UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

FACULTAD 2

Trabajo de diploma para optar por el título de Ingeniero en Ciencias Informáticas

Módulo de recomendación de problemas para el Juez en Línea Caribeño Versión 2.0

Autores: Eiquel Osorio Ramírez
Randy Espino Lobaina

Tutores: MsC. Tomás Orlando Junco Vázquez
MsC. Raciel Yera Toledo
Ing. José Carlos González Fernández

La Habana 2015

PENSAMIENTO

"La inteligencia consiste no sólo en el conocimiento, sino también en la destreza de aplicar los conocimientos en la práctica".

Aristóteles

DECLARACIÓN DE AUTORÍA

Declaramos que somos los únicos autores del trabajo “Módulo de recomendación de problemas para el Juez en Línea Caribeño Versión 2.0” y autorizamos a la Universidad de las Ciencias Informáticas para hacer uso del mismo en su beneficio.

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

Autores:

_____ _____

Eiquel Osorio Ramírez Randy Espino Lobaina

Tutores:

_____ _____

MSc. Tomás Orlando Junco Vázquez Ing. José Carlos González Fernández

DEDICATORIA

Randy:

Este otro éxito en mi vida va dedicado en especial a mi padre que aunque no esté presente físicamente es una victoria y una ilusión que siempre tuvo como padre de su hijo más pequeño y que al final no lo defraudó, "ya puedes descansar tranquilo padre!!!"

Eiquel:

Quiero dedicarle este trabajo especialmente a mis familiares más allegados, a mi mamá, mi papá mi hermana Yara, y a mi hermano Karel.

AGRADECIMIENTOS

Randy:

Agradecer en primer lugar a mi mamá, mi hermano y a mi papá (que toda la vida lo tendré presente) por el apoyo, cariño y formación durante toda mi vida, si no fuera por uds no estaría hoy aquí. A mis tutor MsC. Tomas Orlando Junco Vázquez y el Ing. José Carlos Gonzáles Fernández por la enorme labor pedagógica que ejerció sobre mí, por los debates, por las ideas, por el rigor científico y por la formación ingenieril que me ha dado, son sin dudas un ejemplo a seguir como tutores, gracias.

De la UCI a mis amigos del aula, donde todos hemos sabido vencer esta carrera de 5 años satisfactoriamente. A Eiquel, Javier, Jesús Yasser, Lidia, Dariel, Yaikel, son lo máximo en amigos y aunque nuestras vidas se distancien a partir de ahora no vamos a perder el contacto. No quisiera dejar de mencionar a todas las amistades incluyendo profesores, estudiantes y trabajadores no docentes que me han bridado su apoyo incondicional en momentos puntuales, de corazón nunca los olvidaré, les deseo muchos éxitos en sus vidas, en general a todos y ojala no se me quede nadie. Muchas gracias a todos.

Eiquel:

Quiero agradecer primeramente a mi familia por todo el apoyo que me han dado en la vida y en la carrera si no fuera por ellos no estaría aquí hoy, agradezco a mi hermano Karel por todas las cosas que me enseño en estos 5 años en la universidad, fue como un padre para mí. Agradecimientos a mis tutores, Jose Carlos Tomas y Raciel, les estaré eternamente agradecidos por la grandísima ayuda que me han dado.

A todos mis compañeros con los que compartí aula desde 1ro hasta 5to año y a todos mis amigos en general. También no puede faltar agradecer a mis profesores que sin dudas aportaron en mi formación como ingeniero.

RESUMEN

El presente trabajo se centra en la necesidad de desarrollar la versión 2.0 del módulo de recomendación de problemas para el Juez en Línea Caribeño. Se estudiaron las características principales de los sistemas de recomendación y su necesidad en los jueces en línea. La solución al problema planteado consistió en un sistema de recomendación basado en filtrado colaborativo, para el cual se propuso un perfil algorítmico que integra la teoría más reciente sobre sistemas de recomendación en jueces en línea. Para desarrollar el mismo se utilizó el marco de trabajo Spring 3.0 y la metodología XP para guiar el proceso de desarrollo. En la construcción y despliegue se utilizó el servidor de aplicaciones Web Apache Tomcat 7.0; como sistemas gestor de bases de datos se utilizó PosgreSQL 9.1; como entorno integrado de desarrollo se empleó Spring Tool Suite 4.0, Java, HTML5 y JavaScript como lenguajes de programación, y la librería JavaScript Wijmo para la construcción de gráficos. Como resultado de la propuesta se obtuvo una versión del módulo de recomendación enfocado a orientar a los usuarios del Juez en Línea Caribeño con respecto a qué problemas resolver y, además, mediante 5 interfaces de explicación se muestra a los usuarios los motivos por los cuales son sugeridos los problemas. se realizó la validación de los resultados obtenidos, tanto del perfil algorítmico como del software, y se verificó el cumplimiento de los objetivos propuestos.

Palabras clave: problemas, juez en línea, sistema de recomendación.

ÍNDICE DE CONTENIDO

ÍNDICE DE CONTENIDO

ÍNDICE DE CONTENIDO

ÍNDICE DE CONTENIDO

INTRODUCCIÓN

El 5 de junio de 2010 se publicó en Internet la versión 1.0 del Juez en Línea Caribeño (COJ por sus siglas en inglés) desarrollado principalmente por profesores y estudiantes de la UCI, con el objetivo de proveer un espacio donde las personas de todo el Caribe y otras regiones del planeta puedan auto-prepararse e intercambiar experiencias y conocimientos sobre la resolución de problemas computacionales (COJ, 2010). Aun así Este sistema presenta insuficiencias que impiden brindarle a cada usuario una guía para una preparación enfocada a sus habilidades.

El COJ tiene más de 2300 problemas computacionales y más de 20000 usuarios[1] registrados. Estos problemas son de distintas áreas del conocimiento y complejidad, por lo que en muchas ocasiones resulta complejo para el usuario seleccionar el problema apropiado a resolver. Una solución clásica a esta problemática son los sistemas de recomendación (Resnick, y otros, 1997). Varios autores (Resnick, y otros, 1997); (Burke, 2002); (Ricci, y otros, 2010) coinciden en que un sistema de recomendación es una herramienta de software que, basado en preferencias, sugerencias y valoraciones emitidas por los usuarios, permite guiarlos en su proceso de búsqueda de información relevante.

Actualmente existe en el COJ un sistema de recomendación de problemas, que propone a los usuarios una selección de estos ajustada a su perfil[2]. Sin embargo, para realizar las recomendaciones no se tiene en cuenta información específica de los jueces en línea, sobre todo información referente a la calificación de las soluciones de los usuarios (An e-Learning Collaborative Filtering Approach to Suggest Problems to Solve in Programming Online Judges, 2014). La idea anterior parte de que en la versión actual existen elementos relevantes que no se tienen en cuenta tales como: problemas resueltos por el usuario, problemas que el usuario intentó solucionar sin éxito y el esfuerzo necesario para resolver cada problema. Dado que el actual recomendador ignora estos elementos, provoca que no se extraiga el perfil de valoraciones de los usuarios sobre los problemas, siendo esto de vital importancia para generar las recomendaciones (Resnick, y otros, 1997), por tanto se considera un sistema de recomendación deficiente ya que los problemas sugeridos no están sustentados en base a las valoraciones de los usuarios.

Una vez conocidas las deficiencias, se realizó una evaluación al recomendador mediante métricas de precisión, los resultados obtenidos fueron negativos y se comprobó que no cumple con los estándares mínimos de precisión de los sistemas de recomendación. La principal consecuencia de esto es que las

[1] Estos datos son tomados en diciembre de 2014.

[2] En la versión actual el término perfil se refiere a la preferencia de clasificación de los problemas.

recomendaciones que hoy ofrece COJ a sus usuarios sobre posibles problemas a resolver son insuficientes para potenciar sus habilidades.

Mediante entrevistas con miembros del MPC-TLJ se detectó que en la mayoría de los casos los usuarios no confiaban en las recomendaciones. Una de las causas de la desconfianza es el desconocimiento de las razones por las cuales el sistema de recomendación le sugirió un problema, lo que impide que los usuarios tomen una decisión final correctamente sustentada sobre qué problema resolver. Según (Herlocker, et al., 2000) la ausencia de explicación de las recomendaciones es una carencia de cualquier sistema de recomendación.

Teniendo en cuenta la situación problemática anteriormente planteada se presenta el siguiente **problema de investigación:** ¿Cómo elevar la precisión de las recomendaciones realizadas a los usuarios en COJ, para mejorar el aprovechamiento de los recursos disponibles por parte de los usuarios?

Con el propósito de darle solución a este problema, se determina como **objeto de estudio** Los sistemas de recomendación de información; enmarcándose como **campo de acción** Los sistemas de recomendación en los jueces en línea de programación.

El **objetivo general** de la investigación es desarrollar una versión del módulo de recomendación de problemas para el COJ con el fin de proporcionar a sus usuarios buenas sugerencias sobre los problemas a resolver, para mejorar el aprovechamiento de los recursos.

Como **objetivos específicos** para alcanzar la meta propuesta, se desglosan los siguientes:

1- Realizar un estudio del marco teórico de los sistemas de recomendación, haciendo énfasis en su evolución histórica, su aplicación en los sistemas e-learning[3] y a los jueces en línea.
2- Definir la metodología y herramientas a utilizar en su desarrollo.
3- Elaborar un resumen de las técnicas más factibles para el desarrollo de los sistemas de recomendación para los jueces en línea de programación.
4- Elaborar la propuesta de solución para el módulo de recomendación.
5- Desarrollar el módulo de recomendación para el COJ.

[3] Aprendizaje electrónico. El e-learning consiste en la educación y capacitación a través de Internet.

6- Validar el módulo de recomendación desarrollado.

Para dar cumplimiento a los objetivos propuestos, se definen las siguientes **tareas de la investigación:**

- Realizar un estudio de los principales jueces en línea de programación.
- Realizar un estudio de los sistemas de recomendación y su aplicación a los sistemas e-learning y a los jueces en línea.
- Realizar un estudio sobre explicaciones en los sistemas de recomendación.
- Determinar las técnicas más adecuadas para el dominio de los sistemas de recomendación para jueces en línea.
- Definir la metodología de desarrollo a emplear.
- Definir la plataforma, las principales herramientas, tecnologías y lenguajes de programación y modelado sobre las que el módulo de recomendación será construido.
- Realizar un estudio de las técnicas algorítmicas y/o de inteligencia artificial referenciadas como más efectivas para el desarrollo de los sistemas de recomendación.
- Diseñar las interfaces de explicación de las recomendaciones.
- Definir los requisitos funcionales del módulo de recomendación
- Definir los requisitos no funcionales del módulo de recomendación.
- Implementar las funcionalidades definidas.
- Documentar las pruebas de carga y estrés, de aceptación y de integración y las métricas de evaluación del sistema de recomendación.

Para guiar el desarrollo de la investigación se emplearon los **métodos científicos** siguientes:

Métodos Teóricos:

Inductivo-Deductivo: Se hace uso de deducciones para llegar a tener una visión clara de lo que se quiere hacer y adquirir así nuevos conocimientos.

- A partir de la búsqueda de toda la información referente sobre las técnicas de filtrado más básicas se selecciona una de ellas para solucionar la problemática existente.

Histórico-Lógico**:** Permite estudiar de forma analítica la trayectoria histórico real de los fenómenos, su evolución y desarrollo.

- Este método fue usado con el objetivo de estudiar todo lo relacionado con la historia, desarrollo y evolución del concurso ACM-ICPC en todos sus niveles de competencia, permitiendo tener un mayor conocimiento del empleo de los jueces en línea.

Métodos Empíricos:

Entrevista: Es una conversación planificada entre el investigador y el entrevistado para obtener información.

- Se realizó una serie de entrevistas a un número considerable de usuarios del MPC-TLJ registrados en el COJ para valorar cómo era su interacción con el sistema y los beneficios que le proporcionaba el sistema de recomendación existente.

Observación: Es la percepción planificada dirigida a un fin y relativamente prolongada de un hecho o fenómeno. Es el instrumento universal del científico, se realiza de forma consciente y orientada a un objetivo determinado.

- Fue empleado para conocer y analizar los resultados de las recomendaciones sugeridas a los concursantes por el sistema de recomendación existente, para así evaluar las técnicas más afines al sistema de recomendación a implementar.

CAPÍTULO 1

FUNDAMENTACIÓN TEÓRICA

En este capítulo se analizan los sistemas de recomendación más empleados en la actualidad, el empleo de las herramientas recomendadoras más utilizadas actualmente en los jueces en línea y los sistemas e-learning, así como las características más comunes entre ellos. Además se argumenta la importancia que tiene el uso de las explicaciones en las recomendaciones y la necesidad del empleo de las mismas en COJ. También se seleccionan las tecnologías y metodologías empleadas en su desarrollo.

1.1 Sistemas de recomendación

Hoy en día los sistemas de recomendación se han convertido en un recurso necesario para filtrar información relevante diseminada en grandes volúmenes, con el fin de ayudar a los usuarios a tomar decisiones. Según (Evaluating Collaborative Filtering Recommender Systems, 2004) definen los sistemas de recomendación como aquel sistema que tiene como principal tarea seleccionar ciertos objetos de acuerdo con los requerimientos del usuario. (Resnick, y otros, 1997); (Ricci, y otros, 2010) coinciden en que un sistema de recomendación es una herramienta de software que -utilizando preferencias, sugerencias y valoraciones emitidas por los usuarios- permite guiarlos en su proceso de búsqueda de información relevante.

Los sistemas de recomendación tienen como particularidad que su diseño está orientado a entender las necesidades individuales de los usuarios que asisten, siendo claves para su adopción, la percepción de los consumidores/usuarios sobre el grado en que el sistema de recomendación les comprende y les proporcionan información útil. Un sistema de recomendación es un sistema el cual utiliza las opiniones de los usuarios de una comunidad para ayudar a usuarios de esa comunidad a encontrar contenidos de su gusto entre un conjunto sobrecargado de posibles elecciones (Evaluating Collaborative Filtering Recommender Systems, 2004).

1.1.1 Clasificación de los sistemas de recomendación

A partir de los elementos antes mencionados los sistemas de recomendación han recibido diferentes clasificaciones. Según (Peis, y otros, 2012) se clasifican en sistemas de recomendación de filtrado social, también los denominados de filtrado en colaboración o colaborativos, los basados en contenido y los basados en factores económicos.

En la presente investigación enfocaremos el análisis en los sistemas de recomendación basados en contenido y los colaborativos ya que son los que más aplicación han tenido a los sistemas e-learning. Se excluye el

análisis de los sistemas de recomendación de filtrado social y los basados en factores económicos debido a que su principal aplicación es a sistemas e-commerce[4].

1.1.1.1 Sistemas de recomendación basados en contenido

Los recomendadores basados en el contenido son aquellos sistemas que recomiendan los ítems a los usuarios basándose exclusivamente en la descripción del ítem y de un perfil de los intereses del propio usuario (Swearingen, y otros, 2001). Un sistema de recomendación basado en contenido es aquel en el cual las recomendaciones son realizadas basándose en un perfil creado a partir del análisis del contenido de los ítems que el mismo usuario ha comprado, utilizado o visitado en el pasado (Balabanovic, y otros, 1997). Extraen características de los ítems no conocidos aún por un usuario y las comparan con el perfil del mismo para predecir sus preferencias sobre tales ítems. Lo que pretende este filtrado es recomendar ítems muy similares en su contenido a ítems que ya se sabe que son del agrado del usuario en cuestión.

Poseen un grave problema: la **sobrespecialización**. Este es un fenómeno que se presenta al reducir las recomendaciones a unos contenidos muy similares sin tener en cuenta la posible arbitrariedad de los gustos e intereses de los usuarios. Otro problema de los sistemas de recomendación basados en contenido es que de un objeto sólo se puede conocer una información parcial, normalmente textual, mientras que la información contextual, visual o semántica es más difícil de conocer y por lo tanto se pierden conexiones entre objetos similares de manera menos obvia (Albin, 2009). Además algunos autores consideran que estos sistemas son poco fiables cuando no se tiene mucha información acerca del usuario.

1.1.1.2 Sistemas de recomendación colaborativos

Los sistemas de recomendación basados en un filtrado colaborativo son aquellos en los que las recomendaciones se realizan basándose solamente en los términos de similitud entre los usuarios (Swearingen, y otros, 2001). Específicamente, mediante el filtrado colaborativo los usuarios expresan sus preferencias a través de la ponderación de ítems que se les presenta, sirviendo esto para crear un perfil aproximado de este. Una vez creado cada uno de los perfiles, cuando un nuevo usuario solicita recomendaciones el sistema asocia los "ratings" suministrados por este contra los "ratings" que conforman el

[4] Comercio electrónico consiste en la compra y venta de productos o de servicios a través de medios electrónicos, tales como Internet y otras redes informáticas.

perfil de los usuarios anteriores, construyendo una lista de usuarios semejantes, más conocida como vecinos más cercanos. Combinando esta lista, se devuelve un listado de recomendaciones conteniendo los ítems mejor evaluados por estos usuarios que no han sido aún evaluados por el nuevo individuo. La popularidad de estos sistemas de recomendación ha provocado el descubrimiento de algunos problemas como son la escasez, la escalabilidad y el problema del nuevo ítem, aunque en los últimos tiempos se ha llevado a cabo disímiles investigaciones con el objetivo de minimizar estos problemas.

El filtrado colaborativo es considerado una de las tecnologías más potentes de personalización dentro del amplio concepto de la web adaptativa. Entre las aplicaciones pioneras y más representativas de utilización de esta técnica están amazon.com (Linden, y otros, 2003) y GroupLens (An Open Architecture for Collaborative Filtering of Netnew, 1994), la que constituye el principal referente arquitectónico y algorítmico dentro de este tipo de sistemas de recomendación.

Los sistemas de recomendación basados en filtrado colaborativo han sido capaces de demostrar más nivel de exactitud y eficiencia en las recomendaciones que los sistemas basados en contenido, producto que estos sistemas no dependen de información adicional más allá de los usuarios, los ítems y las interacciones usuario-ítems. Además, el filtrado colaborativo solo necesita calificaciones o ponderaciones sobre un ítem para que ese ítem pueda ser recomendado posteriormente, mientras el filtrado basado en contenido necesita realizar un análisis exhaustivo del contenido del ítem para que el sistema pueda considerar recomendarlo cuando sea solicitado. Más aun, si no existe una manera fácil de extraer automáticamente una característica o contenido de un ítem, entonces el filtrado basado en contenido no puede considerar ese ítem (Schafer, y otros). Los investigadores generalmente consideran que el filtrado colaborativo conduce hacia ítems diferentes o inesperados que son igualmente bien valorados mientras los filtrados basados en contenido tienden a sobrespecializarse a un mismo ítem. Algunas personas denominan esta propiedad de recomendación de ítems diferentes como "novedad o hallazgo afortunado" (Evaluating Collaborative Filtering Recommender Systems, 2004).

1.1.2 Explicaciones en los sistemas de recomendación

A pesar del constante crecimiento de los sistemas de recomendación, su aceptación en diversos dominios no ha sido muy amplia, dado que los usuarios al desconocer su funcionamiento, limitan su utilización (Cleger-Tamayo, y otros, 2012). Una de las principales dificultades radica en que los usuarios tienden a no confiar en

las recomendaciones realizadas debido a que se originan a partir de procesos automáticos, o mejor, son recomendaciones realizadas por la computadora, y perciben que es una recomendación aleatoria no fundamentada, no propia para ellos.

Existen varios sistemas secundarios que se pueden incorporar en los sistemas de recomendación para hacer que el usuario confíe más. Entre esos se encuentran los métodos de transparencia (que permiten al usuario entender cómo funciona el motor del sistema de recomendación) o los métodos de efectividad (que le dicen al usuario por qué se debe considerar o no un producto o ítem). Estas técnicas y métodos son tratados bajo el concepto de explicaciones dentro de los sistemas de recomendación.

Los sistemas recomendadores usualmente proveen explicaciones de sus recomendaciones para ayudar a los usuarios a una mejor selección de productos, actividades o incluso amigos (A generalized taxonomy of explanations styles for traditional and social recommender systems, 2012). Cuando un usuario recibe una explicación puede aceptar una recomendación más fácil porque el sistema provee transparencia a su recomendación. Las explicaciones proveen claridad dentro de las recomendaciones exponiendo el razonamiento y la información que sirven como soporte. Debido a esta virtud, las explicaciones han sido estudiadas en muchas áreas de la investigación tales como modelado de usuario, sistemas e-learning, ingeniería de conocimiento, entre otras.

Algunos sistemas de recomendación de filtrado colaborativo, como Amazon, adoptaron el siguiente estilo de justificación: “Cliente quien compró un elemento X también compró elementos Y, Z,…”. Esto también es llamado estilo de justificación “Humano” (Bilgic, y otros, 2005), el cual está basado en comportamientos de acciones (compra, elemento evaluado, etc.) humanas similares. Al contrario, con el estilo llamado “Elemento”, las justificaciones son de otra forma: “Elemento Y es recomendado porque usted calificó/compró elemento X”. Estos autores (Bilgic, y otros, 2005) afirman que el estilo “Elemento” es mejor que el estilo “Humano”, porque ello permite a los usuarios formular exactamente sus verdaderas opiniones de un elemento.

Las capacidades de explicación proveen una solución hacia la construcción de la confianza y pueden mejorar el rendimiento de filtrado de personas usando sistemas recomendadores. Los usuarios estarán más a gusto al confiar en una recomendación cuando ellos conocen las razones detrás de esa recomendación. Las explicaciones ayudarán el proceso de entendimiento de usuarios, y permitirá a los usuarios conocer dónde

están sus fuerzas y debilidades (Herlocker, y otros, 2000). Además, la capacidad para solicitar una explicación provee un mecanismo para manipular el error posible que viene con una recomendación.

Construir una facilidad de explicación dentro de un sistema recomendador puede beneficiar al usuario de muchas maneras. Alguno de los beneficios que provee están:

- Justificación. La comprensión del usuario del razonamiento detrás de una recomendación, para que el pueda decidir cuanta confianza poner en esa recomendación.
- Participación del usuario. La participación del usuario en el proceso de recomendación, permite al usuario agregar su conocimiento y habilidades de inferencia para completar el proceso de decisión.
- Educación. La educación del usuario como el proceso usado para generar una recomendación, para que él pueda entender mejor las fortalezas y limitaciones del sistema.
- Aceptación. La mayor aceptación de un sistema recomendador como un asistente de decisión, desde sus limitaciones y fortalezas están completamente visibles y sus sugerencias están justificadas.

1.2.3 Recomendación en sistemas de e-learning

Un sistema e-learning es un sistema de educación electrónico a distancia en el que se integran el uso de las tecnologías de la información y otros elementos pedagógico-didácticos para la formación, capacitación y enseñanza de los usuarios o estudiantes en línea.

Muchos autores han coincidido en que con ayuda de sistemas de recomendación se puede incrementar la seriedad, eficiencia y calidad del proceso de aprendizaje que las aplicaciones de e-learning clásicas soportan (Velez Lang, y otros, 2006). La utilización de sistemas recomendadores aplicados al e-learning ha sido una de las soluciones para enfrentar la existencia de desbordamiento de información que imposibilita el acceso efectivo a los recursos disponibles en plataformas, viniendo a perfeccionar la eficiencia y eficacia del e-learning y contribuyendo a incrementar el aprovechamiento por parte de los estudiantes en medio de la situación actual antes descrita (Tan, y otros, 2008).

Uno de los ejemplos más notorios de la aplicación de los sistemas recomendadores en la enseñanza se refleja en la investigación de (García Salcines, y otros, 2008), en la cual se presenta un sistema recomendador colaborativo para cursos de e-learning, el cual permite que profesores de perfil similar compartan los resultados de sus investigaciones, tras aplicar minería de datos de manera local sobre sus propios cursos.

Ejemplos de sistemas de recomendación que aplican este tipo de concepción se evidencian en (Velez Lang, y otros, 2006), (Tan, y otros, 2008) y (Yera, 2010).

1.2 Herramientas recomendadoras en jueces en línea de programación

El sobredimensionamiento de la información en la era moderna de la World Wide Web también ha afectado directamente a los jueces en línea de programación. Como mecanismo de solución se han ido implementando herramientas recomendadoras para brindarle al usuario alternativas de selección sobre los problemas a resolver en dichos jueces en línea. Dentro de las pocas aplicaciones con características recomendadoras conocidas se encuentran **ACM Problem Grading**, el **Next2Solve** y el **RECJUDGE**. A continuación se hace un breve análisis de cada una de ellas.

1.2.1 ACM Problem Grading

Esta herramienta, junto con el **Next2Solve**, fue implementada en apoyo al juez en línea de la Universidad de Valladolid. Básicamente se limita a proporcionar un listado de problemas a resolver, el cual puede ser ordenado por diferentes criterios entre los que se pueden mencionar la cantidad de soluciones aceptadas para cada ejercicio, el porciento de soluciones correctas con respecto al total de soluciones y un puntaje de simplicidad que se calcula tomando como base los dos anteriores criterios.

El principal inconveniente que presenta la herramienta es que sus recomendaciones no se centran en un usuario en particular, solo conjetura que los problemas más fáciles y asequibles para resolver son aquellos que tengan más soluciones enviadas correctas (Toledo, 2010).

1.2.2 Next2Solve

Next2Solve, creado por Igor Navernioukde la Universidad de Toronto, se autodefine como un servicio basado en el **ACM Problem Grading** de Urbaniak. La diferencia mayor entre uno y otro viene dada en que mientras el segundo muestra de manera indistinta el mismo listado para todos los usuarios, Next2Solve solicita la entrada de un identificador del usuario y basado en este muestra una sublista de la lista generada por la aplicación de Urbaniak en la que sólo aparecen los problemas a resolver por el usuario entrado excluyendo aquellos que él ya ha resuelto (Toledo, 2010).

1.2.3 RECJUDGE

El **RECJUDGE** es una herramienta que pretende guiar al estudiante a través del contenido disponible en un juez en línea de programación, tratando de establecer una ruta ideal para el aprovechamiento y asimilación óptima, siendo este además el objetivo general de este tipo de sistemas aplicados a los sistemas e-learning (Velez Lang, y otros, 2006). Esta aplicación se concibe a partir de una integración a un juez en línea de programación y basada en la técnica de filtrado colaborativo, intentar resolver la problemática del exceso de flujo de información a los que se enfrentan profesores y estudiantes diariamente en su preparación; el cual es solucionado generando automáticamente paquetes de actividades a realizar por los estudiantes basados en su rendimiento y progreso actual, siendo resuelto este problema desde el enfoque del problema general de la recomendación (Yera, 2010).

1.2.4 Necesidad de un sistema de recomendación con explicaciones para COJ

El COJ actualmente contiene una cantidad considerable de problemas y usuarios registrados lo que supone una gran sobre carga de información, influyendo a menudo negativamente en la decisión de los usuarios sobre qué problemas resolver. Un sistema recomendador en parte solucionaría el fenómeno de sobrecarga de información en el COJ mediante los procesos de búsqueda y selección de problemas que realiza, pero no garantiza la fiabilidad y credibilidad del usuario a la hora de seleccionar o no el problema sugerido por el recomendador. El hecho de que el recomendador sugiera problemas no asegura que el usuario esté convencido de la recomendación. Las interfaces de explicación garantizarían la fiabilidad y credibilidad que el usuario necesitaría en su proceso de selección de problemas propuestos por el recomendador, al proporcionarle razones detrás de la recomendación, lo cual implicaría una decisión más segura y convincente por parte del usuario sobre los problemas recomendados.

1.3 Tecnologías de desarrollo de software

Las tecnologías que se mencionan a continuación constituyen la base tecnológica con la que se desarrolla COJ, por eso la propuesta a desarrollar debe basarse en las mismas tecnologías utilizadas. Solo se mencionan las tecnologías y una breve descripción de las mismas, destacando principalmente porqué fueron escogidas.

1.3.1 Lenguaje de programación Java

Java surgió en 1991 cuando un grupo de ingenieros de Sun Microsystems trató de diseñar un nuevo lenguaje de programación destinado a correr sobre cualquier plataforma: desde computadoras hasta efectos electrodomésticos. Con este fin fueron de los primeros en introducir el concepto de máquina virtual con el objetivo de que sus aplicaciones fueran totalmente independientes de la plataforma que las procesa.

Sun describe a Java como simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico (García de Jalón, 200). Se ha desarrollado unido a Internet y es por eso que es ampliamente utilizado en la construcción tanto de sistemas distribuidos, como de complejas aplicaciones de gestión. El hecho de la dependencia de una máquina virtual, como mismo constituye una ventaja, es también una desventaja. El depender de un intermediario para ejecutar los programas hace a estos más lentos y los vuelve dependientes de la corrección de la mencionada máquina virtual.

1.3.1.1 Fundamentación de la elección

Se decide el uso de Java principalmente por las siguientes razones:

1. Es un lenguaje cuya línea de expansión y desarrollo se corresponde con la política trazada por la UCI y por el país.
2. La mayoría de las aplicaciones a las que se integrará la solución que se desarrolla, están implementadas en Java, por lo que, al asumirlo como lenguaje, se facilitaría la comunicación con estas.
3. Los servidores en los que se desplegará la solución final actualmente son servidores de aplicaciones Java, siendo costosa la migración a otra tecnología.
4. Actualmente se dispone de un ambiente de producción Java, exigiendo sacrificio en tiempo y esfuerzo la preparación de un ambiente para desarrollar en otra tecnología.

1.3.2 Gestor de base de datos

Se selecciona PostgreSQL como sistema gestor de base de datos ya que es una herramienta que ofrece una buena garantía de integridad y tiene un gran rendimiento y escalabilidad bajo grades cargas de trabajo (Otero,

2007). A raíz de esto y en correspondencia nuevamente con las líneas tecnológicas dictadas por la UCI, se opta por PostgreSQL para el desarrollo de la aplicación.

1.3.3 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 (Degiovannini, 2007).

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 (Smith, 2005) 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 garantizarla utilización exitosa de un framework se aplica tecnología nativa de Java.

1.3.3.1 Capa de presentación

En la capa de presentación se opta por la tecnología nativa de Java, que es Java Server Pages (JSP), surgida casi desde los mismos inicios del lenguaje, altamente validada, totalmente compatible con los servidores de aplicación más utilizados y de fácil integración con tecnologías más actuales como AJAX.

El hecho de desechar frameworks de presentación tales como Struts, JSF, entre otros, viene dado por su orientación a grandes soluciones, donde priman las interfaces complejas y la reutilización masiva de componentes (Williams, 2008), lo que no se ajusta a la aplicación actual a desarrollar.

1.3.3.2 Lógica del negocio

Como parte de la capa 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 (Jhonson, 2002). Entre estas se pueden citar el soporte al patrón Inyección de dependencias (Fowler, 2004) y a la Programación Orientada a Aspectos (O´Regan, 2004). Entre sus principales características están:

- Spring MVC es muy flexible, ya que implementa toda su estructura mediante interfaces. Además, todas las partes del framework son configurables a través de plugins.
- Los controladores de Spring MVC se configuran mediante Inversión de Control como los demás objetos.
- Resulta más fácil probar las partes de Spring MVC debido a que evita forzosamente la herencia de una clase y la dependencia directa en el controlador del servlet que despacha las peticiones (Lobaina, y otros, 2012).

Las principales razones de la elección son la ligereza y transparencia de Spring y que COJ está desarrollado con el mismo, su utilización posibilitaría una fácil integración con COJ.

1.3.3.3 Acceso a datos

Para la implementación del acceso a datos en la aplicación, se selecciona el framework Hibernate, entorno de trabajo que tiene como objetivo facilitar la persistencia de objetos Java en bases de datos relacionales (King, 2005). Es considerado por gran margen el framework más popular de persistencia de Java y uno de los más populares en sentido general.

Hibernate está basado completamente en el patrón "Mapeo Objeto Relacional" (Fussell, 2007) (Larman, 1999) y cuenta con una comprobada efectividad a hora de implementar procesos clave dentro de las aplicaciones, como el manejo de las transacciones, la concurrencia y el caché. Su principal deficiencia viene dada por el rendimiento, el cual es inferior al que se puede alcanzar a través de la ejecución de consultas SQL estándares, así como por algunos problemas de inestabilidad en sus últimas versiones. No obstante, teniendo en cuenta la mediana complejidad del modelo de datos del negocio a desarrollar y la facilidad de migración entre Hibernate y las tecnologías nativas de acceso a datos de Java ante cualquier desastre, se opta por utilizar este framework.

1.3.4 Herramienta de modelado

Las herramientas CASE son herramientas de Ingeniería de Software Asistida por Ordenador (CASE por sus siglas en inglés). Son aplicaciones informáticas destinadas a aumentar la productividad en el desarrollo de software, reduciendo el coste de las mismas en términos de tiempo y dinero.

Las herramientas CASE son útiles en todos los aspectos del ciclo de vida de desarrollo del software, en tareas como el proceso de realizar un diseño del proyecto, cálculo de costes, implementación de parte del código automáticamente con el diseño dado, compilación automática, documentación o detección de errores (Pullés, 2003).

Existen variadas herramientas CASE dentro de las que se encuentran Erwin, Rational Rose, Visual Paradigm, entre otras. Se determinó que se utilizaría Visual Paradigm por ser una herramienta de lenguaje de modelado unificado (UML) profesional que soporta el ciclo de vida completo del desarrollo de software, es compatible con una amplia gestión de casos de uso y diseño de base de datos y proporciona medidas más eficaces en el análisis y diseño de sistemas (UMLy BPMN, 2014).

1.3.5 Completamiento del perfil tecnológico

Para completar el perfil tecnológico de desarrollo se siguió la línea definida por la UCI que agrupa las herramientas de mayor éxito en cada una de las áreas.

1.3.5.1 Servidor web

Servidor web es la tecnología que tiene implícito programas informáticos que procesan aplicaciones web realizando conexiones bidireccionales y/o unidireccionales y síncronas o asíncronas con el cliente, generando o cediendo una respuesta en cualquier lenguaje o aplicación del lado del cliente (Ecured, 2014).

Apache-Tomcat es un servidor web que funciona como un contenedor de servidores desarrollado bajo el proyecto Jakarta en la Apache Software Foundation. Tomcat implementa las especificaciones de los servidores y de Java Server Pages de Sun Microsystems.Su selección fue basada principalmente en que posee licencia libre lo cual facilita posibles actualizaciones.

1.3.5.2 Entorno integrado de desarrollo

Spring Tool Suite es un entorno de desarrollo integrado libre diseñado principalmente para el lenguaje de programación Java. Existe además un número importante de módulos para extenderlo. Spring Tool Suite IDE es un producto libre y gratuito sin restricciones de uso.

Los desarrolladores del COJ prefieren Spring Tool Suite debido a que ya tenían experiencias en su utilización, además porque brinda mejor soporte que otras herramientas analizadas y porque se integra fácilmente con las otras herramientas que fueron seleccionadas. Presenta librería para el trabajo con Spring, se integra de forma natural con el servidor web Apache-Tomcat y tiene una interfaz amigable e intuitiva.

1.3.5.3 Otras tecnologías utilizadas

HTML 5, el lenguaje de marcado de hipertexto, está constituido por un conjunto de etiquetas utilizadas para definir páginas web mediante su estructuración en forma de texto, imágenes, vínculos y otros elementos que la componen (Gómez, 2011).

CSS 3, las hojas de estilo en cascada, es un lenguaje usado para definir la presentación de un documento estructurado escrito en HTML o XML (y por extensión en XHTML), constituye la mejor forma de separar los contenidos y su presentación, además es imprescindible para crear páginas web complejas (Hechavarria, 2011).

Wijmo, es un marco de trabajo que contiene un conjunto de herramientas para el desarrollo y estilo de aplicaciones web modernas. Integra más de 40 elementos de interfaz gráfica de usuario para la representación de imágenes, paneles, gráficos, menús, entre otros (Wijmo, 2014). Para el desarrollo del módulo, específicamente para la visualización de gráficos, fue muy útil su utilización debido a que brinda una amplia gama de interfaces fácilmente comprensibles.

1.4 Metodología de desarrollo del software

Extreme Programming (XP) es una metodología ligera de desarrollo de software que se basa en la simplicidad, la comunicación y la realimentación o reutilización del código desarrollado (Fernández, 2002). XP es una de las llamadas metodologías ágiles de desarrollo de software más exitosas de los tiempos

recientes (Joskowicz, 2008). XP alienta a los desarrolladores a responder a los requerimientos cambiantes de los clientes, aun en fases tardías del ciclo de vida del desarrollo. Tanto líderes de proyecto como clientes y desarrolladores son partes del mismo equipo dedicado a entregar software de calidad. La metodología XP define cuatro variables para cualquier proyecto de software: costo, tiempo, calidad y alcance. Además, se especifica que de estas cuatro variables sólo tres de ellas pueden ser fijadas arbitrariamente por actores externos al grupo de desarrolladores (clientes y jefes de proyecto). El valor de la variable restante podrá ser establecido por el equipo de desarrollo en función de los valores de las otras tres.

Este mecanismo indica que, por ejemplo, si el cliente establece el alcance y la calidad, y el jefe de proyecto el precio, el grupo de desarrollo tendrá libertad para determinar el tiempo que durará el proyecto. Esta metodología de desarrollo propone un ciclo de vida dinámico, donde se admite expresamente que, en muchos casos, los clientes no son capaces de especificar sus requerimientos al comienzo de un proyecto. Debido a esta razón, se trata de realizar ciclos de desarrollo cortos (llamados iteraciones), con entregables funcionales al finalizar cada ciclo. En cada iteración se realiza un ciclo completo de análisis, diseño, desarrollo y pruebas, pero utilizando un conjunto de reglas y prácticas que caracterizan a XP. Una de las desventajas de esta metodología está referida a la baja documentación, pensando en el posterior mantenimiento del sistema. Si bien durante el proyecto el equipo tiene en mente todas sus particularidades, hay que prever qué pasará luego de entregado, cuando el equipo se disuelva, y sea necesario realizar algún cambio o mejora. XP propone la recodificación permanente para asegurar la claridad y simplicidad del código. Sin embargo, es posible que aún un código simple y claro no baste cuando otro equipo de trabajo tenga que tomar el sistema y cambiarlo. Es posible que sea necesario mantener cierta documentación, aunque es compartible que ésta debe ser la mínima necesaria.

Tomando en consideración estas cuestiones se determina emplear la metodología XP en la presente investigación debido a la flexibilidad[5] que provee al equipo de desarrollo en la implementación de la propuesta de solución. Otro motivo que determina esta decisión de selección es la necesaria interacción o comunicación que se requiere con el cliente final (usuarios de COJ) para poder desarrollar una eficiente versión del módulo de recomendación para el Juez en línea Caribeño que provocará una mejor aceptación y productividad en el desempeño de los usuarios. Los desarrolladores actuales cuentan con poca documentación

[5] El término "flexibilidad" en el contexto se refiere a las constantes entrevistas que se requieren con los clientes para determinar el grado de aceptación por parte de los mismos, lo que producirá reajustes en el cronograma y en el tiempo de entrega planificado.

referente al sistema de recomendación que está en funcionamiento en el COJ en estos momentos, debido a la legibilidad y claridad del código que implementó el anterior equipo de desarrollo, esto posibilita una rápida comprensión por parte del equipo de trabajo actual.

1.5 Conclusiones parciales

En este capítulo se realizó el estudio del arte de algunos de los sistemas de recomendación más utilizados en los momentos actuales, basándose principalmente en las técnicas que se utilizan para realizar el filtrado, exponiendo sus características distintivas. Se llegó a la conclusión de que, con la amplia utilización de la Web 2.0, actualmente los sistemas de recomendación son muy útiles y necesarios en los jueces en línea de programación y en los sistemas e-learning. Fueron precisadas las herramientas de recomendación utilizadas en los jueces en línea de programación que más impacto y resultados alcanzan a nivel mundial hoy en día. Además se detalló la relevancia e importancia que tiene las explicaciones dentro de las recomendaciones provistas por un juez en línea aportando confiabilidad y seguridad en las decisiones que toman los usuarios. Finalmente, se determinó seleccionar la XP como metodología de desarrollo de software y algunas de las tecnologías de desarrollo de software a utilizar serán:

- Java como lenguaje de programación.
- PostgreSQL como gestor de base de datos.
- Framework Hibernate como entorno de trabajo para el acceso a datos en la aplicación.
- Spring como framework de apoyo.
- Visual Paradigm como herramientas CASE.
- Apache-Tomcat como servidor web.
- Spring Tool Suite como entorno integrado de desarrollo.

CAPÍTULO 2

CONCEPCIÓN DEL SISTEMA DE RECOMENDACIÓN

En el presente capítulo se describe la propuesta de solución a un sistema de recomendación con el objetivo de lograr un mejor aprovechamiento de los recursos disponibles en el COJ. El sistema se enfoca en brindar a sus usuarios sugerencias de problemas a resolver, basándose en el comportamiento de los usuarios al solucionar problemas. Dentro de la descripción se analizan algunas soluciones desde el punto de vista algorítmico que han sido utilizadas en los sistemas de recomendación, así como el perfil algorítmico del sistema en desarrollo.

2.1 Algoritmos en sistemas de recomendación

Con el objetivo de lograr un sistema de recomendación eficaz y eficiente computacionalmente, se han utilizado diferentes algoritmos que pueden ser agrupados en dos clasificaciones: algoritmos basados en memoria y los basados en modelos.

2.1.1 Algoritmos basados en memoria

Los algoritmos basados en memoria son ampliamente utilizados en sistemas de recomendación basados en filtrado colaborativo. Emplean métricas de similitud para determinar el parecido entre una pareja de usuarios, basándose en la evaluación previa de todos los ítems, dados por los usuarios.

Primero emplean técnicas estadísticas para encontrar a usuarios con un historial de valoraciones similares al usuario actual, denominados K-vecinos. Una vez obtenido el listado de usuarios similares, se combinan sus preferencias para generar un listado con los N elementos más recomendables para el usuario actual. Dentro de las limitaciones que poseen se encuentra la necesidad de disponer de un mínimo de usuarios con un número mínimo de valoraciones cada uno, incluyendo el usuario al cual se le va a recomendar (Galán , y otros, 2007).

2.1.1.1 K - Nearest Neighbor

Los algoritmos K-Nearest Neighbor o vecinos más cercanos fueron los primeros algoritmos de filtrado colaborativo en implementarse y constituyen uno de los métodos preferidos a utilizar en los recomendadores colaborativos (Amatriain, et al., 2011). Estos contienen tres etapas fundamentales: la representación de los datos, la formación de vecinos y la generación de las recomendaciones (Analysis of Recommender Systems' Algorithms, 2003), siendo la segunda etapa la más importante ya que si no se concibe de la forma correcta afecta el rendimiento del sistema y por ende la eficacia de las recomendaciones.

En estos algoritmos para el proceso de formación de vecinos, se han definido disímiles medidas de similitud entre usuarios. Una de las medidas frecuentemente utilizadas para determinar semejanzas dentro de los sistemas de recomendación son las basadas en correlación. Estas son medidas estadísticas que permiten determinar la semejanza entre dos elementos, dados los valores cuantitativos que los describen. El **coeficiente de correlación de Pearson** de demostrada efectividad (An Open Architecture for Collaborative Filtering of Netnew, 1994), está dado, para los usuarios a y b por la siguiente expresión:

$$sim(a,b) = \frac{\sum_z (V_{a,z} - V_a)(V_{b,z} - V_b)}{\sqrt{\sum_e (V_{a,z} - V_a)^2 \sum_z (V_{b,z} - V_b)^2}}$$

Ecuación 1: Función de similitud de Pearson

donde $V_{u,z}$ es la valoración emitida por el usuario u sobre el item z; V_u es la media de todas las valoraciones emitidas por el usuario u, y Z es el conjunto de ítems valorados tanto por el usuario a como por el b. Este método indica la correlación lineal de los elementos de un vector con los del otro (Seagaran, 2005).

Otra de las alternativas utilizadas con frecuencia como medida de similitud es el **coeficiente de Jaccard-Tanimoto**. Este calcula la similitud entre dos conjuntos basándose en la cantidad de elementos comunes que poseen; se define para los usuarios a y b de la siguiente manera:

$$sim(a,b) = \frac{V_c}{V_a + V_b - V_c}$$

Ecuación 2: Función de similitud de Jaccard-Tanimoto

siendo V_a la cantidad de elementos del conjunto a; V_b la cantidad de elementos del conjunto b y V_c la cantidad de elementos comunes de a y b.

Luego de obtener los vecinos más cercanos mediante una determinada medida de similitud, para generar las recomendaciones a un usuario se analizan los ítems que no han sido valorados por este y que han sido valorados por los vecinos, excluyendo los ítems que han sido valorados por el usuario en cuestión. Con este fin la técnica más utilizada es la denominada Most-Frequent Item Recommendation (Toledo, 2010). Esta consiste en llevar un conteo de la frecuencia en que cada uno de los ítems es valorado por parte de los usuario que pertenecen al vecindario del usuario en cuestión. Una vez obtenido el listado de frecuencias de

apariciones de los ítems, estos se ordenan descendentemente considerando la frecuencia de aparición. Finalmente, como recomendaciones se retornan los N primeros elementos del listado.

2.1.2 Algoritmos basados en modelos

Los algoritmos basados en modelo utilizan toda la colección de valoraciones emitidas por un usuario sobre los ítems, con el objetivo de crear un modelo del usuario usado para predecir las valoraciones y generar las recomendaciones. Este grupo de algoritmos está estrechamente relacionado con los sistemas de recomendación basados en contenido, pero también tienen aplicación al filtrado colaborativo.

Se ha demostrado que la utilización de estos algoritmos en sistemas de recomendación, produce buenos resultados, aunque varios autores (Heckerman, et al., 1998) aclaran que pueden sufrir problemas de desbordamiento de memoria ante el intento de generar recomendaciones de alta calidad. En general, ante las consultas responden más rápido que los basados en memoria, pero necesitan de un proceso de aprendizaje intensivo.

2.1.2.2 Agrupamiento

Los algoritmos de agrupamiento o clustering consisten en clasificar usuarios y/o ítems en grupos de acuerdo con sus valoraciones. Estos algoritmos asumen que hay grupos de usuarios o ítems con similares características y, por tanto, una vez que el usuario ha sido asignado a un grupo, la predicción o recomendación se obtiene a partir de las valoraciones otorgadas por otros miembros del mismo grupo. Por ejemplo, como predicción se podría utilizar la media de las valoraciones dentro del grupo.

Estos algoritmos producen recomendaciones que en muchos casos tienen menor calidad que las obtenidas por otras herramientas similares (Item-based collaborative filtering recommendation algorithms, 2001). Sin embargo, su principal objetivo no es el de generar recomendaciones, sino el de reducir el espacio contenedor de todos los datos de los usuarios e ítems, agrupándolos en varios espacios más pequeños con menor cantidad de usuarios y menor número de valoraciones, pudiendo entonces obtener una mayor eficiencia al aplicar algoritmos de recomendación. Esto permite una considerable mejora de las recomendaciones, tanto en exactitud como en rendimiento.

Entre los algoritmos más utilizados para la implementación de técnicas de agrupamiento, se encuentra el k-means clustering (Bock, 2007) y el clúster jerárquico (Seagaran, 2005), los que han tenido aplicación en los sistemas de recomendación (Tang, et al., 2003) y (An adaptive recommendation system without explicit acquisition of user relevance feedback, 2003).

2.2 Limitaciones de los algoritmos en sistemas de recomendación

Es importante tener en cuenta las limitaciones que tienen los sistemas de recomendación; a la hora de desarrollarlos, estas se dan bajo distintos escenarios y diferentes circunstancias. A continuación se hace un breve análisis sobre las más significativas.

2.2.1 Arranque en frío

Constituye uno de los problemas graves a resolver al inicio de la puesta en marcha de todo sistema de recomendación. Describe situaciones en las cuales el recomendador es inhabilitado para hacer recomendaciones significativas debido a falta de valoraciones iniciales. Este problema puede degradar significativamente el rendimiento del sistema de recomendación. Esto puede ocurrir bajo 3 escenarios:

1. Nuevo usuario*:* Cuando un nuevo usuario es registrado en el sistema no se dispone de valoraciones; por tanto, el sistema no puede establecer con precisión su perfil.
2. Nuevo ítem*:* Cuando un nuevo ítem es registrado en el sistema, no existen valoraciones sobre este; por tanto, es un ítem difícil de recomendar.
3. Nuevo sistema*:* Cuando se pone a funcionar un sistema de recomendación en un sistema con poco tiempo de uso, el número de usuarios, ítems y valoraciones es bajo, lo que afecta significativamente la calidad de las recomendaciones.

Se han utilizado varias estrategias para mitigar estas problemáticas; entre ellos está la utilización de criterios emergentes, tales como la popularidad de los ítems (Rashid, et al., 2002), así como la selección aleatoria de ítems que tienen pocas valoraciones (Ben Schafer, et al., 2007).

2.2.2 Escalabilidad

Los algoritmos basados en memoria son los que especialmente sufren problemas de eficiencia y escalabilidad, en gran parte se debe al coste de buscar similitudes entre los patrones de valoraciones de los usuarios. En general, las técnicas que tienen en cuenta un número muy elevado de factores o características acabarán teniendo problemas de rendimiento a medida que aumenta el número de ítems o usuarios (Lopez, 2013). Como se había mencionado en epígrafes anteriores, las técnicas de agrupamiento permitirían reducir el espacio contenedor de los datos y así mitigar esta problemática.

2.3 Propuesta del perfil algorítmico para el sistema de recomendación

Tras el estudio de algunas de las tendencias de desarrollo de sistemas de recomendación, para definir el perfil algorítmico de un sistema recomendado se opta por el desarrollo de un enfoque colaborativo basado en memoria, dada su demostrada efectividad en escenarios de aprendizaje (Zhu, IP, Fok, & Cao, 2007). Específicamente, se tomará como base el trabajo (An e-Learning Collaborative Filtering Approach to Suggest Problems to Solve in Programming Online Judges, Toledo & Mota, 2014), el cual es la única propuesta presente en el estado del arte, centrada en aplicar este enfoque de recomendación en el escenario de un juez en línea.

2.3.1 Recomendación basada en filtrado colaborativo

El filtrado colaborativo basado en memoria puede ser fácilmente adaptado para hacer recomendaciones en este escenario, donde el objetivo es proveer una lista de problemas a resolver. La aplicación de un enfoque usuario-usuario puede ser interpretado como "recomendar al usuario activo esos problemas resueltos por el resto de los usuarios que tienen un comportamiento similar con el usuario activo, considerando problemas resueltos y no resueltos". Con este propósito, es necesario inicialmente obtener un listado de vecinos para el usuario activo, usando una medida de similitud específica. Luego puntear cada problema sin resolver para el usuario activo, considerando los vecinos que los tienen resueltos; y finalmente sugerir aquellos problemas con las puntuaciones más altas.

En epígrafes anteriores se mencionó que los algoritmos de recomendación basados en memoria contenían tres etapas: la representación de los datos, la formación de vecinos y la generación de las recomendaciones. En los siguientes epígrafes se describe cómo se desarrollan esas etapas.

2.3.1.1 Representación de los datos

Básicamente la información necesaria para generar recomendaciones está contenida en la base de datos del juez en línea. Específicamente se dispone de un conjunto de presentaciones, cada una conformada por tres tuplas (usuario, problema, calificación) que representa un intento por resolver un problema específico por un usuario específico, dando la correspondiente calificación, que puede ser: aceptado, respuesta incorrecta, tiempo límite excedido, error en tiempo de ejecución, límite de memoria excedido, error de presentación, o error de compilación.

En este caso, es necesario transformar la información en un formato que puede soportar recomendaciones de filtrado colaborativo (Ben Schafer, et al., 2007). Dicha transformación está compuesta por una matriz de valoraciones (matriz [u, p]), donde cada elemento corresponde con un usuario y un problema dado. Cada posición recibe un valor dependiendo de la calificación y la cantidad de intentos de solución sobre ese problema.

Con el objetivo de incorporar información específica de los jueces en línea para mejorar la precisión de las recomendaciones los autores (An e-Learning Collaborative Filtering Approach to Suggest Problems to Solve in Programming Online Judges, 2014) proponen un método que considera 2 categorías en los problemas aceptados y 2 categorías en los problemas fallidos. Esto asume que usuarios usualmente tienen dificultades para resolver algunos ejercicios, y necesitan realizar muchos intentos fallidos antes de la aceptación final. A diferencia, otros ejercicios pueden ser resueltos usualmente con pocos intentos.

En este caso, la intersección usuario-problema podría ser etiquetada con 5 valores diferentes, usando 2 umbrales “th1” y “th2”, como se presenta a continuación:

- matriz [u,p]=0, si el usuario “u” nunca ha intentado resolver el problema “p”.
- matriz [u,p]=1, si el usuario “u” ha resuelto el problema “p”, necesitando menos de “th1” intentos.
- matriz [u,p]=2, si el usuario “u” ha resuelto el problema “p”, necesitando “th1” o más de un “th1” intentos.
- matriz [u,p]=3, si el usuario “u” no ha resuelto el problema “p”, y su número de fallos está por debajo de “th2”.
- matriz [u,p]=4, si el usuario “u” no ha resuelto el problema “p”, y su número de fallos es igual o está por encima de “th2”.

En esta modificación, las categorías 1 y 2 cubren los problemas resueltos y las categorías 3 y 4 los problemas no resueltos.

2.3.1.2 Formación de vecinos

El proceso de formación de vecinos constituye la etapa más importante dentro de filtrado colaborativo. Consiste en establecer una función de similitud entre dos usuarios, para luego mediante esta función obtener un listado con los usuarios más similares al usuario actual.

En el filtrado colaborativo, usualmente es utilizado como medida de similitud el coeficiente de correlación de Pearson cuando las evaluaciones proveídas por los usuarios son información cuantitativa. En este escenario la matriz usuario-problema solo contiene categorías, y por esta razón la aplicación de Pearson no es adecuada (An e-Learning Collaborative Filtering Approach to Suggest Problems to Solve in Programming Online Judges, 2014). Existen medidas alternativas como es el coeficiente de Jaccard-Tanimoto. Este mide la coincidencia entre dos conjuntos. A continuación en la Figura 1 se define la función de similitud entre dos usuarios. Esta función inicialmente recibe el perfil completo para los usuarios (filas de la matriz usuario-problema), y para realizar comparaciones entre ellos, se excluye el análisis de aquellos problemas etiquetados con 0 para ambos usuarios. Además, para los problemas restantes, calcula la cantidad que contiene la misma categoría en ambos usuarios, y finalmente retorna como el valor de similitud, el radio entre esta cantidad y la cantidad de problemas restantes.

```
procedimiento Similitud ( FilaUsuarior1, FilaUsuarior2)
1.    numerador = 0, denominador = 0
2.    n = longitud(FilaUsuarior1)
3.    para i=1 hasta n
4.          si (FilaUsuarior1[i] != 0 Ó FilaUsuarior2[i] != 0)entonces
5.                denominador++
6.                si(FilaUsuarior1[i] == FilaUsuarior2[i]) entonces
7.                      numerador++
8.    fin para
9.    retornar numerador/denominador
fin procedimiento
```

Figura 1: Función de similitud entre dos usuarios.

2.3.1.3 Generación de las recomendaciones

La Figura 2 presenta el método para sugerir problemas a resolver usando la función de similitud antes descrita. El método recibe el usuario activo para dar recomendaciones, la matriz de valoraciones usuario-problema, y la cantidad de recomendaciones deseadas. En el primer escenario (línea 1-5) se calculan las similitudes entre el usuario activo y el resto de los usuarios, se almacenan en una matriz local "w", y descendentemente ordenan a los usuarios de acuerdo con los valores de similitud. A continuación, se obtienen los k usuarios más similares (línea 6); seguidamente se puntea cada problema sin resolver por el usuario activo, como la sumatoria de los valores de similitud de los usuarios más similares que resolvieron ese problema. Finalmente, se retornan los "n" mejores problemas con las puntuaciones más altas.

```
procedimiento FC (u_activo, matriz, n)
1.    para cada usuario u diferente de u_activo hacer
2.          w[u_activo, u] = Similitud(u_activo, u)
3.          vecinos.adicionar(u);
4.    fin para
5.    ordenar descendentemente vecinos de acuerdo a w[u_activo, u]
6.    KVecinosSuperiores = vecinos.SubLista(0, K)
7.    puntuaciones[] = {0}
8.    para cada usuario u en KVecinosSuperioreshacer
9.          para cada problema p hacer
10.               si ((matriz[u,p]==1 Ó matriz[u,p]==2) Y
      (matriz[u_activo,p]!=1 Y matriz[u_activo,p]!=2)) entonces
11.                     puntuaciones[p] +=w[u_activo, u]
12.         fin para
13.   fin para
14.   retornar lista de n problemas mejor puntuados
fin procedimiento
```

Figura 2: Filtrado colaborativo adaptado a un escenario de juez en línea.

2.3.2 Tratamiento del arranque en frío

A la hora de concebir un sistema de recomendación es importante tener en cuenta sus limitaciones y plantear estrategias para enfrentarlas. El arranque en frío es una de estas limitaciones. El trabajo desarrollado por (An e-Learning Collaborative Filtering Approach to Suggest Problems to Solve in Programming Online Judges,

2014) no contempla este caso, por tanto, a continuación se proponen dos estrategias para enfrentar este problema.

El algoritmo presentado no genera recomendaciones en el caso que el usuario que solicita recomendaciones no tenga ningún problema resuelto. Bajo este escenario se opta por usar un criterio emergente basado en la popularidad de los problemas, dada por la valoración emitida por los usuarios una vez que resuelve un problema. Este método consiste en recomendar de los problemas existentes en el sistema los dos más populares de cada nivel de dificultad, garantizando la diversidad en cuanto a los niveles de habilidades de los usuarios.

Son estos problemas la antesala de la resolución de un buen número de ejercicios en el juez en línea, y la resolución de los mismos implicaría la generación de nuevas buenas recomendaciones que, en caso de su resolución exitosa, permitirían engrosar el perfil del usuario hasta un punto en el que el algoritmo de filtrado colaborativo comenzaría a ser eficaz.

Por otra parte, los problemas que son recientemente registrados en el sistema, inicialmente no tienen valoraciones; por tanto, aparecerán con poca frecuencia en los listados de recomendaciones. En este caso, se opta por adicionar en los listados de recomendaciones aquellos problemas que contengan pocas o ninguna valoración, con el objetivo de que los usuarios resuelvan estos problemas, y generar valoraciones sobre estos.

2.3.3 Mejorando la escalabilidad de las recomendaciones

En epígrafes anteriores se mencionó que una de las limitaciones de los algoritmos de recomendación es la escalabilidad, específicamente en este trabajo, la etapa de formación de vecinos es la que más sufre este problema, dado que para calcular los vecinos más cercanos de un usuario, es necesario calcular su similitud con el resto de usuarios, lo que resulta una tarea muy costosa computacionalmente.

Anteriormente se mencionó que el agrupamiento es una solución a esta problemática, permitiendo reducir el espacio contenedor de los datos. Este tipo de técnicas en este trabajo tendría un impacto muy positivo, dado que como resultado de su aplicación, se obtendrían grupos de usuarios similares, reduciendo el espacio de búsqueda de tal forma que para encontrar los vecinos más cercanos a un usuario, bastaría con buscar en el grupo al que pertenece ese usuario.

Entre los algoritmos clásicos de agrupamiento se encuentra el K-Medoids, el cual surge como alternativa al K-Means ante dominios donde los objetos están descritos por rasgos categóricos, obteniendo en estos casos una mayor efectividad (A K-means-like Algorithm for K-medoids Clustering and Its Performance, 2006). Se selecciona este algoritmo para solucionar el problema de la escalabilidad, ya que se adapta fácilmente a las características del dominio en cuestión, donde los usuarios están descritos por etiquetas (matriz [u, p]) que determinan el comportamiento de un usuario ante un problema determinado.

El algoritmo consiste en reducir la suma de las distancias entre cada objeto y su correspondiente medoide[6] (Ecuación 4). Inicialmente se seleccionan k objetos (medoides), luego se realiza el proceso iterativo de asignación de los objetos al medoide más cercano de acuerdo con una medida de proximidad o de distancia, luego la actualización de los medoides. Este último paso se interpreta como buscar el objeto que mejor represente al grupo.

$$E = \sum_{i=1}^{k} \sum_{p \in M_i} |\boldsymbol{p} - \boldsymbol{m}_i|^2$$

Ecuación 3: Suma del error cuadrático en K-Medoids

El algoritmo trata de minimizar E que es la suma del error cuadrático, p y m_i son los objetos y el medoide del i-ésimo grupo respectivamente. El proceso iterativo continúa, hasta que no haya cambios en los medoides entre una iteración y la anterior o se alcance un número prefijado de iteraciones. Es un algoritmo fácilmente de implementar, aun así, posee las siguientes limitantes (K-means v/s K-medoids: A Comparative Study, 2011):

1. Es necesario conocer el número de grupos (k).
2. Tiende a converger en óptimos locales.
3. El resultado y el tiempo final de funcionamiento dependen de la selección de medoides inicialmente escogida.

[6] Objeto real más representativo de un grupo

Para el proceso de agrupamiento es necesario definir una función de proximidad o distancia entre usuarios, en este caso resulta muy conveniente emplear la medida (Figura 1) utilizada en la etapa de formación de vecinos.

2.4 Propuesta de interfaces de explicación de las recomendaciones

En esta sección se aborda la propuesta de interfaces de explicación de las recomendaciones. La explicación debe ser entendible, capaz de lograr que el usuario confíe en la recomendación que se le hace. Estas pueden ser de diferentes formas, pues depende del criterio que se siga, y algunas pueden ser más aceptadas por los usuarios que otras. Es posible lograr la evaluación de varios tipos de explicaciones, con el objetivo de constatar cuáles son las más efectivas y aceptadas por los usuarios, y así ayudar a que los sistemas de recomendación sean aún más dinámicos y empleados.

En el presente trabajo se realiza la propuesta de interfaces de explicación para el sistema de recomendación basado en filtrado colaborativo y se describe brevemente el diseño de las mismas. Recientemente han sido desarrollados con este propósito trabajos tales como (A generalized taxonomy of explanations styles for traditional and social recommender systems, 2012), aunque se tomará el de (Herlocker, y otros, 2000) como referencia principal por ser el que dio inicio a esta línea, los autores explican cómo pueden diseñarse interfaces de explicación para este tipo de sistemas de recomendación.

2.4.1 Interfaces de explicación basadas en el funcionamiento del sistema

Es importante explicar al usuario cómo funciona el sistema de recomendación, como su interacción con el sistema puede afectar el resultado de las recomendaciones. Para esto es necesario describir qué información fue procesada para localizar sus vecinos, de los vecinos seleccionados: Conocer su perfil, cuán similares son y cuál fue su aportación en el listado de las recomendaciones. Con estas interfaces de explicación se le aseguraría al usuario que el sistema de recomendación ha seleccionado el conjunto de vecinos correcto y por ende que las recomendaciones son buenas (Herlocker, y otros, 2000).

A continuación se describe brevemente la propuesta de 3 interfaces de explicación basadas en el funcionamiento del método de recomendación. Se muestran además los prototipos de interfaces para cada una de ellas.

2.4.1.1 Interfaz de explicación 1

La primera interfaz que se propone muestra al usuario actual cuáles son los usuarios más semejantes a él, permitiéndole establecer una comparación de su perfil con el de los mismos.

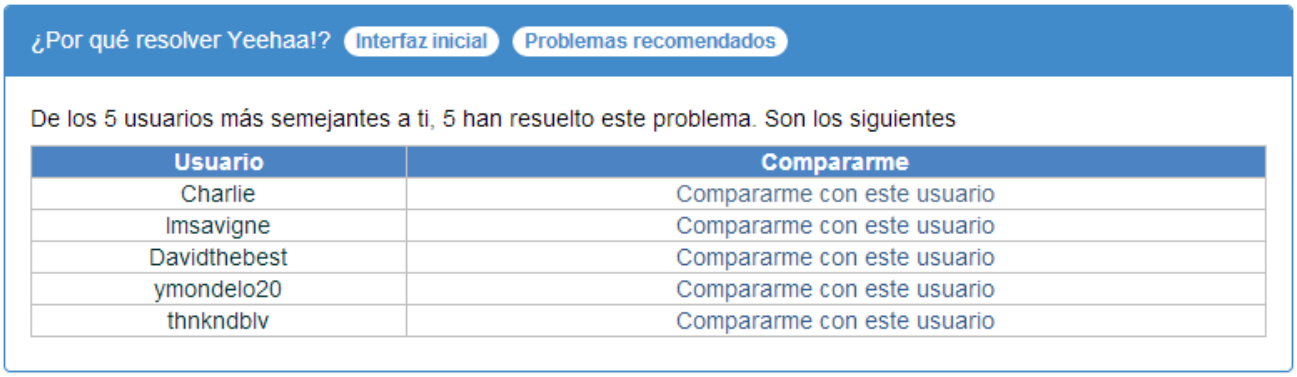


Ilustración 1: Prototipo de interfaz de explicación 1

2.4.1.2 Interfaz de explicación 2

La segunda interfaz de explicación que se propone brinda más detalles del método de recomendación. El usuario podrá conocer el valor de similitud de sus usuarios vecinos, además del peso en el listado de recomendaciones del problema en cuestión.

¿Por qué resolver Yeehaa!? Interfaz inicial Problemas recomendados

Los 5 usuarios más semejantes a usted con su valor de semejanza son:

Usuario	Similitud
Charlie	0.72
lmsavigne	0.59
Davidthebest	0.58
ymondelo20	0.54
thnkndblv	0.53

De ellos los siguientes han resuelto Yeehaa!

Usuario	Similitud
Charlie	0.72
lmsavigne	0.59
Davidthebest	0.58
ymondelo20	0.54
thnkndblv	0.53

Considerando esto el peso final de la recomendación es:

0.72 + 0.59 + 0.58 + 0.54 + 0.53 = 2.96/5 = 0.59

Este problema ocupa el 5 lugar en cuanto al peso en el listado de recomendaciones, considerando todos los problemas y el usuario actual.

Ilustración 2: Prototipo de interfaz de explicación 2

2.4.1.3 Interfaz de explicación 3

La tercera interfaz de explicación que se propone se basa en mostrar la información procesada para determinar el conjunto de vecinos, y muestra una tabla comparativa entre el usuario actual y sus vecinos resaltando aquellos que han resuelto el problema en cuestión.

Ilustración 3: Prototipo de interfaz de explicación 3

2.4.2 Interfaces de explicación basadas en histogramas

Otro de los aspectos que es importante explicar en una recomendación es el historial de valoraciones de los vecinos sobre los problemas recomendados. Mediante un experimento realizado en (Herlocker, y otros, 2000) se comprueba que las interfaces de explicación basadas en histogramas de valoraciones son unas de las más aceptadas por los usuarios. Mediante estas se podría dar al usuario la habilidad de examinar las calificaciones de los vecinos, reforzando la decisión de tomar o desechar la recomendación.

A continuación se describe la propuesta de dos interfaces de explicación basadas en histogramas, y se muestran además los prototipos de las interfaces.

2.4.2.1 Interfaz de explicación 4

La cuarta interfaz que se propone se basa en mostrar en forma de gráfico de barras, los histogramas de las calificaciones de los vecinos asociadas a los intentos de solución del problema en cuestión.

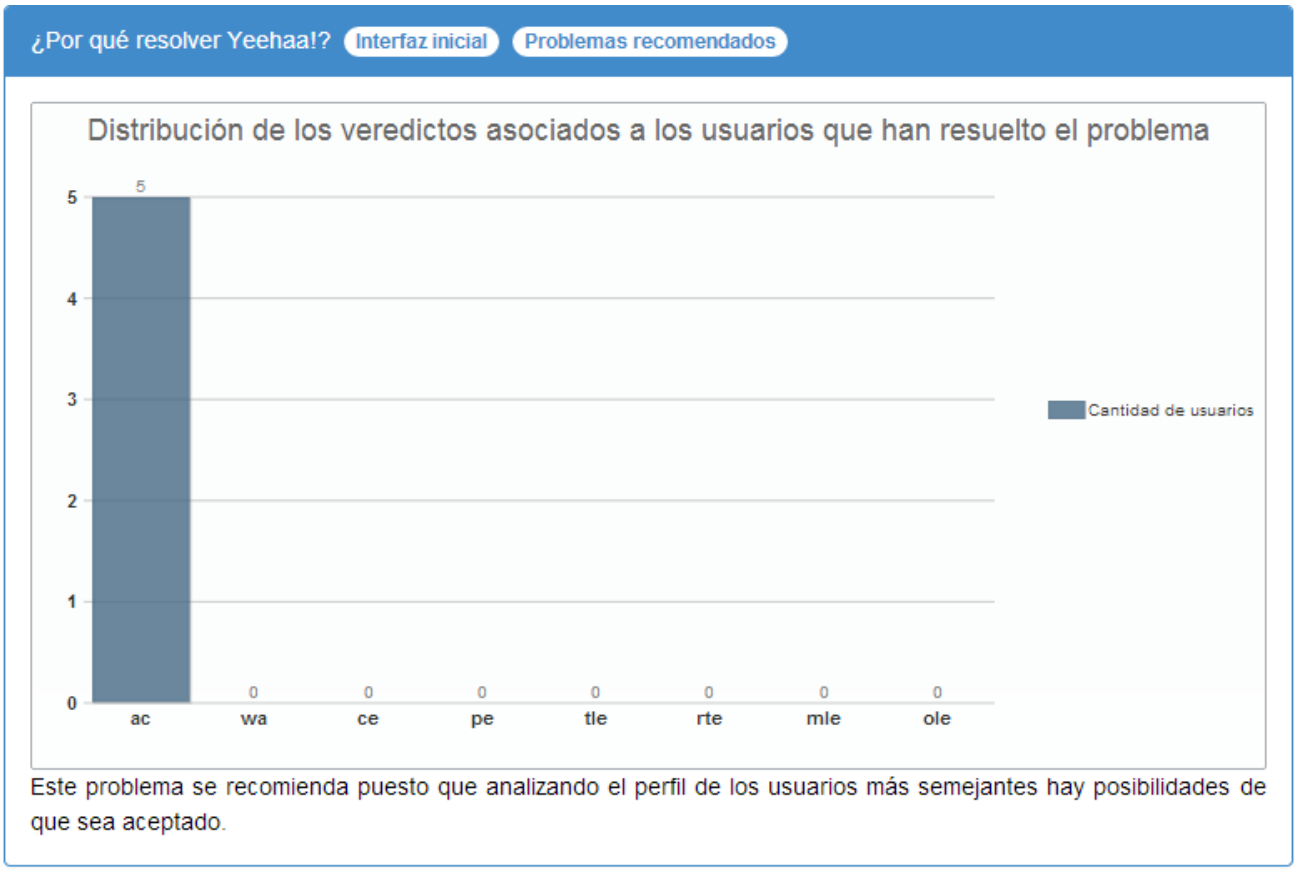


Ilustración 4: Prototipo de interfaz de explicación 4

2.4.2.2 Interfaz de explicación 5

La quinta interfaz de explicación que se propone, muestra de manera gráfica cuán similar es el usuario actual con sus vecinos, teniendo en cuenta problemas resueltos solo por el usuario vecino y los comúnmente resueltos por el usuario actual y sus vecinos.

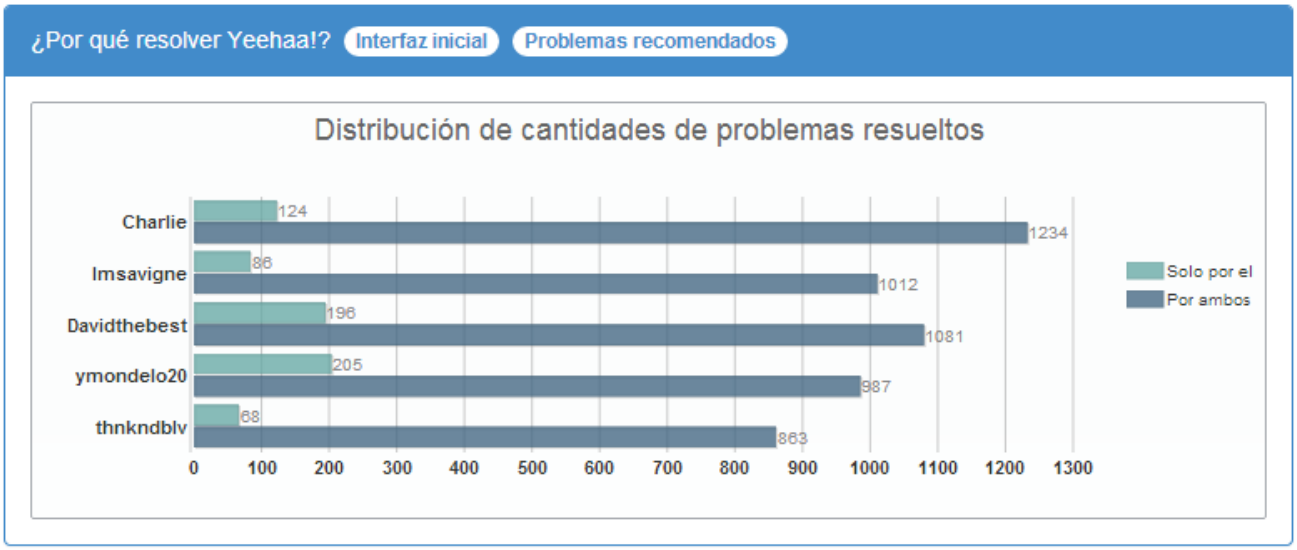


Ilustración 5: Prototipo de interfaz de explicación 5

2.5 Conclusiones parciales

En este capítulo se realizó un análisis de los dos principales enfoques algorítmicos empleados hoy en día para darle solución a la problemática de la recomendación en los jueces en línea: algoritmos basados en memorias y algoritmos basados en modelos; finalmente se selecciona como propuesta de solución los algoritmos basados en memoria debido a su demostrada efectividad y eficiencia. Posteriormente se determinó "el arranque en frío" y la "Escalabilidad" como unas de las limitaciones o problemas más graves que a menudo tienen que lidiar los sistemas de recomendación actuales, así como las estrategias empleadas para solucionar estas inconvenientes en el sistema de recomendación propuesto.

Como parte de la solución para la generación de explicaciones a los usuarios en el sistema de recomendación en desarrollo, se proponen interfaces de explicación que representan un mecanismo que aporta credibilidad y convencimiento en las sugerencias de problemas proporcionadas.

CAPÍTULO 3

DISEÑO E IMPLEMENTACIÓN DEL SISTEMA

En este capítulo se hace énfasis en el diseño y la planificación propios de la metodología de desarrollo utilizada para la implementación del sistema propuesto. La metodología XP hace referencia a los diseños simples y claros. Por ello, XP propone implementar el diseño lo más simple posible, sugiere no adelantar la implementación de funcionalidades que no correspondan a la iteración en la que se esté trabajando. Además se confeccionarán algunos artefactos según la metodología seleccionada.

3.1 Fase de exploración

En esta fase, el cliente define lo que necesita mediante la redacción de sencillas "historias de usuarios" (Joskowicz, 2008). El equipo de desarrollo hace la estimación de los tiempos de desarrollo en base a esta información. Además, es el momento en el que el equipo comienza a familiarizarse con las herramientas, tecnologías y procesos que utilizarán para la elaboración del proyecto.

3.1.1 Historias de usuarios

Las Historia de usuarios (HU) son utilizadas por XP para definir los requisitos del software. En estas el cliente describe brevemente las características que debe poseer el sistema y además es el encargado de asignarles una prioridad (Joskowicz, 2008). También se define el riesgo de desarrollo y el tiempo necesario para la implementación. El programador es el encargado de asignarle un costo de acuerdo con el esfuerzo estimado. Las HU son muy dinámicas y flexibles; además, pueden modificarse en cualquier momento. Estas deben ser suficientemente comprensibles y delimitadas para que se puedan implementar en pocas semanas.

En el presente trabajo se definieron un total de 8 HU. A continuación se muestra la HU de máxima prioridad, el resto podrá consultarlas en el Anexo 1.

Historia de usuario	
Número:1	**Nombre:** Mostrar recomendación a los usuarios autenticados
Requisitos funcionales implícitos: RF1 - Generar el perfil de recomendaciones de los usuarios autenticados RF2 - Mostrar recomendaciones a usuarios autenticados	
Usuarios: Eiquel Osorio Ramírez y Randy Espino Lobaina	
Prioridad en el negocio: Máxima	**Riesgo en desarrollo:** Alto
Puntos estimados: 5	**Iteración asignada:** 1
Descripción: Cuando el usuario está autenticado en el COJ y solicita una recomendación, el sistema genera un perfil de recomendaciones que se utiliza para mostrar las recomendaciones al usuario en cuestión.	
Observaciones: -	

Tabla 1: Historia de Usuario 1

3.1.2 Lista de reserva del producto

Después de analizado el dominio del problema, se identifican los requisitos funcionales y no funcionales que tendrá el sistema. 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 (Pressman, 2005).

La lista de reserva del producto es una tabla que contiene los requisitos funcionales y no funcionales que debe cumplir el sistema a desarrollar (Ver Tabla 4), ordenados según su prioridad de implementación que puede ser: Alta, Media y Baja. Se indica además el tiempo estimado por semanas para la implementación de los requisitos funcionales y el rol que realizó la estimación.

No.	Requisito	Estimación	Estimado por
	Prioridad: Alta		
1	Generar el perfil de recomendaciones de los usuarios autenticados	5	Analista
2	Mostrar recomendaciones a usuarios autenticados	0.5	Analista
	Prioridad: Media		
3	Mostrar Interfaz-1 de explicación de la recomendación	1	Analista
4	Mostrar Interfaz-2 de explicación de la recomendación	1	Analista
5	Mostrar Interfaz-3 de explicación de la recomendación	1	Analista
6	Mostrar Interfaz-4 de explicación de la recomendación	1	Analista
7	Mostrar Interfaz-5 de explicación de la recomendación	1	Analista
	Prioridad: Baja		
8	Valorar las interfaces de explicación de las recomendaciones	1	Analista
9	Mostrar la Interfaz de explicación recomendada	1	Analista
1	Usabilidad: El sistema podrá ser usado sobre ambiente web por personas con pocos conocimientos de informática.		
2	Confiabilidad: - En caso de que el sistema presente alguna falla, los errores se deben mostrar sin detalles de información que puedan comprometer la seguridad e integridad del sistema. Solo podrán mostrarse detalles de la información para los usuarios con privilegios de administración dentro del sistema.		

	- El sistema establecerá mecanismos para garantizar la confiabilidad e integridad de la información ante posibles accesos no autorizados. - Se podrá acceder a la aplicación desde cualquier navegador web, ya sea en Internet Explorer (v8.0 o superior), Mozilla Firefox (v3.0 o superior), Chrome (v5.0 o superior), Opera (v9.0 o superior) y Safari (v4.0 o superior).		
3	Eficiencia: Se esperan alrededor de hasta 50 usuarios conectados concurrentemente. Por tanto cada recomendación realizada debe poseer un tiempo de respuesta rápido.		
4	Interfaz: El diseño cumple con los estándares internacionales de desarrollo web aplicados al COJ, porque presenta una interfaz sencilla e intuitiva que garantiza mayor nivel de usabilidad.		
5	Interfaz de Software: - PC Servidor web. Sistema Operativo: Ubuntu 13.04 o superior. Servidor web: apache Tomcat 7.x o superior. - Lenguaje de programación: PHP 5.3.9 o superior (librerías: pgsql, curl, gd, db, upload progress). - PC Servidor de BD Sistema Operativo: Ubuntu 13.04 o superior. Sistema Gestor de Base de Datos PostgreSQL v		

	8.4 o superior. Se empleará el Sistema Gestor de Base de Datos de PostgreSQL PgAdmin III. - Utilizar el entorno de ejecución java "OpenJDK-JRE'' en su versión 6 o superior.		
6	Requisitos legales, de derecho de autor y otros: - Las tecnologías y herramientas utilizadas están bajo licencia GPL. - El sistema posee una sección donde se publican los términos y condiciones de uso que reglamentan el acceso y uso del COJ y de todos sus subdominios o dominios secundarios incluyendo sus contenidos y servicios, puestos a disposición que deben ser de estricto cumplimiento por parte de los usuarios. - Los derechos de autor sobre la información de terceros que se publica en el COJ pertenecen a sus respectivos autores y su publicación en el COJ se realiza sin ánimos de lucro y con un propósito informativo.		

Tabla 2: Lista de reserva del producto

3.2 Fase de planificación

XP plantea la planificación como un diálogo continuo entre las partes involucradas en el proyecto, incluyendo al cliente, a los programadores y a los coordinadores o gerentes. Durante esta fase se realiza una estimación del esfuerzo en semanas que costará implementar cada historia de usuario. Típicamente esta fase consiste en una o varias reuniones grupales de planificación. El resultado de esta fase es un Plan de Entregas (Joskowicz, 2008).

3.2.1 Plan de iteraciones

Este plan define exactamente cuáles historias de usuario serán implementadas para cada iteración del sistema, según el orden establecido. Al principio de cada interacción se realiza una reunión de planificación.

Cada historia de usuario se traduce en tareas específicas de programación. Así mismo, para cada historia de usuario se establecen las pruebas de aceptación. Estas pruebas se realizan al final del ciclo en el que se desarrollan, pero también al final de cada uno de los ciclos siguientes, para verificar que subsiguientes iteraciones no han afectado a las anteriores. Las pruebas de aceptación que hayan fallado en el ciclo anterior son analizadas para evaluar su corrección, así como para prever que no vuelvan a ocurrir (Cabrera, y otros, 2010).

Iteración 1.
En la primera iteración se implementan las HU con mayor prioridad. Al finalizar se contará con las funcionalidades descritas en la historia de usuario número 1. También se proveerá la primera versión de prueba, dando al sistema las primeras funcionalidades. Esta versión será mostrada al cliente con el objetivo de obtener una retroalimentación para el grupo de trabajo.

Iteración 2.
El objetivo de esta iteración es la implementación de las funcionalidades con prioridad media. Con el cumplimiento de esta se tendrán implementadas las solicitudes del cliente descritas en la historias de usuarios. De esta forma, se obtiene la segunda versión de pruebas del software. Esta versión, junto a las implementaciones anteriores, será mostrada al cliente con el objetivo de realizar cambios en base a la aceptación del mismo.

Iteración 3.
En la última iteración se implementan las funcionalidades con prioridad baja. Con el cumplimiento de esta se tendrán implementadas las solicitudes del cliente descritas en las historias de usuarios. Como resultado de esta iteración se tendrá la versión 1.0 del producto final. A partir de este momento el sistema será puesto a prueba por un período de tiempo, para evaluar el desempeño del mismo.

3.2.2 Plan de duración de las iteraciones.

Como parte del ciclo de vida de un proyecto se crea el plan de duración de cada una de las iteraciones. Este plan tiene como objetivo reflejar la duración de cada iteración, así como el orden en que serán implementadas cada HU en cada una de las iteraciones a realizar, lo que brinda una idea aproximada del tiempo de duración en la confección del sistema en su totalidad.

Iteraciones	Orden de las Historia de usuario a implementar	Duración total de las iteraciones
Iteración 1	1- Mostrar recomendación a los usuarios autenticados**.**	5 semanas
Iteración 2	1- Mostrar interfaz de explicación 1 de recomendaciones a los usuarios autenticados. 2- Mostrar interfaz de explicación 2 de recomendaciones a los usuarios autenticados. 3- Mostrar interfaz de explicación 3 de recomendaciones a los usuarios autenticados. 4- Mostrar interfaz de explicación 4 de recomendaciones a los usuarios autenticados. 5- Mostrar interfaz de explicación 5 de recomendaciones a los usuarios autenticados.	5 semanas
Iteración 3	1- Valorar las interfaces de explicaciones. 2- Mostrar la Interfaz de explicación recomendada a los usuarios autenticados.	2 semanas

Tabla 3: Plan de duración de las iteraciones

3.2.3 Plan de entrega

El cronograma de entregas establece qué historias de usuario serán agrupadas para conformar una entrega, y el orden de las mismas. Este cronograma será el resultado de una reunión entre todos los actores del proyecto (cliente, desarrolladores, gerentes, etc.). Típicamente el cliente ordenará y agrupará según sus prioridades las historias de usuario. El cronograma de entregas se realiza en base a las estimaciones de tiempos de desarrollo

realizadas por los desarrolladores. Luego de algunas iteraciones es recomendable realizar nuevamente una reunión con los actores del proyecto, para evaluar nuevamente el plan de entregas y ajustarlo si es necesario. (Joskowicz, 2008)

Producto	Final 1ra. Iteración 1era. semana de marzo	Final 2da. Iteración 1era. semana de Abril	Final 3ra. Iteración 1era. semana de Mayo
Módulo de recomendación de problemas para el Juez en Línea Caribeño versión 2.0	0.1	0.2	1.0

Tabla 4: Plan de entrega

3.3 Diseño del sistema

Con el objetivo de la comprensión del sistema, en el presente epígrafe se dará una descripción de la arquitectura y diseño del mismo.

3.3.1 Arquitectura del software

La arquitectura es fundamental en cualquier sistema, pues como un modelo de referencia, permite centrarse en la estructura y funciones del mismo, haciendo posible especificar características inherentes a su implementación.

Un concepto ampliamente adoptado por diferentes autores, enuncia como arquitectura de software a "la estructura o estructuras del sistema, lo que comprende a los componentes del software, sus propiedades externas visibles y las relaciones entre ellos." (Escribano, 2002)

La arquitectura de software describe detalles de diseño mediante una colección de componentes y sus relaciones, conformando una vista del sistema en diseño, lo cual incide en aspectos del desarrollo de software como la comprensión, reutilización y construcción de este.

3.3.1.1 Arquitectura basada en el patrón Modelo Vista Controlador

El patrón de arquitectura de software empleado para el desarrollo de la aplicación es la arquitectura Modelo Vista Controlador (MVC). El MVC separa los datos y la lógica de negocio de una aplicación de la interfaz de usuario y el módulo encargado de gestionar los eventos y las comunicaciones. Este estilo arquitectónico propone la construcción de tres componentes distintos denominados modelo-vista-controlador. El componente Modelo propone una representación de la información con el cual el sistema opera o manipula, gestionando todos los accesos a tal información, tanto consultas como actualizaciones, implementando también los privilegios de acceso que se hayan definido en la lógica de negocio. El componente Controlador responde a las acciones de los usuarios e invoca peticiones al componente modelo cuando se hace una solicitud sobre la información, y el componente Vista presenta la información y lógica del negocio en un formato con el cual el usuario sea capaz de interactuar (usualmente interfaz de usuario).

El framework Spring empleado en el desarrollo del presente módulo basa su funcionamiento en la implementación del patrón arquitectónico MVC, propone una estructura basada en modelos (Clases DAO), vistas (Ficheros ".jsp") y controladores (Ficheros ".java"). Una de las razones de la utilización de este patrón de arquitectura es debido a su utilización en el COJ, plataforma donde será implementado este sistema de recomendación.

3.3.2 Modelo de datos

Un modelo de datos es un "mecanismo formal para representar y manipular información de manera general y sistemática" (Silberschatz, y otros, 2007). Peculiarmente un modelo de datos permite:

- Descripción de datos: El tipo de los datos que hay en la base y la forma en que se relacionan.
- Reglas de integridad: Un conjunto de condiciones que deben cumplir los datos para reflejar correctamente la realidad deseada.
- Operaciones: Típicamente, operaciones de agregado, borrado, modificación y recuperación de los datos de la base (Silberschatz, y otros, 2007).

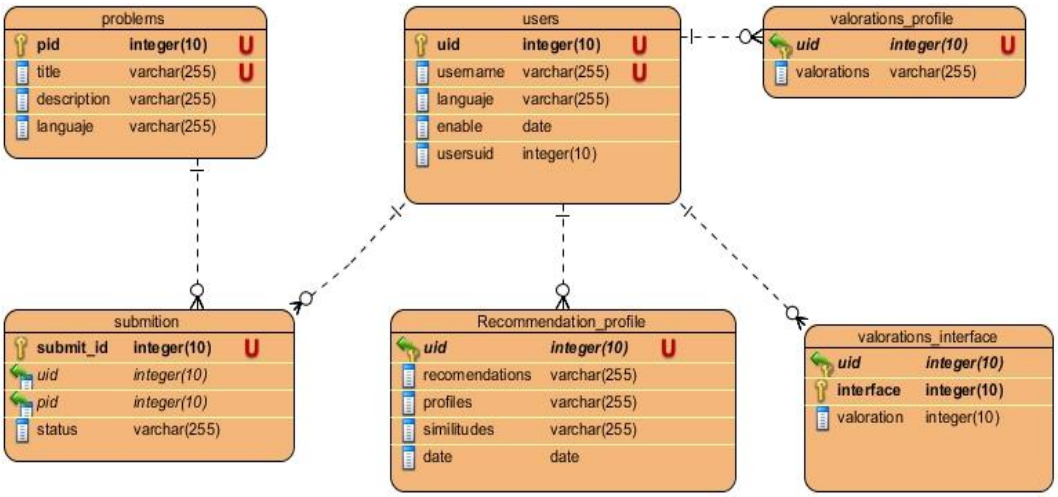


Ilustración 6: Modelo de datos del sistema

3.3.3 Diagrama de paquetes

Un diagrama de paquetes muestra cómo un sistema está dividido en agrupaciones lógicas mostrando las dependencias entre esas agrupaciones. Además cada paquete puede asignarse a una persona o a un equipo, y las dependencias entre ellos pueden indicar el orden de desarrollo requerido. El diagrama que se muestra a continuación representa cómo está estructurado el sistema. Cada paquete puede contener otros paquetes o clases, que tienen interfaces y realizan cierta funcionalidad (UML, 2011).

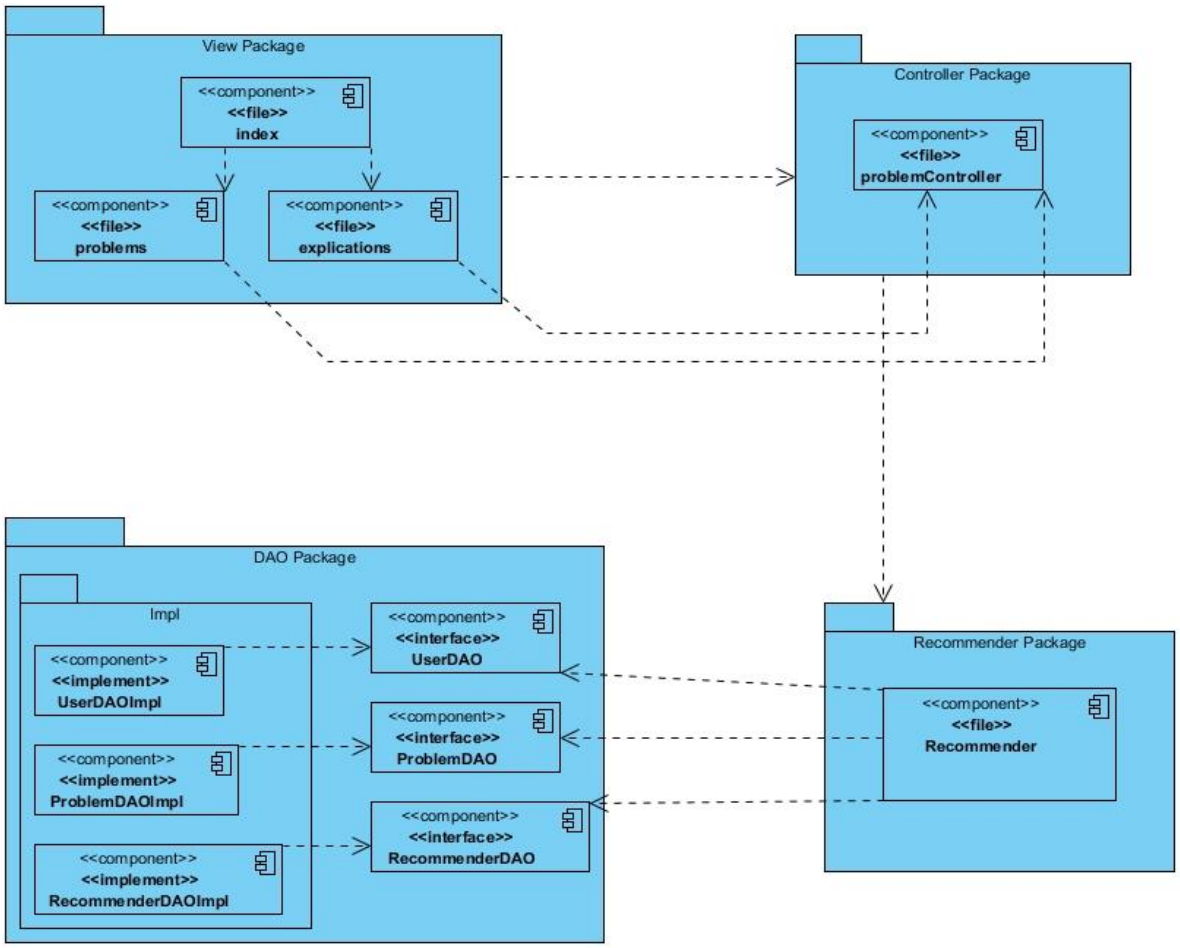
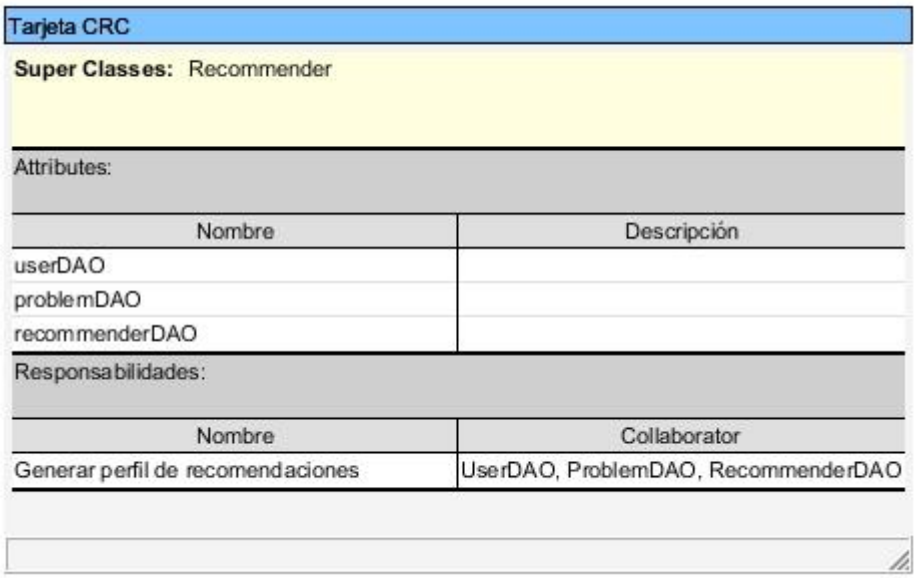


Ilustración 7: Diagrama de paquetes

3.3.4 Tarjetas CRC

En XP, el uso de Tarjetas CRC permite modelar el sistema a través de clases y sus responsabilidades. Estas tarjetas son de muy fácil elaboración y en su construcción participa todo el equipo de desarrollo. Como se verá, están compuestas por el nombre de la clase y dos columnas: la izquierda es usada para definir los atributos y responsabilidades de las clases, mientras que la derecha contiene los nombres de las clases con las que se relaciona para llevar a cabo su labor.



Tarjeta CRC

Super Classes: Recommender

Attributes:

Nombre	Descripción
userDAO	
problemDAO	
recommenderDAO	

Responsabilidades:

Nombre	Collaborator
Generar perfil de recomendaciones	UserDAO, ProblemDAO, RecommenderDAO

Ilustración 8: CRC Generar perfil de recomendaciones

3.3.5 Diagrama de despliegue

El Diagrama de Despliegue es un tipo de diagrama UML que se utiliza para modelar la disposición física de los artefactos software en nodos (usualmente plataforma de hardware) (OMG, 2007). Dentro de los usos que se les da a los diagramas de despliegue son para modelar:

- **Sistemas empotrados**: Un sistema empotrado es una colección de hardware con una gran cantidad de software que interactúa con el mundo físico.
- **Sistemas cliente-servidor**: Los sistemas cliente-servidor son un extremo del espectro de los sistemas distribuidos y requieren tomar decisiones sobre la conectividad de red de los clientes a los servidores y sobre la distribución física de los componentes software del sistema a través de nodos.
- **Sistemas completamente distribuidos**: En el otro extremo encontramos aquellos sistemas que son ampliamente o totalmente distribuidos y que normalmente incluyen varios niveles de servidores. Tales sistemas contienen a menudo varias versiones de componentes de software, alguno de los cuales pueden incluso migrar de un nodo a otro. El diseño de tales sistemas requiere tomar decisiones que permitan un cambio continuo de la topología del sistema.

Ilustración 9: Diagrama de despliegue del sistema

Descripción de los componentes:

- Client PC*:* El Cliente PC es donde se podrá mostrar e interactuar con el juez en línea a través de un navegador web previamente instalado en el ordenador.
- Apache Tomcat: El Servidor Apache Tomcat es donde se encuentra alojado la aplicación web, a la que se conectan todos los clientes por medio de sus estaciones de trabajo.
- PostgreSQL: En el Servidor PostgreSQL es donde se encuentran almacenados todos los datos que permanecen en el sistema.
- TCP/IP: TCP/IP es el protocolo de comunicación entre el servidor web y el servidor de base de datos empleado, este proporciona la transferencia confiable de paquetes de datos sobre la red.
- HTTP: El protocolo HTTP se utiliza para la conexión al servidor web desde las computadoras clientes, este protocolo implementa un canal de comunicación seguro, basado en SSL (Secure Socket Layers) entre el navegador del cliente y el servidor HTTP.

3.4 Fase de implementación del sistema

La fase más importante en los procesos estructurados dentro en la metodología XP es el desarrollo. A medida que se itera se realiza la implementación de las historias de usuario seleccionadas. Desde el inicio se realizan revisiones del plan de iteraciones confeccionado, el cual se modifica y es mejorado en caso de ser necesario, esto se hace de manera continua a lo largo del proyecto.

"El desarrollo de las HU es guiado por las Tareas de Ingeniería, las cuales son consideradas como una herramienta de planificación esencial para el programador" (Harmon, 2008). Estas, especifican de forma técnica los detalles relacionados con la realización de cada HU en cada iteración, enunciando las actividades necesitadas para concretar su implementación.

3.4.1 Iteración 1

En esta iteración se implementa la Historia de Usuario de mayor prioridad en el sistema, con el objetivo de obtener una primera versión del producto con las principales características y funcionalidades que posteriormente serán mostradas al cliente.

Historias de usuarios	Tiempo de Implementación (Semanas)	
	Estimación	Real
1. Mostrar recomendación a los usuarios autenticados.	5	5

Tabla 5: Historias de usuarios desarrolladas en la iteración 1

Tarea de Ingeniería	
Número de tarea : 1	**Número de historia de usuario: 1**
Nombre de la Tarea: Programación de la propuesta de recomendación para usuarios autenticados	
Tipo de tarea: Desarrollo	Puntos Estimados: 5
Fecha de inicio: 2/03/2015	**Fecha fin:** 6/04/2015
Programador responsable: Eiquel Osorio Ramírez y Randy Espino Lobaina	
Descripción: Se realiza la implementación de la propuesta de recomendación de problemas para usuarios autenticados en el sistema.	

Tabla 6: Tarea de ingeniería 1

3.4.2 Iteración 2

En esta iteración se implementan las Historias de Usuarios de prioridad media en el sistema, con el objetivo de obtener una segunda versión del producto con sus respectivas funcionalidades que posteriormente serán mostradas al cliente.

Historias de usuarios	Tiempo de Implementación (Semanas)	
	Estimación	Real
Mostrar interfaz de explicación 1 de recomendaciones a los usuarios autenticados.	1	1
Mostrar interfaz de explicación 2 de recomendaciones a los usuarios autenticados.	1	1
Mostrar interfaz de explicación 3 de recomendaciones a los usuarios autenticados.	1	1
Mostrar interfaz de explicación 4 de recomendaciones a los usuarios autenticados.	1	1
Mostrar interfaz de explicación 5 de recomendaciones a los usuarios autenticados.	1	1

Tabla 7: Historias de usuario desarrolladas en la iteración 2

Para el desarrollo de esta iteración es necesario la realización de 5 tareas de ingeniería. A continuación se muestra la tarea de ingeniería asociada a la segunda HU.

Tarea de Ingeniería	
Número de tarea : 2	**Número de historia de usuario: 2**
Nombre de la Tarea: Implementar la propuesta de interfaz de explicación 1 de las recomendaciones	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha de inicio: 8/04/2015	**Fecha fin:** 15/04/2015
Programador responsable: Eiquel Osorio Ramírez y Randy Espino Lobaina	
Descripción: Se realiza la implementación de la propuesta la primera interfaz de explicación de las recomendaciones.	

Tabla 8: Tarea de Ingeniería 2

3.4.3 Iteración 3

En esta iteración se implementa las HU de menor prioridad en el sistema, con el objetivo de obtener una la última versión del producto que posteriormente será mostrado al cliente.

Historias de usuarios	Tiempo de Implementación (Semanas)	
	Estimación	Real
Valorar las interfaces de explicación	1	1
Mostrar la Interfaz de explicación recomendada a los usuarios autenticados.	1	1

Tabla 9: Historias de usuario desarrolladas en la iteración 3

Tarea de Ingeniería	
Número de tarea : 7	**Número de historia de usuario:** 7
Nombre de la Tarea: Implementar el requisito valorar las interfaces de explicación	
Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha de inicio: 8/04/2015	**Fecha fin:** 15/04/2015
Programador responsable: Eiquel Osorio Ramírez y Randy Espino Lobaina	
Descripción: Se realiza la implementación de la funcionalidad que permita a los usuarios emitir una valoración de una interfaz de explicación.	

Tabla 10: Tarea de ingeniería 7

Tarea de Ingeniería	
Número de tarea : 8	**Número de historia de usuario:** 8
Nombre de la Tarea: Implementar el requisito mostrar interfaz de explicación recomendada	

Tipo de tarea: Desarrollo	Puntos Estimados: 1
Fecha de inicio: 8/04/2015	**Fecha fin:** 15/04/2015
Programador responsable: Eiquel Osorio Ramírez y Randy Espino Lobaina	
Descripción: Se realiza la implementación de la funcionalidad que muestra a los usuarios la interfaz de explicación recomendada a partir de las valoraciones emitidas por los usuarios	

Tabla 11: Tarea de ingeniería 8

3.5 Conclusiones parciales

En este capítulo se abordó todo lo referente al diseño e implementación del sistema. La elaboración de 8 HU posibilitó la captura de los requisitos funcionales y no funcionales recogidos en la lista de reserva del producto. El uso del patrón arquitectónico modelo vista controlador favorecerá la reutilización e implementación de la solución, el empleo de patrones de diseño en la implementación posibilitó una correcta asignación de responsabilidades entre las clases. Además se generaron artefactos tales como: modelo de datos, diagrama de componentes, diagrama de paquetes, tarjetas CRC y diagrama de despliegue, los cuales tributan a una mejor comprensión del sistema. Finalmente el desglose en 8 tareas de ingeniería de las 8 HU permitió a lo largo de 3 iteraciones implementar la solución propuesta.

CAPÍTULO 4

PRUEBAS

El presente capítulo está dedicado a realizar las pruebas necesarias al sistema para validar los resultados obtenidos. Se realiza una validación de la estrategia de recomendación aplicando métricas referenciadas por autores representativos de esta área de investigación. Se valida también el algoritmo de agrupamiento implementado, mediante un experimento. Además se realizan las pruebas de integración, aceptación y de carga al sistema. Finalmente se muestran los resultados de la realización de las pruebas, en diferentes escenarios.

4.1 Evaluación de la estrategia de recomendación

La evaluación de los sistemas de recomendación constituye una de las tareas más complejas dentro de esta área de investigación. Esto viene dado por el hecho de que resulta extremadamente difícil determinar si una estrategia determinada genera recomendaciones provechosas para los usuarios, debido a la gran variedad de perfiles e intereses distintos que pueden tener cada uno de dichos usuarios.

Para determinar la precisión de un sistema de recomendación muchos prestigiosos autores han priorizado, dentro de su línea de investigación, la obtención de métricas eficaces para evaluar a estos. Una de estas métricas y que preferentemente son empleadas para evaluar a los recomendadores es el "Precision y Recall". Estas miden la frecuencia con la que un sistema hace suposiciones correctas con respecto a si un ítem es realmente apropiado o no para un usuario en específico. Recall se encarga de medir la habilidad de un sistema de presentar todos los ítems relevantes, mientras que Precision mide la capacidad de presentar al usuario final sólo los ítems relevantes. (Evaluating Collaborative Filtering Recommender Systems, 2004)

4.1.1 Aplicación del Precision y Recall

Para la aplicación de estas pruebas se utilizó el mismo protocolo de experimentación utilizado en (An e-Learning Collaborative Filtering Approach to Suggest Problems to Solve in Programming Online Judges, 2014). Además, se siguieron las sugerencias de (A Survey of Accuracy Evaluation Metrics of Recommendation Tasks, 2009) para evaluaciones offline a sistemas de recomendación. Estos últimos proponen dividir el conjunto de datos en un conjunto de prueba y otro de entrenamiento, luego aplicar el algoritmo al conjunto de entrenamiento y generar recomendaciones, y finalmente comparar esas recomendaciones contra el conjunto de prueba.

Para aplicar la evaluación offline para cada usuario, se dispone de un conjunto de datos con los problemas que cada usuario ha resuelto, luego se divide este conjunto, en dos subconjuntos, entrenamiento y prueba. Seguidamente al conjunto de entrenamiento se le aplica el algoritmo a validar, y se seleccionan las n-primeras recomendaciones generadas por este. A su vez, los elementos que aparecen tanto en el conjunto de pruebas como en el de las n-primeras recomendaciones, se consideran como parte de un conjunto llamado hit set. En este escenario el Precision y Recall se define de la siguiente manera (Evaluating Collaborative Filtering Recommender Systems, 2004):

$$recall = \frac{TH}{TP}$$

Ecuación 4: Recall

$$precision = \frac{TH}{CPN}$$

Ecuación 5: Precision

Siendo TH el tamaño del conjunto hit set, TP el tamaño del conjunto de prueba y CPN la cantidad de n-primeras recomendaciones.

Es importante resaltar que en la práctica estas métricas normalmente entran en conflicto. Usualmente, a la hora de incrementar el valor de n en las n primeras recomendaciones a obtener, el valor de Recall tiende a incrementarse, contrariamente Precision tiende a disminuir su valor. Por esta razón, se han visto dirigidos los esfuerzos a obtener una métrica que unifique los dos componentes de la anterior. De los varios resultados alcanzados, la de más aplicación ha sido F1 (A re-examination of text categorization methods, 1999), que es a su vez la seleccionada para utilizarse. F1 se define como:

$$F1 = \frac{2 * precision * recall}{recall + precision}$$

Ecuación 6: F1

Para poder determinar el comportamiento general del sistema de recomendación, no es suficiente con tener el valor de estas métricas para cada uno de sus usuarios. Es necesario obtener valores globales de Precision, Recall y F1 que describan el funcionamiento global del sistema. Con este propósito se tomaran las sugerencias de (A Survey of Accuracy Evaluation Metrics of Recommendation Tasks, 2009), que señala computar los promedios de los valores de Precision y Recall para todos los usuarios, y con sus promedios, calcular F1.

Para el cálculo de estas métricas se utilizó una muestra de los primeros 1 000 usuarios del ranking, la cual es una muestra representativa del sistema en general. Se determinó de forma aleatoria los conjuntos de prueba y entrenamiento para cada usuario usando los problemas resueltos por este. Para cada uno de los usuarios se ejecutaron varias corridas en las que se varió, desde 10 hasta 30, el número N de recomendaciones. En la figura se muestran los resultados de estas corridas.

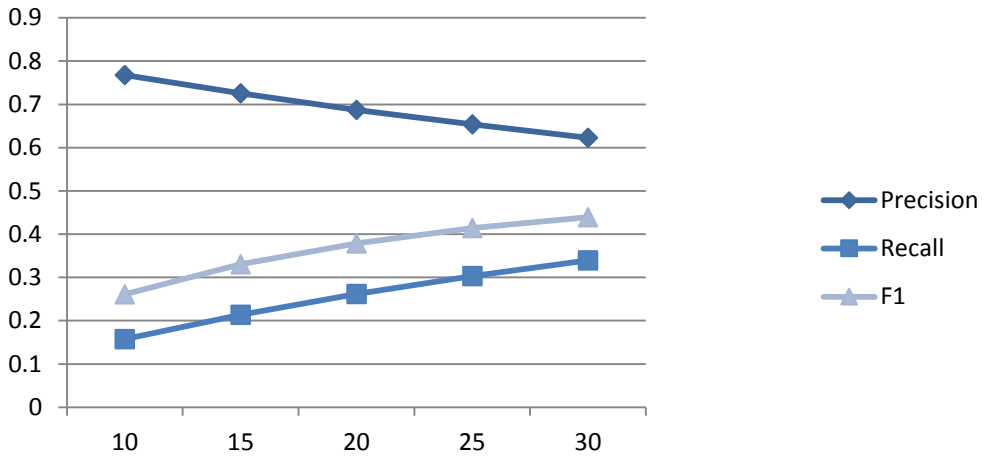


Ilustración 10: Gráfica de los valores globales de Precision, Recall y F1

La gráfica muestra que el algoritmo empleado para generar recomendaciones posee un alto grado de precisión, de los resultados se pueden interpretar que de cada 10 problemas recomendados a un usuario, en 8 de ellos tiene las mayores posibilidades de aceptarlos. Es importante mencionar que los resultados obtenidos luego de aplicar la validación, están en correspondencia con los resultados obtenidos en el trabajo de (An e-Learning Collaborative Filtering Approach to Suggest Problems to Solve in Programming Online Judges,

2014), en donde se valida este algoritmo, y se demuestra que obtiene resultados superiores que otros algoritmos de recomendación.

4.1.1.1 Comparación de las métricas entre el recomendador actual y el desarrollado

La siguiente gráfica muestra una comparación de las métricas de precisión entre el algoritmo de recomendación actual y el desarrollado en este trabajo. Se puede apreciar una considerable superioridad en la precisión de las recomendaciones realizadas por el algoritmo desarrollado en este trabajo.

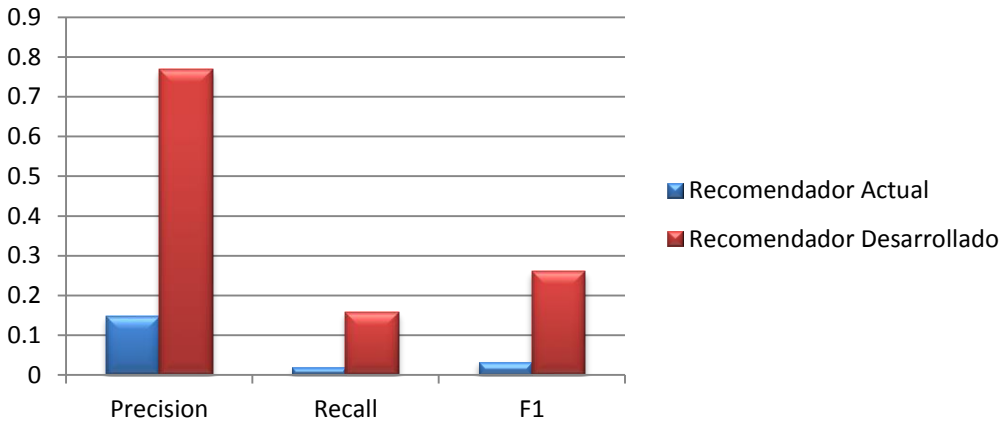


Ilustración 11: Gráfica comparativa entre el recomendador actual y el desarrollado en este trabajo

4.2 Evaluación de la estrategia de agrupamiento

Para la evaluación del algoritmo de agrupamiento empleado, fueron realizadas pruebas destinadas a comprobar el porcentaje de coincidencias al calcular los vecinos para los usuarios, teniendo en cuenta sólo los usuarios del grupo al que pertenece y por otra parte teniendo en cuenta todos los usuarios.

Se realizó una prueba global tomando una muestra de 9 000 usuarios registrados que tiene al menos un problema resuelto. Para cada usuario se calcularon dos listados de los 20 vecinos más cercanos. Para el primer listado se consideraron los usuarios que pertenecen al mismo grupo del usuario en cuestión y para el segundo todos los usuarios de la muestra. Se computaron los porcentajes de coincidencias entre los dos

listados generados para cada usuario obteniendo como resultado final un 99 % de coincidencias, por tanto, la calidad de las recomendaciones no se afectaría empleando el algoritmo de agrupamiento propuesto.

La realización de estas pruebas permitió ajustar el parámetro k para la ejecución del algoritmo de agrupamiento. Se realizaron varias ejecuciones variando este parámetro, resultando ser k=8 uno de los que mejores resultados obtenía en el porcentaje de coincidencias.

4.3 Pruebas de integración

Estas pruebas se realizan para validar que las funcionalidades funcionen correctamente con respecto a su integración con el COJ. Para la realización de las mismas se diseñaron dos casos de pruebas que se describen a continuación.

Caso de prueba: Verificar la integración del módulo de recomendación para la funcionalidad "Mostrar recomendación a usuarios autenticados"
No. De caso de prueba: 1
Módulo a integrar: Recomendación
Condiciones de Ejecución: Algún usuario debe estar previamente autenticado antes de solicitar la recomendación.
Descripción de la prueba: Comprobar que el módulo de Recomendación pueda obtener los datos de los usuarios, para poder realizar la recomendación y que se almacene en la base de datos el perfil de recomendación creado.
Entradas/Pasos de ejecución: – Un usuario debe de estar conectado y autenticado en el sitio. – El usuario debe solicitar la recomendación.
Resultado esperado: Se muestra el listado de problemas recomendados.

Tabla 12: Caso de prueba de integración 1

Caso de prueba: Verificar la integración del módulo de recomendación para la funcionalidad "Mostrar interfaz de explicación recomendada"
No. De caso de prueba: 2
Módulo a integrar: Recomendación
Condiciones de Ejecución: Los usuarios hayan valorado las interfaces de explicación, quedando guardado la valoración en la base de datos.
Descripción de la prueba: Comprobar que el módulo de Recomendación pueda obtener los datos de las interfaces, para poder realizar la recomendación.
Entradas/Pasos de ejecución: – Un usuario debe de estar conectado y autenticado en el sitio. – El usuario debe solicitar la recomendación. – El usuario debe solicitar la interfaz de explicación recomendada.
Resultado esperado: Se muestra la interfaz de explicación recomendada.

Tabla 13: Caso de prueba de integración 2

Los casos de pruebas que se realizaron arrojaron resultados satisfactorios, demostrando que el sistema respondía según lo descrito.

4.4 Pruebas de Aceptación

Las pruebas de aceptación se enfocan en las características generales y funcionalidades del sistema (Pressman, 2005). Estas pruebas derivan de las HU que han sido implementadas y su propósito es garantizar el cumplimiento de los requisitos pautados además de asegurar su correcto funcionamiento.

Caso de prueba de aceptación	
Código de caso de prueba: CP1	**Nombre de la historia de usuario:** Mostrar recomendación a los usuarios autenticados
Responsable de la prueba: Eiquel Osorio Ramírez	
Descripción de la prueba: Prueba de funcionalidad que muestra la recomendación a usuarios autenticados.	
Condiciones de ejecución: Algún usuario debe estar previamente autenticado antes de solicitar la recomendación.	
Entrada/Pasos de ejecución: – Un usuario debe de estar conectado y autenticado en el sitio. – El usuario debe solicitar la recomendación.	
Resultado esperado: Se muestran las recomendaciones al usuario autenticado.	
Evaluación de la prueba: Prueba satisfactoria.	

Tabla 14: Caso de pruebas de aceptación 1

Se realizaron 8 casos de pruebas para cada una de las HU, los resultados obtenidos fueron positivos quedando el cliente satisfecho con cada una de las funcionalidades implementadas.

4.5 Conclusiones parciales

En este capítulo se realizaron diferentes pruebas para validar el módulo de recomendación desarrollado. Las métricas de validación de sistemas de recomendación aplicadas, permitieron llegar a la conclusión que el sistema generalmente proporciona recomendaciones con una alta precisión. Las pruebas realizadas al algoritmo de agrupamiento implementado permitieron comprobar su total validez ya que no afecta el resultado final de las recomendaciones. Mediante la aplicación de las pruebas de integración y aceptación, se pudo comprobar la correcta integración y funcionamiento de los requisitos del sistema.

CONCLUSIONES

El desarrollo del módulo de recomendación de problemas para el COJ posibilitó arribar a las siguientes conclusiones:

- La realización de un estudio del estado del arte de los sistemas de recomendación y de los jueces en línea permite afirmar que los jueces en línea de programación constituyen un área dentro del aprendizaje-electrónico que demanda la incorporación de herramientas recomendadoras con vistas a mejorar el aprovechamiento de sus usuarios.
- A través del estudio de técnicas algorítmicas que se emplean en los sistemas de recomendación, se determinó que el filtrado colaborativo basado en vecinos más cercanos presenta una gran eficiencia y efectividad en entornos de aprendizaje.
- Se determinó que las interfaces de explicación de las recomendaciones, son necesarias en los sistemas de recomendación y que proveen a los usuarios mayor confianza en las sugerencias realizadas por el sistema.
- La aplicación de métricas de validación al algoritmo de recomendación implementado, demostró que el mismo ofrece recomendaciones de alta precisión.
- La incorporación de técnicas de agrupamiento en los sistemas de recomendación reduce los efectos del problema de la escalabilidad y aumenta el rendimiento de los mismos.
- Como producto final se obtuvo un módulo de recomendación de problemas para el COJ con mayor precisión que el anterior, que es capaz de sugerirle a los usuarios los problemas a resolver, y explicar mediante interfaces las razones de las recomendaciones.

RECOMENDACIONES

- Adicionar técnicas de recomendación basadas en contenido al sistema desarrollado, para además lograr recomendaciones enfocadas a las preferencias individuales de cada usuario.
- Aunque los resultados del algoritmo de agrupamiento son positivos, la propuesta es un acercamiento inicial, se recomienda aplicar técnicas más avanzadas de agrupamiento.
- Desarrollar un experimento para evaluar el impacto de las técnicas de arranque en frío propuestas, en la eficacia de la recomendación.

BIBLIOGRAFÍA

A generalized taxonomy of explanations styles for traditional and social recommender systems. Papadimitriou, Alexis, Symeonidis, Panagiotis and Manolopoulos, Yannis. 2012. 3, 2012, Data Mining and Knowledge Discovery, Vol. 24, pp. 555-583.

A K-means-like Algorithm for K-medoids Clustering and Its Performance. Park, Hae-Sang , Lee, Jong-Seok and Jun, Chi-Hyuck . 2006. 2006. Proceedings of ICCIE. pp. 102-117.

A re-examination of text categorization methods. Yang, Yiming and Liu, Xin . 1999. s.l. : ACM, 1999. Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval. pp. 42-49.

A Survey of Accuracy Evaluation Metrics of Recommendation Tasks. Gunawardana, Asela and Shani, Guy. 2009. [ed.] Lyle Ungar. 2009, Journal of Machine Learning Research, Vol. 10, pp. 2935-2962.

Albin, A. 2009. *SISTEMA DE RECOMENDACIÓN COLABORATIVO BASADO EN ALGORITMOS DE FILTRADO MEJORADOS.* 2009. p. 11.

Amatriain, Xavier, et al. 2011. *Data mining methods for recommender systems.* New York : Springer, 2011. pp. 39-71.

An adaptive recommendation system without explicit acquisition of user relevance feedback. Shahabi, C and Chen, Y S. 2003. 2003. Distributed and Parallel Databases. Vol. 14(2), pp. 173-192.

An e-Learning Collaborative Filtering Approach to Suggest Problems to Solve in Programming Online Judges. Toledo, Raciel Yera and Mota, Yailé Caballero. 2014. 2014, International Journal of Distance Education Technologies (IJDET).

An Open Architecture for Collaborative Filtering of Netnew. Resnick, Paul. 1994. 1994. Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work.

Analysis of Recommender Systems' Algorithms. Konstantinos, Margaritis and Vozalis, Emmanouil . 2003. Atenas : s.n., 2003. The 6th Hellenic European Conference on Computer Mathematics & its Applications.

Balabanovic, M and Shoham, Y. 1997. *Content-based, collaborative recomendation on Comunications of ACM 40.* 1997.

Ben Schafer, J, et al. 2007. Collaborative Filtering Recommender Systems. [book auth.] P Brusilovsky, A Kobsa and W Nejdl. *The adaptive web: Methods and Strategies of Web Personalization.* s.l. : Springer, Heidelberg, 2007, Vol. 4321, pp. 291-324.

Bilgic, M and Mooney, R. 2005. Explaining recommendations: satisfaction vs. promotion. *Proceedings recommender systems workshop (IUI conference).* 2005.

Bock, H H. 2007. Clustering methods: a history of k-means algorithms. *Selected contributions in data analysis and classification.* s.l. : Springer Berlin Heidelberg, 2007, pp. 161-172.

Burke, R. 2002. User modeling and user-adapted interaction. *In Hybrid recommender systems: Survey and experiments.* 2002, págs. 331-370.

Cabrera, Gualberto and Heredia, Luis José. 2010. Fase de Planificación. *Portal del Centro de Informática Médica (CESIM).* 2010, pp. 40-43.

Cleger-Tamayo, Sergio, et al. 2012. *Propuesta de Interfaz de Explicación para la recomendacion de noticias.* Granada : s.n., 2012.

COJ. 2010. COJ. [En línea] 5 de Junio de 2010. [Citado el: 21 de Octubre de 2014.] http://coj.uci.cu/general/about.xhtml.

Degiovannini, Marcio. 2007. Comparativa de Frameworks WEB. [Online] Febrero 5, 2007. [Cited: Febrero 22, 2010.] http://www.javahispano.org/contenidos/es/comparativa_de_frameworks_web/.

Ecured. 2014. Servidores Web. [Online] 2014. [Cited: Febrero 25, 2014.] http://www.EcuRed.cu/index.php/Servidores_Web.

Escribano, Gerardo Fernández. 2002. *Introducción a Extreme Programming.* 2002.

Evaluating Collaborative Filtering Recommender Systems. Herlocker, Jonathan L, et al. 2004. 1, 2004, ACM Transactions on Information Systems (TOIS), Vol. 22, pp. 5-53.

Fernández, Gerardo. 2002. *Introducción a Extreme Programming.* 2002.

Fowler, Martin. 2004. Inversion of Control Containers and the Dependency Injection pattern. [Online] Enero 23, 2004. [Cited: Febrero 22, 2010.] http://martinfowler.com/articles/injection.html.

Fussell, Mark. 2007. *Foundations of Object-Relational Mapping.* 2007.

Galán , Nieto and Sergio, Manuel. 2007. *Filtrado colaborativo y sistemas de recomendación.* Madrid : s.n., 2007.

García de Jalón, Javier. 200. *Aprenda Java como si estuviera en primero.* San Sebastián : s.n., 200.

García Salcines, Enrique and Romero Morales, Cristóbal. 2008. *Sistema recomendador colaborativo usando minería de datos distribuidas para la mejora continua de cursos e-learning, IEEE-RITA.* 2008. pp. 19-30.

Gómez, A.C. 2011. XHTML. [Online] 2011. [Cited: Marzo 20, 2014.] http://www.ecured.cu/index.php/XHTML.

Harmon, James. 2008. Using the Dojo JavaScript Library to Build Ajax Applications. *Addison Wesley.* 2008.

Hechavarria, R.G. 2011. CSS. [Online] 2011. [Cited: Marzo 20, 2014.] http://www.ecured.cu/index.php/CSS.

Heckerman, Breese and Cadie. 1998. *Empirical Analysis of Predictive Algorithms for Collaborative Filtering.* 1998.

Herlocker, Jonathan L, Konstan, Joseph A and Riedl, John . 2000. *Explaining Collaborative Filtering Recommendation.* 2000. p. 2.

Item-based collaborative filtering recommendation algorithms. Sarwar, B, et al. 2001. s.l. : ACM, 2001. Proceedings of the 10th international conference on World Wide Web. pp. 285-295.

Jhonson, Rod. 2002. *Expert One-To-One: J2EE Design and Development.* 2002.

Joskowicz, José. 2008. *Reglas y prácticas en eXtreme Programming.* s.l. : Universidad de Vigo, 2008. pp. 4-5.

King, Gaving. 2005. *Hibernate in Action.* 2005.

K-means v/s K-medoids: A Comparative Study. Singh, Shalini S and Chauhan, N C. 2011. 2011. National Conference on Recent Trends in Engineering & Technology. Vol. 13.

Larman, Craig. 1999. *UML y Patrones.* 1999.

Linden, Greg, Smith, Brent and York, Jeremy. 2003. *Amazon.com Recomendations: Item-to-Item Collaborative Filtering, IEEE Internet Computing.* 2003.

Lobaina, J.C and Roque, J.L. 2012. Desarrollo de la versión 2 del Juez en Línea Caribeño. [Online] 2012. [Cited: Diciembre 5, 2013.] http://bibliodoc.uci.cu/RDigitales/2010/octubre/31/TD_05129_12.pdf.

Lopez, Vreixo Formoso. 2013. Filtrado Colaborativo: Principales ventajas y limitaciones. *Técnicas eficientes para la recomendacion de productos basadas en filtrado colaborativo.* Coruña : s.n., 2013, pp. 40-43.

O´Regan, Graham. 2004. Introduction to Aspect- Oriented Programming. [Online] Enero 14, 2004. [Cited: Febrero 22, 2010.] http://onjava.com/pub/a/onjava/2004/01/14/aop.html.

OMG. 2007. *OMG Unified Modeling Language (OMG UML),.* 2007.

Otero, Abraham. 2007. MySQL vs PostgreSQL. ¿Cuándo emplear cada una de ellas? [Online] Septiembre 10, 2007. [Cited: Febrero 22, 2010.] http://www.javahispano.org/contenidos/es/mysql_vs_postgre_cuando_emplear_cada_una_de_ella_11/.

Peis, E, Morales-del-Castillo, J. M and Delgado-López, J. A. 2012. Sistema de recomendación Semánticos. Un análisis del estado de la cuestión - Hipertexto - (UPF). [Online] 2012. [Cited: Enero 3, 2015.] http://www.upf.edu/hipertexnet/numero-6/recomendacion.html#Criterios.

Pressman, R S. 2005. Ingeniería del Software Un enfoque práctico: Sexta edición. [Online] 2005. [Cited: Enero 25, 2015.] http://unpocodejava.wordpress.com/2013/05/09/tecnicas-para-la-captura-de-requisitos/.

Pullés, Y.D. 2003. Herramienta CASE. [Online] 2003. [Cited: Marzo 20, 2014.] http://www.sisman.utm.edu.ec/libros/FACULTAD%20DE%20CIENCIAS%20INFORM%C3%81TICAS/CARRERA%20DE%20INGENIER%C3%8DA%20DE%20SISTEMAS%20INFORMATICOS/09/TECNICAS%20DE%20IV%20GENERACION/Libro.pdf..

Rashid, A M, et al. 2002. Getting to know you: learning new user preferences in recommender systems. *Proceedings of the 7th international conference on Intelligent user interfaces.* s.l. : ACM, 2002, pp. 127-134.

Resnick, P y Varian, R. 1997. Recommender systems. *In comunications of de ACM.* 1997, págs. 56-58.

Ricci, F, y otros. 2010. *Recommender System Handbook.* 2010.

Schafer, J Ben, et al. Comparing Collaborative Filtering to Content-Based Filtering. *Collaborative filtering recommender systems.* pp. 300-301.

Seagaran, Toby. 2005. *Programming collective intelligence.* s.l. : O'Reilly, 2005.

Silberschatz, A, Sudarshan, S and Korth, H F. 2007. *Fundamentos de diseño de bases de datos.* 2007.

Smith, Benji. 2005. Why I hate Framework. [Online] Septiembre 30, 2005. [Cited: Febrero 22, 2010.] http://discuss.joelonsoftware.com/default.asp?joel.3.219431.12.

Swearingen, Kirsten and Sinha, Rashmi. 2001. *Beyond Algorithms: AN HCI Perspective on Recommender Systems. ACM SIGIR 2001 Workshop on Recommender Systems.* 2001.

Tan, Huiyi, Guo, Junfei and Li, Yong. 2008. E-Learning Recommendation System. *IEEE International Conference on Computer Science and Software Engineering.* 2008, pp. 430-433.

Tang, Y T and McCalla, G. 2003. Smart recommendation for an evolving e-learning system. *Workshop on Technologies for Electronic Documents for Supporting Learning.* s.l. : AIED, 2003.

Toledo, Raciel Yera. 2010. *Concepción y desarrollo de un sistema de recomendación para jurados online de programación.* 2010. p. 19.

UML. 2011. Tutorial- UML Package Diagram. [Online] 2011. http://www.visual-paradigm.com/product/vpuml/tutorials/packagediagram.jsp..

UMLy BPMN. 2014. UML, BPMN and Enterprise Architecture Tool for Sofware Development. [Online] 2014. [Cited: Febrero 25, 2014.] http://www.visual-paradigm.com/.

Velez Lang, O and Santos, Carlos. 2006. *Sistemas recomendadores: Un enfoque desde los algoritmos genéticos.* s.l. : Industrial data, 2006.

Wijmo. 2014. Qué es wijmo. *Wijmo.* [Online] 2014. [Cited: Enero 15, 2015.] www.wijmo.com.

Williams, Rob. 2008. JSF: The Good, The Bad, and Yes, The Ugly. [Online] Agosto 25, 2008. [Cited: Febrero 22, 2010.] http://java.dzone.com/articles/jsf-the-good-bad-and-yes-ugly.

Yera, Raciel. 2010. *RECJUDGE: sistema de recomendaciones para jurados online de programación.* 2010. p. 3.