

**Universidad
Rey Juan Carlos**

Escuela Técnica Superior
de Ingeniería Informática

Grado en Ingeniería Informática

Curso 2021-2022

Trabajo Fin de Grado

**DESARROLLO DE UNA INTERFAZ WEB PARA
JUEZ DE CÓDIGO**

Autor: Miguel Sierra Alonso

Tutor: José Manuel Colmenar Verdugo

Cotutor: Raúl Martín Santamaría

Agradecimientos

Este TFG va dedicado para mi familia, amigos y compañeros de carrera con los que he compartido momentos tanto buenos como malos en este trayecto. También a los tutores de este proyecto, además de Isaac también colaborador del mismo, por los consejos y la paciencia a partes iguales. Y por último, mención especial a mi tío, primer ingeniero de la familia, el cuál sin su apoyo en las prácticas de circuitos de primero, esto no hubiese sido posible. Gracias.

Resumen

El proyecto consiste en la elaboración de una interfaz web para un juez de código, esta interfaz tiene por objetivo ser más sencilla e intuitiva que otras de otros jueces ya existentes, además este juez será polivalente y no tendrá por qué solo usarse en competiciones, sino también en la educación, ya sea a niveles universitarios o inferiores, ya que esta interfaz es lo suficientemente intuitiva como para que todo tipo de alumnos y profesores puedan usarla con facilidad.

Esta herramienta ubicada dentro de la gamificación, busca una potenciación del aprendizaje y la motivación gracias al uso de esta interfaz, como está comprobado ya con otras herramientas similares.

En el presente documento, se explican las bases del proyecto, como la metodología, los requisitos, tecnologías usadas, el uso de interfaz e incluso un manual de instalación de la herramienta.

La primera sección más importante del documento es [3](#), habla de la toma de ideas, revisando otros jueces ya existentes y buscando puntos clave para incorporar a nuestro juez e identificando aspectos negativos para no añadir a nuestro proyecto.

Para finalizar el mismo, se demuestran los resultados del proyecto explicando y mostrando como quedó finalmente la interfaz ([5](#)), junto a algunos posibles aspectos a mejorar ([6](#)). Además, se incorpora una guía para ayudar a su instalación ([A.1](#)).

Palabras clave:

- Angular
- Juez de código
- Gamificación
- Interfaz web
- API
- Interacción Persona Ordenador
- Responsive

Índice de contenidos

1. Introducción	4
2. Objetivos	6
3. Marco Teórico y Estado del Arte	7
3.1. Juez de programación	7
3.2. Interfaz de usuario	8
3.3. Referencias existentes	9
3.3.1. Codeforces	9
3.3.2. DOMJudge	12
3.3.3. Kattis	16
3.3.4. ¡Acepta el reto!	20
3.3.5. Conclusión	23
4. Descripción Informática	24
4.1. Requisitos	24
4.2. Metodología	27
4.3. Software y tecnologías utilizadas	29
5. Implementación y Resultados	32
5.1. Conexión <i>Frontend - Backend</i>	32
5.2. Internacionalización	33
5.3. Navegación web	34
5.4. Estructura del proyecto	42
5.5. Problemas surgidos durante el desarrollo	42
5.6. Resumen de resultados	44
6. Conclusiones y Trabajos Futuros	46
Bibliografía	48
Apéndices	50
A. Apéndice	50

A.1. Guía de Instalación	50
------------------------------------	----

1

Introducción

Actualmente en el planeta Tierra se vive una realidad muy distinta a la que había hace escasos dos años, cuando no había que llevar mascarillas por la calle, se podía ir a tu lugar de enseñanza con tranquilidad en transporte público, sin ningún peligro de contagio de una grave enfermedad. Todo esto nos ha hecho cambiar la forma en la que enseñar, la manera de trabajar y en definitiva de vida. Por ello, como hemos podido ver en estos años, hemos tenido que dejar las aulas de manera presencial y realizar nuestros estudios de una manera muy distinta a la que estábamos acostumbrados.

Aunque esta podría ser una buena razón para realizar este proyecto, no surge de aquí, surge de la innovación docente a la hora de dar las clases y realizar pruebas a los alumnos, ya que puede hacer más sencilla una parte de la enseñanza, tanto universitaria como a otros niveles. Ya que hay universidades que siguen haciendo ejercicios o exámenes de programación a papel, lo cual queda anticuado y no permite ver rápidamente si es una solución válida o si la eficiencia del código es correcta. Por ello nacen los jueces de código, una herramienta que es capaz de evaluar una solución a un problema de programación junto a otros aspectos como la eficiencia del código, junto a la interfaz que es con la que realiza la comunicación el usuario, por lo que tiene que ser suficientemente intuitiva para un uso correcto de este sistema.

Además, esto no quita, que se pueda usar en una “competición” dentro del aula. Un buen ejemplo es la plataforma *Kahoot!*[1], esta aplicación es muy famosa en el ámbito educativo, ya que es gratuita y bastante atractiva. En definitiva, gusta mucho tanto a profesores como alumnos. Su funcionamiento es sencillo, el

profesor crea concursos con varias preguntas sobre el contenido que desea repasar con el alumnado, con el objetivo de que afiance esa clase que acaba de dar o algunos conceptos en específico, los alumnos con sus propios dispositivos, ya sea móviles u ordenadores, se conectan a ese concurso y compiten entre ellos por ser el mejor de la clase.

La herramienta que se presenta en este TFG, con el nombre *SuperJudge*, pretende ser algo similar, respecto a los objetivos de la misma, que son ayudar al profesor a dar clases más entretenidas^[2] y por tanto que el alumno “aprenda jugando”, un claro ejemplo de gamificación, ya que todas estas modernas herramientas potencian la motivación y el aprendizaje. Por ejemplo, en nuestra herramienta con distinción de roles, el profesor podrá crear un concurso para la clase y que los alumnos manden sus soluciones a los problemas, compitiendo así entre ellos, además de poder obtener *feedback* de su solución en directo, sin necesidad de esperar varios días como sucede en algunas prácticas de programación, esto consigue un mejor aprendizaje para el alumno sobre cualquier lenguaje de programación^[3], pudiendo practicar, explorar el lenguaje y observar diversas soluciones para el mismo problema. Cabe destacar que con este juez no únicamente se tienen esas ventajas para el alumno, un gran beneficiado también es el profesor ya que evita tener que corregir las tareas de todos los alumnos a mano y poder así centrarse más en otros aspectos de la clase.

Cabe destacar, que nuestro juez puede ser usado en otros ámbitos, como puede ser a la hora de seleccionar a personal para un empleo^[4], ya que, se pueden automatizar las entrevistas de conocimiento técnico, para que el candidato realice alguna prueba, la pase por nuestra herramienta y de esta manera que valore su solución, para poder así darle un feedback a la empresa entrevistadora sobre el candidatos.

En definitiva, el principal objetivo de este TFG, es crear una interfaz web para el *backend*, parte del sistema que procesa los datos de entrada desde el *frontend* con su correspondiente lógica, de un juez de código^[5]. Esta interfaz debe ser atractiva, manejable, moderna y simple de usar. De esta manera esta aplicación se podrá usar en distintos niveles de enseñanza, como pueden ser clases de instituto que tengan asignaturas de programación, carreras universitarias e incluso competiciones de programación como las que realiza cada año nuestra universidad. Se necesita esta interfaz ya que los jueces ya existentes en otras plataformas online no están bien adaptadas al entorno educativo, ya que, no dan la flexibilidad que nuestro juez si consigue ofrecer (3). Además, para facilitar la utilización del *backend* realizado por el compañero, se necesita esta interfaz para que sea fácil de usar para cualquier tipo de usuario.

2

Objetivos

El objetivo principal de este TFG, es diseñar e implementar una interfaz para el *backend* de *SuperJudge*. Esta interfaz tendrá que ser sencilla, intuitiva, atractiva, moderna entre otra cualidades.

Además, es importante que no este orientado a un único lenguaje o propósito, sino que sea polivalente e igual de eficaz para todos los lenguajes y posibles usos de este juez, ya sean prácticas, clases, concursos, exámenes, etc.

Cabe destacar que esta interfaz no tendrá un único idioma, ya que si queremos que sea lo más abierta posible, como por ejemplo para estudiantes extranjeros que vienen a nuestra universidad, en esta primera versión se dejará en español e inglés, pero añadir un nuevo idioma no será complejo para mejoras futuras.

Para concluir, esta nueva herramienta no pretende sustituir a ninguna, si lo miramos desde el ámbito de la enseñanza, sino ayudar al sistema actual, para ofrecer una mejor educación al alumnado. En cambio, si lo miramos desde el punto de las competiciones de programación, si que se pretende mejorar los jueces ya existentes, porque ninguno nos parecerá lo suficientemente sencillo para usarlo de una manera intuitiva, tal y como se explicará en el “Estado del Arte”[\(3\)](#).

También es importante añadir los objetivos personales, el principal es aprender y manejar Angular desde 0, además de mejorar en tecnologías tanto en *frontend* (HTML, CSS y JavaScript) como en *backend* (Spring, Spring Boot Java), en las que ya se tenía base y se han conseguido ampliar conocimientos.

3

Marco Teórico y Estado del Arte

En este apartado se va a detallar qué es un juez de programación y qué es una interfaz web, definidos en la introducción, para afianzar los conceptos en los que se basa este TFG. Además, se va realizar un breve análisis de los distintos ejemplos existentes de jueces de los que se han tomado referencias positivas y puntos negativos para tenerlos en cuenta a la hora de diseñar el proyecto.

3.1. Juez de programación

Un juez de programación, como se ha definido anteriormente, es una herramienta usada para evaluar código enviado para solucionar un problema planteado en el mismo. Esta herramienta es telemática, se puede acceder a ella vía internet desde cualquier dispositivo.

En el propio juez existen problemas que el administrador ha generado para que los usuarios los resuelvan, estos problemas tienen una breve explicación del mismo y casos de prueba que son los que el usuario puede probar antes de mandar su solución al juez para una primera prueba.

Una vez el usuario manda su solución, el juez compila la solución, si esta no compila, la herramienta no será capaz de ejecutarla y acabará así su evaluación informando al usuario que envío el código. El otro caso posible es que compile de manera correcta. Tras la compilación, el juez ejecutará los casos de prueba configurados por el administrador para corroborar si la solución es correcta o no.

Además, los jueces también comprueban que el tiempo de la ejecución y la memoria consumida sean menores a un valor configurado. Si la solución aportada por el usuario supera alguno de esos límites, aunque diese salidas correctas, el resultado sería erróneo.

Por último, los jueces dan *feedback* al usuario, es decir, el juez muestra un error específico en caso de que la solución no sea correcta. Esta puede ser una lista real de respuestas de jueces dada una solución por el usuario:

- Error de compilación.
- Error en tiempo de ejecución.
- Límite de tiempo superado.
- Salida no esperada.
- Correcto.

De esta manera, si el usuario ve que su solución no es correcta, podrá volver a enviarla, siempre y cuando la configuración del juez lo permita, ya que puede ser que restrinja el número de envíos por problema o usuario.

3.2. Interfaz de usuario

Una interfaz de usuario, es la parte de un sistema, con la que el usuario de manera sencilla, puede comunicarse con el sistema en cuestión. Sin necesidad de escribir comandos en una terminal, ni de conocimientos muy altos sobre el mismo.

Esta parte del sistema es muy importante, ya que es con lo que va a estar en contacto el usuario, por ello cabe destacar que una interfaz, en este caso web, tenga las siguientes cualidades^[6]:

- Sencillez.
- Coherencia.
- Intuitiva.
- Robusta.
- Usable.
- Atractiva.
- Adaptable.

De esa manera, el usuario tendrá buena experiencia con el sistema y por tanto el usuario volverá a usarlo cuando lo necesite.

3.3. Referencias existentes

Una vez explicados los dos elementos básicos del proyecto, el juez de programación y el concepto de interfaz, podemos explicar cómo se revisaron distintas interfaces de jueces ya existentes, para poder crear nuestra interfaz aprendiendo de las virtudes y errores que encontramos en las distintas plataformas. Para la toma de requisitos, es necesario este análisis de cómo están implementadas otras interfaces ya existentes. Por ello nos vamos a detener sobretodo en las funcionalidades que queremos en nuestra herramienta, como son la pantalla inicial, la clasificación, listado de problemas y envíos, además de la visualización de sus detalles. Pondremos el foco en el aspecto y sencillez de estas ventanas, ya que estamos buscando una interfaz, como hemos comentado anteriormente, sencilla, atractiva e intuitiva como base[7].

3.3.1. Codeforces

El primero de los jueces de los que se toman referencias es Codeforces[8]. Este es un juez en línea de origen ruso y usado para concursos de programación competitiva, el más famoso para estos concursos. Tras revisar su interfaz para poder tomar referencias para nuestro proyecto, vemos que es una interfaz muy antigua, nada atractiva y algo liosa, es decir, todo lo contrario a nuestros principios deseados para nuestra interfaz.

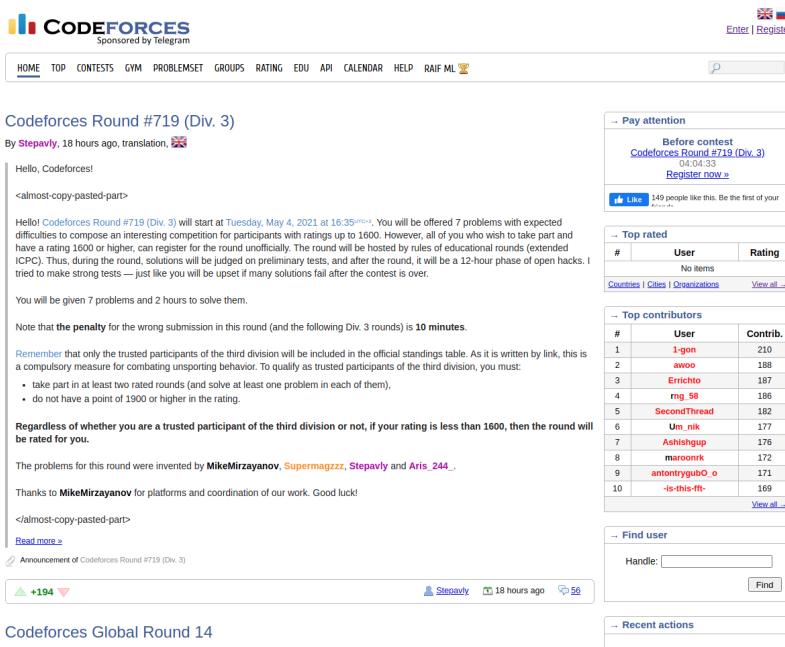


Figura 3.1: Pantalla inicial Codeforces

3.3. Referencias existentes

En esta Figura 3.1 podemos ver lo primero que nos encontramos al acceder, un menú superior donde están las distintas funciones de la web y una pantalla inicial llena de pequeñas cajitas y una especie de tablón de anuncios, todo ello bajo un estilo algo lioso y antiguo, no está todo a primera vista.

The screenshot shows the Codeforces homepage. At the top, there's a navigation bar with links: HOME, TOP, CONTESTS, GYM, PROBLEMSET, GROUPS, RATING, EDU, API, CALENDAR, HELP, RAIF ML, and a search bar. On the right side, there are user icons for misial and Logout, along with a small flag icon. Below the navigation bar, there's a section titled "Current or upcoming contests" with three entries:

Name	Writers	Start	Length	Registrations
Codeforces Round #719 (Div. 3)	Aris 244 MikeMirzayanov Supermegizzz	May/04/2021 16:30 UTC+2	02:00	Before start 03:48:09 Register ↗ 218073 Until closing 05:48:09
Codeforces Round #720 (Div. 2)		May/07/2021 16:30 UTC+2	02:00	Before start 3 days Before registration 20:18:09
Codeforces Raif ML Round 1	raiffeisen	May/17/2021 16:00 UTC+2	6:05:00	Before start 13 days Register ↗ 7764 Until closing 3 weeks

To the right of this is a sidebar with a "Pay attention" section for the round #719, showing "Before contest" details and a "Register now" button. Below that is a "Like" button with 158 likes. Further down, there's a "Contest history" section titled "Past contests" with a table of previous rounds:

Name	Writers	Start	Length	Final standings
Codeforces Global Round 14	FairyPhoenix dragonslayerentaining	May/02/2021 16:35 UTC+2	03:00	Final standings ↗ 22554
Educational Codeforces Round 108 (Rated for Div. 2)	BledDest Neon Fedor aleksey vovuh	Apr/29/2021 16:35 UTC+2	02:00	Final standings ↗ 238923
Contest 2050 and Codeforces Round #718 (Div. 1 + Div. 2)	Futiske Reinhard MihaiRicuOvO xy2906	Apr/03/2021 16:35 UTC+2	02:45	Final standings ↗ 21519
Codeforces Round #717 (Div. 2)	mohammedhab2002	Apr/21/2021 15:35 UTC+2	02:00	Final standings ↗ 21456
Codeforces Round #716 (Div. 2)	mohamedcadry mohammedhab2002	Apr/19/2021 15:35 UTC+2	02:15	Final standings ↗ 214881
Codeforces Round #715 (Div. 1)	Ari Kuroni	Apr/16/2021 16:35 UTC+2	02:15	Final standings ↗ 214523
Codeforces Round #715 (Div. 2)	Ari Kuroni	Apr/16/2021 16:35 UTC+2	02:15	Final standings ↗ 21808
Educational Codeforces Round 107 (Rated for Div. 2)	BledDest Neon Fedor aleksey vovuh	Apr/12/2021 16:35 UTC+2	02:00	Final standings ↗ 21649

At the bottom left, it says "Divide by Zero 2021 and Codeforces Round #714".

Figura 3.2: Listado de concursos Codeforces

En la Figura 3.2 vemos la ventana concursos, es algo confusa, ya que los títulos no están resaltados y dificulta a la hora de elegir un concurso para participar, si únicamente estamos navegando por la web sin tener un concurso preasignado.

El listado de problemas, Figura 3.3, si que se podría trasladar a nuestra interfaz, únicamente mostrando el nombre e *id* del problema junto a unos valores que son la dificultad y número de personas que ha resuelto el mismo, quizás para nuestra interfaz sería útil esta tabla, pero en vez de esos valores de dificultad y usuarios que han resuelto el problema, un link al enunciado en pdf sería algo más útil, ya que esos valores no son del todo claros para el usuario.

La pantalla de visualización de detalles del problema es lo que se puede apreciar en la Figura 3.4. Como se ve, es muy sencilla, no es posible descargar el enunciado, están todos los datos demasiado juntos, sin separación, por lo que no facilita nada al usuario lo que quiere ver en un simple vistazo, además no da facilidad de enviar el problema, ya que está algo escondido, en un lateral, la funcionalidad de subir el código, únicamente permite tomar el fichero y seleccionar el lenguaje, por lo que ni deja ver en el navegador el código que vas a enviar.

Por otra parte, el listado de envíos (Figura 3.5), sí que tiene un diseño ade-

Capítulo 3. Marco Teórico y Estado del Arte

#	Name	standard input/output	solve count
A	Phoenix and Gold	2 s, 256 MB	x13618
B	Phoenix and Puzzle	2 s, 256 MB	x11892
C	Phoenix and Towers	2 s, 256 MB	x7976
D	Phoenix and Socks	2 s, 256 MB	x5857
E	Phoenix and Computers	3 s, 256 MB	x1461
F	Phoenix and Earthquake	3 s, 256 MB	x638
G	Phoenix and Odometers	2 s, 256 MB	x341
H	Phoenix and Bits	4 s, 512 MB	x32
I	Phoenix and Diamonds	5 s, 1024 MB	x27

[Complete problemset](#)

Figura 3.3: Listado de problemas Codeforces

A. Phoenix and Gold

time limit per test: 2 seconds
memory limit per test: 256 megabytes
input: standard input
output: standard output

Phoenix has collected n pieces of gold, and he wants to weigh them together so he can feel rich. The i -th piece of gold has weight w_i . All weights are **distinct**. He will put his n pieces of gold on a weight scale, one piece at a time.

The scale has an unusual defect: if the total weight on it is **exactly** x , it will explode. Can he put all n gold pieces onto the scale in some order, without the scale exploding during the process? If so, help him find some possible order.

Formally, rearrange the array w so that for each i ($1 \leq i \leq n$). $\sum_{j=1}^i w_j \neq x$.

Input
The input consists of multiple test cases. The first line contains an integer t ($1 \leq t \leq 1000$) — the number of test cases.
The first line of each test case contains two integers n and x ($1 \leq n \leq 100$; $1 \leq x \leq 10^4$) — the number of gold pieces that Phoenix has and the weight to avoid, respectively.
The second line of each test case contains n space-separated integers ($1 \leq w_i \leq 100$) — the weights of the gold pieces. It is guaranteed that the weights are **pairwise distinct**.

Output
For each test case, if Phoenix cannot place all n pieces without the scale exploding, print NO. Otherwise, print YES followed by the rearranged array w . If there are multiple solutions, print any.

Example

input	copy
3 3 2 3 2 1 5 3 2 3 4 8 1 5 5	
output	copy
YES 3 2 1 YES 8 1 2 3 4 NO	

Note
In the first test case, Phoenix puts the gold piece with weight 3 on the scale first, then the piece with weight 2, and finally the piece with weight 1. The total weight on the scale is 3, then 5, then 6. The scale does not explode because the total weight on the scale is never 2.

Codeforces Global Round 14

Finished
Practice
Start virtual contest

Virtual participation
Virtual contest is a way to take part in past contests, as close as possible to the real experience. It is supported only ICPC mode for virtual contests. If you've seen these problems, a virtual contest is not for you. If you have not solved them yet, or you just want to solve some problem from a contest, a virtual contest is a good choice. You can archive them in the archive. Never use someone else's code, read the tutorials or communicate with other person during a virtual contest.

Practice
You are registered for practice. You can solve problems unofficially. Results can be found in the contest status and in the bottom of standings.

Clone Contest to Mashup
You can clone this contest to a mashup.
Clone Contest

Submit?

Language: Free Pascal 3.0.2
Choose: Seleccionar archivo | lang1, oracle
Be careful: there is 50 points penalty for submission which has compilation errors or requirements violate failure on the first test, denial of judgement or similar effects. "Passed pretests" submission verdict doesn't mean that your solution is absolutely correct and it will pass system tests.
Submit

Problem tags
constructive algorithms | math | *800

Figura 3.4: Detalles de un problema Codeforces

cuando y se podría adaptar para nuestra interfaz, ya que tiene todo lo necesario, identificador del envío, fecha, usuario, problema, lenguaje, resultado, tiempo y memoria. Lo único que la ventana de detalles del envío, lleva el código enviado, los casos de prueba y el *log*, el cual muestra el detalle de la ejecución, para poder más detalles sobre un posible error. Al hacer click en el identificador nos muestra la ventana de los detalles (Figura 3.6), no da mucha importancia a los que se

3.3. Referencias existentes

The screenshot shows the Codeforces homepage with a navigation bar at the top. Below the navigation bar, there is a table titled "misial submissions" with one row of data. The table columns are labeled: #, When, Who, Problem, Lang, Verdict, Time, and Memory. The single entry in the table is: # 115111610, When May/04/2021 11:49 UTC+2, Who misial, Problem F - Chests and Keys, Lang FPC, Verdict Compilation error, Time 0 ms, Memory 0 KB.

Figura 3.5: Lista de envíos Codeforces

This screenshot shows the "Submission Details" page for a specific submission. It includes the code submitted, the judge log, and the checker log. The code is a Java program that prints "Hello, World" and then enters an infinite loop. The judge log shows a compilation error due to a syntax error in the code. The checker log shows that the program failed to compile.

```
import java.util.Scanner;
public class codigo {
    public static void main(String[] args) {
        // Prints "Hello, World" to the terminal window.
        //System.out.println("HOLA, mundo");
        Scanner sc = new Scanner(System.in);
        while(sc.hasNext()){
            int n = sc.nextInt();
            //System.out.println(n*2);
        }
        throw new RuntimeException("ERROR PROVOCADO");
    }
}
```

-- Judgement Protocol

```
Test: #1, time: 0 ms., memory: 0 KB, exit code: 0, checker exit code: 0, verdict: COMPILATION_ERROR
Input
2 3
3 3
1 1 4
10 10 100
20 15 80
Checker Log
Can't compile file:
Target OS: Win32 (i386)
Compiler: fpc386
program.pas(2,7) Fatal: Syntax error, "BEGIN" expected but "identifier IMPORT" found
Fatal: Compilation aborted
Error: C:\Programs\FPC\bin\i386-Win32\ppc386.exe returned an error exitcode
```

Figura 3.6: Detalles de envío Codeforces

muestran en la tabla, pero es más adecuado que también se viesen en la ventana actual, aparecen todos pegados en la misma línea, por lo que no facilita al usuario visualizarlos de manera sencilla y rápida.

Y por último el ranking, como se puede ver en la Figura 3.7, es muy sencillo, por lo que nos puede servir para nuestra interfaz, ya que ordena a los usuarios por puntuación de los concursos, ordenándolos así en una clasificación.

En definitiva, de este juez podríamos tomar los listados, pero el formato debe ser más atractivo al usuario, ya que durante esta experiencia de uso de este juez, ha sido algo tediosa a la hora de investigar como hacer las distintas funciones básicas además de la sensación de estar usando algo demasiado antiguo por lo que no fomenta mucho la motivación a la hora de usar la herramienta.

3.3.2. DOMJudge

Este juez ya existente es DOMJudge[9], es gratuito y de código abierto[10], es el más famoso para realizar concursos presenciales, por ejemplo, los concursos organizados por ICPC “International Collegiate Programming Contest” con

Capítulo 3. Marco Teórico y Estado del Arte

The screenshot shows the HARBOUR SPACE UNIVERSITY website with a navigation bar including HOME, TOP, CATALOG, CONTESTS, GYM, PROBLEMSET, GROUPS, RATING, EDU, API, CALENDAR, and HELP. A search bar and a 'show unofficial' checkbox are also present. The main content displays the 'Educational Codeforces Round 121 (Rated for Div. 2)' standings. The table lists 16 participants with their names, country flags, and scores. The columns represent problems A through E, with '+' indicating solved problems and 'x' indicating failed attempts. The table includes a header row for the columns and a note at the bottom: 'Double click (or ctrl+click) each entry to view its submission history'.

#	Who	=	Penalty	*	A	B	C	D	E	E
1	noimi	6	189	*	00:00	00:03	00:15	00:36	00:53	01:02
2	eecs	6	192	*	00:02	00:06	00:14	00:58	00:25	00:47
3	olphe	5	93	*	00:01	00:06	00:12	00:24	00:50	
4	HollaQ_Pehv	5	104	*	00:00	00:12	00:17	00:31	00:44	
4	Nyaan	5	104	*	00:01	00:04	00:17	00:34	00:48	
6	tabr	5	108	*	00:00	00:05	00:13	00:22	00:58	
7	Vercingetorix	5	109	*	00:01	00:05	00:17	00:27	00:49	
8	fuppy	5	111	*	00:01	00:08	00:20	00:32	00:50	
9	Kirill22	5	118	*	00:00	00:08	00:17	00:29	00:44	
10	SSRS_	5	121	*	00:00	00:03	00:07	00:30		01:01
11	Jiangly	5	129	*	00:02	00:10	00:23	00:32		00:52
12	flukehn	5	130	*	00:02	00:09	00:19	00:37		00:53
13	A-SOUL_Bella	5	134	*	00:01	00:05	00:17	00:33		01:08
13	andrei_boaca	5	134	*	00:01	00:08	00:18	00:33		01:04
15	prvcislo	5	136	*	00:01	00:10	00:22	00:38		01:05
16	AnandOza	5	139	*	00:01	00:05	00:20	00:30		01:03

Figura 3.7: Ranking Codeforces

origen en Texas, Estados Unidos y sus competiciones regionales como puede ser el SWERC “Southwestern Europe Regional Contest”, donde participan universidades españolas, portuguesas, francesas, italianas, suizas y algunas austriacas, donde el ganador se clasifica para la competición mundial con el resto de campeones regionales. Una vez hecha la introducción, pasamos a analizar lo que nos ataÑe, su interfaz.

The screenshot shows the DOMJudge public interface for the NWERC 2018 competition. The top navigation bar includes links for DOMjudge, Scoreboard, Problemset, Log in, and contest over. The main content displays the 'final standings' for the competition. The table lists 10 teams with their names, country flags, and scores. The columns represent individual team scores and problem results (A through K). The table includes a header row for the columns and a note at the bottom: 'final standings'.

RANK	TEAM	SCORE	A ●	B ●	C ●	D ●	E ●	F ●	G ●	H ●	I ●	J ●	K ●
1	Treenty University of Cambridge	11 1323	170 1 try	103 1 try	56 2 tries	289 1 try	146 1 try	215 2 tries	77 1 try	40 1 try	12 1 try	91 1 try	24
2	Los Patrons University of Oxford	10 1145	217 2 tries	38 1 try	85 2 tries	176 1 try	266 2 tries	157 1 try	12 1 try	18 1 try	72 1 try	22	
3	Double Cycle Cover University of Tübingen	10 1470	267 3 tries	63 1 try	214 3 tries	104 1 try	193 1 try	134 1 try	35 1 try	12 1 try	289 8		
4	Trincherats University of Cambridge	9 788	177 2 tries	59 1 try	31 1 try	180 1 try	101 5 tries	191 1 try	9 1 try	10 1 try	93 1 try		
5	TUMboing Technische Universität München	9 835	188 1 try	49 2 tries	74 1 try	161 1 try	126 5 tries	23 1 try	12 1 try	147 2 tries	16		
6	2 Brits and a Dutchman University of Oxford	9 1021	297 2 tries	147 2 tries	83 1 try	229 1 try	112 1 try	18 1 try	10 1 try	60 1 try	25		
7	Q++ Leiden University	9 1185	267 1 try	123 1 try	77 1 try	205 1 try	135 2 tries	39 2 tries	13 1 try	236 2 tries	30		
8	<DvD> Stairland University	9 1226	293 4 tries	50 1 try	166 1 try	101 3 tries	152 1 try	25 2 tries	8 1 try	209 5 tries	22		
9	Oxford JI-geiko Technische Universität München	9 1498	295 2 tries	74 1 try	135 1 try	281 1 try	210 3 tries	26 1 try	16 2 tries	180 9 tries	41		
10	_=> + <<_1. University of Oxford	8 915	138 1 try	60 1 try	263 1 try	154 3 tries	26 2 tries	23 1 try	117 3 tries	34 1 try			

Figura 3.8: Pantalla pública DOMJudge

Accediendo a su web, podemos encontrar una demo donde visualizar los distintos roles y poder así tomar ideas. Lo primero que podemos apreciar en esta

3.3. Referencias existentes

Figura 3.8, es la ventana pública, donde únicamente muestra el ranking, un acceso a los problemas que veremos más adelante y un botón de login. El ranking, es bastante más interesante que el anterior, ya que da más detalles de la competición con un simple vistazo, además de con distintos colores señalar quien pasó el problema el primero, quien simplemente pasó el problema y quien no. También, tiene un filtro para ayudar a buscar en la clasificación e incluso la posibilidad de marcar como favoritos algunos equipos, para que de esta manera puedas visualizar los equipos seleccionados separados del resto. Este componente podría ser tomado como referencia para nuestro juez, ya que además de ser muy útil, es sencillo y motiva al usuario a realizar problemas para poder llegar lo más arriba posible.

The screenshot shows the DOMjudge public interface with the following layout:

- Header:** DOMjudge, Home, Problemset, Print, Scoreboard, Submit, Logout.
- Title:** Contest problems
- Problems List:** Six problems are listed in two rows of three:
 - Problem A:** Access Points, Limits: 2 seconds / 2 GB. Status: Red dot (passed first). Includes a "problem text" button and a grid of sample results (green, yellow, red). Samples section shows input #1, output #1; input #2, output #2; and a "zip with all samples" button.
 - Problem B:** Brexit Negotiations, Limits: 6 seconds / 2 GB. Status: Blue dot (passed). Includes a "problem text" button and a grid of sample results (green, yellow, red). Samples section shows input #1, output #1; input #2, output #2; and a "zip with all samples" button.
 - Problem C:** Circuit Board Design, Limits: 2 seconds / 2 GB. Status: Green dot (passed). Includes a "problem text" button and a grid of sample results (green, yellow, red). Samples section shows input #1, output #1; input #2, output #2; and a "zip with all samples" button.
 - Problem D:** Date Pickup, Limits: 8 seconds / 2 GB. Status: Red dot (passed first). Includes a "problem text" button and a grid of sample results (green, yellow, red). Samples section shows input #1, output #1; input #2, output #2; and a "zip with all samples" button.
 - Problem E:** Equality Control, Limits: 4 seconds / 2 GB. Status: Green dot (passed). Includes a "problem text" button and a grid of sample results (green, yellow, red). Samples section shows input #1, output #1; input #2, output #2; and a "zip with all samples" button.
 - Problem F:** Fastest Speedrun, Limits: 8 seconds / 2 GB. Status: Blue dot (passed). Includes a "problem text" button and a grid of sample results (green, yellow, red). Samples section shows input #1, output #1; input #2, output #2; and a "zip with all samples" button.

Figura 3.9: Problemas en DOMJudge

Si vemos la pantalla de los problemas (Figura 3.9), da igual la pública que la del equipo una vez iniciada la sesión, ya que en la demo al menos es la misma ventana, por lo que tomaremos la del equipo con su sesión iniciada. En esta ventana se muestran todos los problemas del concurso actual, el problema que se ve aquí, es que no se pueden visualizar los detalles del problema de un simple vistazo, sino que si quieres ver los casos de prueba, debes descargarlos primero y luego abrirlos, al igual que con el enunciado en formato PDF. Además, sería mejor una lista, con las *id* y nombres del problema y poder ver los detalles al hacer click en la fila, no usaría este estilo para la nueva interfaz.



The screenshot shows a web-based submission form titled "Submit". It has fields for "Source files" (with a "Browse" button), "Problem" (a dropdown menu with "Select a problem"), and "Language" (a dropdown menu with "Select a language"). At the bottom are "Cancel" and "Submit" buttons, with the "Submit" button being green.

Figura 3.10: Realizar envío en DOMJudge

La ventana para realizar un envío es mejor que la de Codeforces, ya que es un botón en la parte superior en color verde, con lo que se le facilita al usuario enviar una solución. Al hacer click aparece esta ventana (Figura 3.10), donde se selecciona el archivo fuente, problema al que se quiere realizar el envío y el lenguaje de programación deseado. El primer detalle negativo que se aprecia, es que el usuario no puede ver en el navegador el fichero que está subiendo, por si quiere hacer alguna última revisión o cambio, sería bueno que el sistema permitiese al usuario elegir de que manera quiere enviar la solución, seleccionando el fichero o editándolo en línea. Además, veo mejor poder realizar el envío desde cada problema y evitar así que el usuario tenga que elegirlo en un desplegable antes de enviar, para evitar envíos no deseados a otro problema.

Submissions			
time	problem	lang	result
14:59	F	CPP	WRONG-ANSWER
14:38	F	CPP	WRONG-ANSWER
14:25	F	CPP	WRONG-ANSWER
14:11	F	CPP	WRONG-ANSWER
13:43	F	CPP	WRONG-ANSWER
13:08	A	CPP	CORRECT
12:41	E	CPP	CORRECT
12:27	J	CPP	CORRECT
12:11	J	CPP	WRONG-ANSWER
12:05	G	CPP	CORRECT
11:14	C	CPP	CORRECT
11:06	C	CPP	WRONG-ANSWER
10:49	B	CPP	CORRECT
10:23	H	CPP	CORRECT
10:16	K	CPP	CORRECT
10:12	I	CPP	CORRECT

Figura 3.11: Lista de envíos en DOMJudge

La lista de envíos tiene un diseño adecuado (Figura 3.11), ya que aparecen ordenados por hora, y además muestra el nombre del problema, el lenguaje y el resultado obtenido en la compilación del código enviado, sería una lista muy similar la que se creará en la interfaz de nuestro juez, aunque hay que añadir que

en DOMJudge este listado no es visible para los usuarios, únicamente para los administradores.

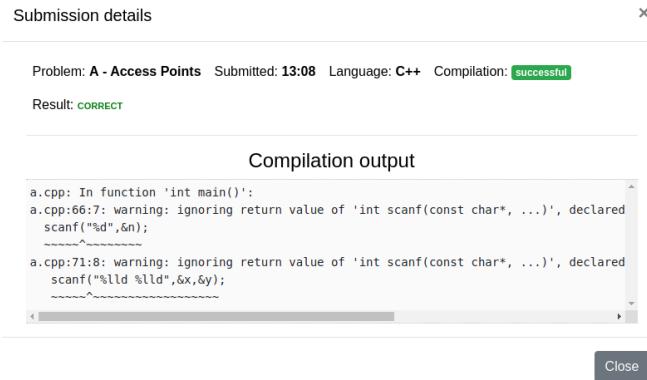


Figura 3.12: Detalles envío en DOMJudge

En la Figura 3.12 se muestran los detalles del envío, pasa algo similar a la de Codeforces, aparecen todos los detalles del envío demasiado pegados, con lo cual dificulta al usuario ver la información que desea de un simple vistazo, además no muestra el código fuente enviado, algo que puede ser útil al usuario para saber porque falló su solución. Es necesario crear una nueva pantalla para nuestro juez para poder visualizar bien los detalles del envío, qué casos de prueba fueron correctos y cuáles no, además de poder visualizar el código enviado.

Sobre este juez, podríamos tomar el ranking y el estilo, ya que es algo más atractivo a la vista que el otro juez ya analizado, aunque tenga algún punto negativo, como la forma de mostrar los problemas que no replicaremos en nuestro juez, es una gran referencia a tener en cuenta para nuestro proyecto ya que es un juez más actualizado que Codeforces, por lo que la interfaz es algo más sencilla y por tanto, más cercana a lo que buscamos.

3.3.3. Kattis

En esta sección analizamos Kattis[11], este juez de programación es gratuito, las empresas usan este juez para reclutar trabajadores, dándole desafíos al candidato y viendo así sus cualidades técnicas en plena acción, además también es usado para campeonatos mundiales de programación (ICPC) como DOMJudge.

Cuando accedemos a su web, como se muestra en la Figura 3.13, se aprecia un menú superior con las distintas opciones a visualizar, y un cuerpo de la pantalla con distintas clasificaciones, problemas sugeridos e incluso anuncios de ofertas de trabajo. A primera vista este juez tiene la interfaz hasta ahora más atractiva y visual que el resto de jueces analizados.

Capítulo 3. Marco Teórico y Estado del Arte

Welcome to the Kattis Problem Archive
Here you can find hundreds of programming problems to solve. If you're new here you're very much welcome! Just [register](#) and start solving.

COMPANY	JOB AD
UNGRDINAL	Help us build the next generation multiplayer server technology

Suggested problems

DIFFICULTY [?]	PROBLEM	SCORE [?]
TRIVIAL	Hello World!	1.2 pt
	Pot	1.2 pt
	Soda Slurper	1.5 pt
	Ptice	1.5 pt
EASY	I Can Guess the Data Structure!	2.8 pt
MEDIUM	WFF 'N PROOF	2.8 pt
	Moogle	5.5 pt
HARD	Escape Plan	5.5 pt

Ranklist

#	USER	SCORE [?]
1	Ivan Petrov	9748.7
2	Josenildo Silva	8226.2
3	Dmitry Lyubshin	8135.7
4	Nick Wu	7808.6
5	Doug Goodman	7720.7
6	Bjarki Ágúst Guðmundsson	7466.1
7	Johan Wind	7257.0
8	Oskar Haarklou Velleborg	6980.4

Universities

#	UNIVERSITY	SCORE [?]
1	National University of Singapore	4187.1
2	KTH Royal Institute of Technology	4027.6
3	Lund University	3595.9
4	Reykjavik University	3109.9
5	Aarhus University	2922.4
6	Mount Allison University	2736.1
7	University of São Paulo	2794.7

Countries

#	COUNTRY	SCORE [?]
1	Sweden	4889.5
2	United States	4858.6
3	Canada	4503.0
4	Iceland	3836.0
5	Singapore	3612.7
6	Denmark	3201.0
7	Norway	2690.6

Figura 3.13: Pantalla pública Kattis

Problems

SOLVED TRIED UNTRIED RSS feed for new problems

NAME	SUBMISSIONS			USERS			DIFFICULTY	
	TOTAL	ACC.	RATIO	FASTEST	TOTAL	ACC.		RATIO
0-1 Sequences	10915	1714	16%	0.00	2220	1183	53%	5.5
10 Kinds of People	25192	5140	20%	0.01	4377	3164	72%	4.8
2048	10306	4725	46%	0.00	4552	3899	86%	2.4
2, 4, 6, Greaat	740	107	14%	0.15	84	37	44%	9.4
3D Printed Statues	13583	6251	46%	0.00	5835	5237	90%	1.9
3D Printer	950	157	17%	0.00	392	120	31%	8.8
4 thought	11785	4041	34%	0.00	3882	3280	84%	2.7
A1 Paper	6451	1948	30%	0.00	1924	1536	80%	3.7
Aah!	34229	15964	47%	0.00	14336	13383	93%	1.5
Abandoned Animal	2449	551	22%	0.03	630	462	73%	5.4
ABC	18602	8458	45%	0.00	8005	7269	91%	1.8
Ab Initio	1737	291	17%	0.09	255	158	62%	7.6
Above Average	12254	5093	42%	0.00	4900	4404	90%	1.8
A+ß Problem	9355	921	10%	0.03	1341	444	33%	8.3
A Brief Gerrymander	519	104	20%	0.11	96	46	48%	8.9
Abstract Art	446	147	33%	0.00	118	89	75%	6.0
Abstract Painting	985	408	41%	0.00	367	301	82%	3.8
Absurdistan Roads II	691	235	34%	0.00	232	155	67%	6.9
Absurdistan Roads III	1724	493	29%	0.02	468	356	76%	5.6
Access Points	477	166	35%	0.01	157	122	78%	5.7
Accounting	1484	436	29%	0.02	402	281	70%	4.7
A Classy Problem	12265	3214	26%	0.01	2779	2161	78%	4.3
ACM Contest Scoring	8186	5130	63%	0.00	4576	4325	95%	1.5

Figura 3.14: Lista de problemas en Kattis

En la Figura 3.14, se aprecia la lista de problemas donde en un formato muy simple y claro aparece toda la información relevante para poder acceder al problema deseado, en la tabla aparecen información de cuantos envíos ha recibido cada problema y cuantos han sido aceptadas, al igual con los usuarios que han intentado el problema, además de una valoración de la dificultad. Si se quiere acceder a los detalles del mismo, bastará con hacer click en el nombre del problema deseado.

Una vez seleccionado el problema deseado, aparecerá la pantalla de la Figura 3.15, donde hasta ahora es la pantalla más clara de todos los jueces analizados

3.3. Referencias existentes

The screenshot shows the Kattis platform interface for the '3-Sided Dice' problem. At the top, there's a navigation bar with links for PROBLEMS, CONTESTS, RANKLISTS, JOBS, and HELP. A search bar and a 'Submit' button are also present. The main content area is titled '3-Sided Dice'. It contains a detailed problem description, input and output specifications, and sample data. On the right side, there's a sidebar with user statistics and problem metadata.

Sample Input:

```
0 0 10000
0 10000 0
10000 0 0
3000 4000 3000
```

Sample Output 1:

```
YES
NO
```

Figura 3.15: Detalle de problema en Kattis

para visualizar la información de la misma, ya que aparece separada en claros apartados, para poder fijarse en el apartado que el usuario desee sin necesidad de ir buscando por la pantalla. Además tiene un claro botón para hacer el envío en la parte superior derecha de color verde, para que el usuario tampoco tenga ningún problema a la hora de intentar realizar un envío de su solución.

The screenshot shows the submission form for the '3-Sided Dice' problem. It features a large text area for uploading a solution file via drag & drop or file selection. Below this is a 'Switch to editor' link. At the bottom, there's a language selection dropdown set to 'C++' and a 'Submit' button.

Figura 3.16: Realizar envío en Kattis

En las figuras anteriores (3.16 y 3.17), podemos apreciar la ventana para realizar un envío, vemos que la herramienta nos da dos opciones, pero no es posible usar ambas a la vez, una de ellas es importar desde el fichero la solución, pero no nos mostrará el texto en el editor, si queremos usar el editor, tendremos que escribir nuestra solución o pegarla desde el fichero. Aquí lo ideal, es combinar ambas opciones, ya que es importante que el usuario si quiere realizar una última revisión o modificación de su solución antes de enviarlo, lo pueda hacer desde la herramienta donde realiza el envío, para aportarle seguridad de que esta mandando lo que realmente quiere.

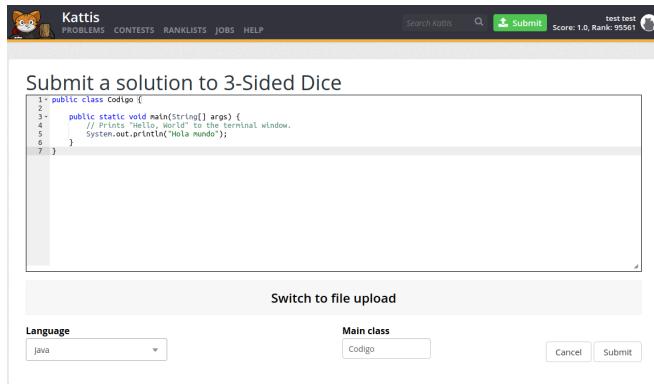


Figura 3.17: Realizar envío en Kattis (editor texto)

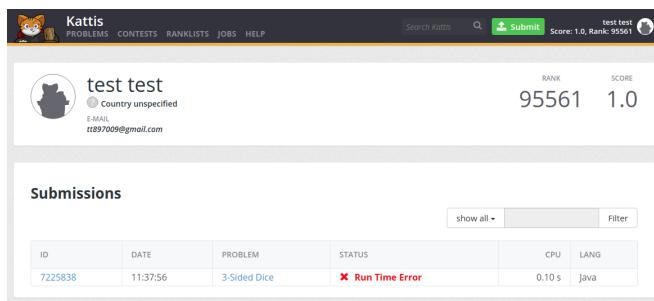


Figura 3.18: Lista de envíos en Kattis

En la Figura 3.18 se muestra el listado de envíos del usuario, este listado es bastante similar a lo que queremos para nuestra interfaz, es una tabla, donde muestra un identificador de envío, la fecha, el problema al que se ha hecho el envío, el resultado de la corrección, el lenguaje y además el tiempo consumido por la máquina para ejecutar la solución enviada. Si hacemos click en el identificador, podremos ver los detalles del envío, tal y como se aprecia en la Figura 3.15.

En la Figura 3.19 se muestran los detalles del envío, al igual que la lista, esta pantalla también resulta muy clara y sencilla para visualizar lo detalles del envío, en la parte superior salen los mismos datos que en el listado, además de mostrar que casos de prueba fueron correctos o erróneos. En la parte inferior, aparecen detalles sobre el fichero enviado, además de poder visualizar el contenido del mismo sin necesidad de descargarlo. Esta ventana salvo alguna posible mejora, como un gráfico o similar para apreciar qué casos de prueba se ejecutaron correctamente, tiene un diseño bastante adecuado para lo que buscamos para nuestra interfaz, por lo que podemos tomar buenas referencias de la misma.

Como conclusión, esta herramienta puede ser utilizada como ejemplo para nuestra interfaz en algunas facetas, como podría ser la modernidad y sencillez a la hora de ir navegando por la web, aunque hay aspectos a mejorar como puede ser a la hora de realizar el envío combinando ambas posibilidades, este análisis

3.3. Referencias existentes

The screenshot shows a Kattis submission page. At the top, there's a navigation bar with links for PROBLEMS, CONTESTS, RANKLISTS, JOBS, and HELP. A user icon and a search bar are also present. The main area is titled "Submission". It displays a table with columns for ID, DATE, PROBLEM, STATUS, CPU, and LANG. The entry is ID 7225838, DATE 11/37/56, PROBLEM "3-Sided Dice", STATUS "Run Time Error", CPU 0.10 s, and LANG Java. Below this is another table for "TEST CASES" with one row for "7225838" showing a failed test case. A link to "download zip archive" is available. The code submitted is "codigo.java":

```

1 import java.util.Scanner;
2
3 public class codigo {
4
5     public static void main(String[] args) {
6         // Prints 'Hello, World' to the terminal window.
7         //System.out.println("Hello, mundo");
8
9         Scanner sc = new Scanner(System.in);
10
11         while(sc.hasNextInt()){
12             int n = sc.nextInt();
13             //System.out.println(n);
14         }
15     }
16     throws new RuntimeException("ERROR PROVOCADO");
17 }
18
19

```

Figura 3.19: Detalles de un envío en Kattis

realizado nos puede resultar muy útil para la toma de requisitos.

3.3.4. ¡Acepta el reto!

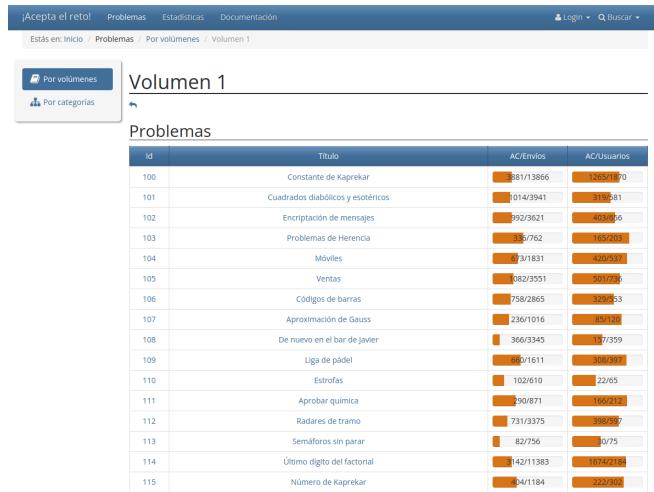
Por último, vamos a comentar el juez ¡Acepta el reto![12], es un repositorio de problemas de programación en español, que además evalúan la solución al problema con un juez en línea. La particularidad de este juez, es que no está tan orientado a concursos como los anteriores jueces, sino a solucionar los problemas cargados en la web.

The screenshot shows the homepage of ¡Acepta el reto!. At the top, there's a navigation bar with links for Problemas, Estadísticas, Documentación, and a login/search bar. The main content area has two sections: "¿Qué es?" and "¿Por dónde empiezo?". The "¿Qué es?" section explains what the site is about, mentioning it's a repository of Spanish programming problems. The "¿Por dónde empiezo?" section provides instructions for new users. Below this is a large button labeled "¡Aceptas el reto?". Further down, there's a section titled "Problema de la semana" with a sub-section "La estación central". This section contains a brief problem statement about finding the central station in a subway network based on travel times.

Figura 3.20: Pantalla de inicio de ¡Acepta el reto!

La pantalla inicial de esta web 3.20, contiene una breve explicación sobre qué

es esta herramienta junto a los primeros pasos a realizar en la misma. Además contiene un problema, el cual va cambiando cada semana para que sea resuelto por el usuario. Cabe destacar el discreto menú de la parte superior, el cual contiene todas la funcionalidades de la herramienta, ver los problemas, las estadísticas, documentación del juez, además de por supuesto el inicio de sesión. En comparación con los anteriores, es muy simple, aunque poco atractiva, lo que habría que mejorar con la implementación de nuestra interfaz.



The screenshot shows a web interface for a programming challenge platform. At the top, there's a navigation bar with links for 'Problemas', 'Estadísticas', and 'Documentación'. Below the navigation is a breadcrumb trail: 'Estás en: Inicio > Problemas > Por volúmenes > Volumen 1'. A sidebar on the left has buttons for 'Por volúmenes' and 'Por categorías'. The main content area is titled 'Volumen 1' and contains a section titled 'Problemas'. It lists 115 problems with columns for 'Id', 'Título', 'AC/Envíos', and 'AC/Usuarios'. Each row also features a small orange progress bar indicating the percentage of solutions accepted relative to the total submissions.

Id	Título	AC/Envíos	AC/Usuarios
100	Constante de Kaprekar	381/13866	1259/1870
101	Cuadrados diabólicos y esotéricos	0/14/3941	0/0/81
102	Encriptación de mensajes	392/3021	403/956
103	Problemas de Herencia	359/762	165/203
104	Móviles	673/1831	420/937
105	Ventas	282/3551	304/173
106	Códigos de barras	758/2865	370/953
107	Aproximación de Gauss	236/7016	85/120
108	De nuevo en el bar de Javier	366/3345	157/359
109	Liga de pádel	583/611	308/397
110	Estrofas	102/610	22/65
111	Aprobar química	390/871	106/12
112	Radares de tramo	731/3375	86/657
113	Semáforos sin parar	82/756	10/75
114	Último dígito del factorial	142/1183	1674/2184
115	Número de Kaprekar	404/1184	223/302

Figura 3.21: Listado de problemas en ¡Acepta el reto!

En la Figura 3.21, se puede apreciar el listado de problemas, bastante simple y acorde a lo que se podría realizar para nuestro juez, cabe destacar que se pueden visualizar los problemas por numero de identificador o por categorías, además se puede observar en las dos columnas de la derecha, unas estadísticas de soluciones aceptadas entre el total de envíos y el total de usuarios. Estas estadísticas, aplicadas al contexto de nuestro juez, el cual será más para el ámbito universitario o concursos de programación, son un algo redundantes ya que estará más acotado el número de usuarios, de problemas y de tiempo para realizarlos, no como en esta herramienta.

Cuando hacemos click en alguno de los problemas del listado de la Figura 3.21, accedemos a la pantalla de la Figura 3.22, donde se muestran todos los detalles y características del problema, como son el enunciado, ejemplos, casos de prueba y métricas a cumplir. Desde esta ventana, además se realizan los envíos de las soluciones y se puede acceder a estadísticas de envíos para este problema en específico. Se puede tomar como buen ejemplo para nuestra interfaz, ya que cumple con lo que buscamos, aunque se buscaría un diseño más atractivo.

En la Figura 3.23 se muestran los envíos en tiempo real de la herramienta, para todos los problemas y de todos los usuarios, ya que como hemos dicho al comienzo este juez no distingue por concursos, por lo que es accesible a todos. Este listado, es casi perfecto para nuestra interfaz, la mejora que se le podría

3.3. Referencias existentes

The screenshot shows a detailed view of a problem titled "Constante de Kaprekar". The interface includes a sidebar with "Enunciado", "Enviar", "Estadísticas", and "Créditos" buttons. The main content area has sections for "Tiempo máximo: 2,000 s" and "Memoria máxima: 4096 KB". It provides instructions: 1. Elige un número de cuatro dígitos que tenga al menos dos diferentes (es válido colocar el dígito 0 al principio, por lo que el número 0009 es válido). 2. Coloca sus dígitos en orden ascendente y en orden descendente para formar dos nuevos números. Puedes añadir los dígitos 0 que necesites al principio. 3. Resta el menor del mayor. 4. Vuelve al paso 2. A note says: "A este proceso se le conoce como la rutina de Kaprekar, y siempre llegará al número 6174 en, como mucho, 7 iteraciones. Una vez en él, el proceso no avanzará, dado que $7641 - 1407 = 6174$ ". Examples show the process for number 3524: 3425 - 2534 = 9981, 9981 - 1899 = 8082, 8820 - 0288 = 8532, 8532 - 2358 = **6174**. Other examples include 1121 needing 5 iterations and 1893 needing 7 iterations.

Figura 3.22: Detalles de un problema en ¡Acepta el reto!

Envío	Usuario	Problema	Resultado	Lenguaje	Tiempo	Memoria	Pos.	Fecha
590856	Santigogin	Me pifio el toro (270)	WA	C++	0.284	1952	-	Hace 36 segundos
590855	JuanCarlos	Chicles de regalo (121)	TLE	Java	-	-	-	Hace 1 minuto
590854	Santigogin	Me pifio el toro (270)	WA	C++	0.26	1952	-	Hace 1 minuto
590853	Santigogin	La abuela María (337)	WA	C++	0.072	1820	-	Hace 1 minuto
590852	pkamacho	La mejor terminación (387)	RTE	Java	-	-	-	Hace 5 minutos
590851	FrancoCollins	Escalera de color (134)	RTE	Java	-	-	-	Hace 7 minutos
590850	FrancoCollins	Escalera de color (134)	RTE	Java	-	-	-	Hace 7 minutos
590849	mistig168	La ardilla viajera (326)	TLE	C++	-	-	-	Hace 10 minutos
590848	Santigogin	La abuela María (337)	WA	C++	0.024	1684	-	Hace 11 minutos
590847	mistig168	La ardilla viajera (326)	RTE	C++	-	-	-	Hace 13 minutos
590846	Aprezeza	Persistencia multiplicativa ...	AC	Java	0.068	1138	14	Hace 14 minutos
590845	Santigogin	La abuela María (337)	WA	C++	0.024	1684	-	Hace 14 minutos
590844	Aprezeza	Persistencia multiplicativa ...	AC	Java	0.061	1138	8	Hace 15 minutos
590843	Aprezeza	Persistencia multiplicativa ...	AC	Java	0.058	1138	5	Hace 16 minutos
590842	Aprezeza	Anillos (183)	AC	Java	0.071	5142	34	Hace 21 minutos
590841	Santigogin	¿En qué volumen? (595)	AC	C++	0.028	1680	142	Hace 21 minutos
590840	Aprezeza	Anillos (183)	RTE	Java	-	-	-	Hace 22 minutos
590839	Santigogin	¿En qué volumen? (595)	WA	C++	0.032	1680	-	Hace 22 minutos

Figura 3.23: Listado de envíos en ¡Acepta el reto!

añadir es un filtro por columna, para poder buscar de una manera más sencilla entre todas las entradas del listado de envíos, ya que al irse actualizando, en algún momento se necesitará visualizar únicamente los de un problema o los de un usuario en concreto.

En definitiva, este juez ha sido sin duda el más sencillo de los analizados, aunque sí que tiene algún defecto a corregir como hemos expuesto en este apartado, aún así se pueden tomar buenas referencias como base para construir nuestra interfaz, solucionando esas desventajas mencionadas además de mejorando los aspectos positivos de este juez.

3.3.5. Conclusión

Tras analizar diversos jueces ya existentes, además de los mencionados en esta memoria (Codeforces, DOMJudge, Kattis y ¡Acepta el reto!), hemos revisado otros como Jutge[13], LightOJ o SPOJ, que se han omitido de la memoria por no aportar ninguna característica relevante para este trabajo. Todos los jueces tienen como punto negativo, aunque no de la misma manera no poder usarlos de una manera sencilla e intuitiva desde un primer momento, cosa que queremos corregir en nuestra interfaz.

Además se busca que nuestra interfaz sea moderna, cosa que con estos jueces analizados el único podría cumplirlo es Kattis, lo queremos para que el usuario tenga la motivación de usarla e incluso de llegar a tomársela como un juego y las interfaces existentes no despiertan eso en el usuario. También, hay otros puntos que facilitarían mucho el uso de estos jueces al usuario. Como un editor de texto en línea, que sea capaz de mostrar el código de con su correspondiente lenguaje, y que así pueda realizar su ejercicio, editar su código o simplemente revisar su solución justo antes de enviarlo. Todo esto desde la propia herramienta, dando una seguridad extra a la persona tras la pantalla.

De esta manera creemos que esta justificado el hecho de crear una nueva interfaz. Ya que, combinando todo lo positivo encontrado, junto a la evolución y eliminación de los aspectos negativos. Se podría obtener una mejor experiencia de usuario y por tanto un alumno más motivado, incrementando así su rendimiento y su aprendizaje sobre la programación.

4

Descripción Informática

En este apartado se va a detallar la toma de requisitos tras el análisis de diversos jueces de programación, la metodología que se ha seguido durante el proyecto, que herramientas se han utilizado y los puntos importantes sobre el proyecto.

4.1. Requisitos

En este apartado se van a detallar los requisitos obtenidos de la investigación y diseño previos para empezar a desarrollar la interfaz a partir de los mismos. Habrá dos tipos de usuarios, unos para la parte de profesor/administrador, genera problemas y concursos, mientras que los otros serían para la parte del alumno, el cual resuelve los problemas creados por el rol anteriormente mencionado. Además, habrá unos requisitos que son para la parte pública de la interfaz, para lo accesible sin realizar un inicio de sesión. Y por último, unos requisitos más generales que engloban a todo el conjunto de la interfaz. En el diagrama de la Figura 4.1, se puede apreciar lo explicado con un diagrama de Venn.



Figura 4.1: Diagrama de Venn de las funciones por cada rol

Requisitos generales:

1. Internacionalización de los textos, para poder así tener la interfaz con cualquier idioma configurado previamente.
2. Interfaz *Responsive*.
3. Respetar los principios de usabilidad. Como por ejemplo:
 - a) Visibilidad del estado del sistema.
 - b) Consistencia y estándares.
 - c) Prevención de errores.
 - d) Diseño estético y minimalista.

Requisitos para la parte pública:

1. Pantalla de inicio sencilla, para facilitar al usuario la utilización de algo nuevo.
2. Visualización de problemas públicos.
3. Visualización de envíos en los problemas públicos junto a sus estadísticas.
4. Visualización de una clasificación de equipos/usuarios públicos.
5. Acceso para iniciar sesión en la aplicación, a la vista para no frustrar al usuario.

Requisitos para el rol profesor/administrador:

1. Creación de concursos, para poder tener recogidos los problemas y poder decidir que usuarios puedan tener acceso a los mismos.
2. Editar, eliminar concursos, para cambiar las características del mismo o borrarlo del sistema.
3. Creación de problemas a partir de un fichero, para que los alumnos puedan solucionarlo.
4. Editar, eliminar problemas, para cambiar las características del mismo o borrarlo del sistema.
5. Asignar problemas a concursos, para que ese problema sea accesible para los usuarios de otro concurso.
6. Visualización de detalles de los problemas, para poder revisar que todo quedó correcto.
7. Visualización de envíos, para ver de un vistazo todos los intentos de los usuarios.
8. Visualización de detalles de los envíos, para ver las soluciones propuestas y los resultados de los alumnos.
9. Realizar envíos a un problema deseado, poder probar una solución sin necesidad de ser alumno.

Requisitos para el rol alumno:

1. Visualización de problemas, para poder ver todos los problemas que el alumno puede realizar.
2. Realizar envíos de soluciones a los problemas, para poder probar el código elaborado por el alumno.
3. Editor de código en la propia herramienta, para poder visualizar ó editar la solución justo antes de proceder al envío.
4. Visualización de envíos, para poder visualizar el resultado de las soluciones propuestas.

4.2. Metodología

En este punto se va a dar detalle sobre la metodología usada durante el desarrollo de este proyecto, además de enfocarnos en las tareas realizadas en cada *sprint*, para valorar el avance que se ha ido generando de una manera mas ordenada.

En este proyecto se han tomado principios de la metodología Agile[14], ya que los requisitos no eran fijos desde un comienzo, sino que han ido variando durante el transcurso de la elaboración del mismo. Además, se han hecho pequeños *sprints*, período breve de tiempo en el que un equipo con metodología Scrum trabaja para completar unos objetivos establecidos, con el fin de poder ir midiendo el progreso del TFG. Cabe destacar que Scrum, se podría definir como, una metodología ágil con el fin de gestionar proyectos basados en desarrollo incremental.

Esta metodología Agile que hemos usado en el desarrollo de este TFG existe desde el año 2001. Nace en Utah, Estados Unidos, de una reunión de un grupo de las principales empresas tecnológicas del momento, donde pusieron en común todas las buenas prácticas de cada una de ellas en la manera de trabajar. Fruto de esta reunión surgió el manifiesto Agile[15], en el cual se detallaban procesos de planificación, creación y testeo del desarrollo software. Si volvemos a nuestro proyecto, se ha usado la metodología ágil Scrum[14], con el fin de facilitar el desarrollo del mismo mediante la creación de diferentes sprints con un número de tareas determinado, además de que estas tareas fuesen cortas y concretas. Además, siendo fiel a la metodología, se han realizado varias reuniones con los tutores, en las cuales se veía el progreso actual del proyecto, que faltaría por añadir, errores a corregir en lo generado hasta el momento e incluso consejos a la hora de desarrollar algunas funcionalidades, también cabe destacar la comunicación fluida vía email, para distintos problemas que surgían o que alteraban la planificación. Otro elemento importante de gestión, ha sido el tablero Kanban, esta herramienta es usada para visualizar el flujo de trabajo, hay una lista de tareas sin hacer, una lista de tareas en progreso y una lista de tareas finalizadas, un ejemplo de este tablero podría ser como se ve en la siguiente figura:

En el desarrollo del proyecto han sido necesarios cinco sprints, vamos a detallar cada uno de ellos con las tareas y problemas que se han ido resolviendo en cada uno de ellos.

Sprint 0:

- Se investigan otros jueces de programación
- Se toman notas para empezar a dar primeros pasos en el desarrollo del trabajo.
- Revisión con tutores para ver como empezar a desarrollar con esos requisitos tomados.

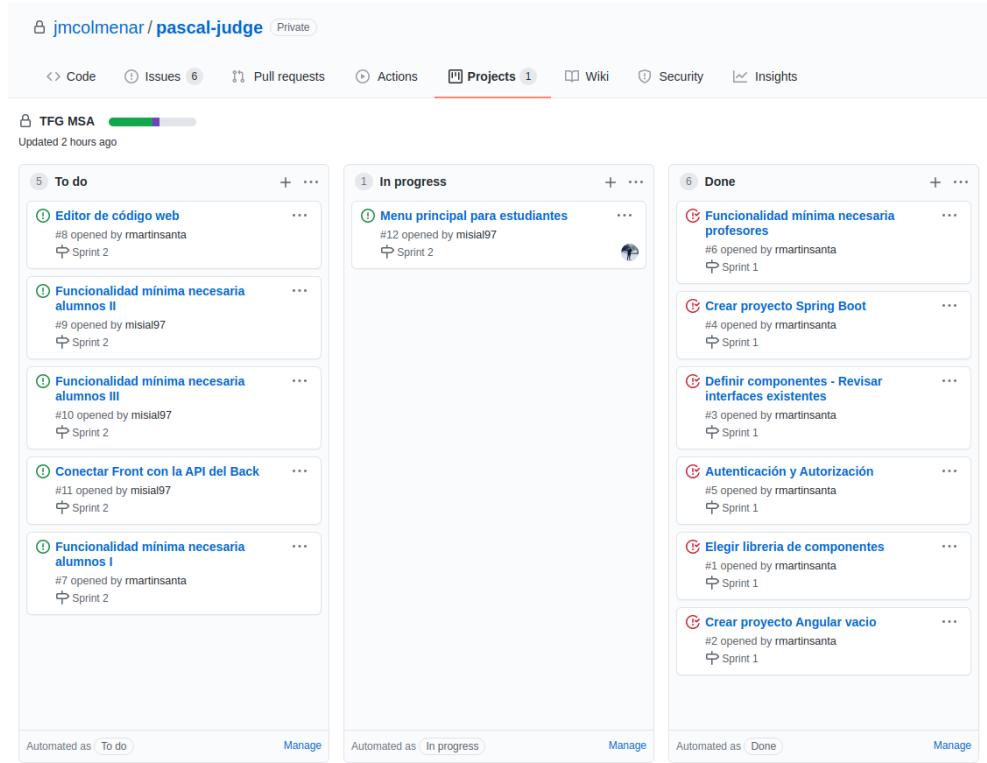


Figura 4.2: Ejemplo de tablero Kanban

Sprint 1:

- Investigar sobre la tecnología
- Montar entorno
- Creación de proyecto Angular
- Se crean primeros controladores en un proyecto SpringBoot (*mock*)
- Funcionalidad *frontend* del profesor
- Revisión con tutores, en esta revisión se comunica que se usará un *backend* real.

Sprint 2:

- Se comienza con funcionalidad de alumno
- Se espera a que llegue el *release* del *backend* para ir adaptando las llamadas a los *endpoints*.
- Se deja funcionando el editor de texto *online Monaco* y llamada a *mock* para los envíos.
- Se comienza a conectar *frontend* y *backend* “real”
- Creación de servicios que conectan *frontend* y *backend*, además de sus objetos respuesta (*POJOs*, “Plain Old Java Object”, es decir, un objeto que simplemente almacena información).
- Reunión con los tutores, nuevas tareas para realizar al proyecto y detalles

a pulir.

Sprint 3:

- Internacionalización. Donde se implementado para los idiomas inglés y español.
- Pantalla de creación de problemas a partir de .zip.
- Pantalla de visualización de problemas con pdf.
- Cambios en el sprint ya que hay tareas que no están muy definidas y se llevan al Sprint 4 y se intercambian por unas de ese cuarto Sprint.
- No hay reunión, ya que se mantienen conversaciones por *email* muy fluidas y no hay nuevas tareas.

Sprint 4:

- Finalizar las tareas marcadas con la etiqueta “*sin_terminar_por_api*”, ya que había que esperar a que la API estuviese completa para estas tareas.
- Creación de la pantalla principal, para tener un perfil público con una mínima funcionalidad.
- Creación del componente Clasificación, para visualizar la tabla de puntuaciones de los equipos.
- Funcionalidad para visualizar en directo los envíos, que se van actualizando cada pocos segundos sin necesidad de recargar. (Se conecta con el *mock* para poder visualizarlo de una manera más sencilla).
- Actualizar de Angular 9 a Angular 12.

4.3. Software y tecnologías utilizadas

En el siguiente apartado vamos a comentar las tecnologías usadas para desarrollar este proyecto.

Lenguajes:

- TypeScript: lenguaje utilizado para desarrollar la lógica en la interfaz.
- HTML: lenguaje utilizado para la maquetación de las pantallas de la aplicación.
- Java 11: lenguaje utilizado para desarrollar un microservicio para poder probar funcionalidades aun no desarrolladas en el *backend*, como por ejemplo el inicio de sesión.
- LATEX: lenguaje utilizado para la generación de la memoria de este TFG.

Tecnologías y herramientas: Para el desarrollo del proyecto han sido importantes el uso de otras librerías para facilitar el mismo y mostrar un aspecto

más profesional para la aplicación. A continuación, se dan detalles de las librerías más importantes utilizadas y para que se usan en el proyecto.

- **Angular:** este *framework* es la base de este desarrollo, usado porque facilita la creación de interfaces ya que da la posibilidad de reutilizar componentes, otro posible *framework* a utilizar podría haber sido React.
- **Spring:** se ha utilizado este framework en el microservicio *mock*, que nos sirve para probar funcionalidad de una manera leal a como sería el comportamiento final entre el *frontend* y *backend*.
- **Insomnia Rest:** cliente REST para realizar peticiones tanto a la API final como al *mock* para realizar pruebas adicionales al uso de la herramienta.
- **Nebular:** Es una librería que es la que da consistencia en el formato y diseño a la aplicación, de ahí se toman todos los elementos como botones, campos de texto... Además es la encargada de dar el tema a la web.
- **Monaco Editor:** Este componente es muy interesante, ya que permite tener un editor de código en la aplicación web y además es multilenguaje, se le puede especificar el lenguaje deseado para que muestre las palabras reservadas de dicho lenguaje para facilitar al usuario la lectura del código realizado, con lo que facilita la resolución de la tarea al usuario.
- **ChartJs:** Esta librería es encargada de realizar gráficos y mostrarlos en la web, en nuestro proyecto únicamente usamos un gráfico de tarta para mostrar al usuario de una manera más clara los casos de prueba que han ido correctamente, que han sido erróneos o que no se han corregido.
- **Ionicons:** Es una librería dedicada a los iconos, en nuestra aplicación no la usamos muchísimo, únicamente en un par de *templates*, pero si que cuando ha sido necesaria deja un resultado muy moderno y más minimalista que una imagen.
- **HttpClient:** Esta librería es muy importante en el proyecto, ya que es la base de la implementación de los servicios que se encargan de ir al *backend* a por la información deseada. *HttpClient* nos permite de una manera muy sencilla realizar distintas peticiones *HTTP* para luego mapear la respuesta a los objetos y poder así manejar la información.
- **Localize:** Lo usamos para poder tener varios idiomas en la aplicación, tanto para textos en las fuentes *HTML*, como en textos que se muestran desde los *TypeScript*. Ahora mismo están en inglés y español, además cabe destacar que se le deberá indicar a la hora de levantar el proyecto el idioma deseado.

Entornos de desarrollo:

- IntelliJ IDEA (JetBrains): entorno de desarrollo utilizado para la parte Java y Spring en el microservicio generado para mockear datos y funcionalidad.
- WebStorm (JetBrains): entorno de desarrollo utilizado para la parte del *frontend* con Typescript y Angular 9, y posterior paso a versión 12.

Características del sistema utilizado:

- Sistema operativo : Ubuntu 20.04, para configurar con mayor facilidad el entorno desde su terminal.
- Procesador: Intel Core i5-10400 CPU @ 2.90GHz
- Memoria RAM: 16GB

5

Implementación y Resultados

En este punto vamos a poner el foco sobre la implementación, estructura del proyecto, detalles a destacar, problemas surgidos durante el desarrollo, además de las pruebas que se han ido realizando para corroborar que es correcto lo que se estaba realizando.

5.1. Conexión *Frontend - Backend*

En esta sección vamos a entrar más en detalle de cómo se tuvo que conectar nuestra interfaz web, con el *backend* a través de la API desarrollada por el TFG que desarrolló el *backend*.

En el comienzo del proyecto al no tener ninguna versión del *backend* estable, se optó por realizar un microservicio en Java con Spring Boot, como *mock*. Este microservicio únicamente tenía controladores para simular la API, y ficheros *json* de respuesta para luego poder procesarlos en *frontend* para visualizar la información de una manera clara para ayudar al usuario.

Una vez más avanzada la parte del compañero, y ya con una API más funcional, se eliminó la conexión con el *mock*, salvo para realizar el inicio de sesión, ya que esa parte no está disponible en el *backend*. El proceso fue algo costoso, ya que hubo que cambiar todas las llamadas al *mock* por el sistema real, además hubo que crear los objetos que concordasen el formato de respuesta para poder manejar la información de una manera sencilla. De esta manera tras finalizar este proceso,

hubo que hacer mejoras en la API del *backend* ya que había funcionalidad sin realizar y algunos fallos que se descubrieron con las pruebas realizadas mediante la interfaz.

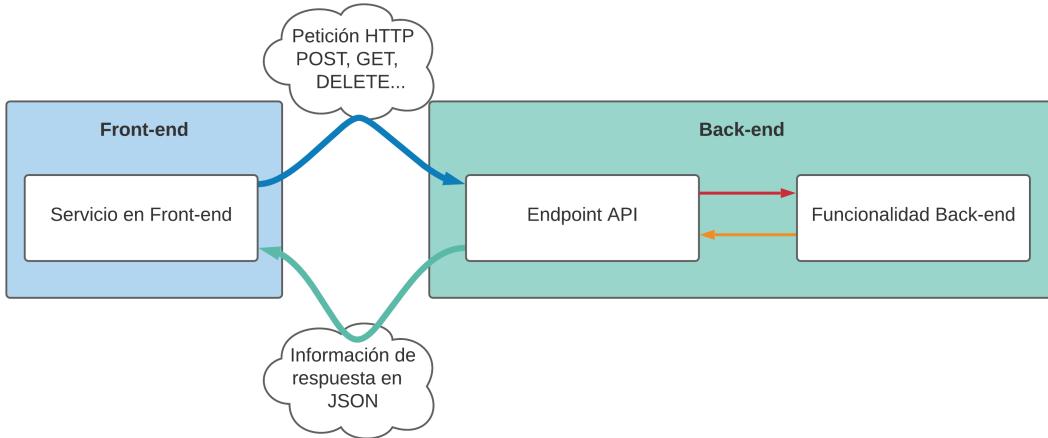


Figura 5.1: Diagrama *Frontend - Backend*

Como podemos ver en la Figura 5.1, la conexión es sencilla, desde la parte del *frontend* se realizan llamadas HTTP hacia la API expuesta por el *backend*, la petición le llega al sistema y devuelve a la interfaz una respuesta que este mostrará al usuario en pantalla para poder realizar su tarea correspondiente. La función desarrollada en este TFG, sería la creación del *frontend*, junto a la conexión al *backend*, mediante la realización de las peticiones HTTP pertinentes y la creación de objetos que devuelve la API, para poder manejarlos desde el *frontend*.

5.2. Internacionalización

En este apartado vamos a comentar como se desarrolló la funcionalidad de internacionalización de la interfaz. Para comenzar tenemos que definir qué es la internacionalización[16]. En la informática, la internacionalización es el proceso de diseñar software de manera tal que pueda adaptarse a diferentes idiomas y regiones sin la necesidad de realizar cambios de ingeniería ni en el código.

Para cumplir este objetivo, nos apoyamos en el *framework* de internacionalización *i18n*. Este funciona a través de identificadores en los textos traducibles, estos identificadores deben estar indicados en los elementos HTML donde se necesite que aparezca dicho texto. Cuando la aplicación levanta configurándole el idioma deseado, busca un fichero del idioma elegido, donde se encuentran todos esos identificadores con el texto deseado en el lenguaje correspondiente.

Esto es muy cómodo, ya que si la aplicación necesita otro idioma, con tan solo coger el fichero donde se encuentran todos los literales de la interfaz y traducir al idioma deseado ya se tendría la totalidad de la herramienta en otro idioma sin necesidad de realizar modificaciones en código fuente.

5.3. Navegación web

En este punto, vamos a dar más detalle de como está estructurada la aplicación web, mostrando el aspecto real de la misma con distintas capturas de pantalla, lo que viene a ser un repaso global de la aplicación web y de como sería una navegación por ella.

Para tener una idea general de como está estructurada la web, se ha creado un diagrama de navegación en la Figura 5.2. En este se puede observar los distintos componentes que se van a explicar a continuación y además diferenciado por roles para poder comprender las diferencias de funcionalidad para cada tipo de usuario.

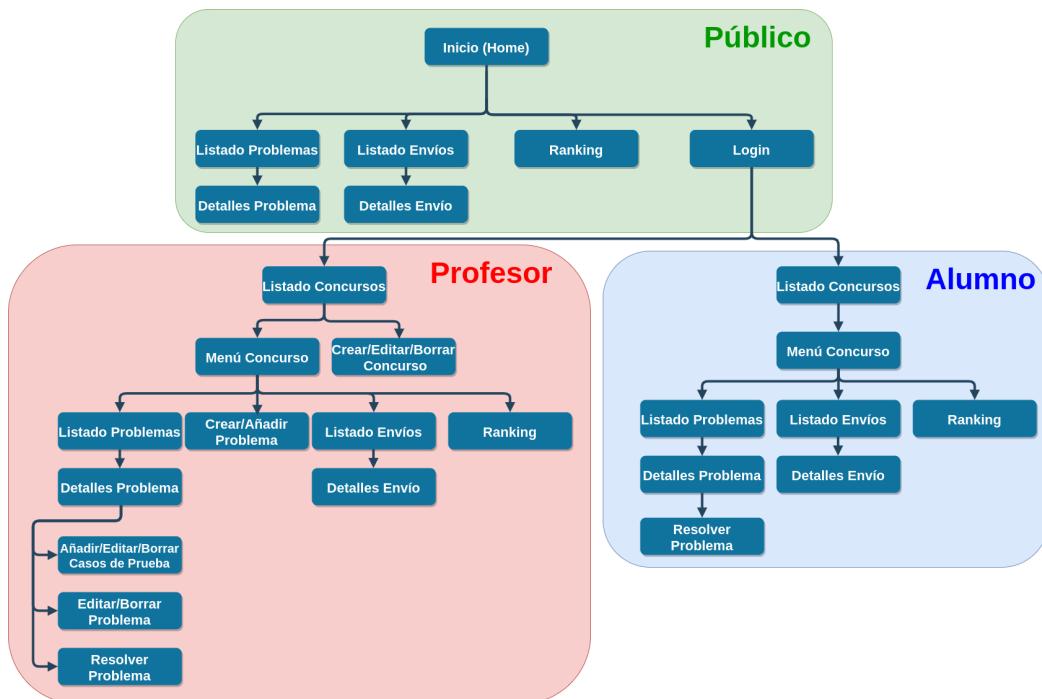


Figura 5.2: Diagrama de navegación web

La Figura 5.3 muestra la pantalla principal de la aplicación, a ella se puede acceder sin iniciar sesión, además tiene unos iconos en la parte media, para poder acceder al listado de problemas, listado de envíos y clasificación del concurso público y poder así visualizar el funcionamiento básico de la web, pero sin poder realizar ningún envío. A continuación, vamos a mostrar estos componentes, que

Capítulo 5. Implementación y Resultados

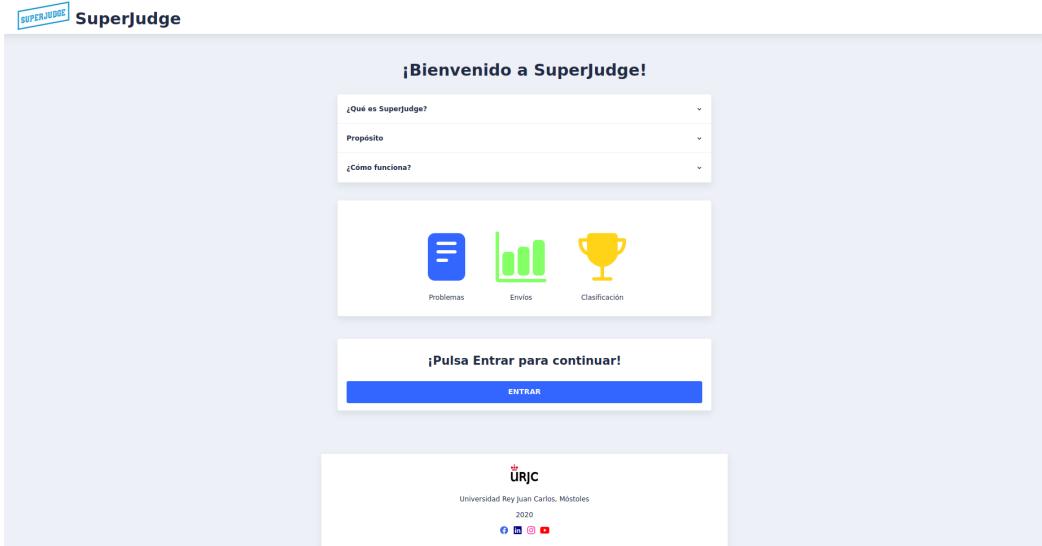


Figura 5.3: Pantalla de inicio

serán prácticamente iguales que los de los usuarios ya con el inicio de sesión hecho, pero con menos funciones.



Figura 5.4: Componente Clasificación

Esta Figura 5.4 muestra el componente *ranking*, el cual tiene como objetivo presentar al usuario la puntuación de todos los equipos integrantes del concurso. Para todos los posibles roles de usuarios es exactamente igual, no se tiene ningún privilegio por ser profesor o alumno en este componente. Cabe destacar, que el componente tiene lógica por debajo para dar toda la información posible al

usuario, como pueden ser los distintos colores o el resumen de la parte inferior. La leyenda de colores y símbolos es la siguiente:

- **Verde claro** : Problema resuelto.
- **Verde oscuro** : Primer equipo en resolver el problema.
- **Rojo** : Problema sin resolver.
- **Mano pulgar arriba** : Numero de problemas resueltos.
- **Mano pulgar abajo** : Numero de intentos fallidos.
- **Reloj** : Mejor tiempo del problema resuelto.

Id	Nombre	PDF
3	CeroVirus	PDF
7	CeroVirus II	PDF
4	Semanas	PDF
2	Primavera	PDF
8	Verano	PDF

Figura 5.5: Componente Listado Problemas

El listado de problemas de nuestro juez, Figura 5.5, tiene el como objetivo, mostrar al usuario el listado de los problemas del concurso deseado. En esta ventana, si que difiere algo de cuando el usuario inicia sesión con el rol profesor, ya que, este es capaz, de eliminar problemas desde esta ventana, el resto de funciones también las tiene pero no ubicadas en la tabla.

Capítulo 5. Implementación y Resultados

The screenshot shows a web-based application interface for 'SuperJudge'. At the top left is the logo 'SUPERJUDGE SuperJudge'. Below it, the title 'Problema CeroVirus' is displayed. A table with columns 'Id', 'Nombre', 'Equipo', 'Límite de Tiempo', and 'Límite de Memoria' is shown. Underneath the table, the section 'Casos de Prueba' contains a PDF viewer. The PDF content discusses a new virus called 'CERO VIRUS' and its propagation mechanism. It also includes a section titled 'Input' with some sample data.

Figura 5.6: Componente Detalles del Problema

En la Figura 5.6, muestra al usuario los detalles del problema seleccionado en la anterior lista, al hacer click en una fila de la Figura 5.5, le lleva a esta pantalla donde puede observar el enunciado entre otros datos. Cabe destacar que, para el perfil alumno desde este mismo componente, puede realizar el envío de la solución para este problema. Para el rol profesor, también tendría acceso a editar los datos del problema.

The screenshot shows a table titled 'Envíos' with columns: Actions, Id, Equipo, Problema, Lenguaje, Fecha, and Resultado. The table lists five entries:

Actions	Id	Equipo	Problema	Lenguaje	Fecha	Resultado
	2	pavloxD	primavera	c++	17/7/2021 17:24:31	OK
	4	pavloxD	CEROVIRUS	java	20/7/2021 13:52:16	KO
	5	pavloxD	semanas	c++	23/7/2021 12:57:24	KO
	6	pavloxD	primavera	java	1/11/2021 12:22:35	OK
	7	other	primavera	java	1/11/2021 12:22:35	Revisando casos prueba...

At the bottom right of the page is the URJC logo.

Figura 5.7: Componente Listado Envíos

La Figura 5.7 muestra el listado de envíos generado para nuestra herramienta, tiene el único objetivo de informar al usuario el listado de los envíos del concurso deseado, es meramente informativa. Además, al igual que en el listado de problemas, podemos hacer click en alguna de las filas, para poder acceder al informe con más detalles sobre ese envío.

En la Figura 5.8 se muestra la ventana de detalles del envío, informa al usuario

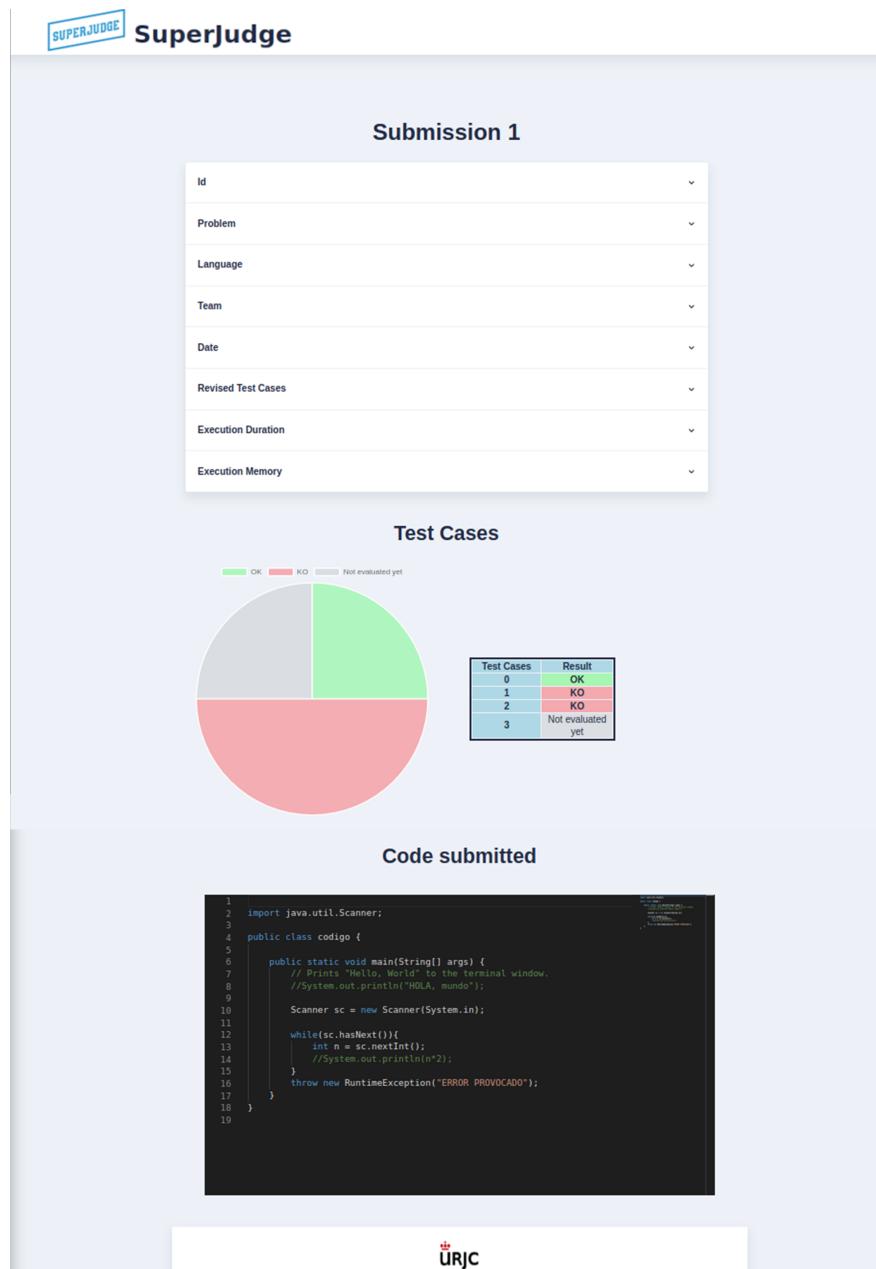


Figura 5.8: Componente Detalles del Envío

los detalles del envío seleccionado en la anterior lista, en este pequeño informe se puede acceder a los detalles del problema que se solucionaba con el envío, parámetros de memoria, duración de la evaluación del problema, entre otros. Además, se le proporciona al usuario un gráfico circular con una leyenda a su derecha, donde muestra para los distintos casos de prueba, los correctos, los erróneos y los que aún no han sido corregidos. Por último, se muestra el código enviado formateado en su lenguaje, algo muy útil para poder ver de un primer vistazo qué ha podido ocurrir.

Capítulo 5. Implementación y Resultados

Una vez vistas todas las ventanas que son accesibles por todos los usuarios, aunque con mayor o menor limitación de funcionalidad, vamos a pasar a los menús específicos para los dos tipos de usuarios que existen en la aplicación.

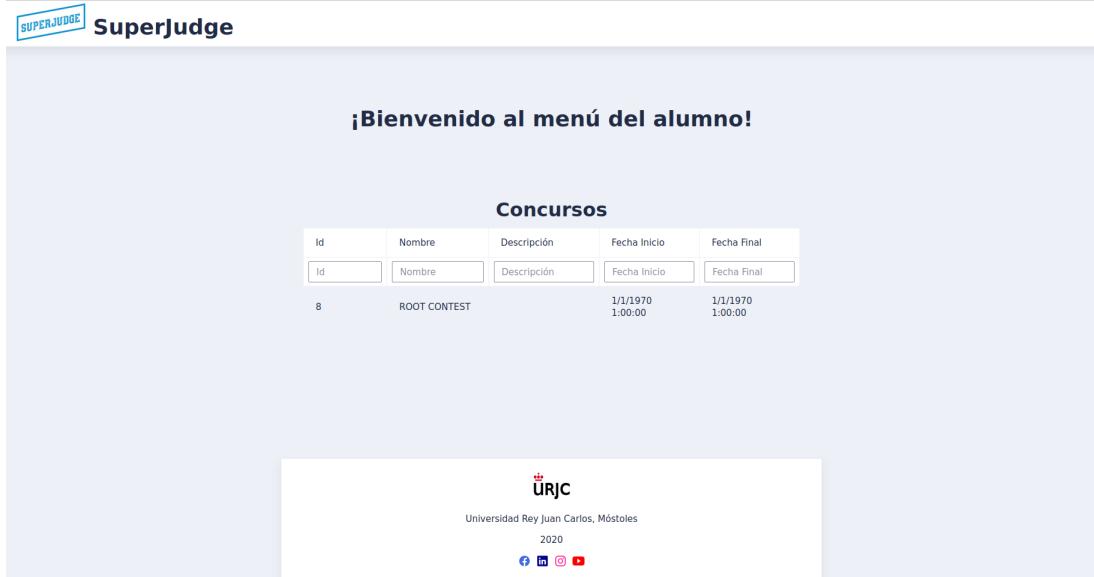


Figura 5.9: Pantalla una vez iniciada la sesión como alumno.

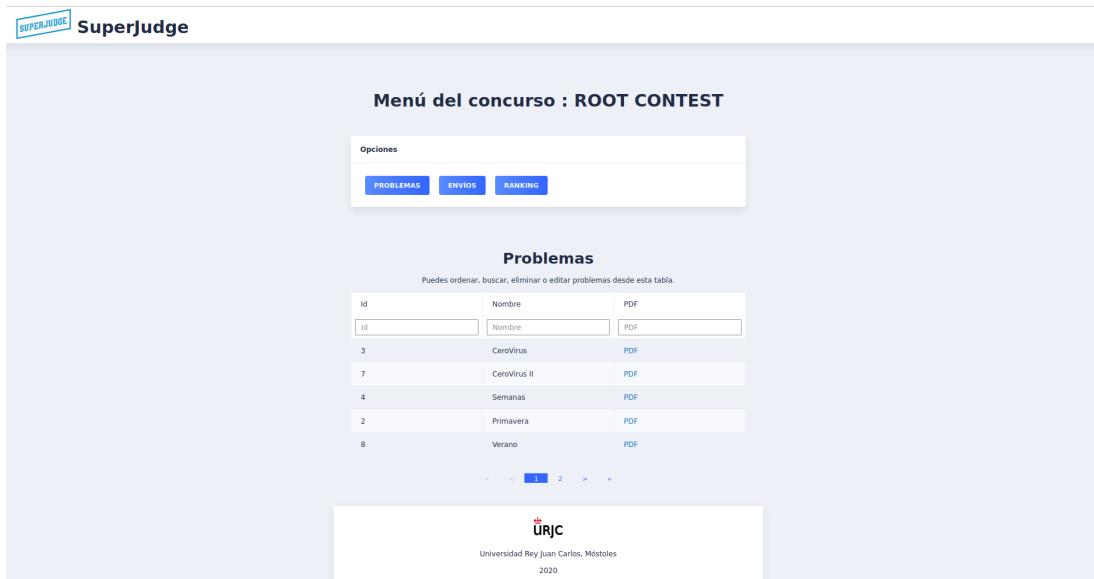


Figura 5.10: Pantalla tras seleccionar el concurso como alumno.

Estas pantallas, Figura 5.9 y Figura 5.10, es lo que el usuario alumno ve una vez ha iniciado sesión. En primer lugar le mostrará el listado de concursos, donde tendrá que seleccionar uno para poder ver los distintos botones de acción. Una

vez ya seleccionado, tendrá a su disposición el listado de problemas, donde como hemos visto antes podrá acceder a los detalles y mandar una solución, también podrá visualizar el listado de envíos en tiempo real y a la clasificación del concurso, al igual que la vista pública.

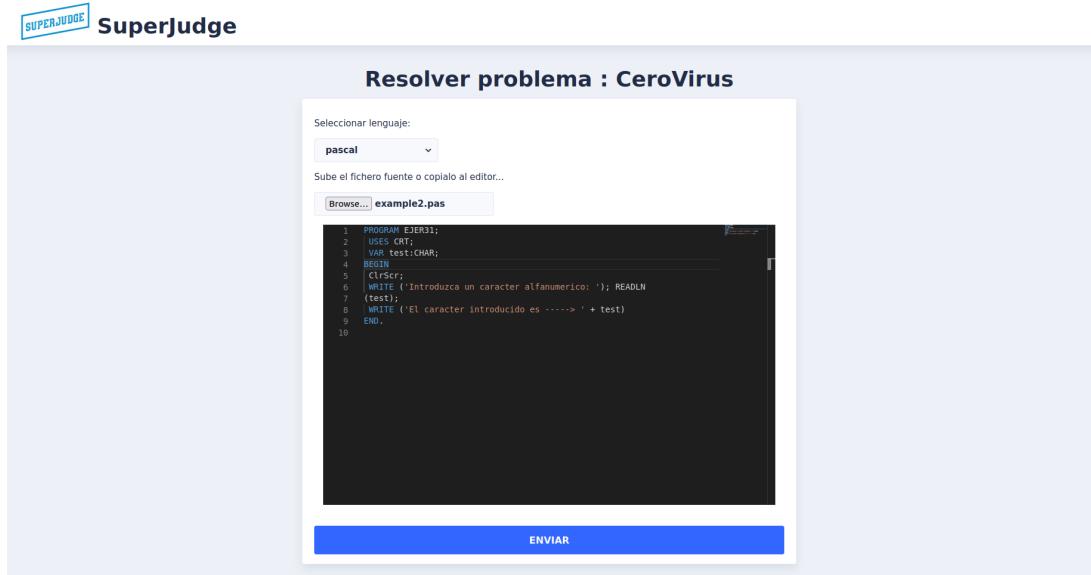


Figura 5.11: Componente Envío de una Solución

En la Figura 5.11 se muestra la vista que el usuario utilizaría a la hora de realizar un envío. Tiene el único objetivo de facilitar al usuario el envío de su solución para el problema deseado. Puede seleccionar el lenguaje en el que quiere tener el editor, para poder ver con mayor comodidad el código de la solución. Además, puede cargar el código al editor seleccionando el fichero directamente desde la web, para acto seguido pulsar en el botón de enviar.

Capítulo 5. Implementación y Resultados

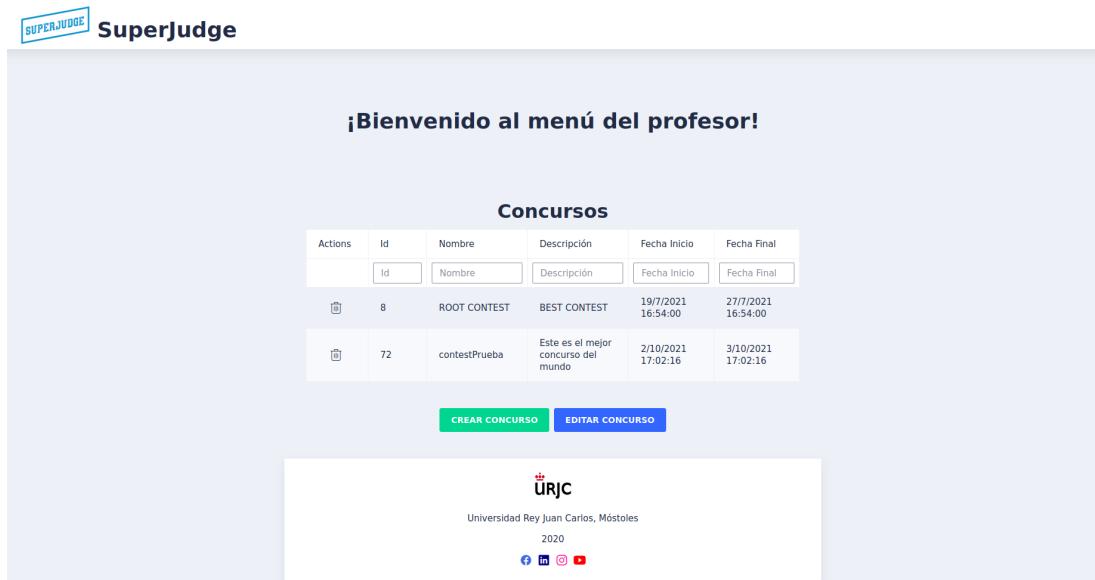


Figura 5.12: Pantalla una vez iniciada la sesión como profesor.

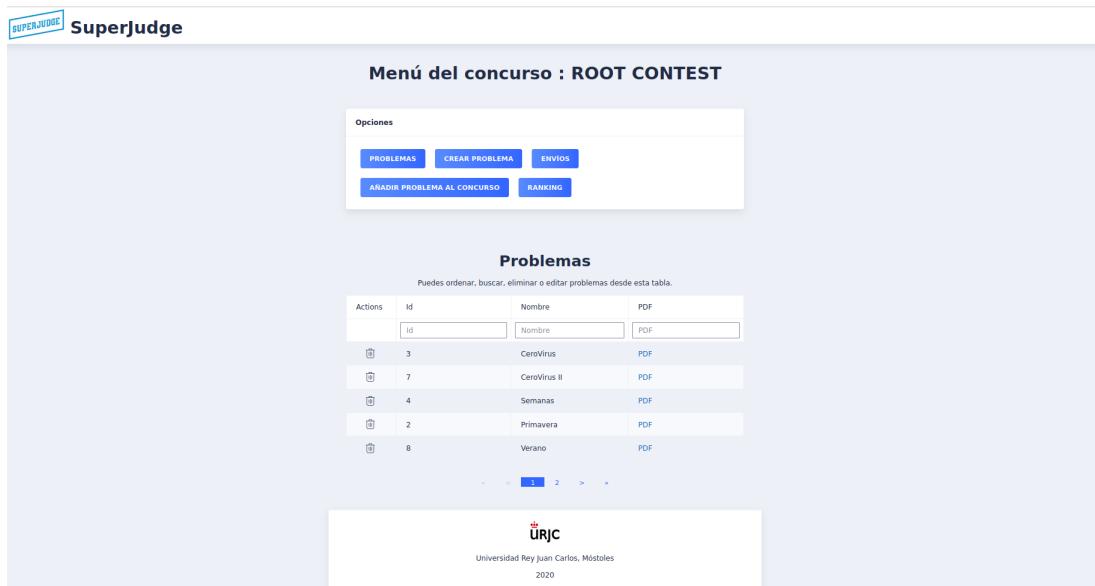


Figura 5.13: Pantalla tras seleccionar el concurso como profesor.

Por último, vamos a mostrar los menús del profesor, Figura 5.12 y Figura 5.13, es lo que el usuario profesor ve una vez ha iniciado sesión, en primer lugar le mostrará el listado de concursos, donde tendrá que seleccionar uno para poder ver los distintos botones de acción. Una vez ya seleccionado, tendrá a su disposición todas las funcionalidades del alumno, además de poder crear, editar y eliminar, tanto concursos, como problemas.

5.4. Estructura del proyecto

La estructura del proyecto es muy importante para que el código sea legible y además de mantenable.

Dentro de /src/app, tenemos distintas carpetas para englobar cada tipo de clases que tenemos en el proyecto y poder así organizarlos de una manera más eficiente.

- **/components:** En este directorio se pueden encontrar a su vez más carpetas, cada una de ellas alberga un componente, es decir, su clase *TypeScript* y su archivo *HTML* correspondiente.
- **/dto:** Dentro de esta carpeta aparecen todos los objetos *TypeScript* necesarios para mapear las respuestas desde el servidor en formato *Json* a estos objetos para poder manejar los datos de una manera más cómoda en esta capa.
- **/services:** En este directorio se encuentran los servicios, estos son encargados de realizar las llamadas la capa *backend* y así dejar la lógica de llamada *HTTP* y mapeo en un sitio aparte. De esta manera el controlador, cuando necesita algún dato, llamaría al servicio deseado y este le devolvería un objeto *TypeScript* de los que están ubicados en el directorio /*dto* y ya manejar esa información para mostrarla al usuario.

5.5. Problemas surgidos durante el desarrollo

Durante el desarrollo del proyecto nos hemos encontrado con numerosos problemas, ya sean por desconocimiento de las tecnologías o por algún error al configurar alguna parte del proyecto.

A continuación, detallo una lista de los problemas más importantes que me han ido surgiendo durante el transcurso de realización de este TFG:

1. Problemas comunes de una nueva tecnología por aprender (Angular), como por ejemplo la familiarización con la sintaxis o la comprensión de funcionalidades de Angular.
2. Problemas a la hora de conectar vía HTTP la interfaz con el microservicio que ejercía de *mock*. El error venía dado por unos headers a la hora de recibir la respuesta, que se soluciona con la etiqueta @CrossOrigin, en la clase marcada con @Controller en el *mock*, para que el navegador maneje de forma segura las solicitudes HTTP de origen cruzado de un cliente cuyo origen es http://localhost:4200 (donde esta ejecutándose la parte *frontend*).

3. Problemas hardware a la hora de montar el entorno y realizar el desarrollo, se tuvo que hacer una ampliación de memoria RAM para poder trabajar, ya que el tener levantado el *frontend* y el *backend* más otras aplicaciones necesarias, hacían que el ordenador tuviese problemas de velocidad por falta de memoria.
4. Algunos problemas a la hora de incorporar el editor de texto Monaco, especialmente porque no lo mostraba, parecía ser problema de CSS, ya que no indicaba el ancho del elemento y no mostraba el componente por pantalla.
5. También hubo problemas cuando se pasó del *mock* al sistema real, ya que no conseguía conectar, el error era igual que el número dos mencionado anteriormente, como solución provisional, ya que al no poder modificar las fuentes del *backend* la solución no podía ser la misma, se optó por crear un proxy en Angular el cual redirige a la API de llamada, de esta manera funciona correctamente la llamada.
6. Como el proyecto se ha tardado algo de tiempo en realizar, aunque se empezó por entonces con la versión más reciente de Angular, la versión 9, hubo que hacer el esfuerzo actualizándolo a la actual que es la 12, en este proceso hubo que realizar algunas modificaciones en dependencias para fuese todo compatible.

5.6. Resumen de resultados

Con la elaboración de este proyecto se ha conseguido dar una interfaz web funcional al usuario para el sistema *backend* desarrollado por otro compañero. Esta interfaz ha respetado los requisitos predefinidos e incluso se ha añadido alguna funcionalidad extra para otorgarle una mayor sensación de robustez.

Aquí pasamos a detallar las **particularidades del proyecto de manera general**:

1. Creación de una *API Mock* para poder probar funcionalidades de la interfaz más fácilmente, como el componente clasificación o el sistema de login.
2. Tener separados los perfiles publico, administrador/profesor y alumno.
3. Interfaz con dos idiomas configurados, inglés y español.
4. Interfaz con un aire moderno y sencilla de manejar

Funcionalidades implementadas en el perfil público:

1. Acceso a visualizar detalles del problema, como el enunciado en PDF, pero no a realizarlos.
2. Visualizar el listado de envíos en el concurso público.
3. Visualizar detalles del envío seleccionado.
4. Visualizar la tabla de clasificación de los equipos del concurso público.

Funcionalidades implementadas en el perfil alumno:

1. Visualizar el listado de concursos en los que participa y acceder a ellos.
2. Visualizar el listado de problemas del concurso seleccionado.
3. Visualizar detalles del problema seleccionado y realizar un envío sobre el mismo.
4. Cargar su fichero de código con la posibilidad de editarlo en el editor *online* con el lenguaje seleccionado.
5. Visualizar el listado de envíos del concurso seleccionado y los detalles del envío.
6. Visualizar la tabla de clasificación de los equipos del concurso seleccionado.

Funcionalidades implementadas en el perfil administrador/profesor:

1. Visualizar el listado de concursos y acceder a ellos.
2. Crear, editar y eliminar concursos.
3. Visualizar el listado de problemas del concurso seleccionado.
4. Crear, editar y eliminar problemas, incluso añadir problemas de un concurso a otro.
5. Visualizar detalles del problema seleccionado y realizar un envío sobre el mismo.
6. Cargar su fichero de código con la posibilidad de editarlo en el editor *online* con el lenguaje seleccionado.
7. Visualizar el listado de envíos del concurso seleccionado y los detalles del envío.
8. Visualizar la tabla de clasificación de los equipos del concurso seleccionado.

6

Conclusiones y Trabajos Futuros

Con la conclusión de este TFG, podemos decir que se han completado todos los objetivos propuestos, por tanto se ha obtenido una interfaz web junto a un sistema previamente elaborado, que es capaz de otorgar al usuario la facilidad para poder utilizar esta herramienta de una manera más simple y más cercana al usuario que no es un profesional en la materia.

Gracias a este proyecto he podido formarme en un campo en el que era totalmente inexperto, como es el *frontend* con Angular, ya que la experiencia laboral que poseo actualmente y los conocimientos que aprendí en la carrera es en su gran mayoría sobre el *backend* y bases de datos. Por lo que ha sido muy satisfactorio a nivel personal poder realizar este proyecto, ya que en su gran parte ha sido aprender esta tecnología y poder así finalizar la realización de este TFG. Todo este aprendizaje y realización del proyecto ha sido también gracias a los tutores implicados en él, ya que la resolución de dudas y el apoyo en ciertos momentos del proyecto han sido indispensable. Además, espero que este proyecto les sea útiles tanto a ellos, como a la ETSII de la Universidad Rey Juan Carlos y que haya valido la pena el esfuerzo puesto en la elaboración de esta interfaz.

Aunque este proyecto es totalmente funcional, en base a los requisitos previos y a aspectos extra que se han ido incorporando al desarrollo, hay algunas funcionalidades que podrían sumarle bastante en la siguiente fase de desarrollo como es el mantenimiento y mejora del sistema, ya que como primera premisa del software, este siempre es mejorable y nunca perfecto.

- Cambio de idioma en caliente sin necesidad de reinicio.

- Añadir un sistema de gestión de cookies para tener mayor control sobre la sesión y cumplir con la Ley General de Protección de Datos Personales.
- Inicio de sesión desde el Aula Virtual.
- Creación de insignias o logros para motivar aún más al alumno.

Bibliografía

- [1] Kahoot! [Online]. Available: <https://es.wikipedia.org/wiki/Kahoot!>
- [2] I. G. Lázaro, “Escape room como propuesta de gamificación en educación,” *Revista Educativa Hekademos*, vol. 27, pp. 71–19, 2019. [Online]. Available: <https://dialnet.unirioja.es/servlet/articulo?codigo=7197820>
- [3] C. E. V. Alma María Pisabarro Marrón, “Gamificación en el aula: gincana de programación,” *Revista de Investigación en Docencia Universitaria de la Informática*, vol. 11, 2018. [Online]. Available: <https://dialnet.unirioja.es/servlet/articulo?codigo=6264619>
- [4] R. Mello, “Predicting the performance of job applicants in coding tests,” in *University Of Gothenburg - Department of Computer Science and Engineering*, 2017. [Online]. Available: <https://gupea.ub.gu.se/handle/2077/52659>
- [5] P. L. Parrilla, *Desarrollo de una plataforma de análisis automático de código*. Escuela Técnica Superior de Ingeniería Informática, URJC, 2021.
- [6] R. A. S. Rueda, “Interfaz web usable: herramienta tecnológica para el proceso de enseñanza-aprendizaje,” *Revista de Comunicación de la SEECEI*, vol. 36, 2015. [Online]. Available: <http://www.seeci.net/revista/index.php/seeci/article/view/141>
- [7] R. A. Carrión, “Usabilidad web: Pensando en el bienestar del usuario,” *Revista Tecnológica Española - RTE*, vol. 27, pp. 67–78, 2014.
- [8] M. MIRZAYANOV, O. PAVLOVA, P. MAVRIN, R. MELNIKOV, A. PLOTNIKOV, V. PARFENOV, and A. STANKEVICH, “Codeforces as an educational platform for learning programming in digitalization,” *Olympiads in Informatics*, vol. 14, pp. 133–142, 2020.
- [9] M. T. Pham and T. B. Nguyen, “The domjudge based online judge system with plagiarism detection,” in *2019 IEEE-RIVF International Conference on Computing and Communication Technologies (RIVF)*. IEEE, 2019, pp. 1–6.
- [10] Repositorio domjudge. [Online]. Available: <https://github.com/DOMjudge/domjudge>
- [11] E. Enström, G. Kreitz, F. Niemelä, P. Söderman, and V. Kann, “Five years with kattis—using an automated assessment system in teaching,” in *2011 Frontiers in education conference (FIE)*. IEEE, 2011, pp. T3J-1.
- [12] P. P. G. Martín and M. A. G. Martín, “¡acepta el reto!: juez online para docencia en español,” in *TICAI 2017: TICs para el Aprendizaje de la Ingeniería*. Universidade de Vigo, 2018, pp. 45–52.
- [13] J. Petit, O. Giménez, and S. Roura, “Jutge. org: an educational programming judge,” in *Proceedings of the 43rd ACM technical symposium on Computer Science Education*, 2012, pp. 445–450.

BIBLIOGRAFÍA

- [14] J. M. V. Andrés Navarro Cadavid, Juan Daniel Fernández Martínez, “Revisión de metodologías ágiles para el desarrollo de software,” *PROSPECTIVA, Universidad Autónoma del Caribe Colombia*, vol. 11, pp. 30–39, 2013. [Online]. Available: <https://www.redalyc.org/pdf/4962/496250736004.pdf>
- [15] L. E. V. A. Eliécer Herrera Uribe, “Del manifiesto ágil sus valores y principios,” *Scientia Et Technica*, vol. 34, 2007. [Online]. Available: <https://www.redalyc.org/articulo.oa?id=84934064>
- [16] G. F. C. Aguilar, “Internacionalización y localización del software en el ámbito mundial,” *INNOVA Research Journal*, vol. 2, pp. 99–111, 2017. [Online]. Available: <https://revistas.uide.edu.ec/index.php/innova/article/view/347>



Apéndice

A.1. Guía de Instalación

Para realizar la instalación y por tanto ejecución del sistema deberemos realizar los siguientes pasos:

1. Tener todo el código fuente, tanto del *backend* como del *frontend*, disponible en el repositorio del proyecto.
2. Levantar el sistema, en las fuentes del *backend*, hay un fichero *docker-compose.yml*, el cuál contiene la configuración para la correct ejecucción del mismo. Se debe ejecutar el siguiente comando desde la ruta donde se encuentre dicho fichero:

```
docker-compose up -d
```

3. Levantar el *mock*, en estos momentos además del sistema, la interfaz necesita una *API Mock*, la cual es capaz de realizar el inicio de sesión para poder visualizar las pantallas de alumno o profesor, o de dar una batería de datos a la clasificación para de esta manera poder probar de una manera más sencilla las funcionalidades. Para ello será necesario, ejecutar el *jar* para el despliegue de esta api, con el comando:

```
java -jar api-mock.jar
```

4. Levantar la interfaz, como explicamos anteriormente, la interfaz esta preparada para dos idiomas, estos deben de ser configurados en el comando que hace que levante la interfaz.

En español: `ng serve --configuration=es`

En inglés: `ng serve --configuration=en`

