UNIVERSIDAD DE LAS CIENCIAS INFORMÁTICAS

Componente para la generación de perfiles de preferencias en el SRI Orión

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

Autores:

Yossan Alejandro Rodríguez Nuñez

Tutores:

MSc. Hubert Vitres Sala

Ing. Yaili Ledea Velázquez

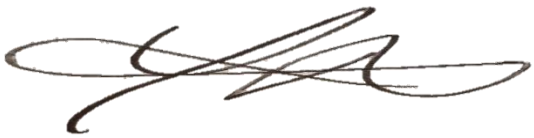
La Habana, junio 2021

“Año 63 de la Revolución”

Declaración de Autoría

Declaro por este medio que yo: **Yossan Alejandro Rodríguez Nuñez,** con carnet de identidad **97091011861** soy el autor principal del trabajo titulado **"Componente para la generación de perfiles de preferencia en el SRI Orión"** y autorizo a la Universidad de Ciencias Informáticas a hacer uso de la misma en su beneficio, así como los derechos patrimoniales con carácter exclusivo.

Para que así conste firmamos la presente a los _04_ días del mes de _Diciembre_ de _2021_

Autor: Yossan Alejandro Rodríguez Nuñez Firma: 

Tutor: MSc. Hubert Vitres Sala Firma: 

Tutor: Ing. Yaili Ledea Velázquez Firma: 

Resumen

Con el objetivo de recuperar y visualizar la información alojada en la web cubana se desarrolló en el país una herramienta de recuperación y visualización llamada Orión. Como deficiencias fundamentales de este buscador se encuentran la dificultad para identificar y comprender la necesidad del usuario escrita en lenguaje natural, la poca precisión y exactitud de los resultados de búsqueda obtenidos, posee limitaciones para recuperar documentos relevantes al introducir términos exactos o ambiguos y al calcular la similitud por la frecuencia de aparición de las palabras clave se recuperan grandes volúmenes de documentos con poca precisión y exactitud. Con el exponencial crecimiento de la web se dificulta el proceso de recuperación de información. Para mejorar este proceso se utilizan varias técnicas, tales como: la expansión y desambiguación de consultas, el uso de bases de conocimiento (ej. Taxonomías, Ontologías, etc.) y la generación de perfiles de preferencia de usuarios. Para darle solución a la problemática anterior se diseña un componente para la generación de perfiles de preferencia de usuarios para el sistema de recuperación de información Orión, haciendo modelado de perfiles de usuario con sus gustos, preferencias e intereses que son la representación interna de sus necesidades de búsqueda. Los aportes fundamentales de la investigación se centran en que el modelado de dichos perfiles de usuarios facilite la recuperación de información relevante esparcida en la web con respecto a las preferencias del usuario disminuyendo el tiempo necesario para encontrar la información y produciendo resultados más precisos y útiles para él mismo.

Palabras Claves: recuperación de información, perfiles de preferencias de usuarios, componente, sistema de recuperación de información

Índice

Introducción

Un perfil es un modelo de un objeto, una representación compacta que describe las características más importantes, creado en la memoria de un ordenador, comprensible por computadores y que es utilizado como representante del objeto. Según Farnadi et al. (2018) la mayor parte del perfilado (*profiling*) está orientada al usuario como organismo principal y más complejo de un sistema. Como ya se ha mencionado, de una forma más genérica, un perfil de usuario es un modelo de usuario creado en la memoria del ordenador, que es utilizado como representante del usuario en las tareas computacionales. El perfil de usuario contiene información modelada sobre el usuario, representada explícita o implícitamente, y cuya explotación permite al sistema incrementar la calidad de sus adaptaciones.

Con relación al dominio, un perfil se compone de las siguientes partes: una parte independiente del dominio, como son el nombre, la edad, el nivel educativo, etc.; otra dependiente del dominio, como por ejemplo los intereses del usuario, sus conocimientos en una determinada área, etc.; y una tercera, que no cae en la estricta definición de ninguna de las dos anteriores, como son las preferencias del usuario, el ambiente, etc. Otra forma de clasificar el perfil de usuario, es considerar su comportamiento con relación al cambio. Alguna información del perfil de usuario es estática, como la fecha de nacimiento, el nombre, etc.; y normalmente es introducida manualmente, de una sola vez por el usuario. Otra sin embargo es dinámica, como por ejemplo los intereses del usuario, que cambian y por consiguiente es aconsejable que sean determinados automáticamente.

El perfil de preferencias de usuarios es un perfil de usuario que se centra principalmente en la característica preferencias del usuario. Por distintas razones los usuarios pueden preferir unos enlaces o nodos en detrimento de otros, y unas partes de páginas en detrimento de otras. Estas preferencias pueden ser absolutas o relativas, esto es, dependen del nodo actual, del objetivo y del contexto actual en general (Cruz, 2003). Las preferencias del usuario difieren de otros componentes del perfil de usuario en distintos aspectos. Al contrario de los otros componentes, las preferencias no pueden ser deducidas enteramente por los sistemas, el usuario tiene que informar directa o indirectamente al sistema de sus preferencias.

En Rodríguez (2016) se plantea que el modelado de perfiles de usuario es una técnica para la recomendación de contenido personalizado. Sin la generación de los perfiles de usuarios, que son la representación interna de sus necesidades de búsqueda, se hace difícil el proceso de recuperación de

información. Esto afecta directamente el conjunto de respuesta que se brinda, debido a que no se adaptan los resultados al usuario ni se ponderan aquellos documentos que podrían tener un mayor significado (Rodríguez, Duque, & Ovalle, 2016). Aciar y Aciar (2013) manifiestan que, para crear un perfil de usuario con sus gustos, preferencias e intereses, se puede utilizar dos formas o métodos de la recolección de información: explícita o implícita.

La información explícita es la que el usuario ingresa al sistema en respuesta a peticiones manifiestas del mismo. Por ejemplo, solicitar al usuario que pondere un conjunto de ítems de una lista de ítems, presentar al usuario dos ítems, y solicitarle que seleccione uno de ellos, que cree una lista de ítems de su preferencia, formulario de registración al sistema, cuestionario con preguntas sobre sus preferencias, intereses, etc. Este tipo de adquisición de información del usuario posee varias desventajas dado que los usuarios tienen una cierta dificultad al autoevaluarse, principalmente en lo que respecta a capacidades y nivel de experto. Por lo que se recomienda presentar un conjunto muy controlado de preguntas, test, ejercicios, etc., con el objeto de obtener una visión más objetiva del usuario. Otra de las desventajas de este tipo de adquisición de información es que, al iniciar la interacción con el usuario con entrevistas, o actividades similares, es la Paradoja del Usuario Activo (Carrol & Rosson, 1987): los usuarios están motivados para empezar la interacción y necesitan concluir su tarea inmediata. No pierden tiempo con cuestionarios, manuales o ayudas en línea. Es una paradoja pues, ahorrarían tiempo a largo plazo, si perdieran algún tiempo inicialmente “optimizando” el sistema. En el caso de fuentes de información concurrentes, los usuarios pueden simplemente no visitar un sitio si tienen que responder primero a una entrevista. Si se lleva a cabo, la fase de adquisición debe ser por eso minimizada e idealmente sólo debe ser administrada después de que el usuario haya obtenido alguna impresión de los beneficios del sitio.

La información implícita es aquella que adquiere el sistema a partir del comportamiento e interacción del usuario con este. Por ejemplo el registro de los *ítems* que el usuario ha visto en una tienda online, analizar el número de visitas que recibe un ítem, analizar las redes sociales de las que el usuario forma parte y de esta manera conocer sus gustos y preferencias, hacer un análisis del historial de compra del usuario, comparar el tiempo que el usuario pasa leyendo sobre ciertos ítems en el sitio en contraste contra el tiempo que dedica a otros ítems, introducción de palabras claves en una búsqueda, analizar la cantidad de clic que hace el usuario sobre determinada sección del sitio. Los métodos de adquisición de información implícita o pasiva parecen ser menos molestos que los activos o explícitos. Por definición, los métodos pasivos no inician ninguna interacción con el usuario. Estos métodos de recopilación de información utilizan a menudo

reglas de adquisición (Cruz, 2003), que son un medio común de generar presunciones acerca del usuario, o sea, típicamente las reglas de inferencia son ejecutadas cuando hay nueva información disponible sobre el usuario. En la mayoría de los casos, las reglas de adquisición se refieren a acciones del usuario observadas, o a una interpretación más o menos directa del comportamiento del usuario. Para este tipo de recopilación de información se pueden utilizar heurísticas como “Si el usuario quiere conocer X, Entonces el usuario podría querer conocer Y” estando X & Y estrechamente relacionados.

Los perfiles de preferencias de usuarios son a menudo utilizados por sistemas de recomendación, estos representan un tipo especifico de filtrado de información que presenta al usuario recursos (películas, música, libros, noticias, páginas web, etc.) que le pueden ser relevantes, atendiendo a dicho perfil de preferencias (gustos, intereses, necesidades), sin necesidad de realizar búsquedas explícitas de dichos recursos (Rodríguez, 2017). Los Sistemas de Recuperación de Información (SRI) utilizan los perfiles de preferencias de usuarios para el procesamiento de consultas, para identificar su necesidad y personalizar los resultados de búsqueda (Viltres, 2018).

Actualmente el desarrollo del dominio .cu avanza a un ritmo creciente. Según (Cubanic, 2021) Centro Cubano de Información de Red, hasta el 14 de junio del 2021 se contaba con 6887 dominios registrados bajo .cu, incluyendo educación, noticias, medio ambiente, deporte, ciencia, cultura e incluso dominios gubernamentales y una amplia gama de información. Para la recuperación y visualización de esta información se ha desarrollado en la Universidad de Ciencias Informáticas (UCI) una herramienta denominada Orión que se encuentra en uso actualmente por la red universitaria. El buscador Orión cuenta con tres mecanismos fundamentales que, al funcionar en conjunto permiten contar con un sistema para acceder a los documentos y sitios publicados en la red. Estos son, una interfaz de tipo web para realizar las consultas de búsqueda, un servidor de indexación de contenidos basado en Solr y como robot de búsqueda o “*spider”* utiliza Nutch (Delgado, 2010). El éxito de Orión depende de que ofrezca a los usuarios resultados útiles y actualizados, de forma tal que en realidad se satisfagan sus necesidades.

Después de realizado un análisis sobre la bibliografía consultada se identificaron una serie de limitaciones que afectan la relevancia de los documentos mostrados a los usuarios y se resumen a continuación:

- Dificultad para identificar y comprender la necesidad del usuario escrita en lenguaje natural
- Poca precisión y exactitud de los resultados de búsquedas obtenidos

- Limitaciones para recuperar documentos relevantes al introducir términos exactos o ambiguos
- Dificultad para obtener las preferencias de búsqueda de los usuarios

Producto de esta situación se plantea el siguiente **problema de investigación:** ¿Cómo contribuir a la generación de perfiles de preferencias de usuarios para el buscador Orión?

Para solucionar el problema planteado, se determinó que el **objeto de estudio** se centra en el proceso de generación de perfiles de preferencias de los usuarios, enmarcado en el **campo de acción** generación de perfiles de preferencias en el sistema de recuperación de información Orión.

Se define como **objetivo general:** Desarrollar un componente para la generación de perfiles de preferencias de usuarios que permita contribuir al proceso de recuperación de información en el buscador Orión.

Para dar cumplimiento al objetivo general, se han trazados los siguientes **objetivos específicos:**

1. Construir los referentes teóricos fundamentales que sustentan la investigación relacionados con la generación de perfiles de preferencias de los usuarios.
2. Diagnosticar el estado de la generación de perfiles de preferencias de los usuarios.
3. Diseñar las funcionalidades del componente para la generación de perfiles de preferencias de los usuarios.
4. Implementar las funcionalidades del componente para la generación de perfiles de preferencias de los usuarios.
5. Validar las funcionalidades del componente para la generación de perfiles de preferencias de los usuarios.

Para dar respuesta a la problemática se fundamenta la siguiente **hipótesis de investigación:** El desarrollo de un componente de generación de los perfiles de preferencias de usuarios en el SRI Orión, que integre técnicas de recolección de información, mejora la calidad de los resultados de búsqueda brindados a los usuarios.

Operacionalización:

Variable Independiente: Generación de perfiles de preferencias de usuarios en el SRI.

Variable Dependiente 1: Calidad de los resultados de búsqueda brindados a los usuarios.

El concepto de calidad suele estar asociado a la satisfacción que los productos generan en un público determinado. Los indicadores de calidad son instrumentos de medición que se emplean para evaluar la calidad de los procesos o productos. O, dicho de otra manera, determinan el nivel de cumplimiento de los objetivos para los cuales se han desplegado una serie de actividades concretas. Según Berdugo (2016) Las dimensiones o factores no son universales como algunos autores como Parasuraman, Zeithaml e Berry han sostenido, sino que son específicos del tipo de servicio (Buttle, 1996). Algunos de los principales indicadores de calidad son conocimiento del servicio, exhaustividad, precisión, fiabilidad, rapidez, costo razonable, satisfacción y aceptación del cliente (Berdugo, 2016). Para este trabajo se han utilizado las dimensiones para la medición de calidad siguientes:

- **Precisión:** Evitar devolver resultados que no correspondan con la consulta realizada (Moreno, 2017; Sanchez, 2018; Otoya, 2019; Sanchez, 2019)
- **Exhaustividad:** Obtener tantos documentos relevantes como sea posible (Moreno, 2017; Sanchez, 2018; Otoya, 2019; Sanchez, 2019)
- **Aceptación de usuario:** Para dar cumplimiento a los objetivos propuestos, se utilizan diferentes métodos científicos. Los principales son (Sampieri, Fernández & Baptista., 2014).

Métodos Teóricos:

Método analítico-sintético: empleado en la descomposición del problema científico en elementos por separado y la profundización en el estudio de cada uno de ellos, para luego sintetizarlos en la propuesta de solución.

Método histórico-lógico: se empleó para el análisis de las principales técnicas de modelado de perfiles de preferencias de usuarios, su surgimiento, su evolución y su estado actual, en función de comprender mejor el objeto de estudio de la investigación.

Método hipotético-deductivo: como guía de la investigación se hace uso de una hipótesis científica. A partir de la observación y el análisis del fenómeno en cuestión, se formuló una hipótesis que será comprobada en el proceso de validación.

Métodos Empíricos:

Análisis documental: se realizaron consultas a documentación científica para el estudio de los referentes teóricos.
Encuesta: mediante su aplicación se obtuvo mediciones cuantitativas de los elementos cualitativos y cuantitativos abordados en la investigación.
Medición: permite medir los resultados al comparar el SRI antes de integrarle el procedimiento y después del mismo.
Experimentación: mediante los experimentos se evaluó la capacidad del procedimiento para contribuir a la calidad de los resultados brindados a los usuarios y la calidad de la información almacenada.
Criterio de expertos empleando el escalamiento de Likert: a partir de su aplicación a expertos se evaluaron los elementos teóricos que fundamentan la investigación.
Técnica Iadov: se aplicó para evaluar el nivel de satisfacción de potenciales usuarios con respecto al procedimiento propuesto.

Estructuración por Capítulos.

El presente documento está estructurado en Introducción, Tres Capítulos, Conclusiones, Recomendaciones, Bibliografía y Anexos:

Capítulo 1: Marco teórico referencial de la investigación, relacionado con el proceso de modelado y generación de perfiles de preferencias de usuarios.

En este capítulo se abordan conceptos fundamentales asociados al dominio del problema expuesto. Se realiza una revisión bibliográfica de los conceptos de recuperación de información y sistemas de recuperación de información. Se abordan los perfiles de usuarios, las preferencias de estos como características inherentes de los mismos y los perfiles de preferencias como producto final de la combinación de ambos. Se presentan los modelos representación de dichos perfiles, así como el proceso de construcción de estos, como se detectan e incluyen las preferencias de los usuarios a sus perfiles, las técnicas de recolección de información y se estudian sistemas clásicos de generación de perfiles basados en los modelos escogidos.

Capítulo 2: Componente para la generación de perfiles de preferencias de usuarios en el sistema de recuperación de información Orión.

Se plantea la propuesta del procedimiento para el modelado de perfiles de preferencias de usuarios en el sistema de recuperación de información Orión. Se enuncia el procedimiento propuesto, sus características y estructura general, así como una descripción de los procesos y actividades por cada fase.

Capítulo 3: Validación del componente para el modelado de perfiles de preferencias de usuarios en el sistema de recuperación de información Orión.

Se valida el cumplimiento del objetivo general de la investigación, a partir de los métodos científicos definidos. Se valida la pertinencia, efectividad y utilidad del componente para la generación de perfiles de preferencias de usuarios en el sistema de recuperación de información Orión. Adicionalmente se emplea la técnica IADOV con el objetivo de conocer el estado de satisfacción de los usuarios y obtener retroalimentación sobre la eficacia del modelo.

Capítulo 1: Marco teórico referencial de la investigación, relacionado con el proceso de modelado y generación de perfiles de preferencias de usuarios

En este capítulo se abordan conceptos fundamentales asociados al dominio del problema expuesto. Se realiza una revisión bibliográfica de los conceptos de recuperación de información y sistemas de recuperación de información. Se abordan los perfiles de usuarios, las preferencias de estos como características inherentes de los mismos y los perfiles de preferencias como producto final de la combinación de ambos. Se presentan los modelos representación de dichos perfiles, así como el proceso de construcción de estos, como se detectan e incluyen las preferencias de los usuarios a sus perfiles, las técnicas de recolección de información y se estudian sistemas clásicos de generación de perfiles basados en los modelos escogidos.

1.1 La recuperación de información y los Sistemas de recuperación de información

La RI surge a finales de la década de 1950, sin embargo, en la actualidad adquiere un rol más importante debido al valor que tiene la información en la generación de conocimiento. La recuperación de información es según Hersh (2021) el campo que abarca la adquisición, organización y búsqueda de información basada en conocimiento. Blázquez (2013) considera la RI como el proceso por el cual las demandas informativas y documentales del usuario son resueltas en un sistema de información, compuesto por un corpus documental de volumen variable; cuyo tratamiento de indexación y almacenamiento hacen posible su estructuración, interrogación y representación por medio del empleo de algoritmos matemáticos, estadísticos y semánticos. Luego de analizados los conceptos anteriormente abordados, En la presente investigación se asume que la RI es el proceso encargado del procesamiento, estructuración y almacenamiento de documentos que permite brindar mediante algoritmos matemáticos, respuestas a las interrogantes de los usuarios relativas a la información publicada en la web.

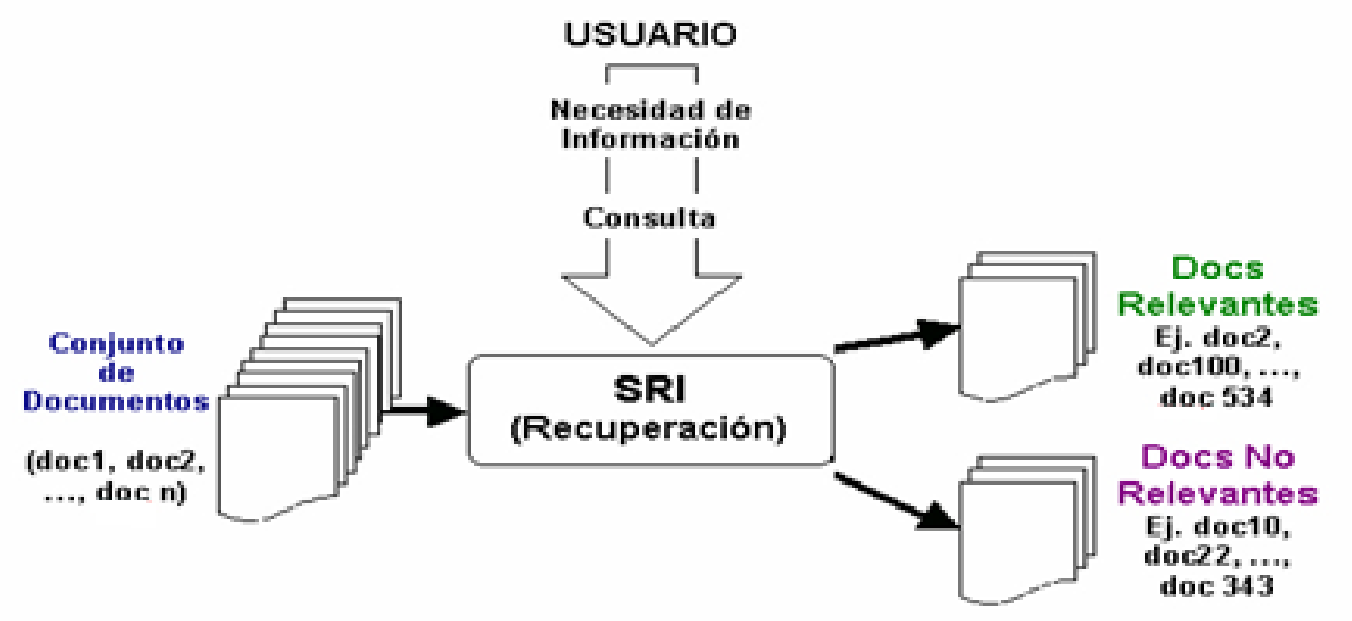


Figura 1: Problema de la RI. Fuente (Tolosa y Bordignon, 2008)

La tarea de recuperar información puede plantearse de diversas formas en relación a la manera en que el usuario interactúa con el sistema, o bien a partir de las facilidades que este le presenta (Ríssola & Tolosa, 2015):

Recuperación inmediata: El usuario define su necesidad de información y posteriormente, obtiene referencias a los documentos que el sistema determina como relevantes.

- Recuperación ad-hoc: en general la necesidad de información que el usuario presenta se traduce a una consulta en texto libre que posteriormente el sistema procesa y evalúa.
- Navegación o *browsing*: el sistema ofrece una interfaz con tópicos por los cuales el usuario navega obteniendo referencias a documentos relacionados. En este caso no se expresa una consulta explícita, hecho que facilita la búsqueda a usuarios que no poseen una necesidad clara.

Recuperación Diferida: El usuario detalla sus necesidades definiendo un perfil de modo tal, que el sistema entrega de forma continua aquellos documentos que se incorporen a la colección. Esta práctica recibe el nombre de filtrado y ruteo.

En la presente investigación se asume el uso de la recuperación diferida por la importancia que posee a largo plazo la estructuración de un perfil que pueda definir a un usuario como individuo y recuperar la información relevante para él, en dependencia de dicho perfil, en este caso perfil de preferencias. De forma general la problemática fundamental de la RI establece que:

- En la web existe un gran número de documentos que conforman una gran colección con diversas categorías.
- Los usuarios con necesidad de respuestas a sus interrogantes, acceden a los SRI e insertan su consulta o criterio de búsqueda, esperando una respuesta que satisfaga sus interrogantes.
- El SRI debe ser capaz de devolver a los usuarios los resultados más relevantes.

1.1.1 Los Sistemas de Recuperación de Información (SRI)

Un SRI es una herramienta que posee un conjunto de componentes que permiten el rastreo, indexación y visualización de la información recolectada, como parte de las respuestas ofrecidas a los usuarios después que ejecutan una consulta (Lafferty & Zhai, 2017). El problema principal de un SRI es brindar a los usuarios exactamente los documentos que satisfagan sus necesidades de búsqueda. Para ello se valen de una arquitectura básica organizada en los componentes que se ilustran en la figura 2.

Estos componentes responden a cuatro procesos fundamentales (Ríssola & Tolosa, 2015; Mora, 2016; Ensias, 2017; Lafferty & Zhai, 2017):

- Proceso de indexación
- Proceso de consulta
- Proceso de evaluación
- Proceso de retroalimentación del usuario

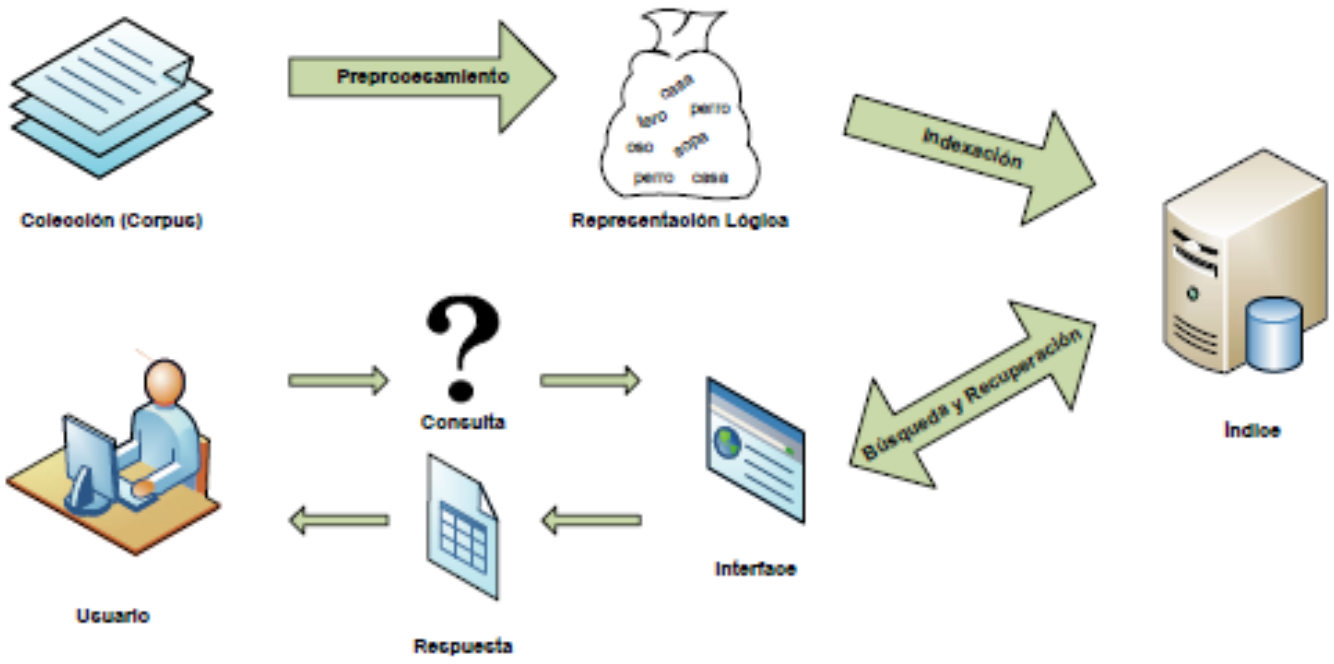


Figura 2: *Arquitectura básica de un SRI. Fuente (Rissola & Tolosa, 2015)*

Esta clasificación no está cerrada, sino que dependiendo del SRI y de sus mecanismos se pueden definir nuevos procesos. El presente trabajo se centra en el componente de procesamiento de datos, específicamente en la generación de los perfiles de usuario en concordancia con Leyva et al. (2018) que propone una arquitectura distribuida base para el despliegue de algunos de los componentes mas importantes de un SRI, entre los que se encuentran:

- Componente de rastreo e indexación
- Componente de procesamiento de datos (Categorización de documentos y definición de los perfiles de usuario)
- Componente de procesamiento estadístico de datos
- Componente de apoyo a la toma de decisiones
- Componente de visualización

1.2 Perfil de usuario

La creación organización y evaluación de la información están determinadas por las necesidades de sus usuarios, ya sean estos reales o potenciales (Hernández, 2018). De una forma genérica, un perfil de usuario es un modelo creado en la memoria del ordenador, que es utilizado como representante del usuario en las tareas computacionales. El perfil de usuario es el perfil más importante y es utilizado en muchas situaciones, entre ellas la personalización de aplicaciones y servicios, en la automatización de tareas, etc. (Cruz, 2003). Quiroz et al. (2018) plantea que el perfil de usuario consta de 5 elementos:

- Datos personales (nombre, edad, dirección, género): es la información personal del usuario, esta información es explícitamente ofrecida por el usuario mediante cuestionarios.
- Estilo de aprendizaje (Activo/Reflexivo, Sensitivo/Intuitivo, Inductivo/Deductivo, Visual/Verbal, Secuencial/Global): se obtiene utilizando el test de Felder y Silverman (1988) en los usuarios.
- Historial de búsqueda.
- Interfaz de usuario: es la forma de interacción que posee el usuario con el sistema, esta puede ser modificada por el usuario o adaptada por el sistema en función a sus gustos y preferencias.
- Preferencias del usuario: se obtienen mediante cuestionarios realizados al usuario o implícitamente al analizar sus interacciones con el sistema.

1.2.1 Preferencias de usuario y perfiles de preferencias

Las preferencias del usuario constituyen una de las características del usuario. Un usuario puede preferir unos enlaces o resultados de búsqueda en lugar de otros. Estas preferencias pueden ser persistentes o relativas, esto, depende de la necesidad actual del usuario, del objetivo y del contexto actual en general. Las preferencias son usadas mayoritariamente por los SRI, para devolver al usuario respuestas específicas y acorde a sus intereses, sean estos, recientes o persistentes (Kacem, 2017). En concordancia con Viltres (2018), el perfil de preferencia del usuario determina el contexto y la intención del usuario para que el SRI seleccione los documentos más relevantes al mismo. El perfil de preferencias dentro de la computadora representa la “forma física” de las necesidades de información del usuario (Hernández, 2018).

Según investigaciones de Kusumaningrum (2017), Hernández (2018) y Viltres (2018) y a los efectos de la presente investigación se define el perfil de preferencias de usuarios como un modelo de las necesidades de búsqueda de los mismos, que están en constante cambio y actualización, permitiendo la comunicación de uno a uno con el SRI garantizando que las respuestas de este, sean apropiadas y específicas al usuario en cuestión, con el objetivo de facilitar la obtención de información.

1.2.2 Adquisición de información para los perfiles de preferencias

Para adquirir la información relevante a un usuario para conformar el perfil de usuario, o en este caso el perfil de preferencias, Aciar y Aciar (2013) manifiestan que, se puede utilizar dos formas o métodos de recolección de información: explícita o implícita. Según Farid (2017) la precisión del perfil de preferencias depende de la cantidad de datos generados mediante la interacción usuario-sistema y los métodos de recopilación de datos pueden ser explícitos, implícitos o híbridos.

- Los enfoques **explícitos** de recopilación de información plantean adquirir la información con técnicas manuales mediante la intervención del usuario, como formularios de registro, cuestionarios o pidiendo a los usuarios que califiquen artículos (Farid, 2017). eBay pide a los usuarios que proporcionen sus opiniones y valoraciones de los servicios y productos que se ofrecen, la empresa utiliza esta información para mejorar las recomendaciones que da a los clientes. Yahoo solicita explícitamente al usuario que proporcione información personal que será almacenada para crear un perfil. Los perfiles explícitos tienen también una naturaleza estática y son válidos sólo hasta que el usuario cambie sus preferencias. Para superar este problema se sugiere extraer la información implícitamente mediante el seguimiento de las actividades y / o comportamiento de los usuarios.
- El método **implícito** intenta inferir la los intereses o el contexto del usuario de los registros procesados o la información recopilados automáticamente sin ningún esfuerzo por parte del usuario. A diferencia del enfoque explicito, el método implícito permite la creación de perfiles automáticos y analiza el patrón de comportamiento del usuario (por ejemplo, historial de uso) para determinar los intereses de un usuario (Farid, 2017). Los métodos de adquisición de información implícita o pasiva parecen ser menos molestos que los activos o explícitos. Por definición, los métodos pasivos no inician ninguna interacción con el usuario (Cruz, 2003). Los inconvenientes de este enfoque son la falta de información para la generación de un perfil, por ejemplo, si el usuario es nuevo o no interactúa lo suficiente con el sistema, y que a veces plantean problemas de privacidad para los usuarios.
- El enfoque **híbrido** recopila la información mediante técnicas semiautomatizadas con participación restringida del usuario, con un enfoque mixto de métodos implícitos y explícitos. Este enfoque ayuda a perfilar eficientemente y mantiene una precisión temporal del perfil a medida que la información se actualiza (Farid, 2017). Kobsa (2001) plantea la recopilación de información de un perfil utilizando retroalimentación explícita en forma de cuestionario solicitando información personal y un conjunto

de intereses y complementando con fuentes de información implícita como marcadores web, historial de navegación, respectivamente.

1.2.3 Aplicaciones de perfiles de preferencias

Los perfiles de preferencias de usuarios son utilizados para brindar servicios personalizados en diferentes áreas, como sistemas de recomendación personalizados, servicios móviles, redes sociales, búsqueda web y navegación, etc. A continuación, se describen varios procesos y sistemas que se beneficiarían de la generación de un buen perfil de preferencias de usuario.

- Búsqueda y navegación web personalizada: la búsqueda personalizada es el proceso de buscar una serie de documentos o páginas web dando preferencia a los intereses del usuario. El interés del usuario es identificado desde el perfil de preferencias. Remesh y Philip (2015) proponen que una búsqueda y navegación personalizadas es un concepto de recuperación de páginas web según las preferencias del usuario, mejorando la navegación y teniendo en cuenta el dinamismo y cambio de las preferencias del usuario mientras se realiza la búsqueda.
- Sistemas de recomendación y predicción: un sistema de recomendación es una subclase del sistema de filtrado de información, que se utiliza para calificar un artículo o servicio en dependencia de las preferencias del usuario (Farid et al., 2018). Un sistema de recomendación de información tiene como función conocer al usuario y eficientemente recomendar artículos que sean de su interés mediante la utilización de su perfil de preferencias.
- Sistemas de aprendizaje adaptativo: el aprendizaje adaptativo es una rama del *e-learning*, que ayuda a los alumnos en su aprendizaje y estudio. Nguyen (2014) propuso un sistema llamado Zebra que proporciona un poderoso mecanismo de inferencia para razonar nueva información sobre los estudiantes basado en sus perfiles.
- Visualización: Yingze et al. (2014) propuso un estudio sobre interfaces interactivas, en el que se describe, cómo la interfaz utilizada en redes sociales puede visualizarse y editarse basado en los perfiles de preferencias de usuarios.
- Servicios personalizados de redes sociales en línea: las redes sociales personalizadas clasifican la información automáticamente en categorías, mostrando al usuario la información más relevante según sus intereses. Se realiza el perfil de preferencias basado en el comportamiento del usuario en

dichas redes, por ejemplo, mediante las opciones de compartir y publicar de Facebook (Kelly & Teevan, 2003).

1.3 Modelado y construcción de perfiles de preferencias

Luego de adquirida la información mediante los métodos de recopilación descritos (explícito, implícito e híbrido) se procede al modelado y construcción de los perfiles de preferencias. Estas acciones representan el almacenamiento y representación de la información del usuario en forma de modelos que sean entendibles por el ordenador (Farid, 2017).

1.3.1 Modelos de representación de perfiles de preferencias

En la presente sección se plantean tres tipos de modelos para la generación de los perfiles de preferencias los cuales son: basado en términos o palabras clave (*keyword-based*), basado en redes semánticas (*semantic-networks*) y basado en conceptos o jerarquías(*concept-based*).

Basado en términos o palabras clave (*Keyword-based*).

Un modelo basado en términos o vectores es la representación más común de los intereses del usuario. Se compone de una colección de datos referidos como un conjunto de términos (o espacio vectorial de términos) por vectores ponderados de palabras clave que se pueden utilizar directamente para ampliar las consultas de los usuarios (Shen et al., 2005). Los métodos Booleano, Frecuencia de aparición del término (TF) y Frecuencia inversa del documento para un término (TF-IDF) (Berry, 2004; Blázquez, 2013), son métodos diferentes para desarrollar una representación para cada término ponderado de los documentos o consultas. Además, la similitud entre dos vectores de términos ponderados es calculado por el método de similitud del coseno (*Cosine Similarity* CS) (Salton et al., 1987). Las palabras clave de intereses del usuario son extraídas de los documentos visitados durante la navegación. Están representados por un solo vector que incluye todo el interés o con múltiples vectores, lo que refleja interés en varios dominios (Gauch et al., 2007) y se puntúan los resultados de búsqueda mediante una puntuación de relevancia representada por un número *float* positivo denominado *score*. En ese caso, a mayor *score* más relevante es la búsqueda (*Elasticsearch of Apache Software Foundation,* 2017). En Este modelo la efectividad de los perfiles de usuario depende del grado de generalización de los vectores. Sus principales inconvenientes son la polisemia y que puede causar una mala interpretación de los intereses de los usuarios.

Basado en redes semánticas (*Semantic-Network-based or Ontology-based*).

Para abordar la polisemia, la ambigüedad y los problemas de sinonimia inherentes con perfiles basados en palabras clave, los perfiles pueden estar representados por un concepto ponderado de cada nodo de la red semántica. Sin embargo, esto también representa una carga para la facilidad de construir tal sistema. Se requiere un mapeo existente entre palabras de interés y conceptos, como: ontología de *WordNet*, o mediante la construcción de un método de sistema de aprendizaje o manualmente (Farid, 2018). Liu (2015) propone generar perfiles de usuarios semánticamente mejorados, que reflejan los intereses e intenciones de los usuarios en un conjunto de categorías específicas. Un proceso de modelado de perfiles basado en ontologías es propuesto por Sieg et al. (2007), que facilita que un gran número de diferentes aplicaciones interactúen semánticamente con una misma base de conocimientos; y también representa intereses en forma de conceptos ontológicos interrelacionado con la ontología fuente predefinida mediante el uso de conceptos de Wikipedia y *WordNet*.

Basado en conceptos o jerarquías (*Concept-based or Hierarchy-based*).

Perfiles basados en conceptos funcionan igual que los perfiles basados en redes semánticas, en la representación por nodos de conceptos y relaciones entre esos nodos. Varios mecanismos son aplicados para asociar un peso a cuánto el usuario está interesado en cada tema. Nanas et al. (2003) han usado diferentes técnicas para categorizar los intereses de los usuarios de forma jerárquica y utilizan para este propósito bases de conocimiento. Los investigadores utilizan el sistema de clasificación de bibliotecas (*library classification system* LCS) para encontrar las jerarquías de los intereses del usuario y utilizan el mecanismo TF-IDF para calcular el peso de las mismas.

1.4 Estudio de homólogos

Con el objetivo de encontrar una solución adaptable se analizaron los principales SRI existentes en el mundo. En dicho análisis se identificaron en los buscadores internacionales Google, Yahoo!, Search y Bing; y en los nacionales C.U.B.A y Lupa los siguientes aspectos: motor de búsqueda, robot rastreador y algoritmos de relevancia empleados.

Como resultado del estudio se destacaron algunas características y funciones más relevantes de dichos buscadores web. Basado en ello se concluye y se comprueba que ninguno de los modelos matemáticos de cálculo de relevancia de información presentes en los buscadores web utilizan el perfil de usuario como se

expone en el artículo científico *"Algorithm for calculating relevance of documents in information retrieval systems"* (Baquerizo et al., 2017).

Tabla 1: *Estudio de los buscadores web en el mundo. Fuente: García et al. (2019)*

Internacionales			
Buscadores Web	**Motor de búsqueda**	**Robot rastreador**	**Algoritmos de relevancia**
Google	Google	Googlebot	PageRank, Modelo Espacio Vectorial
Yahoo	Search Assits	Yahoo Slurp	-
Bing	Bing	Bingbot	-
Nacionales			
C.U.B.A.	Orión	Nutch	Modelo Booleano, Modelo Espacio Vectorial
Lupa	Solr	Lucid Crawler	Modelo Espacio Vectorial

En consecuencia, el motor de búsqueda Orión no cuenta con un algoritmo que elabore las clasificaciones de los resultados según una base de conocimiento del perfil de usuario. Esto pudiera provocar una primera imprecisión en la presentación de los resultados y es que, la información presentada al usuario pudiera no estar relacionada con aquellas materias o aquel ámbito que en realidad le interesa. Esto es válido, sobre todo, cuando se hacen consultas usando términos que se aplican en varios campos o áreas del conocimiento. Igualmente, no devuelve al usuario resultados útiles con un criterio de relevancia personalizada de acuerdo a las categorías de documentos que en realidad interesan al que busca. Esto trae consigo que el usuario pierda el interés y la confianza en la herramienta, si no es capaz de satisfacer sus necesidades de forma óptima. En ese caso, se afecta un factor importante como es la fidelización de los usuarios.

Para el autor Baeza Yates la solución a dicho problema se identifica con dos etapas fundamentales: elección de un modelo que permita calcular la relevancia de un documento frente a una consulta y el diseño de algoritmos que implementen este modelo de forma eficiente. En este sentido es necesario integrar esta variable al cálculo de la relevancia de información, debido a la importancia de conocer la intención del usuario después de la búsqueda. Respecto a ese tema, se evidencia carencia de información en buscadores como **Yahoo**, **Search** y **Bing**, debido a la indiscutible competencia con la mundialmente conocida **Google**.

Selección del modelo de representación de perfiles de preferencias, de los métodos de recolección de información y algoritmos matemáticos utilizados para el desarrollo de la propuesta solución.

Partiendo de los estudios realizados anteriormente, se decide para el desarrollo de un componente para la generación de perfiles de preferencias de usuarios; el método de recolección de información **híbrido**, una primera parte de forma explícita mediante una entrevista inicial al usuario, y una segunda parte implícita utilizando técnicas de minería de datos (Web mining) que monitorean el comportamiento del usuario en el sistema a través de la categorización de documentos relevantes para este.

Para el cálculo de similitud se decide emplear el Modelo Espacio Vectorial (similitud del coseno) por su versatilidad y eficiencia a la hora de generar *rankings* precisión en colecciones de gran tamaño lo que le hace idóneo para determinar la puntuación parcial de los documentos (García et al., 2019). A los efectos del planteamiento anterior, se añade que según Martínez Méndez (2004) el modelo espacio vectorial es el más utilizado en la web.

$$sim(d_i, d_j) = cos(\alpha) = \frac{\sum_{k=1}^{m} d_{ik} \cdot d_{jk}}{\sqrt{\sum_{k=1}^{m} d_{ik}^2} \sqrt{\sum_{k=1}^{m} d_{jk}^2}}$$

Figura 3: Fórmula utilizada para el cálculo de la similitud del coseno

En la figura 3 d_i y d_j son los documentos y d_{ik} representa el peso del rasgo *k* en el documento d_i.

En el proceso de categorización de documentos se empleará el algoritmo desarrollado por los autores; Roberto Passailaigue Baquerizo, Paúl Rodríguez Leyva, Juan Pedro Febles, Hubert Viltres Sala y Vivian Estrada Sentí en su artículo científico *¨Algorithm for calculating relevance of documents in information retrieval systems¨* que utiliza técnicas de minería de datos en el cálculo de relevancia de los documentos ofreciendo al usuario un resultado personalizado para su búsqueda. Este método plantea que obteniendo las categorías presentes en el documento y comparándolas con las categorías relevantes para el usuario, si estas coinciden se creará una nueva variable donde se guardará la puntuación de dicha categoría en un numero *float* positivo de 0 a 1, siendo 1 la mayor puntuación, y, añadido al resultado del cálculo de similitud obtenido mediante la fórmula de similitud del coseno, los documentos podrían obtener una puntuación del 0 al 2, siendo los cercanos a 2 los mas relevantes para el usuario. Según Baquerizo et al. (2017) la utilización de técnicas de minería web para procesar la información almacenada en un SRI e incluyendo variables, así

como el perfil de búsqueda del usuario y la puntuación de las categorías relevantes para este, permite al SRI devolver respuestas más relacionadas con las necesidades de búsqueda de los usuarios.

Para la representación de los perfiles de preferencias se decide emplear el modelo basado en palabras clave **(*Keyword-based*)** dadas las ventajas antes mencionadas y teniendo en cuenta que la efectividad en este modelo depende del grado de generalización d los vectores o palabras clave, esto lo hace el candidato idóneo para la utilización del algoritmo de minería web escogido para la categorización de los documentos. Además, que en este modelo está comprendido el algoritmo utilizado para el cálculo de similitud Modelo Espacio Vectorial.

1.5 Metodología de desarrollo de software

Una metodología es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar nuevos sistemas de información. Una metodología está formada por fases, cada una de las cuales se puede dividir en sub-fases, que guiarán a los desarrolladores de sistemas a elegir las técnicas más apropiadas en cada momento del proyecto y también a planificarlo, gestionarlo, controlarlo y evaluarlo."

En la actualidad se pueden diferenciar dos grandes grupos de metodologías de desarrollo de software: las ágiles y las tradicionales. Para el desarrollo de la solución propuesta se seleccionó la metodología de desarrollo de software Variación AUP para la UCI debido a que es una metodología ágil y es una variación de AUP (Agile Unified Process, Proceso Unificado Ágil) que logra estandarizar el proceso de desarrollo de software en los proyectos de la universidad, además de convertirse en la metodología rectora de su desarrollo productivo, ya que se adapta perfectamente al ciclo de vida de la actividad productiva de los diferentes centros de la institución (Rodríguez, 2015).

1.5.1 Descripción de las fases de la metodología AUP-UCI

De las 4 fases que propone AUP (Inicio, Elaboración, Construcción, Transición) se decide para el ciclo de vida de los proyectos de la UCI mantener la fase de Inicio, pero modificando el objetivo de la misma, se unifican las restantes 3 fases de AUP en una sola, a la que llamaremos Ejecución y se agrega la fase de Cierre. Para una mayor comprensión se muestra la siguiente Tabla 2.

Tabla 2: *Variación de las fases de la metodología AUP a AUP-UCI. Fuente: Rodríguez (2015)*

Fases AUP	Fases Variación AUP-UCI	Objetivos de las fases (Variación AUP-UCI)
Inicio	Inicio	Durante el inicio del proyecto se llevan a cabo las actividades relacionadas con la planeación del proyecto. En esta fase se realiza un estudio inicial de la organización cliente que permite obtener información fundamental acerca del alcance del proyecto, realizar estimaciones de tiempo, esfuerzo y costo y decidir si se ejecuta o no el proyecto.
Elaboración Construcción Transición	Ejecución	En esta fase se ejecutan las actividades requeridas para desarrollar el software, incluyendo el ajuste de los planes del proyecto considerando los requisitos y la arquitectura. Durante el desarrollo se modela el negocio, obtienen los requisitos, se elaboran la arquitectura y el diseño, se implementa y se libera el producto.
	Cierre	En esta fase se analizan tanto los resultados del proyecto como su ejecución y se realizan las actividades formales de cierre del proyecto.

1.6 Tecnologías y herramientas de desarrollo

UML: El Lenguaje Unificado de Modelado (UML) fue creado para forjar un lenguaje de modelado visual común y semántica y sintácticamente rico para la arquitectura, el diseño y la implementación de sistemas de software complejos, tanto en estructura como en comportamiento. Es comparable a los planos usados en otros campos y consiste en diferentes tipos de diagramas. En general, los diagramas UML describen los límites, la estructura y el comportamiento del sistema y los objetos que contiene. Un modelo UML está compuesto por 3 clases de bloques de construcción: los elementos (representaciones abstractas de cosas

reales o ficticias); las relaciones (relacionan los elementos entre sí); y los diagramas (son colecciones de los elementos con sus relaciones) (Lucid Software Inc., 2018).

Java: Java es un lenguaje de programación multiplataforma orientado a objetos de arquitectura neutral, simple, robusto y seguro, con un recolector de basura y que puede alcanzar un alto rendimiento con hardware especializado. Posee un tipado estático y su portabilidad lo hace uno de los lenguajes de programación mas populares y usados en la actualidad, especialmente para aplicaciones de arquitectura cliente-servidor (Netec, 2019).

SQL: SQL es un acrónimo en inglés para "*Structured Query Language*." Un "Lenguaje de Consulta Estructurado. Un tipo de" lenguaje de programación que te permite manipular y descargar datos de una base de datos. Tiene capacidad de hacer cálculos avanzados y álgebra. Es utilizado en la mayoría de empresas que almacenan datos en una base de datos. Ha sido y sigue siendo el lenguaje de programación más usado para bases de datos relacionales.

XML: XML es un lenguaje de marcado similar a HTML. Significa Extensible *Markup Language* (Lenguaje de Marcado Extensible). Esto significa que, a diferencia de otros lenguajes de marcado, XML no está predefinido, por lo que debes definir tus propias etiquetas. El propósito principal del lenguaje es compartir datos a través de diferentes sistemas, como Internet.

NetBeans 12.5: *NetBeans* es un entorno de desarrollo integrado libre, hecho principalmente para el lenguaje Java y posee un numero importante de módulos para su extensión. Al ser de código abierto cuenta con una comunidad en constante crecimiento. Este IDE soporta lenguajes como JavaScript, PHP, HTML5 y CSS y es especialmente útil para el trabajo modular de aplicaciones.

PostgreSQL 13: Consiste en un gestor de bases de datos relacionales, soporta diferentes tipos de datos y está orientado a objetos. Es de código abierto, es decir, cuentan con una comunidad de desarrolladores que implementan mejoras o resuelven errores de forma altruista. Ofrece opciones que otras bases de datos no tienen, como la posibilidad de crear tablas heredadas, esquemas o *triggers*. No solo permite crear procedimientos en lenguaje SQL, sino que también permite utilizar otros lenguajes de programación como Python o Pearl. Permite definir datos que el programa no soporte de serie. Se pueden añadir nuevas funciones o extensiones desarrolladas por terceros, o incluso por uno mismo. También se caracteriza por ofrecer una gran escalabilidad vertical.

pgAdmin 4 v6.1: pgAdmin es una herramienta indispensable para gestionar y administrar PostgreSQL, que muestra una representación grafica de la base de datos y permite administrarla fácilmente.

Visual Paradigm 15.1: Visual Paradigm es una herramienta CASE: Ingeniería de Software Asistida por Computación. La misma propicia un conjunto de ayudas para el desarrollo de programas informáticos, desde la planificación, pasando por el análisis y el diseño, hasta la generación del código fuente de los programas y la documentación.

Solr; Apache Solr es un motor de búsquedas de código abierto que proporciona potentes funcionalidades de búsqueda y navegación por facetas, explora la información desde diversas perspectivas. Las funcionalidades son difíciles de implementar sobre una base de datos relacional. Solr tiene la capacidad de manejar complejos criterios de búsqueda, corrige la ortografía de las consultas, permite destacar resultados, configurar la relevancia de términos, entre otras funcionalidades (Estrada Ramos, 2012).

Conclusiones del capítulo

- Los fundamentos teóricos de la RI confirman la necesidad de la estructuración de un perfil que pueda definir a un usuario como individuo y recuperar la información relevante para él, en dependencia del perfil, en este caso un perfil de preferencias.
- El estudio de los perfiles de usuario permitió definir al usuario como el personaje principal de la trama informática, el principio y fin de cada ciclo de transferencia de la información, las preferencias de los usuarios como características inherentes de los mismos y los perfiles de preferencias como producto final de la combinación de ambos.
- Se pudo observar como el usuario logra, implícita y explícitamente, que el sistema note y construya un perfil de preferencias que lo represente. Así como la importancia y aplicaciones que posee dicho perfil en la recuperación de información.
- El estudio de modelado y construcción de los perfiles de preferencias de usuarios aportan diferentes puntos de vista a tener en cuenta a la hora de generar un perfil, cómo estará estructurado, en base a qué se generará (palabras clave, conceptos o redes semánticas), de dónde viene la información y qué forma tomará luego de creado dicho perfil.
- El estudio de los buscadores permitió analizar los sistemas ya creados para esta función, sus diferencias, ventajas para retomar en la investigación y desventajas para mitigar. Aportaron diversos enfoques de arquitecturas y distintas tecnologías para el desarrollo de un mismo producto, que en este caso son los perfiles de preferencias de usuario.

Capítulo 2: Componente para la generación de perfiles de preferencias de usuarios en el sistema de recuperación de información Orión

En el capítulo se detallan las características del componente de para la generación de perfiles de preferencias de usuario en el sistema de recuperación de información Orión, según plantea la metodología AUP-UCI en su escenario 4, se presenta la propuesta de solución a desarrollar. Se especifican las funcionalidades del componente con la descripción de los requisitos funcionales, no funcionales y las historias de usuario como artefacto principal generado. Se describe la arquitectura a utilizar, el diagrama de clases con los patrones de diseño utilizados y el diagrama de despliegue del sistema.

2.1 Descripción de la propuesta de solución

El proceso del componente para la generación de perfiles de preferencias de usuario en el sistema de recuperación de información Orión comienza con la declaración de las preferencias explícitas del usuario en forma de entrevista inicial donde se le ofrecerá al usuario escoger entre diversas categorías cuáles son del interés de este, esta información es guardada en forma de puntuación, tomando valor 1 las categorías que el usuario escoja y 0 las que no, conformándose así la primera instancia del perfil de preferencias de usuario.

Cuando el usuario interactúe con el sistema mediante la realización de consultas, el motor de búsqueda Solr realizará el tratamiento de consultas, obtendrá las palabras clave y accederá al perfil de preferencias de usuario para obtener el valor de puntuación las categorías y añadirlo al resultado del cálculo de similitud obtenido a través del Modelo de Espacio Vectorial (similitud del coseno) implementado en la clase ¨Similarity¨ de la librería Lucene para realizar la indexación de documentos relevantes al usuario en cuestión.

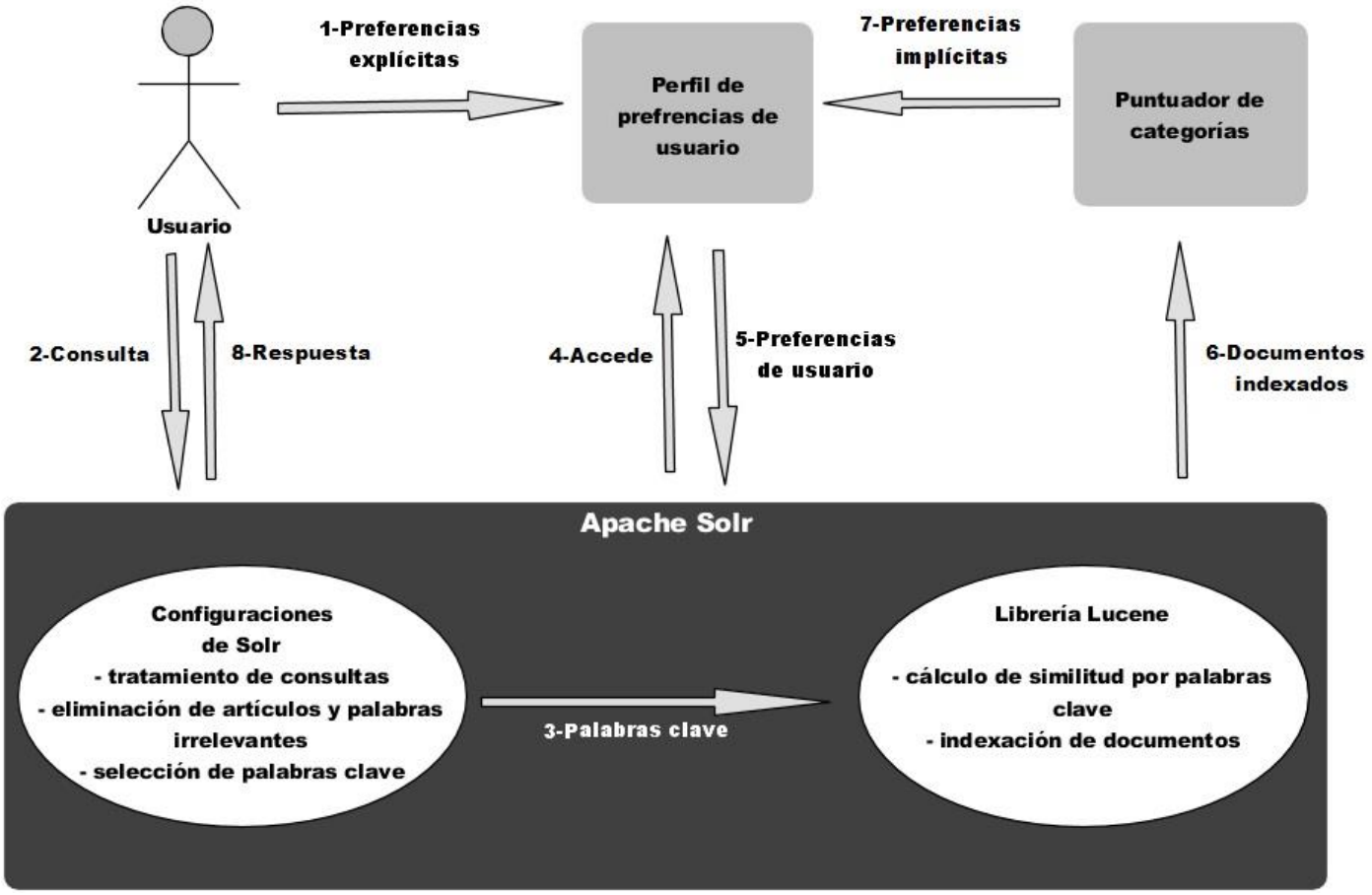


Figura 4: Proceso de generación de perfiles de preferencias de usuarios.

Luego de indexados los documentos relevantes al usuario, se procede a puntuar las categorías presentes en los documentos indexados con un máximo umbral de 1, esta puntuación está relacionada con la cantidad de documentos en los que se encuentra presente una determinada categoría. Obtenidos los valores de puntuación de las categorías de los documentos se procede a actualizar los valores de puntuación de las categorías almacenadas en el perfil de preferencias de usuario realizando un cálculo de promedio en categorías iguales, añadiendo así las preferencias implícitas del usuario.

2.2 Modelo Conceptual

Un modelo del dominio, o modelo conceptual, es una representación visual de las clases conceptuales u objetos del mundo real en un dominio de interés. Es importante resaltar que un modelo del dominio no representa componentes de *software*. No es tratado como un conjunto de diagramas que describen clases de *software*, u objetos *software* con responsabilidades (Larman, 2004). A continuación, se muestra el diagrama de modelo de dominio del componente de generación de perfiles de preferencias para el SRI Orión

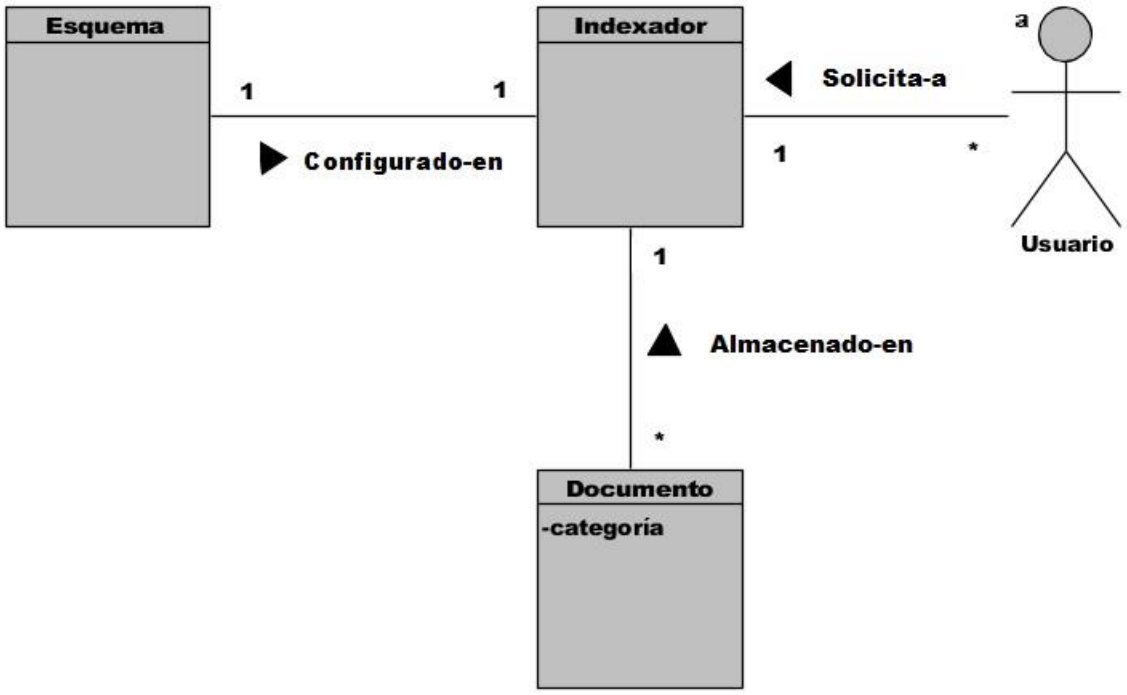


Figura 5: Diagrama de modelo de dominio del componente de generación de perfiles de preferencias para el SRI Orión

Descripción de los objetos presentes en el modelo de estudio:

- ✓ **Usuario:** es el agente que interactúa con el sistema y solicita los documentos.
- ✓ **Esquema:** archivo de configuración que rige al servidor de indexación.
- ✓ **Indexador:** es la herramienta que se encarga de indexar los documentos recuperados de la Web.
- ✓ **Documento:** documento recuperado de la Web.
- ✓ **Categoría:** es el campo que contiene la categoría del documento.

2.3 Especificación de requisitos del Software

Los requisitos del *software* permiten establecer lo que el sistema debe hacer, sus características fundamentales y las restricciones en el funcionamiento del sistema y los procesos de desarrollo del *software*. De manera general, estos requisitos expresan las necesidades objetivas que presentan los usuarios, ante un sistema que resuelve un problema en particular de un determinado dominio (Sommerville, 2005). Estas cualidades se clasifican en:

- ➢ **Requisitos funcionales (RF):** Describen lo que el sistema debe realizar (Sommerville, 2005).
- ➢ **Requisitos no funcionales (RNF):** Son aquellos que no se refieren a las funcionalidades específicas que proporciona el sistema, sino a las propiedades emergentes de éste como fiabilidad, el tiempo de respuesta y la capacidad de almacenamiento (Sommerville, 2005).

Requisitos funcionales

La Tabla 2 muestra un listado de los requisitos funcionales y su correspondiente prioridad (alta, media y baja), con el objetivo de esclarecer la comunicación entre usuario y *software*:

Tabla 3: *Listado de requisitos funcionales.*

No. RF	**Requisito**	**Descripción**	**Prioridad**
1	Obtener las preferencias explicitas.	Se obtienen las preferencias explícitas del usuario a través de una entrevista inicial realizada.	Alta
2	Identificar las palabras clave.	Procesamiento de consultas para la obtención de palabras clave.	Media
3	Obtener los documentos relevantes.	Se obtiene el resultado del cálculo de similitud y se le adiciona al valor de las categorías en el perfil de usuario.	Media
4	Puntuar categorías.	Se puntúa la categoría de los documentos con un máximo umbral de 1, en dependencia las categorías presentes en los documentos indexados y la cantidad de documentos.	Alta
5	Obtener las preferencias implícitas.	Actualiza el perfil de preferencias de usuario con una nueva puntuación de las categorías.	Alta

Requisitos no funcionales

El componente de categorización semántica de documentos debe cumplir con los siguientes RNF:

Tabla 4: *Listado de requisitos no funcionales*

No. RNF	Nombre	Atributo
1	Emplear Java como lenguaje de programación.	*Software*
2	Los servidores donde se instalarán cada uno de los elementos del componente seguirán una distribución GNU/Linux.	*Software*
3	La aplicación se ejecutará en cualquier SO que soporte las librerías de Java 8.	Funcionalidad
4	Utilizar una herramienta que almacene la información indexada.	Funcionalidad
5	Requisitos mínimos en el servidor: 16 Gb de memoria RAM y CPU con 4 *cores* de procesamiento a 3.0 GHz de velocidad.	Hardware
6	Los servidores para la indexación de información deben poseer un disco duro de 2 TB de almacenamiento.	Hardware
7	El componente no debe requerir de información confidencial.	Seguridad
8	El componente debe ser capaz de responder en el menor tiempo posible, en dependencia del tipo de petición y los datos que son manejados.	Rendimiento
9	Será distribuido bajo licencia libre de código abierto GPL (*General Public License*), permitiéndose la modificación, uso y distribución libre.	Licencia
10	Debe permitir su actualización y mantenimiento.	Soporte
11	La herramienta debe ser usable por cualquier usuario, independientemente de si posee experiencia o no.	Usabilidad

12	El componente deberá ser empaquetado en un .jar que garantice su portabilidad e integración con la herramienta de indexado de la información.	Portabilidad

2.4 Historias de usuario

Las Historias de Usuario son un enfoque de requerimientos ágil que se focaliza en establecer conversaciones sobre las necesidades de los clientes. Son descripciones cortas y simples de las funcionalidades del sistema, narradas desde la perspectiva de la persona que desea implementar cada funcionalidad (Izaurralde, 2013).

A continuación, se muestran las historias de usuarios descritas para cada requisito funcional.

Tabla 5: *Historia de usuario #1*

Número: 1	**Nombre de requisito:** Obtener las preferencias explicitas.
Programador: Yossan A. Rodríguez Nuñez	**Iteración Asignada:** 1
Prioridad: Alta	**Tiempo:** 48 horas
Riesgo: Bajo	**Tiempo real:** 40 horas
Descripción: Se realiza una encuesta inicial al usuario en su primer *login* luego de añadido el componente en la que el usuario debe seleccionar entre diversas categorías, las que son de su interés. Las que el usuario marque obtendrán una puntuación de 1 y las que no obtendrán una puntuación de 0, creándose la primera instancia del perfil de preferencias.	
Observaciones: La encuesta solo debe realizarse una vez.	
Prototipo de Interface:	

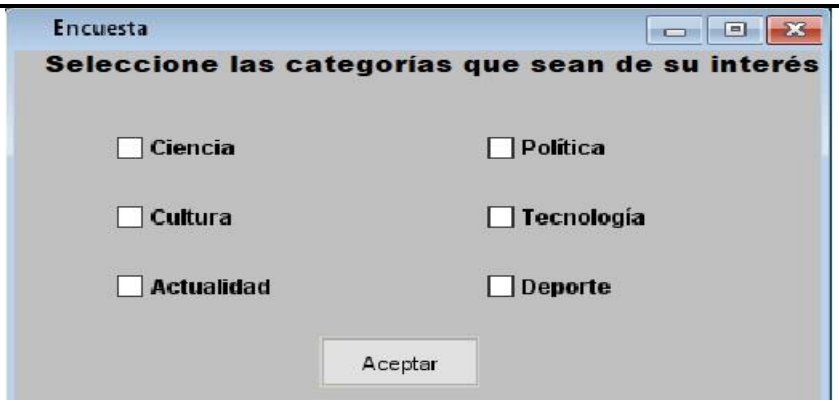


Tabla 6:*Historia de usuario #2*

Número: 2	**Nombre de requisito:** Identificar las palabras clave.
Programador: Yossan A. Rodríguez Nuñez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo:** 48 horas
Riesgo: Alto	**Tiempo real:** 32 horas
Descripción: Comienza con la eliminación de todos los caracteres innecesarios como los números, palabras superfluas, vacías, se eliminan los datos que no aportan información, como son los artículos, preposiciones, conjunciones entre otras palabras a través de las configuraciones de Apache Solr con sus componentes ¨*Tokenizer*¨, ¨*Token Filter*¨, ¨*Merge Policy*¨ que realizan el tratamiento de las consultas y obtienen las palabr	
Observaciones:	
Prototipo de Interface: No aplica.	

Tabla 7: *Historia de usuario #3*

Número: 3	**Nombre de requisito:** Obtener los documentos relevantes.
Programador: Yossan A. Rodríguez Nuñez	**Iteración Asignada:** 1
Prioridad: Media	**Tiempo:** 24 horas
Riesgo: Alto	**Tiempo real:** 16 horas
Descripción: Se obtiene el resultado del cálculo de similitud mediante el Modelo Espacio Vectorial con la fórmula de similitud del coseno y se aplica el algoritmo diseñado por Baquerizo et al. (2017) donde al resultado del cálculo de similitud se le adiciona la puntuación que posee la categoría del documento en el perfil de preferencias de usuario obteniendo un nuevo umbral de 2 siendo los documentos mas cercanos a 2 los mas relevantes para el usuario.	
Observaciones:	
Prototipo de Interface: No aplica.	

Tabla 8: *Historia de usuario #4*

Número: 4	**Nombre de requisito:** Puntuar categorías.
Programador: Yossan A. Rodríguez Nuñez	**Iteración Asignada:** 2
Prioridad: Alta	**Tiempo:** 24 horas
Riesgo: Alto	**Tiempo real:** 18 horas
Descripción:	

Obteniendo las categorías de los documentos relevantes se puntúa la categoría de los documentos con un máximo umbral de 1, si una categoría está presente en varios documentos se realiza un cálculo de promedio entre cuantas veces se repite dicha categoría y la cantidad de documentos indexados.
Observaciones:
Prototipo de Interface: No aplica.

Tabla 9: *Historia de usuario #5*

Número: 5	**Nombre de requisito:** Obtener las preferencias implícitas.
Programador: Yossan A. Rodríguez Nuñez	**Iteración Asignada:** 3
Prioridad: Media	**Tiempo:** 24 horas
Riesgo: Alto	**Tiempo real:** 18 horas
Descripción: Se obtiene el nuevo valor de puntuación de las categorías del perfil de preferencias, si las categorías son iguales, se realiza un cálculo de promedio entre el valor de la categoría del documento y el almacenado en el perfil de preferencias de usuario, actualizando así el perfil de preferencias de usuario.	
Observaciones:	
Prototipo de Interface: No aplica.	

2.5 Arquitectura del sistema

La arquitectura basada en componentes es una rama de la Ingeniería de *Software*, que realiza énfasis en descomponer el *software* en componentes funcionales. La descomposición permite convertir componentes preexistentes en piezas más grandes de *software*, este proceso de construcción de una pieza de *software*, da origen al principio de reutilización del *software* (Scribd, 2018).

Un componente es un objeto de software específicamente diseñado para cumplir con cierto propósito. Los principios fundamentales cuando se diseña un componente es que estos deben ser:

- **Reusable**. Los componentes son usualmente diseñados para ser utilizados en escenarios diferentes por diferentes aplicaciones, sin embargo, algunos componentes pueden ser diseñados para tareas específicas.
- **Sin contexto específico**. Los componentes son diseñados para operar en diferentes ambientes y contextos. Información específica como el estado de los datos deben ser pasadas al componente en vez de incluirlos o permitir al componente acceder a ellos.
- **Extensible**. Un componente puede ser extendido desde un componente existente para crear un nuevo comportamiento.
- **Encapsulado**. Los componentes exponen interfaces que permiten al programa usar su funcionalidad. Sin revelar detalles internos, detalles del proceso o estado.
- **Independiente**. Los Componentes están diseñados para tener una dependencia mínima de otros componentes. Por lo tanto, los componentes pueden ser instalados en el ambiente adecuado sin afectar otros componentes o sistemas.

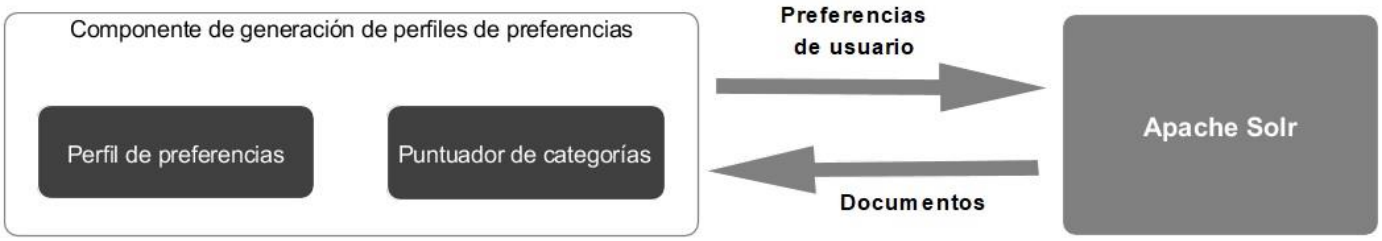


Figura 6: *Arquitectura del componente de generación de perfiles de preferencias.*

El componente **Apache Solr** es el encargado de procesar las peticiones y consultar las preferencias de usuario para obtener e indexar los documentos relevantes para este, posteriormente el **componente de generación de perfiles de preferencias** consultará los documentos obtenidos para puntuar las categorías de los documentos y actualizar el perfil de preferencias.

2.6 Patrones de diseño

Los patrones de diseño representan la descripción de un problema particular y recurrente, que aparece en contextos específicos y presenta un esquema genérico para su solución; este último se especifica mediante la descripción de los componentes que la constituyen, sus responsabilidades, desarrollos y la forma de colaboran entre sí (Larman, 2004).

GRASP

Según Larman (2004), son enfocados en los principios fundamentales de asignación correcta de responsabilidades en el diseño orientado a objetos. Describen los patrones que forman la solución y su aplicación en el desarrollo de los componentes. A continuación, se muestran los patrones utilizados y un ejemplo de cómo se evidencia en la aplicación.

Creador: guía la asignación de responsabilidades relacionadas con la creación de objetos, la intención básica del patrón Creador es encontrar un creador que necesite conectarse al objeto creado en alguna situación (Larman, 2004). En el componente de generación de perfiles de preferencias se encuentra la clase **Preferences_profile**, responsable de crear y cargar los objetos necesarios para el resto de las clases existentes.

Bajo Acoplamiento: impulsa la asignación de responsabilidades de forma en que su localización no incremente el acoplamiento hasta un nivel de resultados negativos que pueden producir un alto acoplamiento. El Bajo Acoplamiento soporta el diseño de clases que son más independientes, reduciéndose el impacto al cambio (Larman, 2004). Las clases **Score_provider** y **Connect_Solr** no tienen relaciones entre ellas, solo se comunican con la clase **Preferences_profile**, y pueden ser reutilizadas como componentes para otros sistemas sin que ocurran grandes impactos por los cambios.

Alta cohesión: La información que almacena una clase es coherente y está relacionada con la clase (Larman, 2004). En el componente es utilizado este patrón en las clases **Score_provider**, **Connect_Solr** y **Preferences_profile**.

Experto: plantea que se debe asignar una responsabilidad al experto en información, o la clase que tiene los datos necesarios para cumplir con este patrón (Larman, 2004). La clase **Category** es un ejemplo de Experto, encargada de la puntuación de las categorías de cada documento y es experta en esa función.

GoF

Describen las formas comunes en que diferentes tipos de objetos pueden ser organizados para trabajar unos con otros. Tratan la relación, combinación y la formación de estructuras de mayor complejidad entre clases (Guerrero et al., 2013). En el desarrollo del componente se identificó ***Iterator*** (Iterador), encargado de realizar recorridos sobre objetos compuestos de forma independiente a su implementación. A continuación, se muestra un ejemplo de su utilización en una lista.

```
Iterator<SolrDocument> iter = list.iterator();
```

2.7 Diagrama de clases de diseño

El diagrama de clases de diseño describe gráficamente las especificaciones de las clases de *software* (Larman, 2004). En el diagrama de clases del componente a desarrollar, se muestra la pertenencia de clases con sus relaciones.

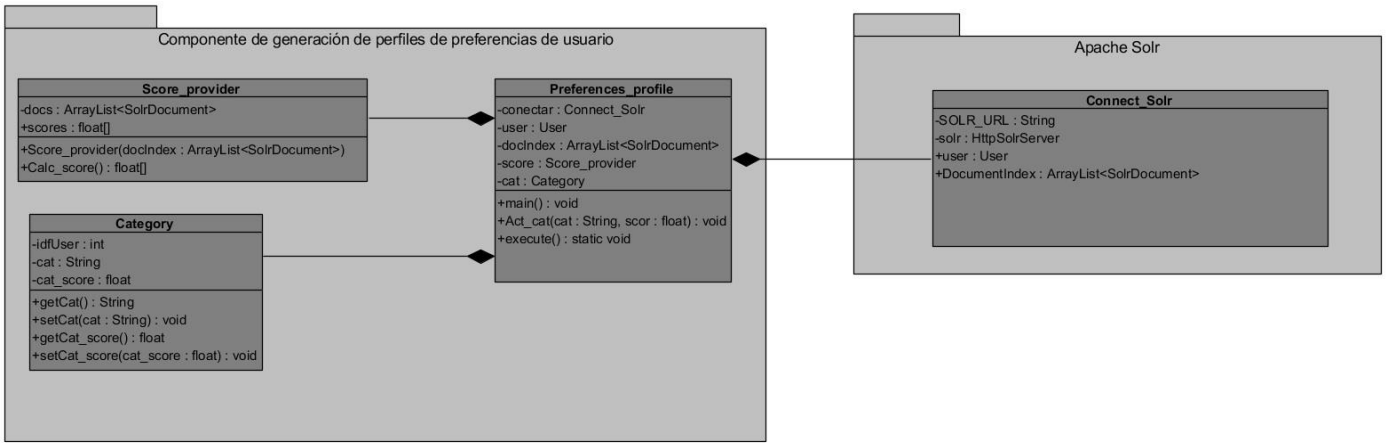


Figura 7: *Diagrama de clases de diseño del componente de generacion de perfiles de preferencias de usuario*

Score_provider: Es la clase encargada de calcular la puntuación de las categorías de los documentos.

Categorizar: Es la clase principal, encargada de establecer el flujo de datos entre el resto de clases.

Category: Es la clase encargada de almacenar el valor de relevancia de cada categoría para el usuario.

Conectar_Solr: Es la clase encargada del flujo de comunicación con el servidor Apache Solr.

2.8 Modelo de datos

Orientado a presentar los elementos que procesan el sistema, la composición y atributos de los mismos, muestra la ubicación en donde serán almacenados estos objetos, su relación y los procesos que los transforman (Pressman, 2010). A continuación, se muestra el modelo de datos con que será implementado el componente de generación de perfiles de preferencias de usuarios.

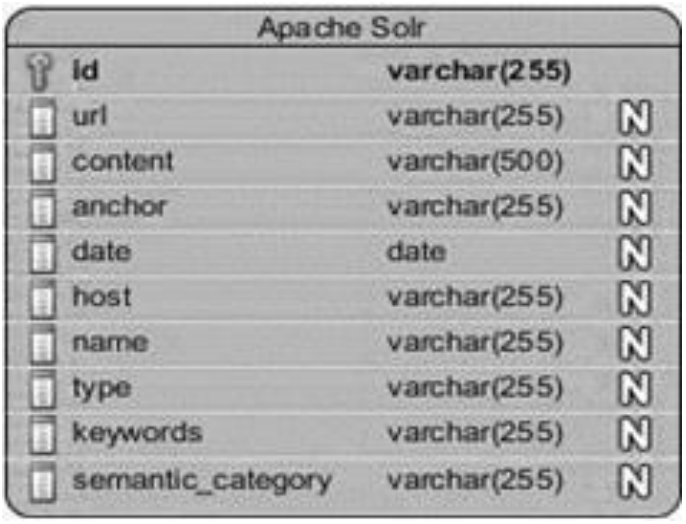


Figura 8: *Diagrama de modelo de datos de Apache Solr*

En la Figura 8 se muestra el nombre de los campos y su formato para el modelo de datos de **Apache Solr**. Para el desarrollo del componente se requiere la extracción del valor de los campos "***id***" y "***content***". Los nuevos campos "***keywords***" y "***semantic_category***", almacenan las palabras clave, y las categorías del documento respectivamente.

2.9 Modelo de despliegue

Un modelo de despliegue representa la relación física que se establece entre los distintos componentes o nodos que describen la topología de un sistema. Estos definen la configuración de los elementos de *hardware*, y detalla cómo los elementos y artefactos del *software* se relacionan en esos nodos (SparxSystems, 2018).

El *hardware* del servidor de indexación Apache Solr debe poseer como requisitos mínimos 16 Gb de memoria RAM y un CPU de 4 núcleos a 3.0 GHz de velocidad. El servidor debe tener 2 Tb de espacio de almacenamiento para los documentos almacenados.

Los componentes trabajaran en cualquier sistema operativo que soporte las librerías de Java 8, pero se recomienda utilizar una distribución GNU/Linux para la instalación de todos los componentes.

Figura 1: *Diagrama de despliegue del componente para la generación de perfiles de preferencias de usuarios.*

La Figura 9 muestra el nodo **Apache Solr** como servidor de indexación; esta herramienta realiza consultas al nodo **componente para la generación de perfiles de preferencias de usuarios** mediante el protocolo HTTPS en el puerto 443.

Conclusiones del capítulo

Con el estudio de los temas referentes al análisis y diseño de la propuesta del componente para la generación de perfiles de preferencias de usuarios se pudo arribar a las siguientes conclusiones:

- La descripción de la propuesta solución se adecua para la solución de la problemática planteada y establece las bases para el correcto desarrollo de la propuesta solución.
- El modelo de dominio se corresponde a la representación visual de los objetos del mundo que interactúan entre sí para la resolución del problema.

- Los requisitos funcionales y no funcionales obtenidos a partir del levantamiento de requisitos responden a que la propuesta de solución satisface las necesidades del cliente.
- Los artefactos generados según la metodología AUP-UCI, la arquitectura y los patrones de diseño crearon las bases necesarias para la construcción de la propuesta de solución.

Capítulo 3: Validación del componente para el modelado de perfiles de preferencias de usuarios en el sistema de recuperación de información Orión

En el presente capítulo se caracterizan los elementos definidos en el proceso de desarrollo, para obtener un producto de calidad. Es presentado y descrito el diagrama de componentes para una mayor comprensión de los elementos físicos del sistema y sus relaciones. En esta sección se incluye la especificación del estándar de codificación desarrollado para la implementación de la aplicación. Se realizan las pruebas del software, con el objetivo de comprobar el correcto funcionamiento del componente y validar la propuesta de solución.

3.1 Diagrama de componentes

El diagrama de componentes permite visualizar la estructura de un sistema informático mediante el análisis de las partes que lo conforman. Los componentes son utilizados como una unidad de composición independiente e indispensable dentro de un sistema, donde se identifica las dependencias que existen entre cada uno de los elementos. Algunos ejemplos de componentes físicos lo constituyen los archivos, módulos, librerías, ejecutables y binarios (Sommerville, 2005).

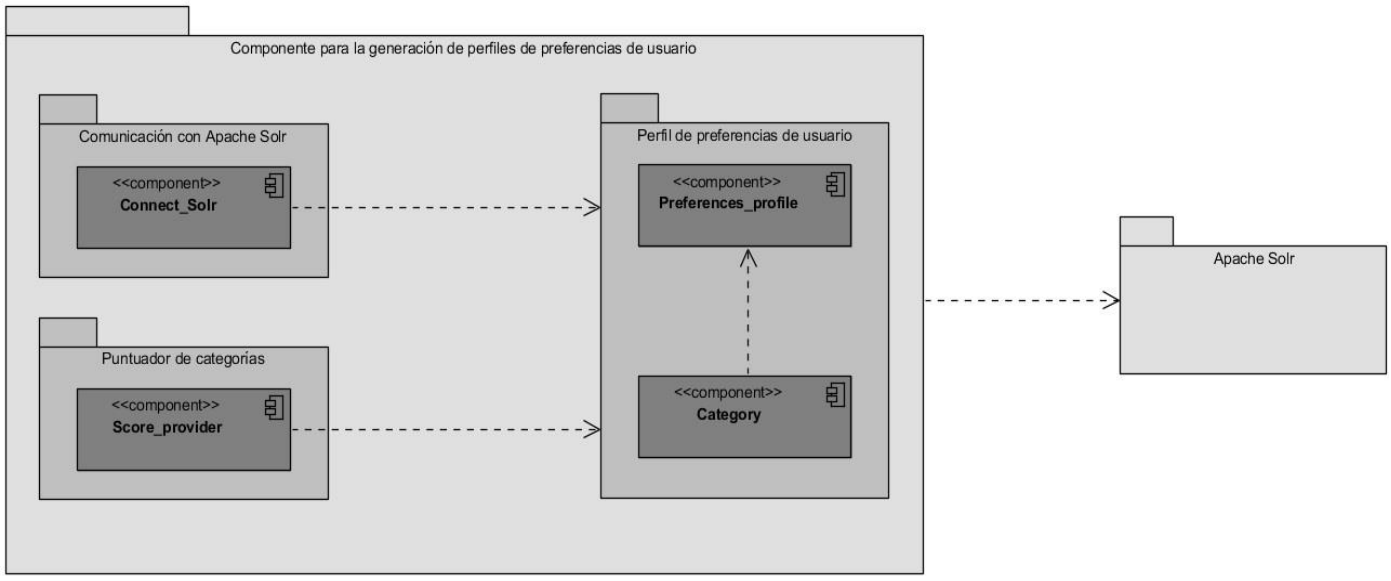


Figura 9: *Diagrama de componentes del Componente para la generación de perfiles de preferencias de usuario*

Descripción de los componentes del diagrama

- **Comunicación con Apache Solr:** Este componente es el responsable de la extracción, inserción y actualización de información con el servidor de indexación **Apache Solr**. Contiene la clase **Connect_Solr** que posee atributos como (user) necesario para identificar el usuario respectivo a cada perfil de preferencias y el atributo (DocumentIndex) que es una lista de documento indexados relevantes para el usuario según el servidor de indexación **Apache Solr**.
- **Puntuador de Categorías:** Contiene la clase *Score_provider* que se encarga de puntuar las categorías presentes en los documentos indexados. Este componente tiene como resultado una lista de números *float* que representan el valor de relevancia de cada categoría en la lista de documentos indexados.
- **Perfil de preferencias de usuario:** Este componente contiene la clase principal *Preferences_profile*, responsable de solicitar y brindar toda la información requerida por el resto de los componentes. Dentro del componente también está la clase *Category*, que almacena el nombre de la categoría y el valor de relevancia de esta para el usuario.
- **Apache Solr:** motor de búsqueda basado en la biblioteca Java del proyecto Lucene, con APIs en XML/HTTP y JSON, resaltado de resultados, búsqueda por facetas, caché, y una interfaz para su administración.

3.2 Estándar de codificación

Los estándares de codificación son un conjunto de reglas o patrones de codificación a seguir por los desarrolladores con el objetivo de establecer un orden y un formato común en el código fuente del sistema en desarrollo. Estos estándares cumplen con el principio de legibilidad y mantenibilidad, correspondientes al objetivo de que el programador entienda el código y sea capaz de modificarlo (Microsoft, 2016). En la implementación del componente es utilizado la siguiente estandarización:

- ✓ Se utiliza el margen adicional de cuatro espacios para diferenciar niveles de programación.

```
class Connect_Solr {
    private String SOLR_URL = "http://localhost:8983/solr/ecured";
```

Figura 10: *Fragmento del código fuente de la clase Connect_Solr.*

- ✓ Las variables son declaradas de forma independiente, en líneas nuevas y no en las mismas sentencias.

```
Score_provider score = new Score_provider();
score.Calc_score();

Connect_Solr conectar = new Connect_Solr();
```

Figura 11: *Fragmento del código fuente de la clase Preferences_profile.*

Las implementaciones de las clases tienen la siguiente estructura:

1. Atributos de la clase.
2. Constructores.
3. Métodos de acceso.
4. Métodos.

```
class Score_provider {

    private ArrayList<SolrDocument> docs ;
    public float [] scores = new float [6];

    public Score_provider(ArrayList<SolrDocument> docIndex) throws FileNotFoundException, IOException {
        docs = docIndex;
        scores = Calc_Score();

    }

    public float[] Calc_Score() throws FileNotFoundException, IOException{
        float [] temp = new float [6];
        for(int i = 0; i <= docs.size(); i++){
            if(docs.get(i).getCategory().equals("Ciencia")){
              temp[0] += 1;
            } else if(docs.get(i).getCategory().equals("Cultura")){
              temp[1] += 1;
            } else if(docs.get(i).getCategory().equals("Actualidad")){
              temp[2] += 1;
            } else if(docs.get(i).getCategory().equals("Política")){
              temp[3] += 1;
            } else if(docs.get(i).getCategory().equals("Tecnología")){
              temp[4] += 1;
            } else if(docs.get(i).getCategory().equals("Deporte")){
              temp[5] += 1;
            }
        }
        for(int i = 0; i <= 6; i++){
            temp[i] /= docs.size();
        }
        return temp;
    }

}
```

Figura 12: *Fragmento del código fuente de la clase Score_provider.*

- ✓ Los atributos relacionados se declaran en una misma línea. El resto se declara en líneas separadas.

```
private String SOLR_URL = "http://localhost:8983/solr/ecured";
private HttpSolrServer solr = new HttpSolrServer(SOLR_URL);

private ArrayList<Category> UserCatScore;
```

Figura 13: *Fragmento del código fuente de la clase Connect_Solr.*

- ✓ Los comentarios se declaran con una gramática correcta y contienen los signos de puntuación apropiados.

```
//put the categories of the document into a variable.
ArrayList<String> docs = document.getCategory();
//declare the variable to return.
float [] score = null;
```

Figura 14: *Fragmento del código fuente de la clase Score_provider.*

Para el correcto funcionamiento del componente para la generación de perfiles de preferencias el servidor de indexación **Apache Solr** provee al componente con un usuario y una lista de documentos indexados por **Solr** a través de la clase **Connect_Solr** encargada de la comunicación entre el componente y el servidor de indexación **Apache Solr**.

```
class Connect_Solr {
    private final String SOLR_URL = "http://localhost:8983/solr/ecured";
    private final HttpSolrServer solr = new HttpSolrServer(SOLR_URL);

    User user = solr.getUser();

    ArrayList<SolrDocument> DocumentIndex = solr.getDocumentList();

}
```

Figura 15: *Fragmento del código fuente de la clase Connect_Solr.*

La clase **Preferences_profile** se comporta como clase principal y es la encargada de tomar los datos ofrecidos para la conexión y trabajar con ellos, esta clase se encarga de la comunicación entre diferentes componentes, guía la asignación de responsabilidades relacionadas con la creación de objetos, obtiene la lista de documentos indexados y el usuario.

```
public class Preferences_profile {

    private Connect_Solr conectar;
    private User user;
    private ArrayList<SolrDocument> docIndex;
    private Score_provider score;
    private Category cat;

    public void main()throws FileNotFoundException, IOException{
        conectar = new Connect_Solr();
        user = conectar.user;
        docIndex = conectar.DocumentIndex;
        score = new Score_provider(docIndex);
        cat = new Category();
    }

    public void Act_cat()throws FileNotFoundException, IOException{
        float [] temp = score.scores;
        if (user.getIdk() == cat.getIdfUser() && cat.getCat().equals("Ciencia")){
            cat.setCat_score((cat.getCat_score() + temp[0])/2);
        } else if (user.getIdk() == cat.getIdfUser() && cat.getCat().equals("Cultura")){
            cat.setCat_score((cat.getCat_score() + temp[1])/2);
        } else if (user.getIdk() == cat.getIdfUser() && cat.getCat().equals("Actualidad")){
            cat.setCat_score((cat.getCat_score() + temp[2])/2);
        } else if (user.getIdk() == cat.getIdfUser() && cat.getCat().equals("Política")){
            cat.setCat_score((cat.getCat_score() + temp[3])/2);
        } else if (user.getIdk() == cat.getIdfUser() && cat.getCat().equals("Tecnología")){
            cat.setCat_score((cat.getCat_score() + temp[4])/2);
        } else if (user.getIdk() == cat.getIdfUser() && cat.getCat().equals("Deporte")){
            cat.setCat_score((cat.getCat_score() + temp[5])/2);
        }
    }

    public static void execute(){}

}
```

Figura 16: Fragmento del código fuente de la clase *Preferences_profile.*

Mediante la variable privada *score* la clase **Preferences_profile** realiza el cálculo de relevancia de cada categoría presente en los documentos indexados utilizando la clase **Score_provider.**

```
class Score_provider {

    private ArrayList<SolrDocument> docs ;
    public float [] scores = new float [6];

    public Score_provider(ArrayList<SolrDocument> docIndex) throws FileNotFoundException, IOException {
        docs = docIndex;
        scores = Calc_Score();

    }

    public float[] Calc_Score() throws FileNotFoundException, IOException{
        float [] temp = new float [6];
        for(int i = 0; i <= docs.size(); i++){
            if(docs.get(i).getCategory().equals("Ciencia")){
              temp[0] += 1;
            } else if(docs.get(i).getCategory().equals("Cultura")){
                temp[1] += 1;
            } else if(docs.get(i).getCategory().equals("Actualidad")){
                temp[2] += 1;
            } else if(docs.get(i).getCategory().equals("Politica")){
                temp[3] += 1;
            } else if(docs.get(i).getCategory().equals("Tecnologia")){
                temp[4] += 1;
            } else if(docs.get(i).getCategory().equals("Deporte")){
                temp[5] += 1;
            }
        }
        for(int i = 0; i <= 6; i++){
            temp[i] /= docs.size();
        }
        return temp;
    }

}
```

Figura 17: Fragmento del código fuente de la clase *Score_provider.*

La clase **Score_provider** proporciona la puntuación de las categorías presentes en los documentos indexados, y con esta puntuación la clase **Preferences_profile** utiliza el método **Act_cat** para actualizar la puntuación de cada categoría perteneciente al perfil de preferencias de usuario en la clase **Category.**

```
public class Category {
    private static int idfUser;
    private String cat;
    private float cat_score;

    public int getIdfUser() {
        return idfUser;
    }

    public String getCat() {
        return cat;
    }

    public float getCat_score() {
        return cat_score;
    }

    public void setCat(String cat) {
        this.cat = cat;
    }

    public void setCat_score(float cat_score) {
        this.cat_score = cat_score;
    }

}
```

Figura 18: Fragmento del código fuente de la clase *Category.*

Como resultado de la actualización de la clase **Category** se obtienen las preferencias implícitas del usuario.

3.3 Estrategia de pruebas

Según Navarro et al. (2009) las pruebas de *software* comprenden una fase del proceso de desarrollo que se centra en asegurar la calidad, fiabilidad y robustez de un *software*, dentro de un contexto o escenario donde está previsto que este sea utilizado. Se encuentra encaminado a medir el cumplimiento de las funcionalidades establecidas por el cliente, reduciendo de esta manera el número de errores no detectados. Esta fase es una de las más costosas del ciclo de vida del *software*.

- ✓ Pruebas unitarias.
- ✓ Pruebas de integración.
- ✓ Pruebas funcionales.

Pruebas unitarias

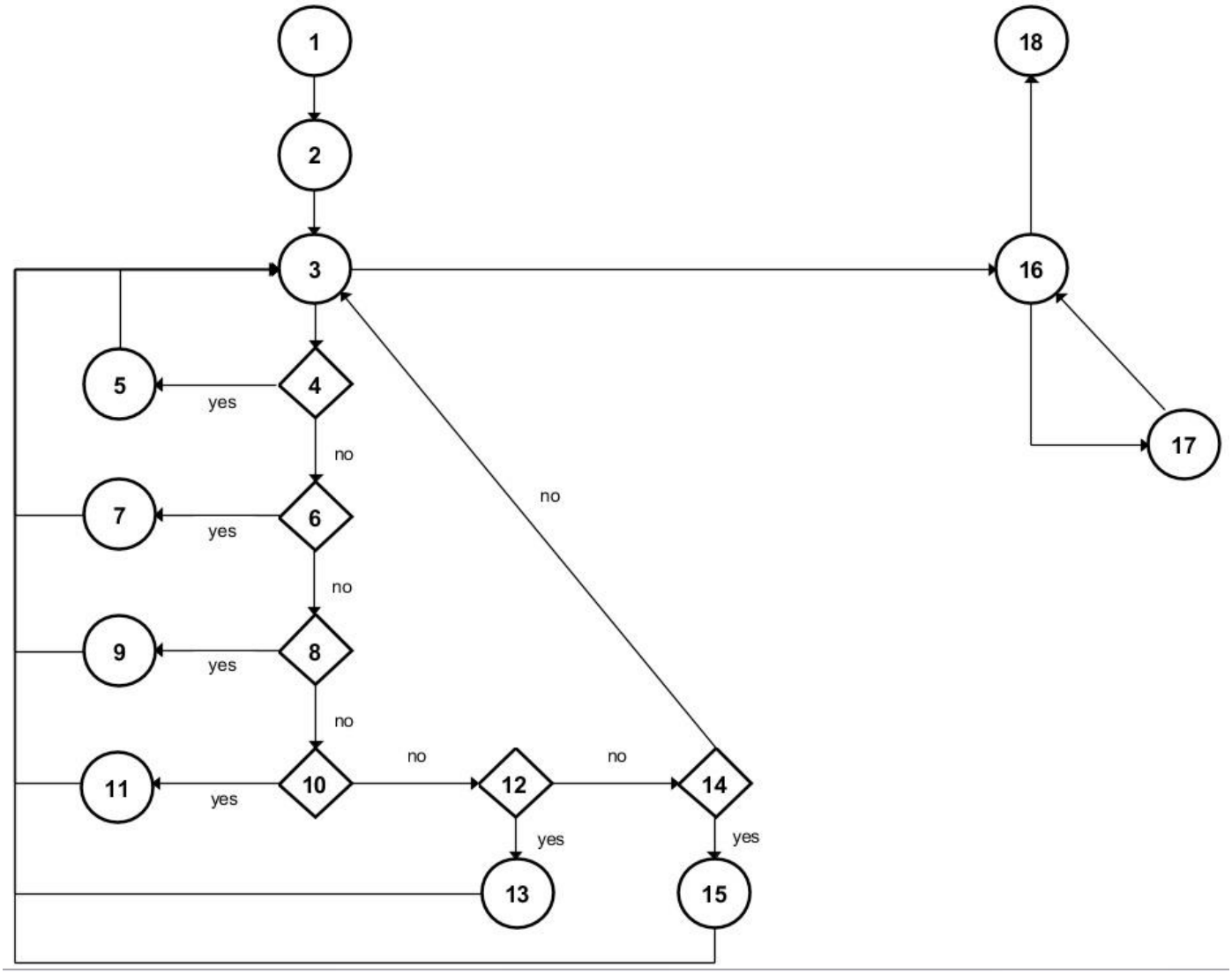
Las pruebas unitarias realizadas utilizan el método de caja blanca. Las unidades de prueba más pequeñas son las operaciones dentro de la clase. Las pruebas unitarias consisten en aislar una parte del código y

comprobar que funciona a la perfección. Son pequeños *tests* que validan el comportamiento de un objeto y la lógica. Las pruebas unitarias suelen realizarse durante la fase de desarrollo de aplicaciones de software o móviles y normalmente las llevan a cabo los desarrolladores (yeeply, 2019). Se decide utilizar el método de camino básico que permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño procedural y utilizar esa medida como guía para la definición de un conjunto básico de caminos de ejecución (Pressman, 2006). A continuación, se realiza la técnica de camino básico al método *Calc_Score* de la clase **Score_provider**.

```
1  public float[] Calc_Score() throws FileNotFoundException, IOException {
2      float[] temp = new float[6];
3      for (int i = 0; i <= docs.size(); i++) {
4          if (docs.get(i).getCategory().equals("Ciencia")) {
5              temp[0] += 1;
6          } else if (docs.get(i).getCategory().equals("Cultura")) {
7              temp[1] += 1;
8          } else if (docs.get(i).getCategory().equals("Actualidad")) {
9              temp[2] += 1;
10         } else if (docs.get(i).getCategory().equals("Política")) {
11             temp[3] += 1;
12         } else if (docs.get(i).getCategory().equals("Tecnología")) {
13             temp[4] += 1;
14         } else if (docs.get(i).getCategory().equals("Deporte")) {
15             temp[5] += 1;
           }
       }
16     for (int i = 0; i <= 6; i++) {
17         temp[i] /= docs.size();
       }
18     return temp;
   }
```


Figura 19: Fragmento del código fuente de la clase *Score_provider.*

Luego de enumerar el código, se diseña la gráfica del método en la Figura 19, que describe el flujo de control lógico mediante la utilización de nodos y aristas.



A partir del grafo obtenido con 18 nodos y 25 aristas se calcula la complejidad ciclomática (V(G)). La complejidad ciclomática se basa en el recuento del número de caminos lógicos individuales contenidos en un programa. Para hallar la complejidad ciclomática, el programa se representa como un grafo, y cada instrucción que contiene, un nodo del grafo. Las posibles vías de ejecución a partir de una instrucción (nodo) se representan en el grafo como aristas.

V(G) = cantidad_aristas – cantidad_nodos + 2

V(G) = 25 – 18 + 2 = 9

Una ruta independiente es cualquier trayecto del programa que ingrese al menos un nuevo conjunto de instrucciones de procesamiento o una nueva condición (Pressman, 2006). La cantidad de rutas independientes se establecen por la complejidad ciclomática; fueron identificadas 9 rutas, tal y como se muestra en la Tabla 10:

Tabla 10: *Listado de caminos independientes.*

No. Ruta	**Camino**
1	1-2-3-16-18
2	1-2-3-16-17-16-18
3	1-2-3-4-6-8-10-12-14-3-16-17-16-18
4	1-2-3-4-5-3-16-17-16-18
5	1-2-3-4-6-7-3-16-17-16-18
6	1-2-3-4-6-8-9-3-16-17-16-18
7	1-2-3-4-6-8-10-11-3-16-17-16-18
8	1-2-3-4-6-8-10-12-13-3-16-17-16-18
9	1-2-3-4-6-8-10-12-14-15-3-16-17-16-18

El valor de V(G) ofrece además un límite superior del número de pruebas que debe diseñarse y ejecutarse para garantizar la cobertura de todas las instrucciones (Pressman, 2006). A continuación, se diseñan casos de pruebas para ser aplicados a cada ruta independiente:

Tabla 11: *Caso de prueba unitaria. Ruta 1.*

Caso de Prueba de Unidad	
No. Ruta: 1	**Ruta:** 1-2-3-16-18
Descripción de la prueba: Obtener valor de puntuación de las categorías de los documentos.	
Entrada: Se trabaja con una lista de documentos tipo Solr que no presentan ninguna de las categorías.	
Resultado esperado: Devolver un arreglo de *floats* vacío.	
Evaluación de la prueba: Satisfactorio. Se obtiene el arreglo vacío.	

Tabla 12: *Caso de prueba unitaria. Ruta 2.*

Caso de Prueba de Unidad	
No. Ruta: 2	**Ruta:** 1-2-3-16-17-16-18
Descripción de la prueba: Obtener valor de puntuación de las categorías de los documentos.	
Entrada: Se trabaja con una lista de documentos tipo Solr que no presentan ninguna de las categorías.	
Resultado esperado: Devolver un arreglo de *floats* vacío.	
Evaluación de la prueba: Satisfactorio. Se obtiene el arreglo vacío.	

Tabla 13: *Caso de prueba unitaria. Ruta 3.*

Caso de Prueba de Unidad	
No. Ruta: 3	**Ruta:** 1-2-3-4-6-8-10-12-14-3-16-17-16-18
Descripción de la prueba: Obtener valor de puntuación de las categorías de los documentos.	

Entrada: Se trabaja con una lista de documentos tipo Solr que no presentan ninguna de las categorías.
Resultado esperado: Devolver un arreglo de *floats* vacío.
Evaluación de la prueba: Satisfactorio. Se obtiene el arreglo vacío.

Tabla 14: *Caso de prueba unitaria. Ruta 4.*

Caso de Prueba de Unidad	
No. Ruta: 4	**Ruta:** 1-2-3-4-5-3-16-17-16-18
Descripción de la prueba: Obtener valor de puntuación de las categorías de los documentos.	
Entrada: Se trabaja con una lista de documentos tipo Solr que presentan la categoría “Ciencia”.	
Resultado esperado: Devolver un arreglo de *floats* con la puntuación de las categorías de los documentos.	
Evaluación de la prueba: Satisfactorio. Se obtiene el arreglo con las categorías de los documentos.	

Tabla 15: *Caso de prueba unitaria. Ruta 5.*

Caso de Prueba de Unidad	
No. Ruta: 5	**Ruta:** 1-2-3-4-6-7-3-16-17-16-18
Descripción de la prueba: Obtener valor de puntuación de las categorías de los documentos.	
Entrada: Se trabaja con una lista de documentos tipo Solr que presentan la categoría “Cultura”.	
Resultado esperado: Devolver un arreglo de *floats* con la puntuación de las categorías de los documentos.	
Evaluación de la prueba: Satisfactorio. Se obtiene el arreglo con las categorías de los documentos.	

Tabla 16: *Caso de prueba unitaria. Ruta 6.*

Caso de Prueba de Unidad	
No. Ruta: 6	**Ruta:** 1-2-3-4-6-8-9-3-16-17-16-18
Descripción de la prueba: Obtener valor de puntuación de las categorías de los documentos.	
Entrada: Se trabaja con una lista de documentos tipo Solr que presentan la categoría “Actualidad”.	
Resultado esperado: Devolver un arreglo de *floats* con la puntuación de las categorías de los documentos.	
Evaluación de la prueba: Satisfactorio. Se obtiene el arreglo con las categorías de los documentos.	

Tabla 17: *Caso de prueba unitaria. Ruta 7.*

Caso de Prueba de Unidad	
No. Ruta: 7	**Ruta:** 1-2-3-4-6-8-10-11-3-16-17-16-18
Descripción de la prueba: Obtener valor de puntuación de las categorías de los documentos.	
Entrada: Se trabaja con una lista de documentos tipo Solr que presentan la categoría “Política”.	
Resultado esperado: Devolver un arreglo de *floats* con la puntuación de las categorías de los documentos.	
Evaluación de la prueba: Satisfactorio. Se obtiene el arreglo con las categorías de los documentos.	

Tabla 18: *Caso de prueba unitaria. Ruta 8.*

Caso de Prueba de Unidad	
No. Ruta: 8	**Ruta:** 1-2-3-4-6-8-10-12-13-3-16-17-16-18
Descripción de la prueba: Obtener valor de puntuación de las categorías de los documentos.	
Entrada: Se trabaja con una lista de documentos tipo Solr que presentan la categoría “Tecnología”.	
Resultado esperado: Devolver un arreglo de *floats* con la puntuación de las categorías de los documentos.	
Evaluación de la prueba: Satisfactorio. Se obtiene el arreglo con las categorías de los documentos.	

Tabla 19: *Caso de prueba unitaria. Ruta 9.*

Caso de Prueba de Unidad	
No. Ruta: 9	**Ruta:** 1-2-3-4-6-8-10-12-14-15-3-16-17-16-18
Descripción de la prueba: Obtener valor de puntuación de las categorías de los documentos.	
Entrada: Se trabaja con una lista de documentos tipo Solr que presentan la categoría “Deporte”.	
Resultado esperado: Devolver un arreglo de *floats* con la puntuación de las categorías de los documentos.	
Evaluación de la prueba: Satisfactorio. Se obtiene el arreglo con las categorías de los documentos.	

Pruebas de integración

Las pruebas de integración se centran en comprobar que los módulos probados por separado funcionen en conjunto, con el objetivo de verificar que interactúan correctamente a través de sus interfaces y cubren las funcionalidades establecidas en los requisitos (Pressman, 2010).

Las pruebas de integración implican probar diferentes módulos de una aplicación de software como grupo. Una aplicación de software se compone de diferentes submódulos que trabajan juntos para diferentes funcionalidades. El propósito de las pruebas de integración es validar la integración de diferentes módulos juntos e identificar los errores y problemas relacionados con ellos (loadview, 2018).

- Se asegura de que todos los módulos de aplicación estén bien integrados y funcionen juntos según lo esperado.
- Detecta problemas y conflictos interconectados para resolverlos antes de crear un gran problema.
- Valida la funcionalidad, fiabilidad y estabilidad entre diferentes módulos.
- Detecta excepciones ignoradas para mejorar la calidad del código.

Pruebas Funcionales

Las pruebas funcionales se llevan a cabo para comprobar las características críticas para el negocio, la funcionalidad y la usabilidad. Las pruebas funcionales garantizan que las características y funcionalidades del software se comportan según lo esperado sin ningún problema. Valida principalmente toda la aplicación con respecto a las especificaciones mencionadas en el documento “Software Requirement Specification” (SRS).

- Se asegura de que el sitio web / aplicación está libre de defectos.
- Garantiza el comportamiento esperado de todas las funcionalidades.
- Garantiza que la arquitectura sea correcta con la seguridad necesaria.
- Mejora la calidad y las funcionalidades generales.
- Minimiza los riesgos empresariales asociados con el sitio web/aplicación.

Se propone realizar las pruebas funcionales con los datos presentes en la tabla 20:

Tabla 20: *Caso de prueba a la HU "Puntuar Categorías"*

Escenario	Descripción	Entrada	Flujo Central	R/ del componente
EC 1.1 Puntuar Categorías	El componente se encarga de puntuar las categorías presentes en los documentos indexados	Doc1. Categoría: Ciencia Doc2. Categoría: Actualidad Doc3. Categoría: Deporte : Ciencia	El componente puntúa las categorías presentes en los documentos indexados calculando la cantidad de veces que se repite la categoría con respecto a la cantidad de documentos indexados	Ciencia: 0.66 Actualidad: 0.33 Deporte: 0.33

Tabla 21: *Caso de prueba a la HU "Obtener preferencias implícitas"*

Escenario	Descripción	Entradas	Flujo Central	R/ del componente
EC 1.1 Obtener preferencias implícitas	El componente se encarga de actualizar el perfil de preferencias del usuario.	Usuario1. Categorías: Ciencia: 0.56 Cultura: 0.50 Actualidad: 0.24 Política: 0.45 Deporte: 0.21 R/ de Puntuar Categorías Ciencia: 0.66 Actualidad: 0.33 Deporte: 0.33	El componente toma la respuesta del componente Puntuar Categorías y utiliza la puntuación de las categorías indexadas para actualizar el perfil de preferencias del usuario.	Usuario1. Categorías: Ciencia: 0.61 Cultura: 0.50 Actualidad: 0.28 Política: 0.45 Deporte: 0.27

Resultados de la prueba:

Como se evidencia en la tabla 22 el perfil de preferencias se actualiza luego de transitar por los componentes. Se puede apreciar que las categorías presentes en los documentos indexados aumentan su puntuación de relevancia para el usuario.

Tabla 22: *Comparación de los perfiles de preferencias.*

Usuario1	
Perfil de preferencias	**Perfil de preferencias actualizado**
Ciencia: 0.56	Ciencia: 0.61
Cultura: 0.50	Cultura: 0.50

Actualidad: 0.24	Actualidad: 0.28
Política: 0.45	Política: 0.45
Deporte: 0.21	Deporte: 0.27

A medida que el usuario interactúe con el sistema sus preferencias se verán actualizadas en tiempo real. garantizando que los documentos indexados estén en concordancia con las necesidades de búsqueda del usuario.

Conclusiones del capítulo

En este capítulo se abordaron aspectos correspondientes a la implementación y validación del componente de categorización semántica de documentos, arribándose a las siguientes conclusiones:

- El diagrama de componentes facilitó la comprensión de la estructura general de la aplicación.
- Al aplicar los estándares de codificación se logró adoptar una estructura homogénea que facilita la comunicación y asegura la calidad, menos errores y fácil mantenimiento.
- Se describe el funcionamiento del componente con ejemplos de código, permitiendo visualizar la programación del componente.
- La aplicación de pruebas unitarias, funcionales e integración permitieron identificar las principales deficiencias en el desarrollo del componente en etapas tempranas, así como solucionar los errores detectados y obtener un producto con un alto valor, pertenencia y utilidad.

Conclusiones

- El análisis de los referentes teóricos permitió definir la utilización del modelo de perfil de preferencias idóneo para el desarrollo de aplicación.
- La modelación de los artefactos permitió obtener una arquitectura sólida de la aplicación y garantizó la base para la organización lógica del código fuente.
- La implementación del componente de generación de perfiles de usuario mejora las respuestas otorgados a los usuarios en el Sistema de Recuperación de Información Orión y proporciona una solución aceptable a la situación problemática existente.
- La propuesta de pruebas escogida permitirá validar el correcto funcionamiento de la propuesta de solución.

Referencias

A learning approach to personalized information filtering. **Sheth, Beerud Dilip. 1994.** s.l. : Massachusetts Institute of Technology., 1994.

A multi-agent system using ontological user profiles for dynamic user modelling. **HAWALAH, Ahmad and FASLI, Maria.** International Conferences on Web Intelligence and Intelligent Agent Technology., pp. 430-437.

Agile and iterative development: a manager's guide. **Larman, C. 2004.** 2004, Addison-Wesley Professional.

Algorithm for calculating relevance of documents in information retrieval systems. **Baquerizo, Roberto Passailaigue and al., et. 2017.** 03, Marzo 2017, International Research Journal of Engineering and Technology (IRJET), Vol. 04. 2395 -0056.

Alipes: A swift messenger in cyberspace. **Widyantoro, D.H., et al. 1999.** 1999, Proceedings of Spring Symposium Workshop on Intelligent Agents in Cyberspace, pp. 62-67.

Amalthaea information discovery and filtering using a multiagent evolving ecosystem. **Moukas, Alexandros. 1997.** 5, 1997, Applied Artificial Intelligence, Vol. 11, pp. 437-457.

Anatomy and Empirical Evaluation of an Adaptive Web-Based Information Filtering System. User Modeling and User-Adapted Interaction. **Micarelli, A. and Sciarrone, F. 2004.** 2-3, 2004, Vol. 14, pp. 159-200.

Apache Solr: un motor de búsqueda de código abierto. **Ramos, L. M. E. 2012.** 2012.

ARQUITECTURA DE HARDWARE PARA SISTEMAS DE RECUPERACIÓN DE INFORMACIÓN. **Leyva, P. R., et al. 2018.** 2018.

Baeza-Yates, R. 2019. Página Web de Ricardo Baeza-Yates. [Online] 2019.

Building and applying a concept hierarchy representation of a user profile. **NANAS, Nikolaos, UREN, Victoria and DE ROECK, Anne. 2003.** 2003, Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval., pp. 198-204.

Capturing interest through inference and visualization: Ontological user profiling in recommender systems. **Middleton, S.E., Shadbolt, N.R. and De Roure, D.C. 2003.** 2003, International Conference on Knowledge Capture,.

Collaborative and Usage-driven evolution of personalized ontologies. **Haase, P., et al. 2005.** 2005, Proceedings of the 2nd European Semantic Web Conference,, pp. 496-499.

COMPONENTE PARA EL CÁLCULO DE LA RELEVANCIA EN EL BUSCADOR ORIÓN. **García, Alexander and Utrera, Eric. 2019.** 02, 2019, Revista Cubana de Ciencias Informáticas, Vol. 13, pp. 91-104. 2227-1899.

Concept Networks for Personalized Web Search Using Genetic Algorithm. **Remesh, K. R. and Philip, Samuel. 2015.** 2015, Procedia Computer Science, Vol. 46, pp. 566-573.

Cubanic. *Cubanic.* [Online] [Cited: 06 14, 2021.] http://www.nic.cu/.

Document language models, query models, and risk minimization for information retrieval. **Lafferty, J. and Zhai, C. 2017.** 2, 2017, ACM SIGIR Forum, Vol. 51, pp. 251-259.

DoLaw: buscador semántico especializado para la legislación peruana de tecnologías de información. **Otoya, D. A. 2019.** 2019.

Estudio, diseño y aplicación de técnicas basadas en Soft Computing para la mejora de la búsqueda conceptual en Internet. **Mora, M. 2016.** 2016, Repositorio Universitario Institucional de Recursos Abiertos.

Etiquetado asistido de documentos de investigación mediante procesamiento de lenguaje natural y tecnologías de la web semántica. **Sanchez, P. H. A., et al. 2018.** 4, 2018, Scientia et technica, Vol. 23, pp. 528-537.

Fundamentos epistémicos de la investigación cualitativa y cuantitativa: Consensos y disensos. **Sanchez, F. A. 2019.** 1, 2019, Revista digital de investigación en docencia universitaria, Vol. 13, pp. 102-122.

Genaum: New semantic distributed search engine. **Ensias, M. V. 2017.** 3-4, 2017, Journal of Mobile Multimedia, Vol. 12.

Generic user modeling systems. User modeling and user adapted interaction. **Kobsa, Alfred. 2001.** 2001.

Gestión Eficiente del Índice Invertido para Flujos de Documentos en Tiempo Real. **Ríssola, E. A. and Tolosa, M. G. H. 2015.** [ed.] Universidad Nacional de Luján. Argentina : s.n., 2015.

Haberfellner, R., et al. 2019. *Systems engineering.* s.l. : Springer International Publishing., 2019.

Hierarchical user interest model based on large log data of mobile internet. **Man, Ning et al. 2016.** 2016, 2016 13th International, pp. 1-5.

ifWeb: A Prototype of User Model-Based Intelligent Agent for Documentation Filtering and Navigation in the World Wide Web. **Asnicar, F. and Tasso, C. 1997.** 1997, Proceedings of the 6th International Conference on User Modeling, pp. 3-11.

Implicit feedback for inferring user preference: a bibliography. **Kelly, Diane and Teevan, Jaime. 2003.** 2003, Acm Sigir Forum, Vol. 37, pp. 18-28.

Implicit user modeling for personalized search. **Shen, Xuehua, Tan, Bin and Zhai, ChengXiang. 2005.** 2005, Proceedings of the 14th ACM international conference on Information and knowledge management., pp. 824-831.

Improving ontology-based user profiles. **TRAJKOVA, Joana and GAUCH, Susan. 2004.** 2004, RIAO, pp. 380-390.

Inferring users information context: Integrating user profiles and concept hierarchies. **Sieg, Ahu, Mobasher, Bamshad and Burke, Robin. 2004.** 2004, 2004 Meeting of the International Federation of Classification Societies, IFCS,.

Information Retrieval. **Hersh, W. 2021.** [ed.] Cham. Springer. Shortliffe E.H. : Biomedical Informatics., 2021. 978-3-030-58721-5.

Introducción a la Recuperación de Información. **Tolosa, H. and Bordignon, R. A. 2008.** s.l. : E-prints in library & information sciences., 2008.

Introduction to Modern Information Retrieval. **Salton, G. and McGill, M. 1987.** s.l. : McGraw-Hill, 1987.

ISOTools. 2015. Blog Calidad y Excelencia. *5 ejemplos de indicadores de calidad que no pueden faltar en tu plan.* 2015.

Izaurralde, Maria Paula. 2013. *Caracterización de Especificación de Requerimientos en entornos Ágiles: Historias.* Universidad Tecnológica Nacional : s.n., 2013. s.n..

Learning and teaching styles in engineering education, Engineering Education,. **Felder, R. M. y Silverman, L.K. 1988.** 7, 1988, Vol. 78, pp. 674-681.

Learning user profiles for personalized information dissemination. **Tan, A. and Teo, C. 1998.** Alaska : s.n., 1998, Proceedings of 1998 IEEE International Joint Conference on Neural Networks, pp. 183-188.

loadview. [Online] [Cited: Noviembre 19, 19.] https://www.loadview-testing.com/es/blog/tipos-de-pruebas-de-software-diferencias-y-ejemplos/.

Los sistemas de recuperación de la información (SRI) de las bases de datos documentales y la calidad de los resultados obtenidos. **Romá, M. T. 2014.** Alicante : Repositorio Institucional Unversidad de Alicante., 2014.

Martínez Méndez, Francisco Javier. 2004. *RECUPERACIÓN DE INFORMACIÓN: MODELOS, SISTEMAS Y EVALUACIÓN.* Murcia : s.n., 2004. 84-932537-7-4.

Método Híbrido de Recomendación Adaptativa de Objetos de Aprendizaje basado en Perfiles de Usuario. **Rodríguez, Paula A., Duque, Néstor D. and Ovalle, Demetrio A. 2016.** 4, La Serena : s.n., 2016, Revista SciELO, Vol. 9. 0718-5006.

Metodología de la investigación. **Sampieri, R., Fernández, C. and Baptista, P.** Mexico : McGraw-Hill.

MICROSOFT. 2021. Revisiones de código y estándares de codificación. [Online] Noviembre 19, 2021.

Modelado de perfiles de usuario para la recomendación de contenido en Twitter. **Rodríguez, María Florencia and Godoy, Daniela Lis. 2016.** Argentina : s.n., 2016. 2451-7615.

Modeling Users' Dynamic Preference for Personalized Recommendation. **Liu, Xin. 2015.** 2015, IJCA, pp. 1785-1791.

Modelo de Interfaz Adaptativa basada en Perfiles de Usuario y Ontologías para Recomendación de Objetos de Aprendizaje. **QUIROZ, Thomás, SALAZAR, Oscar M. and OVALLE, Demetrio A. 2018.** [ed.] Departamento de Ciencias de la Computación y de la Decisión, Facultad de Minas Universidad Nacional de Colombia. 6, 2018, Vol. 29, pp. 295-306. 0718-0764.

MODELO PARA LA RECUPERACIÓN DE INFORMACIÓN CON ANOTACIÓN SEMÁNTICA MODEL FOR THE RECOVERY OF INFORMATION WITH SEMANTIC ANNOTATION. **Viltres, H., et al. 2018.** 2018.

Modern information retrieval. **Baeza, R. and Ribeiro, B. 1999.** New York : ACM press., 1999, Vol. 463.

2019. Netec.com. *cuales-son-las-ventajas-de-java-y-sus-usos.* [Online] 2019. https://www.netec.com/post/cuales-son-las-ventajas-de-java-y-sus-usos.

nfoweb: An Adaptive Information Filtering System for the Cultural Heritage Domain. Applied Artificial Intelligence. **Gentili, G., Micarelli, A. and Sciarrone, F. 2003.** 8-9, 2003, Vol. 17, pp. 715-744.

Ontology Based Personalized Search. **Pretschner, A. and Gauch, S. 1999.** 1999, Proceedings of the 11th IEEE International Conference on Tools with Artificial Intelligence (ICTAI), pp. 391-398.

Open HMI Tester: un Framework Open-source para herramientas de pruebas de software. **Navarro, P. L. M., Pérez, G. M. and Ruiz, D. S. 2009.** 4, s.l. : Actas de los Talleres de las Jornadas de Ingeniería del Software y Bases de Datos, 2009, Vol. 3.

ORION: UN MOTOR DE BÚSQUEDAS PARA LA WEB DE LA UCI. **Delgado, Yusniel and al., et. 2010.** 2010.

Perfil del usuario de información. **Henández, P. 2018.** 2018.

Persona: A Contextualized and Personalized Web Search. **Tanudjaja, F. and Mui, L. 2002.** 2002, Proc 35th Hawaii International Conference on System Sciences,.

Personalized information retrieval based on time-sensitive user profile. **Kacem, Ameni. 2017.** 2017.

Pressman, R. S. 2005. *Pressman, R. (2010). Ingeniería del Software Un Enfoque Práctico. 7ma ed.* s.l. : University of Connecticut., 2005.

Procedimiento para la estructuración y almacenamiento de documentos en el Sistema de Recuperación de Información Orión. **Delgado, Yennifer. 2018.** 2018. Universidad de las Ciencias Informáticas.

Procesamiento Semántico de información en Sistemas de Recuperación de Información. **Viltres, H., et al. 2018.** 1, 2018, SciELO, Vol. 12. 2227-1899..

Propuesta de modelado de una ontología de dominio para la representación de acciones en política-economía. **Moreno, M. J. B. 2017.** [ed.] Universidad de Murcia. 2017.

PVA: A self-adaptive Personal View Agent. Journal of Intelligent Information Systems,. **Chen, C., Chen, M. and Sun, Y. 2002.** 2-3, 2002, Vol. 18, pp. 173-194.

Recomendación contextualizada usando una ontología de contexto genérica y de gran escala construida semiautomáticamente a partir de DBpedia. **Rodríguez, Nicolas, et al. 2017.** 2017.

Recomendador de usuarios en una plataforma colaborativa en base a su perfil y reputación. **Aciar, G. and Aciar, S. 2013.** 2013, Presentado en XIV Argentine Symposium on Artificial Intelligence (ASAI)- JAIIO 42 (2013)., pp. 1-11. 1850-2784.

Recommendation System for Major University Determination Based on Studentâ. **Kusumaningrum, Desi Purwanti, et al. 2017.** 1, 2017, Journal of Applied Intelligent System,, Vol. 2, pp. 21-28.

Rodríguez Sánchez, Tamara. 2015. *Metodología de desarrollo para la Actividad productiva de la UCI.* 2015.

SERVQUAL: Review, critique, research agenda. **Buttle, F. 1996.** 1, 1996, European Journal of marketing, Vol. 30, pp. 8-32.

Sommerville, I. 2005. *Ingeniería del software.* s.l. : Pearson educación., 2005.

SPARXSYSTEMS. UML 2 Tutorials. *Deployment Diagrams.* [Online] [Cited: Noviembre 18, 2021.] https://sparxsystems.com/resources/tutorials/uml2/index.html.

Survey of Text mining: Clustering, Classification, and Retrieval. **Berry, M. W. 2004.** New York, USA : s.n., 2004, Springer Verlag.

Técnicas avanzadas de recuperación de información: procesos, técnicas y métodos. **Blázquez, M. 2013.** Madrid : E-Prints Complutense, 2013.

User profiles for personalized information access. In The adaptive web. **Gauch, Susan, et al. 2007.** 2007, Springer, pp. 54-89.

User Profiling Approaches, Modeling, and Personalization. **Farid, Marina, et al. 2018.** s.l. : Proceedings of the 11th International Conference on Informatics & Systems, 2018.

User profiling through deep multimodal fusion. **Farnadi, G., et al. 2018.** 2018, In Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining, pp. 171-179.

Variables relevantes para la medición de la calidad percibida del servicio bancario. **Berdugo, C. R., Barbosa, R. A. and Prada, L. M. 2016.** 2016.

Web search personalization with ontological user profiles. **Sieg, Ahu, Mobasher, Bamshad and Burke, Robin. 2007.** 2007, Proceedings of the sixteenth ACM conference on Conference on information and knowledge management., pp. 525-534.

WebMate: A personal agent for browsing and searching. **Chen, L. and Sycara, K. 1998.** 1998, Proceedings of the second international conference on Autonomous agents, pp. 132-139.

What characterizes a (software) component? **Broy, M., Deimel, A., Henn, J. et al. 1998.** 1998, Software - Concepts & Tools, Vol. 19, pp. 49-56.

yeeply. [Online] [Cited: Noviembre 19, 2021.] https://www.yeeply.com/blog/que-son-pruebas-unitarias/.