Universidad de las Ciencias Informáticas
Facultad 8

Título:

Concepción y desarrollo de un sistema de recomendación para jurados online de programación

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

Autor: Raciel Yera Toledo
Tutores: Lic. Tomás Orlando Junco Vázquez
Lic. Héctor Matías González
Co-tutor: Ing. Jorge Amado Soria Ramírez

Ciudad de La Habana, Cuba, junio 2010
"Año 52 de la Revolución"

Declaración de autoría

Declaro que soy el único autor del trabajo “Concepción y desarrollo de un sistema de recomendación para jurados online de programación” y autorizo a la Facultad 8 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

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

Autor:

_____

Raciel Yera Toledo

Tutor:

_____

Lic. Tomás O. Junco Vázquez

_____

Lic. Héctor Matías González

Pensamiento

El único modo de hacer un gran trabajo es amar lo que haces. Si no lo has encontrado todavía, sigue buscando. No te acomodes. Como con todo lo que es propio del corazón, lo sabrás cuando lo encuentres.

Steve Jobs

Pero realmente vivía apresado por un sueño. ¿Posibilidades de realizarlo? Muy remotas. Mas, ¡pobre del que no sueñe y luche! Mi sueño era estudiar.

Raimundo Reguera

Milito en el bando de los impacientes, milito en el bando de los apurados, de los que siempre presionan para que las cosas se hagan y de los que muchas veces tratan de hacer más de lo que se puede.

Fidel Castro

Dedicatoria

A mis padres Lorenza y Cristóbal y a mi hermano Rangel, por todo lo que me han dado. Con ellos he adquirido una deuda moral que me resulta imposible de pagar.

A Edianny.

Agradecimientos

Resulta muy difícil completar esta sección tras cinco años de estudio y duro bregar. Muchas son las personas cuyo apoyo ha sido imprescindible a lo largo de esta etapa.

Agradecer ante todo a la Revolución, a la UCI y a Fidel: ellos son los principales responsables de que hoy, al igual que muchos otros compañeros, me pueda estar graduando de ingeniero.

A mi familia, tanto los más cercanos como los más lejanos: todos han puesto su granito de arena con vistas a que hoy pueda culminar mi carrera.

A Edianny, por su amor, comprensión y cariño; sentimientos que día a día humildemente trato de brindarle con la misma intensidad con la que ella me los brinda a mí.

A mis compañeros, los de los grupos 8108 y 8501. Ellos me han demostrado que no se le puede tener miedo a nada en la vida, y que inclusive en los peores momentos, hay que tener confianza y mantener un espíritu valiente y emprendedor. Gracias, amigos.

Agradezco además, a los excelentes profesores que he tenido en estos cinco años, los que siempre me han brindado incondicional apoyo en los momentos difíciles en los que muchas veces me he encontrado.

A todos los estudiantes y profesores de la UCI de una forma u otra han estado vinculados al movimiento de los concursos de programación. Ellos han sido la savia que le ha dado vida a este trabajo, y, personalmente para mí, constituyen y constituirán siempre el mejor paradigma de superación personal y profesional.

A los miembros del Consejo Científico Estudiantil y del Consejo FEU con los que he podido trabajar a lo largo de estos años. Ellos, excelentes amigos que siguen las filosofías expuestas por el gran matemático cubano Raimundo Reguera y por Nuestro Fidel en los pensamientos que se muestran al principio del trabajo, han constituido una escuela para mí, y sin ellos, mis estudios de ingeniería estuvieran todavía incompletos.

Gracias a todos los compañeros que me han dado el privilegio de, desde mi función de alumno ayudante, aportar en algo a su formación. La lista es interminable, pero merecen destacarse los grupos 8105 y 8107, que se convertirían posteriormente en el 8201, donde tengo a varios de mis mejores amigos; ellos saben quiénes son. Algunos son de los que inclusive te llaman al

laboratorio y te dicen que vayas en ese mismo momento a ir a comer espaguetis a su apartamento. Otros son de los que pasan a verte a la 1 de la mañana al laboratorio, notan que tienes hambre, y te dejan todos los panes que habían comprado ellos para llevarse para su casa. Ellos saben quiénes son.

A la gente de Morón y de Ciego, la gente que viene conmigo desde hace 8 años, la gente del IPVCE, los que siempre van a estar conmigo en las buenas y en las malas. Especialmente, a la gente de 10mo A, ellos saben quiénes son. Específicamente la gente de Chambas de 10mo A. Ella sabe quién es.

Para terminar, agradecer a mis tutores, ellos han sido el motor impulsor de este trabajo. A Tomás, por creer en mí cuando nadie más creía y por cargar conmigo durante estos años; a Héctor, por mostrarme la ruta a seguir; y a Amado, por hacerme ver qué era lo que de verdad yo quería. Agradecer a Enrique, persona de mi total confianza cuyas sugerencias fueron claves durante el desarrollo del trabajo, y estoy seguro que seguirán siendo claves.

A todos, muchas gracias.

Resumen

Los sistemas de recomendación son aplicaciones encargadas de realizar sugerencias a los usuarios basándose en la preferencia de estos por un grupo de elementos con características específicas. Tras analizar el dominio actual de los jurados online de programación, herramientas de e-learning que en los últimos tiempos han adquirido gran importancia dentro de la disciplina, se concibió un sistema de recomendación enfocado a orientar a los estudiantes con respecto a qué ejercicios resolver como parte del jurado online. Con este fin, se propuso un perfil algorítmico tomando lo mejor de toda la teoría definida alrededor del problema general de la recomendación. Se realizó el análisis, diseño e implementación de una aplicación para llevar a la práctica el perfil algorítmico definido, incorporándosele además otras funcionalidades con características recomendadoras que de manera directa o indirecta también tributan al aprovechamiento de los servicios del jurado online. Finalmente, se realizó la validación de los resultados obtenidos, tanto del perfil algorítmico como del software, verificándose el cumplimiento de los objetivos propuestos.

Palabras claves: ejercicio, jurado online, sistema de recomendación.

Índice

Introducción

La enseñanza de la programación indudablemente constituye eje central en todas las carreras de computación. A través de esta, los estudiantes adquieren habilidades que posteriormente le van a ser imprescindibles en su desempeño profesional. Contradictoriamente, a nivel regional cada año se han ido acentuando en los alumnos de nuevo ingreso algunos rasgos negativos relacionados con el dominio de esta. Entre estos están la carencia de habilidades a la hora de programar aun habiendo vencido parte de esta materia; el desconocimiento total de los paradigmas directores de la programación, y la falta de conciencia acerca de la necesidad de seguir una disciplina a la hora de escribir código.

Una de las estrategias asumidas para combatir esta desmotivación en los estudiantes ha sido el desarrollo de los concursos de programación. Engendrados en los albores de la década del 70 por el actual director general de la Competencia Internacional Inter-colegios de Programación auspiciada por la ACM (ACM-ICPC), William B. Poucher, en la Universidad de Baylor, han constituido a lo largo de estos años una fuente inagotable de conocimientos que, basándose en la filosofía del aprendizaje competitivo, han favorecido la formación de mejores profesionales de la teoría y la práctica en el mundo de la computación.

En este tipo de eventos, comúnmente se tiene que resolver un conjunto de problemas complejos donde se han de aplicar conocimientos que van desde el diseño y análisis de algoritmos, hasta las más complejas teorías matemáticas, desarrollándose a través de estos, habilidades como el pensamiento algorítmico, la resolución de problemas, el trabajo en equipo, y la laboriosidad.

Con vistas a dar soporte de nuevo tipo a la enseñanza de la programación y a la preparación para este tipo de concursos, se han concebido una serie de herramientas, entre las que sobresalen los jurados online. Estos son aplicaciones web disponibles a tiempo completo que automatizan el proceso de evaluación de soluciones a problemas de programación, y facilitan la presentación de los resultados de dicho proceso. Actualmente en la UCI se cuenta con dos de estas aplicaciones, con más de 700 problemas disponibles, los cuales han tenido más de 130 000 intentos de solución, siendo exitosos alrededor del 34% de estos. Por otra parte, cerca de 6 000 usuarios al menos han intentado darle solución a algún problema.

A pesar de este logro se puede afirmar que, amén de la presencia de esta infraestructura, existe una insuficiente disponibilidad de herramientas informáticas de apoyo a la enseñanza de la programación enfocadas en la adquisición progresiva del conocimiento como parte de estos jurados online. En muchas ocasiones los estudiantes, al interactuar con este tipo de aplicaciones, se pierden en un mar de teoría y de ejercicios generalmente no esenciales ni ordenados lógicamente, lo que deviene en resultados no acordes a su dedicación y tiempo de estudio.

Considerando la anterior situación problémica, unida a la necesidad de reforzar de manera directa la preparación para concursos de programación dada la perspectiva que se ha planteado la UCI, y junto a ella Cuba, de convertirse en abanderada de estos a nivel universitario dentro del área y fuera de esta, lo que incluye una posible candidatura de La Habana para celebrar la Final Mundial de la ACM-ICPC en el período 2015-2020, se presenta el siguiente **problema a resolver:***¿Cómo lograr un mejor aprovechamiento, por parte de los estudiantes, de los recursos disponibles en los jurados online de programación?*

Con el fin de dar respuesta a este problema, se seleccionaron como **objeto de estudio** las *herramientas recomendadoras en el aprendizaje asistido por computadoras.*

El **objetivo general** de la investigación es *desarrollar un sistema de recomendación para ofrecer sugerencias a los usuarios de un jurado online referentes a la trayectoria a seguir, mejorando así el aprovechamiento, por parte de los estudiantes, de los recursos disponibles en los jurados online de programación.*

En la investigación se concibió como **campo de acción** a los *sistemas de recomendación de apoyo a la enseñanza de la programación, a través de los jurados online.*

Como objetivos específicos se definieron los siguientes:

1. Realizar un estudio del estado del arte de los sistemas de recomendación haciendo hincapié en la evolución histórica y teórica, su aplicabilidad al e-learning y a los jurados online, y la metodología y herramientas a utilizar en su desarrollo.
2. Elaborar un compendio de las técnicas más factibles para el desarrollo de los sistemas de recomendación aplicados a los jurados online. Definir el perfil algorítmico de la solución.
3. Analizar, diseñar e implementar una aplicación informática como sistema de recomendación para jurados online de programación.

4. Validar el perfil algorítmico y la aplicación recomendadora construida.

La **idea a defender** se basa en que *la obtención de un sistema de recomendación aplicable a jurados online de programación permitirá orientar a los estudiantes y profesores respecto a los ejercicios a resolver, durante el proceso de enseñanza-aprendizaje de la programación,* teniendo como resultado esperado un sistema de recomendación integrable con un jurado online de programación que facilite la progresión del estudiante que a la vez es usuario de este.

Para la realización de esta investigación se utilizaron métodos teóricos, que permitieron reproducir teóricamente el objeto, en el pensamiento, en toda su objetividad y concreción, así como comprenderlo en su desarrollo, historia y lógica. Los métodos empleados fueron:

1. Analítico-sintético: Al analizar toda la teoría y documentos que permiten la extracción de los elementos más importantes relacionados con las herramientas recomendadoras de apoyo a la enseñanza de la programación.
2. Inductivo-deductivo: Al determinar, de entre todas las características de los sistemas de recomendación, aquellas que son aplicables al objeto de estudio.
3. Histórico-lógico: Al tomar la caracterización de la evolución histórica de los sistemas de recomendación como una de las herramientas para concebir el sistema actual.
4. Modelación: Como forma de representación de todos los datos inherentes al campo de acción.

El uso de los métodos de investigación empírica, por otro lado, conllevó a una serie de procedimientos prácticos con el objeto y los medios de investigación, permitiendo revelar las características fundamentales y relaciones esenciales del primero. Se empleó:

1. La observación: Al permitir observar el fenómeno en su manifestación externa.

Las **tareas de investigación** definidas para dar cumplimiento a los objetivos específicos fueron las siguientes:

1- Realizar un estudio de las principales herramientas recomendadoras desarrolladas en apoyo a la enseñanza de la programación.
2- Identificar el alcance, los propósitos y las tendencias actuales en los sistemas de recomendación concebidos para el e-learning.

3- Determinar puntos comunes entre el dominio de los sistemas de recomendación concebidos para el e-learning y los jurados online de programación.
4- Determinar las técnicas que más se adecuen al dominio de un sistema recomendador para jurados online de programación.
5- Definir la metodología de desarrollo de software a utilizar adecuándose a las particularidades del problema a resolver y del propósito del producto final.
6- Definir la plataforma y principales herramientas sobre las que la aplicación será construida.
7- Realizar el estudio de las técnicas algorítmicas y/o de inteligencia artificial referenciadas como más efectivas para el desarrollo de los sistemas de recomendación.
8- Establecer un modelo de negocio o de dominio para el desarrollo en cuestión.
9- Realizar el diseño e implementación del sistema de recomendación.
10- Validar el sistema de recomendación a través de su integración con el jurado online de programación de la Iniciativa Xtreme, y con el uso de técnicas referenciadas por la bibliografía para el área de los sistemas de recomendación.

El documento está organizado en cuatro capítulos, de la siguiente forma:

Capítulo 1: Se hace un análisis de los principales conceptos relacionados con el dominio. Además, se realiza un estudio del estado del arte de las herramientas recomendadoras orientadas a jurados online, y de los sistemas de recomendación en el amplio sentido de la palabra. Se realiza un estudio de las distintas técnicas, tecnologías y metodologías a utilizar.

Capítulo 2: Se hace una breve descripción de la propuesta, y se realiza un estudio de los métodos algorítmicos utilizados en el dominio, adaptando un subconjunto de estos al campo del problema actual.

Capítulo 3: Se desarrolla la solución propuesta, pasando por las etapas de análisis, diseño e implementación de RUP.

Capítulo 4: En este capítulo se valida el resultado obtenido, utilizando técnicas referenciadas que se enfocan en verificar la efectividad de aplicaciones afines a la que se propone.

Fundamentación teórica

1.1 Introducción

El objetivo de este capítulo es realizar un análisis de los principales conceptos relacionados con el dominio de los sistemas de recomendación. Además, se lleva a cabo un estudio del estado del arte de las aplicaciones recomendadoras orientadas a jurados online, y de los sistemas recomendadores en el amplio sentido de la palabra. Finalmente, se hace una breve descripción de las herramientas, tendencias y tecnologías a utilizar en el desarrollo de la propuesta.

1.2 Generalidades de los sistemas de recomendación

Los sistemas de recomendación utilizan la opinión de los miembros de una comunidad para ayudar a los individuos a identificar la información o los productos más relevantes o interesantes según sus necesidades actuales (1). Constituyen una técnica de filtrado de información, la cual presenta distintos tipos de temas o ítems de información al usuario, basándose en la predicción del "ranking" o ponderación que este le daría a un ítem que el sistema aún no ha considerado; mediando, automatizando o soportando de esta forma, el proceso de realizar recomendaciones (2).

Estos sistemas comienzan a emerger a mediados de los años 90 (3) (4), como solución ante el desbordamiento de información que desde aquel entonces ya estaban ocasionando a nivel global las nuevas tecnologías de la información y las comunicaciones (TICs) y en particular la World Wide Web (www), haciendo difícil para una persona obtener la mejor información dentro

de este cúmulo, al no tener esta, en muchos casos, detalles de cada una de las alternativas a seleccionar.

La amplia utilización y aceptación de este nuevo paradigma ha devenido en que se pueda ya afirmar que se esté abandonando la época de la información y se esté iniciando la época de la recomendación (5).

1.2.1 El problema de la recomendación

El principio de funcionamiento de los sistemas de recomendación está resumido en el problema de la recomendación (6), el cual puede ser formulado de la siguiente forma:

Sea *C* un conjunto de usuarios, y *S* el conjunto de todos los ítems posibles a ser recomendados, tales como libros, películas o restaurantes, pudiendo ser bien grande la cardinalidad de ambos conjuntos. Sea además u una función que mide la utilidad del ítem *s* al usuario *c*, o sea *u*: *C x S* → *R*, donde *R* es un orden total (enteros no negativos o números reales en un determinado rango). Con estos elementos, se desea, para cada usuario $c \in C$, aquel ítem $s' \in S$ que maximice la utilidad al usuario. Más formalmente:

$$\forall c \in C, \;\; s'_c = argmax_{s \in S} u(c, s)$$

1.2.2 El modelo del proceso de recomendación

Dentro de la concepción de sistemas de recomendación, ocupan un papel central el buscador de recomendaciones, el proveedor de preferencias y el sistema recomendador propiamente dicho. El buscador de recomendaciones (generalmente un individuo) puede solicitar sugerencias al sistema recomendador, o este puede automáticamente ofrecerlas sin una solicitud previa. Asimismo, para la construcción de las recomendaciones, el sistema puede capturar de manera directa las preferencias específicas de los individuos, o puede inferirlas a través de preguntas indirectas. Una vez recopilados todos los datos sobre los usuarios, el sistema está listo para recomendar los ítems que el usuario probablemente prefiera, basándose en su perfil, el perfil de otros usuarios, los datos del proveedor de las posibles preferencias, entre otros criterios a tomar en cuenta, siendo utilizadas estas recomendaciones para seleccionar ítems del universo de alternativas. En la práctica, a la hora de concebir un sistema

recomendador, muchas veces no se instancian algunas partes de este modelo, dependiendo esto de los requerimientos y de los objetivos a cumplir.

La figura 1 resume los conceptos y situaciones que suelen presentarse en el modelo general de la recomendación.

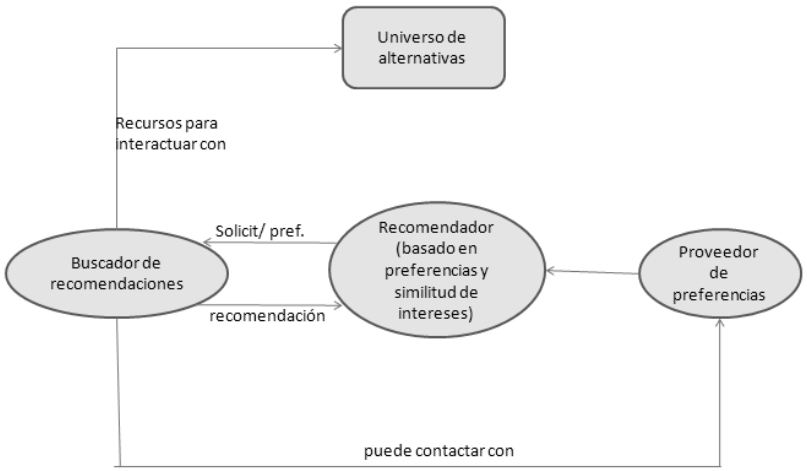


Figura 1. Modelo general del proceso de recomendación

En la concepción y diseño de un sistema recomendador suelen tenerse en cuenta cuatro aspectos fundamentales. Estos son la forma de representar las preferencias (7), la manipulación de los roles dentro del sistema y la comunicación usuario-usuario (8), los algoritmos para manipular las recomendaciones (9) y la forma de interacción usuario-máquina a la hora de representar las recomendaciones (10).

1.3 Clasificación de los sistemas de recomendación

Considerando los aspectos establecidos anteriormente, los sistemas de recomendación han recibido diferentes clasificaciones. A continuación se expone la clasificación más aceptada y reconocida por materiales de referencia.

1.3.1 Recomendadores basados en el 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 (11). Estos tienen en cuenta la filosofía de “muéstrame las que

cosas que he seleccionado, o que me han gustado anteriormente", aprendiendo de esta forma a través de la retroalimentación del individuo con el sistema. Esta retroalimentación puede hacerse de manera explícita, preguntándole directamente a este usuario su criterio acerca de determinado ítem, o de manera indirecta, considerando criterios tales como los sitios frecuentados por el usuario, el tiempo que se demora dentro de una página en específico, y otros que de cierta manera puedan también informar de igual forma las preferencias.

Este tipo de sistemas suele ser utilizado en una gran variedad de dominios que oscilan entre la recomendación de páginas web, artículos noticiosos, restaurantes, programas de televisión, y artículos para vender. A pesar de diferir cada una de estas aplicaciones específicas, todas tienen en común el hecho de contener un modelo que describe los ítems que pueden ser recomendados, un modo de crear un perfil de usuario que describirá los tipos de ítems que él prefiere y un modo de comparar cada uno de los ítems a recomendar con el perfil del usuario, para determinar cuáles de ellos serán recomendados.

Entre los sistemas de recomendación que utilizan de manera exclusiva este paradigma se puede mencionar los referenciados en (12) y en (13).

1.3.2. Sistemas de soporte a la recomendación

Los sistemas de soporte a la recomendación automatizan el proceso de recomendación, sin encargarse de representar preferencias ni de computar recomendaciones. Como su nombre lo indica, estas son herramientas que se dedican a brindar soporte a la actividad de compartir recomendaciones, incluyendo los procesos de construcción y búsqueda de estas, quedando agrupados sus usuarios en dos roles fundamentales: el rol de productor de recomendaciones, y el rol de consumidor de recomendaciones.

A través de este tipo de sistema, el productor, generalmente representado por un pequeño grupo de personas altamente motivado por brindar recomendaciones, introduce datos, preferencias, etc., siendo la aplicación la encargada de computarlas y de posteriormente presentarlas al consumidor de recomendaciones en un formato semejante al formato de entrada.

Los sistemas de soporte a la recomendación son efectivos cuando hay suficientes personas dedicadas a la actividad de recopilación de información con el fin de crear recomendaciones,

teniendo además una correcta implementación de la retroalimentación consumidor-productor, la que determinará la eficacia de las sugerencias realizadas.

Uno de los primeros sistemas en implementar esta enfoque fue Tapestry (14), que era un sistema de mensajes electrónicos que permitía a los usuarios evaluar a los mensajes de bien o mal, o asociarle una anotación con un nuevo mensaje. Tapestry es considerado como uno de los primeros sistemas de recomendación. Actualmente se continúa el desarrollo con base en este tipo de sistemas, pudiendo citarse aplicaciones en internet que lo implementan, como about.com, así como autores que siguen investigando sobre esta línea, como (15).

1.3.3 Sistemas basados en filtrado colaborativo

El filtrado colaborativo abarca técnicas orientadas a identificar personas con similares intereses para, basándose en esto, posteriormente construir las recomendaciones. La exitosa implementación de este enfoque se basa en tres condicionales fundamentales que son: la amplia participación de personas dentro del sistema haciendo fácil para una de ellas el encuentro de otras con intereses similares, la existencia de una forma sencilla de representar los intereses dentro del sistema, y la existencia de algoritmos capaces de relacionar eficientemente las personas correspondientes.

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 perfil de los usuarios anteriores, construyendo una lista de usuarios semejantes, más conocida como usuarios más cercanos. Combinando esta lista, se devuelve un listado de recomendaciones conteniendo los ítems mejor evaluados por estos usuarios y que no han sido aún evaluados por el nuevo individuo.

Entre las principales fortalezas de este enfoque está la posibilidad de una total personalización de las recomendaciones y la simplicidad de la representación desde el punto de vista computacional. Unido a esto, como fortaleza también es de remarcar la existencia de una uniformidad en cuanto a la instanciación de los roles. A diferencia de otras filosofías, en el filtrado colaborativo el constructor de recomendaciones y el consumidor de estas está representado por la misma persona, puesto que mientras más recomendaciones o

evaluaciones realice el usuario, más exacta será la representación de su perfil y por tanto se incrementará la exactitud de las recomendaciones que este reciba de parte del sistema.

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 (16), importante portal de ventas por internet, y GroupLens (3), la que constituye el principal referente arquitectónico y algorítmico dentro de este tipo de sistemas de recomendación.

1.3.4 Sistemas de recomendación híbridos

Los enfoques básicos de la recomendación, anteriormente expuestos, han sido utilizados de manera exitosa en disímiles campos y escenarios. Pese a su difusión, de forma general cada uno de ellos presenta varias desventajas relacionadas principalmente con la necesidad de un gran número de usuarios adheridos al sistema para un correcto funcionamiento del mismo y con la tendencia a una sobre-especialización que va aumentando a medida que aumenta la utilización de la aplicación por parte del usuario.

Con vista a mitigar estas desventajas, una opción muy utilizada es la de combinar la recomendación basada en contenido con el filtrado colaborativo, conformando un sistema de recomendación con características híbridas. A través de estos, la construcción de preferencias se basa tanto en los intereses del usuario, como en los intereses de los usuarios similares, coincidiendo así con dos de las técnicas específicas antes expuestas.

Varios autores han demostrado cómo a través de este enfoque la calidad de las recomendaciones aumenta considerablemente (17), constituyendo de esta forma una alternativa a tener en cuenta a la hora de diseñar un sistema de este tipo.

1.4 Aplicaciones de los sistemas de recomendación

Los sistemas de recomendación han encontrado aplicación en disímiles áreas de la vida cotidiana; es difícil encontrar en estos momentos algún escenario donde la presencia de un sistema de recomendación no se traduzca en mejoras para las personas implicadas en este.

Dentro de todas las aplicaciones de estos sistemas, es de remarcar su amplia y estratégica utilización en el marco del e-Commerce y del e-Learning.

1.4.1 E-Commerce y sistemas de recomendación

En el altamente competitivo mercado de hoy en día, es de vital importancia para las grandes compañías conocer quiénes son sus clientes, cuáles son sus preferencias y cómo evalúan los productos existentes, teniéndose claro que trabajar en base al criterio de los usuarios finales está constituyendo una de las llaves del éxito en muchas empresas con presencia en internet. La manera más efectiva de obtener esta información es a través del estudio detallado del usuario, sin embargo, realizar este es altamente costoso e implica la realización de una planificación cuidadosa.

Con el advenimiento de la web 2.0 muchos sitios web de comercio electrónico han comenzado a retroalimentarse de sus usuarios de manera directa, al permitirle compartir opiniones y evaluar de manera cuantitativa los productos que compran. Esta información recopilada es de un alto valor para la creación de nuevas aplicaciones que perfeccionen la proyección de la empresa o negocio en cuestión, aplicaciones entre las cuales se encuentran sin dudas los sistemas de recomendación.

Uno de los ejemplos más clásicos de la aplicación de los sistemas de recomendación en este campo lo constituye amazon.com, sitio de venta de artículos ya mencionado anteriormente, cuya sección de venta de libros contiene implícitas varias funcionalidades basadas en esta filosofía. Entre estas cabe destacar "*Customers who bought*", la que se encuentra en la página de información de cada libro dentro del catálogo y que contiene dos listas de recomendaciones una con los libros frecuentemente comprados por aquellos clientes que compraron el libro en cuestión y otra con los autores frecuentemente leídos por estos mismos clientes.

Como otras aplicaciones populares en internet que implementan sistemas recomendadores pueden citarse (18) a CDNOW, portal concebido para la venta de CDs de música; eBay, considerado como uno de los mayores portales de compra-venta del mundo; Levis, creada con el objetivo de dar soporte a los clientes de la ropa de la marca homónima; y MovieFinder.com, capaz de localizar filmes con similares características a un filme dado. Todos estos sitios tienen en común el alto nivel de aceptación que tienen por parte de los usuarios de la red global.

1.4.2 E-learning y sistemas de recomendación

El e-learning es un sistema de educación electrónico o a distancia en el que se integra el uso de las tecnologías de la información y otros elementos pedagógicos-didácticos para la formación, capacitación y enseñanza de los usuarios o estudiantes en línea (19). Es ampliamente aceptado que el e-learning puede ser tan rico y tener tanto o más valor que la clase presencial.

El diseño instruccional para el e-learning se ha ido perfeccionando con el transcurso del tiempo y con la acumulación de experiencias. Como consecuencia de esto, actualmente se cuenta con una amplia red de plataformas contenedoras de los más diversos programas educativos, pudiendo mencionarse entre estas el *MIT Open Course Ware* con más de 1800 cursos, *Online-education.net* con más de 600 cursos, y *Learndirect.com* con alrededor de 500 cursos.

Resulta evidente también en esta área, al igual que en el e-Commerce, la existencia de un importante desbordamiento de información que imposibilita el acceso efectivo a los recursos disponibles en plataformas como las mencionadas y a otras aplicaciones de semejante corte. La utilización de sistemas recomendadores aplicados al e-Learning ha sido una de las soluciones para enfrentar esta problemática, 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 (20). Es válido señalar que aunque la introducción de los sistemas recomendadores en este campo tuvo una adopción tardía con respecto a la aplicación en el e-Commerce, estos actualmente revisten la misma influencia e importancia tanto en un campo como en otro.

Pueden citarse varios trabajos que muestran resultados enfocados a implementaciones reales de sistemas recomendadores aplicados al campo de la enseñanza. En (21) 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. Por otra parte, (22) muestra la implementación de un sistema para la recomendación de artículos científicos y técnicos, capaz de retroalimentarse de la web para la obtención de nueva información basándose en la adaptabilidad del perfil de usuario.

1.5 Jurados Online de Programación.

Uno de los ejemplos más representativos de herramientas de e-learning lo constituyen los jurados online de programación. Un jurado online es una aplicación de entornos generalmente académicos, publicada en la web, que provee problemas y evalúa automáticamente las soluciones de sus usuarios en uno de varios lenguajes de programación disponibles (23). Se caracterizan por contener una interfaz bien definida para la interacción con el usuario y un alto de nivel de disponibilidad, permitiendo el libre acceso a la aplicación en todo momento a las personas registradas.

Entre los jurados online de programación más representativos a nivel global pueden mencionarse el Jurado Online de la Universidad de Valladolid insertado en la web desde el año 1998, el Jurado Online de la Universidad de Gdansk en Polonia, más conocido como Jurado Online SPOJ, y el Jurado Online de la Universidad de Pekín, desplegado en China y considerado una de las mayores colecciones de problemas disponibles en la red de redes.

En la Universidad de las Ciencias Informáticas desde hace aproximadamente cuatro años se han comenzado a desplegar jurados online de programación. Actualmente los más importantes son el Jurado Online Xtreme, que pronto será insertado en la red universitaria del Ministerio de Educación Superior de Cuba, y el Jurado Online de la Cátedra de Programación Avanzada (CPAV).

1.6 Herramientas recomendadoras en jurados online de programación

Los jurados online de programación tampoco han estado exentos del sobredimensionamiento de la información en la World Wide Web, expuesto más arriba. Al ser cada día más determinante el papel de la semi-presencialidad y la no presencialidad en la educación actual, ha habido en los últimos años un incremento en el uso de este tipo de herramientas tanto en la docencia de pregrado y postgrado, como también en la preparación específica para concursos de programación.

Con vistas a facilitar la utilización de los jurados online se han concebido aplicaciones con características recomendadoras, fundamentalmente para guiar a los usuarios a la hora de seleccionar los problemas a resolver. Entre las más representativas están el **ACM Problem Grading**, de Sebastian Urbaniak, el **Next2Solve**, de Igor Naverniouk, y el **Problem**

classification, en el Sphere Online Judge. A continuación se hace un breve análisis de cada una de ellas.

1.6.1 ACM Problem Grading

Esta herramienta, junto con el Next2Solve, fue implementada en apoyo al jurado online 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 *score* de simplicidad que se calcula tomando como base los dos anteriores criterios.

El principal inconveniente de esta es el hecho de no enfocarse en un usuario en particular: simplemente supone que los problemas más sencillos a resolver para todos son aquellos que más soluciones correctas tengan. Además, todos los criterios que utiliza son completamente estáticos, siendo diametralmente opuesto a las actuales tendencias dentro de la web.

Como otro elemento negativo a resaltar está el hecho que está fuertemente atada al jurado online de la Universidad de Valladolid a tal punto de que, tras aproximadamente dos años de migrar este a un nuevo servidor y a una aplicación web con tecnología más moderna, aún el ACM Problem Grading sigue basándose en la aplicación antigua para generar los listados, no pudiendo integrarse con la nueva.

1.6.2 Next2Solve

Next2Solve, creado por Igor Naverniouk, estudiante de doctorado de la Universidad de Toronto, se autodefine como un servicio basado en el ACM Problem Grading de Urbaniak. La diferencia radical entre uno y otro viene dada en que mientras ACM Problem Grading 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 excluyéndose aquellos que él ya ha resuelto.

Aunque desde el punto de vista técnico no constituye un gran avance, sí lo es desde el punto de vista de la usabilidad, pues con el hecho de enfocarse en los usuarios de manera particular, marca una nueva tendencia a seguir por las aplicaciones posteriores.

Al depender de los scripts del ACM Problem Grading, actualmente esta aplicación se encuentra también fuera de servicio manifestando además su autor, en su página personal, que no está entre sus prioridades más importantes ponerla nuevamente a funcionar.

1.6.3 SPOJ Problem Guide

SPOJ Problem Guide es una herramienta de recomendación, asociada al jurado online SPOJ, que utiliza un enfoque distinto al presentado en las dos aplicaciones anteriores. En vez de concentrarse en el perfil de los usuarios, esta se centra en los ejercicios, definiendo para cada uno de ellos una serie de datos o etiquetas que luego son utilizadas para agruparlos por categorías. Utilizando estas categorías y a través de opciones de filtrado, los usuarios pueden obtener propuestas de problemas a resolver, basándose en criterios de búsqueda, que coinciden generalmente con los valores de las etiquetas. Unido a esto, los propios usuarios también pueden agregarle nuevas etiquetas a los problemas. Entre los valores que pueden ser tomados por estas etiquetas están backtracking, fuerza bruta, programación dinámica, entre otros.

Uno de los principales inconvenientes que tiene esta propuesta es el hecho de permitir la realización de todas las funcionalidades de manera anónima. El no gestionar de forma individual la información de cada uno de los usuarios implica que no se pueda indicar un paquete específico y personalizado de actividades a realizar, imposibilitando un mejor aprovechamiento de la misma. La realización de todo el intercambio con la aplicación de forma anónima también implica que la información que esta posee carezca de fiabilidad.

1.6.4 Insuficiencia de las herramientas recomendadoras

Tras analizar tres de las aplicaciones más destacadas en el campo de la recomendación aplicada a los jurados online se puede afirmar que estas se encuentran aún en un estado rudimentario y bastante primitivo, no a la par del desarrollo actual de muchas otras tecnologías de e-learning ni de aplicaciones recomendadoras utilizadas en otros campos.

Sólo a través de una combinación efectiva entre los datos asociados a los usuarios con los datos asociados a los problemas, desde el enfoque del problema general de la recomendación, será posible construir aplicaciones recomendadoras de relevancia en el campo de los jurados online.

1.7 Metodología de desarrollo

Para la selección de la metodología de desarrollo a utilizar fueron consideradas tres de las más populares y efectivas. Estas son el Proceso Unificado de Desarrollo (RUP), la Programación Extrema (XP) y Scrum.

RUP está basado en componentes y utiliza el Lenguaje Unificado de Modelado (UML) para preparar todos los esquemas de un sistema software. Es iterativo e incremental, centrado en la arquitectura y guiado por casos de uso. Consta de nueve flujos de trabajos fundamentales, de ellos seis denominados “de Ingeniería”, que son modelamiento del negocio, requerimientos, análisis y diseño, implementación, prueba y despliegue. Los restantes tres flujos de trabajo son considerados de apoyo y son la administración de proyectos, la gestión de configuración y cambios, y finalmente el ambiente (24).

La programación extrema, o XP, es una metodología ágil concebida y desarrollada para dirigir las necesidades específicas del desarrollo de software conducido por equipos pequeños (25). Esta le da poder a los desarrolladores para responder con confianza a los requerimientos cambiantes del consumidor final. Se caracteriza además por fomentar la comunicación desarrollador-cliente desde el primer día. Es considerada ligera, flexible, predecible, de bajo riesgo, y no por ello menos científica. Entre otras ventajas pueden mencionarse los pocos requerimientos de documentación y planificación, así como la exigencia de tener siempre el cliente disponible para el desarrollo, implicando una mejor correspondencia entre el producto y la necesidad del negocio.

Por otra parte, Scrum es otra metodología ágil que comparte muchas de las características de XP. No obstante, son notables algunas diferencias, entre las que están el hecho de que hace mayor énfasis en los flujos de gestión de proyectos y no define una práctica detallada de la ingeniería de software (26). Denomina las iteraciones como “sprints” y requiere la existencia de un equipo de desarrollo auto-organizado que no sea afectado por nuevos cambios en los requerimientos una vez iniciado el sprint, brindando de esta forma poca flexibilidad ante cambios frecuentes.

Una vez estudiadas las posibles metodologías a aplicar, se decide la inclinación por el uso de RUP. Las razones que motivaron esta decisión fueron las siguientes:

- Dadas las peculiaridades del grupo de desarrollo, conformado por solamente un individuo, resulta complejo la implementación de técnicas inherentes a la metodología XP tales como la programación por pares y la refactorización constante.
- A pesar de que RUP requeriría un equipo mayor porque mantiene más artefactos, si se maneja bien el conjunto de procesos, una persona podría hacerse cargo de todos los artefactos de manera secuencial en cada una de las etapas. Este no es el caso de XP, el que define actividades necesariamente paralelas.
- RUP permite además delegar tareas muy específicas en otras personas involucradas de manera indirecta en el proyecto; en el caso de XP, todo aquel que quiera colaborar, ha de involucrarse completamente en el desarrollo del producto.
- La aplicación que se desarrolla es de nuevo tipo, muestra un nuevo paradigma no representado anteriormente; esto implica que es importante desarrollar una buena documentación que represente de manera efectiva los procesos de la aplicación.
- Se aspira a que el proyecto tenga continuidad, por lo cual es necesario facilitar el legado de la aplicación. Una de las formas de lograr esto es a través de artefactos específicos de RUP.
- Por las particularidades del proyecto existe un solapamiento entre el cliente y el equipo desarrollador. Esto implica que no es necesario utilizar una metodología enfocada a facilitar la comunicación con el cliente.

De manera general, se puede afirmar que las dimensiones del proyecto a desarrollar aseguran su exitosa culminación con el uso de la metodología RUP.

1.8 Herramientas y tecnologías a utilizar

1.8.1 Lenguaje de programación

La aplicación a desarrollar es, por definición, una aplicación web. En esta sección se especifican las características más importantes de tres de los lenguajes de programación más utilizados para el desarrollo de aplicaciones web, que son C#, PHP y Java, justificándose la utilización para el desarrollo, de este último.

1.8.1.1 C#

C# es un lenguaje de programación orientado a objetos desarrollado por la compañía Microsoft como parte de la plataforma .NET. Sus principales creadores fueron Scott Wiltamuth y Anders Hejlsberg, este último también diseñador del Turbo Pascal y de la herramienta RAD Delphi.

Entre las características del lenguaje y de la plataforma .NET en general, se destacan la sencillez, la modernidad, la gestión automática de memoria, la seguridad de tipos y la existencia de indexadores, delegados, eventos, y tipos genéricos, entre otros rasgos (27). De forma general es considerado un lenguaje complejo, pero que una vez aprendido es de fácil uso.

1.8.1.2 PHP

PHP, acrónimo recursivo que significa PHP Hypertext Pre-processor, fue creado originalmente por Rasmus Lerdorf en 1994. Es un lenguaje interpretado, diseñado originalmente para la creación de páginas web dinámicas, y con acceso a información almacenada en bases de datos. Fue liberado bajo la PHP License; la Free Software Foundation (FSF) considera como software libre a los productos liberados bajo esta licencia.

Entre sus ventajas más resaltables está el hecho de consumir pocos recursos, por lo que generalmente se ejecuta con rapidez y no tiende a ralentizar el resto de los procesos del sistema operativo (28). Otra importante ventaja es su gran capacidad de conectividad, al utilizar un sistema de extensiones modular hacia interfaces de variado tipo, tales como librerías gráficas, XML, encriptación, etc. Entre las principales desventajas se pueden mencionar que para la realización de proyectos complejos requiere un alto nivel de experiencia, y que por determinadas limitaciones del lenguaje existe un pequeño grupo de tareas que resultan muy difíciles de realizar en el mismo (29).

1.8.1.3 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 del CPU que las procesara.

Sun describe a Java como simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico (30). Se ha desarrollado unido a Internet y es por ello 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 correctitud de la mencionada máquina virtual: en ocasiones el código está bien escrito, pero el programa no trabaja correctamente debido a errores de la máquina virtual.

1.8.1.4 Fundamentación de la elección

Cualquiera de las tres variantes propuestas más arriba es factible para ser utilizada. 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.

1.8.2 Gestor de base de datos

Para la selección del gestor de la base de datos (BD) de la aplicación fueron considerados MySQL y PostgreSQL, por contener como parte de su desarrollo versiones que constituyen tecnología libre.

A pesar de que MySQL es más rápido que PostgreSQL a la hora de resolver consultas, y de tener mejor documentación y herramientas de administración, PostgreSQL ofrece una garantía

de integridad mucho más fuerte que MySQL. Además, presenta una mejor escalabilidad y rendimiento bajo grandes cargas de trabajo (31). 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.

1.8.3 Tecnologías de desarrollo web con Java

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

El uso de estas estructuras ha sido ampliamente discutido existiendo una divergencia de criterios. Indudablemente, abstraen la arquitectura original de la tecnología sobre la que están hechos, lo que representa una ventaja al normalmente disminuir el grado de complejidad de esta y a la vez agregarle capacidades de reutilización claves en el sistema que se desarrolla. Sin embargo, se ha demostrado (33) que el uso de frameworks le agrega complejidad al software en ocasiones hasta un punto en el que el proceso de desarrollo puede llegar a situarse en un callejón sin salida. Resulta altamente complicado encontrar el conjunto ideal de herramientas de este tipo que cumplan eficientemente su función y a la vez no dejen a los desarrolladores tirados en el camino.

Considerando todo lo anterior, se decide el uso de frameworks sólo en las secciones de la aplicación donde sean absolutamente necesarios, valorándose además el grado de aceptación y estabilidad a la hora de seleccionar las herramientas específicas. En las secciones donde no se pueda garantizar, por diversas razones, la utilización exitosa de un framework, se aplica tecnología nativa de Java.

1.8.3.1 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 (34) (35), lo que no se ajusta a la aplicación actual a desarrollar.

1.8.3.2 Lógica del negocio

Como parte de la capa de la lógica del negocio se decide la utilización del framework Spring (36) por permitir implementar varias filosofías consideradas necesarias dentro una aplicación web, independientemente de su tamaño. Entre estas se pueden citar el soporte al patrón Inyección de dependencias (37) y a la Programación Orientada a Aspectos (38). Por encima de todo esto, las principales razones de la selección son la ligereza, la transparencia y el ser no intrusivo (39).

1.8.3.3 Acceso a datos

Para la implementación del acceso a datos en la aplicación, se selecciona el framework Hibernate (40), entorno de trabajo que tiene como objetivo facilitar la persistencia de objetos Java en BD relacionales. Es considerado por gran margen el framework más popular de persistencia de Java y uno de los más populares en sentido general.

Este está basado completamente en el patrón “Mapeo Objeto Relacional” (41) (42) 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.

1.8.4 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.

Acorde con la metodología de desarrollo seleccionada, se optó por utilizar como herramientas CASE una combinación entre el Rational Rose 2003 Enterprise Edition y el Visual Paradigm for UML 6.4 Enterprise Edition. Estos promueven el desarrollo de componentes iterativo e incremental, e incluyen funcionalidades para la generación automática de código a partir de los modelos, asumiendo a UML como lenguaje de modelado. Del primero se destaca su ligereza y su facilidad de uso, y del segundo su fácil integración con el resto de las herramientas de desarrollo.

El IDE seleccionado es el Red Hat Developer Studio 2.0 RC1, entorno que se ha estado empleando en producción desde hace años siempre con resultados muy positivos.

En la construcción y despliegue se decide utilizar el servidor de aplicaciones Apache Tomcat 6.0.14, una de los variantes más empleadas a nivel global para las aplicaciones con el perfil tecnológico anteriormente presentado.

La comunicación con sistemas externos por parte de la aplicación a desarrollar se concibe a través de la utilización de servicios web. Para la elaboración de estos, por los requisitos de ligereza y de limpieza a la hora de establecer la comunicación, se decide implantar la tecnología Axis, implementación Java del protocolo SOAP que pertenece a la segunda generación de las tecnologías de *web-services*.

Por último, como herramienta de alto grado de usabilidad para la administración de la BD se selecciona el EMS PostgreSQL Manager Pro 3.0 y el EMS SQL Manager 2007 for MySQL, siendo también las de más amplia aceptación al igual que las anteriores.

Conclusiones

En este capítulo se realizó un estudio del estado del arte de los sistemas de recomendación, exponiendo de forma breve su basamento teórico y principales características. Se analizaron las diferentes clasificaciones a la que estos pueden ser sometidos, basándose principalmente en la forma de procesar la información. Se llegó a la conclusión de que tienen una amplia aplicación en el mundo de la World Wide Web, comprobándose seguidamente que son también aplicables a los jurados online de programación. Fueron además analizadas algunas

herramientas recomendadoras representativas asociadas a jurados online de reconocimiento mundial. Finalmente, se concluyó que la metodología de desarrollo a utilizar será el Proceso Unificado de Desarrollo (RUP) y que el perfil tecnológico estará compuesto por Java como lenguaje de programación, Spring e Hibernate como framework de apoyo, Red Hat Developer Studio como IDE y Rational Rose y Visual Paradigm como herramientas CASE.

Concepción del sistema de recomendación

2.1 Introducción

En el presente capítulo se describe la propuesta de un sistema de recomendaciones enfocado a facilitar la interacción de sus usuarios con jurados online de programación. Dicho sistema se centra fundamentalmente en sugerir problemas a resolver, basándose en la trayectoria y el perfil de estos. Como parte de esta descripción, se realiza un análisis de diferentes soluciones desde el punto de vista algorítmico que han sido dadas para el problema de la recomendación anteriormente referenciado, así como del flujo de trabajo que deben seguir los sistemas de recomendación destinados a esta área. También se propone el perfil algorítmico del sistema en desarrollo.

2.2 Sistema de recomendación para jurados online de programación

El sistema de recomendación que se propone (43) actúa como punto intermedio entre el usuario y el jurado online (ver figura 2). A la hora de solucionar ejercicios, en lugar de dirigirse directamente al jurado online, este usuario interactuaría con el nuevo sistema (1), el que tendría almacenado previamente su perfil, siendo actualizado a través de una conexión interna con el jurado (2). Una vez poseída la nueva información asociada al usuario actual, casi siempre relacionada con nuevas soluciones o intentos de solución a ejercicios (3), el sistema sugeriría los problemas de mayor certeza de éxito ante intento de solución (4), procediendo

posteriormente el referido usuario a su lectura (5). Este proceso se comporta de manera cíclica, realizándose cada vez que el usuario se conecta a la aplicación.

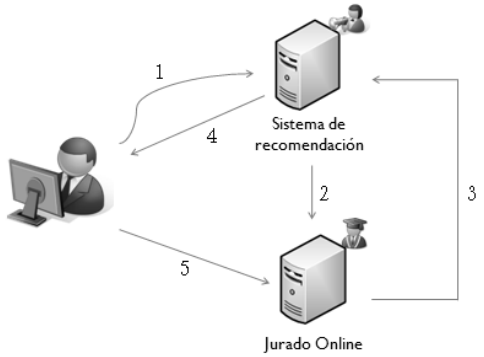


Figura 2. Sistema de recomendación

Para establecer una correcta correspondencia entre los posibles problemas a resolver y los usuarios, es necesario asociarle a ambos una serie de metadatos que permitirán identificarlos dentro de un conjunto de semejantes. En el caso de los usuarios, los datos asociados son recolectados en el momento del primer acceso al sistema, y consisten en la especificación de los temas de programación (grafos, problemas ad-hoc, geometría computacional) que se desean ejercitar con más fuerza en el jurado, así como las habilidades de programación (programación con restricción de tiempo, entrada y salida pesada, etc.) en las que se interesa hacer mayor hincapié. Opcionalmente, el usuario puede indicarle al sistema que sea este el que le sugiera los temas y habilidades a ejercitar, basándose en el perfil asociado a cada uno de los ejercicios que ya ha resuelto.

A cada problema también se le han de asociar los temas y habilidades de programación que ejercita, siendo denominada esta actividad, en el marco de la solución, “etiquetamiento del problema”. De esta tarea se encargan expertos con gran conocimiento en las materias de programación, que a la vez son usuarios del sistema.

Como valor agregado, el sistema permite que los mismos usuarios puedan también hacer recomendaciones: en el caso que alguno de ellos considere que algún problema por determinadas razones no debe dejar de ser resuelto por el resto de la comunidad, se les brinda la posibilidad de subir una recomendación del mismo que será mostrada en una pantalla aparte al resto de los usuarios en dependencia de su mayor o menor interés por los temas que abarca este.

Para facilitar la comunicación entre usuarios con intereses parecidos, se decide incorporar un sistema público de intercambio de mensajes o *shoutbox*, a través del cual el estudiante tendrá comunicación sólo con aquellos usuarios que se inclinaron por seleccionar sus mismos temas a ejercitar, mitigándose así un problema que tiende a aparecer en el campo de los jurados online, que es la dificultad de un usuario novel para encontrar su lugar y su medio de interacción dentro de la gran comunidad que estos tienen implícita..

En la aplicación están comprendidos los roles de administrador, experto y usuario estándar, incluyéndose la gestión de estos por parte del rol de administrador. Finalmente, dentro de esta gestión, es brindada la posibilidad de promover a los usuarios, pudiendo pasar un usuario estándar y un experto a ejercer el rol de experto y de administrador respectivamente; también se considera la acción contraria.

2.3 Algoritmos en sistemas de recomendación

Con el fin de dar una solución eficaz y computacionalmente eficiente al problema de brindar recomendaciones que se correspondan con los intereses de un usuario determinado, se han utilizado diferentes enfoques que pueden ser agrupados en dos categorías fundamentales: los algoritmos basados en memoria y los basados en modelos.

2.3.1 Métodos basados en memoria

Los algoritmos basados en memoria, ampliamente empleados en sistemas de filtrado colaborativo, son en esencia heurísticas que realizan predicciones de las preferencias de los usuarios por determinados ítems, basándose en la evaluación previa de todos los ítems, dada por todos los usuarios. Con el fin de brindar las recomendaciones, los datos normalmente son procesados en el mismo momento en que son solicitadas estas.

Generalmente primero emplean técnicas estadísticas para encontrar a vecinos, es decir, usuarios con un historial de valoraciones sobre los elementos similar al usuario actual. Una vez que se ha construido una lista de estos, se combinan sus preferencias para generar una lista con los N elementos más recomendables para el usuario actual. Entre sus inconvenientes se encuentra la necesidad de disponer de un número mínimo de usuarios con un número mínimo

de predicciones cada uno, incluyendo el usuario para el cual se pretende realizar la recomendación (44).

2.3.1.1 K – Nearest Neighbor

Los algoritmos de vecinos más cercanos, o k-nearest neighbor, fueron de los primeros algoritmos de filtrado colaborativo en implementarse. Están conformados por tres etapas fundamentales, que son la representación de los datos, la formación de vecinos, y la generación de recomendaciones (9), siendo la más crítica la segunda de ellas, por poder afectar el rendimiento del sistema y la eficacia de las recomendaciones en los casos en que no se conciba de la manera más correcta.

Se han definido varias medidas de similitud de elementos para estos algoritmos con vistas a utilizarlos como métrica durante el proceso de formación de vecinos. La más general de todas la constituye la **distancia euclidiana** (45), la cual se define para toda estructura que pueda asumir la forma:

$$(a_1(x), a_2(x)a_3(x), \dots, a_n(x))$$

donde $a_r(x)$ denota el valor del atributo r de la instancia x. Para esta representación, la distancia entre dos instancias x_i y x_j , definida como $d(x_i, x_j)$ viene dada por:

$$d(x_i, x_j) = \sqrt{\sum_{r=1}^{n} (a_r(x_i) - (a_r(x_j))^2}$$

Esta función de cercanía no es propia de sistemas de recomendación. No obstante, su aplicación en algunos casos puede dar resultados satisfactorios si se selecciona de manera eficaz el vector que representa los datos del usuario, con los cuales se determinarán las distancias.

Otra de las funciones de semejanza más populares, en esta caso ya con considerable utilización en el campo de los sistemas de recomendación, lo constituye la **similitud basada en el coseno** (46), la que da una buena medida del “parecido” de dos vectores en un espacio

multidimensional, pudiendo este espacio describir características de usuarios o de ítems, tales como palabras clave, entre otras (47). El cálculo de este parecido viene dado por el coseno del ángulo formado por la representación de estos dos vectores en el espacio n-dimensional, tal y como lo muestra la expresión:

$$sim_{ik} = cos_{ik} = \frac{r_i \cdot r_k}{||r_i|| * ||r_k||} = \sum_j \frac{r_{ij}}{\sqrt{\sum_j r_{ij}^2}} \frac{r_{kj}}{\sqrt{\sum_j r_{kj}^2}}$$

la que permite calcular la proximidad entre dos usuarios u_i y u_k, siendo r_{ij}la evaluación dada por el usuario $u_i$al ítem i_j.

Una manera más sofisticada que la similitud del coseno para establecer semejanzas dentro de los sistemas de recomendación son las similitudes basadas en correlación, las que no son más que medidas estadísticas que permiten establecer semejanzas entre dos elementos, dados determinados valores cuantitativos que los caracterizan. De particular efectividad (3) resulta el **coeficiente de correlación de Pearson**, que viene dado, para los usuarios a y b por:

$$w(a,b) = \frac{\sum_k (V_{a,k} - V_a)(V_{b,k} - V_b)}{\sqrt{\sum_k (V_{a,k} - V_a)^2 \sum_k (V_{b,k} - V_b)^2}}$$

donde $V_{i,k}$es el voto emitido por el usuario i acerca del ítem k, V_i es la media de todos los votos para el usuario i, y K es el conjunto de elementos evaluados tanto por el usuario a, como por el usuario b. Este método indica cuan linealmente correlacionados están los elementos de un vector con los del otro, lo que de manera gráfica se vería si se toma el plano cartesiano, en uno de los ejes se ubica los valores asociados a un vector, en el otro eje los asociados al otro vector, y se determina la cercanía a la existencia de una recta que contenga todos los puntos de intersección de los valores de un eje con los de otro (48). En el caso de dar como resultado 1, esto se interpreta como que existe una correlación total, o en este caso relación de proporcionalidad directa total, entre un vector y otro; en el caso de obtenerse -1, esto indicaría que la relación es de proporcionalidad inversa total. En cambio, la obtención de un 0 como coeficiente de Pearson indica que la relación entre un vector y otro es nula o no existe.

Como una alternativa al coeficiente de correlación de Pearson, con mucha frecuencia se utiliza el **coeficiente de Jaccard-Tanimoto** por resultar más práctico en determinadas circunstancias.

Este se centra en calcular la similitud entre dos conjuntos basándose en la cantidad de elementos comunes que poseen, y se define, siendo N_a la cantidad de elementos del conjunto A, N_b la cantidad de elementos del conjunto B, y N_c la cantidad de elementos de la intersección de A y B, de la siguiente forma:

$$T = \frac{N_c}{N_a + N_b - N_c}$$

Este método tiende a ser utilizado en las recomendaciones basadas en el contenido, a la hora de establecer semejanzas entre ítems, considerando como componentes vectores los usuarios que manifestaron su preferencia por el ítem correspondiente.

Tras obtener los vecinos más cercanos a través de determinado criterio de similitud, sus datos asociados son utilizados para generar las recomendaciones para el usuario actual, debiendo estas estar enfocadas a ítems que aún no hayan sido considerados o evaluados por este. Con dichos fines, la técnica más utilizada es la *Most-Frequent Item Recommendation*, consistente en llevar un conteo de la frecuencia en que cada uno de los ítems es evaluado o considerado por parte de los usuarios que están en el vecindario del usuario activo. Tras completar el listado de frecuencias de aparición de ítems, se procede a excluir aquellos ya evaluados o considerados por el usuario activo, ordenando los restantes de mayor a menor considerando esta frecuencia de aparición, siendo retornados los N primeros elementos de este listado final en calidad de recomendaciones para dicho usuario.

Otra variante a utilizar para la generación de las recomendaciones es la *Association Rule-based Recommendation*. En el dominio de este método, considerando la existencia de n ítems $I = \{i_1, i_2, \dots, i_n\}$, se define una transacción $T \subseteq I$ como un conjunto de ítems que son evaluados o considerados juntos, en una misma interacción del usuario con el sistema. Una regla de asociación entre dos conjuntos de elementos I_x y I_y, donde $I_x, I_y \subseteq I$ y $I_x \cap I_y = \emptyset$, establece que si están presentes los ítems I_x en una transacción T, entonces existe una gran probabilidad de que los ítems de I_y también podrían estar presentes en T. Esta regla de asociación normalmente en lenguaje formal se denota como $I_x \rightarrow I_y$.

Una vez obtenidas las reglas de asociación, es posible la generación de recomendaciones dirigidas a un usuario en específico. Con este propósito, para cada uno de los usuarios que conforman el "vecindario" de este, es creada una transacción conteniendo todos los ítems valorados o considerados en el pasado. Teniendo como entrada estas transacciones, se define un algoritmo de descubrimiento de reglas de asociación para encontrar aquellas que sean

soportadas por determinado o determinados usuarios, esto es, que todos los ítems presentes en la parte izquierda de la regla estén presentes en la transacción del usuario correspondiente. Posteriormente, se forma un listado de elementos no repetidos conformado por los ítems de la parte derecha de las reglas aplicables, que aún no hayan sido considerados por el usuario activo, entregándosele a este en calidad de recomendaciones finales, tras ser ordenado por el grado de confianza de la regla que lo generó. Para los ítems que se generan por más de una regla, se selecciona para el ordenamiento la regla de mayor confianza.

2.3.2 Métodos basados en modelos

En contraste con los algoritmos basados en memoria, los basados en modelos utilizan la colección de evaluaciones o de ítems considerados, con el objetivo de *aprender* un modelo del usuario, el que es entonces usado para mostrar predicciones de evaluación, o mostrar recomendaciones. La implementación efectiva de este "aprendizaje" usualmente implica la utilización de técnicas estadísticas y de aprendizaje automático dentro del sistema de recomendación.

Empíricamente se ha demostrado el incremento del rendimiento en términos general de un sistema de recomendación con la utilización de métodos basados en el modelo, aunque algunos autores (49) señalan que pueden sufrir problemas de desbordamiento de memoria como precio a pagar ante el intento de producir recomendaciones de alta calidad.

Esta familia de algoritmos está intrínsecamente relacionada con los sistemas de recomendación basados en contenido, aunque también es aplicable al filtrado colaborativo.

2.3.2.1 Recomendación bayesiana

Uno de los modelos más utilizados para reflejar las preferencias de los usuarios lo constituyen las redes bayesianas (50). Una red bayesiana es un grafo acíclico dirigido en el que los nodos son variables y los arcos representan relaciones de influencia causal entre ellos. La información cuantitativa a suministrar para cargar la red está dada por las probabilidades a priori de los nodos que no tienen padres, y la probabilidad condicionada de los nodos con padres, tal como se muestra en la figura.

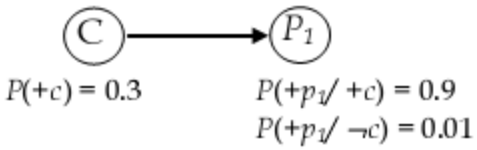


Figura 3. Red bayesiana con parámetros

Las redes bayesianas son aplicables al campo de los sistemas de recomendación, pudiendo servir cada uno de sus nodos como representación de los ítems a seleccionar, con la respectiva probabilidad por parte de estos, de ser seleccionado. Las probabilidades condicionales, por su parte, representarían la probabilidad de seleccionar un ítem determinado, conociéndose previamente la selección de un ítem anterior que guarde intrínseca relación con este.

El aprendizaje de este modelo se divide normalmente en dos categorías fundamentales, que son el paramétrico y el estructural. En el aprendizaje paramétrico, la estructura de la red frecuentemente es definida por un experto mientras que los parámetros son "aprendidos" a través de algoritmos específicos tales como el *Expectation-Maximization* (51), tomando como punto de partida los datos iniciales disponibles acerca del tema en cuestión. Por otra parte, en el aprendizaje estructural, el propio algoritmo debe obtener la estructura de la red, además de determinar sus parámetros. Entre los métodos más usados para este último enfoque está el definido por Chow y Liu en (52).

2.3.2.2 Clustering

Las técnicas de clustering trabajan a través de la identificación de grupos o clusteres con usuarios que evidencien tener semejantes intereses. Una vez que los clusteres son creados, las predicciones de cualquiera de sus individuos miembros pueden ser hechas promediando las opiniones del resto de los usuarios en el cluster correspondiente. Algunas técnicas de clustering consideran además la participación del usuario en más de una de estas estructuras.

Este enfoque directamente aplicado produce recomendaciones que en muchos casos tienen menor calidad que las obtenidas por otras herramientas similares (53). No obstante, es importante resaltar que su principal objetivo no es el de ofrecer sugerencias, sino el de reducir la dimensión del espacio contenedor de todos los datos de los usuarios e ítems,

particionándolos en varios espacios más pequeños con menor cantidad de ítems, menor número de preferencias pre-almacenadas y menor cantidad de usuarios, pudiendo entonces aplicarse algoritmos de filtrado colaborativo que anteriormente no se podían utilizar garantizando la eficiencia, por la gran cantidad de elementos y asociaciones en el espacio inicial. Esto deviene en una mejora notable en la calidad de las recomendaciones, tanto en exactitud como en rendimiento.

Entre los algoritmos más populares para la implementación del clustering de manera general, están el *k-means clustering* (54) y el cluster jerárquico (48), los que han encontrado aplicación en disímiles áreas, incluyendo la de los sistemas de recomendación. Como ejemplos de sistemas de recomendación que han utilizado esta teoría están (22) y (55).

2.3.2.3 Horting

Horting es una técnica basada en grafos, definida en (56) específicamente para su aplicación en sistemas de recomendación. En esta, los nodos del grafo representan los usuarios y las aristas el grado de similitud entre un usuario y otro. Difiere a su vez del método de los vecinos más cercanos al considerar, para la elaboración de las recomendaciones, las relaciones transitivas entre los usuarios como un factor importante a la hora de establecer la proximidad; cuestión que aquellos no toman en cuenta.

La idea central de este enfoque reside en predecir la evaluación del ítem i por parte del usuario j, siendo esta computada como el promedio de una serie de pesos calculados a partir de los correspondientes caminos dirigidos que unen al usuario en cuestión con un conjunto de usuarios ya previamente evaluadores de i.

Desde el punto de vista del rendimiento, se considera que esta técnica es rápida, escalable, precisa y además, con una modesta curva de aprendizaje.

2.4 Limitaciones de los algoritmos en sistemas de recomendación

El hecho de trabajar en sentido general con información con alto grado de incertidumbre ocasiona que los métodos para establecer recomendaciones se encuentren con limitaciones que pueden dificultar la consecución de su objetivo final. A la hora de concebir un sistema de

recomendación, es importante no dejar de tener en cuenta estas limitantes. A continuación se realiza un breve recorrido por algunas de las más significativas.

2.4.1 Arranque en frío

Constituye una de las cuestiones críticas a resolver al inicio del despliegue de todo sistema de recomendación, independientemente de la línea que se haya seguido en su desarrollo. Este problema hace referencia a las situaciones en las que se dispone solamente de un reducido número de evaluaciones o consideraciones para realizar las recomendaciones solicitadas, pudiendo ocurrir específicamente bajo tres escenarios diferentes:

Nuevo usuario: Cuando un nuevo usuario se registra en el sistema de recomendación, existen muy pocos datos disponibles para definir con cierta precisión su perfil.
Nuevo ítem: Cuando un nuevo ítem es adicionado al catálogo, inicialmente no presenta evaluaciones.
Nuevo sistema: Cuando se pone a funcionar una nueva instancia de un sistema de recomendación, el número promedio de evaluaciones o consideraciones por usuario y por ítem es bajo, lo que puede afectar significativamente la calidad de las recomendaciones.

Las principales estrategias para la mitigación de esta problemática han sido la utilización de sistemas híbridos que combinen la recomendación basada en el contenido con el filtrado colaborativo, así como el empleo de criterios emergentes tales como la popularidad y la entropía de los ítems (57).

2.4.2 Dispersión de los datos

En todo sistema de recomendación se ha comprobado que, en la práctica, el número de evaluaciones reales que se posee acerca de un ítem específico queda muy por debajo del número mínimo que se requiere para hacer recomendaciones efectivas, haciéndose particularmente crítico para aquellos usuarios con preferencias alejadas de las de la mayoría. Esto implica que la predicción efectiva a partir de un número reducido de ejemplos juegue un papel clave para el éxito del sistema.

Una manera de enfrentar esta limitante es utilizar datos almacenados en el perfil del usuario a la hora de establecer la función de similitud. Así, para determinar si un usuario es semejante a otro, no sólo se tendría en cuenta si ambos muestran su preferencia por determinados ítems, sino que también entran en consideración la edad, el género, el nivel educacional, entre otros. Esta extensión del tradicional filtrado colaborativo, en ocasiones ha sido referenciada como filtro demográfico.

Se han desarrollado también soluciones teóricas (58) que han ido enfocadas a la reducción de la dimensión de los datos con vistas a elevar la calidad de la información que se extrae de estos.

2.4.3 Sobre-especialización

La sobre-especialización en sistemas de recomendación está directamente asociada con los sistemas basados en el contenido. Este problema aparece cuando solamente se recomiendan ítems que se corresponden fuertemente con el perfil del usuario, limitando a estos a recibir únicamente sugerencias de elementos similares a aquellos que ya ha evaluado o considerado.

Con vistas a garantizar cierta diversidad en cuanto a las recomendaciones brindadas a determinado usuario, en muchos sistemas de recomendación se ha dado cabida a la aleatoricidad, a través de técnicas como la de los algoritmos genéticos. El uso de sistemas de recomendación híbridos, donde se utilicen como basamento para la recomendación tanto la información del propio usuario como la de usuarios semejantes, también se ha utilizado para evitar la tendencia a la aparición de esta problemática.

2.5 Perfil algorítmico propuesto para el sistema de recomendación

El estudio de varias de las tendencias líderes a la hora de definir el perfil algorítmico para un sistema recomendador así como los diferentes puntos neurálgicos que suelen aparecer asociados a estos, permitieron definir la estrategia a aplicar en la aplicación en cuestión.

Se concibieron dos algoritmos fundamentales para acoplar a la aplicación. Uno de ellos tiene un carácter *ad-hoc*, y está basado en ideas generales de filtrado colaborativo apoyándose en la

definición de una función de semejanza entre los usuarios, mientras que el segundo es una implementación de la recomendación basada en reglas de asociación.

2.5.1 Recomendación *ad-hoc* basada en filtrado colaborativo

La filosofía de la recomendación basada en filtrado colaborativo aplicada a los jurados online de programación, se basa en sugerir al usuario aquellos problemas que a pesar de haber sido resueltos por aquellos usuarios semejantes a él, aún no han sido resueltos por este.

Para representar la información asociada a un usuario en específico, se decide asumir un vector con la forma:

$$< cantidadProblemasResueltosTema1, cantidadDeProblemasResueltosTema2, \ldots,$$
$$cantidadProblemasHabilidad1, cantidadProblemasHabilidad2, \ldots \ldots >$$

donde *cantidadProblemasResueltosTema1* representa la cantidad de problemas resueltos, por parte del usuario, del tema1; *cantidadProblemasResueltosTema2* representa la cantidad de problemas resueltos, por parte del usuario, del tema2, etc. La forma de obtener los temas y las habilidades a ejercitar ya se explicó anteriormente en el epígrafe 2.2.

Una vez obtenidos los vectores asociados a todos los usuarios se cuenta con todos los datos necesarios para aplicar la estrategia de recomendación. El principio de funcionamiento de esta se basa en el hecho de que dos usuarios que tengan resueltos la misma cantidad, o una cantidad semejante de ejercicios por cada uno de los temas y habilidades, implica que estos tengan intereses semejantes, pudiendo ser esto considerado para generar las recomendaciones. Expresándolo de una forma un poco más técnica, dos usuarios cuyos vectores asociados tengan una alta correlación.

Específicamente, el algoritmo para ofrecer recomendaciones, al usuario X, de problemas a resolver, es el siguiente:

1- Aplicar correlación de Pearson entre el vector asociado a X y los asociados a los restantes usuarios del jurado, con vistas a obtener un grado de semejanza de cada uno de los usuarios del jurado con el usuario actual.

2- Se multiplica cada grado de semejanza por la razón de problemas aceptados sobre total de envíos de cada uno de los usuarios. A los resultados de la multiplicación se le denomina nivel de confiabilidad con respecto a X.
3- Se crea, con los problemas del jurado, una sublista con aquellos que tengan al menos una tema o habilidad en común con el perfil de X.
4- Para cada problema de esta sublista, se establece una suma de los niveles de confiabilidad con respecto a X, de todos los usuarios que le han dado solución correcta. A esta suma se le denomina ponderación del problema.
5- Se muestra, en calidad de recomendaciones finales esta sublista ordenada descendentemente por la ponderación de los problemas.

2.5.2 Recomendación basada en reglas de asociación

Con el fin de incorporarle características híbridas al sistema de recomendación se define un segundo algoritmo, en este caso basado en reglas de asociación, las que en su conjunto conforman un modelo. Para esto, se realizó una adaptación de la técnica mencionada al final del epígrafe 2.3.1.1 relacionada con este tema.

El objetivo de este método es generar reglas de la forma P → Q, donde tanto P como Q son conjuntos de ejercicios. El significado de estas sería el siguiente: si un estudiante X logró resolver todos los problemas del conjunto P, esto implica que está capacitado para resolver todos los problemas del conjunto Q. Una vez obtenidas un número de reglas de este tipo, para realizarle recomendaciones a un estudiante basta con revisar si para alguna de ellas se cumple que el estudiante haya resuelto todos los ejercicios presentes en el antecedente de la regla, y no todos los ejercicios del precedente. En este caso, se recomiendan los ejercicios del precedente que aún no han sido resueltos por el estudiante.

De manera secuencial, la estrategia sería la siguiente:

1- Obtener las reglas de asociación.
2- Para cada regla de asociación, verificar si se cumple la condición de el estudiante tener resuelto todos los ejercicios que forman parte del antecedente.
3- En caso de cumplirse la condición, revisar si existe algún ejercicio del precedente que aún no haya resuelto.

4- En caso de existir, insertar el problema en una lista de posibles recomendaciones, así como el nivel de confianza de la regla que lo genera.
5- De haber sido anteriormente insertado el problema, actualizar el nivel de confianza, si el valor de este es superior al que tenía asociado dicho problema previamente.
6- Se devuelven, en calidad de recomendaciones, el listado ordenado de manera descendente según los niveles de confianza.

Para obtener las reglas de asociación se concibió la utilización del algoritmo Apriori, definido por Agrawal y Ramakrishnan en (59), y considerado como el método de referencia a la hora de generar este tipo de estructuras, así como uno de los cinco algoritmos de minería de datos más utilizados a nivel global. El volumen de los datos y el contexto en el que se concibe posibilita la utilización de este en su versión original, siendo innecesario aplicar algunas de sus variantes creadas buscando rendimiento más óptimo.

Apriori permite la obtención de reglas de asociación con un nivel de soporte y de confianza mínimo determinado, habiéndose mencionado este segundo concepto ya dentro del algoritmo. En el dominio del problema actual, nivel de soporte de una regla es la probabilidad de que un estudiante haya resuelto todos los ejercicios presentes tanto en el antecedente como en el precedente de la regla, mientras que nivel de confianza es la probabilidad condicional de que un estudiante que haya resuelto los ejercicios del antecedente, haya resuelto también los ejercicios del precedente.

La corrida de Apriori en determinados momentos de la ejecución de la aplicación garantizará la actualización continua de las reglas, permitiendo que las recomendaciones sean realizadas acorde al estado actual del jurado online, agregándole dinamismo al proceso.

2.5.3 Análisis de los algoritmos de recomendación

Se puede afirmar que las dos estrategias de recomendación a utilizar como parte de la aplicación se complementan mutuamente. Por un lado, el algoritmo *Ad-Hoc* de filtrado colaborativo presenta una propuesta dinámica para la generación de recomendaciones que considera a todos los usuarios del jurado basándose en una función de semejanza definida y utilizada para comparar a cada uno de ellos con el usuario a recomendar. Además para establecer las sugerencias, considera el desempeño de cada uno de estos usuarios a la hora

de indicar, al usuario a recomendar, determinados problemas resueltos por estos. Contrario a lo deseado, esta cantidad de factores a considerar puede dar lugar a la aparición de un sobreajuste u *overfitting*, el que, en este caso, surge cuando, ante un desmedido intento de precisión a la hora de generar recomendaciones, estas se hagan con un carácter erróneo, en calidad de ruido, por tal vez asumir algunas hipótesis a raíz de estos factores, que en la práctica sean falsas.

Haciendo referencia a la recomendación basada en reglas, algunos autores como (60) han considerado que estas no ofrecen el nivel de detalle y personalización que otros métodos garantizan. A pesar de esto, se considera que si estas reglas son regeneradas de manera periódica y automática, cuestión no tenida en cuenta por (60), las recomendaciones derivadas ganan en precisión y dinamismo. Por estas razones se decidió la inclusión de este tipo de recomendación ya expuesto en el epígrafe anterior, como complemento al método de filtrado colaborativo. Así, mientras el filtrado colaborativo le añade dinamismo a la recomendación, el criterio basado en reglas garantiza la calidad de estas.

A través de la combinación de estos dos algoritmos se logra mitigar el efecto de algunas limitaciones de los sistemas de recomendación expuestas en 2.5. El hecho de poseer un paquete de reglas de asociación como base para la generación de sugerencias posibilita enfrentar de manera exitosa muchas situaciones asociadas al arranque en frío en el caso de la existencia de nuevos usuarios, dándose más detalles de esto en la siguiente sección. El papel clave de un método de filtrado colaborativo dentro de la arquitectura de recomendación evita la sobre-especialización de los usuarios y, finalmente, con respecto a la limitante de la dispersión, aun cuando por determinado estado de la base de datos del jurado online esta está presente afectando la calidad de las recomendaciones del filtrado colaborativo, el otro mecanismo de recomendación garantiza que el usuario no deje de recibir buenas sugerencias.

2.5.4 Tratamiento del arranque en frío o cold-start

Los algoritmos presentados anteriormente no muestran ninguna recomendación en el caso que el usuario sobre los cuales se corran no tenga ningún ejercicio resuelto. En este caso, en el ejemplo actual se decide recomendar aquellos problemas que más apariciones tengan dentro de los conjuntos de antecedentes en las reglas generadas. Por la propia definición de regla de asociación, son estos problemas la antesala de la resolución de un buen número de ejercicios

en el jurado, 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 estudiante hasta un punto en el que el algoritmo de filtrado colaborativo comenzaría a ser eficaz.

Esta es la estrategia que se define en la aplicación, para el problema del arranque en frío.

2.6 Concepción del sistema de recomendación

A pesar de no estar escrita una definición estándar para el flujo de trabajo que debe llevar a cabo un sistema de recomendaciones aplicado al e-learning, varios autores, de forma independiente, han establecido basamentos para la concepción de estos, basamentos que de una forma u otra, se han convertido en estándares de facto. En el sistema de recomendación que se desarrolla, se decide utilizar el definido en (20), por adaptarse de forma clara al dominio del problema que se plantea.

Siguiendo estas ideas, el proceso de generar recomendaciones consta de un flujo de trabajo compuesto por cinco etapas fundamentales, que son la **recolección de los datos**, la **extracción, transformación y carga** (ETL) de estos, la **generación del modelo**, la **configuración** y la **prestación del servicio**. En la propuesta actual se llevan a cabo estas actividades, mereciendo destacarse las dos primeras.

2.6.1 Recolección de los datos

En la mayoría de los sistemas de recomendación la etapa de la recolección abarca el análisis de los datos históricos y el descubrimiento de los intereses del usuario y de las páginas recientemente visitadas; información que se obtiene fundamentalmente a través de logs de acceso y evaluaciones explícitas por parte de los usuarios. En el caso del sistema de recomendación para jurados online de programación el concepto cambia ligeramente, pues es el jurado la única fuente a partir de la cual se puede obtener información para conformar el perfil de los estudiantes. A pesar de ser la única, aporta datos valiosos para generar buenas recomendaciones, más allá de las dificultades asociadas a los sistemas de recomendación expuesta en epígrafes anteriores.

Como parte del modelo de datos del jurado online se cuenta con tablas como *users*, *solutions* y *problems*, que permiten relacionar a los usuarios con los problemas propuestos. Por otra parte, para la recolección de información inherente al dominio del sistema y no al del jurado online, tal como las preferencias de los usuarios, el etiquetamiento de los problemas y las recomendaciones realizadas por usuarios, se conciben funcionalidades específicas dentro de la aplicación, tal y como se describe al principio del capítulo.

2.6.2 ETL

En esta etapa del flujo de trabajo se realiza la extracción, transformación y carga de los datos hacia la fuente principal, pudiendo hacerse este procesamiento de diferentes maneras, en dependencia de los requerimientos. En la aplicación, la propia herramienta recomendadora es la encargada de garantizar que exista una sincronía total entre su BD y la fuente primaria de información, que es la BD del jurado online, al ejecutar estas tres tareas fundamentalmente cada vez que el usuario entra el sistema, tal y como se muestra en la figura 4.
Este proceso se expone a un mayor nivel de precisión en uno de los diagramas de colaboración que se presenta en el siguiente capítulo.

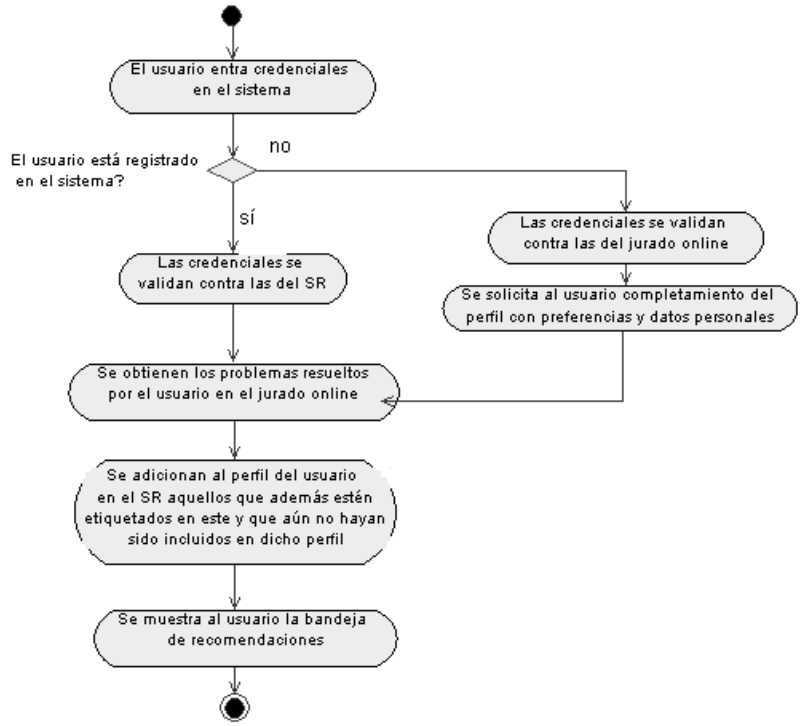


Figura 4. Actividades durante el acceso de un usuario a la aplicación

2.6.3 Carácter cíclico del proceso

Es importante resaltar que tanto en la aplicación actual como en todo sistema de recomendación, el flujo de trabajo expuesto tiene carácter cíclico, pues aun estándose prestando los servicios de recomendación, hay que realizar una y otra vez los procesos de colección de datos, ETL, generación del modelo, y configuración y prestación del servicio, buscando posibles actualizaciones de dicho modelo, para poder brindar en todo momento las mejores recomendaciones. Esto se refleja claramente en la figura 5.

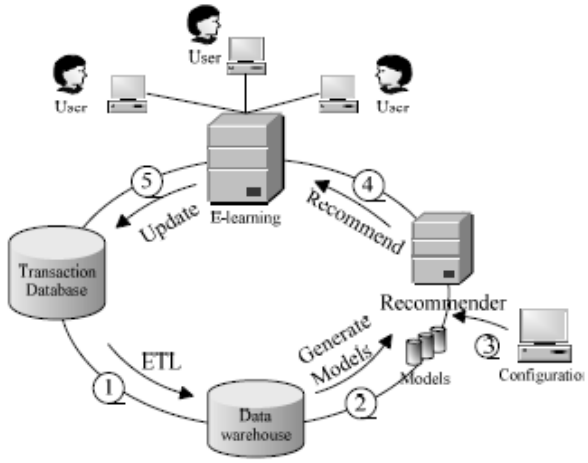


Figura 5. Carácter cíclico del proceso de generar recomendaciones.

Conclusiones

En este capítulo se realizó una descripción detallada del sistema de recomendación, llegándose a la conclusión de que fungirá como mediador entre el usuario y un jurado online de programación ya desarrollado y desplegado, y brindará a los usuarios recomendaciones de los ejercicios a resolver, proponiéndole de esta forma un ruta para la adquisición óptima de conocimiento. Seguidamente se hizo un recorrido por las principales enfoques algorítmicos utilizados para dar solución efectiva al problema de la recomendación, definiéndose claramente dos filosofías distintas para enfrentar este: los algoritmos basados en memoria, y los algoritmos basados en modelo. Como parte de los enfoques se analizaron los métodos de vecinos más cercanos, asociados al primer enfoque; y las redes bayesianas, el clustering y el horting, asociados al segundo. Además, se definió el perfil algorítmico del sistema en desarrollo, conformado por un enfoque híbrido que toma ideas tanto de una como de otra filosofía, siendo esta otra conclusión importante. Finalmente se hizo corresponder el flujo de trabajo del sistema

de recomendación con el definido como estándar para sistemas de e-learning en un artículo proveniente de una revista referenciada, permitiendo obtener como conclusión que el sistema actual tiene muchos puntos comunes con otros de probado éxito.

Desarrollo de la solución propuesta

3.1 Introducción

En el presente capítulo se lleva a cabo el desarrollo del sistema de recomendación para jurados online de programación utilizando la metodología RUP. Se justifica la elaboración de un modelo de dominio que da cabida a la definición de los requisitos funcionales y no funcionales de la aplicación, así como a los casos de uso del sistema. Se presentan además los modelos de clases del análisis, con los respectivos diagramas de colaboración. Se muestra el modelo de diseño y como parte de este los diagramas de clases de diseño; y se hace referencia a los principales patrones de diseño y arquitectura utilizados. El capítulo concluye con una referencia a la implementación de la aplicación.

3.2 Modelo de dominio

Dados los pocos conceptos que abarca el negocio del sistema de recomendación, se opta por definir un modelo de dominio (59). Este es una representación visual de los objetos del mundo real que resultan de interés para el problema, proveniente de la base de conocimientos de expertos, o de conocimiento asociado a sistemas similares. No se desarrolla modelo de negocio además, porque inicialmente no están completamente definidas las fronteras del negocio, ni quienes son las personas que desarrollan cada una de las actividades que forman parte de los procesos de negocio.

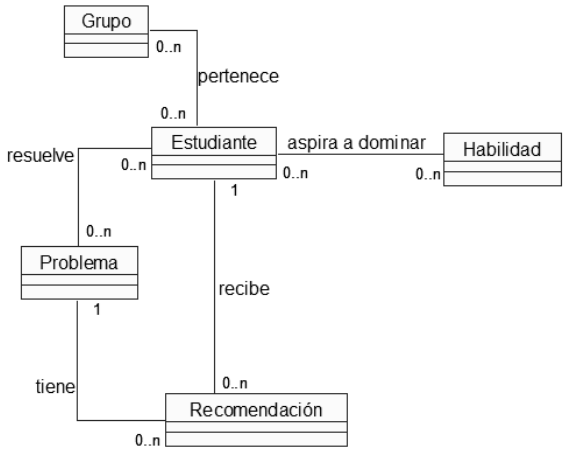


Figura 6. Modelo de dominio del sistema de recomendación

En la figura 6 se presenta el modelo de dominio de la aplicación, el que tiene en cuenta todo lo planteado en las características del sistema y en la descripción de los flujos de trabajo expuestos en el capítulo 2. Se puede apreciar claramente como el estudiante es el concepto central que dicta y genera todas las relaciones.

3.3 Requerimientos

Tras analizar el dominio del problema, resulta fácil identificar los requerimientos funcionales y no funcionales que tendrá la aplicación que se desarrolla. Los requerimientos funcionales son capacidades o condiciones que el sistema debe cumplir, mientras que los no funcionales son propiedades o cualidades que el producto debe tener. Los requerimientos forman parte del modelo del sistema que se desarrolla.

A continuación se presentan los requerimientos funcionales y no funcionales extraídos.

3.3.1 Requerimientos funcionales

RF-1 Iniciar sesión

RF-1.1 Autenticar usuario.

RF-1.2 Cargar datos de usuario en jurado.

RF-1.3 Sincronizar BD de sistema de recomendación con BD del jurado.

RF-1.4 Entrar perfil de usuario.

RF-1.5 Crear perfil de usuario de manera automática.

RF-1.6 Cerrar sesión.

RF-2 Mostrar bandeja de recomendación

RF-2.1 Mostrar recomendaciones automáticas.

RF-2.2 Mostrar recomendaciones de usuario.

RF-2.3 Recomendar ejercicio por usuario.

RF-2.4 Entrar mensaje de ShoutBox.

RF-2.5 Mostrar mensaje de ShoutBox.

RF-2.6 Ver detalles de recomendación.

RF-3 Importar problema de jurado online

RF-3.1 Etiquetar problema.

RF-3.2 Consultar problema.

RF-4 Gestionar usuarios

RF-4.1 Promover usuario.

RF-4.2 Insertar tema.

RF-4.3 Insertar habilidad.

RF-4.4 Generar reglas.

3.3.2 Requerimientos no funcionales

Los requerimientos no funcionales para la aplicación se centran en lograr un funcionamiento eficiente del sistema con vistas a una rápida aceptación por parte de los usuarios. Además, están determinados por la naturaleza del software y del hardware sobre el cual será desplegada la aplicación. Como parte de los anexos se muestran una descripción de los principales requisitos no funcionales de la aplicación a desarrollar.

3.4 Modelo de casos de uso del sistema

3.4.1 Actores del sistema

Un actor es alguien o algo, externo al sistema, que de cierta forma interactúa con este (61). Puede ser una persona, un dispositivo de hardware, u otro sistema. En la tabla 1 se reflejan los actores que componen el sistema en desarrollo, así como la justificación de su selección.

Tabla 1.Actores del sistema

Actores	Descripción
Usuario	Persona que tendrá acceso al sistema como usuario estándar, teniendo la posibilidad de recibir y realizar recomendaciones y enviar mensajes por el shoutbox.
Experto	Además de todos los privilegios del usuario estándar, puede etiquetar problemas.
Administrador	Además de todos los privilegios del experto, podrá realizar la gestión de los usuarios, la inserción de nuevos temas y habilidades, y la generación de reglas.

3.4.2 Diagrama de casos de uso de sistema

Un caso de uso consiste en una secuencia de acciones llevadas a cabo por un sistema, que concluyen con un resultado observable de valor para un actor en particular (61). En la figura 7 se presenta el diagrama de casos de uso del sistema correspondiente a la aplicación en desarrollo.

3.4.3 Descripción de los casos de uso del sistema

La descripción de los casos de uso del sistema contiene una detallada descripción del flujo de eventos a ejecutar, las precondiciones y poscondiciones del caso de uso, así como los requisitos funcionales que este satisface. Como parte de los anexos se muestran las descripciones de los casos de uso más importantes del sistema.

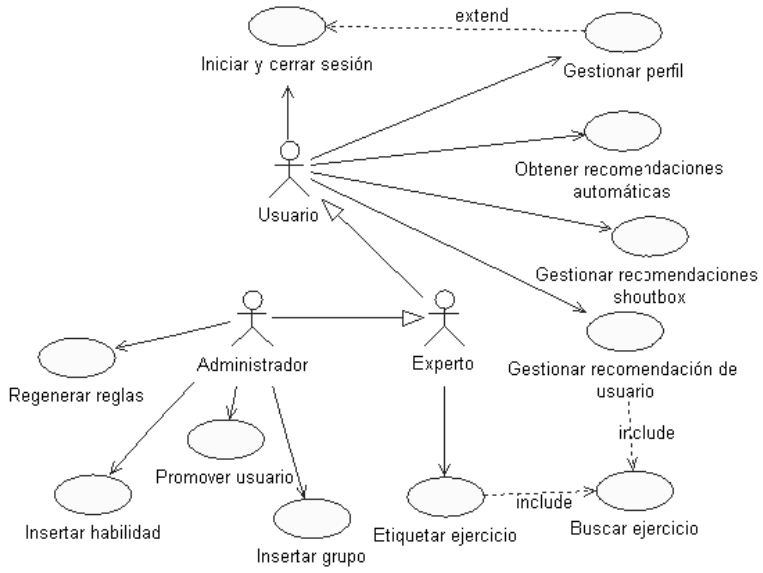


Figura 7. Diagrama de casos de uso del sistema

3.5 Modelo de análisis

El propósito del análisis es transformar los requerimientos del sistema en un formato que se corresponda con el área de conocimiento de los diseñadores del software. De una forma u otra, se encarga de que los requerimientos funcionales del sistema sean manipulables (61), ignorando con propósitos de simplicidad gran parte de los no funcionales, así como las restricciones del ambiente de implementación. Este constituye una abstracción del modelo de diseño.

El modelo de análisis contiene el resultado del análisis de los casos de uso, instanciados en el artefacto diagrama de clases de análisis. Este diagrama está conformado por las clases interfaz, clases control y clases entidad. A continuación se presentan algunos de los diagramas de clases de análisis asociados a los casos de uso del sistema en desarrollo.

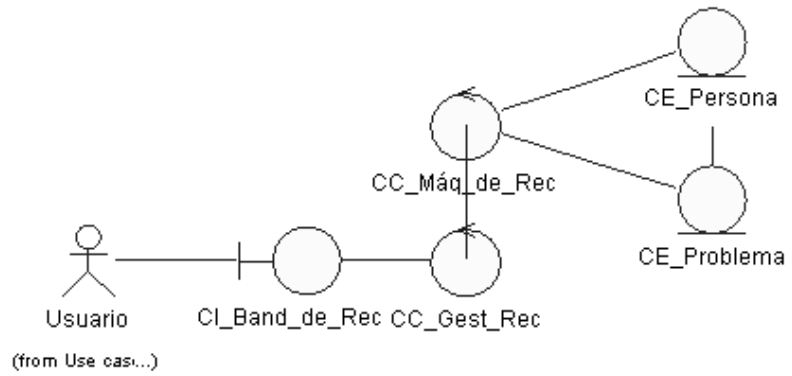


Figura 8. Diagrama de clases de análisis CU Obtener recomendaciones automáticas

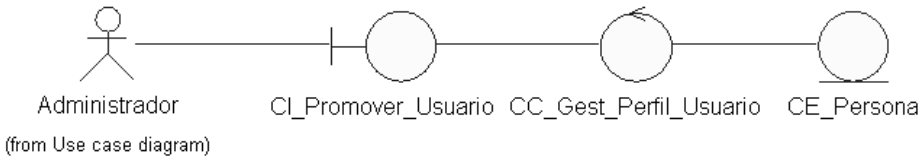


Figura 9. Diagrama de clases de análisis CU Promover usuario.

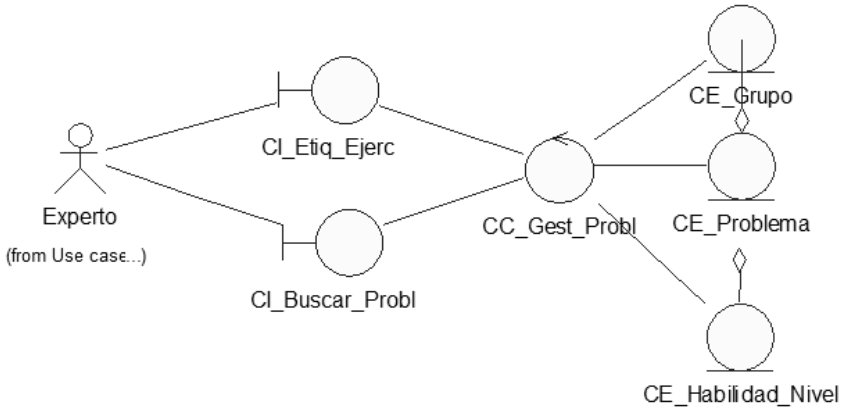


Figura 10. Diagrama de clases de análisis CU Etiquetar ejercicio.

En las figuras 8, 9 y 10 se muestran los diagramas de clases de análisis de tres de los casos de usos más significativos de la aplicación. El resto de los diagramas se muestran como parte de los anexos del trabajo.

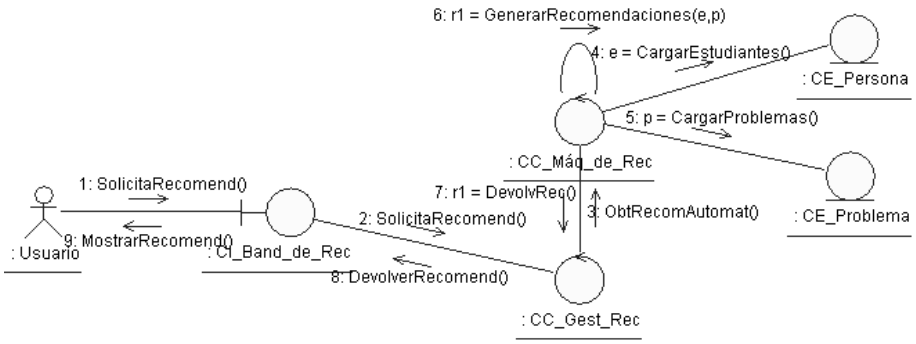
3.6 Diseño

El propósito principal del diseño es conjugar el resultado del análisis con las restricciones impuestas por los requisitos no funcionales, el ambiente de implementación, los requisitos de rendimiento, entre otros (61), dando como resultado un modelo preciso del producto final con un cubrimiento total de los requerimientos. A diferencia del análisis, que en muchos proyectos tiene carácter opcional, el diseño es considerado una disciplina clave dentro de la ingeniería de software, siendo el centro de atención al final de la fase de elaboración e inicio de la fase de construcción, y enfocándose en cómo los requisitos funcionales y no funcionales tienen impacto en el sistema a desarrollar.

El artefacto más importante dentro de la disciplina de diseño lo constituye el modelo de diseño, que no es más que un modelo de objetos que describe la realización de los casos de uso, y sirve como una abstracción del modelo de implementación y del código fuente. Más adelante se muestran varios artefactos de la aplicación en desarrollo intrínsecamente asociados a este modelo de diseño, como son diagramas de interacción y los diagramas de clases de diseño. Además, se muestra el modelo de datos y el modelo de despliegue, considerados también como parte de esta disciplina.

3.6.1 Diagramas de colaboración

El diagrama de colaboración es uno de los artefactos que se construye en la fase terminal del análisis, cuando se está comenzando a trabajar en el diseño de la aplicación. Este se encarga de representar las colaboraciones, en términos de mensajes, entre las diferentes clases del diagrama de clases de análisis. A continuación se presentan los diagramas de colaboración de algunos de los casos de uso en desarrollo. El resto de los diagramas se muestra como parte de los anexos del trabajo.



.

Figura 11. Diagrama de colaboración CU Obtener recomendaciones automáticas.

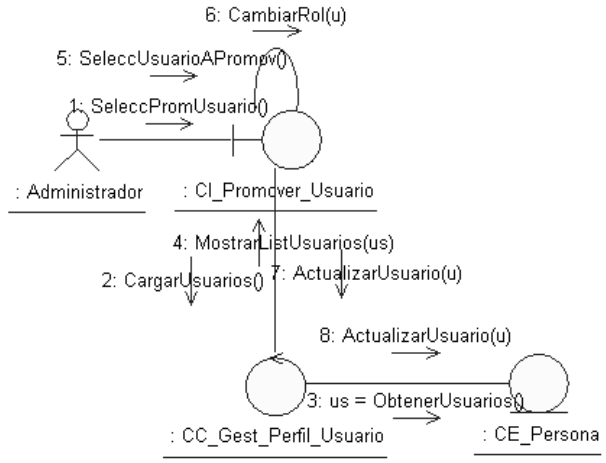


Figura 12. Diagrama de colaboración CU Promover usuario.

3.6.2 Aplicación de patrones de diseño. Arquitectura de la aplicación.

Normalmente los desarrolladores de aplicaciones acumulan un conjunto tanto de principios generales como de soluciones enfocadas en aplicar ciertos estilos que les guíen en el proceso de creación de software (42), recibiendo estos principios y estilos la denominación de patrones de software.

Dentro de los patrones de software revisten especial protagonismo los patrones de diseño. Según (62), un patrón de diseño es una descripción de clases y objetos comunicándose entre sí adaptada para resolver un problema de diseño en un contexto particular. Su objetivo es reutilizar, además de código, otros artefactos de los cuales se tiene certeza respecto a su correcto funcionamiento.

En el diseño de la aplicación se hace un uso intensivo de los patrones GRASP (42), destacándose entre ellos, la presencia del patrón **Experto**, del patrón **Controlador**, del patrón **Alta Cohesión** y del **Bajo Acoplamiento**.

Además se aplican una parte de los patrones GoF definidos en (62), entre los cuales se encuentran:

Singleton: Asegura que una clase tiene una única instancia, y proporciona un mecanismo global de acceso a dicha instancia.

Factoría: Define una interfaz con la responsabilidad de crear objetos. Permite que las clases que la usen sean las que decidan qué clase instanciar.
Fachada: Provee una interfaz única a través de la cual fluye la comunicación con el resto de las interfaces del subsistema.

Es importante señalar que a raíz de la tecnología que se emplea en el desarrollo, que es, como se explicó en el capítulo 1, la plataforma J2EE con su lenguaje Java, también se emplea patrones de diseño que son específicos de esta. Entre estos merecen mencionarse:

Service Locator: Define una interfaz que permite obtener instancias de servicios y componentes localizados en diferentes niveles de la aplicación y facilitar la creación del contexto inicial.
Data Access Object (DAO): Crea una interfaz para encapsular los accesos a las fuentes de datos.

Asimismo, en la aplicación son utilizados otros patrones de diseño que a pesar de no incluirse en ninguna categoría específica, juegan un papel clave dentro de las nuevas tendencias del desarrollo de este tipo de proyectos. Estos son el **Mapeo Objeto-Relacional (ORM)** y la **Inyección de dependencias**. Más abajo se abundará acerca de ellos.

Por otro lado, y a mayor nivel de abstracción, la definición de la arquitectura de software está condicionada por la utilización de frameworks que implementan de una forma u otra, los patrones de diseño mencionados más arriba.

El estilo arquitectónico seleccionado es la arquitectura en n-capas, particularmente una arquitectura de tres capas. Entre las facilidades que brinda este enfoque que justifican su elección, están el hecho de que brinda una abstracción total con respecto al origen de datos, garantiza un bajo costo de desarrollo y mantenimiento de las aplicaciones, facilita la reutilización del código, y provee de una mayor calidad al producto en general.

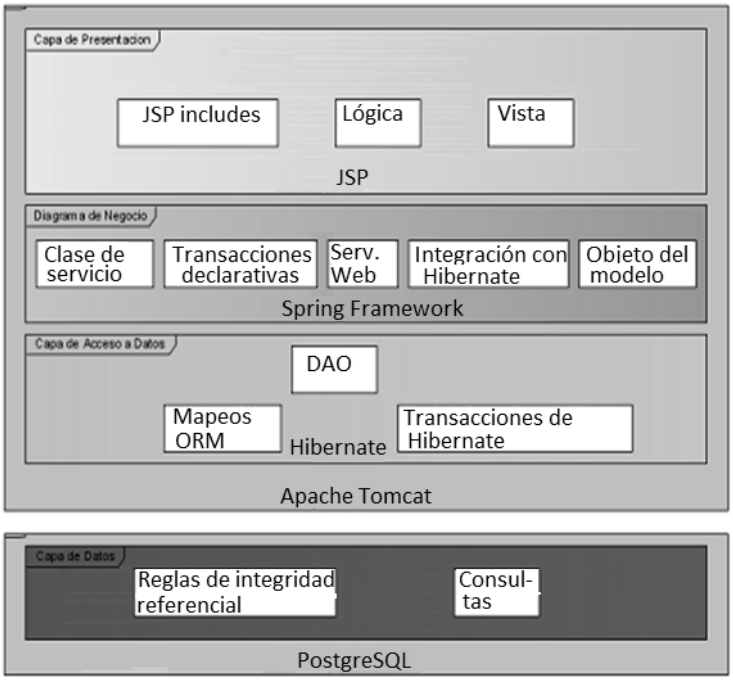


Figura 13. Arquitectura de la aplicación

En la figura se muestra un esbozo de la arquitectura de la aplicación, haciendo énfasis en el perfil tecnológico de desarrollo, presentado de manera independiente en el capítulo 1. En la capa de presentación la tecnología principal que se utiliza son las Java Server Pages, a través de las cuales se genera código HTML que es visualizado por el navegador. En la lógica del negocio se usa el framework Spring, que permite garantizar la presencia de un bajo acoplamiento como parte de la aplicación, con ayuda del patrón Inyección de dependencia. El uso de Spring además permite una futura extensión de la aplicación, al ser este un framework multi-módulo altamente configurable y con variadas funcionalidades como el soporte a la programación orientada a aspectos, los servicios web, y la seguridad de las aplicaciones. Finalmente el acceso a datos se implementa con la ayuda de Hibernate, que es un framework que implementa el patrón Mapeo Objeto-Relacional, consistente en la persistencia, automática y transparente, de objetos en una aplicación, hacia las tablas en una base de datos relacional, además de manejar la concurrencia y las transacciones de manera eficaz.

3.6.3 Sobre la seguridad de la aplicación

Considerando la sencillez de la aplicación en desarrollo y la escasa o nula existencia de información verdaderamente sensible, se decide prescindir de las opciones de Spring para la implementación de la seguridad.

Por tanto, esta se propone manejarla fundamentalmente de una manera ad-hoc. Tal y como anteriormente se ha expuesto, la aplicación contempla la existencia de tres roles principales: el de usuario estándar, el de etiquetador, y el de administrador. Este último es el encargado de promover o despromover a los usuarios hacia y desde los respectivos roles.

No obstante, no se dejaron de utilizar técnicas recomendadas para la concepción de la seguridad, particularmente la implantación de diferentes niveles de seguridad (65), tal y como quedó establecido en los requisitos no funcionales. En la aplicación el acceso a las diferentes funcionalidades se valida a la hora de generar las vistas a mostrar (JSPs), y también a nivel de invocaciones de método, garantizando que la aplicación no invoque ningún método con instrucciones que el usuario autenticado no tenga derecho a ejecutar.

3.6.4 Diagramas de clases de diseño

Una clase de diseño es una descripción de un conjunto de objetos que comparten las mismas responsabilidades, relaciones, operaciones, atributos y semántica (63). Estas se agrupan en los diagramas de clases de diseño.

En las figuras 14 y 15 se presenta un diagrama general de clases de diseño de la aplicación en desarrollo, así el diagrama de clases persistentes. Como parte de los anexos del trabajo, se muestra otro grupo de estos diagramas.

3.6.5 Modelo de datos

El modelo de datos está conformado por 18 tablas, primando las relaciones del tipo muchos-a-muchos. En la figura 16 se muestra íntegramente este modelo.

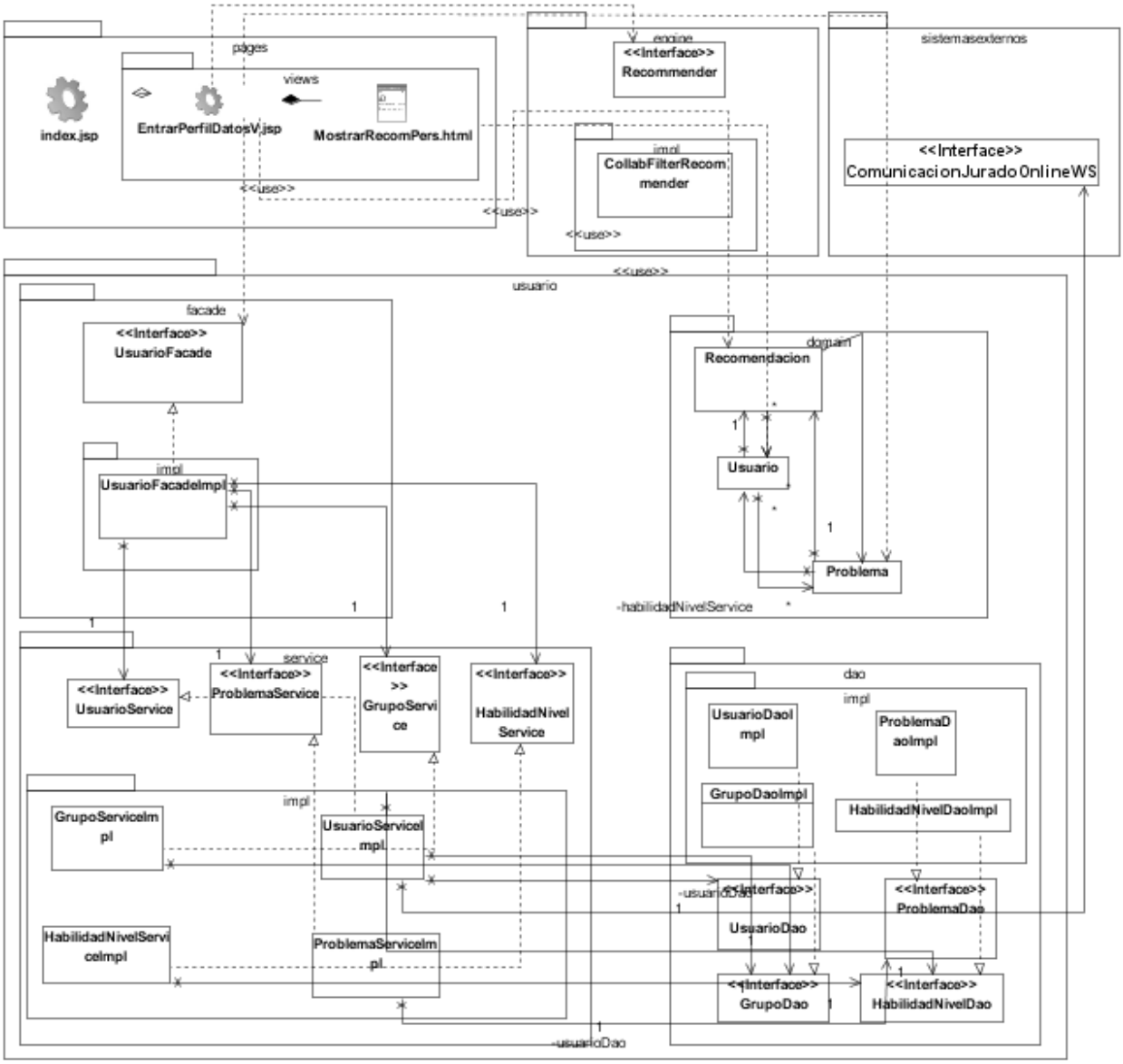


Figura 14. Diagrama de clases de diseño para el CU Obtener recomendaciones automáticas.

3.6.6 Diagrama de despliegue

Según (63), el modelo de despliegue captura los elementos de configuración del procesamiento y las conexiones entre estos elementos. También se utiliza para visualizar la distribución de los componentes de software en los nodos físicos, así como la disposición de los dispositivos que carecen de capacidad de pre-procesado.

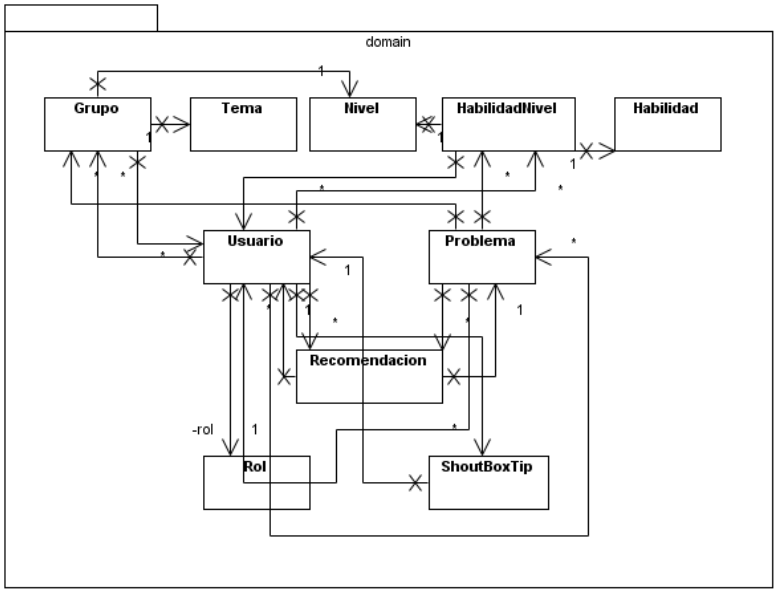


Figura 15. Diagrama de clases de diseño persistentes.

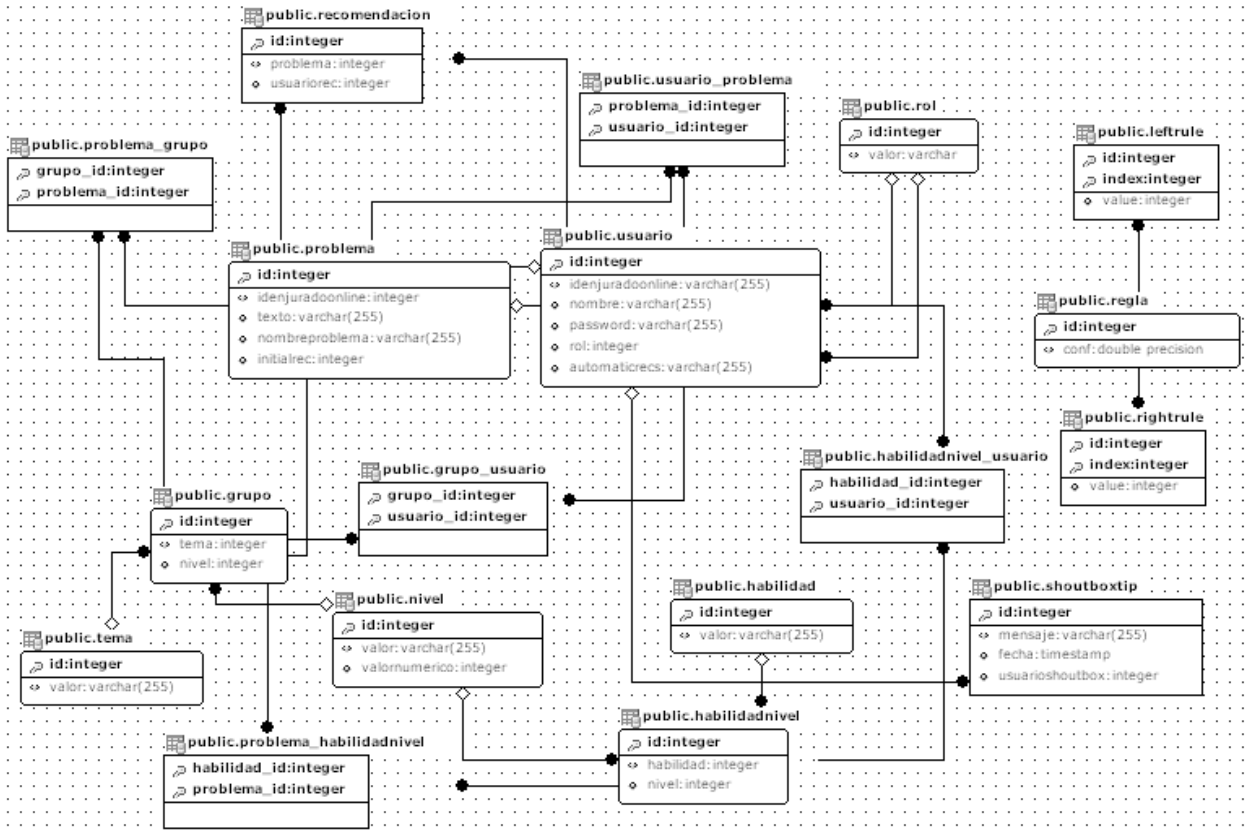


Figura 16. Modelo de datos del sistema en desarrollo.

Como parte del modelo de despliegue de la aplicación actual, el sistema de recomendación se concibe para ubicarse en un servidor independiente al del jurado online de programación, utilizándose un servicio web para garantizar la conectividad entre las dos aplicaciones. El usuario, desde su máquina local, puede acceder a ambas aplicaciones. Sin perder generalidad, la aplicación recomendadora puede ser también ser ubicada en el mismo servidor que el jurado online de programación al que le brindará soporte. En la figura 17 se muestra el diagrama de despliegue de la solución completa.

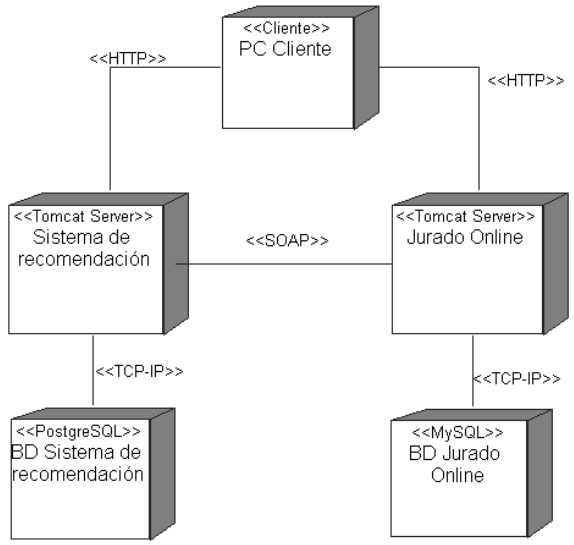


Figura 17. Diagrama de despliegue de la solución

3.7 Implementación

El propósito de la disciplina de implementación es definir la organización del código en términos de subsistemas de implementación organizados en capas. En esta, se toma como punto de partida el análisis y diseño, y se implementa el sistema en términos de componentes, entre los que están ficheros de código fuente, scripts, ficheros de código binario, y ejecutables.

3.7.1 Breve reseña

Para el desarrollo de la aplicación se siguió la estructura tradicional de paquetes, comúnmente utilizada por las aplicaciones Java. La mayoría de los archivos de código fuente se agrupan en un único módulo, que es el módulo usuario, que se divide a la vez en los paquetes facade,

service, dao, domain, config y test. Dentro de estos paquetes se localizan interfaces y las clases que constituyen sus respectivas implementaciones.

En el paquete facade se encuentra la fachada del módulo; este es el único punto de acceso al módulo en su conjunto, desde instancias externas a este. Esta fachada está compuesta por servicios, los cuales son declarados dentro del paquete service, y contienen el grueso de la lógica del negocio de la aplicación. Los servicios a su vez contienen en su interior DAOs, que son interfaces que implementan el patrón Data Access Object, y que encapsulan las invocaciones a la interfaz de comunicación del framework Hibernate. El paquete domain es el encargado de agrupar las clases persistentes de la aplicación, cuyos datos son almacenados con la ayuda del propio Hibernate. Finalmente, los archivos de configuración que permiten el funcionamiento correcto de toda la estructura están localizados dentro del paquete config. Estos archivos de configuración se clasifican en dos categorías principales: los archivos xml del framework Spring, encargados de garantizan la inyección de dependencias dentro de la aplicación asegurando un bajo acoplamiento, y los archivos hbm.xml de Hibernate, encargados de asociar las clases persistentes con tablas dentro de la base de datos, facilitando su almacenamiento. Los xmls de Spring también contienen la definición de la política de transacciones a seguir. Al paquete test se hará referencia en el próximo capítulo.

Además del módulo usuario, en la aplicación se concibieron otros dos paquetes, que son común y engine. Al paquete común se subieron funcionalidades de naturaleza genérica que serían comunes a todos los módulos del sistema. El paquete engine, por otra parte, contiene todo lo relacionado con el motor de recomendación.

Para la implementación de la capa de presentación, en el capítulo 1 se justificó la utilización de la tecnología JSP. Con propósitos organizativos, se creó una estructura de carpetas donde se diferencia las páginas dedicadas a mostrar datos o vistas, y las páginas dedicadas fundamentalmente a contener lógica. Las vistas son asociadas a la página principal a través de etiquetas <*jsp:include*>. En la figura 18 se muestra esta estructura.

Figura 18. Estructura de carpetas para la capa de presentación

Como parte de los anexos se muestran algunas pantallas de la aplicación.

3.7.2 Detalles de implementación del perfil algorítmico

La implementación del perfil algorítmico fue orientada completamente a la optimización, teniendo en cuenta las características de los procesos presentes.

Como parte de la implementación del método de filtrado colaborativo fueron construidos índices temporales utilizados para el acceso a los grupos, habilidades y también estudiantes insertados en la aplicación.

En el caso de determinados procedimientos caracterizados por ser de rendimiento crítico y de ejecución infrecuente, se decidió incorporarlos a la aplicación en calidad de tareas programadas. Tal es el caso de la regeneración de reglas de asociación, a pesar de el administrador poder realizar esta tarea también de manera manual.

Para la implementación de las tareas programadas, fueron utilizadas las clases *CronTriggerBean* y *SchedulerFactoryBean* que brinda el framework Spring.

3.7.3 Diagrama de componentes

A continuación se presentan algunos componentes resultado de la implementación, agrupados en diagramas de componentes. Otro grupo de estos se muestran como parte de los anexos.

```
<bean id="ruleGeneration" class="org.springframework.scheduling.quartz.JobDetailBean">
    <property name="jobClass" value="cu.rec.usuario.scheduling.RuleGenerationJob"/>
    <property name="jobDataAsMap">
    <map>
        <entry key="usuarioFacade" value-ref="usuarioFacade" />
        <entry key="optimizationFacade" value-ref="optimizationFacade" />
    </map>
    </property>
</bean>

<bean id="ruleGenerationTrigger" class="org.springframework.scheduling.quartz.CronTriggerBean">
    <property name="jobDetail" ref="ruleGeneration"/>
    <property name="cronExpression" value="0 23 20 ? * MON" />
</bean>

<bean class="org.springframework.scheduling.quartz.SchedulerFactoryBean">
    <property name="triggers">
        <list>
            <ref bean="ruleGenerationTrigger"/>
        </list>
    </property>
</bean>
```

Figura 19. Fragmento de archivo de configuración para la tarea programada RuleGenerationJob.

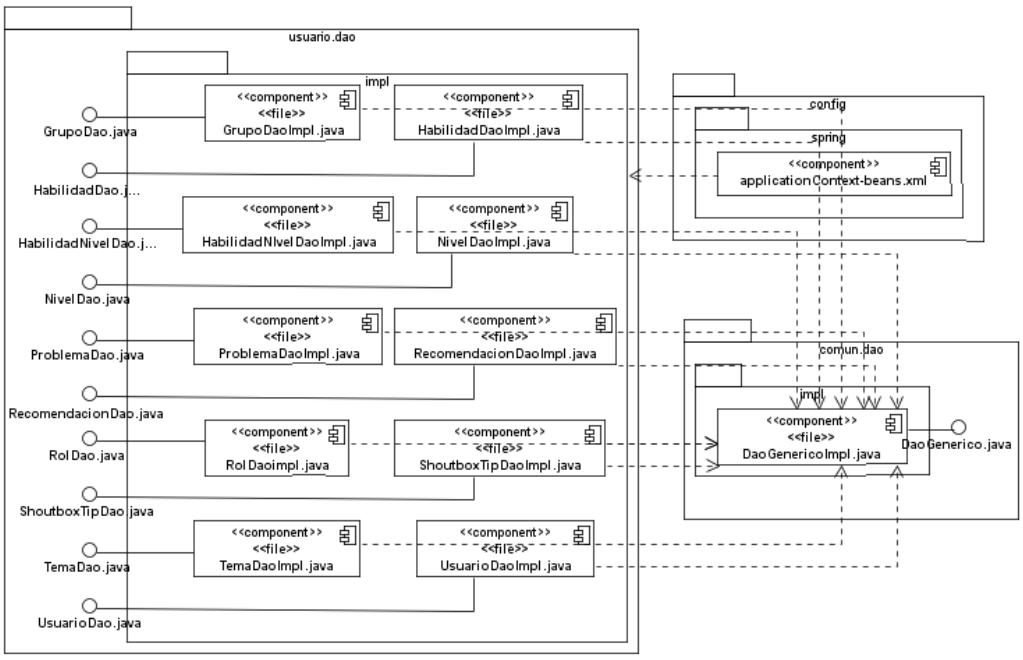


Figura 20. Paquete Dao Módulo Usuario. Diagrama de componentes con sus relaciones

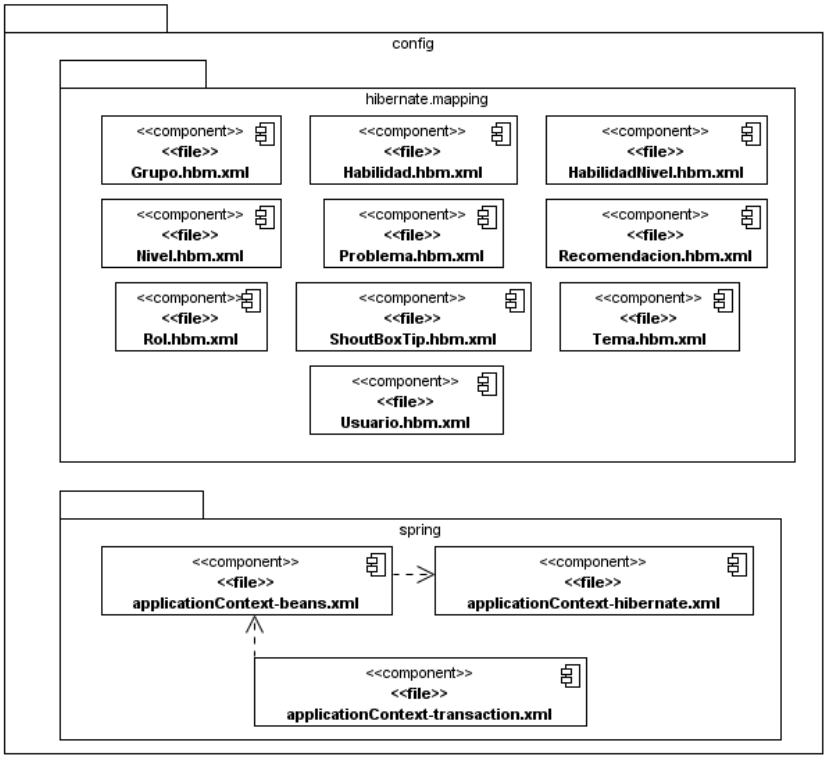


Figura 21. Paquete Config. Diagrama de componentes con sus relaciones

3.7.4 Sobre la comunicación con el jurado online

La aplicación obtenida se encuentra en constante comunicación con el jurado online de programación, permitiendo que a la hora de etiquetar problemas y de generar recomendaciones automáticas y de usuario al individuo autenticado, se tengan en cuenta los ejercicios que en ese preciso momento se encuentran en dicho jurado.

Esta comunicación se garantiza, como anteriormente ya se ha mencionado, a través de un servicio web cuya interfaz se presenta en la figura.

```
public boolean EsCorrectaLaConexion() throws Exception;
public List<Integer> obtenerListadoProblemasAceptados(String idUsuario) throws Exception;
public Integer obtenerCantidadProblemasAceptados(String idUsuario) throws Exception;
public boolean ValidarIDContrasennaContraJurado(String idUsuario, String pwd)throws Exception;
public List<String> consultarProblemas(String nombre);
public String obtenerNombreDadoIDProblema(int idProblema);
public List<String> obtenerListadoUsuariosDadoProblema(int idProblema);
```

Figura 22. Interfaz del servicio web de comunicación

Así, para acoplar la herramienta a cualquier jurado online de programación, basta con que este brinde un servicio web con los procedimientos especificados. La aplicación contiene la implementación de un cliente para consumir servicios web con esta firma. Es válido resaltar que, no obstante esta acoplabilidad, todo el trabajo se desarrolló utilizando el jurado de programación Xtreme, y como parte del mismo, se implementó un web-service con todas las funcionalidades demandadas.

La aplicación asimismo maneja los casos en que no haya comunicación con el jurado online, intentando minimizar las consecuencias de cara al usuario final, de este tipo de situaciones extraordinarias.

Conclusiones

En el capítulo se llevó a cabo el desarrollo del sistema de recomendación. Inicialmente se llegó la conclusión de la factibilidad de la elaboración de un modelo de dominio, por sobre un modelo de negocio. Posteriormente se mostró el diagrama de casos de uso del sistema, así como la especificación de cada uno de ellos. Luego se procedió a llevar a cabo los flujos de análisis, diseño e implementación, exponiéndose artefactos tales como diagramas de clases de análisis, diagramas de colaboración, y diagrama de clases de diseño, entre otros. Se hizo una selección de las mejores prácticas a aplicar dentro de cada una de estas disciplinas, siendo esta otra de las conclusiones importantes dentro del capítulo.

Validación

4.1 Introducción

El presente capítulo está dedicado a validar los resultados obtenidos en los capítulos anteriores. Se realiza una validación de la estrategia de recomendación aplicando métricas referenciadas por autores representativos de esta área de investigación. En adición se muestran los resultados de la realización, en diferentes escenarios y momentos del desarrollo, de las pruebas de software a la aplicación. El peso del flujo de trabajo de pruebas se centró en dos niveles fundamentales: las pruebas de unidad y las pruebas de sistema.

4.2 Evaluación de la estrategia de recomendación

Evaluar un algoritmo de recomendación constituye una de las tareas de mayor complejidad dentro del área de investigación y desarrollo que constituyen los sistemas de recomendación. Esto viene dado por el hecho de que resulta sumamente difícil, y en algunos casos hasta imposible, determinar si una estrategia determinada genera recomendaciones provechosas para los usuarios finales, debido a la infinita cantidad de perfiles distintos que pueden asumir cada uno de dichos usuarios. Por esta razón, muchos prestigiosos autores han priorizado, dentro de su línea de investigación, la obtención de métricas eficaces para evaluar a los sistemas de recomendación.

Para determinar la precisión de un sistema de recomendación, en la mayoría de los casos se han utilizado métricas que a pesar de provenir de otras áreas del conocimiento, tienen aplicación en el campo de este tipo de sistemas. Estas se dividen en dos grupos fundamentales: aquellas que están enfocadas a evaluar cuán cerca está la predicción, por parte de la aplicación, del *rating* que daría un usuario a un determinado *ítem* con respecto a la evaluación real dada por este usuario; y un segundo grupo que se dedica a medir 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 (66).

Dada las características del sistema en cuestión, las métricas pertenecientes al segundo grupo son las más apropiadas para realizar la evaluación de este. Estas son usualmente llamadas métricas de precisión de la clasificación.

A continuación se aplica una de estas métricas al listado de recomendaciones generadas por la combinación de los dos algoritmos del sistema de recomendación desarrollado, a través de un experimento offline (67), utilizando los datos disponibles en el jurado online de programación.

4.2.1 Aplicación de Precision and Recall

Precision and Recall (68) es una de las métricas más populares para evaluar los sistemas de recuperación de información. *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. Es considerada la métrica de precisión de la clasificación por excelencia a utilizar en los sistemas de recomendación.

La aplicación de *Precision and Recall* en este tipo de sistemas difiere ligeramente con respecto a la forma de aplicarla en otros marcos de propósito más general referidos en (68). Como parte de la evaluación offline, normalmente se tienen datos con los ítems que cada usuario ha utilizado o preferido. Para realizar la prueba, se selecciona un usuario específico, y se divide este conjunto de ítems, en este caso problemas resueltos, en dos subconjuntos, llamados conjunto de pre-recomendaciones, y conjunto de pruebas. Seguidamente al conjunto de pre-recomendaciones 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*.

Dentro de este contexto, el *Precision* y el *Recall* se definen de la siguiente manera (69):

$$recall = \frac{tamaño\ del\ hit\ set}{tamaño\ del\ conjunto\ de\ prueba}$$

$$precision\ = \frac{tamaño\ del\ hit\ set}{Cantidad\ de\ n - primeras - recomendaciones}$$

Una de las formas más tradicionales de representar los valores del *precision* y el *recall,* para un usuario determinado, es a través de curvas de *precision y recall* que se representan en el plano cartesiano y en las que por cada uno de los ejes X y Y se ubican respectivamente los valores de *precision* y *recall.*

Es importante resaltar que *precision* y *recall* usualmente entran en conflicto a la hora de la práctica. Normalmente, a la hora de incrementar el valor de n en las n primeras recomendaciones a obtener, aparece una tendencia a incrementarse el valor del *recall* y de disminuir el valor de *precision*. 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* (70), que es a su vez la seleccionada para utilizarse en el asunto actual. *F1* se define como:

$$F1 = \frac{2 * Recall * Precision}{(Recall + Precision)}$$

Para el algoritmo ad-hoc de filtrado colaborativo del sistema de recomendación obtenido, se ejecutó el cálculo de los valores de *recall*, *precision* y F1 para los 50 usuarios punteros del jurado online, considerados los de mejores criterios a la hora de seleccionar los problemas a resolver. El conjunto de pruebas, para cada uno de los usuarios, estuvo conformado por el 20% del total de sus ejercicios resueltos, seleccionados de manera aleatoria. Para cada uno de los usuarios se ejecutaron varias corridas en las que se varió, desde 5 hasta 25, el número N de recomendaciones a ser devueltas por el algoritmo de recomendación, con el objetivo de ver el comportamiento de los tres parámetros que se miden. En la figura se muestran algunos de los resultados de estas corridas.

Para poder comparar el sistema de recomendación con otros similares, no es suficiente con tener los valores de estos parámetros para cada uno de sus usuarios. Se hace necesario

obtener valores globales de *precision*, *recall* y F1 que describan el funcionamiento global del sistema, y no el comportamiento de este ante determinado usuario en específico.

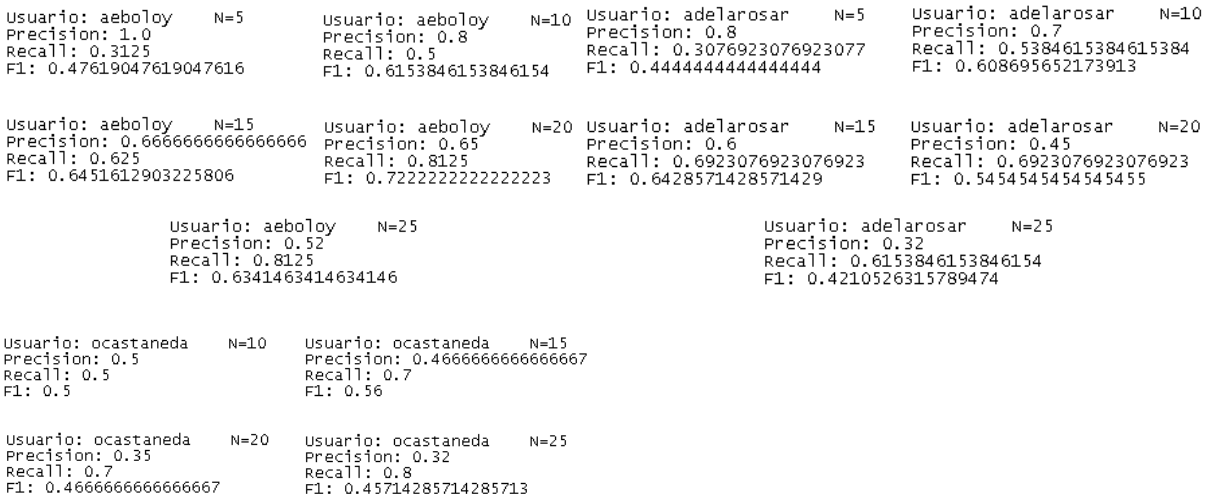


Figura 23. Valores de Precision, Recall y F1 para los usuarios aeboloy, adelarosar y ocastaneda

El cálculo de estos nuevos valores se realizó utilizando las sugerencias brindadas por (71), que señala computar los promedios de los valores de *precision* y *recall* de todos los usuarios para cada valor de N, y con sus promedios, calcular el F1. Los resultados de este proceso se muestran en la figura.

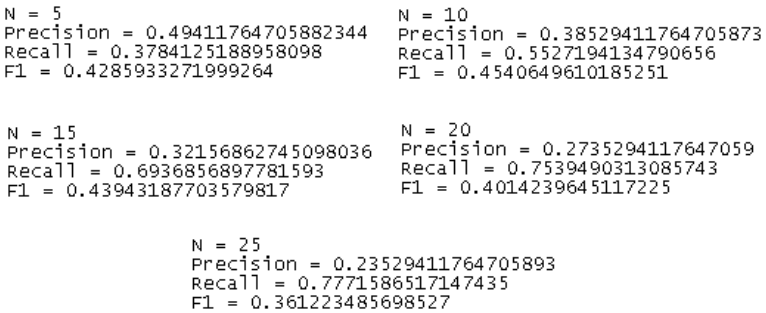


Figura 24. Valores globales de precision, recall y F1 para el sistema de recomendación para jurados online.

Es válido mencionar que los valores de Precision, Recall, y fundamentalmente F1 obtenidos de este último cálculo están en correspondencia con los obtenidos por otros sistemas recomendadores como (72) (73) (74), entre otros citables.

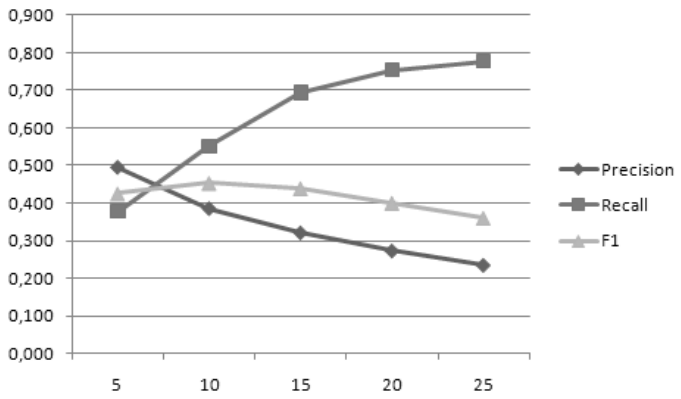


Figura 25. Representación gráfica de los valores globales de precision, recall y F1 para el sistema de recomendación.

Resulta razonable el desbalance entre los valores obtenidos, y los asociados a cada uno de los sistemas recomendadores mencionados. Esto viene dado por las características de las fuentes de datos utilizadas en cada caso, los algoritmos empleados, los objetivos finales del sistema en sí; y por estas razones, los sistemas no son comparables. No obstante, se referenciaron los datos con el objetivo de hacer ver que las recomendaciones brindadas por el sistema desarrollado poseen una calidad semejante a las que se pueden obtener de sistemas con un alto reconocimiento por la comunidad científica internacional. Este es uno de los resultados más importantes del trabajo.

Finalmente, es significativo también destacar que el resultado de que los mayores valores de F1 fueron obtenidos para las corridas en que se devolvieron de 5 a 10 recomendaciones, fue utilizado para determinar la cantidad de recomendaciones a mostrar al usuario en el caso de uso “Obtener recomendaciones automáticas”.

4.3 Pruebas de unidad

El nivel de pruebas de unidad, que puede orientarse tanto a caja blanca como a caja negra, está dirigido hacia los componentes testeables más pequeños del software. Es aplicable a

componentes representados en el modelo de implementación para verificar que los flujos de control y de datos estén cubiertos, y que funcionen como se espera. En el contexto orientado a objetos, la menor unidad a probar es la clase u objeto encapsulado. Una clase puede contener un cierto número de operaciones, y una operación en particular puede existir como parte de un número de clases diferentes. Esta prueba de clases para el software orientado a objetos, es equivalente a las pruebas de unidad para el software convencional.

Para la realización de las pruebas de unidad a la aplicación en desarrollo se seleccionó la herramienta JUnit, considerada por la mayoría de los expertos como la de mayor productividad en este campo (64), sobre la plataforma J2EE.

Un caso de prueba, en su concepto más general, está conformado por un conjunto de entradas de pruebas, condiciones de ejecución y resultados esperados desarrollados para cumplir un objetivo en particular o una función esperada (63). En el contexto de JUnit, un caso de prueba se representa, extendiendo de la clase *TestCase.*

El método más frecuente a utilizar para la implementación de estas pruebas es la creación de una clase de prueba por cada clase de servicio del módulo en cuestión. Dentro de esta se crea un método de prueba por cada método del servicio original, ubicándose dentro de este las diferentes aserciones que validan la correctitud estructural y lógica del servicio. Por otro lado, al extender de *TestCase*, las clases de prueba han de implementar los métodos *setUp* y *tearDown*, ejecutándose el primero siempre al principio de la invocación de cada uno de los métodos de prueba, y el segundo al final de las invocaciones. Normalmente *setUp* se utiliza para establecer un contexto sobre el cual los métodos de prueba puedan ejecutarse, mientras que *tearDown* se destina fundamentalmente a labores de limpieza y de liberación de recursos (64).

Para la concepción de los casos de prueba se tuvieron en cuenta varios de los patrones para el desarrollo guiado por pruebas (*Test Driven Development*) explicados en (65), a pesar de esta filosofía no tener relación directa con la metodología de desarrollo empleada.

Entre los patrones aplicados están el *One Step Test,* que establece criterios para definir el orden de realización de los casos de prueba indicando que, aunque no existe una guía predefinida para esto, el siguiente caso de prueba a ejecutar debe ser siempre aquel que más información aporte, y sobre el que se tenga confianza que se puede implementar de manera correcta.

También se siguió la línea del *Regression Test,* que predica que la primera acción a llevar a cabo una vez que un defecto es reportado, es escribir la más pequeña posible prueba, asociada a este defecto y que falle, y luego, a partir de esta prueba, comenzar las tareas de corrección de los errores.

Finalmente, como otro de los patrones más utilizados puede mencionarse el *Mock Object*, el que sugiere que en los casos en que la obtención de un objeto, necesario para las pruebas, sea excesivamente costosa, se establezca una versión falsa del mismo que externamente se comporte como su contraparte original y se desarrolle el caso de prueba utilizando dicha versión.

Como parte de los anexos se muestran dos de las clases de prueba creadas como parte del proceso de realización de pruebas de unidad con la herramienta JUnit. Es válido aclarar que en el proyecto en desarrollo este nivel de pruebas se realizó a demanda, priorizándose la implementación de pruebas de unidad en aquellas partes del código más propensas a errores.

4.4 Pruebas de sistema

Las pruebas de sistema son aquellas que se hacen cuando el software ya está funcionando como un todo (63). Dentro de las pruebas de sistema se encuentran las pruebas de recuperación, las pruebas de seguridad, las pruebas de resistencia, y las pruebas de rendimiento.

Las pruebas de rendimiento a su vez se clasifican en varias categorías, entre las que están las pruebas de contención, que se enfocan en la capacidad del elemento a probar para manejar de manera aceptable la demanda de múltiples actores, y las pruebas de carga y estrés, las que se usan para validar y valorar la aceptabilidad de los límites operacionales de un sistema bajo carga variable, simulándose la carga de trabajo a la que está sometido en tiempo real., así como para detectar posibles errores que sólo suelen aparecer en situaciones críticas.

Dadas las características del producto que se desarrolla de tener un intercambio continuo con el usuario, se decide la ejecución de pruebas de carga como actividad prioritaria para realizar la validación del mismo.

Además, también se realizaron pruebas de aceptación en busca de posibles *bugs* del software.

4.4.1 Pruebas de carga

Realizar una prueba de carga consiste en exponer un sistema a un uso continuo y constante, controlado estadísticamente (66). Las dos principales razones por las cuales se realiza este tipo de prueba son para dar soporte a las pruebas de confiabilidad del software, y para asegurar de una manera efectiva, la eficacia de los resultados de todas las pruebas de rendimiento. En este tipo de pruebas, la carga suele ser variada desde una cantidad mínima cercana a cero, llegando hasta los valores que el sistema en términos de recursos pueda sostener.

Para la realización de las pruebas a la aplicación en cuestión se seleccionó la herramienta privativa WAPT 6.0, la que provee una manera consistente y eficiente de probar aplicaciones con interfaces web, permitiendo analizar las características de su rendimiento bajo niveles variados de carga.

Este software brinda la posibilidad de, a través de un navegador interno que posee, realizar interacciones con la aplicación web a probar y grabarlas en calidad de casos de prueba, asociados a un perfil de usuario determinado. Una vez que se tiene bien definido las acciones asociadas a un perfil de usuario, WAPT automatiza la prueba de carga al crear usuarios virtuales que ejecutan sucesivamente esas mismas tareas con un período y una frecuencia también especificada.

Las pruebas de carga se realizaron sobre un hardware y un software muy semejante al que se utilizará en el despliegue final del producto, coincidiendo también con el que se dispuso para el desarrollo. Se utilizó como plataforma de Java la JDK 1.6, seleccionando como servidor de aplicaciones el Apache Tomcat 6.0. El ordenador sobre el que se ejecutó la prueba contó un procesador Intel Core 2 Duo 2.20 Ghz y con una memoria RAM de 1 Gb.

Para conformar el plan de pruebas de carga se decidió utilizar las páginas “Ver recomendaciones automáticas” y “Ver mensajes ShoutBox”, por ser dos de las más visitadas dentro de la aplicación.

Para el caso de “Ver recomendaciones automáticas” se creó un grupo de hilos de 20 usuarios, realizando peticiones de manera continua durante dos minutos sobre la aplicación,

computándose un total de 7560 peticiones. El tiempo promedio de respuesta, en el caso más crítico durante el tiempo de prueba, fue de *12,7 s*, tal y como se aprecia en la figura.

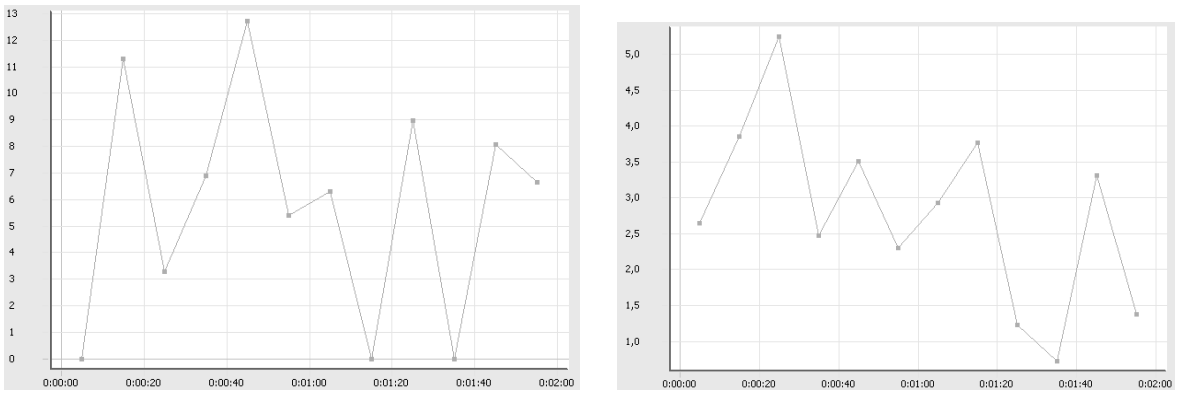


Figura 26. Tiempo de respuesta promedio para la página "Ver recomendaciones automáticas" y "Ver recomendaciones shoutbox" respectivamente

Considerando el volumen de procesamiento asociado a la página, el perfil tecnológico utilizado en su construcción, el tráfico real que tendrá la aplicación en tiempo, así como el resultado de la ejecución de este tipo de pruebas en aplicaciones semejantes, se considera exitosa la realización de la prueba.

Para el caso de "Ver mensajes ShoutBox", se corrió una prueba con características similares, siendo en este caso el tiempo promedio más crítico, de unos *5,24 s.* Esta igualmente se califica de exitosa.

4.4.2 Casos de prueba de aceptación

Como completamiento del paquete de pruebas a realizar a la aplicación, se definieron casos de prueba para aquellos escenarios en que ocurran interacciones bidireccionales con el usuario. Se realizaron pruebas con un equipo seleccionado conformado por usuarios del jurado online anteriormente ya utilizado en otro tipo de pruebas. Como parte de los anexos se presentan estos casos de prueba, así como el resultado final de las mismas.

Conclusiones

En el presente capítulo se realizó la validación del sistema de recomendación para jurados online de programación, trabajándose en dos vertientes principales: el perfil algorítmico de la aplicación, y el software como tal, en fase de despliegue. Para la evaluación del perfil algorítmico se utilizó la métrica *Precision and Recall* para la evaluación del algoritmo ad-hoc de filtrado colaborativo y del algoritmo basado en reglas. Con esta se obtuvieron resultados satisfactorios. Por otro lado, en el caso del software, se realizaron pruebas de unidad que permitieron corregir a tiempo todos los defectos garantizando un completamiento adecuado de cada uno de los componentes concebidos para la aplicación. Finalmente se hicieron pruebas de sistema, que consistieron en pruebas de carga a la aplicación, las que arrojaron resultados semejantes al de otras aplicaciones con el mismo perfil tecnológico y algorítmico; y también se realizaron pruebas de aceptación, las que también fueron exitosas.

Conclusiones

La concepción y desarrollo de un sistema de recomendación para jurados online de programación posibilitó arribar a las siguientes conclusiones:

- La realización de un estudio del estado del arte de los sistemas de recomendación y de los jurados online permitió afirmar que los jurados online de programación constituyen un área dentro del e-learning que demanda la incorporación de herramientas recomendadoras con vistas a mejorar el aprovechamiento de sus usuarios. Sin embargo los disponibles actualmente no cumplen las expectativas de los usuarios de estos jurados online.
- Se definió el perfil algorítmico que se incluyó en la solución, con un enfoque híbrido, a través del estudio de varios algoritmos posibles a aplicar.
- Siguiendo el flujo de trabajo definido en RUP se realizó el desarrollo completo de un sistema recomendador capaz de ser integrado a cualquier jurado online, aumentando sus posibilidades de uso.
- Tras obtener la herramienta se realizaron varias pruebas tanto de su perfil algorítmico como del software en sí, obteniéndose índices aceptables en cada una de ellas, por lo que se concluye un resultado general positivo.

Como producto final del trabajo se obtuvo un perfil algorítmico del sistema de recomendación, y una herramienta que en comunicación con un jurado online de programación, sea capaz de recomendar ejercicios a los usuarios del mismo. Esta herramienta además, tiene otros valores añadidos.

Recomendaciones

Entre las recomendaciones enfocadas a la continuidad del trabajo están:

- Mejorar la respuesta de la estrategia de recomendación ante escenarios de arranque en frío o *cold-start*.
- Integrar la herramienta a otros jurados online de programación distintos del jurado Xtreme, y realizar la medición de los mismos parámetros validadores.
- Incorporar nuevos servicios de valor agregado a la herramienta, enfocados en facilitar aún más la comunicación entre usuarios.

Referencias bibliográficas

1. *IntroductionTo Recommender Systems: Algorithms and Evaluations.* **Konstan, Josep.** 2004, ACM Transactions on Information Systems, pp. 1-4.

2. *Beyond recommender systems: Helping people help each other.* **Terveen, L and Hill, W.** 2001, HCI in the New Millennium, pp. 487-509.

3. *GroupLens: An Open Architecture for Collaborative Filtering of Netnews.* **Resnick, Paul.** 1994. Proceedings of ACM 1994 Conference on Computer Supported Cooperative Work. pp. 175-186.

4. *Recommending and evaluating choices in a virtual community of use.* **Hill, W.** 1995. Proceedings of CHI'95.

5. *The Long Tail.* **Anderson, Chris.** 2004, Wired.

6. **Adomavicius, G. and Tuzhilin, A.** *Recommendation Technologies: Survey of current methods and possible extensions.* University of Minnesota : MISRC working paper 0329, 2003.

7. *Preference Learning in a Recommender System.* **De Gemmis, Marco and Iaquinta, Leo.** Bled, Slovenia : s.n., 2009. European Conference on Machine Learning and Principle and Practice of Knowledge Discovery in Databases.

8. *A recommender system to support the scholarly communication process.* **Rodríguez, Marko and Allen, David.** 2009, The Computing Research Repository.

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

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

11. **Pazzani, Michael and Billsus, Daniel.** Content-Based Recommendation System. *The Adaptive Web: Methods and Strategies of Web Personalization.* s.l. : Springer Berlin / Heidelberg, 2007.

12. *Content-based book recommending using learning for text categorization.* **Mooney, Raymond and Roy, Loriene.** 2000. DL '00: Proceedings of the fifth ACM conference on Digital libraries . pp. 195-204.

13. *Introducing Serendipity in a Content-Based Recommender System.* **Iaquinta, L. and de Gemmis, M.** 2008. Proceedings of Eighth International Conference on Hybrid Intelligent Systems. pp. 168-173.

14. *Using Collaborative Filtering to weave an information tapestry.* **Goldberg, D and Nichols, D.** 1992, Communication of the ACM, pp. 35-60.

15. *Pointing the Way: Active Collaborative Filtering.* **Maltz, D. and Ehrlich, K.** 1995, Proceedings of CHI'95, pp. 202-209.

16. *Amazon.com Recommendations: Item-to-Item Collaborative Filtering.* **Linden, Greg, Smith, Brent and York, Jeremy.** 2003, IEEE Internet Computing, pp. 76-80.

17. *Implementations of Web-based Recommender Systems Using Hybrid Methods.* **Sobecki, Janusz.** 2006, International Journal of Computer Science & Applications, pp. 52-64.

18. *Recommender Systems in E-Commerce.* **Ben Schafer, J and Konstan, J.** 1999. ACM Conference on Electronic Commerce. pp. 158-166.

19. E-learning. *es.wikipedia.org.* [Online] 2008. [Cited: 2 19, 2010.] http://es.wikipedia.org/wiki/E-learning.

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

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

22. *Smart recommendations for an evolving e-learning system.* **Tang, Y and McCalla, G.** 2003. Workshop on Technologies for Electronic Documents for Supporting Learning.

23. *El Jurado Online, una nueva forma de ejercitación y evaluación en las asignaturas de Programación.* **Junco Vázquez, Tomás Orlando and Matías González, Héctor.** 2009. p. XIII Congreso Internacional de Informática Educativa.

24. **Jacobson, Ivar, Booch, Grady and Rumbaugh, James.** *El proceso unificado de desarrollo.* s.l. : Addison-Wesley, 2000.

25. **Beck, Kent.** *Extreme Programming Explained.* s.l. : Notes, 1999.

26. XP vs Scrum. *FEROLEN.* [Online] abril 15, 2008. [Cited: febrero 22, 2010.] http://www.ferolen.com/blog/xp-vs-scrum/.

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

28. **Stewart, Celeste.** The advantage of PHP. *Designer's Playground.* [Online] enero 3, 2006. [Cited: febrero 22, 2010.] http://www.designersplayground.com/articles/118/1/The-Advantages-of-PHP/Page1.html.

29. **Mah, Paul.** Disadvantages of Web Development Using PHP. *ITBusinessEdge.* [Online] junio 15, 2009. [Cited: febrero 22, 2010.] http://www.itbusinessedge.com/cm/blogs/mah/disadvantages-of-web-development-using-php/?cs=33397.

30. **García de Jalón, Javier.** *Aprenda Java como si estuviera en primero.* San Sebastián : s.n., 2000.

31. **Otero, Abraham.** MySQL vs PostgreSQL ¿cuándo emplear cada una de ellas? *javaHispano.* [Online] septiembre 10, 2007. [Cited: febrero 22, 2010.] http://www.javahispano.org/contenidos/es/mysql_vs_postgresql_cuando_emplear_cada_una_de_ellas_11/.

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

33. **Smith, Benji.** Why I Hate Frameworks. *Joel On Software.* [Online] septiembre 30, 2005. [Cited: febrero 22, 2010.] http://discuss.joelonsoftware.com/default.asp?joel.3.219431.12.

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

35. **Schmelzer, Robert.** PHP vs Java: The Competition Part 1. *PHP vs Java: The Competition Part 1.* [Online] noviembre 20, 2006. [Cited: febrero 22, 2010.] http://blog.schmelzer.cc/archives/7-PHP-vs.-Java-The-Competition-Part-I.html.

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

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

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

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

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

41. **Fussell, Mark.** *Foundations of Object-Relational Mapping.* 10 10, 2007.

42. **Larman, Craig.** *UML y Patrones.* s.l. : Prentice-Hall, 1999.

43. *Herramienta de filtrado colaborativo como complemento a un jurado online de programación.* **Yera Toledo, Raciel and Junco Vázquez, Tomás Orlando.** La Habana, Cuba : UCIENCIAS, 2010.

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

45. **Mitchell, Tom.** *Machine Learning.* s.l. : McGraw-Hill, 1997.

46. *User modeling for adaptive news access.* **Billsus, D and Pazzani, M.** 2000.

47. *Aproximando a los sistemas recomendadores desde los algoritmos genéticos.* **Vélez Lags, Oswaldo and Santos, Carlos.** Bogotá, Colombia : s.n., 2005, Revista Colombiana de Computación, pp. 7-23.

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

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

50. **Pearl, J.** *Probabilistic Reasoning in Expert Systems: networks of plausible inference.* San Francisco : Morgan Kauffman Publishers, 1988.

51. *Maximum likelihood estimation from incomplete data.* **Hartley, H.** s.l. : Biometrics, 1958, Vol. 14, pp. 174-194.

52. *Approximating Discrete Probability Distributions with Dependence Trees.* **Chow and Liu.** 1968, IEEE TRANSACTIONS ON INFORMATION THEORY, pp. 462-467.

53. *Item-based collaborative filtering recommendation algorithm.* **Sarwar, Badrul, et al.** Hong-Kong : s.n., 2001. WWW10.

54. *Clustering Methods: A History of k -Means Algorithms.* **Bock, Hans-Hermann.** 2007, Selected Contributions in Data Analysis and Classification, pp. 161-172.

55. *An Adaptive Recommendation System without Explicit Acquisition of User Relevance Feedback.* **Shahabi and Chen.** [ed.] Springer Netherlands. 2003, Distributed and Parallel Databases.

56. *Horting hatches an egg: a new graph-theoretic approach to collaborative filtering.* **Aggarwal, et al.** San Diego, Califormia : s.n., 1999, Proceedings of the fifth ACM SIGKDD international conference on Knowledge discovery and data mining.

57. *Getting to know you: learning new users preferences in recommender systems.* **Rashid, Albert and Cosley.** 2002, Proceedings of Intelligence Conference Intelligent user interfaces .

58. *Learning Collaborative Information Filtering.* **Billsus, D. and Pazzani, M.** 1998, Proceedings of Intelligence Conference Machine Learning.

59. *Fast Algorithms for Mining Association Rules.* **Agrawal, Rakesh and Srikan, Ramakrishnan.** Santiago de Chile : s.n., 1994. Vol. Proceedings of the 20th VLDB Conference.

60. **Pazzani and Billsus.** Content-based Recommendation Systems. *The adaptive web.* s.l. : Springer-Verlag, 2007.

61. Modelo de Dominio. [Online] marzo 22, 2010. http://iie.fing.edu.uy/ense/asign/desasoft/practico/hoja8/ejemplos_clase2.pdf.

62. **Nielsen, Jakob.** useit.com. [Online] [Cited: marzo 15, 2010.] http://www.useit.com/papers/heuristic/heuristic_list.html.

63. **Krutchen, Philippe.** *The Rational Unified Process: An Introduction.* s.l. : Addison-Wesley, 2003.

64. **Gamma, Erich, et al.** *Design Patterns: Elements of Reuseable Object-Oriented Software.* 1995.

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

66. *Evaluating Collaborative Filtering Recommender Systems.* **Herlocker, Jonathan, et al.** 1, 2004, ACM Transactions on Information Systems, Vol. 22.

67. *A Survey of Accuracy Evaluation Metrics of Recommendation Tasks.* **Gunawardana and Shani.** 10, 2009, Journal of Machine Learning Research.

68. *Measures.* s.l. : NIST. The Tenth Text REtrieval Conference (TREC 2001).

69. **Cristache, Alex.** *Hybrid recommender system using association rules.* Auckland, New Zealand : s.n., 2009.

70. *A Re-examination of Text Categorization Methods.* **Yang, Y and Liu, X.** 1999.

71. *Evaluating Recommendation Systems.* **Shani and Gunawardana.** s.l. : Microsoft Research, 2009.

72. *Performance of Recommendation Systems in Dynamic Streaming Environments.* **Nasraoui, Olfa, et al.** 2007.

73. *Collaborative Filtering Recommender Systems based on Popular Tags.* **Liang, Xu and Li.** Sydney, Australia : s.n., 2009.

74. *Recommender System based on Collaborative Behavior of Ants.* **Bedi, Sharma and Kaur.** 2009, Journal of Artificial Intelligence.