UNIVERSIDAD TÉCNICA FEDERICO SANTA MARÍA
DEPARTAMENTO DE INFORMÁTICA
VALPARAÍSO - CHILE

“DESARROLLO DE UN SISTEMA ADAPTATIVO DE RECOMENDACIÓN DE EJERCICIOS DE CORRECCIÓN AUTOMÁTICA EN UN CURSO MASIVO DE INTRODUCCIÓN A LA PROGRAMACIÓN”

HUGO NICOLÁS LEYTON DÍAZ

MEMORIA PARA OPTAR AL TÍTULO DE
INGENIERO CIVIL EN INFORMÁTICA

Profesor Guía: Andrea Vásquez Guerra
Profesor Correferente: Federico Meza

Agosto - 2022

DEDICATORIA

Este trabajo es dedicado a cada una de las personas que a lo largo de todos estos años me apoyaron para que que yo pudiera llegar a esta instancia.

AGRADECIMIENTOS

Agradezco primero a Dios, porque estoy convencido que fue él quien colocó en este trayecto a las personas correctas, a las oportunidades y lecciones que me permitieron llegar hasta aquí.

Quiero agradecer a mis padres, quienes siempre confiaron en mis capacidades y me entregaron lo necesario para que estuviera lo más cómodo posible durante mi etapa universitaria. De mi Padre heredé la creatividad, mientras que de mi Madre la perseverancia, y ambos aspectos me han permitido hasta el día de hoy poner en práctica mi imaginación y perseguir siempre mis sueños.

A mis hermanas Gritsy y Constanza, con quienes siempre he podido contar incondicionalmente y sentir su apoyo a pesar de la distancia.

Agradecer a mi novia Gabriela, quien desde mi segundo año académico estuvo preocupada por mi bienestar, compartiendo lo que tenía para que yo pudiera superar ciertas necesidades. Sin su amor y compañía habría sido difícil llegar hasta aquí, pues ella me mostró la verdad de Dios para que yo pudiera confiar en él.

A mi compañero Rolando, quien conocí en el primer año de universidad. Gracias a su disposición y excelente voluntad logré en muchísimas ocasiones entender contenidos nuevos y preparar de mejor manera los certámenes.

A mis compañeros del equipo de desarrollo de SMOJ: Carlos y Alexander. Gracias a la responsabilidad que demostraron, logramos tener una plataforma robusta y que está al servicio de cerca de 1300 estudiantes cada semestre. Cada reunión que tuvimos me permitió aprender sobre la gestión de equipos y no me arrepiento de haberles propuesto participar

A distintos profesores del Departamento de Informática, comenzando por el profesor Sergio Campos, quien el año 2017 al finalizar la presentación de los proyectos de Introducción a la Ingeniería me entregó su compromiso para continuar el desarrollo de SMOJ con mi equipo. La profesora Cecilia Reyes, que en varias ocasiones me facilitó a mi y a mi equipo el financiamiento necesario para participar en actividades de programación competitiva. También agradezco infinitamente al equipo de coordinadores de la asignatura de programación: Andrea Vásquez, Federico Meza y Pedro Godoy, todos ellos siempre estuvieron comprometidos con el desarrollo y financiamiento de SMOJ, lo cual permitió que hasta el día de hoy esta plataforma siga existiendo y formando parte del plan del curso.

RESUMEN

Resumen— Los cursos masivos de programación suelen integrar herramientas tecnológicas que facilitan el aprendizaje de los estudiantes, pero la mayoría de las veces estas no cuentan con la capacidad para discernir el nivel de dificultad al que el estudiante debe estar expuesto. En esta Memoria se presenta el desarrollo de una plataforma capaz de entregar problemas de programación a estudiantes, ajustados a su nivel de progreso actual y con la finalidad de que ellos no desperdicien tiempo clasificándolos. Los resultados luego de la utilización de la plataforma por parte de los estudiantes demostraron que ellos lograron organizar de mejor manera su tiempo de práctica y que les significó un aporte formativo valioso.

Palabras Clave— Juez de programación en línea; Educación en programación; Herramienta educativa

ABSTRACT

Abstract— Massive programming courses usually integrate technological tools that facilitate student learning, but most of the time those tools do not have the ability to discern the level of difficulty to which the student must be exposed. This work presents the development of a platform capable of providing programming problems to students, adjusted to their current level of progress, so they do not waste time classifying them. The results after the use of the platform by the students showed that they managed to better organize their practice time and that it meant a valuable training contribution.

Keywords— Online programming judge; Programming education; Educational tool

GLOSARIO

API: Application Program Interface.
AWS: Amazon Web Services.
CNED: Consejo Nacional de Educación.
CSV: Comma Separated Values.
EC2: Amazon Elastic Compute Cloud.
IDE: Integrated Development Environment.
LMS: Learning Management Systems.
RDS: Relational Database Service.
SMOJ: Santa María Online Judge.
UTFSM: Universidad Técnica Federico Santa María.
UVA: Unidad Virtual de Aprendizaje.

ÍNDICE DE CONTENIDOS

RESUMEN IV

ABSTRACT IV

GLOSARIO V

ÍNDICE DE FIGURAS IX

ÍNDICE DE TABLAS X

INTRODUCCIÓN 1

CAPÍTULO 1: DEFINICIÓN DEL PROBLEMA 3
- 1.1 Contexto 3
- 1.2 Situación actual 5
- 1.3 Dificultades 6
- 1.4 Propuesta 8
- 1.5 Alternativas a la propuesta 9
- 1.6 Objetivos 10
  - 1.6.1 Objetivo General 11
  - 1.6.2 Objetivos Específicos 11

CAPÍTULO 2: MARCO CONCEPTUAL 12
- 2.1 Enseñanza general de la programación 12
- 2.2 Dificultades para aprender a programar 13
- 2.3 Importancia de la retroalimentación en el aprendizaje de la programación . . . 14
- 2.4 Ejercicios de corrección automática 15
- 2.5 Jueces en línea para retroalimentación inmediata 16
- 2.6 Jueces en línea sin capacidades para estimar el nivel de progreso 16
- 2.7 Sistema 17
- 2.8 Sistema adaptativo 17

CAPÍTULO 3: PROPUESTA DE SOLUCIÓN 18
- 3.1 Capa de datos 19
  - 3.1.1 Cuestionario 20
  - 3.1.2 Distribución 21
  - 3.1.3 Envío 22
  - 3.1.4 Insignia 23
  - 3.1.5 Problema 25
  - 3.1.6 Retroalimentación 26
  - 3.1.7 Usuario 26

3.2 Capa de negocio . . . 27
3.2.1 Responder un cuestionario para una categoría dada . . . 28
3.2.2 Recibir distribuciones. . . . 29
3.2.3 Enviar solución de código. . . . 31
3.2.4 Observar envíos pasados . . . 31
3.2.5 Recibir insignias. . . . 31
3.2.6 Observar un problema dado. . . . 32
3.2.7 Enviar *feedback*. . . . 32
3.2.8 Observar y actualizar perfil. . . . 33
3.2.9 Iniciar y cerrar sesión . . . 33
3.2.10 Observar estadísticas . . . 34
3.2.11 Observar problemas . . . 34
3.2.12 Actualizar el nivel actual de los problemas . . . 35
3.2.13 Generar distribución de problemas . . . 35
3.3 Capa de presentación . . . 36
3.3.1 Colores y fuente . . . 37
3.3.2 Enfoque *Responsive Design* . . . 37
3.3.3 Vista Cuestionario . . . 38
3.3.4 Vista Categorías . . . 39
3.3.5 Vista Problemas . . . 40
3.3.6 Vista Envíos . . . 41
3.3.7 Vista Perfil . . . 43
3.3.8 Vista Estadísticas . . . 44
3.3.9 Vista Administrador de problemas . . . 45
3.4 Arquitectura . . . 46

CAPÍTULO 4: VALIDACIÓN DE LA SOLUCIÓN . . . 48
4.1 Prototipo . . . 48
4.2 Primera habilitación de la plataforma . . . 49
4.2.1 Resultados generales . . . 49
4.2.2 Resultados de envíos realizados . . . 49
4.2.3 Resultados de insignias recibidas . . . 50
4.2.4 Resultados de distribuciones generadas . . . 51
4.2.5 Resultados de *feedbacks* entregados . . . 52
4.2.6 Resultados de cuestionarios intentados . . . 52
4.2.7 Resultados de encuesta . . . 52
4.3 Segunda habilitación de la plataforma . . . 53
4.3.1 Resultados generales . . . 54
4.3.2 Resultados de envíos realizados . . . 54
4.3.3 Resultados de insignias recibidas . . . 55
4.3.4 Resultados de distribuciones generadas . . . 56
4.3.5 Resultados de *feedbacks* entregados . . . 56
4.3.6 Resultados de cuestionarios intentados . . . 56
4.3.7 Resultados de encuesta . . . 57

4.4 Entrevista con profesores 60
4.4.1 Primera entrevista 60
4.4.2 Segunda entrevista 61

CAPÍTULO 5: CONCLUSIONES 62
5.1 Conclusiones Generales 62
5.2 Consideraciones y Limitaciones 63
5.3 Trabajo a futuro 63

REFERENCIAS BIBLIOGRÁFICAS 65

ÍNDICE DE FIGURAS

1 Unidad Virtual de Aprendizaje. 7

2 Árbol del Problema. 8

3 Diagrama simplificado del grupo Cuestionario. 20

4 Diagrama simplificado del grupo Distribución. 21

5 Diagrama simplificado del grupo Envío. 22

6 Diagrama simplificado del grupo Insignia. 23

7 Diagrama simplificado del grupo Problema. 25

8 Diagrama simplificado del grupo Retroalimentación. 26

9 Diagrama simplificado del grupo Usuario. 26

10 Diagrama de Flujo para la primera distribución. 29

11 Diagrama simplificado para las futuras distribuciones. 30

12 Vista de la página inicial de Cody. 36

13 Colores primario y secundario. 37

14 Fuente de texto integrada en Cody. 37

15 Interfaz responsiva de Cody. 38

16 Vista Cuestionario. 39

17 Vista Categorías. 39

18 Vista Listado de problemas. 40

19 Vista Problema. 41

20 Vista Editor de código. 41

21 Vista Envío de código. 42

22 Vista Lista de envíos. 43

23 Vista Perfil. 44

24 Vista Estadísticas. 45

25 Vista Administrador de problemas. 45

26 Vista Visualización de un problema en el Administrador de problemas. 46

27 Diagrama básico de Arquitectura. 47

ÍNDICE DE TABLAS

1 Distribución de estudiantes del curso de programación de la UTFSM en el segundo semestre del año 2021. 6

2 Números identificativos para cada veredicto. 23

3 Puntaje para cada pregunta de un cuestionario. 28

4 Condiciones para generación de distribuciones. 30

5 Condiciones para actualizar el nivel de un problema dado. 35

6 Resultados generales luego de la primera habilitación. 49

7 Resultados de envíos por categoría y nivel luego de la primera habilitación. . . 50

8 Resultados de veredictos luego de la primera habilitación. 50

9 Resultados de insignias luego de la primera habilitación. 51

10 Resultados de distribuciones por categoría y nivel luego de la primera habilitación. 51

11 Resultados de *feedbacks* por categoría luego de la primera habilitación. 52

12 Resultados de cuestionarios por categoría luego de la primera habilitación. . . 52

13 Preguntas para entrevista luego de la primera habilitación. 53

14 Resultados generales luego de la segunda habilitación. 54

15 Resultados de envíos por categoría y nivel luego de la segunda habilitación. . . 54

16 Resultados de veredictos luego de la segunda habilitación. 55

17 Resultados de insignias luego de la segunda habilitación. 55

18 Resultados de distribuciones por categoría y nivel luego de la segunda habilitación. 56

19 Resultados de *feedbacks* por categoría luego de la segunda habilitación. 56

20 Resultados de cuestionarios por categoría luego de la segunda habilitación. . . 57

21 Distribución acerca de la recomendación de la plataforma. 58

22 Palabras más repetidas en opiniones acerca de la usabilidad. 58

INTRODUCCIÓN

En Chile, la cantidad de matrículas de estudiantes de primer año en carreras de tecnología dentro de instituciones de educación superior ha crecido en promedio un 1.3 % cada año durante el periodo 2018-2022, de acuerdo a lo informado en [MIFUTURO, 2022]. Si para el mismo periodo consideramos únicamente carreras relacionadas con la informática, ya sean profesionales o técnicas, el promedio en el aumento anual ha sido de un 13.2 % a partir de los registros contenidos en la base de datos de índices de matrículas en pregrado del Consejo Nacional de Educación[1] (CNED) [CNED, 2022]. Con respecto a la heterogeneidad y utilizando la misma base de datos del CNED, la cantidad de matrículas de hombres ha crecido un 12.5 %, mientras que para las mujeres un 14.2 % y un 35.9 % que proviene de extranjeros.

Es de conocimiento general que las carreras ligadas a la informática imparten en su plan de estudio un curso de introducción a la programación, eso quiere decir que la cantidad de estudiantes inscritos en algún curso como el antes mencionado será directamente proporcional a la cantidad de matrículas que se contabilicen año tras año. También se conoce que tales cursos suelen implicar una exigencia considerable, debido a que los estudiantes requieren de la capacidad para entender e implementar un algoritmo y así resolver cada problema que se les presenta [Gomes y Mendes, 2010].

De acuerdo a un estudio realizado en la *Helsinki University of Technology*[2] [Kinnunen y Malmi, 2006], una de las dos principales razones para la deserción en los cursos de introducción a la programación tiene que ver con la cantidad de tiempo que los estudiantes tienen para su atención, ya sea porque ellos prefieren realizar otra actividad, no invierten suficiente dedicación o porque algún contenido del curso les exigió mucho más tiempo de lo que esperaban. La otra razón reside en la falta de motivación. Los autores concluyen que la intervención en solo algunas de las diferentes dificultades que se presentan para ambas razones no será suficiente para apoyar a los estudiantes.

Los cursos que introducen la programación y que además son masivos a menudo utilizan herramientas tecnológicas que permiten a los estudiantes complementar su estudio poniendo en práctica lo aprendido. El Departamento de Informática de la Universidad Técnica Federico Santa María[3] (UTFSM), pone a disposición de los estudiantes que cursan la asignatura de Programación una plataforma que permite resolver problemas, donde la evaluación de la solución es revisada de manera automática.

La plataforma antes mencionada despliega un conjunto de problemas, pero estos no están clasificados por dificultad. Si lo anterior fuera posible, los estudiantes no perderían tanto tiempo intentando encontrar el problema más adecuado para el nivel de progreso que llevan en ese momento.

[1] `https://www.cned.cl/`
[2] `https://www.aalto.fi/en`
[3] `https://usm.cl/`

En esta memoria se presenta una propuesta para el desarrollo de una nueva plataforma para un curso masivo de programación. Dicha plataforma debe poseer la capacidad de revisar soluciones de manera automática y además contar con autonomía para decidir los problemas que cada estudiante debe intentar, considerando el nivel de progreso que estos llevan en su curso formal.

El presente documento se divide en 5 grandes capítulos.

El primer capítulo consiste en la Definición del Problema, donde se describe el contexto respectivo, así como también lo que se propone, la situación actual y los objetivos que se definieron.

El segundo capítulo expone el Marco Conceptual. En esta sección se describe la base conceptual sobre la que está fundamentada la solución planteada.

La Propuesta de Solución corresponde al tercer capítulo y detalla el desarrollo de la solución propuesta. Se incluye metodología y diagramas que son necesarios para una mejor comprensión de la estrategia seguida.

El cuarto capítulo corresponde a la Validación de la Solución. En este se valida la solución propuesta, demostrando que la implementación fue válida en el entorno donde fue planteada. La validación descrita en este capítulo es apoyada por encuestas y entrevistas.

El quinto y último capítulo consiste en las Conclusiones. Aquí se reflejan los aprendizajes del trabajo realizado, así como también las consideraciones, alcances y el trabajo futuro para aquellos que se sientan motivados por el tema y deseen profundizarlo.

CAPÍTULO 1
DEFINICIÓN DEL PROBLEMA

1.1. Contexto

La gestión del tiempo juega un rol fundamental en la vida de cualquier estudiante universitario, pues continuamente está siendo evaluado y debido a eso requiere una concentración y preparación frecuente. Un estudio [Nasrullah_PhD y Khan_PhD, 2015] llevado acabo en la *Qurtuba University of Science and Technology*[4] demostró que los estudiantes universitarios que gestionan correctamente su tiempo presentan una correlación positiva con respecto a su rendimiento académico. Según el estudio, los estudiantes más aventajados son aquellos que planifican.

Para cada estudiante, la transición entre la enseñanza media y superior trae consigo muchos cambios significativos [Fromme *et al.*, 2008], pues cada estudiante debe encontrar la manera óptima de organizarse y estudiar. Durante la etapa escolar, algunos estudiantes acostumbran a tener hábitos que los despreocupan en cierto grado para atender sus responsabilidades estudiantiles, pero es en la etapa universitaria cuando se dan cuenta de un sin fin de diferencias que existen entre ambas etapas y si son realmente conscientes, entonces podrán encontrar la manera de corregir tales hábitos y lograr así mejores resultados académicos.

La procrastinación es una de las causas que genera la pésima gestión del tiempo. [Yuangga y Sunarsi, 2018] desarrollaron un estudio que tuvo como propósito determinar los efectos de los hábitos de procrastinación y la baja administración del tiempo con respecto a la autoeficacia de los estudiantes. Los resultados demostraron que existe una influencia simultánea entre los hábitos de procrastinación y la baja gestión del tiempo en la autoeficacia de los estudiantes.

Otra causa atribuible al hecho de que los estudiantes no gestionan correctamente su tiempo tiene que ver con la planificación que ellos realizan. Según [Reyes-González *et al.*, 2022], los estudiantes con alto y bajo desempeño se diferencian de acuerdo a las estrategias de planificación y priorización del tiempo que emplean.

Con respecto a los efectos que puede desencadenar la falta de habilidad para gestionar el tiempo en los estudiantes universitarios, [Marcén y Martínez-Caraballo, 2012] afirman que la relación entre el tiempo que dedican y los aprendizajes que alcanzan varía entre estudiantes y esto depende del aprovechamiento de los contenidos y de las aspiraciones que tengan. Los autores también indican que el esfuerzo que dediquen los estudiantes estará orientado hacia la superación de las asignaturas y de sus aspiraciones, por lo que si las calificaciones que obtienen son positivas, se espera que ellos respondan de manera diferente para gestionar su tiempo y el grado de aprovechamiento mencionado.

[4] `https://www.qurtuba.edu.pk/`

Otro de los efectos implicados en la gestión del tiempo corresponde al decrecimiento del rendimiento de los estudiantes a medida que la gestión también disminuye. Así lo confirma la investigación de [Cavero, 2011], donde los autores estudiaron entre otras variables el rendimiento académico resultante. Según el análisis realizado, aspectos de aprendizaje como la regulación del esfuerzo y la buena gestión del tiempo demostraron ser significativos con respecto a la variable mencionada.

Cuando los estudiantes universitarios no se responsabilizan por su avance, se desmotivan gradualmente a tal punto que pueden llegar a tomar decisiones que impliquen desligarse de la asignatura que están cursando. Un buen método de planificación, permitirá que los estudiantes logren insertarse de mejor manera durante los primeros meses en la universidad y se sentirán más seguros al cursar asignaturas donde inicialmente estaban predispuestos [Kraiger *et al.*, 1995].

En Chile, las instituciones de educación superior que ofrecen carreras de ingeniería suelen impartir un curso de introducción a la programación dentro del primer año. Antes de comenzar el curso, habrán estudiantes que ya tienen alguna noción básica acerca de la programación o quizás ya dominan muy bien los contenidos [Alexandron *et al.*, 2012]. Sin embargo, también existirá otro grupo de estudiantes que nunca antes ha tenido alguna experiencia programando o quizás solo han escuchado alguna que otra referencia.

Una asignatura de programación se caracteriza por requerir muchas horas de estudio para entenderla de manera teórica [Atiq, 2018], ya que muy pocos estudiantes han estudiado programación antes de ingresar a la educación superior. Si bien la parte teórica de un curso promedio de programación ya es desafiante para algunos estudiantes, lo es aún más la parte práctica, pues para un mismo ejercicio, pueden llegar a existir una infinidad de maneras de resolverlo. Esto conlleva a que cada estudiante deba dedicar más tiempo a practicar que en comparación con estudiar, porque es posible que la teoría sea levemente extensa, pero podrían haber infinitas maneras de aplicarla.

Acerca de la tasa de aprobación y reprobación que suelen tener los cursos masivos de introducción a la programación, algunos estudios como [Luna *et al.*, 2007] y [Feierherd *et al.*, 2001] no demostraron resultados tan alentadores como se podría esperar, incluso el segundo de ellos menciona una reducción de la asistencia a medida que se realizaron las actividades. En cuanto a conclusiones, el primer estudio propone el aprovechamiento de los recursos docentes, tales como sus roles y tareas asignadas. El segundo estudio en cambio, hace hincapié en que el desarrollo de talleres optativos es un recurso que puede tomarse como medida para superar la separación entre la teoría y la práctica y fortalecer la integración entre docente y alumno.

1.2. Situación actual

En la Universidad Técnica Federico Santa María, el curso de programación se imparte en sus tres campus y dos sedes. En el curso se enseña el lenguaje de programación Python[5] y cuenta con 10 unidades pedagógicas, donde se abordan contenidos básicos y como resultado, permite al estudiante desarrollar el pensamiento algorítmico.

El curso de programación de la UTFSM cuenta adicionalmente con una plataforma tecnológica llamada Santa María Online Judge[6] (SMOJ) que permite la corrección de ejercicios de manera automática. Los estudiantes tienen a su disposición enunciados de ejercicios que son previamente elaborados por los profesores encargados, así, los estudiantes pueden desarrollar la solución del ejercicio dentro de la misma plataforma y luego enviarla para que sea juzgada con distintos casos de prueba, para finalmente entregar un veredicto que indique si la solución logró acertar completamente o si falló en uno o más casos de prueba. Cabe mencionar que SMOJ fue desarrollada en el curso de Introducción a la Ingeniería de la UTFSM y uno de sus tres fundadores es el autor de esta memoria.

Cada una o dos semanas en el curso de programación de la UTFSM se realiza un laboratorio virtual en la plataforma SMOJ para que los estudiantes cuenten con entre 6 y 8 ejercicios. El objetivo es que los estudiantes practiquen los contenidos semanales de una manera asíncrona y que les permita ir mejorando sus habilidades técnicas a lo largo del semestre.

Como se mencionó anteriormente, las soluciones son juzgadas de manera automática y el estudiante recibe un veredicto entre 15 y 30 segundos. Además, se despliega el detalle para cada caso de prueba y en algunos casos es posible que se indique algún *feedback*, con la intención de entregar orientación al estudiante para que vuelva a pensar en la estrategia que está empleando para resolver el problema.

Los ejercicios que son seleccionados para cada laboratorio tienen una complejidad variable, pero todos concuerdan con la unidad semanal que se está estudiando. En SMOJ no se indica que complejidad tiene cada ejercicio y esto es una gran desventaja, ya que los estudiantes suelen comenzar por el primer ejercicio de la lista y puede que se presenten con un ejercicio demasiado complicado para el nivel que presentan en ese momento. El hecho de que los estudiantes inicialmente estén expuestos a un ejercicio con una alta complejidad puede implicar que se desmotiven y que por lo tanto posterguen su estudio. Es posible que también los estudiantes decidan cambiar por otro ejercicio que lo clasifiquen como más adecuado y que dejen parcialmente el que estaban intentando resolver. En el caso de que acierten con la solución, ellos podrían sentirse satisfechos y no continuar practicando, pues ya habrán cumplido con su hábito de estudio. El problema es que ellos se sentirán en su zona de confort y podrían no presentar motivación para intentar resolver algún ejercicio de una complejidad más elevada para así continuar progresando con los contenidos vistos.

[5] `https://www.python.org/`
[6] `https://smoj.inf.utfsm.cl/`

Se evidencia que los actores involucrados en la situación actual contempla tanto a los estudiantes que cursan la asignatura de programación como los profesores quienes dictan cada curso. Por una parte, los estudiantes tienen la intención de practicar y mejorar sus habilidades a través de la plataforma SMOJ, en cambio, los profesores esperan conocer el rendimiento general de su curso o específicamente de un estudiante para que así puedan adecuar de mejor manera sus clases o para conocer en qué unidades los estudiantes tienen más dificultades y así reforzarlas.

1.3. Dificultades

El curso de programación de la UTFSM cuenta cada semestre con cerca de 1300 estudiantes y 38 paralelos, considerando sus 3 campus y 2 sedes. La clasificación de los estudiantes del segundo semestre del año 2021 se detalla a continuación.

Tabla 1: Distribución de estudiantes del curso de programación de la UTFSM en el segundo semestre del año 2021.

Fuente: Elaboración Propia.

Campus ó sede	Paralelos	Estudiantes
Campus Casa Central	16	659
Campus San Joaquín	15	503
Campus Vitacura	5	172
Sede Viña del Mar y Sede Concepción	1	34

El total de estudiantes para el segundo semestre del año 2021 fue de 1368, mientras que el total de paralelos fue de 37.

Actualmente, la asignatura de programación está dividida en una cantidad limitada de unidades denominadas Unidades Virtuales de Aprendizaje (UVA). Cada UVA considera una serie de actividades destinadas al estudiante, tales como la asistencia a las sesiones clase, la resolución de una tarea, etcétera. El espacio que ocupa SMOJ dentro de una UVA consiste en la participación del estudiante para la resolución de ejercicios de corrección automática, considerando una dedicación de 60 minutos. En la siguiente figura se observa la estructura de una UVA.

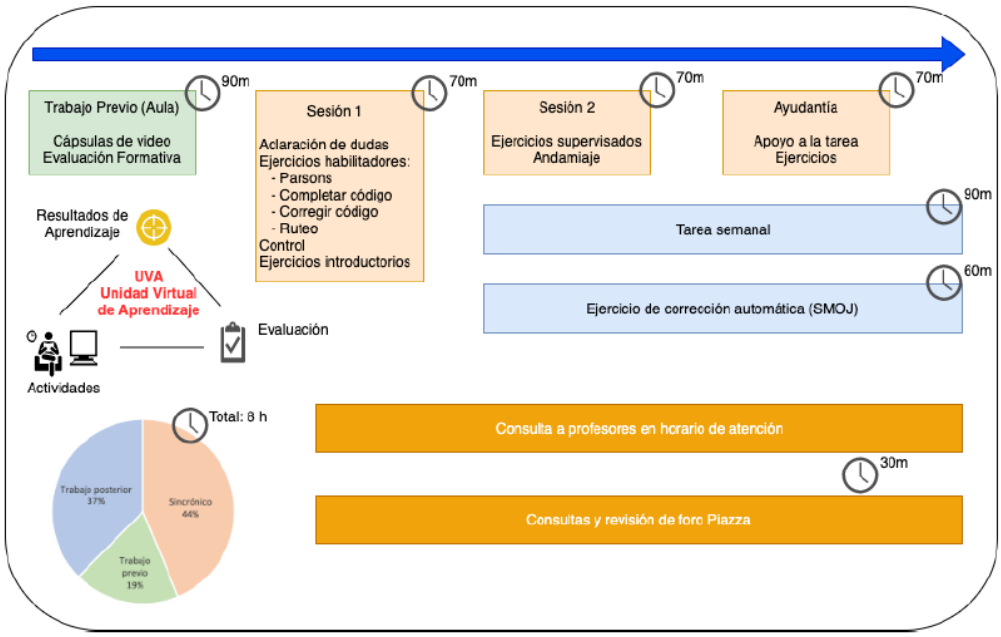

Figura 1: Unidad Virtual de Aprendizaje.
Fuente: Adaptado desde [Vásquez *et al.*, 2021].

Debido a la magnitud de estudiantes que cada semestre cursan la asignatura de programación, ésta es coordinada por un profesor o profesora para cada campus y sede. Lo anterior no significa que los laboratorios sean distintos para cada campus o sede, sino que todos reciben los mismos enunciados para cada laboratorio que se realiza.

Como se ha mencionado antes, SMOJ no clasifica los ejercicios de acuerdo a su complejidad, esto conlleva a que los estudiantes se encuentran con la dificultad para gestionar el tiempo de estudio que ellos disponen, pues no hay una manera personalizada para que sepan de antemano que ejercicios tienen que ir resolviendo de manera progresiva.

Si lo anterior no es solucionado en el corto o mediano plazo, una gran parte de los estudiantes continuará con un desfavorable método para gestionar el tiempo de estudio que ellos tienen para dedicarle a la asignatura, ya que no podrán aprovechar las instancias de estudio que disponen. Una implicancia de lo anteriormente mencionado es el bajo rendimiento académico que se podría desencadenar y sumado a eso, la deserción de la asignatura o lo que es aún peor, la deserción de la universidad.

Como puede verse a continuación en el árbol del problema, se identifica como problema principal la gestión del tiempo que los estudiantes dedican a un curso masivo de programación. Una de las causas que genera el problema mencionado es la transición entre la enseñanza media y superior, ya que de manera generalizada lo que suele ocurrir es que los estudiantes necesitan desarrollar un cambio de paradigma profundo, pero diversos factores ya mencionados previamente afectan el logro de este propósito. Cuando eso ocurre, los es-

tudiantes no reconocen la manera óptima de gestionar su tiempo de estudio y por lo tanto esto puede generar una disminución en las aspiraciones iniciales o la falta de motivación para estudiar. Lo anterior es crítico, ya que para un curso masivo de programación es posible que la tasa de retención de estudiantes disminuya considerablemente.

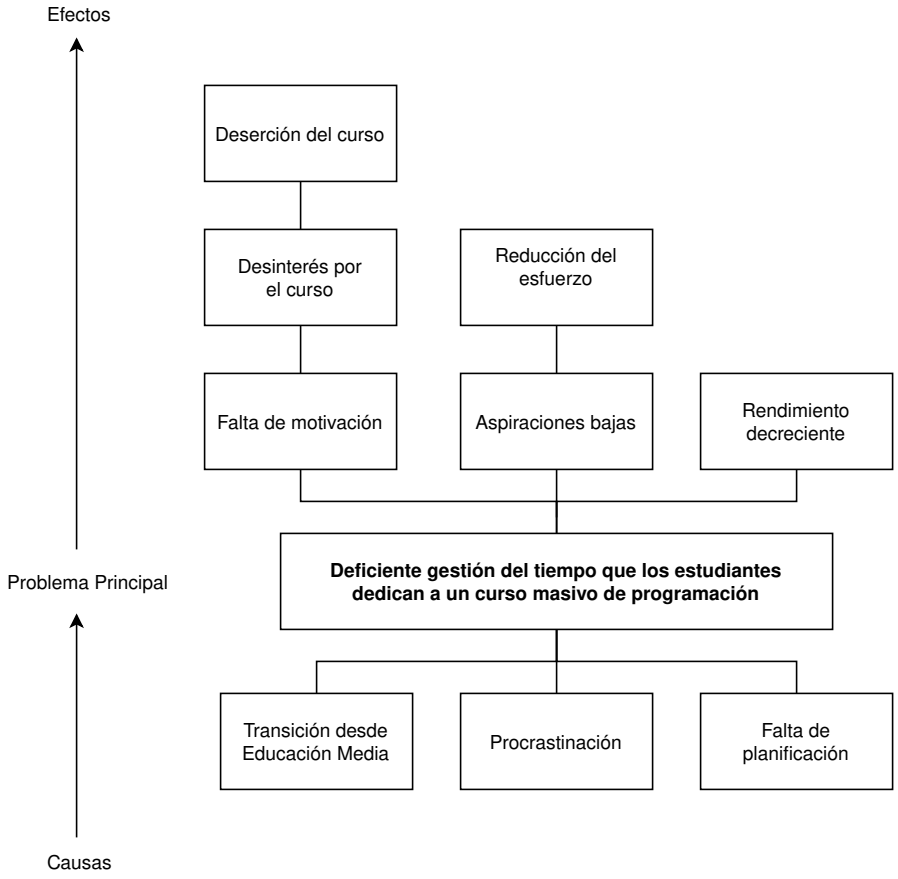

Figura 2: Árbol del Problema.
Fuente: Elaboración Propia.

1.4. Propuesta

Para solucionar la problemática mencionada, se propone el desarrollo de un sistema que permita a los estudiantes resolver ejercicios adecuados a su nivel para que gestionen su tiempo de práctica de una manera óptima, ya que así no perderán tiempo intentando primero los ejercicios que posiblemente sean muy difíciles.

Para ajustar los ejercicios a los perfiles de los estudiantes, se tienen que considerar ciertos parámetros, uno de ellos será la dificultad para cada ejercicio que se encuentra disponible en SMOJ. Esta clasificación estará basada en tres niveles: básico, medio y avanzado. Cada

estudiante que haya resuelto algún ejercicio, tendrá permitido asignar un nivel de dificultad, para que así el problema se caracterice a medida que más estudiantes resuelven el mismo ejercicio. Los docentes también tendrán permitido asignar la dificultad para cada ejercicio y en ese caso se considerará como una asignación mucho más importante, ya que se asume que los docentes son quienes tienen más experiencia juzgando ejercicios de acuerdo a su complejidad. Otro parámetro a considerar será la cantidad de envíos correctos e incorrectos que un estudiante ha realizado para un problema dado, ya que ese valor determinará si el estudiante está o no dominando el contenido.

El impacto que generará la implementación del sistema propuesto será positivo para los estudiantes de un curso masivo de programación, ya que no tendrán que preocuparse por seleccionar ejercicios de una manera manual e identificarlos para clasificarlos de acuerdo al avance que lleven registrado. La contraparte tiene que ver con el estudiante que no tiene dificultades con alguna unidad, en ese caso, el sistema proveerá ejercicios aún más desafiantes para que éste pueda practicar.

1.5. Alternativas a la propuesta

Las siguientes herramientas representan una alternativa siempre y cuando de manera manual algún profesor asigne los ejercicios a los estudiantes de acuerdo a su nivel de avance en el curso. Cada una de las comparaciones se realizan con respecto a SMOJ debido a que es el juez oficial del curso.

HackerRank[7] es un juez en línea que suele ser usado tanto por compañías de tecnologías como por desarrolladores. Para propósitos de práctica, permite crear problemas y sus respectivos casos de pruebas, luego se puede crear un concurso e insertar los problemas creados. La ventaja que tiene HackerRank con respecto a SMOJ, es que HackerRank posee una interfaz mucho más usable e intuitiva para el usuario. Una de las desventajas que presenta HackerRank en comparación a SMOJ, es que los estudiantes en HackerRank tienen que tener asociado un nombre de usuario de acuerdo a un formato convenido previamente, ya que cuando se descargan las calificaciones, es necesario realizar un cruce con la información que se maneja de los estudiantes en el curso.

OmegaUp[8] es un juez en línea de origen mexicano que permite administrar cursos y cuentas de usuarios, además de crear problemas y concursos. La ventaja que tiene OmegaUp con respecto a SMOJ, es que OmegaUp posee un gestor de cursos que puede ser administrado por un profesor, en cambio, SMOJ no tiene esta característica y por lo tanto los cursos tienen que crearse de forma manual desde la administración de la plataforma. La desventaja que presenta OmegaUp en comparación a SMOJ, es que OmegaUp no restringe la creación de nuevos problemas para la plataforma, esto quiere decir que cualquier usuario en OmegaUp

[7] `https://www.hackerrank.com/`
[8] `https://omegaup.com/`

puede crear un problema y quedará público, por lo tanto no hay una revisión previa por parte de algún administrador o profesor que pueda revisar previamente el enunciado y los casos de prueba. SMOJ sólo permite que profesores con un permiso especial puedan crear los problemas y asignarlos a los laboratorios que correspondan.

Mimir Classroom[9] es una plataforma que provee herramientas a profesores para enseñar eficientemente ciencias de la computación sin comprometer la calidad de la educación de los estudiantes. Mimir permite elaborar tareas de programación para que los estudiantes posteriormente sean evaluados, también cuenta con un gestor de cursos y de estudiantes que es utilizada para la organización de alguna escuela o universidad. Una de las ventajas que tiene Mimir con respecto a SMOJ, es que Mimir permite detectar plagio, por lo tanto, es capaz de determinar similitudes de código entre dos o más estudiantes. Otra de las ventajas, es que Mimir cuenta con la integración de un IDE, para que los estudiantes no tengan la necesidad de descargar nada en sus computadores y así trabajar todo desde la misma plataforma. La posibilidad de elaborar tareas que integren archivos con otras extensiones también es una ventaja con la que Mimir cuenta, esto es muy frecuente cuando se elaboran tareas que deban leer archivos de texto o CSV. A pesar de que SMOJ no cuenta actualmente con ninguna de las ventajas que Mimir tiene, es posible integrarlas al mediano plazo debido a que se cuentan con proyectos *open source* que permitan realizar las mismas funciones.

A pesar de todo lo anteriormente descrito, ninguna de las plataformas mencionadas está completamente adaptada para las unidades que se imparten dentro del curso formal de programación, por lo tanto, no resulta una buena idea intentar migrar a una alternativa que pudiera parecer mejor. SMOJ tiene la gran ventaja de ser una plataforma completamente personalizada, pues debido a que es un proyecto desarrollado desde cero y por estudiantes de la Universidad, esto permite ir agregando nuevas funcionalidades que sean requeridas a medida que transcurre el curso.

Con respecto a la recomendación de ejercicios de manera personalizada para cada estudiante, tampoco las tres herramientas mencionadas anteriormente ofrecen esa posibilidad, pero existen otras plataformas que se encargan de clasificar los ejercicios de algunos jueces. Tales plataformas clasifican los problemas de acuerdo a los tópicos asociados que tiene cada problema y así como también la complejidad que representan.

1.6. Objetivos

Tanto el objetivo general como los objetivos específicos se indican a continuación:

[9] `https://www.mimirhq.com/`

1.6.1. Objetivo General

Desarrollar un sistema que permita a los estudiantes de un curso masivo de introducción a la programación practicar con ejercicios de corrección automática ajustados a su nivel de progreso en el curso.

1.6.2. Objetivos Específicos

- Analizar cómo los estudiantes de un curso masivo de introducción a la programación gestionan su tiempo para aprender los contenidos del curso y practicar ejercicios de corrección automática.
- Diseñar el sistema propuesto, definiendo un modelo de datos y tecnologías adecuadas para la implementación.
- Evaluar cuál es la percepción que genera el sistema propuesto en los estudiantes de un curso masivo de introducción a la programación.

CAPÍTULO 2
MARCO CONCEPTUAL

2.1. Enseñanza general de la programación

Distintos enfoques se han popularizado en las últimas décadas para conocer con precisión la manera adecuada acerca de cómo enseñar programación. Antiguamente, la programación fue vista como un medio para que los científicos, matemáticos e ingenieros hicieran que las computadoras ejecutaran instrucciones predefinidas. Sin embargo, [Fincher, 1999] en su estudio confirma que lo anterior es pasado y que la perspectiva actual es ver a la programación como una habilidad, y por eso mismo se ha estudiado los enfoques para enseñarla apropiadamente.

En el estudio mencionado se analizan 4 enfoques: *syntax-free*, *Literacy*, *problem-solving* y *Computation as Interaction*. Cada uno de ellos se describe en los párrafos siguientes.

El primer enfoque está basado en la enseñanza sin considerar la puesta en práctica de los contenidos, pues su propósito es enseñar sin tener un cuenta el desarrollo de código. [Shackelford, 1998] con respecto a la implementación de este enfoque menciona que todos los estudiantes deberían estar expuestos a los principios básicos del pensamiento algorítmico, donde también aporta con 3 objetivos que deberían cumplirse una vez que el enfoque ha sido aplicado: proporcionar una introducción al campo, entregar contenido teórico y habilidades de desarrollo de *software* y por último, preparar a los estudiantes para la programación.

El enfoque de *Literacy* al igual que el anterior, se encuentra exento de la implementación de código. La diferencia reside en que este enfoque se centra en el concepto de abstracción, tomando como referencia el hecho de que los niños desde una edad temprana comienzan a aprender notaciones abstractas con el fin de aprender el significado, como por ejemplo el alfabeto, la escritura, la lectura, etcétera. La manera de trabajar en este enfoque consiste en prestar atención a aspectos como alcanzabilidad, motivación y relevancia.

El tercer enfoque emplea las estrategias de analizar y diseñar, pero al igual que los dos enfoques previos, este no contempla la práctica. Este enfoque no se considera pedagógicamente útil porque el estudiante suele asumir lo que quiere hacer, pero en realidad podría querer hacer más. Otros motivos tienen que ver con que la vía para aprender es la misma que se mencionó en los enfoques anteriores y que se asume inicialmente que en realidad el estudiante no sabe cómo resolver problemas.

El enfoque de *Computation as Interaction* es muy distinto a los antes ya mencionados. En este caso, el contenido se trabaja bajo el paradigma de programación orientada a objetos, con la intención de que los estudiantes asimilen lo que están aprendiendo a través de la

observación de su entorno, ya que los objetos que los rodean pueden modelarse con dicho paradigma. Este enfoque también se destaca por la popularidad de la programación orientada a objetos.

Si bien el estudio que incluyó a los 4 enfoques revisados anteriormente se publicó hace aproximadamente 2 décadas atrás, nuevas maneras de enseñar programación se han investigado y aportado a la comunidad. Uno de ellos es la investigación de [Matthíasdóttir, 2006], en la cual se explica que adicionando a un curso formal sesiones de laboratorio, clases grabadas, pruebas en línea y trabajo individual se podría tener éxito.

Las sesiones de laboratorio consistieron en la resolución de ejercicios mediante un computador, por lo que la práctica se hizo presente. Por otra parte, las clases grabadas trató acerca de la exposición de la teoría y con la ventaja de que los estudiantes pudieran revisarla la cantidad de veces que quisieran. Las pruebas en línea fueron evaluaciones simples y de corta duración con el propósito de mantener a los estudiantes activos durante el periodo académico. Por último, el trabajo individual se desarrolló durante las clases, para que los estudiantes fueran asistidos de manera personal cuando se encontraran desarrollando alguna actividad.

2.2. Dificultades para aprender a programar

Diversos estudios se han realizado para explicar las distintas dificultades que tienen los estudiantes a la hora de aprender a programar. Una de ellas es la investigación de [Derus y Ali, 2012] que tuvo como objetivo conocer las dificultades presentadas por parte de los estudiantes mientras cursaban una asignatura de fundamentos de programación. La metodología que emplearon fue la realización de una encuesta para que pudieran obtener información acerca de los estudiantes, la experiencia previa que tenían y su opinión acerca del curso.

Los resultados de la encuesta realizada demostraron que los estudiantes tuvieron un nivel de comprensión moderado. También se encontró que los contenidos con mayor dificultad fueron aquellos que tienen un concepto abstracto, como por ejemplo la manera en la que una variable registra internamente un valor dentro de la memoria del computador, sobre todo en estructuras como una matriz multidimensional, o en la declaración de bucles o funciones. Por otro lado, la mayoría de los estudiantes estuvieron de acuerdo en el beneficio que les significó las actividades prácticas, como las de tipo laboratorio. Con respecto a los profesores, se identificó que aquellos que no suministraban suficientes ejemplos demostrativos dificultaban el aprendizaje. Los autores finalmente proponen la utilización de herramientas de visualización como una alternativa para entender la programación de una manera práctica, considerando que los estudiantes estuvieron de acuerdo con dicha medida.

Otro estudio realizado fue el de [Milne y Rowe, 2002], que a diferencia del anterior, en este estudio se consideraron tanto las opiniones de los estudiantes como la de los profesores. La metodología utilizada fue la misma que el estudio anterior, en este caso la encuesta estu-

vo compuesta por 3 partes: la primera consideró preguntas acerca de experiencia previa en sistemas operativos y lenguajes de programación, la segunda presentó 28 tópicos de programación para conocer el nivel de conocimiento para cada uno de ellos, y por último, la tercera parte incluyó un espacio para aportar retroalimentación.

Los resultados acerca de la primera parte tuvieron como primer y segundo lugar a los lenguajes de programación Java y C++ con 95 % y 90 % respectivamente. En cuanto a la segunda parte, se llega al misma conclusión que la investigación anterior, es decir, los estudiantes carecen de comprensión para entender lo que sucede en la memoria del computador mientras los programas se ejecutan. Para la tercera parte, la investigación no incluye qué tipo de retroalimentaciones se obtuvieron.

2.3. Importancia de la retroalimentación en el aprendizaje de la programación

La retroalimentación enfocada en el ámbito del aprendizaje de la programación ha significado que distintas investigaciones sean efectuadas para analizar su efectividad. En el caso de [Hao *et al.*, 2022], los autores buscaron entender cómo los estudiantes interactúan con la retroalimentación automatizada y cómo la perciben. El método empleado consistió en la entrega de varios tipos de retroalimentación automatizada a distintos grupos de estudiantes con respecto a los resultados de sus tareas de programación.

Cada estudiante que finalizó su tarea la envió a través de *GitHub*[10] para que posteriormente otras dos herramientas continuaran el flujo, donde una de ellas realizaba *testing* sobre la tarea y la otra entregaba la retroalimentación de manera automática una vez que finalizaba la herramienta anterior. En base a [Narciss y Huth, 2004] se utilizaron los siguientes 3 tipos de retroalimentación:

- Conocimiento de resultados (CR), es este caso, información acerca de si algún caso de prueba estuvo correcto o no.
- Conocimiento de respuestas correctas (CRC), es este caso, comparación entre la salida generada y esperada.
- Retroalimentación elaborada (RE), es este caso, sugerencia de errores típicos cometidos por otros estudiantes en los tres últimos trimestres.

No todos los estudiantes recibieron el mismo tipo de retroalimentación, la asignación fue la siguiente:

- Grupo 1 de 25 estudiantes recibió la retroalimentación de tipo CR.

[10] `https://github.com/`

- Grupo 2 de 25 estudiantes recibió la retroalimentación de tipo CR y CRC.
- Grupo 3 de 26 estudiantes recibió la retroalimentación de tipo CR, CRC y RE.

Para 3 tareas realizadas, los resultados demostraron que los estudiantes del grupo 3 lograron un mejor rendimiento que los grupos 1 y 2, así como también el grupo 2 que fue superior con respecto al grupo 1. La percepción de la retroalimentación en tanto, fue evaluada realizando una encuesta hacia los estudiantes, en la cual se preguntó acerca de qué tanto se aprovechó la retroalimentación, así como también que describieran su experiencia, entre otros.

El estudio finaliza concluyendo que la retroalimentación fue completamente acertada y en el caso de que se hubiera incluido un único tipo de retroalimentación, probablemente no habría tenido la misma efectividad y también podría haber llevado a que los estudiantes optaran por buscar fallos bajo el enfoque de prueba y error.

2.4. Ejercicios de corrección automática

Los ejercicios de corrección automática, como su nombre lo indica, son aquellos que no necesitan una intervención manual para ser corregidos, donde esta intervención podría venir por parte de algún profesor. En cambio, la evaluación de la solución puede estar dada por algún sistema que mantenga automatizada esa tarea.

[Hegarty-Kelly y Mooney, 2021] realizaron un análisis acerca de los sistema de calificación automática considerando a estudiantes de primer año de universidad. Una de las ventajas que mencionan es el hecho de que tales sistemas son útiles para garantizar la equidad de las calificaciones y así como también proveer comentarios oportunos acerca del trabajo hecho.

Para analizar lo mencionado previamente, los investigadores habilitaron un sistema de calificación automática que incluyera los contenidos de programación que los estudiantes se encontraban aprendiendo. Se demostró que el sistema resultó fácil de usar para los estudiantes y sobretodo para aquellos que nunca antes habían estudiado programación. Cabe mencionar que la plataforma aparte de entregar una calificación automática, también proporcionó retroalimentación.

El estudio concluye destacando lo beneficioso que puede resultar el uso de las plataformas de calificación automática, no solo para estudiantes, sino también para profesores, pues ellos tienen la posibilidad de proporcionar comentarios acerca del progreso y realizar un seguimiento del mismo para identificar dónde se encuentran las brechas de conocimiento.

2.5. Jueces en línea para retroalimentación inmediata

Acerca del significado de juez en línea, [Zhou *et al.*, 2018] aporta la siguiente definición: *"un sistema JO (Juez en línea) es un software web para evaluar programas con pruebas de caja negra"*. En Ingeniería de *Software*, las pruebas de caja negra son aquellas que no toman en cuenta la estructura del código y la manera en cómo está implementado, así que los jueces en línea más bien enfocan las restricciones de ejecución hacia el tiempo y memoria que consumen los programas.

El artículo de [Wasik *et al.*, 2018] llamado *"A Survey on Online Judge Systems and Their Applications"* señala que los sistemas de jueces en línea tienen las siguientes aplicaciones:

- Programación Competitiva.
- Educación.
- Compiladores en línea.
- Reclutamiento.
- Servicios de minería de datos.
- Desarrollo de plataformas.

Si centramos nuestro enfoque en la educación, el estudio de [Hidalgo-Céspedes *et al.*, 2020] indica que los jueces ofrecen una serie de ventajas, como mayor objetividad, mayor rigurosidad en la evaluación gracias al conjunto de pruebas y mejores oportunidades de aprendizaje a través de la retroalimentación inmediata.

2.6. Jueces en línea sin capacidades para estimar el nivel de progreso

En la literatura no son muchas las investigaciones centradas en la estimación de los niveles de progreso que llevan los estudiantes mientras utilizan un juez en línea. Sin embargo, una investigación reciente de [Xu *et al.*, 2020] demostró cómo el nivel de los estudiantes puede predecirse utilizando el método *Exploratory Factor Analysis* y recibiendo como entrada los registros históricos que son generados por los mismos estudiantes.

Para la investigación, se tomó en cuenta la participación de 1043 estudiantes matriculados en un curso de introducción a la programación en C. Los datos fueron recopilados en forma de *logs* y el siguiente paso fue determinar una estructura conveniente que permitiera conocer variables candidatas para así incluirlas en la implementación del método *Exploratory Factor Analysis*. Parte de las variables que resultaron candidatas fueron: número de envíos, número de problemas resueltos y número de veces que un estudiante inició sesión dentro del juez.

Posteriormente, se estableció un modelo y se verificó su efectividad utilizando una red neuronal. Los resultados demostraron que el modelo ideal incluyó variables como: la cantidad de problemas simples que los estudiantes resuelven correctamente, la cantidad de problemas complejos que los estudiantes resuelven correctamente, la frecuencia con la que los estudiantes visitan el juez en línea y la cantidad de veces que los estudiantes piden ayuda.

Dentro de las conclusiones, los autores instan a futuras investigaciones a avanzar en la exploración de información, más que en las características que los jueces en línea podrían entregar.

2.7. Sistema

[Johnson, 2021] realiza un breve estudio sobre el concepto de sistema. Entre sus aportes encuentra la definición de parte de [Dictionary, 2022] como *"un grupo de elementos que interactúan regularmente o son interdependientes y forman un todo unificado"*.

Es indispensable que un sistema cuente con tres tipos de componentes: entradas, procesos y salidas. Las entradas son los datos e información que el sistema necesita consumir. Los procesos en cambio, son aquellas acciones que toman las entradas y que en base a eso generan un resultado esperado, dicho de otra manera, la salida del sistema.

2.8. Sistema adaptativo

Según [Institute, 2022], *"un sistema adaptativo (o un sistema adaptativo complejo, CAS) es un sistema que cambia su comportamiento en respuesta a su entorno"*.

Los sistemas que se adaptan continuamente lo hacen para ajustarse a los objetivos, tareas, intereses y otras características pertenecientes a un usuario o grupo de usuarios [González *et al.*, 2008]. De acuerdo a lo anterior, los sistemas adaptan dinámicamente su conducta en base a la interacción entre el usuario y el sistema.

CAPÍTULO 3
PROPUESTA DE SOLUCIÓN

Como se ha mencionado anteriormente, SMOJ es la plataforma que utilizan los estudiantes que cursan la asignatura de programación en la UTFSM. La principal funcionalidad que provee esta plataforma es el juzgamiento de código de manera automática en base a distintos casos de prueba. Cada caso de prueba incluye una entrada y una salida esperada, entonces, cuando el código es juzgado con las respectivas entradas, cada salida generada por este es comparada con cada una de las salidas esperadas. Si todas las comparaciones efectuadas presentan un 100 % de similitud, entonces la plataforma garantiza que el código desarrollado se encuentra correctamente implementado.

A pesar de que SMOJ proporciona los enunciados de los problemas para que estos sean intentados y resueltos por los estudiantes, esta plataforma carece de la capacidad para determinar aquellos problemas que mejor se ajustan al nivel de progreso que llevan los estudiantes. Esto último se traduce en una desventaja, ya que los mismos estudiantes tienen que seleccionar cuál problema deberían intentar primero o cuáles son aquellos que les podría aportar significativamente, teniendo como consecuencia directa una falta en la gestión del tiempo que destinan a practicar.

En una primera instancia, se pensó en integrar el sistema propuesto dentro de SMOJ para que estudiantes y profesores encontraran todo en un mismo lugar, pero debido a que SMOJ en aspectos técnicos no representaba una ventaja para el sistema, se decidió por desarrollarlo de manera independiente con sus propias tecnologías. Este sistema ha sido llamado Cody.

Por necesidades de modularidad, se trabajó con una arquitectura de software basada en capas, donde cada capa se describe a continuación:

1. **Capa de datos:** Es donde los datos son almacenados y accedidos. Ambas operaciones mencionadas se realizan a través de un gestor de base de datos.
2. **Capa de negocio:** Aquí reside la funcionalidad del sistema, además de las reglas que deben de respetarse. Esta capa realiza una comunicación entre la capa de presentación y la capa de datos, actuando como intermediario cuando un usuario realiza una petición con el objetivo de recibir una respuesta.
3. **Capa de presentación:** Aquí se presenta la interfaz gráfica con la que el usuario interactuará para realizar solicitudes a la capa de negocio. En esta capa se priorizará la usabilidad.

Para conocer lo que incluye específicamente cada capa con respecto al sistema propuesto, cada una se detalla a continuación.

3.1. Capa de datos

La capa de datos se trabajó en una instancia de RDS[11] desde Amazon Web Services[12] (AWS), utilizando el motor de MySQL. El uso de RDS es debido a las garantías de disponibilidad y escalabilidad que AWS específica en la contratación del servicio. Por otro lado, MySQL fue seleccionado porque es una base de datos relacional que soporta la exigencia prevista para el sistema.

Habiendo realizado un análisis previo sobre que entidades son las necesarias para esta capa, se crearon en total unas 16 tablas las cuales se listan a continuación:

- **campus:** id_campus, descripcion.
- **caso_prueba:** id_caso_prueba, id_problema, id_subproblema, entrada, salida_esperada, retroalimentacion, estado, puntaje.
- **categoria:** id_categoria, descripcion.
- **cuestionario_alternativa:** id_alternativa, id_pregunta, descripcion.
- **cuestionario_intento:** id_intento, id_usuario, id_categoria, puntaje_obtenido, fecha.
- **cuestionario_pregunta:** id_pregunta, id_categoria, id_alternativa_correcta, descripcion, nivel.
- **distribucion:** id_distribucion, id_usuario, id_categoria, id_problema_1, id_problema_2, id_problema_3, id_problema_4, nivel, fecha.
- **encuesta_respuesta:** id_encuesta, id_usuario, respuesta_1, respuesta_2, respuesta_3, respuesta_4, fecha.
- **envio:** id_envio, id_problema, id_usuario, id_veredicto, codigo_fuente, puntaje_obtenido, fecha, tokens, estado.
- **grupo:** id_grupo, id_campus.
- **insignia:** id_insignia, descripcion, color.
- **insignia_usuario:** id_usuario, id_insignia, fecha, estado.
- **problema:** id_problema, id_categoria, titulo, enunciado, nivel.
- **retroalimentacion:** id_feedback, id_problema, id_usuario, nivel, estado, fecha.
- **subproblema:** id_subproblema, id_problema, descripcion, entrada, salida, notas, codigo_adicional, solucion.

[11] `https://aws.amazon.com/rds/`
[12] `https://aws.amazon.com/`

- **usuario:** id_usuario, nombre, apellido, id_grupo, fecha, estado.

La manera en cómo se agrupan las tablas mencionadas se puede explicar considerando los siguientes 7 grupos:

- Cuestionario
- Distribución
- Envío
- Insignia
- Problema
- Retroalimentación
- Usuario

Cada grupo se detalla a continuación.

3.1.1. Cuestionario

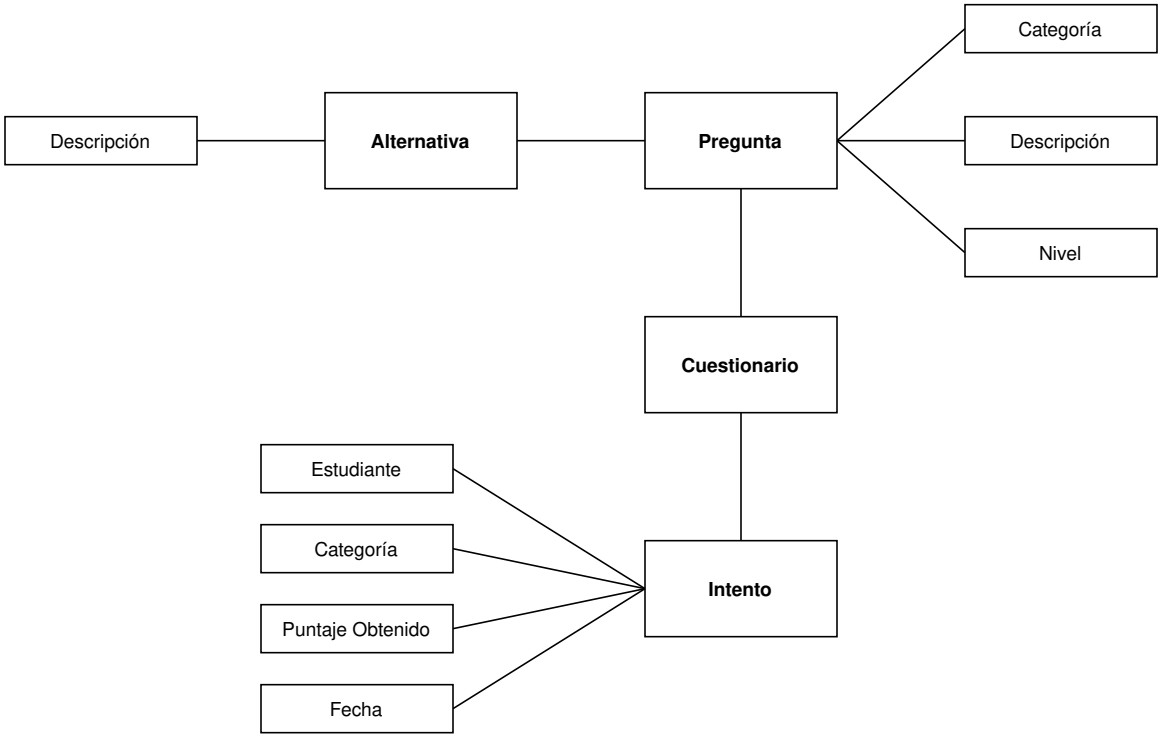

Figura 3: Diagrama simplificado del grupo Cuestionario.
Fuente: Elaboración Propia.

El cuestionario es la evaluación que permite conocer el nivel actual del estudiante para una categoría dada. Las categorías son las siguientes:

- Programas Secuenciales
- Condicionales
- Ciclos
- *Strings*
- Funciones
- Listas
- Tuplas
- Diccionarios

Cada cuestionario está determinado por las categorías anteriores, por lo que siempre existirá la misma cantidad de cuestionarios que categorías. Además, tanto las preguntas como las alternativas asociadas a cada cuestionario están incluidas en su respectiva tabla, así como también los intentos que los estudiantes realizan. Cabe mencionar que un estudiante responde un cuestionario por cada categoría una única vez.

3.1.2. Distribución

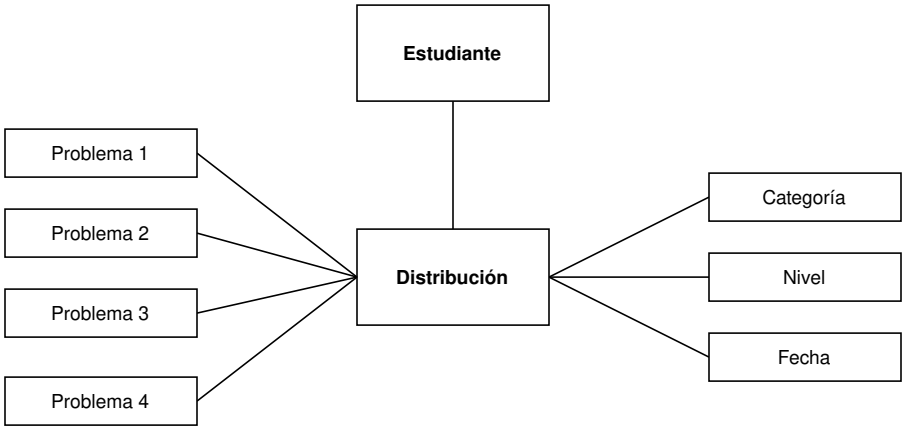

Figura 4: Diagrama simplificado del grupo Distribución.
Fuente: Elaboración Propia.

La distribución es el reparto de ejercicios que al estudiante se le entrega de acuerdo al progreso que lleva en una categoría dada.

Cada una de las distribuciones incluye únicamente 4 problemas. Si bien es una cantidad fija, reducirla o ampliarla conllevaría el desarrollo de una nueva tabla donde se encuentren los problemas para cada distribución.

Las distribuciones tienen también un nivel fácil, medio o difícil, así como también la fecha en la cual fue generada.

3.1.3. Envío

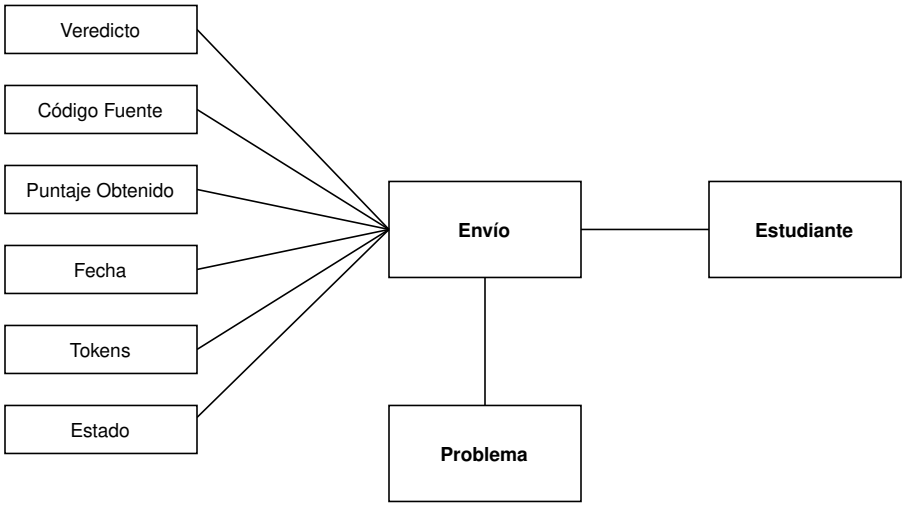

Figura 5: Diagrama simplificado del grupo Envío.
Fuente: Elaboración Propia.

El envío es la entrega que un estudiante realiza cuando intenta resolver un problema.

El valor principal de un envío es el código fuente, ya que eso es lo que se evaluará posteriormente. La fecha también es considerada y el estado del envío permite conocer si el problema asociado debe o no ser considerado para una futura distribución.

Cuando un envío es juzgado, recibe el número identificativo del veredicto asociado, los cuales se listan a continuación:

Tabla 2: Números identificativos para cada veredicto.
Fuente: Elaboración Propia.

Número	Veredicto
1	*In Queue*
2	*Processing*
3	*Accepted*
4	*Wrong Answer*
5	*Time Limit Exceeded*
6	*Compilation Error*
7	*Runtime Error (SIGSEGV)*
8	*Runtime Error (SIGXFSZ)*
9	*Runtime Error (SIGFPE)*
10	*Runtime Error (SIGABRT)*
11	*Runtime Error (NZEC)*
12	*Runtime Error (Other)*
13	*Internal Error*
14	*Exec Format Error*

De acuerdo a la cantidad de casos de prueba resueltos por el código producido por el estudiante, recibirá un puntaje entre 0 y 100. Adicionalmente, los *tokens* que permitirán consultar el detalle de cada uno de los casos de prueba.

3.1.4. Insignia

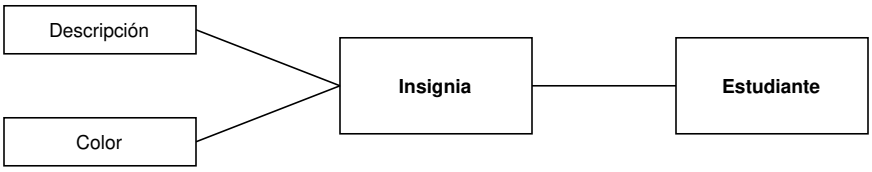

Figura 6: Diagrama simplificado del grupo Insignia.
Fuente: Elaboración Propia.

Una insignia es un reconocimiento que el estudiante recibe por lograr una cierta acción. Cada acción que provoca el recibimiento de una insignia se lista a continuación:

- Realizaste tu primer envío.
- Realizaste 5 envíos.
- Realizaste 10 envíos.

- Realizaste 15 envíos.
- Realizaste 20 envíos.
- Realizaste tu primer envío correcto.
- Realizaste 5 envíos correctos.
- Realizaste 10 envíos correctos.
- Realizaste 15 envíos correctos.
- Realizaste 20 envíos correctos.
- Categoría de Programas Secuenciales completa.
- Categoría de Condicionales completa.
- Categoría de Ciclos completa.
- Categoría de *Strings* completa.
- Categoría de Funciones completa.
- Perfil completado.
- Entregaste tu primer *feedback*.
- Entregaste 5 *feedbacks*.
- Entregaste 10 *feedbacks*.
- Entregaste 15 *feedbacks*.
- Entregaste 20 *feedbacks*.

Cada insignia considera un color, ya que así se logra distinguir entre una insignia y otra.

3.1.5. Problema

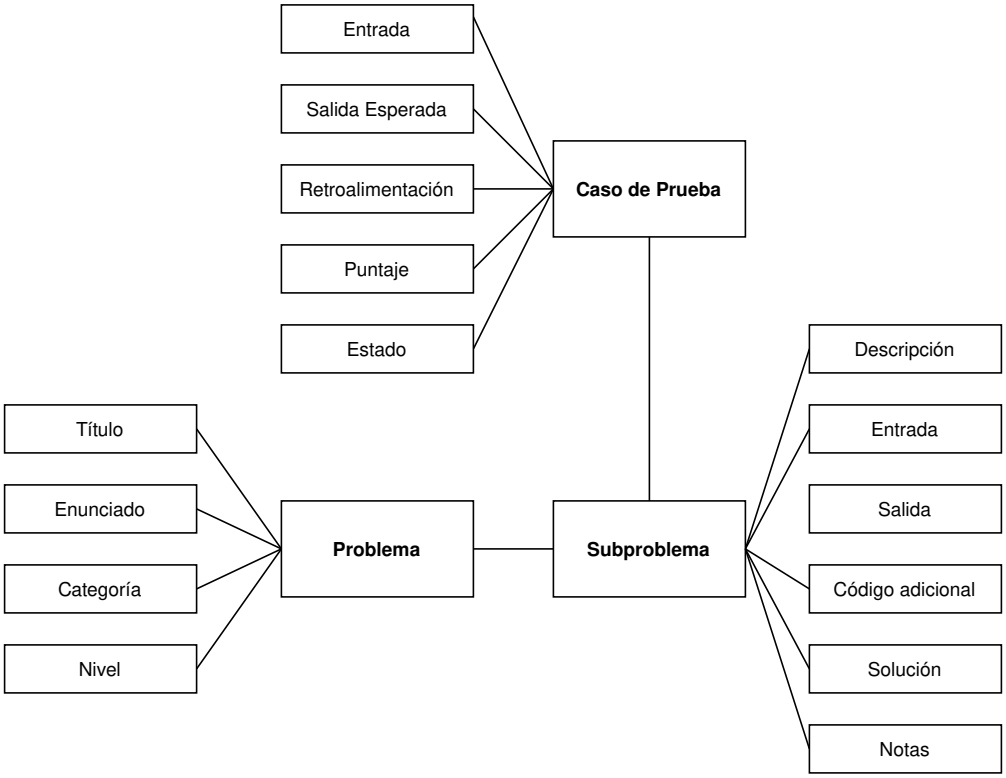

Figura 7: Diagrama simplificado del grupo Problema.
Fuente: Elaboración Propia.

Un problema considera un título y enunciado donde en este último se expresa el contenido sobre lo que se propone resolver por parte del estudiante. Cada problema además incluye una categoría y un nivel, ambos atributos ya se explicaron anteriormente.

Los problemas fueron poblados gracias la base de datos que tiene SMOJ en un ambiente de producción. En SMOJ, los problemas pueden tener hasta 3 subproblemas, por lo que fue necesario elaborar una tabla para subproblemas y así tener homologadas ambas partes. En Cody solo está permitido un subproblema por cada problema, así que en el caso de que algún problema en SMOJ tenga entre 2 y 3 subproblemas, Cody solo considerará el primero.

3.1.6. Retroalimentación

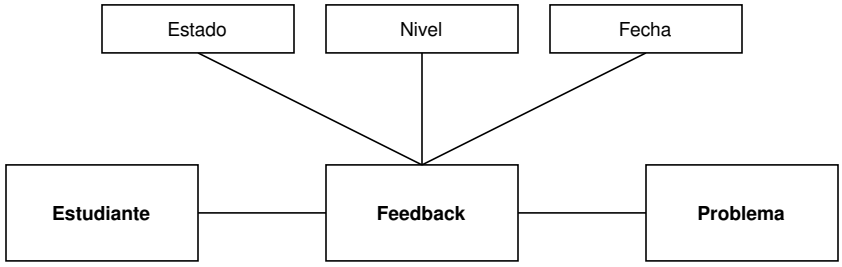

Figura 8: Diagrama simplificado del grupo Retroalimentación.
Fuente: Elaboración Propia.

La retroalimentación es lo que permite conocer la percepción sobre el nivel de un problema dado cuando es resuelto por el estudiante o visto por algún profesor. Un estudiante o profesor puede seleccionar entre las opciones fácil, medio o difícil. Esto último es registrado en la base de datos.

3.1.7. Usuario

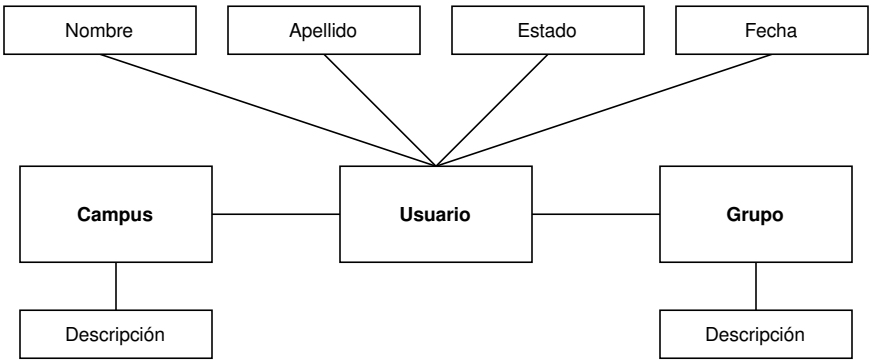

Figura 9: Diagrama simplificado del grupo Usuario.
Fuente: Elaboración Propia.

El usuario registra datos como el nombre, apellido, grupo al cual pertenece, fecha de registro y un estado donde se indica si está activo dentro de la plataforma o no.

El grupo corresponde al paralelo asociado en el curso formal de programación, si bien no es exigido, permite al estudiante ser considerado para la generación de estadísticas que más adelante verá el profesor correspondiente.

Cuando un usuario cumple una serie de condiciones sobre el uso de la plataforma, se despliega una encuesta que debe de responder para que continúe usándola. Las respuestas de esta encuesta se registran en una respectiva tabla.

3.2. Capa de negocio

La capa de negocio fue desarrollada con el lenguaje de programación Python, mediante el micro *framework* Flask[13]. Las razones que se consideraron para la selección de ambos elementos mencionados previamente se listan a continuación:

- Python es un lenguaje de alto nivel.
- Python posee una amplia documentación en la *web*.
- Tanto Python como Flask representan una baja curva de aprendizaje.
- Flask es capaz de adaptarse a cada proyecto instalando las librerías que son necesarias.
- Flask tiene un diseño minimalista, lo cual permite un desarrollo rápido.
- Flask permite un buen manejo de rutas.

Las tres entidades consideradas en esta capa son el estudiante, profesor y sistema. Si bien el profesor puede realizar lo mismo que hace el estudiante, eso no aplica en sentido inverso. El estudiante puede realizar las siguientes acciones:

- Responder un cuestionario para una categoría dada.
- Recibir distribuciones.
- Enviar solución de código.
- Observar envíos pasados
- Recibir insignias.
- Observar un problema dado.
- Enviar *feedback*.
- Observar y actualizar perfil.
- Iniciar y cerrar sesión

[13] `https://flask.palletsprojects.com/en/2.1.x/`

En cuanto al profesor, este puede realizar las siguientes acciones:

- Observar estadísticas.
- Observar problemas.
- Enviar *feedback*.

Por otra parte, el sistema tiene tareas bien específicas que se realizan de manera automática, sin necesidad de que tanto el estudiante como el profesor intervengan. El sistema puede realizar las siguientes acciones:

- Actualizar el nivel actual de los problemas.
- Generar distribución de problemas.

Cada una de las funcionalidades anteriormente listadas se detallan a continuación.

3.2.1. Responder un cuestionario para una categoría dada

Para determinar el nivel que un estudiante tiene en alguna categoría dada, este debe de responder un cuestionario. Este cuestionario se realiza una única vez para cada categoría y no tiene un tiempo límite de duración, además, contempla 9 preguntas que están distribuidas en los niveles de dificultad fácil, medio y difícil. El puntaje para cada pregunta se detalla a continuación:

Tabla 3: Puntaje para cada pregunta de un cuestionario.
Fuente: Elaboración Propia.

Pregunta	Nivel	Puntaje
1	Fácil	0.055
2	Fácil	0.055
3	Fácil	0.055
4	Medio	0.117
5	Medio	0.117
6	Medio	0.117
7	Difícil	0.161
8	Difícil	0.161
9	Difícil	0.162

Es posible notar que el 16.5 % del puntaje total corresponde al nivel fácil, mientras que para los niveles medio y difícil corresponde un 35.1 % y 48.4 % respectivamente. La diferencia

entre estos porcentajes tiene que ver con la intención de valorizar mucho más las preguntas que son de nivel medio y difícil debido a la exigencia que representan.

Una vez que el estudiante termina de responder el cuestionario y aunque no haya respondido alguna de las preguntas, este confirma su envío y posteriormente el sistema configura para él un conjunto de problemas asociados a la categoría dada, para que así pueda comenzar a practicar. Estos problemas están distribuidos de acuerdo al nivel que el estudiante logró respondiendo el cuestionario.

3.2.2. Recibir distribuciones.

La primera distribución que un estudiante recibe es cuando realiza un cuestionario por primera vez, sin embargo, él seguirá recibiendo nuevas distribuciones si es que realiza los problemas propuestos para su nivel. Para un mejor entendimiento acerca de la primera distribución, el proceso se detalla a continuación:

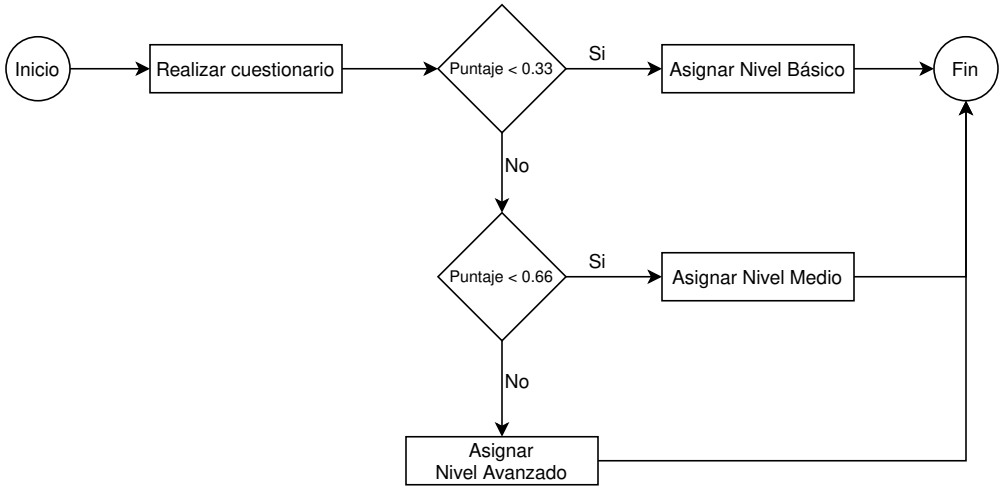

Figura 10: Diagrama de Flujo para la primera distribución.
Fuente: Elaboración Propia.

Está claro que el nivel de la distribución estará determinado por el puntaje total que el estudiante obtenga luego de realizar el cuestionario.

Cada distribución contiene 4 problemas, independiente del nivel de dificultad. Además, estos 4 problemas estarán compuestos por 2 problemas del mismo nivel que la distribución, mientras que los otros dos problemas tendrán cada uno de los niveles restantes. Por ejemplo, si una distribución está determinada por un nivel fácil, habrán dos problemas fáciles, un problema medio y el último de dificultad avanzada.

Si se trata de una primera distribución, entonces los problemas son seleccionados de manera

aleatoria.

Para el caso de las futuras distribuciones, estas estarán condicionadas por una serie de reglas enumeradas que se señalan en el siguiente diagrama:

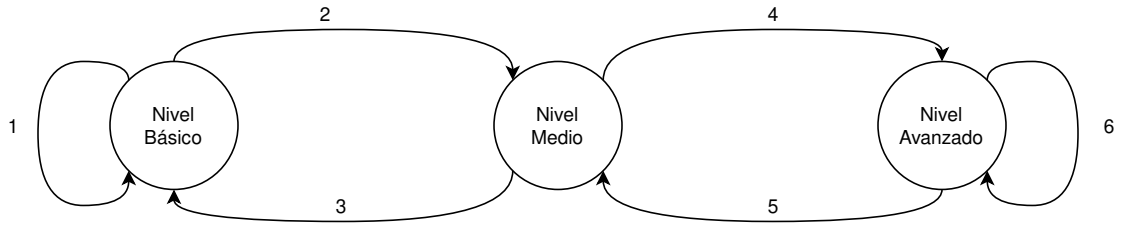

Figura 11: Diagrama simplificado para las futuras distribuciones.
Fuente: Elaboración Propia.

Cada una de las reglas se detalla a continuación:

Tabla 4: Condiciones para generación de distribuciones.
Fuente: Elaboración Propia.

Regla	Condición
1	No resolver ningún problema y haber realizado al menos 8 intentos.
2	Resolver 2 problemas de nivel básico ó resolver al menos 1 problema del nivel medio o difícil.
3	Fallar 2 veces en el problema fácil ó fallar 4 veces en algún problema medio o difícil.
4	Resolver 2 problemas de nivel medio ó resolver el problema de nivel difícil.
5	Fallar 2 veces en el problema fácil o medio ó fallar 4 veces en algún problema difícil.
6	Haber realizado todos los problemas de la distribución.

De acuerdo a lo anterior, es posible notar que la generación de distribuciones siempre es ilimitada. Puede observarse que en ambos extremos, ya sea en el nivel básico o avanzado, aun se siguen generando nuevas distribuciones para que el estudiante no se quede detenido si se trata del nivel básico y para que siempre tenga la posibilidad de exigirse más si se trata del nivel avanzado. Internamente, se prioriza que cada distribución contenga problemas que el estudiante no haya resuelto antes.

3.2.3. Enviar solución de código.

Una vez que el estudiante lee el enunciado de un problema, intentará desarrollar su solución y luego enviarla para ser juzgada de manera automática. Tal juzgamiento se realiza a través de un servicio llamado Judge0[14] que corresponde a una API. Esta API recibe el código junto con los casos de prueba correspondientes, para posteriormente evaluar la solución con los datos de entrada y comparar la salida generada con respecto a la salida esperada. Cabe mencionar que la selección de Judge0 se basó en la experiencia previa por parte del autor de esta Memoria en el proyecto de SMOJ y en las ventajas que se detallan en [Došilović y Mekterović, 2020].

El puntaje que reciba al estudiante estará determinado por el número de casos de prueba correctos con respecto al total de estos. Por ejemplo, si algún problema tiene 10 casos de prueba y el estudiante solo logra resolver 5, él recibirá 50 puntos. Es claro que para tener un problema totalmente correcto, se debe obtener 100 puntos, o en otras palabras, haber resuelto la totalidad de casos de prueba.

En aspectos de arquitectura, la API de Judge0 fue desplegada en una instancia propia de AWS e independiente del proyecto central, ya que es un recurso que necesita responder a muchas solicitudes de manera simultánea.

3.2.4. Observar envíos pasados

Todos los envíos que realizan los estudiantes son persistidos a la base de datos para que posteriormente puedan ser revisados por este. Lo anterior será útil para comparar alguna solución nueva con respecto a una más antigua o también para recordar la estrategia empleada.

El detalle de los casos de prueba que incluye cada envío no se registra en la base de datos principal, ya que tales datos se recuperan consultando a la API de Judge0. Por lo tanto, para obtener un envío completo, se consulta a la base de datos principal y la API de Judge0. Si por alguna razón la API de Judge0 no se encuentra disponible, al menos será desplegado el código de la solución que es obtenido de la base de datos principal.

3.2.5. Recibir insignias.

A medida que el estudiante utiliza la plataforma, este tiene la posibilidad de recibir una serie de insignias ya mencionadas previamente.

[14] `https://judge0.com/`

La estrategia para determinar la asignación de una o más insignias es consultado los requisitos correspondientes cuando el estudiante realiza alguna de las siguientes acciones:

- Realiza un envío.
- Completa todos los problemas para alguna categoría.
- Completa su perfil.
- Realiza un *feedback*.

Por ejemplo, si el estudiante completa su perfil, se revisa si le corresponde alguna insignia por haber realizado algún envío previo, haber completado su perfil, haber enviado algún *feedback* y claramente por si efectivamente ha completado su perfil. A pesar de que se realiza una exploración completa, esto se hace así debido a que el estudiante podría cerrar la ventana del navegador web y si ha cumplido con alguno de los requisitos, no se enterará en ese momento que ha obtenido una insignia.

3.2.6. Observar un problema dado.

Como se ha mencionado antes, cada distribución incluye 4 problemas. El estudiante tiene a su disposición el enunciado para cada problema y allí se señala lo que necesita resolver, así como también observaciones o notas que pueden servir de guía para el desarrollo de la solución.

Es importante mencionar que un estudiante no puede observar un problema que no está incluido dentro de su distribución actual para alguna categoría dada. Esto quiere decir que si el estudiante recibe una nueva distribución, no podrá volver a revisar el enunciado de algún problema incluido en cualquiera de las distribuciones pasadas.

3.2.7. Enviar *feedback*.

Si el estudiante resuelve algún problema, independiente de la categoría asociada, tiene la oportunidad de calificar la dificultad que le exigió el desarrollo de la solución. Los niveles que tiene permitido seleccionar son fácil, medio y difícil.

Esta acción solo la puede realizar una vez por cada distribución, por lo que si en el futuro recibe el mismo problema en una nueva distribución, también tiene otra oportunidad para evaluar la complejidad del problema.

En el caso de que el estudiante no haya enviado el *feedback*, se le recordará hacerlo cada vez que vuelva a intentar el problema, considerando que ya lo resolvió correctamente dentro de la distribución actual.

Al igual que el estudiante, el profesor también tiene la posibilidad de enviar *feedback* hacia un problema en específico. La diferencia residen en el lugar donde el profesor lo realiza. Tal acción es realizada cuando el profesor observa la lista de problemas y selecciona uno en particular. Es en ese momento cuando un formulario se despliega y se le presentan las opciones de fácil, medio y difícil.

El profesor puede seleccionar un nivel de dificultad y más adelante cambiarlo si así lo requiere, por lo que no está limitado para opinar acerca del problema una única vez.

3.2.8. Observar y actualizar perfil.

El perfil del estudiante no incluye muchos detalles, solo presenta información personal básica y tiene la posibilidad de asignar manualmente su paralelo.

El hecho de actualizar el paralelo permite al profesor del mismo tenerlo en cuenta dentro de las estadísticas generales y del curso. Si bien no se fuerza al estudiante a actualizar su paralelo, es el profesor quien debe pedirlo en la aula de clases o similar.

3.2.9. Iniciar y cerrar sesión

El inicio de sesión se realiza desde una plataforma de tipo *Learning Management Systems* o LMS. En tal plataforma es necesario crear un recurso externo que permita la comunicación entre el LMS y Cody. Básicamente un administrador de la plataforma debe asignar al recurso externo los siguientes parámetros:

- Nombre del plataforma.
- *endpoint*.
- Clave pública.
- Clave privada.

Con los valores previamente definidos y correctamente asignados, el estudiante podrá iniciar sesión seleccionado un enlace ubicado en el curso al que pertenece dentro del LMS.

Luego de haber validado los valores mencionados, Cody recibe información del estudiante como el nombre, apellido, nombre completo, correo electrónico, entre otros. Esa misma

información es utilizada para ser persistida a la base de datos si es que el estudiante está ingresando por primera vez. Para diferenciar a un estudiante con respecto a un profesor, el LMS entrega el tipo de cuenta que tiene el usuario, por lo que así es posible diferenciar más adelante las funcionalidades que le corresponde a cada tipo.

Si lo que desea el estudiante es cerrar la sesión, solo basta con que lo realice directamente desde Cody.

3.2.10. Observar estadísticas

Cada profesor tiene la posibilidad de revisar estadísticas de interés que le permitirá conocer de manera general el progreso que lleva su curso. Las siguientes estadísticas se encuentran disponibles:

- Cantidad total de usuarios, envíos, insignias, distribuciones, *feedbacks* y cuestionarios.
- Cantidad de envíos por cada tipo de veredicto.
- Cantidad de envíos por cada categoría y nivel de dificultad.
- Cantidad de distribuciones por cada categoría y nivel de dificultad.
- Cantidad de insignias entregadas por cada tipo de insignia.
- Promedio y cantidad de cuestionarios intentados por cada categoría.

A pesar de que el profesor puede visualizar las estadísticas por curso, también lo puede hacer considerando la totalidad de cursos existentes en la plataforma.

Las estadísticas no son previamente calculadas antes de ser consultadas, es decir, se calculan cada vez que se requieren obtener.

3.2.11. Observar problemas

Los profesores tienen acceso completo para revisar cada uno de los problemas que están incorporados en la plataforma. Además, un sistema de paginación es integrado para no sobrecargar la lista de problemas.

3.2.12. Actualizar el nivel actual de los problemas

Cada 60 minutos, un evento es ejecutado automáticamente para que actualice los niveles de cada problema a los cuales se le ha opinado acerca de su nivel de dificultad.

Si un problema ha sido evaluado como fácil entonces corresponde 1 punto, mientras que para los niveles medio y difícil corresponden 2 y 3 puntos respectivamente. Luego, el nuevo nivel del problema es el promedio obtenido de acuerdo a la totalidad de puntos considerados. La siguiente tabla detalla que nivel tendrá un problema de acuerdo al promedio obtenido.

Tabla 5: Condiciones para actualizar el nivel de un problema dado.
Fuente: Elaboración Propia.

Nivel	Condición
Fácil	Promedio menor a 0.33.
Medio	Promedio mayor o igual a 0.33 y menor a 0.66.
Difícil	Promedio mayor o igual a 0.66.

3.2.13. Generar distribución de problemas

Cuando el sistema tiene que encargarse de generar una distribución para algún estudiante, existen dos posibles casos:

- Es la primera distribución.
- Previamente ya se ha generado una o más distribuciones.

Para el primer caso, el estudiante está recibiendo tal distribución porque ha finalizado un cuestionario. La estrategia es sencilla, solo se obtienen de manera aleatoria 4 problemas que cumplan con la cantidad necesaria para cada uno de los niveles de dificultad.

Con respecto al segundo caso, la decisión acerca de generar o no una nueva distribución es tomada cuando el estudiante realiza el envío para algún problema. En un caso afirmativo, lo que se realiza es la búsqueda de problemas candidatos para la nueva distribución. En esta fase se prioriza a los problemas que no hayan sido resueltos antes, pero en el caso de que haya que repetir alguno, también se acepta.

Ya sea que el estudiante avance o retroceda de nivel, todo está evaluado en las reglas que se mencionaron previamente.

3.3. Capa de presentación

Para la capa de presentación, se decidió por utilizar el *framework* Bootstrap[15] por los siguientes motivos:

- Es gratuito y *open source*.
- Se basa en el concepto de *responsive design*.
- Posee una amplia variedad de componentes fáciles de ocupar.
- Tiene una documentación extensa en la *web*.

Con respecto a la estructura, las vistas se dividen en tres secciones principalmente. Por una parte, está el *header* que permite incluir enlaces hacia otras vistas para un acceso más rápido. Por otra parte, está el *body*, donde se despliega todo el contenido que es de interés para el estudiante o profesor. Por último, se considera un *footer*, el cual se muestra dependiendo de la vista, pero su función es incluir el *copyright*.

A continuación, se muestra una parte de la interfaz correspondiente a la página inicial de Cody.

Figura 12: Vista de la página inicial de Cody.
Fuente: Elaboración Propia.

[15] https://getbootstrap.com/docs/5.1/getting-started/introduction/

3.3.1. Colores y fuente

El color primario de la plataforma es un azul, mientras que el color secundario corresponde a un gris. La selección de tales colores se basó en la idea de mantener una interfaz minimalista y reducida en colores fuertes como el rojo, pensando en que los estudiantes podrían percibir este color como estresante. Ambos colores pueden apreciarse en la siguiente figura:

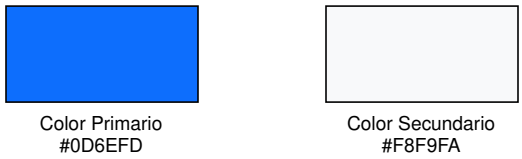

Figura 13: Colores primario y secundario.
Fuente: Elaboración Propia.

Con respecto a la fuente, la elegida fue Segoe UI font family[16], la cual es propiedad de Microsoft Corporation. Esta fuente integrada en la plataforma genera que cualquier tipo de texto luzca agradable para la redacción.

Aa Bb Cc Dd Ee Ff Gg Hh Ii
Jj Kk Ll Mm Nn Oo Pp Qq
Rr Ss Tt Uu Vv Ww Xx Yy Zz
1234567890!@#%&*?,;:$£€
Aa Bb Cc Dd Ee Ff Gg Hh Ii
Jj Kk Ll Mm Nn Oo Pp Qq
Rr Ss Tt Uu Vv Ww Xx Yy Zz
1234567890!@#%&?,;:$£€*

Figura 14: Fuente de texto integrada en Cody.
Fuente: `https://docs.microsoft.com/en-us/typography/font-list/segoe-ui`.

3.3.2. Enfoque *Responsive Design*

El diseño responsivo busca la adaptación de las interfaces sin depender del dispositivo desde donde se están visualizando. Tales dispositivos podrían ser una computadora, tableta,

[16] `https://docs.microsoft.com/en-us/typography/font-list/segoe-ui`

teléfono móvil, reloj inteligente, entre otros. Debido al trabajo realizado con Boostrap, Cody ofrece un diseño adaptativo, por lo que permite que estudiantes y profesores que no cuenten con un computador en el mejor de los casos, puedan de igual manera tener una experiencia de navegación favorable a través de un teléfono móvil o tableta.

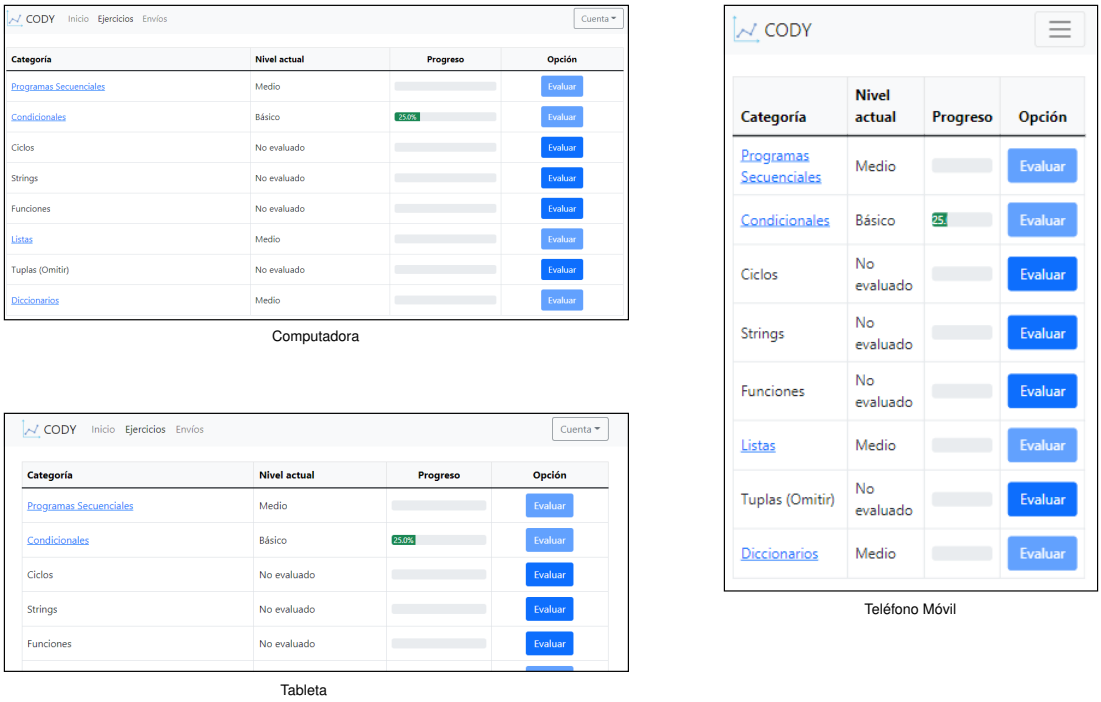

Figura 15: Interfaz responsiva de Cody.
Fuente: Elaboración Propia.

Como puede apreciarse en las imagen anterior, dependiendo de la proporción, el *header* oculta los enlaces y los muestra si el botón ubicado en la esquina superior derecha es seleccionado. Con respecto a la sección del *body*, se observa como el texto varia su despliegue, separando las palabras en más líneas si es necesario.

Cada una de las vistas implementadas se detallan en las siguientes secciones.

3.3.3. Vista Cuestionario

Cada una de las preguntas que componen un cuestionario están incluidas en un bloque bordeado para separarlas unas con otras. En el caso de que alguna pregunta incluya código fuente, este se resalta para diferenciarlo del resto de texto.

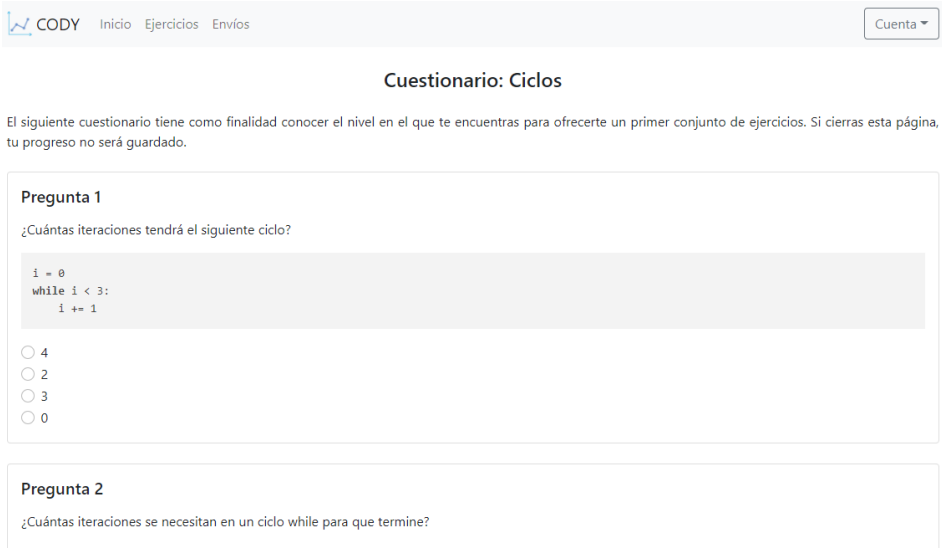

Figura 16: Vista Cuestionario.
Fuente: Elaboración Propia.

3.3.4. Vista Categorías

Las categorías son presentadas en una tabla donde en se informa el nivel que el estudiante tiene para la distribución actual, así como también un botón que al ser seleccionado lleva al estudiante al cuestionario respectivo. La tercera columna indica el nivel de progreso que se lleva hasta ese momento, el cual se calcula considerando el porcentaje de problemas resueltos.

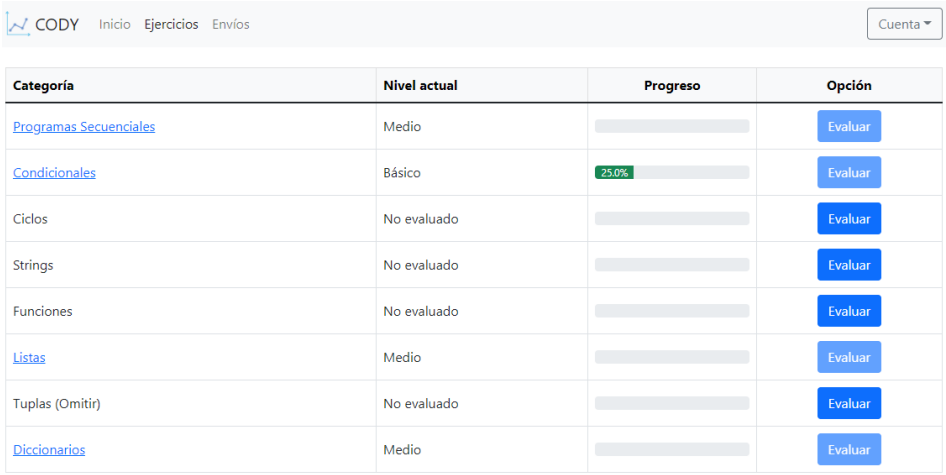

Figura 17: Vista Categorías.
Fuente: Elaboración Propia.

3.3.5. Vista Problemas

La vista para los problemas se divide en dos partes. La primera lista en una tabla los problemas para una distribución dada. La segunda columna de esa misma tabla indica el nivel de dificultad que tiene cada problema, es aquí donde el estudiante puede decidir el problema por el cual quiere comenzar. La tercera columna por otro lado, indica el progreso alcanzado, esto se calcula como el porcentaje de casos de prueba con un estado de *accepted*. Para esto último, siempre se considera el envío con el más alto puntaje.

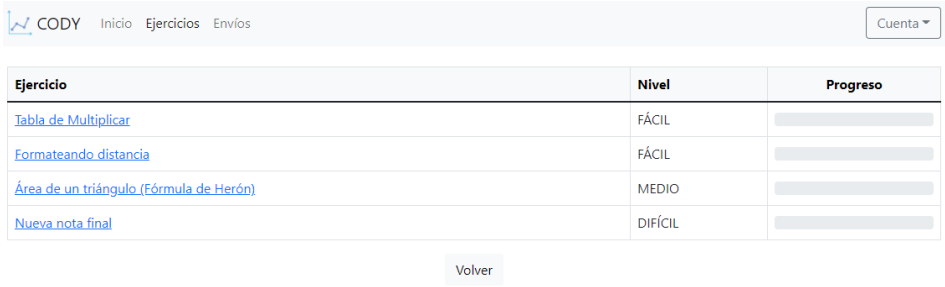

Figura 18: Vista Listado de problemas.
Fuente: Elaboración Propia.

La segunda vista, despliega el contenido específico para un problema dado. Aquí se incluye el título, el enunciado, los casos de prueba y así como también una sección dedicada a la escritura y ejecución de código. Al igual como en los cuestionarios, el código fuente que es escrito en Python es diferenciado del resto de texto. Las fórmulas matemáticas reciben un estilo distinto, ajustándose a lo que se conoce como LaTeX[17].

[17] `https://www.latex-project.org/`

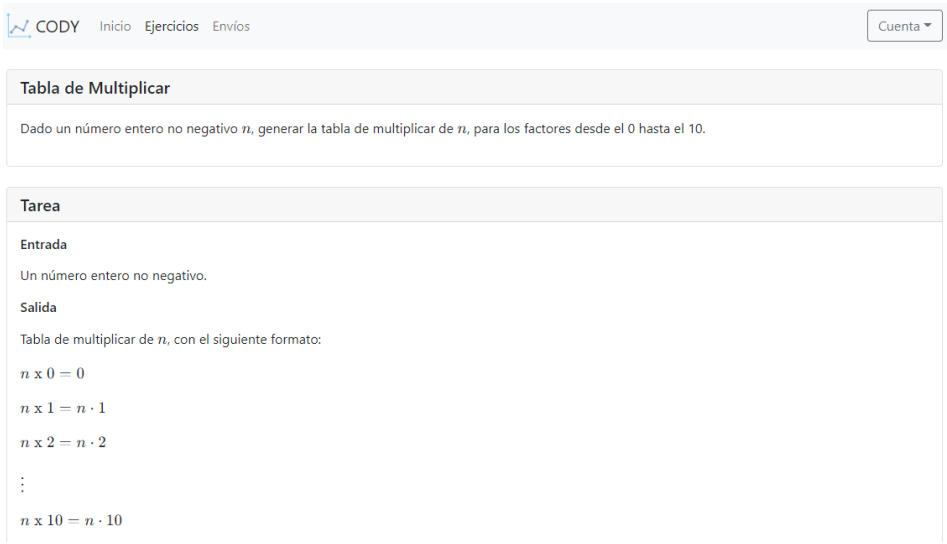

Figura 19: Vista Problema.
Fuente: Elaboración Propia.

3.3.6. Vista Envíos

A pesar de que el editor de código que está integrado en la misma vista para un problema no es una vista independiente, está directamente relacionado con la vista de los envíos porque esta última se complementa desde la otra. El editor incluye un amplio espacio para escribir código, el cual también es posible alargar. En la esquina inferior derecha se añaden dos botones, el de la izquierda permite ejecutar el código considerando los casos de prueba que aparecen en el enunciado como ejemplos, en cambio, el de la derecha realiza lo mismo, pero considerando todos los casos, es decir, públicos y privados.

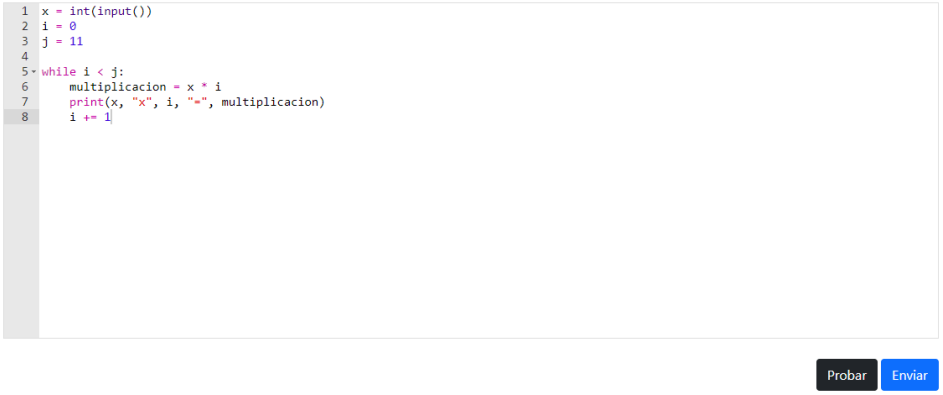

Figura 20: Vista Editor de código.
Fuente: Elaboración Propia.

En la siguiente vista se observa el resultado de la evaluación del código una vez que se envía. Primero se presenta un bloque que incluye el detalle del resultado, el cual cambia de color dependiendo del porcentaje de acierto. Luego y solo en el caso de que el problema sea resuelto completamente, se despliega un bloque de color azul para que el estudiante indique el nivel de dificultad que cree más apropiado para el problema que acaba de resolver. Por último, se lista el detalle para cada caso de prueba, donde los de tipo público muestran toda la información, pero en cambio, los de tipo privado solo muestran el *feedback* que pudieran tener asociado.

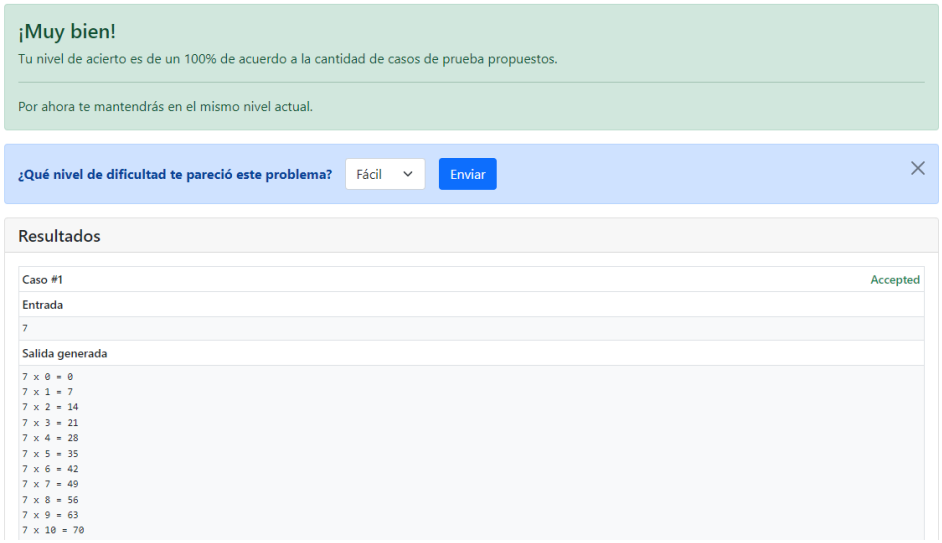

Figura 21: Vista Envío de código.
Fuente: Elaboración Propia.

La última vista asociada a los envíos corresponde a la que incluye la lista de los mismos. Los envíos son ordenados de manera descendiente con respecto a la fecha. La última columna permite abrir un *modal* donde se visualiza el detalle del envío, como el código fuente y los casos de prueba respectivos.

Cabe mencionar que se espera que un estudiante alcance a tener una cantidad de envíos considerable, por lo que en esta vista se integró un paginador para no sobrecargar todo dentro de la misma página.

CODY Inicio Ejercicios Envíos Cuenta

#	Problema	Puntaje	Fecha	Opción
1270	Calculadora	100.0	13:20:09 29-06-2022	Detalle
1269	Calculadora	80.0	13:17:10 29-06-2022	Detalle
1268	Calculadora	80.0	13:14:49 29-06-2022	Detalle
1267	Calculadora	0.0	13:06:57 29-06-2022	Detalle
1266	Calculadora	0.0	13:00:58 29-06-2022	Detalle
1118	Formateando distancia	100.0	16:37:26 28-06-2022	Detalle
788	Horarios	100.0	16:37:26 28-06-2022	Detalle
787	El Vendedor	0.0	16:37:26 28-06-2022	Detalle
786	Hola SMOJ	100.0	16:37:26 28-06-2022	Detalle
785	Hola SMOJ	100.0	16:37:26 28-06-2022	Detalle
784	Hola SMOJ	100.0	16:37:26 28-06-2022	Detalle

Figura 22: Vista Lista de envíos.
Fuente: Elaboración Propia.

3.3.7. Vista Perfil

El perfil se divide en dos secciones, la sección de la parte de arriba se dedica a presentar solo la información personal del estudiante o profesor, donde la única opción disponible para ser actualizada es el paralelo o grupo al cual pertenece.

Por otra parte, la sección ubicada en la parte de abajo incluye el listado de las insignias recibidas, indicando la fecha en la que se obtuvieron y el distintivo en forma de globo que representa a cada una.

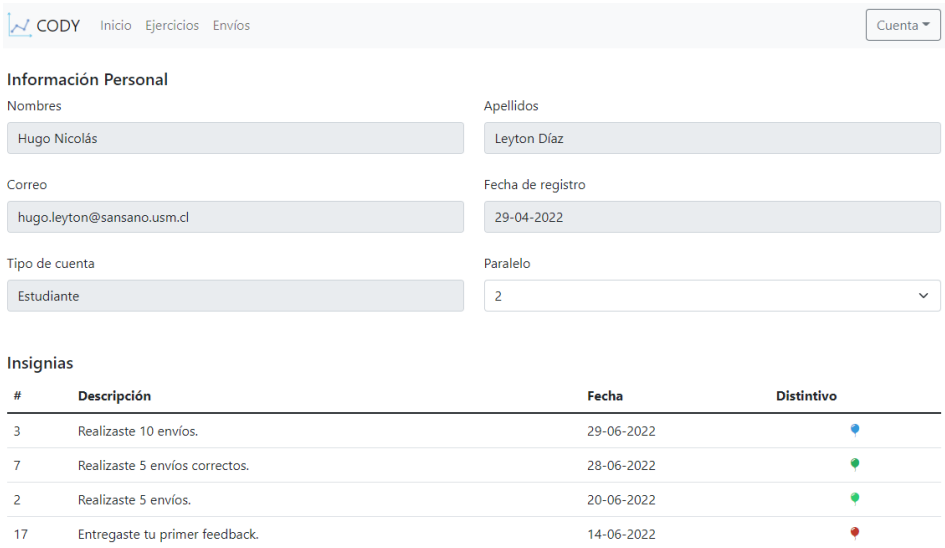

Figura 23: Vista Perfil.
Fuente: Elaboración Propia.

3.3.8. Vista Estadísticas

El profesor tiene la posibilidad de visualizar las estadísticas en una única vista. Cada vez que accede, las estadísticas que se despliegan son las que incluyen a todos los paralelos dentro de la plataforma, sin embargo, un *select* está situado en la parte superior que contiene a todos los paralelos. De esa forma, el profesor puede hacer un filtrado y centrarse en las estadísticas que correspondan a un paralelo en particular.

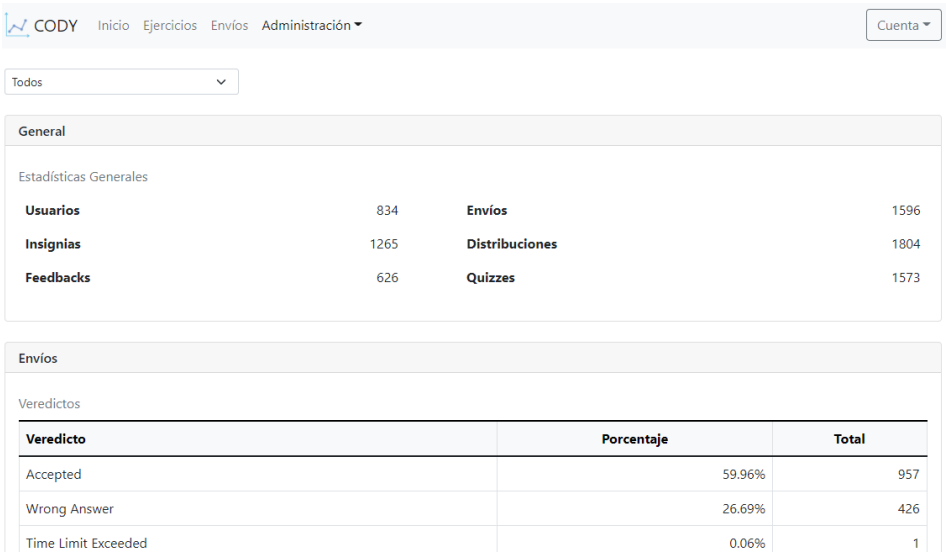

Figura 24: Vista Estadísticas.
Fuente: Elaboración Propia.

3.3.9. Vista Administrador de problemas

La segunda sección a la que solo el profesor tiene acceso, corresponde al lugar donde es posible administrar los problemas. En una primera instancia, se listan en una tabla los problemas ordenados ascendentemente por el título. La tercera columna de la tabla indica el nivel actual que posee el problema y la cuarta permite actualizar la categoría asociada.

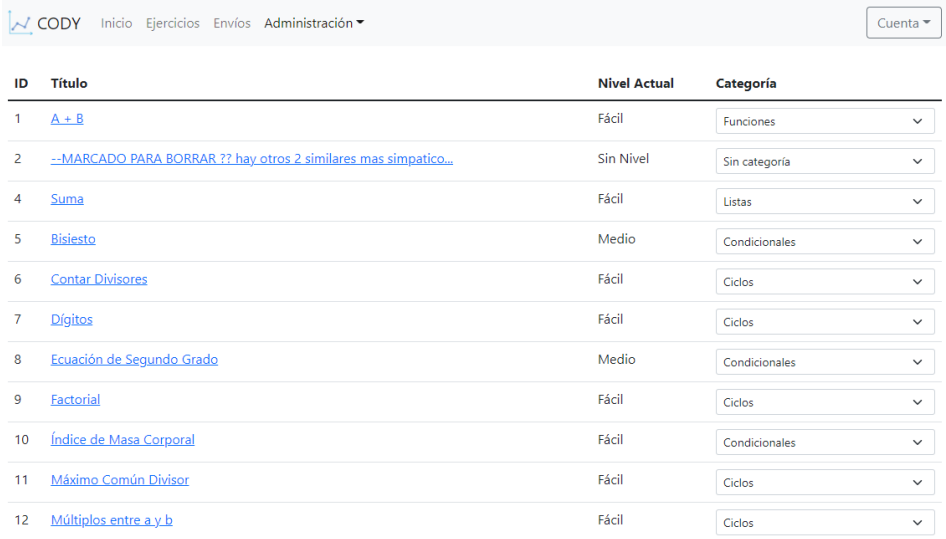

Figura 25: Vista Administrador de problemas.
Fuente: Elaboración Propia.

Si el profesor selecciona el enlace de algún problema situado en la columna del título, entonces un *modal* se abrirá y podrá observar el enunciado. Aquí también está incluido un bloque donde puede opinar acerca del nivel de dificultad que mejor se ajusta al problema.

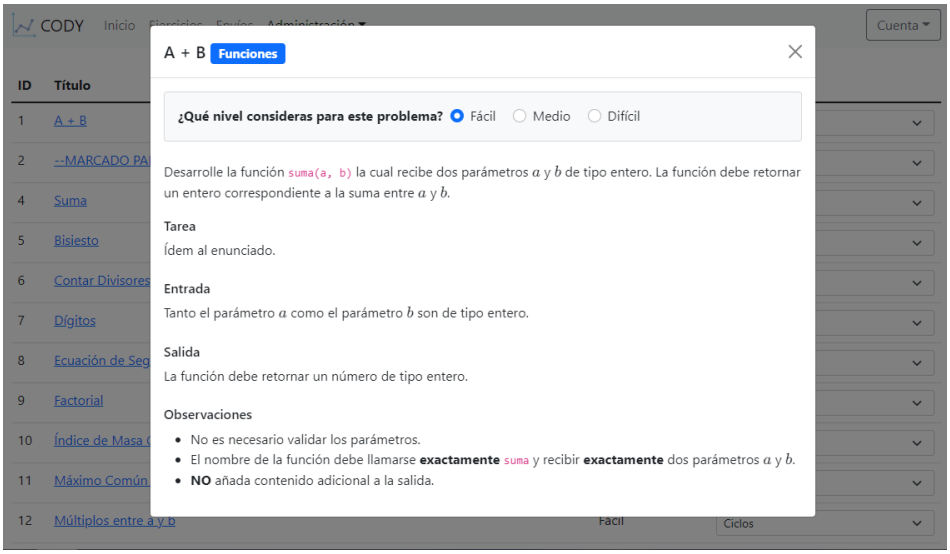

Figura 26: Vista Visualización de un problema en el Administrador de problemas. Fuente: Elaboración Propia.

3.4. Arquitectura

La arquitectura implementada en Cody cuenta con 2 servidores *webs* y dos bases de datos, pero una de estas es externa al servidor con el que se asocia y la otra no.

Cuando una petición se realiza desde el navegador *web*, esta se dirige al servidor de Cody. En ese momento existen dos posibilidades:

- La solicitud tiene como intención juzgar código.
- La solicitud tiene como intención leer o modificar información acerca de un problema, envío, perfil, etc.

Si se trata del primer caso, entonces se realizará una segunda petición pero de manera asíncrona. Esta petición se dirige hacia el servidor de Judge0 que está alojado en una instancia EC2[18] de AWS, y a su vez, Judge0 se encuentra ejecutándose en un contenedor de Docker[19]. Cuando la evaluación del código termina, se devuelve una respuesta que llega al servidor de

[18] `https://aws.amazon.com/ec2/`
[19] `https://www.docker.com/`

Cody para luego ser persistido a la base de datos. Esta última está alojada en una instancia de RDS de AWS.

Por otro lado, si lo que se busca no es juzgar código, entonces la petición se dirige hacia el servidor de Cody, el cual está alojado en otra instancia EC2 de AWS. En el caso de querer consultar a la base de datos de Cody, ya sea para leer o escribir un registro, entonces eso se realiza y se responden las solicitudes correspondientes.

A continuación puede observarse el diagrama de arquitectura correspondiente, aunque a un nivel básico, ya que existen otros componentes que por simplicidad no se incluyen, como las VPC, *Subnets*, *Load Balancers*, etc.

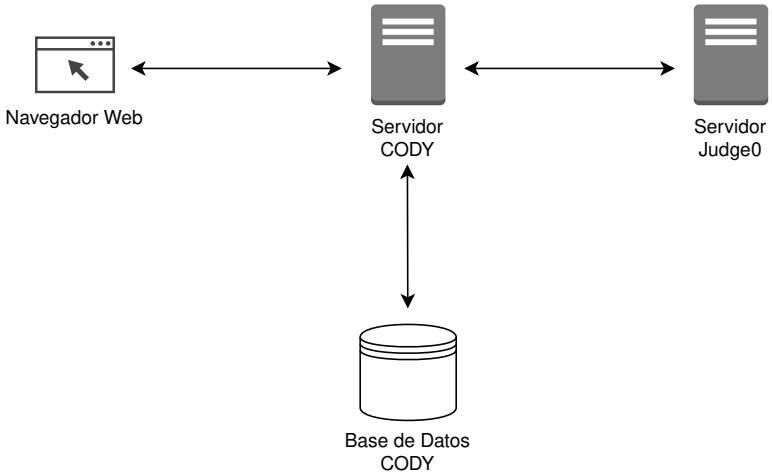

Figura 27: Diagrama básico de Arquitectura.
Fuente: Elaboración Propia.

CAPÍTULO 4
VALIDACIÓN DE LA SOLUCIÓN

Para validar la solución presentada previamente, se realizaron las siguientes actividades:

- Habilitación de la plataforma en una fase de prototipo.
- Habilitación de la plataforma antes de la primera evaluación del curso, con posterior encuesta para estudiantes.
- Habilitación de la plataforma antes de la segunda evaluación del curso, con posterior encuesta para estudiantes.
- Entrevista a profesores para recibir retroalimentación.

Cada una de tales actividades se detallan en las siguientes secciones, junto con los resultados obtenidos.

4.1. Prototipo

El prototipo desarrollado en esta parte representó aproximadamente un 50 % de avance de la plataforma, por lo que no todas las características y funcionalidades estaban finalizadas. Algunas de las funcionalidades terminadas tuvieron que ver con la posibilidad de realizar cuestionarios, visualizar problemas y enviar código.

Se invitó a un estudiante a participar en el uso de la plataforma. Finalizada la sesión, se recibieron las siguientes observaciones:

- Sería de ayuda poder contar con un botón para evaluar el código con los casos de prueba públicos, tal como se incluye en SMOJ.
- Las distintas funcionalidades experimentadas se encontraron útiles para progresar de manera constante.
- La interfaz es atractiva y fácil de entender.

4.2. Primera habilitación de la plataforma

Dos días antes de la primera evaluación del curso, el acceso fue habilitado para que cualquier estudiante pudiera ingresar y hacer uso de la plataforma. Hasta este punto, los ajustes que se necesitaban realizar no eran muchos.

Cabe mencionar que solo se consideraron las siguientes categorías:

- Programas Secuenciales
- Condicionales
- Ciclos
- *Strings*
- Funciones

4.2.1. Resultados generales

De manera general, se obtuvieron los siguientes resultados:

Tabla 6: Resultados generales luego de la primera habilitación.
Fuente: Elaboración Propia.

Descripción	Cantidad
Usuarios registrados	422
Problemas disponibles	81
Envíos realizados	648
Insignias recibidas	577
Distribuciones generadas	769
Feedbacks enviados	262
Intentos en cuestionarios	667

Considerando que la cantidad máxima de estudiantes que podrían haber accedido es cercano a 1300, el resultado obtenido de 422 es significativo, ya que representa cerca del 32 % con respecto al máximo mencionado.

4.2.2. Resultados de envíos realizados

La categoría con más envíos efectuados fue la de Programas Secuenciales. Se esperaba que esta categoría tuviera la máxima cantidad debido a que es la primera que se dicta en el curso

formal, por lo que es aquí donde los estudiantes comienzan a familiarizarse con los contenidos.

Tabla 7: Resultados de envíos por categoría y nivel luego de la primera habilitación. Fuente: Elaboración Propia.

Categoría	Básico	Medio	Avanzado	Cantidad
Programas Secuenciales	156	171	63	390
Condicionales	44	29	22	95
Ciclos	20	6	5	31
Strings	46	23	15	84
Funciones	37	9	2	48

En cuanto a los distintos tipos de veredictos, el 61.11 % con respecto al total correspondió al de tipo *Accepted*, seguido por el de tipo *Wrong Answer* con aproximadamente la mitad del tipo anteriormente mencionado.

Tabla 8: Resultados de veredictos luego de la primera habilitación. Fuente: Elaboración Propia.

Veredicto	Porcentaje	Cantidad
Accepted	61.11 %	396
Wrong Answer	27.31 %	177
Runtime Error (NZEC)	11.42 %	74
Time Limit Exceeded	0.16 %	1

4.2.3. Resultados de insignias recibidas

Las insignias que más se repartieron fueron aquellas que conllevan acciones que se realizan apenas se comienza a tener interacción con la plataforma. El listado completo se muestra a continuación:

Tabla 9: Resultados de insignias luego de la primera habilitación.
Fuente: Elaboración Propia.

Descripción	Cantidad
Realizaste tu primer envío	139
Realizaste 5 envíos	44
Realizaste 10 envíos	19
Realizaste 15 envíos	9
Realizaste 20 envíos	2
Realizaste tu primer envío correcto	117
Realizaste 5 envíos correctos	28
Realizaste 10 envíos correctos	5
Realizaste 15 envíos correctos	2
Categoría de Programas Secuenciales completa	24
Categoría de Condicionales completa	2
Categoría de Ciclos completa	1
Categoría de *Strings* completa	4
Categoría de Funciones completa	1
Perfil completado	69
Entregaste tu primer *feedback*	101
Entregaste 5 *feedbacks*	10
Entregaste 10 *feedbacks*	1

4.2.4. Resultados de distribuciones generadas

De acuerdo a la siguiente tabla, la cantidad de distribuciones generadas fue mayor a medida que el nivel de dificultad creció, esto quiere decir que los estudiantes estuvieron expuestos la mayoría de las veces a una exigencia más alta.

Tabla 10: Resultados de distribuciones por categoría y nivel luego de la primera habilitación.
Fuente: Elaboración Propia.

Categoría	Básico	Medio	Avanzado	Cantidad
Programas Secuenciales	10	86	172	268
Condicionales	5	10	122	137
Ciclos	13	42	52	107
Strings	6	49	115	170
Funciones	20	34	33	87

4.2.5. Resultados de *feedbacks* entregados

Con respecto a los *feedbacks* entregados por los estudiantes, las cantidades reflejan una buena disposición para opinar acerca de las dificultades de los problemas, ya que la entrega de *feedback* es una acción que solo se permite realizar cuando el problema ha sido completamente resuelto y por una única vez. La categoría con más *feedbacks* recibidos fue la de Programas Secuenciales.

Tabla 11: Resultados de *feedbacks* por categoría luego de la primera habilitación. Fuente: Elaboración Propia.

Categoría	Cantidad
Programas Secuenciales	172
Condicionales	35
Ciclos	11
Strings	31
Funciones	13

4.2.6. Resultados de cuestionarios intentados

Los resultados desplegados a continuación demuestran que los estudiantes ingresaron a la plataforma con una buena comprensión teórica previa, pues la mayoría de los promedios listados están más cercanos a 1. La categoría que se impuso con respecto a las otras fue la de Condicionales.

Tabla 12: Resultados de cuestionarios por categoría luego de la primera habilitación. Fuente: Elaboración Propia.

Categoría	Promedio	Cantidad
Programas Secuenciales	0.71	205
Condicionales	0.82	127
Ciclos	0.64	101
Strings	0.73	154
Funciones	0.55	80

4.2.7. Resultados de encuesta

Con el objetivo de recibir retroalimentación por parte de los estudiantes que participaron en la primera habilitación, se invitó a aquellos que tuvieron una alta y baja participación.

Para considerar a los estudiantes con alta participación, se tomaron en cuenta las siguientes condiciones:

- Estudiantes con mayor cantidad de envíos realizados.
- Estudiantes con mayor cantidad de insignias recibidas.

Por otro lado, para seleccionar a los estudiantes con baja participación, también se tomaron ambas condiciones mencionadas pero en el orden inverso.

Cada grupo estuvo compuesto por 10 estudiantes y se les envió un formulario a través de correo electrónico. El formulario consideró las siguientes preguntas:

Tabla 13: Preguntas para entrevista luego de la primera habilitación.
Fuente: Elaboración Propia.

Número	Pregunta
1	¿Qué avances obtuviste gracias al uso de la plataforma?
2	¿Recomendarías la plataforma a otros estudiantes? (sí o no y por qué)
3	¿Qué opinas de la usabilidad de la plataforma?
4	¿Qué mejoras propondrías para la plataforma?
5	Considerando SMOJ y Cody, ¿cuál de ellos crees tú que podría ayudarte más a lograr tus objetivos académicos y por qué?

En total, solo dos estudiantes respondieron la encuesta, uno de cada grupo.

El estudiante perteneciente el primer grupo mencionó que la utilización de la plataforma le contribuyó para prepararse antes de la primera evaluación formal del curso. También indicó que la idea de recibir problemas de manera personalizada le significaba un ahorro considerable de tiempo.

En cuanto al estudiante del segundo grupo, este manifestó al igual que el otro estudiante, que la plataforma le permitía evitar perder tiempo intentando buscar problemas que estuvieran adaptados a su nivel actual. Con respecto a la interfaz, la evaluó de manera positiva.

4.3. Segunda habilitación de la plataforma

La plataforma fue habilitada nuevamente 7 días antes de la segunda evaluación formal del curso, a diferencia de la primera vez que solo consideró 2 días. Para esta oportunidad, se añadieron dos nuevas categorías:

- Listas
- Diccionarios

4.3.1. Resultados generales

Los siguientes resultados reflejan la cantidad total para cada una de las descripciones listadas. La tercera columna indica el porcentaje de aumento con respecto a la primera vez que la plataforma fue habilitada.

Tabla 14: Resultados generales luego de la segunda habilitación.
Fuente: Elaboración Propia.

Descripción	Cantidad	Aumento
Usuarios registrados	834	97.63 %
Problemas disponibles	118	45.68 %
Envíos realizados	1596	146.30 %
Insignias recibidas	1265	119.24 %
Distribuciones generadas	1804	134.59 %
Feedbacks enviados	625	138.55 %
Intentos en cuestionarios	1573	135.83 %

Como puede observarse, la mayoría tuvo un incremento del más del 100 %, donde el primer lugar lo obtuvo la cantidad de envíos realizados. En general, la tabla demuestra claramente el aumento de la participación por parte de los estudiantes.

4.3.2. Resultados de envíos realizados

La categoría de Programas de Secuenciales nuevamente fue la que consiguió el primer lugar, por lo que se reafirma que esta es la categoría que la mayoría de los estudiantes seleccionan para comenzar a practicar.

Tabla 15: Resultados de envíos por categoría y nivel luego de la segunda habilitación.
Fuente: Elaboración Propia.

Categoría	Básico	Medio	Avanzado	Cantidad
Programas Secuenciales	291	322	143	756
Condicionales	103	64	72	239
Ciclos	39	18	17	74
Strings	55	28	19	102
Funciones	67	18	12	97
Listas	65	32	13	110
Diccionarios	106	75	37	218

Con respecto a los veredictos, la cantidad de tipo *Accepted* disminuyó en un 1.15 %, sin embargo, se mantuvo en el primer puesto al igual como en la habilitación pasada.

Tabla 16: Resultados de veredictos luego de la segunda habilitación.
Fuente: Elaboración Propia.

Veredicto	Porcentaje	Cantidad
Accepted	59.96 %	957
Wrong Answer	26.69 %	426
Runtime Error (NZEC)	13.28 %	212
Time Limit Exceeded	0.07 %	1

4.3.3. Resultados de insignias recibidas

La totalidad de insignias recibidas tuvieron un aumento con respecto a la ocasión pasada. La insignia más entregada fue la que se genera cuando el estudiante realiza su primer envío, independiente del tipo de veredicto que haya recibido.

Tabla 17: Resultados de insignias luego de la segunda habilitación.
Fuente: Elaboración Propia.

Descripción	Cantidad
Realizaste tu primer envío	290
Realizaste 5 envíos	105
Realizaste 10 envíos	50
Realizaste 15 envíos	23
Realizaste 20 envíos	12
Realizaste tu primer envío correcto	250
Realizaste 5 envíos correctos	65
Realizaste 10 envíos correctos	18
Realizaste 15 envíos correctos	5
Realizaste 20 envíos correctos	3
Categoría de Programas Secuenciales completa	42
Categoría de Condicionales completa	2
Categoría de Ciclos completa	2
Categoría de *Strings* completa	5
Categoría de Funciones completa	2
Perfil completado	128
Entregaste tu primer *feedback*	224
Entregaste 5 *feedbacks*	35
Entregaste 10 *feedbacks*	2
Entregaste 15 *feedbacks*	1
Entregaste 20 *feedbacks*	1

4.3.4. Resultados de distribuciones generadas

Sin considerar la categoría de Programas Secuenciales, todas las restantes estuvieron medianamente cercanas unas con otras con respecto al total de distribuciones generadas.

Tabla 18: Resultados de distribuciones por categoría y nivel luego de la segunda habilitación. Fuente: Elaboración Propia.

Categoría	Básico	Medio	Avanzado	Cantidad
Programas Secuenciales	20	157	306	483
Condicionales	10	33	206	249
Ciclos	20	72	108	200
Strings	6	62	170	238
Funciones	37	80	63	180
Listas	3	17	165	185
Diccionarios	28	93	148	269

4.3.5. Resultados de *feedbacks* entregados

Los *feedbacks* entregados también tuvieron un aumento con respecto a la habilitación pasada, donde la categoría de Programas Secuenciales mantuvo el primer puesto, con un incremento del 90.12 %.

Tabla 19: Resultados de *feedbacks* por categoría luego de la segunda habilitación. Fuente: Elaboración Propia.

Categoría	Cantidad
Programas Secuenciales	327
Condicionales	71
Ciclos	23
Strings	41
Funciones	29
Listas	51
Diccionarios	83

4.3.6. Resultados de cuestionarios intentados

En la primera habilitación, la categoría de Condicionales fue la que obtuvo el primer lugar con un promedio de 0.82, sin embargo, en esta segunda oportunidad la categoría de Listas fue la que se impuso con un promedio de 0.89, quedándose así con el primer puesto.

Tabla 20: Resultados de cuestionarios por categoría luego de la segunda habilitación. Fuente: Elaboración Propia.

Categoría	Promedio	Cantidad
Programas Secuenciales	0.70	365
Condicionales	0.81	222
Ciclos	0.67	189
Strings	0.73	220
Funciones	0.54	165
Listas	0.89	175
Diccionarios	0.66	238

4.3.7. Resultados de encuesta

Una encuesta fue implementada dentro de la plataforma para que los estudiantes la respondieran una vez que cumplían con alguna de las siguientes condiciones:

- La encuesta no había sido respondida antes.
- El total de envíos era mayor o igual a 3.
- El total de distribuciones generadas era mayor o igual a 4.

Los valores mínimos para el total de envíos y distribuciones tienen que ver con los resultados expuestos después de la primera habilitación, así que esos valores se seleccionaron pensando en lo se esperaba que tuviera un estudiante con una participación suficiente.

Cabe mencionar que la encuesta consideró las mismas 5 preguntas que se realizaron en la primera habilitación (Ver tabla 13). Además, 198 estudiantes la respondieron.

Con respecto a la primera pregunta, acerca de los avances que lograron los estudiantes gracias al uso de la plataforma, las opiniones con mayor frecuencia que se registraron se listan a continuación:

- Hubo una mejora con respecto a la resolución de problemas.
- Se logró poner en práctica lo aprendido en clases.
- Existió una mejor preparación para la evaluación formal del curso.
- Fue posible repasar los contenidos y retenerlos con mayor frecuencia.
- Se consiguió conocer el nivel de progreso actual a medida que se avanzó.

- Se obtuvo una mejor organización para ejercitar.

La otra parte de las respuestas consideradas como negativas solo reflejaron que la plataforma no les significó ningún avance, pero estas respuestas solo representaron el 8.54 % con respecto al total.

La segunda pregunta buscó conocer si los estudiantes recomendarían o no la plataforma. Aquí, el por qué de las opiniones afirmativas que más destacaron fueron las siguientes:

- Facilita la práctica.
- Permite conocer el progreso actual.
- Refresca los contenidos previos.
- Es útil para preparar futuras evaluaciones formales.
- Posibilita recibir retroalimentación.
- Beneficia la organización del estudio.

La siguiente tabla detalla la distribución entre estudiantes que sí considerarían recomendar la plataforma, así como aquellos que no lo harían y otros que están en una posición neutra o no contestaron.

Tabla 21: Distribución acerca de la recomendación de la plataforma.
Fuente: Elaboración Propia.

Respuesta	Cantidad
Sí	85.35 %
No	2.02 %
Neutro o no responde	12.63 %

La tercera pregunta de la encuesta trató sobre la usabilidad de la plataforma. Para explorar las opiniones, se lista a continuación las palabras más repetidas:

Tabla 22: Palabras más repetidas en opiniones acerca de la usabilidad.
Fuente: Elaboración Propia.

Palabra	Cantidad
Buena	62
Fácil	38
Intuitiva	9

Por otro lado, se obtuvieron las siguientes sugerencias:

- Desplegar el resultado de los cuestionarios una vez que se finaliza.
- Añadir videos para complementar las categorías.

La cuarta pregunta tuvo como finalidad conocer cuales aspectos podrían ser mejorados, pensando en una futura versión de la plataforma. A continuación se enumeran aquellas que tuvieron una amplía mayoría:

- Mostrar solución de los problemas una vez resueltos.
- Interfaz con modo oscuro.
- Añadir una sección con problemas para ser intentados de manera libre.
- Mejorar la retroalimentación que entregan los problemas.
- Añadir problemas más difíciles que los actuales.
- Integrar problemas de desarrollo o de selección múltiple.

Para la quinta y última pregunta, los estudiantes tuvieron que comparar entre SMOJ y Cody para determinar cuál de ellos sería más apropiado para conseguir los logros académicos deseados.

Un 57.58 % de los estudiantes indicó que preferiría Cody en vez de SMOJ, algunas de las principales razones encontradas en las opiniones se listan a continuación:

- Cody permite conocer el nivel actual de progreso.
- Cody tiene una interfaz más cómoda.
- Cody es sencillo de utilizar.

Otro 14.65 % decidió inclinarse hacia SMOJ. Aquellos estudiantes basaron su decisión mayoritariamente en las siguientes razones:

- SMOJ contiene más ejercicios por categoría.
- Existe una costumbre en el uso de SMOJ.

Por último, el 27.77 % restante se mantuvo en una posición neutral o no respondió.

4.4. Entrevista con profesores

Para finalizar el proceso de validación, se realizaron dos entrevistas a través una conversación, donde la finalidad fue obtener retroalimentación y que los profesores experimentaran el uso de la plataforma, recordando que ellos tienen acceso adicional a dos vistas de administración.

4.4.1. Primera entrevista

Para la primera entrevista, se consideró a un profesor que contaba con una amplía experiencia en el uso de SMOJ.

La opinión acerca de la interfaz fue positiva, destacando los colores y la ubicación de los distintos elementos. Además, se recibieron las siguientes sugerencias que podrían ser integradas:

- En la vista para administrar paralelos, incluir un filtro que relacione los campus con sus respectivos paralelos.
- En la vista para administrar paralelos, agrupar las secciones de una manera distinta para evitar hacer *scroll*.
- En la vista para administrar problemas, incluir un filtro por categoría.
- En vez de tener dos botones en la vista de los problemas para probar y enviar código, podría establecerse la regla de que si los casos de prueba han sido resueltos, entonces se envía automáticamente.

En cuanto a las funcionalidades, estas fueron valoradas como eficaces. Se recibieron las recomendaciones sobre nuevas implementaciones que podrían realizarse:

- Incluir estadísticas específicas para cada problema.
- Implementar acciones para crear, editar y eliminar problemas.
- Añadir una categoría nueva que incluya problemas más difíciles que los actuales, para así motivar a estudiantes más avanzados.
- Revisar los parámetros de la función *input* para que los estudiantes no envíen código que incluya mensajes adicionales.
- Ofrecer una retroalimentación más descriptiva si se encuentran errores por parte del intérprete de Python que son fácilmente identificables.

Una recomendación que es necesaria destacar tiene que ver con el hecho de actualizar el nivel de dificultad para cada problema. La implementación actual utiliza el promedio que proviene desde los *feedbacks* que los estudiantes y profesores envían. Sin embargo, el entrevistado sugiere analizar en dos partes los *feedbacks*, en un grupo se considerarían los que provienen de estudiantes y en otro el de profesores. Esto se haría así porque podría darse el caso en que ambos grupos tendieran a extremos opuestos, por lo que sería importante observar eso.

4.4.2. Segunda entrevista

La segunda entrevista tuvo como participante a una profesora con poca experiencia en jueces como SMOJ.

La interfaz la consideró adecuada, resaltando los colores y el estilo del texto. Se recibieron las siguientes recomendaciones, específicamente para la vista de administración de paralelos:

- Añadir un filtro que permita revisar las distribuciones a lo largo del tiempo.
- Añadir cada una de las categorías en la tabla de las insignias entregadas.
- Solo desplegar las insignias con representen las cantidades mayores.

Con respecto a la funcionalidad de la plataforma, esta también tuvo un recibimiento positivo y debido a eso no se entregaron observaciones.

CAPÍTULO 5
CONCLUSIONES

5.1. Conclusiones Generales

En esta Memoria se estudió cómo los estudiantes de un curso masivo de programación gestionan su tiempo de practica en plataformas que evalúan soluciones de manera automática. A partir de lo anterior se diseñó y desarrolló una plataforma tecnológica que fuera capaz de proveer ejercicios de programación ajustados al nivel de progreso de los estudiantes, para que estos últimos no tuvieran la necesidad de clasificarlos por su cuenta y para que así consiguieran una mejor gestión y ahorro de su tiempo.

La plataforma *web* basó su desarrollo en una arquitectura monolítica, empleando la metodología por capas, donde la capa de datos consistió en un motor de base de datos MySQL, alojado en una instancia de RDS en AWS. Por otro lado, la capa de negocio se trabajó con el lenguaje de programación Python, a través del micro *framework* Flask. La capa de presentación en tanto, fue implementada con el motor de plantillas Jinja2. Tanto la capa de negocio como de presentación, se integraron en una única instancia de EC2 de AWS. Para la tarea de juzgar código de manera automática, se optó por la utilización de un servicio externo llamado Judge0 que pudo ser añadido a otra instancia de EC2 dentro de AWS.

Las principales funcionalidades implementadas están orientadas hacia el estudiante, el cual en una primera instancia tiene la posibilidad de responder un cuestionario para una categoría dada. Luego que el cuestionario es respondido, una distribución de ejercicios es presentada para que comience su práctica. A medida que trabaja en los problemas, su nivel de progreso puede ir disminuyendo o aumentando, para que así regularmente obtenga una nueva distribución de ejercicios, siempre ajustados a su nivel actual. El profesor en tanto, cuenta con la posibilidad de revisar estadísticas generales acerca de los paralelos que se encuentran en la plataforma, con la finalidad de que se mantenga informado sobre el avance de estos.

Para conocer la percepción de los estudiantes hacia la plataforma, esta se habilitó en 2 ocasiones. La primera ocasión ocurrió 2 días antes de la primera evaluación del curso. Una vez que finalizó dicha evaluación, se seleccionaron a 20 estudiantes para que respondieran una encuesta, donde 10 de ellos hicieron un uso frecuente de la plataforma, mientras que los otros 10 restantes fue muy poco fue lo que usaron. La segunda vez que la plataforma fue habilitada, se permitió su uso 7 días antes de la segunda evaluación del curso. En esta oportunidad, una encuesta se desplegó a cada estudiante que cumplió ciertas reglas de participación.

Tanto la encuesta que se desplegó para la primera y segunda habilitación tuvieron las mismas preguntas. Para ambas ocasiones, los resultados demuestran que los estudiantes lograron una serie de beneficios, tales como una mejor organización a la hora de ejercitar, poner en

práctica lo aprendido en clases, preparar de mejor manera las evaluaciones, entre otros. También se recibieron sugerencias para mejorar la interfaz, así como también para añadir nuevas funcionalidades.

Para complementar la validación, se entrevistó a dos profesores del Departamento de Informática de la Universidad Técnica Federico Santa María. Uno de ellos contaba con una experiencia previa en jueces en línea, mientras que el otro no lo suficiente. Ambos profesores destacaron las distintas funcionalidades e interfaces. Adicionalmente, el primero aportó con sugerencias para implementar nuevas características en las interfaces de administración, así como también nuevas funcionalidades orientadas hacia el beneficio pedagógico de los estudiantes.

5.2. Consideraciones y Limitaciones

La arquitectura que respalda a la plataforma desarrollada fue de tipo monolítica, lo cual está bien para los cerca de 1300 estudiantes que cursan la asignatura de programación cada semestre. Sin embargo, no existe garantía en términos de escalabilidad debido al tipo de arquitectura mencionada.

Tanto para la evaluación del prototipo como la primera habilitación tuvieron poca participación por parte de los estudiantes, por lo que se desearía que hubiese existido una participación mayor para así haber recibido mucha más retroalimentación que permitiera mejorar la plataforma antes de la segunda habilitación.

Una consideración a tomar en cuenta es que no se realizó un estudio acerca de la efectividad del diseño de la interfaz, así como tampoco la experiencia de usuario, por lo que no se cuenta con suficiente certeza para saber si la selección de los colores, los mensajes y la distribución de las secciones fueron las más apropiadas para el público objetivo.

Por último, tampoco se formalizaron y realizaron pruebas unitarias, de integración o de carga. Esta fase del desarrollo más bien consistió en pequeños casos de prueba para verificar el correcto funcionamiento, pero no se establecieron pruebas suficientes como para garantizar que la plataforma se encontrara lo más libre de *bugs* posible.

5.3. Trabajo a futuro

Una de las tareas que quedan pendientes para toda aquella persona que quiera profundizar más en el tema de esta Memoria, es desarrollar una arquitectura que permita la escalabilidad, para así permitir que muchísimos más estudiantes puedan utilizar la plataforma sin ningún tipo de inconveniente técnico. Un primer enfoque podría ser la utilización de balanceadores de carga y el uso de *clusters*.

También, se sugiere la implementación de algún servicio que sea capaz de detectar altas similitudes de código, para que así los estudiantes no tomen ventaja de manera irregular sobre otros que realizan un trabajo honesto.

Para finalizar, otra característica que sería interesante desarrollar, es la integración de esta plataforma en otras instituciones educativas, para de esta manera evaluar la flexibilidad que presenta al incorporar contenidos similares o extras, así como también el desarrollo de interfaces y funcionalidades que se ajusten a los nuevos requerimientos.

REFERENCIAS BIBLIOGRÁFICAS

[Alexandron *et al.*, 2012] Alexandron, G., Armoni, M., Gordon, M., y Harel, D. (2012). The effect of previous programming experience on the learning of scenario-based programming. En *Proceedings of the 12th Koli Calling International Conference on Computing Education Research*, pp. 151–159.

[Atiq, 2018] Atiq, Z. (2018). Emotions experienced by first-year engineering students during programming tasks. En *Proceedings of the 2018 ACM conference on international computing education research*, pp. 258–259.

[Cavero, 2011] Cavero, M. Á. B. (2011). Voluntad para estudiar, regulación del esfuerzo, gestión eficaz del tiempo y rendimiento académico en alumnos universitarios. *Revista de investigación educativa*, 29(1):171–185.

[CNED, 2022] CNED (2022). Base de datos de pregrado (2005-2022) y posgrado (2005-2021) por programa. `https://www.cned.cl/estadistica/indices-bd-matricula`. Accedido: 21-08-2022.

[Derus y Ali, 2012] Derus, S. y Ali, A. M. (2012). Difficulties in learning programming: Views of students. En *1st International Conference on Current Issues in Education (ICCIE 2012)*, pp. 74–79.

[Dictionary, 2022] Dictionary, M. W. (2022). Definition of System. `https://merriam-webster.com/dictionary/system`. Accedido: 08-08-2022.

[Došilović y Mekterović, 2020] Došilović, H. Z. y Mekterović, I. (2020). Robust and scalable online code execution system. En *2020 43rd International Convention on Information, Communication and Electronic Technology (MIPRO)*, pp. 1627–1632. IEEE.

[Feierherd *et al.*, 2001] Feierherd, G. E., Depetris, B. O., y Jerez, M. (2001). Una evaluación sobre la incorporación temprana de algorítmica y programación en el ingreso a informática. En *VII Congreso Argentino de Ciencias de la Computación*.

[Fincher, 1999] Fincher, S. (1999). What are we doing when we teach programming? En *FIE'99 Frontiers in Education. 29th Annual Frontiers in Education Conference. Designing the Future of Science and Engineering Education. Conference Proceedings (IEEE Cat. No. 99CH37011*, volumen 1, pp. 12A4–1. IEEE.

[Fromme *et al.*, 2008] Fromme, K., Corbin, W. R., y Kruse, M. I. (2008). Behavioral risks during the transition from high school to college. *Developmental psychology*, 44(5):1497.

[Gomes y Mendes, 2010] Gomes, A. J. y Mendes, A. J. (2010). A study on student performance in first year cs courses. En *Proceedings of the fifteenth annual conference on Innovation and technology in computer science education*, pp. 113–117.

[González *et al.*, 2008] González, H. M., Duque, N. D., y Ovalle, D. A. (2008). Modelo del estudiante para sistemas adaptativos de educación virtual. *Revista Avances en sistemas e informática*, 5(1):199–206.

[Hao *et al.*, 2022] Hao, Q., Smith IV, D. H., Ding, L., Ko, A., Ottaway, C., Wilson, J., Arakawa, K. H., Turcan, A., Poehlman, T., y Greer, T. (2022). Towards understanding the effective design of automated formative feedback for programming assignments. *Computer Science Education*, 32(1):105–127.

[Hegarty-Kelly y Mooney, 2021] Hegarty-Kelly, E. y Mooney, D. A. (2021). Analysis of an automatic grading system within first year computer science programming modules. En *Computing Education Practice 2021*, pp. 17–20.

[Hidalgo-Céspedes *et al.*, 2020] Hidalgo-Céspedes, J., Marín-Raventós, G., y Calderón-Campos, M. E. (2020). Online judge support for programming teaching. En *2020 XLVI Latin American Computing Conference (CLEI)*, pp. 522–530. IEEE.

[Institute, 2022] Institute, N. E. C. S. (2022). Definition of Adaptive System. `https://necsi.edu/adaptive`. Accedido: 08-08-2022.

[Johnson, 2021] Johnson, L. (2021). What is a system? *Student Works*.

[Kinnunen y Malmi, 2006] Kinnunen, P. y Malmi, L. (2006). Why students drop out cs1 course? En *Proceedings of the second international workshop on Computing education research*, pp. 97–108.

[Kraiger *et al.*, 1995] Kraiger, K., Salas, E., y Cannon-Bowers, J. A. (1995). Measuring knowledge organization as a method for assessing learning during training. *Human factors*, 37(4):804–816.

[Luna *et al.*, 2007] Luna, C. D., Pedemonte, M., Viera, M., y Fraschini, E. (2007). Organización para un curso de programación en un contexto de masividad. *Revista Iberoamericana de Tecnología en Educación y Educación en Tecnología*, (2):83–91.

[Marcén y Martínez-Caraballo, 2012] Marcén, M. y Martínez-Caraballo, N. (2012). Gestión eficiente del tiempo de los universitarios: evidencias para estudiantes de primer curso de la universidad de zaragoza. *Innovar*, 22(43):105–130.

[Matthíasdóttir, 2006] Matthíasdóttir, Á. (2006). How to teach programming languages to novice students? lecturing or not. En *International Conference on Computer Systems and Technologies-CompSysTech*, volumen 6, pp. 15–16.

[MIFUTURO, 2022] MIFUTURO (2022). Bases de datos de matriculados en educación superior, entre los años 2007 y 2022, tanto de pregrado, como posgrado y postítulo. `https://www.mifuturo.cl/bases-de-datos-de-matriculados/`. Accedido: 21-08-2022.

[Milne y Rowe, 2002] Milne, I. y Rowe, G. (2002). Difficulties in learning and teaching programming—views of students and tutors. *Education and Information technologies*, 7(1):55–66.

[Narciss y Huth, 2004] Narciss, S. y Huth, K. (2004). How to design informative tutoring feedback for multimedia learning. *Instructional design for multimedia learning*, 181195.

[Nasrullah_PhD y Khan_PhD, 2015] Nasrullah_PhD, S. y Khan_PhD, M. S. (2015). The impact of time management on the students' academic achievements.

[Reyes-González *et al.*, 2022] Reyes-González, N., Meneses-Báez, A. L., y Díaz-Mujica, A. (2022). Planificación y gestión del tiempo académico de estudiantes universitarios. *Formación universitaria*, 15(1):57–72.

[Shackelford, 1998] Shackelford, R. (1998). En *Introduction to Computing and Algorithms*.

[Vásquez *et al.*, 2021] Vásquez, A., Meza, F., y Barrera, P. G. (2021). Emergency remote teaching model for massive programming classes. En *2021 XLVII Latin American Computing Conference (CLEI)*, pp. 1–9. IEEE.

[Wasik *et al.*, 2018] Wasik, S., Antczak, M., Badura, J., Laskowski, A., y Sternal, T. (2018). A survey on online judge systems and their applications. *ACM Computing Surveys (CSUR)*, 51(1):1–34.

[Xu *et al.*, 2020] Xu, B., Yan, S., Jiang, X., y Feng, S. (2020). Scfh: A student analysis model to identify students' programming levels in online judge systems. *Symmetry*, 12(4):601.

[Yuangga y Sunarsi, 2018] Yuangga, K. D. y Sunarsi, D. (2018). The influence of procrastination and low time management on student self efficacy (at ma soebono mantofani). *PINISI Discretion Review*, 2(1):85–92.

[Zhou *et al.*, 2018] Zhou, W., Pan, Y., Zhou, Y., y Sun, G. (2018). The framework of a new online judge system for programming education. En *Proceedings of ACM turing celebration conference-China*, pp. 9–14.