el alumno como: uno con un resumen de las insignias obtenidas y las que faltan por obtener, un tablón de noticias donde los alumnos pueden enterarse de las últimas novedades actualizadas por los profesores para informar sobre exámenes, actividades, etc. Finalmente, hay un apartado de perfil donde existe información del alumno y donde también se puede poner un avatar.

Por otro lado, en la parte del profesor, mostrada en la Figura 2, existen varios campos importantes de configuración. Primero aparece el configurador de las actividades permitiendo añadir al sistema nuevas actividades definiendo previamente ciertos campos de configuración.

Además, otro apartado importante es el configurador de los módulos de las prácticas, donde se puede configurar cada comprobador de la práctica de forma independiente y también el lenguaje que se usará, junto con los fragmentos fuente (ficheros de texto con partes pequeñas de código del programa). Estos fragmentos se combinan siguiendo ciertas reglas basadas en plantillas, generando así un fichero de texto para que los alumnos puedan realizar las comprobaciones pertinentes.

Figura 2. Página principal de DRACO por parte del profesor

En la página del profesor también se puede ver la clasificación del curso con más información que los alumnos no disponen en su apartado de clasificación, además de incluir la posibilidad de exportar datos del curso.

Adicionalmente, también se pueden configurar las insignias que se pueden obtener dentro de DRACO y por último un apartado para configurar el sistema, las monedas y la tienda.

Para la realización de este Trabajo de Fin de Grado se ha necesitado instalar, configurar y entender el proyecto y el entorno para poder trabajar en local. Se ha utilizado XAMPP para disponer de un servidor local; para el desarrollo del trabajo ha sido necesario tener conocimientos previos programando en PHP, SQL, CSS3 y HTML5. También ha sido importante entender los conceptos de gamificación y las pautas de WCAG para que la web sea accesible para todas las personas con discapacidad y que puedan utilizarla con facilidad.

El principal objetivo de este Trabajo de Fin de Grado es aportar nuevas funcionalidades para los correctores de las prácticas, en concreto, para incorporar los correctores de la asignatura de Traductores de Lenguajes.

En la parte del profesor, habrá que diseñar y desarrollar las páginas de configuración del generador de código intermedio (GCI) y del generador de código objeto (GCO) para que pueda configurar ambos módulos de dicha práctica. Para la configuración de ambos generadores se ha utilizado de guía los otros comprobadores de la parte de Procesadores de Lenguajes realizando todas las modificaciones necesarias, así como crear nuevas tablas de la base de datos y adaptando las tablas similares. También se han tenido que configurar nuevas insignias para la asignatura.

Para la parte del alumno, habrá que diseñar y desarrollar las páginas con la interfaz del comprobador del GCI y GCO, en las se pueden realizar las entregas de los módulos, así como el sistema de puntos que se le asignará a cada alumno. Para poder realizar el comprobador del GCI hay que realizar, de manera similar a la que se hace en el análisis léxico, una definición de cuartetos para dicho comprobador y su implantación con la interfaz y las funcionalidades. El proceso de comprobación es distinto a los otros analizadores de la parte de Procesadores, ya que no pueden existir errores y, por tanto, esas pantallas del proceso del comprobador no son necesarias. Después, tras definir dichos cuartetos, se le solicitará al alumno un fichero y se arrancará el comprobador para mostrar los resultados. Para el comprobador del GCO, no será necesaria ninguna definición como en el anterior módulo, pero será necesaria alguna pregunta al alumno para definir el funcionamiento del comprobador. Finalmente, al igual que en el módulo de GCI, se le solicitará un fichero al alumno para arrancar el comprobador y mostrar los resultados correspondientes. Los comprobadores de GCI y GCO propiamente dichos serán desarrollado por otros alumnos, por lo que el presente Trabajo de Fin de Grado depende de la finalización de dichos módulos, aunque se preparará todo el entorno de DRACO para que puedan integrarse una vez finalizados.

A todo esto, hay que recalcar la importancia de las estadísticas y los resultados de las comprobaciones de los alumnos, para que queden todos los datos registrados de los nuevos comprobadores. Por eso hay que añadir en la parte del profesor las estadísticas correspondientes y los registros de las comprobaciones de cada módulo.

También hay que añadir que, en el apartado de configuración del sistema es necesario disponer de un parámetro para determinar la asignatura activa. Por tanto, dependiendo de la asignatura activa, el alumno podría ver solamente los módulos correspondientes a dicha asignatura. Este parámetro también afecta a las insignias y a las actividades ya que para cada curso son distintas. En

cambio, en la vista del profesor, el parámetro no afecta ya que así se pueden configurar ambas asignaturas desde las mismas páginas.

Por otro lado, otro de los objetivos del trabajo es realizar tareas de mantenimiento en el sistema para mejorar el funcionamiento de este, incluyendo tareas de mantenimiento perfectivo y correctivo entre las cuales se incluyen, por ejemplo:

- Añadir información adicional al perfil del alumno.
- Tenía que quedar registrado de alguna manera cuando el alumno arrancaba una comprobación y quedaba sin entregar porque se le acababa el tiempo.
- A raíz de ese problema, también se añadió un nuevo campo *score* que indicaba también si una comprobación se había quedado sin entregar.
- Este nuevo campo *score* se ha utilizado para indicar la puntuación que ha obtenido un grupo en una comprobación de un módulo dentro de la parte de estadísticas en la vista del profesor.
- Cada módulo de las prácticas puede tener distintas fases activas que, dependiendo de la fecha, el porcentaje de puntuación total de puntos obtenidos por completar dichos módulos es distinto, y se ha tenido que implementar la comprobación de nivel para cada fase, es decir, que cada fase iba a tener un nivel mínimo establecido para poder acceder y realizar las comprobaciones.
- En la parte del administrador, en el resumen de las estadísticas individuales de cada grupo, para cuando una prueba no se ha completado, no deben aparecer algunas opciones.
- Otros problemas que se detecten durante el desarrollo del Trabajo.

2 Estado de la cuestión

En este apartado se presentarán todas las herramientas y conocimientos necesarios previos para la realización del Trabajo de Fin de Grado. Se detallarán conceptos técnicos sobre programación web, la accesibilidad necesaria en la plataforma donde se ha trabajado, el funcionamiento de compiladores y el significado de un sistema de gamificación para la educación.

2.1 HTML

HyperText Markup Language o más conocido como HTML, es un lenguaje de programación que da estructura y significado a las páginas web.

El origen se remonta a 1980, cuando el físico Tim Berners-Lee, trabajador del CERN (Organización Europea para la Investigación Nuclear), propuso un nuevo sistema de hipertexto para facilitar el intercambio de documentos [1].

Los sistemas de hipertexto ya existían con anterioridad, como Xanadu, creado por Ted Nelson y que fue el primero en darle ese nombre. Estos sistemas no lograron una implementación completa debido a varios problemas, incluyendo la falta de estandarización en muchos casos [2].

Posteriormente, cuando Tim Berners-Lee terminó de desarrollarlo, lo presentó en una convocatoria donde se decidiría cuál sería el sistema de hipertexto para internet. Y finalmente, tras asociarse con el ingeniero de sistemas Robert Cailiau, lograron presentar la propuesta ganadora conocida mundialmente como *World Wide Web* (o W3).

En 1991, se publicó bajo el nombre de *HTML Tags*, el primer documento oficial con la descripción de HTML.

El desarrollo de este lenguaje como estándar, comenzó en 1993 mediante la propuesta de la IETF (*Internet Engineering Task Force*). Aunque se definieran etiquetas importantes como tablas, imágenes o formularios, no se consolidaron como estándar oficial. En 1995, la IETF publicó el HTML 2.0, que se convirtió en el primer estándar oficial del lenguaje y, en 1997, el W3C (*World Wide Web Consortium*) publicó la versión HTML 3.2 que incorporó avances significativos como los *applets* de Java o la disposición de textos alrededor de imágenes.

Posteriormente, en 1998 se publicó el HTML 4.0, con mejoras importantes como la introducción de hojas con el estilo de CSS y la posibilidad de incluir *scripts* o programas en las páginas web. También se mejoró la accesibilidad de las páginas, la creación de tablas más complejas y el uso de los formularios [3].

Más tarde, en 2004, ante la lentitud del W3C para actualizar el estándar de HTML, un grupo de empresas liderado por Apple, Mozilla y Opera creó el WHATWG (*Web Hypertext Application Technology Wroking Group*) para desarrollar HTML 5. Su objetivo era crear un lenguaje más compatible y flexible con los nuevos avances multimedia y las necesidades del mercado, incluyendo funcionalidades como audio, video, Canvas y API para aplicaciones web [4].

Finalmente, en 2007 el W3C se unió al desarrollo de HTML 5, que publicaron el primer borrador en 2008 y en 2014 se lanzó la versión final. Actualmente, HTML 5 se ha convertido en un estándar dominante en el sector para la creación de páginas web y su desarrollo continúa para una futura versión 5.1.

Los documentos HTML definen la estructura y el contenido de las páginas web, permitiendo a los navegadores interpretar y mostrar toda la información de manera coherente. Este lenguaje de programación se compone de etiquetas que dan formato y que se describen entre corchetes angulares (< y >) y se dividen en dos partes, la apertura y el cierre. Cuando se cierra la etiqueta, se utiliza una barra diagonal antes del nombre de la etiqueta. Entre estas etiquetas se escriben la información que se quiere que se despliegue con el formato requerido. La estructura es la siguiente:

```
<etiqueta>Contenido</etiqueta>
```

Las dos partes principales de un documento HTML son la cabecera que se identifica con la etiqueta `<head>` y el cuerpo que se identifica con la etiqueta `<body>` mostrados en la Figura 3. La cabecera contiene información sobre la página como el título, y adicionalmente puede contener la codificación de caracteres y los metadatos. Esta información por lo general no se muestra en la página web, pero es importante para los navegadores y motores de búsqueda.

Por otra parte, el cuerpo, es el elemento que contiene todo el contenido de la página web. Este contenido sí que es visible para los usuarios, y también pueden interactuar con él. Además, es obligatorio usar `<!DOCTYPE html>` al principio de cada documento, para identificar la versión de HTML utilizado, siendo necesaria para la declaración y el correcto funcionamiento del código.

```
<!doctype html>
<html lang="es">
 <head>
  <meta charset="utf-8" />
  <meta name="viewport" content="width=device-width" />
  <title>Mi página de prueba</title>
 </head>
 <body>
  <img src="images/firefox-icon.png" alt="Mi imagen de prueba" />
 </body>
</html>
```

Figura 3. Ejemplo de estructura básica de un documento HTML

Las etiquetas también pueden tener asociados unos atributos. Un atributo es información adicional acerca del elemento pero que no aparece en el contenido real del elemento. El nombre de un atributo debe ir seguido de un signo de igual (=), y finalmente, las comillas de apertura y cierre, encerrando el valor del atributo como se ve en la Figura 4.

Figura 4. Ejemplo de estructura básica para un atributo

Algunas de las etiquetas más importantes [5] y que más se han utilizado para este Trabajo de Fin de Grado son:

- `<form>` que define un formulario para recolectar información que introduce el usuario. Se usa principalmente para enviar respuestas en actividades y en las pruebas de las prácticas.

- `<input>` crea un campo de entrada de datos. Existen diferentes tipos de `input` según la información que se quiera capturar. Entre ellos se encuentran texto, casillas de verificación, números, etc.
- `<button>` genera un botón interactivo que realiza una acción deseada cuando el usuario lo pulsa.
- `<label>` representa una etiqueta para un elemento en una interfaz de usuario. Se debe asociar esta etiqueta con el elemento mediante el atributo `for`. Sirve para mejorar la accesibilidad y la usabilidad.
- `<table>` define una tabla completa. Se usan las etiquetas `<th>` para definir celdas de encabezado dentro de la tabla y `<td>` para añadir celdas de datos a la tabla.
- `<img>` permite insertar imágenes en la página web.
- `<a>` permite crear hipervínculos a otros recursos que se pueden especificar normalmente con el atributo `href.`
- `<strong>` consigue destacar un texto y ponerlo en negrita. Por otro lado, también está la etiqueta `<em>` para dar énfasis a un texto y ponerlo en cursiva.

2.2 CSS

CSS (*Cascading Style Sheets*) es un lenguaje de programación que se utiliza para darle un aspecto a una página web cuando se muestra en el navegador.

Alrededor de los años 70, aparecieron las primeras hojas de estilos, poco después del lenguaje de etiquetas SMGL, ya que se necesitaba un mecanismo para aplicar estilos a documentos electrónicos [6].

Surgieron dos propuestas independientes para crear lenguajes de hojas de estilo para aplicarle estilo a los documentos HTML: CHSS (*Cascading HTML Style Sheets*) fue propuesto por Håkon Wium Lie y SSP (*Stream-based Style Sheet Proposal*) entre 1994 y 1995. Poco después, se combinaron basándose en los principios de ambas propuestas para crear CSS.

A lo largo de los años, han ido saliendo nuevas versiones de CSS hasta la actual, "CSS3", que introdujo muchas mejoras respecto a la versión anterior, incluyendo mejoras de funcionamiento y rendimiento, así como la posibilidad de realizar animaciones, entre muchas otras cosas [7].

CSS usa selectores como se ven en la Tabla 1 para identificar elementos HTML a los que se quieren aplicar los estilos.

Selector	**Ejemplo**	**Descripción**
`#id`	#nombre	Selecciona el elemento con `id="nombre"`
`.class`	.intro	Selecciona todos los elementos con `class="intro"`
`element.class`	p.intro	Selecciona solo los elementos `<p>` con `class="intro"`
*	*	Selecciona todos los elementos
`element`	p	Selecciona todos los elementos `<p>`
`element.element`	div, p	Selecciona todos los elementos `<div>` y los elementos `<p>`

Tabla 1. Selectores en CSS

Una vez seleccionados los elementos, se aplican propiedades para personalizar la apariencia del estilo como el color, los bordes, el tamaño y las animaciones, entre otras [8].

Este lenguaje, como su propio nombre indica, funciona en cascada. Esto quiere decir que los estilos se aplican de forma jerárquica siguiendo reglas de cascada y herencia [9].

- Cascada: Tiene mayor prioridad el selector que tenga más especificidad, es decir, el que tenga más propiedades definidas en el estilo.
- Herencia: Los estilos se heredan en el árbol DOM (*Document Object Model*).

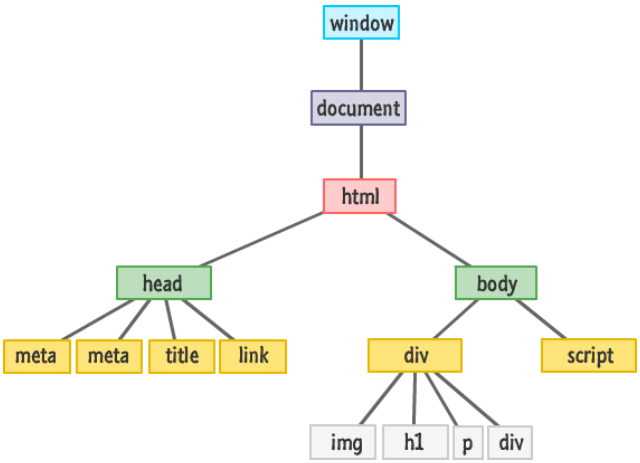

Figura 5. Esquema básico del árbol DOM

Los estilos de CSS se pueden integrar en el documento HTML de diversas maneras [10]:

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
         initial-scale=1.0">
    <title>Documento</title>
    <style>
        h1{
            color: red;
            font-size: 24px;
        }
    </style>
</head>
<body>
    <h1>Encabezado 1</h1>
    <p>Primer párrafo</p>
</body>
</html>
```

Figura 6. Ejemplo de aplicación de CSS interno en un documento HTML

- CSS Interno: que se define dentro del documento HTML entre las etiquetas `<style>` y `</style>` en el encabezado y solo se aplica a dicho documento como se ve en la Figura 6. Se usa generalmente para pequeñas modificaciones que se vayan a usar específicamente en ese documento.
- CSS Externo: primero se define un archivo CSS. Posteriormente se enlaza al documento HTML utilizando la etiqueta `<link>` como se ve en la Figura 7.

Esto permite que el código sea más fácil de interpretar, esté más organizado y se pueda utilizar en varias páginas.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width,
          initial-scale=1.0">
    <title>Ejemplo de CSS externo</title>
    <link rel="stylesheet" href="styles.css">
</head>
<body>
    <h1>Ejemplo de CSS externo</h1>
    <p>Este es un ejemplo de cómo aplicar estilos CSS
       externos a un documento HTML.</p>
    <p>Los estilos se definen en un archivo CSS separado llamado
       "styles.css".</p>
</body>
</html>
```

Figura 7. Ejemplo de aplicación de CCS Externo

- CSS En línea: se definen los estilos dentro de los elementos HTML utilizando el atributo `style` como se ve en la Figura 8. Estos estilos tienen la mayor prioridad en la cascada ya que anulan el resto de los estilos aplicados a ese elemento previamente.

```
<!DOCTYPE html>
<html lang="es">
<head>
    <meta charset="UTF-8">
    <title>Título de la página web</title>
    <title>Documento</title>
</head>
<body>
    <h1 style="color: red;">Encabezado 1</h1>
    <p style="font-weight: bold;">Primer párrafo</p>
</body>
</html>
```

Figura 8. Ejemplo de aplicación de CSS En línea

2.3 PHP

PHP es un lenguaje de programación del lado del servidor, de código abierto y que se utiliza para crear páginas web interactivas y dinámicas. Este lenguaje se integra con HTML y se ejecuta en el servidor web. Esto implica que el código de PHP se procesa antes de que la página se envíe al navegador que está utilizando el usuario.

PHP es un lenguaje de código abierto que fue creado por el programador Rasmus Lerdorf en 1994. Inicialmente, se desarrolló un conjunto de *scripts* CGI (*Common Gateway Interface*) escritos en C conocidos como *Personal Home Page Tools*. Estos *scripts* se utilizaban principalmente para rastrear visitas en una página web y administrar un currículum [11].

En 1995, Rasmus Lerdorf decidió incorporar en el lenguaje ciertas características de Perl, creando así PHP/FI, la primera versión de PHP que ya incluía un pequeño soporte para HTML.

Posteriormente, en 1996, salió de forma oficial la versión 2.0 de PHP/FI. Esta nueva generación comenzó a evolucionar siguiendo el propósito original, que fue que PHP pase de ser un conjunto de herramientas, a un lenguaje de programación.

Más tarde, en 1997, Zeev Suraski y Andi Gutmans, volvieron a escribir el analizador subyacente. Esta nueva versión, que se publicó como PHP 3.0, que se asemeja a la actual, tenía una gran extensibilidad, además de ofrecer soporte nativo para bases de datos, funciones definidas por el usuario y una sintaxis orientada a objetos [12].

Poco después, en 1998, Andi Gutmans y Zeev Suraski, comenzaron a trabajar de nuevo para mejorar la modularidad del código base de PHP, y esta vez, reescribieron el núcleo del lenguaje. Dos años más tarde, en el año 2000, se lanzó PHP 4.0, que incorporó el motor Zend (formado a partir de los nombres de sus creadores, Zeev y Andi) y aportaba un mejor rendimiento y estabilidad al lenguaje. Esta versión, incorporaba soporte para más servidores web y sesiones HTTP [13].

Más adelante, en 2004, se lanzó la versión PHP 5.0, en 2015, PHP 7.0, y finalmente PHP 8.0 en 2020, aportando el compilador JIT (*Just In Time*), tipos de unión y tipo de retorno estático entre muchas otras características [14].

PHP trabaja con un modelo de solicitud y respuesta; cuando el usuario solicita una página web, el servidor actúa detectando e identificando el código y se lo envía a un intérprete de PHP. El intérprete ejecuta el código procesando y generando contenido dinámico. Finalmente, el resultado en formato HTML se envía al navegador del usuario y el navegador se encarga de interpretar el HTML mostrando la página web (Figura 9).

Figura 9. Esquema explicativo del funcionamiento de PHP con el servidor web

Antes de empezar a programar con PHP, hay que configurar un entorno instalando un sistema de bases de datos y un servidor web compatible con el lenguaje. Una posibilidad interesante para conseguirlo es usar XAMPP Server, que incluye ambas configuraciones.

Un código de PHP siempre empieza por `<?php` y finaliza con `?>`. Este lenguaje tiene variables con tipos de datos predefinidos como *String, Integer o Boolean*, y se declaran como `$nombre_de_variable = valor`. Como muchos otros lenguajes de programación, PHP te permite utilizar operadores, controlar el flujo del programa con estructuras de control *if, else, switch, for* y *while,* y conectarse a una base de datos para manipular datos.

Además, PHP tiene un tipo de variables llamadas superglobales, como se puede ver en la Figura 10. Estas variables son siempre accesibles desde cualquier clase, función o fichero. Las variables superglobales [15] más utilizadas en este Trabajo de Fin de Grado son:

- `$_SERVER` que almacena información sobre cabeceras, rutas y ubicaciones de los *scripts*.
- `$_SESSION` permite almacenar información específica del usuario a través de varias páginas web dentro de la misma sesión. Esto significa que, si un usuario inicia sesión en una página web, mientras esté en el mismo navegador, se guarda esa sesión activa por mucho que refresque o cierre la pestaña actual o abra una nueva pestaña.
- `$_POST` es un *array* que contiene los datos que se envían al servidor desde un formulario mediante el método POST. Se suele utilizar para enviar información sensible o que no se quiere mostrar en la URL.
- `$_GET` es un *array* que contiene los datos que se envían al servidor desde un formulario mediante el método GET. Los datos se pueden ver en la URL después de un signo de interrogación.
- `$_REQUEST` es un *array* que tiene almacenado todos los datos que se envían al servidor desde un formulario tanto por el método POST como por el método GET.

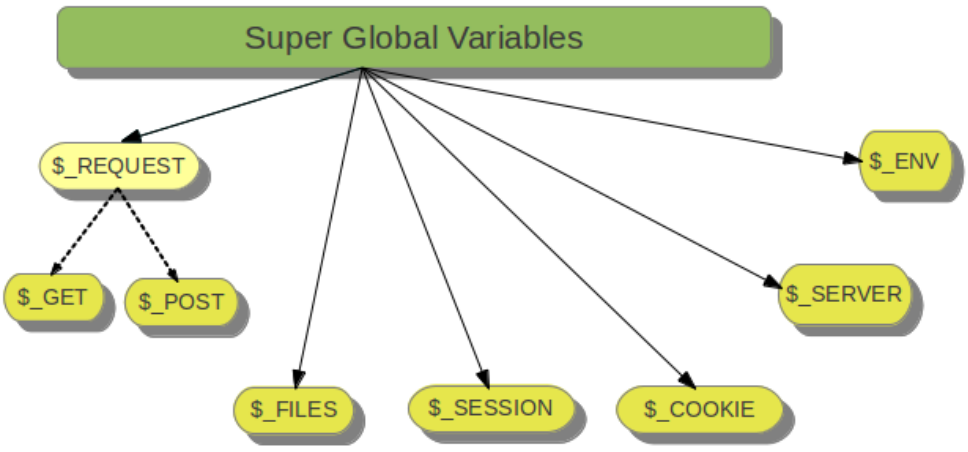

Figura 10. Variables superglobales en PHP

2.4 SQL

La historia de SQL comienza en la década de los 70, cuando Donald David Chamberlin y Raymond F. Boyce desarrollan SEQUEL (*Structured English Query Language*), el precursor de SQL, como parte del proyecto SystemR en IBM. El desarrollo de SEQUEL fue gracias a Edgar Frank Codd, quien introdujo el modelo relacional de datos, y este lenguaje se utilizó para la gestión de este tipo de bases de datos.

Poco después, en los años 80, SEQUEL se renombra a SQL (*Structured Query Language*) y pasa a convertirse en el lenguaje estándar para trabajar con bases de datos relacionales hasta la actualidad [16].

Una base de datos es una colección organizada de datos y están formadas por tablas. Las columnas representan el tipo de datos que se guardarán, y las filas indican los registros de la tabla de la base de datos, según la Figura 11. Por lo general, cada registro en una tabla se identifica con un identificador único, considerado una clave primaria.

IdAmigo	Apellidos	Nombres	Direccion	Telefono	E-mail	FechaNacimien
14	Cardona	Lorena	Villas del jardin	3141586	lafufis5000@ho	06/02/1984
15	castro	luis carlos	cuba	3335478		31/12/1985
16	eustaquia	anacleta	el cartucho	311542455		25/05/1983
12	Franco	Daniel	la pradera	3301898	dani_45@hotm	02/01/1983
7	Garcia	Mafe	Villa del prado	3384405	mafe1989@hot	21/05/1989
9	Gomez	Natalia	centro	3335880	nattygoa2@hot	28/09/1988
11	Gonzales	Sara	centro	3644523	sarita45@hotm	24/08/1987
4	Marin Rojas	Anyela	Tejares de la lo	3287144	anye71@hotma	11/10/1989
6	Ordoñez	David	centro	3384405	crwlyn1@hotm	29/11/1987
3	Osorio	Lorena	Campestre A	3323331	lorito241@hotm	27/11/1988
13	Restrepo	Felipe	Alamos	3211456	pipe_54@hotm	25/05/1987
5	Rodriguez	Angela	Tejares de la lo	3282057	angelapatico17(	17/09/1988
1	Rua Perez	Anyelo	cll12 # 25-12	3680150	anyelov2007@h	28/01/1989
10	Sanmartin	Juan David	Campestre A	3325486	juandavidsanma	27/03/1990
8	Toro	Monica	Poblado II	3380412	monicatorod@h	05/11/1985
2	Zarate	Katherin	Kennedy	3312657	kazaga29@hot	24/08/1990

Figura 11. Tabla de una base de datos relacional

Para este Trabajo de Fin de Grado se ha utilizado MySQL, que es un sistema de gestión de bases de datos de código abierto relacionales, lo que permite manipular datos de forma estructurada y utiliza el lenguaje SQL.

2.3.1 Definición de datos (DDL)

Hace referencia a los comandos SQL que diseñan la estructura de las bases de datos y de las tablas [17] y son los siguientes (se puede ver un ejemplo en la Figura 12):

- `CREATE` que crea una tabla o una base de datos entre otras cosas.
- `ALTER` que modifica la estructura de tablas, vistas, etc.
- `DROP` elimina una tabla o una base de datos entre otras cosas.

```
CREATE TABLE usuarios (
    id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    email VARCHAR(100) UNIQUE,
    edad INT
);
ALTER TABLE usuarios ADD COLUMN fecha_nacimiento DATE;
DROP TABLE usuarios;
```

Figura 12. Ejemplo comandos básicos de definición de datos

2.3.2 Consulta de datos (DQL)

Hace referencia a los comandos SQL que permiten consultar datos en una base de datos relacional [17] y son los siguientes (se puede ver un ejemplo en la Figura 13):

- `SELECT` selecciona datos de una tabla.
- `FROM` ayuda a especificar de qué tabla se buscarán los datos.
- `ORDER BY` ordena los resultados en el orden deseado.

- `WHERE` filtra la consulta mediante una condición para obtener un resultado más específico.
- `GROUP_BY` agrupa registros obtenidos de una consulta.

```
SELECT * FROM usuarios;
SELECT * FROM usuarios WHERE edad > 18;
SELECT * FROM usuarios ORDER BY nombre ASC;
SELECT COUNT(*) AS total_usuarios, pais FROM usuarios GROUP BY pais;
```

Figura 13. Ejemplo de consulta de datos en SQL

2.3.3 Manipulación de datos

Son comandos que escriben información nueva o modifican los datos existentes en la base de datos [17] (se puede ver un ejemplo en la Figura 14).

- `INSERT INTO` que inserta nuevos registros en la base de datos.
- `DELETE FROM` que elimina registros de una base de datos.
- `UPDATE` que actualiza los datos existentes de una tabla.

```
INSERT INTO usuarios (nombre, email, edad) VALUES ('Juan',
'juan@correo.es', 30);
UPDATE usuarios SET edad = 31 WHERE nombre = 'Juan';
DELETE FROM usuarios WHERE correo = 'pedro@correo.es';
```

Figura 14. Ejemplo de manipulación de datos en SQL

2.5 WCAG

La primera versión de WCAG (*Web Content Accesibility Guidelines*) fue desarrollada y publicada por W3C en 1999, y se centraba en mejorar la accesibilidad web mediante el código HTML y CSS. En 2008, la versión 2.0 amplió su alcance a todas tecnologías de programación [18].

Posteriormente, a finales de 2023 se publicó la última versión 2.2 que actualizó los cuatro principios básicos de accesibilidad previamente definidos en la versión 2.0. Actualmente los cuatro principios de WCAG y sus pautas son [19]:

- Perceptible. Se centra en hacer que la información y los componentes de la interfaz de usuario sean perceptibles para todos los usuarios. Incluye cuatro pautas:
    - Texto alternativo. Dar información textual equivalente para elementos no textuales como imágenes, vídeos y audio.
    - Contenido multimedia dependiente del tiempo. Ofrecer contenidos sincronizadas para contenido multimedia.
    - Adaptabilidad. Hay que asegurar que el contenido se pueda presentar en diferentes formatos sin perder información.
    - Distinguible. Facilitar la visualización y audición del contenido, incluyendo la distinción entre elementos importantes y secundarios.
- Operable. Se enfoca en garantizar que los usuarios puedan interactuar con la interfaz de usuario y navegar por el contenido de una página web de manera efectiva. Incluye cinco pautas:
    - Accesibilidad al teclado. Permitir que todas las funcionalidades sean accesibles mediante el teclado, sin necesidad de tener un ratón.
    - Tiempo suficiente. Otorgar tiempo suficiente a los usuarios para leer y comprender el contenido, completar formularios y realizar acciones.

    - Prevención de ataques epilépticos. Hay que asegurar que el contenido de la página web no causa ataques epilépticos.
    - Navegación. Ofrecer mecanismos de navegación claros y consistentes para que los usuarios puedan encontrar lo que buscan de forma rápida y sencilla.
    - Modalidades de entrada. Facilitar la interacción con el contenido web utilizando diversos métodos de entrada, como puede ser la voz, o mediante gestos.
- Comprensible. Este principio busca que la información y las operaciones del sitio web sean comprensibles para todos los usuarios. Incluye tres pautas:
    - Legibilidad. Hay que asegurar que el texto sea legible y comprensible.
    - Previsibilidad. Hacer que la apariencia y el funcionamiento del sitio web sean predecibles, con una estructura y una navegación lógica.
    - Asistencia a la entrada de datos. Ayudar a los usuarios a evitar y corregir errores al ingresar información en formularios.
- Robusto. Se centra en garantizar que el contenido web sea interpretable por una amplia variedad de agentes de usuario, incluyendo productos de apoyo. La única pauta es:
    - Compatibilidad. Maximizar la compatibilidad del sitio web con navegadores web actuales, así como con tecnologías de asistencia.

Las pautas de accesibilidad se establecen por tres niveles de conformidad: A, AA y AAA [20]. El nivel A es el nivel más importante y establece las pautas mínimas obligatorias que deben cumplir todas las páginas web, que se basan en que el contenido web debe ser perceptible y operable. El nivel AA establece criterios más estrictos que se centran en hacer que la web sea más fácil de usar y de comprender. Finalmente, el nivel AAA, es considerado el nivel menos importante de accesibilidad, y se consigue realizando un esfuerzo adicional para facilitar al máximo el uso de la web por todos los usuarios [21].

2.6 Gamificación

La gamificación consiste en el uso de mecánicas, elementos y técnicas de diseño que utilizan juegos y actividades en un ámbito educativo o profesional para motivar a los usuarios a que consigan mejores resultados, ya sea para mejorar alguna habilidad, obtener nuevos conocimientos o favorecer la participación [22].

En los años 80, comenzó a utilizarse este sistema en varios ámbitos, incluido el *marketing*, para atraer y retener clientes mediante regalos y puntos, y, gracias al profesor Malone, se trasladó este concepto a la educación.

En 2003, un programador británico llamado Nick Peeling, comenzó a utilizar el término *gamification*, haciendo referencia a *game* en inglés. Pero no fue hasta 2010, hasta que se empezó a difundir el término en conferencias y congresos resaltando “la importancia de la experiencia lúdica”.

Posteriormente, los creadores de videojuegos Cunningham y Zichermann, la definieron como el “uso del pensamiento lúdico y las mecánicas de juego para fomentar la participación de los usuarios y resolver problemas” [23].

Dentro de la gamificación, intervienen tres elementos fundamentales que son las dinámicas, las mecánicas y los componentes que fueron definidos en la pirámide de Gamificación por Kevin Werbach [24] como se puede ver en la Figura 15.

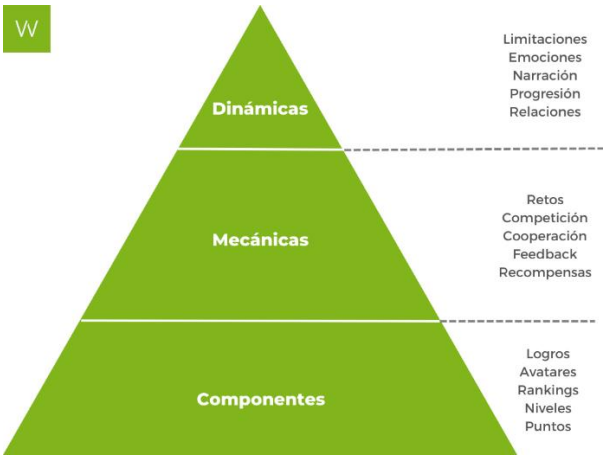

Figura 15. Pirámide de los elementos de la gamificación

- Las dinámicas están relacionadas con la motivación de los usuarios para participar e interactuar con los juegos. Existen varios tipos de dinámicas entre los que destacan:
    - Limitaciones: Ofrece la posibilidad de buscar las mejores alternativas para la resolución de un problema en un entorno limitado.
    - Narrativa: Ayuda a que el jugador se sienta parte del juego y muestre mayor interés en él.
    - Progreso: La sensación de sentir que va progresando es muy importante para que el jugador vea que está mejorando dentro del juego y quiera resolver desafíos más complejos.
    - Relaciones: Fomenta la comunicación y la colaboración para alcanzar un objetivo común.
- Las mecánicas son las reglas que definen el funcionamiento del juego. Son los elementos básicos y herramientas que permiten al diseñador del sistema gamificado generar un compromiso por parte de los usuarios a través de estas mecánicas, entre las cuales se pueden destacar:
    - Retos: Son objetivos que se les propone a los jugadores y les pueden dar puntos u obtener otra recompensa. Además, pueden tener una dificultad elegida por el administrador.
    - Competición: Genera un entorno competitivo donde hay más interés por parte de los usuarios en tener mejores resultados que el resto, y, por tanto, se esfuerzan más.
    - Área social de cooperación: Zonas que permiten la interacción con otros jugadores. Suelen ser muy útiles cuando un jugador se queda "atascado" en un nivel y puede pedir ayuda. Así, favorece el altruismo, un bien ajeno, sin pedir nada a cambio.
    - *Feedback*: Mensajes que recibe el jugador con información acerca de su progreso para hacer sentir mejor al jugador acerca de sus logros, o bien para decirle en que se ha equivocado para que pueda aprender de sus errores.
    - Recompensas: Son premios que se le otorgan al jugador por completar desafíos u otras tareas. Estas recompensas pueden ser muy variables y suelen destacar: nuevas habilidades, nuevas funcionalidades, obtención de puntos o monedas, insignias, etc.
- Los componentes son recursos que se utilizan para crear una tarea o actividad y están asociadas a las dinámicas y las mecánicas vistas anteriormente. Suelen destacar:

- Logros: Es el reconocimiento virtual por completar actividades y objetivos que motivan a progresar.
- Avatares: Son representaciones gráficas de una persona o un personaje y representan la identificación del usuario en el sistema.
- *Ranking*: Tabla que representa la clasificación del usuario respecto a los otros jugadores y se ajustan generalmente dependiendo del nivel o los logros de los jugadores.
- Niveles: Representan el progreso de un jugador en el juego. Los jugadores pueden subir de nivel normalmente cuando se llega a una cierta cantidad de puntos. Dependiendo del nivel de cada jugador se pueden desbloquear nuevas ventajas o funcionalidades.
- Puntos: Se les otorgan puntos cuando completan actividades, alcanzan un objetivo o también por participar en retos.
- Regalos: Recompensas por completar dichas actividades y objetivos.
- Insignias: Son símbolos virtuales que demuestran que se ha obtenido un logro específico.

Por otro lado, a lo largo de los años, se ha ido implementando cada vez más la gamificación para la educación. Para ello, hay que buscar las necesidades de los alumnos, y con ellas, diseñar el sistema gamificado con los elementos descritos anteriormente [25] [26] [27] [28].

2.7 Compiladores

Un compilador es un software informático que traduce un código de un lenguaje de programación de alto nivel, como puede ser C o Java, a un código de bajo nivel, que generalmente es el lenguaje máquina o ensamblador.

En los inicios de la Informática, los códigos que se escribían estaban en lenguaje máquina. Este lenguaje era un conjunto de instrucciones binarias muy complicadas de leer e implementar; por tanto, para facilitar esta tarea se desarrollaron los lenguajes ensambladores. Es por ello que, en 1952, la informática Grace Murray Hopper creo el A-0, el primer compilador de la historia, capaz de convertir lenguaje matemático en lenguaje máquina. Seis años después, presentó el B-0, un lenguaje/compilador que podría traducir instrucciones del inglés [29].

En 1957, un grupo dirigido por John Backus consiguió desarrollar un compilador en IBM específico para el lenguaje FORTRAN, y que facilitó la programación de aplicaciones científicas y de ingeniería.

En la década de los 70, se introdujeron nuevos lenguajes de programación como COBOL o Pascal. Estos lenguajes se caracterizaban por ser de alto nivel, es decir que eran más entendibles para el humano, y no es tan complicado como el lenguaje máquina [30] [31].

Hoy en día, los compiladores son una parte fundamental e imprescindible para la programación ya que pueden traducir el código de alto nivel a lenguaje máquina para que el ordenador pueda ejecutarlo. Antes de poder convertirlo en dicho programa ejecutable pasa por varias etapas de análisis y síntesis, como se ve en la Figura 16.

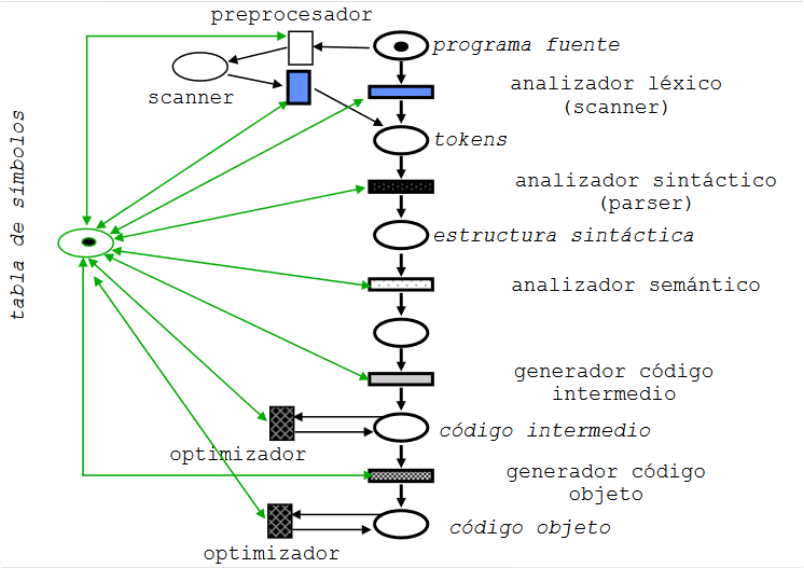

Figura 16. Fases de un compilador

2.7.1 Analizador Léxico

El analizador léxico es la primera etapa de un compilador. Su función es leer el código fuente carácter a carácter y dividirlo en unidades más pequeñas llamados *tokens*, siguiendo el proceso de un autómata finito determinista. Además, inserta en la tabla de símbolos los identificadores.

2.7.2 Analizador Sintáctico

El analizador sintáctico va recibiendo la secuencia de *tokens* generada y comprueba si esta secuencia sigue las reglas gramaticales del lenguaje de programación. Si es correcta la validación, se genera un árbol sintáctico que representa la estructura del código. El análisis sintáctico puede ser de dos tipos:

- Descendentes: Parten del axioma, y van efectuando derivaciones por la izquierda hasta obtener la secuencia de derivaciones que reconoce la sentencia.
- Ascendentes: Parten de la sentencia de entrada, y van aplicando reglas de producción hacia atrás, hasta llegar al axioma.

Además, el analizador sintáctico detecta errores cuando encuentra un *token* inesperado.

2.7.3 Analizador Semántico

Esta es la última fase antes de la fase de síntesis, y, por tanto, prepara el programa para ser traducido. El analizador semántico utiliza el árbol sintáctico generado por el analizador anterior y valida el significado del código fuente realizando una comprobación de tipos, la accesibilidad de las variables y detectando errores semánticos, como el uso de variables no declaradas con anterioridad [32] [33] [34].

2.7.4 Tabla de símbolos

La tabla de símbolos es una estructura de datos que los compiladores utilizan para almacenar información sobre los identificadores del código fuente, como variables, funciones, clases, interfaces, etc. Esta tabla se utiliza tanto en el análisis léxico, como en el análisis semántico y en la fase de síntesis, y sirve principalmente para comprobar si una variable ya ha sido declarada con anterioridad, verificar el tipo de variables declaradas y comprobar el alcance de

un nombre, entre otras funcionalidades y se suelen implementar como árboles de búsqueda binaria o tablas *hash*.

El diseño de una tabla de símbolos permite actuar a dicha tabla como un diccionario en la que se asocia cada *token* con sus atributos. La tabla debe gestionar estos nombres y sus propiedades y permitir el uso de métodos de búsqueda, inserción, eliminación y gestión de alcance de los *token*s [35] [36].

2.7.4 Generador de Código Intermedio (GCI)

El generador de código intermedio se encarga de traducir el código fuente, escrito en alto nivel, a un lenguaje intermedio que sea más fácil de procesar y traducir. Para empezar, el GCI recibe el árbol sintáctico y genera una representación intermedia del código, en un lenguaje independiente de la máquina, que se va a representar en forma de cuartetos. Este lenguaje mantiene la estructura del código fuente original, pero no tiene detalles específicos del lenguaje ni de la arquitectura del ordenador.

Generar el código intermedio tiene la ventaja de que, al ser independiente de la máquina, se puede traducir a diferentes arquitecturas. Esto permite aplicar técnicas de optimización antes de la traducción final que mejoran su rendimiento, y, además, puede ser traducido a diferentes lenguajes máquina permitiendo que el código se pueda ejecutar en diversas plataformas [34] [37].

2.7.5 Generador de Código Objeto (GCO)

El generador de código objeto toma el código generado por el GCI y lo convierte en un código para la máquina (que suele ser ensamblador o código máquina). Implica tres pasos clave:

1. Selección de instrucciones. El GCO selecciona las instrucciones específicas del conjunto de instrucciones de la máquina que mejor implementan las operaciones del código intermedio.
2. Asignación de registros. El GCO decide qué valores se almacenarán en los registros del ordenador durante la ejecución del programa. Esta asignación optimiza el acceso a memoria y mejora el rendimiento además de reducir el tamaño del archivo y eliminar código no necesario.
3. Finalmente, el código objeto se convierte en instrucciones de máquina reales que el ordenador puede ejecutar [38] [39] [40].

3 Planteamiento del problema

3.1 Introducción

En este capítulo se presenta la especificación de requisitos software para las ampliaciones y modificaciones realizadas durante este Trabajo de Fin de Grado en el sistema web DRACO.

3.1.1 Propósito

El propósito de esta especificación de requisitos es definir las mejoras y novedades que se implementarán en el sistema DRACO. En estos requisitos se pueden ver de forma más detallada los objetivos descritos al principio del trabajo para la mejora del sistema.

3.1.2 Ámbito del sistema

El sistema DRACO es un sitio web utilizado por las asignaturas de Procesadores de Lenguajes y Traductores de Lenguajes en la ETSIINF. Se busca poder ampliar el acceso a la práctica de la asignatura para la optativa Traductores de Lenguajes.

DRACO es un sistema funcional que lleva varios años usándose para estas asignaturas y se caracteriza por tener un módulo de actividades, donde los alumnos pueden realizar actividades publicadas por el profesor. Y por otro lado tiene el módulo de prácticas, donde los alumnos pueden realizar las comprobaciones pertinentes de los analizadores desarrollados en sus prácticas.

Además, el sistema cuenta con la vista del alumno y la vista del profesor. En la vista del alumno se encuentran los módulos de actividades y prácticas, un apartado para el perfil y más elementos relacionados con la gamificación. En la vista del profesor, se pueden configurar las prácticas y las actividades, así como ver estadísticas, configurar datos del sistema, etc.

3.1.3 Definición de abreviaturas

DRACO	Dinámica de Refuerzo del Aprendizaje de Compiladores
PDL	Procesadores de Lenguajes
TDL	Traductores de Lenguajes
HTML	Lenguaje de marcas de hipertexto (*Hyper-Text Markup Language*)
CSS	Hojas de estilo en cascada (*Cascading Style Sheets*)
PHP	Procesador de Hipertexto (*Hypertext Pre-Processor*)
WCAG	Pautas de Accesibilidad para el Contenido Web (*Web Content Accessibility Guidelines*)
ETSIINF	Escuela Técnica Superior de Ingenieros Informáticos
SQL	Lenguaje de Consulta Estructurada (*Structured Query Language*)
TFG	Trabajo de Fin de Grado
GCI	Generador de Código Intermedio
GCO	Generador de Código Objeto

3.2 Descripción general

Para este TFG se pueden dividir los requisitos del sistema en las siguientes secciones:

- Definición del GCI y del GCO: En este apartado, los grupos de alumnos pueden definir la configuración para su práctica.
- Comprobador del GCI y el GCO: En esta sección se van a definir los requisitos para que los alumnos puedan realizar las pruebas de los módulos del GCI y el GCO pasando por un comprobador.
- Configurador para el GCI y el GCO: Para la parte del profesor, se definirán los requisitos para que el administrador pueda configurar ambos módulos GCI y GCO de la práctica.
- Mantenimiento del sistema: Aquí se definen los requisitos de las modificaciones necesarias a realizar en el sistema DRACO para un mejor funcionamiento.

3.3 Especificación de requisitos

En este apartado se van a definir los requisitos para las secciones descritas en el apartado 3.2. También se han especificado algunos requisitos que están sujetos a cambios y el motivo correspondiente.

3.3.1 Requisitos funcionales

3.3.1.1 Definición del GCI

Requisito 1.1: Acceso a la definición del GCI. Los estudiantes podrán acceder a la definición del GCI desde el apartado de prácticas y desde el desplegable del apartado de prácticas en el inicio de DRACO.

Requisito 1.2: Descripción de la definición. Habrá una descripción extensa y muy clara de cómo definir los cuartetos y el formato de sus argumentos. Se incluye en esta descripción información sobre los puntos que se obtendrán, cómo definir los cuartetos y el funcionamiento de cómo funciona la definición. Deberá permitirse continuar con la definición o volver atrás.

Requisito 1.3: Cuartetos disponibles. Cuando el alumno entre a la definición, podrá ver la lista de cuartetos disponible y que se puedan utilizar. Para cada cuarteto, se podrá rellenar el nombre del operador de dicho cuarteto que va a usar el alumno en su práctica.

Requisito 1.4: Clases disponibles. Cuando el alumno entre a la definición, podrá ver una lista de las clases de argumentos (que se refieren al nombre de cada tipo de argumento), donde se podrán rellenar los nombres de las clases que ha usado el alumno en su práctica.

Requisito 1.5: Verificación de la definición. El sistema tiene que comprobar que las condiciones de definición del GCI se cumplen, tanto las condiciones de los cuartetos, como las condiciones de las clases.

> **Requisito 1.5.1**: El nombre del operador del cuarteto está formado por letras, dígitos y subrayados, siendo el primero siempre una letra; no se puede repetir el nombre del operador y no es necesario rellenar el nombre en todos los cuartetos.
>
> **Requisito 1.5.3**: El nombre de las clases está formado por letras, dígitos y subrayados, siendo el primero siempre una letra; se pueden repetir más de una vez, pero no puede haber una clase que se llame como un operador, y viceversa.

Requisito 1.6: Mensaje de error. Si las condiciones (Requisito 1.5) no se cumplen, el sistema no debe guardar ninguna información y debe mostrar un mensaje de error.

Requisito 1.7: Guardar definición para el grupo. Si no hay errores en la definición de nombres de operador y clases, cuando el usuario solicite guardar los nombres, el sistema almacenará en la base de datos los operadores elegidos para cada grupo que los haya definido, teniendo en cuenta el número de versión si hubiera otras definiciones previas que habían sido utilizadas en alguna comprobación.

Requisito 1.8: Mantener el formulario. Cuando se realice una definición incorrecta, la página web deberá guardar la información del formulario para que el alumno pueda corregir el error de forma más sencilla sin necesidad de volverlo a rellenar desde el principio.

Requisito 1.9: Obtención de puntos. El sistema debe otorgar los puntos correspondientes cuando se realiza la definición correcta del GCI por primera vez. Las siguientes veces no se otorgan puntos.

> **Requisito 1.9.1**: Si el alumno completa al menos 10 cuartetos y al menos 3 clases se le otorgan puntos.

Requisito 1.10: Intentos ilimitados. Una vez realizada la definición, se puede volver a modificar las veces que el alumno quiera.

Requisito 1.11: Versión de la definición. Cada vez que un alumno realice una nueva definición, se guarda la nueva definición como una nueva versión en la base de datos; de esta manera quedan registradas todas las definiciones realizadas por cada grupo, siempre que se hayan utilizado en alguna comprobación.

3.3.1.2 Definición del GCO

Requisito 2.1: Acceso a la definición del GCO. Los estudiantes podrán acceder a la definición del GCO desde el apartado de prácticas y desde el desplegable del apartado de prácticas en el inicio de DRACO.

Requisito 2.2: Descripción. Habrá una descripción sobre cómo realizar la definición del GCO.

Requisito 2.3: Pregunta en la definición. Cuando el alumno vaya a realizar una definición, se le realiza una pregunta sobre el orden de la pila y tendrá dos respuestas a elegir (ascendente o descendente). En el futuro se podrán añadir nuevas preguntas al alumno, que surgirán cuando esté desarrollado el módulo.

Requisito 2.4: Selección de respuesta. Una vez que el alumno haya elegido las respuestas, se guardarán en la base de datos para su respectivo grupo.

Requisito 2.5: Intentos ilimitados. El alumno puede cambiar el valor de la definición del GCO las veces que quiera.

Requisito 2.6: Obtención de puntos. El sistema debe otorgar los puntos correspondientes cuando se realiza la definición completa del GCO por primera vez. Las siguientes veces no se otorgan puntos.

3.3.1.3 Comprobador del GCI

Requisito 3.1: Ubicación del comprobador. Tanto en el apartado de prácticas como en el desplegable de dicho apartado en la página de inicio debe aparecer a los estudiantes el comprobador del GCI.

Requisito 3.2: Descripción del comprobador. Cuando un alumno acceda al comprobador, debe aparecer una descripción sobre los pasos a seguir para realizar la comprobación.

Requisito 3.3: Información de la prueba. Cuando se seleccione la opción de "Realizar prueba", aparecerá una pantalla con información sobre el máximo de puntos a obtener y el tiempo máximo que tiene el alumno para realizar la prueba una vez se descargue el fichero fuente.

Requisito 3.4: Entrega de la lista de cuartetos. El traductor realizado en la práctica por los alumnos deberá entregar obligatoriamente los resultados generados en un archivo de texto (lista de cuartetos).

Requisito 3.5: Entrega de los resultados. Se le dará al alumno un tiempo determinado para realizar la prueba con el fichero descargado. Deberá subir el fichero una vez generado por su práctica, enviándolo a DRACO.

Requisito 3.6: Generación de fichero fuente. Cuando se inicie la prueba, se genera un fichero fuente basado en las opciones de la práctica del grupo al que pertenezca el alumno. Este requisito y sus subrequisitos están sujetos a cambios (el corrector se está implementando en otros Trabajos de Fin de Estudios).

> **Requisito 3.6.1**: DRACO evaluará el fichero de cuartetos e informará al alumno si el fichero es correcto y corresponde con una traducción válida del fichero fuente.
>
> **Requisito 3.6.2**: En caso de detectarse algún error en el fichero o en la traducción, se informará al alumno de los errores detectados.
>
> **Requisito 3.6.3**: DRACO dará los puntos correspondientes, según la configuración establecida por el profesor.

3.3.1.4 Comprobador del GCO

Requisito 4.1: Ubicación del comprobador. Tanto en el apartado de prácticas como en el desplegable de dicho apartado en la página de inicio debe aparecer el comprobador del GCO.

Requisito 4.2: Descripción. Cuando un alumno acceda al comprobador, debe aparecer una descripción sobre los pasos a seguir para realizar la comprobación.

Requisito 4.3: Información de la prueba. Cuando se elija "Realizar prueba" se mostrará información sobre el máximo de puntos a obtener y el tiempo máximo que tiene el alumno para realizar la prueba una vez se seleccione "Descargar fichero".

Requisito 4.4: Entrega de los resultados. Se le dará al alumno un tiempo determinado para realizar la prueba con el fichero descargado. Deberá subir el fichero una vez generado por su práctica, enviándolo a DRACO.

Requisito 4.5: Generación de fichero fuente. Cuando se inicie la prueba, se genera un fichero fuente basado en las opciones de la práctica del grupo al que pertenezca el alumno. Este requisito y sus subrequisitos están sujetos a cambios (el corrector se implementará en otros Trabajos de Fin de Estudios).

> **Requisito 3.6.1**: DRACO evaluará el fichero de ensamblador e informará al alumno si el fichero es correcto y corresponde con una traducción válida del fichero fuente.
>
> **Requisito 3.6.2**: En caso de detectarse algún error en el fichero o en la traducción, se informará al alumno de los errores detectados.
>
> **Requisito 3.6.3**: DRACO dará los puntos correspondientes, según la configuración establecida por el profesor.

3.3.1.5 Configuración del GCI y GCO

Requisito 5.1: Ubicación de la configuración. El profesor podrá acceder a la configuración de la definición y el comprobador del GCI y GCO desde la sección de prácticas.

Requisito 5.2: Edición de la definición. Para la configuración de la definición del GCI y GCO se podrá editar datos para configurar los puntos máximos a otorgar por realizar la definición. También se puede definir la fecha y hora del inicio y el fin de cada fase de pruebas, el porcentaje de puntos que se otorgan y el nivel necesario para poder acceder a cada fase.

Requisito 5.3: Edición de la configuración. Para la configuración de la comprobación del GCI y GCO se podrán editar datos para configurar los puntos por realizar la prueba, las penalizaciones en función de los fallos detectados, el tiempo disponible para realizar la prueba, el tiempo de espera para poder repetir una prueba, los niveles de dificultad, etc. Se podrá también definir la fecha y hora del inicio y el fin de cada fase pruebas, el porcentaje de puntos que se otorgan y el nivel necesario para poder acceder a cada fase.

Requisitos 5.4: Apartados de la configuración. Para la comprobación del GCI y GCO se podrá configurar el módulo. Habrá un apartado de inicio, gestión de ficheros fuente, gestión de plantillas, comprobador de plantillas, consulta de datos de los grupos, estadísticas, configuración de datos y gestión del comprobador.

Requisito 5.5: Configurar datos. Se podrán configurar los datos del comprobador, dividido en los apartados: Tiempos correspondientes a pruebas, Puntuaciones y penalizaciones otorgadas a los alumnos por conseguir determinadas metas en cada prueba, Multiplicadores según la dificultad de las pruebas realizadas por los alumnos y número de intentos que tendrá cada grupo y Probabilidades de que se le asigne una prueba de una determinada dificultad (nivel de dificultad difícil, intermedia y fácil).

Requisito 5.6: Estadísticas. En la parte de estadísticas se guardará y mostrará el número del grupo, las pruebas correctas, las pruebas incorrectas, las pruebas no completadas y las pruebas totales.

Requisito 5.7: Exportación de datos. Al exportar los datos del comprobador, se exportará en formato *CSV* la misma información que la descrita en el requisito 5.6.

Requisito 5.8: Elección del lenguaje. Se podrá seleccionar el lenguaje que se desea administrar.

Requisito 5.9: Gestión de ficheros fuente. Se podrá subir al sistema uno o varios fragmentos. Aparecerá las plantillas en el sistema con la opción de mostrar ficheros para una plantilla. También se mostrará un listado de los ficheros en el sistema y se podrán filtrar por: Inicio, Intermedio y Final. Por último, habrá un listado de ficheros obsoletos con la opción de descargar.

Requisito 5.10: Gestión de plantillas. Habrá una descripción sobre información de las plantillas. Se podrá crear una plantilla completando un formulario, o también se podrán importar dichas plantillas. También habrá un listado de las plantillas que hay en el sistema.

> **Requisito 5.10.1**: En el formulario de la plantilla, se rellenará un nombre, número de campos, número de campos como máximo, la dificultad y una descripción.

Requisito 5.10.2: Se puede seleccionar una plantilla ya creada que aparezca en el listado de plantillas para poder configurarla. Cuando se configura se puede modificar el formulario ya relleno descrito en el Requisito 5.10.1.

Requisito 5.11: Comprobar plantillas. El administrador podrá comprobar la plantilla. Una vez pulsado el botón de comprobar la plantilla, aparecerá una descripción de la comprobación y la posibilidad de elegir entre Comprobación rápida y Comprobación extendida. Deberá informarse de los problemas detectados.

Requisito 5.12: Gestión del comprobador. Se podrán exportar los datos del comprobador en formato CSV, borrar los datos de los ficheros y las plantillas, borrar ficheros obsoletos, descargar fragmentos fuentes del sistema, exportar información de los fragmentos de la base de datos y, por último, importar datos de los fragmentos fuente a la base de datos.

3.3.1.6 Tareas de mantenimiento

Requisito 6.1: Deberá incorporarse información adicional para mostrar en la página del perfil del alumno. Esta información será su nivel, los puntos y las monedas.

Requisito 6.2: A la hora de realizar una comprobación, si un alumno deja la comprobación a medias, tiene que quedar registrado en la base de datos que ha iniciado una comprobación que no ha terminado.

Requisito 6.3: Cada módulo de las prácticas (tanto de PDL como TDL) puede tener distintas fases activas y, dependiendo de la fecha, el porcentaje de puntuación del total de puntos obtenidos por completar dichos módulos es distinto. Se quiere incluir que cada fase requiera de un nivel mínimo del usuario para poder acceder a la actividad correspondiente.

Requisito 6.4: En la parte del profesor, cuando se quiere mirar la información de un usuario, no deben aparecer los *links* de las páginas del alumno, si no los tiene definidos.

Requisito 6.5: En la parte del administrador, en el resumen de las estadísticas individuales de cada grupo, para cuando una prueba no se ha completado, no debe de aparecer la opción de consultar los ficheros entregados por el alumno ni la de realizar prueba, únicamente la opción de generar el fichero de prueba.

3.3.2 Requisitos de interfaces externas

DRACO consulta la base de datos de grupos de prácticas establecido por el sistema de grupos de la asignatura.

4 Solución

4.1 Diseño

En este apartado se presenta el diseño para la resolución de los requisitos planteados en el capítulo anterior. Para ello se usarán DFD (Diagrama de Flujo de Datos).

4.1.1 Diseño de las Bases de Datos

En primer lugar, se mostrará el modelo E/R de las bases de datos como se ve en la Figura 17 para comprender la relación entre las tablas y la utilización de estas para la posterior implementación. Para DRACO hay múltiples bases de datos entre las cuales se incluyen una para cada módulo de cada asignatura. El siguiente diagrama corresponde al modelo E/R de la base de datos "draco_cgci" que se ha utilizado para la realización de este Trabajo de Fin de Grado. La BBDD correspondiente al módulo de GCO se llama "draco_cgco" y sigue el mismo modelo quitando la tabla "type_palabraRes_GCI".

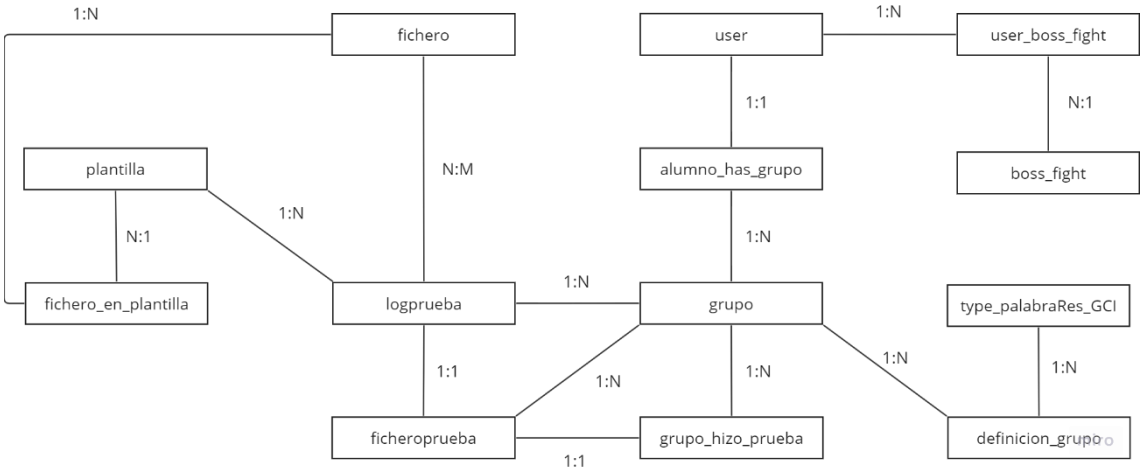

Figura 17. Modelo E/R de DRACO sobre las bases de datos utilizadas

4.1.2 Tablas de las Bases de Datos

A continuación, se explicarán de forma detallada las tablas de las bases de datos que hay actualmente en dentro del sistema que se han usado para el desarrollo de este trabajo.

La tabla 'user' registra todos los datos de cada uno de los alumnos en el curso.

Campo	**Descripción**
`enrollment_number`	Número de matrícula *(Primary key)*
`PIdN`	DNI, NIE o Pasaporte
`name`	Nombre
`surname`	Apellidos
`birth_date`	Fecha de nacimiento
`sex`	Género
`class_group`	Grupo de clase al que pertenece
`nickname`	Nombre de usuario en DRACO
`email`	Correo electrónico
`avatar`	Avatar, foto de perfil del alumno

Campo	Descripción
points	Puntos
level	Nivel
betting_chips	Monedas
personal_web	Página web personal
facebook_account	Cuenta de Facebook
twitter_account	Twitter
password_hash	Hash de la contraseña del usuario
Status_id	Campo sin uso actual
Degree_code	Código del grado en el que está matriculado el alumno
consecutivedays	Número de días que el alumno ha entrado de forma consecutiva a Draco

La tabla 'alumno_has_grupo' registra la relación entre el alumno y su grupo.

Campo	Descripción
idAlumno_ag	Clave primaria del alumno en la tabla 'user'
IdGrupo_ag	Clave primaria del grupo en la tabla 'grupo'

La tabla 'grupo' contiene información sobre los intentos de cada grupo de alumnos.

Campo	Descripción
IdGrupo	Numero de grupo *(Primary key)*
Dificultad	Límite de dificultad de las pruebas que pueden realizar dicho grupo
Intentos	Intentos restantes del grupo para el módulo correspondiente
Superados	Intentos completados por el grupo en el tiempo de prueba establecido
IntentosGastados	Intentos totales realizados por el grupo

La tabla 'grupo_hizo_prueba' registra los datos de una prueba realizada por un grupo.

Campo	Descripción
IdGrupo	Número de grupo
IdPrueba	Identificador de prueba único *(Primary Key)*
Fecha	Fecha de cuando se realizó la prueba
FormaFichero	Versión de la definición de cuando se realizó la prueba
Puntuación	Puntos obtenidos por el grupo tras realizar la prueba
score	Puntuación de 0 a 10 obtenida en la prueba

La tabla 'logprueba' almacena un registro de las pruebas realizadas por cada grupo.

Campo	Descripción
IdLog	Identificador del *log (Primary Key)*
IdPrueba	Identificador de la prueba que se realizó

Campo	Descripción
IdGrupo	Número del grupo que realizó la prueba
IdPlantilla	Identificador de la plantilla que se utilizó para la generación del fichero
IdFichero	Identificador del fichero que se utilizó para la prueba
OpcionesElegidas	Indica las opciones elegidas dentro de cada fichero para formar el fichero fuente de la prueba

La tabla 'ficheroprueba' contiene registrados los ficheros entregados para la realización de una prueba

Campo	Descricpción
IdPrueba	Clave primaria de la tabla 'grupo_hizo_prueba'
IdGrupo	Número de grupo
FechaLimite	Fecha límite en la que los alumnos pueden realizar la prueba con el fichero
Ruta	Ruta del fichero en el sistema
IdLog	Identificador del *log* correspondiente al IdLog de la tabla 'logprueba'
Dificultad	Dificultad del fichero utilizado
Actual	Indicador que representa si es el último fichero generado para el grupo

La tabla 'fichero' registra la información de los fragmentos fuente que están en el sistema y que se usará para la generación del fichero

Campo	Descripción
Id	Identificador del fichero *(Primary Key)*
Nombre	Nombre del fichero correspondiente al fragmento
Tipo	Tipo de fragmento (Inicial, Intermedio, Final) e indica la posición que ocuparán en el fichero
VersionActual	Indicador de si se está utilizando la versión más reciente del
IdLenguaje	Identificador del lenguaje al que pertenece

La tabla 'plantilla' tiene registrados los campos necesarios para la generación de las plantillas

Campo	Descripción
IdPlantilla	Identificador de la plantilla generada *(Primary Key)*
Nombre	Nombre de la plantilla
NumCampos	Número mínimo de fragmentos que se utilizan en la plantilla
NumeroFicherosLibres	Número de ficheros que no tienen posición fija en la plantilla
Dificultad	Dificultad de la plantilla
FechaInicial	Fecha de inicio a partir de la cual se podrá usar dicha plantilla
FechaFinal	Fecha final a partir de la cual la plantilla dejará de estar disponible para su uso
Descripción	Descripción de la plantilla
Comprobada	Indica si la plantilla ha sido comprobada

Campo	Descripción
IdLenguaje	Identificador del lenguaje al que pertenece la plantilla

La tabla 'fichero_en_plantilla' relaciona el fichero generado con la plantilla

Campo	Descripción
IdFichero	Clave primaria del fichero de la tabla 'fichero'
IdPlantilla	Clave primaria de la plantilla de la tabla 'plantilla'
Posición	Posición que tiene la plantilla en el fichero

La tabla 'boss_fight' contiene información sobre los distintos módulos de las prácticas de PDL y TDL.

Campo	Descripción
id	Identificador del módulo *(Primary Key)*
type	Tipo del módulo relativo a la práctica
status	Estado actual del módulo
title	Nombre descriptivo del módulo
req_level	Nivel mínimo necesario para acceder a cada fase del módulo
idate	Fecha de inicio de la fase del módulo
ddate	Fecha de fin de la fase del módulo
minpoints	Puntos mínimos a obtener cuando se completa el módulo en la fase
maxpoints	Puntos máximos a obtener cuando se completa el módulo en la fase
percentage	Porcentaje aplicado a los puntos obtenidos en la fase del módulo

La tabla 'user_boss_fight' tiene registrada la actividad de los alumnos cuando realizan alguna comprobación o definición de los módulos de la práctica.

Campo	Descripción
User_enrolment_number	Clave primaria de la tabla 'user'
Boss_id	Clave primaria de la tabla 'boss_fight'
datedone	Fecha en la que se realizó dicha tarea
points_award	Puntos otorgados al alumno tras realizar la tarea
test_id	Identificador de la tarea que se realizó
doneby	Matricula del alumno que realizo la tarea

La tabla 'type_palabraRes_GCI' define e identifica las palabras reservadas establecidas de los cuartetos y las clases de argumentos disponibles en el sistema.

Campo	Descripción
palRes_id	Identifica de forma única cada palabra (*Primary key*)
palRes_name	Descripción usada por el sistema para los nombres de las palabras reservadas
esOperador	Determina si es un operador o una clase de argumento del cuarteto

La tabla 'definicion_grupo' de la base 'draco_cgci' almacena la información de la definición de los *tokens* de los cuartetos y las clases de argumentos realizada por los grupos de prácticas en el sistema web DRACO.

Campo	Descripción
id_definicion	Identifica el registro de forma única (*Primary Key*)
palRes_id	Hace referencia al campo palRes_id de la tabla 'type_palabraRes_GCI'
idGrupo	Identifica el grupo de prácticas que realiza la definición
lexemaGrupo	Almacena la palabra reservada que utilizará el grupo
versionDefinicion	Número que identifica el número de versión de la definición de palabras reservadas para el GCI por un grupo

La tabla 'definicion_grupo' de la base 'draco_cgco' almacena la información de la definición del tipo de análisis escogida por los grupos de prácticas en el sistema web DRACO.

Campo	Descripción
id_definicion	Identifica el registro de forma única (*Primary Key*)
idGrupo	Identifica el grupo de prácticas que realiza la definición
tipo_definicion	Almacena el tipo de análisis que ha escogido el grupo
versionDefinicion	Número que identifica el número de versión de la definición

4.1.1 Definición del GCI

A continuación, se muestra en la Figura 18 un diagrama de flujo de datos sobre la realización de la definición del GCI que tratan los requisitos desde 1.1 hasta el 1.11.

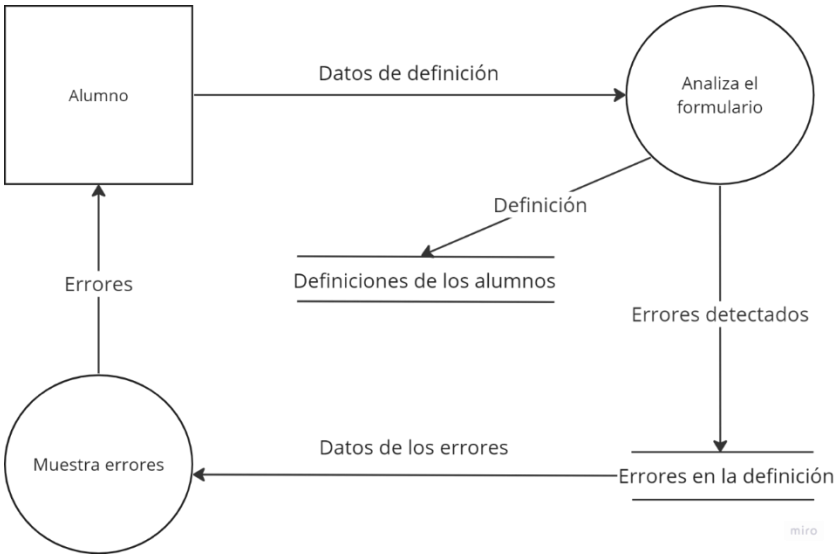

Figura 18. DFD nivel 0 para la definición de GCI

El proceso de realizar una definición se hace mediante los datos que introduce el alumno en el formulario y su posterior tratamiento de errores.

Antes de que el alumno vaya a realizar una definición, el administrador debe haber configurado qué operadores y clases se van a poder definir para cada grupo.

Posteriormente, el sistema muestra toda la información y datos guardados por el administrador y despliega todas las posibles opciones que se pueden definir por el grupo.

- Analiza el formulario. Una vez el alumno completa el formulario y quiere guardar la definición, se realiza un análisis para cada campo introducido

y se cumplen los requisitos definidos por la asignatura de TDL para realizar la práctica correspondiente.

- Guarda la definición para el grupo. Si no se detectan errores tras realizar el análisis, se guarda la definición para el grupo. Para poder guardar la definición, primero hay que realizar una comprobación previa: comprobar si se ha realizado alguna prueba si el alumno ya tenía alguna definición hecha previamente. En caso de que no se haya hecho ninguna prueba teniendo ya una definición hecha, se borra la actual y se pone la nueva, sin cambiar el número de versión. En caso de que sí se haya realizado alguna prueba con la definición actual, se guarda la nueva definición con un numero de versión superior, y para las siguientes pruebas se utilizaría la que tenga el número más alto de versión. Posteriormente, se otorgan puntos a todos los miembros del grupo la primera vez que se realiza la definición; el resto de las veces que se quiera cambiar dicha definición no se otorgan puntos.
- Muestra errores. Cuando el alumno quiera guardar la definición y existan errores en el análisis de los campos introducidos, se guardarán en una lista todos los errores. Después se imprimirán los errores en la página. Estos errores indicarán qué campo está mal y su correspondiente causa. Aparte de mostrar los errores, se guarda el formulario en la página para que el alumno no pierda los datos introducidos y los tenga que rellenar de nuevo.

4.1.2 Definición del GCO

A continuación, se muestra en la Figura 19 un diagrama de flujo de datos sobre la realización de la definición del GCI que tratan los requisitos desde 2.1 hasta el 2.6.

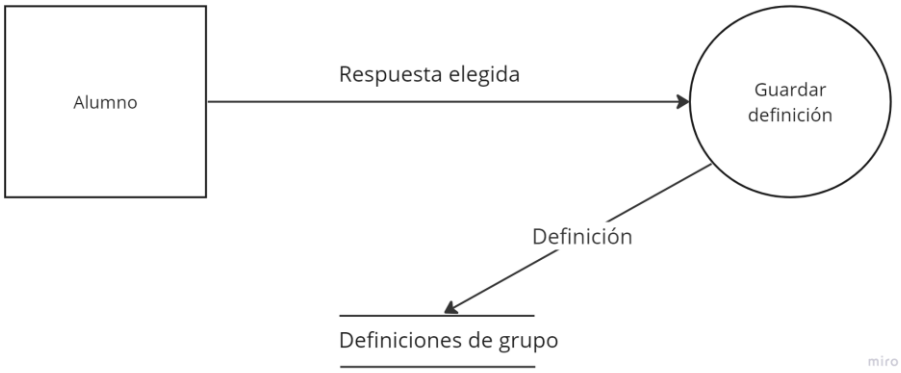

Figura 19. DFD 0 para la definición de GCO

El proceso para realizar la definición del GCO se hace respondiendo al formulario.

Se muestra al alumno que vaya a realizar la definición las dos opciones para que escoja el tipo que quiera. La selección será única, es decir, que no se podrán seleccionar las dos a la vez.

- Guardar definición. Una vez el alumno haya realizado la selección, ya podrá guardar su definición. Antes de guardar la definición se comprueba si se ha hecho alguna prueba con la definición anterior, si la había. En caso de que no se haya hecho dicha prueba, se elimina la definición anterior y se registra la nueva en la Base de Datos. En caso de que sí se haya realizado una prueba, se aumenta el número de versión de la

definición, y para futuras pruebas, se utilizará la definición con el mayor número de versión. Posteriormente se otorgan puntos a cada alumno pertenecientes al grupo del alumno que realizó la definición. Solo se otorgan puntos la primera vez que se realiza y se pueden hacer definiciones ilimitadas.

4.1.3 Comprobador del GCI

Para los requisitos desde el 3.4 al 3.6 se ha diseñado un DFD de introducción sobre el funcionamiento general del comprobador del GCI en la Figura 20.

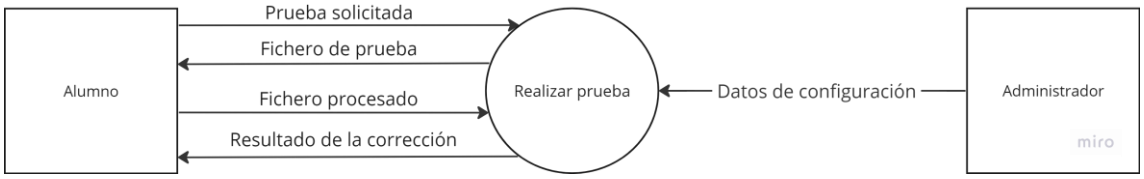

Figura 20. DFD de nivel 0 sobre el comprobador de GCI

Posteriormente, se introduce un DFD de nivel 1 en la Figura 21 donde se expone de forma más específica el funcionamiento del comprobador.

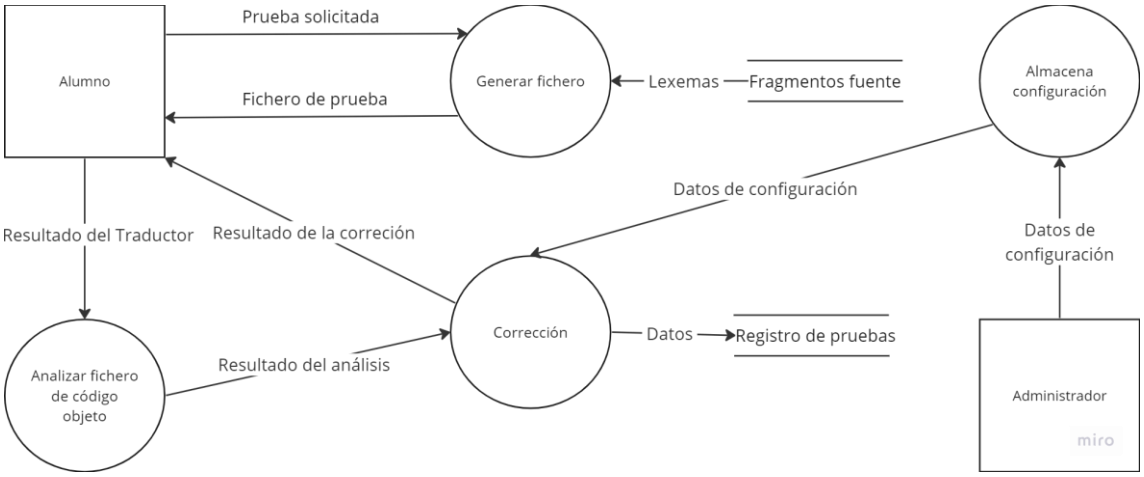

Figura 21. DFD de nivel 1 sobre el comprobador de GCI

- Generar fichero. El alumno solicita iniciar una comprobación y el sistema le entrega un fichero con el programa que el grupo tiene que traducir con su Traductor. Dicho fichero se genera con los fragmentos fuente que el administrador introdujo previamente en el sistema.
- Almacena configuración. Obtiene los parámetros necesarios para realizar la corrección y otorgar la puntuación necesaria a cada grupo que realizó su correspondiente prueba.
- Corrección. Aquí se realiza el proceso de corrección donde se obtiene el resultado del análisis realizado por el comprobador y envía dicho resultado al alumno. Dicho resultado contiene, si los hay, los errores que tenía el código intermedio generado por su Traductor, y la puntuación obtenida por el grupo. La puntuación otorgada al grupo se obtiene de los parámetros de configuración que se ha definido con antelación por parte del administrador. Finalmente se recogen los datos de la corrección, y se quedan registrados como un *log*.

4.1.4 Comprobador del GCO

Para los requisitos desde el 4.4 al 4.6 se ha diseñado un DFD de introducción sobre el funcionamiento general del comprobador del GCO en la

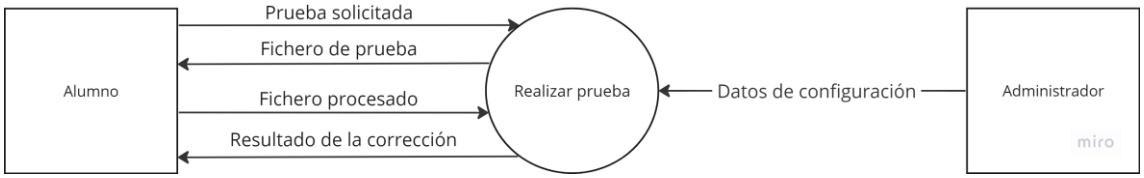

Figura 22. DFD de nivel 0 sobre el comprobador de GCO

Posteriormente, se introduce un DFD de nivel 1 en la Figura 23 donde se expone de forma más específica el funcionamiento del comprobador.

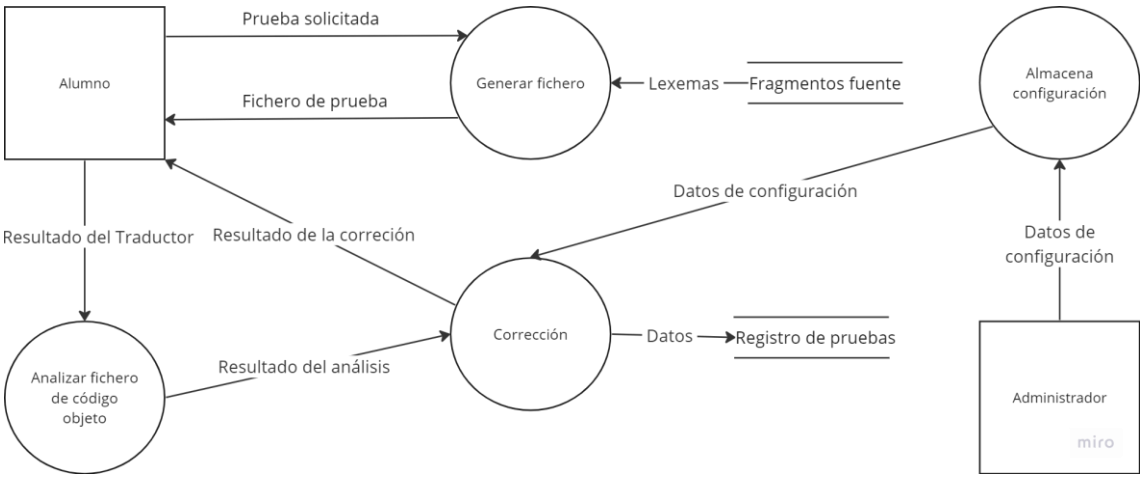

Figura 23. DFD de nivel 1 sobre el comprobador de GCO

- Generar fichero. El alumno solicita iniciar una comprobación y el sistema le entrega un fichero con el programa que el grupo tiene que traducir con su Traductor. Dicho fichero se genera con los fragmentos fuente que el administrador introdujo previamente en el sistema.
- Almacena configuración. Obtiene los parámetros necesarios para realizar la corrección y para otorgar la puntuación necesaria a cada grupo que realizó su correspondiente prueba.
- Corrección. Aquí se realiza el proceso de corrección donde se obtiene el resultado del análisis realizado por el comprobador y envía dicho resultado al alumno. Dicho resultado contiene, si los hay, los errores que tenía el código objeto generado por su Traductor, y la puntuación obtenida por el grupo. La puntuación otorgada al grupo se obtiene de los parámetros de configuración que se ha definido con antelación por parte del administrador. Finalmente se recogen datos de la corrección, y se quedan registrados como un *log* donde se recoge toda la información importante acerca de la prueba.

4.1.5 Configuración del GCI y GCO

Para la configuración del GCI se va a dividir el diseño en varios apartados.

4.1.5.1 Configurar datos

En el proceso de configurar datos, el administrador configura los parámetros para la realización de la práctica y se ha diseñado un DFD que se puede ver en la Figura 24.

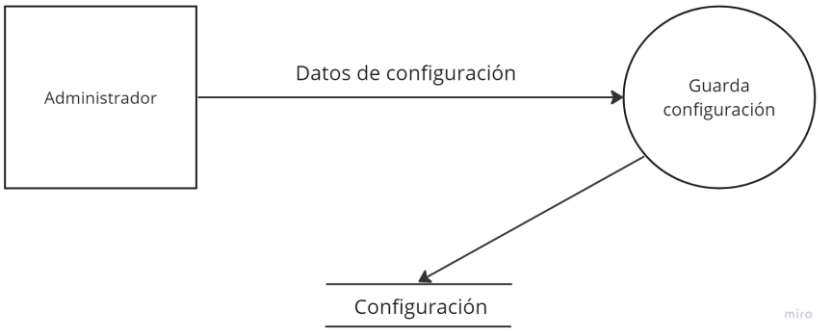

Figura 24. DFD de nivel 0 sobre la configuración de GCI y GCO

- Guarda configuración. Se guardan los datos introducidos mediante un formulario en un fichero de configuración. Este fichero contiene los siguientes parámetros:
  - Tiempo límite
  - Tiempo dependiendo del tipo de prueba
  - Tiempo entre pruebas
  - Puntos que se otorgan
  - Multiplicadores
  - Intentos
  - Probabilidades
  - Mínimo de pruebas

4.1.3.2 Gestión del comprobador

El administrador tiene un apartado para la gestión del comprobador y se ha diseñado un DFD en la Figura 25 con las posibilidades que tendrá esta parte de la configuración.

- Exportar datos del comprobador. El administrador tendrá la opción de exportar los datos de los grupos que han realizado pruebas en dicho comprobador. Estos datos se descargan en un fichero CSV para el administrador cuyo contenido incluye las pruebas completadas, las pruebas erróneas, las pruebas no completadas y las pruebas totales realizadas por cada grupo.
- Borrar datos. Se da la opción de que el administrador pueda borrar los datos de los ficheros y las plantillas. Se elimina toda la información de los *logs,* los ficheros fuente y las plantillas. También se vacían los directorios que contienen los fragmentos fuente del profesor, los fragmentos obsoletos y los ficheros temporales.
- Borrar ficheros obsoletos. Por otro lado, se ofrece la opción de únicamente borrar los ficheros obsoletos.
- Exporta fragmentos del sistema. El administrador podrá descargar en un archivo zip todos los fragmentos fuente actuales, excluyendo los obsoletos, que contenga el sistema. Estos fragmentos se obtienen de la tabla “fichero” donde la versión actual del fichero sea la 1.
- Exporta información de fragmentos. También el administrador podrá descargar en un archivo SQL las tablas de datos relacionadas con los

fragmentos fuente creador por el profesor. En dicho proceso se exportan las tablas “fichero”, “plantilla” y “fichero_en_plantilla” del módulo correspondiente.

- Importar datos. Por último, el administrador también podrá importar un fichero SQL con los datos de las tablas relacionadas con los fragmentos fuente. Antes de importar se deben eliminar los registros que han podido quedar previamente en la tabla para posteriormente poder importar los nuevos datos mediante el fichero SQL.

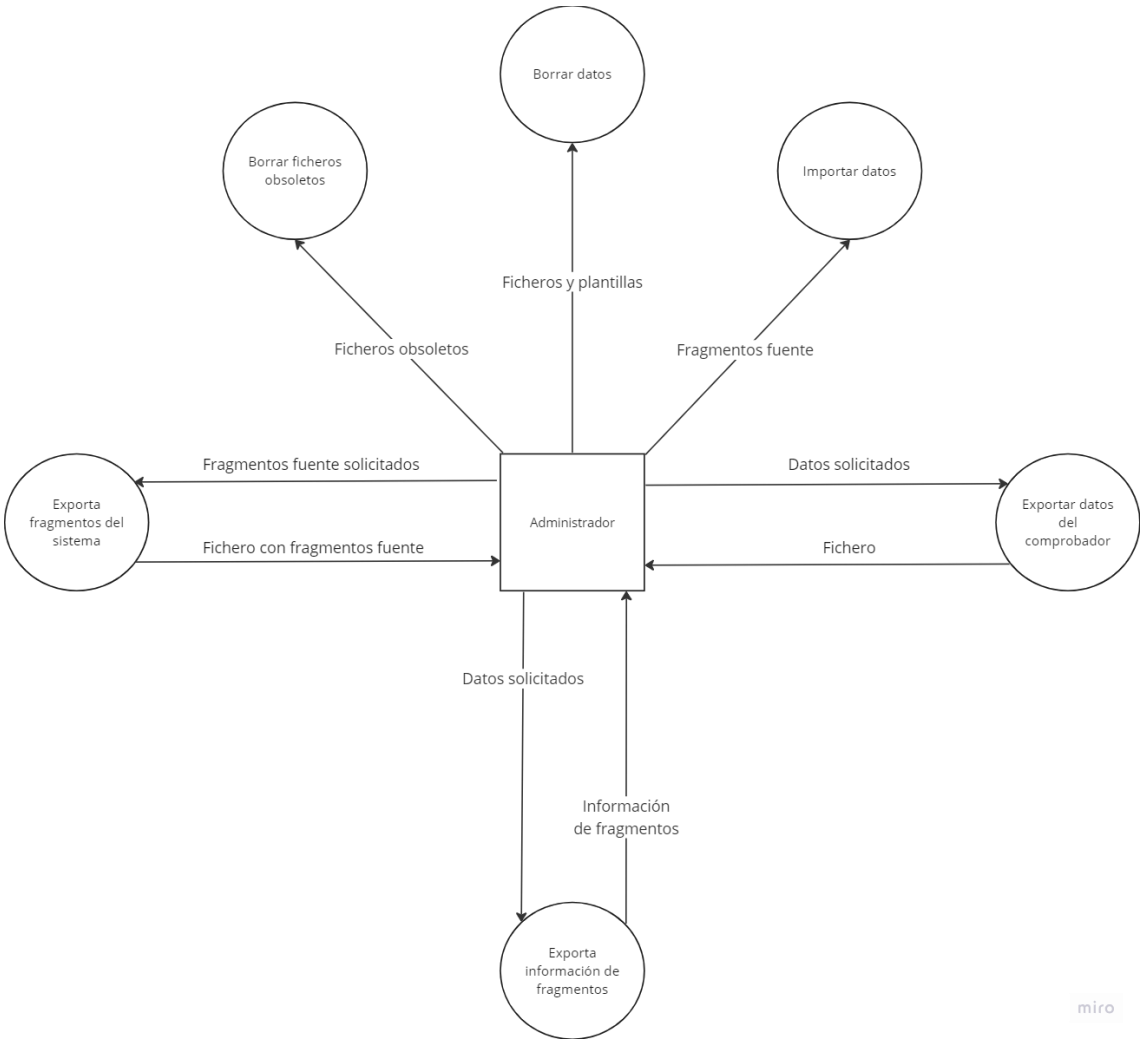

Figura 25. DFD sobre la gestión del comprobador de GCI y GCO

4.1.3.3 Estadísticas

Se ha diseñado un DFD mostrado en la Figura 26 que cubre los requisitos 5.6 y 5.7 sobre la consulta de estadísticas del comprobador.

Figura 26. DFD acerca de la consulta de estadísticas

- Consultar estadísticas. Cuando el administrador solicite ver las estadísticas de las comprobaciones del módulo de la práctica, se

mostrarán los datos en forma de tabla y diagrama circular mostrando las pruebas correctas, erróneas, no realizadas y las totales. Estos datos se recogen de la tabla “grupo_hizo_prueba”. Además, debajo de las gráficas, aparecerá una tabla con dichos datos de cada grupo. Al seleccionar un grupo, aparecerá un diagrama de circular sobre las pruebas realizadas por cada miembro del grupo, así como un resumen de dichas pruebas

- Exportar datos del comprobador. Se le ofrece al administrador la posibilidad de exportar los datos de las pruebas realizadas en dicho comprobador por todos los grupos. Se descargará un fichero CSV incluyendo los datos de las estadísticas indicadas en el apartado anterior para cada grupo.

4.1.3.3 Gestión de ficheros fuente y plantillas

Se ha diseñado un DFD inicial sobre le gestión de ficheros fuente en la Figura 27. Posteriormente, en la Figura 28 se muestra también la gestión y comprobación de plantillas que se debe hacer después del proceso de gestión de fragmentos fuente.

Figura 27. DFD de nivel 0 sobre la gestión de fragmentos fuente en GCI y GCO

En el DFD de la Figura 28 se define el siguiente nivel de gestión y comprobación de plantilla.

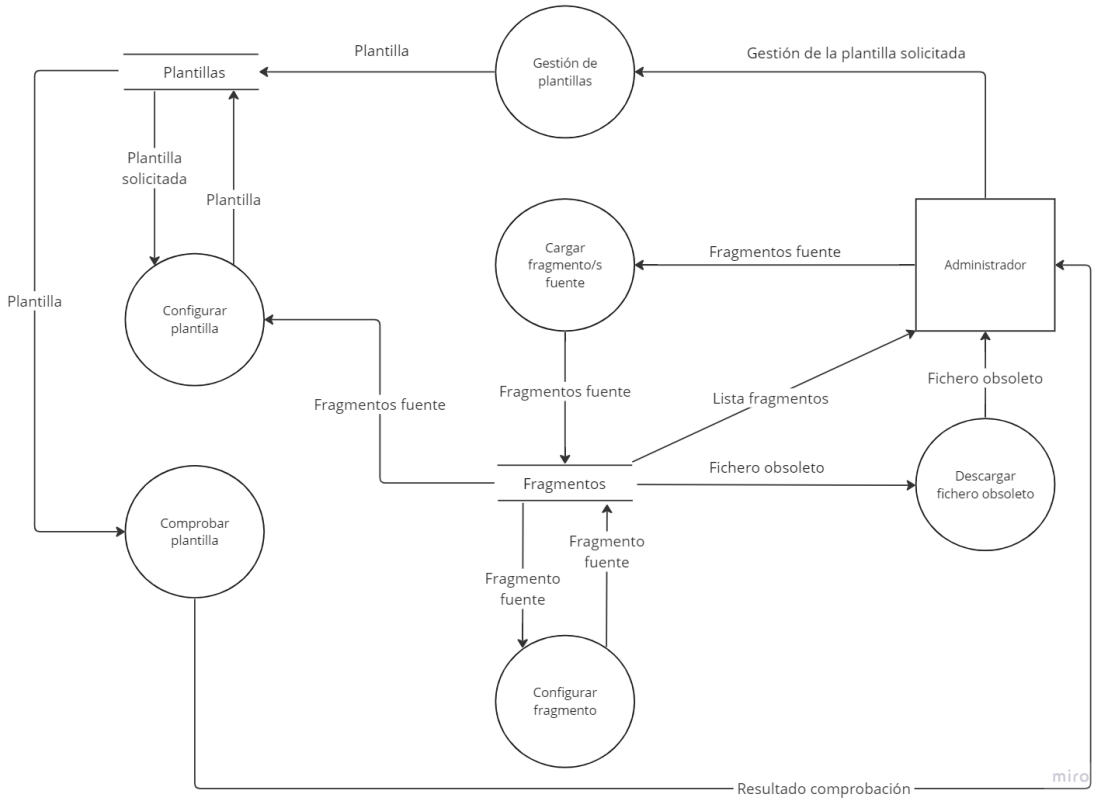

Figura 28. DFD de nivel 1 incluyendo la gestión y configuración de plantillas en GCI y GCO

- Cargar fragmentos fuente. El administrador puede introducir fragmentos fuente de forma individual, o varios fragmentos mediante un archivo ZIP. Los fragmentos fuente son ficheros que contienen pequeños fragmentos de código y se combinan entre sí para generar plantillas. Cuando se cargan en el sistema, para cada uno de los fragmentos se pregunta de qué tipo será según la posición en la plantilla que va a ocupar: Inicio, Intermedio o Final.
- Configurar fragmento. Una vez cargados los fragmentos al sistema, se puede seleccionar un fragmento concreto para poder configurarlo. Dicha configuración da la posibilidad de:
    - Indicar el tipo de fragmento que será
    - Darle una posición nueva dentro de una plantilla que ya está usando dicho fragmento
    - Eliminar fragmento de la plantilla que lo está utilizando
    - Añadir dicho fragmento a una plantilla ya existente
    - Añadir restricciones respecto al fragmento actual. Dichas restricciones indican qué fragmentos fuente y cuántas veces se deben repetir en la plantilla antes de que aparezca el fragmento actual.
    - Modificar el contenido del fragmento actual. Dicha modificación se realiza subiendo otro fragmento fuente.
    - Ver el fragmento en detalle
    - Eliminar el fragmento fuente del sistema
- Descargar fichero obsoleto. También se podrá descargar un fichero obsoleto seleccionado.
- El administrador también puede acceder a la gestión de plantillas. En dicha gestión podrá crear una plantilla o importarla mediante un zip.
    - Si la decide crear manualmente, deberá indicar el nombre, el número de campos, el número de campos como máximo y la dificultad de la plantilla. De forma opcional puede añadir una descripción de dicha plantilla.
    - Por otro lado, también puede importar las plantillas mediante un archivo ZIP.
- Una vez hay alguna plantilla en el sistema, se puede configurar de forma individual. Dicha configuración dispone de:
    - Cambiar el nombre, el número de campos, el número máximo de campos, la dificultad y la descripción de dicha plantilla seleccionada.
    - Eliminar la plantilla del sistema.
    - Aparece una lista de los fragmentos fuente en el sistema para poder añadirlos a la plantilla indicando qué posición ocupará.
    - Aparece una lista sobre los fragmentos fuente disponibles en la plantilla, y, seleccionando el fichero de inicio, final y los ficheros intermedios deseados, el administrador tendrá la posibilidad de ver dicha combinación de los ficheros seleccionados.
- También al seleccionar una plantilla, el administrador puede realizar dos comprobaciones sobre dicha plantilla. Una comprobación rápida o extendida. Si, en alguna plantilla hay algún error, se notificará al administrador.

4.1.6 Tareas de mantenimiento

En este apartado, se van a mostrar las tareas de mantenimiento efectuadas durante la realización de este TFG.

Tarea 1. Información adicional en el perfil del alumno.

Los alumnos disponen de un apartado de perfil dentro de DRACO. Dicho apartado de perfil contiene información acerca del alumno. Se ha querido añadir más información. Se ha decidido añadir los puntos, el nivel y las monedas correspondientes al alumno.

Tarea 2. Comprobación a medias.

Tanto en la asignatura de Procesadores de Lenguajes, como en la asignatura de Traductores de Lenguajes, los alumnos pueden acceder a los módulos de las prácticas y pueden realizar comprobaciones de cualquiera de los módulos de su práctica. Cuando se arranca una prueba, automáticamente se descarga el fichero para ser utilizado como entrada de la práctica del alumno. Posteriormente a la descarga, el alumno tiene un tiempo límite para poder realizar la entrega del fichero pasado por el Procesador o por el Traductor implementado por su grupo.

Si el fichero traducido no se entrega en el límite de tiempo establecido, no había manera de saber si el alumno había dejado la comprobación a medias. Se ha añadido un nuevo campo en la tabla "grupo_hizo_prueba" llamado *score*. Dicho campo calcula una puntuación tras una entrega, y se queda en `null` cuando la comprobación no se ha terminado.

Tarea 3. Niveles en las fases.

En la edición de datos de los módulos de los comprobadores de prácticas, tanto de PDL como de TDL, cada módulo puede tener distintas fases en diferentes fechas. Cada fase tiene asignado una fecha de inicio, una fecha de terminación, un porcentaje de puntos, que se aplica a los puntos configurados que se otorgan cuando se realiza una comprobación.

En la interfaz de la edición de datos aparecía un campo para configurar el nivel de la cada fase, pero no se estaba guardando en ningún lado, y, por tanto, no tenía utilidad, ya que solo afectaba al comprobador completo.

Se agregó el campo "req_level" en la tabla "boss_fight" que va a contener los niveles de cada fase. Este nivel es el nivel mínimo que ha de tener el alumno para poder realizar una comprobación en dicha fase.

Tarea 4. Páginas web del alumno.

Cuando el alumno accede a su perfil, puede configurar su página web personal, su cuenta de *Facebook* y su cuenta de *Twitter*. Ahora cuando el profesor quería ver el perfil del alumno desde la parte de administrador, y el alumno no tenía ninguno de esos enlaces configurados, aparecía un enlace para cada apartado que abría otra pestaña de la propia página; en este caso, se volvía a abrir el perfil del alumno en otra pestaña del navegador.

El sistema deberá comprobar que dichos campos están configurados para poder mostrar la URL correcta en su caso, o no mostrar nada en caso contrario.

Tarea 5. Estadísticas individuales de cada grupo.

En el diseño original, cuando un grupo realizaba una comprobación, esta quedaba registrada en la tabla "grupo_hizo_prueba". Sin embargo, si un grupo iniciaba una comprobación y no la finalizaba, dicha información no se mostraba en las estadísticas individuales de cada grupo en la parte del administrador. Debido a los resultados de la tarea 2, comenzó a aparecer información que no era útil para cuando las pruebas no habían sido completadas, como por ejemplo las opciones de "Tokens empleados" (en el comprobador léxico) y "Realizar prueba". Esto resultaba confuso y poco práctico, ya que no tenía sentido

mostrar opciones relacionadas con una prueba que el alumno no había finalizado. Por lo tanto, se implementó un mecanismo para detectar estos casos y omitir el acceso a dichas opciones, dejando únicamente el botón de generar fichero de prueba para que el administrador pudiera ver el fichero que se había utilizado en dicha comprobación.

4.2 Implementación

Para la realización de este Trabajo de Fin de Grado se han utilizado diversas tecnologías.

Para la programación y desarrollo del código se ha tenido que utilizar diversos lenguajes de programación en conjunto:

- HTML5 para estructurar el contenido de las páginas web
- PHP7 para gestionar la lógica del servidor y la interacción con la base de datos
- MySQL para almacenar y recuperar datos de manera eficiente de las tablas en la base de datos
- CSS3 para diseñar y estilizar la apariencia de las páginas web

Por otro lado, para desarrollar el código se ha utilizado Visual Studio Code [41] en la versión 1.89.1, que cuenta con soporte para los lenguajes utilizados.

XAMPP Server [42]ha sido necesario para crear un entorno local de desarrollo que incluye Apache, PHP y MySQL, facilitando pruebas y desarrollo. XAMPP en su versión 8.2.12 incluye PhpMyAdmin que gestiona y administra bases de datos MySQL de manera gráfica simplificando las tareas de creación, modificación, importación, exportación y consulta de datos.

Además, se ha utilizado GitLab [43] para gestionar el control de versiones del código fuente.

Para finalizar, se ha utilizado Xdebug [44], una herramienta de alta utilidad para poder depurar el código PHP. Para configurar Xdebug había que instalarse la extensión en Visual Studio Code y desde XAMPP, en la configuración “php.ini” dentro del módulo de Apache había que añadir el código que aparece en la Figura 29.

```
zend_extension = xdebug
xdebug.mode = debug
xdebug.start_with_request = yes
```

Figura 29. Código para utilizar Xdebug

Las definiciones y comprobadores de GCI y GCO tienen que aparecer cuando la asignatura de TDL está activa como se ve en la Figura 30. Estos módulos se caracterizan por ser de dicha asignatura por el campo “curso” en la tabla “boss_fight”. De esta manera en el apartado de prácticas solo aparecerán los módulos de GCI y GCO. Para implementar esto, se ha utilizado una variable global dentro de los ficheros PHP llamada “course” que representa la asignatura que se tratará dependiendo de cuál está en curso. Dicha asignatura se configura desde la vista del administrador, en la sección de configuración que corresponde con el fichero “configuration_database.php”. Una vez seleccionada la asignatura en curso, se utiliza la variable global mencionada anteriormente para separar los módulos de Traductores de Lenguajes de los módulos de Procesadores de Lenguajes. Dicha comprobación se realiza en el fichero “Index.php” que redirecciona dependiendo del valor que tenga la variable, a los ficheros correspondientes a GCI y GCO, o a los módulos de la parte de PDL.

Figura 30. Pantalla de inicio de prácticas de TDL

4.2.1 Definición del GCI

Para la parte de la definición del GCI se han tenido que definir dos tablas de datos nuevas: “type_palabrares_gci” y “definicion_grupo”. En la primera tabla están los lexemas definidos por el profesor y en la segunda, se guardan las definiciones de cada grupo. Cuando se accede a la definición del GCI aparece una lista con los operadores y las clases a definir como se ve en la Figura 31 y en la Figura 32. De esta forma se utilizan dos ficheros: “DefinirOperadores.php” y “DefinirOperadoresAction.php”. La primera es la que se encarga de mostrar la interfaz de la definición y la segunda, se encarga de operar con la base de datos para guardar y mostrar los datos para cada grupo.

Figura 31. Lista de cuartetos en la definición de GCI

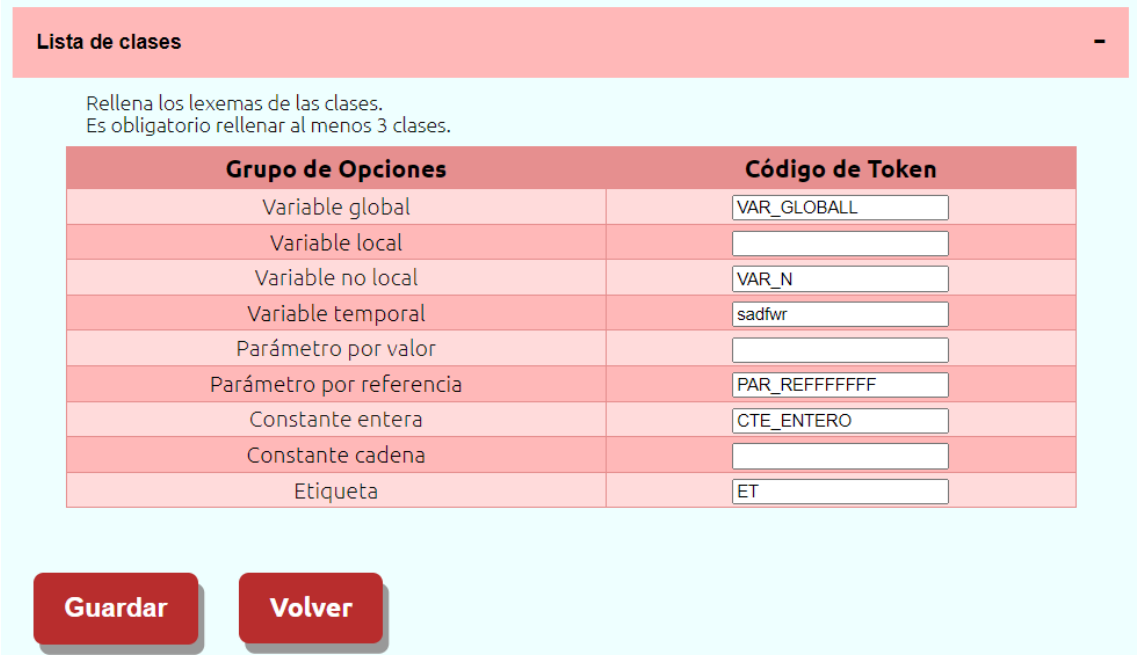

Figura 32. Lista de clases en la definición de GCI

El fichero "DefinirOperadoresAction.php" realiza varias comprobaciones a la hora de guardar la definición para cada grupo. Primero, comprueba que cada operador tiene el formato adecuado y, posteriormente, comprueba que se ha rellenado el número mínimo de operadores y de clases que exige la práctica. A pesar, de que el alumno no haya definido el número mínimo de operadores y clases, sí que permite guardar la definición para el grupo dentro de la base de datos en la tabla "definición_grupo", pero no otorga ningún punto, y tampoco le permite realizar comprobaciones del GCI. Finalmente, para guardar la definición, se mira la tabla de "grupo_hizo_prueba" para comprobar si se ha realizado alguna prueba con la definición actual del alumno (si la había). En caso de que se haya realizado una prueba, se guarda la nueva definición con un nuevo número de versión. En caso de que no se haya realizado ninguna prueba con la definición actual, no tiene sentido mantener dicha definición, y, por tanto, se borra de la tabla y se inserta la nueva definición con el mismo número de versión que había anteriormente.

Si al querer guardar la definición existe algún error, no se realizará ninguna acción de modificación en la base de datos, y se mostrarán los errores como se ven en la Figura 33 al inicio de la página guardando cada error en un *array* llamado "errores", y cuya implementación se muestra en la Figura 34.

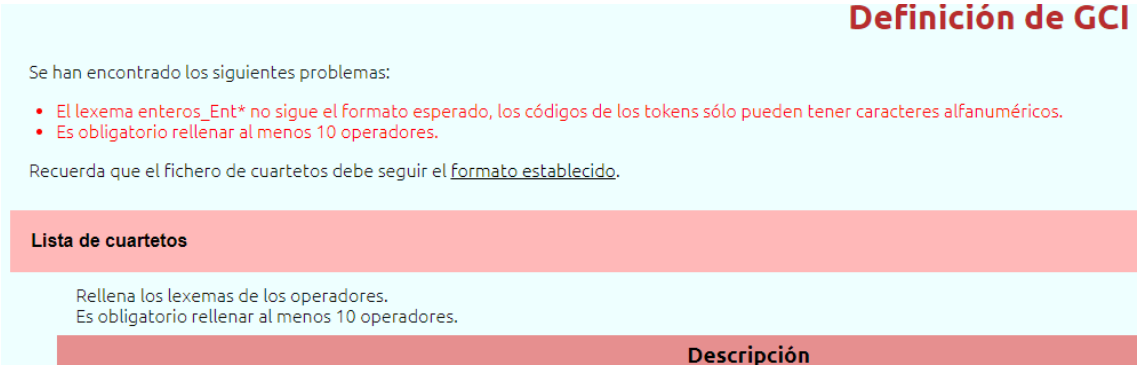

Figura 33. Mostrar errores en la definición

```
if(empty($errores)){
        echo "<p class='ok'>Se dio de alta correctamente la configuración del
             lenguaje.</p>";

        //Damos los puntos en caso de que fuera la primera vez que se guarda
        //la configuración
        $sql_query = "SELECT `maxpoints` FROM `boss_fight` WHERE `id`= 8;";
        $query_result = mysqli_query($db_link, $sql_query);
        $row = mysqli_fetch_array($query_result);
        $puntos = round($row['maxpoints']*$_SESSION['per_confgci']);
        $sql_query = "SELECT * FROM `user_boss_fight` WHERE `Boss_id`= 8 AND
                     `User_enrolment_number`='$user_id';";
        $query_result = mysqli_query($db_link, $sql_query);
        if(mysqli_num_rows($query_result)==0)
                echo "<p class='ok'>Al ser la primera vez que se guarda la
                     configuración de tu grupo de prácticas habéis obtenido
                     ".$puntos." puntos.</p>";
        darPuntosGrupo($puntos, 8, $idGrupoPracticas, 0, $user_id);
        disconnect_db($db_link);

        echo "<br><label for='volver' title='volver' class='sr-only'>
              Volver</label>";
        $indexDir = isset($_GET['CAL']) ? '../GCI/Index.php' :
        '../Index.php';
        echo "<a href='$indexDir'><input class='button-practicas' id='volver'
             type='button' name='volver' value='Volver'></a>";
}
```

Figura 34. Tratamiento de la definición al querer guardar

También es importante recalcar que, si el alumno comete errores en la definición, se mantiene el formulario con los campos que había completado, para que no haya necesidad de que el alumno lo tenga que volver a rellenar, y resulte más fácil corregir los errores. Cuando el alumno pulsa el botón de guardar la definición, el fichero "DefinirOperadoresAction.php" detecta que se ha pulsado el botón y ejecuta sus acciones necesarias con el formulario. En el caso de que se detecten errores, se guardan en un *array* llamado 'errores' y se realizan las operaciones pertinentes como se ve de forma simplificada en la Figura 35.

Posteriormente, se añade una condición para comprobar si ha habido errores en el guardado, para poder mostrar de nuevo el formulario, con los datos que había rellenado el alumno, sin perderlos como se puede ver en la Figura 36.

```
if(empty($errores)){
echo "<p class='ok'>Se dio de alta correctamente la configuración del
        lenguaje.</p>";
//Resto de acciones de guardado
}
else{
       echo "<p>Se han encontrado los siguientes problemas:</p>";
       echo "<ul class='ErrorAlumList'>";
       foreach ($errores as $error){
              echo "<li>$error</li>";
       }
       echo "</ul>";
}
```

Figura 35. Tratamiento de errores en el fichero "DefinirOperadoresAction.php"

```
//Contiene las acciones que se realizan al dar a guardar.
include 'DefinirOperadoresAction.php';

//Si se entra en la página o si ha habido errores
if (!isset($errores) || !empty($errores)) {
//Mostrar formulario
}
```

Figura 36. Mostrar formulario dentro del fichero "DefinirOperadores.php"

4.2.2 Definición del GCO

Para la definición de GCO se ha tenido que crear la tabla “definicion_grupo”. Esta tabla, como se ha visto anteriormente, contiene información sobre el tipo de pila que decide usar cada grupo para su práctica. De esta forma se utilizan los ficheros “DefinirSentido.php” y “DefinirSentidoAction.php”. El primero es el que se encarga de mostrar la interfaz de la definición, como se ve en la Figura 37, y, el segundo, se encarga de operar con la base de datos para guardar y mostrar los datos para cada grupo.

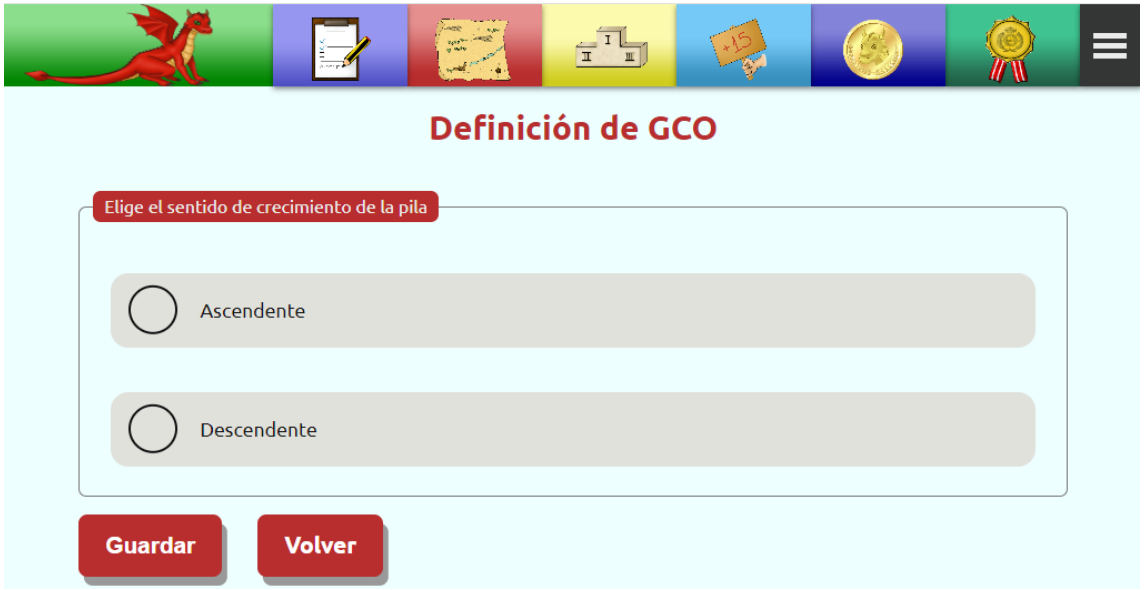

Figura 37. Cuestionario de la definición del GCO

El alumno podrá seleccionar únicamente una de las dos opciones para guardar su definición. Si no selecciona nada, no tiene ninguna acción. De la misma forma que en la definición del GCI, se realiza una comprobación antes de guardar la información en la base de datos. El sistema comprueba si el alumno ya ha realizado una comprobación con la definición actual; en caso de que no lo haya realizado, se elimina de la tabla dicho registro y se inserta el nuevo con el mismo número de versión que había anteriormente. En caso de que sí se haya realizado alguna comprobación, se añade una nueva entrada con el nuevo y un número de versión superior.

4.2.3 Comprobador del GCI

Se ha implementado la interfaz de acceso al comprobador del GCI y se ha dejado preparado para la futura implementación del comprobador. Cuando el alumno accede al comprobador se ha puesto una descripción de cómo funciona dicho comprobador y qué acciones son necesarias previas a la realización de la prueba como se ve en la Figura 38.

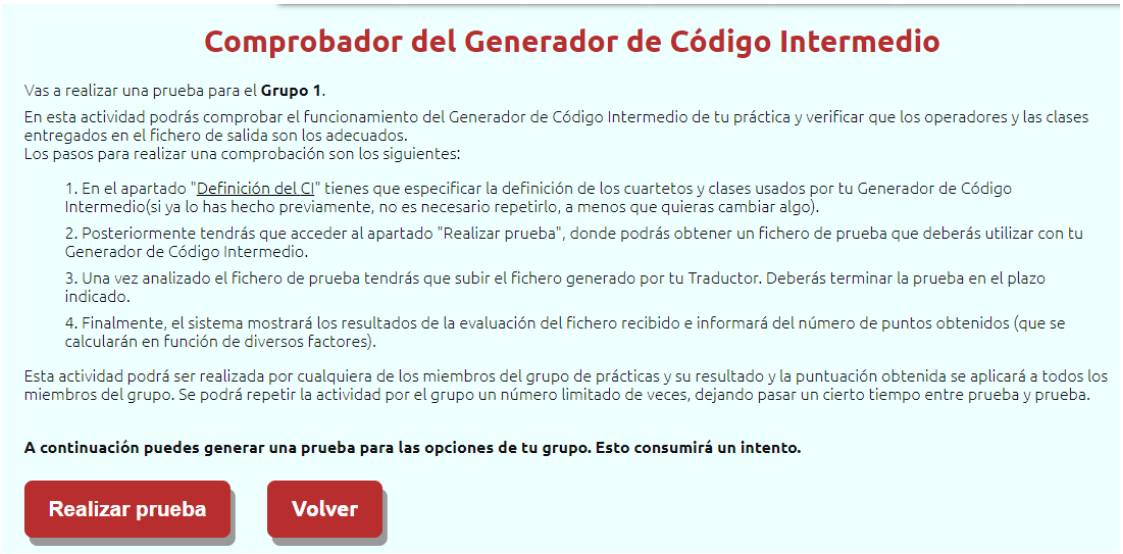

Figura 38. Descripción del comprobador de GCI

En caso de que la prueba no se haya entregado en el plazo de tiempo asignado, al volver a entrar al comprobador, aparecerá el mensaje de error mostrado en la Figura 39. Esto se ha hecho comprobando en la tabla "ficheroprueba" que la fecha límite sea menor que la actual.

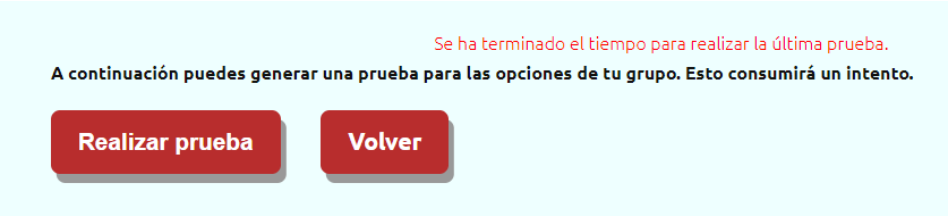

Figura 39. Tiempo límite excedido

Para poder realizar la prueba, se comprueba en la tabla "definicion_grupo" que el grupo al que pertenece el alumno tiene alguna definición hecha; si no la tiene, aparecerá un mensaje de error indicando que tiene que realizar la definición de GCI. En este mensaje de error se incluye un acceso para que el alumno pueda acceder de forma más rápida a ficha definición, como se puede ver en la Figura 40.

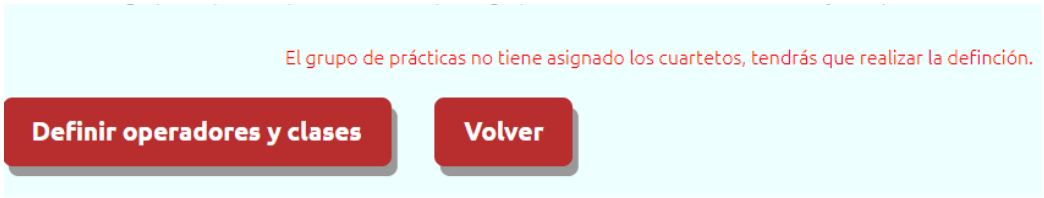

Figura 40. El alumno no tiene una definición realizada

Una vez el sistema detecte que el alumno tiene una definición hecha, se pasa a la siguiente pantalla pulsando el botón de "Realizar prueba" donde ya le aparece al alumno la posibilidad de poder descargar el fichero de prueba, como se ve en la Figura 41. En esta pantalla aparece una pequeña descripción sobre el tiempo límite que tiene el alumno para subir el fichero y la puntuación máxima que puede obtener. El tiempo se obtiene del fichero de "configuración.txt" ubicado en el sistema, y la puntuación se obtiene de la fase activa que configuró previamente el administrador.

Figura 41. Pantalla para descargar el fichero de prueba

Cuando el alumno pulsa el botón "Descargar fichero de prueba", lleva a la pantalla (Figura 42) del comprobador donde permite seleccionar un archivo de su sistema y subirlo. También se da la opción de volver a descargar el fichero y de regresar, por si el alumno necesita realizar alguna otra comprobación. Cuando vuelve a la comprobación, el tiempo ha seguido transcurriendo.

Figura 42. Pantalla para subir fichero al comprobador

Figura 43. Errores orientativos sobre cómo se verían los errores en el comprobador de GCI

Una vez se pulsa el botón “Subir archivo” se ejecutará el comprobador sobre el fichero que ha subido el alumno. Dicho comprobador, como se ha mencionado anteriormente, está a la espera de desarrollarse, pero aquí ya se ha implementado que se muestre como salida los posibles errores, en caso de que hubiera, e información sobre los puntos obtenidos para cada miembro del grupo como se puede ver en la Figura 43. Toda esta información queda registrada en la tabla “grupo_hizo_prueba” para dejar constancia sobre que pruebas ha realizado cada grupo en dicho módulo.

4.2.4 Comprobador del GCO

Se ha implementado la interfaz de acceso al comprobador del GCO y se ha dejado preparado para la futura implementación del comprobador. Cuando el alumno accede al comprobador se ha puesto una descripción de cómo funciona dicho comprobador y qué acciones son necesarias previas a la realización de la prueba como se ve en la Figura 44.

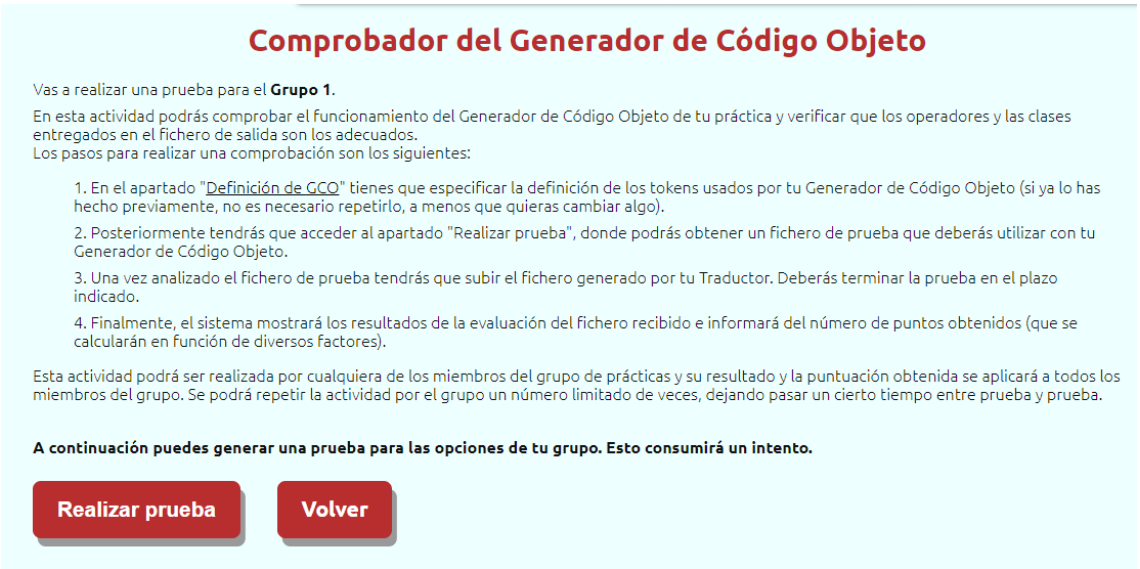

Figura 44. Descripción del comprobador de GCO

En caso de que la prueba no se haya entregado en el plazo de tiempo asignado, al volver a entrar al comprobador, aparecerá el mensaje de error mostrado en la Figura 45. Esto se ha hecho comprobando en la tabla “ficheroprueba” que la fecha límite sea menor que la actual.

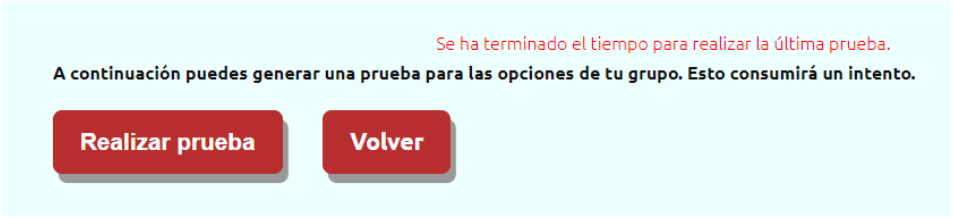

Figura 45. Tiempo límite excedido

Para poder realizar la prueba, se comprueba en la tabla “definicion_grupo” que el grupo al que pertenece el alumno tiene alguna definición hecha; si no la tiene, aparecerá un mensaje de error indicando que tiene que realizar la definición de GCO. En este mensaje de error se incluye un acceso para que el alumno pueda acceder de forma más rápida a ficha definición, como se puede ver en la Figura 46.

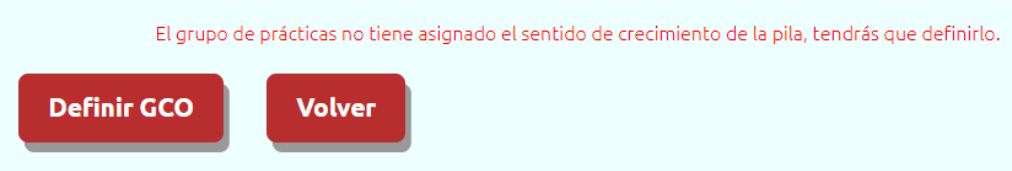

Figura 46. El alumno no tiene una definición realizada

Una vez el sistema detecte que el alumno tiene una definición hecha, se pasa a la siguiente pantalla pulsando el botón de "Realizar prueba" donde ya le aparece al alumno la posibilidad de poder descargar el fichero de prueba, como se ve en la Figura 47. En esta pantalla aparece una pequeña descripción sobre el tiempo límite que tiene el alumno para subir el fichero y la puntuación máxima que puede obtener. El tiempo se obtiene del fichero de "configuración.txt" ubicado en el sistema, y la puntuación se obtiene de la fase activa que configuró previamente el administrador.

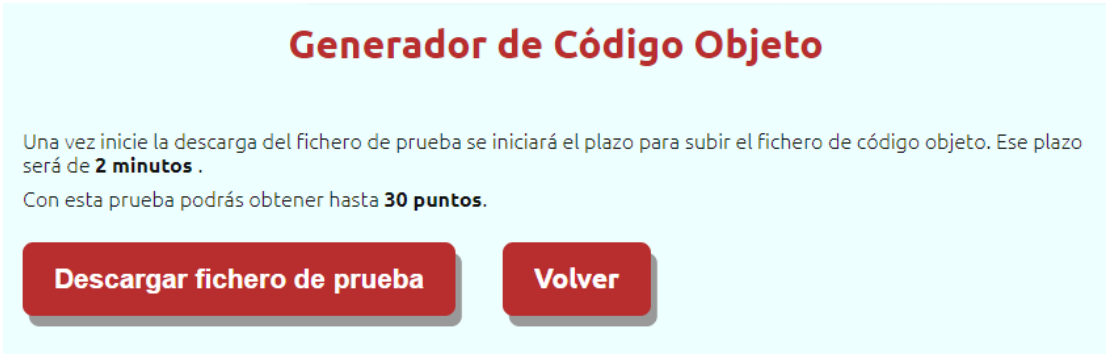

Figura 47. Pantalla para descargar el fichero de prueba

Cuando el alumno pulsa el botón "Descargar fichero de prueba", lleva a la pantalla (Figura 48) del comprobador donde permite seleccionar un archivo de su sistema y subirlo. También se da la opción de volver a descargar el fichero y de regresar, por si el alumno necesita realizar alguna otra comprobación. Cuando vuelve a la comprobación, el tiempo ha seguido transcurriendo.

Figura 48. Pantalla para subir fichero al comprobador

Una vez se pulsa el botón “Subir archivo” se ejecutará el comprobador sobre el fichero que ha subido el alumno. Dicho comprobador, como se ha mencionado anteriormente, está a la espera de desarrollarse, pero aquí ya se ha implementado que se muestre como salida los posibles errores, en caso de que hubiera, e información sobre los puntos obtenidos para cada miembro del grupo como se puede ver en la Figura 49. Toda esta información queda registrada en la tabla “grupo_hizo_prueba” para dejar constancia sobre que pruebas ha realizado cada grupo en dicho módulo.

Figura 49. Errores orientativos sobre cómo se verían los errores en el comprobador de GCO

4.2.5 Configurador del GCI y GCO

Para comenzar, se pueden editar los datos de las fases de cada módulo de la práctica. Añadiendo los datos correspondientes y guardando dicha información de cada fase dentro de la tabla “boss_fight”. Como la tabla solo tiene un registro por cada módulo, se ha decidido guardar la información de las fechas de inicio y fin de cada fase y el nivel mínimo para poder acceder a cada fase como en los módulos que hay en la asignatura de PDL. Es decir, que se guardan por ejemplo, para el campo ‘idate’ (fecha_inicio) como fecha1;fecha2;fecha3;etc. Así cada fecha le corresponde a la fase1, fase2, fase3, etc.

En las estadísticas de los comprobadores, se ha optado por seguir el mismo tipo de diagramas que en la asignatura de PDL, usando así un diagrama circular para los grupos y un diagrama de barras para las pruebas realizadas. Además, se ha añadido una tabla debajo de dichos diagramas indicando los datos específicos sobre las pruebas realizadas. Todo lo descrito anteriormente se puede observar en la Figura 50.

El administrador también puede gestionar los ficheros fuente y las plantillas con todas las opciones descritas anteriormente. Se ha implementado la parte de gestor del comprobador donde va a permitir al administrador exportar los datos del comprobador recogiendo los datos de la tabla “grupo_hizo_prueba” en un archivo CSV y exportar los fragmentos de la base de datos en un fichero SQL.

Por otro lado, también puede descargar los fragmentos en formato ZIP donde se incluyen todos los ficheros. Y, por último, se permite eliminar los datos de ficheros y eliminar ficheros obsoletos.

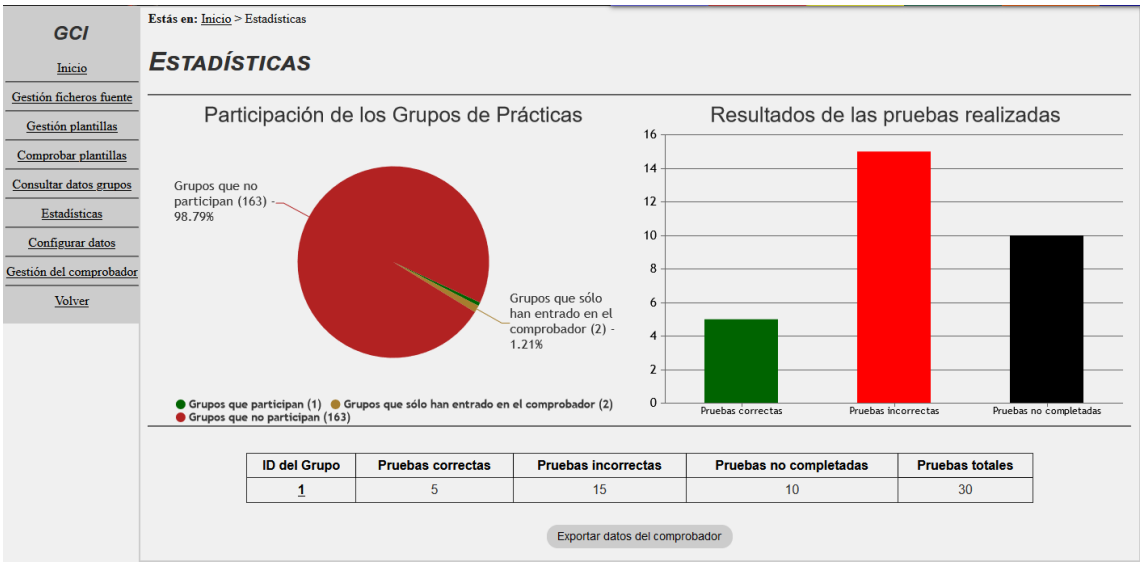

Figura 50. Estadísticas del comprobador de GCI

4.2.5 Tareas de mantenimiento

A continuación, se va a explicar la implementación de las tareas de mantenimiento que han tenido lugar en el sistema. Estas modificaciones afectan a las dos asignaturas, ya sea para Procesadores de Lenguajes como para Traductores de Lenguajes.

4.2.5.1 Perfil del alumno

Como se ha mencionado anteriormente, se ha añadido información adicional al mostrar el perfil del alumno dentro del sistema. Anteriormente no se mostraba dicha información, y se decidió mostrar al principio del perfil, la cantidad de puntos y monedas, y el nivel del usuario como se puede ver enmarcado en rojo en la Figura 51. Dicha información se obtiene de la tabla "user" en el fichero "get_alumno.php" para así poder añadirlo al fichero "Perfil.php". Para poder representar el nivel y las monedas se utiliza la función `floor()` para redondear dichos números decimales hacia abajo al entero más cercano, de esta manera se asegura que el alumno vea un número entero en dichos valores, en lugar de decimales.

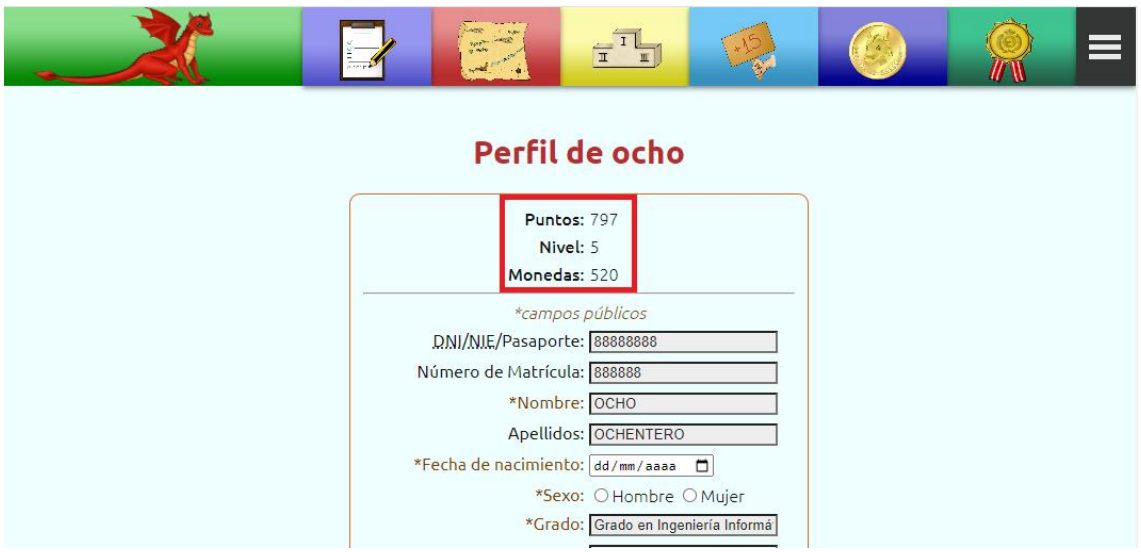

Figura 51. Perfil con la nueva información incorporada

4.2.5.2 Pruebas incompletas

Cuando un alumno inicia una comprobación, tiene un tiempo límite para poder realizar la entrega del fichero. Si el tiempo transcurrido desde que se descargó el fichero de prueba supera el tiempo límite, y no se ha entregado ningún fichero, se considera que no se ha realizado la prueba, por tanto, que no se completó.

Para la realización de esta tarea, se observó que cuando una prueba no se realizaba, se borraba el registro entero de la tabla "grupo_hizo_prueba", por tanto, no se podía obtener información sobre las pruebas que quedaban incompletas. Se ha quitado dicha línea de código para que no se elimine el registro, y se ha añadido el campo "score" en la tabla mencionada anteriormente que indica la puntuación de cero a diez. Este campo se puede usar para saber si una comprobación se ha realizado mirando el valor de este campo "score": si está a `null`, como se puede ver en la Figura 52, significa que la prueba no ha sido finalizada.

IdGrupo	IdPrueba	NumSignifadoToken	Fecha	FormaFichero	ComprobacionAtributos	Puntuacion	score
1	60	1	2024-05-21 16:24:25	0	0	83	8.25
1	61	0	2024-05-21 16:34:29	0	0	0	*NULL*
1	62	1	2024-05-21 17:01:03	0	0	2	0.05

Figura 52. Registros de pruebas de un grupo de la BBDD

4.2.5.3 Niveles de fases activas

Anteriormente, existía en la interfaz de edición de los módulos de la práctica un campo para rellenar que era el de "Nivel necesario". Dicho campo se podía rellenar, pero no tenía ningún uso y no se guardaba en ninguna tabla de la base de datos. De esta manera, en el fichero "editAction.php" ahora se guarda en la tabla "boss_fight" en el campo "req_level" de la misma forma que se guardaban las fechas y los porcentajes de cada fase, es decir, cada nivel también irá separado por '**;**' con el siguiente formato: nivel1;nivel2;nivel3, etc. De esta forma se ha añadido otro campo dentro de la interfaz donde se editan las fases llamadas "Nivel" como se puede ver en la Figura 53 enmarcado en rojo.

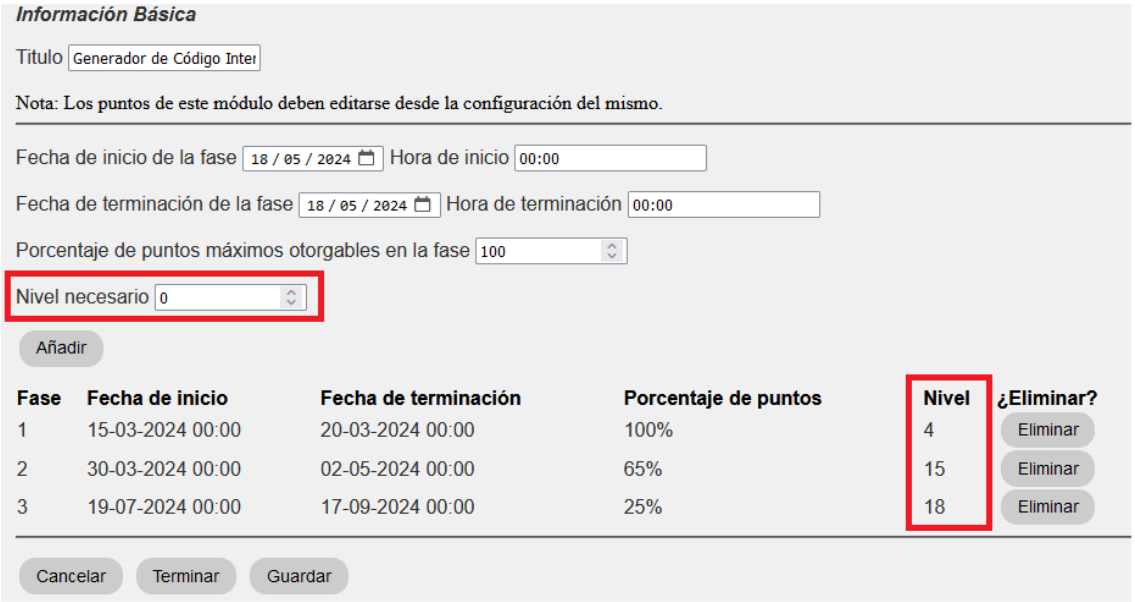

Figura 53. Nueva interfaz del editor de fases de los módulos

Posteriormente, en el fichero de “IndexAction.php” se obtiene la información del nivel utilizando la función `explode()` para obtener la información del nivel de la fase actual y se añade la condición adicional de que el nivel del alumno sea mayor que el nivel de la fase como se ve en la Figura 54.

```
$levels = ($practica["req_level"]) ? explode(";", $practica["req_level"]) :
            array();
foreach($idates as $pos => $idate){
      $ddate = isset($ddates[$pos]) ? $ddates[$pos] : "";
      $percentage = isset($percentages[$pos]) ? $percentages[$pos] : "";
      if($idate < $now_today && $ddate > $now_today &&
            $nivel_alumno >= $levels[$pos]){
                  //Resto de código
            }
      }
```

Figura 54. Fragmento de código sobre el tratamiento de nivel de acceso a una fase

Además, se tuvo que añadir a la condición de mostrar los módulos activos la comprobación de que el alumno tiene mayor nivel que el de la fase activa.

4.2.5.4 Enlaces del perfil

Anteriormente, como se mencionó, en el perfil del alumno en la vista del administrador siempre se mostraban los enlaces configurados por los alumnos a pesar de que no lo estuvieran. La forma de arreglar esto dependía de la forma en la que se guardaban estos enlaces en la tabla “user”. Cuando no había enlace, se guardaba como `null` y esto generaba el problema. Ahora se ha dejado el campo vacío dentro de la tabla y se ha modificado el código para que cuando detecte que está vacío, no se imprima.

En la Figura 55 se puede ver que no hay ningún enlace por parte del alumno; en cambio en la Figura 56 el alumno ha configurado su página web y su cuenta de *Twitter*, y al administrador le aparecen dichos enlaces.

Figura 55. Perfil del alumno sin ningún enlace configurado

Figura 56. Perfil del alumno con dos enlaces configurados

4.2.5.5 Estadísticas de grupo

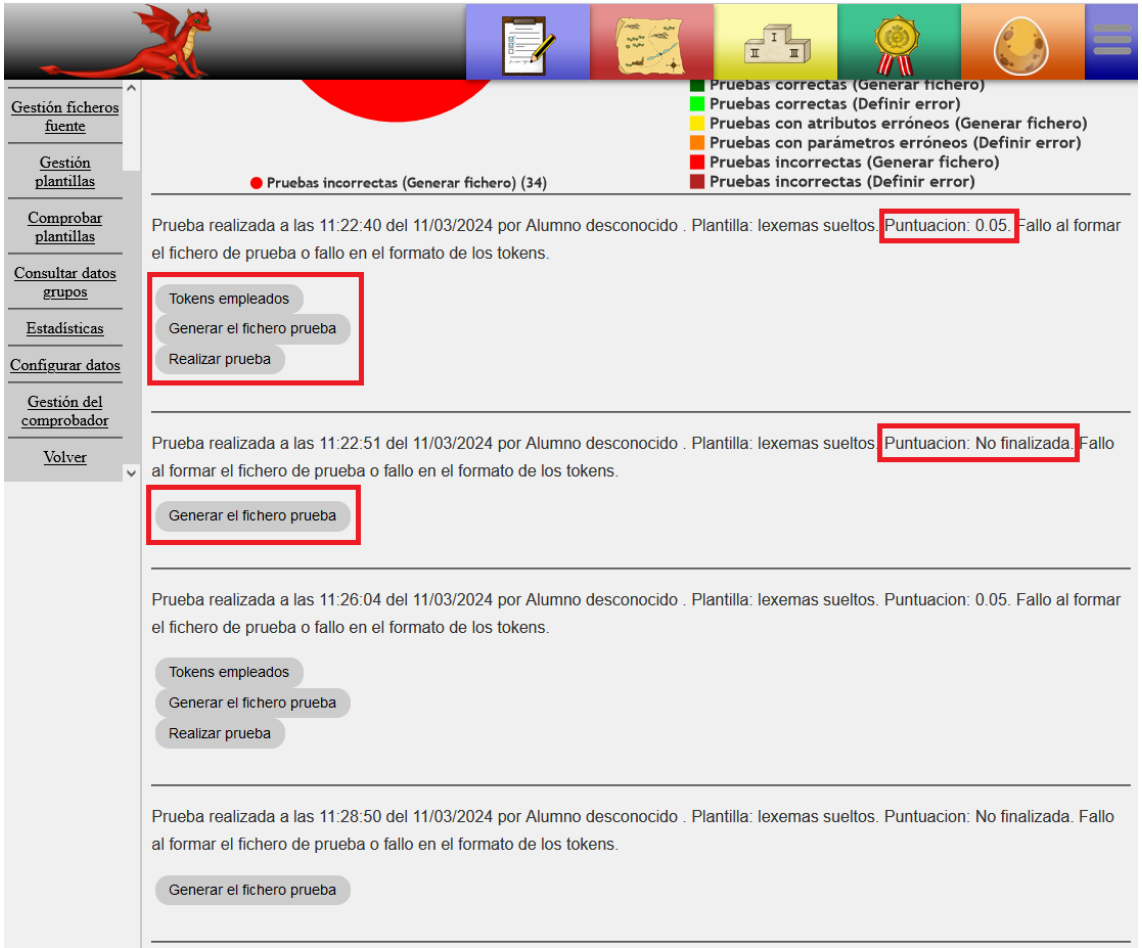

Figura 57. Resumen de pruebas en las estadísticas de grupo

Como se puede ver en la Figura 57, en las pruebas “No finalizadas” solo aparece la opción de generar el fichero prueba. Esto evita que salgan botones para repetir la prueba o descargar ficheros de los alumnos, dado que no tiene sentido dichas acciones para las pruebas sin terminar. Además, se ha modificado la información que se imprime al mostrar el resumen de cada prueba. Antes, se usaba la información del campo “Puntuación” de la tabla “grupo_hizo_prueba”, pero como se incorporó el nuevo campo “score” en dicha tabla que indicaba si una prueba había sido finalizada o no, era un buen indicativo para mostrarlo en este resumen, proporcionando además una información más clara del resultado de una prueba.

Todo esto se ha conseguido modificando el fichero de “MostrarDatos.php” de todos los módulos de ambas asignaturas, añadiendo una condición comprobando que la puntuación sea igual a cero para cuando se van a mostrar dichos botones de “Tokens empleados” (y similares) y “Realizar prueba”. En caso de que la puntuación no sea igual a `null`, significa que la prueba sí que se realizó y no se quedó sin realizar, por tanto, se mostrarían dichos botones.

4.2.6 Accesibilidad

Para garantizar que el sistema sea inclusivo y accesible, se han aplicado los criterios de las Directrices de Accesibilidad para el Contenido Web (WCAG) [45]. A continuación, se detallan algunos de estos criterios y cómo se han integrado en el proyecto:

Uso de Colores (Criterio 1.4.3): Se ha asegurado que el contraste entre el texto y el fondo sea adecuado para usuarios con dificultades visuales. Esto se logró utilizando herramientas de contraste de color para verificar que se cumpla el ratio mínimo de 4.5:1.

Formularios Accesibles (Criterio 3.3.2): Todos los formularios en el sistema incluyen etiquetas claras y descripciones para cada campo, facilitando su comprensión y uso por parte de personas con discapacidades cognitivas.

Uso del Teclado (Criterio 2.1.1): El sistema es completamente navegable usando solo el teclado, lo cual es fundamental para usuarios con dificultades de destreza. Se han implementado accesos directos y órdenes lógicas de tabulación.

Identificación de Errores (Criterio 3.3.1): Los mensajes de error en los formularios son claros y específicos, y proporcionan orientación sobre cómo corregir los errores. Esto es crucial para mejorar la experiencia del usuario y evitar frustraciones.

Tamaño del Texto y Zoom (Criterio 1.4.4): El texto del sistema es escalable sin pérdida de funcionalidad, permitiendo a los usuarios aumentar el tamaño del texto según sus necesidades sin que ello afecte la usabilidad del sistema.

Navegación Consistente (Criterio 3.2.3): La estructura de navegación del sistema tiene coherencia en todas las páginas, lo cual ayuda a los usuarios a predecir dónde encontrar información y cómo moverse por el sistema.

Abreviaturas y Acrónimos (Criterio 3.1.4): Se ha proporcionado la expansión de las abreviaturas y acrónimos utilizados, asegurando que todos los usuarios, independientemente de su nivel de conocimiento previo, puedan comprender el contenido.

Contenido no textual (Criterio 1.1.1): Todas las imágenes en DRACO incluyen descripciones alternativas para garantizar que las personas con discapacidades visuales puedan acceder a la información.

Lenguaje Sencillo (Criterio 3.1.5): El lenguaje utilizado en el sistema es claro y sencillo, evitando jergas técnicas y proporcionando explicaciones adicionales cuando es necesario. Esto facilita la comprensión del contenido para todos los usuarios.

Temporalidad de los Contenidos (Criterio 2.2.1): Se ha asegurado que los usuarios tengan suficiente tiempo para leer y utilizar las funcionalidades del sistema dentro de la página web sin que se cierre automáticamente o se actualice sin previo aviso, lo cual es crucial para usuarios con discapacidades cognitivas o motoras.

4.3 Pruebas

En este apartado se resumen las pruebas realizadas durante el desarrollo de este TFG para la posible detección de errores. Se divide esta sección en dos apartados: pruebas unitarias y pruebas de integración. En las pruebas unitarias, cuando se mencionan los módulos de las prácticas, se hace referencia a la definición de GCI y GCO y a los comprobadores de GCI y GCO.

4.3.1 Pruebas unitarias

Prueba 1. Mostrar los módulos de las prácticas

Los módulos de las prácticas tienen que estar accesibles desde el botón de Prácticas en la pantalla de inicio, desde el desplegable que aparece cuando se pone el ratón encima del botón de "Prácticas" en la pantalla de inicio, y desde el menú principal pulsando el botón "Prácticas".

Esta prueba ayudó a detectar que no aparecía el botón de "Prácticas" en el menú de DRACO cuando debería aparecer, y se solucionó.

Prueba 2. Acceso a la definición de GCI

Pulsando "Definición del CI", lleva a la descripción sobre la definición y posteriormente, pulsando el botón "Continuar", se accede al formulario de la definición.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 3. Errores al guardar la definición de GCI

Se ha probado que si el alumno no cumple con el formato que deben tener los lexemas de los operadores y las clases aparece un mensaje de error indicando qué lexemas están mal escritos.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 4. Guardar la definición de GCI

Se ha probado que, si el alumno no rellena el mínimo de operadores y de clases y se pulsa el botón "Guardar", el sistema guarda la definición para el grupo del alumno que la realizó tras realizar las comprobaciones necesarias y ver que no hay errores. También se comprobó que en este caso no se otorgan puntos al grupo.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 5. Mantener definición de GCI

Si el alumno quiere guardar la definición y hay algún error que lo impide, se mantiene el formulario con los campos rellenos que había definido el alumno.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 6. Guardar la definición de GCI

Cuando se pulsa el botón "Guardar", el sistema guarda la definición para el grupo del alumno que la realizó tras realizar las comprobaciones necesarias y ver que no hay errores.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 7. Acceso a la definición GCI

Cuando el alumno accede a la definición, se ha comprobado que se muestran los campos rellenos correspondientes a la definición que ya tenía hecha previamente, en caso de que la hubiera.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 8. Acceso a la definición de GCO

Pulsando "Definición del GCO", lleva a la descripción sobre la definición y posteriormente, pulsando el botón "Continuar", se accede al formulario de la definición.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 9. Selección en la definición de GCO

El alumno solo puede escoger una de las 2 opciones que se le propone antes de querer guardar la definición.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 10. Guardar la definición de GCO

Cuando se pulsa el botón "Guardar", el sistema guarda la definición para el grupo del alumno que la realizó, haciendo las comprobaciones necesarias y si todo es correcto.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 11. Acceso al comprobador de GCI

Al pulsar "Generador de Código Intermedio" lleva a la descripción sobre la definición y posteriormente, pulsando el botón "Realizar prueba", se dará la opción de descargar el fichero.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 12. Acceso al comprobador de GCO

Pulsando "Generador de Código Objeto", lleva a la descripción sobre la definición y posteriormente, pulsando el botón "Realizar prueba", se dará la opción de descargar el fichero.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 13. Mostrar módulos para configuración

En la vista del administrador se ha comprobado que esté disponible la configuración de los módulos de las prácticas.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 14. Activar y desactivar módulo

Para cada uno de los módulos de las prácticas se ha comprobado que, si se desactiva un módulo, ya no aparece disponible para los alumnos, y si se activa,

se ha comprobado que el sistema activa la fase que comprende la fecha actual, y vuelve a aparecer disponible para los alumnos.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 15. Editar módulos

Se ha comprobado que se pueden añadir fases con nivel mínimo de acceso para cada módulo.

Esta prueba ayudó a detectar errores sobre cómo se guardaba el nivel mínimo de cada fase y se solucionó.

Prueba 16. Configurar módulos

Se ha comprobado que el administrador puede configurar los parámetros del módulo elegido desde el apartado de "Configurar datos". Cuando se vuelve a acceder a dicha sección, se muestran los resultados guardados con anterioridad en el caso de que los haya.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 17. Configurar fragmentos fuente

Se ha comprobado que el administrador puede subir al sistema fragmentos fuente siguiendo el flujo de importación de fragmentos fuente desde el apartado de "Gestión fragmentos fuente".

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 18. Configurar plantillas

Se ha comprobado que el administrador puede subir plantillas al sistema y puede configurar dichas plantillas de forma individual desde la sección "Gestión de plantillas".

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 19. Comprobar plantillas

Se ha comprobado que se pueden realizar las dos comprobaciones disponibles desde el apartado de "Comprobar plantillas".

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 20. Estadísticas

El administrador puede acceder a las estadísticas generales del módulo seleccionado. Gracias a esta prueba se han detectado errores en el campo de "Pruebas no Realizadas". También se han detectado errores en el diagrama de barras, que no coincidía con los datos de la tabla.

Tras repetir las pruebas, se verifica que ahora se ven de forma correcta los diagramas y las tablas de las estadísticas desde el apartado de "Estadísticas".

Esta prueba ha ayudado a detectar un error en el diagrama de barras ya que no coincidía con los datos de las tablas que aparecían debajo pero ya se muestran correctamente el resumen de las pruebas y el diagrama de barras.

Prueba 21. Estadísticas de grupo

Se ha comprobado que se puede acceder a las estadísticas de cada grupo desde el número de grupo que aparece en la tabla de la pantalla de estadísticas. Se muestran correctamente el resumen de las pruebas y el diagrama de barras.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 22. Perfil del alumno

Se ha comprobado que, dentro del perfil del alumno, aparezca, al principio, el nivel, puntos y número de monedas correspondientes.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 23. Comprobaciones a medias

Se ha comprobado que, cuando se arranca una comprobación y no se entrega dentro del tiempo establecido, queda registrado en la base de datos que la prueba no ha sido realizada.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 24. Configurar nivel en las fases

Se ha comprobado que se puede definir un nivel mínimo para el acceso a cada fase de cada módulo de las prácticas.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 25. Acceso a las fases

Se ha comprobado que se puede acceder a un módulo activo si se tiene el nivel suficiente, y si no se tiene dicho nivel mínimo no aparece dicho módulo.

Se realizó la prueba y funcionaba correctamente como se esperaba.

Prueba 26. Enlaces de los alumnos

Se ha comprobado que, dentro del perfil del alumno en la vista del administrador, solo aparezcan los enlaces que dicho alumno haya configurado.

Gracias a esta prueba, se han detectado errores en el código sobre cómo se comprobaban la existencia de dichos enlaces, y la forma en la que se guardaban en la base de datos y se solucionó.

Prueba 27. Estadísticas de grupo

Se ha comprobado que, una vez se haya realizado una comprobación, dentro las estadísticas individuales del grupo, en el registro de dicha prueba, no aparezcan las opciones de realizar de nuevo la prueba, o la de qué *tokens* se han empleado.

4.3.2 Pruebas de integración

A continuación, se van a explicar las pruebas de integración que se han realizado para comprobar el funcionamiento del sistema y los posibles errores detectados.

Para realizar las pruebas de las definiciones y las comprobaciones de los módulos se ha utilizado un usuario de pruebas dentro del sistema que actúa como alumno.

Comprobadores

El alumno descarga el fichero de prueba y posteriormente puede subir su fichero procesado al sistema para que realice las comprobaciones necesarias del fichero subido.

El alumno puede leer los errores que ha tenido en su comprobación del fichero procesado y también puede ver la puntuación obtenida en dicha comprobación.

En caso de que el alumno no tenga una definición hecha, no se le permite hacer una comprobación y, en lugar de darle la opción de realizar la prueba, se le da la opción de realizar la definición.

El alumno puede introducir datos en el formulario par el caso de la definición de GCI, o responder a la pregunta en el caso de la definición del GCO.

Se han realizado varias pruebas exhaustivas sobre el funcionamiento de las pruebas de los comprobadores, y se detectó un error. Dicho error era que se le permitía al alumno realizar la comprobación sin tener previamente ninguna definición hecha.

Configuración de los comprobadores

La configuración de los comprobadores de Traductores de Lenguaje en la vista del administrador se han integrado con el resto de los comprobadores en DRACO, es decir, con los de Procesadores de Lenguajes, teniendo un funcionamiento homogéneo entre todos ellos.

Se ha añadido al sistema una comprobación del curso activo, para poder saber si la asignatura activa es la de PDL o la de TDL, por tanto, se ha podido verificar que, dependiendo del resultado de dicha comprobación, el alumno va a ver los módulos correspondientes a la asignatura en curso.

Además, se han hecho pruebas para verificar que el administrador puede configurar cualquiera de los módulos de forma independiente en la parte de configuración en la vista del profesor, independientemente de la asignatura activa.

Tareas de mantenimiento

Las pruebas de integración de las tareas de mantenimiento son muy parecidas a las pruebas unitarias mencionadas anteriormente y los errores detectados también. Se podría añadir que el administrador puede ahora ver en la interfaz, el nivel que tiene cada fase de cada módulo tanto de PDL como de TDL, puede consultar los enlaces configurados de cada alumno, y puede descargar el fichero de prueba generado en una comprobación no finalizada, dentro del registro de pruebas de cada grupo.

5 Resultados y conclusiones

Durante la realización de este Trabajo de Fin de Grado, se han implementado nuevas características y funcionalidades al sistema web DRACO. Estas nuevas implementaciones permiten una mejor gestión y evaluación de las prácticas y ejercicios de los estudiantes. Ahora, el sistema puede proporcionar una retroalimentación más precisa y detallada, lo que facilita tanto la labor de los profesores al corregir las prácticas como el aprendizaje de los estudiantes al identificar más claramente sus errores y áreas de mejora.

Estas nuevas implementaciones abren un nuevo camino en la página web para la asignatura de Traductores de Lenguajes, ya que no disponía de un sistema de comprobadores y definiciones como lo tenía la asignatura de Procesadores de Lenguajes, que está muy bien valorado por los alumnos. La inclusión de estos comprobadores es sumamente útil para los estudiantes, ya que les permite verificar el correcto funcionamiento de sus prácticas de manera autónoma e intuitiva. Tener acceso a un comprobador de prácticas ahorra mucho tiempo y también mejora la calidad del aprendizaje. Los estudiantes pueden identificar y corregir errores por sí mismos, lo que fomenta un aprendizaje muy activo. Además, este sistema ayuda a estandarizar las evaluaciones, asegurando que todos los estudiantes sean evaluados bajo los mismos criterios y condiciones.

Gracias a este TFG, se ha conseguido desarrollar un sistema funcional de definiciones para los módulos, aunque temporal ya que está sujeto a modificaciones futuras, y una interfaz preparada para la implementación de los comprobadores.

Una de las principales novedades de este TFG es que se ha añadido una sección de definiciones sobre cada módulo. Esta sección permitirá a los alumnos definir los operadores y las clases de argumentos, para el comprobador del generador de código intermedio, y el sentido de crecimiento de la pila, para poder realizar las futuras comprobaciones del generador de código objeto.

También, se han realizado tareas de mantenimiento que han mejorado las funcionalidades y usabilidad de DRACO. Dichas actualizaciones mostrarán información adicional en el perfil del alumno, lo cual es más cómodo para que puedan ver sus monedas, puntos y nivel de forma más rápida.

Asimismo, se implementó la novedad de registrar cuándo una comprobación quedaba incompleta, es decir, cuándo el alumno no entregaba el fichero resultante. Anteriormente, el profesor no podía ver la información de las pruebas no completadas, lo que le impedía tener una visión completa del trabajo realizado por los estudiantes. Ahora, con esta nueva funcionalidad, el profesorado puede realizar un seguimiento más preciso del rendimiento de los estudiantes, identificando no solo las prácticas finalizadas sino también aquellas que quedaron incompletas. Esto permite a los administradores obtener una comprensión más profunda de las dificultades y el progreso de los alumnos.

Por otro lado, se introdujeron mejoras para los módulos de las prácticas estableciendo niveles mínimos para cualquiera de las fases activas que hubiera en el sistema, lo que asegura para el alumno, tener una progresión adecuada según el nivel del estudiante en el sistema.

En la parte de administración, se optimizó la interfaz del administrador para que no se muestren enlaces inexistentes cuando fuera a ver el perfil de un alumno, mostrando así una interfaz más limpia y libre de errores y confusiones.

Estas mejoras no solo han optimizado la funcionalidad del sistema, sino que también han contribuido a una experiencia más eficiente y agradable tanto para los alumnos, como para los profesores.

A lo largo del desarrollo de este Trabajo de Fin de Grado, se ha optado en la reutilización de algunas partes de código del sistema, realizada por otros desarrolladores. Para mantener el código fuente similar al que ya había, se han utilizado funciones y fragmentos de código ya existentes en lugar de escribir todo desde cero. De esta manera se ha conseguido mantener un código uniforme y ahorrar tiempo, pero no por ello quiere decir, que no haya habido ninguna dificultad debido a la complejidad y la cantidad de *scripts* que hay.

Uno de los mayores retos fue comprender el código PHP utilizado en DRACO. Al principio del desarrollo, PHP resultó un lenguaje complicado debido a su sintaxis particular y la manera de fusionarse con el lenguaje HTML y CSS. Posteriormente a eso, hubo que comprender el flujo de los códigos fuentes y como interactuaban con la base de datos.

Además, había mucha falta de documentación sobre el código, y la cantidad de fragmentos de código comentados, añadió más dificultad a su comprensión. Sin embargo, a medida que se iban realizando pequeñas tareas de mantenimiento, se lograba una familiarización con el código comprendiendo mejor su funcionamiento y pudiendo desarrollar las nuevas implementaciones con más soltura.

Como conclusiones personales, la realización de este Trabajo de Fin de Grado, me he hecho comprender más en profundidad qué es un sistema gamificado, y cómo funciona el desarrollo web. Trabajar en DRACO para poder mejorarlo no sólo me ha ayudado a mejorar mis habilidades en desarrollo web, sino también la resolución de problemas y comprender un código fuente de un proyecto tan grande como este me ha dado más soltura para posibles situaciones futuras. A pesar de que me costó comprender el código PHP, a medida que iba desarrollando, iba entendiendo mejor su funcionamiento y me ha dado confianza como desarrollador y sobre todo he aprendido la importancia de la claridad que debe tener un código y lo accesible que tiene que ser para personas con discapacidad.

Además, este proyecto me ha permitido apreciar el trabajo en conjunto gracias al trabajo de otros desarrolladores para la reutilización de ciertos fragmentos y funciones del código. Aprender a integrar nuevas características y funcionalidades dentro de un sistema ya existente, como DRACO, mejorando y respetando el trabajo previo ha sido una buena lección para futuros proyectos.

Sí que he echado en falta más documentación para la comprensión del código para que así, de esta manera, los futuros desarrolladores que vayan a implementar nuevas características y funcionalidades no pierdan tanto tiempo al principio en entender todo el código fuente.

En resumen, este Trabajo de Fin de Grado me ha ayudado a fortalecer mis conocimientos sobre el desarrollo web y su accesibilidad, así como el trabajo en equipo que llevaré conmigo en mi carrera profesional.

6 Análisis de impacto

En esta sección se describirá el análisis de impacto sobre los resultados obtenidos tras la realización de este Trabajo de Fin de Grado.

En el contexto personal, he podido aprender muchas cosas nuevas sobre los lenguajes PHP, MySQL, HTML y CSS. He podido aprender más sobre cómo funciona el desarrollo web dentro de un servidor, y no en un navegador como puede ser con JavaScript, y cómo es trabajar en un proyecto grande, en el que han participado también otros alumnos. Gracias a esto, he aprendido a poder adaptarme al código escrito por otras personas y he conseguido seguir el flujo de todo el código fuente antes de empezar a realizar la implementación de mi TFG. Por último, es necesario mencionar el conocimiento adquirido sobre las pautas WCAG que han resultado ser muy importantes para personas con discapacidad.

También este TFG beneficia a las asignaturas de Procesadores de Lenguajes y Traductores de Lenguajes dentro de la Escuela Técnica Superior de Ingenieros Informáticos de la UPM. Y gracias a este trabajo, se ha conseguido avanzar en la implementación de unos nuevos comprobadores para la asignatura de TDL que aún no estaban implementados. De esta manera, los alumnos que vayan a cursar dicha asignatura optativa podrán realizar sus comprobaciones de las prácticas de forma más eficiente y dentro de un sistema gamificado que les ayude a tener más motivación.

A continuación, se muestran los ODS (Objetivos de Desarrollo Sostenible) [46] que son de utilidad en el desarrollo y solución de este Trabajo de Fin de Grado.

- Objetivo 4: Garantizar una educación inclusiva, equitativa y de calidad y promover oportunidades de aprendizaje durante toda la vida para todos.
    - Meta 4.3: DRACO facilita el acceso a los mismos recursos educativos de calidad para cualquier persona sin discriminar el género de la persona.
    - Meta 4.4: DRACO incentiva el aprendizaje continuo y fomenta el aprendizaje para que los alumnos obtengan competencias relevantes de cara al futuro.
    - Meta 4.5: DRACO cumple con las pautas de accesibilidad y no discrimina a ninguna persona con discapacidad.
- Objetivo 5: Lograr la igualdad entre los géneros y empoderar a todas las mujeres y niñas.
    - Meta 5.1: DRACO es un entorno libre de discriminación hacia las mujeres
    - Meta 5.5: El sistema está diseñado para que tanto mujeres como hombres obtengan la misma puntuación por sus actividades dentro de DRACO, de esta manera, las mujeres pueden liderar el ranking al igual que los hombres.
- Objetivo 8: Promover el crecimiento económico inclusivo y sostenible, el empleo y el trabajo decente para todos.
    - Meta 8.4: Al ser DRACO un sistema web, no consume recursos materiales adicionales no necesarios.
    - Meta 8.6: DRACO pertenece a una asignatura de la UPM, y se encarga de proporcionar conocimientos informáticos a los alumnos para que, en un futuro, se puedan dedicar a ello, mejorando así, la empleabilidad.

- Objetivo 9. Construir infraestructuras resilientes, promover la industrialización sostenible y fomentar la innovación.
    - Meta 9.5: DRACO puede ayudar a los alumnos a mostrar interés por los compiladores y fomentar así a que participen en proyectos de investigación e innovación.
- Objetivo 10. Reducir la desigualdad en y entre los países.
    - Meta 10.2: DRACO tiene un diseño de una web inclusiva y accesible para todos los estudiantes dejando clara la igualdad de oportunidades entre todo tipo de personas sin importar su procedencia, género, raza, religión o discapacidad.

7 Futuras líneas de trabajo

DRACO es un sistema que lleva en desarrollo más de 10 años y cada año se van añadiendo nuevas características y mejoras, pero siempre hay hueco para incorporar nuevas funcionalidades.

Nuevos comprobadores

En cuanto a la parte de DRACO que pertenece a la asignatura de Traductores de Lenguajes, solo se ha implementado la interfaz de cómo serían los comprobadores y las definiciones para poder realizar las pruebas. Para el futuro, habría que implementar la funcionalidad de los comprobadores para que dichos módulos puedan corregir los ficheros pasados por parámetro, y así los alumnos podrían disponer de un sistema funcional para la asignatura. Ya se ha comenzado a trabajar en esta línea futura, pues de forma paralela a la realización de este Trabajo de Fin de Grado, hubo dos alumnos desarrollando los comprobadores de ambos módulos, pero aún sigue en proceso dicho desarrollo. Cuando se termine, se podrá integrar con la interfaz y las bases de datos creadas durante este TFG.

Insignias

También para la parte de Traductores de Lenguajes, habría que añadir nuevas insignias para cuando los alumnos realicen actividades o tareas dentro del sistema.

Marco de avatar

En cuanto a las recompensas, se podría implementar un marco para el avatar. Esto quiere decir, que, dependiendo del nivel del alumno, u otro parámetro, se pueda añadir un marco al avatar de bronce, plata, platino, diamante, etc. Esto ayudaría a los alumnos a seguir participando en tareas y actividades para poder obtener una distinción que pocos alumnos puedan obtener y distinguirse del resto por su trabajo continuo favoreciendo la motivación.

Mapa de actividades

También se podría implementar, tanto para Procesadores de Lenguajes como para Traductores de Lenguajes, un mapa ilustrativo para las actividades que llegue a una meta y sea un tesoro, por ejemplo. Dicho tesoro puede ser una recompensa que pueda ayudar de alguna manera al alumno para la realización de las prácticas, como pueden ser intentos extra o DRACoins. Dicho mapa sería más interactivo y agradable a la vista para los alumnos y puede promover a realizar las actividades más que una lista como está actualmente.

Comunicación entre alumno y profesor

Dentro de DRACO, la interacción entre el alumno y el profesor es nula. Para esto se podría implementar un pequeño sistema dentro de DRACO para poder pedir tutorías o poder enviar correos al profesor de forma directa sin necesidad de utilizar WebMail por ejemplo. Que el sistema pueda detectar quien es el alumno que está realizando la consulta y se envié un correo al profesor mediante un formulario. Siguiendo esta misma línea, que los alumnos tengan un apartado para poder reportar errores que puedan ser de utilidad para la resolución de fallos en el sistema con más prioridad, en lugar de que se detecten con el tiempo.

Comprensión del código

El entendimiento del código es complicado para empezar a desarrollar nuevas funcionalidades y características, ya que se pierde mucho tiempo en comprenderlo. Para ello, habría que limpiarlo de todos los fragmentos de código comentados ya que resultan innecesarios, y dificultan la comprensión. También, se ha detectado que alguna tabla en las bases de datos contiene datos irrelevantes que no se usan y muchas veces se repiten registros de forma innecesaria, por lo que habría que revisar todas las bases de datos. También ayudaría tener una documentación útil explicando qué hace cada *script* y cómo se compagina con el resto.

Filtros en las estadísticas

El administrador puede mirar las estadísticas individuales de cada grupo y durante la realización de este Trabajo de Fin de Grado, se han realizado severas pruebas, y se ha echado en falta un filtro para mostrar las pruebas según un filtro de fecha, por ejemplo. Dicho esto, se podría incorporar un filtro para mostrar las pruebas realizadas por un grupo en un comprobador.

8 Bibliografía

[1] R. Barzanallana, «Introducción a HTML y CSS,» [En línea]. Available: https://www.um.es/docencia/barzana/DAWEB/2017-18/daweb-tema-1-introduccion-html-css.html. [Último acceso: 4 Marzo 2024].

[2] A. Hern, «World's most delayed software released after 54 years of development,» 6 Junio 2014. [En línea]. Available: https://www.theguardian.com/technology/2014/jun/06/vapourware-software-54-years-xanadu-ted-nelson-chapman. [Último acceso: 4 Marzo 2024].

[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: 5 Marzo 2024].

[4] Whatwg, «HTML,» [En línea]. Available: https://html.spec.whatwg.org/. [Último acceso: 5 Marzo 2024].

[5] M. W. Docs, «Conceptos básicos de HTML,» [En línea]. Available: https://developer.mozilla.org/es/docs/Learn/Getting_started_with_the_web/HTML_basics. [Último acceso: 5 Marzo 2024].

[6] Uniwebsidad, « Breve historia de CSS,» [En línea]. Available: https://uniwebsidad.com/libros/css/capitulo-1/breve-historia-de-css. [Último acceso: 13 Marzo 2024].

[7] J. A. L. Quesada, «Librosweb,» [En línea]. Available: http://dis.um.es/~lopezquesada/documentos/IES_1213/LMSGI/curso/UT5/libroswebcss/www.librosweb.es/css/capitulo1/breve_historia_de_css.html. [Último acceso: 14 Marzo 2024].

[8] W. Schools, «CSS,» [En línea]. Available: https://www.w3schools.com/css/. [Último acceso: 14 Marzo 2024].

[9] J. t. expert, «Jerarquía CSS,» 16 Marzo 2023. [En línea]. Available: https://juniortoexpert.com/es/jerarquia-css/. [Último acceso: 16 Marzo 2024].

[10] M. W. Docs, «Cascada y herencia,» [En línea]. Available: https://developer.mozilla.org/es/docs/Learn/CSS/Building_blocks/Cascade_and_inheritance#especificidad. [Último acceso: 16 Marzo 2024].

[11] E. Virtual, «La evolución del PHP: Pasado, presente y tendencias futuras,» [En línea]. Available: https://einavirtual.com/php/historia-es. [Último acceso: 8 Marzo 2024].

[12] P. PHP, «Historia de PHP y proyectos relacionados,» [En línea]. Available: https://programadorphp.es/docs/php_manual_espanol/history.html. [Último acceso: 8 Marzo 2024].

[13] PHP, «History of PHP,» [En línea]. Available: https://www.php.net/manual/en/history.php.php. [Último acceso: 8 Marzo 2024].

[14] V. Duka, «Introducción A PHP 7: Qué Hay De Nuevo Y Qué Se Ha Ido,» 2016. [En línea]. Available: https://www.toptal.com/php/introduccion-a-php-7-que-hay-de-nuevo-y-que-se-ha-ido. [Último acceso: 9 Marzo 2024].

[15] d. ade, «PHP 5 Global Variables – Superglobals,» 18 Octubre 2017. [En línea]. Available: https://ictacademy.com.ng/php-5-global-variables-superglobals/. [Último acceso: 9 Marzo 2024].

[16] E. d. E. e. C. y. Tecnología, «Lenguaje SQL, historia y conceptos básicos,» 14 Septiembre 2016. [En línea]. Available: https://www.universidadviu.com/es/actualidad/nuestros-expertos/lenguaje-sql-historia-y-conceptos-basicos. [Último acceso: 12 Marzo 2024].

[17] A. AWS, «¿Qué es SQL (lenguaje de consulta estructurada)?,» [En línea]. Available: https://aws.amazon.com/es/what-is/sql/. [Último acceso: 12 Marzo 2024].

[18] S. Tech, «WCAG: qué son y para qué sirven,» 14 Junio 2023. [En línea]. Available: https://smowl.net/es/blog/wcag/. [Último acceso: 18 Marzo 2024].

[19] S. L. Mora, «Accesibilidad Web,» [En línea]. Available: https://accesibilidadweb.dlsi.ua.es/?menu=principios-2.1. [Último acceso: 19 Marzo 2024].

[20] T. Estudio, «WCAG 2.1: qué son y cómo respetarlas,» 7 Febrero 2022. [En línea]. Available: https://torresburriel.com/weblog/wcag-2-1-que-son-y-como-respetarlas/. [Último acceso: 19 Marzo 2024].

[21] W3, «Understanding Conformance,» [En línea]. Available: https://www.w3.org/WAI/WCAG21/Understanding/conformance.html. [Último acceso: 20 Marzo 2024].

[22] V. Gaitán, «Gamificación: el aprendizaje divertido,» [En línea]. Available: https://www.educativa.com/blog-articulos/gamificacion-el-aprendizaje-divertido/. [Último acceso: 26 Marzo 2024].

[23] W. Jara, «Gamificación: Un mundo para educar,» 6 Diciembre 2019. [En línea]. Available: https://matematicas69026909.wordpress.com/2019/12/06/historia-de-la-gamificacion/. [Último acceso: 26 Marzo 2024].

[24] J. C. Martín, «Cómo aplicar la gamificación en entornos formativos,» 13 Febrero 2023. [En línea]. Available: https://www.wetak.com/como-aplicar-la-gamificacion-en-entornos-formativos/. [Último acceso: 14 Abril 2024].

[25] P. Lobato, «¿Qué es la gamificación?,» [En línea]. Available: https://www.smartmind.net/blog/que-es-la-gamificacion/#:~:text=Los%20tres%20elementos%20de%20la,las%20mecánica. [Último acceso: 28 Marzo 2024].

[26] O. B. Gené, «Fundamentos de la gamificación,» [En línea]. Available: https://oa.upm.es/35517/1/fundamentos%20de%20la%20gamificacion_v1_1.pdf. [Último acceso: 28 Marzo 2024].

[27] R. Rosas, «Qué es la gamificación, elementos y beneficios,» [En línea]. Available: https://rosanarosas.com/que-es-gamificacion-como-funciona/. [Último acceso: 29 Marzo 2024].

[28] G. d. Canarias, «Gamificación,» [En línea]. Available: https://www3.gobiernodecanarias.org/medusa/ecoescuela/pedagotic/files/2018/11/gamificacion.pdf. [Último acceso: 29 Marzo 2024].

[29] M. Á. G. García, «Grace Hopper, la brillante inventora del primer compilador,» 28 Febrero 2018. [En línea]. Available: https://www.exevi.com/grace-hopper-la-brillante-inventora-del-primer-compilador/. [Último acceso: 14 Abril 2024].

[30] C. D. Gomez, «La evolución de los compiladores,» University of San Carlos of Guatemala, Mayo 2023. [En línea]. Available: https://www.researchgate.net/publication/370684300_La_evolucion_de_los_compiladores. [Último acceso: 1 Abril 2024].

[31] A. M. Soledad, «La evolución de los compiladores,» 23 Julio 2021. [En línea]. Available: https://www.researchgate.net/publication/370684300_La_evolucion_de_los_compiladores. [Último acceso: 1 Abril 2024].

[32] M. H. C. Zacatelco, «Compiladores,» [En línea]. Available: https://www.google.com/url?sa=t&source=web&rct=j&opi=89978449&url=https://www.cs.buap.mx/~hilda/notascompila.doc&ved=2ahUKEwiuwMLhk6uFAxXmwAIHHUuWCS0QFnoECEoQAQ&usg=AOvVaw2-SyKSgTIxLq8Tz2RWgTIC. [Último acceso: 2 Abril 2024].

[33] J. Calvo, «¿Que es un Compilador en programación?,» 17 Abril 2018. [En línea]. Available: https://www.europeanvalley.es/noticias/que-es-un-compilador-en-programacion/. [Último acceso: 3 Abril 2024].

[34] I. P. Pérez, «Fases de un Compilador y sus Fundamentos Teóricos,» [En línea]. Available: http://cidecame.uaeh.edu.mx/lcc/mapa/PROYECTO/libro32/13_fases_de_un_compilador_y_sus_fundamentos_tericos.html. [Último acceso: 5 Abril 2024].

[35] D. Singh, «Designing and Implementing Symbol Tables,» [En línea]. Available: https://noobtomaster.com/compiler-design/designing-and-implementing-symbol-tables/. [Último acceso: 01 Junio 2024].

[36] T. Point, «Compiler Design - Symbol Table,» [En línea]. Available: https://www.tutorialspoint.com/compiler_design/compiler_design_symbol_table.htm. [Último acceso: 01 Junio 2024].

[37] GeeksforGeeks, «Generación de código intermedio en diseño de compiladores,» 19 Enero 2024. [En línea]. Available: https://www.geeksforgeeks.org/intermediate-code-generation-in-compiler-design/. [Último acceso: 6 Abril 2024].

[38] GeeksforGeeks, «Introducción del código objeto en el diseño del compilador,» 12 Mayo 2023. [En línea]. Available: https://www.geeksforgeeks.org/introduction-of-object-code-in-compiler-design/?ref=lbp. [Último acceso: 6 Abril 2024].

[39] V. Jiménez, « Procesadores de lenguaje,» [En línea]. Available: https://www3.uji.es/~vjimenez/AULASVIRTUALES/PL-0910/T5-GENERACION/codigo.apun.pdf. [Último acceso: 6 Abril 2024].

[40] J. N. Parra, «Generación de código,» [En línea]. Available: http://webdiis.unizar.es/~neira/12048/generaciondecodigo.pdf. [Último acceso: 6 Abril 2024].

[41] V. S. Code, «April 2024,» [En línea]. Available: https://code.visualstudio.com/updates/v1_89. [Último acceso: 01 Junio 2024].

[42] Xampp, «XAMPP,» [En línea]. Available: https://www.apachefriends.org/index.html. [Último acceso: 01 Junio 2024].

[43] GitLab. [En línea]. Available: https://gitlab.com/. [Último acceso: 01 Junio 2024].

[44] Seed, «¿Cómo configurar Xdebug & VSCode?,» 18 Mayo 2021. [En línea]. Available: https://www.seedem.co/es/blog/como-configurar-xdebug-vscode. [Último acceso: 01 Junio 2024].

[45] S. L. Henry., «Sumario de WCAG 2,» 1 Mayo 2024. [En línea]. Available: https://www.w3.org/WAI/standards-guidelines/wcag/es. [Último acceso: 28 Mayo 2024].

[46] N. Unidas, «Objetivos de desarrollo sostenible,» [En línea]. Available: https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/. [Último acceso: 2024 Mayo 28].

Este documento esta firmado por

	**Firmante**	CN=tfgm.fi.upm.es, OU=CCFI, O=ETS Ingenieros Informaticos - UPM, C=ES
	Fecha/Hora	Mon Jun 03 10:48:09 CEST 2024
	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)