UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 4

CONCEPCIÓN Y DESARROLLO DE UN MÓDULO PARA LA DETECCIÓN DE PLAGIO EN EL JUEZ EN LÍNEA CARIBEÑO

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas

Autores: Yisel Quiñones Guerrero

Leandro González Vallejo

Tutores: Ing. Enrique José Altuna Castillo

Lic. Tomás Orlando Junco Vázquez

La Habana, Cuba, junio de 2012

"Año 54 de la Revolución"

Declaración de Autoría

Declaramos que somos los únicos autores del trabajo **“Concepción y desarrollo de un módulo para la detección de plagio en el Juez en Línea Caribeño”** y autorizamos a la Facultad 4 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los ____ días del mes de _____ del año _____.

Autor: Yisel Quiñones Guerrero

Autor: Leandro González Vallejo

Tutor: Ing. Enrique José Altuna Castillo

Tutor: Lic. Tomás Orlando Junco Vázquez

"El secreto del éxito es la constancia en el propósito."

Benjamín Disraeli

"La virtud, como el arte, se consagra constantemente a lo que es difícil de hacer, y cuanto más dura es la tarea, más brillante es el éxito."

Aristóteles

"La vida sin ideas de nada vale. No hay felicidad mayor que la de luchar por ellas."

Fidel Castro Ruz

Este trabajo está dedicado:

A nuestros padres, que son nuestra vida y razón de ser. Por su apoyo, amor y confianza.

A nuestras familias que son una sola.

Agradecimientos

Son muchas las personas que han colaborado con este trabajo. Agradecemos a nuestros tutores Tomás y Enrique por señalarnos el camino y guiarnos en la travesía, sin sus revisiones y oportunos comentarios no hubiésemos alcanzado la meta. A Raciel por inspirarnos y ser nuestro ejemplo a seguir. A Lara por inyectarnos un poco de su inmensa creatividad. A los profesores: Dovier por su apoyo y sugerencias, Danay por sus consejos, Yusdel por su preocupación, Marcel por ayudarnos a entender lo que en realidad queríamos hacer. A Yadilka por su interés e importantes sugerencias que fortalecieron la calidad de la solución en su etapa final.

En cinco años de Universidad son muchas las personas, amigos y recuerdos que marcan nuestras vidas para siempre.

De Yisel

Agradezco a mis padres, por ser los pilares fundamentales de la mujer que soy hoy, por su apoyo, dedicación, confianza, por su amor incondicional, por guiarme por el buen camino, por ser mis padres, ellos son mi razón de ser. A mi hermano por ser tan especial y cuidarme siempre. A mi hermana por su dulzura. A mis queridos abuelos por sus consejos, ayuda y por confiar siempre en mí. Gracias a Dios por la dicha de tenerlos a todos a mi lado.

Agradezco a mi familia linda, a cada uno de ellos que forman una parte importante en mi vida.

Agradezco a la UCI y a la Revolución por permitirme obtener el título de ingeniera. A todos los profesores que de una forma u otra han influenciado en mí con su sabiduría.

Agradezco a todos los compañeros y amigos que he hecho en esta casa grande. A las chicas del apartamento, a Karen, Judith, Olia, Linet, Mayelín, Arlene, Celia, por hacer más divertida la UCI. A Maday y Regla mi familia de primer año. A Milene, Lety, Patry y Yadira, grandes amigas que no olvidaré nunca.

Agradezco a Ernesto, Nadian, Rolando y Wilson por ser amigos tan especiales. A los varones de mi grupo un piquete inigualable, a Juanqui quién no puedo dejar de mencionar, a todos gracias por su amistad.

Mi último agradecimiento es para lo mejor que me ha dado la UCI, a Leandro, mi amigo, mi pareja de estos 5 años, mi compañero de Tesis. En las buenas y en las malas, mi alegría, mi sostén en las recaídas. Gracias por existir en mi vida. Te amo.

De Leandro

Agradezco a mis padres por haberme motivado a continuar mis estudios. A mi hermano por su cariño y confianza. A mi abuelo por su devoción. A mi cuña Yuli por ser la hermana que nunca tuve. A mis sobrinitas por la alegría que dan a mi vida. Agradezco a todos mis familiares que de una forma u otra han colaborado con mis logros personales, hoy comparto mi mayor trofeo con ellos; sea mi título la mayor muestra de agradecimiento.

Gracias a todos los profesores que han sembrado en mí una semilla de conocimiento.

Agradezco a la iniciativa Xtreme por abrir un poco mis cerrados ojos, cambiaron la manera en que pensaba pasar mi carrera. Junto con ellos les doy gracias a todos los grandes amigos que hice en el mundo de la programación competitiva, gracias por cada ejercicio que me enseñaron a resolver y por ser amigos aun cuando la gloria de unos dependía del fracaso de otros.

Agradezco a Ernesto, Lety, Nadian, Mada, Patry, Raidel, Rolando y Wilson por acompañarme en los comienzos, son y serán mis mejores amigos, la distancia es incapaz de borrar los recuerdos que tenemos juntos. A Yadira y Raúl por hacer más placentera la convivencia. Dicen que los primeros años son los más duros, para mí no fue así gracias a ustedes. Agradezco a mi piquete de Venezuela, por hacerme sentir tan bien acompañado incluso lejos de casa. En especial a Willy y Pablo que fueron dos de las mejores cosas que traje de la misión, y ni me los pesaron en la aduana. Agradezco a las chicas súper poderosas del 140104, fue lindo compartir con ellas el día a día. A la enorme cantidad de amistades que alegraron mis días en la UCI.

A mi obsesión, mi amiga, mi novia, mi compañera, mi familia. Ayer y hoy mi compañera de tesis y a partir de mañana mi futuro. A Yisel porque sin ella la UCI no hubiera sido igual. Gracias por todo lo que me has dado y lo que has significado en mi vida. Te amo.

Resumen

El plagio es una actitud vergonzosa que debe evitarse y combatirse. **¿Cómo facilitar la detección de plagio en los jueces en línea de programación?** es el problema científico que fundamenta la necesidad de **desarrollar un módulo para el Juez en Línea Caribeño que facilite la detección de soluciones plagiadas;** siendo este el objetivo general de la presente investigación. Tras describir el plagio como fenómeno y su detección en términos matemáticos, se caracterizaron y valoraron como soluciones candidatas tres de los sistemas más notorios mundialmente. Se documentaron los elementos tenidos en cuenta para la selección de las herramientas, lenguajes de programación y otras tecnologías. Conforme a la elección del Proceso Unificado como metodología a utilizar en el desarrollo, se documentó la captura de requisitos, descripción de casos de usos, proceso de modelado entre otros elementos que hicieron posible una implementación fluida y libre de excesivos cambios o errores. Se describieron los criterios y ecuaciones matemáticas utilizadas para emitir un dictamen de plagio, además de las funcionalidades y características del módulo creado. Se implementaron seis analizadores léxicos que permiten la detección de plagio en los lenguajes de programación C, C++, Java, Pascal, Python y CSharp; usados en el 99.4 % del total de soluciones enviadas al Juez en Línea Caribeño. Se garantizó la calidad del módulo y la veracidad de sus resultados con varios ciclos de prueba y el análisis de los criterios de aceptación de los administradores, organizadores, entrenadores y prestigiosos miembros del Movimiento de Programación Competitiva Tomás López Jiménez.

Palabras clave: código fuente, detección de plagio, evaluación automática, juez en línea, programación.

Índice

Introducción

La Programación, en el contexto informático, es el proceso de diseñar, codificar, depurar y mantener un conjunto de instrucciones destinadas a resolver un problema determinado mediante el uso de la computación (1). Unido a otras disciplinas como la Ingeniería de Software, ha permitido el desarrollo de programas que en menos de un siglo pasaron de simples procesamientos estadísticos a la gestión de grandes cantidades de información y complejos cálculos. Bjarne Stroustrup, creador del exitoso lenguaje de programación[1] C++, afirmó: "*La única forma de aprender a programar es programando*" (2), por lo que en todo el mundo especialistas de la Informática dedican una considerable cantidad de tiempo a practicar e intentan aumentar sus conocimientos de determinado lenguaje de programación y sus habilidades para escribir algoritmos[2].

Un Juez en Línea es una aplicación web con un conjunto permanente de problemas que deben ser resueltos mediante la programación. Su principal función es evaluar automáticamente los intentos de solución de los programadores. Ganan popularidad como plataforma para el entrenamiento desde el año 1997 con la publicación del Juez en Línea de la Universidad de Valladolid (UVA) en España, creado por el profesor de matemáticas Miguel Ángel Revilla (3). El UVA junto al Sphere (SPOJ) de la Universidad de Gdansk en Polonia y al PKU (POJ) de la Universidad de Peking en China, son tres de los jueces en línea más representativos en el mundo.

En la Universidad de las Ciencias Informáticas (UCI) se desarrolla una línea de investigación relacionada con las herramientas de apoyo al Proceso de Enseñanza y Aprendizaje, particularmente en las asignaturas de programación con los jueces en línea (4). Su uso contribuye al aumento significativo que ha experimentado el movimiento de programación competitiva en el período 2009-2012. Se han celebrado diversas competencias, programas de entrenamiento, se percibe un creciente interés de los estudiantes por el tema y se insertó la UCI en el movimiento Internacional de Competencias de Programación Inter-Colegios de la Asociación para Máquinas Computadoras (ACM-ICPC).

[1] Un **lenguaje de programación** es un idioma artificial diseñado para la comunicación hombre-máquina.
[2] Un **algoritmo** es una serie finita de pasos para resolver un problema (58).

Durante el año 2006, surge en la antigua Facultad 8[3] de la UCI el primer Juez en Línea de esta institución, desarrollado por la iniciativa "*Xtreme Programming*"[4] (Xtreme). Hasta la actualidad cuenta con 2699 usuarios registrados que han enviado 79220 intentos de solución a 662 ejercicios y 69 competencias (5). También se encuentra en uso desde el año 2007 el Juez en Línea de la Cátedra de Programación Avanzada (CPAV) con 6527 usuarios, 163119 envíos, 711 problemas y 157 competencias realizadas (6).

En este contexto surge el Juez en Línea Caribeño (COJ por sus siglas en inglés de *Caribbean Online Judge*) convirtiéndose en la plataforma principal de entrenamiento para los programadores. Los resultados de este proyecto trascienden las fronteras cubanas, incluso de la Región Caribeña. Desde su publicación el 5 de junio del 2010, se han registrado 6112 usuarios de 393 instituciones pertenecientes a 97 países, se han enviado 246155 intentos de solución y fueron celebradas 146 competencias (7).

Los administradores del COJ han detectado recientemente un número considerable de soluciones muy similares, incluso algunas idénticas. El plagio es una violación grave que puede ser reflejada en diferentes esferas. La Real Academia Española al definir la palabra "plagio", nos remite directamente a la acción y efecto de "plagiar", entendiendo esta última como "copiar en lo sustancial obras ajenas, dándolas como propias" (8). Un motivo para cometer plagio es la intención de hacer más rápido y con menos esfuerzo determinada tarea sin valorar que moralmente no es correcto ni honesto apropiarse de ideas ajenas. En este sentido la ley contempla algunos tipos de plagio como delito y es sancionado de acuerdo a su gravedad.

En el Juez en Línea CPAV se desarrolló un sistema basado en la comparación de texto plano, que aunque no tenía en cuenta la semántica ni otros datos estructurales de las soluciones, sirvió para alertar sobre la gravedad del asunto. El 9 de septiembre del 2009 la primera ejecución de dicho sistema detectó 948 posibles casos de plagio correspondientes a 374 usuarios, en una segunda ejecución el 31 de mayo del 2010 fueron 271 los casos detectados y 211 los usuarios involucrados (9).

Como se planteó anteriormente, el COJ no escapa a esta situación, pero actualmente el proceso de detección de plagio es ineficiente, basándose solo en el criterio de expertos que deben hacer un escrutinio manual de las soluciones enviadas por los usuarios. Se hace necesario mejorar el proceso de detección

[3] La antigua **Facultad 8** es actualmente la Facultad 4, en el año 2010 la UCI cambia por cuestiones organizativas la constitución de sus facultades.
[4] La Iniciativa ***Xtreme Programming*** o simplemente Xtreme surge en la Facultad 8 de la UCI con el objetivo de promover la programación competitiva.

de plagio, pero no existe ninguna herramienta para la detección automática que se adecue a las características de un Juez en Línea; específicamente ninguna puede responder a la interrogante: ¿En el problema P, fue la solución A del usuario Pepe copiada de la solución B del usuario Juan? Las herramientas existentes solo concentran su funcionamiento en la comparación del código fuente[5], obviando características importantes del problema y el usuario. Por ejemplo, es diferente el análisis que debe hacerse cuando la solución requiere el uso de un algoritmo clásico que implica un bloque de código que "comúnmente" puede ser similar.

Atendiendo a la anterior problemática, se identifica el siguiente **problema científico:** *¿Cómo facilitar la detección de plagio en los jueces en línea de programación?*

Se selecciona como **objeto de estudio** para la investigación el *proceso de detección de plagio en programas informáticos.*

El **objetivo general** de la investigación es *desarrollar un módulo para el Juez en Línea Caribeño que facilite la detección de soluciones plagiadas.*

El **campo de acción** se enmarca en los *sistemas para la detección de plagio en programas informáticos en el contexto de los jueces en línea.*

Se proponen para este trabajo los siguientes **objetivos específicos:**

1- Realizar un estudio de los últimos avances sobre los sistemas para la detección de plagio determinando la metodología, herramientas y tendencias más factibles en su desarrollo.
2- Definir el perfil algorítmico que se utilizará para la detección del plagio, considerando las características inherentes a los jueces en línea.
3- Analizar, diseñar e implementar un módulo para el Juez en Línea Caribeño que facilite el proceso de identificación y gestión de soluciones plagiadas.
4- Validar el módulo obtenido como parte del proceso de investigación y desarrollo así como los resultados arrojados por el perfil algorítmico definido.

La **idea a defender** es que *un módulo para la detección de plagio en el Juez en Línea Caribeño facilitará el proceso de identificación de las soluciones plagiadas.*

[5] El **código fuente** es el conjunto de líneas de texto con las instrucciones escritas en un lenguaje de programación.

El **resultado esperado** es *un módulo que sea capaz de detectar y gestionar las soluciones plagiadas en el Juez en Línea Caribeño*.

Para la realización de esta investigación se utilizaron los **métodos científicos** detallados a continuación:

Métodos teóricos:

- Analítico-sintético: para el análisis de la teoría, documentación y demás antecedentes relacionados con el objeto de estudio, permitiendo concluir las características principales que debe tener el módulo para la detección de plagio propuesto, así como las herramientas y tecnologías óptimas para su desarrollo.
- Histórico-lógico: para estudiar la evolución histórica de los sistemas para la detección de plagio y caracterizar el avance en las tecnologías utilizadas para su desarrollo, particularizando en el área correspondiente a la detección en programas informáticos.
- Modelación: para representar los procesos, datos y características inherentes al campo de acción.

Métodos empíricos

- Observación: para entender mejor el fenómeno, interactuando con él de forma sistémica, tanto interna como externa. La observación externa que se llevó a cabo fue incluida, tanto abierta como cerrada.

Para cumplir con los objetivos fueron definidas las siguientes **tareas de investigación:**

1- Caracterizar el plagio como problema ético, sus antecedentes, tendencias actuales y consecuencias de su práctica.
2- Realizar un estudio de los sistemas para la detección de plagio, clasificaciones y aplicaciones.
3- Identificar y caracterizar las herramientas más representativas para la detección de plagio en programas informáticos.
4- Realizar un estudio sobre el uso de aplicaciones para la detección de plagio en jueces en línea.
5- Definir la metodología de desarrollo de software adecuada para el éxito del proyecto.
6- Definir las herramientas óptimas para el desarrollo del módulo para la detección de plagio.
7- Elaborar un modelo de dominio o negocio que represente los conceptos principales que se identifican en el problema.

8- Realizar el análisis, según la metodología de desarrollo seleccionada, que permita la construcción del módulo para la detección de plagio.
9- Diseñar, según la metodología de desarrollo seleccionada, un módulo para la detección de plagio en el Juez en Línea Caribeño.
10- Elaborar un perfil algorítmico para la solución.
11- Implementar la solución basándose en el diseño y el perfil algorítmico propuesto.
12- Validar el módulo desarrollado.

El documento está organizado en tres capítulos que detallan la investigación previa, análisis, diseño, elaboración del perfil algorítmico, implementación y prueba de la solución propuesta; quedando la **estructura capitular** de la siguiente manera:

Fundamentación teórica: Se documentan los principales conceptos relacionados con el plagio y los programas detectores. Se analizan las aplicaciones más comunes de este tipo de sistemas y los algoritmos que las respaldan. Se explica el objetivo y funcionamiento de los jueces en línea, ejemplificando con el Juez en Línea Caribeño. Se argumenta la necesidad de facilitar el proceso de detección de soluciones plagiadas y por último se realiza un estudio de las metodologías y herramientas necesarias para el desarrollo de la solución.

Desarrollo de la solución propuesta: Se recogen los detalles principales relacionados con el análisis y diseño de la solución propuesta. Se define un modelo de dominio que sirve de punto de partida para la identificación de requisitos funcionales, casos de uso y actores. Se documenta el proceso de realización de diagramas y modelos definidos por la metodología de desarrollo seleccionada. Finalmente, se elabora el perfil algorítmico utilizado en la solución y se documentan los principales detalles de su base lógica y funcionamiento.

Implementación y Prueba: Se reseña el proceso de elaboración de los diagramas de componentes y despliegue así como otros aspectos relacionados con la implementación. Se valida el resultado obtenido utilizando la técnica de Iadov para la determinación del Índice de Satisfacción Grupal y se valora el funcionamiento del módulo implementado así como sus dictámenes.

Fundamentación teórica

1.1 Introducción

En el presente capítulo se lleva a cabo la fundamentación teórica del trabajo, teniendo como principal objetivo investigar, estudiar y analizar los elementos principales relacionados con los sistemas para la detección de plagio como dominio de la investigación. Se fundamenta la necesidad de combatir el plagio en el COJ y la creación de un módulo para su detección como posible solución al problema. Se realiza un estudio de las aplicaciones existentes para la detección de plagio, enfatizando en las herramientas que están orientadas a los jueces en línea. Se culmina el capítulo con la elaboración del perfil tecnológico realizando la selección de las herramientas, tecnologías y metodologías a usar en el desarrollo del proyecto.

1.2 Sobre el plagio

El plagio es una violación grave que puede ser reflejada en diferentes esferas de la sociedad. La Real Academia Española al definir la palabra “plagio”, nos remite directamente a la acción y efecto de “plagiar”, entendiendo esta última como “copiar en lo sustancial obras ajenas dándolas como propias” (8).

El *Oxford English Dictionary* define plagio como “La acción o la práctica de plagiar; la apropiación ilícita o el robo, y publicación como propios, de las ideas, o de la expresión de las ideas (literario, artístico, musical, mecánico, etc.) de otros (10).

El delito del plagio viola los derechos de autor y paternidad que rodean la creación de una obra. El derecho de autor es la protección que le otorga el Estado al creador de las obras literarias o artísticas desde el momento de su creación y por un tiempo determinado. Por otro lado se define como propiedad intelectual una disciplina normativa que protege las creaciones intelectuales provenientes de un esfuerzo,

trabajo o destreza humanos, dignos de reconocimiento jurídico. La propiedad intelectual comprende: el derecho de autor, los derechos conexos[6] y la propiedad industrial (11).

Actualmente el auge de Internet ha hecho que sea más fácil plagiar el trabajo de otros (12). Es común que muchos sitios copien y peguen información sin la autorización correspondiente, engañando a sus lectores al presentar los datos como propios; de igual manera las personas copian y pegan de sitios e investigaciones científicas sin referenciar la fuente, a estas acciones se les conoce como plagio de texto. Un caso particular lo constituye el plagio de algoritmos y códigos fuentes de programas informáticos. Sobre todo en el ámbito académico el plagio es un comportamiento inaceptable y actualmente existen varias herramientas informáticas que permiten detectarlo.

1.3 Generalidades de los sistemas para la detección de plagio

Los sistemas para la detección de plagio son herramientas con tecnologías informáticas que realizan la búsqueda de similitudes entre muestras con el fin de encontrar fragmentos plagiados total o parcialmente. Se basan en diversas técnicas de recuperación y extracción de información así como de reconocimiento de formas y teorías de la información (12). Estos sistemas comienzan a surgir a mediados de los años 70 como posible solución ante la inaceptable tendencia presente en investigaciones científicas, obras, algoritmos y códigos de programación (13).

1.3.1 El problema de la detección de plagio

El problema de la detección automática de plagio puede verse como un problema de búsqueda y/o clasificación (12). Los elementos de referencia ***D*** y el elemento sospechoso ***S*** son suficientes para describir el planteamiento de la detección de plagio:

Sean ***S*** un elemento sospechoso y ***D*** un conjunto de elementos de referencia, el objetivo de la detección automática de plagio es encontrar aquel elemento $d \in D$ que haya sido utilizado como fuente para obtener el elemento ***S***, el cual presumiblemente es un caso de plagio. Dicha búsqueda puede llevarse a un nivel

[6] **Derechos conexos** es un término relacionado con la ley de derechos de autor y copyright, para referirse a derechos similares, los derechos conexos son parte del marco jurídico del copyright.

más específico: Sea $S_i \in S$ un fragmento plagiado, el objetivo es encontrar aquel fragmento $d_j \in d$ tal que d_j es la fuente del fragmento plagiado S_i.

1.4 Aplicaciones de los sistemas para la detección de plagio

Los sistemas para la detección automática de plagio tienen aplicación en diferentes esferas de la vida profesional, a continuación se hará referencia a su utilización en el entorno académico, empresarial y el derecho de autoría.

1.4.1 Los sistemas para la detección automática de plagio en el entorno académico.

El tema del plagio académico se refleja en los trabajos encargados a los alumnos, es actualmente un problema importante y siempre lo ha sido. En 1995 Alschuler y Blimling hablaban de epidemia refiriéndose al aumento en esta tendencia; pero quizás es más común ahora, debido a la colosal cantidad de información disponible en Internet y en general a las facilidades que proporcionan las nuevas tecnologías de la información y las comunicaciones (TIC). En España el 61.1 % del alumnado universitario afirma haber incorporado en sus trabajos fragmentos de páginas web como si fueran propios al menos una vez; estudios confirman la práctica generalizada de copiar y pegar sin citar las fuentes (14). El plagio de ensayos, seminarios, artículos y prácticas de programación es un problema serio en la enseñanza actual. Los profesores no se dan cuenta en algunas ocasiones de que el trabajo de sus estudiantes no es del todo honesto, debido a que la detección manual es complicada y en ocasiones los pedagogos no cuentan con el tiempo suficiente para detectar la situación.

Si cierto es que la tecnología ha ayudado de una forma u otra a facilitar el plagio, no se puede descartar que a la vez ha ayudado a prevenirlo y a detectarlo, ya que existen varias herramientas informáticas desarrolladas con este fin.

Como es tarea de todos eliminar estas incidencias, muchas son las universidades de todo el mundo que han creado sistemas de este tipo para combatir el plagio en sus centros de estudios y brindar sus servicios a otras esferas y entidades, entre estas universidades se puede citar:

- MOSS, en la Universidad de Stanford, EUA.
- JPlag, en la Universidad de Karlsruhe, Alemania.

- YAP3, en la Universidad de Sydney, Australia.

En las universidades británicas y norteamericanas por ejemplo, existe un protocolo por el que debe pasar cualquier trabajo de sus estudiantes. Entre los programas más utilizados con este objetivo están: Turnitin, Copycatch y Eve2 (15).

1.4.2 Los sistemas para la detección automática de plagio en el entorno empresarial.

Con el arribo de la Web 2.0, las empresas han entrado a una nueva era de la informatización, ahora no están ajenas a las tendencias actuales y a los beneficios de Internet. Las que se han apoyado en el marketing a través de los medios digitales, han obtenido resultados satisfactorios en el mundo competitivo y empresarial. El fácil acceso a la información que brinda la red ha favorecido un aumento, en cierta medida, de los casos de plagio entre empresas. Esto conlleva a que otras empresas les copien productos, servicios e ideas, por lo que deben protegerse ante aquellas plagiadoras que se valen del esfuerzo y resultados de otros para atribuirse méritos ajenos (16).

Existen también herramientas para la detección automática de plagio disponibles para que las empresas se protejan. WCopyFind es un software desarrollado por Bloomfield de la Universidad de Virginia, Estados Unidos, en el año 2004 y especializado en la detección de plagio en el entorno empresarial (16).

1.4.3 Los sistemas para la detección automática de plagio en la protección al Derecho de Autoría.

“El plagio constituye el más grave atentado al derecho de autor, pues en esencia significa desconocer la paternidad del autor, y por consiguiente, la relación que le une con la obra sustrayéndole a todo conocimiento e ignorándole toda aportación creativa” (17).

Actualmente las investigaciones científicas, artículos, libros, literatura, páginas webs que se encuentran en la red, se ven amenazadas por personas que no respetan los derechos de autor y se atribuyen estas creaciones como propias. En ocasiones consientes del delito, pero en otras, es por desconocimiento de la cultura de referenciar y citar a los legítimos autores. Existen, hoy en día, formas más avanzadas de proteger los derechos de autoría haciendo uso de la tecnología. Los sistemas de detección automática de plagio se crean para que cumplan la función de identificar las obras que han sido plagiadas en su totalidad o fragmentos de las mismas.

A continuación se mencionan algunos programas que se encargan de detectar plagios en libros, literatura de todo tipo, imágenes y audio:

- Para detectar el plagio en texto el sistema Viper basa su funcionamiento en buscar entre los documentos locales y en Internet. Existen otros con este mismo objetivo como PlagiarismChecker, Plagium, Chimpsky o CopyTracker.
- Para identificar plagio en las imágenes GazoPa y TinEye, aprovechando sofisticados algoritmos, buscan fotografías similares entre diferentes servidores.
- En el audio también se detecta el plagio, algunas de las herramientas informáticas que se pueden usar son Shazam y SoundHound (18).

1.5 Algoritmos para la detección de plagio en código fuente.

Los algoritmos generalmente utilizados en la detección de plagio en código fuente van desde criterios muy sencillos basados en métricas de software como el conteo de variables, hasta complejas funciones de codificación y comparación de porciones de texto. Se han logrado identificar tres tendencias fundamentales encabezadas por el uso de los algoritmos Running-Karp-Rabbin, Greedy-String-Tiling y técnicas basadas en funciones "*hash",* explicadas más adelante, para generar claves que representan de manera casi unívoca a un documento, texto o archivo (19). A continuación se detalla la base lógica y principales características de estos algoritmos:

1.5.1 Algoritmo Running-Karp-Rabbin.

La base del algoritmo Running-Karp-Rabbin es el *hash Karp-Rabin* utilizado para crear claves en porciones no comparadas del código fuente, las coincidencias son buscadas sobre dicha clave y el tiempo de ejecución es optimizado con el uso de tablas *hash* para el almacenamiento. Contrario a realizar una búsqueda exhaustiva de coincidencias, se puede obtener de la tabla las posiciones iniciales de todas las porciones de código de longitud **S** procesadas en orden descendente para intentar encontrar primero coincidencias mayores, que tengan el mismo valor asignado por la función *Karp-Rabin*. Luego de encontrar una coincidencia de la porción de código y la clave a ella asignada, se continúa buscando nuevos elementos hasta que el algoritmo se detiene debido a una diferencia o el fin de la entrada (20).

1.5.2 Algoritmo Greedy-String-Tiling.

El algoritmo Greedy-String-Tiling intenta encontrar similitudes de código comenzando por valores extremos y disminuyendo la longitud hasta un mínimo elegido. En cada ciclo de búsqueda se analizan las porciones de tamaño **S** que coincidan en ambos códigos, las mismas son añadidas a la lista de coincidencias en una estructura del tipo: **"posición fuente, posición destino, longitud"** reflejando una similitud encontrada comenzando en la **posición fuente** del primer código y en la **posición destino** del segundo, con una **longitud** determinada por el tercer parámetro. La porción de código es marcada para no tenerla en cuenta en futuras iteraciones y se continúa el proceso con valores **S** menores hasta llegar al umbral inferior de la detección (21).

1.5.3 Algoritmos basados en funciones Hash.

El termino *hash* o picadillo es asociado a las funciones *hash* y las tablas *hash* indistintamente. En resumen no es más que la codificación de un texto en un valor acotado, mediante una función matemática. El objetivo principal de este procedimiento es aprovechar las capacidades de direccionamiento directo de estructuras como los arreglos, ya que, gracias a la conversión de un texto en su correspondiente clave numérica, esta puede ser utilizada como índice del arreglo y realizar operaciones de búsqueda en orden constante, en este caso la complejidad algorítmica está determinada por el rendimiento de la función utilizada para generar la clave *hash* correspondiente.

En la detección de plagio, el uso de funciones para convertir el código en una clave numérica, tiene como principal objetivo ganar en velocidad de comparación, debido a que el proceso de obtención de los elementos es constante y óptimo, como antes se señalaba. El funcionamiento, apartando la complejidad de la función *hash*, es bastante simple y sin alejarse mucho de la comparación realizada por otros algoritmos para la detección de plagio, pero con una complejidad computacional menor (22).

1.6 Herramientas para la detección de plagio en código fuente

Existe un considerable número de herramientas para la detección de plagio en código fuente bajo distintos tipos de licencia disponibles en Internet, ya sea en forma de servicio web o como una aplicación descargable, incluso por medio de correo electrónico. Dichas aplicaciones tienen sus más antiguos

precedentes en un artículo de Karl J. Ottenstein (23). Se destacan entre ellas el YAP3 de Michael Wise, el MOSS de Alex Aiken y el JPlag de Guido Malpohl. Ellas representan tendencias diferentes en la forma de abordar el problema de la detección de plagio, por lo que se seleccionan como objetos de investigación. A continuación se detallan algunas de sus principales características:

1.6.1 YAP3

YAP3 por sus siglas en inglés de ***Y****et* ***A****nother* ***P****lague* o traducido “otra plaga más”, es un programa desarrollado en el año 1996 que basa su funcionamiento en utilidades de la consola Unix, fue programado en Perl y los scripts pueden ser bajados desde la web del autor (24). Es una herramienta gratuita con fines no comerciales disponible para los lenguajes C, Pascal y Lisp. Se basa en una mezcla de Running-Karp-Rabbin con Greedy-String-Tiling. Comienza su procesamiento del código eliminando los comentarios y constantes de cadena, convierte todos los textos a letras minúsculas por lo que no tiene en cuenta esta característica. Mapeando los sinónimos a una palabra común simplifica el engaño basado en remplazos de este tipo. Su estrategia para evadir los reordenamientos del código se fundamenta en procesar las funciones de acuerdo a su orden de llamada.

Sus principales desventajas vienen dadas por la rústica representación de los resultados, no puede ser usado para fines comerciales y está disponible para pocos lenguajes de programación.

1.6.2 MOSS

MOSS de sus siglas en inglés para ***M****easure* ***O****f* ***S****oftware* ***S****imilarity* o traducido “medición de similaridad de programas”. Es un servicio de Internet desarrollado en el año 1994. Funciona sometiendo a su consideración un grupo de ficheros y el sistema responde con un conjunto de páginas HTML detallando las similitudes detectadas. Es una herramienta gratuita aunque requiere obtener una cuenta vía correo electrónico del propietario; su uso está limitado a fines no comerciales. El MOSS se encuentra disponible para los lenguajes de programación: C, C++, Java, C#, Python, Visual Basic, JavaScript, FORTRAN, ML, Haskell, Lisp, Scheme, Pascal, Modula2, Ada, Perl, TCL, Matlab, VHDL, Verilog, Spice, MIPS assembly, a8086 assembly, a8086 assembly, MIPS assembly, HCL2.

Basa su funcionamiento en la comparación de huellas digitales sacadas de la estructura de los documentos, o códigos fuentes en este caso, luego divide los conjuntos de huellas en porciones

basándose en algoritmos n-gramas[7], a las cuales se adjunta información estructural como el número de la página referenciada. Inicialmente el algoritmo crea un índice a distintos lugares dentro de los documentos, identificados con su respectiva huella, para luego crear de todas las coincidencias la huella distintiva del documento. Como último paso las coincidencias son ordenadas de mayor a menor grado según su nivel de relevancia (23).

MOSS es un poderoso sistema del cual se pueden señalar como características positivas la cantidad de lenguajes de programación que soporta y la calidad de sus resultados. En sentido negativo, la imposibilidad de utilizarlo comercialmente y la complejidad en la implementación de los algoritmos que lo sustentan.

1.6.3 JPlag

JPlag viene de una mezcla entre los prefijos **J** de Java y **Plag** de *Plagiarism*. Es un sistema para la detección de plagio en conjuntos de códigos fuentes que se encuentra en forma de servicio web desde el año 1996 y su uso es gratuito aunque limitado por la creación de una cuenta. Esta aplicación programada en Java toma como entrada un conjunto de programas y para cada par computa las regiones similares; como resultado genera un reporte en forma de página HTML. JPlag se basa en un algoritmo similar al usado por YAP3 pero con algunas optimizaciones para mejorar su eficiencia. Trabaja convirtiendo el programa de alguno de los lenguajes que acepta: Java, C#, C y C++, en símbolos básicos o *tokens* para luego tratar de completar su estructura con subcadenas de estos símbolos correspondientes al programa con el que se está comparando. El porciento de plagio es deducido con respecto a la cantidad de subcadenas que puedan completarse.

Su principal desventaja es que la cuenta necesaria para su uso está sujeta a la aceptación del propietario. Además al estar disponible solo para cuatro lenguajes de programación y el COJ actualmente contar con la posibilidad de enviar soluciones en nueve lenguajes ofrece a los usuarios del Juez en Línea una vía de escape traduciendo a un lenguaje no contemplado en el módulo para la detección de plagio.

[7] La **n-grama** es una representación redundante de texto que consiste en fragmentos solapados, ya sea a nivel de carácter o de palabra, cuya longitud es **n**. Por ejemplo, las 3-gramas, a nivel de caracteres de “ejemplo” son [eje, jem, emp, mpl, plo]; y los 2-gramas a nivel palabra de “este es solo un ejemplo” son [este es, es solo, solo un, un ejemplo] (19).

1.7 Jueces en línea de programación

Un Juez en Línea de Programación es una aplicación web cuya principal funcionalidad consiste en evaluar, de forma automática, propuestas de solución a un conjunto permanente de problemas o en el contexto de una competencia de programación. Adicionalmente se puede tomar como factor para determinar el potencial de un Juez en Línea la cantidad de lenguajes de programación que soporta para evaluar las soluciones, la cantidad de problemas, las modalidades de competencia, además de los clásicos patrones de usabilidad, confiabilidad y disponibilidad.

Hay decenas de jueces en línea alrededor del mundo destacándose el UVA creado en 1995 por Miguel Ángel Revilla y publicado por la Universidad de Valladolid en abril de 1997. Este Juez ha recibido más de 10000000 de códigos fuentes para evaluarlos (3); se destaca también como Juez oficial de la ACM-ICPC. El Sphere Online Judge (SPOJ) de la Universidad de Gdansk en Polonia, una plataforma capaz de evaluar soluciones en más de 40 lenguajes, con más de 11600 problemas y disponible en varios idiomas; alrededor de 21300 usuarios envían cerca de 100000 soluciones mensuales (25).

Además del PKU (POJ) de la Universidad de Peking en China vale la pena mencionar al joven Codeforces, publicado en internet desde el 2010 convirtiéndose en el primer Juez de la Web 2.0, desarrollado por Mike Mirzayanov, su principal característica es la constancia con que ha venido celebrando competencias casi semanales.

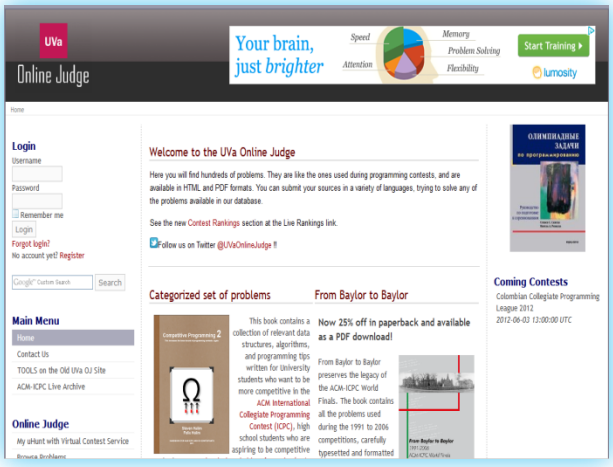


Figura 1: Interfaz principal de UVA.

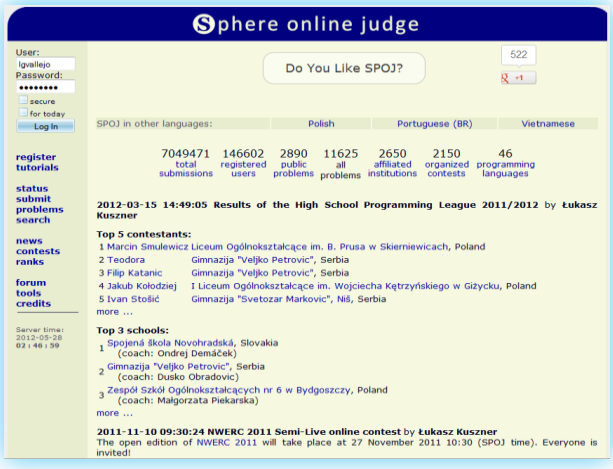


Figura 2: Interfaz principal del SPOJ.

La UCI comienza a incursionar en el desarrollo de jueces en línea desde el año 2006 y hasta la actualidad se han publicado más de 5 aplicaciones de este tipo, tomaron mayor relevancia el Juez en Línea de la iniciativa Xtreme y la Cátedra de Programación Avanzada (CPAV). Con la experiencia obtenida con estos proyectos surge, en el marco de la inserción de la UCI en el movimiento ACM-ICPC y como vanguardia en Cuba y el Caribe, la iniciativa de publicar un juez para la Región; surge así el 5 de Junio del 2010 el Juez en Línea Caribeño (COJ).

Figura 3: Interfaz principal del COJ

1.8 Plagio en los jueces en línea

El plagio también está presente en los jueces en línea y es una de las dificultades que presentan actualmente. Se han identificado usuarios que envían soluciones de otros, dándolas como suyas. El 9 de septiembre del 2009 el CPAV detectó mediante una aplicación para la comparación de texto 948 posibles casos de plagio correspondientes a 374 usuarios, en una segunda corrida de la aplicación el 31 de mayo del 2010 fueron 271 posibles casos y 211 los usuarios involucrados (9). La necesidad de detectar el plagio se ha tornado primordial para garantizar la fiabilidad de los resultados en las competencias y las estadísticas; además esta práctica antiacadémica y antiética atenta contra el aprovechamiento que pueden hacer los estudiantes de una aplicación tan útil para las asignaturas de programación.

1.8.1 Herramientas para la detección de plagio en jueces en línea

Se llevó a cabo una intensiva búsqueda de herramientas para la detección de plagio usadas por los jueces en línea, pero no se encontró resultado alguno. Mediante la investigación y comunicación vía correo electrónico con quince administradores de los jueces en línea alrededor del mundo, solo se ha confirmado que el SPOJ intentó usar una herramienta de este tipo; limitándose a consumir los servicios prestados por el MOSS. No se ha logrado identificar una tendencia a vincular aplicaciones para detectar plagio en los jueces en línea.

1.8.2 Conclusiones parciales

Luego de analizar los principales algoritmos para la detección de plagio, los sistemas ya existentes e investigar su presencia en los jueces en línea, se concluye que no es posible contar con la experiencia de proyectos similares para afrontar la presente problemática. No es conveniente el uso de alguno de los sistemas para la detección existentes, debido a que sus ventajas no se complementan y sus desventajas vienen dadas por factores como condiciones para otorgar la licencia, deficiencia en la cantidad de lenguajes de programación contemplados y principalmente la calidad de los dictámenes al tenerse solo en cuenta la estructura del código fuente. Estos sistemas son incapaces de valorar la repercusión que tienen sobre las similitudes detectadas, la pertenencia del problema a una categoría determinada. Es lógico que el código de diferentes usuarios usado para implementar un algoritmo de camino mínimo, por ejemplo el Dijkstra, sea similar ya que el mismo puede encontrarse en varias bibliografías y es considerado un algoritmo clásico (26). Por otra parte y ante la necesidad de implementar un sistema propio para el Juez en Línea Caribeño, se valoró el funcionamiento y ventajas de las principales tendencias algorítmicas para la detección de plagio, quedando fundamentada la elección de Greedy-String-Tiling para el uso en la comparación estructural del código, sobre la base de la complejidad computacional y de la implementación. Se determina en este caso que la razón “esfuerzo de implementarlo sobre la calidad de sus dictámenes” es superior a otros algoritmos.

1.9 Metodología de desarrollo

La solución en desarrollo, al formar parte modular del COJ, está sujeta a determinadas tecnologías de las que el mismo hace uso, no obstante se hizo un estudio de posibles alternativas y sus características. Se valoró la elección de Scrum, la Programación Extrema (XP) y el Proceso Unificado de Desarrollo (RUP) como posibles metodologías a utilizar.

1.9.1 SCRUM

La metodología SCRUM toma su nombre de una posición entrelazada en círculo que toman los integrantes de los equipos de rugby para tomar decisiones sobre el juego. Sus principios fundamentales fueron desarrollados en procesos de reingeniería por Goldratt, Takeuchi y Nonaka en la década de 1980 y aplicados al proceso de desarrollo de software por Jeff Sutherland en 1993, siendo formalizada con la

colaboración de Ken Schwaber en una presentación en OOSPLA 96. Los principios de la misma han evolucionado con las contribuciones de varios autores, entre los que se pueden citar Cohn, Beedle y Schwaber (27).

En SCRUM se define un marco para la gestión de proyectos. Está especialmente indicada cuándo se requiere de un rápido cambio de requisitos. Sus principales características se pueden resumir en dos: el desarrollo de software se realiza mediante iteraciones, denominadas Sprint, con una duración de 30 días; y las reuniones a lo largo del proyecto, entre ellas destacan la reunión diaria de 15 minutos del equipo de desarrollo para coordinación e integración (27).

1.9.2 Programación Extrema (XP)

La Programación Extrema (XP) es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en el desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los programadores y propiciando un buen clima de trabajo. Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las soluciones implementadas y buena actitud a la hora de enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes (27). Se considera ligera, flexible, predecible y de bajo riesgo. Tiene pocos requerimientos de documentación y planificación. El cliente forma parte integral del proceso de desarrollo (28).

1.9.3 Proceso Unificado de Desarrollo (RUP)

Figura 4: Logo de RUP

Es una metodología para la ingeniería de software, que va más allá del mero análisis y diseño orientado a objetos para proporcionar una familia de técnicas que soportan el ciclo completo de desarrollo de software. El resultado es un proceso basado en componentes, dirigido por los casos de uso, iterativo e incremental y centrado en la arquitectura (29). Utiliza el Lenguaje Unificado de Modelado (UML) para preparar todos los esquemas de un sistema software. Consta de nueve flujos de trabajos fundamentales, de ellos seis denominados “de Ingeniería”, que son modelamiento del negocio, requerimientos, análisis y diseño, implementación, prueba

y despliegue. Los restantes tres flujos de trabajo son considerados de apoyo y son la administración de proyectos, la gestión de configuración y cambios, y finalmente el ambiente (30).

Analizando las características principales de cada opción y teniendo en cuanta las ventajas y desventajas que cada una supone se opta por el uso de RUP basándose en los siguientes puntos:

- La gran cantidad de artefactos que propone RUP necesitaría un mayor equipo de desarrollo, aunque dichos artefactos no tienen que desarrollarse al mismo tiempo lo que posibilita que una menor cantidad de personal se haga cargo de ellos en un orden bien definido.
- Solo estarán trabajando a tiempo completo en el proyecto los desarrolladores, en este caso dos; XP define que todo aquel que desee colaborar debe formar parte integral del equipo de desarrollo, por lo que cualquier asesoramiento o ayuda sería problemático.
- Se pretende continuar con la investigación en futuros niveles científicos, por lo que es necesaria una correcta documentación del proceso de desarrollo, cualidad reconocida de RUP.
- No es factible el uso de XP, cuyas técnicas son basadas en equipos de desarrollo bien definidos, no siendo el caso en esta investigación al solo tratarse de dos individuos.
- XP define actividades paralelas por definición lo que complejiza el proceso para solo dos personas.
- RUP proporciona un expediente de proyecto de muy buena calidad lo que facilita la continuidad del desarrollo en manos de un nuevo equipo y la integración con nuevas ideas.

Basándose en estos puntos se considera que RUP facilitará el desarrollo exitoso del proyecto y la correcta documentación del mismo.

1.10 Herramientas y tecnologías a utilizar

1.10.1 Lenguaje de programación

Entre las posibles opciones para la elección del lenguaje de desarrollo se valoraron fundamentalmente tres, a continuación se exponen las principales características de PHP, C# y Java. Se escogen estas tres para estudiarlas teniendo en cuenta su empleo en la realización de aplicaciones similares.

1.10.1.1 C#

Scott Wiltamuth, Anders Hejlsberg y la compañía Microsoft fueron los creadores de C#, lenguaje de programación orientado a objetos que ha deleitado a los programadores de todo el mundo y está incluido en la plataforma .NET. Entre sus principales características se puede señalar la sencillez, modernidad, gestión automática de memoria, indexadores, eventos, delegados y tipos genéricos que lo convierten en una poderosa opción (31).

1.10.1.2 PHP

En el año 1994 Rasmus Lerdorf crea PHP, acrónimo recursivo que significa PHP Hypertext Pre-processor un lenguaje interpretado ideal para el desarrollo de sitios web dinámicos. Se pueden resaltar entre sus principales ventajas su escaso consumo de recursos, su rapidez y amplia capacidad de conectividad. Lamentablemente requiere un alto nivel de experiencia para la realización de tareas complejas. (32)

1.10.1.3 Java

Sun Microsystems se gana el protagonismo al diseñar Java en el año 1991, la principal característica buscada con su creación se convierte en la principal ventaja, y está dada por ser un lenguaje multiplataforma, capaz de correr no solo en computadoras sino en celulares, hasta equipos electrodomésticos, cuyo único requisito es tener la máquina virtual de Java, concepto desarrollado por Sun con el objetivo de proveer independencia de la arquitectura sobre la que corre la aplicación. Es un lenguaje de programación no propietario.

Figura 5: Logo de Java

Sun describe a Java como simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico (33). Su estrecha unión con internet está dada primeramente por el auge que alcanzaron ambos en fechas similares. Se puede señalar como desventaja la dependencia de la máquina virtual para que los programas funcionen, también al tener que pasar por este paso intermedio se ralentiza un poco el proceso total de ejecución.

1.10.1.4 Fundamentación de la elección

Se decide el uso de Java por las razones siguientes:

- Es un lenguaje muy utilizado por la UCI al no ser propietario.
- El COJ está desarrollado en este lenguaje.
- Los servidores actuales sobre los que está desplegado el COJ son servidores de aplicaciones Java.
- El ambiente de trabajo utilizado actualmente por el equipo de desarrollo del COJ es Java, otro lenguaje se saldría de los esquemas del proceso integral de desarrollo de la aplicación cliente.

1.10.2 Gestor de base de datos

A continuación se detallan las principales características de MySQL y PostgreSQL como posibles gestores de base datos a utilizarse para el desarrollo.

MySQL por su parte es más ligero que PostgreSQL y la resolución de consultas tiende a realizarse más rápido, son factores también positivos la amplia documentación existente y la calidad de las herramientas de administración. PostgreSQL por su parte provee bases de datos con altos niveles de integridad y consistencia, su comportamiento ante consultas concurrentes es mucho mejor que MySQL, es escalable y estable antes cargas excesivas (34).

Figura 6: Logo de PostgreSQL

Finalmente, se selecciona PostgreSQL para el desarrollo de la aplicación en correspondencia de la tecnología actualmente usada por el COJ y las líneas estratégicas de la UCI y atendiendo fundamentalmente a su superioridad en el manejo de la concurrencia.

1.10.3 Tecnologías de desarrollo web con Java

Desde sus inicios, Java ha sido un lenguaje completamente orientado a la arquitectura, reafirmando esto el hecho de que fue de los primeros en incorporar el concepto de framework. En el desarrollo de software, un framework es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado (32).

El uso de estas estructuras ha sido ampliamente discutido existiendo una divergencia de criterios. Indudablemente, abstraen la arquitectura original de la tecnología sobre la que están hechos, lo que representa una ventaja al normalmente disminuir el grado de complejidad de esta y a la vez agregarle

capacidades de reutilización claves en el sistema que se desarrolla. Sin embargo, se ha demostrado (33) que el uso de frameworks le agrega complejidad al software en ocasiones hasta un punto en el que el proceso de desarrollo puede llegar a situarse en un callejón sin salida. Resulta altamente complicado encontrar el conjunto ideal de herramientas de este tipo que cumplan eficientemente su función. Considerando todo lo anterior, se decide el uso de frameworks sólo en las secciones de la aplicación donde sean absolutamente necesarios, valorándose además el grado de aceptación y estabilidad a la hora de seleccionar las herramientas específicas. En las secciones donde no se pueda garantizar, por diversas razones, la utilización exitosa de un framework, se aplica tecnología nativa de Java (35).

1.10.3.1 Capa de presentación

La aplicación no contará con interfaces demasiado complejas, más bien se limitará a la organización de información, gestión de usuario y representación de resultados por lo tanto el uso de frameworks no se considera necesario. Se elige de esta forma a Java Server Pages (JSP) tecnología complementaria del lenguaje java que entre sus principales ventajas tiene la sencilla integración con tendencias más modernas como Ajax.

1.10.3.2 Lógica del negocio

Como parte de la lógica del negocio se decide la utilización del framework Spring por permitir implementar varias filosofías consideradas necesarias dentro una aplicación web, independientemente de su tamaño (36). Entre estas se pueden citar el soporte al patrón “Inyección de dependencias” y a la Programación Orientada a Aspectos. Por encima de todo esto, las principales razones que motivaron su elección son la ligereza, la transparencia y el ser no intrusivo (37).

1.10.3.3 Acceso a datos

Para la implementación del acceso a datos en la aplicación, se valoró el framework Hibernate (40), entorno de trabajo que tiene como objetivo facilitar la persistencia de objetos Java en Bases de Datos (BD) relacionales. Es considerado por gran margen el framework más popular de persistencia de Java y uno de los más populares en sentido general. Se basa principalmente en el patrón “Mapeo Objeto Relacional” y funciona efectivamente para el manejo de procesos claves dentro de las aplicaciones, como el manejo de transacciones, la concurrencia y la caché (38). Se tomó la decisión de no usarlo debido a

que a pesar de las ventajas que ofrece, es negativa su influencia en el rendimiento tan necesario en el módulo que se pretende desarrollar y sobre todo en el Juez en Línea.

1.10.4 Herramientas CASE para el modelado

Computer Aided Software Engineering (CASE) o Ingeniería de Software Asistida por Computación, es la aplicación de métodos y técnicas a través de las cuales se hacen útiles a las personas comprender las capacidades de las computadoras, por medio de programas, de procedimientos y su respectiva documentación. Las herramientas CASE representan una forma de modelar los procesos que son generados en un proyecto real (39).

1.10.4.1 Visual Paradigm

Figura 7: Logo del Visual Paradigm

Visual Paradigm es una herramienta UML (Unified Modeling Language) profesional que soporta el ciclo de vida completo del desarrollo de software: análisis y diseño Orientados a Objetos, construcción, pruebas y despliegue. El software de modelado UML ayuda a una más rápida construcción de aplicaciones de calidad, mejores y a un menor coste. Permite dibujar todos los tipos de diagramas de clases, código inverso, generar código desde diagramas y generar documentación (40).

El Visual Paradigm constituye una herramienta de software libre de probada utilidad para el analista de un proyecto. Se caracteriza por la disponibilidad en múltiples plataformas (Windows, Linux). Posee características como, diseño centrado en casos de uso y enfocado al negocio que genera un software de mayor calidad. Además de que el modelo y el código permanecen sincronizados en todo el ciclo de desarrollo. Posee la ventaja que permite la generación de BD, transformación de diagramas de Entidad-Relación en tablas de BD. Posee compatibilidad entre ediciones (41).

1.10.4.2 Rational Rose

Rational Rose herramienta CASE desarrollada por Rational Corporation basada en el UML, que permite crear los diagramas que se van generando durante el proceso de Ingeniería en el Desarrollo del Software. Para el ambiente de modelado, soporta la generación de código a partir de modelos en C++, CORBA,

Java, Visual Basic, entre otros. Esta herramienta permite que el equipo de desarrollo entienda mejor el problema, que identifique las necesidades del cliente en forma más efectiva y comunique la solución propuesta de forma más clara. Rational permite completar una gran parte de las disciplinas (flujos fundamentales) de RUP. Presenta cuatro Vistas: la Vista de Casos de Uso, en la que se representa el comportamiento deseado del sistema, la Vista Lógica que muestra la estructura y el comportamiento del sistema, la de Componentes y por último la Vista de Despliegue, en la que se modela la distribución o despliegue de los nodos de procesamiento del sistema. Posee librerías para la compatibilización de ingeniería inversa sobre Java, XML_DTD, CORBA y Visual Basic (42).

El Rational Rose Enterprise Edition, presenta ciertas desventajas que influyen de manera negativa en su uso, pues es un software propietario, lo que dificulta su adquisición. Solo cuenta con versiones en inglés, por lo que sus manuales de ayuda a pesar de ser muy útiles se encuentran en el mismo idioma.

1.10.4.3 Fundamentación de la elección

La herramienta CASE que se utilizará para el sistema en desarrollo será el Visual Paradigm, eficaz para el proceso de modelado y la representación de los flujos inherentes a RUP, metodología de desarrollo elegida. Posee una interfaz amigable. Pertenece a la familia de software libre y la tendencia que se lleva en la UCI es promover su uso. Además se tuvo en cuenta el conocimiento del equipo de desarrollo y la documentación accesible para esta.

1.10.5 Herramientas de Desarrollo del Software

Una de las claves para obtener el éxito de un proyecto de desarrollo de software, es tener en cuenta, la utilización de tecnologías, estándares y herramientas a usar. Un IDE (Entorno de Desarrollo Integrado / *Integrated Development Enviroment*) es un entorno de programación que ha sido empaquetado como un programa de aplicación, o sea, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica.

1.10.5.1 Eclipse

Eclipse es un entorno de desarrollo integrado, de Código abierto y Multiplataforma. Mayoritariamente se utiliza para desarrollar lo que se conoce como "Aplicaciones de Cliente Enriquecido", opuesto a las aplicaciones "Cliente-liviano" basadas en navegadores. Es una potente y completa plataforma de

Programación, desarrollo y compilación de elementos tan variados como sitios web, programas en C++ o aplicaciones Java. Integra herramientas y funciones necesarias para el trabajo, recogidas además en una atractiva interfaz que lo hace fácil y agradable de usar.

Eclipse provee al programador con Frameworks muy ricos para el desarrollo de aplicaciones gráficas, definición y manipulación de modelos de Software y Aplicaciones web. Pose arquitectura plug-in que permite escribir cualquier extensión deseada en el ambiente, como sería Gestión de la configuración. Se provee soporte para Java y CVS en el SDK de Eclipse (43).

1.10.5.2 NetBeans

NetBeans es un IDE que permite que las aplicaciones sean desarrolladas a partir de un conjunto de Componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software. Es una herramienta para programadores pensada para escribir, compilar, depurar y ejecutar programas. El IDE NetBeans es un producto libre y gratuito sin restricciones de uso (44).

Figura 8: Logo del NetBeans

1.10.6 Servidor de aplicaciones

Tomcat es un Servidor web con soporte de Servlets y JSPs. Incluye el compilador Jasper, que compila JSPs convirtiéndolas en servlets. El motor de servlets de Tomcat a menudo se presenta en combinación con el servidor web Apache.

Tomcat puede funcionar como servidor web por sí mismo. En sus inicios existió la percepción de que el uso de Tomcat de forma autónoma era sólo recomendable para entornos de desarrollo y entornos con requisitos mínimos de velocidad y gestión de transacciones. Hoy en día ya no existe esa percepción y Tomcat es usado como servidor web autónomo en entornos con alto nivel de tráfico y alta disponibilidad.

Dado que Tomcat fue escrito en Java, funciona en cualquier sistema operativo que disponga de la máquina virtual de Java (45).

El Apache Tomcat 7.0.12, actualmente utilizado por el COJ, se mantiene para el desarrollo de la solución actual.

1.11 Conclusiones

En este capítulo se profundizó en los conceptos relacionados con el dominio de los sistemas para la detección de plagio y el plagio como problema. Mediante un estudio de los algoritmos más usados en los sistemas para la detección de plagio se llegó a la conclusión de que la mejor variante para su uso en la implementación es el Greedy-String-Tiling basándose fundamentalmente en la relación entre la complejidad de su implementación y los dictámenes que produce. Se realizó un análisis de las aplicaciones existentes documentando sus principales características: lenguajes que soportan, algoritmos en que basan su funcionamiento, forma de acceder al servicio, presentación de los resultados, rendimiento y tipo de patente. Atendiendo a las tendencias actuales y principalmente a la experiencia de la UCI se eligieron las siguientes herramientas y se fundamentó la elección de las mismas para completar las necesidades del proceso de desarrollo, como metodología de desarrollo RUP, lenguaje de programación se seleccionó a Java complementando con el framework Spring para la gestión del negocio, el gestor de bases de datos PostgreSQL, NetBeans 7.1 como IDE de programación y el Visual Paradigm como herramienta CASE.

Desarrollo de la solución propuesta

2.1 Introducción

La función del presente capítulo es abordar el proceso de desarrollo del módulo para la detección de plagio en el COJ, específicamente en las etapas de análisis, diseño e implementación utilizando la metodología de desarrollo RUP. Se explica la elaboración de un modelo de dominio, que da lugar a la identificación de los requisitos funcionales y no funcionales con los cuales debe cumplir la aplicación, así como los casos de uso del sistema. Se muestran además los modelos de clases del análisis, con sus respectivos diagramas de colaboración, el modelo de diseño y como parte de este los diagramas de clases del diseño. Se define la arquitectura y se hace referencia a los principales patrones de diseño utilizados. El capítulo termina con una explicación detallada del perfil algorítmico, base fundamental de la investigación.

2.2 Modelo de dominio

El problema investigado está enmarcado en el Juez en Línea Caribeño, en el cual no se perciben claramente los límites del proceso de detección de plagio, haciéndose difícil determinar el conjunto estructurado de las actividades que se desarrollan en el negocio. Por estas razones se ha determinado realizar un Modelo Conceptual o Modelo de Dominio. No se desarrolla Modelo de Negocio, además, porque inicialmente no están completamente definidas las fronteras del negocio, ni quiénes son las personas que desarrollan cada una de las actividades que forman parte de sus procesos. El Modelo Conceptual es una representación visual de los objetos, conceptos y procesos del mundo real que resultan de interés para el problema, proveniente de la base de conocimientos de expertos, o de conocimiento asociado a sistemas similares. En este caso fueron los administradores, parte del equipo de

desarrollo del COJ y características identificadas en el estudio del arte, los que posibilitaron la selección de las clases del dominio que conforman dicho modelo.

2.2.1 Clases del Modelo de Dominio:

Usuario: Persona registrada en el COJ, que tiene permisos concedidos para enviar intentos de soluciones al sistema.

COJ: Juez en Línea de Programación que tiene asociado usuarios y un conjunto permanente de problemas.

Problema: Situación a resolver mediante la concepción de un algoritmo escrito en uno de los leguajes de programación soportados por el Juez en Línea Caribeño (COJ).

Sumisión: Solución en forma de código fuente que da un usuario a uno de los problemas publicados.

Inspector de plagio: Usuario con permisos para chequear las sumisiones en busca de plagio.

Reporte de plagio: Es la información que se genera en el proceso de inspección de las sumisiones.

Juez de plagio: Usuario con permiso para emitir criterios sobre los resultados generados, experto que revisa el reporte de plagio y dictamina si realmente está en presencia de soluciones plagiadas.

Código fuente: Conjunto de líneas que conforman un bloque de texto, escrito por el usuario según las reglas sintácticas y semánticas de algún lenguaje de programación, su objetivo es intentar dar solución a un problema.

En la (Figura 9) se presenta el diagrama del modelo de dominio correspondiente al sistema en desarrollo.

2.3 Requerimientos

Luego de analizarse el dominio del problema, es necesario definir lo que hará el sistema, para ello se determinan los requisitos que, según Jacobson (30), no son más que las condiciones o capacidades que tienen que ser alcanzadas por un sistema para satisfacer las necesidades del cliente. Los mismos se clasifican en requisitos funcionales y no funcionales. Los requisitos funcionales son capacidades o

condiciones que el sistema debe cumplir, mientras que los no funcionales son propiedades o cualidades que el producto debe tener (30).

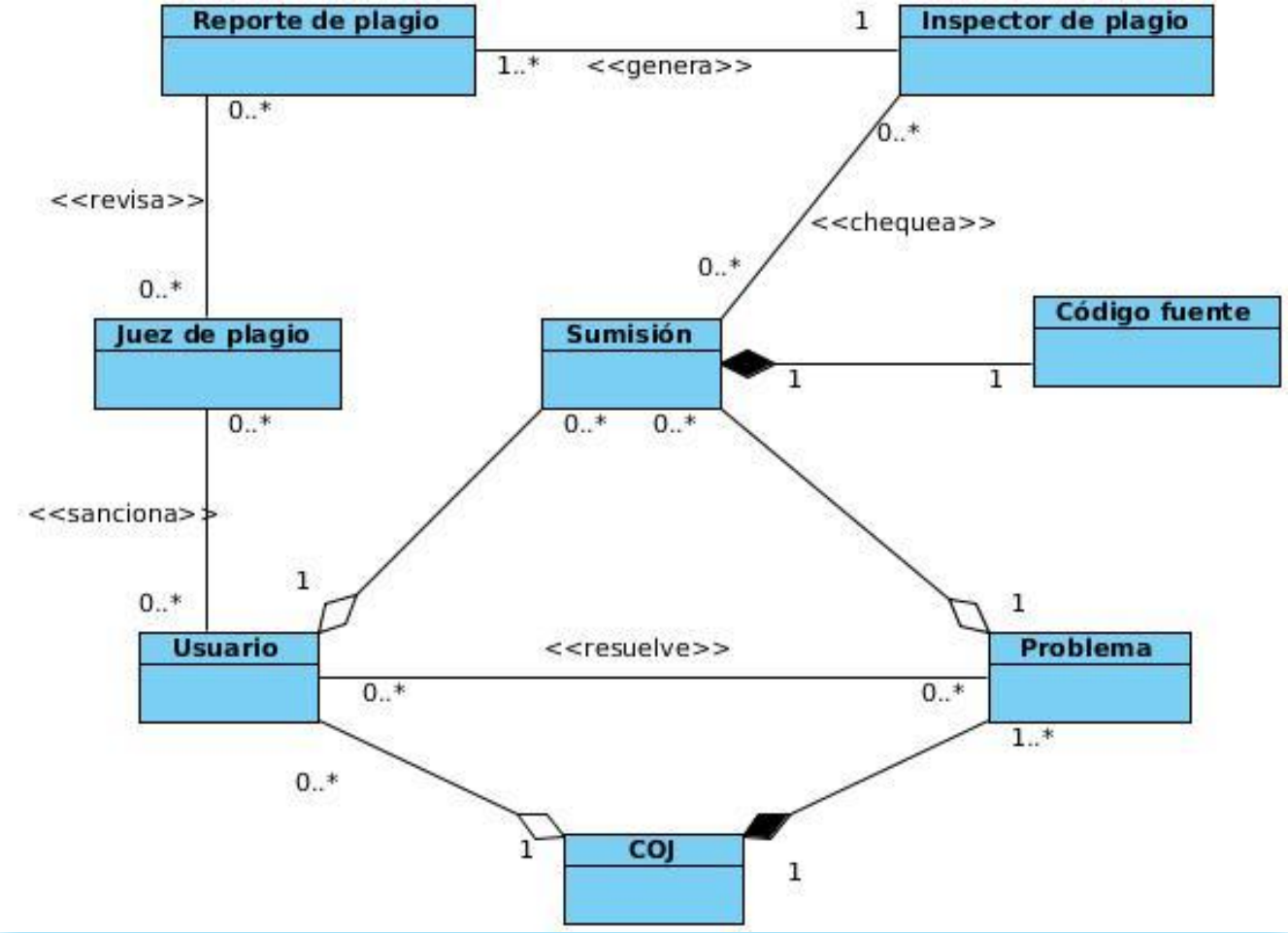


Figura 9: Diagrama del modelo de dominio

A continuación se presentan los requisitos funcionales y no funcionales que han sido identificados y que deberá cumplir la aplicación.

2.3.1 Requisitos funcionales

El sistema debe:

RF1: Permitir al usuario inspeccionar plagio entre un conjunto de sumisiones.

RF2: Permitir al usuario inspeccionar plagio entre las sumisiones de un problema.

RF3: Permitir al usuario inspeccionar plagio entre una sumisión y un conjunto de sumisiones.

RF4: Permitir al usuario inspeccionar plagio en un rango de sumisiones.

RF5: Mostrar los resultados de la inspección de plagio realizada por el usuario.

RF6: Mostrar el resultado de la detección de plagio entre dos sumisiones.

RF7: Mostrar el resultado de un algoritmo para la detección sobre dos sumisiones.

RF8: Calcular la probabilidad de plagio entre sumisiones.

RF9: Calcular la confiabilidad de los resultados de la detección de plagio entre sumisiones.

RF10: Mostrar similitudes entre códigos.

RF11: Permitir al usuario seleccionar una sumisión del conjunto de sumisiones actuales del COJ.

RF12: Permitir al usuario seleccionar un problema del conjunto de problemas existentes en el COJ.

RF13: Permitir al Juez de plagio bloquear al usuario como sanción por el plagio cometido.

RF14: Permitir al Juez de plagio desbloquear al usuario sancionado.

RF15: Permitir al Juez de plagio enviar una notificación al usuario ante su posible implicación en un caso de fraude detectado.

RF16: Permitir al Juez de plagio emitir un criterio sobre los resultados de las detecciones. El Juez de plagio dejará plasmado en su revisión una evaluación y las observaciones de la misma.

RF17: Permitir al Juez de plagio eliminar los criterios emitidos.

2.3.2 Requisitos no funcionales:

Los requerimientos no funcionales para la aplicación se centran en lograr un funcionamiento eficiente del sistema con vistas a una rápida aceptación por parte de los usuarios. Además, están determinados por la

naturaleza del software y del hardware sobre el cual será desplegada. Como parte de los anexos se muestran una descripción de los principales requisitos no funcionales de la aplicación a desarrollar (35).

2.4 Modelo de casos de uso del sistema

2.4.1 Actores del sistema

Un actor es una persona o proceso externo al sistema que interactúa con él, pueden representar el rol que juega una o varias personas, un equipo o un sistema automatizado (30). En la (Tabla 1) se presentan los actores del sistema en desarrollo y su descripción.

Tabla 1: Actores del sistema

Actor	Descripción
Inspector de plagio	Actor que tiene la tarea de detectar plagio en las sumisiones de los usuarios. Si encuentra algún incidente de plagio genera un reporte de plagio.
Juez de plagio	Usuario con permiso para emitir criterios sobre los resultados generados, experto que revisa el reporte de plagio y dictamina si realmente está en presencia de soluciones plagiadas.

2.4.2 Diagramas de casos de uso de sistema

Los diagramas de casos de uso documentan el comportamiento del sistema desde el punto de vista del usuario. Cuenta con dos tipos usuarios, cada uno se representa mediante un actor. Los actores utilizan el sistema al interactuar con los casos de uso (CU). Un caso de uso es una secuencia de acciones que el sistema lleva a cabo para ofrecer un resultado de valor para un actor, es decir, un caso de uso proporciona un resultado observable para el usuario (30). Los casos de uso determinan los requisitos funcionales del sistema y representan las funciones puede ejecutar. Todos los actores y casos de uso del sistema forman un Modelo de Casos de Uso. La ventaja principal de este modelo es la facilidad para interpretarlos, lo que hace que sean especialmente útiles en la comunicación con el cliente, los usuarios y los desarrolladores de la aplicación.

En la (Figura 10) se muestra el Modelo de Casos de Uso del módulo para la detección de plagio en el COJ.

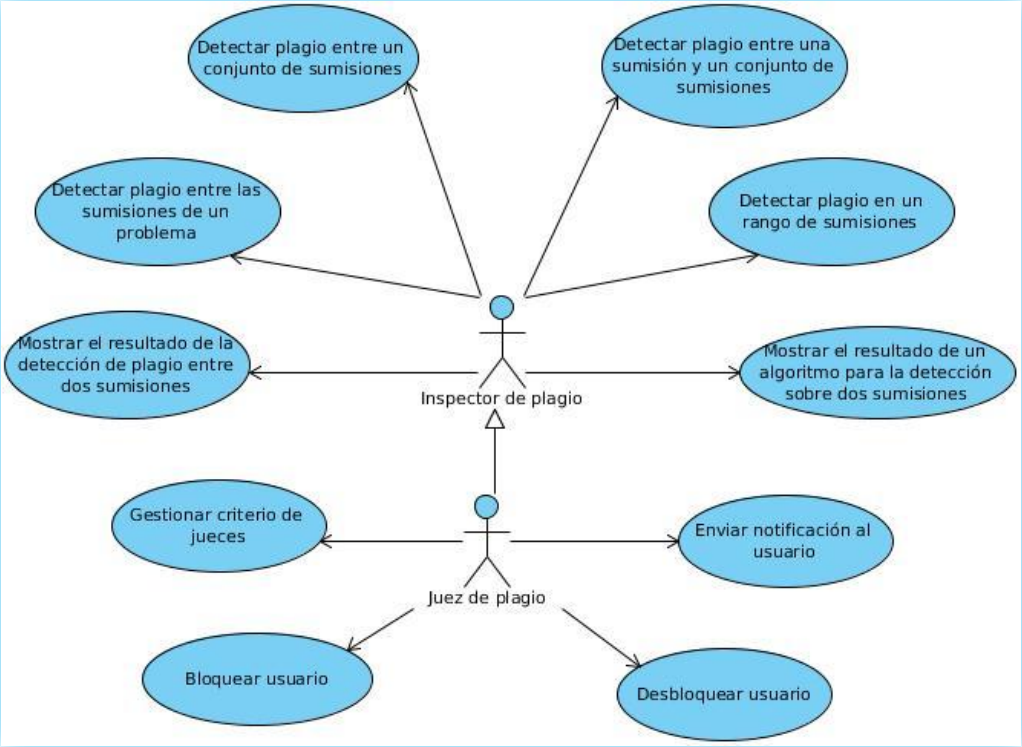


Figura 10: Diagrama del Modelo de Casos de Uso.

2.4.3 Descripción textual de los casos de uso del sistema

Las descripciones de los casos de uso del sistema contienen los detalles del flujo de eventos a ejecutar, el actor que lo inicializa, las precondiciones y poscondiciones del caso de uso, así como las referencias a los requisitos funcionales que los satisfacen. Las mismas son fundamentales en la comunicación exitosa del equipo de desarrollo.

A continuación, se presenta la descripción textual del CU1 Detectar plagio entre un conjunto de sumisiones en la (Tabla 2), el resto de las descripciones textuales de los restantes CU se encuentran en el (Anexo 2) de la (Tabla 12) a la (Tabla 20).

Tabla 2: CU1 Detectar plagio entre conjunto de sumisiones.

Caso de Uso:	Detectar plagio entre un conjunto de sumisiones.
Actores:	Inspector de plagio
Resumen:	El caso de uso inicia cuando el inspector de plagio solicita detectar plagio entre un conjunto de sumisiones del panel Herramientas de administración, posteriormente selecciona la opción detectar plagio. El sistema detecta el plagio y finaliza el caso de uso.
Precondiciones:	Usuario autenticado y con permisos de administración.
Referencias	RF1, RF6, RF9, RF12
Prioridad	Alta
Flujo Normal de Eventos	
Acción del Actor	**Respuesta del Sistema**
1. Selecciona la opción **Detectar plagio entre un conjunto de sumisiones** que pertenece a Detectar plagio dentro de las herramientas de administración.	2. Muestra los campos para la inserción de los identificadores de las sumisiones que se desean inspeccionar.
3. Inserta los identificadores de las sumisiones que desea inspeccionar. 4. Oprime **Detectar plagio**	5. Comprueba que los campos no estén vacíos, además que los identificadores solo contengan números.
	6. Visualiza los resultados de la inspección de plagio a través de una tabla, que muestra el porcentaje de plagio entre las sumisiones.
	7. Termina el caso de uso
Flujos Alternos	

Acción del Actor	**Respuesta del Sistema**
3a Selecciona los identificadores de la lista de sumisiones.	
	3a.1. Regresa al paso 5 del Flujo Normal de Eventos y finaliza el caso de uso.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
4a Campos Vacíos.	
	4a.1. Muestra en mensaje indicando que no deben haber campos vacíos
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
4b Identificadores solo contienen números.	
	4b.1. Muestra un mensaje indicando datos incorrectos.
Flujos Alternos	
Acción del Actor	**Respuesta del Sistema**
5a Selecciona la opción resetear valores.	
	5a.1. Limpia los campos para la inserción nuevamente de los identificadores de las sumisiones.
	5a.2. Regresa al paso 3 del Flujo Normal de Eventos y finaliza el caso de uso.
Poscondiciones	Se han mostrado y almacenado en la base de datos los resultados de la detección de plagio entre las sumisiones seleccionas.

2.5 Modelo de análisis

El análisis realizado consiste en obtener un enfoque más detallado del sistema que se preocupa de ver que hace y especificar qué debe hacer, teniendo en cuenta los requisitos funcionales ya establecidos. El resultado del análisis es el Modelo de Análisis. La importancia de hacer este modelo está dada por las facilidades que brinda al equipo de desarrollo, pues refina los requisitos, da una visión interna del sistema a los desarrolladores y por eso está descrito con su lenguaje.

El Modelo de Análisis constituye la primera aproximación al Modelo de Diseño, es el resultado del análisis de los casos de uso, instanciados en el artefacto diagrama de clases del análisis que está conformado por las clases del análisis (interfaz, control y entidad) las cuales encapsulan las diferentes funcionalidades que representan los casos de uso. Proporciona una estructura centrada en el mantenimiento de aspectos tales como la flexibilidad y la reutilización. Además hace abstracciones, evita resolver algunos problemas y tratar algunos requisitos que serán pospuestos para el diseño y la implementación (30).

2.5.1 Diagramas de clases del análisis

Un diagrama de clases es un tipo de diagrama estático, se utiliza en la descripción de la estructura del sistema en desarrollo mostrando sus clases del análisis, el actor con el que interactúan y las relaciones entre ellas. Los estereotipos empleados en las clases del análisis Interfaz, Control y Entidad se muestran en la (Figura 11).

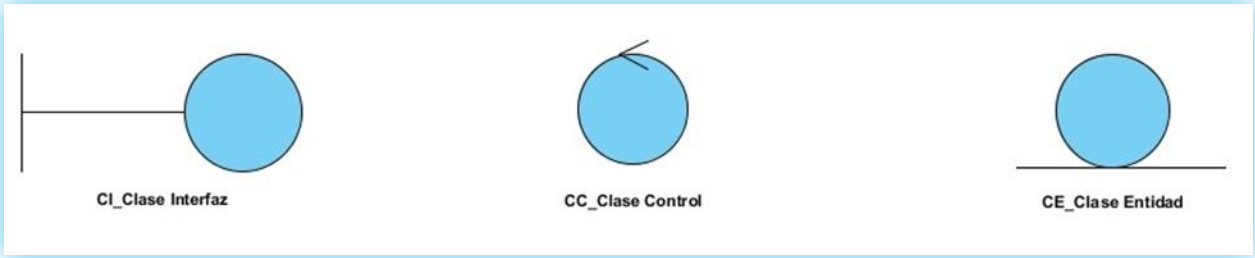


Figura 11: Estereotipos de las clases del análisis

A continuación se presenta el Diagrama de Clases del Análisis (DCA) del CU Detectar plagio entre un conjunto de sumisiones en la (Figura 12) el resto de los diagramas se encuentran en el (Anexo 3) de la (Figura 23) hasta la (Figura 32)

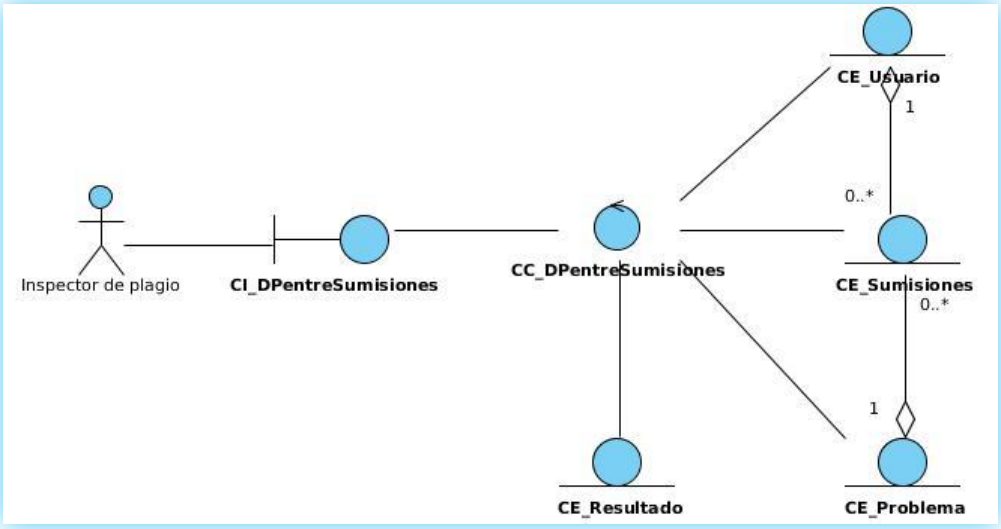


Figura 12: Diagrama de clases del análisis CU1 Detectar plagio entre un conjunto de sumisiones.

2.5.2 Diagramas de interacción

En el análisis, las interacciones entre objetos se pueden representar a través de diagramas de colaboración o de secuencia. Para la solución del módulo en desarrollo se emplean los diagramas de colaboración, pues el objetivo fundamental es identificar los requisitos y las responsabilidades sobre los objetos, y no identificar secuencias de interacción detalladas y ordenadas cronológicamente, para lo cual serían más factibles los diagramas de secuencia.

Los estereotipos empleados se muestran en la (Figura 11) Interfaz, Control y Entidad respectivamente. A continuación se presenta el diagrama de colaboración correspondiente al CU Detectar plagio entre un conjunto de sumisiones en la (Figura 13), el resto de los diagramas se encuentran en el (Anexo 4) de la (Figura 33) hasta la (Figura 41).

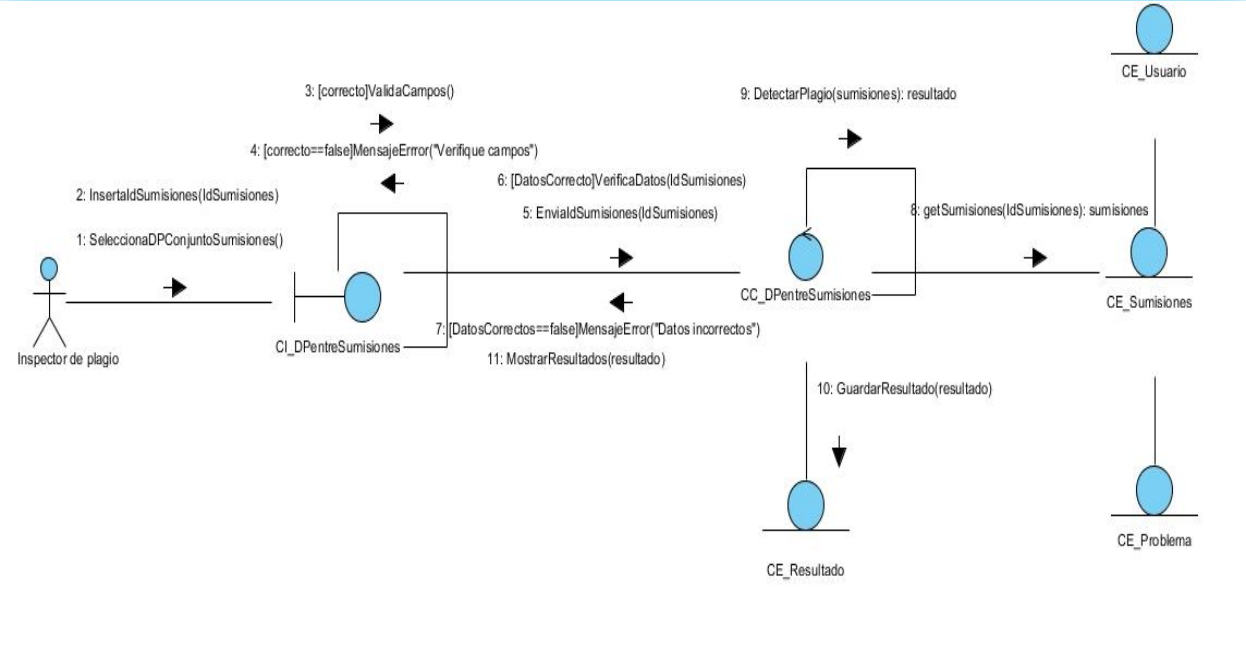


Figura 13: Diagrama de colaboración CU1 Detectar plagio entre conjunto sumisiones

2.6 Diseño

El diseño realizado es un refinamiento del análisis que tiene como propósito principal conjugar las restricciones impuestas por los requisitos no funcionales con el ambiente de implementación, define cómo cumple el sistema sus objetivos. El diseño propuesto es suficiente, dando como resultado un modelo preciso del producto final con un cubrimiento total de los requerimientos del sistema y asegurando que pueda ser implementado sin imprecisiones (30).

El artefacto más importante dentro de la disciplina de diseño es el modelo de diseño que se crea tomando el modelo del análisis como entrada principal, y se adapta a un entorno de implementación particular. El modelo de diseño es un modelo de objetos que describe la realización de los CU, y sirve como una abstracción del modelo de implementación y el código fuente (30). Es usado para concebir un documento del diseño del sistema de software. Es abarcador, compuesto por artefactos que engloban todas las clases del diseño, subsistemas, paquetes, colaboraciones, y las relaciones entre ellos.

2.6.1 Diagramas de clases del diseño

Los estereotipos empleados para la confección de los diagramas de clases del diseño con extensiones web son: Server Page, Client Page, Form, las relaciones y asociaciones especiales entre las páginas clientes, las páginas servidoras y los formularios Build, Link, Redirect, Submit e Include (Tabla 3).

Tabla 3: Leyenda de los estereotipos web.

Estereotipos	Icono	Descripción
<<Server Page>>		Representa la página web que tiene código que se ejecuta en el servidor.
<<Client Page>>		Una instancia de Página Cliente es una página web, con formato HTML. Mezcla de datos, presentación y lógica. Son interpretadas por el navegador.
<<Form>>		Colección de elementos de entrada que son parte de una página cliente.
Relaciones		**Descripción**
<<Build>>		Asociación especial que relaciona las páginas cliente con las páginas servidor, expresa como las páginas que se encuentran en el servidor construyen las páginas en el cliente.
<<Link>>		Se refiere a la asociación del hipervínculo; esta siempre se origina desde una página cliente y apunta hacia otra página cliente o una página de servidor.
<<Redirect>>		Es la acción de redireccionar el procesamiento de una página a otra.
<<Submit>>		Es la relación entre una página servidor y un formulario, a través de esta asociación el formulario manda los valores de sus campos al servidor, para ser procesados
<<Include>>		Es la relación, cuando una página servidor incluye a otra página servidor.

A continuación, se muestra el Diagrama de Clases del Diseño (DCD) con estereotipos web correspondientes al CU Detectar plagio entre un conjunto de sumisiones en la (Figura 14), el resto de los diagramas se encuentran en el (Anexo 5) de la (Figura 42) hasta la (Figura 50).

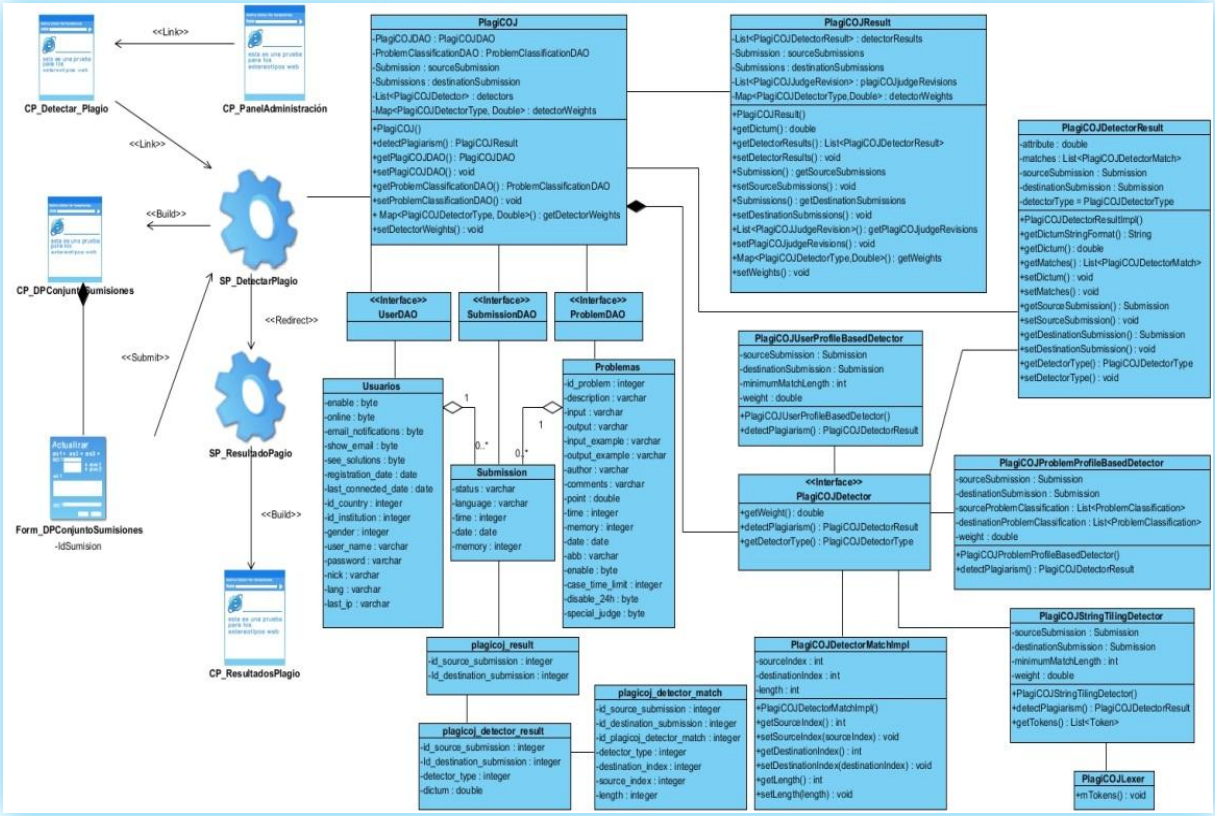


Figura 14: Diagrama de clases del diseño. CU1 Detectar plagio entre un conjunto de sumisiones.

2.6.2 Modelo de datos

El Modelo de Datos es el lenguaje orientado a describir la Base de Datos, permite además describir los elementos de la realidad que intervienen en el problema dado y la forma en que se relacionan estos elementos entre sí (Figura 15).

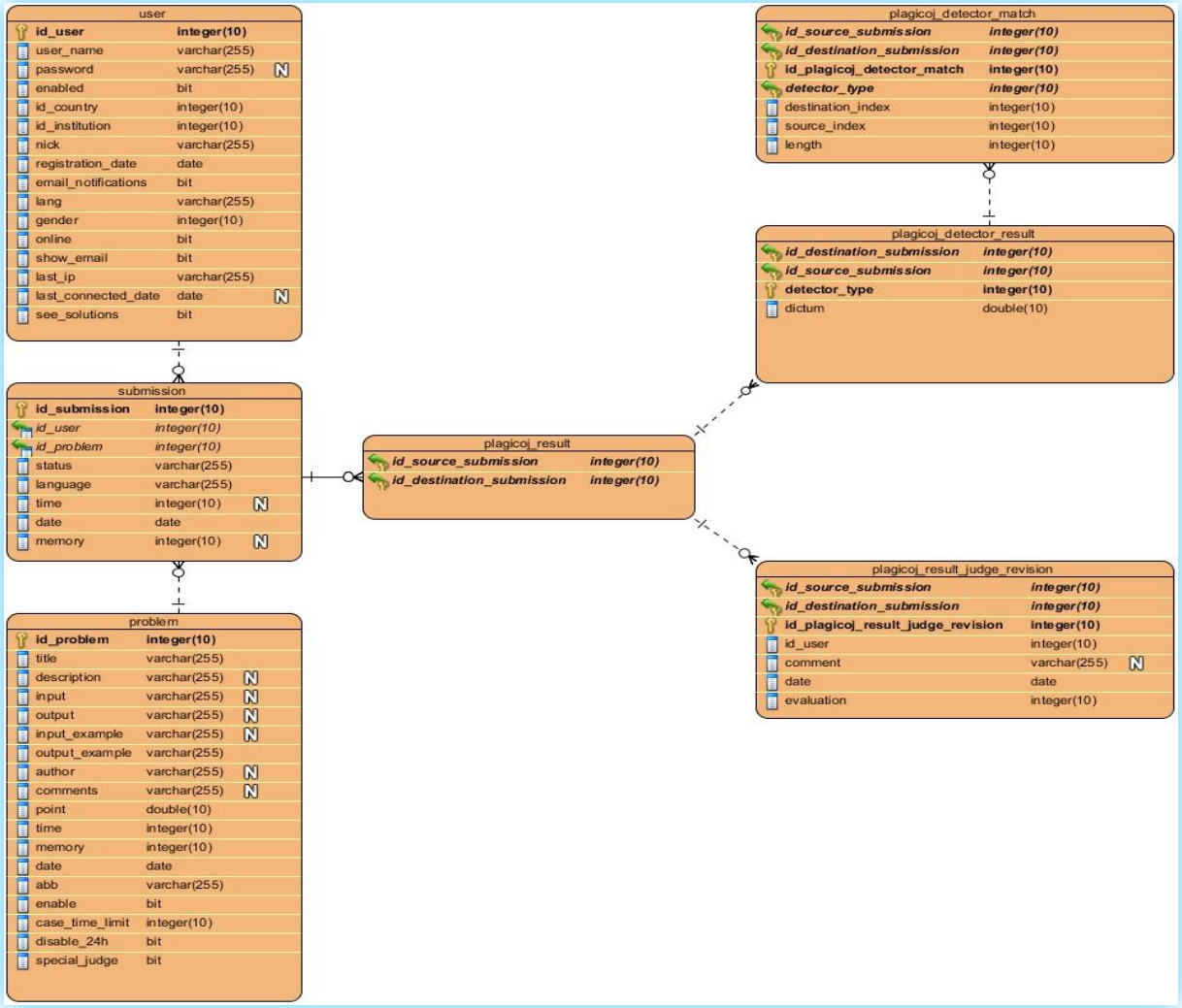


Figura 15: Modelo de datos.

2.7 Aplicación de patrones de diseño.

Usualmente los desarrolladores de software por su experiencia en el tema, acumulan un repertorio tanto de principios generales como de soluciones apoyadas en la aplicación de ciertos estilos que los guían en la creación de software, siendo estos principios y estilos denominados patrones de software (46).

Los patrones de diseño son soluciones simples y elegantes a problemas específicos y comunes del diseño, basados en la experiencia y en la evidencia de su funcionamiento.

En el diseño de la aplicación se hace uso de los patrones GRASP; acrónimo de *General Responsibility Assignment Software Patterns,* patrones generales de software para asignar responsabilidades (46), destacándose los que se mencionan a continuación.

El patrón **Experto**: Se basa en que la responsabilidad de realizar una tarea es de la clase que posee los datos involucrados; una clase contiene toda la información necesaria para realizar la tarea que tiene encomendada.

El patrón **Controlador**: Se asocia con operaciones del sistema y respuestas a sus eventos, tal como se relacionan los mensajes y los métodos. El controlador delega en otros objetos el trabajo que se necesita hacer pero coordina o controla la actividad.

El patrón **Alta Cohesión**: Se manifiesta en la medida en que se relacionan las clases y el grado de focalización de las responsabilidades de un elemento. Cada elemento del diseño debe realizar una labor única dentro del sistema, no desempeñada por el resto de los elementos y autoidentificable, una clase con baja cohesión hace muchas cosas no relacionadas o hace demasiado trabajo.

El patrón **Bajo Acoplamiento**: Asigna la responsabilidad de controlar el flujo de eventos del sistema a clases específicas. Esto facilita la centralización de actividades como las validaciones y la seguridad de la aplicación.

Además se aplican algunos patrones *Gang of Four* (GoF), también conocidos como los patrones de la pandilla de los cuatro, entre los que es válido mencionar **Abstract Factory** que proporciona una interfaz para crear familias de objetos o que dependen entre sí, sin especificar sus clases concretas. Permite que las clases que la usen sean las que decidan qué clase instanciar (47). Se emplean otros patrones GoF pero están directamente incluidos en las tecnologías que se usan en el desarrollo de la aplicación como los framework Spring y Acegi. También se usa el patrón **Data Access Object** (DAO), este crea una interfaz para encapsular los accesos a las fuentes de datos.

Igualmente, en la aplicación se utiliza otro patrón de diseño que a pesar de no incluirse en ninguna categoría específica, juega un papel clave dentro de las nuevas tendencias del desarrollo de este tipo de proyectos, este es la **Inyección de dependencias** base del funcionamiento del Framework Spring**.**

2.8 Arquitectura de la aplicación.

La IEEE define la arquitectura de software como la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos, el ambiente y los principios que orientan su diseño y evolución (48).

El estilo arquitectónico o el patrón de arquitectura de software empleado para el desarrollo de la aplicación es la arquitectura Modelo Vista Controlador (MVC). Esta se basa en separar los datos de la aplicación, la interfaz de usuario, y la lógica del negocio en tres capas diferentes, es decir, que agrupa el código según su función. El modelo define el comportamiento, los datos de la aplicación y la lógica del negocio; la base de datos pertenece a esta capa. La vista muestra la información que emplean los usuarios para interactuar con la aplicación. La capa controlador es la que descifra las acciones realizadas por los usuarios, realiza las llamadas al modelo para obtener la información y se las envía a la vista para que las muestre a los usuarios. La vista y el controlador dependen del modelo, el cual no depende de las otras clases. Esta separación permite construir y probar el modelo independientemente de la representación visual (49).

Es seleccionado este estilo por las ventajas que brida su estructura, además de ser el que utiliza el COJ, plataforma donde será desplegado el módulo para la detección de plagio.

2.9 Seguridad de la aplicación

Para la seguridad de la aplicación se utiliza *Acegi Security System*, un framework creado por Ben Alex e íntimamente ligado al proyecto Spring. Comúnmente conocido como Acegi, brinda las funcionalidades necesarias para adoptar mecanismos de seguridad en aplicaciones Java utilizando características de programación orientada a aspectos (50), de forma transparente para el desarrollador, sin necesidad de escribir código, utilizando para ello el soporte prestado por el Framework Spring (51).

La arquitectura de Acegi está fuertemente basada en interfaces y en patrones de diseño, proporcionando las implementaciones más comúnmente utilizadas y numerosos puntos de extensión donde nuevas funcionalidades pueden ser añadidas.

No se dejaron de utilizar técnicas recomendadas para la concepción de la seguridad, particularmente la implantación de diferentes niveles de seguridad, tal y como quedó establecido en los requisitos no funcionales. En la aplicación el acceso a las diferentes funcionalidades se valida a la hora de generar las vistas y también a nivel de invocaciones de método, garantizando que la aplicación no haga una llamada a ningún método con instrucciones que el usuario autenticado no tenga derecho a ejecutar.

En el acceso a la base de datos se previene la inyección SQL con el método previsto por la clase JdbcTemplate de Spring (Figura 16).

```
public void insertPlagiCOJResultJudgeRevisions(
    int sourceSubmissionId,
    int destinationSubmissionId,
    String userId,
    int evaluation,
    String comment) {
        getJdbcTemplate().update(
    "INSERT INTO plagicoj_result_judge_revision VALUES(?,?,?,?,?)",
        new Object[]{
            sourceSubmissionId,
            destinationSubmissionId,
            userId,
            evaluation,
            comment
        });
}
```

Figura 16: Ejemplo de prevención contra inyección SQL.

2.10 Perfil algorítmico

La base algorítmica se debe sustentar en tres criterios que al complementarse generen un dictamen. Se implementará un detector para el análisis de la estructura del código fuente basado en el algoritmo Greedy-String-Tilling (22), otro basado en las características del problema y un último detector cuyo criterio decida la probabilidad de que un usuario, según su historial, pueda haber cometido plagio.

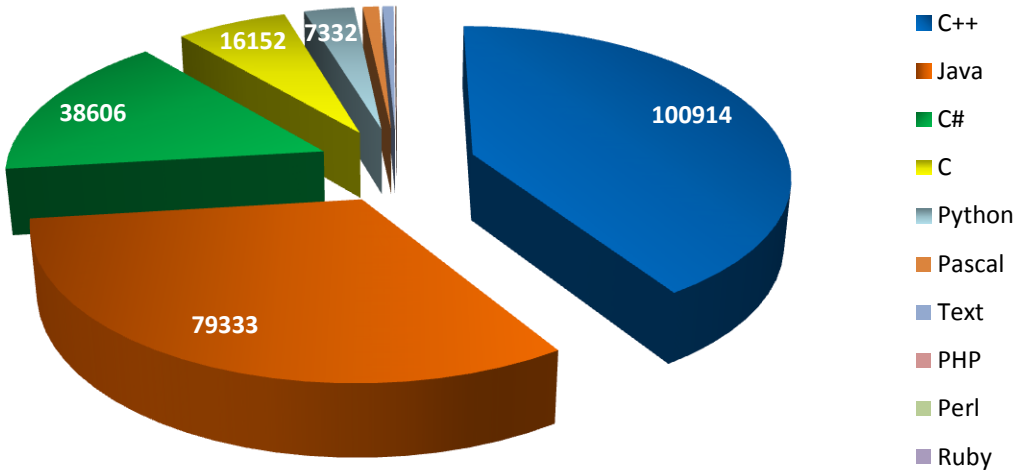
2.10.1 Detector de plagio basado en la estructura del código fuente.

Con respecto al algoritmo Greedy-String-Tilling (Figura 17) puede ser encontrado en varias bibliografías por ejemplo en (22); no siendo este el caso de los otros que fueron desarrollados desde cero, sobre la base de la experiencia en el campo de acción y criterios no necesariamente complejos; pero que sirven para equilibrar la ineficacia de los resultados obtenidos con los sistemas de detección existentes al aplicarlos al entorno de los jueces en línea.

```
do {
    maxMatch = getMinimumMatchLength();
    ArrayList<PlagiCOJDetectorMatchImpl> matches
    = new ArrayList<PlagiCOJDetectorMatchImpl>();
    for (int i = 0; i < lenghtSource; i++) {
        for (int j = 0; j < lenghtDestination; j++) {
            int length = 0;
            while (i + length < lenghtSource
                    && j + length < lenghtDestination
                    && sourceCodeTokens.get(i + length).getType()
                    == destinationCodeTokens.get(j + length).getType()
                    && !marksSource[i + length]
                    && !marksDestination[j + length]) {
                ++length;
            }
            if (length == maxMatch
                    && (matches.isEmpty() || (matches.get(
                            matches.size() - 1).getSourceIndex()
                            + length <= i && matches.get(
                            matches.size() - 1)
                            .getDestinationIndex()
                            + length <= j))) {
                matches.add(new PlagiCOJDetectorMatchImpl(i, j,
                        length));

            } else if (maxMatch < length) {
                maxMatch = length;
                matches.clear();
                matches.add(new PlagiCOJDetectorMatchImpl(i, j,
                        length));
            }

        }
    }

    for (int i = 0; i < matches.size(); i++) {
        for (int j = 0; j < matches.get(i).getLength(); j++) {
            marksSource[matches.get(i).getSourceIndex() + j] = true;
            marksDestination[matches.get(i).getDestinationIndex()
                    + j] = true;
        }
        tiles.add(matches.get(i));
    }
} while (maxMatch > getMinimumMatchLength());
```

Figura 17: Algoritmo Greedy-String-Tiling utilizado para la detección de plagio en la estructura del código.

Se implementaron analizadores léxicos para los lenguajes de programación C, C++, Java, Pascal, Python y CSharp, que son los de mayor uso en el COJ representando el 99.4% del total de las soluciones enviadas hasta el momento al Juez en Línea (52).



Gráfica 1: Cantidad de sumisiones por lenguaje enviadas al COJ.

Con respecto a las parejas de fragmentos de código detectados, el dictamen final queda formulado según la (Ecuación 1) que a grandes rasgos expresa que la similitud entre dos códigos está dada por la razón entre "la suma de las longitudes de los fragmentos similares o *coincidencias,* por dos debido a que están presentes en ambos códigos"; sobre la logitud total (22).

Tabla 4: Leyenda para la (Ecuación 1).

Variable	Significado
A	Solución fuente.
B	Solución destino.
S	Conjunto de similitudes detectadas entre la solución A y B.
l	Longitud de la similitud.
c	Código fuente de la solución.

$$similitud(A,B) = \frac{2(\sum_{x \in S} x_l)}{|A_c| + |B_c|}$$

Ecuación 1: Similitud de acuerdo a las coincidencias detectadas por el algoritmo Greedy String Tilling.

2.10.2 Detector de plagio basado en el perfil del problema objetivo de la solución.

Con respecto al detector basado en las características del problema, se comenzó con la creación de un sistema de clasificación en el Juez en Línea (Figura 53), capaz de definir para cada problema cuales algoritmos clásicos son necesarios en su solución. Si resolver un problema implica encontrar la distancia mínima de un nodo a los demás y sobre esa distancia realizar algún tipo de procesamiento, por ejemplo ir y volver a la mayor cantidad de nodos contando con un límite de distancia total a recorrer, lo cual podría verse desde una óptica golosa; entonces será muy probable encontrar entre las clasificaciones de este problema: Dijkstra y Goloso. La comparación de las estructuras de problemas solubles mediante algoritmos clásicos necesariamente arrojará altos niveles de similitud, desde este punto de vista se considera ad-hoc como la clasificación prototipo para cualquier detección. Ad-hoc es una frase proveniente del latín que significa “para esto”, en el ámbito informático es usada con frecuencia para referenciar algoritmos cuyo desarrollo no está sujeto a ningún patrón, categoría o que tenga similitud con algo conocido y por consiguiente cuyo resultado no es generalizable.

Se estimó, según las clasificaciones de todos los problemas del Juez en Línea y las soluciones de los usuarios, la longitud promedio de código que debería emplearse para implementar determinado algoritmo. El corazón de esta idea es un procedimiento almacenado cuyo objetivo es actualizar las longitudes de código para cada clasificación (Figura 52). De acuerdo a las clasificaciones coincidentes en los ejercicios comparados, el dictamen se calcula con la (Ecuación 2), donde el denominador se refiere al máximo de las longitudes de las sumisiones enviadas al Juez en Línea.

Tabla 5: Leyenda para la (Ecuación 2).

Variable	Significado
A	Solución fuente.
B	Solución destino.
c	Código fuente.
s	Solución enviada al Juez en Línea.
f	Conjunto de clasificaciones del problema objetivo de la solución.
l	Longitud estimada.

$$similitud(A,B) = \frac{|A_c| + |B_c| - 2\sum_{F\,\epsilon\,[A_f \cap B_f]} F_l}{2\left(\underset{1\leq s\leq\infty}{max}|s_c|\right)}$$

Ecuación 2: Similitud de acuerdo a las coincidencias detectadas con el comparador de problemas.

2.10.3 Detector de plagio basado en el perfil del usuario

Por último se realiza un estudio del historial de cada usuario con el objetivo de determinar la probabilidad de que hayan cometido plagio. Dicho criterio se basa en los dictámenes realizados a otros ejercicios en los que el usuario ha estado involucrado (Ecuación 3).

Tabla 6: Leyenda para la (Ecuación 3)

Variable	Significado
A	Usuario fuente.
B	Usuario destino.
E	Cantidad de ejercicios existentes en el Juez en Línea.
d	Dictámenes de las detecciones en que el usuario ha estado involucrado.
c	Cantidad de detecciones en que el usuario ha estado involucrado.
r	Cantidad de ejercicios resueltos por el usuario.

$$similitud(A,B) = \frac{\frac{\sum_{x\,\epsilon\,B_d} x}{B_c} + \frac{B_r}{E}}{2}$$

Ecuación 3: Similitud de acuerdo al algoritmo comparador de acuerdo al historial del usuario.

Finalmente, los resultados se ponderan, se unen y se evalúan por los administradores que laboran como jueces de plagio, los cuales son los encargados de emitir criterios sobre el dictamen final de la (Ecuación 4) y los dictámenes de las detecciones independientes (Figura 55), el objetivo de esos criterios es enriquecer la calidad de los dictámenes del módulo en futuras versión y determinar las acciones a llevar a cabo con los usuarios involucrados en un plagio, detectado por el módulo y ratificado por los jueces. Se

brinda la funcionalidad de notificación a los usuarios, sanción temporal de denegación de servicios y sanción permanente, pérdida de su cuenta en el sistema.

Tabla 7: Leyenda para la (Ecuación 4)

Variable	Significado
A	Solución fuente.
B	Solución destino.
D	Conjunto de detectores aplicados a las soluciones A y B.
d	Dictamen de la detección.
p	Ponderación del detector.

$$Dictamen(A,B) = \frac{\sum_{x \in D} x_d * x_p}{\sum_{x \in D} x_p}$$

Ecuación 4: Dictamen general del módulo para la detección de plagio.

2.11 Conclusiones

El capítulo presentó la concepción del módulo para la detección de plagio en el COJ. Se llevó a cabo por su factibilidad la elaboración del Modelo de Domino a partir de los conceptos fundamentales que giran en torno al problema, que constituyó la entrada a la identificación de los requerimientos funcionales y no funcionales que debe cumplir la aplicación. Estos elementos permitieron la creación de los casos de uso representados en el Modelo de Casos de Uso del Sistema, así como las descripciones textuales de cada uno de ellos. Luego se procedió con los flujos de análisis y diseño, exponiéndose los artefactos principales, tales como diagramas de clases de análisis, diagramas de colaboración, y diagrama de clases de diseño utilizando estereotipos web, entre otras actividades fundamentales como antesala a la etapa de Implementación y Prueba. Se hizo una selección de las mejores prácticas a aplicar dentro de cada una de estas disciplinas, como la aplicación de patrones de diseño, la selección del estilo arquitectónico y la seguridad de la aplicación. Se realizó el perfil algorítmico, arrojando algoritmos eficientes y eficaces, base fundamental para la exitosa implementación del módulo para la detección de plagio en el Juez en Línea Caribeño.

Implementación y Prueba

3.1 Introducción

Continuación se documentan los aspectos fundamentales de la implementación y validación del módulo para la detección de plagio. Se describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo éstos se organizan de acuerdo a los nodos específicos en el modelo de despliegue. Una vez terminadas las actividades de implementación, se efectuarán las pruebas que se le realizarán al sistema desarrollado para validar los resultados obtenidos. El peso del flujo de trabajo de pruebas se centró en el sistema y en la validación de la aplicación mediante la técnica de Iadov para conocer la satisfacción, alrededor de la creación y eficiencia del módulo para la detección de plagio.

3.2 Modelo de Implementación

La implementación se comienza con el resultado del diseño y se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts, ficheros de código binario, ejecutables y similares.

El diseño realizado propuso un plano del modelo de implementación, por lo que sus últimas actividades están vinculadas a la creación del modelo de despliegue. En la implementación se describe cómo los elementos del modelo del diseño se implementan en términos de componentes y cómo estos se organizan de acuerdo a los nodos específicos en el modelo de despliegue. Los artefactos principales generados en esta etapa, son el diagrama de despliegue y el de componentes, ambos conforman lo que se conoce como el modelo de implementación, donde se describen los componentes, su organización y dependencias entre los nodos físicos en los que funcionará la aplicación (30).

La mayor parte de la arquitectura del sistema se capturó durante el diseño, siendo el propósito principal de la implementación desarrollar la arquitectura propuesta y el sistema como un todo.

3.2.1 Diagrama de despliegue

El Diagrama de Despliegue modela la arquitectura en tiempo de ejecución y muestra la disposición física de los nodos que componen el sistema. El mismo presenta la configuración de los elementos de hardware (nodos) y muestra cómo los elementos y artefactos del software se trazan en esos nodos y se encuentran conectados por enlaces de comunicación. Un nodo es un recurso de ejecución tal como un computador, un dispositivo o memoria, es decir, un elemento de hardware o software. Estos se representan con la forma de una caja en tres dimensiones. Las conexiones que se establecen son asociaciones de comunicación entre los nodos, y se representan mediante un estereotipo que identifica el protocolo de comunicación que el mismo utiliza.

El diagrama que aparece a continuación muestra las relaciones físicas entre los componentes hardware y software en el sistema final. Como parte del diagrama de despliegue de la aplicación actual, el módulo para la detección de plagio se ubicará en el mismo servidor donde se encuentra el COJ. Los usuarios que lo usarán, accederán mediante el Juez en Línea Caribeño. Como protocolo de comunicación entre el servidor web y el servidor de base de datos se emplea TCP/IP (*Transmition Control Protocol*/*Internet Protocol*), este provee la transferencia confiable de paquetes de datos e información sobre la red. Para la conexión al servidor web desde las computadoras clientes se utiliza HTTPs (*HyperText Transfer Protocol*, Protocolo de Transferencia de Hipertexto), el mismo implementa un canal de comunicación seguro, basado en SSL (*Secure Socket Layers*) entre el navegador del cliente y el servidor HTTP (53). El diagrama de Despliegue del Módulo para la detección de plagio se presenta en la (Figura 18).

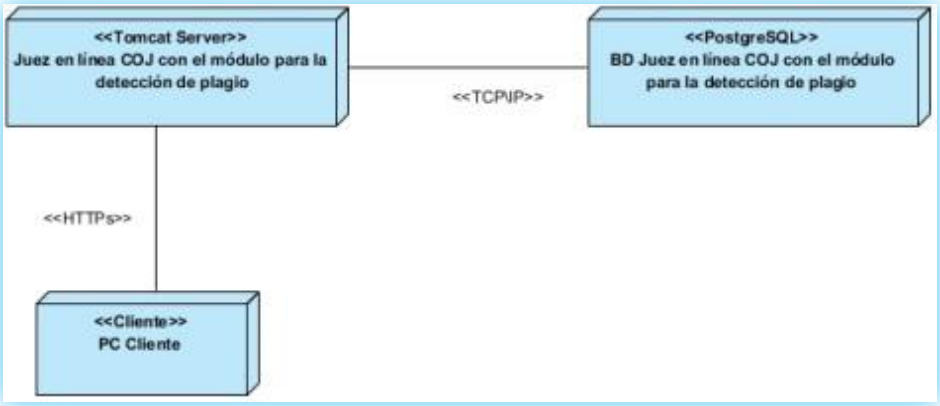


Figura 18: Diagrama de Despliegue del Módulo para la detección de plagio.

3.2.2 Diagrama de componentes

El diagrama de componentes es usado para organizar el modelo de implementación en términos de subsistemas de implementación, paquetes, componentes y mostrar las relaciones entre los elementos que conforman la implementación. El uso más importante de estos diagramas es mostrar la estructura de alto nivel del modelo de implementación, especificando los subsistemas de implementación, sus dependencias a la hora de importar el código y para organizar los subsistemas de implementación en capas y en paquetes.

Mediante este diagrama se visualizan además las dependencias de compilación de los ficheros de código, relaciones de derivación entre ficheros de código fuente y ficheros que son resultados de la compilación, dependencias entre los elementos de la implementación y los correspondientes elementos de diseños que son implementados. Se usa para representar la vista estática de la aplicación, la estructura y las relaciones lógicas entre los componentes tales como código fuente, librerías, tablas de la base de datos, de forma general componentes que conforman el módulo para la detección de plagio.

Desde el punto de vista del diagrama de componentes se tienen en consideración los requisitos relacionados con la facilidad de desarrollo, la gestión del software, la reutilización, las restricciones impuestas por los lenguajes de programación y las herramientas utilizadas en el desarrollo. Los elementos de modelado dentro de un diagrama de componentes serán componentes y paquetes. En cuanto a los

componentes, sólo aparecen tipos de componentes. Las instancias específicas de cada tipo se encuentran en el diagrama de despliegue (54).

El diagrama de componentes fue elaborado de forma tal que reflejara las divisiones e interdependencias entre las clases y demás elementos. Si se explica comenzado por la adquisición de los datos y siguiendo el proceso hasta su presentación al usuario, se puede observar como el paquete inferior del diagrama (Figura 19) representa las tablas de la base de datos, a las cuales se tiene acceso solamente a través del paquete *Data Access Layer* o DAO, tal y como está previsto por el patrón del mismo nombre. Los elementos representados en este paquete son los encargados de implementar los métodos de acceso a la base de datos para la clase a la que están destinados.

En el nivel superior se refleja el núcleo del módulo para la detección de plagio, en él están los detectores, los analizadores léxicos encargados de traducir el código en formato texto a un flujo coherente de elementos sintácticos. También se encuentran un conjunto de pequeñas clases entidad para representar valores y conjuntos de datos, por ejemplo: PlagiCOJResult y PlagiCOJJudgeRevisión; representantes de los resultados del módulo y las revisiones de los Jueces respectivamente. El elemento PlagiCOJ es el encargado de controlar todo el proceso. Están representadas además las Interfaces creadas para la mayoría de las clases.

En una capa independiente están representados los Controladores o *Controllers,* que de acuerdo al estilo arquitectónico MVC se encargan de enlazar los resultados del modelo con las vistas, brindando una interfaz que asegura la independencia en estos procesos.

Por último se representa como un conjunto de pequeños paquetes las vistas de la aplicación unidas a otros elementos como las librerías, mayoritariamente complementarias a Spring. Los ficheros de configuración son también representados en un paquete independiente, la mayoría de estos elementos son ficheros XML encargados del mapeo de las dependencias. Además se pueden observar los ficheros JavaScript organizados junto al resto de los pertenecientes al COJ y los ficheros de estilo o *Cascading Style Sheets* (CSS) almacenados independientemente con el objetivo de mantener los diseños los más claros posibles y apartar las cuestiones estilísticas y las de presentación neta de valores, de esta manera ambos proceso pueden atacarse por separado y en la práctica ha generado mejores resultados que la introducción del estilo en lo elementos HTML.

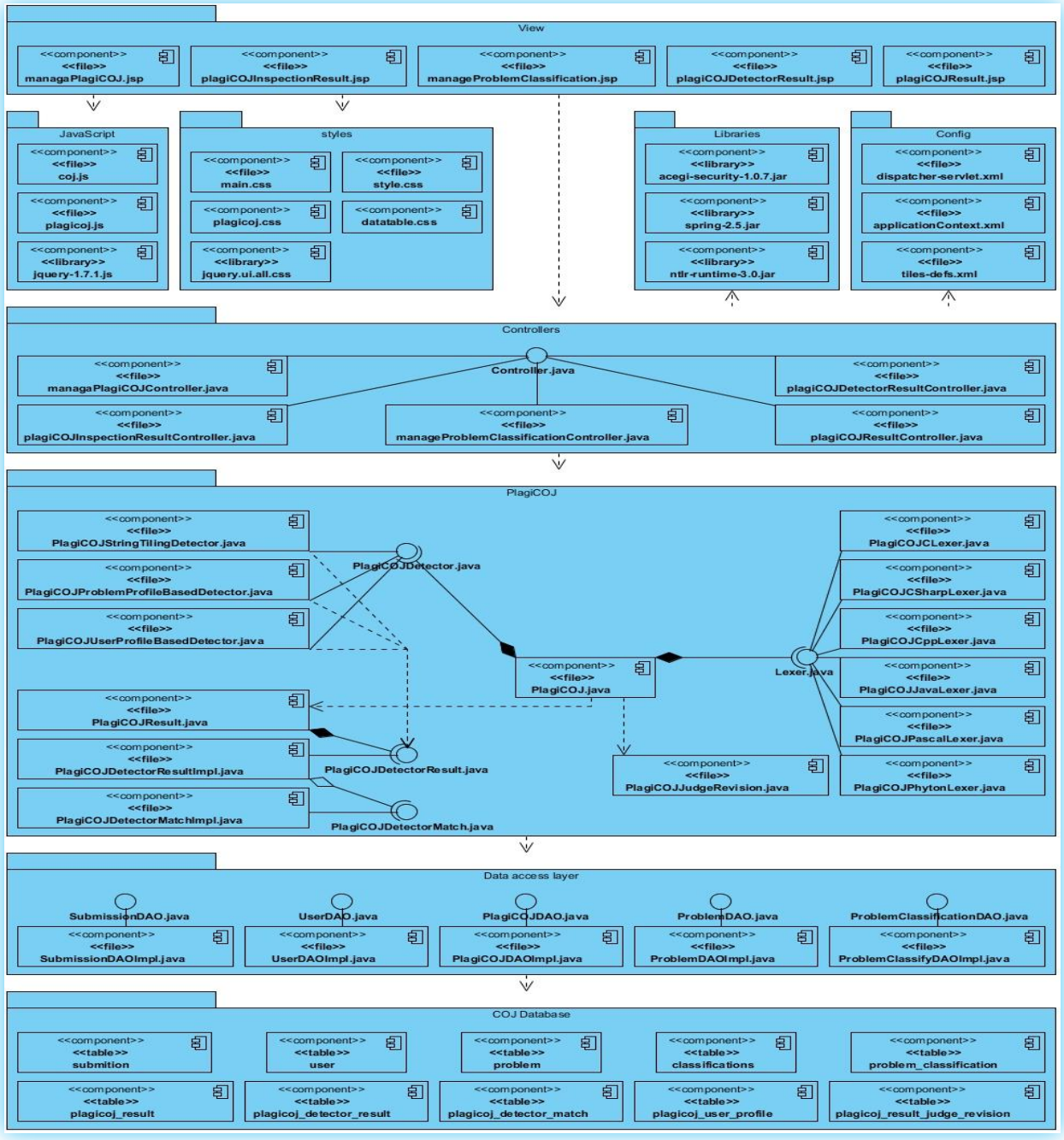


Figura 19: Diagrama de Componentes del módulo para la detección de plagio.

3.3 Reseña de la Implementación

La implementación del módulo comenzó con la creación de la vista para la selección de las sumisiones que se desean procesar y su inclusión dentro de los vínculos del panel administrativo (Figura 20). Este sería el primer paso para el acceso a las posibilidades de detección de plagio y la pantalla principal del módulo.

Figura 20: Vínculo al módulo para la detección de plagio en el panel de administración del COJ

Al describir el proceso teniendo en cuenta la arquitectura, lo primero que se debería señalar es la creación de las tablas previstas en el modelo de datos y la clase **PlagiCOJDAO** con los métodos necesarios para dar soporte a las consultas que se deseen hacer a la base de datos.

Las clases encargadas de la lógica del negocio se organizaron en paquetes según sus relaciones y dependencias (Figura 21). El paquete padre se nombró **PlagiCOJ** y dentro de él se crearon espacios para los Analizadores Léxicos, detectores y las revisiones de los jueces, por mencionar algunos.

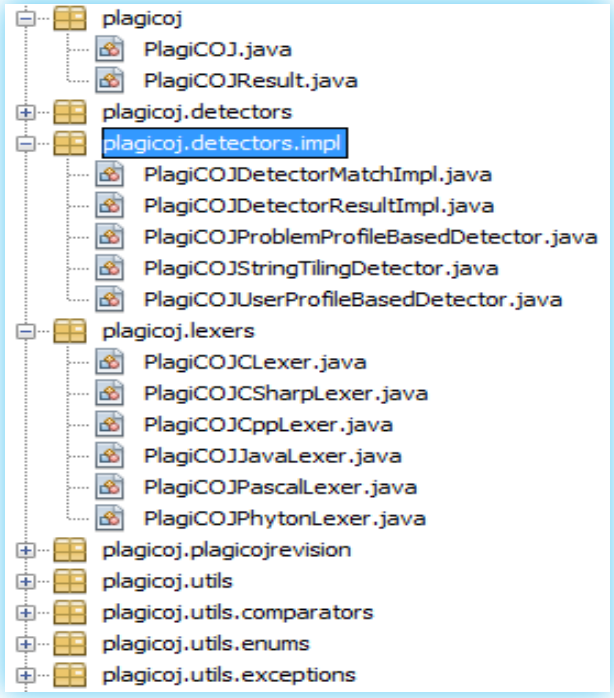


Figura 21: Estructura organizativa en paquetes.

En cuanto al soporte para diferentes lenguajes de programación se implementaron Analizadores Léxicos para los lenguajes C, CSharp, C++, Java, Pascal y Python, representando estos el 99.4% del total de sumisiones que se han realizado al Juez en Línea. La estructura de tokens generada es compatible por lo que es posible la comparación estructural entre códigos escritos en lenguajes de programación diferentes.

Se implementaron tres detectores. El **PlagiCOJStringTilingDetector** basado en el algoritmo Greedy-String-Tiling y su principal función es ofrecer un dictamen de acuerdo a la estructura del código fuente de

las soluciones. El **PlagiCOJProblemProfileBasedDetector** encargado de establecer una diferencia entre códigos utilizados para resolver un problema **ad-hoc** del cual no se tiene un patrón de solución y algoritmos clásicos cuyo conocimiento implicará que la mayoría de las soluciones sean similares. Por último, dentro del alcance previsto inicialmente para el proyecto, se implementó el **PlagiCOJUserProfileBasedDetector** que establece el tratamiento según el historial del usuario, diferenciado a quién nunca ha cometido fraude del que lo comete a menudo.

Para el diseño de las vistas se crearon los ficheros Java Server Page (JSP) los cuales fueron mapeados en los ficheros de configuración de Spring a sus correspondientes controladores. Ambos ficheros fueron almacenados, de acuerdo a la estructura prevista por el COJ, dentro de una carpeta con nombre **plagicoj.** Para la internacionalización se utilizó el mecanismo orientado a contextos de Spring dando soporte para dos idiomas español e inglés. El uso de JavaScript en el lado del cliente se apoyó en el framework jQuery en su versión 1.7 y algunas tareas, principalmente de estilo, se hicieron basadas en su interfaz de usuario (UI) versión 1.8. Se muestran algunas interfaces del módulo creado de la (Figura 53) a la (Figura 60) en el (Anexo 6).

3.4 Pruebas de software

Para que el módulo para la detección de plagio se considere listo para ser desplegado y en funcionamiento en el COJ, debe someterse previamente a una etapa de pruebas rigurosas, con el fin de analizar si cumple con las funcionalidades requeridas. Las pruebas que se le realizarán a la aplicación son un elemento crítico para la garantía de la calidad, representan además la revisión final de las especificaciones, los requerimientos, el análisis, diseño e implementación del módulo. El principal objetivo de estas pruebas es descubrir errores; estos serán corregidos posteriormente para que la aplicación final cumpla con lo previsto y tenga la eficiencia y la calidad requerida. Aunque las pruebas no pueden asegurar la ausencia de defectos; sí pueden demostrar la existencia de estos (54).

Las pruebas a la aplicación se centrarán en dos niveles fundamentales: las pruebas de unidad y las pruebas de sistema. A continuación se presenta una descripción de las pruebas realizadas al sistema y las estrategias utilizadas para validar la solución desarrollada.

3.5 Pruebas de sistema

En el proceso de realización de las pruebas al módulo para la detección de plagio, se aplica el nivel de prueba de sistema. Las pruebas de sistema se usan para verificar la aplicación final, asegurando el correcto funcionamiento como un todo y que la misma realiza las funciones requeridas.

3.5.1 Tipos de pruebas del sistema

Según el nivel establecido, se seleccionaron dos tipos de pruebas capaces de mostrar la calidad de la aplicación desarrollada. Se pretende además evaluar y verificar las funcionalidades de la misma, así como la seguridad del módulo.

Prueba de Seguridad: Intenta verificar que los mecanismos de protección incorporados en el sistema lo protegerán de accesos impropios.

Prueba funcional: Tiene como objetivo asegurar el cumplimiento apropiado de los requisitos funcionales, entrada de datos, procesamiento y obtención de resultados. La principal intención de estas pruebas es medir la correspondencia entre la arquitectura de información propuesta y las funciones que realmente fueron implementadas (54).

Para realizar este tipo de pruebas es necesario tener definidos los requerimientos a verificar con los casos de prueba afiliados a cada uno de ellos. Estos serán los encargados de verificar la aplicación implementada, se utilizará el método de caja negra.

3.5.1.1 Diseño de casos de prueba.

El método de prueba de caja negra se aplica a la interfaz de la aplicación. Pretende demostrar que las funcionalidades del módulo son operativas, las entradas se aceptan correctamente y que se producen los resultados esperados. Dentro del método de caja negra se utiliza la técnica “Partición Equivalente” siendo considerada como una de las más efectivas en la evaluación de los valores válidos, inválidos y los que no es necesario proporcionar un valor del dato en las entradas existentes en la aplicación. La Partición Equivalente divide el dominio de entrada de un programa en clases de datos a partir de las cuales se derivan casos de prueba (54).

Tabla 8: Caso de prueba del CU1 Detectar plagio entre un conjunto de sumisiones.

Escenario	Descripción	ID sumisión	Agregar ID sumisión	Respuesta del sistema	Flujo central
EC 1.1	Selecciona la opción Detectar plagio entre un conjunto de sumisiones.	-	-	Muestra el campo para la inserción del identificador de una sumisión de las que desea inspeccionar y el botón Añadir para agregar más sumisiones al conjunto de sumisiones.	Panel de administración/ Detectar plagio entre un conjunto de Sumisiones.
EC 1.2	Inserta los identificadores de las sumisiones que desea inspeccionar y oprime Detectar plagio.	V	V	Visualiza los resultados de la inspección de plagio a través de una tabla, que muestra el porcentaje de plagio entre las sumisiones.	Panel de administración/ Detectar plagio entre un conjunto de Sumisiones/ Inserción de los datos/ Oprime detectar plagio.
EC 1.3	Datos Incorrectos.	I	I	Mensaje de error, datos incorrectos	Panel de administración/ Detectar plagio entre un conjunto de Sumisiones/ Inserción de los datos/ Oprime detectar plagio.
		V	I		
		I	V		
EC 1.4	Campos	NA	NA	Mensaje de error,	Panel de

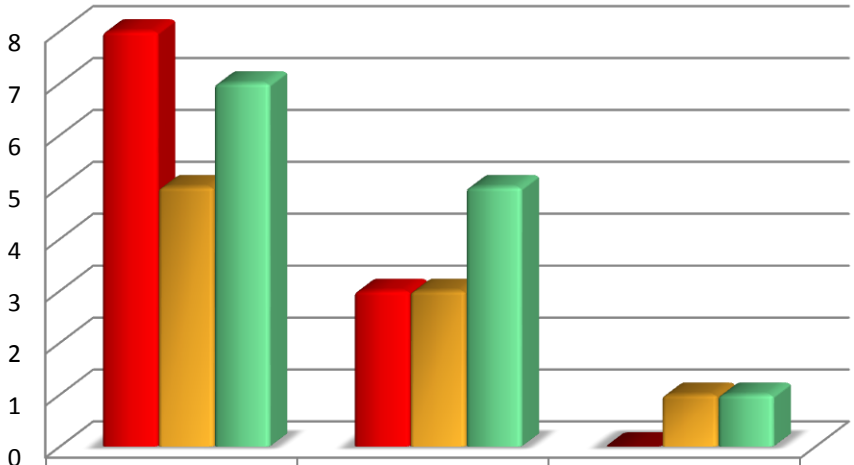
	obligatorios.			campos obligatorios.	administración/ Detectar plagio entre un conjunto de Sumisiones/ Oprime detectar plagio.
EC 1.5	Selecciona los ID de la lista de sumisiones.	V	V	Visualiza los resultados de la inspección de plagio a través de una tabla, que muestra el porcentaje de plagio entre las sumisiones.	Panel de administración/ Detectar plagio entre un conjunto de Sumisiones/ Selecciona los ID de la lista de sumisiones/ Oprime detectar plagio.
EC 4	Selecciona la opción resetear valores.	NA	NA	Limpia los campos para la inserción nuevamente de los identificadores de las sumisiones.	Panel de administración/ Detectar plagio entre un conjunto de Sumisiones/ Inserción de los datos/ Oprime detectar plagio/ Selecciona la opción resetear.

Tabla 9: Descripción de las variables. Caso de prueba del CU1 Detectar plagio entre un conjunto de sumisiones.

No	Nombre del campo	Clasificación	Valor Nulo	Descripción
1	ID Sumisión	campo de texto	No	El campo solicita el id de la sumisión que desea inspeccionar.
2	Agregar ID Sumisión	campo de texto	No	El campo solicita el resto de los id de sumisiones que desea agregar al conjunto que desea inspeccionar.

3.5.2 Resultados de las pruebas

Se realizaron tres iteraciones de prueba en las que no solo se detectaron varias No Conformidades, también surgieron ideas y validaciones necesarias a partir de las recomendaciones de los probadores. En el transcurso de las iteraciones la cantidad de errores tuvo una tendencia a disminuir hasta el punto de en la 3ra iteración notificarse solo una No Conformidad no significativa y una recomendación. A continuación se grafica el resumen de los resultados de las pruebas:



	1ra Iteración	2da Iteración	3ra Iteración
Significativas	8	3	0
No significativas	5	3	1
Recomendaciones	7	5	1

Gráfica 2: Resumen de las No Conformidades arrojadas por las pruebas.

Entre las No Conformidades más significativas resaltan las citadas a continuación:

- Excesivo tiempo de respuesta para rangos amplios de detección.
- Poca claridad en los mensajes mostrados por el módulo, surgió como una recomendación la necesidad de mostrar al usuario el tiempo estimado que tardarán los resultados en calcularse.

- Datos repetidos e inconsistentes con la búsqueda, como parte de esta No Conformidad surgen la idea de agregar varios filtros que posibilitan acotar el rango de búsqueda y añadir una optimización extra al tiempo total de ejecución.

Las No Conformidades fueron corregidas y originaron nuevos ciclos de prueba, la mayoría de las recomendaciones fueron aceptadas. En el caso de las dirigidas a factores subjetivos como el rendimiento, se optimizó el tiempo de repuesta, utilizando técnicas de extracción mínima de información en la base de datos y almacenando varios valores anteriormente calculados en cada petición.

3.6 Validación de la solución desarrollada

La validación de la solución desarrollada asegura que el producto final cumpla con los requerimientos establecidos para el software, acredita además la calidad y eficiencia del módulo. En la actualidad existen disímiles métodos para determinar la validez de un producto, los cuales presentan diferentes enfoques. Una vez realizadas las pruebas a las interfaces de la aplicación, las entradas y salidas de datos, así como el tiempo de las respuestas, se procede a la validación de los algoritmos y al grado de aceptación de los clientes. Para ello se aplica la técnica de Iadov para medir el nivel de satisfacción del personal que utilizará el módulo implementado.

3.6.1 Aplicación de la técnica de Iadov

La técnica de Iadov fue creada originalmente por Kuzmina en 1970, utilizada por López Rodríguez y González Maura en el 2002, para el estudio de la satisfacción por la profesión en carreras pedagógicas. Luego, González Maura y López Rodríguez en ese mismo año realizaron una transformación a esta técnica y la plantean como alternativa para el diagnóstico de la motivación profesional en profesores de Educación Física (55).

Constituye actualmente un camino indirecto para el conocimiento de la satisfacción, se utilizan criterios que establecen relaciones entre tres preguntas que se intercalan dentro del cuestionario y cuya existencia es desconocida por los involucrados. Estas cuestiones se relacionan a través del “Cuadro lógico de V. A. Iadov” (Tabla 10) desarrollado por los citados autores.

Para la recogida de los datos que permitirán conocer la satisfacción, alrededor de la creación y eficiencia del módulo para la detección de plagio, se confeccionó un cuestionario conformado por diez preguntas, de ellas siete cerradas[8] y tres abiertas[9] que posteriormente van a ser analizadas mediante la técnica V. A. Iadov. El cuestionario se aplica a los administradores del COJ, el Comité de Entrenadores del Movimiento de Programación Competitiva Tomás López Jiménez y varios programadores. Las preguntas están basadas principalmente en la necesidad de la creación de un módulo para la detección de plagio y la eficiencia de las detecciones de plagio realizadas por los algoritmos implementados, el cuestionario se puede encontrar en el (Anexo 8).

Tabla 10: Cuadro lógico de V. A. Iadov

5. ¿Qué usted cree del módulo para la detección de plagio en el Juez en Línea Caribeño?	1. ¿Cree usted que el plagio es un comportamiento correcto?								
	No			No sé			Si		
	3. ¿Cree usted que la dificultad en la detección de plagio hace necesaria la creación de un módulo que facilite el proceso?								
	Si	No sé	No	Si	No sé	No	Si	No sé	No
Me gusta mucho	1	2	6	2	2	6	6	6	6
No me gusta tanto	2	2	3	2	3	3	6	3	6
Me da lo mismo	3	3	3	3	3	3	3	3	3
Me disgusta más de lo que me gusta	6	3	6	3	4	4	3	4	4
No me gusta nada	6	6	6	6	4	4	6	4	5
No sé qué decir	2	3	6	3	3	3	6	3	4

[8] **Pregunta cerrada:** El encuestado está forzado a elegir entre las opciones preestablecidas que se le presentan.
[9] **Pregunta abierta:** El encuestado tiene la libertad de dar su opinión sin tener restricciones.

El resultado de la interrelación de las tres preguntas cerradas conlleva a un número que indica la posición de cada uno de los encuestados en la escala de satisfacción siguiente:

1. Clara satisfacción
2. Más satisfecho que insatisfecho
3. No definida
4. Más insatisfecho que satisfecho
5. Clara insatisfacción
6. Contradictoria

La aplicación de la técnica de Iadov y de las preguntas adicionales mostradas en el cuestionario, constituyen un instrumento de gran utilidad para el estudio y evaluación de la satisfacción-insatisfacción de los encuestados.

Para obtener el índice de satisfacción grupal (ISG) se establece una escala numérica entre +1 y -1 con los diferentes niveles de satisfacción como se muestra en la (Tabla 11) (56).

Tabla 11: Escala numérica de satisfacción

Valor	Interpretación
+1	Máximo de satisfacción
0. 5	Más satisfecho que insatisfecho
0	No definido y contradictorio
-0. 5	Más insatisfecho que satisfecho
-1	Máxima insatisfacción

El índice de satisfacción grupal se calcula por la siguiente fórmula:

$$\mathrm{ISG} = \frac{\mathrm{A}(+1) + \mathrm{B}(0.5) + \mathrm{C}(0) + \mathrm{D}(-0.5) + \mathrm{E}(-1)}{\mathrm{N}}$$

En esta fórmula A, B, C, D, E, representan el número de sujetos con índice individual (1, 2, 3, 4, 5 ó 6) y N representa el número total de sujetos del grupo.

El ISG arroja valores entre +1 y -1. Los valores que se encuentran comprendidos entre - 1 y - 0, 5 indican insatisfacción; los comprendidos entre - 0, 49 y + 0, 49 evidencian indefinición o contradicción y los que caen entre 0, 5 y 1 indican que existe satisfacción.

Estos valores se pueden representar en un eje de coordenada como se aprecia en la (Figura 22)

Figura 22: Eje numérico de satisfacción

El procedimiento de cálculo del Índice de Satisfacción Grupal (ISG) de la encuesta realizada a 15 personas queda reflejado a continuación:

$$\mathrm{ISG} = \frac{\mathrm{A}(+1) + \mathrm{B}(0.5) + \mathrm{C}(0) + \mathrm{D}(-0.5) + \mathrm{E}(-1)}{\mathrm{N}}$$

$$\mathrm{ISG} = \frac{10(+1) + 4(0.5) + 0(0) + 1(-0.5) + 0(-1)}{15}$$

$$\mathrm{ISG} = 0.76$$

El resultado arrojado demuestra clara satisfacción con una ISG de 0.76 dentro del rango [0.5, 1].

3.7 Conclusiones

En este capítulo se realizaron los diagramas de despliegue y de componentes. Se realizó una reseña de las cuestiones más importantes relacionadas con la implementación. Por último se documentó el proceso de prueba del módulo, enfatizando en el uso de la técnica Iadov para el cálculo de la satisfacción. Luego de aplicarles una encuesta a 15 personas el Índice de Satisfacción Grupal resultantes determinó que el módulo para la detección de plagio cuenta con un buen nivel de aceptación entre sus usuarios.

Conclusiones

El proyecto para la creación de un módulo para la detección de plagio en el Juez en Línea Caribeño, transitó por varias etapas hasta convertirse, finalmente, en un excelente resultado disponible entre las herramientas administrativas del COJ en http://coj.uci.cu. Para su desarrollo primeramente fueron estudiados el plagio como problema ético, las variantes actuales para su detección y los resultados obtenidos mediante el uso de aplicaciones similares a la que finalmente fue desarrollada durante esta investigación. El estudio permitió concluir que no existe un software, algoritmo, ni otra clase de sistema para la detección de plagio que se adecúe completamente a las características propias de un Juez en Línea, teniendo en cuenta, que en este tipo de entorno es necesario valorar criterios que van más allá de la comparación de códigos fuentes**.**

Se realizó el análisis y diseño del módulo para la detección de plagio de acuerdo a la metodología de desarrollo RUP que fue la seleccionada para llevar a cabo el proceso de desarrollo, lo cual constituyó los cimientos para la correcta implementación de la propuesta.

La base algorítmica para la detección de plagio constituye el núcleo del módulo creado. Se elaboró de forma que tuviera en cuenta el código fuente de las soluciones, las características del problema al cual estas corresponden y el historial o comportamiento previo de los respectivos autores. Luego de su diseño, implementación y de chequeado su funcionamiento práctico se concluye, en consonancia con el objetivo general de la investigación, que la propuesta algorítmica sobre la base de los criterios antes mencionados, proporciona dictámenes coherentes en la comparación de soluciones enviadas al Juez en Línea Caribeño.

En general se obtuvo un módulo que agiliza el proceso de detección de soluciones plagiadas, su uso es netamente administrativo y su funcionamiento no es invasivo. El módulo fue probado y sus dictámenes validados por el Colectivo de Entrenadores del Movimiento de Programación Competitiva Tomás López Jiménez.

Recomendaciones

La estructura híbrida del sistema para la detección permite la inclusión relativamente sencilla de nuevos modos de detección, los cuales correctamente ponderados y unificados con el resto de los resultados pueden perfeccionar aún más los dictámenes. Se recomienda la implementación de detectores que tengan en cuenta métricas de software, comportamiento en tiempo de ejecución, estilo de código de los usuarios, entre otros factores de avanzada cuya complejidad motivó que quedaran fuera de la investigación, al menos en su fase actual.

Se cree oportuno continuar optimizando los tiempos de respuesta del sistema tanto con mejoras a los algoritmos como el almacenamiento en caché de sus resultados.

Desarrollar nuevas formas de representación de los resultados orientadas de alguna manera a que los usuarios sean partícipes del proceso de detección, justificando o refutando los dictámenes que haga el sistema sobre sus soluciones.

La utilidad de una herramienta para la detección de plagio se torna aún mayor aplicada a un Juez en Línea para uso académico. Se recomienda su implementación en el *Academic Online Judge*.

Referencias bibliográficas

1. **Gutierrez, Andrea.** *Fundamentos de la programacion de computadoras.* s.l. : Institución Educativa Simón Bolívar, 2012.

2. *Consejos para tener éxito Programando.* **Perera, Raydelto Hernández.** 2009.

3. **Europa PRESS.** El Juez Online de la UVA alcanza los diez millones de accesos procedentes de todo el mundo. *Europa PRESS.* 15 de 2 de 2012. Sacado de http://www.europapress.es/castilla-y-leon/innova-00439/noticia-innova-juez-online-uva-alcanza-diez-millones-accesos-procedentes-todo-mundo-20120420142158.html.

4. *El Jurado en Línea, una nueva forma de ejercitación y evaluación en las asignaturas de programación.* **Junco Vázquez, Tomás Orlando.** Ciudad de la Habana, Cuba : Informática 2009, 2009. ISBN-978-959-286-010-0.

5. **Iniciativa Xtreme Programming.** Xtreme Judge. [En línea] 2006. [Citado el: 26 de 5 de 2012.] http://xtremejudge.uci.cu.

6. **Jimenez, Reymis Monier.** Juez en Línea Cátedra de Programación Avanzada (CPAV). [En línea] 2007. [Citado el: 26 de 5 de 2012.] http://cpav.uci.cu.

7. **COJ development team(CDEVT).** Caribbean Online Judge. [En línea] UCI, 5 de 6 de 2010. [Citado el: 26 de 5 de 2012.] http://coj.uci.cu.

8. **Real Academia Española.** Diccionario de la Lengua Española. *http://buscon.rae.es/draeI/.* [En línea] 2001. [Citado el: 23 de 5 de 2012.] http://buscon.rae.es/draeI/.

9. **Jimenez, Reymis Monier.** Juez en Línea Cátedra de Programación Avanzada (CPAV). [En línea] 2007. [Citado el: 19 de 03 de 2012.] http://cpav.uci.cu/viewpage.php?page_id=3.

10. **Universidad de Oxford.** *http://dictionary.oed.com/.* [En línea] 11 de 2010. [Citado el: 23 de 5 de 2012.] http://dictionary.oed.com/.

11. **Castro, Sonia Jannett Girón.** *Libro anotaciones sobre plagio.* [ed.] Universidad Sergio Arboleda. Primera edición. 2008.

12. **Cedeño, Luis Alberto Barrón.** *Detección automática de plagio en texto.* Universidad Politécnica de Valencia. Valencia : s.n., 2008. Tesis desarrollada dentro del Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital.

13. *JPlag: Finding plagiarisms among a set of programs.* **Universidad Karlsruhe.** 28 de marzo de 2000, Technical Report 2000-1, University of Karlsruhe.

14. *El ciberplagi acadèmic: esbrinant-ne les causes per tal d'enllestir les solucions.* **COMAS, RUBÉN y SUREDA, JAUME.** 10, s.l. : UOC, 2008, Digihtum.

15. **Universidad de Alcalá.** *Tutorial AlfaBuah.* Biblioteca, Universidad de Alcalá. 2011. Tutorial.

16. **Balaguer, Enrique Valles.** Empresa 2.0: Detección de plagio y análisis de opiniones. [ed.] Universidad Politécnica de Valencia. [Tesis desarrollada dentro del Máster en Inteligencia Artificial, Reconocimiento de Formas e Imagen Digital]. Valencia : s.n., 2010.

17. *El plagio como ilícito penal.* **Balbuena, Pedro Virgilio.** 2004, Ventana Legal.

18. Softonic. [En línea] 2012. [Citado el: 25 de 5 de 2012.] http://onsoftware.softonic.com/detectar-plagio-texto-imagenes-audio.

19. **Barrón Cedeño, Alberto, Vila, Marta y Rosso, Paolo.** *Detección automática de plagio: de la copia exacta a la paráfrasis.* Valencia : Natural Language Engineering Lab, 2011.

20. **Wise, Michael J.** *YAP3: Improved detection of similarities in computer program and other texts.* Sydney, Australia : Department of Computer Science, University of Sydney, 1996.

21. —. *String Similarity via Greedy String Tiling and Running Karp−Rabin Matching.* Sydney, Australia : Department of Computer Science, University of Sydney, 1993.

22. **Kazim, Norulazmi Bin.** *Plagiarims Detection System for Java Programming Assigments by Using Greedy String Tilling Algorithm.* Malaysia : College of Arts and Sciences, 2008. pág. 15, Tesis para obtar por el grado MSc. (ICT).

23. *An algorithmic approach to the detection and prevention of plagiarism.* **Ottenstein, Karl J.** s.l. : ACM SIGSCE, 1976, págs. 30–41.

24. **Wise, Michael J.** Wise's Web Words. [En línea] [Citado el: 24 de 5 de 2012.] http://www.pam1.bcs.uwa.edu.au/~michaelw.

25. **Sphere Research Labs.** SPOJ. [En línea] Sphere Research Labs, 2012. [Citado el: 27 de 5 de 2012.] http://www.spoj.pl/info.

26. *Aplicación de la teoría de grafos y el Algoritmo de Dijkstra para determinar las distancias y las rutas más cortas en una ciudad.* **Restrepo C, Jorge Hernán y Sánches C, John Jairo.** [ed.] Redalyc. 26, s.l. : Scientia Et Technica, 2004, Scientia Et Technica. Disponible en http://redalyc.uaemex.mx/src/inicio/ArtPdfRed.jsp?iCve=84911640021. ISSN 0122-1701.

27. *Las Metodologías de Desarrollo Ágil como una Oportunidad para la Ingeniería del Software Educativo.* **Orjuela Duarte, Ailin y Rojas C., Mauricio.** Colombia : s.n., 2 de 6 de 2008, Avances en Sistemas e Informática, Vol. 5. Disponible en: http://redalyc.uaemex.mx/src/inicio/ArtPdfRed.jsp?iCve=133115027022. ISSN (Versión impresa): 1657-7663.

28. **Beck, Kent.** *Extreme Programming Explained.* First Edition . s.l. : s.l. : Notes, 1999.

29. *Metodologías de desarrollo de software. ¿Cuál es el camino?* **Delgado Expósito, Erly.** Matanzas, Cuba : s.n., 3 de 12 de 2008, Revista de Arquitectura e Ingeniería, Vol. 2. Disponible en: http://redalyc.uaemex.mx/src/inicio/ArtPdfRed.jsp?iCve=193915935003. ISSN (Versión impresa): 1990-8830.

30. **Jacobson, Ivar, Booch, Grady y Rumbaugh, James.** *El proceso unificado de desarrollo de software.* Primera edición. Madrid : Pearson Educación S.A, 2000.

31. **Seco, José Antonio González.** *El lenguaje de programación C#.* 2002.

32. **Stewart, Celeste.** The advantage of PHP. Designer's Playground. [En línea] 3 de enero de 2006. [Citado el: 22 de febrero de 2010.] http://www.designersplayground.com/articles/118/1/The-Advantages-of-PHP/Page1.html.

33. **García de Jalón, Javier , y otros, y otros.** *Aprenda Java como si estuviera en primero.* s.l. : San Sebastián, 2000.

34. **Otero, Abraham.** MySQL vs PostgreSQL ¿cuándo emplear cada una de ellas? *javaHispano.* [En línea] 10 de 9 de 2007. [Citado el: 22 de 2 de 2010.] http://www.anchor.com.au/hosting/dedicated/mysql_vs_postgres.

35. **Toledo, Raciel Yera.** *Concepción y desarrollo de un sistema de recomendación para jurados online de programación.* Universidad de las Ciencias Informáticas. La Habana : s.n., 2010. pág. 45, Tesis de grado.

36. **Johnson, Rod.** *Expert One-To-One: J2EE Design and Development.* s.l. : Wrox, 2002.

37. **Walls, Craig.** *Spring in Action Second Edition.* s.l. : Manning, 2008.

38. **King, Gavin.** *Hibernate in Action.* s.l. : Manning, 2005.

39. **Corporación C&TA.** Ciencia y Técnica Administrativa (C&TA). [En línea] 2008. [Citado el: 16 de 3 de 2012.] http://www.cyta.com.ar/biblioteca/bddoc/bdlibros/proyectoinformatico/libro.

40. Visual Paradigm. [En línea] 2011. [Citado el: 13 de 1 de 2011.] http://www.visual_paradigm.com/product/.

41. Ecured. [En línea] [Citado el: 16 de 3 de 2012.] http://www.ecured.cu/index.php/Visual_Paradigm.

42. Ecured. [En línea] [Citado el: 16 de 3 de 2012.] http://www.ecured.cu/index.php/Rational_Rose.

43. Ecured. [En línea] [Citado el: 16 de 3 de 2012.] http://www.ecured.cu/index.php/Eclipse,_entorno_de_desarrollo_integrado.

44. Ecured. [En línea] [Citado el: 16 de 3 de 2012.] http://www.ecured.cu/index.php/NetBeans.

45. Ecured. [En línea] [Citado el: 16 de 3 de 2012.] http://www.ecured.cu/index.php/Tomcat.

46. **Larman, Craig.** *UML y Patrones: Una introducción al análisis y diseño orientado a objetos y al proceso unificado.* Segunda edición. s.l. : Prentice-Hall, 2002.

47. **Gracia, Joaquin.** IngenieroSoftware. [En línea] mayo de 2005. [Citado el: 5 de 4 de 2012.] http://www.ingenierosoftware.com/analisisydiseno/patrones-diseno.php.

48. *Seminario de Arquitectura de Software.* **Reynoso, Billy.** s.l. : Universidad de Buenos Aires, Architect Academy, 2005.

49. *El patrón de diseño Modelo Vista Controlador (MVC) y su implementación en Java Swing.* **Bascón Pantoja, Ernesto.** 4, s.l. : Acta Nova, 2004, Vol. 2.

50. **O´Regan, Graham.** Onjava.com. [En línea] 14 de 1 de 2004. [Citado el: 19 de 2 de 2012.] http://onjava.com/pub/a/onjava/2004/01/14/aop.html..

51. **González, Carlos Sánchez.** *Seguridad no intrusiva con Acegi Security System for Spring.* Softgal. España : s.n., 2010.

52. **COJ, development team.** Caribbean Online Judge. [En línea] 5 de 6 de 2010. [Citado el: 3 de 5 de 2012.] http://coj.uci.cu/24h/statistics.xhtml.

53. **YourBussinet.** Programación Web. [En línea] 2003. [Citado el: 26 de 3 de 2012.] http://www.programacionweb.net.

54. **Pressman, Roger S.** *Ingeniería de Software, Un enforque práctico.* Séptima. New York : McGraw-Hill, 2010. ISBN 978-0-07-337 597 -7.

55. *Las clases de Educación Física y el deporte extraescolar entre el alumnado almeriense de rimaria. Una aplicación práctica mediante la técnica de Iadov.* **Gómez López, Manuel, y otros, y otros.** España : s.n., 2006, Revista Digital - Buenos Aires.

56. *La técnica de Iadov: Una aplicación para el estudio de la satisfacción de los alumnos por las clases de educación física.* **Rodríguez, Alejandro López y González Maura, Viviana.** Universidad de la Habana, Cuba : s.n., 4 de 2002, Revista Digital - Buenos Aires.

57. *Rational Unified Process.* s.l. : Rational Software Company, 2003.

58. **Ruz, Víctor Valenzuela.** *Manual de análisis y diseño de algorítmos.* Copiapó, Chile : Instituto Nacional de Capacitación (INACAP), 2003.