RESUMEN TRABAJO DE GRADO

AUTOR(ES):

NOMBRE(S): ANGIE MELISSA **APELLIDOS**: DELGADO LEÓN

NOMBRE(S): GERSON YESID **APELLIDOS**: LÁZARO CARRILLO

FACULTAD: INGENIERÍA

PLAN DE ESTUDIOS: INGENIERÍA DE SISTEMAS

DIRECTOR:

NOMBRE(S): FREDY HUMBERTO **APELLIDOS**: VERA RIVERA

TÍTULO DEL TRABAJO (TESIS): DESARROLLO E IMPLEMENTACIÓN DE UN MARCO DE TRABAJO PARA EL GRUPO DE ESTUDIO EN PROGRAMACIÓN COMPETITIVA DEL PROGRAMA DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD FRANCISCO DE PAULA SANTANDER.

RESUMEN:

El presente proyecto presenta un marco de trabajo desarrollado para ser implementado en el grupo de estudio de Programación Competitiva del programa Ingeniería de Sistemas de la UFPS. En él se tienen en cuenta aspectos metodológicos, informáticos y específicos de las competencias de programación para crear diferentes componentes informáticos y organizacionales que permitan el mejoramiento continuo de dicho grupo. Todos los componentes se integran en un marco de trabajo donde cada elemento cumple una función específica, y cada labor realizada es medida para analizar su utilidad. Entre los componentes creados encontramos una metodología de trabajo para el manejo del grupo, una plataforma de Software (Training Center), más de 20 videos con explicaciones de algunas de las temáticas necesarias para las competencias, manuales de competencia con más de 80 algoritmos en java y C++, y presentaciones explicativas de más de 20 temáticas utilizadas en las competencias.

PALABRAS CLAVE: Programación Competitiva, ICPC, Programación, Maratones de Programación, Algoritmos

CARACTERISTICAS:

PÁGINAS: 223 **PLANOS**: ___ **ILUSTRACIONES**: 20_ **CD ROM**: 1

DESARROLLO E IMPLEMENTACIÓN DE UN MARCO DE TRABAJO PARA EL GRUPO DE ESTUDIO EN PROGRAMACIÓN COMPETITIVA DEL PROGRAMA DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD FRANCISCO DE PAULA SANTANDER

ANGIE MELISSA DELGADO LEÓN

GERSON YESID LÁZARO CARRILLO

UNIVERSIDAD FRANCISCO DE PAULA SANTANDER

FACULTAD DE INGENIERÍA

PLAN DE ESTUDIOS DE INGENIERÍA DE SISTEMAS

SAN JOSÉ DE CÚCUTA

2018

DESARROLLO E IMPLEMENTACIÓN DE UN MARCO DE TRABAJO PARA EL GRUPO DE ESTUDIO EN PROGRAMACIÓN COMPETITIVA DEL PROGRAMA DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD FRANCISCO DE PAULA SANTANDER

ANGIE MELISSA DELGADO LEÓN

GERSON YESID LÁZARO CARRILLO

Trabajo de grado para optar al título de Ingenieros de Sistemas

DIRECTOR

FREDY HUMBERTO VERA RIVERA

INGENIERO DE SISTEMAS

Magíster en ingeniería área informática y ciencias de la computación

UNIVERSIDAD FRANCISCO DE PAULA SANTANDER

FACULTAD DE INGENIERÍA

PLAN DE ESTUDIOS DE INGENIERÍA DE SISTEMAS

SAN JOSÉ DE CÚCUTA

2018

Universidad
Francisco de Paula Santander

NIT. 890500622 - 6

ACTA DE SUSTENTACIÓN DE TRABAJO DE GRADO

www.ufps.edu.co

FECHA: *05 DE ABRIL DE 2018*
HORA: *2:30 P. M.*
LUGAR: *4 PISO AULA SUR - UFPS*
PLAN DE ESTUDIOS: *INGENIERÍA DE SISTEMAS*

TÍTULO DEL TRABAJO DE GRADO: **"DESARROLLO E IMPLEMENTACIÓN DE UN MARCO DE TRABAJO PARA EL GRUPO DE ESTUDIO EN PROGRAMACIÓN COMPETITIVA DEL PROGRAMA DE INGENIERÍA DE SISTEMAS DE LA UNIVERSIDAD FRANCISCO DE PAULA SANTANDER"**

ING. MARÍA DEL PILAR ROJAS PUENTES
ING.JAIRO FUENTES CAMARGO
ING.MILTON JESÚS VERA CONTRERAS

DIRECTOR: FREDY HUMBERTO VERA RIVERA

NOMBRE DEL ESTUDIANTE	CÓDIGO	CALIFICACIÓN	
		NÚMERO	LETRA
ANGIE MELISSA DELGADO LEÓN	1150990	5,0	CINCO, CERO
GERSON YESID LÁZARO CARRILLO	1150972	5,0	CINCO, CERO

LAUREADA

FIRMA DE LOS JURADOS

FIRMA DE LOS JURADOS

ING. MARÍA DEL PILAR ROJAS PUENTES

ING.JAIRO FUENTES CAMARGO

ING.MILTON JESÚS VERA CONTRERAS

OSCAR ALBERTO GALLARDO PÉREZ
Coordinador Comité Curricular

Dedicatoria

A todos y cada uno de los estudiantes que han hecho parte del grupo de estudio en Programación Competitiva UFPS. Ellos hacen parte de este proyecto.

Agradecimientos

Agradecimiento especial a todo el Programa de Ingeniería de Sistemas, especialmente a los ingenieros Fredy Vera y Milton Vera quienes han dirigido el grupo desde su creación. A los ingenieros Hugo Morales y Fabio Avellaneda, quienes a través de la Red de Programación Competitiva fueron un apoyo constante a este proyecto. A la Liga Colombiana de Programación, cuyos resultados fueron fundamentales en nuestros análisis. A la comunidad de programación competitiva en Colombia, compuesta por alumnos y exalumnos siempre dispuestos a ayudar a resolver cualquier duda que se presentara. Y muy especialmente, a aquellos estudiantes del programa que creyeron en nuestra iniciativa y se unieron al grupo de estudio brindándonos su tiempo y trabajando fuerte para que el nombre de la Universidad sea referente en competencias de programación.

Contenido

Lista de tablas

Lista de figuras

Lista de anexos

Resumen

El presente proyecto presenta un marco de trabajo desarrollado para ser implementado en el grupo de estudio de Programación Competitiva. En él se tienen en cuenta aspectos metodológicos, informáticos y específicos de las competencias de programación para crear diferentes componentes tanto a nivel informático como a nivel organizacional que permita el mejoramiento continuo de dicho grupo. Todos los módulos propuestos se integran al final en un único marco de trabajo donde cada elemento cumple una función específica, y cada labor realizada es medida para analizar su utilidad.

Se espera que la implementación de este marco de trabajo en el grupo de estudio permita la mejora a corto y largo plazo de los resultados obtenidos en maratones de programación por estudiantes de la carrera a nivel nacional e internacional. En el tiempo que lleva este proyecto en desarrollo, versiones preliminares del marco de trabajo fueron implementadas en el grupo de estudio y los resultados hasta el momento han sido completamente satisfactorios:

Como parte de los resultados que se lograron con la realización de este proyecto contamos con la publicación del articulo Desarrollo e implementación de un marco de trabajo para el entrenamiento en programación competitiva en la revista Universidad, ciencia y tecnología de la Universidad Nacional Experimental Politécnica Antonio José de Sucre, la construcción de más de 20 videos con explicaciones de algunas de las temáticas necesarias para las competencias, notebooks con más de 80 algoritmos en java y C++, presentaciones explicativas de más de 20 temáticas utilizadas en las competencias, la organización de 4 maratones UFPS (2 de ellas con ejercicios completamente inéditos escritos por el equipo organizador de la universidad y que fueron utilizados por toda Latinoamérica), la participación por 4 años consecutivos en la maratón

nacional y latinoamericana de programación ACM-ICPC y la plataforma de entrenamiento para programación competitiva UFPS – Training center.

Introducción

Durante los últimos años, diferentes estudiantes de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander han representado a la institución en competencias de programación, a nivel local, nacional y continental. Estas competencias son por lo general pruebas en las cuales se miden diferentes habilidades de los estudiantes de Ingeniería de Sistemas y carreras afines: Trabajo en equipo, estructuras de datos, paradigmas de desarrollo, utilización de conocimientos en otras áreas (matemáticas, física, etc.), entre otras (Bloomfield & Sotomayor, 2016). Aunque existen multitud de competencias, la más famosa a nivel Universitaria (Competencia ACM ICPC) nos permite hacernos una idea de la importancia que este evento tiene a nivel mundial: 50145 estudiantes, de 2948 universidades en 103 países participaron en la última edición (ICPC, 2017).

El desempeño para la Universidad en estas competencias ha sido bueno, abriéndose en poco tiempo un espacio entre las principales Universidades que generalmente disputan estas competencias, y empezando a obtener reconocimiento por su trabajo. Sin embargo, no basta la dedicación de algunos estudiantes y los conocimientos adquiridos en el aula de clase para alcanzar los primeros puestos. Para obtener los mejores resultados hace falta tener un plan de estudio serio, una motivación general, una estrategia definida para mejorar, retroalimentación en el proceso, y mucha constancia por parte de los estudiantes.

Por estos motivos, luego de analizar materiales de diversa índole con los cuales diferentes universidades de Colombia y el mundo afrontan estas competencias - especialmente el material de aquellas universidades que son referencia en la materia - hemos emprendido la tarea de

diseñar un marco de trabajo que ayude a los estudiantes que hacen parte de este proceso a formar equipos cada vez más competitivos, que logren mejores resultados cada año.

Diseñar este marco de trabajo ha supuesto varios retos, pero al mismo tiempo ha sido una tarea sumamente enriquecedora para quienes realizamos el proyecto, y esperamos que sea igualmente enriquecedora para todos aquellos que la aprovechen en los años sucesivos. El primer reto consistió en diseñar un marco de trabajo al mismo tiempo que se aplicaba en el grupo de estudio. Originalmente este proyecto se inició como un trabajo de aula en el segundo semestre de 2015, convirtiéndose en una iniciativa de proyecto de grado para mediados del 2016. Sin embargo, desde el mismo 2015 se creó el grupo de estudio en Programación Competitiva, y por tanto cada uno de los elementos que fueron incorporándose en este documento, también fueron probándose en un ambiente real, con estudiantes que estaban deseosos de aprender y quienes enriquecieron este proceso en cada momento. Por eso se debe adoptar una metodología que permita investigar, proponer, sintetizar, poner en práctica, y concluir a raíz de los resultados en diferentes iteraciones, pues desde el estado inicial del proyecto se contó con el grupo de estudio.

El siguiente reto lo supuso la temática que debían manejar los estudiantes para superar sus expectativas en competencias. Si bien muchas de estas temáticas son cubiertas en las áreas de programación que se dictan en la carrera, muchas otras son profundizaciones de temas específicos que no hacen parte del pensum y que, por tanto, debíamos interiorizar desde otras fuentes. Además, las temáticas que aparecen en una competencia pueden ser tan extensas que sería difícil sintetizarlas todas, y por tanto era necesario establecer un currículo que aportara al estudiante unas bases sólidas para poder desempeñarse en estas competencias. En este punto el

apoyo constante de diferentes docentes de la carrera fue fundamental, pues sus conocimientos tanto en docencia como en las temáticas en cuestión fueron de valiosa ayuda.

Junto a estos aparecieron otros cientos de retos que se encuentran documentados a lo largo de este texto. Algunos más serios que otros, pero todos sin duda enriquecieron este proceso, y permitieron construir un marco de trabajo funcional para el grupo de estudio.

Como no podía ser de otra forma, realizar este proyecto implicó buscar soluciones tanto en metodologías de enseñanza tradicionales, como en la integración de tecnologías de la información para potenciar el aprendizaje. El hecho de haber iniciado el grupo de estudio al mismo tiempo que diseñábamos el marco de trabajo nos permitió visualizar algo importante: Más allá de las competencias, la plataforma en torno a la que gira el marco de trabajo podía ser un complemento también de las clases tradicionales. Las áreas relacionadas con los algoritmos y la programación - que no son pocas en el pensum - no solo son la base de las competencias de programación, también pueden ir de la mano con ellas, sin necesidad de modificar la forma en que son dictadas, siendo un apoyo tecnológico que permite al estudiante fortalecer su conocimiento, mientras el docente lleva un fácil seguimiento.

A continuación, se presenta una descripción del proyecto, su desarrollo y la forma en que este se ha llevado a cabo de forma evolutiva, hasta su fase actual, que sin duda puede seguir siendo actualizada conforme el tiempo traiga nuevas tecnologías, pero manteniendo siempre su base fundamental.

1. Presentación general del Proyecto

El grupo de estudio en Programación Competitiva hace parte del semillero de investigación en Linux y Desarrollo de Software Libre (SILUX) del Programa Ingeniería de Sistemas en la Universidad Francisco de Paula Santander. La línea principal de este grupo de estudio es preparar a los estudiantes del programa para representar a la Universidad en diferentes competencias de programación que se realizan a nivel local, nacional e internacional.

Este grupo es relativamente reciente - inició en el año 2015, como parte de un proyecto de aula y semillero, y por tanto se encuentra aún en fase de consolidación. Una parte fundamental en esta consolidación es contar con un marco de trabajo que permita una mayor apropiación de conocimiento de sus estudiantes y un mejor desempeño en las diferentes competencias que se realizan. Justamente, es ese el propósito de este proyecto.

Hasta el día de hoy, el grupo de estudio ha alcanzado logros importantes. Si bien estos logros se detallarán más adelante, a manera de resumen, podemos contar la participación por 4 años consecutivos en la Maratón Nacional de Programación, clasificación en estos 4 años a fase latinoamericana, y muy buenos resultados en la Liga Colombiana de Programación y la Red de Programación Competitiva.

Al plantear un marco de trabajo para el grupo de estudio, se busca que estos resultados no se vean interrumpidos, sino que se conviertan en una constante, a pesar del cambio de nombres de los estudiantes que hacen parte de la representación. Por obvias razones, con el paso del tiempo, los estudiantes que componen el grupo de estudio van cambiando, y es aquí donde el marco de trabajo formalmente definido cobra importancia para que esto no resulte un inconveniente, y no haya que empezar de cero en cada ocasión.

2. Planteamiento del problema

Las Maratones de Programación son competencias que ponen a prueba la capacidad de los estudiantes para dar solución a un conjunto de problemas en un espacio de tiempo determinado (Halim & Halim, Competitive Programming, 2013). Los problemas que se plantean en una maratón de programación no se centran en un solo tema específico, sino que abarcan un conjunto variado de temáticas de diferentes áreas del conocimiento y niveles de dificultades. Debido a esto es necesario que los participantes tengan un amplio conjunto de conocimientos que les permita enfrentar problemas de cualquier temática, habilidades de análisis de algoritmos para analizar en tiempo real si la solución propuesta es eficiente para la problemática en cuestión y habilidades asociativas para trabajar en equipo sin dificultad por varias horas seguidas, con la tensión que estas competencias implican.

Las Maratones de Programación como competencias grupales en las cuales los estudiantes se enfrentan a diferentes problemas que deben solucionar a través de un programa informático, ineludiblemente fomentan el aprendizaje colaborativo (Universidad EAFIT, 2014) y el aprendizaje basado en problemas (Vizcarro, y otros, 2009) (Servicio de Innovación Educativa de la Universidad Politécnica de Madrid , 2008), donde la construcción del conocimiento no recae exclusivamente en un docente o tutor, sino que surge del trabajo en equipo y el constante compartir conocimientos entre los miembros del equipo. El grupo de estudio en programación competitiva de la Universidad Francisco de Paula Santander surge precisamente como una manera de incentivar este tipo de aprendizaje dentro de las aulas de la universidad con el fin de fortalecer la preparación de los estudiantes en diferentes temáticas que puedan ser aplicadas en una maratón de programación y en su propia carrera. Estas temáticas cubren temas de programación, matemáticas, lógica, entre otros.

El grupo de estudio en programación competitiva de la Universidad Francisco de Paula Santander es actualmente parte del semillero SILUX (Semillero de Investigación en Linux y Desarrollo de Software Libre) y es dirigido de manera cooperativa entre estudiantes que han participado en competencias ACM-ICPC Nacionales y Regionales junto con un docente quien además de coordinar esfuerzos en la parte administrativa del grupo, oficia como coach (director) de los equipos que representan a la universidad en las diferentes competencias en las cuales participan. Este grupo se encuentra abierto a todos aquellos estudiantes de Ingeniería de Sistemas y otras carreras con habilidades en programación quienes pueden participar de forma abierta en las diferentes actividades desarrolladas dentro del grupo.

Desde la creación del grupo de estudio en febrero del 2015 se ha participado activamente en redes de entrenamiento como la Red de Programación Competitiva (RPC) y la Colombian Collegiate Programming League (CCPL) en las cuales los estudiantes del grupo compiten y se preparan junto a otras Universidades del país y del continente. Junto con estos entrenamientos se desarrolla un conjunto de actividades como charlas, sesiones de entrenamiento y asesorías en algunas de las temáticas necesarias en las competencias, todo esto con base en los conocimientos previos de los participantes del grupo y siempre tomando en cuenta el desempeño general de los participantes así como sus opiniones para definir una estructura para las sesiones y sus actividades complementarias acorde a las fortalezas, debilidades y necesidades de los estudiantes.

Desde la formación del grupo, se ha buscado que los estudiantes fortalezcan sus habilidades en programación, lo cual será benéfico no solo para las maratones de programación sino también para su vida profesional, y al lograr dicho fortalecimiento, el desempeño de la Universidad en estas competencias será cada vez más notorio, buscando ocupar con el paso del tiempo mejores

puestos que consoliden a la Universidad como referente a nivel nacional e internacional en estas competencias. Por esta razón, desde el inicio del grupo se ha realizado constante retroalimentación con los alumnos participantes, consenso con los docentes que han oficiado como coach, y opinión de participantes y coach de otras universidades referentes a nivel nacional para responder la pregunta crucial de la cual surge esta investigación a partir de lo previamente expuesto: **¿Cómo mejorar las habilidades algorítmicas de los estudiantes del programa de Ingeniería de Sistemas permitiéndoles enfrentarse a maratones de programación con resultados satisfactorios?**

Es en este punto donde surge la necesidad de plantear mecanismos que dicten las estrategias y pasos a seguir por todos los participantes del grupo, que permitan establecer objetivos para los estudiantes del grupo de estudio (y para el propio grupo como tal), y escalar en los resultados obtenidos frente a las demás universidades de la región y del país.

3. Justificación

Como se ha expuesto previamente, el grupo de estudio en programación competitiva busca continuamente preparar a los estudiantes de manera efectiva para obtener los mejores resultados en competencias de programación, de manera que en el ámbito personal los participantes puedan alcanzar nuevos logros académicos (que pueden tener repercusión en su futuro laboral), mientras la institución adquiere reconocimiento por la representación realizada. En años previos la participación de la Universidad en estos eventos ha sido satisfactoria, pero para lograr un amplio reconocimiento y figurar en la élite nacional e internacional en competencias de programación, se hace necesario tomar medidas adicionales en la forma en la que este proceso de entrenamiento se realiza. Por lo tanto, es necesario definir estrategias claras y concisas que permitan llevar a cabo esta mejora continua, como un proceso pedagógico donde se tengan en cuenta múltiples factores que pueden influir en esta preparación.

Un punto de partida al construir este proyecto, es que las estrategias adoptadas en busca del mejoramiento de los estudiantes en competencias de programación, no pueden ser estrategias aisladas, independientes unas de otras. Somos conscientes de las múltiples formas en las cuales puede enfocarse esta preparación (recopilación de material documental para estudio, creación de competencias internas, clases magistrales preparadas, solo por nombrar algunas). Pero para que la preparación sea integral y completa, es necesario un punto de encuentro donde diferentes estrategias y herramientas de aprendizaje puedan integrarse y trabajar de la mano en busca de mejores resultados.

Es por esto que la construcción de un marco de trabajo completo que oriente los procesos realizados en el grupo de estudio de programación competitiva es necesario para formalizar el proceso de entrenamiento a través de la preparación, adecuación y aplicación de técnicas,

metodologías y herramientas que se adapten al grupo de estudio. Para lograr esta adaptación, el proyecto debe contemplar las siguientes características del grupo:

- **Es abierto**: El grupo de estudio no es un espacio con matrícula previa, sino un espacio abierto a todo el que desee aprender.
- **Es heterogéneo**: Los estudiantes del grupo no pertenecen a un semestre específico (pueden hacer parte estudiantes del primero al último semestre), ni deben cumplir alguna materia como prerrequisito para ingresar.
- **Es un espacio colaborativo**: Se busca que el grupo de estudio sea un espacio donde todo aquel que quiera aportar pueda hacerlo.

Desde este punto de vista, es necesario que el proyecto se resuma en la creación de un marco de trabajo completo, que defina las herramientas y materiales, la metodología de trabajo y las prácticas que se acogerán para el trabajo dentro del grupo, que tome como punto de partida estas características y se acople a ellas para lograr un mejor desempeño en competencias de programación mientras se fortalecen las habilidades en programación de los estudiantes.

Este marco de trabajo englobaría una colección de múltiples componentes (humano, software, documental, de aprendizaje, de colaboración, entre otros) articulados mediante una completa metodología establecida y definida en cada una de sus etapas, girando alrededor de una plataforma de software que permita llevar a cabo los procesos de preparación y entrenamiento con mejores resultados. El desarrollo de todos estos componentes y su posterior articulación con la metodología y la plataforma de estudio permitirán que los estudiantes se encuentren en un entorno de aprendizaje inmersivo, donde cada detalle habrá sido desarrollado buscando estimular su aprendizaje en la mayor medida posible.

Actualmente, el grupo está conformado por estudiantes de todos los semestres quienes buscan reforzar sus conocimientos con las temáticas típicas de una maratón de programación que se ajusten a su nivel actual de conocimiento. Entre las temáticas que se manejan actualmente, se pueden contar: algoritmos de grafos, Backtracking, procesamiento de Strings, geometría computacional, programación dinámica, teoría grafos y teoría de números, entre otros.

Cabe destacar que todas las temáticas que los estudiantes aprenden como parte del syllabus del grupo de estudio, si bien están enmarcadas en el contexto de una competencia de programación, son habilidades que resultan de enorme utilidad tanto en el desarrollo de sus materias en la línea de desarrollo de software, como en su vida profesional futura. Así pues, el desarrollo del marco previsto no solo ayudará al estudiante a mejorar sus resultados en competencia, sino también a mejorar sus resultados académicos, y a fortalecer habilidades propias de un ingeniero desarrollador de software.

Además, establecer un marco de trabajo y en especial la metodología de trabajo correctamente documentada y estructurada permite que sea perdurable en el tiempo, el cual se ha convertido en uno de los pilares de este proyecto: Cada una de las actividades y de las acciones realizadas sigue un lineamiento definido, que puede seguir realizándose en semestres sucesivos aun cuando los estudiantes y/o docentes encargados de la organización del grupo de estudio cambien. Esto es muy importante, pues los estudiantes tienen un paso limitado por la Universidad (según el pensum, de 10 semestres) lo que hace que los miembros del grupo están cambiando inevitablemente, semestre a semestre. Al sintetizar todo el proceso realizado en este marco de trabajo, se busca que, sin importar estos cambios en sus integrantes, el grupo siga mejorando y fortaleciéndose semestre a semestre.

Como bien se analizará en el marco teórico, diversas universidades en Colombia y en el exterior han definido sus propias estrategias, cursos, y actividades curriculares y extracurriculares para llevar a cabo sus procesos de preparación en competencias de programación, mejorando sustancialmente sus resultados con el tiempo, y siendo reconocidas por sus amplios logros. Lo que buscaremos será justamente definir un marco con los lineamientos para el proceso propio de preparación para la Universidad Francisco de Paula Santander, que permita acceder a esta élite en las competencias de Programación, teniendo en cuenta nuestro contexto actual y la preparación previa de los estudiantes. Como elemento distintivo, este marco de trabajo busca la integralidad, combinando múltiples enfoques que girarán en torno a la plataforma digital en la cual los estudiantes podrán acceder a un entorno completo de preparación, mientras los estudiantes y coach llevan a cabo un completo proceso de seguimiento y tutoría.

4. Objetivos

4.1. Objetivo General

Desarrollar e implementar un marco de trabajo para formalizar la metodología de trabajo del grupo de estudio de programación competitiva de la Universidad Francisco de Paula Santander.

4.2. Objetivos Específicos

- Fomentar la participación de los estudiantes en las competencias de programación para continuar representando a la universidad a nivel nacional e internacional.
- Construir un banco documental que abarque todos los contenidos que se trabajan durante el semestre y que sirva para evidenciar las actividades realizadas.
- Desarrollar una plataforma web de entrenamiento para realizar un seguimiento del trabajo realizado dentro del grupo de estudio.
- Diseñar un plan de trabajo para integrar todos los componentes desarrollados de manera tal que sea replicable en el futuro.

5. Delimitación

5.1. Alcances

El presente proyecto se realiza para generar y desarrollar un marco de trabajo para el grupo de estudio en programación competitiva del programa ingeniería de Sistemas en la Universidad Francisco de Paula Santander. Esto incluye: Plan de trabajo, estructura metodológica, material de apoyo en el proceso pedagógico y la plataforma de entrenamiento que será el centro de todos los demás componentes. Una vez culminado el desarrollo del proyecto, el grupo de estudio podrá seguir integrando nuevas herramientas, técnicas o metodologías adaptadas al marco de trabajo, que no estarán cubiertas por el proyecto actual.

Las herramientas y el material generado estarán disponibles para su utilización por parte de todos los estudiantes de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander, especialmente aquellos estudiantes del grupo de estudio en programación competitiva.

5.2. Limitaciones y delimitaciones

Teniendo en cuenta que el grupo de estudio sobre el cual se realiza este trabajo es relativamente nuevo (menos de un año en funcionamiento en el momento de iniciarse este proyecto) y hasta la fecha de inicio no se había establecido ninguna estructura metodológica para su funcionamiento, la investigación se ha delimitado en los siguientes aspectos:

5.2.1. Delimitación temporal. En el primer semestre del año 2015, se estableció el grupo de estudio en programación competitiva, y desde aquel momento se han llevado a cabo diferentes actividades con los estudiantes que han permitido estudiar a fondo las mejores maneras para lograr el cumplimiento de los objetivos propuestos, utilizando la retroalimentación obtenida para el desarrollo posterior del marco de trabajo.

El desarrollo formal e implementación del marco de trabajo (con un trabajo planificado de 6 meses antes de su primera versión funcional), realizado durante el segundo semestre del año 2016 entró en funcionamiento para el trabajo en el grupo de estudio en el primer semestre del año 2017, y de esta forma fue utilizado en la preparación de los equipos que participaron en la Maratón Nacional de Programación 2017.

En base a la información recolectada de las diferentes fuentes de información previstas, y la retroalimentación recibida durante la aplicación de esta primera versión del marco de trabajo, se realizaron cambios y mejoras para completar el marco de trabajo tal como se describe en este documento, con el cual se realizará la preparación de los equipos que participarán en la Maratón Nacional de Programación en el año 2018.

Si bien la primera versión desarrollada y utilizada en el 2017 (con resultados positivos como se demostrará más adelante) y la versión completa mostrada en este documento son versiones utilizables y técnicamente completas, evitaremos utilizar el término “versión final”, pues conforme avance el tiempo es posible que existan nuevas herramientas de entrenamiento y nuevos retos que afrontar que conlleven a una actualización de lo aquí expuesto. Sin embargo, estas actualizaciones futuras no hacen parte de este proyecto, que está delimitado hasta lo entregado en este documento.

5.2.2. Delimitación espacial. Todo el proyecto se centra para su uso en el grupo de estudio de Programación Competitiva UFPS.

5.2.3. Delimitación metodológica. El proyecto incluye sólo aquellos aspectos indicados en los alcances. Cualquier elemento no incluido en los alcances puede ser parte de proyectos sucesivos derivados de este**.**

6. Marco Referencial

6.1. Antecedentes

Analizar los trabajos teóricos y prácticos que se han realizado en ramas afines a la nuestra es fundamental para detallar cómo en otras instituciones se llevaron a cabo los procesos, en qué contexto fueron realizados para analizar su impacto en nuestro proyecto, cuáles elementos o prácticas pueden replicarse por sus buenos resultados y qué problemas han surgido en sus desarrollos, para evitarlos.

En primer lugar, se ha realizado un análisis de los proyectos investigativos realizados en Colombia y otros países para tener como referencia en la realización de nuestro proyecto. Junto a esto, se analizaron también proyectos prácticos que se han desarrollado en diferentes universidades fomentando la participación en competencias de programación, y finalmente analizamos los antecedentes propios del grupo de estudio, los cuales son de gran importancia porque aportan el contexto para comparar la situación de dicho grupo con los demás antecedentes externos.

6.1.1. Antecedentes investigativos.

Con estos objetivos en mente, se analizan 3 trabajos investigativos Nacionales y 3 internacionales realizados en el contexto de la Programación Competitiva y que aportan un punto de vista importante para el proyecto en curso actual. Los trabajos nacionales al respecto, se centran en mejorar el desempeño de sus universidades en las competencias ACM ICPC bajo diferentes enfoques:

- *Desarrollo e implementación de un programa de trabajo para el semillero de programación* (Echavarría, 2013). Mediante este proyecto se desarrolla un cronograma y

contenido de clases para el semillero de Programación de la Universidad EAFIT, buscando preparar a estudiantes novatos para las maratones de programación organizadas por ACIS/REDIS y por ACM-ICPC, fomentando la apropiación progresiva del conocimiento necesario para poder resolver problemas de este tipo. Se busca establecer un programa fijo, que se mantenga aún si los directores del semillero cambian.

- *Herramienta de estudio para las maratones de programación promovidas en el programa de ingeniería de sistemas y computación de la universidad tecnológica de Pereira* (Jimenez Becerra, 2012). El proyecto anterior se centraba al 100% en el aspecto metodológico como eje central del proceso. Este proyecto se centra en el mismo problema, bajo un enfoque diferente: Desarrolla una herramienta de estudio para participar en el proceso de entrenamiento de maratones de programación, aun cuando las maratones originales han culminado. En este proyecto se analizan diferentes alternativas de software libre que pueden implementarse en la Universidad Tecnológica de Pereira como herramienta de preparación, y se elige la utilización del software PC^2 para este propósito. Con esto se busca una mayor preparación de los estudiantes que disponen de una plataforma para enviar sus soluciones en cualquier momento, y es posible hacer un seguimiento sobre los competidores y sus respectivas soluciones.
- *Las Maratones de Programación, un paso más al campo de la Investigación* (Guerrero Calvache, Diaz Riascos, Córdova Pérez, & Hernandez, 2017). Este artículo presentado en la Universidad de Nariño describe el planteamiento inicial de un plan para el mejoramiento de los equipos que representan a su universidad en maratones de programación.

Por su parte a nivel internacional, un referente invaluable es URI Online Judge, el juez de problemas de programación online abierto de la Universidad Regional Integrada do Alto Uruguai e das Missoes (Brasil). En torno a este se encuentra gran material documental sobre la forma en que herramientas de este tipo pueden jugar un papel fundamental en la enseñanza de programación. A continuación, se exponen 3 artículos que plantean la importancia y trascendencia de esta plataforma, la cual resulta un antecedente de gran importancia para nuestra investigación:

- *Uri Online Judge Académico: Una herramienta para clases de algoritmos y programación (Nombre original en inglés: URI Online Judge Academic: A Tool for Algorithms and Programming Classes)* (Bez, Tonin, & Rodegheri, URI Online Judge Academic: A tool for algorithms and programming classes, 2014). Esta herramienta cuenta con más de 1000 ejercicios de diferente temática y diferente dificultad, y se encuentra abierta al público general para que cualquier estudiante pueda prepararse para una competencia de programación. En este estudio sus creadores exponen su enfoque educativo: Como utilizar esta plataforma (y, por consiguiente, cómo utilizar las maratones de programación) como herramienta pedagógica para la enseñanza en clases de algoritmos y programación.
- *URI Online Judge Academic: Integración y consolidación de la herramienta en el proceso de enseñanza/aprendizaje (Nombre Original en portugués: "URI Online Judge Academic: Integração e Consolidação da Ferramenta no Processo de Ensino/Aprendizagem")* (Bez, Tonin, & Selivon, URI Online Judge Academic: Integração e Consolidação da Ferramenta no Processo de Ensino/Aprendizagem, 2015).

En este trabajo los autores de URI Online Judge exponen como el uso de la herramienta de maratones (la misma expuesta en la investigación anterior) funciona efectivamente fortaleciendo el proceso de enseñanza y aprendizaje, no sólo como parte de una preparación para competencias de programación, sino como herramienta de integración primordial en las clases presenciales de algoritmos y áreas relacionadas.

- *Uri Online Judge: Una herramienta para profesores (Nombre Original en inglés: URI Online Judge Academic: A Tool for Professors)* (Bez, Ferreira, & Tonin, URI Online Judge Academic: A Tool for Professors, 2013). En este artículo se describe como la plataforma de maratones de programación URI Online Judge a través de su módulo "Academic" puede modernizar y aportar nuevas formas de enseñanza en el aula de clase, orientando, simplificando, controlando y evaluando el proceso.

6.1.2. Antecedentes prácticos. Diferentes Universidades en los 5 continentes han definido sus propios lineamientos para llevar a cabo su preparación de cara a los distintos certámenes de programación competitiva y más específicamente a las competencias de ACM ICPC. Si bien cada centro de estudios tiene sus propios métodos y herramientas, cabe destacar que muchas de las actividades realizadas tienen aspectos en común, que pueden ser un punto de partida importante para el trabajo a implementar en la UFPS, con las características necesarias para adaptarse a nuestro propio entorno. Algunos de los antecedentes a destacar en este ámbito a nivel mundial son:

- SLPC ("Stanford Local Programming Contest") (Wang, 2017) y CS 97SI: Introduction to Programming Contests, Universidad de Stanford (Park, 2016): SLPC es la maratón de Programación Interna de la Universidad de Stanford, y como preparación para sus

estudiantes oferta el curso CS 97SI, mediante el cual sus estudiantes aprenden sobre las generalidades de una maratón de programación y los temas más comunes en competencia. Los ejercicios de este curso son todos de competencias de programación, y el material generado (temarios, diapositivas, problemas propuestos y un libro corto de referencia con una colección de algoritmos codificados en C++ para su uso durante competencia. El curso es impartido por el PhD. Jaehyun Park, coach de los equipos de la Universidad en competencias ICPC.

- "CS3233 - Competitive Programming" Universidad Nacional de Singapur (Halim, CS3233 - Competitive Programming, 2017). Este curso prepara a los estudiantes de dicha universidad que desean participar en competencias ICPC, y a los estudiantes que desean mejorar sus habilidades en entrevistas técnicas de TI. El curso es impartido por Steven Halim, quien además es coach de los equipos de su Universidad y es una de las figuras más importantes de Programación Competitiva en el mundo. Este curso se acompaña del libro Competitive Programming 3 (Halim & Halim, Competitive Programming, 2013) del mismo autor, el cual define desde las estrategias para afrontar una competencia de programación, hasta las temáticas generales de estudio. El libro se complementa con una herramienta llamada "UVA Hunting" que realiza seguimiento a cada uno de los temas especificados, con ejercicios del juez virtual UVA de la universidad de Valladolid.

También en Colombia se han realizado avances en este sentido:

- El grupo de Investigación en Informática Educativa "GUÍAME" De la Universidad Nacional de Colombia sede Medellín ha desarrollado la herramienta "PPC - Programming Practice Center" (PPC - Competitive Programming Practice Center, 2014)

con el fin de apoyar a profesores y monitores en los cursos de programación. Más específicamente, dicha plataforma permite definir una serie de problemas para que los estudiantes resuelvan a lo largo del semestre. Esta plataforma se basa en el API del juez online Uva, el cual contiene la base de datos de ejercicios de maratones de programación más amplia existente.

6.1.3. Antecedentes del grupo de estudio en programación competitiva. La Universidad Francisco de Paula Santander ha contado con participación en diferentes competencias de programación, siendo la principal ACM - ICPC. En el año 2012 un equipo representó a la Universidad en la fase Nacional de ACM ICPC, en el año 2014 la representación fue de 2 equipos, y desde el año 2015 ininterrumpidamente se han presentado 3 equipos en la competencia nacional ICPC anual.

En estas representaciones, se ha logrado clasificar un equipo a la fase regional latinoamericana en los años 2014, 2015 y 2016, y 2 equipos en el año 2017 (cabe destacar que luego del trabajo realizado durante un año en la construcción de este proyecto, se logró por primera vez la clasificación de 2 equipos a fase regional, mostrando los resultados del trabajo. El resumen pormenorizado se efectuará más adelante).

6.2. Marco Teórico

La construcción de un marco de trabajo que fortalezca el proceso de preparación y entrenamiento de los estudiantes del programa de Ingeniería de Sistemas de la UFPS para las competencias de programación tipo ACM-ICPC conlleva una preparación teórica en tres ramas del conocimiento diferentes que en el proceso deben entrelazarse: En primer lugar, es necesario el estudio y análisis de diferentes metodologías de aprendizaje que se ajusten a las necesidades del grupo de estudio y que se puedan aplicar en el desarrollo de este proyecto. En segundo lugar, debe realizarse un completo y centrado análisis de las temáticas que se abordarán en el grupo de estudio. Sea cual sea la metodología y herramientas empleadas, este marco de trabajo surge como una herramienta para la enseñanza y el mejoramiento de las habilidades de programación y, por tanto, es necesario definir cuáles serán esas enseñanzas que conformen el syllabus a fortalecer en el grupo. Por último, pero no menos importante, debe estudiarse el componente tecnológico que acompañará a la metodología propuesta y la forma en que este pueda potenciar el aprendizaje. Así pues, los 3 componentes forman un solo conjunto, donde el primero (la metodología) es el carril por el cual el segundo (el conocimiento) puede fluir, y el tercero (el componente tecnológico) funciona como aglutinante de los dos primeros.

Antes de poder seleccionar una metodología de aprendizaje es necesario comprender cuales son los factores que motivan el aprendizaje independiente en estudiantes universitarios. Dado que el grupo de estudio no es una materia común y es completamente opcional, es fundamental mantener la motivación de los estudiantes para que sigan asistiendo y haciendo parte de las actividades planeadas. Para elevar esta motivación deben establecerse factores inmediatos desde que el estudiante inicia su preparación. Según Ngan y Law (Ngan & Law, 2015), estos factores

pueden ser establecer metas desafiantes, tener un sistema de aprendizaje virtual sólido y promover la sana competencia.

Tomando en cuenta estos factores motivacionales, la cantidad de estudiantes con los que cuenta el grupo de estudio y sus diferentes niveles de conocimiento, es necesario plantear estrategias metodológicas para el grupo de estudio en programación competitiva que se adapten a estas características. Es por esto que la idea resultante es aplicar el aprendizaje colaborativo complementado con el aprendizaje basado en problemas. A continuación, se exponen las ideas subyacentes tras cada uno de estos enfoques, y su importancia para el proyecto actual:

El **aprendizaje colaborativo** es un enfoque que trata de organizar las actividades dentro del aula para convertirlas en una experiencia social y académica de aprendizaje. Los estudiantes trabajan en grupo para realizar las tareas de manera colectiva. El aprendizaje en este enfoque depende del intercambio de información entre los estudiantes, los cuales están motivados tanto para lograr su propio aprendizaje como para acrecentar los logros de los demás. Como características de este tipo de aprendizaje, encontramos (Universidad EAFIT, 2014):

La **interdependencia positiva**, donde cada estudiante trabaja en pro de unas metas en común, que son establecidas entre todos, realizando unas tareas que también son repartidas por consenso. De esta forma todos los estudiantes que hacen parte del grupo se sienten parte del proceso y las responsabilidades no recaen en un solo estudiante, sino en el grupo entero. Teniendo en cuenta que uno de los objetivos del grupo es funcionar sin necesidad de un rol de "profesor/organizador" todo el tiempo, y que se busca que aun cuando los estudiantes del grupo cambien semestre a semestre este pueda seguir funcionando sin problema, la interdependencia positiva es un pilar clave en la construcción del marco de trabajo.

Del mismo modo, se encuentra **la interacción cara a cara**, gracias a la cual los estudiantes se habitúan al trabajo en equipo desde el marco del respeto y apoyo mutuo, puntos fundamentales tanto en las competencias ACM ICPC (Donde se trabaja en equipos de 3 integrantes con un solo computador, lo que obliga a una perfecta coordinación como equipo) como en la vida profesional.

Otra característica de este enfoque de aprendizaje es la **contribución individual**, aunque va de la mano con la interdependencia positiva. Esta es la capacidad del estudiante de "asumir un papel participativo en el proceso, a través de actividades que le permitan exponer e intercambiar ideas, aportar opiniones y/o experiencias, convirtiendo así la tarea del equipo en un foro abierto a la reflexión y al contraste crítico de pareceres y opiniones" (Universidad EAFIT, 2014).

Por último, este enfoque mejora diferentes **habilidades personales y de grupo**, pues mientras los estudiantes fortalecen sus habilidades argumentativas, colaborativas y asociativas, grupalmente se fortalece la capacidad de planificar, auto organizar, autorregular y tomar decisiones en equipo.

De esta forma, con el aprendizaje colaborativo se solucionan muchos de los problemas iniciales de la conformación del grupo de estudio. Ahora bien, para lograr un verdadero impacto en el aprendizaje de los estudiantes que se preparan para competencias de programación, es necesario introducirlos en el ambiente propio de estas competencias. Es allí donde surge el **aprendizaje basado en problemas** (APB) (Vizcarro, y otros, 2009). Este enfoque por su parte es un método de aprendizaje basado en el principio de usar problemas como punto de partida para para la adquisición e integración de nuevos conocimientos. Esta metodología busca que los estudiantes sean el centro del aprendizaje, desarrollando competencias como trabajo en equipo,

habilidades de comunicación y técnicas de análisis y resolución de problemas (Servicio de Innovación Educativa de la Universidad Politécnica de Madrid , 2008).

Uno de los principales puntos a favor del aprendizaje basado en problemas es que fomenta la posibilidad de interrelacionar diferentes asignaturas académicas. Esto es fundamental en las competencias de programación, pues en dichas competencias se requieren conocimientos en programación (estructuras de datos, diseño de algoritmos, etc.), diferentes ramas de la matemática (Geometría, estadística, métodos numéricos, etc.) entre otras asignaturas que aparecen con mayor o menor regularidad. En un solo problema pueden mezclarse diferentes temáticas que incluso, no necesariamente todos los estudiantes deben conocer, pero uniendo los conocimientos individuales y fomentando el trabajo en equipo, pueden alcanzar los objetivos.

Lo anterior sostiene la base teórica que responde al "cómo" llevar a cabo el proceso. Pero resuelto el "cómo", se pasa al "qué", y es necesario centrar los puntos que nos llevarán a construir el conjunto de temáticas que serán cubiertas en el grupo de estudio bajo el enfoque previamente descrito. Sin embargo, la construcción de este syllabus no es tan compleja, pues de hecho existe notable material de referencia. En este ámbito, Steven Halim y Felix Halim (Halim & Halim, Competitive Programming, 2013) han definido en su libro *Competitive Programming* una serie de temáticas de usual aparición en competencias. La misma tarea ha desarrollado Antti Laaksonen en *Competitive Programmer's Handbook* (Laaksonen, 2017), y Steven Skiena junto a Miguel Revilla en *Programming Challenges: The Programming Contest Training Manual* (Skiena & Revilla, 2003). Si bien los 3 presentan algunas diferencias en sus temáticas, muchas de ellas resultan similares, y partiendo de ellas puede formarse un consenso sobre los puntos principales a cubrirse en el grupo.

Dado que estos 3 libros son netamente centrados en la programación competitiva, la base brindada por *Introduction to Algorithms* del MIT (Cormen, Leiserson, Rivest, & Clifford, 2012) nos aporta apoyo académico a las mismas temáticas. En cierta forma, los temas cubiertos por este texto son similares a los cubiertos por los mencionados previamente; pero la diferencia entre el enfoque competitivo y el enfoque netamente académico crea un contraste en buenas prácticas de código que intentamos no descuidar en nuestro grupo, y por eso mantenemos ambos enfoques a la vista en la construcción del syllabus.

Teniendo en cuenta que el grupo acoge estudiantes de todos los semestres, también deben cubrirse las temáticas más básicas, que no se encuentran en libros tan especializados. Para este segmento tomamos como punto de partida *Fundamentos de Programación* (Joyanes Aguilar, 1996).

Finalmente, el marco de trabajo necesita un componente tecnológico que aúne todo lo realizado: Donde pueda encontrarse el material de aprendizaje, aprender nuevos temas, enfrentarse a los retos que una competencia de programación propone a los estudiantes, y que se encuentre disponible para su uso durante las sesiones del grupo, pero también de forma externa. Por tanto, una herramienta online que concentre estos materiales es lo ideal. Como ejemplo, puede citarse *URI Online Judge* (Bez, Ferreira, & Tonin, URI Online Judge Academic: A Tool for Professors, 2013) como un espacio donde los docentes establecen control del trabajo independiente de sus estudiantes, *Hackerearth* que acompaña problemas a desarrollar con material de aprendizaje sobre dichos temas y *Uva Online Judge* que contiene la más grande colección de problemas de programación en internet. Estas tres herramientas han sentado unas bases en la programación competitiva, y algunos de sus conceptos trascienden en la enseñanza.

Finalmente, cabe decir que la programación va más allá de las competencias. La carrera de Ingeniería de Sistemas en la UFPS tiene como misión el desarrollo y gestión de sistemas de información, y una de las fases en el desarrollo es la programación. No en vano, una línea dentro del pensum se dedica a la Programación de Computadores, cubriendo diferentes materias electivas y opcionales, como Fundamentos de Programación, Programación Orientada a Objetos I y II, Análisis de Algoritmos, Estructuras de datos, entre otras. Todas estas materias tienen un alto componente práctico, en donde el estudiante debe enfrentarse a entornos de programación, sea cual sea el lenguaje o el tema visto. En estos contextos, el grupo de estudio y el pensum están íntimamente relacionados. Los estudiantes profundizan y cuestionan en el grupo lo visto en sus materias. Es una forma de contrastar sus aprendizajes, y fortalecerlos. Por dicha causa, el proyecto planteado, tal como hemos visto con *URI Online Judge* puede ir más allá del grupo de estudio, y ser parte activa de las clases de programación.

6.3. Marco Legal

La Constitución Política de Colombia en su artículo 69 establece que "*Se garantiza la autonomía universitaria. Las universidades podrán darse sus directivas y regirse por sus propios estatutos, de acuerdo con la ley.*" (Asamblea Nacional Constituyente, 1991) Haciendo uso de esta facultad dada por el estado, la Universidad Francisco de Paula Santander en el acuerdo 056 del 07 de septiembre del 2012 (Consejo Superior Universitario UFPS, 2012), establece la organización del sistema de investigación de la Institución. En el artículo 1 de dicho acuerdo establece regirse, entre otras, por las siguientes políticas: Promoción de una cultura de generación, transferencia y aplicación de conocimiento, fortalecimiento de grupos, semilleros y centros de investigación y fomento a la formación de investigadores mediante la participación activa de los estudiantes en semilleros de investigación.

Bajo esta política institucional, la Universidad tiene en SILUX uno de los semilleros de investigación establecidos con participación constante de estudiantes. Una de las líneas investigativas para este semillero, definidas por su director es Programación Competitiva, que tiene como sublíneas resolución de problemas, programación, lenguajes, paradigmas y herramientas. Es justo para cubrir esta línea que se ha creado el grupo de estudio en Programación competitiva como parte del semillero.

El actual marco de trabajo para este grupo de estudio permite reforzar la forma en que el semillero SILUX se rige y apoya estas políticas instituciones, y la comunicación con quienes han ejercido como directores del semillero ha sido constante y retroactiva.

7. Diseño Metodológico

7.1. Tipo de Investigación

Según Raúl Dean, la investigación tecnológica "*designa un ámbito de producción de conocimiento tecnológico validado, que incluye tanto el producto cognitivo, -teorías, técnicas, tecnologías, maquinarias, patentes, etc.- como las actividades que desarrollan los ingenieros para producir y validar dichos productos y conocimientos.*" (Dean, 2017) Desde este punto de vista, este proyecto se encuentra enmarcado en el ámbito de la investigación tecnológica aplicada ya que se busca desarrollar e implementar un marco de trabajo para el entrenamiento de los estudiantes del grupo de estudio en programación competitiva de la Universidad Francisco de Paula Santander, integrando y desarrollando herramientas tecnológicas para lograr este propósito.

7.2. Fuentes de información

7.2.1. Fuentes de información primaria. Las fuentes primarias contienen información de gran valor, fruto de estudios originales que permiten orientar la investigación. Consideramos como fuentes de información primaria para nuestra investigación:

- Retroalimentación de los estudiantes del grupo: Durante el desarrollo del proyecto se obtuvo múltiple retroalimentación a través de encuestas, mesas redondas y retroalimentación directa con los estudiantes.

- Resultados en competencias: Mientras el proyecto se llevaba a cabo, los estudiantes del grupo participaron en múltiples competencias de Programación a nivel interno, nacional y continental, de donde pudieron obtenerse sus resultados y compararlos con respecto a la Universidad, la región, el país y el continente.
- Artículos científicos, revistas científicas e informes de ponencias: Para asegurar que el proyecto funcionara correctamente se recopiló información previa de diferentes Universidades que sobresalen en competencias de Programación, sentando una base sólida de antecedentes a tener en cuenta.
- Guías metodológicas y pedagógicas: Para el diseño de la metodología de trabajo colaborativa basada en problemas, se analizaron guías metodológicas de estos dos enfoques que aportaran una visión clara de sus ventajas, desventajas y características.

7.2.2. Fuentes de información secundaria. Junto a las fuentes primarias que aportan el valor principal a esta investigación, se encuentran las fuentes de información secundarias que permiten añadir valor agregado en aspectos específicos a la investigación. En este caso, las fuentes de información secundarias son**:**

- Documentación de lenguajes utilizados en la plataforma de software.
- Textos de referencia en metodologías de enseñanza.
- Extensa colección de problemas de programación competitiva en diferentes plataformas.
- Documentación de plataformas de entrenamiento en maratones de programación.
- Normas técnicas.

7.3. Recolección de la información

El proceso de recolección de información es crucial para alcanzar los objetivos establecidos en el proyecto.

Es por esto que se hace necesaria en primer lugar una búsqueda y revisión de la literatura que sirva de base para el proyecto, así como la revisión detallada de los antecedentes y avances que se están realizando a nivel nacional y mundial en los procesos de preparación y entrenamiento de estudiantes para las competencias de programación tipo ACM-ICPC.

Durante el desarrollo del proceso, se mide regularmente los avances de los estudiantes y su satisfacción con respecto al grupo. Con lo primero, aseguramos que los estudiantes están aprendiendo y mejorando en competencias de programación, objetivo primordial del grupo y del proyecto, y con lo segundo, buscamos asegurar la permanencia de los estudiantes para mantener un grupo constante de personas semestre a semestre. Por esto, para medir los avances de los estudiantes recolectamos información cuantitativa de múltiples fuentes para evaluar su mejoramiento:

- Número de problemas resueltos por temática (de estos datos puede hacerse un fácil seguimiento sobre cuales temas son los que presentan mayores dificultades en el aprendizaje).
- Seguimiento al trabajo semanal por cantidad de ejercicios resueltos.
- Resultados obtenidos en competencias de la Liga Colombia de Programación Competitiva (CCPL) realizadas aproximadamente una vez al mes, en donde además de la UFPS participan universidades de toda Colombia (permitiendo realizar una perspectiva con respecto al país).

- Resultados obtenidos en competencias de la Red Programación Competitiva (RPC) realizadas aproximadamente una vez al mes, en donde además de la UFPS participan universidades de toda Latinoamérica (permitiendo realizar una perspectiva con respecto a la región).

Con respecto al segundo ítem se realizan encuestas de satisfacción, y análisis sobre el número de asistentes semanales.

7.4. Análisis de la información

El proceso de análisis de información consiste en revisar los resultados obtenidos a través de la recolección de información de cada una de las variables establecidas para determinar la veracidad o falsedad de las hipótesis a probar establecidas en el proyecto.

7.5. Población

Para la presente investigación la población corresponde a los estudiantes del Programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander que pertenecen al grupo de estudio en programación competitiva (Dependiendo el semestre académico, esta cifra se mantiene entre los 30 y 60 estudiantes).

7.6. Muestra

Dado que la población tiene un tamaño manejable, y que los mecanismos de recolección de información son lo suficientemente versátiles, es posible analizar toda la población sin perder

simplicidad. Por tanto, la muestra son los estudiantes del Programa de Ingeniería de Sistemas de la Universidad Francisco de Paula Santander que pertenecen al grupo de estudio en programación competitiva.

7.7. Instrumentos

Previamente se han mencionado algunos instrumentos para la recolección de información. A continuación, se detallan formalmente:

- Consolidados de los resultados en las competencias de entrenamiento desarrolladas por la Red de Programación Competitiva y la Liga Colombiana de Programación Competitiva en las que participan los estudiantes del grupo de estudio.
- Consolidados de los resultados en las competencias oficiales en las que participan los estudiantes del grupo de estudio en representación de la Universidad.
- Consolidado de los temas cubiertos en las sesiones del grupo de estudio.
- Métricas de usabilidad y pruebas de rendimiento aplicadas a la plataforma de entrenamiento que se desarrollara en el proyecto.
- Encuesta para conocer el nivel de satisfacción de los estudiantes que pertenecen al grupo de estudio respecto a la metodología empleada.

8. Metodología del Proyecto

El proyecto se realiza siguiendo una metodología iterativa compuesta de 6 fases (Donde la primera fase no hace parte de las iteraciones, como se ilustra en el siguiente gráfico):

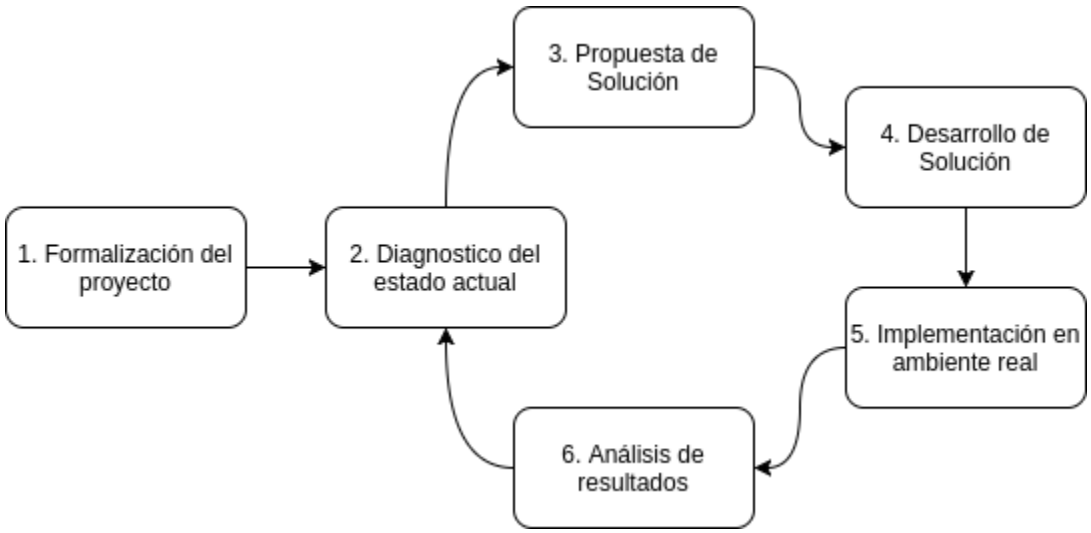


Figura 1. Diagrama de la metodología de desarrollo del marco de trabajo

Inicialmente se ha realizado la formalización el proyecto: Análisis del problema, detalles preliminares, y formación del estado del arte y referente teórico. A continuación, se diagnostica como es el estado inicial del grupo, sin un marco de trabajo definido y se propone una solución al respecto. Después de refinar esta propuesta de solución se lleva a cabo, y se implementa directamente en el grupo de estudio. Una vez implementada se recaba información en un tiempo prudencial y se analizan los resultados para volver a un diagnóstico del estado actual, ya con un marco de trabajo funcional, con el fin de detectar problemas y mejoras, volviendo a poner en marcha las fases 2,3, 4,5 y 6, sin reinventar el trabajo, sino mejorando en cada paso aquellos aspectos que pueden ser mejorados. Al momento de entregar este proyecto se han realizado dos iteraciones completas que permiten aportar los resultados satisfactorios esperados. Si en un

futuro nuevas circunstancias precisan de cambios en el marco de trabajo, puede seguirse el mismo proceso, realizando una nueva iteración.

Fase 1: Formalización del proyecto: Esta fase cubre todo el planteamiento inicial, análisis del problema, descripción del problema, desarrollo y entrega del anteproyecto, formalización del proyecto, análisis de antecedentes y construcción de marco teórico.

Fase 2: Diagnóstico del estado actual: Comprende el análisis de las necesidades del grupo para analizar qué se necesita implementar. En la primera iteración, el diagnóstico es completo, para implementar desde cero. En la segunda en cambio, es un diagnóstico sobre elementos que podrían mejorarse.

Fase 3: Propuesta de solución: En esta fase se parte del diagnóstico del estado actual para proponer un marco de trabajo para el grupo de estudio, con sus componentes y especificaciones. En las iteraciones sucesivas, en este paso se proponen cambios y mejoras sobre la solución ya implementada.

Fase 4: Desarrollo de la solución: En esta fase se realizan/mejoran los componentes propuestos, según los resultados de la fase anterior. En esta fase se completa una nueva versión del marco de trabajo.

Fase 5: Implementación en ambiente real: En este punto, el marco de trabajo tiene una nueva versión que es puesta en práctica directamente con los estudiantes del grupo de estudio. Se utilizan los componentes creados/mejorados en un ambiente real, y se extraen datos para analizar los beneficios/fallas que puedan presentarse.

Fase 6: Análisis de resultados: Todas las iteraciones conllevan un análisis de resultados, en el cual se miden diversos factores para analizar si el Marco ha funcionado, y qué factores podrían mejorarse. Mientras el proyecto se encuentra en desarrollo, esta fase siempre aporta la materia

prima para iniciar una nueva iteración, pues de los resultados obtenidos se definen qué factores deben mejorarse, y de allí parte el nuevo ciclo.

Cabe mencionar que en el marco de trabajo se incluye un componente software, que, como tal, se realiza siguiendo una metodología específica de desarrollo de software que será detallada más adelante. En este punto se menciona únicamente la metodología del proyecto completo como tal.

9. Diagnóstico situacional

Antes de iniciar el desarrollo del proyecto, el programa de Ingeniería de Sistemas de la UFPS había tenido representación en las siguientes competencias oficiales de programación:

Tabla 1. Desempeño UFPS en maratones de programación antes del inicio de este proyecto. Fuente: ACIS

Año	Competencia	Equipo	Puesto*	Comentario
2011	Maratón Nacional de Programación ACIS/REDIS - Colombia	UFPS-Cúcuta	> 36	-
2014	Maratón Nacional de Programación ACIS/REDIS - Colombia	Candelaria UFPS	> 33	Recibe invitación a fase regional latinoamericana por desempeño. **
		Yajar	> 33	-
	Maratón Regional de Programación ICPC - Latinoamérica Norte	Candelaria UFPS	18	-
2015	Maratón Nacional de Programación ACIS/REDIS	Candelaria	29	Clasificado a fase regional latinoamericana.
		UFPS Team	> 40	-
		Null Pointer	> 40	-
	Maratón Regional de Programación ICPC - Latinoamérica Norte	Candelaria	18	-

- **(*)** Los puestos que contienen un >, implican que el equipo no logró resolver ningún ejercicio durante la competencia. De esta forma, si un equipo posee puesto > 33,

significa que, en dicha competencia, 33 equipos obtuvieron al menos un ejercicio, y los demás se dan como empatados en la posición 34, o > 33 (Se utiliza la segunda notación, para no confundir con puestos fijos).

- **(**)** De la competencia Nacional clasifica un número determinado de equipos a la fase latinoamericana. Si el número de equipos que resuelven al menos un ejercicio en competencia no alcanza a cubrir la cantidad de cupos asignados, la organización suele invitar a algunos de los equipos que no puntuaron, según su desempeño durante la competencia, en el entrenamiento previo, y sus resultados durante el año en la Liga Colombiana de Programación (CCPL)

Más adelante estos resultados serán contrastados con los obtenidos durante los años 2016 y 2017, años en los cuales el proyecto se encontraba en ejecución.

El grupo de estudio en Programación Competitiva se conformó en el año 2015 con el fin de mejorar los resultados en competencias de programación de los estudiantes de la Universidad Francisco de Paula Santander.

Tabla 2. Número de estudiantes en el grupo de estudio

AÑO	SEMESTRE	ESTUDIANTES TOTALES	ESTUDIANTES ACTIVOS
2015	I	19	8
	II	30	7
2016	I	36	12

Se consideran estudiantes activos aquellos que hicieron parte de por lo menos la mitad de las actividades realizadas durante el semestre en el grupo.

Desde el primer momento se diagnosticó un problema en la retención de los estudiantes en el grupo. Durante los 3 semestres en los cuales el grupo de estudio funcionó antes del desarrollo de este marco de trabajo, se visualizó como la cantidad de estudiantes que asistieron en al menos una ocasión, creció notablemente (casi se duplicó desde el primer semestre del 2015 hasta el segundo semestre del 2016). Sin embargo, el número de estudiantes activos fue bajo, añadiendo además que la cantidad más alta de estudiantes durante cada semestre se encontraba en el inicio de las sesiones. A medida que el tiempo pasaba, el número de estudiantes decaía, hasta finalizar semestres únicamente con aquellos que habían sido más activos.

Para lograr una mayor permanencia de los estudiantes, se realizó una pequeña encuesta abierta al iniciar el segundo semestre del 2016 con aquellos que habían pertenecido al grupo en los semestres anteriores. En esta encuesta buscábamos conocer las razones que motivaban a los estudiantes a hacer parte del grupo, y las dificultades que habían tenido en el proceso. En la encuesta presentamos dos preguntas con diferentes opciones (selección múltiple con múltiple respuesta) y un espacio para añadir aquellas que no hubiéramos considerado. Con esta encuesta se buscó en primer lugar, determinar qué motiva a los estudiantes para hacer parte del grupo, y de esa forma, enfocar el proyecto en la consecución de esas motivaciones y, en segundo lugar, determinar las dificultades que pudieran llevar a la deserción, para mitigarlas en la medida de lo posible.

Tabla 3. Motivación de los estudiantes para participar en el grupo de estudio

Motivación	**No. Estudiantes**
Representar a la carrera en diferentes competencias de programación de forma oficial.	16

Reforzar los temas vistos en las materias de la carrera	13
Aprender nuevas temáticas	11
Obtener proyección profesional y oportunidades laborales	4
Pertenecer a un semillero	1
Las competencias de programación son divertidas	1

Tabla 4. Dificultades que encontraron los estudiantes en el grupo de estudio

DIFICULTAD	**NO. ESTUDIANTES**
Muchas temáticas son de semestres superiores al que me encuentro.	9
Las competencias tienden a ser muy complicadas.	7
El grupo no es lo suficientemente organizado.	8
El horario de competencias no me resulta cómodo.	4
No conseguí equipo.	4

Esta encuesta fue realizada a una muestra de 21 estudiantes que hicieron parte del grupo en los 3 semestres previamente mencionados, y se llevó a cabo al iniciar el segundo semestre del 2016. Estos resultados presentan información relevante: En primer lugar, los estudiantes buscan participar en competencias oficiales representando a la Universidad a nivel nacional e internacional. Por tanto, este punto será muy importante en el momento de diseñar el marco de trabajo, pues se buscará que el desempeño durante las diferentes actividades realizadas sea el criterio principal bajo el cual se definan los estudiantes que realizan dichas representaciones. Del mismo modo, en las temáticas a tratar se buscará lograr un equilibrio entre temas comunes de la

carrera a profundizar, y temas que están fuera del pensum pero que pueden ser útiles en dichas competencias.

Del mismo modo, la dificultad principal se encuentra en que algunas de las temáticas tratadas en el grupo son demasiado complicadas para el semestre académico que cursan los estudiantes. Entonces, el marco debe tener especial cuidado en proponer actividades que sean aptas para estudiantes sin importar su semestre actual. La organización del grupo se encuentra en constante mejora, y con el marco debe estandarizarse, así que este es un tema inherente al proyecto. También debe fomentarse el trabajo en grupo (aspecto fundamental de las competencias). La dificultad de las competencias siempre será alta (estas competencias son en su mayoría externas a la Universidad y no podemos definir su dificultad), pero, por tanto, el objetivo es preparar a los estudiantes para que esto no sea un problema, sino una forma de mejorar. Finalmente, el horario de las competencias es lo único que desafortunadamente no podemos tratar, pues son definidos por la organización y no está a nuestra disposición modificarlo. Sin embargo, el grupo tiene más actividades aparte de las competencias, de las cuales los estudiantes pueden hacer parte sin la obligación de competir.

Al iniciar el segundo semestre de 2016, el grupo se encuentra conformado por 36 personas de la siguiente forma:

Tabla 5. Composición del grupo de estudio al iniciar el segundo semestre del 2016

Semestre	**No. Estudiantes**
I	2
II	5
III	10
IV	5

V	4
VI	5
VII	3
VIII	0
IX	0
X	2

Esta información será importante en el momento de definir las temáticas a tratar.

10. Propuesta de Solución

La propuesta de solución planteada se encuentra estructurado en 5 ejes, donde para cada uno de ellos se han establecido un conjunto de actividades y unos resultados esperados.

Tabla 6. Ejes del marco de trabajo

EJE	DESCRIPCIÓN
Eje 1: Promoción	El objetivo del primer eje del marco de trabajo es fomentar la participación de los estudiantes en las competencias de programación para continuar representando a la universidad a nivel nacional e internacional.
Eje 2: Metodología	La metodología de trabajo del grupo define la forma de trabajar y la forma de llevar a cabo todas las actividades. Este eje estructura todo el trabajo a realizar y define como se orientará el grupo a lo largo del tiempo.
Eje 3: Producción documental	El tercer eje del marco de trabajo tiene como objetivo construir un banco documental que abarque todos los contenidos que se trabajan durante el semestre y que sirva para evidenciar las actividades realizadas.
Eje 4: UFPS Training Center	Todas las actividades desarrolladas en el grupo de estudio tendrán un punto de encuentro a través de una plataforma web llamada UFPS Training Center (Plataforma tecnológica). En este eje del

	proyecto se busca desarrollar dicha plataforma para realizar un seguimiento del trabajo realizado dentro del grupo de estudio.
Eje 5: Integración del Marco de Trabajo	El quinto eje del proyecto tiene como objetivo diseñar un plan de trabajo para integrar todos los componentes desarrollados de manera tal que sea replicable en el futuro.

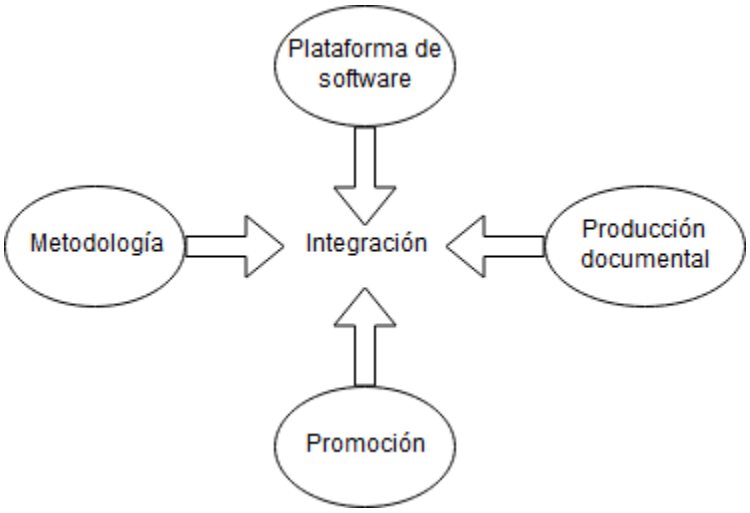


Figura 2. Integración del marco de trabajo

11. Promoción

El objetivo del primer eje de la investigación es fomentar la participación de los estudiantes en las competencias de programación para continuar representando a la universidad a nivel nacional e internacional. Los estudiantes de la carrera deben conocer la existencia del grupo de estudio, así como contar con información clara sobre su funcionamiento, los días en los cuales se realizan reuniones o competencias, y sus mecanismos de aprendizaje y clasificación a representaciones oficiales; para esto se han establecido las siguientes actividades, las cuales se llevan a cabo según la regularidad establecida:

11.1. Reuniones de información e invitación

Al iniciar cada semestre, se realiza una reunión de carácter informativo con el fin de invitar a los estudiantes de la carrera a hacer parte del grupo, explicar sus objetivos, su funcionamiento y definir los horarios de las sesiones a realizarse. Esta reunión se realiza según disponibilidad de espacio y tiempo, en los días siguientes a la reunión de invitación a semilleros que realiza el programa semestralmente. De esta forma, en la invitación a semilleros (momento en el cual se encuentran presentes gran parte de los estudiantes del programa) se indica el horario de esta reunión, y así es posible alcanzar un mayor número de estudiantes. Esta reunión se ha realizado de forma ininterrumpida desde el inicio de este proyecto con una asistencia considerable, y se ha realizado en conjunto entre los autores de este proyecto y el director del semillero SILUX, al cual pertenece el grupo. A lo largo del tiempo, esta reunión puede (y debería) seguir siendo organizada por quienes en cada momento de tiempo tomen el control del grupo, siempre de la mano del director del semillero.

11.2. Invitaciones al grupo por diferentes medios de difusión

A lo largo del semestre, el grupo realiza prácticas, charlas y competencias internas. También, en la medida de lo posible, se participa en competencias de RPC, CCPL y en ocasiones especiales otras competencias (Facebook Hacker Cup, Google Code Jam, entre otras). Para cada una de estas actividades, es necesario que los estudiantes estén informados sobre los horarios y requisitos para participar. Para esto se han establecido dos canales directos de comunicación: Corre electrónico y un grupo en Facebook (Programación Competitiva UFPS) que cuenta con 363 personas que pertenecen a la universidad o a las comunidades de programación competitiva en Colombia y Latinoamérica. El correo electrónico se utiliza para todas aquellas informaciones importantes que son relevantes únicamente para los estudiantes de la universidad mientras que en el grupo en Facebook se comparte toda clase de material e información que pueda ser útil no solamente para los estudiantes de la universidad sino para la comunidad de programación competitiva en general. Eventos especiales como participaciones en eventos, o la Maratón de Programación UFPS son también publicitadas por los medios de comunicaciones digitales de la carrera y la Universidad gracias a la gestión del director del semillero SILUX, y por medio de anuncios en tableros informativos.

11.3. Sesiones semanales de entrenamiento

Antes de iniciar el proyecto, se detectó que el no contar con una periodicidad definida para las actividades incrementa la deserción de estudiantes en el grupo. Por tanto, se llevan a cabo sesiones semanales con un horario convenido entre todos los miembros del grupo al iniciar cada

semestre. (Esta actividad es independiente de las competencias organizadas por redes externas a la Universidad).

11.4. Competencias de entrenamiento de RPC y CCPL

La Red de Programación Competitiva (RPC) y la Liga Colombiana de Programación (CCPL) realizan durante el año entre 12 y 14 competencias cada una, siendo por lo general los sábados de 1pm a 6pm. Estas dos redes son un punto de medición directo para comparar a la Universidad frente a las demás Universidades de país y de la región que generalmente participan en dichas competencias, y por tanto siempre que sea posible, la Universidad oficia como sede presencial de dichas competencias. Esto se realiza de la mano del director del semillero SILUX para obtener los permisos necesarios para el uso de salas y equipos. En algunas ocasiones, ambas redes organizan competencias para el mismo día, lo que obliga a elegir solo una de las dos. Pero mientras son organizadas en fechas diferentes, se participa en todas las fechas.

11.5. Recolección de datos de los participantes en cada sesión con fines estadísticos y de análisis

En cada una de las sesiones, competencias y actividades realizadas se recopila información con fines estadísticos y de control. Dado que la carrera cuenta con su propio formato de asistencia con fines de control de calidad, se utiliza este formato en cada una de las sesiones, y de él se extraen los datos necesarios antes de ser entregados como parte de la gestión documental de SILUX.

12. Metodología de trabajo en el grupo de estudio

Como se mencionaba en el diagnóstico, uno de los problemas que llevaban a la deserción de los estudiantes del grupo a lo largo del semestre era la falta de organización en las actividades realizadas. En los inicios del grupo, intentábamos realizar una sesión semanal que podía ser, o bien una charla sobre algún tema en específico (en cuyo caso la sesión se realizaba en un día de lunes a viernes) o bien competencias con RPC/CCPL (en cuyo caso la sesión se realizaba el sábado). En aquel entonces no habíamos definido claramente los mecanismos de promoción explicados en el capítulo anterior, ni existían unas temáticas definidas para explicar, sino a medida que se acercaba la fecha, se elegía un tema arbitrariamente.

Este mecanismo era a simple vista bastante desorganizado. Por tanto, uno de los ejes funcionales de este proyecto consiste en definir una metodología que los estudiantes líderes del grupo puedan aplicar y los asistentes del grupo llevar a cabo para mantener un norte durante el desarrollo de las actividades del semestre.

La metodología de trabajo propuesta parte de las actividades mencionadas en el eje anterior, y de los mecanismos de aprendizaje mencionados en el marco teórico.

12.1. Roles y responsabilidades

En el grupo, todos los estudiantes tienen las mismas responsabilidades, y los mismos incentivos. Sin embargo, para fines organizativos se hacen algunas distinciones:

Tabla 7. Roles y responsabilidades

Rol	Descripción
Líderes del grupo	Estudiantes que realizan las labores administrativas del grupo, y gestionan la comunicación entre el docente director del semillero, y los estudiantes que conforman el grupo. Son los encargados de promover acciones y cambios en el grupo de ser necesario.
Estudiantes con experiencia	Estudiantes que han asistido por lo menos un semestre al grupo, y han participado en competencias de RPC/CCPL. Ellos pueden llevar a cabo socializaciones, dirigir sesiones y proponer ejercicios.
Estudiantes nuevos	Estudiantes que asisten durante el semestre en curso por primera vez al grupo.

Cabe resaltar que esta clasificación se realiza solo con fines organizativos (resulta más fácil para los procesos del grupo tener definidos los estudiantes que pueden realizar una u otra actividad). Sin embargo, nada impide a un estudiante nuevo llevar a cabo socializaciones o sesiones, ni a un estudiante con experiencia proponer acciones directas para el grupo. Ante todo, el grupo mantiene un enfoque abierto.

12.2. Aprendizaje colaborativo y aprendizaje basado en problemas

En primer lugar, establecemos la periodicidad de las sesiones de entrenamiento como semanal, y las competencias siguen estando regidas por los calendarios de RPC/CCPL. El hecho de hacer sesiones solo en las semanas que no había competencias, implicaba pérdida de ritmo

cuando se encontraban competencias seguidas, además de terminar contando con pocas sesiones. De esta forma, podemos contar desde el inicio de semestre con una cantidad de semanas fijas sobre las cuales definir las temáticas a tratar. Las sesiones semanales dan inicio con la reunión de socialización, y terminan una semana antes de finalizar las clases del semestre (no se realiza sesión en la última semana para evitar sobrecargar a los estudiantes que están a punto de iniciar exámenes finales). Por la misma razón, durante las dos semanas de previos se realiza una sola sesión, dejando libre una de las dos semanas para que los estudiantes se enfoquen en sus previos (puede ser la primera o segunda semana, según socialización previa con los estudiantes).

Una sesión se divide en dos partes: Sección básica y sección avanzada. En la sección básica se inicia desde fundamentos de programación y temas de iniciación (enfocados siempre en maratones de programación), mientras en la sección avanzada se enseñan temáticas para las cuales se requiere tener mayores conocimientos previos. Los estudiantes pueden participar en cualquiera de las dos secciones según su nivel académico actual. De esta forma se garantiza que estudiantes de cualquier semestre pueden participar, eligiendo las temáticas básicas aquellos que están en primeros semestres, y las avanzadas los de semestres finales. De cualquier forma, cabe indicar que no se impone ninguna restricción sobre la participación: Cada estudiante es el que define según su conocimiento a cuál sección desea asistir, o incluso puede participar de las dos al tiempo.

Cada sección se divide a su vez en dos partes: En una primera parte se realiza una socialización sobre un tema específico o sobre un conjunto de problemas previamente vistos, y en la segunda parte se realizan los ejercicios propuestos. Cabe indicar que, en la sección de ejercicios propuestos, se realiza trabajo individual asistido: Cada estudiante debe solucionar los

problemas individualmente, pero se permite (e incluso, se recomienda) la socialización para resolver dudas e inquietudes.

Existen unas temáticas principales (se especificarán en el siguiente capítulo) que por su importancia en competencias son temas de obligatoria revisión en el grupo. Según la cantidad de sesiones que se planteen, existe otro grupo de temáticas opcionales, tomando como referencia los lineamientos de Competitive Programming 3 (Halim & Halim, Competitive Programming, 2013). Los líderes del grupo con el apoyo del director del semillero definen el orden y las temáticas que harán parte del grupo durante el semestre entero. Posteriormente, entre los estudiantes que más han participado en competencias se definen quienes tienen mayor práctica en un tema, para dirigir su socialización. En temáticas de mayor dificultad puede solicitarse la participación de un docente del programa, dependiendo de su disponibilidad.

En estas sesiones se utiliza como eje el aprendizaje colaborativo (Universidad EAFIT, 2014): Cada sesión la dirige inicialmente entre uno y tres estudiantes, explicando el tema. Estos estudiantes recibirán un incentivo en el ranking (se explicará en los próximos párrafos). Estos incentivos permiten que los estudiantes decidan tomar la iniciativa de forma voluntaria, y tomar la voz en las temáticas que mejor manejan. Una vez se hace la explicación de un tema, se procede a socializar entre todo el grupo. Se busca el consenso grupal, que todos resuelvan sus dudas e inquietudes. En muchas ocasiones es común que los estudiantes no expresen sus preguntas en un primer instante; para incentivar la participación, quienes dirigen la sesión y los líderes del grupo pueden empezar a plantear preguntas abiertas, para verificar si los estudiantes han entendido correctamente, o si aún hay falencias. Esta actividad dura en total una hora.

A continuación, se plantean una serie de ejercicios a realizar. En las primeras iteraciones de este proyecto, estos ejercicios se plantean en diferentes plataformas de internet. Para la sesión

final, los ejercicios están accesibles directamente en el “UFPS Training Center”, plataforma de entrenamiento. Es en este punto donde se pone en práctica el aprendizaje basado en problemas (Servicio de Innovación Educativa de la Universidad Politécnica de Madrid , 2008). Los estudiantes se enfrentan a un conjunto de problemas de diferente nivel de dificultad, y buscan encontrar la solución óptima en el menor tiempo posible. Sin embargo, no por incluir un nuevo enfoque se deja de lado el aprendizaje colaborativo; por el contrario, los dos enfoques se complementan a la perfección. Los estudiantes a cargo de la sesión y los líderes del grupo constantemente se dirigen a los diferentes estudiantes para resolver sus dudas e inconvenientes, y se permite la colaboración entre los estudiantes para mejorar sus resultados.

Tanto la sección básica como la avanzada tienen la misma estructura, resultando al final de 2 horas cada sección. Si bien el horario se establece a conveniencia de los estudiantes, se busca que ambas sesiones se realicen el mismo día, una tras la otra para permitir que los estudiantes que lo deseen asistan directamente a las dos secciones.

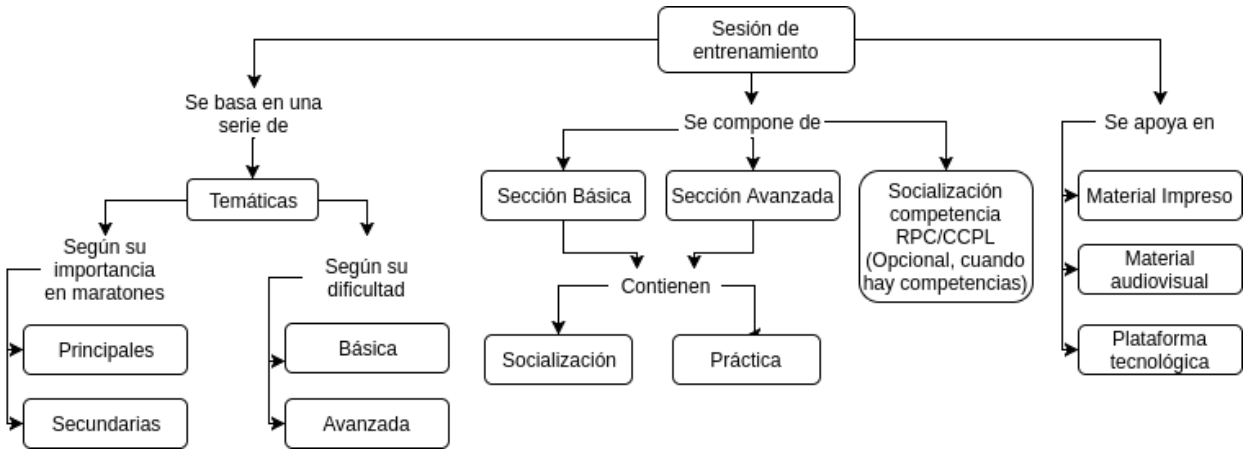


Figura 3. Estructura de una sesión de entrenamiento

12.3. Integración de las competencias en la metodología

Las competencias realizadas por RPC/CCPL cuentan con una plataforma, lenguajes, problemas y horarios definidos por la organización, y por tanto no está en nuestras manos realizar cambios sobre ellos (aunque en realidad tampoco sería necesario realizar cambios: Estas competencias siguen al pie de la letra las reglas de las competencias oficiales, y por tanto permiten ejercitarse en un ambiente exactamente igual al de dichas competencias).

Sin embargo, en la semana siguiente a una competencia de RPC/CCPL se dedican 20 minutos para socializar los ejercicios que se hayan resuelto por alguno de los equipos. Esta socialización es llevada a cabo sin ningún orden específico, cada ejercicio resuelto puede ser explicado por quien tome la iniciativa, y pueden realizarse preguntas. Se incentiva que después de estas socializaciones los equipos vuelvan a revisar las competencias pasadas e intenten resolver los ejercicios que no lograron realizar en la competencia, teniendo en cuenta las indicaciones de sus compañeros. Lo fundamental de esta socialización es descubrir las dificultades que tuvieron los estudiantes y responder sus inquietudes para fortalecer sus capacidades.

12.4. Actividades adicionales

Los ejercicios propuestos en la sesión tienen una hora para ser resueltos. Sin embargo, se mantiene un plazo de una semana para realizarlos, permitiendo su continuación en casa en caso de no terminarlos en la sesión.

Adicional a esto, se realizan algunas competencias nocturnas para realizarse desde casa. Estas actividades son opcionales, pero se recomienda su participación para obtener incentivos en el grupo.

12.5. Trabajo individual y trabajo grupal

Las competencias de programación se realizan de forma grupal. Sin embargo, es necesario que cada miembro del grupo se encuentre en la capacidad de resolver problemas con facilidad, para que todos los miembros sumen esfuerzos, en lugar de restar. Por tanto, las actividades propuestas buscan tener espacios individuales y espacios grupales, para fortalecer las capacidades de cada estudiante, y su trabajo en equipo.

A continuación, se presentan las diferentes actividades del grupo, y su carácter individual o grupal:

Tabla 8. Actividades de promoción del grupo.

Actividad	Carácter
Llevar a cabo una sesión o socialización	Individual o grupal*
Espacio para la solución de ejercicios	Individual**
Competencias nocturnas	Individual
Competencias organizadas por RPC	Grupal
Competencias organizadas por CCPL	Grupal

*Una socialización es, por defecto, grupal (entre todo el grupo de estudio) pero su organización y dirección puede estar a cargo de un solo estudiante, o varios, según disponibilidad.

** Si bien se explicó que no está prohibida la socialización en este espacio, se cuentan los ejercicios de cada estudiante de forma individual.

12.6. Ranking e Incentivos

Anteriormente se han especificado diferentes actividades, y la principal complicación en este punto surge en lograr la participación activa de los estudiantes en cada una de esas actividades. En este punto, de nuevo surge una de las ventajas del aprendizaje basado en problemas: La sana competencia y el entorno competitivo promueven el esfuerzo de los estudiantes por lograr buenos resultados (Servicio de Innovación Educativa de la Universidad Politécnica de Madrid , 2008). Este punto lo llevamos al siguiente nivel: La realización de un ranking de los estudiantes del grupo.

Este ranking no es más que una calificación de los estudiantes con respecto a sus logros en el grupo. Sin embargo, la construcción de este ranking debe ser muy cuidadosa: En el deben incluirse todas las actividades que hacen parte del grupo. Realizar una sesión suma puntos, cada ejercicio resuelto suma, cada competencia, cada acción en específico que un estudiante realice. Al iniciar cada semestre, se define con el director del semillero y con los estudiantes los porcentajes específicos que cada actividad tendrá para el ranking.

El ranking tiene dos componentes: Individual y grupal. El individual, por supuesto, mide las actividades que cada estudiante realiza. El grupal, además de contar los resultados en RPC y CCPL asigna un porcentaje al desempeño individual de cada estudiante que conformaba ese grupo. Este ranking tiene una fecha definitiva de cierre, y los dos equipos que a esa fecha logren

ocupar los dos primeros puestos en el ranking, ganan el derecho a representar a la Universidad en la Maratón Nacional de Programación.

Esto, por tanto, permite un beneficio en 3 niveles diferentes: En primer lugar, incentiva a los estudiantes a participar en el grupo (se ha implementado desde el inicio de este proyecto, y los resultados, como se verá más adelante, han sido satisfactorios). En segundo lugar, permite elegir para las representaciones a aquellos estudiantes que en realidad lo merecen por su buen desempeño. En tercer lugar, el programa y la Universidad cuentan en estas competencias con estudiantes que la representen de forma adecuada, permitiendo posicionarse como una de las instituciones a tener en cuenta a nivel nacional.

Además de esto, algunas empresas como Google realizan eventos en los cuales invitan estudiantes de la Universidad a hacer parte de ellos. Este ranking se convierte en la mejor manera de elegir los estudiantes que asisten a estos eventos.

12.7. Integración con herramientas

En los capítulos posteriores se analizarán diferentes herramientas documentales y tecnológicas que hacen parte de este proyecto. Estas herramientas se integran a la metodología y hacen parte de ella en cada paso. Las sesiones de entrenamiento no son concebidas como clases magistrales, sino como experiencias educativas. Para construir esta experiencia resulta necesario complementar la socialización y el componente humano previamente expuesto con materiales y herramientas que fortalezcan y contribuyan al entendimiento y mejoramiento de los estudiantes.

En primer lugar, nos centraremos en el material para las competencias de programación. En una competencia de RPC/CCPL, o competencia oficial Nacional, Regional o Mundial, se

encuentra prohibida la utilización de cualquier material digital o acceso a internet. Por tanto, en esta parte, la metodología debe apoyarse en material impreso. Para tal fin, en el siguiente capítulo se establece el manual de competencias (Notebook), el cual se comparte con los miembros del grupo y se promueve su impresión y utilización en dichas competencias. En fases nacional y regional 2016 y 2017 este manual ha sido de utilización obligatoria por los miembros de los equipos participantes, y su utilización ha sido recurrente en competencias locales.

En segundo lugar, la metodología utilizada durante las sesiones de entrenamiento permite integrar una mayor cantidad de herramientas. Por un lado, las socializaciones pueden apoyarse en material explicativo (presentaciones) que estén disponibles para su visualización en cualquier navegador. De esta forma, los estudiantes pueden seguir la socialización directamente desde sus pantallas o desde el video beam. Por otro lado, existen materiales adicionales (externos o propios del grupo) que se proponen como guías para revisar posteriormente. Estos materiales se comparten, si sus derechos lo permiten, con los estudiantes del grupo. Del mismo modo, como se ha precisado previamente, las sesiones contienen un componente completamente práctico en el cual los estudiantes ponen a prueba sus capacidades algorítmicas en el desarrollo de problemas propuestos. En este espacio, los estudiantes cuentan con herramientas tecnológicas que les permiten probar directamente sus soluciones en un entorno controlado y saber su calificación. Todas estas herramientas mencionadas en este punto serán explicadas a profundidad en los siguientes capítulos.

12.8. Sesiones de entrenamiento a través de diferentes semestres

Las temáticas existentes en maratones de programación son extensas y un semestre no es suficiente para cubrirlo todo. Sin embargo, cada semestre hay alumnos nuevos en el grupo, así

como alumnos que continúan desde el semestre anterior. Por tanto, el grupo debe aportar semestre a semestre información introductoria que permita a los nuevos estudiantes empezar desde cero, pero también aportar nueva información, de modo que los estudiantes antiguos no se mantengan en una constante repetición de las mismas temáticas.

Para enfrentar esta problemática, de nuevo la solución surge en la división de la sesión en dos secciones: básica/avanzada. Se recomienda a los estudiantes nuevos, por lo menos en las primeras sesiones a las que asisten a hacer parte de la sección básica. De esta forma tendrá la introducción a las maratones y conceptos iniciales, y si posteriormente decide que tiene bases suficientes para pasar a la sección avanzada puede hacerlo. Por tanto, la sección básica se repite con un esquema similar semestre a semestre, que permite la iniciación en maratones de estudiantes nuevos.

Por su parte, las temáticas de la sección avanzada son un poco más flexibles y configurables, pudiendo establecerse cambios en conjunto entre el director, el grupo y los líderes al inicio del semestre. Existen suficientes temáticas avanzadas para cubrir varios semestres, pero como se ha dicho antes, algunas hacen parte de las maratones con tanta frecuencia, que tienen una importancia primordial. Por tanto, esta sección se conforma como un balance entre las temáticas principales que han sido vistas antes, como temáticas nuevas que pueden aparecer esporádicamente. Así, los estudiantes se mantienen entrenando en temas de importancia, y siguen aprendiendo nuevas cosas aun cuando llevan varios semestres en el grupo.

13. Producción documental

El segundo eje de este marco de trabajo tiene como objetivo construir un banco documental que abarque todos los contenidos que se trabajan durante el semestre y que sirva para evidenciar las actividades realizadas, así como para ser una fuente de información y conocimiento.

Toda esta producción documental debe ser pública para los estudiantes del grupo, de forma que puedan acceder a él tanto en las actividades del grupo, como en sus casas de forma independiente. Según las necesidades y conveniencia, puede existir documentación en diferentes formatos: Impresa, digital o audiovisual, las cuales se complementarán de forma efectiva.

Para esto se han establecido las siguientes actividades:

13.1. Planteamiento de la estructura del banco documental

El sitio en el cual se encuentre la información debe estar estructurado de forma que sea fácilmente accesible. Los estudiantes deben poder encontrarla con facilidad y estar disponible todo el tiempo. Este por tanto debe funcionar como un repositorio, en el cual los estudiantes puedan añadir contenido que consideren de importancia, y además debe mantenerse abierto a la actualización: Con el tiempo los materiales se hacen obsoletos y debe ser fácil poder modificar esos contenidos según corresponda.

Además, el contenido del banco contendrá múltiple material de diversa índole, y por tanto el banco necesita adaptarse a diferentes tipos de contenido.

Para esto es posible apoyarnos en herramientas externas. Aprovechando el apoyo del Semillero SILUX, el cual fomenta el uso de software libre, optamos por utilizar Github y sus facilidades para el manejo de herramientas libres adaptadas a la educación. De esta forma,

podemos organizar la información de forma completa y correcta, y permitimos que el espacio sea abierto para su acceso y modificación controlada por todos los miembros del grupo.

13.2. Creación de los repositorios y la organización

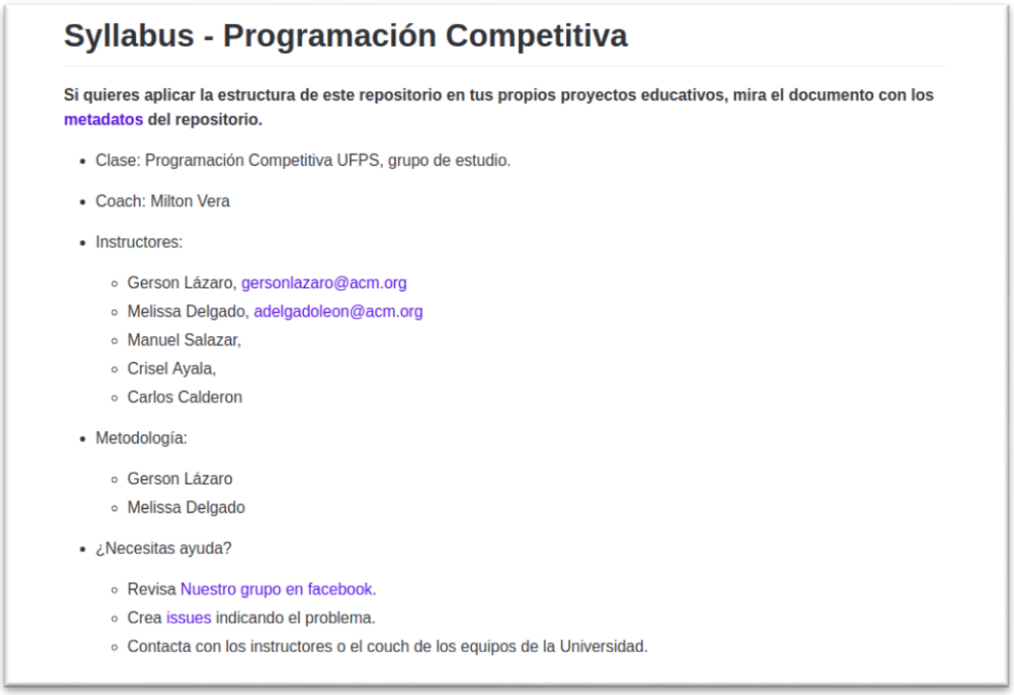
Con base en lo anterior, se creó una organización a través de GitHub en la cual mantener toda la información del Grupo de Estudio llamada “Programación Competitiva UFPS” (accesible a través de https://github.com/programacionCompetitivaUFPS). Esta organización es administrada por las personas que en cada semestre se encuentren a cargo del grupo, pero bajo el principio de colaboración de Github, todos los demás pueden colaborar sin necesidad de ser administradores.

Figura 4. Vista inicial de la organización en GitHub

Dentro de ella se crearon los repositorios que contienen la información estructurada bajo la cual se guía el grupo. Los repositorios principales de esta organización son: Syllabus, soluciones, presentaciones y notebook (serán expuestos a continuación). Junto a estos se encuentran otros repositorios temporales u organizativos que, si bien contienen información de utilidad para el grupo, no se describirán por carecer de importancia teórica (Por ejemplo, plantillas para la realización de las presentaciones o informes de las reuniones realizadas).

13.3. Repositorio Syllabus

El primer repositorio disponible es el “Syllabus”, el cual puede accederse desde el siguiente enlace: https://github.com/ProgramacionCompetitivaUFPS/Syllabus. Ha sido diseñado siguiendo la Guía de Github para el aula de clase (GitHub, 2016). En ella se encuentra una corta introducción al grupo, las temáticas a tratar, los métodos de entrenamiento, información de los horarios del grupo, información sobre las competencias y demás información de interés. También se aportan enlaces a material externo útil, y está estructurado de forma que pueda ser reproducible por otros proyectos similares. Este es el punto de partida para los estudiantes del grupo.



Syllabus - Programación Competitiva

Si quieres aplicar la estructura de este repositorio en tus propios proyectos educativos, mira el documento con los metadatos del repositorio.

- Clase: Programación Competitiva UFPS, grupo de estudio.
- Coach: Milton Vera
- Instructores:
  - Gerson Lázaro, gersonlazaro@acm.org
  - Melissa Delgado, adelgadoleon@acm.org
  - Manuel Salazar,
  - Crisel Ayala,
  - Carlos Calderon
- Metodología:
  - Gerson Lázaro
  - Melissa Delgado
- ¿Necesitas ayuda?
  - Revisa Nuestro grupo en facebook.
  - Crea issues indicando el problema.
  - Contacta con los instructores o el couch de los equipos de la Universidad.

Figura 5. Fragmento inicial del Syllabus (Acceder al sitio para ver una versión completa y actualizada)

13.4. Repositorio de Material de apoyo y enseñanza en las sesiones del grupo

En las sesiones realizadas en el grupo se expone diferente material que se encuentra accesible para su estudio posterior. Este material se compone de:

13.4.1. Presentaciones explicativas: A través de diapositivas se realizan algunas de las explicaciones teóricas (complementándose con el tablero y herramientas visuales). Estas presentaciones son realizadas en forma web, y accesibles a través del navegador en cualquier dispositivo de escritorio o móvil, con lo cual los estudiantes pueden accederla en todo momento. Se ha diseñado además la plantilla web para estas presentaciones, de tal forma que los estudiantes puedan acceder a ella y puedan crear nuevas explicaciones.

13.4.2. Videos explicativos: Este es un proyecto complementario que originalmente no hace parte del proyecto presentado, pero ha complementado las presentaciones, permitiendo a los estudiantes ver las explicaciones desde la comodidad de sus casas, y repetirlas tantas veces como sea necesario. Este es un proyecto del semillero Silux, pero sus contenidos al ser abiertos, y creados por los mismos autores de este trabajo, han sido incluidos en el repositorio.

13.4.3. Material de lectura adicional: Documentos adicionales que pueden colaborar en el desarrollo del grupo. Estos pueden ser materiales externos de otros autores (libros, documentos o entradas de blog) en cuyo caso se adjunta el link con los créditos respectivos, o documentos propios del grupo, como la guía de iniciación en C++ para personas conocedoras de Java, documento creado por los autores de este proyecto para dar una introducción fácil a C++ a los estudiantes del grupo.

Todos estos materiales son accesibles para su edición y descarga a través del repositorio “slides” del grupo, y puede ser visualizado desde la plataforma de software explicada posteriormente.

13.5. Repositorio de Manual de Competencias (Notebook)

Durante las competencias de programación oficiales a nivel universitario no se permite el uso de ningún dispositivo electrónico, pero si es válido utilizar todo el material impreso que se desee llevar. Dado que la competencia es contra reloj, se busca que los estudiantes cuenten con un manual muy ligero con los códigos de diferentes algoritmos que puedan necesitar durante la competencia.

Este manual no debe tomarse como una guía de enseñanza, sino como una herramienta a utilizar en el momento de competir. Por tanto, en este documento no se consiguen explicaciones o tutoriales (estos son cubiertos en las sesiones con otros materiales disponibles en el repositorio) sino solo códigos, tips y datos a tener en cuenta a la hora de programar.

Este notebook se realiza en dos versiones diferentes: C++ y Java (Lenguajes de programación aceptados en competencia). Ambas versiones contienen el mismo material, y simplemente cambian el lenguaje del código, para que cada estudiante utilice aquel cuyo lenguaje prefiera. En épocas recientes se han añadido dos nuevos lenguajes oficiales a las competencias (Python y Kotlin) y en caso de que a futuro un número importante de estudiantes del grupo se decante por alguno de estos dos lenguajes, fácilmente podría “traducirse” a ellos y crear nuevas variaciones del notebook. Hasta el momento no ha sido necesario, pues todos los estudiantes utilizan java o C++ como lenguaje principal durante las competencias.

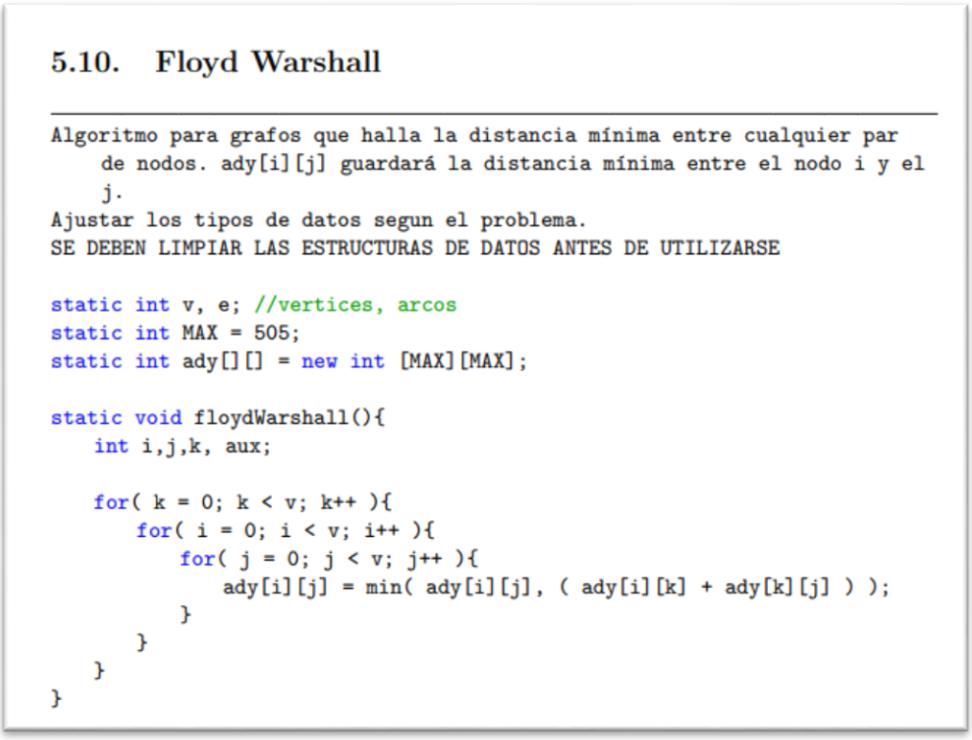
13.6. Proceso de creación de los notebooks

Dentro de la organización en GitHub, uno de los repositorios es el llamado “notebook”. Puede accederse desde: https://github.com/ProgramacionCompetitivaUFPS/notebook. En dicho repositorio se encuentra una carpeta para cada uno de los lenguajes que tienen un notebook actualmente, y dentro de ellas hay una carpeta por cada temática general. Allí se encuentran ubicados los códigos de cada temática, en archivos .java (lenguaje java) o .cpp (lenguaje c++).

A medida que el grupo realiza sus sesiones y se ven nuevas temáticas, son los mismos estudiantes los que proponen que algoritmos deberían estar en el notebook. Pueden hacerlo a través del repositorio, o indicándolo a los encargados del grupo para que ellos agreguen el respectivo issue. Es importante que cada código solicitado aparezca en un issue, pues es de allí de donde se toman las referencias para los nuevos temas a agregar.

Cualquier estudiante que desee puede clonar el repositorio, y contribuir con sus soluciones a los issues pendientes. Una vez un código nuevo es añadido, este es sometido a prueba, y si cumple con los requerimientos, el issue se cierra, y el código en cuestión es incorporado al repositorio.

De esta forma, el notebook se actualiza constantemente. Cuando se acercan las competencias oficiales de programación, se hace necesario contar con estos códigos consolidados en un pdf imprimible para llevar al sitio de competencia. Para esto se ha utilizado “notebook generator” creado por Manuel Pineda de la Universidad Tecnológica de Pereira (Pineda, 2015), utilidad que toma la carpeta con los códigos y lo convierte en un documento látex correctamente maquetado.



5.10. Floyd Warshall

```
Algoritmo para grafos que halla la distancia mínima entre cualquier par
    de nodos. ady[i][j] guardará la distancia mínima entre el nodo i y el
    j.
Ajustar los tipos de datos segun el problema.
SE DEBEN LIMPIAR LAS ESTRUCTURAS DE DATOS ANTES DE UTILIZARSE

static int v, e; //vertices, arcos
static int MAX = 505;
static int ady[][] = new int [MAX][MAX];

static void floydWarshall(){
    int i,j,k, aux;

    for( k = 0; k < v; k++ ){
        for( i = 0; i < v; i++ ){
            for( j = 0; j < v; j++ ){
                ady[i][j] = min( ady[i][j], ( ady[i][k] + ady[k][j] ) );
            }
        }
    }
}
```

Figura 6. Captura de pantalla del algoritmo de Floyd Warshall tomado directamente del Notebook Java.

Desde el inicio de este proyecto, se han lanzado cuatro versiones de los notebooks (actualizándose con nuevos contenidos).

Tabla 9. Versiones de los notebooks del grupo.

Versión	Año	Comentario
v1.0.0	2015	Versión lanzada para utilizarse en las competencias nacional y regional 2015. Versión preliminar, lanzada antes de formalizar este proyecto.
v2.0.0	2016	Actualización de la versión anterior, lanzada para utilizarse en la Maratón de Programación UFPS 2016, y las competencias nacional y regional del mismo año

v3.0.0	2017	Actualización lanzada para utilizarse en la maratón de Programación UFPS 2017.
v4.0.0	2017	Actualización lanzada para utilizarse en la maratón Nacional y Regional 2017.

Actualmente, el notebook se encuentra estructurado de la siguiente forma:

Tabla 10. Estructura de los notebooks

	Categoría	**Descripción**	**Contenido**
1	Input/Output	Indicaciones sobre lectura y escritura de dato a través de la entrada/salida estándar, y formas de optimizar el tiempo de ejecución de estas operaciones.	Java: 2 códigos de ejemplo C++: 1 código de ejemplo
2	Estructuras de datos	Códigos para crear e inicializar estructuras de datos complejas que no se encuentran incluidas en los lenguajes por defecto: Conjuntos disyuntos, árboles segmentados (Algunas estructuras como String Hashing, Suffix Array o grafos se encuentran en otras categorías).	2 códigos de ejemplo en cada lenguaje (Java/C++)
3	Programación dinámica	Ejemplos clásicos de Programación dinámica que pueden aplicarse en problemas de diferentes contextos: Problema de la mochila, subsecuencia común más larga, etc.	4 códigos de ejemplo en cada lenguaje (Java/C++)

4	Geometría Computacional	Clases para construir elementos geométricos de forma óptima (punto, línea, vector, polígono) y operaciones sobre ellos (distancias euclidianas, envolventes convexas, entre otros)	10 códigos de ejemplo en cada lenguaje (Java/C++)
5	Grafos	Construcción óptima de la librería para ejercicios de grafos, y códigos para llevar a cabo los principales algoritmos sobre ellos: Dijkstra, Prim, Kruskall, Floyd Warshall, entre otros.	Java: 19 códigos de ejemplo C++: 15 códigos de ejemplo
6	Matemáticas	Algoritmos matemáticos y de teoría de números: Test de primalidad de Miller Rabin, algoritmo de Euclides, Criba de Eratóstenes, transformada rápida de Fourier, entre otros.	Java: 12 códigos de ejemplo C++: 13 códigos de ejemplo
7	Strings	Algoritmos de búsqueda eficiente sobre Strings: String Hashing, suffix arrays, entre otros.	9 códigos de ejemplo en cada lenguaje (Java/C++)
8	Tips y fórmulas	Secuencias comunes (Fibonacci, triangulares, tribonacci, etc.), detalles sobre complejidad algorítmica a tener en cuenta, y otros tips de competencia.	4 ítems (esta sección no contiene código y es similar en ambos lenguajes)

14. UFPS Training Center

Todas las actividades desarrolladas en el grupo de estudio tienen un punto de encuentro a través de una plataforma web llamada UFPS Training Center. Este tercer eje de proyecto es el componente software que funciona como punto de encuentro de todos los demás componentes.

La plataforma de entrenamiento cuenta con ejercicios de práctica de diferentes temáticas y niveles de dificultad los cuales pueden ser enviados y calificados a través de la misma, explicaciones teóricas y multimediales de las diferentes temáticas, un modo de aprendizaje guiado a través de las distintas categorías de ejercicios, seguimiento del desempeño de los estudiantes que puede ser utilizado tanto en el grupo como por los docentes del programa que dictan temáticas afines y deseen integrarlo a sus áreas, competencias de programación en tiempo real, y sistema de ranking para potenciar el aprendizaje basado en problemas.

La intención con esta plataforma es dar al estudiante todas las herramientas que necesita para llevar a cabo su entrenamiento en un solo sitio: desde el UFPS Training Center el estudiante puede estudiar, puede practicar, puede competir, y puede mejorar sus habilidades tanto en programación competitiva como en las propias áreas de programación de la carrera. De esta forma, el estudiante tiene al alcance de su mano todo lo necesario para aprender, y quienes dirigen el grupo tienen al alcance de su mano todos los reportes de competencias y rendimiento para tomar decisiones en el grupo. Del mismo modo, los profesores de la carrera que dictan materias relacionadas con la programación pueden encontrar en esta plataforma una aliada para sus clases prácticas, tareas y evaluaciones.

14.1. Análisis preliminar

Dada la finalidad del grupo de estudio: formar estudiantes con capacidades en programación competitiva, es obligatorio contar con un componente de software donde poder poner a prueba y mejorar las habilidades de los estudiantes. Buscamos por tanto contar con una plataforma que tuviera como eje la resolución de ejercicios de competencias de programación, en la cual se pudiera hacer seguimiento del progreso individual y colectivo, sugiriendo planes guiados de entrenamiento y en los cuales pueda encontrar a mano el material de entrenamiento y guía que lo ayude a resolver dichos ejercicios.

Antes de iniciar la construcción de una herramienta de software propia, en el grupo se analizaron diferentes herramientas ya existentes que pudieran suplir las necesidades existentes. Algunas de ellas fueron probadas en el grupo de estudio, otras fueron analizadas según su funcionalidad.

Un detalle importante es que las desventajas mencionadas a continuación, son desventajas con respecto a las necesidades del grupo expuestos previamente, y no necesariamente desventajas generales de la plataforma.

Tabla 11. Herramientas externas utilizadas en el grupo de estudio

Herramienta	Descripción	Ventajas	Desventajas
UVA Online Judge	Accesible a través de https://uva.onlinejudge.org/		
	Juez Online automatizado de problemas de	- Miles de ejercicios disponibles. - Ejercicios de	- Disponibilidad: UVA Online Judge presenta problemas para el acceso

	programación de la Universidad de Valladolid.	diferentes competencias oficiales.	regularmente. - Imposibilidad de asignar ejercicios a usuarios. - Imposibilidad de ver soluciones realizadas. - Imposibilidad de crear competencias.
COJ Online Judge	Accesible a través de http://coj.uci.cu		
	Juez Online automatizado de problemas de programación del Caribe (Caribbean Online Judge)	- Ejercicios de múltiples temáticas. - Estadísticas completas de ejercicios y usuarios.	- Imposibilidad de asignar ejercicios a usuarios. - Imposibilidad de crear competencias. - Imposibilidad de realizar seguimiento.
URI Online Judge	Accesible a través de https://www.urionlinejudge.com.br		
	Juez Online Automatizado de problemas de programación de la Universidad Regional Integrada (Brasil)	- Mecanismo de seguimiento. - Posibilidad de asignar ejercicios a usuarios. - Modo “coach” que permite llevar a cabo múltiples actividades.	- Base de datos de ejercicios limitada (muchas de las temáticas explicadas en el grupo no tienen ejercicios en URI) - Imposibilidad de crear nuevos problemas.

Codeforces	Accesible a través de http://codeforces.com		
	Juez Online de Programación Competitiva.	- Gran cantidad de ejercicios. - Ranking de usuarios.	- Imposibilidad de añadir ejercicios. - Imposibilidad de asignar ejercicios a usuarios. - Pocos ejercicios de nivel básico (es una plataforma por lo general avanzada).
A2 Online Judge	Accesible a través de https://a2oj.com		
	Juez de Programación Online.	- Permite crear competencias. - Permite añadir ejercicios desde diferentes jueces.	- No se pueden añadir ejercicios desde cero (deben estar en un juez externo previamente). - Solo modo competencia (no hay forma de practicar fuera de competencia). - Solo ejercicios, imposibilidad de añadir/ver información adicional (tutoriales o guías).
VJudge	Accesible a través de https://vjudge.net		
	Juez de Programación	- Permite crear competencias.	- No se pueden añadir ejercicios desde cero (deben

	Online.	- Permite añadir ejercicios desde diferentes jueces.	estar en un juez externo previamente). - Solo modo competencia (no hay forma de practicar fuera de competencia). - Solo ejercicios, imposibilidad de añadir/ver información adicional (tutoriales o guías).
HackerEarth	Accesible a través de https://www.hackerearth.com		
	Plataforma para desarrolladores: Entrenamiento, problemas de programación, materiales de entrenamiento y preparación para entrevistas técnicas.	- Posee material además de los ejercicios. - Ejercicios categorizados. - Se pueden crear competencias (a través de una cuenta de embajador estudiantil)	- No es posible añadir nuevos materiales. - En las pruebas realizadas con esta herramienta, hubo inconformidad de los estudiantes (errores del juez, lentitud, dificultad para entender la plataforma).
Hacker Rank	Accesible a través de https://www.hackerrank.com		
	Plataforma para entrenar	- Plataforma moderna. - Múltiples categorías.	- Enfocada principalmente en la sección de trabajos

	programación competitiva y conseguir trabajos tecnológicos a través de sus ejercicios.		tecnológicos, un enfoque que no se ajusta a la finalidad del grupo.
Kattis	Accesible a través de https://open.kattis.com		
	Plataforma moderna de maratones de programación.	- Tiene una de las interfaces gráficas más modernas y atractivas entre los softwares de su tipo. - Usada de forma oficial en competencias en algunas regiones de América y Europa.	- Imposibilidad de añadir materiales diferentes a ejercicios. - Imposibilidad de añadir nuevos ejercicios. - Imposibilidad de realizar seguimiento.
ICPC Live Archive	Accesible a través de https://icpcarchive.ecs.baylor.edu		
	Plataforma que contiene todos los ejercicios de maratones oficiales regionales y	- Todos los ejercicios hacen parte de competencias oficiales y, por tanto, son de alta calidad.	- Imposibilidad de asignar ejercicios a usuarios. - Imposibilidad de ver soluciones realizadas. - Imposibilidad de crear

	mundiales a lo largo del tiempo para entrenar.		competencias.
Acepta el reto	Accesible a través de https://www.aceptaelreto.com		
	Juez online con ejercicios de programación.	- Juez con solo problemas en español.	- Número bajo de ejercicios (comparado con los demás jueces previamente expuestos)
BOCA	Versión para instalar en https://www.ime.usp.br/~cassio/boca		
	Plataforma de código libre para ejecutar maratones de programación.	- Software open source que puede instalarse para correr maratones propias. - Posibilidad de añadir ejercicios nuevos.	- Solo maratones de programación: No se puede añadir ejercicios de entrenamiento, ni guías, ni ningún otro tipo de material.
PC^2	Versión para instalar en http://pc2.ecs.csus.edu		
	Plataforma de código libre para ejecutar maratones de programación de la universidad estatal de	- Software open source que puede instalarse para correr maratones propias. - Posibilidad de añadir ejercicios	- Solo maratones de programación: No se puede añadir ejercicios de entrenamiento, ni guías, ni ningún otro tipo de material.

	california, de sacramento.	nuevos.

Las celdas marcadas con color de fondo representan herramientas que fueron utilizadas dentro del grupo de estudio durante una o varias sesiones. Las no subrayadas solo fueron estudiadas y probadas como parte del proyecto, pero no de forma pública en el grupo.

Después de analizar todas estas herramientas y de utilizar muchas de ellas, siempre existía un punto muerto: A través de estas herramientas fue posible llevarse a cabo el entrenamiento, pero existieron siempre dificultades para coordinar todas las herramientas que intentamos colocar a disposición de los estudiantes. En primer lugar, en la mayoría de plataformas no era posible añadir nuestros propios ejercicios, estando obligados a usar las bases de datos que las propias plataformas proveían. En algunas ocasiones en las cuales era necesario tratar una temática compleja, era difícil encontrar ejercicios que se ajustaran justamente al tema explicado. Del mismo modo, cualquier guía o material adicional debía entregarse a los estudiantes a través de correo electrónico o link, pues estas plataformas no permiten añadirlo.

Con el objetivo de analizar los resultados de los estudiantes y elegir los de mejores resultados para representar a la Universidad, necesitamos tener un reporte y seguimiento de su actividad. De nuevo, esto resultó ser una tarea compleja pues las plataformas no proveen elementos con estos fines, y en muchas ocasiones debieron realizarse reportes manuales, revisando uno a uno los estudiantes a través de sus propias cuentas para determinar su rendimiento.

Por lo tanto, podemos concluir que si bien fue posible realizar este entrenamiento utilizando estas plataformas preexistentes - a las cuales de hecho agradecemos por haber sido de gran ayuda

mientras conformábamos este grupo, y que seguimos recomendando como espacios de entrenamiento adicional - sin duda para el grupo de estudio necesitábamos algo más.

Se necesita una herramienta completa, que pueda gestionar todas las actividades del grupo: Entrenamientos, sesiones y competencias. Allí debe ser posible acceder al material complementario generado, pero además resolver ejercicios de práctica y competir contra los demás estudiantes del grupo. También debe ser posible definir nuestras propias líneas para proponer los ejercicios que consideramos que los estudiantes deben realizar, y hacer seguimiento individual y grupal a esos resultados. Además, debe ser posible añadir ejercicios, tanto por parte de los líderes del grupo, como de los profesores para sus propias materias. Por último, integrarse como una herramienta que pueda complementar las actividades de los docentes en su aula de clase.

14.2. Requerimientos

Una vez se ha hecho notoria la necesidad de una herramienta de software, procedemos al análisis de requerimientos de la herramienta. Para esto, utilizamos tres mecanismos:

- Entrevista grupal abierta con el grupo de estudio, en la cual se pide a los estudiantes que opinen sobre las funcionalidades que debe tener dicha herramienta para su correcto funcionamiento.
- Entrevista con director de semillero Silux (En el momento del análisis, el director era el Ing. Fredy Vera).
- Reunión interna entre los líderes del grupo de estudio.

A raíz de estas reuniones, se definen los siguientes requerimientos:

Tabla 12. Requerimientos funcionales iniciales

Id.	Nombre	Descripción
RF-01	Listar Problemas	El sistema debe permitir a los estudiantes ver la lista de problemas disponibles en la plataforma categorizados según su temática.
RF-02	Agregar problemas	El sistema debe permitir a los profesores/coach agregar nuevos problemas a la plataforma.
RF-03	Calificar problemas	El sistema debe calificar las soluciones a los problemas que los estudiantes envíen, indicando si su solución es correcta, o si ha cometido errores.
RF-04	Corregir problemas	El sistema debe permitir a autores y administradores editar los problemas existentes.
RF-05	Eliminar problemas	El sistema debe permitir a los administradores eliminar problemas.
RF-06	Acceder al foro de discusión	El sistema debe permitir a los estudiantes publicar y leer el foro de discusión de cada problema, y a los profesores responder las problemáticas surgidas en este proceso.
RF-07	Filtrar problemas	El sistema debe permitir a los estudiantes filtrar los problemas bajo diferentes criterios de búsqueda.
RF-08	Categorizar problemas	El sistema debe permitir la categorización de los problemas según su tipo, y mostrarlos a los estudiantes según la categoría elegida.

RF-09	Ver ranking	El sistema debe generar un ranking según los ejercicios resueltos entre los estudiantes.
RF-10	Crear competencias	El sistema debe permitir la creación de competencias (maratones) de programación en vivo.
RF-11	Añadir material de ayuda	El sistema debe permitir la adición de material de ayuda para los estudiantes (diapositivas, videos o pdf).
RF-12	Ver material de ayuda	El sistema debe permitir a los estudiantes visualizar el material de ayuda a medida que soluciona los problemas.
RF-13	Crear Modo guiado	El sistema debe permitir a coach/profesores añadir un modo guiado con problemas y materiales específicos para hacer seguimiento a sus estudiantes.
RF-14	Recepcionar solución de tareas	En el modo guiado, los estudiantes envían las tareas propuestas por el coach/docente a la plataforma. El sistema debe permitir la recepción de estas tareas.
RF-15	Ver información del grupo en modo guiado	El sistema debe permitir al coach/docente ver la información general de las soluciones de las tareas que han enviado sus estudiantes.
RF-16	Ver soluciones de los estudiantes en modo guiado	El sistema debe permitir a los profesores/coach ver las soluciones de sus estudiantes a las tareas en el modo guiado.
RF-17	Ver estadísticas	El sistema debe permitir al estudiante ver sus propias estadísticas de envíos y soluciones.
RF-18	Eliminar usuarios	El sistema debe permitir al administrador eliminar usuarios no

			deseados.

Tabla 13. Requerimientos no funcionales iniciales

Id	Nombre	Tipo	Descripción
RNF-01	El sistema debe mantenerse estable aún en momentos de afluencia de público.	Eficiencia	Durante una sesión de entrenamiento o una competencia hay en promedio 30 estudiantes. En una clase práctica puede haber hasta 50 estudiantes. El sistema debe mantenerse estable aún si todos estos usuarios están conectados al tiempo.
RNF-02	Los envíos de soluciones no deben afectar la estabilidad del sistema.	Eficiencia	La calificación de una solución enviada por un estudiante implica su ejecución. Algunos problemas no tienen soluciones de complejidad algorítmica baja, y la ejecución de cada solución puede tardar varios segundos. Aún si la calificación presenta congestión por esta razón, el resto de la plataforma debe mantenerse estable y rápida.
RNF-03	Las contraseñas de los usuarios deben almacenarse encriptadas.	Seguridad	Las contraseñas de los usuarios se guardarán de forma encriptada en la base de datos, de tal forma que ni siquiera los administradores puedan acceder a ellas.
RNF-04	Las soluciones	Seguridad	Los estudiantes envían soluciones en forma de

	enviadas por los usuarios deben ejecutarse en modo “sandbox”.		código de programación que se califican automáticamente en la plataforma. Estos códigos deben ejecutarse en un modo “sandbox” que impida que un código malicioso realice daños en la plataforma o demás datos alojados en el servidor.
RNF-05	El tiempo de aprendizaje de un usuario con la plataforma debe ser menor a 20 minutos.	Usabilidad	La interfaz gráfica de la plataforma debe ser lo suficientemente simple para que los usuarios logren entenderla y utilizarla completamente en, como máximo, 20 minutos.
RNF-06	Manuales de usuario	Usabilidad	La plataforma debe contar con manuales de usuario debidamente documentados y claros.
RNF-07	El sistema debe funcionar en cualquier plataforma moderna.	Usabilidad/ Producto	La plataforma debe funcionar de forma similar en sistemas Windows/Linux/Mac OS. Esta debe operar a través de cualquier navegador web moderno (Google Chrome, Mozilla Firefox, Internet Explorer 10+, Microsoft Edge, Safari, Opera, Vivaldi, Maxthon o Brave Browser). En cualquier navegador no mencionado en esta lista la aplicación podría funcionar si utiliza tecnologías de HTML/CSS/Javascript modernas.

14.3. Metodología de desarrollo de software

Si bien el proyecto cuenta con una metodología previamente explicada, para el desarrollo de la aplicación fue necesario definir una metodología diferente que se ajustara a la creación de software. Luego de analizar los requerimientos, se puede observar como el software necesario se estructura a lo largo de una serie de módulos (autenticación, problemas, material, modo guiado, administración, competencias, estadísticas/perfiles) que pueden desarrollarse de forma escalonada e irse añadiendo al software a medida que se realizan. Una vez se añade un nuevo módulo, pueden realizarse sobre el todo tipo de pruebas de funcionamiento, para, una vez integrado por completo, continuar con la creación del siguiente módulo.

Por tanto, la metodología elegida para realizar esta herramienta es una metodología iterativa e incremental.

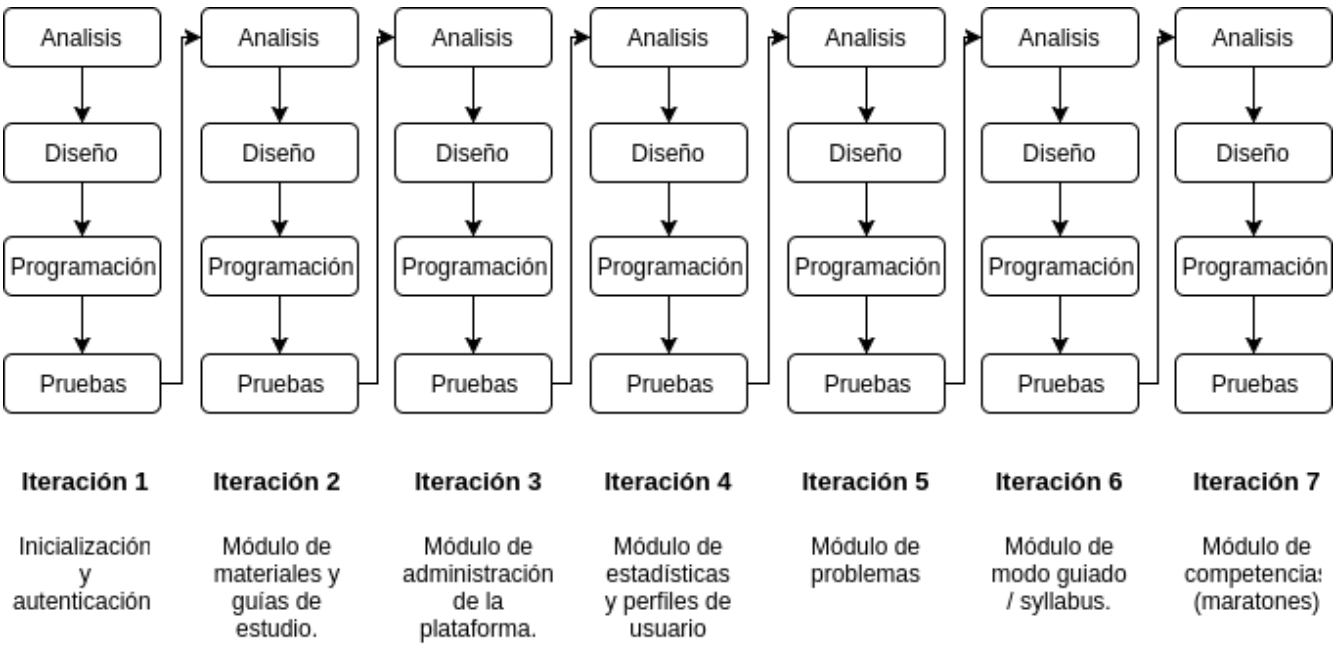


Figura 7. Metodología de desarrollo

La metodología iterativa e incremental nos permite ir realizando nuevos incrementos sobre la plataforma que han sido completamente testeados antes de pasar a la siguiente fase. Como se puede ver en el diagrama anterior, se han planteado 7 iteraciones, en cada una de las cuales se agrega un módulo hasta llegar a la iteración 7 en donde se encuentra la plataforma completa desarrollada.

14.4. Diseño del sistema

La documentación completa del diseño del sistema se adjunta en los anexos y para una mejor visualización de los diagramas, también pueden visualizarse como anexo digital (Ver anexos). De forma descriptiva se añaden a continuación el planteamiento del diseño de clases, el diagrama de base de datos, el diagrama de arquitectura bajo el cual se ha creado la plataforma utilizada y un resumen tanto de la arquitectura completa como de las necesidades arquitecturales que llevaron a este diseño.

En primer lugar, la plataforma se desarrolla bajo un esquema orientado a objetos y eventos. El diagrama de clases simplificado que representa el diseño de la aplicación es el siguiente (Para ver el diagrama de clases con todos los métodos y atributos, se recomienda ver el anexo previamente mencionado):

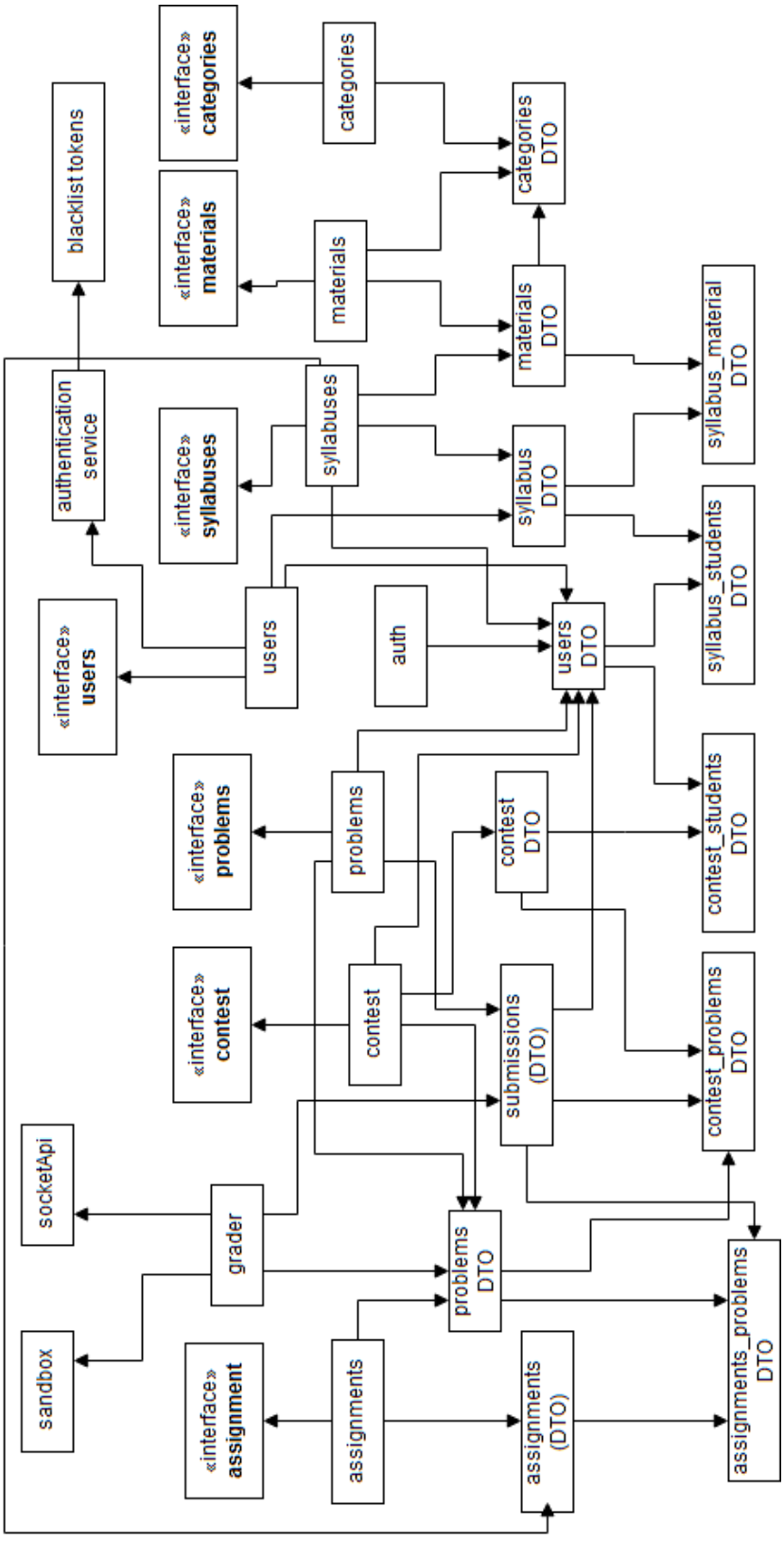


Figura 8. Diagrama de clases simplificado de la plataforma.

Esto se complementa con el siguiente modelo de datos:

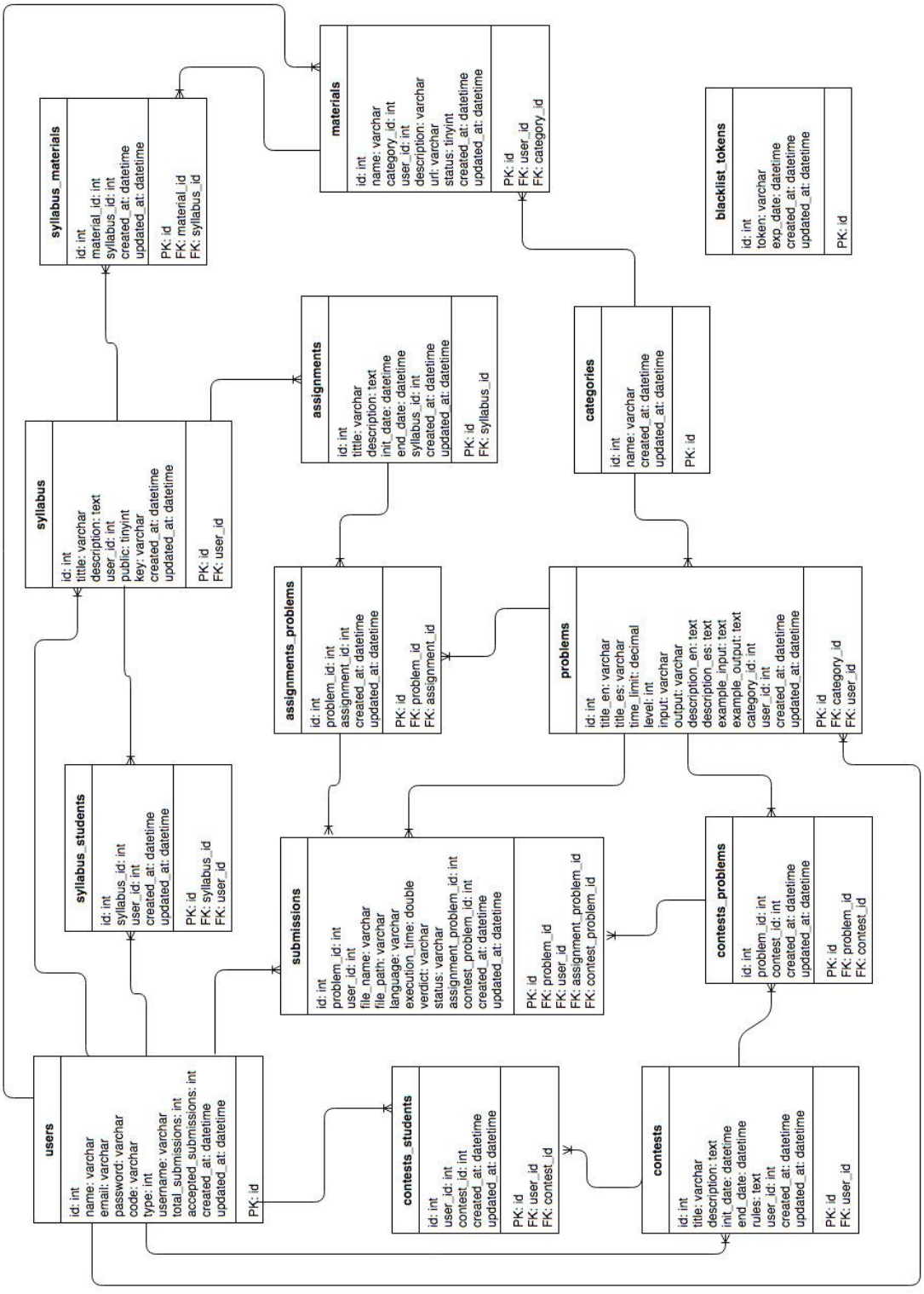


Figura 9. Diagrama de bases de datos.

14.5. Arquitectura de la aplicación

La aplicación debe resolver varios asuntos complejos:

En primer lugar, el hecho de ejecutar códigos escritos por los usuarios internamente obliga a mantener altos controles de seguridad para evitar la ejecución de códigos maliciosos, o incluso la ejecución de códigos que por simples errores de programación puedan generar comportamientos no deseados en el sistema.

En segundo lugar, es conocido de antemano que las calificaciones de las soluciones enviadas tendrán una demora prevista de incluso varios segundos, pues existen problemas cuyas únicas soluciones posibles tienen complejidades algorítmicas elevadas (incluso del orden O(2^n) u O(n!). Si bien se recomienda a quienes diseñan los problemas utilizar una cantidad de datos de prueba acorde con la complejidad algorítmica esperada, es inevitable que algunos algoritmos tarden un tiempo mayor a 1 segundo en ejecutarse. Si además la plataforma recibe varios algoritmos de este tipo al mismo tiempo, se puede presentar una carga considerable. Por tanto, la arquitectura debe garantizar independencia entre el sistema con el cual el usuario interactúa, y el motor de calificación de códigos ejecutado en el segundo plano. De esta forma, aun cuando las calificaciones tarden unos segundos en su procesamiento, el resto de la plataforma debe continuar su funcionamiento sin interrupciones ni demoras. El usuario debe poder continuar leyendo nuevos problemas o realizando cualquier acción interna mientras la calificación es entregada.

En tercer lugar, durante competencias es necesario contar con un factor realtime. No basta con calificar las soluciones enviadas, la plataforma debe informar a los usuarios en tiempo real sobre las calificaciones a sus problemas y los cambios en el puntaje de los demás estudiantes.

No menos importante, el sistema debe ser escalable. Originalmente se define este sistema para evaluar códigos en Python, Java y C++, que son los lenguajes oficiales utilizados en competencias de programación, y además Java es el lenguaje utilizado en las clases de programación de la Universidad. Sin embargo, siendo conscientes de que la informática se mantiene en constante actualización, es necesario que el sistema pueda adaptarse a nuevos lenguajes si en un futuro la organización o la carrera deciden optar por nuevas tecnologías (De hecho, Python ha sido agregado apenas al iniciar este proyecto, y ya se estudian nuevas adiciones por parte de la organización). Por tanto, el sistema debe ser escalable no solo en respuesta a usuarios sino también en posibles lenguajes a calificar. Su arquitectura debe permitir la integración de estos nuevos lenguajes sin necesidad de reescribir herramientas enteras.

Estas tres características a tener en cuenta conforman la base sobre la cual plantear la arquitectura de la plataforma.

Par asegurar la seguridad de la plataforma, los códigos enviados por los estudiantes serán calificados y ejecutados en un ambiente aislado. Este ambiente se logrará utilizando Docker. Docker es un software de código abierto que proporciona contenedores para la ejecución de tareas con una capa adicional de abstracción y virtualización a través del aislamiento de recursos. (Cuesta & González, 2016). De esta forma, el código se ejecuta sobre un contenedor que se encuentra aislado del resto del sistema operativo y de la aplicación, pero sin necesidad de ejecutar máquinas virtuales completas que conllevarían una degradación en el rendimiento por el consumo de recursos.

Sin embargo, Docker solo otorga la virtualización, el calificador debe ser construido sobre él. Esta es una de las partes más complejas de la aplicación, por la variedad de tecnologías que deben confluir en una sola: Los compiladores de Java, Python y C++ (incluyendo la máquina

virtual de Java) deben estar disponibles para la ejecución de los códigos y llamarse de forma automatizada utilizando Bash a través de Docker. Este debe ser además accesible desde la aplicación (es decir, desde fuera del contenedor) para que el sistema envíe los códigos y las entradas con las cuales debe ejecutarse. En este proceso deben utilizarse además múltiples herramientas libres de Linux como auxiliares en la ejecución: sed para realizar operaciones con los archivos, dev-essentials que contiene los headers necesarios para la compilación en C++, time para medir el tiempo de ejecución de los programas enviados por los estudiantes, entre otros detallados en la documentación técnica de la herramienta.

En primera instancia, sería posible mantener un solo contenedor que realice todas las calificaciones. Sin embargo, esto implicaría que en el momento de añadir un nuevo lenguaje deba editarse absolutamente todo el código del calificador, y volver a crear el contenedor con una nueva configuración. En vez de esto, optamos por crear un contenedor por cada lenguaje, lo que nos permite editar solo unas pequeñas configuraciones, definir las herramientas a utilizar, y el nuevo lenguaje estará disponible.

El servidor de la aplicación debe en este caso ser un punto de comunicación entre el usuario y el calificador, siendo externo a este último para no sufrir detrimento en su rendimiento, como se ha mencionado anteriormente. Por tanto, las necesidades para las operaciones del servidor se centran en realizar operaciones no bloqueantes (no tener que esperar hasta la calificación para continuar), ser ligero y eficiente aun cuando se realizan múltiples peticiones que conlleven uso intensivo de datos, y orientarse a eventos, pues es necesario que las calificaciones disparen eventos que cambien las estadísticas, ranking y notifique al usuario de sus resultados. Por esta razón se ha elegido utilizar node.js en el servidor, un entorno basado en JavaScript (lenguaje orientado a eventos) y que responde fielmente a estas necesidades.

Una de las herramientas más conocidas al trabajar node.js es socket.io, que a través de web sockets permite la comunicación realtime entre cliente y servidor, lo cual permitirá perfectamente mantener a los usuarios notificados de sus resultados, bajo el mismo enfoque orientado a eventos.

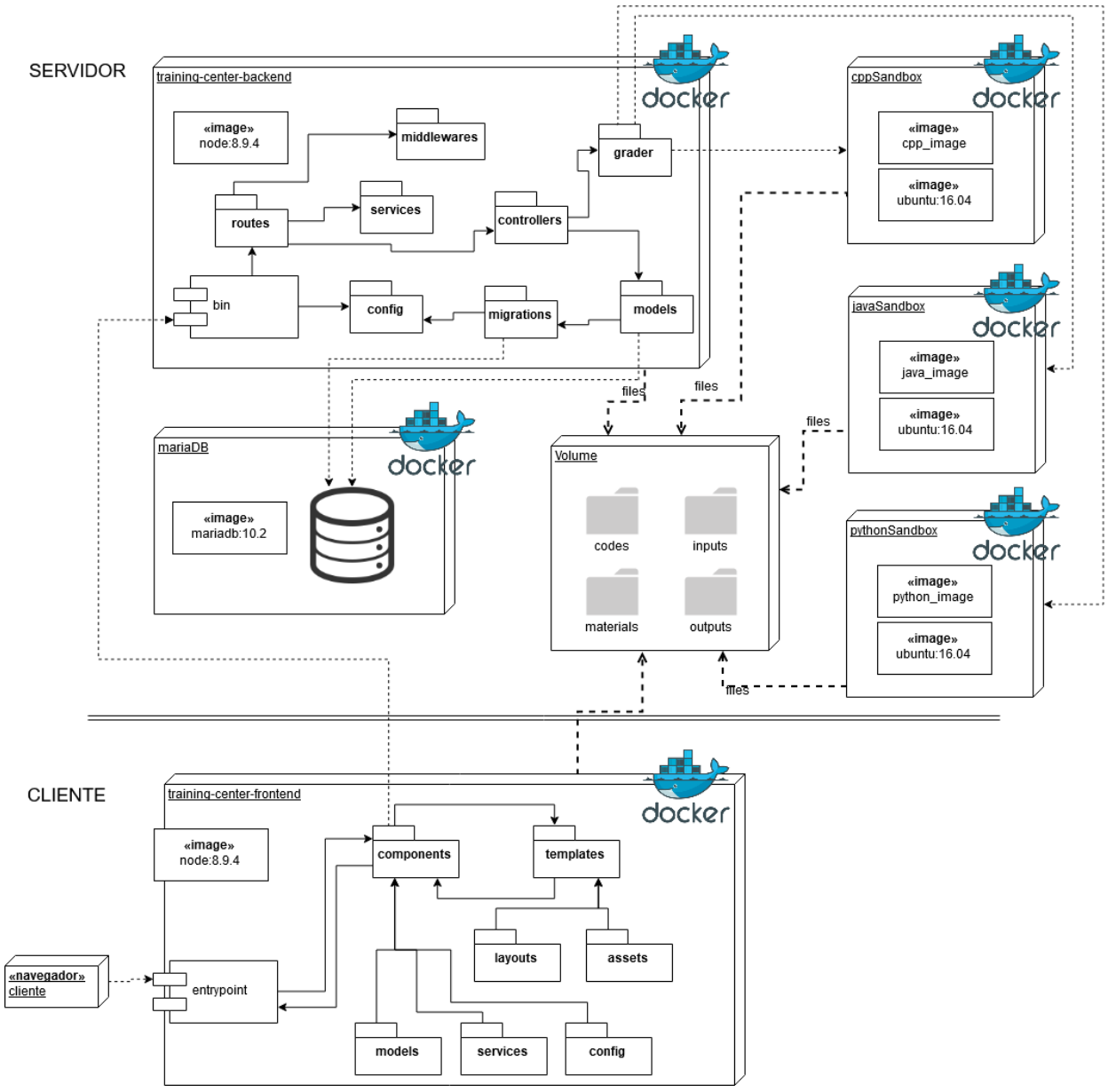


Figura 10. Diagrama arquitectural..

La plataforma está diseñada bajo una arquitectura de cliente - servidor (dos capas). Cada una de las capas es diseñada como una aplicación completa y funcional, con interdependencia una de la otra. A continuación, se describe el funcionamiento de cada capa.

14.6. Capa servidor

El servidor es el encargado de procesar la información enviada desde el cliente y devolver las respuestas que sean necesarias. Esta capa contiene en su interior varios componentes interrelacionados entre sí:

training-center-backend: El núcleo central de este servidor es el "training-center-backend", aplicación que funciona como eje central de procesamiento obteniendo la información necesaria de los demás componentes y que mantiene una constante comunicación con la aplicación cliente. A continuación, se definen los subcomponentes de esta aplicación, para posteriormente definir su funcionamiento en conjunto:

- **Bin** (binario): Se encarga de recibir las peticiones desde el cliente y devolver las respuestas.
- **Router** (enrutador): Conjunto de utilidades para analizar las peticiones recibidas y "enrutarlas" hacia el componente encargado de su procesamiento.
- **middleware**: Son pequeños bloques de código que se ejecutan entre la petición que hace el usuario hasta que la petición llega al servidor, definiendo el flujo y restricciones de la aplicación.
- **Services** (servicios): Utilidades genéricas que pueden ser de ayuda en diferentes operaciones y se encapsulan para evitar redundancia de código.
- **Controllers** (controladores): Lógica de la aplicación.

- **Config** (Configuración): Archivos de configuración de la aplicación.
- **Models** (Modelos): Modelos de datos (en este punto se gestiona la conexión con la base de datos)
- **Migrations** (Migraciones): componente para la gestión de la base de datos.
- **Grader** (Calificador): componente encargado de comunicarse con los contenedores seguros para la ejecución de las soluciones de los estudiantes.

En conjunto, el funcionamiento del backend es el siguiente: “bin” se encuentra permanentemente escuchando peticiones desde el cliente. Al llegar una petición, esta es capturada por un enrutador (router) específico que la redirige al controlador en el que debe procesarse, utilizando los servicios (services) que sean necesarios, y en el proceso de redirección la solicitud cruza los filtros (middlewares) para definir entre otros aspectos, que el usuario esté autorizado. En caso de ser necesario, el controlador se comunica con los modelos (models), los cuales a través de migraciones (migrations) obtienen y actualizan la información almacenada en la base de datos. Durante todo el proceso, se toman en cuenta los parámetros establecidos globalmente en el archivo de configuración (config).

Un caso especial ocurre cuando la solicitud recibida es un ejercicio para ser calificado. En este caso, el controlador llama al calificador (grader), el cual se comunicará con el entorno seguro específico para el lenguaje de la solución que se encargará de su ejecución y calificación.

- **cppSandbox:** Contenedor Docker que funciona a modo de entorno seguro (sandbox) para ejecutar en entorno seguro las soluciones recibidas en lenguaje C++ de los estudiantes. Este contenedor cuenta con el compilador GNU G++ en su versión 5.4.0

(Este compilador se encuentra bajo licencia libre GPLv3). Recibe las peticiones desde el calificador y toma la solución y las entradas desde las carpetas compartidas en “volume”.

- **javaSandbox:** Contenedor Docker que funciona a modo de entorno seguro para ejecutar en entorno seguro las soluciones recibidas en lenguaje java de los estudiantes. Este contenedor cuenta con el compilador incluido en el JDK (Java Development Kit) en su versión 8 (Compartido bajo licencia libre Oracle Binary Code License). Recibe las peticiones desde el calificador y toma la solución y las entradas desde las carpetas compartidas en “volume”.
- **pythonSandbox:** Contenedor Docker que funciona a modo de entorno seguro para ejecutar en entorno seguro las soluciones recibidas en lenguaje python de los estudiantes. Este contenedor cuenta con el compilador python en su versión 3 (Compartido bajo licencia libre GPL). Recibe las peticiones desde el calificador y toma la solución y las entradas desde las carpetas compartidas en “volume”.
- **mariaDB:** Gestor de base de datos. La plataforma cuenta con una base de datos que almacena toda la información no estática de la aplicación usando este gestor.
- **volume:** Es un espacio de almacenamiento compartido. Aquí se almacenan los archivos estáticos de la plataforma, y de allí son tomados los códigos, entradas y salidas por los contenedores para realizar las calificaciones.

14.7. Capa cliente

La capa del cliente es una aplicación web completa que se ejecuta en cualquier navegador web moderno. Cuenta con los siguientes componentes:

- **entrypoint (index.html)**: Punto de comunicación entre el navegador y la aplicación. Todas las solicitudes e interacciones de la aplicación son hechas a través de este componente.
- **Components (componentes):** Los componentes (también llamados módulos) definen la lógica de la aplicación en el cliente. Se comunican en tiempo real con las plantillas (templates).
- **Templates (plantillas):** Son unidades de diseño de interfaz de usuario conectados a los componentes para tener la información actualizada en tiempo real.
- **Layouts (diseños):** Los layouts definen diseños generales que pueden ser reutilizadas por diferentes plantillas.
- **Assets (componentes gráficos):** Imágenes, estilos, y demás componentes gráficos.
- **Models (modelos):** Los modelos permiten mantener una estructura de objetos en la aplicación cliente.
- **Services (servicios):** Servicios que permiten encapsular tareas repetitivas para evitar duplicaciones de código.
- **Config (configuración):** Configuraciones generales de la aplicación.

Estos componentes se relacionan de la siguiente manera: El usuario ingresa a la aplicación a través del punto de entrada (index.html), que inmediatamente lo redirige al componente solicitado (mediante un enrutamiento transparente realizado por las mismas configuraciones de los componentes). Cada componente define una lógica del negocio y tiene adjunto una plantilla con la cual realiza una conjunción de datos (data binding) que permite comunicación en tiempo real: Los cambios realizados en el componente luego de un procesamiento o de una conexión al

servidor se despliegan inmediatamente en la interfaz, mientras los cambios ingresados por el usuario a la plantilla se actualizan inmediatamente al componente.

Para esta labor el componente se basa en los modelos que definen el esqueleto de la aplicación, y los servicios que son utilidades transversales a la aplicación. Por su parte, las plantillas se ayudan de los diseños y los componentes gráficos para desplegar la información correctamente y de forma agradable al usuario.

Toda la ejecución sigue los parámetros establecidos en config.

Cabe destacar que la comunicación con el servidor ocurre en los componentes, aunque para una mayor modularidad y simplicidad, generalmente estos componentes se apoyan en servicios para realizar estas tareas.

14.8. Desarrollo de la plataforma

A continuación, la lista de tecnologías utilizadas en el desarrollo de esta plataforma:

Tabla 14. Tecnologías usadas en la plataforma

Nombre	Tipo	Descripción
Javascript	Lenguaje de programación	Lenguaje usado en la programación del frontend y backend de la aplicación
G++	Compilador	Compilador de código libre utilizado para compilar los códigos que los usuarios envían en C++.
Python (Compilador)	Compilador	Compilador de código libre utilizado para compilar los códigos que los usuarios envían en Python (Lenguaje)

JDK	Kit de desarrollo	Kit de herramientas de desarrollo de Java, que incluye entre otras cosas el compilador del lenguaje, el intérprete, y la máquina virtual (En la versión utilizada la máquina virtual se incluye en el JDK, otras distribuciones pueden manejarla por separado).
Docker	Contenedores	Contenedores para ejecutar los códigos en un ambiente aislado.
Node.js	Entorno de ejecución	Herramienta usada en el backend de la aplicación.
Express.js	Framework	Framework para el desarrollo del backend de la aplicación.
Sequelize	ORM	Herramienta para el mapeo de datos haciendo uso de la programación orientada a objetos.
Aurelia.js	Framework	Framework para la creación de Single Page Application. Utilizada en el frontend de la aplicación.
Socket.io	Librería	Librería para el manejo de eventos en tiempo real.

Tal como se definió en la metodología, la plataforma se basó en los siguientes módulos:

Tabla 15. Módulos de la plataforma

Módulo	**Descripción**
Inicialización y autenticación	Inicialización del esquema, diseño y estructura de la aplicación, y autenticación de los usuarios en la plataforma.

Materiales y guías de estudio	Creación, actualización y revisión de materiales y guías de estudio, y despliegue a los usuarios para su visualización.
Administración de la plataforma	Herramienta para la administración de usuarios, problemas y guías en la plataforma (solo usuarios administradores).
Estadísticas y perfiles de usuario	Módulo para desplegar estadísticas, rankings, perfiles de usuario, y resultados a los docentes/coach.
Problemas	Creación, edición y eliminación de problemas, visualización de los mismos y motor de calificación.
Modo guiado/syllabus	Módulo para la supervisión de los docentes/coach al trabajo de los estudiantes.
Competencias (maratones)	Módulo para la creación, administración y desarrollo de maratones de programación, así como para la ejecución de las mismas.

14.9. Diseño de interfaces de la plataforma

El diseño de la plataforma se centra en cumplir los estándares de diseño de la Universidad, tanto en sus colores como en sus componentes. A continuación, se presentan algunas capturas conceptuales del diseño de la plataforma.

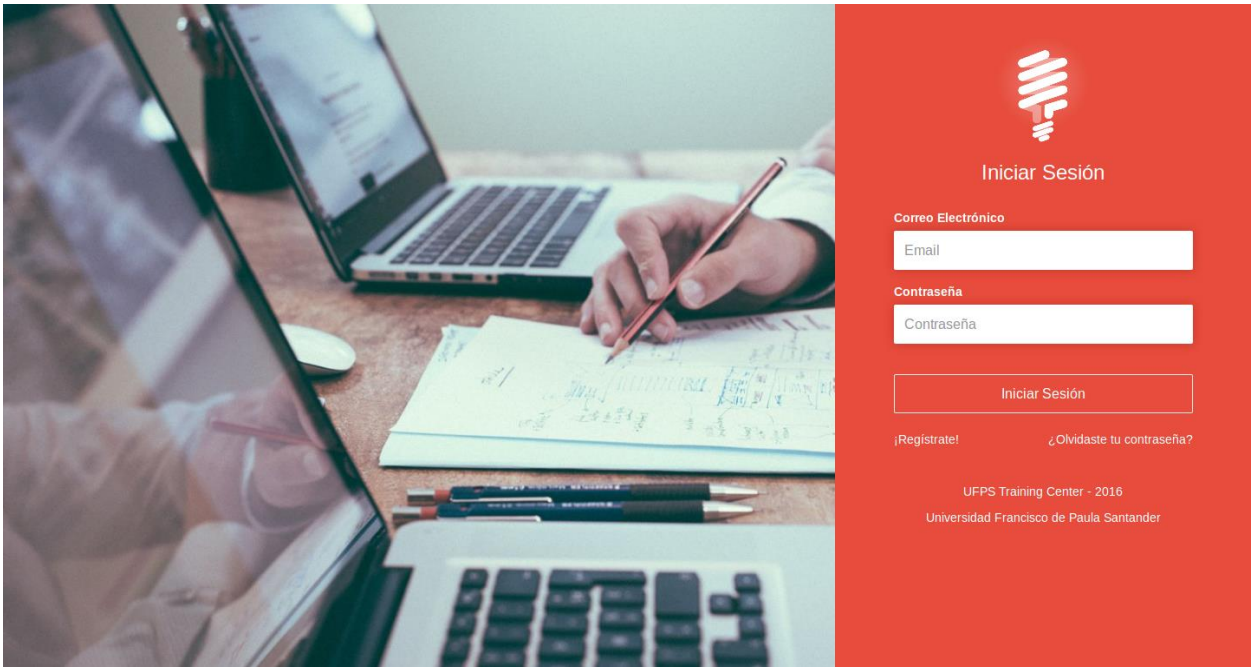


Figura 11. Interfaz de inicio de sesión

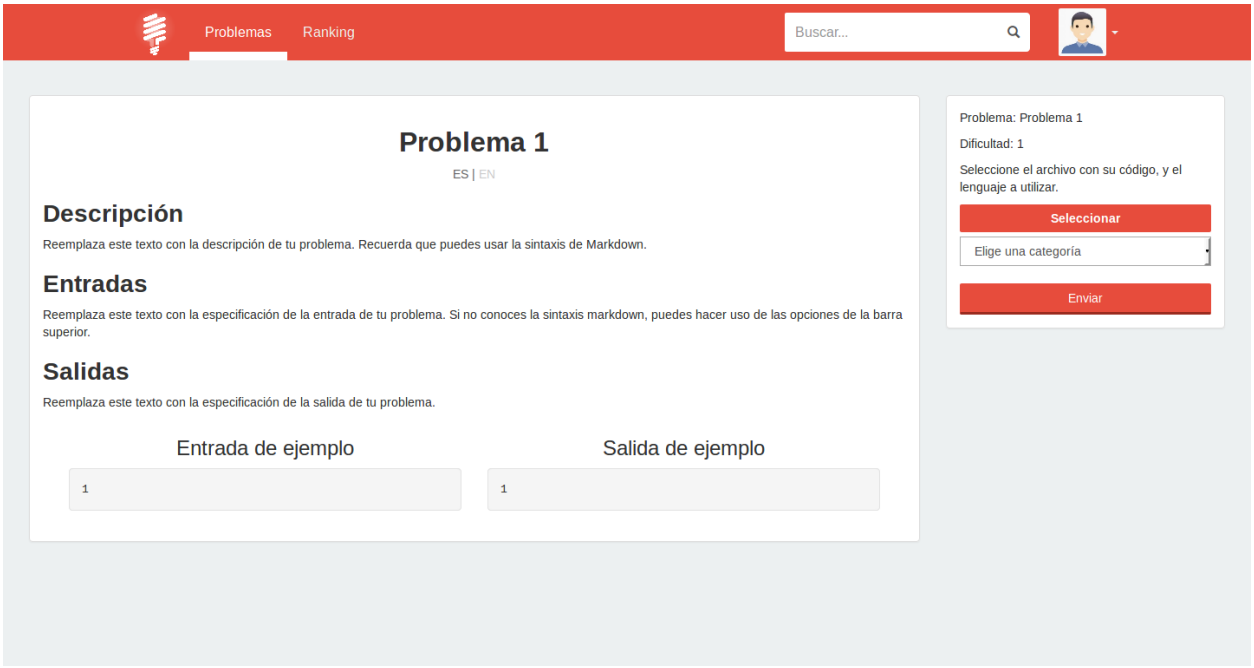


Figura 12. Interfaz de visualización de problemas.

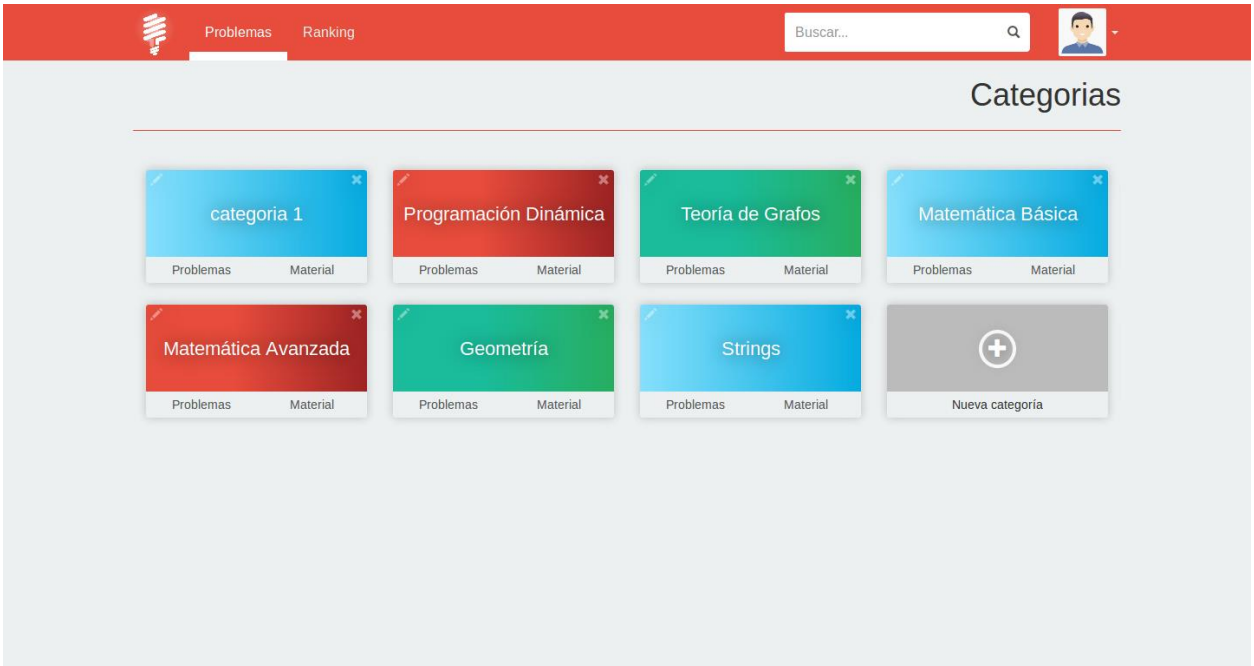


Figura 13. Interfaz de categorías disponibles.

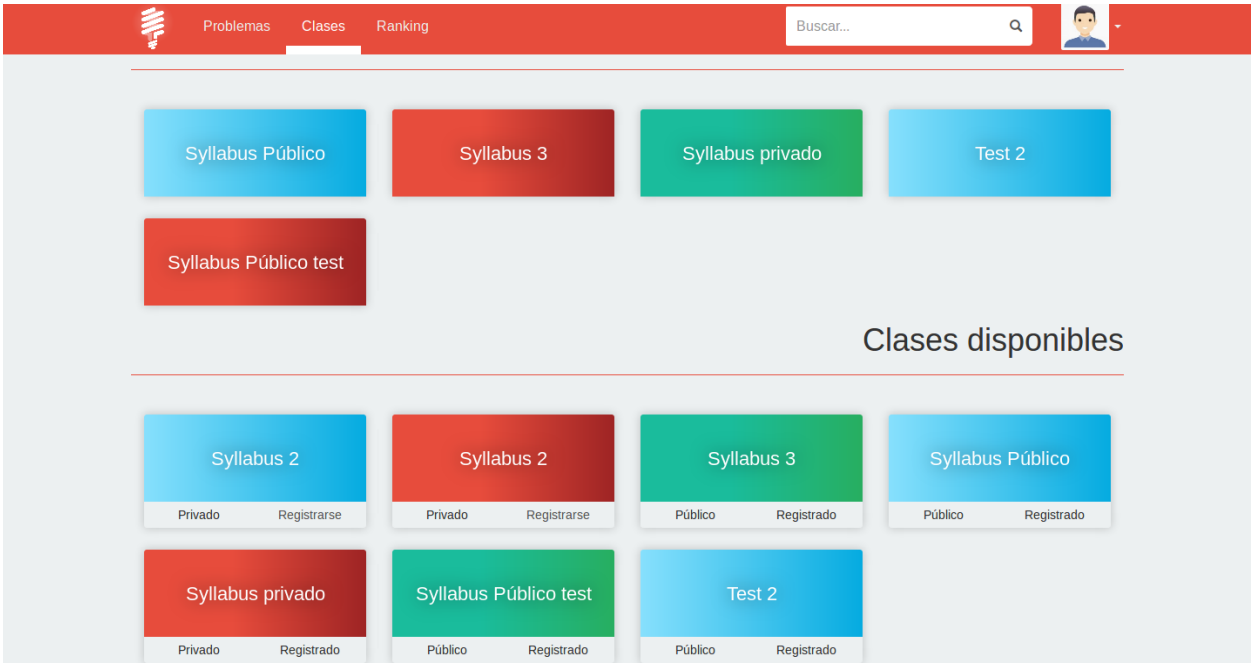


Figura 14. Interfaz de clases matriculadas y disponibles.

15. Integración del Marco de Trabajo

Al iniciar este capítulo se cuenta con una metodología para el trabajo de los estudiantes, unas temáticas de estudio definidas, un amplio material documental y de apoyo, herramientas de promoción para el grupo y una plataforma de software para entrenar. Este eje es el punto de unión entre todos estos elementos, y responde a la pregunta ¿Cómo integrar todas las herramientas creadas en un solo marco de trabajo útil para los estudiantes?

15.1. Temáticas del grupo de estudio

En primer lugar, se ha mencionado previamente en la metodología que las sesiones se realizan en base a un conjunto de temáticas de necesario conocimiento para participar en maratones de programación. Utilizando como referencia Competitive Programming (Halim & Halim, 2013), Competitive Programmer's Handbook (Laaksonen, 2017) y Programming Challenges (Skiena & Revilla, 2003), y la experiencia propia en las competencias en las cuales hemos participado desde el 2014, hemos definido un conjunto de temáticas. Este conjunto se clasifica de dos formas:

- Según su importancia en competencias de Programación: Se considera "Principal" aquel tema que aparece regularmente en competencias (por ejemplo, recorridos sobre grafos), o que es una base de programación importante para resolver problemas de programación (por ejemplo, entrada y salida estándar). Se considera "Opcional" aquel tema que aparece esporádicamente en maratones de programación, y que por tanto puede ser incluido en las sesiones si se cuenta con el tiempo suficiente.

- Según su dificultad: Se considera “básico” un tema que no necesita mayor fundamentación previa para ser entendido (por ejemplo, arreglos). Se considera “intermedio” un tema que, si bien no requiere conocimientos previos específicos, si necesitan de habilidad en programación para ser entendidos. Se considera “avanzado” un tema que requiere de conocimiento y práctica previa para poder ser interiorizado. Estos temas por lo general son enfocados a los estudiantes de semestres superiores.

La primera calificación sirve para definir la prioridad que se asignan a los temas en el grupo. Los temas principales deben ser cubiertos en su totalidad, mientras los temas opcionales se consideran según la cantidad de sesiones con las que se cuente en el semestre en curso. La segunda clasificación sirve para definir qué temas se estudian en las sesiones con estudiantes de semestres avanzados, y cuales con estudiantes de semestres básicos (Los temas de dificultad intermedio según decisión del grupo, pueden ser vistos en las sesiones básicas, avanzadas, o en ambas).

A continuación, se presenta la lista de temáticas que hasta el momento han sido analizadas para hacer parte del grupo de estudio en programación competitiva. Sin embargo, deben tenerse en cuenta varios aspectos:

En primer lugar, el mundo de la programación es demasiado extenso como para cubrirlo todo en una sola lista. Sin embargo, en determinados momentos (especialmente después de las maratones mundiales de programación) temáticas que eran poco conocidas cobran importancia, y por tanto esta lista es revisada con regularidad. Se recomienda a quienes dirigen el grupo que revisen la lista al iniciar cada semestre con el fin de incluir nuevas temáticas o eliminar algunas (la aparición de nuevos algoritmos puede convertir en obsoletos algunos de los aquí

especificados).

En segundo lugar, debe considerarse que el grupo no intenta ser un reemplazo de las clases propias del pensum de la carrera, sino un complemento. Por eso, nuestro enfoque es explicar a los estudiantes que es cada tema, como utilizarlo, y cuando utilizarlo. La fundamentación teórica no se centra en detalles, más allá de los necesarios para entender dicha temática. Un ejemplo de esto son las listas enlazadas. Esta temática es cubierta en la asignatura estructuras de datos, y en ella además de aprender que es y para qué sirve, se profundiza en los conceptos de nodo, de enlaces simples, dobles o circulares, y por lo general, se crea una implementación propia para aclarar cada detalle en profundidad. En el grupo en cambio, somos conscientes que, si bien es bueno conocer todos estos detalles, en una Maratón de Programación el tiempo es un limitante, y no sería óptimo crear una implementación propia cuando los lenguajes utilizados ya proveen una. Por tanto, el enfoque en el grupo se centra en explicar que es una lista enlazada, cuando es correcto utilizarla, y directamente como utilizar la implementación que los lenguajes proveen.

Tabla 16. Clasificación de Temas y categorías

Categoría	**Tema**	**Importancia en competencias**	**Dificultad**
Introducción a la programación competitiva	Introducción a competencias de programación	Principal	Básico
	Entrada/Salida estándar	Principal	Básico
	Instrucciones Condicionales	Principal	Básico
	Ciclos	Principal	Básico
	Recursión	Principal	Básico

	Arreglos	Principal	Básico
	Introducción a la complejidad algorítmica	Principal	Intermedio
Búsqueda y ordenamiento	Ordenamiento (Incluido en el lenguaje)	Principal	Básico
	Búsqueda binaria	Principal	Intermedio
Estructuras de datos lineales (incluidas por defecto en Java y C++)	Estructura tipo vector (ArrayList/Vector)	Principal	Básico
	Listas enlazadas (List, LinkedList)	Secundario	Básico
	Pilas (Stack)	Principal	Básico
	Colas (Queue)	Principal	Básico
	BitSets	Secundario	Intermedio
Estructuras de datos no lineales (incluidas por defecto en java y C++)	Treeset (Set en C++, TreeSet en java)	Principal	Intermedio
	Treemaps (Map en C++, TreeMap en java)	Principal	Intermedio
	Colas de Prioridad	Secundario	Intermedio
	Disjoint Set (Conjuntos disyuntos)	Secundario	Avanzado
Estructuras de datos no lineales (no incluidas por defecto en java y C++)	Arboles segmentados	Principal	Avanzado
	Arboles Binarios Indexados (Fenwick)	Secundario	Avanzado

Grafos	Teoría de Grafos (Introducción)	Principal	Básico
	Representación de grafos (Matriz de adyacencia, lista de adyacencia, lista de arcos)	Principal	Básico
	Grafos implícitos	Principal	Intermedio
	Árboles (n-arios y binarios)	Secundario	Intermedio
	Árboles binarios completos	Secundario	Intermedio
	Recorridos sobre grafos (Recorrido en Profundidad, recorrido en anchura)	Principal	Intermedio
	Componentes conexos en grafos	Secundario	Avanzado
	Arboles cobertores minimales (algoritmo de Kruskal y Algoritmo de Prim)	Principal	Avanzado
	Algoritmos de ruta más corta (Algoritmo de Dijkstra, Algoritmo de Floyd-Warshall)	Principal	Avanzado
	Camino Euleriano y Hamiltoniano	Secundario	Avanzado
	Puentes e ismos en grafos	Secundario	Avanzado
	Ordenamiento topológico en grafos	Secundario	Avanzado
	Flujo Mínimo - Máximo	Secundario	Avanzado
Paradigmas de diseño algorítmico	Algoritmos de Fuerza Bruta	Secundario	Intermedio
	Backtracking	Secundario	Avanzado

	Encuentro a medio camino (Meet in the middle)	Secundario	Avanzado
	Algoritmos Avaros	Principal	Intermedio
	Programación Dinámica	Principal	Avanzado
	Algoritmos “Divide y Vencerás”	Principal	Intermedio
Matemáticas	Manejo de números grandes	Principal	Básico
	Sucesiones conocidas (Fibonacci, tribonacci, triangulares, catalán, etc.)	Principal	Básico
	Aritmética Modular	Principal	Básico
	Números Primos (Algoritmo de Eratóstenes)	Principal	Básico
	Función Euler Totient	Secundario	Intermedio
	Factorización de enteros (método tradicional, Algoritmo de Pollard Rho)	Secundario	Avanzado
	Eliminación gaussiana	Secundario	Avanzado
	Problemas Josefianos	Secundario	Intermedio
Estadística	Combinatoria	Principal	Básico
	Probabilidad	Secundario	Básico
Strings	Manejo eficiente de strings (StringBuilder)	Principal	Básico
	Algoritmo de Knuth - Morris - Pratt	Principal	Avanzado

	(KMP)		
	Algoritmo Z	Secundario	Avanzado
	Suffix Tree	Secundario	Avanzado
	Suffix Trie	Secundario	Avanzado
	Suffix Array	Secundario	Avanzado
	String Hashing	Secundario	Avanzado
Geometría	Puntos	Secundario	Intermedio
	Rectas	Secundario	Intermedio
	Segmento de recta	Secundario	Intermedio
	Polígonos regulares e irregulares	Secundario	Intermedio
	Polígonos convexos y cóncavos	Secundario	Avanzado
	Envolvente Convexa	Secundario	Avanzado
	Distancias (Distancia Euclidiana, Distancia Manhattan, Distancia de punto a Línea)	Principal	Básico
	Sweep Line	Secundario	Avanzado

15.2. Pruebas del marco de trabajo en el grupo de estudio

El marco de trabajo se ha construido de forma incremental, y a medida que sus componentes alcanzan una nueva versión total o parcial se implementan en el grupo. Muchas de estas pruebas fueron satisfactorias, otras llevaron a un análisis retrospectivo para mejorar con el paso del

tiempo. A continuación, se resumirá la forma en la cual el proyecto fue implementando a través del tiempo hasta alcanzar su estado actual.

- **Semestre I de 2016:** Se realizan sesiones obteniendo retroalimentación sobre las necesidades de los estudiantes con el grupo de estudio.
- **Semestre II de 2016:** Se propone una metodología preliminar, que alterna clases magistrales (poco espacio para socialización) con competencias de RPC y CCPL de forma excluyente: Sólo se realiza una u otra actividad semanal, nunca las dos al tiempo. También se construye la primera versión de la lista de temas que se tratan en el grupo (Syllabus). En este semestre se participa en la maratón nacional de programación y se logra la clasificación de un solo equipo a la fase regional, por debajo de las expectativas (se había propuesto el objetivo de clasificar por lo menos dos equipos).

  El análisis retrospectivo realizado al finalizar el semestre nos indica que realizar las actividades de forma excluyente está afectando la dinámica del grupo: En primer lugar, el número de sesiones se vio muy reducido, por lo cual muchas temáticas no pudieron tratarse. En segundo lugar, llegaron a presentarse hasta 3 competencias seguidas, lo que significó una pérdida de ritmo en las sesiones que estuvieron detenidas por esas 3 semanas. Como si fuera poco, los estudiantes se mostraron confundidos sobre las semanas en las cuales había sesión, y en las cuales había competencias.

  Esta fue la primera versión del marco de trabajo funcional puesto en marcha, y el punto de partida para el marco actual.
- **Semestre I de 2017:** Los aprendizajes del análisis retrospectivo semestre anterior son la base para los cambios introducidos durante este semestre: Las sesiones y las competencias dejan de ser excluyentes: Se realizan sesiones semanales de entrenamiento

sin importar si hay o no competencias de RPC/CCPL durante esa semana. Se establecen medios de comunicación directos con los estudiantes (Grupo en Facebook, mencionado en el Capítulo de Promoción) para evitar la desinformación sobre los horarios de sesiones y competencias.

También en este semestre se definen los incentivos con su estructura actual. Hasta este momento, los estudiantes que obtenían el primer puesto en la maratón UFPS, y quienes habían presentado mejor rendimiento en el semestre eran elegidos para representar a la Universidad en la fase nacional. El criterio para definir el rendimiento en el semestre se basaba únicamente en los ejercicios resueltos en competencias RPC/CCPL y en sesiones. Para motivar la participación activa de los estudiantes en las diferentes facetas del grupo, se define un ranking más completo para elegir los beneficiados con la representación en Maratón Nacional de Programación, que tiene en cuenta todos los aspectos del grupo de forma global, y resulta exitoso. Si bien los resultados cuantitativos serán expuestos más adelante, cabe resaltar que durante este semestre la plataforma utilizada calificó como correctos más de 1000 soluciones enviadas por estudiantes del grupo.

En este semestre las sesiones se dinamizan. Hasta el momento el aprendizaje era repartido entre sesiones magistrales y aprendizaje basado en problemas. En este punto reemplazamos las sesiones magistrales por socializaciones basadas en aprendizaje colaborativo. Esto resulta benéfico, pues dado que el grupo es conformado solo por estudiantes, no siempre las sesiones magistrales resultaban lo suficientemente claras o atractivas para los demás integrantes (esto es fácilmente explicable, pues ninguno de los integrantes posee preparación académica ni experiencia en enseñanza). Por esto el aprendizaje colaborativo y las socializaciones otorgan una nueva dinámica más efectiva.

Durante este semestre se realizaba una sesión semanal con un tema predefinido en cada ocasión. Estos temas iban desde los más simples al inicio de semestre, hasta los más complejos al final de semestre. En el análisis retrospectivo pudimos observar como los estudiantes de semestres iniciales tuvieron problemas para adaptarse al grupo al llegar a temas intermedios y avanzados pues la curva de aprendizaje era demasiado elevada. Por otro lado, las temáticas dictadas durante el semestre fueron las mismas vistas en el semestre anterior, y para quienes llevan algún tiempo en el grupo, esto resultó en algunas ocasiones, redundante. Esto llevó a plantear una última mejora a la estructura de las sesiones para el siguiente semestre.

Esta fue la segunda versión funcional del marco de trabajo. Los resultados fueron mucho más satisfactorios, como se verá en la sección de resultados.

- **Semestre II de 2017:** Para el grupo es fundamental la presencia de estudiantes de semestres iniciales, quienes aún tengan varios años de estudio por realizar, de forma que puedan realizar una trayectoria larga en competencias. Por esto las mejoras en este semestre se centran en asegurar su permanencia, continuidad y adaptación al grupo. Por esto se decide expandir las sesiones, haciendo que cada una de ellas tenga dos secciones: Una con un tema básico, y una con un tema avanzado. De esta forma, los estudiantes según su experiencia y nivel eligen en cuál de las dos ubicarse (o hacerlo en las dos si lo desean).

  De esta misma ampliación en las sesiones surge la solución para el problema de la redundancia en los temas para los estudiantes que participan en el grupo por varios semestres. La sección básica contiene siempre los temas introductorios, siendo el punto de partida para los estudiantes nuevos, y la sección avanzada no tiene unos temas

totalmente definidos, sino una amplia colección de posibles temáticas (previamente listadas, y que además puede seguir expandiéndose cada semestre). Por tanto, en cada semestre al iniciar el grupo entre todos los estudiantes se define un syllabus “avanzado” a tratar durante el semestre, conteniendo algunas de las cosas vistas antes para fortalecer, y algunos temas nuevos para generar variaciones y evitar la monotonía.

Durante este semestre se estandarizan también los materiales, se añade contenido audiovisual de ayuda, y se completa el manual de competencias a su versión actual para la utilización de los estudiantes.

- **Actualidad:** Durante el semestre anterior se alcanzó una versión estable del marco de trabajo después de múltiples mejoras y de un continuo aprendizaje en su desarrollo. Sin embargo, durante todos los semestres previamente descritos, la dispersión de la información siempre fue un punto complicado: Cada uno de los materiales usados (audiovisuales, digitales, de entrenamiento) se encuentra en un sitio diferente, y los problemas propuestos dependen siempre de plataformas externas, y aunque se utilizaron múltiples plataformas en el proceso como se describió anteriormente, no se logró llenar las expectativas con ninguna de ellas.

  Por eso se integra al marco de trabajo actual la plataforma “UFPS Training Center” donde el marco de trabajo tiene su punto de encuentro y los estudiantes un centro de experiencia de aprendizaje.

  La metodología, las sesiones, y el material no presentan cambios, pero ahora la herramienta software utilizada centraliza todo el proceso para hacer más fácil de guiar, dirigir y supervisar.

16. Maratón de Programación UFPS

En el año 2014 cuando la Universidad volvió a participar en Maratones de Programación a nivel nacional, se realizó la maratón UFPS 2014 con el apoyo de la RPC para realizar la elección de los equipos que representarían a la Universidad en la competencia nacional..

Para el 2015 empezaba a tomar forma el grupo de estudio, y se realizó la Maratón de Programación UFPS 2015. En esa ocasión participaron 12 equipos de la UFPS, junto a 1 equipo de la Universidad Simón Bolívar y 2 equipos de la Universidad de Pamplona. Sin embargo, al igual que el año anterior, no se contó con una competencia propia: En realidad se aprovechó la competencia 05 de RPC (con la respectiva autorización de la red) y de esa forma se pudo elegir a los ganadores de nuestra competencia.

Ya en el 2016 con la construcción en primer lugar del anteproyecto y luego del proyecto, se planeó una Maratón Interna anual, con ejercicios propios escritos por integrantes de la UFPS (docentes y alumnos que no participen en la competencia), y colaboradores externos. Esta competencia se integra en la sección de incentivos: Se ha descrito previamente como las actividades realizadas en el grupo dan un puntaje, que genera un ranking y al final los estudiantes más destacados representan a la Universidad en la Maratón Nacional. Pero como se pueden enviar 3 equipos por facultad, decidimos utilizar este mecanismo para otorgar dos cupos, y el tercero se entrega a aquel equipo que gane la maratón de Programación UFPS. Esto ha resultado un éxito, pues los estudiantes se preparan durante todo el semestre para participar en la competencia y cuentan con la posibilidad de competir directamente por un cupo en fase nacional.

Una vez definida la integración de la competencia dentro del marco, nos enfocamos en la organización de la Maratón como tal. Dicha Maratón se organiza un sábado cada año en las últimas semanas del primer semestre. Luego de un acuerdo con la Red de Programación

Competitiva, nos han permitido utilizar su infraestructura para llevar a cabo la competencia con nuestros propios ejercicios allí. Este acuerdo tiene una característica especial: La competencia se realiza de forma interna en la UFPS, pero también se encuentra activa para todos los sitios en los cuales la Red tiene presencia. De esta forma la Maratón de Programación UFPS ha llegado en los últimos dos años a diferentes países de Latinoamérica, otorgando reconocimiento a nuestra Universidad y grupo de estudio.

Figura 15. Cartel promocional Maratón de Programación UFPS 2016

En la Maratón de Programación UFPS 2016 se contó con 12 problemas inéditos, y la participación de 12 equipos de la Universidad (36 estudiantes). Gracias a la red de programación competitiva esta competencia llegó a 183 equipos (549 personas) distribuidos en 15 sedes presenciales de 5 países (Colombia, Bolivia, Chile, Cuba y México), sin mencionar los estudiantes en modalidad virtual, que podían participar desde cualquier sitio del mundo. El ganador de la competencia interna fue el equipo compuesto por los estudiantes David Tolosa, Juan Manuel Huertas y Denis González. El ganador a través de RPC fue el equipo “Eeny Meeny miny moe” de la Universidad Central de Venezuela.

Figura 16. Cartel promocional Maratón de Programación UFPS 2017

Para el 2017 la maratón UFPS tuvo una variación: Los estudiantes participaron en un ambiente privado (diferente al del resto de Latinoamérica). Para esta competencia se escribieron en total 17 ejercicios diferentes. Tanto en la versión interna de la UFPS como en la versión abierta de RPC se presentaron 12 ejercicios, teniendo ambas competencias 7 ejercicios en común, mientras los otros 5 eran diferentes en cada una. Con esto se logró escribir ejercicios que evaluaran temas específicos para los estudiantes UFPS según lo visto en el grupo de estudio. En la Universidad participaron 18 equipos, 54 estudiantes. En Latinoamérica a través de la Red de Programación Competitiva participaron 209 equipos, para un total de 627 participantes. En la competencia interna el ganador fue el equipo compuesto por Crisel Ayala, Angie González y Brayan Arias. El ganador a través de RPC fue el equipo “Standard Move” de la Universidad Nacional de Córdoba (UNC) Argentina.

17. Resultados

El objetivo general de este proyecto fue “desarrollar e implementar un marco de trabajo para formalizar la metodología de trabajo del grupo de estudio de programación competitiva de la Universidad Francisco de Paula Santander”. Como se ha visto a través de los capítulos anteriores, este marco de trabajo efectivamente ha sido desarrollado e implementado en el grupo de estudio, y se propone como el eje a través del cual los estudiantes y docentes que con el paso del tiempo se encuentren frente al grupo guíen su trabajo.

En el desarrollo se realizaron múltiples mediciones de diferentes resultados que en conjunto permiten ver un balance general de proyecto. Diferentes aspectos fueron medidos y analizados para poder constatar si realmente los objetivos específicos fueron cumplidos.

17.1. Fomento en la participación de los estudiantes

El primer objetivo específico fue “fomentar la participación de los estudiantes en las competencias de programación para continuar representando a la universidad a nivel nacional e internacional”. Con este fin se establecieron las estrategias de promoción previamente descritas, y para analizar el impacto que tuvieron estas estrategias en el grupo, se midió la participación, semestre a semestre. En este caso, se persiguen dos fines en el mismo objetivo: En primer lugar, contar con un número creciente de estudiantes del cual formar un grupo lo suficientemente competitivo y, en segundo lugar, mantener un alto número de estudiantes activos a lo largo de cada semestre.

Con este objetivo en mente, realizamos mediciones a través de la asistencia de los estudiantes. Consideramos que un estudiante es activo, si asistió por lo menos a la mitad de las actividades realizadas durante el semestre en curso.

Tabla 17. Número de estudiantes en el grupo de estudio

Año	**Semestre**	**Estudiantes totales**	**Estudiantes activos**	**Porcentaje de estudiantes activos**
2015	I	19	8	42.10%
	II	30	7	23.33%
2016	I*	36	12	33.33%
	II**	34	13	38.23%
2017	I	37	24	64.86%
	II	41	21	51.21%

* Durante este semestre se presentó el anteproyecto. Los datos previos a este son de referencia para tener un punto de comparación con los resultados obtenidos una vez el proyecto inició su ejecución.

** Desde este semestre los resultados hacen parte de medición del proyecto actual.

En cada una de las actividades realizadas en el grupo, se repasó la asistencia tanto con fines de monitoreo del grupo, como de control del semillero SILUX. Estas cifras son extraídas directamente de las asistencias tomadas en el grupo, las cuales se han entregado como parte del control documental del semillero al final de cada semestre.

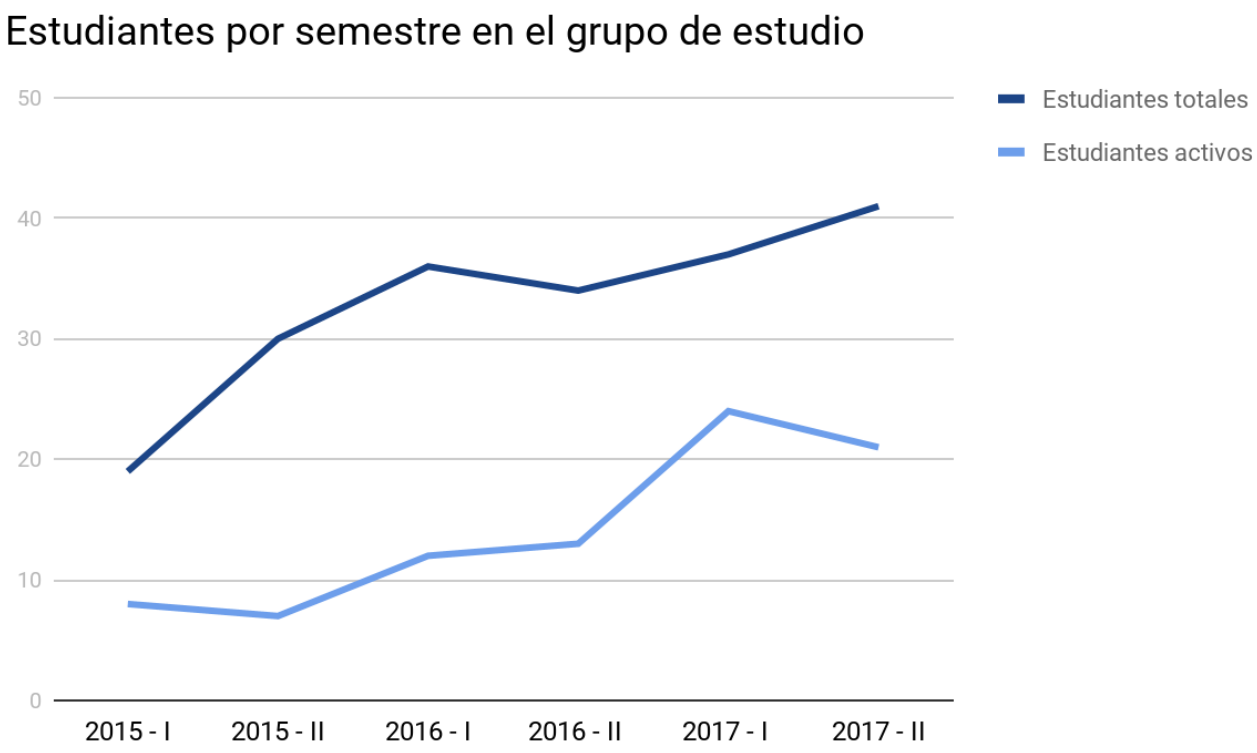


Figura 17. Gráfico de estudiantes activos y totales en el grupo de estudio en los últimos 6 semestres

Existen dos comportamientos a tener en cuenta al analizar este gráfico: El primero, es que la cifra de estudiantes que cada semestre hacen parte del grupo de estudio mantiene un crecimiento casi ininterrumpido, que ha ido mejorando desde la implementación de los mecanismos de promoción establecidos. En segundo lugar, puede observarse un comportamiento particular en el número de estudiantes activos por semestre; si bien el número es creciente a través del tiempo, los aumentos más considerables se dan siempre en los primeros semestres de cada año. Esto se debe según nuestro análisis, a los incentivos: Por calendario académico, resulta obligatorio definir los estudiantes que representarán a la Universidad en la Maratón Nacional de

Programación a más tardar, antes de salir a vacaciones de mitad de año. Por tanto, los estudiantes tienen mayores incentivos en los primeros semestres, y de ahí este comportamiento particular.

También es notable que el grupo tuviera una caída en la cantidad de estudiantes participantes al iniciar el segundo semestre de 2016, justo cuando se colocaba en pruebas por primera vez el marco de trabajo. A raíz de esto se mejoraron los canales de promoción y comunicación para iniciar de nuevo el crecimiento desde el siguiente semestre.

Al final de cada semestre, hemos realizado una encuesta con los estudiantes que hacen parte del grupo para analizar su nivel de satisfacción con el grupo. Esta encuesta es genérica, y se recomienda a quienes tomen el liderazgo del grupo en el futuro seguir realizando esta encuesta como un buen punto de retroalimentación para conocer las áreas donde se está fallando y poder mejorar a partir de estas.

Con esta encuesta corta se busca analizar la satisfacción de los estudiantes con respecto a cinco puntos específicos: Los temas vistos, las explicaciones recibidas, los materiales de apoyo, los ejercicios propuestos y la metodología trabajada.

Para realizar esta encuesta se ha utilizado Google Forms, de forma que los estudiantes pudieran responder a través de internet. Esta encuesta fue enviada cada semestre a todos los estudiantes que hicieron parte del grupo (tomando muestra = población por el número limitado de estudiantes y por la facilidad de analizarlos a todos en conjunto), logrando la participación de un buen número de ellos en cada una de las ocasiones. Se solicitó a los estudiantes que evaluaran cada una de las características previamente descritas con un número entre 1 y 5, donde 1 es completamente insatisfecho y 5 completamente satisfecho. Adicional, los estudiantes tienen la posibilidad de añadir sus comentarios libremente al final de la encuesta. Los promedios de los resultados en cada caso fueron los siguientes.

Tabla 18. Resultados de la encuesta de satisfacción realizada a los estudiantes

Característica evaluada	**2016 - II**	**2017 - I**	**2017 - II**
Temas vistos en el grupo de estudio.	4,18	4,3	4,68
Claridad en las explicaciones y socializaciones.	3,82	4,3	4,75
Utilidad del material de apoyo.	4,45	4,8	4,93
Calidad de los ejercicios propuestos.	4,82	4,65	4,87
Calidad de la metodología trabajada.	4,27	4,55	4,81
Total de encuestados:	11	20	16

A través de los semestres evaluados la satisfacción de los estudiantes ha variado de la siguiente forma:

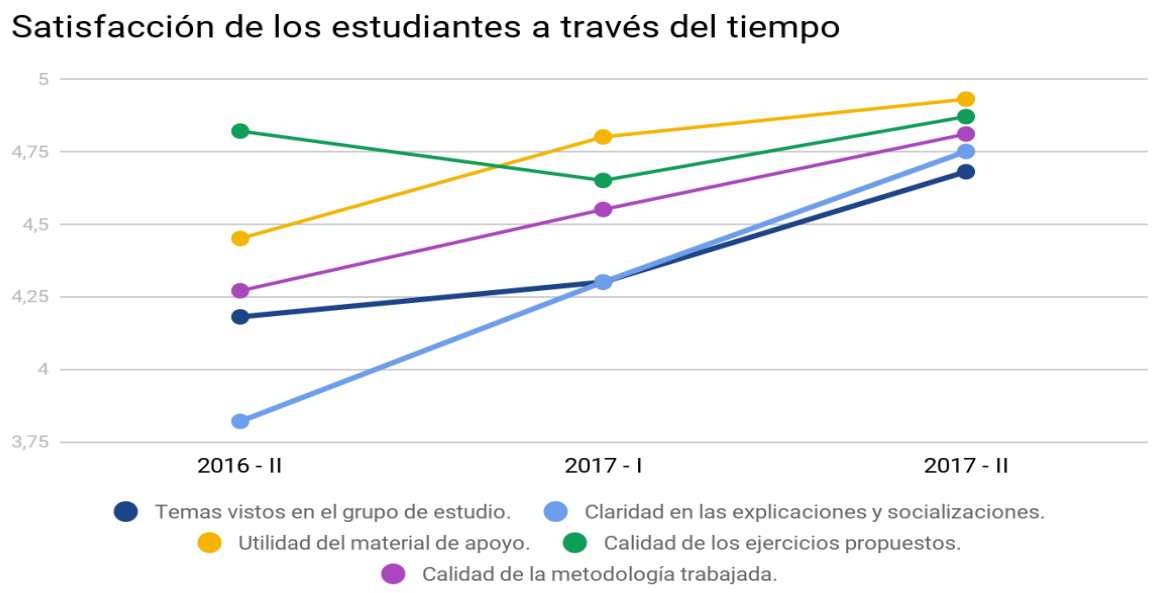


Figura 18. Satisfacción de los estudiantes a través del tiempo

Durante el primer semestre de puesta en práctica del marco de trabajo de forma preliminar, segundo semestre de 2016, la satisfacción expresada por los estudiantes en términos generales fue buena. Los ejercicios propuestos en el grupo fueron ampliamente aprobados por los estudiantes, pero sin embargo la principal falencia se encontró en la claridad de las explicaciones (en aquel momento solo se hacían explicaciones con poco espacio para socializaciones, y por tanto en la encuesta se preguntaba solo por la claridad en las explicaciones. El texto de la encuesta fue ajustado en el siguiente semestre para incluir las socializaciones en la pregunta). Por tanto, un propósito para el siguiente semestre fue hacer explicaciones más claras y más participativas, así como apoyarse más en el material de apoyo, dado que este había sido calificado con muy buenos resultados por los estudiantes. Como se ha dicho previamente, la encuesta al final permitía agregar comentarios aparte de las preguntas si el estudiante así lo deseaba, y para este semestre las opiniones recibidas se centraban en 3 aspectos: Informar mejor cuando habían competencias de RPC y CCPL los días sábados, ser más organizados en la forma de llevar a cabo las sesiones, y no imponer un ritmo de trabajo tan fuerte para estudiantes de semestres inferiores (las respuestas exactas dadas por los estudiantes así como los resultados tabulados de la encuesta se adjuntan como anexos). Estas 3 recomendaciones fueron también tenidas en cuenta, centrando esfuerzos en mejorar nuestra comunicación, organización y material para estudiantes de primeros semestres.

Para el primer semestre de 2017, la utilidad del material de apoyo se incrementó, posiblemente debido al énfasis que se hizo a los estudiantes a utilizar el manual de competencias (notebook). Hasta el semestre anterior, el manual estaba relegado solo a su uso en las competencias oficiales nacional y regional, pero en este semestre se motiva su utilización en las sesiones y en competencias de RPC/CCPL, siendo bien recibido. Por el contrario, la satisfacción

de los estudiantes con respecto a los ejercicios cayó un poco. Durante este semestre cambiamos la plataforma utilizada, de UVA Online Judge a URI Online Judge que, aunque tiene características más modernas y completas que la anterior, posee una colección de ejercicios mucho más limitada que pudo influir en este punto. Si bien la claridad en las explicaciones y socializaciones sigue siendo el punto más bajo en las calificaciones, tuvo un aumento notable desde 3.82 hasta 4.3, luego de añadir más participación y apoyo en el material. Una de las sugerencias dadas en este semestre como retroalimentación es explicar los ejercicios de las competencias de los sábados después de las competencias para seguir mejorando, y esta sugerencia es tomada en cuenta e implementada para el siguiente semestre.

Para el segundo semestre de 2017 la calificación de todos los indicadores subió con respecto al semestre anterior. En este semestre por primera vez la claridad en las explicaciones deja de ser el ítem con la calificación más baja, puesto que ocupa ahora los temas vistos en el grupo. Si bien esto no resulta preocupante, pues a pesar de ser el tema con la calificación más baja cuenta con 4.68 sobre 5, una calificación muy satisfactoria, los temas se han ampliado después de una revisión al material teórico existente para seguir fortaleciendo este punto.

17.2. Resultados documentales

No entraremos en detalles en esta sección dado que previamente se han descrito los materiales documentales y audiovisuales que hacen parte de este proyecto, su proceso de producción y su integración dentro del marco de trabajo. Por tanto, simplemente listamos los resultados obtenidos al respecto, para cumplir con el objetivo de “construir un banco documental

que abarque todos los contenidos que se trabajan durante el semestre y que sirva para evidenciar las actividades realizadas". Los productos obtenidos en este punto son:

- Un syllabus de temáticas de programación competitiva.
- 20 presentaciones digitales en formato web con explicaciones de diferentes temáticas que pueden ser accedidas a través de cualquier dispositivo con conexión a internet.
- 2 editoriales explicativas sobre las maratones UFPS 2016 y UFPS 2017.
- 2 manuales de competencia (Notebooks) En java y C++, cada uno de los cuales se encuentra por su cuarta versión actualmente.

Junto a esto, existen algunos materiales que no fueron planteados originalmente como parte de este proyecto, pero que sin embargo se han integrado a él para complementarlo:

- 20 videos explicativos de diferentes temáticas de maratones de programación (más de 160 minutos de video). Estos videos hacen parte originalmente de un proyecto realizado entre el Ingeniero Milton Vera, director del semillero SILUX, Crisel Ayala, estudiante del grupo de estudio y los dos autores de este proyecto, para fortalecer el entrenamiento fuera de aula, y facilitar el proceso de enseñanza cuando existe cambio en quienes dirigen el grupo. Este contenido se encuentra alojado en YouTube, es accesible a través del "UFPS Training Center".
- "C++ para maratonistas Java", una iniciativa planteada por los autores de este proyecto y llevada a cabo de forma independiente, es un libro digital corto explicando C++ a través de ejemplos comparativos con Java, y enfocado en competencias de programación. Es diseñada especialmente para introducir en C++ a los estudiantes del grupo que nunca lo han utilizado pero que manejan java correctamente.

17.3. Resultados en software

Otro objetivo específico fue “desarrollar una plataforma web de entrenamiento para realizar un seguimiento del trabajo realizado dentro del grupo de estudio” y esto se ha logrado creando el UFPS Training Center, a través del cual los estudiantes llevarán a cabo su proceso de aprendizaje mientras los líderes/coach del grupo realizan seguimiento. No se entrará en detalle en esta sección, pues la construcción del software ha sido detallada previamente.

17.4. Resultados en la implementación general del proyecto

Finalmente, entre nuestros objetivos iniciales se planteó “diseñar un plan de trabajo para integrar todos los componentes desarrollados de manera tal que sea replicable en el futuro”. Todos los componentes han sido desarrollados, y puestos en práctica a lo largo de los semestres anteriores. En el cambio de cada semestre se realizaron cambios, pero siguió siendo replicable. Para el siguiente semestre los líderes del grupo cambiarán, pero quienes tomarán el relevo ya han adelantado estudios con el marco de trabajo y continuarán realizando su implementación.

Con todo esto en mente, el nuevo punto de análisis se centra en los resultados. Después de todo, el grupo de estudio existe para mejorar los resultados en maratones de programación a nivel oficial de los estudiantes que representan a la Universidad en los diferentes eventos en los cuales tiene representación.

Analizaremos en primer lugar la posición que logró la Universidad (el equipo con mejor de la Universidad en cada caso) en las diferentes competencias de RPC desde el inicio del segundo semestre de 2016 (inicio de la construcción del marco de trabajo) hasta la actualidad. Realizaremos esta comparación solo con los equipos colombianos. Dado que no existe un

conjunto fijo de países participantes, es variante a lo largo de las competencias y por tanto analizar los datos entre todos los países puede resultar sesgado (Por ejemplo, en algunas ocasiones las competencias RPC son clasificatorias oficiales en México, y en estas competencias la cantidad de equipos mexicanos puede llegar hasta los 500, sesgando cualquier estadística que puede obtenerse al respecto).

Tampoco el número de equipos participantes colombianos es un número fijo. En el espacio analizado la competencia con menor participación contó con 35 equipos, mientras la de mayor participación contó con 130. Por tanto, analizar únicamente la posición tampoco resulta muy eficaz, pues por nombrar un ejemplo, un puesto 30 sería un resultado muy bajo en el primer caso, pero medio/alto en el segundo caso. Por tanto, para hacer un comparativo efectivo tomamos como referencia el esquema que usa el ICFES Saber PRO para la evaluación por medio de percentiles (ICFES, 2017). En esta prueba, además del puntaje cuantitativo que recibe el estudiante, se le indica un número de percentil entre 1 y 100 representando su posición con respecto a los demás estudiantes que presentaron la prueba. Por ejemplo, si un estudiante se encuentra en percentil 72, implica que, si distribuimos la totalidad de estudiantes en 100 grupos según sus resultados, siendo el grupo 1 el de peores resultados y el grupo 100 el de mejores, el estudiante en cuestión estaría en el grupo 72, con mejores resultados que los 71 anteriores, y peor resultado que los 18 restantes.

Así pues, podemos tener un reporte claro que, del desempeño en cada competencia, pero surge un nuevo inconveniente: Dividir el conjunto en 100 percentiles es efectivo para el ICFES pues son miles de estudiantes, pero no en competencias donde en muchos casos no llegamos a 100 equipos. En una competencia con 40 equipos, cada equipo ocuparía más de dos percentiles. Por tanto, seguimos su esquema, fórmula y modo de conteo, pero en lugar de percentiles usamos

deciles, que nos dividen el conjunto en 10 “grupos significativos”, siendo mucho más ajustado para el caso.

Analizando el mejor puesto alcanzado por un equipo UFPS en competencia, tenemos entonces:

Tabla 19. Resultados UFPS en RPC

Competencia	**Mejor Puesto equipo UFPS en Colombia**	**Cantidad Equipos Colombianos**	**Decil**
RPC 06 / 2016	11	48	8
RPC 07 / 2016	5	35	9
RPC 08 / 2016	8	45	9
RPC 09 / 2016	6	62	10
RPC 10 / 2016	8	44	9
RPC 11 / 2016	6	77	10
RPC 13 / 2016	11	87	9
RPC 14 / 2016	12	77	9
RPC 01 / 2017	13	90	9
RPC 02 / 2017	15	113	9
RPC 03 / 2017	10	133	10
RPC 04 / 2017	12	130	10
RPC 05 / 2017	5	62	10
RPC 07 / 2017	7	54	9
RPC 08 / 2017	6	66	10

RPC 09 / 2017	8	96	10
RPC 10 / 2017	9	98	10
RPC 11 / 2017	10	107	10
RPC 12 / 2017	11	107	9
RPC 13 / 2017	9	78	9
RPC 14 / 2017	12	83	9

Para un análisis más claro de estos datos, podemos analizar el siguiente gráfico.

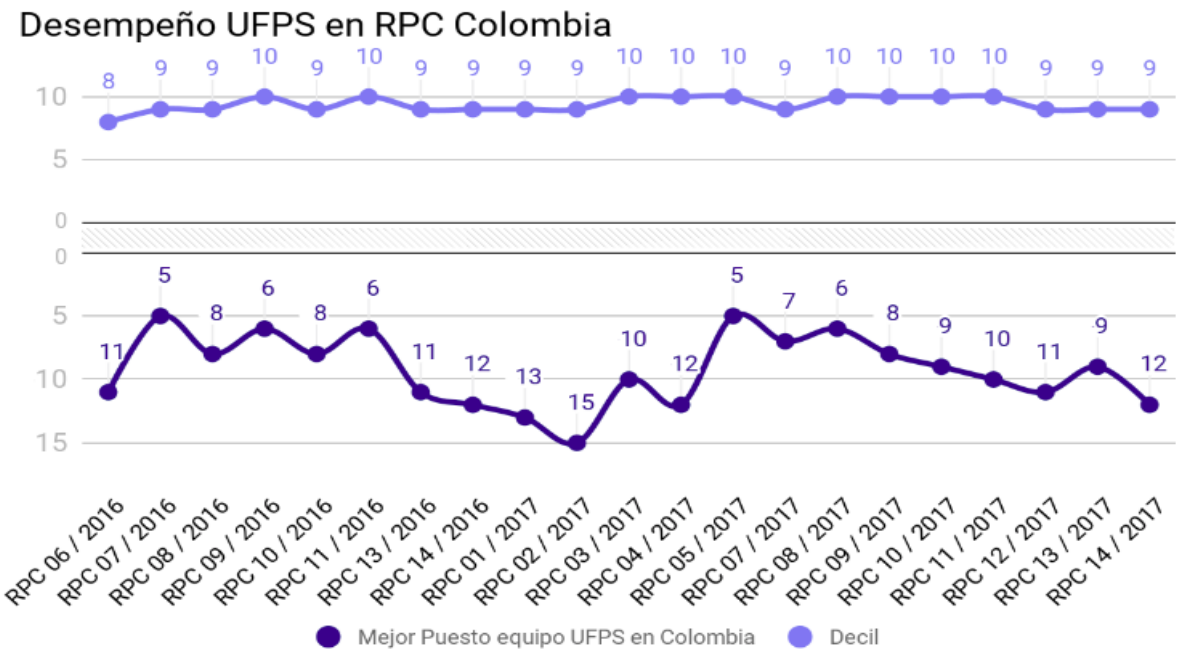


Figura 19. Resultados UFPS en RPC

De estos resultados hay varias cosas para destacar: En primer lugar, desde el inicio del marco de trabajo, la UFPS ha tenido al menos un equipo en las primeras 15 posiciones en Colombia en cada competencia RPC, sin importar cuantos equipos había en competencia. Mejor

aún, salvo en la primera competencia cuando el marco aún estaba en fase muy temprana, se logró mantener siempre al mejor equipo de la Universidad en los dos deciles superiores. Es de resaltar también la cantidad de veces que se alcanzó el percentil 10 entre el RPC 10 y el RPC 11 de 2017, coincidiendo justamente con el momento en que el grupo de estudio solidifica su marco de trabajo.

Realizando el mismo análisis con los datos obtenidos de las competencias de la Liga Colombiana de Programación Competitiva (CCPL) tenemos que:

Tabla 20. Resultados UFPS en CCPL

Competencia	**Mejor Puesto equipo UFPS**	**Cantidad Equipos Participantes**	**Decil**
CCPL 07 / 2016	7	38	9
CCPL 08 / 2016	5	35	9
CCPL 09 / 2016	7	57	9
CCPL 11 / 2016	13	42	7
CCPL 01 / 2017	5	78	10
CCPL 03 / 2017	7	87	10
CCPL 04 / 2017	9	77	9
CCPL 05 / 2017	4	72	10
CCPL 06 / 2017	2	58	10
CCPL 08 / 2017	5	70	10
CCPL 09 / 2017	11	54	8
CCPL 11 / 2017	7	52	9

CCPL 12 / 2017	5	46	9

Resultados de los cuales extraemos la siguiente información de forma gráfica:

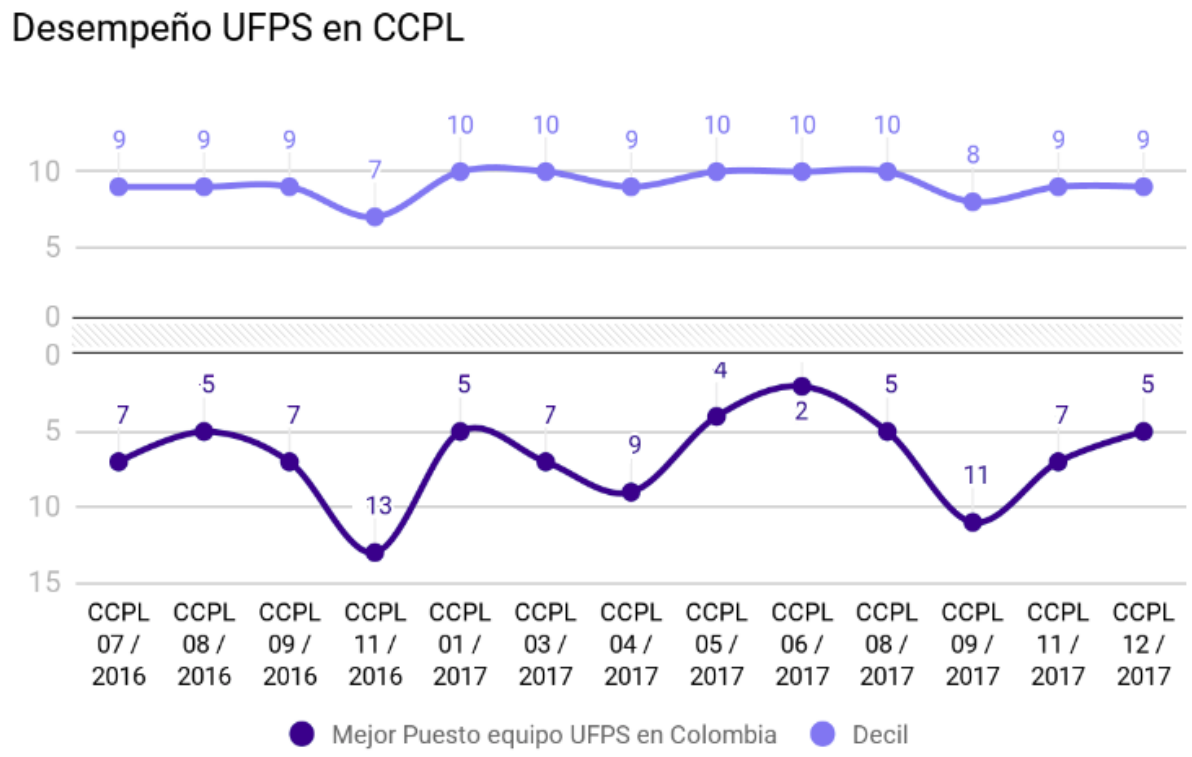


Figura 20. Resultados UFPS en CCPL

En CCPL contamos con menos competencias, pues no ha sido posible participar en todas (en la mayoría debido a cruce de horarios con RPC). Sin embargo, los resultados de nuevo han sido satisfactorios: Desde el inicio de la implementación del marco de trabajo, se ha mantenido un decil alto, la mayoría de veces entre 9 y 10 (con solo dos ocasiones estando por debajo de este margen, con 7 y 8).

Es además de destacar que la UFPS empieza a obtener reconocimiento por sus resultados en CCPL a nivel nacional, alcanzando en el tiempo medido cuatro quintos puestos, un cuarto puesto, e incluso un segundo puesto.

Con todo este entrenamiento realizado los estudiantes van a la Maratón Nacional de Programación. En esta competencia realizada una vez al año se pone a prueba todo el avance durante el año previo, y es el principal punto de observación para ver nuestros avances. En esta competencia se plantearon tres objetivos a alcanzar durante los años previos:

- Clasificar por lo menos dos equipos a fase regional latinoamericana (Hasta el 2016 avanzaban los primeros 40, desde el 2017 los primeros 45 a nivel nacional)
- Clasificar por lo menos uno de los equipos de forma directa (Llamamos clasificación directa a lograr un cupo directamente a través de los ejercicios resueltos en competencia. Por el alto nivel de dificultad que presenta esta competencia, generalmente no se llenan los 40 o 45 cupos durante la competencia, y los organizadores otorgan los cupos sobrantes a los demás equipos según su desempeño en entrenamientos, CCPL y su cercanía a resolver ejercicios durante la competencia).
- Obtener el primer lugar de la región Santanderes. La maratón nacional se lleva a cabo en simultánea en 6 universidades distribuidas geográficamente por Colombia. Si bien no es obligación participar en una sede específica (cada Universidad elige en qué sede participar), por cercanía tradicionalmente las Universidades de los dos Santanderes se dan cita en la sede de la Universidad Pontificia Bolivariana en Bucaramanga. El objetivo en este caso es tener el primer puesto entre los equipos de esta sede.

Estos objetivos fueron definidos a largo plazo en conjunto por los equipos y el coach participantes en la Maratón Nacional del año 2014, y desde la entrada en funcionamiento del marco de trabajo se alineó el trabajo con el fin de hacer realidad este objetivo. Los resultados han sido los siguientes:

Tabla 21. Desempeño UFPS en maratones nacionales de programación

Año	Estado	Puesto UFPS en Colombia	Puesto UFPS en región Santanderes	Total clasificados	Clasificados directamente	Clasificados por invitación
2014	Antes de iniciar el proyecto actual	> 33	> 2	1	0	1
2015		29	2	1	1	0
2016	Con el proyecto actual en ejecución	> 27	> 0	1	0	1
2017		16	1	2	1	1

- El color verde implica cumplimiento del objetivo, mientras el color rojo no cumplimiento.
- Cuando se utiliza la notación > implica que no se logró resolver ejercicios y por tanto no se cuenta con un puesto fijo. (> 33) significa que 33 equipos solucionaron al menos un ejercicio, y por tanto el equipo en cuestión estaría detrás de esos 33, sin un puesto

definido.

Durante el año 2016 no se lograron los objetivos propuestos en la Maratón Nacional a pesar de estar en plena implementación del marco. Esto llevó a un análisis retrospectivo completo que ha sido especificado en los capítulos anteriores, con diferentes cambios que propiciaron por fin el logro de los 3 objetivos propuestos.

Los resultados obtenidos durante el 2017 superaron las expectativas. En primer lugar, se alcanzó el puesto 16 entre 120 equipos, lo que nos ubica en el decil 9 de dicha competencia. En segundo lugar, por primera vez un equipo UFPS logró el primer lugar de la prueba entre los equipos de los Santanderes, posicionándose como un referente de la región. En tercer lugar, gracias a su buen desempeño la organización invitó un segundo equipo UFPS a hacer parte de la competencia latinoamericana.

Una vez cumplidos los 3 objetivos, se recomienda a los equipos y coach actuales definir nuevos objetivos retadores a mediano plazo para continuar el proceso de crecimiento y consolidación de la UFPS en competencias de este tipo.

Por su parte, estos han sido los resultados obtenidos en competencias regionales latinoamericanas:

Tabla 22. Desempeño UFPS en maratones regionales latinoamericanas de programación

Año	**Puesto en región Latinoamérica norte**	**Puesto entre los equipos colombianos**	**Logros**
2014	18	15	-
2015	18	15	-

2016	20	17	-Equipo ganador del warmup. -Primer equipo en resolver un problema en competencia.
2017	31	26	-Primer equipo en resolver un problema en competencia.

En competencia regional latinoamericana la Universidad también ha desempeñado un gran papel. En los 4 años consecutivos logrando la clasificación ha conseguido resolver por lo menos dos ejercicios cada año sin excepción. Si bien los resultados el último año no fueron los esperados, por segundo año consecutivo un equipo de la institución logró ser el equipo más rápido de toda la competencia en resolver un problema, valiéndole un amplio reconocimiento por este motivo.

Una vez alcanzados los objetivos en la fase nacional y ya con un marco de trabajo y un grupo consolidado, consideramos que en futuras ocasiones deben establecerse objetivos anuales que tengan como centro los resultados en esta fase latinoamericana.

18. Conclusiones

El fin primordial de este proyecto es mejorar los resultados de los estudiantes de la UFPS que participan en maratones de programación. Y esto no es algo que se logre de un día para otro; requiere disciplina, esfuerzo y mucho entrenamiento por parte de los estudiantes. Pero para que este esfuerzo valga la pena, debe estructurarse, apoyarse en bases sólidas, tener en cuenta todos los aspectos y medir activamente los resultados. Para eso surge el marco de trabajo, para definir los lineamientos y las acciones que lleven al grupo de estudio en programación competitiva de la UFPS a ser un referente nacional en competencias de programación.

La práctica constante es completamente necesaria para obtener buenos resultados. Por eso los estudiantes resuelven ejercicios en la Red de Programación Competitiva, en la Liga Colombiana de Programación, en las sesiones del grupo y en su casa, de forma individual y grupal. Su trabajo es su forma de progreso, y el grupo le aporta la guía, las herramientas y la metodología. El nivel debe ir mejorando con el tiempo, y por tanto lo ejercicios propuestos son también cada vez más retadores, de forma que sus resultados sean cada vez mejores.

Con el paso del tiempo, los estudiantes de la Universidad resuelven cada vez más ejercicios y son cada vez más reconocidos a nivel nacional. La Universidad Francisco de Paula Santander ha dejado de ser un simple asistente, a ser una institución a tener en cuenta en las competencias de programación, y los resultados lo confirman. El objetivo, hasta el momento, se está cumpliendo. Pero este no es motivo para detener el trabajo o para relajarse; todo lo contrario, justo cuando el marco de trabajo para el grupo de estudio ha alcanzado un nivel de madurez suficiente para guiar el proceso de aprendizaje, es cuando más debe haber coordinación entre estudiantes y docentes para que semestre a semestre, el grupo de estudio, guiado por este proyecto siga midiendo y mejorando sus resultados a nivel nacional e internacional.

Uno de los puntos más complejos de la realización del proyecto se encontró en la continuación a través del tiempo. Los estudiantes, e incluso los docentes cambian a través del tiempo, y se convirtió en una labor imperativa establecer mecanismos y procesos que sean fácilmente reproducibles para que, a pesar del cambio en el personal, el trabajo siga realizándose de la mejor forma posible. Al iniciar este proyecto, era una labor relativamente fácil la implementación del trabajo realizado, pues los autores de este proyecto éramos justamente quienes ejercíamos como líderes del grupo. Sin embargo, para analizar el traspaso y el cambio fuimos dando espacio a nuevos estudiantes de continuar el proceso. El segundo semestre de 2017 fue un proceso completo de transición llevado a cabo con éxito. Nuevos estudiantes se vincularon al grupo, y algunos de los que llevaban más tiempo asistiendo tomaron el liderazgo. Hemos logrado que la importancia del proceso recaiga en el proceso en sí, y no en las personas a cargo, para hacerlo perdurable en el tiempo.

Para garantizar que el proceso resultara efectivo, es necesario establecer mediciones que permitan definir la mejora, o los puntos débiles para intervenir. La ventaja en este caso, es que existen múltiples variables de importancia muy fáciles de medir: Los resultados de los estudiantes, la asistencia al grupo, la cantidad de ejercicios resueltos, la cantidad de soluciones por temas, sus intervenciones y sus resultados en competencias oficiales y de entrenamiento. Y todo esto ha sido medido. En el proceso, no siempre las mediciones arrojaron los resultados esperados, pero gracias a la metodología utilizada se pudieron corregir situaciones sobre la marcha, y al final alcanzar los puestos deseados.

En definitiva, el marco de trabajo propuesto constituye un punto de confluencia para el grupo de estudio, y un lineamiento claro a seguir para avanzar en el proceso de mejora continua que permita a la UFPS alcanzar los mejores resultados en competencias de programación.

19. Recomendaciones

Basaremos nuestras recomendaciones en dos aspectos: investigativas y prácticas.

En las recomendaciones investigativas, incluimos todo aquello que a nivel de investigación pueda surgir de este proyecto o del grupo. En primer lugar, en el espacio investigativo del programa: semilleros, grupos de estudio y materias de investigación la gran mayoría de proyectos se encuentra enfocado en la realización de sistemas de información, con toda la razón, pues este es el eje central de la misión de la carrera. Sin embargo, en semilleros como este existe el espacio para ahondar en investigaciones específicas de algoritmos, estructuras de datos complejas, complejidades algorítmicas y temáticas adjuntas al grupo de estudio. No solo existe el espacio, sino además el ambiente propicio para poner en práctica las pruebas prácticas necesarias y el apoyo suficiente para llevarlo a cabo. De hecho, estos materiales podrían fortalecer el estudio en el grupo.

Del mismo modo, la plataforma desarrollada en este proyecto se ha realizado de forma que pueda extenderse fácilmente a nuevos lenguajes y herramientas. Por tanto, nuevos proyectos que extiendan esta plataforma son completamente bienvenidos.

Por su parte, en las recomendaciones prácticas tratamos aspectos exclusivos de como continuar la implementación de este proyecto en el grupo. En primer lugar, es fundamental que quienes cada semestre sean los líderes del grupo se mantengan incentivando a los nuevos estudiantes a continuar, y a tomar el liderazgo más adelante, para no llegar a un punto muerto donde nadie quiera tomar la partida.

En segundo lugar, los materiales tienden a quedarse obsoletos con el tiempo. Por esto se recomienda encarecidamente que una vez estos materiales se hagan obsoletos sean actualizados.

También es importante que todo cambio es bienvenido, siempre y cuando responda a un intento mejora, y sus resultados sean medidos para verificar su impacto. Este marco no pretende ser una caja negra rígida que indica cómo hacer las cosas nos guste o no. Por el contrario, es un punto de consenso donde todos realizamos nuestros aportes para el beneficio común.

Finalmente, pero no menos importante, la maratón de Programación UFPS y la participación en la Maratón Nacional deben seguirse promoviendo año a año pues han resultado ser los medios más efectivos para incorporar estudiantes al grupo, y para incentivar su participación continua.

Referencias Bibliográficas

Asamblea Nacional Constituyente. (1991). *Constitución Política de Colombia.* Bogotá DC.

Bez, J. L., Ferreira, C., & Tonin, N. (2013). URI Online Judge Academic: A Tool for Professors.

Bez, J. L., Tonin, N., & Rodegheri, P. (2014). URI Online Judge Academic: A tool for algorithms and programming classes. *2014 9th International Conference on Computer Science & Education.* Vancouver: IEEE.

Bez, J. L., Tonin, N., & Selivon, M. (2015). URI Online Judge Academic: Integração e Consolidação da Ferramenta no Processo de Ensino/Aprendizagem.

Bloomfield, A., & Sotomayor, B. (2016). A Programming Contest Strategy Guide. *SIGCSE '16 Proceedings of the 47th ACM Technical Symposium on Computing Science Education*, 609-614.

Consejo Superior Universitario UFPS. (2012). *Acuerdo 056.* San José de Cúcuta.

Cormen, T., Leiserson, C., Rivest, R., & Clifford, S. (2012). *Introduction to Algorithms* (II ed.). Massachusetts: MIT Press.

Cuesta, B., & González, S. (2016). *Introducción a docker.*

Dean, R. (2017). *La investigación tecnológica en las ciencias de la ingeniería y la innovación tecnológica.* From Universidad Nacional del Rio Cuarto: https://www.unrc.edu.ar/publicar/23/dossidos.html

Echavarría, A. (2013). Desarrollo e implementación de un programa de trabajo para el Semillero de Programación EAFIT. , 4p.

GitHub. (2016). *Classroom Guide: A step-by-step guide on setting up your GitHub Organization for classroom use.* From Github for Education: https://education.github.com/guide

Guerrero Calvache, S. M., Diaz Riascos, E. D., Córdova Pérez, F. E., & Hernandez, G. J. (2017). Las Maratones de Programación, un paso más al campo de la Investigación. *RTE - Revista Tecnológica SPOL, 28*(5), 424-434.

Halim, S. (2017). *CS3233 - Competitive Programming*. From NUS - National University of Singapur: http://www.comp.nus.edu.sg/~stevenha/cs3233.html

Halim, S., & Halim, F. (2013). *Competitive Programming* (3 ed.). Singapur: Lulu.

ICFES. (2017). *Guía de interpretación y uso de resultados del examen Saber PRO.* Bogotá CD: |CFES.

ICPC. (2017). *ICPC Fact Sheet.*

Jimenez Becerra, L. M. (2012). *Herramienta de estudio para las maratones de programación promovidas en el programa de ingeniería de sistemas y computación de la Universidad Tecnológica de Pereira.* Pereira: Universidad Tecnológica de Pereira.

Joyanes Aguilar, L. (1996). *Fundamentos de Programación* (Segunda ed.). Madrid: McGraw Hill.

Laaksonen, A. (2017). *Competitive Programmer's Handbook.* Helsinki.

Ngan, S.-C., & Law, K. (2015). Exploratory Network Analysis of Learning Motivation Factors in e-Learning Facilitated Computer Programming Courses. *The Asia-Pacific Education Researcher*, 705-717.

Park, J. (2016). *CS 97SI: Introduction to Programming Contests*. From Universidad de Stanford: http://web.stanford.edu/class/cs97si/

Pineda, M. (2015). *Notebook Generator*. From https://github.com/pin3da/notebook-generator

PPC - Competitive Programming Practice Center. (2014). From Guiame: http://guiame.org/index.php?option=proyectos

Servicio de Innovación Educativa de la Universidad Politécnica de Madrid . (2008). *Aprendizaje basado en problemas, guías rápidas sobre nuevas metodologías.* Madrid: Servicio de Innovación Educativa de la Universidad Politécnica de Madrid .

Skiena, S., & Revilla, M. A. (2003). *Programming Challenges: The Programming Contest Training Manual (Texts in Computer Science).* New York: Springer.

Universidad EAFIT. (2014). Aprendizaje colaborativo/Cooperativo. *Conexiones*.

Vizcarro, C., Juarez, E., Romero, A., García-Sevilla, J., Prieto, A., Diaz, D., . . . Navarro, J. J. (2009). *La metodología del aprendizaje basado en problemas.* Universidad de Barcelona: Universidad de Barcelona.

Wang, J. (2017). *Stanford Local programming Contest: Preparation*. From Stanford Local programming Contest: https://cs.stanford.edu/group/acm/SLPC/training.php

Anexos

Anexo 1. Anexos Digitales

Este proyecto contiene varios anexos digitales en pdf, imágenes y software, que se adjuntan a través del siguiente link:

http://www.ufpstrainingcenter.com/anexos

A través de ese link podrá acceder a:

- Plataforma de entrenamiento UFPS Training Center.
- Repositorios mencionados en el documento (syllabus, soluciones, notebooks)
- Guía metodológica del grupo de estudio.
- Manuales de competencia (Notebooks) en C++ y Java.
- Documentación técnica de la plataforma UFPS Training Center.
- Diagramas de la plataforma UFPS Training Center.

Anexo 2. Modelo de encuesta para conocer las motivaciones y dificultades de los estudiantes del grupo de estudio

Encuesta motivacional grupo de estudio en programación competitiva

Encuesta para conocer las motivaciones que llevan a los estudiantes a hacer parte del grupo de estudio en programación competitiva.

*Obligatorio

¿Cual es su motivación para hacer parte del grupo de estudio? *

- [ ] Representar a la carrera en diferentes competencias de programación de forma oficial.
- [ ] Pertenecer a un semillero
- [ ] Aprender nuevas temáticas
- [ ] Reforzar los temas vistos en las materias de la carrera
- [ ] Obtener proyección profesional y oportunidades laborales
- [ ] Otro: _____

¿Que dificultades tuvo para hacer parte del grupo de estudio? *

- [ ] Las competencias tienden a ser muy complicadas.
- [ ] Muchas temáticas son de semestres superiores al que me encuentro.
- [ ] El horario de competencias no me resulta cómodo.
- [ ] No tengo equipo para competir
- [ ] No tuve dificultades con el grupo
- [ ] Otro: _____

ENVIAR

Anexo 3. Metadatos y resultados de la encuesta para conocer las motivaciones y dificultades de los estudiantes del grupo de estudio

Fecha: Encuesta realizada entre el 24 y 27 de agosto del 2017

Modalidad: Virtual, a través de Google Forms

Muestra: 21 estudiantes que asistieron al grupo, bien fuera de forma activa o esporádica, durante el semestre anterior a la realización de la encuesta (semestre I de 2016)

Resultados:

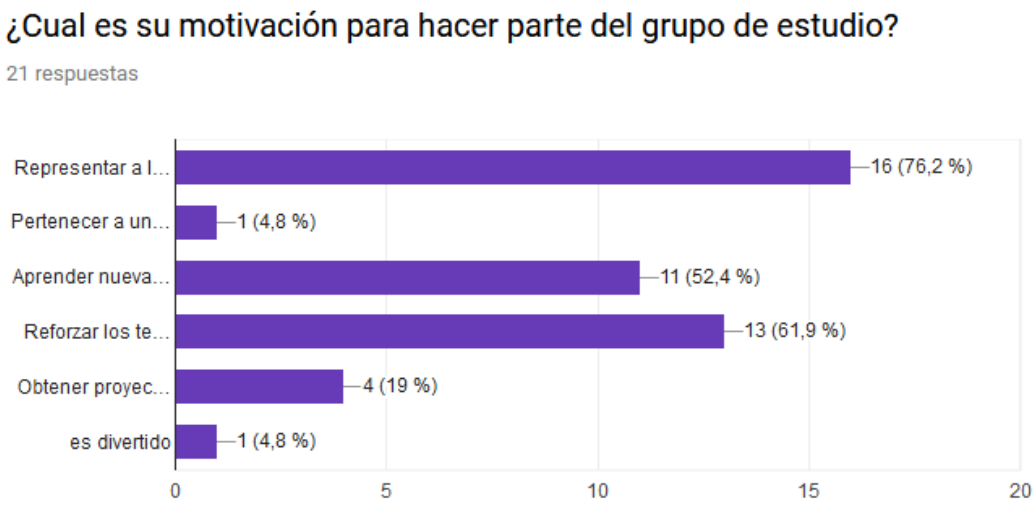


Gráfico 1 Anexo C. Resultados a la pregunta sobre la motivación para hacer parte del grupo. Gráfico tomado directamente de los resultados de Google Forms.

La pregunta proponía 5 opciones de respuesta, y un espacio abierto para añadir más opciones. Solo un usuario añadió una opción más: “es divertido”.

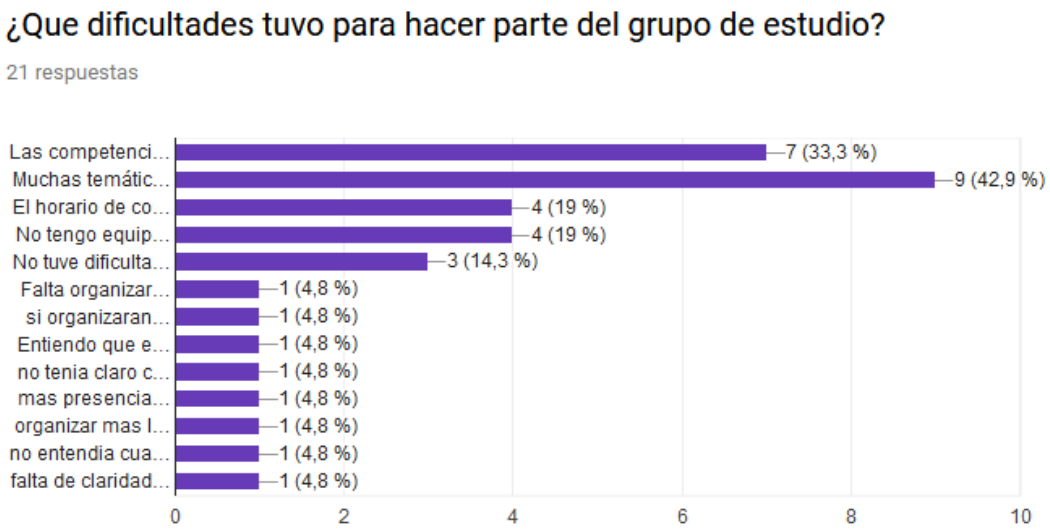


Gráfico 2 Anexo C. Resultados a la pregunta sobre las dificultades para hacer parte del grupo. Gráfico tomado directamente de los resultados de Google Forms.

La pregunta proponía 5 opciones de respuestas con dificultades que los autores consideramos posibles, junto a una opción para añadir nuevas dificultades. Y una opción más indicando: No tuve dificultades en el grupo. Solo 3 estudiantes marcaron esta opción. Por su parte, en el gráfico aparecen 8 opciones diferentes con un solo voto cada una (todas añadidas por los estudiantes en la opción “otra”, pero luego de analizarlas, concluimos que todos se refieren a lo mismo: La desorganización del grupo durante dicho semestre, y por tanto fue tenida en cuenta como una sola. Las 8 opciones añadidas por los estudiantes y que aparecen en la parte inferior del gráfico son:

- No tenía claro cuando se reunían y cuando no
- Falta organizar más las actividades
- Más presencia del director del semillero

- Si organizaran más el grupo sería mejor
- Organizar más los horarios de las sesiones
- No entendía cuando hacían sesiones y cuando no, por eso falté a varias y al volver ya había perdido el ritmo
- Entiendo que es organizado solo por estudiantes, pero si intentaran dar más orden a los temas sería más fácil entender
- Falta de claridad en los horarios

Las opciones anteriores son transcritas de la misma forma que fueron escritas por los estudiantes, corrigiendo solo cuestiones ortográficas. En general la queja era la desorganización, especialmente en los horarios, lo cual fue tenido en cuenta en los semestres sucesivos.

Anexo 4. Modelo de encuesta para conocer la satisfacción de los estudiantes del grupo de estudio al finalizar cada semestre

Encuesta de satisfacción grupo de estudio UFPS

Esta encuesta es anónima. Puede responder con total tranquilidad, sus respuestas serán usadas para mejorar el grupo.
Para las siguientes preguntas, responda de 1 a 5 donde 1 es totalmente insatisfecho y 5 totalmente satisfecho

*Obligatorio

¿Cual es su nivel de satisfacción respecto a los temas tratados en el grupo de estudio? *

	1	2	3	4	5	
Totalmente insatisfecho	○	○	○	○	○	Totalmente satisfecho

¿Cual es su nivel de satisfacción respecto a las explicaciones y socializaciones realizadas en el grupo? *

	1	2	3	4	5	
Totalmente insatisfecho	○	○	○	○	○	Totalmente satisfecho

¿Cual es su nivel de satisfacción respecto a los materiales de apoyo usados en el grupo? *

	1	2	3	4	5	
Totalmente insatisfecho	○	○	○	○	○	Totalmente satisfecho

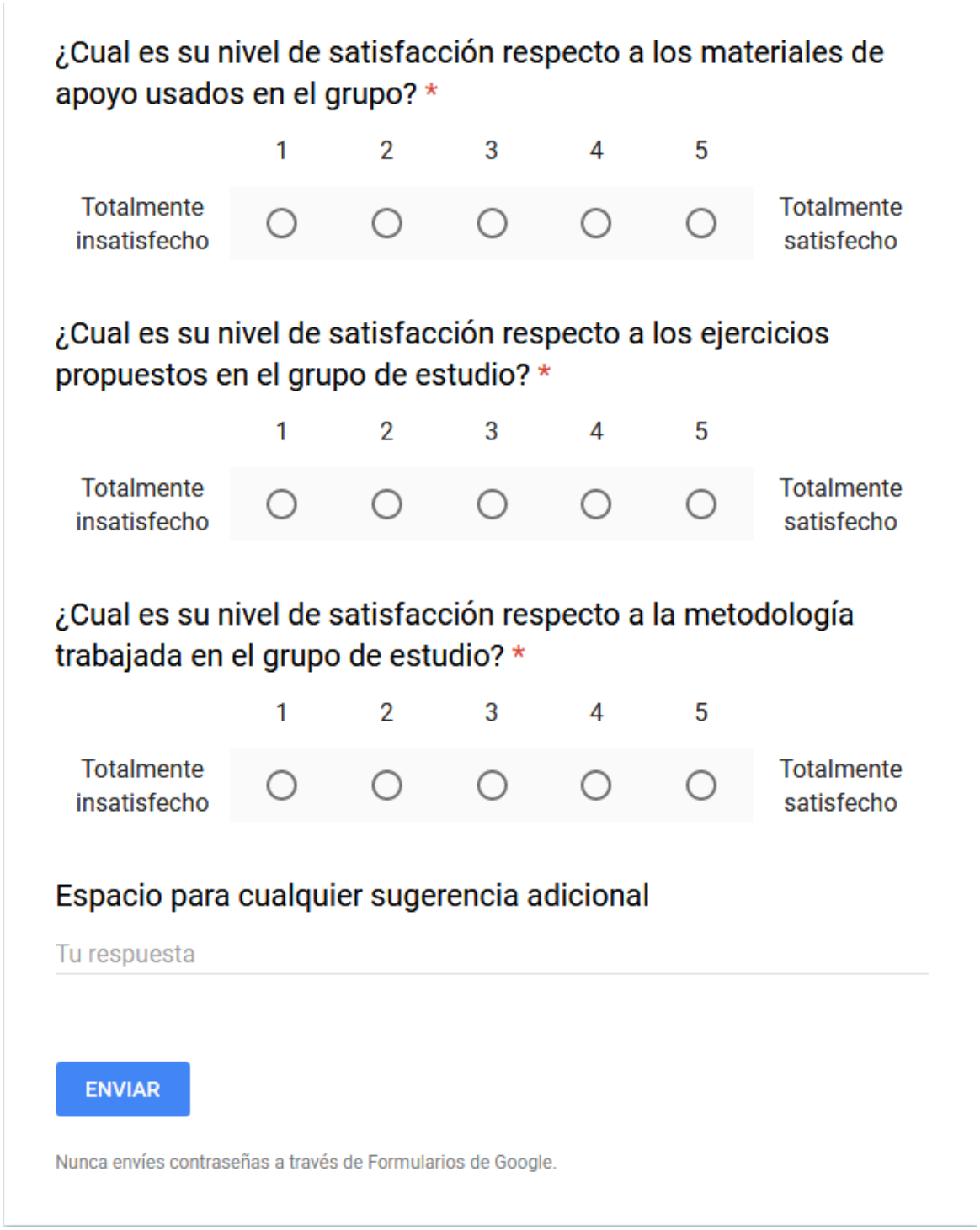


Gráfico 1 Anexo D. Modelo de la encuesta. Cabe destacar que en la primera versión de la encuesta (al finalizar el primer semestre de 2016 la segunda pregunta solo preguntaba por explicaciones y no por socializaciones, debido a las características del proceso en ese momento)

Anexo 5. Metadatos y resultados de la encuesta para conocer la satisfacción de los estudiantes del grupo de estudio al finalizar cada semestre

La misma encuesta ha sido aplicada al finalizar 3 semestres diferentes para comparar sus resultados a través del tiempo:

Tabla 1 Anexo E. Fechas de la encuesta

Semestre	Fecha de realización	Tamaño de la muestra
2016 – II	06 de diciembre del 2016 al 08 de diciembre del 2016	11
2017 – I	21 de junio del 2017 al 24 de junio del 2017	20
2017 – II	24 de enero del 2018 al 27 de enero del 2018	16

Modalidad: Virtual, a través de Google Forms

Tabla 2 Anexo E. Resultados en el segundo semestre de 2016

Fecha/hora de respuesta	Pregunta 1	Pregunta 2	Pregunta 3	Pregunta 4	Pregunta 5
6/12/2016 11:23:21	3	3	5	5	3
6/12/2016 11:38:13	4	3	5	5	5
6/12/2016 13:01:20	5	5	5	5	5
6/12/2016 13:19:33	4	4	4	5	4
6/12/2016 15:39:43	5	4	4	4	5
6/12/2016 19:12:31	4	3	3	4	4
6/12/2016 21:36:47	5	5	5	5	4

7/12/2016 10:24:12	4	3	5	5	4
7/12/2016 15:31:32	5	4	4	5	5
8/12/2016 14:26:11	2	3	4	5	3
8/12/2016 17:12:57	5	5	5	5	5
Promedio	4,181818182	3,818181818	4,454545455	4,818181818	4,272727273

Adicionalmente, 3 usuarios dejaron sus comentarios de retroalimentación a través de la encuesta. Dichos comentarios fueron (en palabras de los estudiantes, solo se ha corregido la ortografía de ser necesario):

- informar mejor cuando hay competencias los sábados
- Falta más organización en las actividades.
- Es muy difícil para alguien de segundo semestre llevar el ritmo

Tabla 3 Anexo E. Resultados en el primer semestre de 2017

Fecha/hora de respuesta	**Pregunta 1**	**Pregunta 2**	**Pregunta 3**	**Pregunta 4**	**Pregunta 5**
21/06/2017 11:21:08	4	4	5	5	4
21/06/2017 11:23:14	4	4	4	4	5
21/06/2017 11:40:18	5	4	5	5	4
21/06/2017 15:20:52	5	5	5	5	5
21/06/2017 17:12:01	4	3	4	4	4
21/06/2017 18:43:43	5	5	5	5	5

21/06/2017 20:11:00	5	4	5	5	5
21/06/2017 23:05:17	4	4	5	5	5
22/06/2017 13:43:11	4	4	4	4	4
22/06/2017 17:22:32	5	4	5	5	4
22/06/2017 18:14:25	3	4	5	3	4
22/06/2017 20:41:34	3	3	5	5	5
22/06/2017 22:51:49	3	5	5	5	5
23/06/2017 8:11:58	5	5	5	5	5
23/06/2017 12:06:09	4	5	5	5	4
23/06/2017 13:43:25	5	4	5	5	5
23/06/2017 18:52:02	4	4	4	4	4
23/06/2017 21:15:43	5	5	5	4	5
24/06/2017 7:54:23	5	5	5	5	4
24/06/2017 19:23:03	4	5	5	5	5
Promedio	4,3	4,3	4,8	4,65	4,55

Adicionalmente, 2 usuarios dejaron sus comentarios de retroalimentación a través de la encuesta. Dichos comentarios fueron (en palabras de los estudiantes, solo se ha corregido la ortografía de ser necesario):

- no avanzar tan rápido en los temas
- explicar los ejercicios de las competencias de los sábados

Tabla 4 Anexo E. Resultados en el segundo semestre de 2017

Fecha/hora de respuesta	Pregunta 1	Pregunta 2	Pregunta 3	Pregunta 4	Pregunta 5
24/01/2018 23:54:38	5	5	5	5	5
25/01/2018 0:01:06	5	5	5	5	5
25/01/2018 0:05:34	5	5	5	5	5
25/01/2018 11:27:22	4	4	5	5	4
25/01/2018 11:35:34	4	5	4	5	4
25/01/2018 11:51:22	4	5	5	5	5
25/01/2018 12:07:39	5	5	5	5	5
25/01/2018 12:19:18	5	4	5	5	5
25/01/2018 12:34:15	5	5	5	5	5
25/01/2018 12:43:23	5	5	5	5	5
25/01/2018 14:21:39	5	5	5	5	5
25/01/2018 14:43:58	5	5	5	5	5
25/01/2018 14:44:12	4	5	5	5	5
25/01/2018 15:01:21	5	5	5	5	5
25/01/2018 16:29:27	5	5	5	4	5
25/01/2018 17:30:39	4	3	5	4	4
Promedio	4,6875	4,75	4,9375	4,875	4,8125

Adicionalmente, 2 usuarios dejaron sus comentarios de retroalimentación a través de la encuesta. Dichos comentarios fueron (en palabras de los estudiantes, solo se ha corregido la ortografía de ser necesario):

- crear más eventos internos en la universidad para incentivar más alumnos
- Falta más motivación en el segundo semestre del año para los que no clasifican a maratón nacional ni regional.

Anexo 6. Guía Metodológica del Marco de Trabajo

Esta guía se comparte en modo digital (Ver Anexo A) como documento independiente, y se entrega semestralmente a los líderes del grupo para que interioricen las herramientas con las que cuentan.

Introducción

Una de las partes fundamentales en el marco de trabajo desarrollado para el grupo de estudio en programación competitiva es la metodología. La importancia radica en que si bien existen varios componentes independientes (material escrito, sesiones, plataforma de software, entre otros) sin una metodología que marque el norte y el proceso a seguir en el grupo, realmente este marco no tendría mucho sentido. La metodología de trabajo surge para coordinar todos los esfuerzos independientes en un solo esfuerzo hacia el cual coordinar el trabajo realizado para alcanzar los objetivos planteados.

Esta guía metodológica busca ser un documento corto y ameno al cual los estudiantes que toman el liderazgo del grupo puedan referirse para la dirección y organización de las tareas planteadas.

Sin embargo, este documento no es una línea rígida para seguir al pie de la letra. Por el contrario, define unos lineamientos generales permitiendo que cada lider adopte sus propias estrategias como parte de el. Incluso es posible en ciertas circunstancias planear actividades especiales (competencias temáticas, conferencias, etc.) que se salgan de los lineamientos propuestos. Estas actividades extra-metodológicas permiten evitar la monotonía.

Roles y responsabilidades

En el grupo, todos los estudiantes tienen las mismas responsabilidades, y los mismos incentivos. Sin embargo, para fines organizativos se hacen algunas distinciones:

Rol	Descripción
Líderes del grupo	Estudiantes que realizan las labores administrativas del grupo, y gestionan la comunicación entre el docente director del semillero, y los estudiantes que conforman el grupo. Son los encargados de promover acciones y cambios en el grupo de ser necesario.
Estudiantes con experiencia	Estudiantes que han asistido por lo menos un semestre al grupo, y han participado en competencias de RPC/CCPL. Ellos pueden llevar a cabo socializaciones, dirigir sesiones y proponer ejercicios.
Estudiantes nuevos	Estudiantes que asisten durante el semestre en curso por primera vez al grupo.

Cabe resaltar que esta clasificación se realiza solo con fines organizativos (resulta más fácil para los procesos del grupo tener definidos los estudiantes que pueden realizar una u otra actividad). Sin embargo, nada impide a un estudiante nuevo llevar a cabo socializaciones o sesiones, ni a un estudiante con experiencia proponer acciones directas para el grupo. Ante todo, el grupo mantiene un enfoque abierto.

Sesiones de entrenamiento

Se llaman sesiones de entrenamiento a las actividades realizadas semanalmente en el grupo diferentes a las competencias. En ellas se busca que los estudiantes aprendan nuevas temáticas y fortalezcan aquellas que ya conocen.

Estas actividades se deben iniciar en la medida de lo posible en la segunda semana de clases de cada semestre, y finalizar una semana antes de terminar dicho semestre. El horario es elegido

en la primera sesión de común acuerdo entre todos los integrantes del grupo. Preferiblemente, debe evitarse que esta actividad se planee para los días lunes (la cantidad de lunes festivos durante el semestre puede afectar el número de sesiones disponibles). En semanas de previos se recomienda realizar una pausa (generalmente durante la segunda semana de previos) para dar espacio a los estudiantes de dedicarse a su material al 100%.

Una sesión se divide en dos partes: Sección básica y sección avanzada. En la sección básica se inicia desde fundamentos de programación y temas de iniciación (enfocados siempre en maratones de programación), mientras en la sección avanzada se enseñan temáticas para las cuales se requiere tener mayores conocimientos previos. Los estudiantes pueden participar en cualquiera de las dos secciones según su nivel académico actual. De esta forma se garantiza que estudiantes de cualquier semestre pueden participar, eligiendo las temáticas básicas aquellos que están en primeros semestres, y las avanzadas los de semestres finales. De cualquier forma, cabe indicar que no se impone ninguna restricción sobre la participación: Cada estudiante es el que define según su conocimiento a cuál sección desea asistir, o incluso puede participar de las dos al tiempo.

Cada sección se divide a su vez en dos partes: En una primera parte se realiza una socialización sobre un tema específico o sobre un conjunto de problemas previamente vistos, y en la segunda parte se realizan los ejercicios propuestos. Cabe indicar que, en la sección de ejercicios propuestos, se realiza trabajo individual asistido: Cada estudiante debe solucionar los problemas individualmente, pero se permite (e incluso, se recomienda) la socialización para resolver dudas e inquietudes.

En estas sesiones se utiliza como eje el aprendizaje colaborativo. Cada sesión la dirige inicialmente entre uno y tres estudiantes, explicando el tema. Estos estudiantes recibirán un

incentivo en el ranking. Estos incentivos permiten que los estudiantes decidan tomar la iniciativa de forma voluntaria, y tomar la voz en las temáticas que mejor manejan. Una vez se hace la explicación de un tema, se procede a socializar entre todo el grupo. Se busca el consenso grupal, que todos resuelvan sus dudas e inquietudes. En muchas ocasiones es común que los estudiantes no expresen sus preguntas en un primer instante; para incentivar la participación, quienes dirigen la sesión y los líderes del grupo pueden empezar a plantear preguntas abiertas, para verificar si los estudiantes han entendido correctamente, o si aún hay falencias. Esta actividad dura en total una hora.

A continuación, se plantean una serie de ejercicios a realizar. los ejercicios están accesibles directamente en el "UFPS Training Center", plataforma de entrenamiento. Es en este punto donde se pone en práctica el aprendizaje basado en problemas. Los estudiantes se enfrentan a un conjunto de problemas de diferente nivel de dificultad, y buscan encontrar la solución óptima en el menor tiempo posible. Sin embargo, no por incluir un nuevo enfoque se deja de lado el aprendizaje colaborativo; por el contrario, los dos enfoques se complementan a la perfección. Los estudiantes a cargo de la sesión y los líderes del grupo constantemente se dirigen a los diferentes estudiantes para resolver sus dudas e inconvenientes, y se permite la colaboración entre los estudiantes para mejorar sus resultados.

Tanto la sección básica como la avanzada tienen la misma estructura, resultando al final de 2 horas cada sección. Si bien el horario se establece a conveniencia de los estudiantes, se busca que ambas sesiones se realicen el mismo día, una tras la otra para permitir que los estudiantes que lo deseen asistan directamente a las dos secciones.

Integración de las competencias en la metodología

Las competencias realizadas por RPC/CCPL cuentan con una plataforma, lenguajes, problemas y horarios definidos por la organización, y por tanto no está en nuestras manos realizar cambios sobre ellos (aunque en realidad tampoco sería necesario realizar cambios: Estas competencias siguen al pie de la letra las reglas de las competencias oficiales, y por tanto permiten ejercitarse en un ambiente exactamente igual al de dichas competencias).

Sin embargo, en la semana siguiente a una competencia de RPC/CCPL se dedican 20 minutos para socializar los ejercicios que se hayan resuelto por alguno de los equipos. Esta socialización es llevada a cabo sin ningún orden específico, cada ejercicio resuelto puede ser explicado por quien tome la iniciativa, y pueden realizarse preguntas. Se incentiva que después de estas socializaciones los equipos vuelvan a revisar las competencias pasadas e intenten resolver los ejercicios que no lograron realizar en la competencia, teniendo en cuenta las indicaciones de sus compañeros. Lo fundamental de esta socialización es descubrir las dificultades que tuvieron los estudiantes y responder sus inquietudes para fortalecer sus capacidades.

Actividades adicionales

Los ejercicios propuestos en la sesión tienen una hora para ser resueltos. Sin embargo, se mantiene un plazo de una semana para realizarlos, permitiendo su continuación en casa en caso de no terminarlos en la sesión.

Adicional a esto, se realizan algunas competencias nocturnas para realizarse desde casa. Estas actividades son opcionales, pero se recomienda su participación para obtener incentivos en el grupo.

Trabajo individual y trabajo grupal

Las competencias de programación se realizan de forma grupal. Sin embargo, es necesario que cada miembro del grupo se encuentre en la capacidad de resolver problemas con facilidad, para que todos los miembros sumen esfuerzos, en lugar de restar. Por tanto, las actividades propuestas buscan tener espacios individuales y espacios grupales, para fortalecer las capacidades de cada estudiante, y su trabajo en equipo.

A continuación, se presentan las diferentes actividades del grupo, y su carácter individual o grupal:

Actividad	**Carácter**
Llevar a cabo una sesión o socialización	Individual o grupal*
Espacio para la solución de ejercicios	Individual**
Competencias nocturnas	Individual
Competencias organizadas por RPC	Grupal
Competencias organizadas por CCPL	Grupal

***Una socialización es, por defecto, grupal (entre todo el grupo de estudio) pero su organización y dirección puede estar a cargo de un solo estudiante, o varios, según disponibilidad.

** Si bien se explicó que no está prohibida la socialización en este espacio, se cuentan los ejercicios de cada estudiante de forma individual.

Ranking e Incentivos

El ranking no es más que una calificación de los estudiantes con respecto a sus logros en el grupo. Sin embargo, la construcción de este ranking debe ser muy cuidadosa: En el deben incluirse todas las actividades que hacen parte del grupo. Realizar una sesión suma puntos, cada ejercicio resuelto suma, cada competencia, cada acción en específico que un estudiante realice.

Al iniciar cada semestre, se define con el director del semillero y con los estudiantes los porcentajes específicos que cada actividad tendrá para el ranking.

El ranking tiene dos componentes: Individual y grupal. El individual, por supuesto, mide las actividades que cada estudiante realiza. El grupal, además de contar los resultados en RPC y CCPL asigna un porcentaje al desempeño individual de cada estudiante que conformaba ese grupo. Este ranking tiene una fecha definitiva de cierre, y los dos equipos que a esa fecha logren ocupar los dos primeros puestos en el ranking, ganan el derecho a representar a la Universidad en la Maratón Nacional de Programación.

Esto, por tanto, permite un beneficio en 3 niveles diferentes: En primer lugar, incentiva a los estudiantes a participar en el grupo (se ha implementado desde el inicio de este proyecto, y los resultados, como se verá más adelante, han sido satisfactorios). En segundo lugar, permite elegir para las representaciones a aquellos estudiantes que en realidad lo merecen por su buen desempeño. En tercer lugar, el programa y la Universidad cuentan en estas competencias con estudiantes que la representen de forma adecuada, permitiendo posicionarse como una de las instituciones a tener en cuenta a nivel nacional.

Además de esto, algunas empresas como Google realizan eventos en los cuales invitan estudiantes de la Universidad a hacer parte de ellos. Este ranking se convierte en la mejor manera de elegir los estudiantes que asisten a estos eventos.

Integración con herramientas

Las sesiones de entrenamiento no son concebidas como clases magistrales, sino como experiencias educativas. Para construir esta experiencia resulta necesario complementar la

socialización y el componente humano previamente expuesto con materiales y herramientas que fortalezcan y contribuyan al entendimiento y mejoramiento de los estudiantes.

En primer lugar, nos centraremos en el material para las competencias de programación. En una competencia de RPC/CCPL, o competencia oficial Nacional, Regional o Mundial, se encuentra prohibida la utilización de cualquier material digital o acceso a internet. Por tanto, en esta parte, la metodología debe apoyarse en material impreso. Para tal fin, en el siguiente capítulo se establece el manual de competencias (Notebook), el cual se comparte con los miembros del grupo y se promueve su impresión y utilización en dichas competencias. En fases nacional y regional 2016 y 2017 este manual ha sido de utilización obligatoria por los miembros de los equipos participantes, y su utilización ha sido recurrente en competencias locales.

En segundo lugar, la metodología utilizada durante las sesiones de entrenamiento permite integrar una mayor cantidad de herramientas. Por un lado, las socializaciones pueden apoyarse en material explicativo (presentaciones) que estén disponibles para su visualización en cualquier navegador. De esta forma, los estudiantes pueden seguir la socialización directamente desde sus pantallas o desde el video beam. Por otro lado, existen materiales adicionales (externos o propios del grupo) que se proponen como guías para revisar posteriormente. Estos materiales se comparten, si sus derechos lo permiten, con los estudiantes del grupo. Del mismo modo, como se ha precisado previamente, las sesiones contienen un componente completamente práctico en el cual los estudiantes ponen a prueba sus capacidades algorítmicas en el desarrollo de problemas propuestos. En este espacio, los estudiantes cuentan con herramientas tecnológicas que les permiten probar directamente sus soluciones en un entorno controlado y saber su calificación. Todas estas herramientas mencionadas en este punto serán explicadas a profundidad en los siguientes capítulos.

Sesiones de entrenamiento a través de diferentes semestres

Las temáticas existentes en maratones de programación son extensas y un semestre no es suficiente para cubrirlo todo. Sin embargo, cada semestre hay alumnos nuevos en el grupo, así como alumnos que continúan desde el semestre anterior. Por tanto, el grupo debe aportar semestre a semestre información introductoria que permita a los nuevos estudiantes empezar desde cero, pero también aportar nueva información, de modo que los estudiantes antiguos no se mantengan en una constante repetición de las mismas temáticas.

Para enfrentar esta problemática, de nuevo la solución surge en la división de la sesión en dos secciones: básica/avanzada. Se recomienda a los estudiantes nuevos, por lo menos en las primeras sesiones a las que asisten a hacer parte de la sección básica. De esta forma tendrá la introducción a las maratones y conceptos iniciales, y si posteriormente decide que tiene bases suficientes para pasar a la sección avanzada puede hacerlo. Por tanto, la sección básica se repite con un esquema similar semestre a semestre, que permite la iniciación en maratones de estudiantes nuevos.

Por su parte, las temáticas de la sección avanzada son un poco más flexibles y configurables, pudiendo establecerse cambios en conjunto entre el director, el grupo y los líderes al inicio del semestre. Existen suficientes temáticas avanzadas para cubrir varios semestres, pero como se ha dicho antes, algunas hacen parte de las maratones con tanta frecuencia, que tienen una importancia primordial. Por tanto, esta sección se conforma como un balance entre las temáticas principales que han sido vistas antes, como temáticas nuevas que pueden aparecer esporádicamente. Así, los estudiantes se mantienen entrenando en temas de importancia, y siguen aprendiendo nuevas cosas aun cuando llevan varios semestres en el grupo.

Anexo 7. Modelo de analisis UFPS Training Center

Introducción

El presente documento expone y define el modelo de análisis propuesto en el desarrollo de la plataforma de software UFPS Training Center, la cual hace parte del proyecto de grado "Desarrollo e implementación de un marco de trabajo para el grupo de estudio en Programación competitiva del programa Ingeniería de Sistemas UFPS".

Alcance

El presente documento cubre únicamente los aspectos relacionados con el proceso de análisis. Como tal, el documento es parte de una documentación más completa que cubre las demás fases de desarrollo, y se distribuyen de manera digital como un anexo al informe final del proyecto.

Esta plataforma de software está planeada para iniciar su funcionamiento en el primer semestre del 2018 como parte de los trabajos del grupo de estudio, una vez este inicie sus funciones. Sin embargo, la plataforma hace parte de un marco de trabajo con más componentes metodológicos, digitales y físicos que ya se encuentran en funcionamiento. Para más detalles de los demás componentes y de la integración de la plataforma con ellos, ver el documento principal del proyecto.

Definiciones, acrónimos y abreviaturas

- **ICPC:** International Collegiate Programming Contest, Maratón de Programación Oficial Universitaria más importante a nivel mundial.

- **Maratón de Programación:** Competencia donde se ponen en prueba las habilidades algorítmicas y de trabajo en equipo de los estudiantes en un entorno controlado.
- **UFPS Training Center:** Plataforma de entrenamiento en Programación Competitiva para los estudiantes de la UFPS.

Requerimientos del sistema

(En los anexos digitales los requerimientos son presentados en un documento independiente del análisis)

Listado de requerimientos funcionales

Id.	Nombre
RF-01	Listar Problemas
RF-02	Agregar problemas
RF-03	Calificar problemas
RF-04	Corregir problemas
RF-05	Eliminar problemas
RF-06	Acceder al foro de discusión
RF-07	Filtrar problemas
RF-08	Categorizar problemas
RF-09	Ver ranking
RF-10	Crear competencias
RF-11	Añadir material de ayuda
RF-12	Ver material de ayuda
RF-13	Crear Modo guiado

RF-14	Recepcionar solución de tareas
RF-15	Ver información del grupo en modo guiado
RF-16	Ver soluciones de los estudiantes en modo guiado
RF-17	Ver estadísticas
RF-18	Eliminar usuarios

Especificación de requerimientos funcionales

ID del Requerimiento:	*RF-01*
Nombre del Requerimiento:	Listar Problemas
El sistema debe desplegar al usuario la lista de problemas almacenados en la plataforma con la posibilidad de filtrar bajo diferentes criterios.	
Prioridad: Media	

ID del Requerimiento:	*RF-02*
Nombre del Requerimiento:	Agregar problemas
El sistema debe permitir a los docentes y administradores añadir nuevos problemas para que los estudiantes los solucionen.	
Prioridad: Media	

ID del Requerimiento:	*RF-03*
Nombre del Requerimiento:	Calificar problemas
El sistema debe calificar automáticamente las soluciones enviadas por los estudiantes a los diferentes problemas e indicarles su calificación.	
Prioridad: Muy alta	

ID del Requerimiento:	*RF-04*
Nombre del Requerimiento:	Corregir problemas
Los administradores tienen la potestad de corregir problemas si tienen problemas en su formato, estructura o entradas y salidas.	
Prioridad: Media	

ID del Requerimiento:	*RF-05*
Nombre del Requerimiento:	Eliminar problemas
El administrador puede borrar problemas de la plataforma si no cumple con las características que debería contar un problema de programación.	
Prioridad: Media	

ID del Requerimiento:	*RF-06*
Nombre del Requerimiento:	Acceder al foro de discusión
El sistema debe permitir las discusiones de los estudiantes en cada problema.	
Prioridad: Baja	

ID del Requerimiento:	*RF-07*
Nombre del Requerimiento:	Filtrar problemas
El sistema permite que los estudiantes filtren los problemas por dificultad, autor, idioma y otros filtros.	
Prioridad: Baja	

ID del Requerimiento:	*RF-08*
Nombre del Requerimiento:	Categorizar problemas
Los problemas pueden agruparse bajo diferentes categorías globales.	

Prioridad: Media

ID del Requerimiento:	*RF-09*
Nombre del Requerimiento:	Ver ranking
El sistema debe generar ranking de los usuarios según el número de problemas resueltos.	
Prioridad: Bajo	

ID del Requerimiento:	*RF-10*
Nombre del Requerimiento:	Crear competencias
El sistema debe permitir crear competencias (maratones) de programación en tiempo real.	
Prioridad: Alto	

ID del Requerimiento:	*RF-11*
Nombre del Requerimiento:	Añadir material de ayuda
El sistema debe permitir que los usuarios agreguen material de ayuda (pdf, videos).	
Prioridad: Media	

ID del Requerimiento:	*RF-12*
Nombre del Requerimiento:	Ver material de ayuda
Los estudiantes pueden ver el material de ayuda para mejorar su entrenamiento.	
Prioridad: Media	

ID del Requerimiento:	*RF-13*
Nombre del Requerimiento:	Crear Modo guiado

Los profesores pueden crear modos guiados (clases o syllabus) para que sus estudiantes resuelvan los ejercicios que propone y vean sus materiales.
Prioridad: Alta

ID del Requerimiento:	*RF-14*
Nombre del Requerimiento:	Recepcionar solución de tareas
En el modo guiado, los estudiantes envían las tareas propuestas por el coach/docente a la plataforma. El sistema debe permitir la recepción de estas tareas.	
Prioridad: Media	

ID del Requerimiento:	*RF-15*
Nombre del Requerimiento:	Ver información del grupo en modo guiado
El sistema debe permitir al coach/docente ver la información general de las soluciones de las tareas que han enviado sus estudiantes.	
Prioridad: Media	

ID del Requerimiento:	*RF-16*
Nombre del Requerimiento:	Ver soluciones de los estudiantes en modo guiado
El sistema debe permitir a los profesores/coach ver las soluciones de sus estudiantes a las tareas en el modo guiado.	
Prioridad: Media	

ID del Requerimiento:	*RF-17*
Nombre del Requerimiento:	Ver estadísticas
El sistema debe permitir al estudiante ver sus propias estadísticas de envíos y soluciones.	
Prioridad: Media	

ID del Requerimiento:	*RF-18*
Nombre del Requerimiento:	Eliminar usuarios
El sistema debe permitir al administrador eliminar usuarios no deseados.	
Prioridad: Media	

Requerimientos no funcionales

Id	**Nombre**	**Tipo**	**Descripción**
RNF-01	El sistema debe mantenerse estable aún en momentos de afluencia de público.	Eficiencia	Durante una sesión de entrenamiento o una competencia hay en promedio 30 estudiantes. En una clase práctica puede haber hasta 50 estudiantes. El sistema debe mantenerse estable aún si todos estos usuarios están conectados al tiempo.
RNF-02	Los envíos de soluciones no deben afectar la estabilidad del sistema.	Eficiencia	La calificación de una solución enviada por un estudiante implica su ejecución. Algunos problemas no tienen soluciones de complejidad algorítmica baja, y la ejecución de cada solución puede tardar varios segundos. Aún si la calificación presenta congestión por esta razón, el resto de la plataforma debe mantenerse estable y rápida.
RNF-03	Las contraseñas de los usuarios deben almacenarse encriptadas.	Seguridad	Las contraseñas de los usuarios se guardarán de forma encriptada en la base de datos, de tal forma que ni siquiera los administradores puedan acceder a ellas.
RNF-04	Las soluciones enviadas por los usuarios deben ejecutarse en modo “sandbox”.	Seguridad	Los estudiantes envían soluciones en forma de código de programación que se califican automáticamente en la plataforma. Estos códigos deben ejecutarse en un modo “sandbox” que

			impida que un código malicioso realice daños en la plataforma o demás datos alojados en el servidor.
RNF-05	El tiempo de aprendizaje de un usuario con la plataforma debe ser menor a 20 minutos.	Usabilidad	La interfaz gráfica de la plataforma debe ser lo suficientemente simple para que los usuarios logren entenderla y utilizarla completamente en, como máximo, 20 minutos.
RNF-06	Manuales de usuario	Usabilidad	La plataforma debe contar con manuales de usuario debidamente documentados y claros.
RNF-07	El sistema debe funcionar en cualquier plataforma moderna.	Usabilidad/Producto	La plataforma debe funcionar de forma similar en sistemas Windows/Linux/Mac OS. Esta debe operar a través de cualquier navegador web moderno (Google Chrome, Mozilla Firefox, Internet Explorer 10+, Microsoft Edge, Safari, Opera, Vivaldi, Maxthon o Brave Browser). En cualquier navegador no mencionado en esta lista la aplicación podría funcionar si utiliza tecnologías de HTML/CSS/Javascript modernas.

Análisis de casos de uso

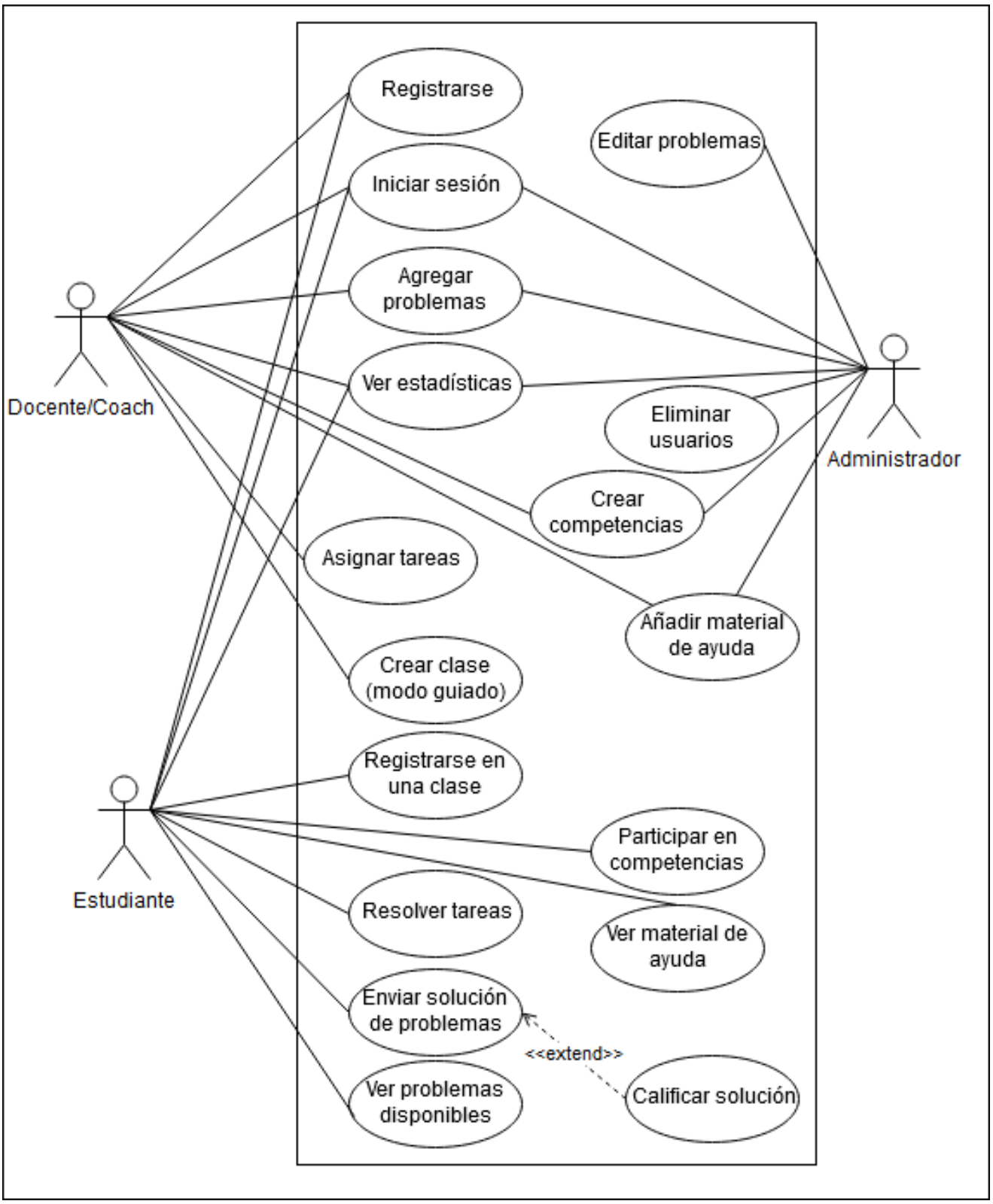
Módulos del sistema

A raíz de los requerimientos obtenidos se establece que la plataforma contendrá los siguientes módulos.

Módulo	**Descripción**

Usuarios	Gestiona los usuarios, registros, inicios de sesión y mecanismos de autenticación en la plataforma.
Materiales y guías	Gestiona los materiales escritos y audiovisuales de ayuda subidos a la plataforma.
Administración	Gestiona los mecanismos de administraciones, solo disponibles para usuarios de tipo administrador.
Estadísticas y rankings	Gestiona las estadísticas de usuarios de los estudiantes y los rankings comparativos de desempeño.
Problemas	Gestiona la creación, edición y solución de problemas dentro de la plataforma.
Syllabus	Gestiona las clases o modos guiados donde un docente o coach puede hacer seguimiento a sus estudiantes.
Competencias	Gestiona maratones de programación en tiempo real.

Diagrama general de casos de uso



Análisis del caso de uso: Registrarse

Descripción

Permite a un usuario estudiante o docente (coach) registrarse en la plataforma para acceder posteriormente a sus funcionalidades.

Flujo de eventos básico

1. El usuario ingresa en la interfaz de registro.
2. El usuario ingresa los datos solicitados completamente.
3. El usuario pulsa registrarse y envía los datos.
4. La plataforma verifica que los datos sean válidos.
5. La plataforma registra al usuario.
6. El usuario es redirigido al inicio de sesión.

Flujo de eventos alternativo

Datos incompletos:

1. El usuario ingresa en la interfaz de registro.
2. El usuario ingresa los datos incompletos.
3. El usuario pulsa registrarse.
4. La plataforma no envía los datos, sino que indica que hay campos incompletos.

Usuario ya registrado:

1. El usuario ingresa en la interfaz de registro.
2. El usuario ingresa los datos solicitados completamente.
3. El usuario pulsa registrarse y envía los datos.
4. La plataforma verifica los datos y encuentra que el email ya ha sido registrado.
5. La plataforma informa al usuario.

6. El usuario es redirigido al inicio de sesión.

Postcondiciones

El caso de uso registro es fundamental para todos los casos posteriores.

Análisis del caso de uso: Iniciar sesión

Descripción

Permite a usuarios de cualquier tipo ingresar a la plataforma con sus datos de acceso.

Flujo de eventos básico

1. El usuario ingresa a la interfaz de inicio de sesión.
2. El usuario escribe sus datos de acceso (O los toma del navegador si ha sido guardado previamente).
3. El usuario envía los datos.
4. La plataforma verifica los datos.
5. La plataforma redirige al usuario a su inicio dentro de la aplicación.

Flujo de eventos alternativo

Combinación de usuario/contraseña incorrecta:

1. El usuario ingresa a la interfaz de inicio de sesión.
2. El usuario escribe sus datos de acceso (O los toma del navegador si ha sido guardado previamente).
3. El usuario envía los datos.
4. La plataforma verifica que los datos no coinciden (correo no registrado o contraseña no coincide con el correo).
5. La plataforma informa al usuario del fallo.

Precondiciones

El usuario debe estar previamente registrado para poder iniciar sesión.

Postcondiciones

El sistema debe guardar la sesión del usuario para mantenerlo activo aún si hay recargas en la página.

Análisis del caso de uso: Agregar problemas

Descripción

Permite a un usuario del tipo docente o administrador añadir un nuevo problema en la plataforma.

Flujo de eventos básico

1. El usuario pulsa en agregar problemas.
2. El usuario añade los datos básicos del problema (Título, cuerpo, entradas y salidas de ejemplo)
3. El usuario añade la entrada de prueba que usará la plataforma para calificar el problema.
4. El usuario añade la salida de prueba que debe coincidir con la generada por las soluciones de los usuarios.
5. El usuario envía los datos.
6. La plataforma almacena el problema dejándolo accesible para enviar soluciones.

Flujo de eventos alternativo

Datos incompletos:

1. El usuario pulsa en agregar problemas.

2. El usuario añade los datos básicos del problema, omitiendo alguno o varios de ellos.
3. El usuario añade la entrada de prueba que usará la plataforma para calificar el problema.
4. El usuario añade la salida de prueba que debe coincidir con la generada por las soluciones de los usuarios.
5. El usuario envía los datos.
6. La plataforma informa que faltan datos.

Precondiciones

El usuario debe estar registrado e iniciar sesión como administrador o docente.

Postcondiciones

Una vez el problema es añadido debe estar inmediatamente disponible para su solución por parte de los estudiantes.

Análisis del caso de uso: Ver estadísticas

Descripción

Permite a usuarios de cualquier tipo ver las estadísticas (rankings) públicos dentro de la plataforma.

Flujo de eventos básico

1. El usuario pulsa en “ranking”.
2. La plataforma despliega un ranking de los usuarios, con controles para filtrar.

Análisis del caso de uso: Asignar tareas

Descripción

Un usuario de tipo docente/coach puede asignar ejercicios para que sus estudiantes los realicen en un plazo de tiempo determinado.

Flujo de eventos básico

1. El docente entra en la clase (modo guiado) al que desea asignar la tarea.
2. El docente pulsa en nueva tarea.
3. El docente ingresa los datos de fechas de apertura y cierre de la tarea.
4. El docente elige los ejercicios a agregar en la tarea.
5. El docente envía la tarea.
6. La plataforma almacena la tarea para desplegarla a los estudiantes.

Flujo de eventos alternativo

1. El docente entra en la clase (modo guiado) al que desea asignar la tarea.
2. El docente pulsa en nueva tarea.
3. El docente ingresa los datos de fechas de apertura y cierre de la tarea (uno de los dos datos incompletos).
4. El docente elige los ejercicios a agregar en la tarea (o deja en blanco el espacio.
5. El docente envía la tarea.
6. La plataforma informa que debe completar los campos para asignar la tarea.

Precondiciones

1. Solo un usuario docente puede añadir tareas.
2. Primero debe haber creado una clase, y la tarea la asigna a dicha clase.

Postcondiciones

La tarea se encuentra inmediatamente disponible para ser resuelta por los estudiantes inscritos en esa clase.

Análisis del caso de uso: Crear clase (modo guiado)

Descripción

Un docente puede crear una clase para compartir tareas y materiales con un conjunto específico de estudiantes.

Flujo de eventos básico

1. El docente se dirige a la sección clases.
2. El docente marca la opción nueva clase.
3. El docente proporciona la información solicitada.
4. El docente confirma la operación marcando enviar.
5. La plataforma guarda la nueva clase.

Precondiciones

El usuario debe estar registrado e iniciar sesión como docente.

Análisis del caso de uso: Registrarse a una clase

Descripción

Un usuario estudiante puede registrarse a una clase específica, bien sea de forma abierta o con una contraseña asignada por el docente en caso de ser una clase privada.

Flujo de eventos básico (Clase pública)

1. El estudiante entra en la sección clases
2. La plataforma despliega la lista de clases públicas y privadas disponibles.

3. El estudiante marca “registrarse” en una clase pública.
4. La plataforma redirige al estudiante a los materiales de la clase.

Flujo de eventos alternativo

Registro en clase privada

1. El estudiante entra en la sección clases
2. La plataforma despliega la lista de clases públicas y privadas disponibles.
3. El estudiante marca “registrarse” en una clase privada.
4. Se despliega un espacio para ingresar la clave de la clase.
5. El estudiante ingresa la clave.
6. La plataforma redirige al estudiante a los materiales de la clase.

Precondiciones

El usuario debe estar registrado y haber iniciado sesión como estudiante.

Postcondiciones

Para ver las tareas y materiales de una clase es necesario estar registrado.

Análisis del caso de uso: Resolver tareas

Descripción

Un estudiante puede enviar la solución a los problemas propuestos en una tarea.

Flujo de eventos básico

1. El estudiante entra en la clase específica.
2. El estudiante abre la tarea actual.
3. El estudiante abre uno de los problemas propuestos en la tarea.

4. El estudiante programa una solución.
5. El estudiante envía dicha solución.
6. La plataforma recibe la solución.
7. La plataforma envía al estudiante la calificación de dicha tarea.
8. Se repiten los pasos 3 al 7 para cada problema propuesto.

Precondiciones

El usuario debe estar registrado y haber iniciado sesión como estudiante, y estar matriculado en la clase específica.

Postcondiciones

Las soluciones correctas de los estudiantes se desplegarán al docente.

Análisis del caso de uso: Enviar solución de problemas

Descripción

Un estudiante puede enviar solución de los diferentes problemas disponibles en la plataforma en cualquier momento.

Flujo de eventos básico

1. El estudiante abre un problema específico.
2. El estudiante programa una solución para el problema y la guarda en un archivo único.
3. El estudiante selecciona el archivo en la plataforma.
4. El estudiante envía su solución.
5. La plataforma añade la solución a una cola de soluciones recibidas.

6. La plataforma califica la solución.
7. La plataforma informa al usuario la calificación obtenida.

Precondiciones

El usuario debe estar registrado y haber iniciado sesión como estudiante.

Postcondiciones

La calificación puede tardar en caso de congestión en la plataforma. Sin embargo, una vez sea calificado, se notificará al usuario en tiempo real, sin necesidad de recargar.

Análisis del caso de uso: Ver problemas disponibles

Descripción

Un usuario puede ver la lista de problemas disponibles en la plataforma.

Flujo de eventos básico

1. El estudiante ingresa en la sección problemas.
2. La plataforma despliega la lista de problemas con controles para filtrar.

Precondiciones

El usuario debe estar registrado y haber iniciado sesión (con cualquier rol)

Postcondiciones

Una vez listado el problema, un estudiante puede abrirlo y proceder a solucionarlo.

Análisis del caso de uso: Editar problemas

Descripción

Un usuario administrador puede editar problemas si se encuentran en ellos errores de redacción o en los casos de prueba.

Flujo de eventos básico

1. El administrador abre el problema en cuestión.
2. El administrador pulsar editar problema.
3. La plataforma carga el contenido del problema en forma de menú editable.
4. El administrador corrige la información necesaria.
5. El administrador guarda los cambios.
6. La plataforma almacena estos nuevos cambios.

Precondiciones

El usuario debe estar registrado y haber iniciado sesión como administrador.

Postcondiciones

El problema editado está disponible con sus cambios inmediatamente después de editarse.

Análisis del caso de uso: Eliminar usuario

Descripción

Un usuario administrador puede eliminar usuarios si detecta en ellos comportamientos abusivos (envío de altos volúmenes de soluciones spam, soluciones que contienen comportamientos maliciosos que buscan afectar la plataforma, entre otros) o si el propio usuario lo solicita.

Flujo de eventos básico

1. El administrador carga la lista de usuarios.

2. El administrador localiza al usuario malicioso.
3. El administrador selecciona eliminar usuario.
4. La plataforma pregunta si está seguro de eliminar dicho usuario.
5. El administrador confirma.
6. El usuario es borrado de la plataforma.

Precondiciones

El usuario debe estar registrado y haber iniciado sesión como administrador.

Postcondiciones

Inmediatamente eliminado, el usuario no podrá iniciar sesión en la plataforma.

Análisis del caso de uso: Crear competencias

Descripción

Un administrador o docente puede crear una maratón de programación en la plataforma.

Flujo de eventos básico

1. El usuario selecciona la opción maratones.
2. El usuario elige la opción crear nueva maratón.
3. El usuario elige los problemas que harán parte de la maratón.
4. El usuario activa la maratón.
5. La plataforma almacena la información.
6. La plataforma genera la competencia una vez iniciada.

Precondiciones

El usuario debe estar registrado y haber iniciado sesión como administrador o docente.

Análisis del caso de uso: Añadir Material de ayuda

Descripción

Los administradores y docentes pueden añadir material de ayuda (slides, videos, etc) a la plataforma para guiar a los estudiantes.

Flujo de eventos básico

1. El usuario entra a la opción añadir material.
2. El usuario marca la información secundaria del material.
3. El usuario adjunta el link del material.
4. La plataforma verifica la información.
5. La plataforma añade el material.

Precondiciones

El usuario debe estar registrado e iniciar sesión como docente o administrador.

Postcondiciones

El material adjunto estará disponible para los estudiantes inmediatamente después de ser añadido.

Análisis del caso de uso: Participar en competencias

Descripción

Un usuario estudiante puede participar en competencias y enviar soluciones en ellas.

Flujo de eventos básico

1. El estudiante se registra en la competencia.
2. La plataforma muestra la interfaz de competencia.
3. El estudiante elige uno de los problemas.

4. La plataforma despliega dicho problema.
5. OPCIONAL: El estudiante programa una solución a este problema.
6. OPCIONAL: El estudiante envía la solución a este problema.
7. La plataforma califica su solución.
8. La plataforma actualiza el ranking de la competencia.
9. Se repite el proceso del 3 al 9 tantas veces como el estudiante desee.

Precondiciones

El usuario debe estar registrado y haber iniciado sesión como estudiante.

Análisis del caso de uso: Ver material de ayuda

Descripción

El estudiante puede ver el material de ayuda disponible en la plataforma.

Flujo de eventos básico

1. El estudiante entra en la sección material.
2. El estudiante elige un material.
3. La plataforma despliega en pantalla el material.

Precondiciones

El usuario debe estar registrado y haber iniciado sesión como estudiante.

Análisis del caso de uso: Calificar solución

Descripción

La plataforma debe ser capaz de analizar la solución del usuario, ejecutarla en un ambiente seguro e indicarle si es correcto o incorrecto.

Flujo de eventos básico

1. La plataforma obtiene la primera solución pendiente de la cola de soluciones enviadas por los estudiantes.
2. La plataforma copia la entrada y la salida de ese problema a un entorno seguro.
3. La plataforma copia el código de la solución al mismo entorno seguro.
4. La plataforma compila el código.
5. La plataforma ejecuta la solución en el entorno seguro con las entradas indicadas.
6. La plataforma compara las diferencias entre el archivo de salida original y el generado para ver si coinciden.
7. La plataforma emite una calificación válida o inválida según su coincidencia.
8. La plataforma elimina los archivos copiados del entorno seguro.

Flujo de eventos alternativo

Solución que excede el tiempo límite:

1. La plataforma obtiene la primera solución pendiente de la cola de soluciones enviadas por los estudiantes.
2. La plataforma copia la entrada y la salida de ese problema a un entorno seguro.
3. La plataforma copia el código de la solución al mismo entorno seguro.
4. La plataforma compila el código.
5. La plataforma ejecuta la solución en el entorno seguro con las entradas indicadas.
6. El código tarda más tiempo del aceptado para ese ejercicio en arrojar una respuesta.
7. La plataforma cancela la ejecución del archivo.
8. La plataforma emite una calificación de “Tiempo límite excedido”.
9. La plataforma elimina los archivos copiados del entorno seguro.

Solución con error de compilación:

1. La plataforma obtiene la primera solución pendiente de la cola de soluciones enviadas por los estudiantes.
2. La plataforma copia la entrada y la salida de ese problema a un entorno seguro.
3. La plataforma copia el código de la solución al mismo entorno seguro.
4. La plataforma intenta compilar el código.
5. La plataforma obtiene un error al compilar.
6. La plataforma emite una calificación de “Error de compilación”.
7. La plataforma elimina los archivos copiados del entorno seguro.

Solución con error en tiempo de ejecución:

1. La plataforma obtiene la primera solución pendiente de la cola de soluciones enviadas por los estudiantes.
2. La plataforma copia la entrada y la salida de ese problema a un entorno seguro.
3. La plataforma copia el código de la solución al mismo entorno seguro.
4. La plataforma compila el código.
5. La plataforma ejecuta la solución en el entorno seguro con las entradas indicadas.
6. La ejecución se rompe sin arrojar un retorno correcto.
7. La plataforma emite una calificación de “Error en tiempo de ejecución”.
8. La plataforma elimina los archivos copiados del entorno seguro.

Precondiciones

Se ejecuta mientras haya soluciones en la cola.

Anexo 8. Modelo de diseño y arquitectura UFPS Training Center

Introducción

El presente documento expone y define el modelo de diseño propuesto en el desarrollo de la plataforma de software UFPS Training Center, la cual hace parte del proyecto de grado "Desarrollo e implementación de un marco de trabajo para el grupo de estudio en Programación competitiva del programa Ingeniería de Sistemas UFPS".

Alcance

El presente documento cubre únicamente los aspectos relacionados con el proceso de diseño, definición de la arquitectura, modelos de clases y datos. Como tal, el documento es parte de una documentación más completa que cubre las demás fases de desarrollo, y se distribuyen de manera digital como un anexo al informe final del proyecto.

Esta plataforma de software está planeada para iniciar su funcionamiento en el primer semestre del 2018 como parte de los trabajos del grupo de estudio, una vez este inicie sus funciones. Sin embargo, la plataforma hace parte de un marco de trabajo con más componentes metodológicos, digitales y físicos que ya se encuentran en funcionamiento. Para más detalles de los demás componentes y de la integración de la plataforma con ellos, ver el documento principal del proyecto.

Definiciones, acrónimos y abreviaturas

- **Aplicación Backend:** Aplicación ejecutada en el servidor.
- **Aplicación Frontend:** Aplicación para el cliente.

- **ICPC:** International Collegiate Programming Contest, Maratón de Programación Oficial Universitaria más importante a nivel mundial.
- **Maratón de Programación:** Competencia donde se ponen en prueba las habilidades algorítmicas y de trabajo en equipo de los estudiantes en un entorno controlado.
- **SPA:** Single Page Application, aplicación web completa que se ejecuta a través de una única página.
- **UFPS Training Center:** Plataforma de entrenamiento en Programación Competitiva para los estudiantes de la UFPS.

Arquitectura del sistema

Diagrama general de arquitectura

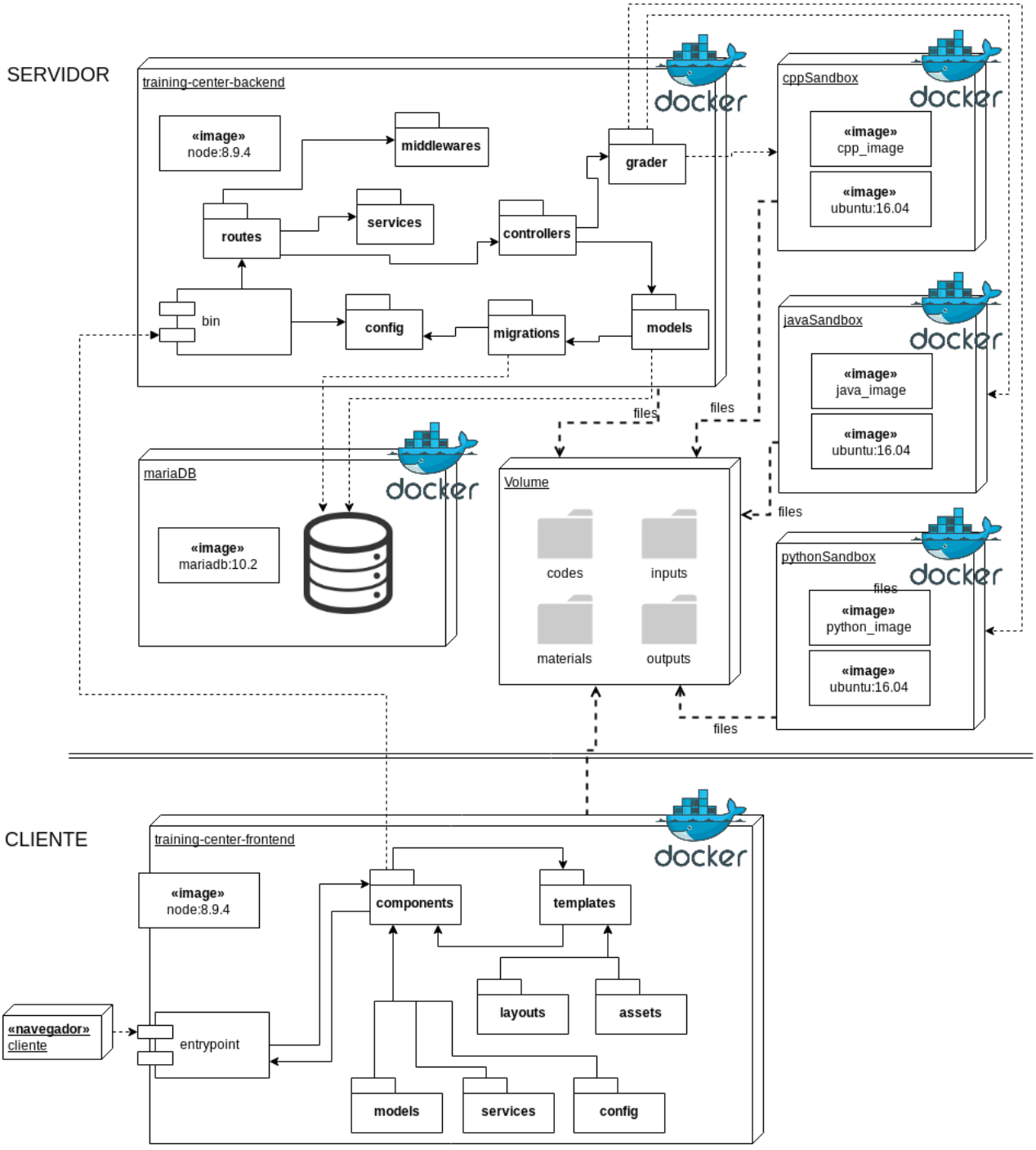


Figura 1. Diagrama arquitectural del sistema (para ver en mejor calidad, abrir el anexo digital "diagrama arquitectural.png")

Especificación general de la arquitectura

La plataforma está diseñada bajo una arquitectura de cliente - servidor (dos capas). Cada una de las capas es diseñada como una aplicación completa y funcional, con interdependencia una de la otra. A continuación, se describe el funcionamiento de cada capa.

Capa servidor

El servidor es el encargado de procesar la información enviada desde el cliente y devolver las respuestas que sean necesarias. Esta capa contiene en su interior varios componentes interrelacionados entre sí:

- **training-center-backend:** El núcleo central de este servidor es el "training-center-backend", aplicación que funciona como eje central de procesamiento obteniendo la información necesaria de los demás componentes y que mantiene una constante comunicación con la aplicación cliente. A continuación, se definen los subcomponentes de esta aplicación, para posteriormente definir su funcionamiento en conjunto:
    - bin: Se encarga de recibir las peticiones desde el cliente y devolver las respuestas.
    - router: Conjunto de utilidades para analizar las peticiones recibidas y "enrutarlas" hacia el componente encargado de su procesamiento.
    - middleware: Son pequeños bloques de código que se ejecutan entre la petición que hace el usuario hasta que la petición llega al servidor, definiendo el flujo y restricciones de la aplicación.
    - services: Utilidades genéricas que pueden ser de ayuda en diferentes operaciones y se encapsulan para evitar redundancia de código.
    - controllers: Lógica de la aplicación.

- config: Archivos de configuración de la aplicación.
- models: Modelos de datos (en este punto se gestiona la conexión con la base de datos)
- migrations: componente para la gestión de la base de datos.
- grader: componente encargado de comunicarse con los contenedores "sandboxing" para la ejecución de las soluciones de los estudiantes.

En conjunto, el funcionamiento del backend es el siguiente: "bin" se encuentra permanentemente escuchando peticiones desde el cliente. Al llegar una petición, esta es capturada por un router específico que la redirige al controlador en el que debe procesarse, utilizando los servicios que sean necesarios, y en el proceso de redirección la solicitud cruza los middlewares para definir entre otros aspectos, que el usuario esté autorizado. En caso de ser necesario, el controlador se comunica con los modelos (models), los cuales a través de migrations obtienen y actualizan la información almacenada en la base de datos. Durante todo el proceso, se toman en cuenta los parámetros establecidos globalmente en config.

Un caso especial ocurre cuando la solicitud recibida es un ejercicio para ser calificado. En este caso, el controlador llama al grader, el cual se comunicará con el entorno seguro específico para el lenguaje de la solución que se encargará de su ejecución y calificación.

- **cppSandbox:** Contenedor Docker que funciona a modo de sandbox para ejecutar en entorno seguro las soluciones de los estudiantes recibidas en lenguaje C++. Este container cuenta con el compilador GNU G++ en su versión 5.4.0 (Este compilador se

encuentra bajo licencia libre GPLv3). Recibe las peticiones desde el grader y toma la solución y las entradas desde las carpetas compartidas en “volume”.

- **javaSandbox:** Contenedor Docker que funciona a modo de sandbox para ejecutar en entorno seguro las soluciones de los estudiantes recibidas en lenguaje java. Este container cuenta con el compilador incluido en el JDK (Java Development Kit) en su versión 8 (Compartido bajo licencia libre Oracle Binary Code License). Recibe las peticiones desde el grader y toma la solución y las entradas desde las carpetas compartidas en “volume”.
- **pythonSandbox:** Contenedor Docker que funciona a modo de sandbox para ejecutar en entorno seguro las soluciones de los estudiantes recibidas en lenguaje python. Este container cuenta con el compilador python en su versión 3 (Compartido bajo licencia libre GPL). Recibe las peticiones desde el grader y toma la solución y las entradas desde las carpetas compartidas en “volume”.
- **mariaDB:** Gestor de base de datos. La plataforma cuenta con una base de datos que almacena toda la información no estática de la aplicación usando este gestor.
- **volume:** Es un espacio de almacenamiento compartido. Aquí se almacenan los archivos estáticos de la plataforma, y de allí son tomados los códigos, entradas y salidas por los contenedores para realizar las calificaciones.

Capa cliente

La capa del cliente es una aplicación web completa que se ejecuta en cualquier navegador web moderno. Cuenta con los siguientes componentes:

- **entrypoint (index.html)**: Punto de comunicación entre el navegador y la aplicación. Todas las solicitudes e interacciones de la aplicación son hechas a través de este componente.
- **components:** Los componentes (también llamados módulos) definen la lógica de la aplicación en el cliente. Se comunican en tiempo real con los templates.
- **templates:** Son unidades de diseño de interfaz de usuario conectados a los componentes para tener la información actualizada en tiempo real.
- **layouts:** Los layouts definen plantillas de diseño que pueden ser reutilizadas por diferentes templates.
- **assets:** Imágenes, estilos, y demás componentes gráficos.
- **models:** Los modelos permiten mantener una estructura de objetos en la aplicación cliente.
- **services:** Servicios que permiten encapsular tareas repetitivas para evitar duplicaciones de código.
- **config:** Configuraciones generales de la aplicación.

Estos componentes se relacionan de la siguiente manera: El usuario ingresa a la aplicación a través del entrypoint, que inmediatamente lo redirige al componente solicitado (mediante un enrutamiento transparente realizado por las mismas configuraciones de los componentes). Cada componente define una lógica del negocio y tiene adjunto un template con el cual realiza un "data binding" que permite comunicación en tiempo real: Los cambios realizados en el componente luego de un procesamiento o de una conexión al servidor se despliegan inmediatamente en la interfaz, mientras los cambios ingresados por el usuario al template se actualizan inmediatamente al componente.

Para esta labor el componente se basa en los models que definen el esqueleto de la aplicación, y los services que son utilidades transversales a la aplicación. Por su parte, los templates se ayudan de los layouts y los assets para desplegar la información correctamente y de forma agradable al usuario.

Toda la ejecución sigue los parámetros establecidos en config.

Cabe destacar que la comunicación con el servidor ocurre en los componentes, aunque para una mayor modularidad y simplicidad, generalmente estos componentes se apoyan en servicios para realizar estas tareas.

Comunicación entre cliente y servidor

La comunicación entre cliente y servidor se realiza utilizando REST (transferencia de estado representacional) utilizando JSON como el formato de medios transmitidos.

Esto nos permite una separación e interdependencia sencilla entre el cliente y el servidor, facilidad para posibles migraciones y en caso de posteriores aplicaciones (por ejemplo, aplicaciones móviles para complementar la plataforma) muy fácilmente podrían conectarse al mismo servidor, siendo necesario solo el desarrollo del cliente.

Por su parte, desde el navegador todas las conexiones se realizan utilizando la interfaz Fetch de javascript, en lugar del clásico XMLHttpRequest. Si bien Fetch no tiene compatibilidad con navegadores antiguos (especialmente internet explorer), utilizar fetch trae como ventaja fundamental su apoyo directo en “promesas” que nos permite simplificar la aplicación en momentos de espera de las conexiones. Sin embargo, para no perder del todo la compatibilidad, se utiliza un polyfill desarrollado por github y compartido bajo código libre que permite la utilización de fetch en navegadores que no lo soportan nativamente.

Arquitectura encapsulada

En el diagrama de arquitectura general se puede observar como los componentes (excepto volumes que es un folder de intercambio) se encuentran definidos como contenedores docker. Esto permite encapsular fácilmente los diferentes componentes, permitiendo un despliegue mas simple y controlado bajo cualquier hardware sin necesidad de excesivas configuraciones.

De esta manera, cada componente resulta en un contenedor independiente, que puede comunicarse con los demás con facilidad, pero abstrayendo sus operaciones a su espacio contenido. Del mismo modo, esto permite una mayor facilidad para migración.

Escalabilidad

Hay dos cuestiones de escalabilidad que hemos priorizado desde el inicio de la plataforma. En primer lugar, si bien el flujo de usuarios se encuentra acotado por los estudiantes del grupo de estudio, o en el mejor de los casos de la carrera y estos no son números muy altos, si debo considerarse que los usuarios utilizarán la plataforma generando operaciones pesadas (calificación de sus soluciones) que pueden tardar varios segundos de procesamiento. Estos procesamientos de larga duración no deben afectar el rendimiento de la plataforma. En segundo lugar, aunque originalmente se diseña la plataforma para funcionar con java, C++ y Python por ser los lenguajes actualmente aceptados en competencias, sabemos que a futuro podría ser necesario incorporar nuevos lenguajes, bien sea porque las competencias de programación hacen oficial un nuevo lenguaje aceptado, o porque en la Universidad se desea trabajar un lenguaje diferente. Por tanto, surge la necesidad de integrar fácilmente nuevos lenguajes a la plataforma.

La primera cuestión es inmediatamente resuelta por la arquitectura, en combinación con el paradigma de node.js (se explicará más adelante). Al separar el frontend del backend, cada

usuario puede acceder a la aplicación en su computador sin importar la saturación que esté ocurriendo en el servidor. Pero no solo esto, sino que la aplicación del servidor también está separada (por un lado, el backend procesando las comunicaciones, y por otro los contenedores sandbox ejecutando las soluciones de los usuarios). Esta separación permite que el backend siga respondiendo las peticiones de los usuarios mientras la saturación queda únicamente en las calificaciones. Así los demás usuarios podrán seguir leyendo problemas, solucionándolos y utilizando la plataforma sin problema. Inevitablemente esto implica que las calificaciones si pueden tardar unos segundos debido a la saturación, pero gracias al enfoque el usuario puede seguir trabajando normalmente mientras espera su calificación, y una vez esta es obtenida se despliega en tiempo real en su interfaz.

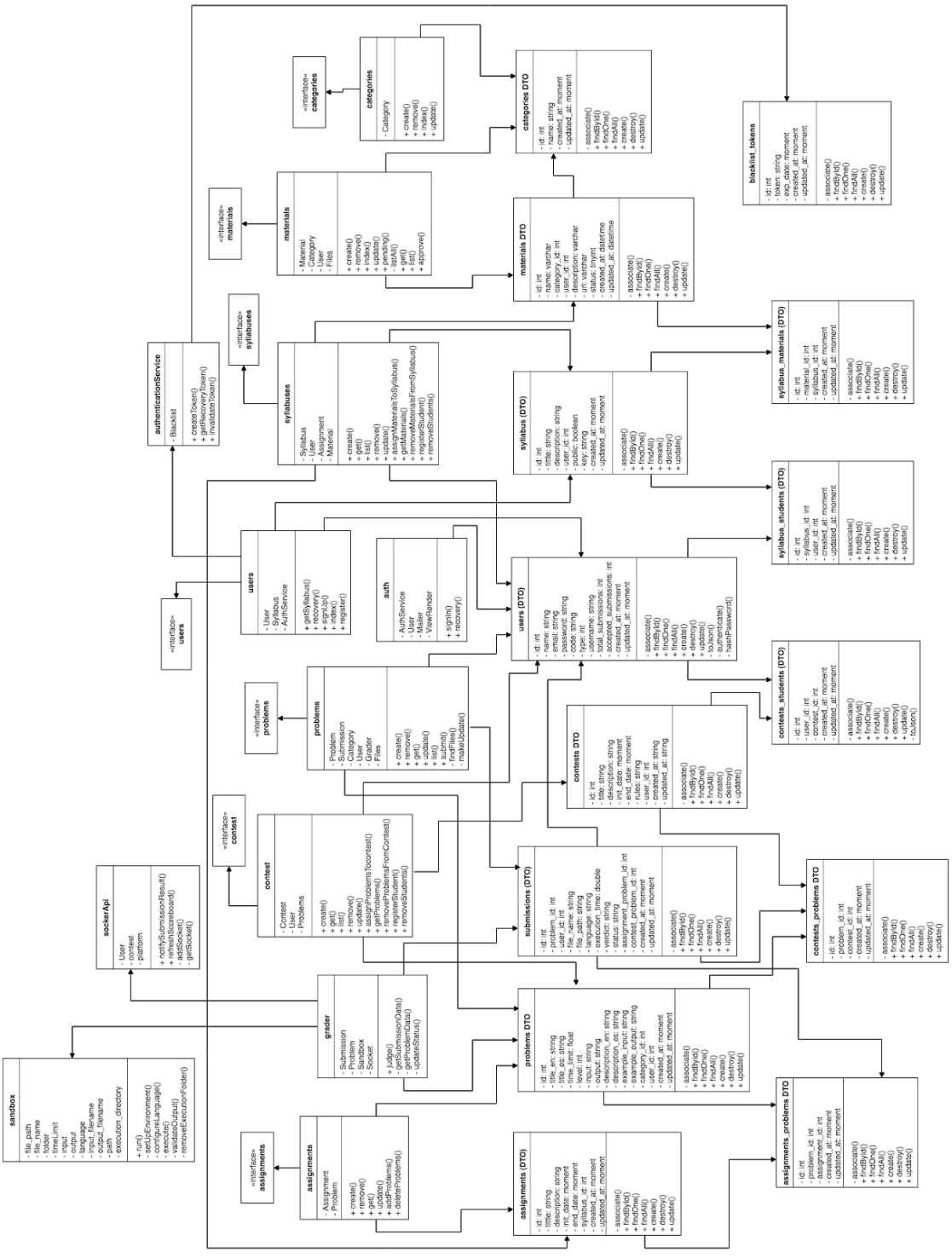
La segunda cuestión es aún más arquitectural: Un primer enfoque permitiría tener un único contenedor sandbox de calificación con las herramientas necesarias para la calificación en todos los lenguajes disponibles. Sin embargo, en el momento de intentar añadir nuevos lenguajes, inevitablemente sería necesario una reestructuración de todo el código para añadir las nuevas herramientas e instrucciones que el lenguaje requiera. Esto implica un gasto considerable de tiempo y recursos, así como mayores riesgos de errores inducidos en el cambio. Justo por eso se plantea un contenedor independiente por cada lenguaje. De esta forma, en el momento de añadir un nuevo lenguaje, basta con crear un nuevo contenedor con sus configuraciones y archivos necesarios. Posteriormente se configura el componente grader para indicarle que hay un nuevo lenguaje disponible, e inmediatamente después puede someterse a pruebas, pues ya estará listo para usar.

Herramientas

Las herramientas a utilizar en la construcción de UFPS Training Center son las siguientes:

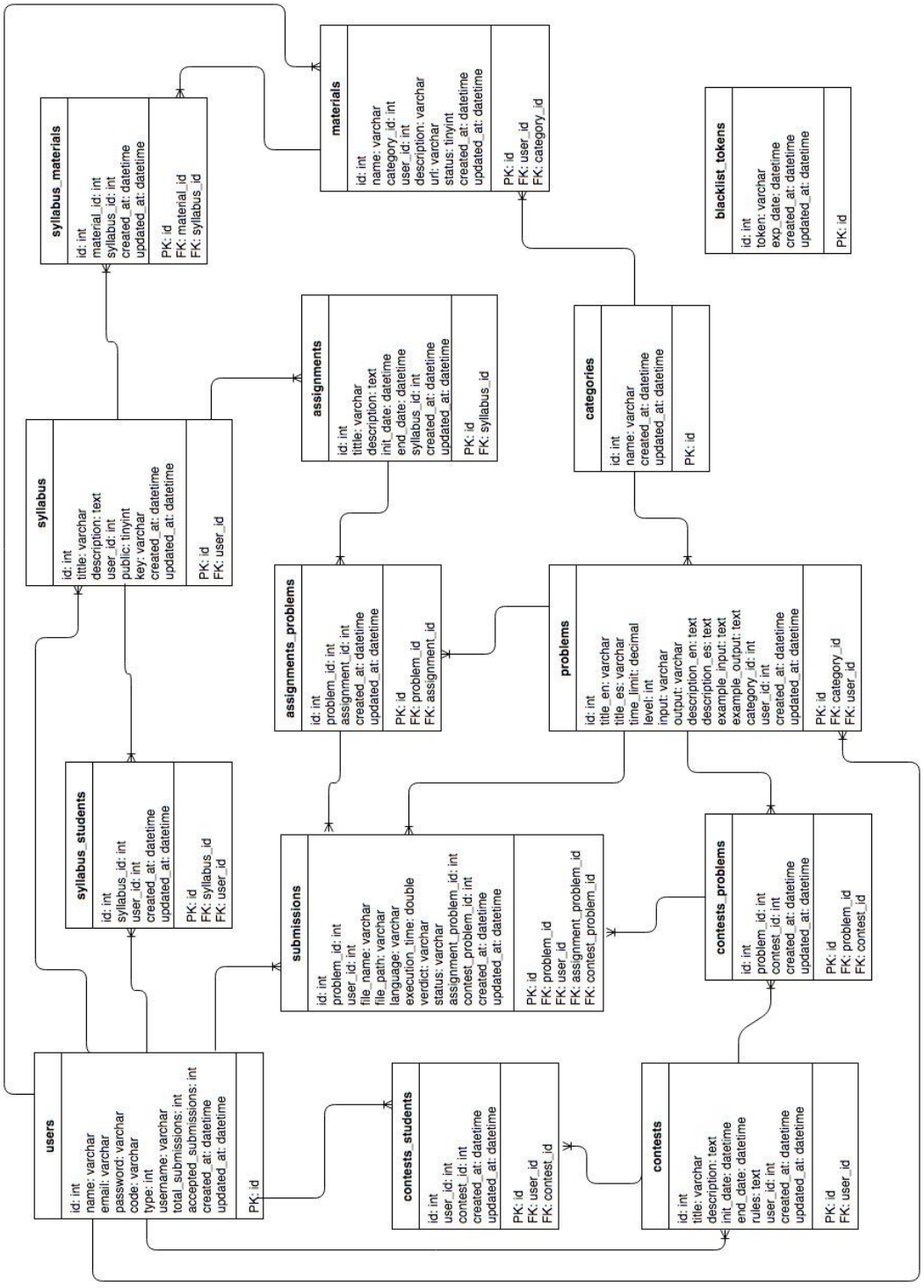
- **Javascript**: Lenguaje de programación usado en el cliente y el servidor.
- **Node.js:** Entorno de ejecución para javascript en el servidor. Es elegido porque su modelo de operaciones E/S sin bloqueo y su orientación a eventos encaja a la perfección con las necesidades de la aplicación.
- **Express.js:** Framework API REST para la aplicación backend.
- **Docker:** Contenedores de software que aportan una capa adicional de abstracción. En el proyecto, docker tiene dos usos diferentes: En primer lugar para encapsular los diferentes componentes, y en segundo lugar como un entorno seguro de ejecución.
- **GNU G++:** Compilador de código libre desarrollado por GNU para C++. Este es utilizado para calificar las soluciones C++ enviadas por los estudiantes.
- **JDK:** Java Development Kit, herramientas para la compilación y ejecución de Java.
- **Python:** Compilador para el lenguaje del mismo nombre, utilizado para calificar las soluciones python enviadas por los estudiantes.
- **Bash:** Lenguaje de consola, utilizado para la comunicación entre la plataforma y el calificador de soluciones enviadas por los estudiantes.
- **time y timeout:** Dos utilidades linux desarrolladas por GNU y compartidas bajo código libre, utilizadas para saber el tiempo de ejecución de las soluciones y para cortar su ejecución si exceden el tiempo especificado.
- **Aurelia:** Framework para la realización de la aplicación cliente bajo el esquema SPA (Single Page Applications).
- **Socket.io:** Comunicación en tiempo real para enviar calificaciones.
- **Sequelize:** ORM para Node.js.

Modelo de clases



Se recomienda visualizar este diagrama directamente como imagen entre los anexos digitales para una mejor visualización.

5. Modelo de datos



Se recomienda visualizar este diagrama directamente como imagen entre los anexos digitales para una mejor visualización.

Diccionario de datos

Tabla	users		
Descripción	Usuarios de la plataforma		
Campo	**Tipo**	**Adicional**	**Descripción**
id	int	PK	Identificador del usuario.
name	varchar		Nombre del usuario.
email	varchar		Email del usuario.
password	varchar		Contraseña del usuario (guardada encriptada con SALT HASH)
code	varchar		Código del usuario en la Universidad.
type	int		0: estudiante 1: docente/coach 2: administrador
username	varchar		Nombre del usuario para mostrar.
total_submissions	int		Total de soluciones enviadas.
accepted_submissions	int		Total de soluciones correctas.
created_at	datetime		Fecha de creación del usuario.
updated_at	datetime		Fecha de actualización del usuario.

Tabla	contest_students		
Descripción	Tabla de transición que relaciona los estudiantes inscritos en una maratón específica.		
Campo	**Tipo**	**Adicional**	**Descripción**
id	int	PK	Identificador de la relación.
user_id	int	FK	Identificador del usuario.
contest_id	int	FK	Identificador de la maratón.
created_at	datetime		Fecha de creación del registro.

updated_at	datetime		Fecha de última actualización del registro.

Tabla	contest		
Descripción	Maratón de programación		
Campo	**Tipo**	**Adicional**	**Descripción**
id	int	PK	identificador de la maratón
title	varchar		Título de la maratón.
description	varchar		Descripción de la maratón.
init_date	datetime		Fecha de inicio de la maratón.
end_date	datetime		Fecha de finalización de la maratón.
rules	text		Reglas de la maratón
user_id	int	FK	Usuario creador de la maratón.
created_at	datetime		Fecha de creación del registro.
updated_at	datetime		Fecha de última actualización del registro.

Tabla	submissions		
Descripción	Soluciones enviadas a la plataforma.		
Campo	**Tipo**	**Adicional**	**Descripción**
id	int	PK	Identificador de la solución.
problem_id	int	FK	Identificador del problema solucionado.
user_id	int		Identificador del estudiante que envía la solución.
file_name	varchar		Nombre del archivo enviado.
file_path	varchar		Ruta al archivo enviado.
language	varchar		Nombre del lenguaje de programación.
execution_time	double		Tiempo de ejecución de la solución.
veredict	varchar		Calificación dada por la plataforma.

status	varchar		Status del problema.
assignment_problem_id	int	FK	Identificador del problema en la tarea (si aplica).
contest_problem_id	int	FK	Identificador del problema en el contest (si aplica).
created_at	datetime		Fecha de creación del registro.
updated_at	datetime		Fecha de última actualización del registro.

Tabla	contest_problems		
Descripción	Tabla de transición que asocia una maratón de programación con sus problemas.		
Campo	**Tipo**	**Adicional**	**Descripción**
id	int	PK	Identificador de la relación.
problem_id	int	FK	Identificador del problema.
contest_id	int	FK	Identificador de la maratón.
created_at	datetime		Fecha de creación del registro.
updated_at	datetime		Fecha de última actualización del registro.

Tabla	syllabus_students		
Descripción	Tabla de transición que asocia clases con los estudiantes matriculados en ella.		
Campo	**Tipo**	**Adicional**	**Descripción**
id	int	PK	Identificador de la relación.
syllabus_id	int	FK	Identificador de la clase.
user_id	int	FK	Identificador del estudiante.
created_at	datetime		Fecha de creación del registro.
updated_at	datetime		Fecha de última actualización del registro.

Tabla	assignments_problems		
Descripción	Tabla de transición para asociar tareas con sus problemas.		
Campo	**Tipo**	**Adicional**	**Descripción**
id	int	PK	Identificador de la relación.
problem_id	int	FK	Identificador del problema.
assignment_id	int	FK	Identificador de la tarea.
created_at	datetime		Fecha de creación del registro.
updated_at	datetime		Fecha de última actualización del registro.

Tabla	problems		
Descripción	Problemas almacenados en la plataforma.		
Campo	**Tipo**	**Adicional**	**Descripción**
id	int	PK	Identificador del problema.
title_en	varchar		Título en inglés.
title_es	varchar		Título en español.
time_limit	decimal		Tiempo límite del problema.
level	int		Nivel entre 1 (muy facil) y 10 (muy dificil).
input	varchar		Ruta a la entrada del problema.
output	varchar		Ruta a la salida del problema.
description_en	varchar		Descripción del problema en inglés.
description_es	varchar		Descripción del problema en español.
example_input	text		Entrada de ejemplo.
example_output	text		Salida de ejemplo.
category_id	int	FK	Identificador de la categoría del problema.
user_id	int	FK	Identificador del autor del problema
created_at	datetime		Fecha de creación del registro.

updated_at	datetime		Fecha de última actualización del registro.

Tabla	syllabus		
Descripción	Clases almacenadas en la plataforma		
Campo	**Tipo**	**Adicional**	**Descripción**
id	int	PK	Identificador de la clase.
title	varchar		Título de la clase.
description	varchar		Descripción de la clase.
user_id	int	FK	Identificador del docente/coach creador de la clase.
public	tinyint		0: clase privada 1: clase pública
key	varchar		En caso de ser privada, el docente/coach asigna una clave que los estudiantes deben conocer para acceder. Esta se guarda en este campo.
created_at	datetime		Fecha de creación del registro.
updated_at	datetime		Fecha de última actualización del registro.

Tabla	assignments		
Descripción	Tareas almacenadas en la plataforma		
Campo	**Tipo**	**Adicional**	**Descripción**
id	int	PK	Identificador de la tarea.
title	varchar		Título de la tarea.
description	varchar		Descripción de la tarea.
init_date	datetime		Fecha de inicio de la tarea.
end_date	datetime		Fecha de finalización de la tarea.
syllabus_id	int	FK	Syllabus al cual pertenece la tarea.
created_at	datetime		Fecha de creación del registro.
updated_at	datetime		Fecha de última actualización del registro.

Tabla	syllabus_materials		
Descripción	Tabla de transición que asocia materiales con el syllabus al que pertenecen		
Campo	**Tipo**	**Adicional**	**Descripción**
id	int	PK	Identificador de la relación.
material_id	int	FK	Identificador del material.
syllabus_id	int	FK	Identificador del syllabus.
created_at	datetime		Fecha de creación del registro.
updated_at	datetime		Fecha de última actualización del registro.

Tabla	categories		
Descripción	Categorías de ejercicios y materiales		
Campo	**Tipo**	**Adicional**	**Descripción**
id	int	PK	Identificador de la categoría.
name	varchar		Nombre de la categoría.
created_at	datetime		Fecha de creación del registro.
updated_at	datetime		Fecha de última actualización del registro.

Tabla	materials		
Descripción	Materiales disponibles en la plataforma.		
Campo	**Tipo**	**Adicional**	**Descripción**
id	int	PK	Identificador del material.
name	varchar		Nombre del material.
category_id	int	FK	Id de la categoría a la cual pertenece el material.
user_id	int	FK	Id del usuario que subió el material a la plataforma.
description	varchar		Descripción del material.
url	varchar		URL del material.

status	tinyint		1: Interno 2: Externo
created_at	datetime		Fecha de creación del registro.
updated_at	datetime		Fecha de última actualización del registro.

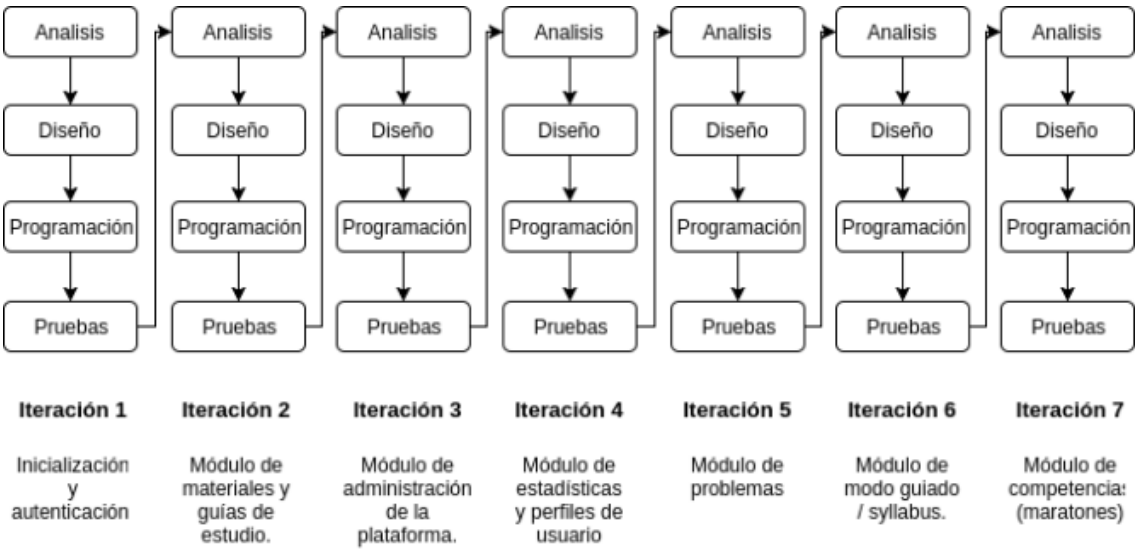
Tabla	blacklist_tokens		
Descripción	Lista de tokens de usuario que han sido revocados.		
Campo	**Tipo**	**Adicional**	**Descripción**
id	int	PK	Id del token
token	varchar		Token JWT.
exp_date	datetime		Fecha de vencimiento del token.
created_at	datetime		Fecha de creación del registro.
updated_at	datetime		Fecha de actualización del registro.

Anexo 9. Metodología de desarrollo del UFPS Training Center

La plataforma de software “UFPS Training Center” hace parte del proyecto de grado “Desarrollo e implementación de un marco de trabajo para el grupo de estudio en programación competitiva”. Este proyecto de grado cuenta con una metodología definida en los documentos del proyecto. A continuación, se detallará específicamente la metodología de desarrollo de software utilizada en la creación de la plataforma software.

Luego de analizar los requerimientos, se puede observar como el software necesario se estructura a lo largo de una serie de módulos (autenticación, problemas, material, modo guiado, administración, competencias, estadísticas/perfiles) que pueden desarrollarse de forma escalonada e irse añadiendo al software a medida que se realizan. Una vez se añade un nuevo módulo, pueden realizarse sobre el todo tipo de pruebas de funcionamiento, para, una vez integrado por completo, continuar con la creación del siguiente módulo.

Por tanto, la metodología elegida para realizar esta herramienta es una metodología iterativa e incremental.



La metodología iterativa e incremental nos permite ir realizando nuevos incrementos sobre la plataforma que han sido completamente testeados antes de pasar a la siguiente fase. Como se puede ver en el diagrama anterior, se han planteado 7 iteraciones, en cada una de las cuales se agrega un módulo hasta llegar a la iteración 7 en donde se encuentra la plataforma completa desarrollada.

Anexo 10. Ficha técnica del software UFPS Training Center

PRODUCTO			
CARACTERÍSTICAS DEL PRODUCTO			
Nombre del Producto	UFPS Training Center		
Proyecto padre	Desarrollo e implementación de un marco de trabajo para el grupo de estudio en programación competitiva.		
Versiones Anteriores	-	**Versión Actual**	1.0.0
DESCRIPCIÓN DEL PRODUCTO			
Descripción General del Producto	Plataforma para el entrenamiento de programación competitiva de los estudiantes del grupo de estudio en programación competitiva.		
Objetivo	Mejorar los resultados de los estudiantes UFPS en competencias de programación competitiva por medio de una plataforma ágil y completa.		
ARQUITECTURA			
Descripción	Aplicación de cliente /servidor. El cliente se realiza siguiendo un enfoque multicapa y el cliente bajo un enfoque MVVM		
REQUERIMIENTOS DEL PRODUCTO			
Requisitos del Sistema (Servidor)			
Hardware	Requisitos mínimos: Memoria RAM: 1GB CPU: 1GHz Disco duro: 20 GB Transferencia: 20GB/Mes		
Software	Sistema operativo Linux (En Windows la plataforma funciona, pero no el calificador) Node.js NPM MariaDB		

	Docker Bash
Requisitos del Sistema (Cliente)	
Hardware	Requisitos mínimos: Memoria RAM: 512mb CPU: 512MHz
Software	Navegador web moderno. Garantizado el funcionamiento en: Microsoft Edge, Mozilla Firefox, Google Chrome, Opera, Maxthon y Vivaldi en sus versiones más recientes. Demás navegadores modernos con soporte para HTML5 deben funcionar correctamente. No funciona en Internet Explorer 8 y versiones anteriores. En versiones mas recientes funciona, con algunos detalles de visualización de interfaz. Se recomienda su utilización en computadores en lugar de dispositivos móviles.
Otros	Es necesaria una conexión a internet.
REQUERIMIENTOS	
Requerimientos Funcionales Generales	• Listar problemas • Agregar problemas • Calificar problemas • Corregir problemas • Eliminar problemas • Filtrar problemas • Categorizar problemas • Ver ranking • Crear competencias • Añadir material de ayuda

	• Ver material de ayuda • Crear modo guiado • Recibir solución de problemas • Ver estadísticas • Gestionar usuarios
Requerimientos Adicionales	• Los envíos de soluciones no deben afectar la estabilidad del sistema. • Las contraseñas de los usuarios deben almacenarse encriptadas. • Las soluciones enviadas por los usuarios deben ejecutarse en modo “sandbox”. • El sistema debo funcionar en cualquier plataforma moderna.
Requisitos Legales Aplicables	• Cualquier usuario puede eliminar su cuenta y toda su información a la plataforma realizando esta petición al administrador cumpliendo con la ley de habeas data.
Clientes del producto	Grupo de estudio en programación competitiva UFPS