Escuela Superior
de Ingeniería y Tecnología
Universidad de La Laguna

Trabajo de Fin de Grado

CODE-ULL: Sistemas para la ejecución, validación y puntuación de código fuente

CODE-ULL: Systems for execution, validation and scoring of source code

José Lozano Armas

La Laguna, 12 de marzo de 2025

Dra. Dª. **Coromoto León Hernández**, profesora Catedrática del área de Lenguajes y Sistemas Informáticos adscrita al Departamento de Ingeniería Informática y de Sistemas de la Universidad de La Laguna, como tutora

I N F O R M A

Que la presente memoria titulada:

”*CODE-ULL: Sistemas para la ejecución, validación y puntuación de código fuente*”

ha sido realizada bajo su dirección por D. **José Lozano Armas**.

Y para que así conste, en cumplimiento de la legislación vigente y a los efectos oportunos firma la presente en San Cristóbal de La Laguna, a 12 de marzo de 2025

Agradecimientos

Quiero agradecer la oportunidad de desarrollar este Trabajo Fin de Grado a mi tutora Doña Coromoto León, así cómo al Aula Cultural de Pensamiento Computacional por la organización de las Olimpiadas.

También quiero dar las gracias a mi familia y amigos, los cuales siempre me apoyaron desde el principio de la carrera hasta el desarrollo de este trabajo final que culmina con esta memoria.

Licencia

Resumen

A lo largo de esta memoria se explicará todo lo relacionado con el desarrollo de "CODE-ULL", una aplicación web, que permite la ejecución, validación y puntuación de código fuente que soluciona un problema de programación. El sistema se utiliza para evaluar el código enviado por los usuarios en función de criterios como corrección, eficiencia y tiempo de ejecución.

"CODE-ULL" permite gestionar distintos tipos de usuarios. El administrador tiene permisos para gestionar usuarios y crear preguntas sobre programación informática. Los participantes podrán registrarse para intentar resolver dichas preguntas y subir al sistema el código con la solución.

El sistema ejecuta y valida código automáticamente en varios lenguajes de programación. Compara la salida del código enviado con la salida esperada para verificar si es correcta. Evalúa el rendimiento en términos de tiempo de ejecución. Proporciona retroalimentación inmediata sobre si la solución es correcta o tiene errores. Soporta múltiples lenguajes de programación como C++, Java, Python, etc. Además, muestra un listado categorizado de todos los participantes ordenado en base a los puntos obtenidos y la hora en la que se ha enviado la solución.

Palabras clave: aplicación web, validación, puntuación, ejecución, prototipo, arquitectura react-flask, sistema de rankings, gestión del tiempo.

Abstract

Throughout this report, everything related to the development of "CODE-ULL" will be explained, a web application that allows the execution, validation and scoring of source code that solves a programming problem. The system is used to evaluate the code sent by users based on criteria such as correctness, efficiency and execution time.

"CODE-ULL" allows managing different types of users. The administrator has permissions to manage users and create questions about computer programming. Participants can register to try to solve these questions and upload the code with the solution to the system.

The system automatically executes and validates code in various programming languages. It compares the output of the submitted code with the expected output to verify if it is correct. It evaluates performance in terms of execution time. It provides immediate feedback on whether the solution is correct or has errors. It supports multiple programming languages such as C++, Java, Python, etc. In addition, it shows a categorized list of all participants ordered based on the points obtained and the time at which the solution was submitted.

Keywords: web application, validation, puntuation, ejecution, prototype, architecture react-flask, Rankings system, time management.

Índice general

Índice de Figuras

Índice de Tablas

Capítulo 1

Introducción

A lo largo de este capítulo nos dedicaremos a dar un contexto del porqué la realización del trabajo de fin de grado, así cómo plantear que antecedentes hay sobre los sistemas para la ejecución, validación y puntuación de código fuente, cuáles son nuestros objetivos con la realización de este proyecto, y qué planificación hemos llevado a cabo para la realización del mismo.

1.1. Contexto

Desde hace algunos años en la Universidad de La Laguna se han estado realizando unas jornadas denominadas “Las Olimpiadas del pensamiento computacional”, las cuales a día de redacción de este documento van ya por su sexta edición. Estas jornadas tienen como objetivo el enseñar a jóvenes de los centros educativos más cercanos a la universidad las bases de la programación, así cómo hacer hincapié en la importancia de desarrollar sus capacidades en el pensamiento computacional y facilitarles el familiarizarse con los conocimientos necesarios para tener un correcto desempeño en este campo.

Debido a esto surge la necesidad de desarrollar un sistema que permita a los monitores de dichas jornadas, el crear pruebas para puntuar a los estudiantes de las mismas, y que dicho sistema sea capaz de validar el resultado obtenido por cada uno de ellos.

1.2. Antecedentes y estado actual del tema

Para poder hacer un correcto desarrollo del proyecto fue importante el investigar primero acerca de sistemas, herramientas o cualquier otro medio con características parecidas a las que queríamos para nuestro sistema. Debido a esto primero hubo que definir correctamente que se consideraba cómo un sistema de validación.

1.2.1. Sistemas de validación

Los sistemas de validación de código son recursos que están cada vez más presentes en todo tipo de ámbitos, especialmente aquellos relacionados con la educación. Estos sistemas permiten recibir una entrada, la cual puede ser un código de programación escrito en algún lenguaje específico, y luego a través de un lógica interna del sistema se procesa dicha información, se ejecuta y se puntúa. De esta manera se consigue hacer un sistema capaz de valorar la calidad del código introducido.

1.2.2. Herramientas y plataformas

Cómo bien habíamos mencionado, especialmente en el ámbito académico existen muchas plataformas que usan este tipo de prácticas de cara al evaluar distintos tipos de código, tales como pueden ser: Jutge, Codecademy o en general cualquier sistema digital para aprender a programar. Para hacer esto se apoyan principalmente en una serie de herramientas cómo pueden ser las CVT (Code validation tools) [14], estas herramientas o subprogramas toman registro de varias entradas cómo son el código introducido, el resultado obtenido, y el resultado esperado, para en base al mismo desarrollar una tabla con toda la información que se obtenido tras desarrollar dicho procesamiento.

Algunos ejemplos de estas herramientas puede ser por ejemplo Exli [4], esta se trata de una herramienta inteligente la cual recibe cómo entrada una serie de pruebas unitarias, y a partir de las mismas desarrolla “inline tests” (pruebas en línea que son parte de un sistema automatizado), para tomar registro de la calidad de los códigos introducidos. Para usarla correctamente primero es necesario cumplir con una serie de requerimientos como son por un lado el tener al menos 10GB de espacio disponible, y por el otro el tener instalado Docker, el cual a su vez deberá tener instalado características cómo: Conda (última versión), Python 3.9 o posterior, Java 8, JUnit 4 y Maven 3.8.3 o posterior. A partir de aquí simplemente es cuestión de instalar y configurar el entorno, por lo que bastaría con crear el Docker usando:

```
#  docker build -t exli .
#  docker run -it exli /bin/bash
```

Para luego crear un ambiente de Python para la herramienta Exli usando:

```
#  cd exli/python && bash prepare-conda-env.sh
#  conda activate exli
```

Llegados a este punto se recomienda visualizar el repositorio en cuestión para ver todo el proceso de cómo ejecutar correctamente, aunque cabe resaltar que el programa te da una breve descripción si se invoca con la opción –help. El porqué de mencionar Exli, es debido principalmente a que una herramienta con estas características se podría implementar o al menos sus funcionalidades en nuestro sistema, específicamente a la hora de llevar a cabo las pruebas del mismo.

Esta no es la única herramienta que existe, otro claro ejemplo es CPLAS (C Programming Learning Assistant System)[16]. CPLAS es un sistema para aprender C, el cual su lógica se basa en el uso de varios ficheros para evaluar la calidad del código del usuario. Para ello cuenta con un fichero con el código fuente, otro que recibe los posibles errores, otro con el resultado esperado, el fichero de entrada, el de salida y finalmente el de salida de los test. Con toda esta información se procede a desarrollar una tabla en la cual se registra la información del usuario y valora la calidad de la misma, basándose en las pruebas superadas y el tiempo de CPU invertido. Uno de los motivos principales de porqué mencionar esta herramienta, es por que tal y como se ha explicado, esta comparte muchas similitudes con los objetivos que deberá cumplir nuestra aplicación web una vez finalizada. Volviendo a esta herramienta en cuestión, cabe mencionar que este sistema de ficheros y pruebas gestionan la lógica que hay detrás de la página web que lleva el manejo de los ejercicios, desafortunadamente esta no dispone de un código fuente abierto,

pero se trata de una buena referencia del tipo de sistema que queremos llevar a cabo, el cual explicaremos más adelante. Además, desde dicha página se puede acceder a un menú superior el cual cuenta con enlaces a otras pruebas de distintos tipos de lenguajes de programación, característica que de ser necesario se podría aplicar a nuestro sistema si así se quisiese.

Por otra parte, otro ejemplo de una herramienta parecida es APLAS (Android Programming Learning Assistance System)[18], esta se utiliza para enseñar a desarrolladores los conceptos básicos de programación para Android. La lógica de su sistema es bastante parecida a lo mostrado en los ejemplos anteriores, sin embargo esta difiere en que en su caso usa una serie de test unitarios para validar las distintas pruebas que se le ponen a los usuarios. Así mismo, para el uso de la misma se usa como intermediario un sistema web con un servidor, el cual recibe la entrada que hace el usuario o estudiante y mediante una fórmula matemática le pone una nota acorde al trabajo que este haya hecho. El porqué de mencionar esta herramienta es debido a que de cara a suministrar un servicio adecuado a nuestro sistema, es necesario establecer un servidor para poder gestionar las pruebas y sus puntuaciones adecuadamente, además de la posibilidad de mantener un registro de las acciones que se realizará sobre la aplicación.

Actualmente otro tipo de sistemas con características similares a las que buscamos son aquellos que involucran el uso de Autograder [24]. Esta herramienta se utiliza para corregir código mediante pruebas, permite ser altamente personalizable, además de que tiene la posibilidad de detectar incluso plagio, característica que no es necesaria en principio para nuestro sistema, pero que puede ser una buena referencia por si se quisiese añadir en un futuro, aparte de las características antes mencionadas que de por sí encajan bastante con el producto final que queremos realizar. Para utilizar la misma basta simplemente con tener instalado los compiladores o intérpretes para los lenguajes de programación que acepta siendo estos: Java, C, C++ y CPython desde la versión 3.8 hasta la 3.11. Es importante mencionar que se puede usar perfectamente tanto en Linux cómo en OS X, pero en Windows puede fallar en caso de usar pruebas que incluyan el uso de Shebang. Dicho esto para instalarlo y usarlo base con usar los siguientes comandos:

```
# pip install autograder
# pip install -U --no-cache-dir autograder
# autograder guide directorio/donde/quieres/crear/todo
```

Tras aplicar este último comando no solo crearemos el entorno para usar la herramienta, si no que el asistente nos permitirá configurar todo incluyendo el lenguaje que queremos implementar así como pruebas de ejemplo para el mismo.

Por otra parte otro ejemplo pero en este caso a un nivel superior de herramientas que sirven de cara a la validación de código, es el uso de LLM4VV [11]. El mismo se basa en la práctica del uso de LLM (Large Language Models) para crear pruebas dirigidas, estas aprovechan el uso del lenguaje humano mediante prompts recurriendo por ejemplo a la API de modelos de IA cómo puede ser el caso de ChatGPT, para desarrollar pruebas en base a las necesidades que se les planteen. El motivo de mencionar la misma, es por si se viese necesario añadir algún tipo de funcionalidad que utilice IA de cara al uso de nuestra aplicación, sabiendo que existen modelos o herramientas como LLM4VV, se podría tener un buen punto de partida para en caso de quererlo intentar añadir sus funcionalidades a nuestra aplicación web. Por otra parte de cara a su instalación como tal no se indica

ningún tipo de requisito previo, simplemente bastaría con acudir a su repositorio para clonar la información del mismo y desde ahí operar con él.

Tras haber hecho una revisión acerca de las diferentes herramientas y sistemas, podemos sacar en clave una serie de características con las que compararlas, las cuales reflejaremos en la Tabla 1.1.

Nombre sistema	Tipo de pruebas	Uso de IA	Plataforma web	¿Qué es?	Dificultad
Exli	Inline Test	No	No	Herramienta	3
CPLAS	Unit Testing	No	No	Plataforma	2
APLAS	Unit Testing	No	No	Plataforma	2
LLM4VV	PPP	Si	No	Herramienta	4
Autograder	Unit Testing	No	No	Herramienta	1

Tabla 1.1: Comparativa de herramientas evaluadas

1.2.3. Características de nuestro sistema y comparativa con los antes mencionados

Para finalizar y volviendo a las características del sistema partiendo de lo antes visto, nuestra aplicación debería poder diferenciar entre docentes y usuarios. Haciendo que los docentes sean los cuales introduzcan las pruebas a realizar, así cómo los tests de validación que tendrían que pasar las mismas, además de las puntuaciones que estos vean pertinentes para cada prueba. Por otro lado los usuarios serían aquellos que harían las pruebas desarrolladas por los anteriores, para una vez finalizada la experiencia poder dar una puntuación a los mismos en base a las pruebas superadas y el tiempo requerido. Teniendo pruebas cuya lógica sea parecida a la de las pruebas unitarias antes mencionadas en los sistemas de la sección anterior, para poder realizar correctamente una plataforma que sea gestionada a través de nuestra aplicación web.

1.3. Objetivos

A raíz de lo explicado en el punto anterior, el objetivo principal de este trabajo es el de realizar un análisis de los sistemas de ejecución, validación y puntuación de código que existen actualmente, así cómo ahondar en los posibles antecedentes que haya sobre este tipo de sistemas. Para luego establecer una propuesta de sistema en base a la información recopilada y que esta pueda ser implementada en unas futuras jornadas.

Este sistema se tratará de una aplicación web la cual permitirá tener tres tipos de usuarios. Por un lado estarán los monitores y administradores, estos tendrán la posibilidad de desarrollar las pruebas, así cómo de establecer puntuaciones para las mismas, teniendo aparte los administradores la posibilidad de tener acceso a gestionar usuarios y la memoria interna del sistema. Por otro lado estarán los usuarios, los cuales se tratará de los estudiantes, que deberán poder visualizar las pruebas desarrolladas por los docentes, introducir el respectivo código en el sistema, para que luego este sea evaluado de manera automática por el mismo, dando una puntuación al trabajo del usuario en base al número de pruebas superadas y el tiempo empleado. Registrando el tiempo de este último solo en caso de pasar todas y cada una de las pruebas de una determinada pregunta, para en caso de empate poder ordenar las puntuaciones obtenidas adecuadamente.

1.4. Planificación

Para poder llevar a cabo correctamente el proyecto tuvimos que desarrollar una planificación la cual se basó en una serie de fases o tareas, en pos de garantizar el correcto desarrollo del mismo, las cuales fueron las siguientes:

1.4.1. Revisión bibliográfica

Cómo primera tarea se planteó la necesidad de hacer una revisión bibliográfica acerca de todos los tópicos que se puedan considerar interesantes de cara a la realización del proyecto, comenzando por: ¿Qué es un sistema de validación?, ¿Qué características tiene?, ¿Qué ejemplos existen?, ¿Qué tipo de tecnologías usan?, etcétera.

1.4.2. Diseño de requisitos

En esta fase del proyecto se tuvo que diseñar los requisitos del mismo, estipulando que lenguajes de programación se utilizarían para el desarrollo de la aplicación, así cómo para el control de sus pruebas, además de esbozar un diseño de cómo se vería dicho producto final y mostrar una primera versión del mismo.

1.4.3. Implementación del sistema

Durante el desarrollo de esta fase, se indicó que para cumplir la tarea era que necesario que el sistema sea capaz de crear las distintas pruebas, siendo posible el poder introducir un título, descripción de lo que se pide, y la posibilidad de introducir código por pantalla y que este sea registrado de manera interna por el sistema, para que luego en la siguiente tarea poder crear el sistema de puntuación y pruebas.

1.4.4. Incorporación de sistema de pruebas y validación

Esta tarea fue destinada al desarrollo de las pruebas del sistema de validación, crear el sistema de puntuación en base al superar las mismas y la gestión del tiempo para en caso de empate. Además de la posibilidad de poder gestionar un ranking con las puntuaciones de los distintos usuarios en base a las preguntas, así cómo de cualquier posible mejora al sistema.

1.4.5. Documentación

Finalmente cómo fase final se estableció esta última tarea donde se generaría la documentación del proyecto tanto del código, cómo de la memoria, así cómo el desarrollo de la presentación final del proyecto sumado al video y guión de la misma.

Capítulo 2

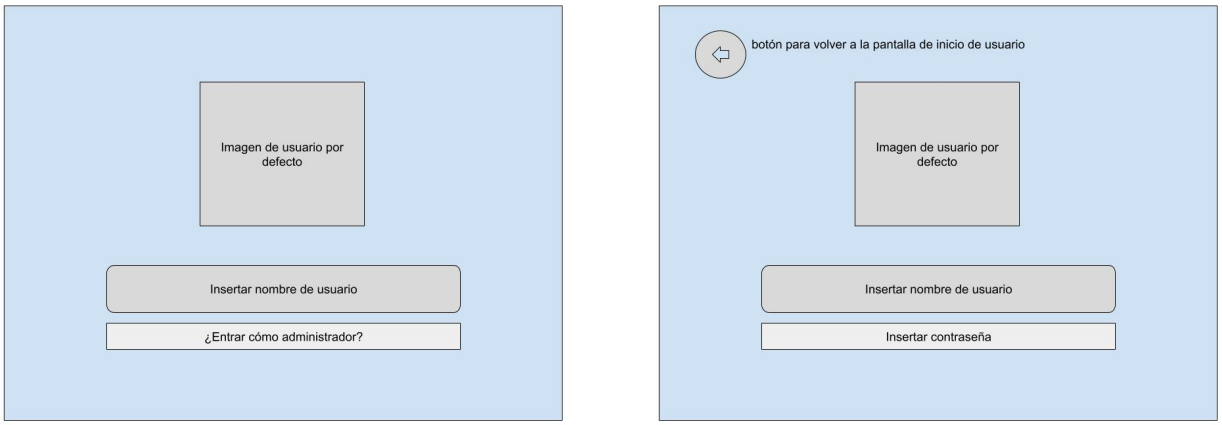
Diseño del sistema

En el capítulo anterior hemos hablado acerca del porqué de la realización del proyecto así como de sus antecedentes. En este entraremos más en detalle en el diseño del mismo, así como en tópicos cómo con qué arquitectura está diseñado, que tecnologías han sido utilizadas, o como el sistema para manejar sus datos.

2.1. Arquitectura del sistema

En el capítulo anterior se hizo referencia a las características que consideramos oportunas para nuestro sistema, donde es importarte resaltar que una de las herramientas que más interesante y parecida nos pareció a nuestra idea original fue la de la plataforma de CPLAS, por lo que a la hora de desarrollar la arquitectura de nuestro sistema “CODE-ULL”, la lógica y diseño de la misma está muy presente, debido a que esta también era un sistema basado en bloques de preguntas.

Partiendo de esta idea, ahora era cuestión de determinar el cómo se gestionaría todo lo relacionado con la arquitectura, por ello se hicieron una serie de bocetos iniciales que más adelante conformarían las distintas pantallas que gestionarían la aplicación web. Para ello se pensó primero en una pantalla inicial que permitiese el gestionar tanto los docentes cómo a los usuarios, cómo se podrá ver a continuación en la Figura 2.1:



Boceto pantalla de inicio (usuario) Boceto pantalla de inicio (admin)

Figura 2.1: Bocetos de cómo se gestionarían las pantallas de usuario

Cómo pueden ver la idea era simple, mostrar una pantalla de registro la cual originalmente te dejase entrar como un estudiante, pero que tuviese una opción para en caso de quererlo, registrarte cómo docente, siendo el caso de la imagen de la derecha, situada en la anteriormente mencionada Figura 2.1.

En caso de acceder mediante cualquiera de estas pantallas, se llevaría al usuario a una pantalla desde donde gestionar los bloques de la aplicación, más adelante veremos que esta idea cambiaría ligeramente para el caso de los administradores, pero volviendo la pantalla de bloque esta tendría la estructura reflejada en la Figura 2.2:

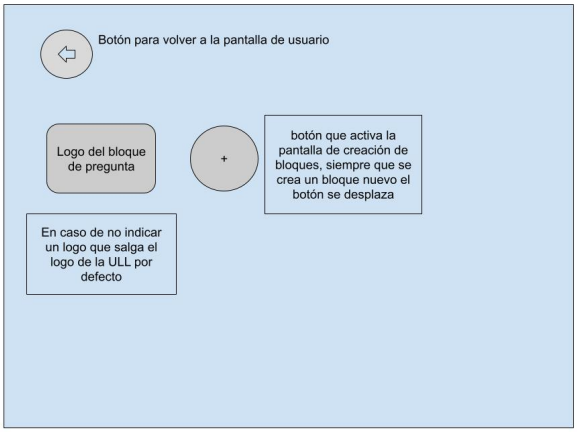


Figura 2.2: Boceto pantalla creación de bloques

En ella en el caso de los administradores podrían ver cada entrada a los bloques creados situados a lo largo de la pantalla, mostrando cada bloque en base a un logo, el cual en el momento de la creación se trataría del logo de la universidad por defecto. Aparte ellos podrían seguir creando dichos bloques al recurrir a un botón extra el cual se ubicaría siempre al lado de la última entrada creada, teniendo tambíen la página una opción para volver a la pantalla de inicio. Por su parte los estudiantes podrían visualizar lo mismo con la diferencia de que estos no tendría la posibilidad de crear nuevos bloques.

En caso de seleccionar alguno de estos bloques, era necesario tener algún tipo de pantalla intermedia la cual permitiese tanto crear las preguntas cómo listarlas, debido a esto se planteó la idea reflejada en la Figura 2.3:

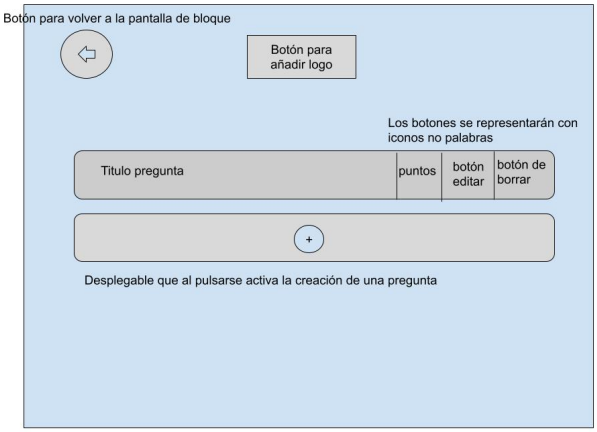


Figura 2.3: Boceto pantalla bloque interno

Originalmente se había planteado que las preguntas se creasen mostrando el titulo, los puntos y otra serie de elementos para editar y borrar, además de que dicha pantalla intermedia permitiera el poder modificar el logo asociado al bloque. Más adelante veremos que esta idea se cambió ligeramente. Por su parte cada entrada de esta pantalla debía estar asociada a las preguntas en sí, las cuales debían tener un título, descripción y la posibilidad de guardar y registrar toda su información, cómo se verá a continuación en la Figura 2.4.

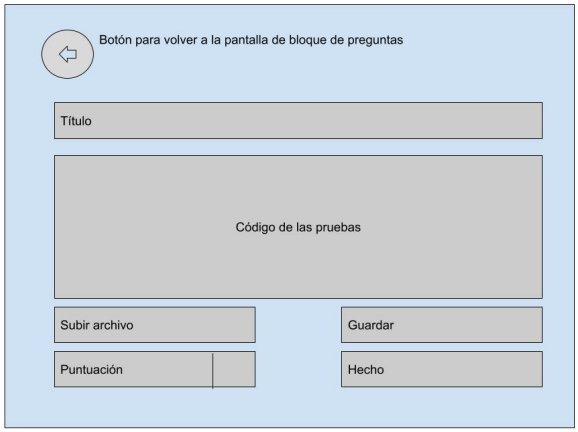


Figura 2.4: Boceto pantalla creación de preguntas

Esta pantalla sería igual en el caso de ser un usuario, solo que la diferencia principal de la misma vendría a la hora de subir el código, aparte de que de los mismos no podrían ver la puntuación de cada pregunta. Finalmente uno de los últimos bocetos que se plantearon acerca de las futuras pantallas de la aplicación, fue el cómo se gestionarían los rankings, el cual se podrá ver reflejado en la Figura 2.5.

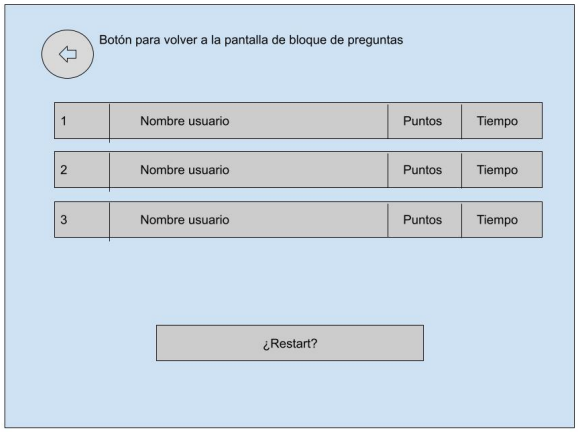


Figura 2.5: Boceto pantalla de rankings

Esta pantalla debería tener una opción para volver al apartado de bloques y en la misma se listaría cada jugador mostrando su posición, nombre, puntos y tiempo tardado en resolver cada bloque. Para concluir cabe mencionar que en próximos apartados de la memoria veremos que muchos de estos diseños cambiaron debido a nuevas necesidades surgidas, además de que también se decidió crear pantallas extras para distintas funcionalidades que serán explicadas más adelante.

2.2. Tecnologías utilizadas

Para poder llevar a cabo el desarrollo en sí, fue necesario determinar los lenguajes en los que desarrollar la aplicación, por ello, se optó principalmente por el uso de Javascript y CSS para gestionar todo aquello relacionado con la parte Frontend, es decir, lo visual de la página, y dejar lo interno (Backend) en manos de algún lenguaje que facilitase las conexiones con un servidor así como de manejar archivos, el cual se trató de Python.

Partiendo de estas ideas, se investigó acerca de herramientas o frameworks que facilitasen el trabajo, dando lugar a que a la hora de desarrollar cualquier aspecto relacionado con la interfaz se usase React, el cual es un framework de Javascript que permite desarrollar páginas en base a componentes. Además este tenía la ventaja de que era compatible con la tecnología principal que se decidió que gestionaría la lógica interna, la cual se trató del framework de Flask, el cual permite gestionar servidores, así como hacer peticiones a los mismos. Por otro lado, otra ventaja que proporcionaba Python era que es un lenguaje que aporta una gran cantidad de librerías para manejar ficheros, directorios o incluso expresiones regulares, las cuales serían utilizadas para funcionalidades que se explicarán más adelante.

2.3. Manejo de datos

Llegados a este punto es obvio pensar que para que un proyecto así funcione correctamente es necesario tener alguna manera de gestionar su información, debido a que en el mismo se manejan bloques, preguntas, usuarios... Por esto mismo se plantearon varias opciones de cómo se gestionaría dicha base de datos, recurriendo finalmente a una mezcla entre dos principales medios.

Por un lado y originalmente, se había recurrido a la opción del localStorage que proporciona el propio framework de React para gestionar todo lo relacionado con la memoria, debido a que este permitía que la propia página en la que se desarrollaba la aplicación gestionará todo. El problema de recurrir únicamente a este, es el hecho de que hacía que el programa fuese poco portable, además de que podía dar lugar a errores a la hora de gestionar ciertas operaciones del sistema, por lo que el uso de localStorage se limitó únicamente para ciertas operaciones relacionadas con las sesiones de los usuarios, cómo por ejemplo conservar el nombre de los estudiantes, dejando todo lo relacionado con gestionar la información de los docentes, así cómo de las pruebas en el otro medio principal para gestionar la información, siendo este las estructuras JSON (JavaScript Object Notation).

De cara a poder hacer lo más robusto y exportable posible el sistema que gestionaba la información, se optó por que el propio servidor recurriese a crear, leer y modificar archivos JSON con la información de la aplicación. Esto debido a que Python da la posibilidad de usar con mucha facilidad librerías relacionadas con el manejo de expresiones regulares y ficheros. Lo cual permite que a la hora de registrar la información de ya sea un usuario, un bloque o una prueba, sea posible guardar en estos JSONs toda la información recopilada, permitiendo que luego la parte de React se capaz de recibir toda esta información y procesarla facilmente debido a que es un tipo de estructura muy familiar con Javascript. Además un extra añadido del porque situar todo en base a JSONs, fue que hacía mucho más fácil el manejar y registrar toda la información asociada a los rankings, botones y la propia información de los usuarios.

Capítulo 3

Implementación

En capítulos anteriores ahondamos en el porqué de la creación del sistema, así como de antecendentes y la estructura del mismo, La intención de este capítulo es la de abordar todo lo relacionado con la implementación del sistema, su manera de uso y las consideraciones necesarias para entender el correcto funcionamiento del sistema.

3.1. Sistema

Cómo bien se explicó anteriormente la aplicación cuenta con tres tipos de usuarios principales. Siendo estos por un lado los administradores y monitores (docentes), y por el otro los estudiantes (usuarios). Cuando se accede al sistema las primeras pantallas que se podrán apreciar serán las de registro. Estas tienen la función de dar acceso a los usuarios o a los docentes en caso de introducir sus credenciales correctamente, cómo se podrá ver reflejado en la Figura 3.1:

Pantalla de inicio (usuario) Pantalla de inicio (admin)

Figura 3.1: Pantallas de inicio

Y según el rol con el que se haya accedido al sistema, se podrán tener una serie de funcionalidades u otras. Las cuales empezaremos a explicar en las siguientes secciones, comenzando por el funcionamiento de la aplicación para los usuarios.

3.1.1. Actuando como usuario

En el caso de actuar cómo un usuario, este por defecto será enviado a la pantalla general de bloques, desde la cual podrá seleccionar uno de estos para comenzar a realizarlo, dicha pantalla se verá reflejada a continuación en la Figura 3.2

Figura 3.2: Pantalla bloques general (usuario)

Cabe mencionar que aunque es posible retroceder en la aplicación usando el botón de retroceder de cualquier página web, se recomienda utilizar el botón de “volver a la página” anterior que proporciona la aplicación, ya que en caso de estar navegando por varias pantallas el hacer esto facilita el movimiento.

Volviendo a la pantalla en sí, si se selecciona uno de los bloques creados lo que pasará es que se redirigirá al usuario a la página de interna de los bloques, la cual se puede ver reflejada en Figura 3.3.

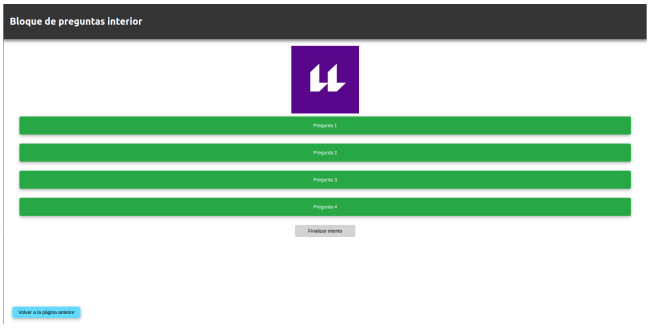


Figura 3.3: Pantalla bloques interna (usuario)

Desde esta página el usuario podrá seleccionar cada una de las preguntas creadas por los docentes, además de también contar con un botón el cual permite finalizar con el cuestionario enviando al usuario a la página de rankings, la cual será explicada en los apartados siguientes. Volviendo a la página en sí, esta solo sirve para que los usuarios puedan seleccionar las preguntas a resolver, por lo que en caso de seleccionar alguna se pasaría a la pantalla de preguntas, la cual se refleja en la Figura 3.4.

Figura 3.4: Pantalla pregunta (usuario)

Esta pantalla tiene cómo principal objetivo mostrar la información de las preguntas que los docentes hayan creado, por ello cuando un usuario accede a la misma su único

deber es leer la descripción de lo que se pide en la prueba y subir un archivo a la misma, donde la extensión de dicho archivo podrá ser tanto en Python, cómo C, C++, Ruby o Javascript. Y una vez subido el mismo deberá darle al botón de confirmar reflejado en la Figura 3.4, para poder corregir dicha entrega.

En caso de resolver todas las pruebas asociadas a la pregunta el sistema notificará con un mensaje de felicitaciones y redirigirá automáticamente al usuario a la pantalla del interior de bloques vista en la Figura 3.3. En caso de no pasar todas las pruebas se notificará indicando que no se ha completado del todo la pregunta y se mantendrá en la misma página, aunque el usuario siempre podrá salir de la pregunta e intentarlo en otro momento. Cómo última aclaración se indicará que en caso de no introducir un código con las extensiones antes mencionadas, se tomará cómo no resuelta ninguna de las pruebas y por tanto no se puntuará esa pregunta hasta añadir un código que sea correcto y acorde a lo antes mencionado.

3.1.2. Actuando como docente

Volviendo ahora al caso de los docentes este es un poco distinto. Para empezar, en el caso de registrarse con éxito se pasa a una pantalla, la cual se trata de un panel de control, el cual se podrá ver en la Figura 3.5.

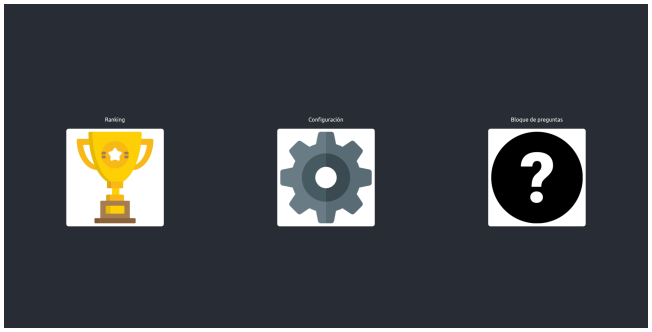


Figura 3.5: Panel de control

En esta pantalla se podrán ver tres botones principales, los cuales son el asociado a la pantalla de rankings, la cual será explicada en el próximo capítulo, la pantalla de configuración y la pantalla de los bloques de preguntas. Es importante mencionar que esta vista es para el caso de aquellos docentes que sean “administradores”, la única diferencia entre ellos y los “monitores”, es que los monitores no tienen acceso a la zona de configuración, por tanto no podrán verla.

Volviendo a la pantalla en sí, en caso de acceder a la zona de preguntas se pasará a la pantalla de bloques general, pero con los permisos de un docente tal y cómo se puede ver en la Figura 3.6.

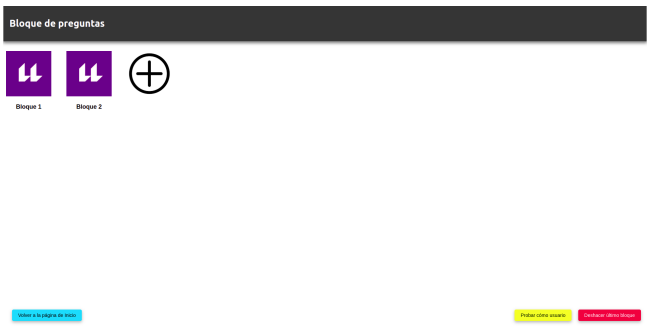


Figura 3.6: Pantalla bloques general (admin)

Desde esta misma para que un docente pueda crear un bloque simplemente deberá recurrir al botón con el símbolo del “más”, en caso de pulsarlo el sistema de manera automática irá desarrollando las operaciones necesarias para preparar los directorios y archivos de registro asociados a cada bloque. Cabe mencionar que en el momento de que haya al menos un bloque se activará un botón que permitirá borrar el último creado, y que independientemente de esto tanto en esta página cómo en las siguientes, existirán dos botones extras, uno para volver a la página anterior, en este caso se pasa directamente a la página de registro vista en la Figura 3.1 y otro para actuar temporalmente cómo un usuario y ver todo lo mencionado en la sección anterior.

En caso de seleccionar un bloque se pasaría a la página de bloques interna, la cual se puede ver en la Figura 3.7.

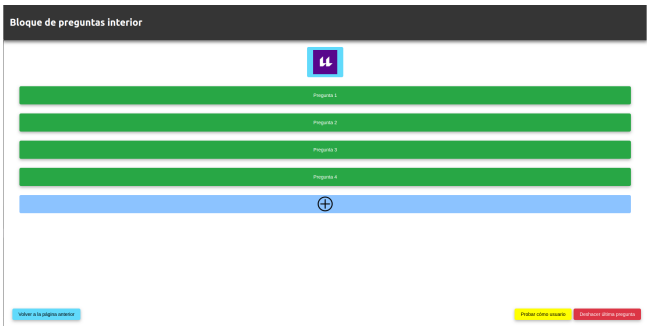


Figura 3.7: Pantalla bloques general (admin)

Esta página tiene funcionalidades muy simples, desde aquí los docentes en el caso de pulsar el botón del “más”, generarán nuevas entradas que se trataran de botones, los cuales darán acceso a las distintas preguntas generadas. Así mismo, otra particularidad de esta página es que en la parte superior de la misma se situa un botón desde el cual se podrá modificar la imagen del bloque asociado, y además existe el respectivo botón para borrar las últimas preguntas creadas.

Partiendo de la idea de la página anterior, en caso de pulsar alguna de estas entradas se accederá a la página de preguntas, la cual se podrá ver en la Figura 3.8.

Figura 3.8: Pantalla preguntas (admin)

Esta pantalla permite a los docentes gestionar las entradas a las preguntas, en ella para poder registrar una pregunta será necesario que el usuario le de un título, descripción y cómo mínimo una prueba, la cual estará basada en un fichero de entrada, un fichero resultado y un número con los puntos asociados, todos estos apartados serán explicados en detalle más adelante, pero de momento solo mencionar que el número de pruebas es ilimitado y que una vez añadida toda esta información, será importante pulsar el botón de confirmar para registrarlo todo con éxito.

3.1.3. Gestión interna

Esta sección esta destinada a explicar todo lo relacionado con las funciones del administrador, a excepción de todo lo relacionado con la creación de bloques y preguntas, visto en la anterior sección. Si volvemos a la Figura 3.5 en la cual se podía ver un panel de control, si accedemos a la pantalla de configuración nos redirigirá a la pantalla que se ve en la Figura 3.9.

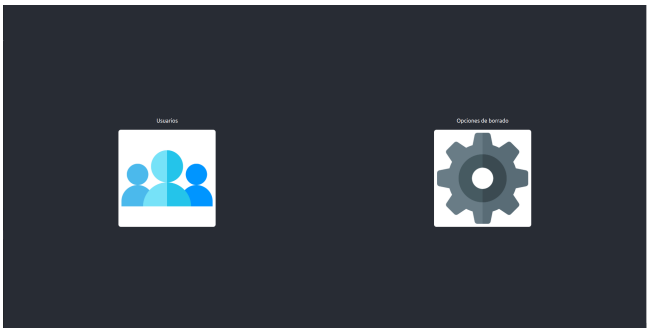


Figura 3.9: Opciones

Desde esta misma se accede a las funcionalidades extra con las que cuenta el administrador, las cuales son por un lado la gestión de usuarios y por el otro las opciones de borrado. Por la parte de la gestión de usuario en caso de pulsarse se pasaría a la siguiente pantalla, reflejada en la Figura 3.10.

Figura 3.10: Creación de usuarios general

Esta misma sigue una lógica parecida a la pantalla de bloques general, donde cada entrada en lugar de ser un bloque hace referencia a un usuario. Así mismo, desde aquí los administradores podrán pulsar el botón de “más” para añadir usuarios, así como acceder a los mismos para modificar sus datos. Por otro lado y por motivos de seguridad, existe un usuario por defecto el cual se trata de “admin”, al cual se podrá modificar su contraseña, pero nunca se podrá borrar su registro o modificar su rol, en caso de intentar esto el sistema notificará con error.

Entrando más en detalle acerca de cómo se crea un usuario, en caso de pulsar el botón de “más” o de seleccionar un usuario en concreto, se accederá a la pantalla de la Figura 3.11.

Figura 3.11: Creación de usuarios interna

Desde esta misma se podrá dar imagen, nombre y contraseña a un usuario, además de que podrá seleccionar un rol para el mismo, pudiendo elegir entre “administrador” y “monitor”. Para concluir con este apartado, mencionar que solo se podrá modificar o crear la información del mismo en caso de pulsar el botón de confirmar.

Finalmente y recapitulando a la pantalla de opciones vista en la Figura 3.9, en caso de seleccionar el apartado de opciones de borrado se llegará a la pantalla de la Figura 3.12.

Figura 3.12: Opciones de borrado

Desde esta pantalla los administradores contarán con una serie de botones los cuales permiten borrar la información de los estudiantes, la información de los bloques de pregunta, y todo lo relacionado con los usuarios registrados, a excepción del antes mencionado usuario “admin”, el cual es un usuario que viene por defecto con el sistema.

3.2. Métodos de uso

Una vez explicado todo lo relacionado con el sistema es importante hacer mención a una serie de detalles, comenzando es importante que para que todo funcione correctamente tanto la parte Frontend (todo lo relacionado con React), cómo la parte Backend (todo lo relacionado con Flask) ambas esten activadas. Para hacer esto es necesario situarse en el origen del repositorio y realizar los siguientes comandos para activar las funcionalidades de React y Flask respectivamente:

```
#  npm start
#  python3 server.py
```

Así mismo, se recomienda que para usar todo perfectamente se tenga dos terminales activas corriendo cada comando, además de que una vez terminada la actividad con el proyecto, cancelar la ejecución de los mismos en lugar de cerrar la terminal directamente, para evitar que se esté ejecutando en segundo plano. Finalmente en el archivo principal del repositorio, el cual es denominado “App.js”, este contiene en su parte superior una constante denominada “route to server”, en la cual figura la IP de la máquina que ejecuta la aplicación web, debido a esto y en pos de hacer un correcto uso de la aplicación, se deberá modificar la dirección de dicha IP, por la de la máquina invocante.

3.3. Dirección del repositorio

En este último apartado podrán acceder a todo lo relacionado con el proyecto, accediendo al código a través del siguiente enlace: https://github.com/JoseLozanoArmas/TFG.git

Capítulo 4

Validación y puntos

En este capítulo se entrará más en detalle acerca del cómo se gestionan las distintas validaciones y puntuaciones del sistema, así como las características de sus pruebas.

4.1. Formato de las pruebas

Cómo bien se explicó anteriormente, la creación de las pruebas cae en manos de los docentes, ya sean monitores o administradores. Cómo se vió en la Figura 3.8 las pruebas se tratan de dos ficheros a los que se le asocia una puntuación. Estos ficheros se tratan por un lado de un fichero con la entrada que deberá recibir los programas de los usuarios, y por el otro de un segundo fichero el cual tendrá el resultado esperado de la salida de dichos programas. Para facilitar su uso, se recomienda que ambos ficheros dispongan de una extensión “txt” para facilitar su uso. Así mismo, en ambos casos los ficheros deberan tener entradas que sigan el siguiente formato:

```
info_1
info_2
info_3
```

Donde cada “info” hace referencia a una posible entrada del fichero en caso de ser el fichero de entrada o un resultado esperado en el caso de ser el fichero resultado, siendo posible introducir más de una entrada por línea. Es importante mencionar que independientemente de cuantas entradas se introduzca por fichero, así cómo de cuantas pruebas se creen, será responsabilidad de los usuarios de que tengan su código preparado para que este reciba un fichero cómo entrada y que a la hora de corregir se tomará en cuenta lo que ellos impriman por pantalla.

4.2. Lógica de la corrección

Cómo se explicó en el punto anterior las pruebas se basan en dos ficheros, estos ficheros una vez subidos el sistema registrará sus direcciones en un JSON junto con la puntuación de la prueba, donde dicho JSON tendrá el nombre de la pregunta en cuestión y será situado en la carpeta del bloque que se está desarrollando. Esto se hace con la idea de facilitar las labores de corrección, debido a que el sistema de la aplicación lo que hace manera interna es lo siguiente:

La corrección comienza cuando los docentes tienen registradas sus pruebas y un usuario, es decir, un alumno, sube su código a una pregunta y pulsa el botón de confirmar. Cuando esto se hace se invoca un método el cual recibirá la identificación del bloque y de la pregunta, sumado a la entrada del usuario. Con todo esto, se adquiere la entrada del usuario, así como el JSON antes mencionado de registro, para así poder leer y procesar los ficheros de prueba, los cuales irá evaluando uno a uno. Acto seguido, el sistema mediante el uso de una librería de Python, la cual es denominada subprocess, ejecutará la entrada del usuario en cuestión para luego evaluarla. Es importante aclarar que cómo se explicó anteriormente, cada usuario debe adaptar su código para que pueda procesar el fichero de entrada. Una vez recibido el fichero y la entrada del usuario, se ejecuta y se hace una captura del resultado de la terminal, la cual se compara con el resultado esperado indicando si se ha pasado la prueba o no, este proceso a su vez se repite con cada una de las pruebas registradas en dicha pregunta.

Llegados a este punto es importante hacer unas últimas aclaraciones y es que solo en caso de superar todas las pruebas se registrará el tiempo en las que se resolvieron, para en caso de empate poder ordenar los rankings adecuadamente. En caso de que esto pase se notificará al usuario y se le redirigirá a la página interna de las preguntas vista anteriormente en la Figura 3.3, para que pueda continuar con el desarrollo del bloque. En caso de no haber pasado todas las pruebas se mandará una notificación al usuario y se le mantendrá en la página por si se diese el caso de querer intentarlo de nuevo. Registrando únicamente los puntos obtenidos de las pruebas superadas, más no el tiempo en el que se resolvió por lo explicado anteriormente.

Para finalizar mencionar que todos aquellos códigos que no respeten las extensiones antes mencionadas, que tengan errores de compilación (en el caso de los lenguajes compilados) o errores de sintaxis que impidan su correcta ejecución, serán considerados cómo **erróneos**.

4.3. Rankings

Finalmente una vez explicado todo lo relacionado con la corrección y pruebas podemos retomar todo lo relacionado con los rankings, lo cual se quedó pendiente cuando explicamos los elementos del panel de control, localizado en la Figura 3.5.

Si nos fijamos en el mismo veremos un botón que hace referencia a los rankings, el cual al pulsarse nos mandará a la pantalla general desde donde se visualizan los mismos, la cual se refleja en la Figura 4.1.

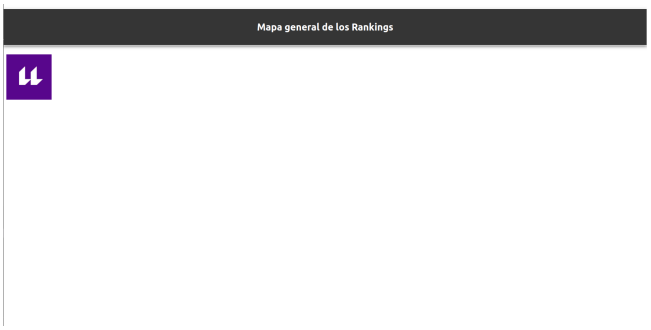


Figura 4.1: Pantalla general de los rankings

Esta pantalla solo es visible para los docentes y en ella se muestran solo los bloques creados, donde si se pulsa alguno de estos, se redirige al docente en cuestión a una pantalla interna donde se reflejan las puntuaciones obtenidas en dicho ranking, cómo se podrá ver en la Figura 4.2.

Figura 4.2: Pantalla interna de los rankings

Cómo último apunte mencionar que los usuarios tambíen pueden acceder a esta pantalla al pulsar el botón de “finalizar intento”, localizado en la pantalla interna de bloques que se pudo ver en la Figura 3.3. Así mismo la información de esta página se irá actualizando a medida que la actividad se vayan realizando y los usuarios vayan llevando a cabo las distintas preguntas diseñadas previamente por los docentes.

Capítulo 5

Conclusiones y líneas futuras

En conclusión el sistema realizado se trata de una aplicación web, la cual permite dar la posibilidad a los docentes de crear preguntas distribuidas en base a bloques, donde cada docente podrá introducir pruebas con sus respectivos tests, para que luego los alumnos puedan introducir código para resolver dichas preguntas. Haciendo que el sistema una vez se haya subido el código, sea capaz de ejecutar, validar y puntuar la entrada de cada usuario, reflejando en un ranking cada puntuación de los alumnos, así como la hora en la que pasó las pruebas en caso de resolver todos los tests de las mismas.

Así mismo cómo bien se explicó antes, CODE-ULL es un sistema cuya estructura principal parte de la misma que muchas herramientas antes mencionadas cómo CPLAS, APLAS, etcétera. Los cuales demostraron ser buenas herramientas para enseñar en sus respectivos campos, facilitando una gran cantidad de tareas relacionadas con la docencia.

De cara al futuro, mencionar que dicho sistema a su vez tiene un diseño exportable y fácil de usar, debido a que cuenta con un interfaz de fácil uso y un sistema de registro en base JSONs. Debido a esto se espera que el sistema desarrollado sea posible de implementar en unas futuras jornadas de pensamiento computacional, facilitando así las labores del profesorado a la hora de realizar las mismas, y ayudando a mejorar la experiencia de aquellos que quieran participar tanto en las jornadas cómo en el uso del proyecto.

Capítulo 6

Summary and Conclusions

In conclusion, the system is a web application, which allows teachers to create questions, distributed on blocks, where each teacher can enter tests with their respective tests, so that students can then enter code to solve these questions, making the system, once the user's enter is completely uploaded to be able to run, validate and score the entry of each user, reflecting each student's score in a ranking, as well as the time in which they passed the tests in case of solving all the question's tests.

As explained before, CODE-ULL is a system whose main structure is based on the same structure as many of the tools mentioned before, such as CPLAS, APLAS, etcetera. All of them proved to be good tools for teaching in their respective fields, facilitating a large number of tasks related to teaching.

With a view to the future, it is worth mentioning that this system has an exportable and easy-to-use design, because of the system has an interface easy to use as also a register system based on JSONs. Because of this is hoped that it will be possible to implement in future computational thinking workshops, making easier the work of teachers when carrying out their tasks, and helping to improve the experience of those who wish to participate in the workshops or using the project.

Capítulo 7

Presupuesto

El objetivo de este capítulo es el de desplegar una tabla en la que se mostrará un presupuesto acerca del trabajo realizado para desarrollar el proyecto. Valorando valores cómo la investigación, desarrollo, etcétera.... La cual se podrá ver reflejada a continuación en la Tabla 7.1:

Categoría	**Descripción**	**Tiempo estimado**	**Costo Hora**	**Costo Total**
Costo de investigación	Investigación previa al desarrollo	55 h	8 €/h	440 €
Costo de desarrollo	Desarrollo de la aplicación	90 h	12 €/h	1080 €
Costo de depuración	Búsqueda de errores	66 h	12 €/h	792 €
Costo de despliegue	Despliegue del servidor	N/A	N/A	175 €
Costo total				**2487 €**

Tabla 7.1: Tabla de presupuestos

Bibliografía

[1] Aprende con Alf. Librería time. https://aprendeconalf.es/docencia/python/manual/datetime/, 2024.

[2] Arial Dev. Desplegar páginas web. https://blog.arial.dev/como-desplegar-una-aplicacion-web-desde-tu-ordenador, 2024.

[3] DotNet Full Stack Dev. Uso de roles. https://dotnetfullstackdev.medium.com/role-based-authorization-and-authentication-in-react-with-auth-handlers-specif 2024.

[4] EngineeringSoftware. Exli. https://github.com/EngineeringSoftware/exli, 2024. Último acceso: 9 de marzo de 2025.

[5] Python Software Foundation. Librería os. https://docs.python.org/es/3.10/library/os.html, 2024.

[6] Python Software Foundation. Librería re. https://docs.python.org/es/3.13/library/re.html, 2024.

[7] Python Software Foundation. Librería subprocess. https://docs.python.org/es/3.13/library/subprocess.html, 2024.

[8] LenguajeCSS. Guía css. https://lenguajecss.com/css/introduccion/guia-css/, 2024.

[9] LogRocket. Almacenamiento local. https://blog.logrocket.com/using-localstorage-react-hooks/, 2024.

[10] Microsoft. Creación de aplicación web con python a través de vscode. https://learn.microsoft.com/es-es/visualstudio/ide/quickstart-python?view=vs-2022, 2024.

[11] C. Munley, A. Jarmusch, and S. Chandrasekaran. Llm4vv: Developing llm-driven testsuite for compiler validation. *Future Generation Computer Systems*, 2024.

[12] Okayama University. Ejemplo de página con las características de nuestro sistema. https://www.cc.okayama-u.ac.jp/funabiki/PLAS/C_C++/C_C++_programming.html, 2024.

[13] Pallets Projects. Flask framework (para poder crear las páginas). https://flask.palletsprojects.com/en/stable/, 2024.

[14] A. Pnueli, O. Shtrichman, and M. Siegel. The code validation tool (cvt). *International Journal on Software Tools for Technology Transfer (STTT)*, 2(2):192–201, 1998.

[15] Pallets Projects. Leer json con flask. https://flask.palletsprojects.com/en/stable/patterns/javascript/, 2024.

[16] A. A. Puspitasari, N. Funabiki, X. Lu, H. Qi, H. H. S. Kyaw, and K. Ueda. An implementation of code validation function in c programming learning assistant system. *International Journal of Information and Education Technology (IJIET)*, 9(1):24–30, 2023.

[17] React Team. React framework (otra opción para poder crear las páginas). https://es.react.dev/learn, 2024.

[18] Y. W. Syaifudin, P. Y. Saputra, T. Fatmawati, N. I. Susanti, A. R. Patta, and A. Calvina. Application of unit testing and integration testing for automatic grading mechanism in android programming learning assistance system. In *2024 ASU International Conference in Emerging Technologies for Sustainability and Intelligent Systems (ICETSIS)*, pages 1883–1887. IEEE, 2024.

[19] TecAdmin. Desplegar páginas web usando flask en red local. https://tecadmin.net/how-to-configure-flask-application-visible-on-the-network/, 2024.

[20] Byron Vargas. Creación de un servidor. https://www.byronvargas.com/web/como-se-crea-un-servidor/, 2024.

[21] Byron Vargas. Guía de creación de aplicaciones web. https://www.byronvargas.com/web/como-se-crea-una-app-web/, 2024.

[22] YouTube. Flask y react. https://www.youtube.com/watch?v=D1W8H4Rkb9A, 2024.

[23] YouTube. Manejo de fichero json. https://www.youtube.com/watch?v=Y2FpY61h5Z8, 2024.

[24] A. Zmievska. Autograder. https://github.com/zmievsa/autograder, 2024. Último acceso: 9 de marzo de 2025.