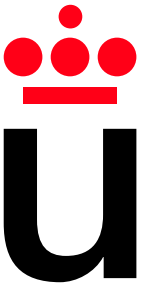

Universidad
Rey Juan Carlos

Escuela Técnica Superior
de Ingeniería Informática

Grado en Ingeniería del Software

Curso 2024-2025

Trabajo Fin de Grado

DISEÑO DE UNA INTERFAZ ACCESIBLE EN LA HERRAMIENTA EDUCATIVA DE PROGRAMACIÓN IUDEX

Autor: Adaya María Ruiz Mayoral
Tutores: Raúl Martín Santamaría
Isaac Lozano Osorio

Agradecimientos

Me gustaría dedicar este Trabajo de Fin de Grado a mis amigos y mi familia, a los que agradezco su acompañamiento y apoyo a lo largo de estos cuatro años de carrera universitaria.

También me gustaría agradecer a mis compañeros y amigos de programación competitiva, además de a la asociación de alumnos Dijkstraidos, por enseñarme un ámbito de la informática tan interesante y por todo lo aprendido y vivido gracias a ellos.

Por último, agradezco a mis tutores por guiarme y apoyarme durante el desarrollo de este trabajo, el cual no habría sido posible sin ellos.

Resumen

En un mundo en constante evolución tecnológica, el ámbito de la enseñanza no es una excepción. Hemos presenciado un proceso de digitalización que ha transformado la forma de educar, especialmente en áreas relacionadas con la informática. A raíz de este, han surgido herramientas para facilitar el aprendizaje en un contexto digital, como los jueces de código, que permiten a estudiantes o desarrolladores enviar soluciones a problemas de programación y que estas sean evaluadas de forma automática. Estas plataformas han ganado popularidad en el ámbito académico por su retroalimentación inmediata y reducción de la carga de trabajo de los docentes. Por estos motivos, algunos alumnos de la Universidad Rey Juan Carlos han desarrollado un juez de evaluación de código llamado **iudex**.

El principal objetivo de esta herramienta es ser utilizada en un ámbito educativo, como apoyo para docentes y alumnos en asignaturas de programación. Por esto, es importante que la interfaz sea funcional e intuitiva, y permita su uso a cualquier persona sin suponer una dificultad añadida al aprendizaje. La motivación de este trabajo es mejorar la interfaz de esta herramienta para que cumpla estos requisitos. Para conseguir cumplir este objetivo, se realizará una evaluación de la interfaz existente para detectar problemas y posibles mejoras. A continuación, se diseñará un prototipo de la interfaz deseada, para el que se tendrá en cuenta el contraste de los colores, la tipografía a usar y la estructura de navegación, entre otros.

Finalmente, se implementará la interfaz teniendo en cuenta las conclusiones previas, procurando conseguir la mayor funcionalidad posible, y añadiendo diversas prestaciones que no estaban presentes en la interfaz anterior, como el inicio de sesión institucional de la Universidad Rey Juan Carlos, el uso de la API o la posibilidad de realizar los problemas y que sean evaluados.

Palabras clave:

- Interfaz web
- Juez automático
- Accesibilidad
- Autenticación

Siglas y terminología

- API: Interfaz de Programación de Aplicaciones, del inglés Application Programming Interface, es un conjunto de reglas y protocolos que permiten a diferentes aplicaciones comunicarse entre sí, definiendo cómo deben interactuar los componentes de software y proporcionando una interfaz para acceder a funciones o servicios específicos.

- IDE: entorno de desarrollo integrado, del inglés Integrated Development Environment, proporciona herramientas como un editor de código, un depurador y un compilador o intérprete en una única interfaz para facilitar el desarrollo de software.

- Backend: parte de la aplicación que gestiona la lógica del negocio, el acceso a la base de datos y el procesamiento de datos. No es visible para el usuario final y se encarga del funcionamiento interno del software.

- Frontend: parte de la aplicación que interactúa directamente con el usuario. Es la interfaz visual con la que los usuarios interactúan, encargándose de la presentación y experiencia de usuario.

- WebSockets: protocolo de comunicación que permite una conexión bidireccional y en tiempo real entre un cliente y un servidor a través de una única conexión TCP, utilizados en aplicaciones que requieren actualizaciones instantáneas.

- Framework: conjunto de herramientas, bibliotecas y convenciones que proporcionan una estructura estándar para desarrollar aplicaciones y facilitan el desarrollo.

- Mock: técnica utilizada en pruebas de software que consiste en crear objetos simulados que replican el comportamiento de componentes reales.

- JWT: estándar de token basado en JSON, del inglés JSON Web Token, se utiliza para transmitir información entre distintas partes de manera segura.

- Proxy: intermediario que actúa en la comunicación entre un cliente y un servidor.

- SPA: del inglés Single Page Application, se refiere a una aplicación web que funciona dentro de una única página, actualizando dinámicamente el contenido sin recargar la página completa.
- DTO: del inglés Data Transfer Object, son objetos que se utilizan para transportar datos entre diferentes capas de una aplicación.
- Token: unidad de autenticación que se utiliza para verificar la identidad del usuario o el acceso a un recurso.

Índice de contenidos

Índice de figuras

1 Introducción

En este capítulo se analizará el uso actual de los jueces de evaluación de código, además de los motivos de su reciente popularidad y del incremento de su utilización en el contexto educativo. A continuación, se compararán algunos de estos jueces junto a sus características más destacables, puesto que serán una referencia para el desarrollo de este Trabajo de Fin de Grado y para la mejora del juez de evaluación de código *iudex*. También se comentará el estado del arte de la aplicación, analizando el estado del backend y del anterior frontend.

1.1. Contexto y alcance

Desde la aparición de Internet, la tecnología ha estado incorporándose poco a poco en el ámbito educativo, en todos los niveles. Especialmente a raíz de la pandemia provocada por el COVID-19, hemos presenciado un gran aumento en la digitalización de la enseñanza, ya que el mundo se ha visto obligado a adaptarse a un modelo educativo online [1]. Este cambio se presentó de forma repentina e inesperada, por lo que las herramientas y metodologías usadas inicialmente han tenido que evolucionar hasta conseguir adaptarse a las necesidades de los alumnos y los docentes, que han descubierto un gran potencial en la aplicación de la tecnología a la educación.

Un punto clave para que esta digitalización sea posible han sido los recursos digitales amigables, gamificados y fáciles de usar, puesto que hacen que los contenidos sean más atractivos para el alumnado y aumentan su motivación. Además de esto, contribuyen al desarrollo de habilidades y competencias digitales. Este

tipo de recursos favorecen la innovación y el perfeccionamiento del desempeño docente, por lo que es importante continuar utilizándolos y mejorándolos a pesar de que la educación haya vuelto a la presencialidad [2].

Uno de los recursos que han surgido a raíz de la digitalización de la enseñanza son los que se utilizan para la evaluación de competencias automatizada, especialmente utilizados en el ámbito de la ingeniería informática y ramas similares, tanto en el mundo laboral como en el educativo, como por ejemplo los jueces de evaluación de código. Hay investigaciones [3] que demuestran que, en general, estas herramientas resultan muy útiles y beneficiosas para el aprendizaje. Principalmente, se ha demostrado que reducen la carga de trabajo de los docentes, mejoran el aprendizaje de los alumnos y aumentan la cantidad de trabajo que realizan, además de ser bien recibidos por los estudiantes y permitir que puedan realizar actividades sin necesitar la presencia del profesor, siguiendo su propio ritmo [3].

Una manera de hacer que estas herramientas sean bienvenidas por los alumnos es la gamificación, que es un concepto que se refiere al uso de elementos del diseño de videojuegos en otros contextos no relacionados con el juego, para hacer que un producto o servicio sea más atractivo, divertido y motivador [4]. Este concepto se puede observar en los jueces de evaluación de código, puesto que motivan a los estudiantes a programar y a hacer código de mayor calidad, ya que se proponen los problemas de programación como un juego. La dinámica de los concursos motiva a los estudiantes a completar todos los problemas, como si completaran los niveles de un juego. Además, recompensa la velocidad y la corrección del resultado mejorando la posición en la clasificación del concurso, lo que les motiva a coger más soltura programando para poder hacer los problemas mejor y más rápido.

Además, los jueces de evaluación de código también suponen una opción muy interesante para evaluar los conocimientos de los estudiantes de programación, puesto que son una alternativa a programar a papel que facilita tanto al estudiante como al docente la ejecución de exámenes. Al programar a papel, los alumnos no disponen de flexibilidad para modificar el código ya escrito, además de que carecen de las herramientas para detección de errores o erratas que proporciona los IDEs, lo que hace que el código sea menos legible y desorganizado, lo que dificulta la labor del docente. A pesar de esto, hay que tener en cuenta que realizar los exámenes en un ordenador también tiene sus inconvenientes, el principal de ellos sería la facilidad de recurrir al uso de métodos no legítimos por parte de los alumnos [5].

Debido a estos motivos, los jueces de código se están popularizando en el ámbito de la informática para propósitos tanto educativos como lúdicos. A continuación se profundizará en las características y ventajas de los jueces, además se realizará un análisis de los más populares y/o interesantes.

1.2. Jueces de código

Los jueces de evaluación de código son plataformas que permiten, esencialmente, compilar y ejecutar un código fuente contra un conjunto de casos de prueba (entradas y salidas) definidos para un problema, y verificar si los resultados de dicho código fuente son los esperados. Para evaluar el código no solo tiene en cuenta los resultados, sino también el tiempo de ejecución y la memoria necesitada, lo que hace que la evaluación sea más completa, teniendo en cuenta tanto salidas como optimización. Tienen grandes ventajas, como que proporcionan una retroalimentación instantánea sobre el código evaluado, además de que la evaluación realizada es completamente objetiva, sin sesgo humano.

Uno de sus principales usos se da en la programación competitiva, que consiste en competiciones en las que programadores resuelven problemas de programación en un tiempo limitado. En estas competiciones se emplean los jueces de evaluación de código, puesto que permiten crear concursos con los problemas a resolver, facilitando a los participantes consultar cada problema e intentar resolverlo. El factor más crucial en estas competiciones es el tiempo, por lo que la rápida retroalimentación que aportan los jueces de código hace que sean muy utilizados en este contexto. La competición más antigua y de las más conocidas es la *International Collegiate Programming Contest* (ICPC), que se originó en la década de 1970 y no ha dejado de incrementar su popularidad, con un total de 75.000 participantes de 3.450 universidades de 111 países distintos en su edición de 2023. Esta competición se desarrolla utilizando el juez en línea *DOMjudge*[1], ya que permite crear instancias privadas en las que realizar los concursos [6]. También existen otros jueces usados esencialmente para programación competitiva, como *Codeforces*[2] o *¡Acepta el reto!*[3], que se usan principalmente para entrenar y practicar, ya que cualquier persona puede acceder a sus concursos y problemas públicos.

Otro ámbito en el que los jueces de código se han incorporado es el laboral, ya que para los puestos de trabajo relacionados con la programación en muchos casos se realizan entrevistas técnicas para conocer mejor las habilidades del posible empleado, e intentar predecir su actuación en la empresa. Estas pruebas técnicas pueden consistir en problemas de programación competitiva, ya que es una buena manera de comprobar la velocidad y capacidad de resolución de problemas del candidato, por lo tanto, también se utilizan jueces de código para llevarlas a cabo [7].

El ámbito en el que nos centraremos es el académico, puesto que el propósito del juez **iudex** es ser utilizado en este. Los jueces de código también pueden ser utilizados en asignaturas de programación para asignar tareas que ayuden al alumnado a practicar y estudiar o para evaluar sus conocimientos, de una

[1] `https://www.domjudge.org/`
[2] `https://codeforces.com/`
[3] `https://aceptaelreto.com/`

forma mucho más atractiva para ellos y facilitando la labor de los docentes. Estas plataformas ofrecen ventajas muy interesantes, ya que ofrecen a los alumnos una rápida retroalimentación sobre su solución a un problema o proyecto, incluyendo detalles sobre errores en caso de que se hayan cometido. Adicionalmente, registran información sobre el número de intentos efectuado o el tiempo que se ha tardado en realizarlos, lo que facilita la evaluación que llevan a cabo los docentes [8]. A continuación, se analizarán algunos de los jueces de código más utilizados y algunos que se han considerado interesantes para el desarrollo de este trabajo por su desempeño en el ámbito académico.

1.2.1. DOMjudge

El principal juez de código usado como referencia para este trabajo es el anteriormente mencionado *DOMjudge* [9], es un juez gratuito y de código abierto, especialmente popular para la realización de concursos presenciales con un grupo de problemas y límite de tiempo fijo, como el *International Collegiate Programming Contest* (ICPC) o el *AdaByron*, gracias a la posibilidad de creación de instancias privadas. Además, este juez ya es usado en algunas asignaturas de la Universidad Rey Juan Carlos, como por ejemplo Diseño y Análisis de Algoritmos, además de en cursos sobre programación competitiva organizados por asociaciones de la universidad, por lo que los alumnos están familiarizados con su interfaz, la cual se analizará a continuación.

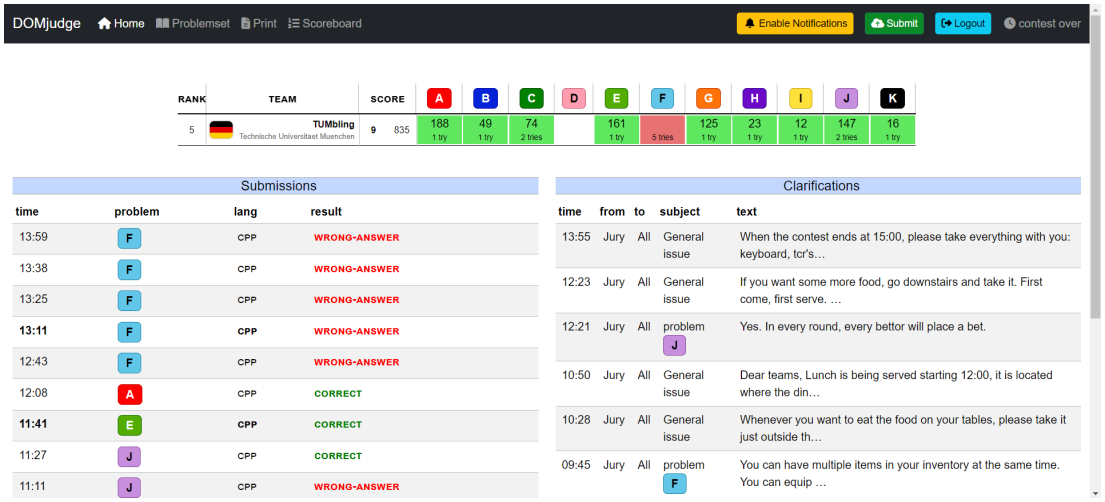

Figura 1.1: Pantalla inicial de *DOMjudge*

Al acceder como equipo participante de un concurso podemos ver una pantalla inicial, mostrada en la figura 1.1, con nuestra posición en la clasificación y el estado actual de nuestros envíos, pudiendo ver fácilmente qué problemas hemos completado, fallado o cuáles no hemos intentado. Además, podemos ver una tabla

con nuestros envíos en orden cronológico, indicando el resultado de cada uno, el momento en el que se ha realizado y el lenguaje usado. Por último, podemos ver un apartado de *clarifications*, donde se muestran las aclaraciones que se han realizado a lo largo del concurso, sobre dudas generales o sobre problemas específicos. Esta ventana tiene un diseño muy amigable para el usuario, ya que, a pesar de tener mucha información, la muestra de una forma sencilla de interpretar gracias a la codificación por colores y el diseño estructurado en tablas.

Además, podemos ver en la barra superior un menú sencillo para navegar por la página, con iconos que ayudan a interpretar el propósito de cada sección. Esta misma barra también dispone de un botón para realizar envíos que está visible en todo momento, lo que es una característica muy útil para los alumnos, puesto que en otros jueces puede llegar a ser complicado encontrar el lugar para realizar los envíos, lo que puede provocar una ligera pérdida de tiempo en momentos de tensión e incrementar el nerviosismo. También cabe destacar el contador en tiempo real que indica cuánto tiempo falta para que el concurso finalice y la barra de progreso, ya que al estar siempre visibles ayudan a que los usuarios no pierdan la noción del tiempo.

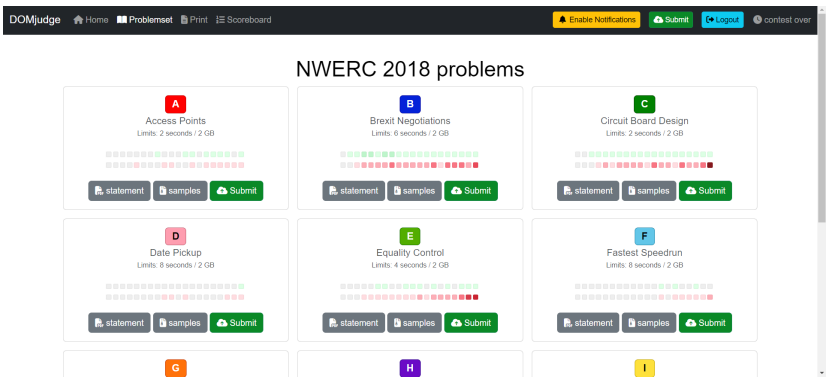

Figura 1.2: Vista de problemas de *DOMjudge*

Respecto a la ventana para visualizar los problemas, en la figura 1.2 podemos ver que están organizados en tarjetas de forma clara y separada, además de disponer de botones claramente etiquetados, lo que evita confusiones. Podemos ver el límite de espacio y tiempo, abrir el PDF con el enunciado del problema en otra ventana o descargar los casos de prueba. En este caso, la opción de descargar los casos de prueba no es la mejor para principiantes, puesto que estos generalmente tendrían que descomprimir el archivo y abrirlo con otra herramienta, lo que puede suponer demasiado tiempo. Esta característica es más adecuada para usuarios con un alto nivel de conocimientos de programación, ya que su propósito ideal es usarlos para ejecutar directamente un código por línea de comandos usando los archivos descargados como archivo de entrada. Como el propósito de *iudex* es principalmente educativo, se utilizará una alternativa más sencilla y amigable para principiantes.

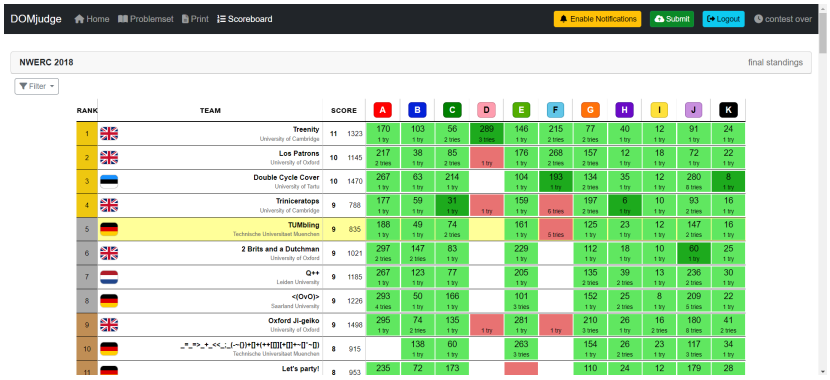

Figura 1.3: Vista de la clasificación de *DOMjudge*

Por último, en la vista de equipo, tenemos la clasificación del concurso. Podemos ver en la figura 1.3 que también resulta fácil de interpretar para el usuario, ya que está codificada por colores y el diseño de las columnas es claro y ordenado. Se puede ver claramente la posición, puntuación y números de problemas resueltos de cada equipo, además de disponer de la opción de filtrar la información. Además, al final de la pantalla podemos ver un resumen de las estadísticas de cada problema y una leyenda con el significado de los colores de cada celda.

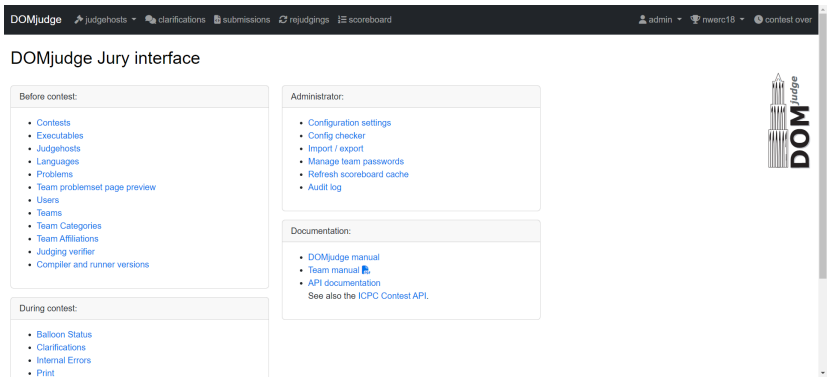

Figura 1.4: Vista de administrador de *DOMjudge*

Respecto a las vistas de juez y de administrador, son iguales a diferencia de que la vista de administrador dispone de un botón de *judgehosts* en la barra superior, y del botón *team* en la vista de juez, que permite acceder a la vista de equipo. En la figura 1.4 podemos ver que en la parte derecha disponemos de dos desplegables para cambiar de concurso, activar notificaciones o cerrar sesión, entre otros; y en la parte izquierda disponemos de los botones *clarifications*, *submissions*, *scoreboard* y *rejudge*. En la vista de *clarifications* o aclaraciones y de *submissions* o envíos se presenta una tabla que se puede filtrar, y que muestra los detalles de cada elemento cuando se selecciona. Además, en la vista inicial de ambos roles podemos ver listas de diferentes elementos como concursos, problemas o usuarios, que al ser seleccionados nos redirigen a una pantalla más detallada

en la que podemos editar, añadir o borrar algunos de estos elementos, o realizar otros tipos de gestiones.

1.2.2. ¡Acepta el reto!

El juez de código *¡Acepta el reto!* [10] es una plataforma web desarrollada por la Universidad Complutense de Madrid, que consiste en un almacén de problemas de programación con un juez en línea para la corrección automática de estos problemas. No permite la creación de concursos, es una herramienta usada para practicar, ya que tiene una gran colección de problemas. Ahora nos centraremos en analizar su interfaz.

Figura 1.5: Pantalla inicial de *¡Acepta el reto!*

Al igual que en los jueces que hemos analizado anteriormente, podemos ver que dispone de una barra superior de navegación que está visible en todo momento, lo que facilita al usuario el movimiento por la web. También dispone de un botón para iniciar sesión o ver el perfil del usuario, y otro para buscar problemas o usuarios. En la página de inicio mostrada en la figura 1.5 disponemos de unas secciones informativas, que explican en qué consiste la página y cómo empezar a usarla, lo que usaremos como referencia para el desarrollo de este trabajo, ya que ayuda a usuarios inexpertos a entender el propósito y funcionamiento de un juez de código. Además cabe destacar que dispone de un diseño minimalista y claro, destacando claramente los títulos de las secciones y usando una tipografía legible y con un tamaño adecuado.

Respecto a la vista de problemas, se puede ver que están bien organizados, en categorías o en volúmenes, lo que facilita encontrar problemas de un tipo específico o por su identificador. Además, al igual que en todas las pantallas, en todo momento podemos ver en qué parte de la página estamos, lo que facilita la navegación por la página y concretamente por las clases de problemas. En la figura 1.6 podemos ver la vista de detalles de un problema, donde se muestra el enunciado, los casos de prueba y varias secciones interesantes, como estadísticas

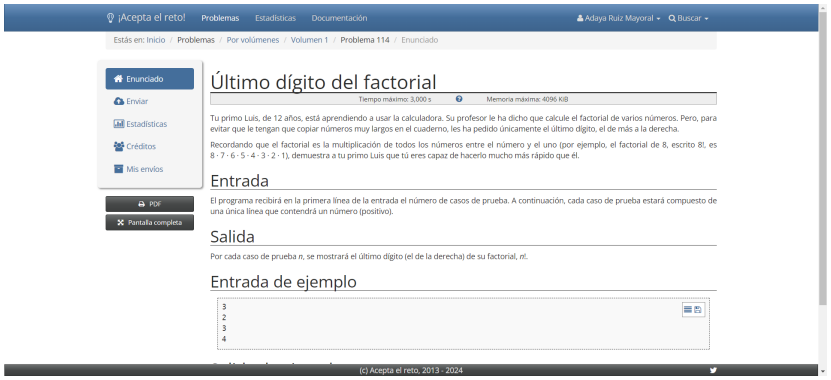

Figura 1.6: Vista de detalles de un problema de *¡Acepta el reto!*

del problema o envíos del usuario. Esta vista es muy completa, ya que disponemos de las opciones de copiar o descargar los casos de prueba, descargar el problema en PDF o ponerlo en pantalla completa, lo que ayuda a enfocar la atención del usuario únicamente en el enunciado del problema junto a sus entradas y salidas. Además, el botón de envío se encuentra visible al principio de la página, aunque se podría mejorar destacándolo más y haciendo que este sea persistente en la vista de la interfaz.

En la vista de envío podemos destacar que dispone de un editor de código, con algunas funcionalidades añadidas. Permite que el usuario añada comentarios para diferenciar el envío que va a realizar, y también permite pegar una plantilla en el editor de código, que puede ser útil para principiantes o para ahorrar tiempo al codificar el problema. Por último, respecto a la vista de envíos, podemos considerar que tiene un diseño limpio y claro, aunque sería mejorable añadiendo la opción de filtrarlos, ya que aparecen todos los envíos de todos los usuarios y problemas. También cabe destacar la documentación, que se encuentra muy completa y accesible, ayudando a todo tipo de usuarios a solucionar sus dudas y problemas.

Podemos concluir que este juez de código tiene un diseño de interfaz interesante y sencillo, que tendremos en cuenta para el desarrollo de *iudex*. Además, también tendremos en cuenta características como la información mostrada en la pantalla inicial.

1.2.3. CodeOcean

A diferencia de los jueces mencionados anteriormente, cuyo principal enfoque es la programación competitiva, ahora analizaremos otras aplicaciones más enfocadas al ámbito educativo.

CodeOcean[4] [11] es una herramienta de evaluación automática basada en la web, que ofrece un entorno de desarrollo y ejecución para problemas de programación. Permite desarrollar el código en un editor y ejecutarlo en su servidor, ofreciendo su salida, además de ponerle nota y proveer retroalimentación gracias a los test unitarios que incluye. Además, se puede integrar con entornos de aprendizaje masivo en línea (MOOCs), lo que facilita su uso en otras plataformas de enseñanza. Finalmente, cabe destacar que dispone de la posibilidad de realizar preguntas en la propia plataforma sobre el programa que se está desarrollando, ya que tiene un foro de discusión.

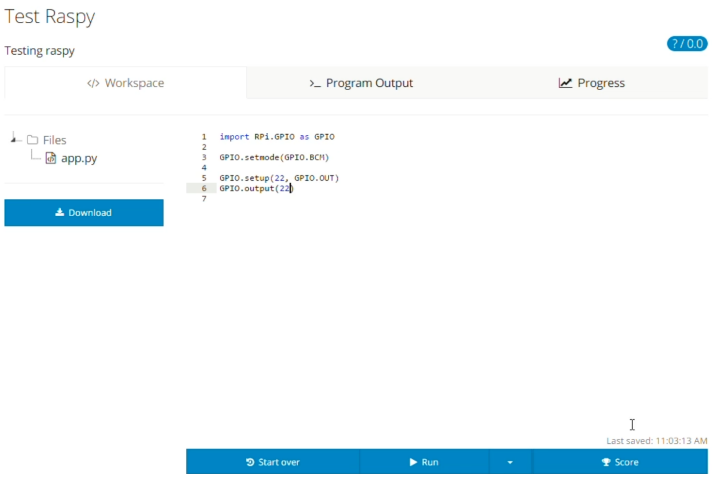

Figura 1.7: Vista de desarrollo de CodeOcean

1.2.4. INGInious

INGInious[5] [12] es una plataforma de evaluación automática de código con el propósito de ser usada especialmente en entornos académicos. Permite el uso de cualquier lenguaje de programación, además de la integración con plataformas educativas como *edX* o *Moodle*. Se basa en la creación de tareas, cuyos criterios de corrección son personalizables por los docentes. Dichas tareas deberán ser completadas por los alumnos, que pueden intentarlas de forma ilimitada, recibiendo el resultado de cada uno de sus intentos.

Respecto a su interfaz, visible en la figura 1.8, dispone de un menú en el lateral izquierdo donde se puede cambiar el idioma de la página, acceder al listado de cursos e iniciar sesión o registrarse. Al seleccionar uno de los cursos, se muestra un listado de las diferentes tareas que lo componen, las cuales también pueden ser seleccionadas para ver su descripción y realizarlas. Durante la realización de

[4] `https://github.com/openHPI/codeocean`
[5] `https://inginious.org/`

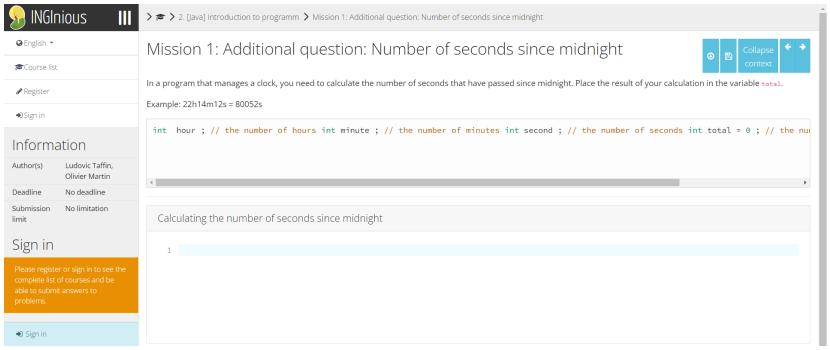

Figura 1.8: Vista de realización de tarea de INGInious

una tarea se dispone de diferentes opciones, como cargar un archivo, guardar el que se ha desarrollado y realizar un envío.

1.2.5. Codeboard

Codeboard[6] [13] es una aplicación web que consiste en un entorno de desarrollo con la opción de correcciones automáticas. Los docentes pueden asignar tareas a sus alumnos y, si lo desean, pueden activar la opción de corrección automática para que se puntúe la tarea. Esta corrección se puede realizar mediante entrada y salida o mediante tests unitarios, lo que está a elección del creador del proyecto. Su interfaz es sencilla, similar a un típico entorno de desarrollo, con botones para compilar, ejecutar o enviar el código, entre otros. Adicionalmente, permite la colaboración en tiempo real, lo que representa una funcionalidad muy interesante que puede facilitar la realización de trabajos grupales y el seguimiento del trabajo de los alumnos por parte de los docentes.

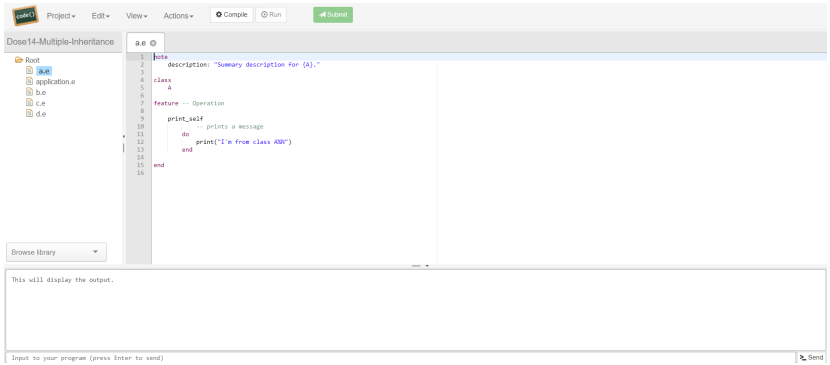

Figura 1.9: Vista del editor de código de *Codeboard*

[6] `https://codeboard.io/`

1.2.6. Virtual Programming Lab

Virtual Programming Lab[7] [14] es un complemento de la plataforma *Moodle* que permite asignar tareas a los alumnos y resolverlas en la misma página. Dispone de un editor de código que permite ejecutar, evaluándolo automáticamente y dando retroalimentación de la ejecución, además de los tests que han sido fallados o superados. También permite puntuar cada tarea según la cantidad de tests que han sido superados. Además, tiene restricciones para la edición del código y no permite pegar, por lo que ayuda a prevenir el plagio.

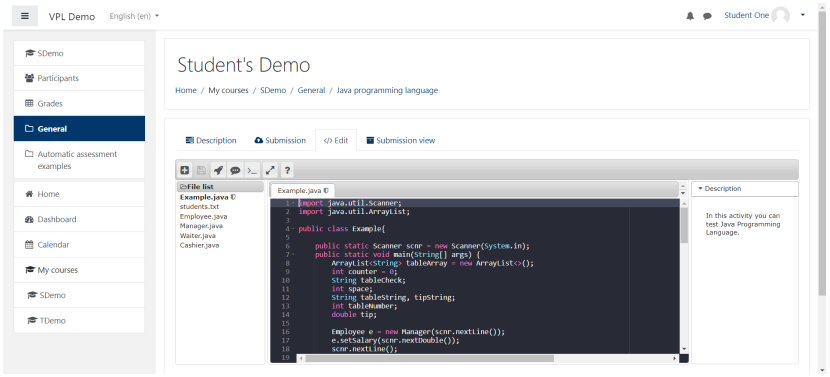

Figura 1.10: Vista de editor de código de *Virtual Programming Lab*

Respecto a su interfaz, visible en la figura 1.10, podemos ver que dispone de un menú en el lado izquierdo donde se puede navegar a las distintas tareas, ver los participantes de la asignatura o consultar las notas del usuario, entre otras opciones. También cabe destacar el desplegable para cambiar el idioma de la página en la parte superior, que aumenta su accesibilidad. En la figura 1.10 tenemos la vista del editor de código de una tarea, que podemos ver que es muy completo y nos permite crear varios archivos, además de ejecutarlos y editarlos. La interfaz de la tarea es clara, con una pestaña conteniendo su descripción, otra para realizar el envío de uno o más archivos y una última pestaña para ver la información del envío realizado, además del ya comentado editor de código.

A continuación, comentaremos el estado del arte de la aplicación *iudex*, analizando los distintos trabajos que se han realizado sobre la misma.

1.3. Estado del arte

Iudex es una plataforma de evaluación de código automática, con su nombre proveniente del latín *juez*, que consiste en una aplicación web de código libre con

[7] `https://vpl.dis.ulpgc.es/`

un propósito principalmente educativo, con el objetivo de ser usada en asignaturas de programación, bases de datos o algoritmos, entre otras, para que los alumnos puedan resolver tareas o incluso realizar pruebas prácticas. Está desarrollada usando Java, Typescript y HTML principalmente, y se puede acceder al repositorio que la contiene en el siguiente enlace `https://github.com/URJC-CP/iudex`. Ha sido desarrollada por varios alumnos de la Universidad Rey Juan Carlos, y, a pesar de encontrarse en una fase avanzada, aún necesita trabajo, especialmente su frontend.

A continuación se explicará en profundidad cada uno de los trabajos realizados previamente, de los cuales cuatro se han dedicado al backend y uno de ellos al frontend de la aplicación.

1.3.1. Backend

Como se ha comentado brevemente en el párrafo anterior, en esta sección se describirá el contenido de los cuatro trabajos dedicados al backend.

En el primero de ellos [15], se analizó la arquitectura que se iba a implementar, basada en servicios. Los servicios definidos fueron la interfaz web, que es la parte de la aplicación con la que interactúa el usuario, el juez, que se divide en una API y una base de datos y es la parte funcional y lógica de la aplicación, el orquestador de contenedores, que se encarga de la correcta ejecución de los códigos de la aplicación en distintas instancias, y RabbitMQ, que actúa como mediador de las comunicaciones entre los servicios del juez y el orquestador de contenedores. Además de realizar este diseño, se implementó el servicio de RabbitMQ y el orquestador de contenedores, que fue diseñado de manera que permitiera la ejecución de diferentes lenguajes.

Tras haber escogido la estructura de la aplicación, en el segundo trabajo [16] se implementó el juez previamente especificado, creando un sistema de clases y entidades para organizar y manipular la información relacionada con los distintos modelos, cuyo esquema facilitó también la construcción del correspondiente a la base de datos. También se implementó la API, que permitiría el fácil uso de la aplicación, y se automatizó su documentación mediante el uso de la librería *Swagger*, la cual permite realizar llamadas a la API desde una ruta del servidor web.

El tercero de estos trabajos [17] se centró en realizar mejoras a la aplicación ya existente, además de añadir algunas funcionalidades nuevas, como la adición de los lenguajes C, C++, MySQL o MariaDB a los lenguajes que el juez puede ejecutar, nuevas entradas en la API y la adición de un marcador para poder mostrar la clasificación de los concursos. Además, se mejoró la mantenibilidad y usabilidad de la aplicación, aumentando la calidad del código mediante el uso de herramientas como SonarCloud, entre otros.

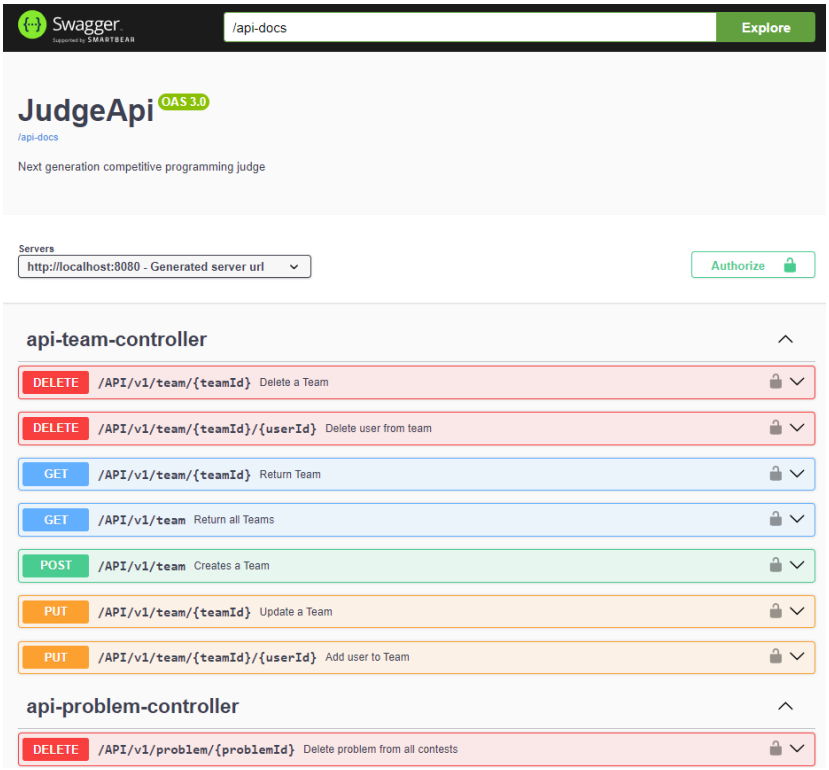

Figura 1.11: Captura de la documentación de la API generada con *Swagger*

Por último, el quinto trabajo sobre la aplicación [18], que se ha desarrollado paralelamente a este, también se centra en la mejora de la aplicación y la implementación de nuevas funcionalidades. La mejora de la calidad de la aplicación se ha realizado mediante eliminación de líneas duplicadas y la resolución de bugs y fallos, además de añadir y arreglar pruebas automáticas. También se han incluido WebSockets para reducir la carga de la aplicación al ser usada por varios usuarios simultáneamente, y se ha desarrollado un sistema de autenticación delegada, que permite el inicio de sesión institucional y que será incluido en este trabajo.

1.3.2. Estado de la interfaz

Respecto a la interfaz de *iudex*, fue desarrollada en el cuarto trabajo [19] que se realizó sobre esta aplicación. En este trabajo se pretendió implementar una interfaz fácil de usar, que permitiera utilizar las funciones ya implementadas en el backend. Algunos de los requisitos planteados incluían la internacionalización de los textos, el control de roles y el respeto de los principios de usabilidad, además de requisitos específicos para las vistas de cada rol. Esta interfaz fue desarrollada usando el *framework* Angular, con lenguajes como Typescript y HTML, y con la utilización de la librería Nebular. A continuación analizaremos algunos aspectos de la interfaz que se desarrolló en este trabajo, además de las pantallas diseñadas.

- Internacionalización: se desarrolló utilizando el *framework i18n*, usando el paquete *Localize* de Angular, que requiere que se reinicie la aplicación para cambiar el idioma, lo que no es ideal, ya que sería preferible tener la posibilidad de cambiarlo dinámicamente.

- Conexión backend: al inicio del desarrollo no se contaba con una versión final de la API real, por lo que se desarrolló una API *mock*, la cual se intentó sustituir por la API real al finalizar el trabajo mediante la implementación de los servicios en el frontend, pero no se consiguió.

- Gestión de roles: se definieron dos tipos de usuarios, alumnos y profesores, con distintos permisos. Además de estos se definieron vistas y recursos públicos que se podrían visualizar a pesar de no poseer ningún rol.

- Navegación web: en algunos casos, podemos encontrar botones en la interfaz que indican claramente su función y la pantalla a la que redirigen, sin embargo, se echa en falta algún componente que esté visible en todo momento y facilite al usuario moverse por la página, ya que en varias ocasiones la única forma de salir de una vista es seleccionando la opción de retroceso del propio navegador. Esto provoca otros problemas en la navegación, ya que dificulta ir directamente a la página deseada.

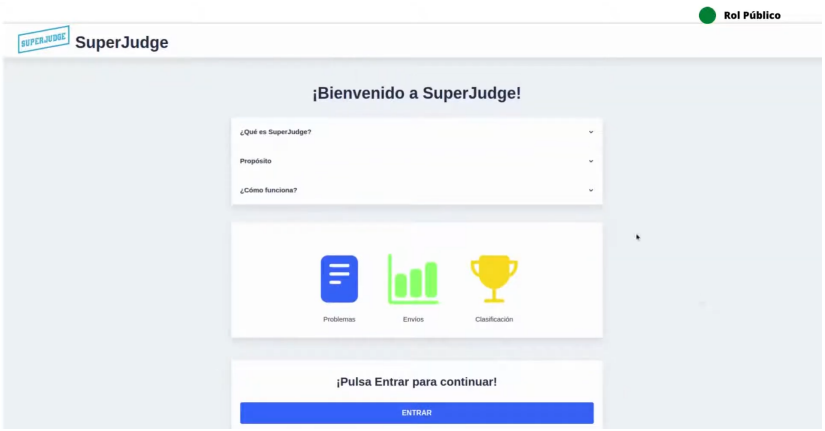

Figura 1.12: Pantalla inicial de la anterior interfaz de *iudex*

- Vista pública: sin necesidad de iniciar sesión, los usuarios pueden acceder al concurso público y ver los problemas, envíos y clasificación de este. En la pantalla inicial mostrada en la figura 1.12 podemos ver menús desplegables con información sobre la aplicación, botones para acceder a los recursos previamente mencionados, y un botón para iniciar sesión. Al seleccionar estos recursos, podemos ver una lista de problemas o envíos, con una opción para filtrar por texto las columnas. Si visualizamos los detalles de uno de los problemas o envíos, se muestran menús desplegables con cada campo de información del elemento, además del PDF con el enunciado en el caso de

los problemas, y un gráfico sobre los casos de prueba superados y el código enviado en el caso de los envíos. En el caso de la clasificación, encontramos una tabla que muestra el estado del concurso público, con celdas codificadas por colores según si un problema se ha resuelto o no, aunque carece de una leyenda de colores para informar al usuario.

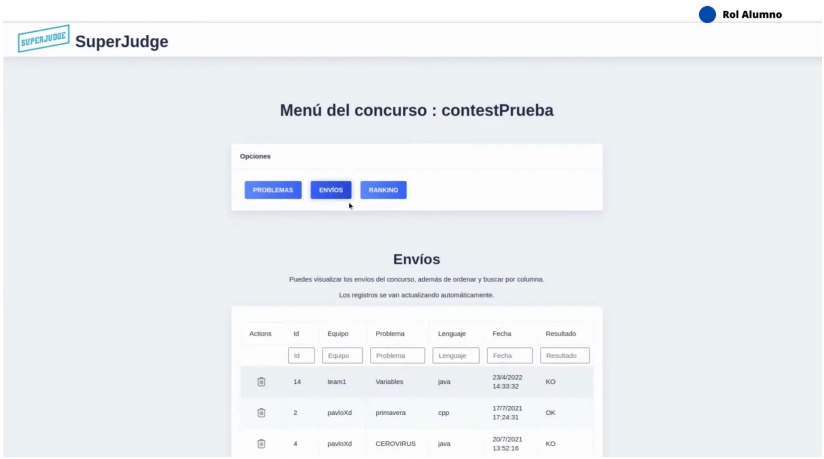

Figura 1.13: Vista de listado de envíos de un concurso con el rol de alumno en la anterior interfaz de *iudex*

- Vista de alumnos: al iniciar sesión con un perfil de alumno, se muestra una pantalla con la lista de concursos disponibles. Al seleccionar uno de ellos se muestran una serie de opciones en la parte superior de la pantalla, para mostrar los problemas, envíos o clasificación del concurso. Al seleccionar uno de estos recursos, se muestra un listado en formato de tabla, al igual que en la interfaz pública, que se puede visualizar en la figura 1.13. Las interfaces de detalles de problemas también es igual a la interfaz pública, con la diferencia de un botón para realizar un envío. En el caso de los envíos, no se pueden visualizar sus detalles. Para realizar un envío, se dispone de una interfaz en la que seleccionar el lenguaje deseado y subir el código a enviar, además de disponer de un editor de código.

- Vista de profesores: al iniciar sesión con un perfil de profesor, las pantallas son iguales a las de la vista de alumno, pero con opciones adicionales. En el caso de la lista de concursos, se da la opción de borrar, editar y añadir un concurso. En la vista mostrada al seleccionar un concurso, dispondremos de las opciones de crear un problema o añadir un problema al concurso, además de las que se encontraban disponibles en la vista de alumnos. En la pantalla de detalles de un problema, que podemos ver en la figura 1.14, dispondremos de las opciones de añadir, editar o borrar casos de prueba, además de las opciones de borrar o editar el problema. También se dispone de la opción de eliminar envíos en la vista en la que se muestran todos los envíos de ese concurso.

Figura 1.14: Vista de detalles de un problema con el rol de profesor de la anterior interfaz de *iudex*

Respecto a la usabilidad y accesibilidad de la interfaz web desarrollada, podemos concluir que presenta varias carencias, algunas intrínsecas al propio diseño y otras derivadas de aspectos más técnicos. Esto se debe a que por varios motivos, al desplegar la web, actualmente surgen muchos problemas como que no se puede cambiar de idioma, la superposición de algunas vistas, varios textos incompletos o que no se llegan a mostrar, al igual que los gráficos y el editor de código. Algunos problemas independientes de los provocados por aspectos técnicos son la falta de herramientas que ayuden a la navegación, dificultad de selección de elementos, desplegables innecesarios para mostrar información muy breve, y diseño poco intuitivo para las acciones exclusivas del rol de profesor, entre otros. Finalmente, el mayor inconveniente de esta interfaz es que utilizaba datos simulados, ya que no tenía una conexión real con la API. Esto además implicaba la imposibilidad de utilizar las funcionalidades del backend, como realizar envíos, iniciar sesión o modificar recursos, entre otras. Algunos de estos problemas se comentarán con más detalle durante la explicación de las decisiones de diseño tomadas para el nuevo diseño de la interfaz.

1.4. Estructura del documento

Tras la introducción realizada en este capítulo 1, en este Trabajo de Fin de Grado se explicará el diseño de la nueva interfaz de *iudex*, además de su implementación. En el capítulo 2 se enumerarán los objetivos que han motivado la realización de este trabajo, y se comentará la metodología utilizada para su desarrollo. A continuación, en el capítulo 3 se profundizará en la descripción informática del trabajo realizado, incluyendo los requisitos para el mismo y las herramientas que han sido utilizadas. Posteriormente, en el capítulo 4 se expondrán los resultados

del trabajo llevado a cabo, además de validar el correcto funcionamiento del mismo. Por último, en el capítulo 5 se reflexionará sobre el trabajo realizado y se comentarán los posibles trabajos futuros propuestos.

2

Objetivos

En este capítulo se enumerarán los objetivos de este Trabajo de Fin de Grado, de los cuales se especificarán los objetivos principales del trabajo, así como los objetivos secundarios derivados de estos. Asimismo, se explicará la metodología utilizada para su desarrollo.

2.1. Objetivos principales

El objetivo principal de este proyecto es el diseño y la implementación de una nueva interfaz web para la aplicación *iudex*, con el propósito de aumentar la usabilidad y accesibilidad de la misma. Las mejoras realizadas tendrán el objetivo de obtener una interfaz más moderna, intuitiva y efectiva para los usuarios, consiguiendo la claridad de las páginas y la uniformidad del diseño de las distintas secciones.

Además de esto, se enfocará la aplicación a un ámbito principalmente educativo, con el objetivo de que pueda ser usada en las distintas asignaturas de programación, ya sea para la realización de ejercicios, prácticas o exámenes, permitiendo una evaluación más precisa y justa de las competencias de programación de los usuarios.

2.2. Objetivos secundarios

Derivado de los anteriores, se han establecido los siguientes objetivos secundarios.

- Estudiar en profundidad el *framework* Angular y PrimeNG, la biblioteca de componentes especialmente diseñada para su integración.
- Ampliar conocimientos de los lenguajes Typescript, HTML y CSS.
- Aprender a utilizar herramientas para la internacionalización dinámica de aplicaciones, como la biblioteca *ngx-translate*.
- Profundizar en la tecnología de contenedores *Docker*, así como de la elaboración de archivos *Dockerfile* y *Docker Compose*.
- Aumentar conocimientos sobre integración de backend y frontend con el uso de una API desarrollada con el *framework Spring Boot*.
- Aprender sobre el uso del estándar de *JWTs* para la autenticación de usuarios.
- Utilizar una metodología de desarrollo para realizar un proyecto, siguiendo una serie de requisitos tanto funcionales como no funcionales.
- Familiarizarse con el mantenimiento y la evolución de una aplicación web.
- Obtener conocimientos sobre el diseño de interfaces siguiendo los estándares de usabilidad y accesibilidad.
- Aprender a utilizar LaTeX y Overleaf para la redacción de documentos académicos de calidad.

2.3. Metodología

Para el desarrollo de este trabajo se ha tomado la decisión de seguir una metodología ágil. Este tipo de metodologías surgieron en la época de los noventa, con intención de reducir la probabilidad de fracaso de un proyecto software, intentando reducir la burocracia que suponía el uso de metodologías tradicionales. Este concepto se consolidó en 2001, con la creación del *Manifiesto por el desarrollo ágil de software* [20], en el que se acuerdan cuatro valores indicando las prioridades de este tipo de metodologías, priorizando los individuos e interacciones sobre los principios y herramientas, el software funcional sobre la documentación exhaustiva, la colaboración con el cliente sobre la negociación contractual y la respuesta

al cambio sobre ceñirse a un plan. También se estipulan doce principios del software ágil que guían la aplicación de estos valores, promoviendo la flexibilidad, la colaboración y la entrega continua de software de calidad.

Una de las metodologías ágiles más representativas es Scrum, que ha sido la seleccionada para el desarrollo de este proyecto. Esta metodología nos proporciona un enfoque que divide el proyecto en iteraciones llamadas *Sprint*, las cuales tienen asignadas una serie de reglas y tareas específicas que deberán realizarse. Antes de iniciar cada iteración se ha definido el plan de trabajo (*Sprint planning*), durante ellas se realizaron reuniones ocasionales para resolver los obstáculos surgidos (*Daily Scrum*), y al finalizar cada iteración se ha realizado una revisión del trabajo realizado (*Sprint Review* y *Sprint Retrospective*). Además, se definen los roles *Scrum Master*, *Product Owner* y *Development Team*, en este caso los dos primeros han sido asumidos por los tutores del trabajo, ya que adoptan los roles de coordinar al desarrollador y de maximizar el valor del producto desarrollado. En esta metodología también se definen varios artefactos, que sirven como referencia del progreso para el equipo, como el *Product Backlog*, que es una lista de los requerimientos del producto a desarrollar, la cual se ha realizado en este trabajo mediante la creación de *Issues* en GitHub [21]. Para mejorar la visualización de la misma y del progreso del desarrollo, se ha utilizado la herramienta *GitHub Projects* para visualizar estas tareas en un tablero Kanban. A continuación se detallarán los siete *Sprints* que han sido necesarios para el desarrollo de este proyecto.

- Sprint 1: Para comenzar este trabajo se analizó el estado de la anterior interfaz y de la API, además de realizar una investigación sobre el diseño usable y accesible de interfaces. Tras esto, se identificaron los requisitos de la nueva interfaz.

- Sprint 2: Durante este sprint se realizó el diseño de la nueva interfaz con la herramienta Figma, siguiendo los principios de usabilidad y accesibilidad investigados previamente e incluyendo las funcionalidades ya existentes en la API.

- Sprint 3: Para comenzar con el desarrollo de la interfaz, se decidió crear un nuevo proyecto de Angular, se programaron los servicios y modelos necesarios, se realizó la conexión con el backend mediante un *proxy*, se crearon los ficheros para la internacionalización y se creó el componente correspondiente a la pantalla inicial de la aplicación.

- Sprint 4: Este sprint debería haber correspondido a la implementación de la autenticación, pero esto se pospuso debido a problemas con el backend, por lo que se eliminó temporalmente el control de roles de la API para poder comenzar a desarrollar las vistas del rol de estudiante. Se implementó la pantalla con el listado de concursos del usuario, la pantalla inicial de un

concurso, la pantalla con el listado de problemas de un concurso y la pantalla con la clasificación.

- Sprint 5: Una vez fueron solucionados los problemas con el backend, se eliminó el *proxy* y se implementó el inicio de sesión delegado. Posteriormente, se modificó el *Dockerfile* de la aplicación para que el frontend se compile al levantar el contenedor.
- Sprint 6: Tras implementar la autenticación, se desarrollaron las pantallas para los usuarios con rol de juez y administrador, las cuales son los listados de concursos, problemas, envíos, usuarios y resultados, además de la pantalla de clasificación.
- Sprint 7: Durante este sprint se aumentó la accesibilidad de la aplicación, cambiando el modo de realizar la internacionalización para poder realizar traducciones dinámicamente, y se implementó la posibilidad de cambiar entre tema claro y oscuro, obteniendo así el producto final.

3

Descripción informática

En este capítulo se describirá en profundidad el desarrollo inicial del trabajo, describiendo el análisis de requisitos realizado inicialmente en la sección 3.1, las herramientas y tecnologías utilizadas en la sección 3.2 y finalmente, en la sección 3.3, el diseño del prototipo de la aplicación.

3.1. Análisis de requisitos

Tras estudiar el estado de la aplicación en la sección 1.3, para especificar qué se buscaba en el producto a desarrollar, se realizó un análisis de requisitos tanto funcionales como no funcionales, que se presentan a continuación.

3.1.1. Requisitos funcionales

- Los usuarios no registrados podrán acceder a una pantalla inicial con información sobre la aplicación y un botón para iniciar sesión.
- Los usuarios podrán iniciar sesión utilizando sus credenciales de la Universidad Rey Juan Carlos, con un sistema de autenticación delegada.
- Todos los usuarios tendrán en todo momento la opción de cambiar el idioma de la interfaz, además de la opción de cambiar el estilo de la interfaz entre tema claro y oscuro.

- Respecto a la gestión de roles, existirá una jerarquía con los roles alumno, juez y administrador, donde los alumnos solo podrán hacer las acciones correspondientes a su rol, los jueces podrán realizar las acciones correspondientes a su rol y al de alumnos, y los administradores podrán realizar las acciones propias de su rol y de cualquier otro usuario.

- Requisitos para el rol de alumno:

  - Los alumnos podrán visualizar sus estadísticas personales y el listado de concursos disponibles para cada una de sus asignaturas, además de los listados de problemas y envíos de cada concurso, y su clasificación.
  - Los alumnos podrán visualizar el archivo PDF de cada problema, además de copiar la entrada y salida de sus casos de prueba.
  - Los alumnos podrán realizar envíos mediante la subida de un archivo o usando el editor de código incorporado en la interfaz.

- Requisitos para el rol de juez:

  - Los jueces podrán visualizar los listados de concursos, problemas y envíos, además de la clasificación de todos los concursos.
  - Los jueces podrán borrar concursos, problemas y envíos. Además, podrán añadir/editar concursos y problemas.

- Requisitos para el rol de administrador:

  - Los administradores podrán visualizar los listados de usuarios y resultados.

- La aplicación dispondrá de una barra de navegación visible en todo momento que permitirá a los usuarios moverse entre pantallas, incluyendo un botón para cerrar sesión y la opción de cambiar de rol en caso de que el usuario tenga el rol de juez o administrador.

- En la barra de navegación de las pantallas dentro de un concurso, el alumno dispondrá de un botón para realizar envíos y una cuenta atrás del tiempo restante del concurso.

- La aplicación dispondrá de notificaciones emergentes para informar al usuario o notificar errores.

- La aplicación deberá contener apartados explicativos donde se considere necesario, para explicar leyendas de colores o siglas, entre otros.

3.1.2. Requisitos no funcionales

- La aplicación será desarrollada utilizando el *framework* Angular, acompañado de tecnologías como HTML, CSS y Typescript.
- Se utilizará la biblioteca PrimeNG para el desarrollo.
- La aplicación seguirá la guía de identidad visual corporativa de la Universidad Rey Juan Carlos.
- El sistema podrá ser desplegado utilizando únicamente contenedores *Docker*, compilando el código del frontend en los mismos.
- La aplicación debe cumplir con los principios de usabilidad y accesibilidad, asegurando que sea fácil de usar independientemente del tipo de usuario.
- La aplicación debe estar orientada a ser utilizada en el ámbito académico.

3.2. Herramientas y tecnologías utilizadas

En este apartado se desarrollan las distintas herramientas y tecnologías que se han requerido para el desarrollo de este proyecto, incluyendo tanto las tecnológicas como las organizativas, las cuales se comentarán a continuación. Los IDEs utilizados han sido IntelliJ IDEA y Visual Studio Code.

3.2.1. GitHub

GitHub [22] es una plataforma de código libre basada en la nube para almacenar y compartir código, que también permite desarrollarlo junto a otros usuarios. El trabajo colaborativo es su principal función, el cual es posible gracias a la herramienta de código abierto Git, sobre la que está construida esta plataforma. Git es un sistema de control de versiones que detecta automáticamente los cambios realizados en un archivo, por lo que es una herramienta muy útil para el desarrollo de código en equipo y para mantener un seguimiento del mismo.

En este trabajo se ha utilizado GitHub para almacenar el código del proyecto, subiéndolo a medida que se iba desarrollando, y permitiendo que los tutores verificaran que los cambios realizados eran correctos al finalizar cada parte. También se han aprovechado sus funcionalidades relacionadas con la organización, permitiendo crear algunos artefactos de la metodología usada, como el *Product Backlog* y el tablero Kanban mencionados previamente en la sección 2.3, para los que se ha utilizado *GitHub Projects*.

3.2.2. Figma

Figma[1] es una herramienta de diseño y generación de prototipos colaborativa, principalmente basada en la web. Está enfocada al diseño de pantallas de aplicaciones o páginas web, permitiendo incluso la incorporación de elementos interactivos en los prototipos desarrollados.

Esta plataforma ha sido utilizada para realizar el diseño inicial de todas las interfaces de *iudex* y un prototipo que simulaba la navegación, los cuales posteriormente servirían de referencia para su desarrollo.

3.2.3. Angular

Angular [23] es un *framework* de desarrollo web de código abierto basado en Typescript, un lenguaje de programación que extiende Javascript. Fue creado por Google y se utiliza para la construcción de aplicaciones SPA. Las aplicaciones desarrolladas usando Angular se organizan en componentes en los que se combinan HMTL, CSS y lógica en Typescript, los cuales son reutilizables y modulares. También dispone de bibliotecas integradas que cubren numerosas funcionalidades, de las cuales se han utilizado *Router* para el enrutamiento, *HttpClient* para la comunicación entre servidor y cliente, y *ngx-translate* para las traducciones dinámicas, entre otras. Angular ha sido utilizado para realizar todo el frontend de este proyecto.

3.2.4. PrimeNG y PrimeFlex

Para facilitar la implementación del diseño de la interfaz, se decidió usar PrimeNG[2], una biblioteca de componentes de interfaz preconstruidos y personalizables para Angular. Esto se ha complementado con el uso de PrimeFlex[3], que es un *framework* de utilidades CSS diseñado para trabajar con PrimeNG, que permite crear un diseño responsivo y gestionar la disposición de los componentes.

Se ha decidido utilizar esta biblioteca por su gran variedad de componentes altamente personalizables, además de por su editor de temas, que permite crear el CSS de la aplicación de una forma sencilla y visual, por lo que se ha utilizado para crear los temas claro y oscuro de la interfaz. Otro ámbito a tener en cuenta es que los componentes de PrimeNG cumplen con las normativas de accesibilidad, lo que ayuda a que la interfaz desarrollada sea accesible.

[1] `https://www.figma.com`
[2] `https://primeng.org/`
[3] `https://primeflex.org/`

3.2.5. Spring

Spring [24] es un *framework* de desarrollo de aplicaciones de código abierto, que provee una infraestructura para construir aplicaciones basadas en Java. El backend de la aplicación *iudex* está desarrollado haciendo uso de este *framework*, ya que dispone de muchas herramientas que lo convierten en uno de los más completos para desarrollo web.

En el caso de este trabajo, Spring se ha usado principalmente para el controlador SPA, que permite la redirección correcta de las rutas de la aplicación. También se ha usado para modificar la gestión de roles, realizar modificaciones relacionadas con la autenticación o pequeños arreglos en los puntos de servicio de la API.

3.2.6. Docker

Docker [25] es una plataforma de código abierto que permite desarrollar, enviar y ejecutar aplicaciones dentro de contenedores, empaquetando el código de una aplicación y todas sus dependencias. La aplicación *iudex* ya utilizaba la tecnología de contenedores de Docker, y contaba con dos archivos de configuración *Docker Compose* que permitían levantar la aplicación con un solo comando. Durante este trabajo se han modificado estos archivos para permitir que el frontend desarrollado pueda ser compilado en el mismo contenedor, ya que el anterior estaba integrado de forma estática previamente compilado, dificultando la realización de cambios.

3.2.7. LaTeX

LaTeX [26] es un sistema de preparación de documentos usado principalmente para la elaboración de textos científicos y técnicos, el cual ha sido utilizado para la redacción de la memoria de este trabajo para aumentar su calidad y profesionalidad. Esto ha sido realizado mediante la utilización de la aplicación web Overleaf, que permite escribir y editar documentos en línea de forma colaborativa, facilitando así las revisiones por parte de los tutores.

3.3. Diseño

En este apartado se analizará el diseño inicial de la interfaz de *iudex*, el cual se elaboró mediante la realización de un prototipo en Figma que se puede encontrar en el siguiente enlace `https://www.figma.com/design/rF1HQlC9EhMWcNFCjbx411/`

IUDEX?node-id=0-1&t=O8oeoiZLVr7lixZv-1.

Antes de comenzar el diseño se realizó un análisis de los puntos de servicio de la API y los roles que podían utilizarlos, para saber qué acciones se deberían incluir en cada vista. Respecto a la estética de la interfaz, se tomó la decisión de seguir el Manual de Identidad de la Universidad Rey Juan Carlos [27], puesto que el propósito de esta aplicación es ser utilizada principalmente en diferentes asignaturas de esta Universidad. Además, siguiendo los colores y fuentes de este documento, aseguramos que serán accesible. También se utilizó el logo de la Universidad como referencia para diseñar un nuevo logo para *iudex*, que se puede visualizar en la parte superior izquierda de todas las vistas.

El diseño ha sido realizado en modo oscuro, aunque la aplicación real también dispondrá de una opción para visualizarla en modo claro, debido a que las interfaces con fondo oscuro tienen numerosas ventajas, como por ejemplo: la reducción del cansancio ocular, mayor comodidad de visualización en espacios con poca luz, mejor legibilidad del contenido y reducción de distracciones. Nótese que, la interfaz con fondo oscuro es mayoritariamente preferido por los programadores, que es el público al que se enfoca esta aplicación web [28]. También se han tomado en consideración las *Directrices de Accesibilidad para el Contenido Web* (WCAG) [29] y los *Principios de usabilidad web* de Jakob Nielsen [30] para hacer que la interfaz sea accesible y usable, los cuales serán abordados con más profundidad más adelante. A continuación, se comentarán las interfaces diseñadas, en las que se ha cubierto la totalidad de las funcionalidades implementadas en la API, a pesar de que algunas de ellas no serán implementadas por estar fuera del alcance de este trabajo.

Figura 3.1: Pantalla inicial en el prototipo de Figma de *iudex*

- Pantalla inicial: para el diseño de esta vista, se ha usado como referencia la pantalla inicial de la interfaz anterior, eliminando las opciones de visualizar problemas, envíos y clasificación públicos, puesto que se ha eliminado esta

función para los usuarios no registrados. Además, como se puede visualizar en la figura 3.1 se ha situado el botón para iniciar sesión en el medio de la pantalla para centrar la atención en él, ya que a menos que sea el primer acceso del usuario, su principal objetivo será iniciar sesión.

- Pantalla de inicio de sesión: únicamente aparece en el prototipo como referencia, no será necesario implementar una vista de inicio de sesión, ya que se iniciará sesión desde la página de la Universidad Rey Juan Carlos.

- Pantalla de inicio de alumnos: al iniciar sesión como usuario con el rol de alumno, se redirigirá a esta pantalla en la que se pueden visualizar algunas estadísticas del usuario y el listado de concursos de cada asignatura en formato de tarjetas, con un botón llamativo para acceder a ellos. Respecto a la anterior interfaz, esta no disponía de las estadísticas del alumno, además de que presentaba el listado de concursos en un formato de tabla que dificultaba la diferenciación entre concursos y no indicaba de forma clara cómo acceder a ellos. También se dispondrá de un menú desplegable en la parte derecha de la barra de navegación donde se encontrará la opción de cerrar sesión o, en caso de tener un rol de juez o administrador, adicionalmente se encontrará la opción de cambiar a la vista del rol deseado.

Figura 3.2: Pantalla inicial del rol alumno en el prototipo de Figma de *iudex*

- Pantallas de concurso: para las pantallas de los concursos se ha tomado inspiración de *DOMjudge*, ya que es un sistema con el que los alumnos están familiarizados y que permite que se enfoquen en el concurso. En la barra de navegación de estas pantallas están las opciones, inicio, problemas y clasificación que redirigirán a la pantalla correspondiente, destacando en la que se encuentra el usuario. Además, se dispone de una cuenta atrás en el centro y a la derecha encontraremos el desplegable mencionado en el punto anterior y el nombre del concurso, que al ser seleccionado redirigirá al usuario a la vista inicial con el listado de concursos. Adicionalmente,

en todo momento estará visible en la barra de navegación un botón para realizar envíos, permitiendo que los alumnos no pierdan tiempo buscándolo.

- Pantalla inicial del concurso: en esta pantalla, mostrada en la figura 3.3, encontramos en la parte superior una tabla de clasificación en la que solo está la información del usuario, permitiendo que visualice fácilmente qué problemas ha solucionado, fallado o aún no ha intentado, además de su posición en la clasificación global. Bajo esta tabla, en el lado izquierdo se visualiza una tabla con los envíos del usuario con su resultado, facilitando que vea el motivo por el que ha fallado su código o si ha sido aceptado. Por último, a la derecha de esta tabla dispone de un editor de código en el que podrá escribir su solución a los problemas, elegir el lenguaje y compilarla o enviarla.

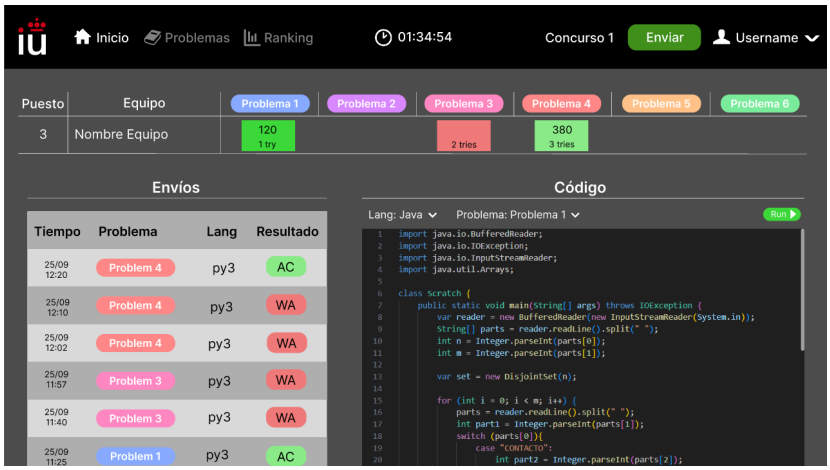

Figura 3.3: Pantalla inicial del concurso en el prototipo de Figma de *iudex*

- Pantalla de clasificación del concurso: en esta pantalla, correspondiente a la figura 3.4, se ha mantenido el formato de tabla con celdas de colores de la anterior interfaz, cambiando únicamente el estilo de la misma para que sea más minimalista y profesional.

- Pantalla de problemas del concurso: esta pantalla también ha sido inspirada por *DOMjudge*, ya que se ha buscado mejorar la vista de tabla de la antigua interfaz, permitiendo diferenciar los problemas fácilmente y acceder más rápido a las opciones de abrir el PDF del problema, ver los casos de prueba y realizar un envío. Para esto se ha utilizado un formato de tarjetas, con los botones de acciones en distintos colores, para que sean fácilmente diferenciables.

- Pantallas de juez y administrador: en las pantallas del rol de juez, podemos ver también una barra de navegación con las opciones concursos, problemas, equipos, envíos, clasificación y *rejudge*, aunque las pantallas de equipos

y *rejudge* no se implementarán por el momento, ya que los equipos no serán necesarios en el contexto educativo y la funcionalidad de *rejudge* aún no está implementada. En las pantallas de administrador se dispondrán de las opciones, usuarios y resultados. Además, también se encontrará un desplegable en el lado derecho para cerrar sesión o cambiar de vista de rol, al igual que en las pantallas de alumnos.

- Pantalla de clasificación de jueces: respecto a la pantalla de clasificación en la vista correspondiente al rol de juez, es igual a la que se puede visualizar en la pantalla de alumnos, pero con la opción de seleccionar el concurso del que se quiere consultar.

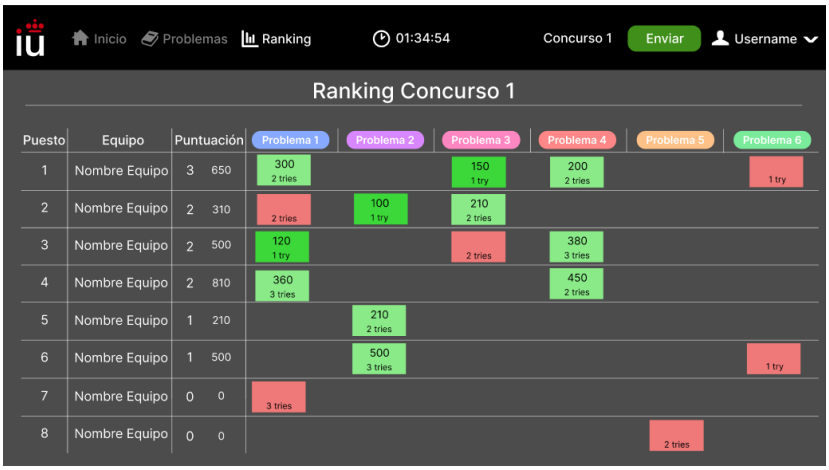

Figura 3.4: Pantalla de clasificación del concurso en el prototipo de Figma de *iudex*

- Tablas de información de rol juez y administrador: en las distintas pantallas correspondientes a los elementos que se pueden seleccionar en la barra de navegación, los usuarios con roles de juez o administrador podrán visualizar un listado de los recursos seleccionados, con la opción de filtrarlos y acceder a ellos. Un ejemplo de esta interfaz se encuentra en la figura 3.5.

Figura 3.5: Pantalla de tabla de resultados del rol administrador en el prototipo de Figma de *iudex*

- Pantalla de detalles de rol juez y administrador: en las pantallas de detalle de un recurso seleccionado, se ha cambiado el formato de la anterior interfaz, que usaba menús desplegables para cada campo de información de forma innecesaria. En este nuevo diseño, ejemplificado en la figura 3.6, se mostrará el nombre del campo en el lado izquierdo de una tabla y a la derecha la información correspondiente al mismo, permitiendo una consulta más rápida y sencilla. Además de esto, se encontrará la opción de editar el recurso o eliminarlo, en caso de que sea posible.

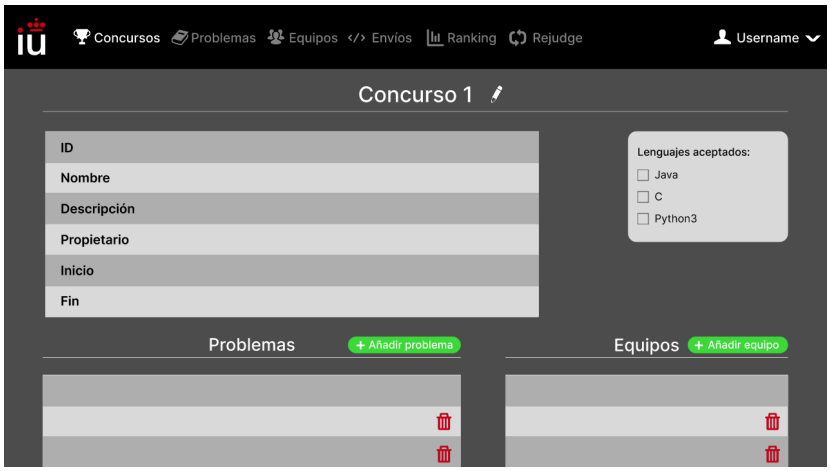

Figura 3.6: Pantalla de detalles de problema del rol juez en el prototipo de Figma de *iudex*

A continuación se hablará sobre el proceso de implementación de este diseño para que sea una interfaz web real.

4

Solución tecnológica y resultados

Antes de comenzar con el desarrollo de la solución, se tuvo que tomar la decisión de conservar el frontend anterior o reimplementar y crear un nuevo proyecto. Tras evaluar el estado del mismo, se decidió reimplementar debido a que su mantenimiento sería más costoso que empezar de cero, ya que el nuevo diseño es bastante distinto al anterior, se decidió usar una biblioteca de componentes distinta, más completa y flexible, además de que la versión de Angular era muy antigua y la lógica no funcionaba bien con la API real. Por ende, se creó un proyecto nuevo con una versión más actualizada de Angular, específicamente la 16.2, la cual no se ha llegado a actualizar más debido a las restricciones de PrimeNG, ya que en estos momentos la última versión de Angular para la que proporciona soporte a largo plazo es la 16.

En este capítulo se describirá la implementación de la nueva interfaz, destacando sus aspectos clave. Adicionalmente, se analizará la accesibilidad y usabilidad de la aplicación final en la sección 4.7, y finalmente se adjuntará un vídeo de la interfaz a modo de demostración de resultados.

4.1. Servicios, modelos y conexión con backend

Tras crear el nuevo proyecto de Angular, se comenzó el desarrollo creando los servicios y modelos necesarios para la comunicación con la API, ya que se podían tomar como referencia algunos del anterior frontend. En los servicios desarrollados también se incluyeron llamadas a los puntos de servicio correspondientes

a las nuevas funcionalidades implementadas en trabajos anteriores a este, para poder ser utilizados posteriormente desde los componentes. Estos servicios han sido implementados con TypeScript utilizando la biblioteca integrada de Angular *HttpClient*, que permite realizar peticiones HTTP, en este caso a la API de *iudex*. La respuesta de estas peticiones se almacena en los modelos desarrollados, llamados DTOs, lo cual no se hacía en el código de la anterior interfaz, por lo que en este nuevo código se dispone de una mayor seguridad de tipos, asegurando que los datos recibidos son del tipo esperado.

Para conseguir conectar los servicios con la API se utilizó inicialmente un *proxy*, puesto que el frontend y el backend se ejecutaban en distintos puertos y eso impedía su comunicación. Este *proxy* permitió que la conexión funcionara utilizando rutas relativas sin necesidad de establecer directamente en el código la ruta con el puerto del backend, solucionando otro problema del anterior frontend. Posteriormente, cuando se empezó a implementar el inicio de sesión delegado, apareció la necesidad de que ambas partes de la aplicación se ejecutaran en el mismo puerto, y se tomó una primera aproximación de integrar de forma estática el frontend previamente compilado, que era la que se había tomado también en el anterior. Esto fue únicamente temporal, ya que disminuía el rendimiento de la aplicación y dificultaba su desarrollo, puesto que cualquier cambio en el frontend requería volver a compilarlo e integrarlo, lo cual era un proceso excesivamente lento. Para conseguir un método más óptimo de compilación que permitiera ejecutar toda la aplicación en el mismo puerto, se crearon nuevos archivos *Dockerfile* y *Docker Compose* que compilaran ambas partes en el contenedor, facilitando el despliegue y permitiendo un desarrollo más rápido. Además de esto, para conseguir que se redirigieran correctamente las rutas del frontend, se creó una clase SPA Controller, ya que no existía previamente, permitiendo también recargar la página en cualquier ruta correspondiente a la interfaz sin recibir un mensaje de error.

4.2. Navegación

Respecto a la navegación por la interfaz, se ha utilizado la biblioteca integrada de Angular *Router*, que permite que la aplicación tenga una navegación fluida sin recarga, para la que se han utilizado nombres de rutas bien definidos y explicativos. Al igual que en el diseño de la interfaz, el usuario será redirigido principalmente gracias a la barra de navegación, permitiendo también cambiar de vista a la correspondiente a otro rol en caso de que esté autorizado. A continuación, en la figura 4.1 vemos el diagrama de navegación definitivo, indicando las pantallas correspondientes a cada rol, además de la jerarquía de los mismos. Adicionalmente, desde cualquiera de las pantallas representadas, a excepción de las ilustradas en blanco, se permite cambiar a la vista de otro rol en caso de que sea juez o administrador, lo cual redirigiría a la pantalla inicial del rol seleccionada,

marcada por el icono .

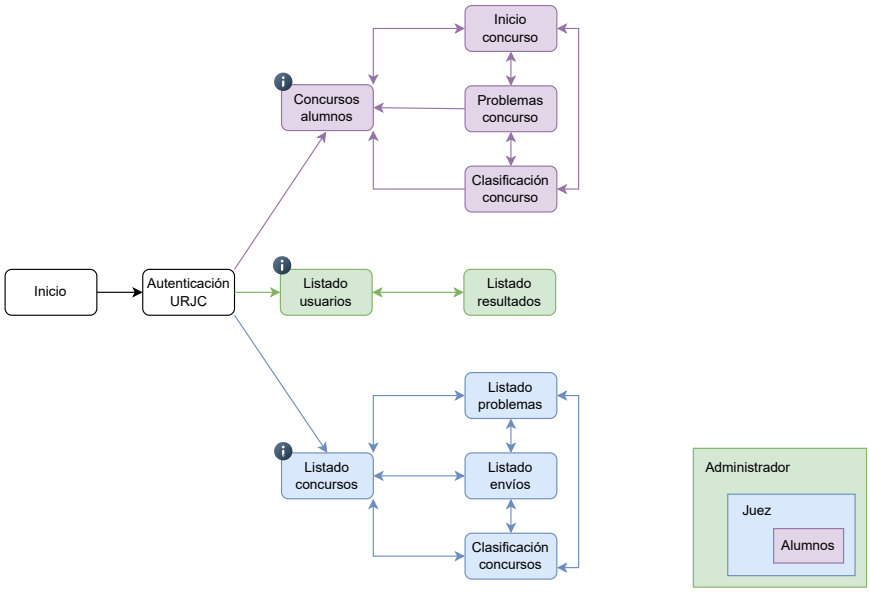

Figura 4.1: Diagrama de navegación de la interfaz de *iudex*

4.3. Estilos

Como se ha comentado previamente en la sección 3.2.4, se ha utilizado PrimeNG como biblioteca de componentes y PrimeFlex para complementarla. Para crear el estilo de los componentes se utilizó la herramienta *Designer* de PrimeNG, que permite modificar un tema ya existente, en este caso Soho Light y Soho Dark, que han sido modificados para utilizar los colores usados en el prototipo de la interfaz. La herramienta mencionada permitió obtener dos archivos CSS correspondientes al modo claro y oscuro de la interfaz, los cuales se pueden cambiar fácilmente seleccionando un botón situado en el pie de página de la aplicación, el cual llama al servicio implementado *ThemeService*, que se encarga de cambiar el archivo CSS utilizado.

4.4. Componentes

A continuación se analizarán los diferentes componentes que han sido implementados para el desarrollo de la interfaz web de *iudex*.

- Barra de navegación (Navbar): el componente correspondiente a la barra de navegación se implementó inicialmente con el componente de PrimeNG

p-menubar, el cual era demasiado estricto respecto al posicionamiento de los elementos, por lo que posteriormente se cambió a *p-toolbar*. Los elementos que aparecen dentro de este componente varían en función del rol del usuario y la vista en la que se encuentra, siendo iguales a los especificados en el prototipo de la interfaz, a excepción del elemento "equipos", el cual se decidió no incluir, ya que no será usado, puesto que no habrá equipos en la aplicación. Esta barra de navegación es visible en todas las pantallas a excepción de la inicial, ya que no tiene ninguna función en esa vista.

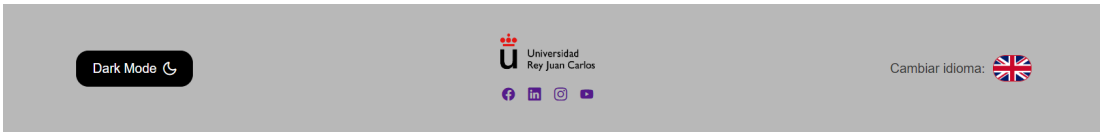

Figura 4.2: Pie de página de la nueva interfaz de *iudex*

- Pie de página (Footer): en el componente del pie de página, ilustrado en la figura 4.2, encontramos el logo de la Universidad Rey Juan Carlos junto a algunos enlaces de sus redes sociales, los botones de cambio entre temas oscuro y claro, y el botón para cambiar el idioma de la interfaz entre inglés y español. Este componente se encuentra visible en todas las pantallas de la aplicación para que el usuario pueda cambiar el tema o el idioma siempre que lo desee.

- Inicio (Home): este componente es prácticamente igual a la pantalla inicial del prototipo, a excepción de la barra de navegación, la cual se eliminó, puesto que, como se ha comentado anteriormente, no es necesaria. Se ha utilizado un botón para el inicio de sesión y el componente *p-dropdown* para los menús desplegables con información sobre la aplicación.

- Pantalla inicial de alumnos (StudentContests): en este componente se da la bienvenida al usuario y se muestra una tarjeta implementada con el componente *p-card* con estadísticas correspondientes a sus envíos totales, concursos en los que ha participado y envíos correctos. Bajo este elemento, vemos que los concursos están organizados por asignaturas, aunque por el momento no está implementado en el backend, por lo que vemos una única sección "Concursos asignatura 1" como referencia. En esta sección podemos ver los concursos del usuario en formato de tarjeta con el nombre del concurso, su inicio y final, y un botón para acceder al mismo. A diferencia del prototipo, el nombre del concurso no tiene color, ya que en el backend los concursos no tienen colores asignados.

- Pantalla inicial del concurso (StudentHome): este componente también se ha implementado siguiendo el diseño del prototipo, en la parte superior encontramos una tabla con la clasificación del usuario en el concurso, en el

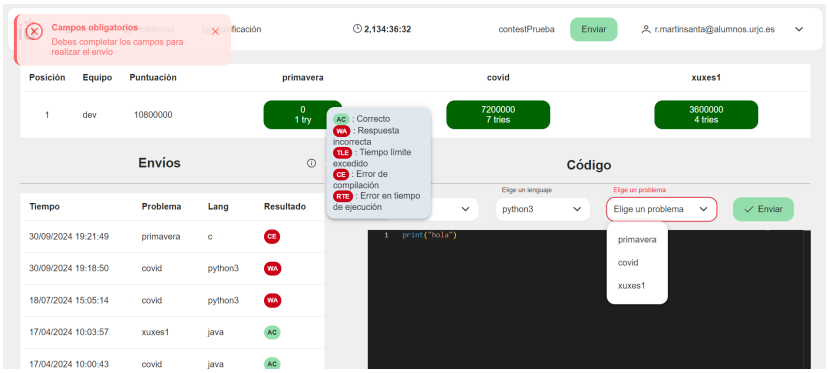

Figura 4.3: Pantalla inicial de concurso de la nueva interfaz de *iudex*

lado izquierdo una tabla con los envíos del usuario y en el derecho un editor de código. Se ha añadido información sobre el significado, los resultados de los envíos en la parte superior de la tabla correspondiente, haciendo uso del componente *p-tooltip*. Además, en el editor de código disponemos de la opción de cambiar el tema del mismo, con tres temas distintos a elegir. También podemos ver dos menús desplegables para seleccionar el lenguaje de programación y el problema al que corresponda el código desarrollado, además de un botón para realizar un envío con el mismo. Inicialmente, también se quería incluir la opción de compilar el código sin enviarlo, pero esta funcionalidad no está disponible en la API. Adicionalmente, se han incluido notificaciones para informar al usuario de si el envío se ha realizado correctamente o si hay algún campo incompleto que lo impida. Estas notificaciones se pueden visualizar en la figura 4.3, junto a la información de los resultados de los envíos.

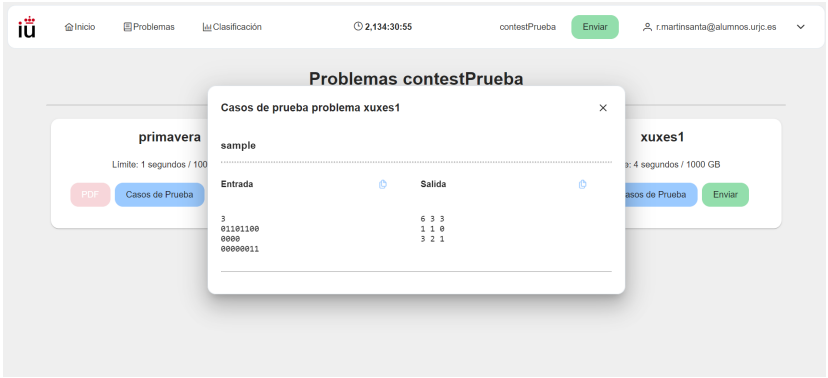

Figura 4.4: Pantalla de problemas del concurso de la nueva interfaz de *iudex* con modal de casos de prueba

- Pantalla de problemas del concurso (StudentProblems): en este componente se presentan los problemas del concurso en formato de tarjeta, con el nombre del mismo, sus límites tanto de tiempo como de memoria, los botones para

visualizar su PDF y casos de prueba, además de un botón para realizar un envío. Al igual que los concursos, los nombres de los problemas tampoco tienen color, ya que los colores asignados pueden presentar problemas en términos de accesibilidad por falta de contraste con el texto. Respecto al botón para visualizar el PDF del problema, se encarga de abrirlo en una nueva pestaña y solo está disponible en caso de que el problema tenga PDF, en caso contrario estará desactivado. El botón de casos de prueba abre un modal en el que se muestran estos, como se puede visualizar en la figura 4.4, con la adición de la opción de copiar la entrada o salida de cada uno al portapapeles, acompañada de una notificación de confirmación en caso de que se haya copiado correctamente. Por último, el botón de realizar un envío abre otro modal correspondiente al componente *SubmitButton* del cual se hablará a continuación, en la que estará preseleccionado el problema correspondiente.

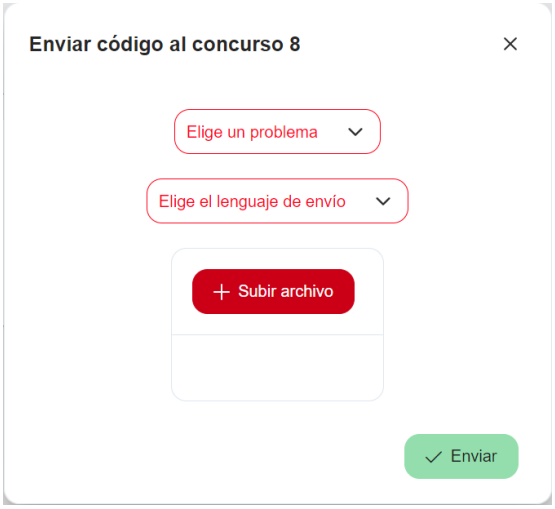

Figura 4.5: Botón de envío de la nueva interfaz de *iudex*

- Botón de envío (SubmitButton): se tomó la decisión de crear un componente únicamente para el modal de realización de un envío, puesto que se reutilizará en la barra de navegación y en cada uno de los problemas de la vista de problemas de cada concurso. Al igual que en el editor de código, disponemos de dos menús desplegables donde seleccionar el lenguaje usado y el problema al que se desea realizar el envío, junto a un botón para seleccionar el archivo de código, que adicionalmente dispone de un área a la que también podemos arrastrar y soltar archivos, permitiendo al usuario usar la opción que prefiera. También se han incluido notificaciones para informar al usuario si hay algún campo incompleto que le impida realizar el envío.

- Clasificación del concurso (StudentRanking): a pesar del nombre del componente, este ha acabado reutilizándose para la vista de clasificación de los

Figura 4.6: Vista de juez de la clasificación del concurso de la nueva interfaz de *iudex*

jueces, puesto que la estructura necesaria es la misma. Podemos ver en la parte superior de ambas pantallas el nombre del concurso, que en el caso de la pantalla de jueces es un desplegable con los diferentes concursos, junto a un icono implementado con el componente *p-tooltip* que contiene la leyenda de colores de las celdas de la tabla, como se puede visualizar en la figura 4.6. En la parte inferior podemos observar la tabla con los usuarios participantes en el concurso, sus posiciones, puntuación y el resultado que han obtenido en cada problema. Inicialmente, se quería añadir un resumen de resultados en la parte inferior de la tabla, ya que esto estaba implementado en la anterior interfaz, pero no se ha realizado debido a que esta información no está disponible en los puntos de servicio de la API.

Figura 4.7: Vista de resultados de administrador de la nueva interfaz de *iudex*

- Tablas para jueces y administradores (JudgeAdminTables): se ha utilizado un único componente para todas las tablas correspondientes a las pantallas de jueces y administradores, ya que tienen la misma estructura pero con distinto contenido. Este contenido varía en función de los parámetros de la ruta, que indican el tipo de recurso que se está consultando. En todas las tablas se dispone de la opción de ordenar cada una de las columnas en

orden ascendente o descendente, y en el caso de las pantallas de concursos, problemas y envíos también se dispone de la opción de borrar elementos. En la figura 4.7 podemos ver el listado de resultados de la vista de administrador filtrado por resultado, en este caso la tabla está incompleta, ya que la API no devuelve la información de envío y problema, pero se han dejado las columnas correspondientes como referencia.

4.5. Autenticación

La implementación de la autenticación delegada ha sido problemática debido a que inicialmente el punto de servicio de inicio de sesión redirigía al proveedor de identidades, pero no se devolvía el token necesario, ya que se encontraba en una página externa a la que no se podía acceder desde el frontend. Para solucionar esto, se modificó la implementación del punto de servicio de autenticación para que redirija a una página de la que se pueda obtener un token temporal, el cual posteriormente se intercambia por el token de sesión del usuario y el token de actualización en un nuevo punto de servicio. Una vez adquiridos los tokens, se redirigirá al usuario a la pantalla inicial correspondiente a su rol más alto en la jerarquía mencionada previamente.

Estos tokens son guardados en el almacenamiento local, ya que son necesarios para realizar peticiones HTTP, conocer el estado de la sesión del usuario y actualizarla en caso de que el token de acceso haya caducado. Esto se realiza gracias al servicio implementado en el frontend *AuthInterceptor*, el cual se encarga de interceptar todas las peticiones realizadas a la API para añadir el token de acceso en la cabecera. Además, en caso de que el token de acceso haya caducado, se encarga de utilizar el token de actualización para conseguir un nuevo token de acceso, y así permitir que el usuario no tenga que volver a iniciar sesión.

Adicionalmente, fue necesario crear un nuevo punto de servicio que devolviera la información del usuario actual, ya que no era posible recuperarla únicamente con el token. Por último, para el cierre de sesión, en caso de que el usuario seleccione el botón correspondiente en la barra de navegación, se borrarán los tokens de acceso y actualización del almacenamiento local y será redirigido a la pantalla inicial de la aplicación.

4.6. Internacionalización

Con el propósito de aumentar la accesibilidad y usabilidad de la página se ha implementado la opción de cambiar de idioma utilizando el *framework i18n*, lo que inicialmente se intentó conseguir con el uso del paquete *Localize* de Angular,

pero tras comprobar que la opción de cambiar el idioma dinámicamente utilizando este paquete no era compatible con la forma de desplegar la aplicación, se tomó la decisión de utilizar la biblioteca *ngx-translate*. Para la utilización de esta biblioteca se crearon dos archivos *json* con las traducciones en inglés y español, los cuales son los idiomas que serán soportados de momento. Para presentar esta opción al usuario se implementó un botón situado en el pie de página que se comunica con el servicio de Angular *Translate Service*, de forma que al seleccionar el botón se cambiará el idioma de la aplicación, sin necesidad de refrescar la página. Para añadir nuevos idiomas simplemente será necesario crear un archivo *json* con las traducciones para el nuevo idioma y añadir la opción correspondiente en el botón del pie de página.

4.7. Accesibilidad y usabilidad

Con el propósito de asegurar la accesibilidad de la interfaz desarrollada, se han utilizado como referencia las *Directrices de Accesibilidad para el Contenido Web* (WCAG) [29], que son un conjunto de normas internacionales cuyo objetivo es ofrecer un estándar que garantice que el contenido digital sea accesible para todas las personas. A continuación se describirán las decisiones que se han tomado e implementado con el fin de cumplir los principios de este estándar.

- Perceptible: con el objetivo de cumplir este principio, se ha añadido texto alternativo a las imágenes y se han utilizado iconos como apoyo al texto en vez de usarse como sustitución del mismo. Adicionalmente, se han seguido las recomendaciones respecto al tamaño del texto, siendo el normal de un tamaño de 16 píxeles y los grandes de un mínimo de 24 píxeles, lo que asegura su legibilidad y fácil diferenciación. Respecto al contraste de color, se estipula un contraste de 4.5:1 en el caso del texto normal, y un contraste de 3:1 en el caso del texto grande y componentes de interfaz de usuario, los cuales se han cumplido en ambos estilos de la interfaz, tanto claro como oscuro, validándolo gracias a la web `https://www.siegemedia.com/contrast-ratio`. La fuente usada para que el texto sea fácilmente legible es Inter, la cual es recomendada por el Manual de Identidad Corporativa de la Universidad Rey Juan Carlos [27], además de ser una tipografía diseñada específicamente para mejorar la legibilidad en interfaces digitales. Por último, se han incluido etiquetas y descripciones claras en los formularios, botones y demás elementos de la interfaz.

- Operable: uno de los aspectos de la interfaz que cumple este principio es la navegación, la cual es fácil y clara, como se ha visto en la sección 4.2, utilizando enlaces en la cabecera fácilmente accesibles como punto de referencia. Adicionalmente, se permite la navegación utilizando únicamente

el teclado, garantizando el acceso a personas con discapacidades motoras o que usen dispositivos de entrada alternativos. Además, se proporciona al usuario tiempo suficiente para interactuar con el contenido o leerlo, y se ha incluido manejo de errores en todos los formularios, utilizando notificaciones para informar en caso de que algún campo esté incompleto. Finalmente, se ha asegurado que los elementos interactivos como los botones tengan un tamaño adecuado, siendo siempre igual o mayor al tamaño mínimo recomendado por las WCAG de 44x44 píxeles.

- Comprensible: para cumplir este principio, se ha utilizado un lenguaje claro y simple, con explicaciones para términos más técnicos, como los resultados de los envíos, e información sobre el funcionamiento y propósito de la aplicación en la pantalla de inicio. Adicionalmente, se proveen indicaciones claras para que el usuario sepa cómo rellenar los formularios, además del control de errores mencionado en el punto anterior. También se provee consistencia en los elementos de navegación, entre otros elementos, ya que siempre se encuentran en el mismo lugar. Finalmente, la internacionalización de la página permite al usuario identificar o cambiar el idioma, lo que asegura que el contenido sea fácil de entender.

- Robusto: este principio se cumple gracias a que la aplicación podrá ser usada desde cualquier navegador y sistema operativo una vez esté desplegada en producción. Además, el código CSS y HTML de la aplicación es válido y está bien estructurado.

Respecto a la usabilidad de la aplicación, se han seguido los *Principios de usabilidad web* de Jakob Nielsen [30] para asegurar y validar la misma. A continuación, se explicará cómo se cumple cada uno de ellos.

- Visibilidad del estado del sistema: para informar al usuario de lo que está sucediendo en la aplicación, se han incluido notificaciones que informan de si un envío se ha realizado correctamente o hay un error que lo haya impedido, el nombre de la sección actual destacado en la barra de navegación y el tiempo restante del concurso.

- Relación entre el sistema y el mundo real: para cumplir este principio se ha utilizado un lenguaje familiar para el usuario, claro y sencillo. Además, se han utilizado iconos que recuerdan a objetos del mundo real para apoyar algunos textos y una leyenda de colores con el objetivo de reducir la carga cognitiva.

- Control y libertad del usuario: se permite a los usuarios navegar libremente por la interfaz, teniendo siempre la opción de volver atrás, moverse entre secciones o cerrar formularios en caso de que ya no deseen completarlos. Adicionalmente, se provee la opción de subsanar errores, complementada

por información sobre cómo hacerlo, por ejemplo, en el caso de los formularios.

- Consistencia y estándares: con el objetivo de tener un diseño uniforme en toda la interfaz, se han utilizado los mismos colores, tipografías, iconos y elementos visuales como botones o desplegables en todas las pantallas, teniendo todos ellos un estilo, comportamiento y posición consistente. Adicionalmente, se han utilizado iconos, flechas desplegables y colores estándar, como un icono de una casa para representar la pantalla inicial o el color rojo para representar lo incorrecto y el verde para lo correcto.

- Prevención de errores: como se ha mencionado en puntos anteriores, además de impedir el envío de formularios incompletos, para evitar la entrada de datos incorrectos, en los formularios de la aplicación no se permite la entrada de parámetros de forma textual, sino que se ha decidido implementar menús desplegables en los que únicamente se puede seleccionar una de las opciones presentes. Adicionalmente, si se desea borrar un elemento en el caso de usuarios con rol de juez o administrador, se mostrará una ventana de confirmación para evitar eliminarlos por error.

- Reconocer antes que recordar: para minimizar la carga de memoria del usuario, se han implementado las opciones más usadas de forma que siempre estén visibles y fácilmente accesibles, como el botón de realizar envíos o los botones de navegación. Adicionalmente, como se ha mencionado anteriormente, se han utilizado iconos intuitivos, además de menús claros, sencillos y consistentes que permiten que el usuario interactúe con la interfaz basándose en el reconocimiento de elementos visuales.

- Flexibilidad y eficiencia de uso: se ha diseñado la aplicación con el objetivo de que pueda ser usada por todo tipo de usuarios, tanto principiantes como expertos, por ejemplo, mediante la inclusión de información sobre el funcionamiento de la aplicación y sobre los términos más técnicos. Adicionalmente, se permite la personalización de la interfaz en cualquier momento, ya que se permite el cambio de estilo y de idioma desde cualquier pantalla, además de la disposición de algunos elementos, como los modales de realización de envío o visualización de casos de prueba.

- Diseño estético y minimalista: el diseño de la interfaz se ha centrado en la simplicidad y calidad visual, evitando incluir elementos innecesarios y distribuyendo los necesarios de una forma lógica, usando el espacio de forma eficiente. Además, se ha evitado el uso excesivo de colores y elementos decorativos que no aporten información relevante, para evitar sobrecargar al usuario. Como se ha mencionado anteriormente, se ha priorizado el uso de una tipografía legible, además del contraste suficiente entre los colores del fondo y el texto, obteniendo finalmente una interfaz clara y profesional.

- Reconocer, diagnosticar y corregir errores: en los diferentes formularios de la aplicación, se impide que sean enviados con información incompleta y se notifica al usuario para que pueda corregir dichos errores. Además, los mensajes de error mostrados son claros, utilizando un lenguaje sencillo de entender.
- Ayuda y documentación: para garantizar la ayuda al usuario, se proporciona toda la información necesaria sobre el propósito y funcionamiento de la aplicación en la página inicial, además de diferentes descripciones emergentes con información sobre términos técnicos.

Para finalizar, se ha realizado un vídeo[1] de la interfaz desarrollada en el que se pueden verificar los diferentes aspectos y funcionalidades comentados en este capítulo.

[1] videoInterfazIUDEX.mov

5

Conclusiones y trabajos futuros

En este capítulo se desarrollarán las conclusiones obtenidas del desarrollo de este trabajo. Adicionalmente, se realizarán propuestas para posibles trabajos futuros, para que otras personas puedan aportar a este proyecto.

5.1. Conclusiones

Este trabajo se ha centrado en el desarrollo de la mejora de la interfaz del juez automático de código *iudex*, para conseguir aumentar su funcionalidad, accesibilidad y usabilidad. Para ello se ha creado un prototipo y posteriormente se ha implementado una nueva interfaz, siguiendo los requisitos recopilados y asegurando un rendimiento adecuado.

Esta aplicación podrá ser usada en asignaturas de programación de la universidad, para evaluar a los alumnos o permitir que realicen ejercicios prácticos. Además, puesto que es una aplicación de código abierto, podrá ser utilizada o modificada por otras universidades, asociaciones o usuarios, ya sea con propósitos académicos u orientados a otros ámbitos, como el de la programación competitiva.

A lo largo del desarrollo de este trabajo se han ampliado conocimientos sobre el *framework* Angular y los lenguajes HTML, CSS y Typescript, además de aprender a usar nuevas bibliotecas como *ngx-translate* o *PrimeNG*, las cuales desconocía. También se han adquirido nuevos conocimientos sobre autenticación en aplicaciones web, y se han ampliado los relativos a integración de backend y

frontend de aplicaciones web y a la tecnología de contenedores *Docker*. Adicionalmente, se han obtenido conocimientos sobre el diseño accesible y usable, además de familiarizarse con el desarrollo de proyectos siguiendo una metodología ágil.

Se considera que se han cumplido los objetivos propuestos, obteniendo un producto final que presenta una gran mejoría respecto al estado del arte, habiendo conseguido una interfaz con buen rendimiento, fácil de usar y apta para todo tipo de usuarios.

5.2. Trabajos futuros

A continuación se proponen algunos posibles trabajos futuros para mejorar la aplicación y conseguir un producto realmente completo.

Para mejorar y completar el frontend de la aplicación, se propone la implementación de las vistas de detalles de recursos de usuarios con roles de juez o administrador, las cuales han sido diseñadas en este trabajo, pero no se han implementado por estar fuera del alcance del mismo. También se propone la adición de detalles como una barra de progreso en las vistas de un concurso, para incluir una representación visual del tiempo que queda para que finalice el concurso. Adicionalmente, se sugiere la implementación de WebSockets en el frontend, ya que estos fueron desarrollados en paralelo a este trabajo y, por tanto, no ha sido posible incluirlos. Por último, se propone la realización de pruebas de usuario final, con el objetivo de obtener métricas y aspectos reales a mejorar con el uso real del juez, para validar la interfaz de manera más precisa y obtener conclusiones, lo cual no se ha realizado en este trabajo por falta de tiempo.

Con el objetivo de perfeccionar el backend, se propone utilizar el orquestador de contenedores ya existente para incluir las opciones de compilar el código sin realizar un envío y de *rejudge*, para permitir la reevaluación de los envíos. Además de esto, se sugiere realizar una revisión de los datos proporcionados por la API para que sean más completos, por ejemplo, como ha sido mencionado en la sección 4.4 en el caso de los resultados o de la clasificación, en la que sería deseable disponer de un resumen como el *mockeado* en la anterior interfaz.

Por último, se sugiere la implementación de un sistema que permita la comunicación entre docentes y alumnos, para permitir la resolución de dudas o asistencia técnica. Incluir un servicio de generación de informes para facilitar el trabajo del docente, o la implementación de restricciones para evitar plagios durante pruebas de evaluación, evitando que los usuarios puedan pegar texto externo.

Bibliografía

[1] K. Nikolopoulou, "Digital education in the post-covid era: Challenges and opportunities to explore," in *Towards a Collaborative Society Through Creative Learning*, T. Keane, C. Lewin, T. Brinda, and R. Bottino, Eds. Springer Nature Switzerland, 2023.

[2] S. E. Manyari Del Carpio, J. H. Vargas Manyari, and I. E. Cruz Oyola, "Recursos digitales favorecen el proceso de enseñanza y aprendizaje en tiempos de pandemia," *Horizontes Revista de Investigación en Ciencias de la Educación*, vol. 7, pp. 397 – 402, 03 2023. [Online]. Available: http://www.scielo.org.bo/scielo.php?script=sci_arttext&pid=S2616-79642023000100397&nrm=iso

[3] J. C. Paiva, J. P. Leal, and A. Figueira, "Automated assessment in computer science education: A state-of-the-art review," *ACM Trans. Comput. Educ.*, vol. 22, no. 3, jun 2022. [Online]. Available: https://doi.org/10.1145/3513140

[4] S. Deterding, "Gamification: toward a definition," in *Design, ACM CHI 2011*, D. Tan and B. Begole, Eds., Vancouver, 2011, pp. 12–15. [Online]. Available: http://gamification-research.org/wp-content/uploads/2011/04/02-Deterding-Khaled-Nacke-Dixon.pdf

[5] Genbeta, "Crítica a los exámenes de programación en papel," 2024, accedido: 3 de septiembre de 2024. [Online]. Available: https://www.genbeta.com/desarrollo/critica-a-los-examenes-de-programacion-en-papel

[6] I. C. P. Contest, "The icpc international collegiate programming contest," Fact Sheet, 2024. [Online]. Available: https://icpc.global/worldfinals/fact-sheet/ICPC-Fact-Sheet.pdf

[7] R. Mello, "Predicting the performance of job applicants in coding tests," Research Article, 2024. [Online]. Available: https://gupea.ub.gu.se/handle/2077/52659

[8] J. L. Bez, N. A. Tonin, and P. R. Rodegheri, "Uri online judge academic: A tool for algorithms and programming classes," in *2014 9th International Conference on Computer Science & Education*, 2014, pp. 149–152.

[9] J. Eldering, N. Gerritsen, K. Johnson, T. Kinkhorst, M. Pluijmaekers, M. Vasseur, and T. Werth, *About DOMJudge*, October 2023.

[10] 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.

[11] T. Staubitz, H. Klement, R. Teusner, J. Renz, and C. Meinel, "Codeocean - a versatile platform for practical programming exercises in online environments," in *2016 IEEE Global Engineering Education Conference (EDUCON)*. IEEE, 2016.

[12] G. Derval, A. Gego, P. Reinbold, B. Frantzen, and P. V. Roy, "Automatic grading of programming exercises in a mooc using the inginious platform," in *European Stakeholder Summit on experiences and best practices in and around MOOCs (EMOOCS'15)*, 2015.

[13] P. Antonucci, C. Estler, D. Nikolic, M. Piccioni, and B. Meyer, "An incremental hint system for automated programming assignments," in *Proceedings of the 2015 ACM Conference on Innovation and Technology in Computer Science Education (ITiCSE '15)*. New York, NY, USA: Association for Computing Machinery, 2015.

[14] J. C. R. del Pino, E. R. Royo, and Z. J. H. Figueroa, "Vpl: Laboratorio virtual de programación para moodle," in *Actas de las Jornadas sobre la Enseñanza Universitaria de la Informática (JENUI)*, 2010.

[15] P. L. Parrilla, J. S.-O. Calvo, and I. L. Osorio, "Diseño e implementación de una arquitectura basada en contenedores para la automatización de la evaluación de código," Trabajo de Fin de Grado en Ingeniería de Computadores, 2020/2021.

[16] ——, "Desarrollo de una plataforma de análisis automático de código," Trabajo de Fin de Grado en Ingeniería Informática, 2020/2021.

[17] S. Giri, J. M. C. Verdugo, and R. M. Santamaría, "Nuevas funcionalidades en juez de código online," Trabajo de Fin de Grado en Ingeniería Informática, 2020/2021.

[18] I. P. Ventosa, R. M. Santamaría, and I. L. Osorio, "Evolución y pruebas automáticas de la herramienta educativa iudex," Trabajo de Fin de Grado en Ingeniería Informática, 2021/2022.

[19] M. S. Alonso, J. M. C. Verdugo, and R. M. Santamaría, "Diseño e implementación de una arquitectura basada en contenedores para la automatización de la evaluación de código," Trabajo de Fin de Grado en Ingeniería Informática, 2021/2022.

[20] A. C. Kent Beck, Mike Beedle, "Manifiesto por el desarrollo Ágil de software," https://agilemanifesto.org/iso/es/manifesto.html, 2001.

[21] A. Navarro Cadavid, J. D. Tamayo Martínez, and J. Morales Vélez, "Revisión de metodologías ágiles para el desarrollo de software," *Revista de metodologías ágiles para el desarrollo de software*, 2013.

[22] "About GitHub and Git - GitHub Docs — docs.github.com," https://docs.github.com/en/get-started/start-your-journey/about-github-and-git.

[23] "Angular — angular.dev," https://angular.dev/overview.

[24] R. Johnson, J. Hoeller, and K. Donald..., *Spring Framework Reference Documentation*, 2014. [Online]. Available: https://docs.spring.io/spring-framework/docs/3.2.17.RELEASE/spring-framework-reference/pdf/spring-framework-reference.pdf

[25] "What is Docker? — docs.docker.com," https://docs.docker.com/get-started/docker-overview/.

[26] "Introduction to LaTeX — latex-project.org," https://www.latex-project.org/about/.

[27] Universidad Rey Juan Carlos, *Manual de Identidad Corporativa*, 2023, accessed: 2024-09-23. [Online]. Available: https://www.urjc.es/images/facultades/Manual_de_Identidad_Corporativa.pdf

[28] T. Kozon, "Shedding light on dark mode: Key considerations and advantages when adapting websites," 2023. [Online]. Available: https://boringowl.io/en/blog/shedding-light-on-dark-mode-key-considerations-and-advantages-when-adapting-websites

[29] W. W. A. I. (WAI), "Sumario de WCAG 2 — w3.org," https://www.w3.org/WAI/standards-guidelines/wcag/es.

[30] J. Nielsen, *Usability Engineering*. San Francisco, CA: Morgan Kaufmann, 1994. [Online]. Available: https://www.nngroup.com/books/usability-engineering/