Juez en línea para introducir programación con Blockly

Online judge for children using Blockly

Guzmán Garrido Alique
César Garza Sánchez
Jesús Alejandro Sánchez Couto

Universidad Complutense de Madrid

Facultad de Informática

Trabajo de Fin de Grado en Ingeniería Informática

Tutores Marco Antonio Gómez Martín
Pedro Pablo Gómez Martín

Curso 2020/2021

Resumen

En este trabajo se desarrolla una herramienta que se pueda utilizar en la enseñanza para usuarios que estén poco familiarizados con la programación. Se trata de un juez en línea que utiliza Blockly, una librería de Google que permite programar de manera visual. Al tratarse de una programación por medio de bloques hace que la barrera de entrada sea mucho más leve y fácil de aprender. Se proporcionan una serie de problemas a los usuarios para que pongan a prueba los conocimientos aprendidos y su mejora de pensamiento computacional.

Se estudia también cuan importante es la enseñanza de la programación en edades tempranas y trabajar el pensamiento computacional desde el instituto.

Palabras clave

Blockly, Programación, Pensamiento computacional, Juez en línea, Python, Enseñanza, JavaScript, Algoritmos.

Abstract

The purpose of this project is developing a tool that can be used for teaching users who are not very familiar with programming. It is an online judge that uses Blockly, a Google library that allows visual programming. Being a block-oriented programming makes the entry barrier much lighter and easier to learn. A series of problems are provided to users to test the knowledge they have learned and their improvement of computational thinking.

It is also studied how important is it to teach programming at an early age and working on the computational thinking since highschool. The progress of this kind of thinking as exersises and tests are carried out is also observed.

Keywords

Blockly, Programming, Computational thinking, Online judge, Python, Education, Algorithms, Javascript

Índice

Lista de figuras

1. Introducción

1.1 Objetivos

El objetivo de este proyecto es desarrollar una herramienta web que permita a los personas sin conocimientos de informática o sin nociones de pensamiento computacional familiarizarse con estos conceptos en un entorno simplificado, amigable y sencillo.

Esta página debe permitir a los usuarios resolver problemas de distinta dificultad desde cualquier dispositivo con acceso a Internet, con una mínima asistencia por parte de los docentes. Una vez entregada la solución, la página la compilará, ejecutará y comparará su salida con los casos de prueba de ejemplo. En caso de coincidir en todos, el veredicto será positivo. Si difiere en algún caso, será negativo.

El usuario podrá revisar las soluciones que ha enviado, volver a intentar las entregas fallidas tras modificarlas, además de ver su progreso y acceder a su cuenta desde cualquier lugar mediante su usuario y contraseña. También podrán ver la salida de su código, así como casos de prueba extra en caso de que necesiten más ayuda para resolver un ejercicio.

Los administradores podrán añadir o modificar problemas que ya existan, además de poder importarlos o exportarlos a una base de datos en masa. También serán capaces de modificar los permisos de los usuarios.

Todo el código está disponible en Github en `https://github.com/cesarcgs/TFGBlockly` y la página es accesible desde `https://tfg-blockly.netlify.app/`. El usuario de prueba es ***usuario1*** y la contraseña ***usuario1*** también. La primera carga de la página tarda unos segundos ya que el worker de la API ha de empezar a funcionar.

1.2 Estructura de la memoria

La presente memoria muestra tanto el desarrollo de la página web, mencionando las tecnologías y la metodología utilizadas, así como las pruebas realizadas sobre usuarios, comprobando la efectividad de la herramienta.

Para comenzar, se introduce el concepto de jueces en línea, seguido de varios ejemplos de distintas implementaciones de jueces ya utilizadas a nivel global. Se hace después una introducción a la librería Blockly de Javascript, explicando brevemente su funcionamiento, así como el uso educativo

de este y otros lenguajes. A continuación, se habla en profundidad de las tecnologías utilizadas en la página web, tanto en el servidor como en el cliente, tras lo cual se explica en detalle el funcionamiento del sistema y los cambios realizados.

Por último, se muestran los resultados del estudio realizado entre personas de diferente perfil para comprobar la validez de la herramienta, mostrando los avances de dichos usuarios al enfrentarse a un pequeño test sobre Blockly y Python después de resolver ejercicios utilizando la herramienta.

1.3 Metodología e integración

Para la división de trabajo y organización del equipo hemos utilizado metodología ágil Kanban autogestionándonos el trabajo y aumentando la productividad.

En cuanto al código hemos seguido el flujo Git Flow, lo que nos ayuda en la integración continua, ya que se tendrá en la rama *master* siempre una versión sin errores.

Para la integración continua hemos utilizado Travis CI, Heroku y Netlify. Travis CI se encarga de clonar el repositorio de GitHub en un entorno virtual y realizar pruebas sobre este. Si funcionan todas las pruebas, se considera "superado" y Travis CI podrá levantar la parte del servidor en un servicio, Heroku en este caso. El cliente se despliega en Netlify que permite ejecutar páginas web estáticas y aplicaciones web en la nube a través de un servicio de backend sin servidor. Netlify actualiza el cliente igual que Travis CI, en este caso, cada vez que se hace un *push* a *master*. De este modo, tanto servidor como cliente están siempre levantados con la última versión estable.

1. Introduction

1.1 Goals

The goal of this project is to develop a web page that allows people without programming skills or notions of computational thinking to become familiar with these concepts in a simplified, friendly and simple environment.

This page should allow users to solve problems with different difficulty from any device with Internet access, with minimal assistance from teachers. After the solution is delivered, the page will compile it, run it, and compare its output to the sample test cases. If all the results are correct, the verdict will be positive. If it differs in any case, it will be negative.

The user will be able to review the solutions they have sent, retry failed submissions after modifying them, as well as view their progress and access their account from anywhere using their username and password. They will also be able to see the output of their code, as well as extra test cases in case they need more help solving an exercise.

Administrators will be able to add or modify problems that already exist, as well as being able to import or export them to a database in bulk. They will also be able to modify user permissions.

All the code of the proyect is available in the Github repository `https://github.com/cesarcgs/TFGBlockly` and the web page is `https://tfg-blockly.netlify.app/`. The test username is ***usuario1*** and the password is ***usuario1*** as well

1.2 Structure of the memory

This report shows both the development of the website, mentioning the technologies and methodology used, as well as the tests carried out on users, verifying the effectiveness of the tool.

To begin with, the concept of online judges is introduced, followed by several examples of different implementations of judges already used globally. An introduction to the Blockly Javascript library is then made, briefly explaining its operation, as well as the educational use of this and other languages. Next, the technologies used on the website are discussed in depth, both on the server and on the client, after which the operation of the system and the changes made are explained in detail.

Finally, the results of the study carried out among people of different profiles to check the validity

of the tool are shown, showing the progress of these users when facing a small test on Blockly and Python after solving exercises using the tool. The first load of the web will take some seconds due to the API's worker has to get started.

1.3 Metodology and integration

To organize the team and divide the work, we have used Kanban agile methodology to self-manage the work and increase productivity.

As for the coding, we have followed the Git Flow, which will help us in the continuous integration, since the master branch will always have a version without errors.

For continuous integration we used Travis CI, Heroku and Netlify. Travis CI was responsible for cloning the GitHub repository in a virtual environment and performing the tests on it. If all the tests work, it's considered "passed" and Travis CI will be able to deploy the server side on a service, Heroku in this case. The client is deployed on Netlify, which supports running static web pages and web applications in the cloud via a serverless backend service. Netlify updates the client just like Travis CI, in this case, every time a push is made to master. This way, both server and clients are always up with the latest stable version.

2. Estado del Arte

2.1 Enseñanza de la programación

2.1.1 Beneficios de la programación

Actualmente la tecnología forma parte de la vida cotidiana de muchas personas, desde enviar mensajes a través de teléfono móvil hasta gestionar recibos o comprar en el supermercado. La tecnología está tan presente en el día a día que es necesario tener unos conocimientos mínimos para poder entender cómo funciona y hacer uso de ella de forma más segura.

A día de hoy, los jóvenes entran en contacto con la tecnología a edades muy tempranas, y muestran un dominio en el uso de estas a los dos o tres años. A estos se les conoce como *nativos digitales*, mientras que a las personas que no han nacido en la era de la información, sino que se han adaptado al uso de las tecnologías, se les atribuye el término *inmigrantes digitales*. La atribución de estos términos a diferentes personas conlleva una serie de problemas para los nativos, los jóvenes, ya que muchos adultos asumen que no es necesario educarles en este aspecto, viendo cómo parecen saber moverse por las redes y aplicaciones. En consecuencia, las habilidades que desarrollan los nativos no suele ir más allá de usar dichas herramientas, sin plantearse qué esta ocurriendo, cómo, ni qué riesgos puede conllevar. Impartir conocimientos de la programación les haría más conscientes de estos problemas y solventaría en parte la exclusión que algunas personas *fuera de la norma* sufren, ya sea por no estar tan expuestas a las tecnologías o por simplemente no disponer de herramientas propias, como un teléfono móvil [1].

Sin embargo, conocer las bases de la informática no solo permite aprender a escribir un programa sencillo y entender en términos generales cómo funciona una aplicación cualquiera por debajo. La principal ventaja de aprender a comprender y escribir programas es el desarrollo de un pensamiento computacional. Esta forma de pensamiento es útil a la hora de resolver un problema difícil transformándolo en otro más sencillo que se sabe cómo resolver [2]. Además, ayuda a desarrollar otras aptitudes tales como la creatividad, la abstracción, el ensayo-error o la recursividad, que beneficia a un individuo a tratar problemas en su día a día, incluso cuando no está necesariamente interactuando con una máquina. Adicionalmente, también ayuda a practicar el trabajo en equipo y acostumbrarse a interactuar con otras personas para llevar a cabo un proyecto, lo cual también favorece a las personas en el ámbito social.

Finalmente, los puestos de trabajo relacionados con la informática solo van en aumento. Un gran

beneficio de generalizar la informática es la promoción de sus carreras universitarias y la formación de más personas que podrían ocupar estos puestos.

2.1.2 Programación en los colegios

Aunque ya hay países en los cuales forma parte de la educación obligatoria, como Reino Unido o Estados Unidos [3], actualmente en la mayoría de las aulas se utilizan programas como herramientas para impartir otros conocimientos en lugar de ser el objeto de estudio.

En el caso de España, la informática ya forma parte de la educación obligatoria, siendo su asignatura correspondiente en la ESO "Tecnología, Programación y Robótica". En esta asignatura se enseñan, junto con otros conocimientos de informática, las bases de la programación a través de Scratch, un lenguaje de programación que utiliza una interfaz visual y un sistema de bloques para simplificar la escritura de un programa.

Además, los contenidos de la programación ya forman parte del currículo en otros países. El primero en establecerla como asignatura obligatoria fue Reino Unido en 2014, previendo un aumento en los puestos de trabajo relacionados con la informática y la ciberseguridad, y es ahora una asignatura obligatoria hasta los 16 años. En 2007, de acuerdo a un estudio de *Royal Society*, la impresión mayoritaria que la informática como asignatura daba a los jóvenes era que resultaba aburrida y repetitiva, motivo por el cual el número de personas que cursaban esta asignatura era muy bajo [4]. Sin embargo, *The Welcome Trust*, una fundación de caridad centrada en la investigación de la salud con sede en Londres, realizó otra encuesta a 4.000 personas en 2017, y las carreras relacionadas con la informática acumularon un 11% de los encuestados, llegando a ser la tercera carrera de ciencias más popular [5].

	Todos los jóvenes	Hombres %	Mujeres %
Medicina	27	14	44
Ingeniería	24	34	10
Ciencias de la Computación	11	17	3
Psicología	8	3	16
Magisterio	7	5	10
Veterinaria	6	3	10
⋮	⋮	⋮	⋮

Fuente: *Wellcome trust 2017. Young people's views on science education. Science Education Tracker Research Report.*

2.1.3 Lenguajes de programación para principiantes

Una de las formas de introducir la programación a los alumnos en escuelas es mediante lenguajes de programación sencillos. Este tipo de programas han estado presente durante un tiempo, aunque solo recientemente se ha planteado usarlos en las escuelas para impartir clases de programación.

- **Logo**: A finales de los 60, Logo fue desarrollado por Wally Feurzeig, Seymour Papert y Cynthia Solomon [6]. Se caracteriza por tener una tortuga que actúa a modo de pincel y dibuja formas según los comandos que se le hayan dado al ejecutar el código. Su lenguaje es muy similar al natural, con comandos tales como forward o right, y simplificaba el aprendizaje de conceptos como la recursión o el razonamiento de sincronicidad corporal, un proceso por el cual el programador entiende, razona y predice el movimiento de la tortuga si ejecutara ciertos comandos. Además, al ser un lenguaje interpretado, no necesita compilar para depurar el código. Tuvo mucho éxito hasta finales de los 90, donde su uso empezó a disminuir a favor de otros lenguajes sencillos.

- **Mindstorm**: Más que como un lenguaje de programación, LEGO, junto con otras entidades como MIT, lanzaron una serie de juguetes con fines educativos. Similar a los LEGO actuales, dispone de varias piezas diferentes que pueden combinarse para crear robots concretos, pero en este caso, varias de estas piezas vienen con diferentes funcionalidades: un motor, sensores, cables, bluetooth o el ordenador, desde el cual se puede mandar una serie de comandos al resto de piezas. Los comandos se representan a modo de bloques, incluyendo variables, constantes, acciones, sensores e incluso bloques personalizados que pueden combinarse entre sí.

- **Scratch**: Scratch fue el primer lenguaje de programación con una interfaz gráfica, con su primera versión desarrollada en 2007 por MIT. De forma similar a Blockly, es de código abierto y utiliza un sistema de bloques para dar lugar a animaciones, juegos y aplicaciones sencillas, con la diferencia de que no se basa en otros lenguajes de programación. Dispone de una interfaz de bloques intuitiva y fácil de usar. Una gran ventaja que tiene Scratch es que todos los programas deben almacenarse en su servidor, de modo que programadores con menos experiencia tienen a disposición proyectos de otros programadores más experimentados a través de los cuales pueden aprender.

- **Alice**: Es un lenguaje de programación basado en objetos lanzado en 1998, permitiendo, de forma similar a Scratch, arrastrar y combinar bloques para definir un comportamiento. Sin embargo, a diferencia de Scratch, Alice permite la creación de objetos 3d y eventos, así como su uso en el programa de bloques. Dispone de varias ventanas donde se muestran el árbol de objetos, dichos eventos, el programa en sí y la escena resultante del programa. Además, en caso de fallo, también muestra una ventana con los pasos que ha dado hasta llegar al error, simplificando la depuración.

- **Kodu**: También utiliza un sistema de bloques para escribir un programa. En este caso,

sin embargo, el objetivo es crear un juego 3d con diferentes personajes independientes, cada uno de los cuales tiene un comportamiento propio definido por el programador, en lugar de limitarlo a una animación invariable.

Los lenguajes de programación con interfaces visuales e intuitivas son ideales para que jóvenes de 8 a 16 años empiecen a adquirir aptitudes tales como el pensamiento crítico, persistencia, habilidades cognitivas o creatividad [7]. Este tipo de lenguajes son los más usados para introducir la programación a alumnos de primaria, siendo Scratch el más popular, mientras que en secundaria se empiezan a introducir otros lenguajes de programación tales como Python o JavaScript, aunque muchos centros escolares aún hacen uso de interfaces visuales en secundaria [8].

Centros que usan el lenguaje %			
Primaria		Secundaria	
Scratch	38	Python	21
Logo	17	Scratch	19
Kodu	15	JavaScript	10
Otro	4	Kodu	8
Blockly	4	Visual Basic	6
JavaScript	3	Logo	5
Microsoft TypeScript	3	Small Basic	5
Espresso	3	Lenguajes C	4
⋮	⋮	⋮	⋮
Swift	1	Blockly	3

Fuente: *Pye Tait*.

2.2 Jueces en línea

Un juez de programación en línea se trata de un software (web habitualmente) que emite un veredicto sobre la corrección y eficiencia de la solución a ciertos problemas de programación. Estos son proporcionados por los administradores del sistema mientras que las soluciones son proporcionadas por el usuario. Por lo tanto está compuesto tanto por un corrector automático, así como por un repositorio con los enunciados de los problemas disponibles. La naturaleza de la mayoría de estos ejercicios es lógica o matemática, como pueden ser problemas de combinatoria, teoría de números, algoritmia o estructuras de datos. Estos sistemas tienen diversos usos, como puede ser la enseñanza, la selección de personal, la minería de datos o la organización de concursos de programación, siendo esta última la más extendida.

2.2.1 Jueces de programación competitiva

Se trata del uso más extendido de estos programas, donde equipos de participantes se enfrentan a un mismo conjunto de problemas con el objetivo de terminarlos de la manera más rápida y eficaz posible. El veredicto de dichos sistemas es habitualmente llevado a cabo en máquinas anfitrionas o "jueces", tras someter el resultado a extensos casos de prueba. La respuesta del juez suele basarse en un sistema de todo-o-nada, es decir, solo considera correcta una entrega si ha superado todos los casos de pruebas sin excepción. Sin embargo, puesto que este comportamiento puede ser frustrante, algunos recurren a distintos sistemas de puntuación que premian la corrección. Estos concursos tienen un crecimiento cada vez más grande, habiendo participado más de 55.000 personas en las fases clasificatorias de algunos de ellos [9].

Hay una gran variedad de software preparado para organizar estas competiciones, tanto a pequeña escala para un grupo reducido de participantes, tanto como para una gran cantidad de universidades a nivel mundial.

Algunos de las competiciones más famosas son ICPC[1] (International Collegiate Programming Contest), IOI[2] (International Olympiad in Informatics) o Google Code Jam[3] (patrocinado por Google).

CMS

CMS, o Contest Management System[4] (figura 2.1) es un software utilizado para organizar concursos de programación. Ha sido desarrollado por gente involucrada en la organización de este tipo de eventos a nivel nacional e internacional, habiéndose creado pensando en la seguridad, la portabilidad, la facilidad de uso y la adaptabilidad a diferentes situaciones.

El sistema esta organizado de manera modular, con diferentes servicios ejecutándose potencialmente en varias máquinas, proporcionando extensibilidad replicando dichos servicios en otras máquinas. Los servicios principales son LogService, ResourceService, Checker, EvaluationService, Worker, ScoringService, ProxyService, PrintingService, ContestWebServer y AdminWebServer.

El sistema ha sido utilizado en competiciones de renombre, como la IOI (International Olympiad of Informatics) [10] durante 5 años o la CEOI (Central European Olympiad in Informatics).

DOMjudge

DOMjudge[5] (figura 2.3) es un sistema utilizado para gestionar concursos de programación. Posee un mecanismo para entregar soluciones a problemas y juzgar las soluciones de manera totalmente automática, además de interfaces web para los equipos, el jurado y los espectadores.

[1] `https://icpc.global/`
[2] `https://ioinformatics.org/`
[3] `https://codingcompetitions.withgoogle.com/codejam`
[4] `https://cms-dev.github.io/`
[5] `https://www.domjudge.org/`

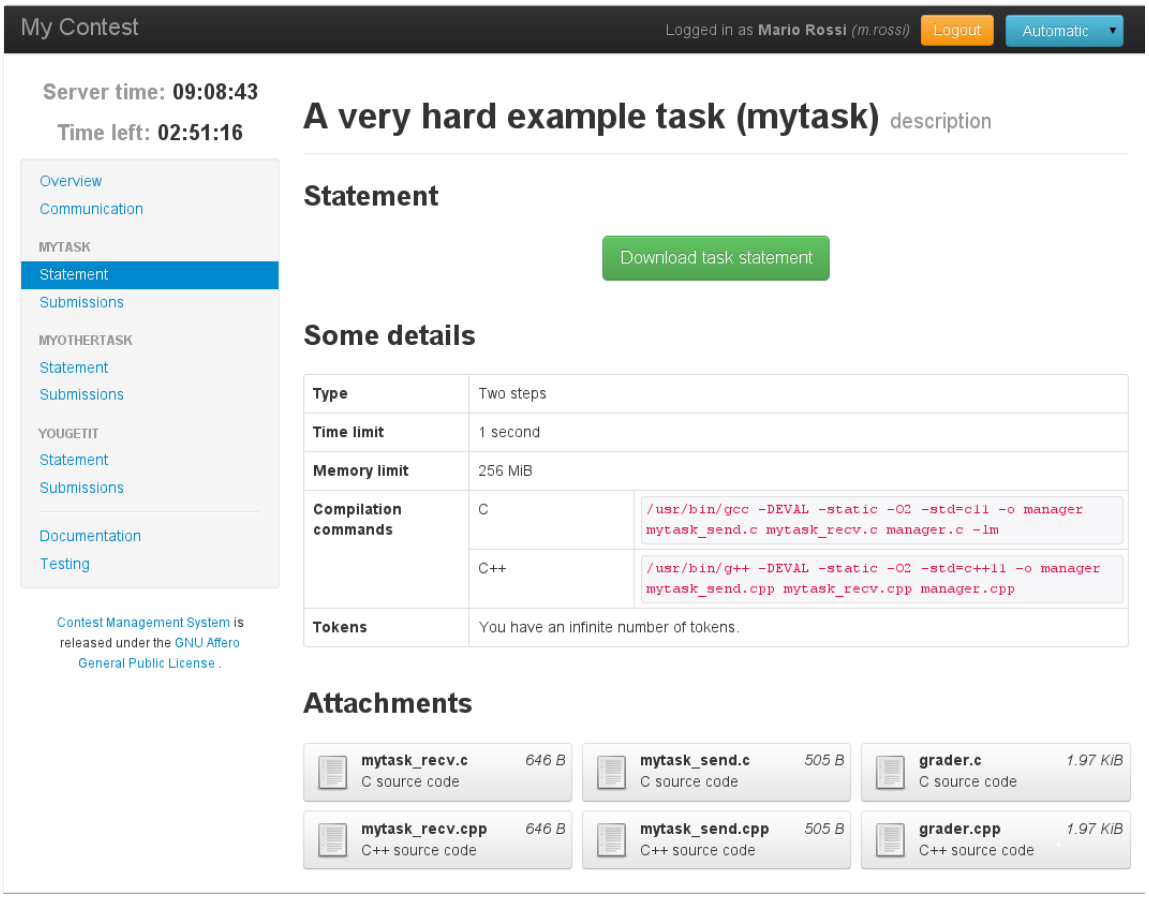


Figura 2.1: Interfaz de un problema en un concurso de CMS. Imagen extraída de la página de CMS

Está pensado principalmente para ser usado en concursos de programación, donde existe un tiempo especificado de entrega y ciertos problemas, pero puede ser adaptado a otros contextos como la enseñanza.

El sistema presenta una muy buena escalabilidad [11], teniendo un sistema modular para agregar lenguajes extra o compiladores. La interfaz de los equipos es simple, mientras que la parte de los administradores posee características muy interesantes, como son la posibilidad de probar el efecto de cambios en envíos realizados por los participantes, aclaraciones o información detallada sobre entregas. Cuenta además con una API para poder aumentar las funcionalidades del sistema con aplicaciones secundarias.

2.2.2 Jueces educativos

Este tipo de sistemas provee a los profesores de una herramienta con la que ofrecer a los alumnos ejercicios (normalmente de iniciación a la programación) que estos puedan resolver con mayor flexibilidad. El uso de estos sistemas proporciona varias ventajas. En primer lugar, provee de una mayor objetividad y rigurosidad a la enseñanza, permitiendo a los profesores verificar con facilidad la exactitud de las soluciones, aliviando la carga de trabajo por parte del profesor. En segundo

lugar, los estudiantes reciben una retroalimentación casi inmediata de manera que pueden realizar más rápidamente los problemas. Por último, el servicio goza de una disponibilidad absoluta [12], no limitado al horario lectivo. Esto permite a los alumnos entregar los ejercicios incluso desde su casa con su propio ordenador, cuando tengan tiempo disponible.

Algunos de los jueces más importantes de este tipo son *¡Acepta el reto!*[6], HackerRank[7], u online-judge.org[8].

¡Acepta el reto!

¡Acepta el reto![9](figura 2.2) es una pagina creada en 2014 por dos profesores de la Facultad de Informática de la Universidad Complutense de Madrid, que habían contribuido a la puesta en marcha de ProgramaMe, un concurso de programación para alumnos de Ciclos Formativos de Formación Profesional. A raíz de la existencia de esos problemas que carecían de utilidad más allá de ser resueltos sin poder probar su corrección, decidieron desarrollar este juez en línea con la ayuda de estudiantes.

El juez acepta soluciones tanto en C, C++ como Java, corrigiéndolas automáticamente. Dependiendo del resultado, el juez puede devolver una respuesta diferente: AC (Aceptada), WA (Respuesta incorrecta), CE (Error de Compilación), RTE (Error durante la ejecución), TLE (Tiempo límite superado), MLE (Límite de memoria superado), OLE (Límite de salida superado), RF (Función restringida) o IE (Error interno).

Este juez en línea dispone de más de 500 problemas publicados separados en distintas categorías, además de contar con más de 15000 usuarios. Por último, los enunciados están en castellano, a diferencia de la mayoría de los jueces [13]. Esto supone normalmente una barrera para los estudiantes de los primeros cursos de los grados.

2.2.3 online-judge-mean

Online-judge-mean[10][11] es una aplicación web desarrollada por jojozhuang, cuya función es resolver preguntas sobre algoritmia. El usuario puede enviar sus respuestas y comprobar si ha superado todos los casos de prueba. Otras funciones que tiene son:

- Autenticación basada en tokens: Permite el registro, acceso, auto-registro, reinicio de contraseña, etc.
- Gestión de usuarios: Crear, modificar o eliminar usuarios
- Gestión de problemas: Creación, modificación o eliminación de preguntas.

[6] `https://www.aceptaelreto.com/`
[7] `https://www.hackerrank.com/`
[8] `https://onlinejudge.org/`
[9] `https://www.aceptaelreto.com/`
[10] `https://github.com/Bit-Developer/online-judge-mean`
[11] `https://online-judge.netlify.app/`

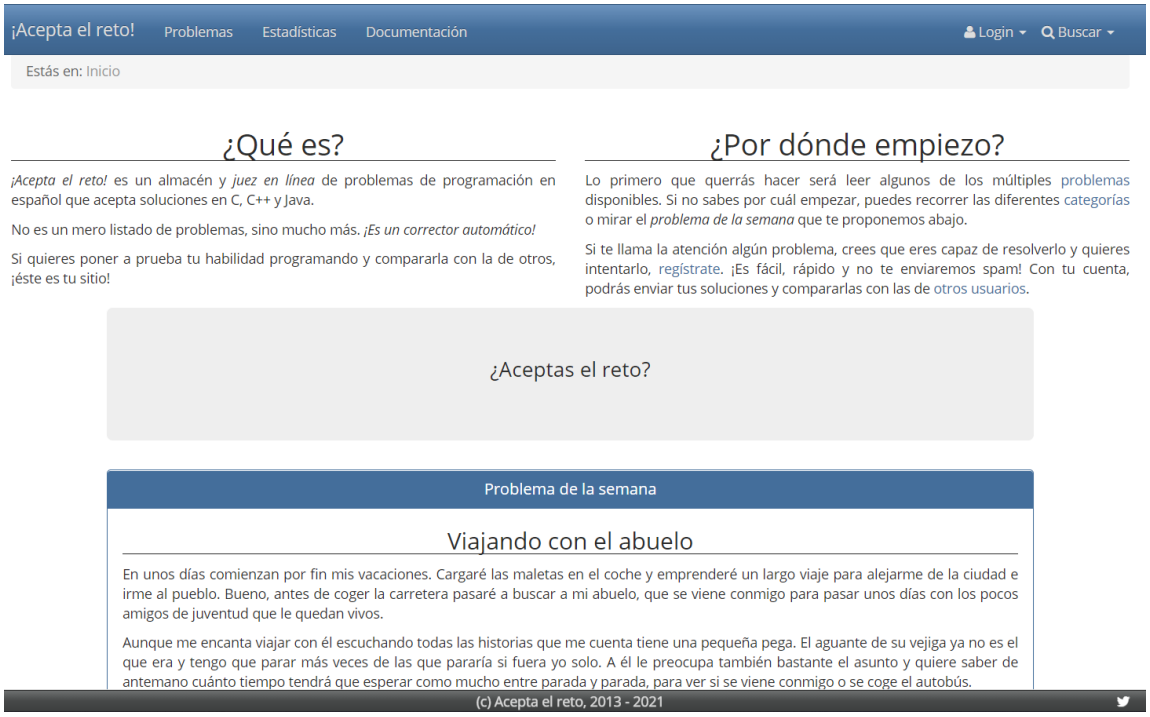


Figura 2.2: Captura de pantalla de Acepta el Reto

- Gestión de la base de datos: Integra un sistema de importación y exportación de datos mediante archivos .csv para los usuarios, problemas y entregas.
- Sistema de evaluación: Corrección y feedback de las entregas.
- Lenguajes de programación: El sistema soporta entregas en 3 lenguajes diferentes: Python, Javascript y Java.
- Interfaz de Usuario: Cuenta con un editor de código, barra de progreso.

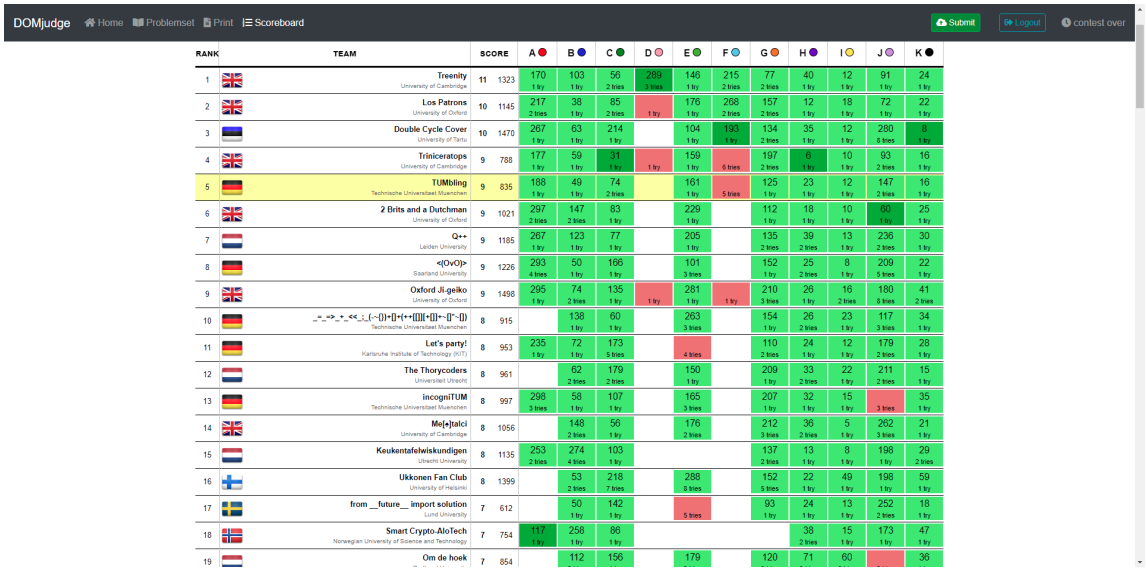
Se hablará en más profundidad sobre este sistema en el capítulo 3.

2.3 Blockly

2.3.1 Qué es Blockly

Blockly es un proyecto de Google, una librería Javascript de código abierto que permite programar de una forma enteramente visual [14].

Su primera versión fue lanzada en 2012 por Neil Fraser, Quynh Neutron, Ellen Spertus y Mark Friedman a modo de sustituto de OpenBlocks, otra herramienta de programación con bloques. A diferencia de otros lenguajes de programación, Blockly añade una capa más de abstracción a través de bloques que, de forma similar a Scratch, pueden moverse y combinarse para crear programas.



RANK	TEAM	SCORE	A	B	C	D	E	F	G	H	I	J	K
1	Treenity University of Cambridge	11 1323	170 1 try	103 1 try	56 2 tries	289 3 tries	146 1 try	215 2 tries	77 2 tries	40 1 try	12 1 try	91 1 try	24 1 try
2	Los Patrons University of Oxford	10 1145	217 2 tries	38 1 try	85 2 tries	1 try	176 1 try	268 2 tries	157 2 tries	12 1 try	18 1 try	72 1 try	22 1 try
3	Double Cycle Cover University of Tartu	10 1470	267 1 try	63 1 try	214 1 try		104 1 try	193 1 try	134 2 tries	35 1 try	12 1 try	280 3 tries	6 1 try
4	Trinieratops University of Cambridge	9 788	177 1 try	59 1 try	31 1 try	1 try	159 1 try	6 tries	197 2 tries	6 1 try	10 1 try	93 2 tries	16 1 try
5	TUMbling Technische Universitaet Muenchen	9 835	188 1 try	49 1 try	74 2 tries		161 1 try	5 tries	125 1 try	23 1 try	12 1 try	147 2 tries	16 1 try
6	2 Brits and a Dutchman University of Oxford	9 1021	297 2 tries	147 2 tries	83 1 try		229 1 try		112 1 try	18 1 try	10 1 try	60 1 try	25 1 try
7	Q++ Leiden University	9 1185	267 1 try	123 1 try	77 1 try		205 1 try		135 2 tries	39 2 tries	13 1 try	236 2 tries	30 1 try
8	<(OvO)> Saarland University	9 1226	293 4 tries	50 1 try	166 1 try		101 3 tries		152 1 try	25 2 tries	8 1 try	209 5 tries	22 1 try
9	Oxford Ji-geiko University of Oxford	9 1498	295 1 try	74 2 tries	135 1 try	1 try	281 1 try	1 try	210 3 tries	26 1 try	16 2 tries	180 6 tries	41 2 tries
10	_=_=>_+_<<_;-{--{}+[]+{++[][][+[]]+--[]^-[]} Technische Universitaet Muenchen	8 915		138 1 try	60 1 try		263 3 tries		154 1 try	26 2 tries	23 1 try	117 3 tries	34 1 try
11	Let's party! Karlsruhe Institute of Technology (KIT)	8 953	235 1 try	72 1 try	173 5 tries		4 tries		110 2 tries	24 1 try	12 1 try	179 2 tries	28 1 try
12	The Thorycoders Universiteit Utrecht	8 961		62 2 tries	179 2 tries		150 1 try		209 1 try	33 2 tries	22 1 try	211 2 tries	15 1 try
13	incogniTUM Technische Universitaet Muenchen	8 997	298 3 tries	58 1 try	107 1 try		165 3 tries		207 1 try	32 1 try	15 1 try	3 tries	35 1 try
14	Me[+]talci University of Cambridge	8 1056		148 2 tries	56 1 try		176 2 tries		212 3 tries	36 2 tries	5 1 try	262 3 tries	21 1 try
15	Keukentafelwiskundigen Utrecht University	8 1135	253 2 tries	274 4 tries	103 1 try				137 2 tries	13 1 try	8 1 try	198 1 try	29 2 tries
16	Ukkonen Fan Club University of Helsinki	8 1399		53 2 tries	218 7 tries		288 8 tries		152 5 tries	22 1 try	49 1 try	198 1 try	59 1 try
17	from __future__ import solution Lund University	7 612		50 1 try	142 1 try		5 tries		93 1 try	24 1 try	13 1 try	252 2 tries	18 1 try
18	Smart Crypto-AloTech Norwegian University of Science and Technology	7 754	117 1 try	258 1 try	86 1 try					38 2 tries	15 1 try	173 1 try	47 1 try
19	Om de hoek	7 854		112	156		179		120	71	60		36

Figura 2.3: Interfaz de un problema en un concurso terminado en DOMjudge

2.3.2 Perspectiva del cliente

Cada tipo de bloque representa un elemento o una acción, no necesariamente en relación con la informática: un bloque podría usarse para inicializar variables o realizar operaciones matemáticas, y otro podría usarse para desplazar a un personaje por un mapa o girar una figura. Los bloques se pueden seleccionar en una tabla, arrastrar y colocar donde uno prefiera dentro de un canvas, o lienzo. Se pueden combinar entre sí como piezas de un puzle y vienen diseñados de forma que al usuario le sea fácil intuir qué bloques pueden combinarse con qué y en qué orden.

De esta forma, el resultado es una aglomeración de bloques de diferentes colores, tamaños y formas con un sentido y fin claros.

2.3.3 Instalación: NPM

La librería Blockly se puede descargar a través de diferentes enlaces en su Github[12]. Dispone de una carpeta "demos", a través de la cual se pueden probar varias funcionalidades del cliente, como un canvas que mantiene un tamaño fijo, o lo ajusta según se cambia el de la página; ejecutar o mostrar el código resultante de un conjunto de bloques...

Para poder utilizar la librería Blockly, solo se necesita descargar sus ficheros e importarlos en el código donde se quiera usar. Sin embargo, al estar incluida en los registros de NPM, introducirlo en un proyecto existente se reduce a un solo comando.

NPM, o *Node Package Manager*, es un gestor de paquetes que permite descargar librerías de código para utilizarlas en un proyecto. Todas las librerías descargadas con NPM se almacenan en una carpeta llamada "node_modules". Del mismo modo, también gestiona la dependencias necesarias

[12] `https://github.com/google/blockly`

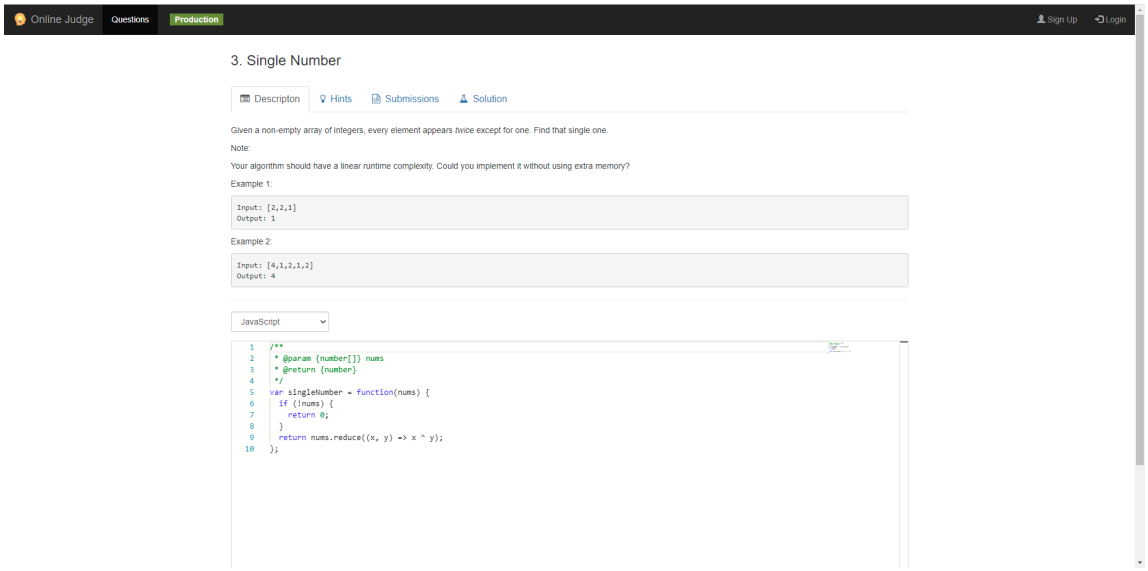


Figura 2.4: Interfaz de un problema en online-judge-mean

para poder utilizar las librerías en el proyecto sin que de error porque se intente utilizar una librería que no se encuentre en el proyecto o no se haya relacionado con los ficheros que la usan.

2.3.4 Los bloques

Todos los bloques de la librería se encuentran en varios archivos .js en la carpeta *blocks*. Estos archivos contienen listas donde se dan valores a los diferentes elementos que componen un bloque. Estas listas están escritas en JSON, pero Blockly permite también permite crear bloques en Javascript. Ambos inicializan los mismos valores y se pueden combinar y usar indistintamente, salvo en el caso de los mutadores o validadores, los cuales solo pueden definirse en Javascript.

Algunos elementos que componen un bloque cualquiera son:

- TYPE: El identificador del bloque, que se usará más adelante para incluirlo en la página e interactuar con él.
- MESSAGE0: Texto qué aparecerá dentro del bloque una vez sea visible en la página. Si existen huecos donde puede enlazarse con otros bloques o elementos que el cliente puede modificar, se deben representar individualmente con el prefijo % seguido de un número, en orden: %1, %2, %3...
- ARGS0: Si se han usado prefijos en message0, se concreta aquí para cada variable si se espera otro bloque, un valor o una de varias opciones predeterminadas. No se define en otro caso.
- OUTPUT: Define el tipo del resultado de la operación que representa el bloque. Por ejemplo, en el caso de una operación matemática, se espera un *Number*, y en el caso de una comparación un *Boolean*. Si se define aparecerá un conector a la izquierda.

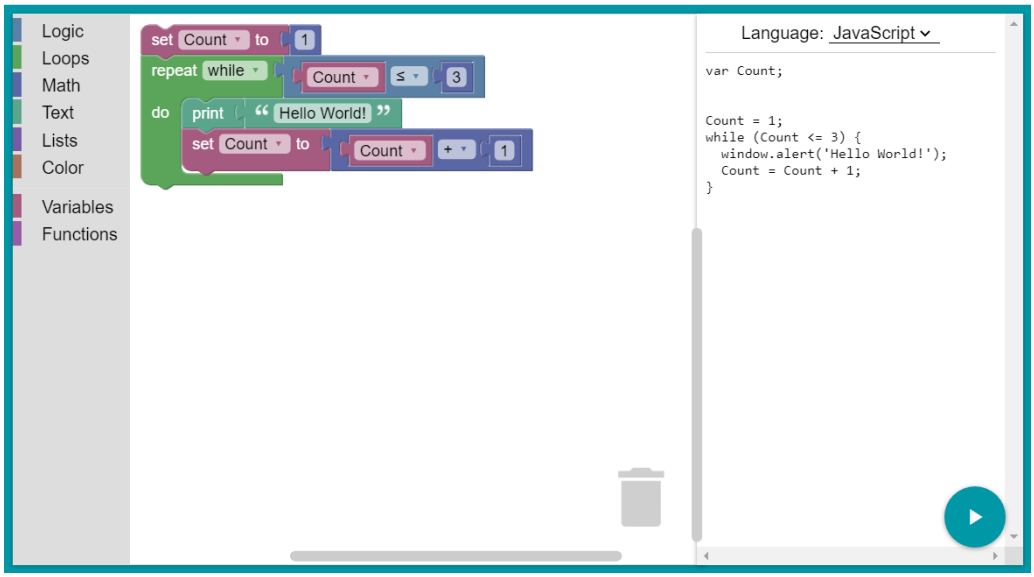


Figura 2.5: Captura de un programa en blockly.

- PREVIOUSSTATEMENT y NEXTSTATEMENT: Indican si se pueden enlazar con otros bloques por arriba y por debajo respectivamente y cuáles son, admitiendo cualquiera si su valor es *null*. No se definen si no se quiere añadir conectores.
- COLOUR: El color del bloque.
- TOOLTIP: Es un texto explicativo que aparece cuando el cursor se encuentra encima del bloque. Se puede usar de forma opcional para explicar su funcionalidad o proporcionar otra información adicional.
- MUTATOR: Define si un bloque puede ser modificado por el cliente con una serie de opciones que se le proporciona. Se puede declarar en formato JSON junto con el resto de elementos del bloque, y las extensiones en sí se definen aparte como se haría cualquier bloque independiente, pero todo su comportamiento se debe escribir en Javascript. Un mutador incluye:
  - `decompose`: Si un bloque tiene mutadores definidos, tendrá un icono de un engranaje desde el cual se puede abrir una ventana. Desde esta ventana se podrán ver otros bloques complementarios que se pueden usar para modificar el bloque principal. La función decompose organiza la forma en la que se muestran y combinan los bloques en esta ventana.
  - `compose`: Esta función se encarga de reconstruir el bloque principal de acuerdo a la combinación de bloques complementarios. Se ejecuta cada vez que se incluye o se elimina un bloque complementario en la ventana de *decompose*.
  - `saveConnections`: Según se van modificando bloques con la función compose, es necesario que las conexiones con otros bloques no se pierdan debido a la adición o sus-

tracción de mutadores dentro al bloque. Esta función se llama antes de *compose* para asegurar que esto ocurra, en el caso de que se haya definido.

– Se pueden incluir otras funciones que añadan otras funcionalidades a los mutadores y asegurarse, por ejemplo, de que el número de valores de entrada es el correcto. Por ejemplo, el bloque *math_number_property*, que comprueba si un número es par, impar, negativo, entero... dispone de un mutador con una extensión que añade una entrada adicional a la derecha, en el caso de que se quiera comprobar si el número es divisible por otro.

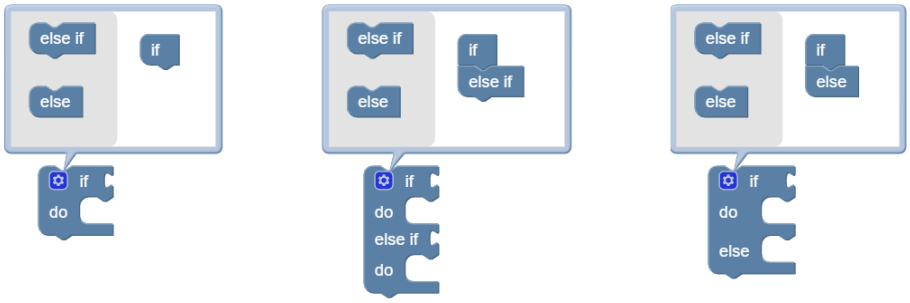


Figura 2.6: Ejemplo UI de un mutador else_if.

Además de los bloques por defecto, también se pueden crear bloques personalizados manualmente o a través de la herramienta online Blocky Developer Tools, y pueden importarse y usarse directamente incluyendo la ruta del archivo .js donde se hayan escrito.

2.3.5 El espacio de trabajo

Antes de introducir el lienzo donde funcionará Blockly, es necesario indicar las rutas en la que se encuentran el script de blockly, `blockly_compressed`; así como sus bloques, `blocks_compressed`; los personalizados si hubiera alguno y el lenguaje que se prefiera, que en el caso del español es `msg/js/es.js`, para poder utilizar funciones de Blockly en el código sin que de errores.

Para poder utilizar bloques en una página, es necesario escribir un div con id “blocklyDiv” en la parte del código html en el que queramos usarlos. Blockly lo utiliza para generar un área de fondo blanco por el cual se pueden arrastrar y colocar bloques, pero no añade estos mismos.

Para disponer de los bloques en el espacio de trabajo, hace falta incluir una toolbox en formato xml. La toolbox es el menú que aparece a la izquierda del canvas desde el cual un usuario puede seleccionar bloques y colocarlos en él. Cada bloque se incluye individualmente en el xml con un elemento `<block type="nombre"></block>`, donde `type` es el campo del bloque mencionado en el punto anterior. En el caso de que los bloques tengan espacio para poner valores u otros bloques, estos se representan dentro del `<block>` correspondiente con otros tres elementos, cada uno encapsulando al siguiente en orden: `<value>`, `<shadow>` y `<field>`. Estos son utilizados internamente por Blockly para

buscar y leer los valores de dichos valores en los bloques a la hora de traducirlos a otro lenguaje de programación, de lo cual se habla más adelante. Opcionalmente, los bloques se pueden agrupar como uno quiera dentro de categorías de la forma `<category name="nombre">`Los bloques`</category>`. De esta forma, en lugar de los bloques, la toolbox mostrará los nombres de las categorías, que se expandirán hacia dentro del canvas para mostrar los bloques si se hace click en ellos. La toolbox se debe inyectar en el espacio de trabajo a través de la función $Blockly.inject('blocklyDiv', \{toolbox :$ *elemento que tenga la toolbox*$\})$.

Finalmente, el espacio de trabajo puede ser fijo o variable. En el segundo caso, es necesario crear otro elemento de id "blocklyArea" de tamaño absoluto, de modo que su tamaño varíe en proporción a la vista del navegador, y expanda el código donde se inyecta la toolbox con el comportamiento que permite aumentar o reducir el tamaño del espacio de trabajo. Este código desplaza el centro de *blocklyDiv* en la misma cantidad que lo ha hecho *blocklyArea* al modificar el tamaño de la ventana e iguala el tamaño del primero al del segundo, en píxeles.

2.3.6 De Blockly a código

Blockly actualmente dispone de varios ficheros .js que permiten traducir los bloques colocados en el espacio de trabajo en otros 5 lenguajes de programación diferentes (dart, javascript, php, python y lua). Cada uno de estos lenguajes tiene su propia subcarpeta dentro de la carpeta *generators* con todas las traducciones para los bloques por defecto. En el caso de los bloques personalizados, se puede simplemente incluir la traducción en el mismo archivo donde se crearon los bloques en sí.

La traducción de un bloque a un lenguaje concreto se realiza mediante la operación

Blockly.*lenguaje_en_el_que_se_quiera_traducir*['nombre del bloque'] = function(block){...}

Se puede convertir el código en Blockly del espacio de trabajo a otro lenguaje con la función

Blockly.*lenguaje_en_el_que_se_quiera_traducir*.workspaceToCode(workspace)

Esta función devuelve un código en el lenguaje correspondiente en formato string.

3. Estructura de online-judge-mean

El objetivo del TFG es hacer un juez en línea en el que los usuarios programen con Blockly. La página no la hemos programado desde cero, sino que se buscaron jueces ya implementados desde los que partir y poder ampliar y añadir Blockly como lenguaje adicional.

Se hizo un estudio de los posibles jueces y de algunas herramientas de gestión de concursos. Estas últimas fueron desestimadas puesto que necesitamos un juez levantado en todo momento, sin límite de tiempo y con una interfaz intuitiva y sencilla. Por ello, nos decantamos por online-judge-mean.

3.1 Qué es online-judge-mean

Como ya mencionamos en la sección 2.2.3, online-judge-mean es la base desde la que ha partido el proyecto. Es una aplicación de juez en línea de código abierto con disponibilidad completa.

Desde la página principal (figura 3.1), se puede registrar un usuario o iniciar sesión con uno ya existente. Al hacer click en *Login* (figura 3.2a), la información se valida y se pasa al servicio de autenticación, donde comprueba si el usuario existe y permite el inicio de sesión si es el caso.

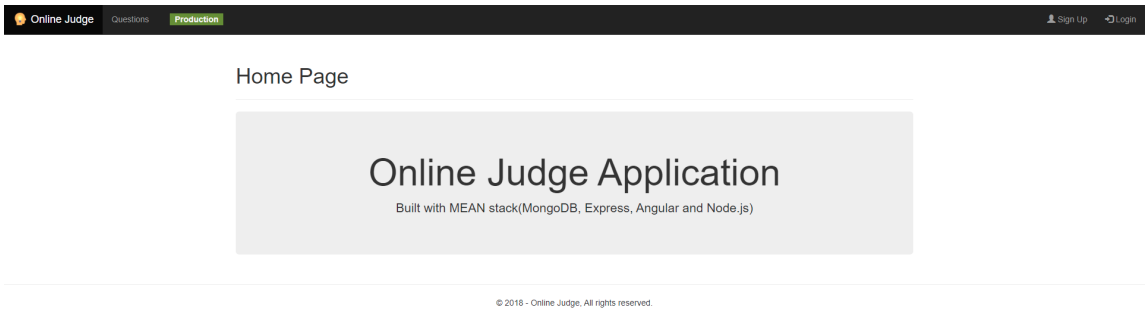
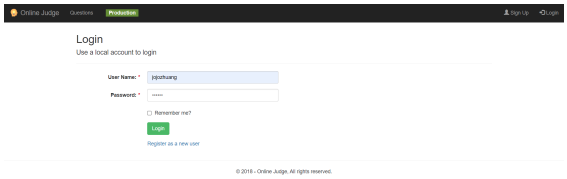


Figura 3.1: Página principal de online-judge-mean

La lista de problemas (figura 3.2b) se puede acceder desde la página principal. Para cada problema se puede distinguir:

- Un número para indicar el orden de aparición de los problemas.
- El nivel de dificultad. Pueden tomar los valores *easy*, *medium* o *hard*.
- La frecuencia del problema.
- La valoración del problema, con valores entre 1 estrella y 5 estrellas.



(a) Página de *Login*

(b) Página de Problemas

Desde esta lista se puede acceder a cada problema de forma individual (figura 3.3). La página de un problema dispone de diferentes pestañas:

- Description: Aporta la información necesaria para poder entender y resolver el ejercicio.
- Hint: Esta pestaña tiene un elemento que se puede expandir para mostrar una pista, normalmente indicando algo que deber usar.
- Submissions: Muestra todas las entregas que el usuario ha realizado en el ejercicio concreto, indicando la fecha y hora de la entrega, el tiempo de ejecución, el lenguaje que se ha utilizado y si la respuesta fue correcta. Se puede acceder a cada entrega individualmente para visualizar el código.
- Solution: La solución del ejercicio. Se puede ver el resultado en otros lenguajes, aunque no todos los problemas tienen la solución implementada.

Debajo de estas pestañas hay un campo en el que se puede escribir la respuesta al ejercicio. Además, incluye una lista desplegable desde la que se puede indicar en cuál de los tres lenguajes soportados se quiere intentar el problema: Java, JavaScript o Python. Actualmente, este campo tiene ya la solución escrita si ha sido aportada por el desarrollador, como en el caso del ejercicio Two Sum, al final de la lista.

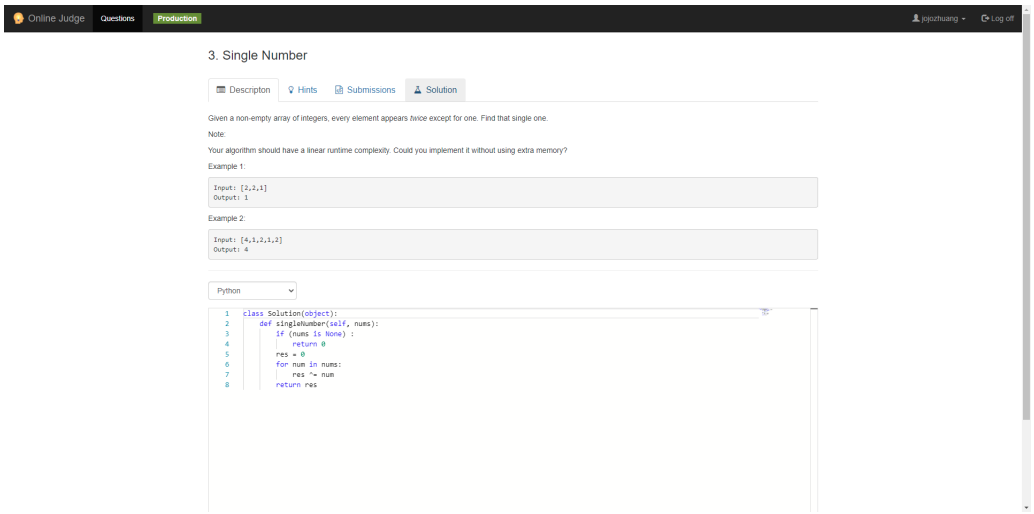


Figura 3.3: Página de un problema cualquiera en online-judge-mean

3.2 Tecnologías utilizadas

Las lenguajes de programación en los que está desarrollado son JavaScript, TypeScript, HTML y Python. La aplicación ha sido creada utilizando MEAN stack, un framework que utiliza MongoDB, Express, Angular y Node.js.

3.2.1 Lenguajes de programación

JavaScript

JavaScript (JS) es, junto con HTML y CSS uno de los principales lenguajes utilizados en la programación web. Sigue el estándar de ECMAScript, que asegura la interoperabilidad de las páginas web en diferentes navegadores.

JavaScript fue desarrollado en 1995 por Netscape, en colaboración con Sun Microsystems. El objetivo de este lenguaje era ampliar lo que se podía hacer en una página web, dado que en ese momento los buscadores solo podían mostrar documentos fijos en html. La popularidad de JavaScript llevó a Microsoft a desarrollar su propia versión, denominada JScript. En consecuencia, Netscape y Sun estandarizaron JavaScript con ayuda de ECMA, una organización internacional de estándares sin ánimo de lucro para sistemas de comunicación. Esto resultó en ECMAScript, un estándar de JavaScript que permitió que las páginas web pudieran funcionar sin problemas en diferentes navegadores. [15]

JavaScript usa un sistema de llaves similar al de otros lenguajes de programación populares como C, C++, Java o PHP. Su compilación en tiempo de ejecución permite crear páginas web interactivas y reduce el tiempo de carga de la página, al actuar desde el lado del cliente. Sin embargo, también se puede usar en el backend, o lado del servidor, como al conectar con una base de datos. Para ello conviene utilizar entornos para comunicar JavaScript con el sistema operativo, como node.js, del cual se hablará en el punto 3.2.3.

De las desventajas que tiene JavaScript, la más destacable es la dificultad a la hora de depurar. JavaScript permite acceder a elementos inexistentes, lo cual puede dar lugar a errores que no son reflejados en la página web, no mostrando nada en su lugar. Además, al ser un lenguaje interpretado, resulta muy complicado encontrar errores en líneas de código que raramente o nunca se ejecutan, como el caso de un if-else cuya condición es cierta en muchos casos.

En online-judge-mean, JavaScript se utiliza en todo el backend: la conexión y comunicación con la base de datos, la corrección de ejercicios a través del juez , los modelos...

TypeScript

Es un lenguaje de programación bajo una licencia de código abierto, es decir, se puede utilizar, modificar y redistribuir libremente. Está basado en JavaScript, al que se le han añadido definiciones

de tipos estáticas. Sin embargo, TypeScript no se puede utilizar directamente en navegadores, sino que debe compilarse en código JavaScript antes de ejecutarlo.

Es un superconjunto de JavaScript, lo cual implica grades similitudes entre ambos lenguajes. Esto permite pasar código del uno al otro sin demasiadas complicaciones y usarlo casi indistintamente. Sin embargo, aún puede haber errores si se usaron palabras en JavaScript que TypeScript reconoce como tipos.

La primera diferencia notable con JavaScript es que TypeScript usa escritura y comprobación estáticas, es decir, los tipos de cada variable se comprueban y verifican antes de ejecutar el código. En JavaScript no es necesario definir el tipo de una variable, este es en su lugar identificado durante la ejecución del código cuando se le atribuye un valor. En TypeScript, al contrario, las variables tienen un tipo que se determina al crear la variable, aunque, al estar basado en JavaScript, también está preparado para identificarlo al atribuirles un valor. Además, se realizan comprobaciones durante el proceso de compilación y se avisa de errores si los hubiera, lo cual también previene que se de el caso de rutas raramente tomadas durante su ejecución mencionada anteriormente.

Esencialmente, se pueden usar indistintamente y para los mismos fines, tanto en frontend como en backend, con la diferencia de que TypeScript resultará en un código más robusto.

Este lenguaje se utiliza en online-judge-mean para definir el comportamiento de los diferentes elementos de una página en concreto, detectar *cookies*, capturar errores al cargar la página... todo excepto el backend, que está escrito en JavaScript.

Python

Python es un lenguaje bajo una licencia de código abierto. Es uno de los lenguajes de programación de alto nivel orientado a objetos más popular de acuerdo al índice de TIOBE, donde ha subido un puesto entre Junio de 2020 y Junio de 2021 y es actualmente el segundo lenguaje de programación más usado [16]. Esto, en parte, se debe a la extensa librería de Python, que junto con la simplicidad del lenguaje, permite que los programadores puedan centrarse menos en *cómo* están escribiendo el programa o en qué librerías necesitan importar en su código y más en resolver el problema en cuestión [17].

De forma similar a JavaScript, Python, es un lenguaje interpretado y realiza comprobaciones dinámicamente, es decir, que no necesita compilarse y los errores se detectan durante la ejecución del código. Las comprobaciones las realiza según se ejecuta el código, parando la ejecución en caso de error. Esto es ventajoso a la hora de depurar porque solo muestra un error en caso de fallo.

Sin embargo, la ejecución de código en Python es más lenta en comparación con otros lenguajes, como Java o C++. Además, tiene que usar mucha más memoria que otros lenguajes. Un número puede ocupar poca memoria en otros lenguajes de programación, en Python las variables son en su

lugar objetos en el momento que se crean, necesitando mucha más memoria hasta que se les asigna un valor.

En online-judge-mean, Python se usa únicamente como uno de los posibles lenguajes a usar para resolver los ejercicios.

HTML

HyperText Markup Language o *HTML* es un lenguaje de marcado para crear páginas web. Utiliza una serie de etiquetas, *tags*, que sirven para identificar el tipo del elemento que se ha creado, ya sea texto o multimedia. Estos *tags* se pueden configurar para modificar diferentes características tales como su posición, el color de fondo o el tamaño de fuente. Sin embargo, una página web por defecto muestra el contenido de las etiquetas en orden de aparición, de arriba a abajo, si no se ha indicado su posición.

La última versión de HTML es HTML5. Esta versión incluye nuevas etiquetas que especializan la ya existente *div* para diferenciar mejor entre diferentes partes de la página tales como la cabecera, el contenido, el pie de página o links de navegación. Además, intenta separar marcaje de diseño, promoviendo CSS para el segundo, de modo que el archivo HTML solo mostraría qué elementos hay y de qué forma se combinan. Si bien aún se pueden configurar etiquetas desde el HTML, esta separación hace el código más fácil de mantener, rediseñar, optimizar y reutilizar.

3.2.2 MongoDB

MongoDB es una base de datos NoSQL basada en documentos. Utiliza archivos BSON o *Binary JSON* para almacenar la información en colecciones.

Entre sus características más destacables están:

- Soporte de consultas *Ad-hoc*: Se trata de consultas sin parametrizar que permiten acceder a los valores de la base de datos en tiempo real.
- Amplia gama de tipos de indexación: Permite crear y modificar índices en cualquier momento para acomodarlos a la evolución de una aplicación.
- Conjuntos de réplica: Un servidor principal recibe todas las operaciones de escritura y propaga esas mismas operaciones a otros servidores secundarios, replicando la información.
- *Sharding*: La división de conjuntos de datos entre varias colecciones o *shards* permite una mejor escalabilidad horizontal.
- Equilibrio de carga: gracias al *sharding* y a la réplica, MongoDB soporta equilibrio de carga a gran escala. Puede manejar varias peticiones de lectura y escritura concurrentes para el mismo dato asegurando la consistencia del mismo.

MongoDB posee además un servicio en la nube, Atlas, que permite alojar bases de datos de manera gratuita siempre y cuando su tamaño no supere los 512 MB. El servicio proporciona además una interfaz para poder consultar y manejar los datos de manera sencilla e intuitiva.

3.2.3 Node.js

Node.js es un entorno de ejecución de JavaScript multi-plataforma de código abierto. Permite utilizar código JavaScript fuera de un navegador, pudiendo usarse en el backend. Esto proporciona la oportunidad de ejecutar scripts desde el lado del servidor para generar páginas web dinámicas antes de enviar el documento html al buscador del usuario. Sus características han hecho que sea una de las plataformass más populares para el desarrollo de aplicaciones para nube o para móvil [18].

De todas las ventajas que tiene usar Node.js, las que más destacan son:

- Añadir nuevos módulos al proyecto es muy sencillo gracias a npm o *Node Package Manager*, que permite incluirlos con una linea de comando.
- Posee una arquitectura impulsada por eventos capaz de gestionar E/S asíncrona.
- Utiliza el motor V8 de Google Chrome, que, a través de un código en C++, convierte el código JavaScript en lenguaje máquina, lo que agiliza su ejecución.

3.2.4 Express

Express.js, o simplemente Express, es un framework de aplicaciones web de back-end para Node.js. Está diseñado para desarrollar aplicaciones web y APIs.

La ventaja principal de este framework es la sencillez con la cual implementa el enrutamiento, simplificando el proceso a una sola función para cada petición HTTP (GET, POST, etc). Además, cuenta con módulos, llamados *middleware*, que permiten expandir aún más las funcionalidades de éste. Algunas de estas extensiones permiten, por ejemplo, establecer las *cookies* de sesión o proteger de vulnerabilidades *CSRF*.

3.2.5 Angular

Angular es un framework de aplicación web de código abierto construida sobre TypeScript y desarrollado por Google. Está formado tanto por un framework basado en componentes, así como por una colección de librerías con una gran cantidad de características, incluyendo enrutamiento, gestión de formularios o comunicación cliente-servidor.

Los componentes son ficheros .ts que se encargan de preparar la información que se mostrará en los ficheros HTML correspondientes. Un componente contiene un decorator "@Component" donde se especifica el archivo .css, con el estilo del HTML de la página, y una plantilla HTML, o dirección a esta, que mostrará la información. Este sistema de componentes permite también separar la parte

visual, el HTML, de la funcionalidad de la página, así como expandir el vocabulario de HTML para una mejor comunicación entre ambos archivos. Cuenta también con un sistema de inyección de dependencias que evita la redundancia y que permite utilizar funciones y variables de otros archivo .ts como si fueran propias con tan solo una línea de código.

3.3 Base de Datos

Además de los problemas, la base de datos almacena también toda la información sobre lo usuarios y sus entregas. Para ello, dispone de 3 colecciones, *submissions*, *problems* y *users*.

- Questions contiene toda la información pertinente a los ejercicios. Los diferentes parámetros que utiliza son:
  - *question ID*: ID de la pregunta. Auto-genererado por MongoDB.
  - *title*: El título del ejercicio. Es el nombre que se utiliza para mostrar por pantalla.
  - *uniquename*: Se basa en el título, sustituyendo espacios por guiones, y solo usa minúsculas. Solo puede haber un ejercicio con un uniquename concreto.
  - *sequence*: El número utilizado para ordenar los ejercicios en la página donde aparecen todos los problemas. Es aportado al crear el ejercicio por el administrador.
  - *description*: Contiene el enunciado que explica el problema en cuestión, qué se espera, y algunos ejemplos, en formato HTML.
  - *difficulty*: Indica cuán difícil se estima que sea el ejercicio. Utiliza los valores 10, 20 y 30 para facil, medio o dificil respectivamente.
  - *hints*: Muestra ayuda extra en caso de ser necesaria.
  - *frequency*: Valor establecido por el administrador que intenta simular el número de intentos de un ejercicio.
  - *rating*: Al igual que *frequency*, se trata de un valor añadido por el administrador que intenta diferenciar los ejercicios dependiendo de su popularidad.
  - *Java/Javascript/Python function*: Se trata de 3 campos que almacenan las soluciones al ejercicio en cada uno de los lenguajes soportados.
  - *solution*: Este campo guarda una explicación teórica de la solución del ejercicio, indicando la complejidad.
- En users se almacenan todos los usuarios de la página web. Sus parámetros son:
  - *userID*: ID del usuario, auto-generado por MongoDB.

  - *username*: Nombre del usuario en concreto. No puede repetirse y se utiliza para buscar todas las entregas que ha realizado o las de un ejercicio en concreto.
  - *email*: Correo electrónico del usuario. Al igual que el nombre, no puede repetirse.
  - *role*: Puede ser admin o regular. Regular permite resolver los ejercicios mientras que admin, además, autoriza a crear o modificar tanto problemas como otros usuarios.
  - *hash* y *salt*: Se utilizan para almacenar la contraseña de forma segura.
- Submissions contiene todas las entregas realizadas por todos los usuarios. Incluye:
  - *submission ID*: ID de la entrega, auto-generado por MongoDB.
  - *username*: Nombre del usuario que ha realizado la entrega.
  - *questionname*: Nombre del problema en cuestión. Equivale al uniquename del problema correspondiente y se puede utilizar junto con username para buscar todas las entregas de un usuario en un ejercicio concreto.
  - *language*: Lenguaje utilizado en la entrega, puede ser Java, Python o JavaScript.
  - *solution*: Contiene la solución entregada por el usuario.
  - *status*: Indica si las respuesta es correcta o no.
  - *timesubmitted*: Contiene la fecha y hora en la que se realizó la entrega.
  - *runtime*: Indica cuánto tardó la respuesta en ejecutarse.

4. Extensión de online-judge-mean para soportar Blockly

4.1 Juez

EL juez original estaba diseñado para funcionar con 3 lenguajes, Java, Javascript y Python. Puesto que Blockly puede traducir su código a Python, el resto de lenguajes no eran necesarios, se modificó para que solo aceptara Python. También se barajó la opción de dejar Javascript solo, ya que Blockly puede traducirse a Javascript. Sin embargo, se desechó esa idea puesto que los usuarios podrían ver sus entregas pasadas tanto en Blockly como traducidas a otro lenguaje, y Python es más entendible visualmente para personas con menor experiencia.

Por otro lado, se creó un fragmento de código que envuelve el enviado por el usuario, lo cual permite la comprobación de la corrección de una entrega de manera mucho más sencilla. Esta envoltura cambia la salida estándar momentáneamente a un fichero temporal, además de proporcionar formato al código Blockly traducido a Python para su correcta ejecución. Tras ejecutarse la entrega, este fichero se compara con un archivo que contiene los resultados esperados, haciendo mucho más sencilla la tarea de crear código para testar cada ejercicio. Esta mejora reduce la longitud de los testadores muy significativamente además de su complejidad. Por último, separa el archivo que ejecuta la entrega de los casos de prueba y de los resultados esperados, haciendo más difícil que el usuario pueda acceder a ellos.

Finalmente, se prescindió de la necesidad de que el usuario tenga que leer todos los casos de prueba. Este funcionamiento es muy común en otros jueces, donde el usuario tiene que envolver todo su código con la solución en un bucle con cierta condición para leer todos los casos de prueba del ejercicio. Esa funcionalidad pasó a formar parte de la carcasa previamente mencionada, puesto que no cumple ninguna función a la hora de desarrollar el código para resolver los ejercicios, convirtiéndose más en un trámite repetitivo que en un conocimiento útil para el usuario.

4.2 BBDD

La base de datos sufrió varias modificaciones, principalmente en la estructura de los datos guardados en las colecciones, quedando éstas sin modificar.

Las entregas que se guardan en la base de datos se han modificado quitando el tiempo de actualización, pues era redundante con el tiempo de creación. Este tiempo de modificación solo difiere en unos milisegundos del tiempo de creación por lo que fue retirado. Por otro lado, se crearon tres nuevos parámetros. Los dos primeros almacenan los números de fallos y aciertos respectivamente por todos los usuarios en conjunto, que se utiliza para calcular la frecuencia de cada ejercicio en lugar de escoger un número arbitrario al crear una nueva pregunta o editar una ya existente. El tercero de ellos guarda la solución entregada por el usuario en un formato utilizable por Blockly, lo cual permite volver a intentar entregas pasadas. Esto se logra traduciendo primero el espacio de trabajo de Blockly a XML y posteriormente a texto plano. Este texto contiene tanto la meta-información de los bloques (tipo de bloque, nombre del mismo, color, etc) así como el valor y el orden que ha escrito el usuario al entregarlo. Cuando el usuario accede a la pestaña de entregas pasadas, ese código se vuelve a traducir a XML y posteriormente se inyecta en el espacio de Trabajo

Los problemas guardados también sufrieron cambios, siendo el más importante la adición de un nuevo campo que posibilita cambiar el tipo de bloques disponibles para cada ejercicio. Es una funcionalidad muy interesante ya que permite rebajar el número de bloques necesarios en los primeros problemas al mínimo. Con esta reducción se crea una primera impresión mucho más favorable para los usuarios menos avanzados, haciendo que se puedan centrar en familiarizarse con la herramienta. Otro cambio ha sido suprimir el campo donde se guardaba la solución anteriormente, ya que carece de sentido poder acceder a la solución en cualquier momento.

4.3 Problemas

La resolución de problemas ha sido la más modificada, ya que se ha cambiado el lenguaje de resolución por completo, siendo Blockly el único aceptado. Por otro lado, al ser los problemas más sencillos, la parte explicativa de los ejercicios no era necesaria. En su lugar, añadimos casos de prueba extra, que pueden ser de mayor utilidad a la hora de saber por qué no funciona una entrega.

La pestaña de soluciones también fue cambiada por completo (figura 4.1), siendo ahora utilizada para que el usuario pueda ver el resultado de su código para los casos de prueba que aparecen en el enunciado. Este cambio es interesante ya que permite tener un conocimiento más conciso de por qué una entrega no es correcta. Para reforzar esto, el usuario también podrá acceder a sus entregas pasadas para ver el código utilizado y editarlas directamente, sin tener que volver a empezar desde cero otra vez.

Por último, se han añadido varios ejercicios de dificultad variada, lo que permite a los usuarios tener siempre un reto al que enfrentarse hasta que estén listos para dar el salto a la programación con un lenguaje escrito. Esto, sumado a poder ver qué ejercicios el usuario ha intentado o resuelto, está orientado a hacer la página más atractiva para aquellos que estén empezando a programar.

Este progreso comienza con la utilización del bloque “imprimir” y el bloque de “número”. Posterior-

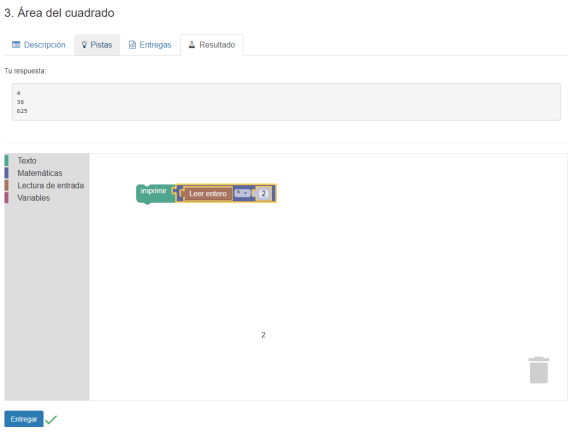


Figura 4.1: Interfaz actualizada de la pestaña de soluciones, ahora llamada Resultado

mente se introducen las variables con los bloques “establecer variable a” y “variable”, permitiendo la resolución de ejercicios más complejos. A continuación, los bloques de suma de dos operandos así como “Leer entero” y “Leer palabra” permiten resolver cualquier problema matemático o de bucles de bajo nivel. Finalmente, la manipulación de cadenas (añadirle caracteres a una palabra, leer la posición “X” de una cadena o bien obtener la longitud de una frase) se menciona brevemente como último paso.

A continuación se explicarán cada uno de los ejercicios[1]:

- 1 es soledad: Se trata del primer ejercicio al que se enfrentarán los usuarios. Este ejercicio está diseñado para familiarizarles con el entorno sobre el cuál van a trabajar en la página. También introduce el concepto más sencillo y básico de todos los problemas, imprimir por pantalla. Es una acción que tendrán que utilizar durante la resolución de todos los ejercicios y un concepto fundamental a la hora de enfrentarse a cualquier juez en línea. El objetivo de este ejercicio es imprimir el número 1 por pantalla, para lo cual sólo se dispone de el bloque de mostrar por pantalla y otro que permite utilizar números. Esto centra la dificultad de la prueba en saber manejar la herramienta.

- 2 son compañía: El segundo ejercicio se trata de una ampliación del primero, introduciendo el concepto de variables para que se empiecen a familiarizar con su uso. Las variables son una parte imprescindible en cualquier lenguaje y sea cual sea el uso de que se le dé y en los jueces en línea no es una excepción. La diferencia de este ejercicio con respecto al anterior es la necesidad de utilizar dos impresiones de números por pantalla en lugar de una, teniendo que ser ambos números iguales. El hecho de incluir las variables a pesar de no ser necesarias para la resolución correcta del problema, hace que el usuario desarrolle la buena costumbre de utilizarlas para crear un código más limpio, claro y legible.

- El área del cuadrado: Éste tercer ejercicio se trata del primero en el que los usuarios tendrán

[1] `https://tfg-blockly.netlify.app/questions`

que hacer uso de la lectura de entrada. Con esto se introducen las tres funciones principales que tendrán que manipular para obtener el resultado esperado. Para leer estos datos disponen del bloque "Leer entero", que les permitirá adquirir los datos de entrada que les proporciona la página en forma de casos de prueba. Una vez almacenado ese valor en una variable, el usuario podrá manipularla para obtener el valor del área del cuadrado multiplicándolo por si mismo. Finalmente, solo faltará mostrar ese resultado por pantalla.

- El área del triángulo: La resolución y la complejidad de este ejercicio es muy similar al anterior. Lo único que difiere es la necesidad de utilizar 2 lecturas de entrada por cada caso de prueba, además de tener que introducir el bloque de operación matemática dentro de otro. Al igual que con el uso de las variables en el segundo ejercicio, esto último no es necesario para resolver el problema pero introduce el concepto para que el usuario pueda utilizarlo en el futuro. Una vez leídos los datos, solo hay que multiplicar ambas variables y dividirlos entre dos, mostrando el resultado por pantalla.

- 3 son multitud: Este ejercicio incluye de nuevo un concepto no visto antes en ninguno de los anteriores, los bucles. Diseñado para solamente centrarse en ellos, se han quitado todos los bloques que no son estrictamente necesarios. El usuario puede utilizar o bien un bucle "while" o bien un bucle "for". EL objetivo final del ejercicio es, utilizando cualquiera de los dos, mostrar por pantalla los números del 1 al 10.

- La madre de Jaimito: Se introduce un nuevo concepto en este ejercicio, la lectura de palabras. A pesar de que funcionalmente es muy similar al bloque de "Leer Entero", para los usuarios con poca experiencia puede suponer un reto combinar ambos conceptos. Esto, sumando a la necesidad de utilizar bucles y variables, hace que se convierta en el primer ejercicio donde el usuario tendrá que utilizar estos 3 elementos, terminando así el período de introducción a conceptos. Para resolver el problema planteado, el usuario tendrá que mostrar por pantalla tantas veces como se indique la palabra recibida.

- Multiplicar a la antigua: La complejidad de este ejercicio reside en la necesidad de leer y manipular 3 variables de entrada distintas. Esto, sumado al requerimiento añadido de tener que utilizar una cuarta variable que no se menciona en el enunciado para guardar el resultado durante cada iteración del bucle, lo convierte en un ejercicio complicado para usuarios que no tengan claros todos estos conceptos. Para poder superar correctamente esta prueba, hay que imprimir la suma de los dos segundos números tantas veces como indique el primero.

- El mundo tras el espejo: Este ejercicio se trata del último y es por tanto el más complejo. Mezcla todos los conceptos estudiados con anterioridad, además de añadir la manipulación de palabras y caracteres. Por ello, establece el límite de dificultad de los ejercicios de la página, siendo todos los problemas anteriores más sencillos. Implementar la solución en Blockly de ejercicios más complicados se convierte en una ardua tarea debido a las limitaciones del

lenguaje, haciéndolo contraproducente. Es recomendable utilizar otro lenguaje si el usuario está al nivel de poder resolver ejercicios con más dificultad que este.

4.4 Blockly

El proyecto completo ha sido cambiado para solo utilizar Blockly, siendo el único lenguaje aceptado. Este cambio es visible al resolver problemas y al ver entregas pasadas, además de todo lo mencionado previamente. Para ello, ha sido necesario crear un espacio de trabajo de Blockly donde previamente estaba el editor de texto y otro al visualizar el código de entregas pasadas, donde aparecen tanto el código en Blockly como traducido ya a Python, para que el usuario se familiarice con él.

Por otro lado, se ha hecho uso de la posibilidad de generar bloques personalizados, siendo utilizados principalmente para leer datos de entrada en los problemas. Estos bloques son traducidos a la función "readline()" de Python, y después se realiza una conversión de tipo a entero o cadena de los datos leídos dependiendo del bloque utilizado. Para ello ha sido necesario crear un nuevo archivo Javascript, llamado "lecturaDeEntrada.js" donde está la declaración de esos bloques en formato JSON, añadiéndolos a la lista de bloques disponibles. En ese mismo archivo se encuentran también las funciones que traducen los bloques una vez en el espacio de trabajo a código Python.

4.5 Misceláneo

La página ha sido traducida completamente al castellano, además de haber simplificado el lenguaje utilizado para usuarios con menores conocimientos técnicos sobre la programación. También se han quitado del sistema las opciones que o bien no son utilizadas, como los editores de código o bien no tienen sentido en una página de esta índole, como el sistema de puntuación de los problemas.

5. Fase de pruebas

Anteriormente se ha hablado del pensamiento computacional y de la importancia de su desarrollo en edades tempranas. Existen diferentes plataformas que hacen uso de programación con bloques, como se comentó en el Capítulo 2. En este proyecto se ha apostado por la enseñanza a través de problemas del tipo de los jueces en línea, en este caso, programando en Blockly. En este capítulo se ha probado el juez para demostrar si la hipótesis es cierta.

En esta fase de pruebas 18 individuos realizarán los ejercicios propuestos en el juez en línea y posteriormente se les hará un breve examen sobre Blockly y otro sobre Python (Anexo 1). Las edades y conocimientos sobre la programación de los participantes es la siguiente:

Muestra de participantes	
Edad	Conocimiento
26	Básico
58	Nulo
59	Nulo
28	Nulo
29	Básico
17	Medio
24	Nulo
53	Nulo
16	Nulo
26	Básico
24	Básico
34	Nulo
41	Nulo
16	Nulo
56	Avanzado
40	Nulo
67	Nulo
59	Nulo

Tabla de edades y conocimientos informáticos

5.1 Diseño del experimento

Se hacen todas las pruebas de manera individual de manera que ningún individuo se vea influenciado por lo que ha hecho el anterior a él o ya se conozca los ejercicios.

La prueba comenzará con una breve presentación (Anexo 1), de unos 5-10 minutos aproximadamente, en la que se introducirá Blockly y la manera de utilizarlo. Durante ese tiempo el individuo puede interrumpir y realizar todas las preguntas que crea necesarias para entender bien el concepto. También se explicará el funcionamiento del juez en cuanto a la entrada de texto y números que debe procesar durante los ejercicios.

Después de la presentación, se hará una demostración práctica del ejercicio 4 y se le dejará con la herramienta para que se familiarice con su funcionamiento.

A continuación, se dejará al individuo realizar las pruebas en orden. Durante la lectura de cada enunciado se podrán realizar preguntas y serán contestadas para que sea claro el objetivo del ejercicio. En el momento que empiece a resolverlo se comenzará a cronometrar el tiempo, y se dejará libremente al usuario realizar la prueba, independientemente de las dudas que le surjan. Hasta los 3:00 minutos del cronómetro no se contestarán las preguntas que se realicen, aunque sí quedarán apuntadas, pero a partir de esa marca sí se intervendrá para guiar un poco al usuario para que consiga desbloquearse y pueda continuar el ejercicio. Se anotarán los intentos fallidos y la causa de los mismos para ver si se repiten ciertos comportamientos en diferentes individuos. El cronómetro se pausará cuando se envíe el intento correcto o cuando el usuario desista. Si terminado un ejercicio, el usuario sigue teniendo alguna duda sobre el mismo se resolverá para que lo comprenda correctamente.

Las pruebas con los ejercicios terminan cuando el usuario decide que el nivel supera su capacidad. Entonces se procederá a una breve prueba sobre Blockly seguida de otra de Python. Estas pruebas se encuentran en el Anexo 1 después de la introducción a Blockly. En las pruebas de Blockly se pondrán casos que no se han practicado en los ejercicios del juez, tales como listas, y se utilizarán bloques que no hayan visto hasta entonces. Empieza por una dificultad sencilla, haciendo uso de condicionales pero luego se introducen las listas, los bucles y algoritmos más complejos. Durante la realización de ambas pruebas no se contestarán las dudas que surjan, aunque sí se anotarán.

Para que el usuario pueda vincular los bloques vistos en Blockly con el código en Python, se traducirán al castellano las palabras como while, for, if, else, etc. No habrá ninguna presentación previa sobre el lenguaje Python, pero aún así se pedirá interpretar el código mostrado. El objetivo de la prueba con un lenguaje que no conozca el usuario es conocer si después de un tiempo haciendo uso de la plataforma es capaz de entender un poco un lenguaje completamente desconocido para él hasta el momento. Pese a no conocer el lenguaje, sí se habrá interiorizado la estructura que siguen los programas y los elementos imperativos clásicos, aunque no se haya tenido contacto previo con

la sintaxis. De esta manera sería evidente hasta cierto punto, un aumento en su pensamiento computacional.

Por último, serán anotadas todas las observaciones que quiera realizar el usuario sobre la página, su funcionamiento, dificultad de los ejercicios y comentarios sobre las pruebas realizadas al finalizar.

5.2 Resultados

5.2.1 Ejercicios de la página web

Ejercicio 1 - 1 es soledad

La mayoría de las personas entendían que lo primero que necesitaban era un bloque que pudiera mostrar algo, "imprimir" en este caso. Limitar los tipos de bloque disponibles dependiendo del ejercicio resultó beneficioso. Aunque sabían qué necesitaban, no sabían si tenían que usar el bloque *imprimir*, aunque fuera el único bloque con acción que se les proporcionaba. Otra de las dudas que surgieron fue si lo que debían "imprimir" era un uno en texto plano o el número uno, pero, al solo haber dos bloques disponibles, dedujeron la funcionalidad de *imprimir* y que debían usar ambos bloques, y todos optaron por imprimir el número.

Con este ejercicio comprendieron que *imprimir = responder*.

Ejercicio 2 - 2 son compañía

Una de las posibles respuestas a este ejercicio es el Ejercicio 1 dos veces, pero también se les dio la opción de usar variables para asegurarse de que ambos bloques *imprimir* tenían lo mismo. La gran mayoría hizo justamente eso, y otros intentaron usar variables. Esto les supuso complicaciones, ya que no estaban acostumbrados al concepto de las variables ni a usarlas, que se aclararon al responder dudas tras pasar tres minutos.

Una duda común en ambas posibles respuestas fue si se podían repetir bloques, aunque se resolvió rápido.

Ejercicio 3 - Área del cuadrado

Este es el primer ejercicio en el que no les queda otra más que usar variables y datos de entrada, lo cual, al no estar acostumbrados a usarlos, hizo el ejercicio más complejo.

El mayor obstáculo en este ejercicio fue el uso del bloque *leer entero*. Algunos no tenían claro si se refería a leer un número entero o a leer todos los casos de prueba, mientras que otros, aunque entendieran lo que hacía, no sabían donde usarlo. Además, al ser el primer ejercicio con varios casos de prueba en el enunciado, algunos usuarios trataron de hacer una solución para cada caso en vez de hacer una solución genérica como se había mostrado en la demostración.

Algunos usuarios tardaron tan poco debido a que este fue el ejercicio mostrado antes de empezar y retuvieron la solución. Como se ve en el gráfico, no es lo común.

Ejercicio 4 - Área del triángulo

Si bien este ejercicio es como el anterior con un valor y una operación más, eso no lo hizo más fácil. Es más, el hecho de que necesitaran una multiplicación y una división solo lo hizo más difícil, puesto que la mayoría no sabían que podrían combinar bloques de operaciones matemáticas. Otros dos problemas que surgieron en menor medida fueron el orden de las operaciones y el uso de leer entero en las variables cuando hay más de un dato de entrada por caso de prueba. Las personas que se encontraron con el primer caso intentaron leer los datos del caso de prueba después de usar las variables, y en el segundo, asumieron que las variables podían almacenar más de un valor y que *leer entero* leería ambas base y altura al usarse una vez.

Sin embargo, a pesar de todos estos problemas, el diagrama muestra una mejoría en los tiempos de la mayoría de la muestra, lo que implica que el ejercicio anterior resultó muy útil a la hora de enseñar cómo utilizar variables y operaciones matemáticas en Blockly.

Ejercicio 5 - 3 son multitud

Este es el primer ejercicio en el que disponen de bloques para realizar bucles. En los dos ejercicios anteriores se utilizaban los datos de entrada, pero no se requería ningún bucle, al contrario que en este ejercicio. Si bien éstos disponen de valores por defecto para poder usarlos directamente, algunos no sabían si se podían usar o si debían modificarlos o usar bloques de matemáticas para darles valor. También hubo complicaciones a la hora entender qué hace exactamente un bucle, el hecho de que el código que contiene se repite, y usar la variable que uno de los bucles crea automáticamente. Sin embargo, si bien algunos lo resolvieron antes, la gran mayoría lo resolvió poco después de haber respondido las dudas una vez pasados los tres primeros minutos.

Ejercicio 6 - La madre de Jaimito

El mayor obstáculo en este ejercicio fue leer los datos de entrada. Ya que uno de los datos es un entero y otro una palabra, necesitan usar dos bloques distintos para leer sus valores. El problema que muchos encontraron y que les costó entender fue que el orden en el que se leen los datos importa, puesto que los datos de entrada se leen de un solo archivo de texto.

Ejercicio 7 - Multiplicar a la antigua

Muchas personas decidieron parar en este punto, reduciendo la muestra a menos de la mitad. Las dos complicaciones más comunes fueron:

- *La necesidad de un acumulador.* Todos vieron rápidamente que necesitaban variables para

los números de cada caso de prueba, pero les faltó crear una variable más para acumular la suma y poder "imprimir" el resultado de cada iteración.

- *La diferencia entre establecer un valor y añadirlo.* Aún habiendo resuelto el problema del acumulador, un problema que encontraron al usarlo fue el bloque que debían usar para ello. El bloque que permite tomar un valor y sumarlo al de una variable es *añadir "valor" a "variable"*, pero el que están todos acostumbrados a usar es *establecer "variable" a*, que en su lugar remueve el valor de una variable por completo y lo sustituye por otro, de modo que no acumula.

En la mayoría de casos el tiempo que tardaron en resolverlo era ligeramente mayor que en el ejercicio anterior, pero en otros pocos, la diferencia fue mucho mayor. Sin embargo, consiguieron resolverlo, a pesar del tiempo, lo que indica que entendieron y resolvieron los problemas mencionados.

Ejercicio 8 - El mundo tras el espejo

En este caso solo un usuario fue capaz de solucionarlo, el único usuario con conocimiento avanzado de la programación. Esto se debió principalmente a que trabaja de informático y estos conocimientos no eran nuevos para él. Tardó 428 segundos, un poco menos que en el ejercicio anterior. El resto de individuos no siguieron al ver la dificultad del ejercicio.

Tabla de valores

Valores	Participantes	Mediana	Menor Valor	Mayor Valor	Primer Cuartil	Tercer Cuartil	Itercuartílico	Outliers
Ejercicio 1	18	53.5	14	221	20	103.75	83.75	221
Ejercicio 2	18	52	23	252	41	97.75	56.75	252, 180, 151
Ejercicio 3	18	320	16	960	271.25	506	234.75	960, 767
Ejercicio 4	18	313	145	840	209.5	440.25	230.75	840, 711
Ejercicio 5	18	269	41	572	183	350.75	167.75	572
Ejercicio 6	13	546	198	982	382	671	289	982
Ejercicio 7	7	591	409	737	437	656	209	none

Diagramas de los resultados

A continuación se muestra el diagrama de cajas y bigotes de cada ejercicio. En el eje Y se muestra el tiempo (en segundos) que se ha tardado en completar el ejercicio. En el eje X se muestran cada uno de los ejercicios.

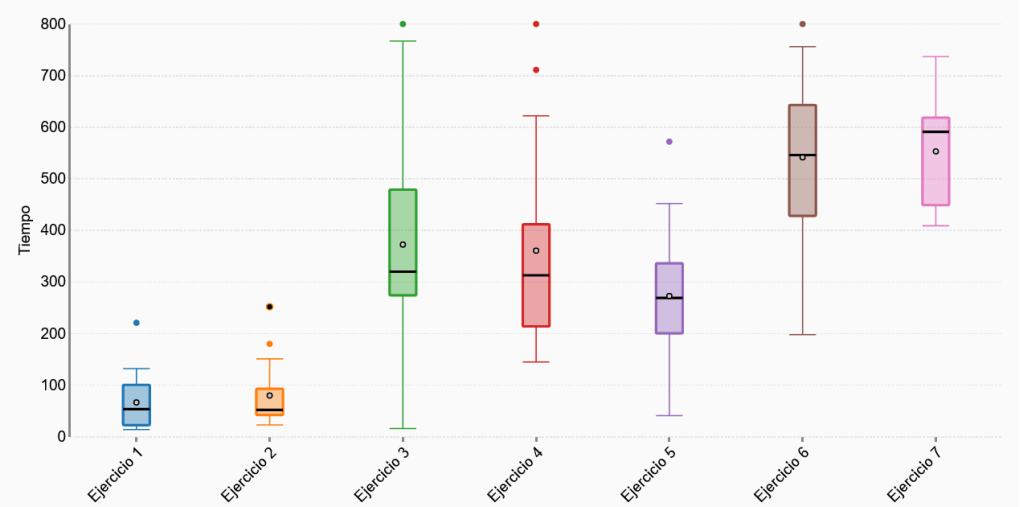


Figura 5.1: Distribución del tiempo necesario para completar cada ejercicio

5.3 Conclusiones de las pruebas

La gran cantidad de dudas y complicaciones a lo largo de las pruebas era de esperar, puesto que la mayoría de las personas de la muestra tenían conocimientos nulos de la programación (12 de los 18 individuos) y apenas rudimentarios de las tecnologías, al nivel de los *nativos digitales*. Sin embargo, todos consiguieron resolver más de la mitad de los ejercicios, algunos de ellos habiéndose quedado apenas a un paso de resolverlos todos, lo cual implica que la aplicación junto a la resolución de preguntas ayuda a comprender conceptos básicos de la programación tales como las variables e iteraciones.

5.3.1 Resultados de los test

Después de intentar resolver los ejercicios de la plataforma, los individuos se enfrentarán a un examen compuesto por 5 preguntas de Blockly y 5 de Python (Anexo 1).

Estos son los resultados de los tests realizados a los participantes:

	Resultados Test Blockly					Resultados Test Python				
Tests	1	2	3	4	5	1	2	3	4	5
Media de aciertos	0.94	0.61	0.94	0.94	0.27	0.78	0.94	0.94	0.44	0.28
Número de aciertos	17	7	15	14	1	10	17	16	5	2
Número de ayudas	1	6	3	4	6	6	1	2	5	4
Número de fallos	0	5	0	0	11	2	0	0	8	12

De acuerdo a los resultados de los tests de Blockly, se puede ver que los ejercicios de Blockly ha ayudado a comprender el funcionamiento del sistema de bloques y su comportamiento según el orden en el que se combinen, salvo por el quinto ejercicio, que ha resultado en un gran número

de fallos dada su complejidad en comparación con el resto, incluyendo acciones dentro del bucle y tanto fuera como dentro de la condicional. Así mismo, la gente que no acertó en la funcionalidad del ejercicio 2 se debe mayoritariamente a que el bloque *longitud de* pasa desapercibido al combinarse con *imprimir* dada la paleta de colores.

Sin embargo, el test de Python, puesto que no les hemos explicado cómo funciona Python ni qué significa cada palabra de sus ejercicios, muestra cómo esta serie de ejercicios y el test de Blockly han ayudado a entender conceptos de la programación que no solo tienen uso en lenguajes basados en bloques, tales como el uso de variables, iteraciones, funciones lógicas y listas. Aunque cabe decir que la gran mayoría tuvo problemas a la hora de comprender cómo Python accede a un elemento de una lista.

5.3.2 Conclusiones finales

Las pruebas con personas sin conocimientos informáticos muestra que la aplicación puede ser utilizada sin problemas por personas que están empezando a programar. Los ejercicios han ayudado tras una breve explicación a entender qué son y cómo funcionan elementos básicos que utiliza un programa, así como a hacerles capaces de leer un programa sencillo en otros lenguajes y saber de qué se encarga sin sentirse completamente perdidos. Cabe decir, sin embargo, que esta no es una herramienta que enseña sino que ayuda a aprender y comprender información que se ha aportado de forma apropiada anteriormente con la práctica. Esto se puede ver en los resultados de los tests de Blockly y Python, que, incluso después de haber pasado por siete de los ocho ejercicios de la página web y sacar las respectivas soluciones, algunas personas muestran fallos en los mismos conceptos mencionados anteriormente u otros que se han añadido en los tests, como la indexación, que, si bien no han tenido que usarlo en los ejercicios, no fueron capaces de identificarlo en el test de Python.

6. Contribuciones

Durante todo el desarrollo del proyecto, los tres miembros hemos estado en constante comunicación, tanto para el reparto de trabajo como para la toma de decisiones. Esto ha propiciado que todos estuviéramos dentro de todas las tareas en mayor o menor medida.

Guzmán Garrido Alique: Mi labor principal a lo largo del proyecto ha sido el desarrollo (tanto diseño como implementación) de funciones utilizadas en varios lugares de la página, desde la parte correspondiente a los problemas hasta cambios en la manera en la que se guardan la información en la base de datos.

- Primera instalación del sistema en local, tanto la base de datos como el cliente y el servidor para probar las primeras funcionalidades desarrolladas.
- Incorporación de Blockly, tanto en la web como las modificaciones internas necesarias para acoplarlo al juez. Esta tarea incluye la instalación de Blockly mediante npm al comienzo del proyecto, generar el código para poder ver el espacio de trabajo y añadirlo a la interfaz (tarea realizada junto a Jesús) y traducción de ese código Blockly a Python. También he sido el encargado de implementar una manera de mostrar una cantidad diferente de bloques disponibles para resolver cada ejercicio. Esto comprende los cambios a los modelos de la base de datos así como su implementación en código. Finalmente, la traducción de Blockly a XML para su guardado en la base de datos y su posterior recuperación, fue desarrollada junto a César.
- Modificación de los testeadores de los ejercicios. Tanto el diseño e implementación de los nuevos testeadores más sencillos así como los cambios necesarios para acomodarlos al sistema de juicio. Sumado a esta modificación hay que añadir también el cambio realizado al sistema para que el usuario no tenga que iterar a través de todos los casos de prueba, convirtiéndose en un proceso automático.
- Creación de los bloques personalizados utilizados para leer la entrada de los casos de prueba.
- Diseño de problemas e implementación de los mismos, principalmente aquellos de dificultad más avanzada. Esto implica por un lado la redacción de los enunciados y por otro la creación de los casos de prueba, los resultados esperados para cada caso de prueba y el testeador, así como los casos de prueba extra que se mostrarán al usuario. Por otro lado, el diseño de la curva de dificultad para que sea lo más suave posible para los usuarios, fue realizado por mí.
- Planteamiento y desarrollo de los cambios referentes a las pestañas de problemas, como los

casos de prueba adicionales o el cambio de la antigua pestaña de solución a la actual, que muestra los primeros resultados para ayudar al usuario. Esto conllevó rediseñar cómo devuelve la respuesta el sistema de juicio para que la respuesta de ciertos casos de prueba estuviera disponible.

- Diseño e implementación de los algoritmos encargados de mostrar la frecuencia de los ejercicios y el progreso del usuario en ellos.
- Limpieza de funciones no necesarias, tanto en el código, así como en la interfaz del usuario.

César Garza Sánchez: En el proyecto he ejercido tanto de desarrollador junto con Guzmán como de gestor del equipo para la organización y el flujo correcto de trabajo.

- Gestión del equipo y del proyecto. Proponer herramientas como Trello para organizar las tareas y poder llevar un seguimiento de las mismas, así como repartir las tareas a realizar y fijar fechas para cerrar secciones. Todo el flujo de correos con nuestros tutores para pedir feedback y comentar nuestros distintos problemas en las reuniones con los mismos.
- Gestión del repositorio de GitHub. Ayudando a los otros componentes a seguir el GitFlow y todo el sistema de ramas para no tener conflictos en el código desarrollado. También me he encargado de revisar que el código que se desplegaba en la rama master era correcto y estaba testado, llevando las pull requests y organizando las ramas principales del repositorio.
- Integración continua. Investigación de las posibles plataformas que se podían utilizar para que hubiese la última versión estable desplegada en todo momento. Integrar la página con las herramientas Travis CI, Heroku y Netlify, así como las variables de entorno de producción, de manera dinámica cada vez que se actualiza la rama master.
- Encapsulación del código Blockly. Cuando se realiza una entrega, el código de Blockly se transforma a Python, pero ha de ser tratado e introducido dentro de una clase además de desviar la salida estándar temporalmente a un archivo de texto para ser comparado en un futuro con la solución.
- Proporcionar conocimientos relacionados con la API de la plataforma. Ayudando a mis compañeros a comprender cómo funciona una API desarrollada en Node.js, su flujo y los endpoints.
- Recuperar entregas pasadas. Desarrollo, junto a Guzmán, de la funcionalidad que permite recuperar una entrega pasada, para poder modificarla y enviar una nueva entrega. Se realiza mediante el guardado en base de datos del código Blockly en XML y su conversión cuando se quiera editar.
- Diseño de la fase de pruebas. He sido el encargado de diseñar el diseño del experimento, tanto organizando una presentación previa como realizando los dos exámenes posteriores (Blockly y Python). También he propuesto el modo a proceder en cada prueba así como los datos que

se debían recoger de cada individuo. He llevado la organización del Excel utilizado para los resultados de las pruebas para su posterior uso en los diagramas.

- Desarrollo, junto a Guzmán, de los algoritmos y llamadas necesarias para mostrar si un ejercicio ha sido ya resuelto con el usuario que tenga la sesión iniciada.
- Estructura utilizada en Latex, junto con la propuesta del uso de la página Overleaf para poder escribir la memoria coordinadamente. Organizar la estructura de la memoria y la creación de los distintos comandos que se han reescrito Overleaf.
- Investigación y mejora del routing de la página junto a Guzmán para poder acceder a las urls de manera directa sin necesidad de interactuar con la página. Por defecto, todos los enlaces a los cuales no se haya accedido desde la propia página no son soportados.

Jesús Alejandro Sánchez Couto: A lo largo del proyecto me he encargado de desarrollar parte de la aplicación junto con Guzmán y traducir las diferentes páginas de la aplicación, así como de subir la base de datos a la nube y solventar las inconsistencias entre esta y el resto de la aplicación.

- Incorporación de Blockly en el proyecto junto con Guzmán, principalmente la interfaz para el usuario de la página web. Esto incluye eliminar la parte de la interfaz que había anteriormente para resolver problemas en Python, JavaScript o Java e incluir el espacio de trabajo que usan los usuarios para crear código en Blockly, los bloques disponibles y la visualización del código Blockly de entregas pasadas.
- Eliminación de las implementaciones de Java y JavaScript que proporcionaba Online-Judge-Mean para la resolución de ejercicios. Puesto que nuestros ejercicios se resuelven con Blockly, solo se necesita traducir la solución a un solo lenguaje para su corrección. También me he encargado de la edición de los modelos en TypeScript relevantes para que la falta de estos lenguajes como opciones no resulte en error al compilar el código.
- Edición de los textos de la aplicación que aparecen en ingles al español, tanto en páginas visibles para un usuario cualquiera como en páginas solo disponibles para administradores, así como los mensajes de aviso durante el registro o inicio de sesión o al intentar editar el email o contraseña de un usuario.
- Modificación de la página de creación y edición de un ejercicio de acuerdo a los cambios en nuestro proyecto. Esto incluye la eliminación de los lenguajes ya mencionados en el segundo punto de mis contribuciones, la solución por defecto que aporta cada ejercicio para cada lenguaje y la frecuencia, puesto que ya no es un valor arbitrario, sino que se calcula.
- Subida de la base de datos a MongoDB Atlas para el correcto funcionamiento de la aplicación una vez la subimos a Netlify y Heroku, así como la edición de los elementos de cada entrada para evitar inconsistencias a medida que se avanzaba y se realizaban cambios en el proyecto.

- Edición de la página de edición del perfil de un usuario para que no puedan cambiar su nombre. Online-Judge-Mean permite editar el nombre de un usuario, lo cual causa problemas a la hora de buscar sus respectivas entregas.

7. Conclusiones y trabajo futuro

7.1 Conclusiones

El proyecto desde un inicio fue trabajado como si se tratase de un juez en línea como los que hemos utilizado a lo largo de nuestra formación (*aceptaelreto*, *DomJudge*...). El reto consistía en poder integrar la librería de Blockly dentro de un esqueleto que ya vino proporcionado (online-judge-mean). Puesto que la licencia que tenía el repositorio del que utilizamos toda la estructura de la web era MIT, y dicha licencia no asegura el correcto funcionamiento del proyecto, la realidad fue que la mayor parte del esfuerzo estuvo en hacer que se integrase todo correctamente y adaptándolo a lo que queríamos ofrecer al usuario.

La librería de Blockly nos ofrecía una variedad de lenguajes a los que se podía convertir y terminamos decantándonos por Python. Escogimos este lenguaje principalmente por ser uno de los más sencillos de aprender y cabía la posibilidad de que los usuarios aprendiesen un poco de Python a medida que resuelven problemas en Blockly.

Una parte importante del proyecto ha sido la fase de pruebas con usuarios, puesto que el objetivo final de este es la enseñanza de la programación. El proceso ha sido duro, ya que ninguno sabíamos lo complicado que es explicar programación, aunque sea básica. Puesto que las pruebas se realizaron con el objetivo de ver cómo mejoraba el pensamiento computacional de los individuos a poco que se trabajara la programación y que supusimos que Blockly era muy intuitivo, hicimos explicaciones cortas. El problema reside en que para alguien ajeno a la programación, puede ser complicado hasta Blockly. A pesar de estos inconvenientes, se puede ver la evolución de los usuarios en el pensamiento computacional, puesto que partiendo de cero en la mayoría de los casos, eran capaces de resolver muchas preguntas de las pruebas que se planteaban después de los problemas.

Hemos focalizado en la importancia de la enseñanza de la programación desde edades tempranas para mejorar habilidades que sin la programación no serían posibles, como es el pensamiento computacional. La página web sería una buena inclusión en la educación, puesto que hace entretenido el aprendizaje y también permite al alumno practicar por su cuenta. Es una buena herramienta de soporte para los profesores y para gente ajena al centro educativo que simplemente quiera enfrentarse a ciertos retos para aprender a programar.

7.2 Trabajo futuro

En lo que respecta al futuro, hay algunos aspectos de la aplicación que nos gustaría mejorar en futuras iteraciones, como por ejemplo la cantidad de ejercicios disponibles en el sistema. Sería interesante que la gente dispusiera de más ejercicios, a fin de suavizar la curva de dificultad de estos, creando una experiencia más agradable para los usuarios. Algunos de estos problemas podrían ser similares a los que ya existen, o bien explorar campos nuevos como pueden ser el uso de estructuras de datos básicas (listas, colas o pilas).

Con respecto a la interfaz de la página, la inclusión de una nueva pestaña desde la que el usuario pudiera ver o bien sus entregas o bien todas las de la página ordenadas por la fecha de resolución. A pesar de que no se pudiera ver la solución de otras personas, sería una buena motivación ver el progreso de otros compañeros de clase. Por otro lado, añadir botones que permitan desplazarse al problema anterior o posterior, facilitaría la navegación entre ejercicios.

Otro punto a seguir desarrollando es la creación de problemas. En la actualidad, al crear un nuevo problema, es necesario generar archivos para el juez (principalmente el testador y los casos de prueba) en el sistema de archivos. Contemplamos la opción de añadir esos archivos directamente desde la página web creando una capa de abstracción, lo cuál simplificaría el proceso en el futuro.

El sistema de juicio de las preguntas podría mejorarse para que mostrara al usuario una retroalimentación mejor en caso de error. Sin embargo, al ser el objetivo de esta página gente con escasos conocimientos de programación, sería necesario ajustar los errores a un lenguaje más corriente para que todo el mundo pueda entenderlos. Para llevar a cabo este cambio, sería necesario rediseñar el método actual, puesto que esa traducción a lenguaje común no es posible actualmente.

Por último, ofrecer a los profesores la posibilidad de crear competiciones organizadas por ellos mismos para que miembros de una misma clase o centro puedan competir entre ellos a través de la página. Además, podrían ver el avance de cada alumno y organizar el concurso todo desde un mismo lugar, con todas las herramientas necesarias para la administración de la misma. Sin embargo, la carga adicional que implica este módulo no es asumible por Heroku tal y como está desplegada la aplicación actualmente, por lo que sería necesario un servidor dedicado o pasar a un plan de pago en dicha plataforma.

7. Conclusions and future work

7.1 Conclusions

We have treated the project as if it wee an online judge like the ones we have used throughout our studies at university (*aceptaelreto*, *DomJudge*...). The challenge was being able to integrate Blockly librery into a skeleton that was already provided (online-judge-mean). As the license of the repository used as the entire structure of the web was MIT, and this license does not ensure the correct performance of the project, the reality was that the most of the effort was making everything integrate correctly and adapt it to what we wanted to offer the user.

The Blockly library offered us a variety of programing languages that could be parsed to, and we ended uo choosing Python. We chose this language primarily because it is one of the easiest to learn, and there was a chance that users would learn a bit of Python as they solve the problems in Blockly.

An important part of the project has been the testing phase with users, since the final objective is the teaching of programing. The process has been hard, since non of us knew how complicated it is to explain programing, even if it is basic. As the tests were carried out with the objective of seeing how the computational thinking of the users improved as soon as they start programming and we assumed that Blockly was very intuitive, we did short explanations. The problem is that for users who are not familiarized with programing, even Blockly can be complex. Despite this drawbacks, you can see the evolution of users in computational thinking, even starting from scartch in most cases, they were able to solve many tests questions that were posed after the problems.

We have focused on the importance of teaching programming from an early age to improve skills that would not be possible without programming, sucha as computational thinking. The website would be a good inclusion in education, as it makes learning fun and also allows the student to practice on ther own. It is a good support tool for teachers and for people ourside the educational center who simply want to face certain challenges to learn programming.

7.2 Future work

There are some aspects of the application that we would like to improve in future iterations, such as the number of exercises available in the system. It would be interesting to make more exercises available to people, in order to smooth the challenge curve of the exercises, creating a more pleasant

experience for the users. Some of these tasks could be similar to those that already exist, or explore new fields such as the use of basic data structures (lists, queues or stacks).

Regarding the page interface, the inclusion of a new tab from which the user could view either their submissions or all submissions on the page sorted by resolution date. Although it would not be possible to see other people's solutions, it would be a good motivation to see the progress of other classmates. On the other hand, adding buttons to move to the previous or next task would make it easier to navigate between exercises.

Another point to be further developed is the creation of exercises. Currently, when creating a new one, it is necessary to generate files for the judge (mainly the tester and test cases) in the file system. We are considering the option of adding those files directly from the web page creating an abstraction layer, which would simplify the process in the future.

The judging system of the questions could be improved to show the user better feedback in case of error. However, as the target of this page is people with little programming knowledge, it would be necessary to adjust the errors to a more common language so that everyone can understand them. To make this change, it would be necessary to redesign the current method, since such a translation into common language is not currently possible.

Finally, offering teachers the possibility of creating competitions organized by themselves so that members of the same class or school can compete against each other through the site. In addition, they would be able to see the progress of each student and organize the competition all from one place, with all the necessary tools for the administration of the competition. However, the additional load that this module implies is not bearable by Heroku as the application is currently deployed, so it would be necessary to have a dedicated server or move to a paid plan on the platform.

Bibliografía

[1] Nativos digitales inmigrantes digitales. *On the Horizon*, 9, December 2001.

[2] Jeannette M. Wing. Computational thinking. *Communications of the ACM*, 49(3):33–35, 2006.

[3] Un análisis de la situación sobre el estado de la enseñanza de la programación en primaria y su didáctica. Technical report, Universidad Rey Juan Carlos, URJC, 2017.

[4] After the reboot: computing education in schools. Technical report, Royal Society, 2017.

[5] Young people's views on science education. Technical report, The Wellcome Trust, 2017.

[6] George Lukas. Uses of the logo programming language in undergraduate instruction. In *Proceedings of the ACM Annual Conference - Volume 2*, ACM '72, page 1130–1136, New York, NY, USA, 1972. Association for Computing Machinery.

[7] John Maloney, Mitchel Resnick, Natalie Rusk, Brian Silverman, and Evelyn Eastmond. The scratch programming language and environment. *ACM Trans. Comput. Educ.*, 10(4), November 2010.

[8] The Royal Society. After the reboot: computing education in uk schools. Technical report, The Royal Society, 2017.

[9] ACM/ICPC. Fact sheet – The 44th Annual World Finals of the ACM Internacional Collegiate Programming Contest. Technical report, ICPC, July 2020.

[10] Luca Weherst Stefano Maggiolo, Giovanni Mascellani. CMS: a Growing Grading System. *Olympiads in Informatics*, 8:123–131, 2014.

[11] Thijs Kinkhorst. DOMjudge at Amrita. Technical report, Amrita University, 2013.

[12] Competitive learning in informatics: The uva online judge experience. Technical report, Universidad de Valladolid, 2016.

[13] Marco Antonio Gómez-Martínn Pedro Pablo Gómez-Martín. ¡Acepta el reto!: juez online para docencia en español. In *Actas de las Jornadas sobre Enseñanza Universitaria de la Informática*, pages 289–296, 2017.

[14] Erik Pasternak, Rachel Fenichel, and Andrew N. Marshall. Tips for creating a block language with blockly. In *2017 IEEE Blocks and Beyond Workshop (B B)*, pages 21–24, 2017.

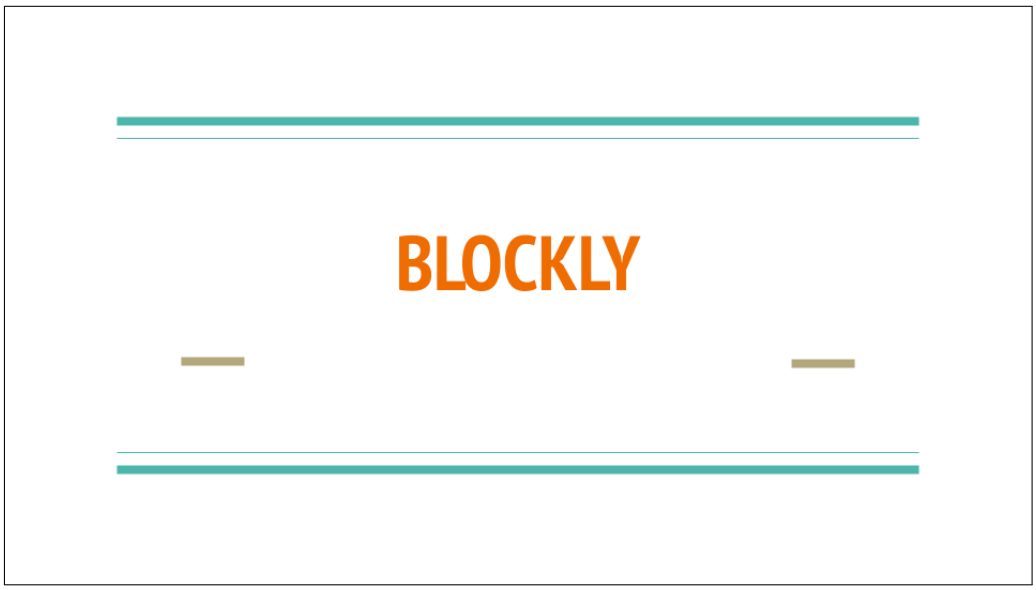
[15] Jeremy Keith. *DOM Scripting*. Friends of ED, 2005.

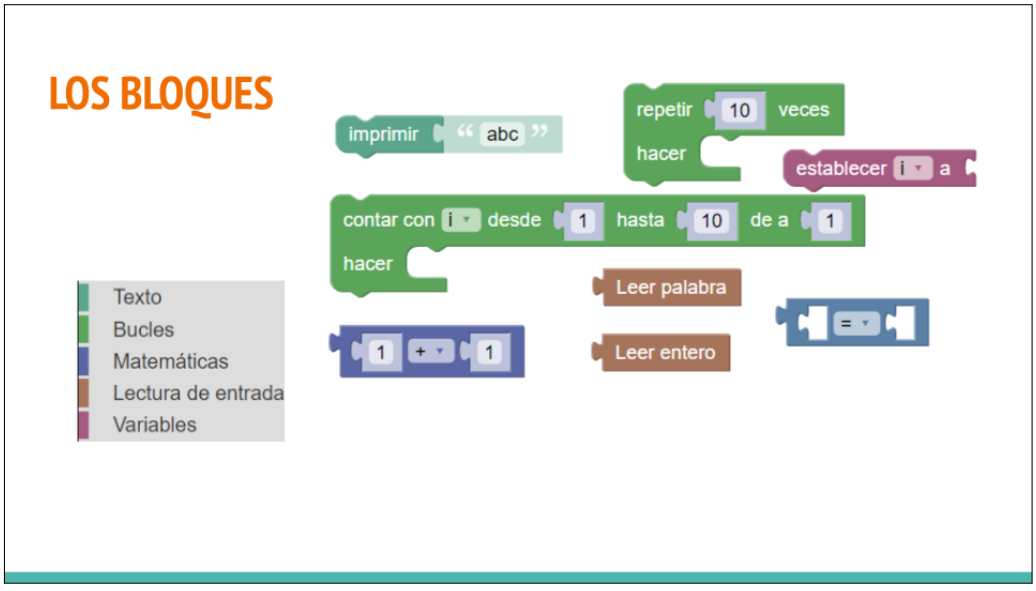
[16] TIOBE Index. June headline: Python has never been so close to position #1 before.

[17] Python to learn programming. *Journal of Physics: Conference Series*, 423, January 2013.

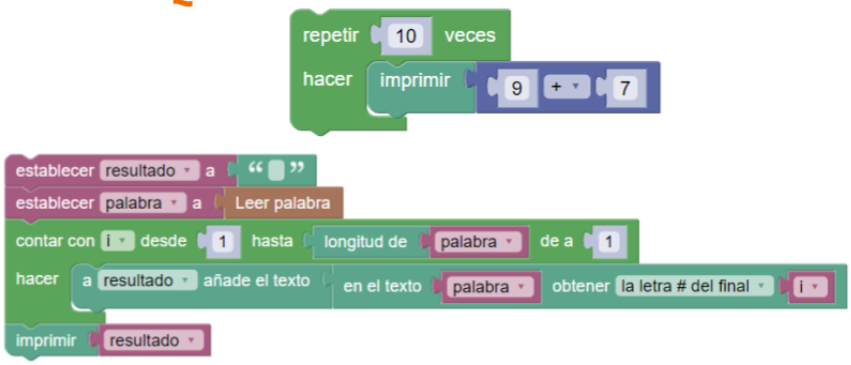
[18] Andrew Low, Joran Siu, Ivy Ho, and Gary Liu. Introduction to node.js. CASCON '14, page 283–284, USA, 2014. IBM Corp.

Anexo 1


BLOCKLY


LOS BLOQUES
imprimir
abc
repetir
10
veces
hacer
establecer
i
a
contar con
i
desde
1
hasta
10
de a
1
hacer
Leer palabra
Leer entero
1
+
1
=
Texto
Bucles
Matemáticas
Lectura de entrada
Variables

UNIR LOS BLOQUES



RESULTADOS


```
5
```


```
texto de ejemplo
texto de ejemplo
texto de ejemplo
texto de ejemplo
```


```
0
6
12
18
24
```

LEER DATOS DE ENTRADA

```
4
7
11
58
```

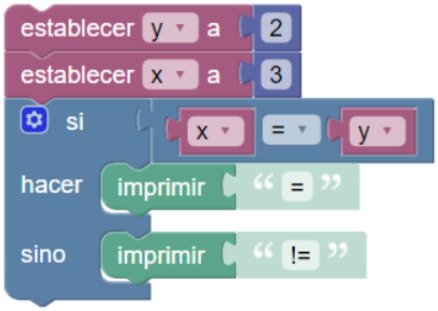

```
El número es par
El número es impar
El número es impar
El número es par
```

```
Reloj
Python
Tortilla
Madrid
```


```
R
P
T
M
```

TEST BLOCKLY

1



2

3


establecer lista de palabras a
crear lista con
Pez
Horario
Estruendoso
Tú
para cada elemento palabra en la lista lista de palabras
hacer
si
longitud de palabra > 5
hacer
imprimir
Esta palabra es larga
sino
imprimir
Esta palabra es corta

4


establecer y a 25
si
raíz cuadrada y es entero
hacer
imprimir
Es entero

5


establecer y a 5
establecer x a 0
repetir mientras y > 0
hacer
si
x es divisible por 3
hacer
imprimir x
añadir -1 a y
añadir 1 a x

TEST PYTHON

1

```
x = 0
y = 10
while x < y:
    print(x)
    x = x + 1
```

2

```
edad = 18
ingresos = 23000
if edad > 25 and ingresos > 12500:
    print('A pagar a Hacienda!')
else:
    print('Te libras de pagar!')
```

3

```
lista = ['Lunes', 'Martes', 'Miércoles', 'Jueves', 'Viernes']
for dia in lista:
    if dia == 'Viernes':
        print('Empieza el finde!')
    else:
        print('Qué bajón de día...')
```

4

```
lista_compra = ["Patatas", "Cebolla", "Sardinas", "Leche"]
x = 0;
while x < len(lista_compra):
    print(lista_compra[x])
    x = x + 1
```

5

```
a = 0
b = 1
print(a)
print(b)
for i in range(0,9):
    aux = b
    b = a + b
    a = aux
    print(b)
```


GRACIAS POR TU
COLABORACIÓN

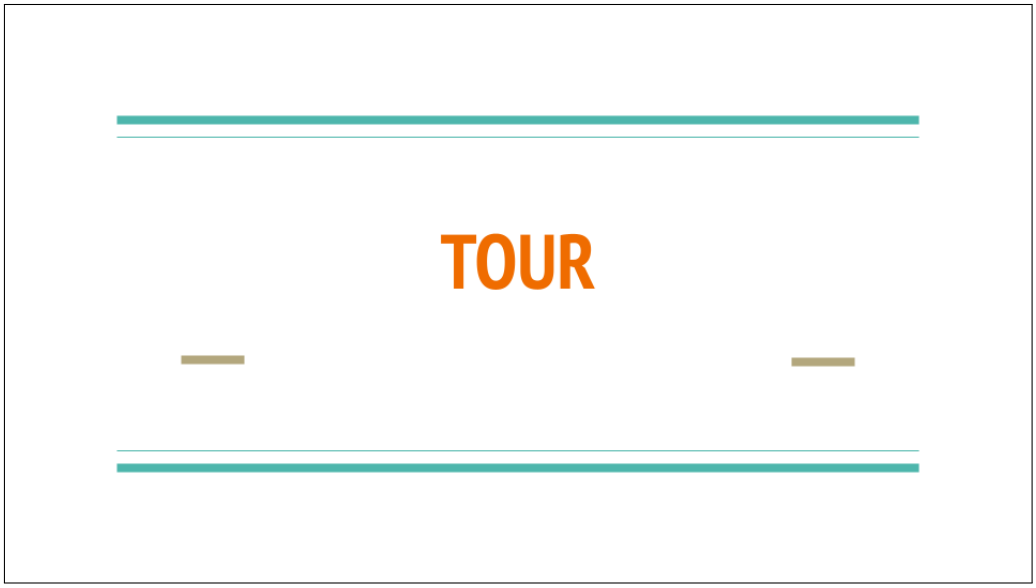
Anexo 2

Usuario normal: usuario1

Contraseña normal: usuario1

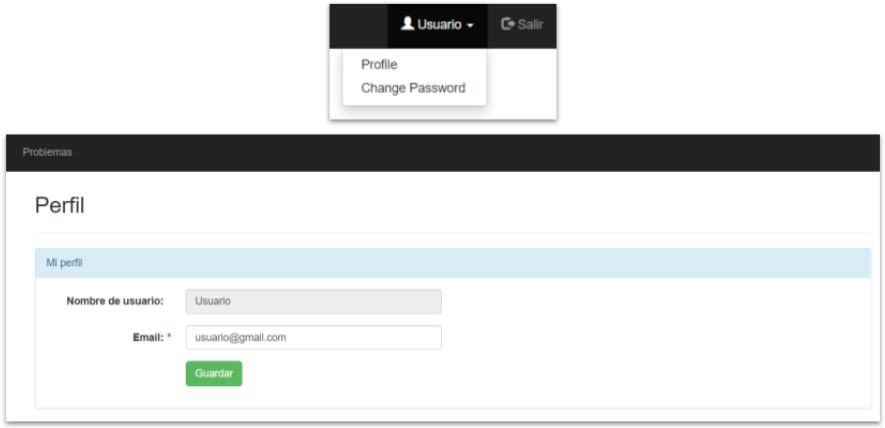
Usuario administrador: admin

Contraseña administrador: 123456



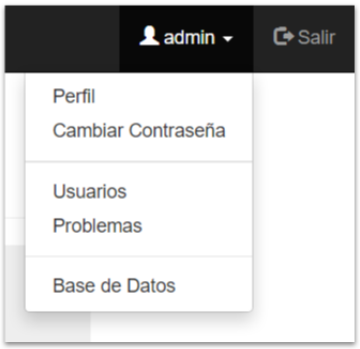
Registrarse o acceder como un usuario.

Opciones de un Usuario Cualquiera: Editar Perfil

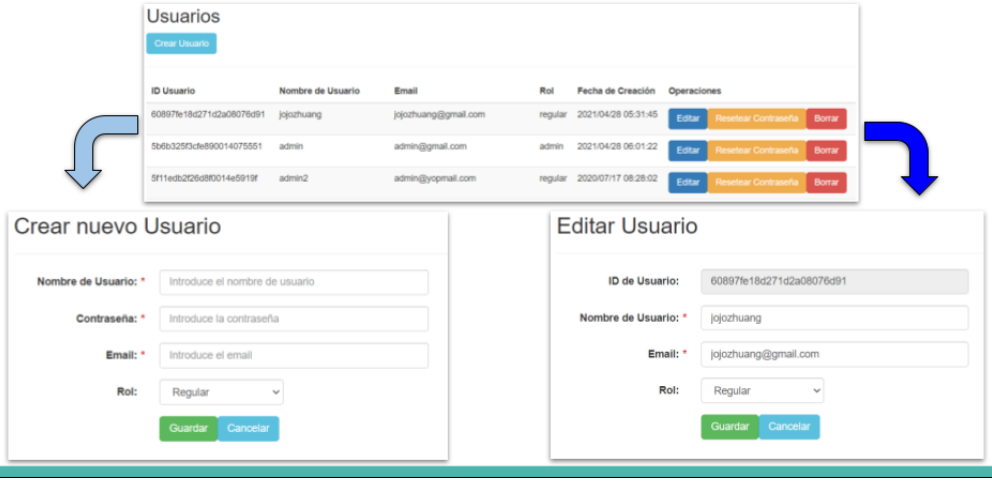


Opciones de un Usuario Cualquiera: Cambiar Contraseña

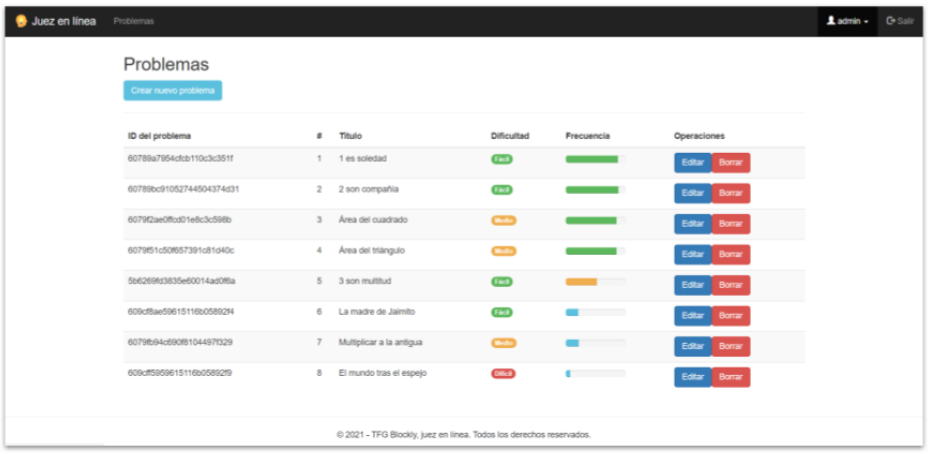
Opciones de un Administrador



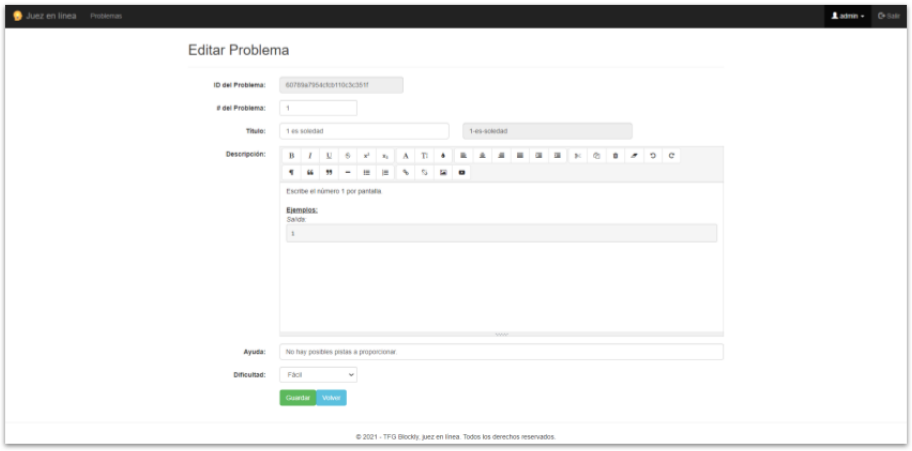
Administrador: Usuarios



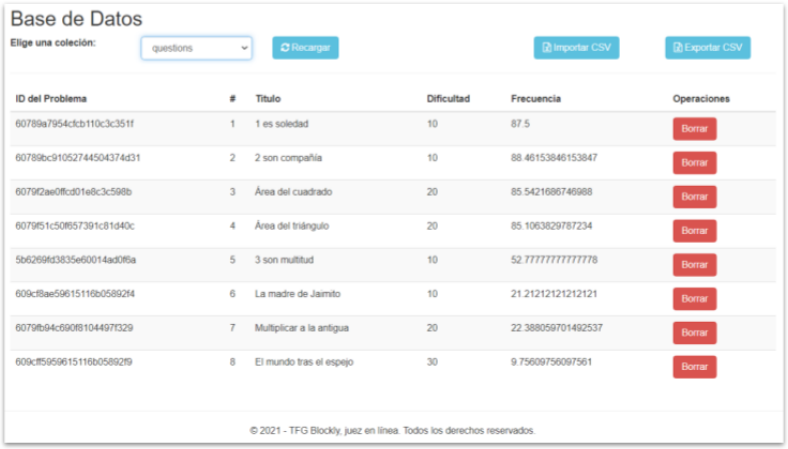
Administrador: Ejercicios



Administrador: Crear y Editar Ejercicios



Administrador: Base de Datos

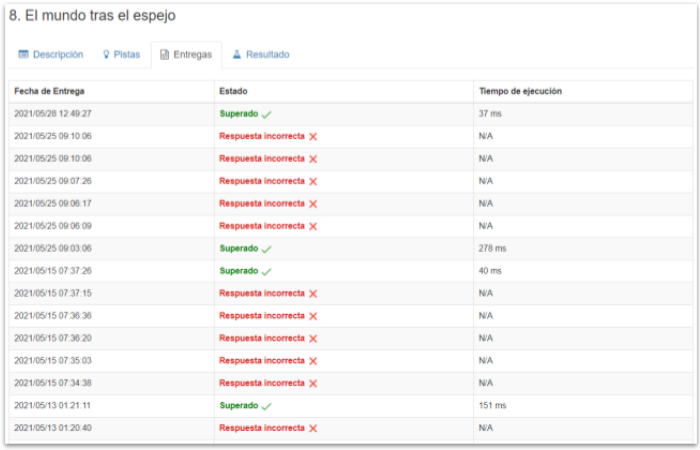


Lista de Ejercicios

Ejemplo de un Ejercicio: Enunciado

Ejemplo de un Ejercicio: Pistas

Ejemplo de un Ejercicio: Entregas



Fecha de Entrega	Estado	Tiempo de ejecución
2021/05/28 12:49:27	Superado	37 ms
2021/05/25 09:10:06	Respuesta incorrecta	N/A
2021/05/25 09:10:06	Respuesta incorrecta	N/A
2021/05/25 09:07:26	Respuesta incorrecta	N/A
2021/05/25 09:06:17	Respuesta incorrecta	N/A
2021/05/25 09:06:09	Respuesta incorrecta	N/A
2021/05/25 09:03:06	Superado	278 ms
2021/05/15 07:37:26	Superado	40 ms
2021/05/15 07:37:15	Respuesta incorrecta	N/A
2021/05/15 07:36:36	Respuesta incorrecta	N/A
2021/05/15 07:36:20	Respuesta incorrecta	N/A
2021/05/15 07:35:03	Respuesta incorrecta	N/A
2021/05/15 07:34:38	Respuesta incorrecta	N/A
2021/05/13 01:21:11	Superado	151 ms
2021/05/13 01:20:40	Respuesta incorrecta	N/A

Ejemplo de un Ejercicio: Resultado

8. El mundo tras el espejo

Descripción | Pistas | Entregas | Resultado

Aquí podrás ver el resultado de tu código para los casos de prueba que aparecen en la descripción

```
Por favor, entrega una solución válida para ver el resultado
```

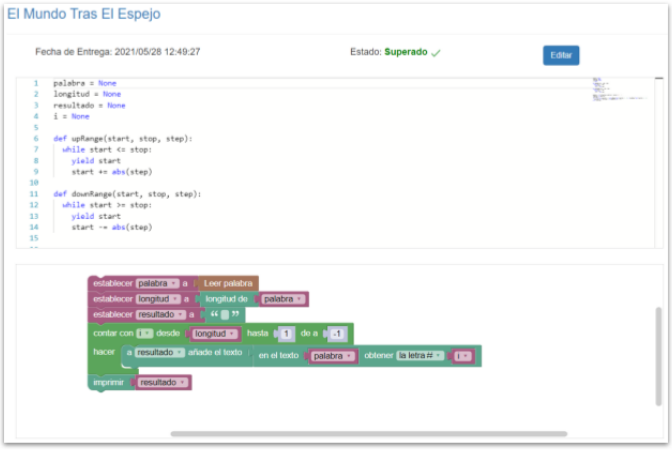
8. El mundo tras el espejo

Descripción | Pistas | Entregas | Resultado

Tu respuesta:

```
joleR
nohtyP
allitroT
dirdaM
```

Ejemplo de un Ejercicio: Entrega



Ejemplo de un Ejercicio: Blockly