

**ESCUELA POLITÉCNICA NACIONAL
FACULTAD DE INGENIERÍA DE SISTEMAS**

**DESARROLLO DE UN SISTEMA DE RECOMENDACIÓN PARA
ARTÍCULOS DE REVISTAS CIENTÍFICAS ABIERTAS UTILIZANDO
TECNOLOGÍAS DE DATOS ENLAZADOS**

**TRABAJO DE TITULACIÓN PREVIO A LA OBTENCIÓN DEL
TÍTULO DE INGENIERO EN SISTEMAS INFORMÁTICOS Y DE
COMPUTACIÓN**

LENIN SANTIAGO MONTALVO LIMA

lenin.montalvo@epn.edu.ec

JONATHAN PAUL PACHACAMA QUINGA

jonathan.pachacama@epn.edu.ec

DIRECTORA: ING. MARIA ASUNCION HALLO CARRASCO, PHD.

maria.hallo@epn.edu.ec

Quito, mayo de 2019

DECLARACIÓN DE AUTORÍA

Nosotros, Montalvo Lima Lenin Santiago y Pachacama Quinga Jonathan Paul, declaramos bajo juramento que el trabajo aquí descrito es de nuestra autoría que no ha sido previamente presentado para ningún grado o calificación profesional; y, que he consultado las referencias bibliográficas que se incluyen en este documento.

A través de la presente declaración cedemos los derechos de propiedad intelectual correspondientes de este trabajo, a la Escuela Politécnica Nacional, según lo establecido por la Ley de Propiedad Intelectual, por su Reglamento y por la normatividad institucional vigente.

Lenin Santiago Montalvo Lima

Jonathan Paul Pachacama Quinga

CERTIFICACIÓN

Certifico que el presente trabajo fue desarrollado por Montalvo Lima Lenin Santiago y Pachacama Quinga Jonathan Paul, bajo mi supervisión.

PhD. María Asunción Hallo Carrasco
DIRECTOR DE PROYECTO

AGRADECIMIENTO

Agradezco a mis padres Silvio Montalvo y Judith Lima por todo su apoyo y cariño incondicional; a mis hermanos Oscar Montalvo y Alex Montalvo por su ayuda; a mi tutora, la Dra. María Hallo, por su guía y ayuda en el proyecto; a mis familiares, profesores y amigos quienes me ayudaron con sus consejos y enseñanzas a lo largo de mi vida estudiantil.

YNWA.

Lenin Montalvo

DEDICATORIA

Dedico este logro es a mis padres Silvio Montalvo y Judith Lima, todos mis méritos a ellos se debe; en cuanto a mis errores, yo soy el único responsable.

Lenin Montalvo

AGRADECIMIENTO

A Dios por darme la vida, por su misericordia que se renueva cada día y sobre todo por su eterno amor que me ha alcanzado.

A mi familia por su paciencia y confianza puesta en mi para terminar todos mis proyectos, por cada consejo, cada palabra de aliento y sobre todo por sus oraciones.

A la PhD María Hallo por su guía a través de este proyecto de titulación.

A todos mis profesores que han aportado de forma directa o indirectamente en mi crecimiento profesional.

A Estefanía Contreras por su amor y su apoyo en todo momento,

Jonathan Pachacama

DEDICATORIA

Dedico este proyecto de titulación a Dios que siempre será lo principal en mi vida.

A mis padres Yolanda y Carlos que con su esfuerzo y amor me han apoyado cada día para avanzar en mi vida profesional, por sus consejos y sobre todo su buen testimonio ya que siempre serán un ejemplo para mí.

A mis hermanos Miriam y Henry por estar siempre a mi lado son una bendición en mi vida.

A todos mis amigos con los que hemos compartido buenos momentos en especial a Estefanía Contreras, mi amada, quien ha sido una pieza fundamental en todo este camino.

Jonathan Pachacama

CONTENIDO

DECLARACIÓN DE AUTORÍA	I
CERTIFICACIÓN	II
AGRADECIMIENTO	III
DEDICATORIA	IV
AGRADECIMIENTO	V
DEDICATORIA	VI
CONTENIDO	VII
ÍNDICE DE FIGURAS	IX
ÍNDICE DE TABLAS	XII
RESUMEN	1
ABSTRACT	2
CAPITULO 1: INTRODUCCIÓN	3
1.1. Descripción del problema	3
1.1.1. Situación actual	3
1.1.2. Situación propuesta	3
1.2. Alcance	4
1.3. La web actual	4
1.4. La web semántica.....	6
1.4.1. El lenguaje RDF	7
1.4.2. Vocabularios RDF y prefijos:	9
1.5. El lenguaje de consulta <i>sparql</i>	10
1.5.1. Operadores SPARQL.....	12
1.5.2. Funciones.....	13
1.6. Interfaz de consulta a repositorios digitales RDF	14
1.6.1. Repositorio digital <i>RKB (Resilience Knowledge Base)</i>	15
1.6.2. Comparativa IEEE, ACM y DBLP.....	17
1.6.3. Extracción de información de un artículo científico	18
1.7. Sistemas de recomendación	20
1.8. Selección de lenguajes y herramientas	23
1.8.1. Tecnología FRONT-END	23
1.8.2. Tecnología BACK-END.....	23
1.8.3. Entornos de desarrollo integrado (IDE).....	24
1.9. Metodología utilizada (SCRUM).....	24
1.9.1. SCRUM	25

1.9.2.	Componentes de <i>Scrum</i>	25
1.9.3.	Eventos de <i>scrum</i>	26
1.9.4.	Artefactos de <i>scrum</i>	27
CAPITULO 2: DESARROLLO DE LA METODOLOGÍA		28
2.1	Roles	28
2.2	Requerimientos	28
2.3	Iteraciones	37
2.3.1.	Primera iteración.....	37
2.3.2.	Segunda iteración	48
2.3.3.	Tercera iteración	54
2.3.4.	Cuarta iteración.....	63
2.4.	Producto final.....	71
CAPITULO 3: PRUEBAS DE ACEPTACIÓN Y EVALUACIÓN DEL SISTEMA		86
3.1.	Requisitos físicos del sistema	86
3.2.	Pruebas de aceptación	87
3.2.1.	Pruebas de aceptación de la primera iteración.....	87
3.2.2.	Pruebas de aceptación de la segunda iteración.....	90
3.2.3.	Pruebas de aceptación de la tercera iteración	94
3.2.4.	Pruebas de aceptación de la cuarta iteración	95
3.3.	Pruebas de integridad.....	97
3.4.	Evaluación del software.....	100
3.4.1.	Encuesta de satisfacción de los usuarios	100
3.4.2.	Análisis de resultados de la encuesta.....	101
CAPITULO 4: CONCLUSIONES Y RECOMENDACIONES		105
4.1.	Conclusiones	105
4.2.	Recomendaciones	105
BIBLIOGRAFÍA:.....		106
GLOSARIO		109
ANEXOS.....		111

ÍNDICE DE FIGURAS

Figura 1. Web actual.....	5
Figura 2. Componentes de la web semántica.	7
Figura 3. Representación de un grafo RDF.....	8
Figura 4. Triple RDF	8
Figura 5. Representación de una URI.....	8
Figura 6. Declaración de prefijo para la DBpedia.	10
Figura 7. Declaración de prefijos en la iniciativa <i>RKB Explorer</i>	10
Figura 8. Ejemplo de consulta SPARQL	11
Figura 9. Resultado de consulta SPARQL	11
Figura 10. Operador filter.....	12
Figura 11. Consulta con operador OPTIONAL.....	13
Figura 12. Consulta SPARQL con operador IsIRI	13
Figura 13. Consulta SPARQL con la función BOUND.....	14
Figura 14. Consulta SPARQL con función IsLiteral.....	14
Figura 15. Ejemplo de la interfaz de consulta de RKB Explorer	16
Figura 16. Consulta SPARQL para extraer información de un artículo científico en la interfaz de <i>RKB Explorer</i>	18
Figura 17. Resultados de consulta SPARQL de la Tabla 5 en formato JSON	19
Figura 18. Esquema del procedimiento de recomendación en <i>lucene [19]</i>	21
Figura 19. Proceso de búsqueda en <i>apache lucene [19]</i>	22
Figura 20. Módulos del sistema de manejo de artículos científicos de acceso directo	40
Figura 21. Arquitectura para mejorar un sistema de gestión de referencia con recomendaciones de datos vinculados abiertos [1].....	42
Figura 22. Sistema de manejo de artículos científicos con la plantilla.....	43
Figura 23. Archivo <i>config.php</i> (wikindx5.3.2).....	44
Figura 24. Pantalla principal wikindx.....	45
Figura 25. interfaz login.	45
Figura 26. Prototipo de interfaz registro de usuario.....	46
Figura 27. Esfuerzo realizado para la primera iteración.	47
Figura 28. Tareas pendientes para la primera iteración.	47
Figura 29. Pantalla de biblioteca.....	51
Figura 30. Barra de búsqueda.	51
Figura 31. Botones biblioteca.....	51
Figura 32. Artículo seleccionado en biblioteca.	52

Figura 33. Eliminar artículo <i>wikindx</i>	52
Figura 34. Editar artículo <i>wikindx</i>	52
Figura 35. Cita bibliográfica <i>wikindx</i>	53
Figura 36. Metadatos del recurso en <i>wikindx</i>	53
Figura 37. Esfuerzo desarrollado para la segunda iteración.....	53
Figura 38. Tareas pendientes para el segundo sprint.	54
Figura 39. Diagrama de clases básico de la librería apache lucene en java	58
Figura 40. Resultado de 4 archivos indexados.....	58
Figura 41. Resultado de ejemplo del sistema de recomendación	59
Figura 42. Extracción de código para indexación.....	59
Figura 43. Consulta sparql para extraer los datos a indexar	60
Figura 44. Artículo guardado con éxito	60
Figura 45. Lista de recomendación de la IEEE	61
Figura 46. Lista de recomendaciones de la ACM.....	61
Figura 47. Lista de recomendaciones de la DBLP	62
Figura 48. Esfuerzo desarrollado para la tercera iteración	62
Figura 49. Tareas pendientes para la tercera iteración.	63
Figura 50. Modal de edición de metadatos para IEEE, DBLP y ACM.....	66
Figura 51. Edición de metadatos de un artículo científico	67
Figura 52. Mensaje de artículo guardado.....	67
Figura 53. Artículo de la DBLP.....	68
Figura 54. Página oficial del artículo en la IEEE.....	68
Figura 55. Artículo de la ACM	69
Figura 56. Página oficial del artículo en la ACM.....	69
Figura 57. Metadatos de un artículo de la IEEE	70
Figura 58. Página oficial del artículo	70
Figura 59. Esfuerzo realizado para la cuarta iteración	71
Figura 60. Tareas pendientes para la cuarta iteración.	71
Figura 61. Modelo vista controlador [28].	72
Figura 62. Arquitectura modelo vista controlador de la aplicación web	73
Figura 63. Modelo conceptual de la base de datos investigador utilizada	74
Figura 64. Modelo físico de la base de datos investigador utilizada	75
Figura 65. Modelo conceptual de la base de datos wikindx5 utilizada.....	76
Figura 66. Modelo físico de la base de datos wikindx5 utilizada	77
Figura 67. Registro de usuario.....	78
Figura 68. Inicio de sesión	78

Figura 69. Esquema de funcionamiento de módulos	79
Figura 70. Diagrama de clases para el servidor de recomendación.....	81
Figura 71. Proceso de indexación y búsqueda.....	82
Figura 72. Método GET al servidor de recomendación	84
Figura 73. Resultados de una solicitud GET a la API de recomendación.....	85
Figura 74. Arquitectura para la integración del sistema de manejo de artículos científicos de acceso abierto con el sistema de recomendación con datos enlazado	85
Figura 75. Resultados pregunta 1 de la encuesta	101
Figura 76. Resultados pregunta 2 de la encuesta	102
Figura 77. Resultados pregunta 3 de la encuesta	102
Figura 78. Resultados pregunta 4 de la encuesta	103
Figura 79. Resultados pregunta 5 de la encuesta	103

ÍNDICE DE TABLAS

Tabla 1. Interfaces sparql en la web	15
Tabla 2. Comparativa de predicados entre repositorios de la DBLP, IEEE y ACM.....	17
Tabla 3. Predicados o enlaces de dos recursos RDF para la ACM.....	18
Tabla 4. Resultado con URIs retornados.	18
Tabla 5. Consulta para extraer literales	19
Tabla 6. Resultados como literales	19
Tabla 7. Roles del equipo <i>scrum</i>	28
Tabla 8. Estimación y prioridad para listar los requerimientos.....	29
Tabla 9. Requerimientos (<i>product backlog</i>).....	30
Tabla 10. Historia de usuario 1	31
Tabla 11. Historia de usuario 2	32
Tabla 12. Historia de usuario 3	32
Tabla 13. Historia de usuario 4	33
Tabla 14. Historia de usuario 5	34
Tabla 15. Historia de usuario 6	34
Tabla 16. Historia de usuario 7	35
Tabla 17. Historia de usuario 8	36
Tabla 18. Historia de usuario 9	36
Tabla 19. Requerimientos para la primera iteración.....	37
Tabla 20. Tareas para la primera iteración.....	38
Tabla 21. Datos generales de la primera iteración	39
Tabla 22. Tareas finalizadas de la primera iteración	39
Tabla 23. Descripción de la configuración de <i>wikindx</i>	44
Tabla 24. Requerimientos para la segunda iteración	48
Tabla 25. Tareas para la segunda iteración	49
Tabla 26. Datos generales de la segunda iteración	50
Tabla 27. Tareas finalizadas de la segunda iteración.....	50
Tabla 28. Requerimiento para la tercera iteración.....	54
Tabla 29. Tareas para la tercera iteración (Continua en la siguiente página).....	55
Tabla 30. Datos generales de la tercera iteración	57
Tabla 31. Tareas finalizadas de la tercera iteración	57
Tabla 32. Requerimientos para la cuarta iteración	63
Tabla 33. Tareas para la cuarta iteración.....	64
Tabla 34. Datos generales de la cuarta iteración	65

Tabla 35. Seguimiento de tareas de la cuarta iteración.....	65
Tabla 36. Características físicas para el motor de recomendación	86
Tabla 37. Herramientas para el servidor <i>sa//s</i>	86
Tabla 38. Herramientas para el servidor de recomendación	87
Tabla 39. Casos de prueba de aceptación.....	88
Tabla 40. Caso de prueba de aceptación 2.....	88
Tabla 41. Caso de prueba de aceptación 3.....	89
Tabla 42. Caso de prueba de aceptación 4.....	90
Tabla 43. Caso de prueba de aceptación 5.....	91
Tabla 44. Caso de prueba de aceptación 6.....	92
Tabla 45. Caso de prueba de aceptación 7.....	92
Tabla 46. Caso de prueba de aceptación 8.....	93
Tabla 47. Caso de prueba de aceptación 9.....	94
Tabla 48. Caso de prueba de aceptación 10.....	95
Tabla 49. CASO DE PRUEBA DE ACEPTACIÓN 11.....	96
Tabla 50. Caso de prueba de aceptación 12.....	96
Tabla 51. Caso de prueba de aceptación 13.....	97
Tabla 52. Prueba de integridad 1	98
Tabla 53. Prueba de integridad 2.....	99
Tabla 54. Prueba de integridad 3.....	99
Tabla 55. Prueba de integridad 4.....	100
Tabla 56. Respuestas a preguntas de encuesta	104
Tabla 57. Promedios por categoría y general	104

RESUMEN

La gestión de referencias es una tarea difícil para los investigadores. Las publicaciones encontradas en la búsqueda, en su mayoría, no son tan relevantes. El número de publicaciones crece cada año y la iniciativa de acceso abierto ayuda a tener una mejor difusión a algunas de esas publicaciones [1].

Las recomendaciones suelen ser en base a las preferencias de anteriores usuarios, arrojando así resultados que podrían no ser relevantes para quien está buscando información en ese instante [2].

Pero ahora con la web semántica, junto con los datos enlazados abiertos (*Open Linked Data*) y las nuevas herramientas que se ofrecen con ella, se pueden desarrollar sistemas de recomendación más certeros identificando así cuales son los resultados que el usuario espera ver.

La cantidad de datos en la web crece continuamente y es imposible procesarla directamente por un humano [3]. En la web semántica se crean tripletas RDF (*Resource Description Framework*) para representar el conocimiento, que pueden ser procesadas por las computadoras y por aplicaciones que se pueden comunicar entre sí. Además, la web semántica utiliza algunos metadatos legibles por las máquinas para enriquecer los documentos en la *www* (*World Wide Web*), de modo que Internet se convierta en un medio de intercambio de información universal [4].

Se propone desarrollar un módulo de recomendación basado en datos enlazados abiertos. Este módulo funcionará con metadatos extraídos de un sistema de manejo de artículos científicos que se encuentra conectado al repositorio de DOAJ (*Directory of Open Access Journals*), Para luego almacenarlo en la base de datos de WIKINDEX, que es un sistema bibliográfico y de redacción de citas/notas. WIKINDEX también permite la creación de artículos en línea [5]. Estos metadatos serán extraídos al módulo de recomendación para ser procesados y entregar los artículos recomendados relacionados al conjunto de datos abiertos de la ACM (*Association for Computing Machinery*), IEEE (*Institute of Electrical and Electronics Engineers*) y DBLP (*Digital Bibliography & Library Project*).

Palabras Clave: RDF, web semántica, recomendación, ontología, datos enlazados, citas, bibliografía.

ABSTRACT

The management of references is a difficult task for researchers. The publications found in the search, for the most part, are not so relevant. The number of publications grows every year and the open access initiative helps to better disseminate some of these publications [1].

The recommendations are usually based on the preferences of previous users, thus yielding results that may not be relevant for those who are looking for information at that moment [2].

But now with the semantic web along with the open linked data and the new tools offered with it, more accurate recommendation systems can be developed, identifying the results that the user expects to see.

The amount of data on the web grows continuously and it is impossible to process it directly by a human [3]. In the semantic web, RDF (Resource Description Framework) triples are created to represent knowledge, which can be processed by computers and applications that can communicate with each other. In addition, the semantic web uses some metadata readable by machines to enrich the documents on the www (World Wide Web), so that the entire Internet becomes a universal information exchange [4].

It is proposed to develop a recommendation module based on open linked data. This module will work with metadata extracted from a management system for scientific articles that is connected to the DOAJ database (Directory of Open Access Journals), it will be stored in the WIKINDEX database, which is a software bibliographic quotation/notes and can create articles online [5]. This metadata will be extracted to the recommendation module to be processed and then will be delivered recommended articles related to the open data set of the ACM (Association for Computing Machinery), IEEE (Institute of Electrical and Electronics Engineers) and DBLP (Digital Bibliography & Library Project).

Keywords: RDF, semantic web, recommendation, ontology, linked data, cites, bibliography.

CAPITULO 1: INTRODUCCIÓN

1.1. Descripción del problema

1.1.1. Situación actual

En la actualidad la gestión de referencias de artículos científicos para investigadores es una tarea ardua porque la información que se encuentra disponible en la web es complicada de encontrar y gestionar, debido a ello y a la gran cantidad que existen, los artículos encontrados no son relevantes. La iniciativa de acceso abierto y el crecimiento de artículos cada año ayudan a tener una mejor organización y difusión de los artículos científicos [1].

Una de las iniciativas de acceso abierto es *RKB (Resilience Knowledge Base) Explorer* que usa repositorios de datos enlazados abiertos. Estos repositorios contienen artículos de la IEEE (*Institute of Electrical and Electronics Engineers*), la ACM (*Association for Computing Machinery*) y la DBLP (*Digital Bibliography & Library Project*). Todas las publicaciones que existen en esos organismos son almacenadas en formato RDF (*Resource Description Framework* o Marco de Descripción de Recursos) por la iniciativa *RKB Explorer*.

Los datos están ahí, el problema radica en que no hay ningún gestor y buscador de estos artículos que sea manejable por la comunidad en general, la razón radica en que para la extracción de este tipo de información se necesita de lenguajes de consulta sofisticados, en este caso se hace uso del lenguaje de consulta SPARQL (Protocolo y lenguaje de consulta RDF). Además, no están conectados a un sistema de recomendación, lo que tiene como consecuencia que los resultados obtenidos no sean relevantes.

1.1.2. Situación propuesta

Como solución a este problema se plantea el desarrollo de una aplicación web que permita la búsqueda de artículos RDF y el almacenamiento de los metadatos en una base de datos. De tal manera que el módulo de búsqueda esté conectado a un módulo de recomendación para que los artículos devueltos sean relevantes.

La **aplicación web** permitirá al usuario:

- La extracción de metadatos de artículos científicos de un repositorio de artículos de datos abiertos como DOAJ (*Directory of Open Access Journals*).

- El almacenamiento de los metadatos del artículo científico en la base de datos de *WIKINDX*, que es un sistema bibliográfico y de redacción de citas/notas y creación de artículos en línea.
- El uso de los metadatos en el módulo de recomendación para ser procesados y entregar los artículos recomendados relacionados al conjunto de datos abiertos de la ACM, IEEE y DBLP.

1.2. Alcance

Con la aplicación web propuesta los principales beneficiados serán investigadores, profesores y estudiantes, principalmente de la Escuela Politécnica Nacional. La aplicación web les permitirá buscar artículos científicos relacionados con su tema de interés, y gestionarlos en bases de datos locales, obteniendo resultados que les aportará grandes beneficios en su investigación.

La aplicación web usará un servicio REST en el cual está incorporado un indexador de texto, un motor buscador y un motor de recomendación. Este servicio fue desarrollado con el fin de que la aplicación web pueda acceder a sus funciones. Pero a la vez, este servicio podrá ser usado por otras aplicaciones que requieran consultar los artículos que se encuentran indexados en el servidor. La indexación de artículos solo se encargará el administrador de la aplicación.

Los repositorios digitales RDF a usarse para la extracción e indexación de datos de artículos científicos son los de la iniciativa *RKB Explorer*; los repositorios que se escogerán de dicha iniciativa serán de la ACM, IEEE y DBLP.

Los artículos científicos que almacenan dichos repositorios son relacionados a las ciencias de la computación, ingeniería de software, redes y telecomunicaciones. Por lo tanto, la investigación que se realice en la aplicación web será más provechosa en esas ramas. Además, los artículos son almacenados en el idioma inglés, por lo que será necesario conocimientos en este idioma para los interesados en usar este software.

El investigador tendrá acceso a un gestor de referencias *wikindx*, desde el cual podrá obtener la cita bibliográfica de su artículo en diferentes formatos. Además, podrá obtener y editar la información de este.

1.3. La web actual

La web actual está compuesta por documentos, estos documentos están compuesto por texto, imágenes, videos, etc. Los documentos se relacionan entre sí mediante enlaces o

mejor conocidos como hipervínculos [6], [7]. Una de las formas en la que se ve un documento web se lo muestra en la **Figura 1**.



Figura 1. Web actual

Como se puede apreciar en la **Figura 1** la página web presenta mucho texto y enlaces que nos llevan a otros documentos de la web. Los problemas que se presentan en la web actual son varios, a continuación, se los describirá a cada uno [6].

- **Heterogeneidad:** Los datos generados por cada organización web tiene diferentes formatos, por lo que consumir información de una cierta entidad a otra puede resultar un tanto complicado. Por ejemplo, la Wikipedia no puede intercambiar información con la IEEE.
- **Masividad:** Actualmente la cantidad de información que se genera día a día va creciendo exponencialmente. Debido a la gran cantidad de información generada actualmente se deben de generar técnicas y metodologías que permitan recolectar información y darle un buen uso. La información es muy variada, existe gran información que es genérica o específica que está en un formato hecho para las personas, y para ellas es muy difícil buscar e ir accediendo a toda esta información.
- **Cambios rápidos:** Cada día son borrados y almacenados una gran cantidad de datos. Con las redes sociales, sitios de compras por internet, correos electrónicos, servicios de mensajería instantánea y muchos servicios más, se transmiten grandes cantidades de información y de igual manera se desecha gran cantidad de información.

- **Interpretación de la información:** La información es hecha para que sean consumidas por humanos, es muy difícil para un computador poder interpretar la información que se encuentra en un documento de la web actual. Si un computador puede interpretar la información, sabrá qué información debe leer e interpretar y se pueden generar enlaces que ayuden a encontrar información relevante acerca de un tema.

Esos son los desafíos que se presentan en la web actual. Se busca una manera de manejar la gran cantidad de datos e información que existe y que sea interpretable por el computador para poder automatizar procesos de búsqueda, recopilación de información, recomendación, etc. Estos procesos son muy largos y requieren grandes cantidades de procesamiento. Como solución, en los últimos años se han propuesto soluciones de la web semántica.

1.4. La web semántica

Según Tim Barners-Lee: “La Web Semántica es una extensión de la web actual en la cual se da un significado bien definido a la información, permitiendo mejorar la colaboración entre personas y computadoras en la web” [7]. En otras palabras, la web semántica es el salto de una web hecha para humanos hacia una web hecha para máquinas. La web semántica brinda un mejor acceso a los datos que se encuentran en la *web*.

Para que las computadoras puedan interpretar la información que se encuentra en la *web* se debe especificar un lenguaje en común. Un lenguaje que permita especificar los diferentes componentes en la *web* y las relaciones que existen entre dichos componentes. Otro punto para tomar en cuenta para extraer datos de la *web* es tener un lenguaje de consulta entendible para un computador que permita extraer esa información y analizarla.

La *World Wide Web Consortium*, que es el organismo regulador de la web, ha elaborado una serie de recomendaciones para que los computadores sean capaces de entender la información en la web. Una de estas recomendaciones establece que debe haber un lenguaje estándar para todos los computadores.

La W3C además ha elaborado una pirámide de estándares para poder llevar a cabo el proyecto de la web 3.0 o web semántica [6], [7]. Entre los principales estándares se encuentran: El *RDF*, *RDF Schema* y la *Ontology OWL*.

En la **Figura 2** se puede visualizar todos los componentes que corresponden a los estándares de la web semántica [4].

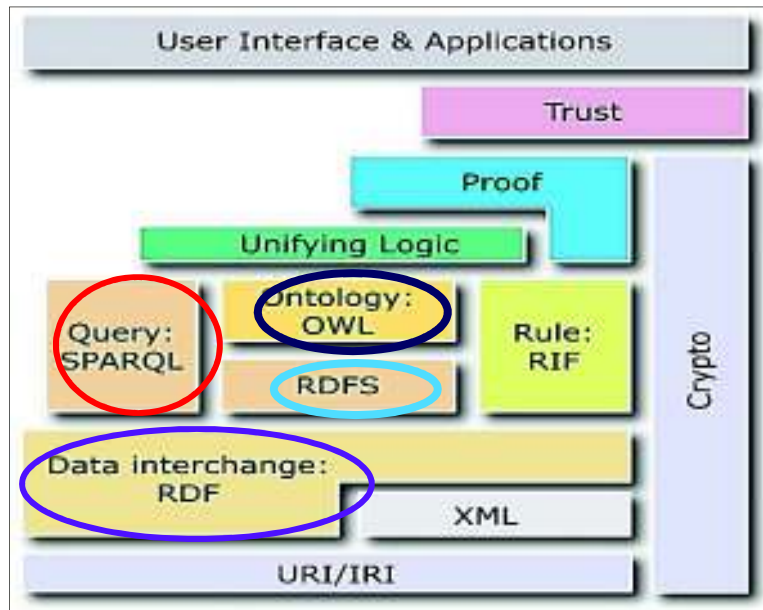


Figura 2. Componentes de la web semántica.

En los niveles inferiores se encuentran los componentes más básicos de la web 3.0 y en el superior los componentes de confianza. En este proyecto se hará énfasis en los aspectos básicos de la web semántica, es decir en los niveles inferiores. A continuación, se expandirá la información de estos componentes básicos [7].

1.4.1. El lenguaje RDF

El lenguaje RDF es una propuesta de la W3C para la representación de la información de recursos que se encuentran en la web. Como características básicas se tiene:

- RDF está basado en el uso de grafos dirigidos y etiquetados.
- El lenguaje RDF es entendido y procesado por un computador.
- Los componentes básicos para la construcción de un RDF son los URI y los literales.

El lenguaje RDF, del inglés *Resource Description Framework*, se utiliza para representar entre otras cosas: información personal, redes sociales y/o metadatos. Así también como un medio de unificación de fuentes de información dispares [8].

RDF se puede representar de manera gráfica como un grafo con nodos y arcos etiquetados, tal como se muestra en la **Figura 3**.

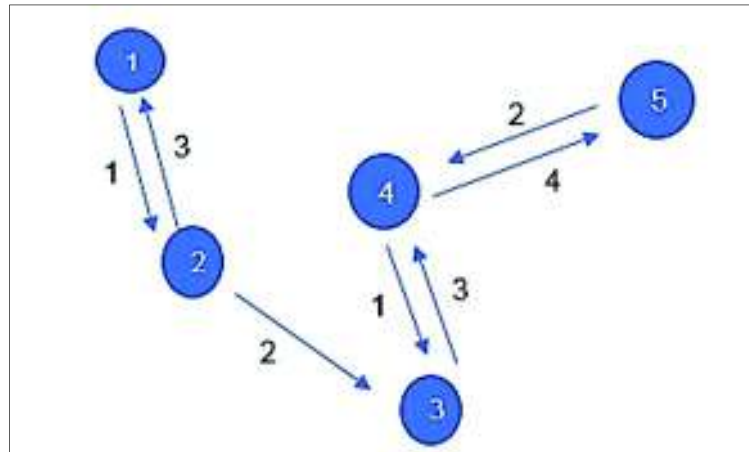


Figura 3. Representación de un grafo RDF

Cada nodo representa al componente básico de un RDF tal como es el URI. Cada arco representa un predicado entre el URI y el valor. También se puede describir al grafo RDF como un conjunto de triples RDF.

En la **Figura 4** se muestra como está formado un triple RDF:

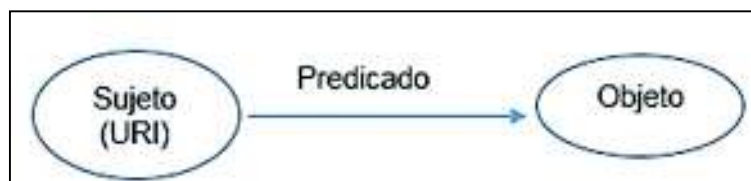


Figura 4. Triple RDF

- **Sujeto (URI):** Es un Identificador de recursos uniforme. Un recurso podría ser: un artículo científico, un libro de literatura, el autor de una novela, etc. Cada recurso tendrá su propio URI. En el triple RDF también se lo conoce como sujeto.

La **Figura 5** muestra la representación de un URI



Figura 5. Representación de una URI

Las partes en la que se compone un URI son las siguientes:

- **http:** Protocolo de comunicación el cual permite la transferencia de la información en la World Wide Web. Se lo conoce como el camino del URI [7].
- **ieee.RKB Explorer.com:** Es similar a un dominio web y es en donde se encuentra el recurso. Se lo conoce como la autoridad del URI [7].

- ***/id/publication-04223238:*** Es el identificador del recurso al cual se desea acceder. En este ejemplo se trata de un artículo científico. Se lo conoce como el camino del URI. Cabe recalcar la diferencia que existe entre un URI y una URL. La URI por un lado es identificador de un recurso en la web, mientras tanto la URL (Localizador de Recursos Uniforme) es un URI pero que identifica a una página web.
- ***El predicado:*** El predicado es el nexo que existe entre el sujeto y el objeto, es el que hace que exista una relación entre dos recursos en la web. Por ejemplo, se quiere saber cuál es la relación existente entre Suiza y Albert Einstein, como premisa sabe que nació en suiza, pero para hacer interpretable la relación para un computador debe de haber un nexo entre los dos recursos. Este nexo es el predicado ***dbo:citizenship***. Con ese predicado se puede saber la ciudadanía que tuvo Albert Einstein a través de una consulta que sea entendida por el navegador a través de la web [7].
- ***El objeto:*** Es el valor que se le da al sujeto, o el valor que se le da al atributo. Por lo general los objetos suelen ser literales. Un literal es una cadena de caracteres encerradas por comillas, para que un literal pueda ser entendido e interpretado por el computador debe tener asociado un tipo [7]. Por ejemplo: ***akt:has-title***. “*Software Prefetching*”, ello indica que lo que está encerrado entre comillas es un literal de tipo título. Así mismo un objeto puede ser un URI, y a su vez referencia a otros URIs.

1.4.2. Vocabularios RDF y prefijos:

Los vocabularios RDF especifican un lenguaje estándar para el entendimiento y comunicación con las bases de datos RDF. Actualmente en la web existen varios tipos de vocabularios RDF. Algunos vocabularios describen: las relaciones entre personas de redes sociales, los elementos de un artículo científico, las coordenadas geográficas, etc. [7].

Antes de poder realizar consultas SPARQL es necesario definir los vocabularios que se van a usar. Para la definición de los vocabularios se hace la declaración de los prefijos. Un prefijo es la declaración de un vocabulario RDF. Para declarar un prefijo se antepone la palabra *PREFIX*, luego se escribe el nombre del vocabulario RDF seguido de la URI del recurso de dicho vocabulario, por ejemplo:

PREFIX + nombre de vocabulario RDF + <URI del recurso>

La **Figura 6** muestra un ejemplo de declaración de prefijos de DBpedia.

```
@prefix rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
```

Figura 6. Declaración de prefijo para la DBpedia.

1.5. El lenguaje de consulta *sparql*

El término *sparql* se refiere a la vez a un lenguaje de consulta y a un protocolo. En su mayor parte el término *sparql* es usado para hacer referencia al lenguaje de consulta RDF. Haciendo uso de este término, el lenguaje de consulta es similar a SQL de manera sintáctica. Este lenguaje realiza consultas a grafos RDF mediante la coincidencia de patrones [9]. Además, define el formato de un documento XML para la representación de resultados de una consulta *sparql* [10].

Las principales características de este lenguaje son:

- Patrones conjuntivos básicos.
- Filtros de valores.
- Patrones opcionales.
- Patrones de Disyunción.

En cuanto al protocolo *sparql*, es un método para invocación de consultas *sparql* remotas y de sus resultados. Su función principal es la de proponer una interfaz simple que pueda soportar a los protocolos SOAP o HTTP con el fin de que un cliente pueda emitir consultas SPARQL hacia una interface de consulta [9].

Como se ha mencionado en la sección anterior, Antes de realizar una consulta *sparql* es necesario declarar los diferentes vocabularios RDF que serán necesarios para el entendimiento de las diferentes relaciones existente entre los datos que se requiere consultar. En la **Figura 7** se muestra un ejemplo de declaración de prefijos para la iniciativa *RKB Explorer*

```
PREFIX id: <http://ieee.rkbexplorer.com/id/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX akt: <http://www.aktors.org/ontology/portal#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX akt: <http://www.aktors.org/ontology/portal#>
PREFIX akts: <http://www.aktors.org/ontology/support#>
PREFIX iai: <http://www.iai.uni-sb.de/resist#>
```

Figura 7. Declaración de prefijos en la iniciativa *RKB Explorer*

Una vez declarados los prefijos necesarios se procede a realizar la consulta SPARQL. Los ejemplos que se describen a continuación son desarrollados en la interfaz de consulta de la iniciativa *RKB Explore*. La interface de consulta es <http://acm.RKBExplorer.com/sparql/>. Esta interfaz nos permite realizar consultas SPARQL al repositorio RDF de la ACM.

Un ejemplo de una consulta *sparql*/se la expresa en la **Figura 8**:

```

1      2      3
SELECT ?titulo WHERE {
      4
<http://acm.rkbexplorer.com/id/100233> akt:has-title ?titulo
      5
}
      6

```

Figura 8. Ejemplo de consulta SPARQL

En la **Figura 8** muestra números que representan lo siguiente:

1. La cláusula SELECT que identifica a las variables que van a aparecer en el resultado de una consulta.
2. Es el nombre de la variable que se precede con un signo de interrogación, puede haber más de una variable.
3. La cláusula WHERE que provee un patrón de grafo básico que coincide con el grafo de datos RDF.
4. Es el sujeto del triple RDF que generalmente siempre vendrá a ser un URI.
5. Es el predicado del triple RDF que fue declarado al inicio de la consulta e identifica al vocabulario RDF a usar y con el cuál se hace comprender la relación que se está buscando.
6. Es la variable declarada al inicio de la estructura en el paso 2 y pasa a ser el objeto o el valor del triple RDF. Aparecerá en el resultado final de la consulta.

El resultado del ejemplo de consulta se lo expresa en la **Figura 9**:

Query: [show]		
Results:		
Result	Binding	Value
1	?titulo	Analyzing and visualizing performance of memory hierarchies

1 result(s) found in 0.0 seconds.

Figura 9. Resultado de consulta SPARQL

El resultado presentado en el gráfico tiene el formato de una tabla HTML y la contiene tres atributos:

- **Result.** Presenta el número del resultado.
- **Binding.** Es la columna donde se indica la variable buscada.
- **Value.** Es la columna donde se presenta al resultado obtenido para la variable en la columna *binding*. Cabe recalcar que el resultado obtenido puede ser otra URI a la cual se le puede se le puede realizar otra consulta.

1.5.1. Operadores SPARQL

Los operadores del SPARQL permiten realizar consultas más complejas y de mayor precisión en cuanto a los resultados devueltos.

AND

El operador *AND* permite realizar una conjunción de patrones y se lo representa con un punto (.). Es insertado dentro de la cláusula *where* y después del triple, con ello se permitirá poner los triples que sean necesarios dentro de una consulta [11].

DISTINCT

Hay veces en que una consulta puede devolver varios resultados repetidos, para evitar eso se hace uso del operador *DISTINCT*. Este operador debe ser escrito después de la sentencia *SELECT*[11].

FILTER

La operación *filter* permite obtener una consulta dando como parámetro el valor de una variable que se desea que obtenga la consulta. Dentro de la función se puede hacer uso de otros operadores como es el caso del operador and (&&) y el operador or (||) [10] ,[11].

En la **Figura 10** se muestra un ejemplo donde se desean encontrar artículos de dos autores diferentes.

```
SELECT DISTINCT *
WHERE { ?articulo akt:has-title ?nombre_articulo.
        ?articulo akt:has-author ?autor.
        ?autor    akt:full-name ?nombre_autor.
        filter(?nombre_autor="Fancong Zeng" || ?nombre_autor="Pankaj Jalote")
}
```

Figura 10. Operador filter

OPTIONAL

Existen casos en los que cuando se realiza una consulta *sparql* se retorna un resultado vacío y eso es porque la variable declarada en la consulta no existe en el grafo RDF hacia donde se realiza la consulta [9], [11]. Para el control de ese tipo de inconsistencias es necesario el operador *OPTIONAL*, el cual permite que el resultado de una variable sea opcional tal como se ve en la **Figura 11**.

```
SELECT DISTINCT ?titulo ?abstract WHERE {  
    <http://ieeex.rkbexplorer.com/id/publication-00532622> akt:has-title ?titulo.  
    OPTIONAL{<http://ieeex.rkbexplorer.com/id/publication-00532622> extn:has-abstract ?abstract}  
}
```

Figura 11. Consulta con operador OPTIONAL

Algunos artículos almacenados en los repositorios de la *RKB Explorer* no contiene el campo de *abstract* por lo que es necesario el operador *OPTIONAL*. Lo mismo aplica a otro tipo de repositorios con diferentes campos.

1.5.2. Funciones

IsIRI(¿VAR)

Esta función retorna verdadero si el valor retornado es una URI. Esta función va como parámetro dentro del operador *FILTER* [9], [11]. El valor retornado también puede ser su negación anteponiendo el signo de interrogación antes de la función

En la Figura 12 se muestra cómo se retorna todos los objetos o los resultados que sean una URI dentro de un grafo RDF de un artículo científico.

```
SELECT * WHERE {  
    <http://ieeex.rkbexplorer.com/id/publication-04223238> ?predicado ?objeto.  
    FILTER(IsIRI(?objeto))  
}
```

Figura 12. Consulta SPARQL con operador IsIRI

BOUND(¿VAR)

Retorna verdadero si una variable contiene un valor. A diferencia del operador *OPTIONAL*, esta función solo retornará los resultados que contengan el valor especificado. Al igual que el operador *IsIRI()*, este también puede ser negado anteponiendo el signo de admiración (!): *FILTER(!Bound(?abstract))*[9], [11]. Un ejemplo de esta función se muestra en la Figura 13.

```

SELECT ?articulo ?abstracto WHERE {
  ?articulo rdf:type akt:Proceedings-Paper-Reference.
  ?articulo extn:has-abstract ?abstracto.
  FILTER(BOUND(?abstracto))
}

```

Figura 13. Consulta SPARQL con la función BOUND

IsLITERAL(?VAR)

A diferencia de la función IsIRI(), esta función retorna los valores que sean considerados literales. Los literales son los valores u objetos que son una cadena de caracteres. Con la negación de IsIRI a parte de literales se obtienen ontologías, por lo que esta función es de mucha utilidad si se requiere extraer información [11].

Un ejemplo de esta función se muestra en la **Figura 14**.

```

SELECT * WHERE {
  <http://ieee.rkbexplorer.com/id/publication-04223217> ?predicado ?objeto.
  FILTER(IsLiteral(?objeto))
}

```

Figura 14. Consulta SPARQL con función IsLiteral

1.6. Interfaz de consulta a repositorios digitales RDF

Es una interfaz en la que los usuarios (humanos o aplicaciones) pueden acceder para realizar consultas *sparql* y retornar los resultados de acuerdo con la consulta enviada. Para las aplicaciones estas interfaces pueden ser usados como APIs de las cuales se consume el servicio mediante una petición y se retorna los resultados en diferentes formatos. Para los humanos es una interfaz gráfica en la cual se pueden realizar las consultas *sparql* y obtener los resultados de dicha consulta [12].

Una interfaz de *sparql* puede retornar diferentes formatos. Los formatos varían dependiendo de quién los consuma: un ser humano o una máquina. Para los humanos generalmente los resultados se suelen presentar en tablas HTML (*Hyper Text Markup Language*). En cambio, cuando una aplicación quiere consumir dichos resultados, es necesario transmitir esa información en un lenguaje legible para la aplicación, entre dichos formatos se encuentran: RDF/XML o JSON [12].

Un SPARQL puede ser categorizado como genérico o específico. Una interfaz genérica trabaja sobre cualquier repositorio de datos, tal es el caso de la interfaz de *OpenLink Virtuoso* que trabaja sobre el repositorio de datos de la DBpedia, de la *BBC Programmes and Music*, *Bio2RDF*, etc.

Las interfaces específicas trabajan tan solo para un repositorio de datos específico como es el caso de la interfaz de *RKB Explorer* que trabaja sobre sus mismos repositorios [12]. Actualmente se puede observar las interfaces de consulta para SPARQL que están disponibles desde la web de la w3c.

La **Tabla 1** muestra una parte de la información de las 3 primeras interfaces de consulta *sparql*. La lista completa se puede encontrar en el siguiente enlace: <https://www.w3.org/wiki/SparqlEndpoints>.

Project	status	SPARQL endpoint	Webform
Wikidata	(2017-02-23) alive	endpoint	GUI
BBC Programmes and Music	(2010-06-29) alive	endpoint	Ajax based Visual Query Builder
Bio2RDF	(2010-01-07) alive	List of 40 SPARQL endpoints	n/a

Tabla 1. Interfaces sparql en la web

1.6.1. Repositorio digital *RKB (Resilience Knowledge Base)*

Para la realización de este proyecto se ha tomado en cuenta que los datos que se requieren extraer de las consultas SPARQL sean de tipo bibliográfico y que esos datos sean de un artículo científico. Para cumplir tal requerimiento se ha elegido a los repositorios de datos de la iniciativa *RKB Explorer*.

RKB Explorer es una aplicación web semántica para usuarios que requieran consultar a la base de *ReSIST Knowledge Base* (RKB). Esta base fue construida como parte del proyecto ReSIST (*Resilience for Survivability in IST*), que ha ido recopilando información de muchas fuentes bibliográficas, dándoles una cierta estructura que va a permitir consultar la información por tema. Dando una especial atención a los temas relacionados a Informática fiable [13]. Entre los diferentes temas que divide a la información que ofrece la RKB se encuentran [13].

- Personas del proyecto RKB.
- Proyectos que ha descubierto el proyecto RKB que suele estar actualizado en todos los proyectos financiados por la Unión Europea (UE) a través de la base de datos

CORDIS (*Community Research and Development Information Service*), y muchos de los financiados por la NSF (*National Science Foundation*) de los EE. UU.

- Mecanismos resilientes.
- Metadatos sobre los mecanismos que han sido estudiados en el proyecto RESIST.
- Publicaciones científicas en el campo de las ciencias de la computación, incluidos los de la DBLP, IEEE y ACM.
- Organizaciones académicas, gubernamentales e industriales.
- Áreas de Investigación en las ciencias de la computación.

De todos los temas listados anteriormente que contiene el proyecto RKB, se hará uso de las publicaciones tanto de la DBLP, IEEE y ACM que están almacenadas en la base de datos del RKB.

Para ingresar a los datos de cada repositorio digital del proyecto RKB, mediante consultas SPARQL, se lo hará mediante sus respectivas interfaces. Su interfaz es similar en cuanto a presentación gráfica, pero la dirección de ingreso y los tipos de datos que contienen son diferentes. Para el ingreso se los hace mediante la siguiente dirección web ***http://[nombre_dataset].RKB Explorer.com/sparql***.

En **nombre_dataset** se ingresa el nombre del repositorio: DBLP, IEEE o ACM.

La **Figura 15** muestra la interfaz de consulta de *RKB Explorer*.

SPARQL Query Interface

This interface permits queries to be made over the information held within the repository, using the SPARQL Query Language.

Result format:

Query:

```
PREFIX id: <http://ieee.rkbexplorer.com/id/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX akt: <http://www.aktors.org/ontology/portal#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX akt: <http://www.aktors.org/ontology/portal#>
PREFIX akts: <http://www.aktors.org/ontology/support#>
PREFIX iai: <http://www.iai.uni-sb.de/resist#>

SELECT * WHERE [ ?s rdfs:label ?o ] LIMIT 10
```

Enviar consulta

Figura 15. Ejemplo de la interfaz de consulta de RKB Explorer

1.6.2. Comparativa IEEE, ACM y DBLP

Los repositorios de datos de la IEEE, ACM y DBLP conectados a la base RKB tienen algunas diferencias. Los datos que se almacenan y las ontologías que se usan son diferentes. Por lo tanto, una consulta realizada en el repositorio de datos de la ACM no podría utilizarse en el repositorio de datos de la IEEE o la DBLP, por lo que es necesario conocer los diferentes campos y datos que se encuentran en cada una.

Cada URI de la publicación está conectado a esos valores mediante triples de RDF. Estos triples se descomponen en: el sujeto, el predicado (que es un enlace), y el objeto. El sujeto y el objeto se encuentran conectados mediante el predicado. A continuación, la **Tabla 2** muestra los predicados que se usan en los tres repositorios [14].

DBLP	IEEE	ACM	Significado
akt:sub-area-of	akt:sub-area-of	akt:sub-area-of	Sub área de
akt:article-of-journal	iai:is-related-to	--	Artículos relacionados
akt:has-author	akt:has-author	akt:has-author	Autor
akt:full-name	akt:full-name	akt:full-name	Nombre del autor
akt:has-title	akt:has-title	akt:has-title	Título del artículo
owl:sameAs	akt:paper-in-proceedings	--	Congresos o conferencias donde se han publicado
akt:has-date	akt:has-date	akt:has-date	Fecha
akt:has-web-address	akt:has-web-address	--	Dirección web del artículo
akt:has-volume	extn:has-abstract	--	Abstract del artículo
akts:years-of	akts:years-of	akts:years-of	Año de publicación
akts:month-of	--	akts:month-of	Mes de publicación
--	iai:has-ieee-keyword	akt:address-generic-area-of-interest	Palabras Clave
akt:has-affiliation	--	--	Afiliación
akt:cites-publication-reference	iai:is-very-strongly-related-to	akt:cites-publication-reference	Citas
akt:edited-by	iai:is-strongly-related-to	akt:has-publication-reference	Artículos fuertemente relacionados

Tabla 2. Comparativa de predicados entre repositorios de la DBLP, IEEE y ACM.

1.6.3. Extracción de información de un artículo científico

Para extraer los datos de un artículo científico mediante una consulta SPARQL se debe usar: la URI, su predicado y la variable del atributo que se desea conocer. La **Tabla 3** muestra los principales campos de un artículo científico en relación con la ACM. Los mismos campos se encuentran en la DBLP e IEEE.

Título del artículo	akt:has-title
Autor	akt:has-author
Fecha	akt:has-date
Citas	akt:cites-publication-reference

Tabla 3. Predicados o enlaces de dos recursos RDF para la ACM

La **Figura 16** muestra un ejemplo de consulta *sparql* que se usa en la interfaz de *RKB Explorer* para la URI <http://acm.RKB Explorer.com/id/1006979>.

```
PREFIX id: <http://acm.rkbexplorer.com/id/>
PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
PREFIX akt: <http://www.aktors.org/ontology/portal#>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
PREFIX akt: <http://www.aktors.org/ontology/portal#>
PREFIX akts: <http://www.aktors.org/ontology/support#>
SELECT ?titulo ?autor ?fecha ?citas WHERE {
  <http://acm.rkbexplorer.com/id/106979> akt:has-title ?titulo.
  <http://acm.rkbexplorer.com/id/106979> akt:has-author ?autor.
  <http://acm.rkbexplorer.com/id/106979> akt:has-date ?fecha.
  <http://acm.rkbexplorer.com/id/106979> akt:cites-publication-reference ?citas
}
```

Figura 16. Consulta SPARQL para extraer información de un artículo científico en la interfaz de *RKB Explorer*

En la **Tabla 4** se puede observar los resultados de la consulta donde se muestran: el título, el autor y la fecha del recurso consultado. Las citas son otro recurso RDF y para obtener un valor textual se deberá cambiar la sintaxis de la consulta como se muestra en la **Tabla 5**.

Result	Binding	Value
1	?titulo	Software prefetching
	?autor	http://acm.rkbexplorer.com/id/person-60855-09416367142050e9164517da8205073f
	?fecha	http://www.aktors.org/ontology/date#1991-04-01 [local]

Tabla 4. Resultado con URIs retornados.


```

SELECT DISTINCT ?titulo ?autor ?fecha ?cita WHERE {
  <http://acm.rkbexplorer.com/id/106979> akt:has-title ?titulo.
  <http://acm.rkbexplorer.com/id/106979> akt:has-author ?autorRDF.
  ?autorRDF akt:full-name ?autor
  <http://acm.rkbexplorer.com/id/106979> akt:has-date ?fechaRDF.
  ?fechaRDF akts:has-pretty-name ?fecha
  <http://acm.rkbexplorer.com/id/106979> akt:cites-publication-reference ?citaRDF
  ?citaRDF akt:has-title ?cita
}

```

Tabla 5. Consulta para extraer literales

En la **Tabla 6** se muestra los resultados legibles al ser humano.

La interfaz de *RKB Explorer* tiene la opción de cambiar el formato del resultado y en vista

Result	Binding	Value
1	?titulo	Software prefetching
	?autor	Allan Porterfield
	?fecha	1991-01-01
	?cita	Lockup-free instruction fetch/prefetch cache organization

Tabla 6. Resultados como literales

de que el sistema de recomendación va a ser un aplicativo web la mejor opción para obtener los resultados es el formato JSON.

La **Figura 17** muestra los resultados de la consulta en formato JSON.

```

{
  "head": {
    "vars": [ "titulo", "autor", "fecha", "cita" ]
  },
  "results": {
    "distinct": true, "ordered": false,
    "bindings": [
      {
        "titulo": { "value": "Software prefetching", "type": "literal" },
        "autor": { "value": "David Callahan", "type": "literal" },
        "fecha": { "value": "1991-04-01", "type": "literal" },
        "cita": { "value": "Register allocation & spilling via graph coloring", "type": "literal" }
      }
    ]
  }
}

```

Figura 17. Resultados de consulta SPARQL de la **Tabla 5** en formato JSON

1.7. Sistemas de recomendación

Definición de sistemas de recomendación.

Los sistemas de recomendación son herramientas de toma de decisiones que proporcionan a un usuario información suficiente en relación con lo que buscan [15]. Los sistemas de recomendación son también técnicas de software que ofrecen sugerencias para que los elementos sean de utilidad para un usuario [16].

Tipos de sistemas de sistemas de recomendación

Existen varios tipos de sistemas de recomendación. En forma general [16] propone 6 tipos de sistemas de recomendación

- **Filtrado colaborativo:** Este sistema se basa en la recopilación de experiencias de navegación y evaluaciones de usuarios sin ningún conocimiento sobre los elementos que se sugieren.
- **Basada en contenido:** Este sistema se basa en descripciones bien definidas de los elementos. La similitud de los elementos se calcula en función de las características asociadas con los elementos comparados
- **Basada en el conocimiento:** Este sistema intenta hacer coincidir las preferencias del usuario con las propiedades del elemento (características).
- **Demográfico:** Este tipo de sistema recomienda elementos según el perfil demográfico del usuario.
- **Basado en la comunidad:** Este tipo de sistema recomienda elementos según las preferencias de los amigos de los usuarios.
- **Enfoques híbridos:** Este sistema intenta integrar características de varios métodos, se basan en la combinación de las técnicas mencionadas anteriormente.

El sistema de recomendación desarrollado en el presente proyecto utiliza la recomendación basada en contenido ya que la recomendación se realiza en base a los metadatos de un artículo científico, en específico el título que es una descripción bien definida de un elemento, y el *abstract* en el cual se resume de lo que se trata el artículo científico.

Modelo espacio vectorial (*Vector Space Model*)

El modelo utilizado en el sistema de recomendación es el VSM (*Vector Space model*) que es un modelo algebraico para filtrado, recuperación, indexado y cálculo de relevancia de información. VSM es una representación espacial de documentos de texto [17].

Este modelo parte de la premisa de que se pueden representar los documentos como vectores de términos, y de esa manera los documentos pueden ubicarse en un espacio vectorial de n dimensiones [17]. Cada término es definido como una dimensión y cada documento es expresado como un vector [18]. De esta manera cada documento ocupa un lugar en el espacio vectorial, y si varios documentos tienen similares dimensiones quedan más próximos entre sí, agrupándose por su grado de semejanza y creando así grupos de documentos similares o dicho de otra manera se crean documentos relevantes para la mismas clase de información [17].

Para representar a los documentos, se utiliza un vector de pesos no binarios, donde cada peso tiene un término de índice $d_i = (t_i1, t_i2, t_i3, \dots, t_in)$ [17].

Para el cálculo de la similaridad entre un texto de consulta, que también pasa a ser un documento, y los documentos en el espacio vectorial (o entre los mismos documentos del espacio vectorial) se usa la medida del ángulo entre los vectores de los documentos. Si el ángulo es pequeño, entonces los documentos son similares, y si el ángulo es grande entonces los documentos son de temas distintos [18].

Herramientas para implementación de sistema de recomendación

Para la implementación de este modelo se hace uso de la librería *apache lucene* desarrollado en Java. La librería permite programar operaciones tales como: indexación, búsqueda y clasificación.

La indexación que realiza el modelo con la librería *apache lucene* se muestra en la **Figura 18**:

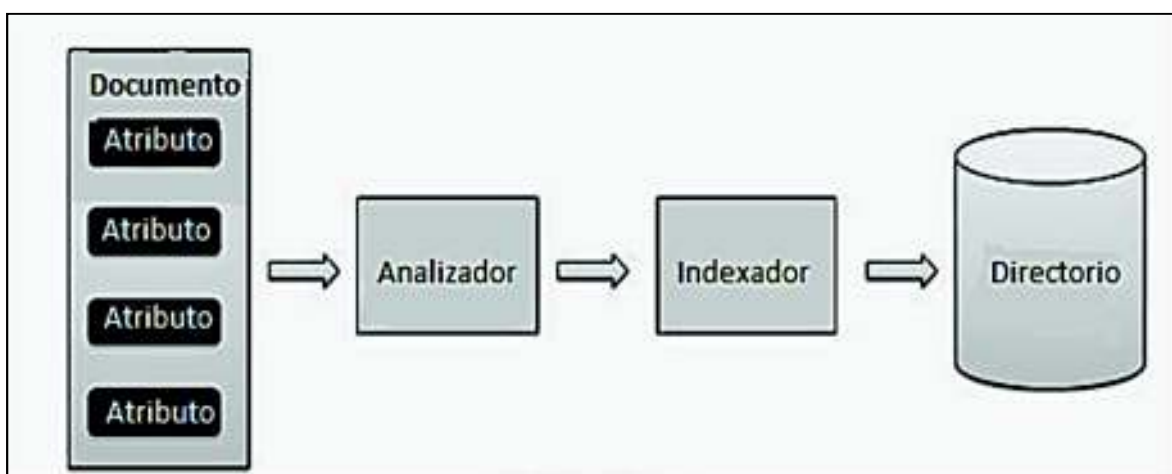


Figura 18. Esquema del procedimiento de recomendación en *lucene* [19]

Los pasos que se realizó para la implementación del motor de recomendaciones son los siguientes:

1. El documento es un archivo plano el cual contiene los campos que representan el atributo que se desea indexar.
2. El **analizador** es un módulo que se encarga de realizar una inspección a los diferentes campos para verificar que cumplan con los requisitos para ser indexados.
3. El **indexador** indexa cada atributo del documento que haya pasado el análisis, así la información puede ser recuperada en base a claves.
4. Una vez indexados los documentos se procede a crear una interface para que el usuario pueda realizar una búsqueda.
5. Cuando el usuario ingresa un **texto de búsqueda** el programa creará un **objeto de consulta** con un **analizador de consultas**. El objeto es usado para consultar en la base de datos de índices.
6. Con el **objeto de consulta** construido se procede a **buscar** en la base de datos de índices para obtener los resultados más relevantes.
7. Se presentan los **documentos relevantes** que contienen términos similares a los ingresados inicialmente por el usuario.

La **Figura 19** muestra de forma más detallada el proceso de búsqueda que realiza el motor de recomendaciones.

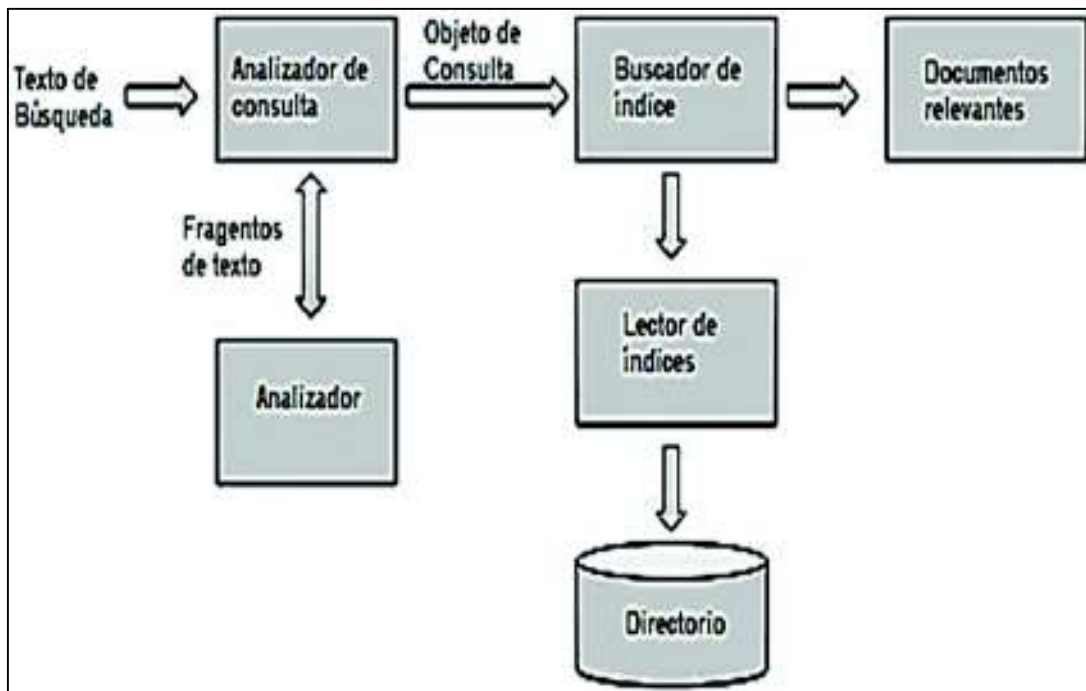


Figura 19. Proceso de búsqueda en *apache lucene* [19]

1.8. Selección de lenguajes y herramientas

1.8.1. Tecnología FRONT-END

Para el desarrollo de la parte visual de la aplicación se hace uso del motor de vista EJS (*Embedded JavaScript templating*) y del marco estructural para aplicaciones web dinámicas Angularjs.

EJS

EJS es un lenguaje simple de creación de plantillas que permite generar un marcado HTML con JavaScript [20]. EJS es el motor de vista predeterminado en Sails.js por lo que se utiliza para crear en su mayoría el diseño de las diferentes páginas de la aplicación.

ANGULARJS

Es una plataforma de desarrollo que facilita la creación de componentes visuales de aplicaciones con la web. Angular combina plantillas declarativas, inyección de dependencias, herramientas de extremo a extremo y mejores prácticas integradas para resolver los desafíos de desarrollo [21]. Angularjs se utiliza específicamente en la creación del diseño de la página para el módulo de recomendación.

1.8.2. Tecnología BACK-END

Para el desarrollo del servicio de obtención de datos y conexión a bases de datos se hace uso del marco de trabajo Sails.js en la versión 0.12.14 y para el módulo de recomendación se hace uso del Java EE.

SAILSJS

Es un marco de trabajo para Node.js que utiliza el patrón Modelo Vista Controlador. Utiliza el lenguaje de programación JavaScript y permite conectar a cualquier base de datos gracias a una capa de ORM (mapeo Objeto-relacional), que proporciona una capa de acceso a datos sencilla que funciona, independientemente de la base de datos que esté utilizando [22].

Permite realizar relaciones (uno a muchos, muchos a muchos) entre diferentes modelos y esos modelos pueden pertenecer a diferentes bases de datos, no importa si estas bases de datos son no relacionales [22].

Con Sails.js se puede tener un API que permita realizar las operaciones CRUD (*Create, Read, Update and Delete*) sin tener que escribir una línea de código para ello [22].

Java EE

El JEE (*Java Enterprise Edition*) permite la creación de proyectos web dinámicos, y dentro de estos la programación de web services.

Los tipos de *webservices* que se pueden desarrollar son el SOAP (*Simple Object Access Protocol*) y el REST (*Representational State Transfer*) del cuál hará uso esta aplicación. En esta herramienta, a diferencia del sails.js, si hay que programar las operaciones CRUD.

1.8.3. Entornos de desarrollo integrado (IDE)

WebStorm

Es un IDE ligero pero potente, perfectamente equipado para el desarrollo complejo del lado del cliente y desarrollo con Node.js del lado del servidor [23].

Este IDE permite trabajar con facilidad con los diferentes marcos de desarrollo como angular, node.js, vue.js. También con marcos de desarrollo móviles como Ionic, Cordoba, React Native [23].

Tiene funcionalidades de ayuda tanto para el lado de cliente como para el lado del servidor con node.js. Además, tiene integrado un terminal de comandos, este es necesario para crear, compilar y ejecutar los diferentes proyectos [23].

Visual Studio Code. Es una herramienta de desarrollo hecha por Microsoft, que es gratuita y de código abierto. Esta herramienta es ideal para la programación de aplicaciones web, ejecución de scripts de Nodejs y depuración de código. Además, viene integrado un controlador de versiones GIT.

Eclipse EE. Este editor de desarrollo cuenta con la creación de proyectos web dinámicos, generando automáticamente todos los archivos y carpetas que van a ser usadas para la creación de un proyecto web. Tiene la posibilidad de ser usado con Maven, que es un repositorio de librerías y dependencias, y se encarga de descargarlas automáticamente cuando el proyecto web lo necesite.

Con el uso de diferentes librerías se pueden crear servicios API REST para cualquier tipo de proyecto web, y tiene la posibilidad de trabajar sobre cualquier servidor tipo Apache, Wildfly, etc.

1.9. Metodología utilizada (SCRUM)

Al ser una aplicación en la que hay varios involucrados, entre desarrolladores e interesados en el software; además, al existir varios cambios constantes en el transcurso del desarrollo,

y en la que hay que tener constante comunicación con los interesados del producto software, la metodología ideal para el desarrollo del aplicativo web es **Scrum**. Entre las razones por las que esta metodología es la idónea, se encuentran:

- Es una metodología adaptable a los cambios.
- En el proceso de desarrollo de la metodología, las pruebas funcionales son frecuentes
- Se tiene un contacto frecuente con el cliente.
- Se entregan productos de forma iterativa e incremental aumentando así la oportunidad de retroalimentación.
- Se realiza una inspección frecuente de los artefactos *Scrum* para analizar si están cumpliendo los objetivos, y en caso de que no lo haga, los procesos deberán de ajustarse.

1.9.1. SCRUM

Scrum es un marco de trabajo el cual se basa en procesos empíricos a través de la transparencia en los procesos, una inspección continua y una adaptación para que el producto no se desvíe de los límites aceptables. *Scrum* minimiza la complejidad en el desarrollo de productos lo que satisface las necesidades de los clientes. Scrum es simple, fácil de entender, pero difícil de dominar [24].

1.9.2. Componentes de *Scrum*

El Equipo Scrum

Consiste en un **dueño de producto**, el **equipo de desarrollo** y un **facilitador**. Los equipos *Scrum* son multifuncionales y auto organizados. Este equipo *Scrum* entrega un producto incremental por medio de interacciones con el objetivo de obtener retroalimentaciones más eficientes. Está diseñado para mejorar la productividad, creatividad y la flexibilidad. Cada entrega asegura una versión funcional del producto [24].

El Dueño de Producto

El dueño del producto es el responsable de gestionar la **lista de productos** y de maximizar el valor del producto. El **dueño del producto** no es un comité sino una única persona, es quien se encarga de la priorización en la lista de trabajo y toda la organización debe respetar su decisión, sobre todo el **equipo de trabajo** que no puede actuar en base a lo que diga cualquier persona [24].

El Equipo de Desarrollo

Los miembros del **equipo de desarrollo** son multifuncionales y auto organizados, cada miembro individual puede tener habilidades específicas en un área, pero la responsabilidad de entregar un **incremento** (*sprint*) “terminado” recae en todos. El equipo consiste en profesionales los cuales, sin importar el título que tengan, todos se identifican como desarrolladores. El tamaño del **equipo de desarrollo** no debe ser menor a 3, ni superar los 9 miembros, con el propósito permanecer ágil y así poder completar una cantidad de trabajo significativa [24].

Facilitador

El **facilitador** es el encargado de que el *Scrum* se entienda y adapte. El **facilitador** es un líder quien ayuda a las personas externas a entender las interrelaciones útiles y no útiles que existen en el equipo *Scrum* para maximizar el valor creado por el **equipo Scrum**. Además, el facilitador se asegura de que se trabaje en base a teorías, reglas y prácticas del *Scrum* [24].

1.9.3. Eventos de *scrum*

Scrum tiene eventos predefinidos para crear regularidad y minimizar la necesidad de reuniones no definidas en *Scrum*. Todos los eventos tienen una duración máxima, es decir que tienen una duración fija. A los eventos se les considera como bloques de tiempo. Los eventos ayudan en la adaptación e inspección de algún aspecto. La falta de alguno de los eventos reduce la transparencia por lo que da oportunidad a la pérdida de inspección y adaptación [24].

La iteración

Es el corazón de *Scrum*, básicamente es un contenedor de eventos por lo que se constituye como un bloque de tiempo de un mes o menos durante el cual se crea un incremento de producto “terminado”. Una **iteración** comienza inmediatamente después de que se termine la **iteración** anterior. La **iteración** alcanza un objetivo y tiene un plan flexible, un diseño y una definición de cómo se construirá. Si el objetivo de la iteración queda obsoleto solo el dueño del producto puede cancelarlo [24].

Planificación de la iteración

La **planificación de la iteración** tiene un máximo de duración de ocho horas para un *sprint* de un mes. Mediante un trabajo colaborativo entre el **equipo Scrum** se crea el plan para

definir el trabajo que se va a realizar en una **iteración**, lo que es conocido como la planificación de la iteración [24].

Objetivo del Incremento

Es el objetivo que se va a cumplir en el sprint mediante la implementación de la **lista de producto**. Para el cumplimiento del objetivo de la **iteración** se debe de implementar la funcionalidad y tecnología [24].

Iteración diaria

Es una reunión de un bloque de tiempo de 15 minutos para sincronizar actividades y crear un plan para las siguientes 24 horas llevado a cabo por el equipo de desarrollo. Las reuniones se llevan a cabo a la misma hora y en el mismo lugar. Como entrada tiene al trabajo realizado desde la última iteración diaria y se define el trabajo que podría finalizarse para ese día [24].

Revisión de la iteración

Es una reunión informal y restringida en la cual se presenta el avance de la tarea con el fin de obtener retroalimentación. Usualmente para la **iteración** de un mes se realiza una reunión de cuatro horas. Para iteraciones más cortas se las hace en tiempos más cortos.

El resultado final es una **lista de producto** revisada, en la cual se definen los elementos de la **lista de producto** del siguiente sprint [24].

Retrospectiva de la iteración

Tienen lugar después de la **revisión de la iteración** y antes de la siguiente planificación de la **iteración**. Es una oportunidad para que el **equipo Scrum** identifique falencias y mejoras en sí mismo y de crear un plan de mejoras que serán implementadas para la siguiente **iteración** [24].

1.9.4. Artefactos de *scrum*

Lista de Producto

Es una lista en la cual se encuentran todas las características, funcionalidades, requisitos, mejoras y correcciones en la que se constituyen los cambios a realizarse sobre el producto para entregas futuras. Cada elemento tiene como característica la descripción, el orden, la estimación y el valor.

El responsable de esta lista es el **dueño del producto** y por el deben de ser aprobados todos los cambios a realizarse en la **lista de producto**.

Lista de Pendientes de la iteración

Es una lista de los elementos que son seleccionados de la **lista de producto** y constituyen las tareas para la siguiente iteración. Viene acompañado con un plan el cual especifica el **objetivo de la iteración** y como conseguirlo.

Incremento

Son los elementos completados de una **lista de producto** durante una iteración sumado a los **incrementos** de las **iteraciones** anteriores. Al final de cada iteración se debe de tener un **incremento** terminado, es decir que está listo para usarse.

CAPITULO 2: DESARROLLO DE LA METODOLOGÍA

En este capítulo se detalla cómo se desarrolló la aplicación de acuerdo con la metodología escogida *Scrum*. En *Scrum*, cada sprint es una iteración que entrega un incremento terminado como resultado. Para la aplicación web desarrollada se presenta como evidencia un incremento terminado. Este incremento se lo considera terminado cuando, a cada tarea asignada a dicha iteración, cambie su estado progresivamente de “pendiente” a “en curso” y finalmente a “terminada”.

2.1 Roles

La **Tabla 7** define los roles del equipo de desarrollo:

Rol	Persona Encargada	Descripción
Propietario del Producto	PhD. María Hallo	Es quien da a conocer los requisitos que deben de especificarse en la Lista de Producto
Facilitador Scrum	PhD. María Hallo	Es quién planifica las tareas que el equipo <i>scrum</i> debe realizar.
Grupo Scrum	Lenin Montalvo	Son los encargados del desarrollo de la aplicación con las diferentes tecnologías de desarrollo.
	Jonathan Pachacama	

Tabla 7. Roles del equipo *scrum*

2.2 Requerimientos

Los requerimientos son definidos por propietario del producto. Los requisitos del cliente en la metodología *Scrum* se lo denomina **Pila del producto**. La pila de producto es una lista ordenada de todo aquello que el propietario del producto cree que necesita el producto

[25]. En la **Tabla 8** se presenta la prioridad y la estimación que se asignan a los requerimientos. En la **Tabla 9** se listan los requerimientos que el dueño del producto definió. Los requerimientos de la tabla contienen: un identificador para cada requerimiento, un nombre, una descripción, el número de prioridad y la estimación.

	ESTIMACIÓN	PRIORIDAD
1	Fácil	Baja
2	Media	Medio
3	Compleja	Medio Alto
4	Muy compleja	Alto

Tabla 8. Estimación y prioridad para listar los requerimientos

ID requerimientos	Nombre	Nro. De Prioridad	Descripción	Estimación
RQ001	Utilizar el sistema de manejo de artículos científicos de acceso abierto [26].	4	El investigador podrá hacer uso de una aplicación de búsqueda ya implementada.	2
RQ002	Utilizar el gestor de referencias del sistema de manejo de artículos científicos de acceso abierto.	4	El investigador podrá hacer uso de un gestor de referencias, específicamente <i>wikindx</i> .	3
RQ003	Registrar usuarios.	4	El investigador podrá tener una cuenta para utilizar la aplicación.	2
RQ004	Utilizar biblioteca.	3	El investigador tendrá acceso a una biblioteca donde podrá visualizar los artículos guardados.	2
RQ005	Visualizar citas bibliográficas.	4	El investigador podrá generar referencias bibliográficas del artículo seleccionado.	4
RQ006	Obtener artículos relacionados.	4	El investigador podrá encontrar artículos similares de un artículo guardado previamente en <i>wikindx</i> haciendo uso del módulo de recomendación.	3

Continuación de la Tabla 9

RQ007	Ampliar resultados.	4	El investigador podrá expandir la información de un artículo seleccionado de la lista retornada del módulo de recomendación.	4
RQ008	Manipular resultados.	3	El investigador podrá manipular y guardar los resultados de las recomendaciones obtenidas.	4
RQ009	Redireccionar artículo.	3	El investigador podrá dirigirse a la dirección web del artículo recomendado.	2

Tabla 9. Requerimientos (*product backlog*)

Definición de historias de usuarios

Las **historias de usuario** son usadas en las metodologías ágiles para la recolección de requisitos [25]. La historia de usuario puede estar compuesta de diferentes partes dependiendo del tipo de proyecto, para este proyecto se optaron las siguientes partes [25]:

- **ID:** Es un identificador único para la historia de usuario.
- **Usuario:** La persona a quien va encargada la funcionalidad.
- **Nombre de historia:** Es un título descriptivo para la historia de usuario.
- **Prioridad en negocio:** Nos permite determina el orden en lek que las historias de usuario van a implementarse.
- **Riesgo en desarrollo:** Es un riesgo técnico o funcional asociado a la implementación de la historia de usuario.
- **Puntos estimados:** Es un indicador que establece la complejidad de la historia de usuario, en el cual se puede estimar el tiempo a demorarse en completar esa historia.
- **Programador responsable:** Es quien se encarga del desarrollo de la historia de usuario.
- **Iteración asignada:** Es la iteración en la cual se va a desarrollar la historia de usuario.
- **Descripción:** Es una descripción que responde a las siguientes preguntas: ¿Quién se beneficia?, ¿Qué se quiere?, ¿Cuál es el beneficio?

- **Observaciones:** Sirve para hacer aclaraciones acerca de las restricciones y dependencias que se puede tener en el desarrollo de la historia de usuario.

Historias de usuarios para la primera iteración.

Una vez definido los requerimientos para la primera iteración, se definirán en las **Tablas 10, 11 y 12** las historias de usuario para la primera iteración.

Utilizar el sistema de manejo de artículos científicos de acceso abierto.

En esta historia de usuario se utilizará el buscador del sistema de manejo de artículos científicos de acceso abierto [26].

Historia de usuario	
Número: 1	Usuario: Investigador
Nombre historia: Utilizar el sistema de manejo de artículos científicos de acceso abierto	
Puntos estimados: 2	Iteración asignada: 1
Programador responsable: Jonathan Pachacama	
Descripción: El investigador podrá utilizar el buscador del “sistema de manejo de artículos científicos” para poder expandir sus fuentes bibliográficas.	
Observaciones: El desarrollo del sistema de recomendación para artículos de revistas científicas abiertas utilizando tecnologías de datos enlazados se realizará en base al sistema de manejo de artículos científicos de acceso abierto[26]	
Prioridad en negocio: Alto	Riesgo en desarrollo: Media

Tabla 10. Historia de usuario 1

Utilizar el gestor de referencias del sistema de manejo de artículos científicos de acceso abierto.

En esta historia de usuario se utiliza el gestor de referencias del sistema de manejo de artículos científicos de acceso abierto [26].

Historia de usuario	
Número: 2	Usuario: Investigador
Nombre historia: Utilizar el gestor de referencias (<i>wikindx</i>) del sistema de manejo de artículos científicos de acceso abierto.	
Prioridad en negocio: Medio Alto	Riesgo en desarrollo: Media
Puntos estimados: 3	Iteración asignada: 1
Programador responsable: Jonathan Pachacama	
Descripción: El investigador podrá guardar en el gestor de referencias <i>wikindx</i> los artículos encontrados en el repositorio de DOAJ para poder generar citas bibliográficas.	
Observaciones: El sistema mostrará un mensaje de confirmación de que se guardó el artículo	

Tabla 11. Historia de usuario 2

Registrar usuarios

En esta historia de usuario se registrará el usuario con el que se utilizará la aplicación, la historia de usuario se la presenta en la **Tabla 12**.

Historia de usuario	
Número: 3	Usuario: Investigador
Nombre historia: Registrar usuarios	
Prioridad en negocio: Alto	Riesgo en desarrollo: Media
Puntos estimados: 2	Iteración asignada: 1
Programador responsable: Lenin Montalvo y Jonathan Pachacama	
Descripción: El investigador podrá registrarse en la aplicación, donde ingresará la información necesaria para tener su propio perfil y gestionar sus artículos guardados o creados.	
Observaciones: El sistema mostrará una pantalla para ingresar a la aplicación	

Tabla 12. Historia de usuario 3

Definición de historias de usuarios para la segunda iteración.

Las **Tablas 13** y **14** muestran las historias de usuario para la segunda iteración correspondientes a utilizar biblioteca y visualizar citas bibliográficas de acuerdo con los requerimientos definidos por el dueño del producto.

Utilizar biblioteca

En esta historia de usuario se podrá hacer uso de una biblioteca donde se enlistarán los resultados de los artículos guardados en las bases de datos.

Historia de usuario	
Número: 4	Usuario: Investigador
Nombre historia: Utilizar biblioteca	
Prioridad en negocio: Medio Alto	Riesgo en desarrollo: Media
Puntos estimados: 2	Iteración asignada: 2
Programador responsable: Lenin Montalvo y Jonathan Pachacama	
Descripción: El investigador podrá visualizar una lista de resultados de los artículos guardados en el <i>wikindx</i> según los parámetros que ingrese, para poder editar y eliminar un artículo seleccionado utilizando la interface de <i>wikindx</i> .	
Observaciones: <ul style="list-style-type: none">- Los artículos serán presentados en una lista que contiene 10 artículos y se podrá visualizar más artículos mediante una paginación no numerada,- Los artículos se podrán buscar al ingresar el título completo o tan solo algunas palabras.	

Tabla 13. Historia de usuario 4

Visualizar citas bibliográficas

En esta historia de usuario el investigador podrá visualizar las citas bibliográficas de los artículos guardados en *wikindx*.

Historia de usuario	
Número: 5	Usuario: Investigador
Nombre historia: Visualizar citas bibliográficas	

Continuación de Tabla 14

Prioridad en negocio: Alto	Riesgo en desarrollo: Media
Puntos estimados: 4	Iteración asignada: 2
Programador responsable: Lenin Montalvo y Jonathan Pachacama	
Descripción: El usuario podrá generar la cita bibliográfica de un artículo utilizando la interface de <i>wikindx</i> para poder facilitar la tarea de crear una cita bibliográfica.	
Observaciones: Las citas serán generadas solo con los metadatos guardados en la base de datos sincronizada con <i>wikindx</i> .	

Tabla 14. Historia de usuario 5

Definición de historias de usuarios para la tercera iteración.

Para la tercera iteración solo se definió una historia de usuario correspondiente a la parte del módulo de recomendaciones.

Obtener artículos relacionados.

La **Tabla 15** indica la historia de usuario para obtener artículos relacionados.

Historia de Usuario	
Número: 6	Usuario: Investigador
Nombre historia: Obtener artículos relacionados	
Prioridad en negocio: Alta	Riesgo en desarrollo: Baja
Puntos estimados: 3	Iteración asignada: 3
Programador responsable: Lenin Montalvo y Jonathan Pachacama	
Descripción: El investigador podrá encontrar artículos similares de un artículo guardado previamente en <i>Wikindx</i> haciendo uso del módulo de recomendación, para poder ampliar sus fuentes bibliográficas.	
Observaciones: - El módulo de recomendación tan solo retornará los artículos de los repositorios de datos seleccionado: IEEE, ACM o DBLP. - Los resultados se presentarán en una lista, siendo el primero el de mayor relevancia.	

Tabla 15. Historia de usuario 6

Definición de historias de usuarios para la cuarta iteración.

Para la última iteración se definió las siguientes historias de usuario correspondientes los requerimientos de: ampliar resultados, manipular resultados y redireccionar artículos.

Las **Tablas 16, 17 y 18** muestran las historias de usuario 7, 8 y 9

Ampliar resultados

En esta iteración se tiene como objetivo poder ampliar los resultados obtenidos del módulo de recomendación. Por ello se tomará en cuenta el entregable realizado en la tercera iteración, que es en donde se presenta la lista de los artículos relacionados según el repositorio de datos elegido.

Historia de usuario	
Número:7	Usuario: Investigador
Nombre historia: Ampliar resultados.	
Prioridad en negocio: Alta	Riesgo en desarrollo: Media
Puntos estimados: 4	Iteración asignada: 4
Programador responsable: Lenin Montalvo y Jonathan Pachacama	
Descripción: El investigador podrá expandir la información de un artículo seleccionado de la lista retornada del módulo de recomendación, para poder visualizar sus metadatos.	
Observaciones: <ul style="list-style-type: none">- Se hará uso de consultas <i>sparql</i>/ para realizar la consulta a los repositorios <i>RKB Explorer</i>.- El investigador no introducirá las consultas <i>sparql</i>, estas funcionarán implícitamente en la codificación del programa.	

Tabla 16. Historia de usuario 7

Manipulación de resultados

En esta historia de usuario se podrá ingresar los metadatos faltantes en un artículo y guardarlos en la base de datos del gestor de referencia (*wikindx*).

Historia de Usuario	
Número: 8	Usuario: Investigador
Nombre historia: Manipular resultados	

Continuación de Tabla 17

Prioridad en negocio: Medio Alta	Riesgo en desarrollo: Media
Puntos estimados: 4	Iteración asignada: 4
Programador responsable: Lenin Montalvo y Jonathan Pachacama	
Descripción: El investigador podrá manipular los metadatos de los artículos obtenidos y guardarlos, para poder corregir o aumentar información que no se obtuvo del repositorio.	
Observaciones: - Los resultados de ACM, DBLP e IEEE no contienen metadatos suficientes para generar citas bibliográficas completas, el investigador debe ingresar manualmente los campos necesarios.	

Tabla 17. Historia de usuario 8

Redireccionar artículo

En esta historia de usuario se podrá redireccionar a la página web del artículo recomendado.

Historia de usuario	
Número: 9	Usuario: Investigador
Nombre historia: Redireccionar artículo	
Prioridad en negocio: Medio Alta	Riesgo en desarrollo: Media
Puntos estimados: 2	Iteración asignada: 4
Programador responsable: Lenin Montalvo y Jonathan Pachacama	
Descripción: El investigador podrá dirigirse a la dirección web del artículo recomendado, para obtener el artículo completo.	
Observaciones: - No todos los artículos contienen la información de su dirección web.	

Tabla 18. Historia de usuario 9

2.3 Iteraciones

2.3.1. Primera iteración

Planificación de la primera iteración

La primera iteración comienza a partir del 30 de enero del 2018 con una jornada de 4 horas diarias durante 22 días de lunes a viernes. Los días 12 y 13 de febrero se consideran festivos debido al feriado de carnaval y no se realizó ningún avance en esos días. La iteración terminará el 28 de febrero del 2018 de acuerdo con las acotaciones del facilitador del proyecto. Para la primera iteración se tomará en cuenta los requerimientos RQ001, RQ002 y RQ003 de la lista de requerimientos, tal como se muestra en la **Tabla 19**.

ID requerimientos	Nombre	Nro. De Prioridad	Descripción	Estimación
RQ001	Utilizar el sistema de manejo de artículos científicos de acceso abierto [26].	4	El investigador podrá hacer uso de una aplicación de búsqueda ya implementada.	2
RQ002	Utilizar el gestor de referencias del sistema de manejo de artículos científicos de acceso abierto.	4	El investigador podrá hacer uso de un gestor de referencias, específicamente <i>wikindx</i> .	3
RQ003	Registrar usuarios.	4	El investigador podrá tener una cuenta para utilizar la aplicación.	2

Tabla 19. Requerimientos para la primera iteración

Objetivo de la primera iteración

El objetivo para esta iteración es el levantamiento del sistema de manejo de artículos científicos de acceso abierto [26], para su posterior integración con el módulo de recomendación. Además de una implementación para registrar y autenticar usuarios. Una vez configurado el sistema con su respectiva base de datos, con su gestor de referencia utilizados y un Login funcional se podrá continuar con la siguiente iteración.

Lista de tareas para la primera iteración.

En la **Tabla 20** se detalla las tareas designadas para los requerimientos RQ001, RQ002 y RQ003 considerando su estimación en tiempo y el responsable de cada tarea. De acuerdo con la tabla de tareas se trabajará un total de 80 horas para la primera iteración.

ID Tareas	Nombre	Responsable	Tiempo Estimado	Estimación
T-001	Adaptación del sistema de manejo de artículos científicos de acceso abierto [26].	Jonathan Pachacama	20	4
T-002	Configuración del gestor de base de datos <i>phpMyadmin</i> .	Jonathan Pachacama	4	3
T-003	Pruebas de la integración de la aplicación sistema de manejo de artículos científicos de acceso abierto [26] y consumo de la API de <i>RKB Explorer</i> .	Jonathan Pachacama	8	3
T-004	Adaptación de los modelos definidos en el sistema de manejo de artículos científicos de acceso abierto [26] implementado en <i>sails.js</i> .	Jonathan Pachacama	4	3
T-005	Elaboración del modelamiento de la base de datos del <i>wikindx</i> .	Jonathan Pachacama	4	4
T-006	Elaboración del modelamiento de la base de datos "investigador".	Equipo	4	3
T-007	Elaboración de prototipo interfaz de registro y perfil de usuario.	Equipo	4	2
T-008	Elaboración prototipo interfaz de <i>Login</i> .	Equipo	4	2
T-009	Programación de funcionalidad de <i>login</i> .	Equipo	8	2
T-010	Programación de funcionalidad de registro de usuario.	Equipo	8	2
T-011	Pruebas de funcionamiento del registro de usuarios y funcionamiento de <i>Login</i> .	Equipo	12	3
Total, Horas			80	

Tabla 20. Tareas para la primera iteración

Revisión de la primera iteración

Para realizar los seguimientos de cada una de las iteraciones se usa una plantilla *Scrum Anexo I*.

Para el seguimiento de la primera iteración se determinará la configuración que se muestra en la **Tabla 21**.

Proyecto			
DESARROLLO DE UN SISTEMA DE RECOMENDACIÓN PARA ARTÍCULOS DE REVISTAS CIENTÍFICAS ABIERTAS UTILIZANDO TECNOLOGÍAS DE DATOS ENLAZADOS			
Nº de sprint	Inicio	Días	Jornada
1	30-ene.-18	22	4
TAREAS		EQUIPO	FESTIVOS
TIPOS	ESTADOS		
Análisis	Pendiente	Jonathan	12-feb.
Codificación	En curso	Lenin	13-feb.
Prototipado	Terminada	Equipo	
Pruebas	Eliminada		
Reunión			

Tabla 21. Datos generales de la primera iteración

De acuerdo con el formato establecido, las tareas presentan el estado de “terminada” una vez se haya cumplido con las horas definidas. En la **Tabla 22** se presentan en detalle los días y las horas asignadas a cada tarea correspondientes al primer sprint con su respectivo responsable y tipo de tarea realizada.

PILA DEL SPRINT				
Backlog ID	Tarea	Tipo	Estado	Responsable
Utilizar el sistema de	Adaptación del sistema de manejo de artículos científicos de acceso	Codificación	Terminada	Jonathan
	Configuración del gestor de base de datos	Codificación	Terminada	Jonathan
	Pruebas de la integración de la	Pruebas	Terminada	Jonathan
Utilizar el gestor de referencias	Adaptación de los modelos definidos en el sistema de manejo de artículos científicos de acceso	Codificación	Terminada	Jonathan
	Elaboración del modelamiento de la base de datos del Wikindx.	Análisis	Terminada	Jonathan
Registrar usuarios	Elaboración del modelamiento de la base de datos "investigador".	Análisis	Terminada	Equipo
	Elaboración de prototipo interfaz de registro y perfil de usuario.	Prototipado	Terminada	Equipo
	Elaboración prototipo interfaz de	Prototipado	Terminada	Equipo
	Programación de funcionalidad de	Codificación	Terminada	Equipo
	Programación de funcionalidad de re	Codificación	Terminada	Equipo
	Pruebas de funcionamiento del regis	Pruebas	Terminada	Equipo

Tabla 22. Tareas finalizadas de la primera iteración

Integración del sistema de manejo de artículos científicos de acceso abierto y el sistema de recomendación.

Para desarrollar el sistema de recomendación previamente se tuvo que configurar e instalar el sistema de manejo de artículos científicos de acceso abierto [26]. Este sistema forma parte del proyecto interno del DICC PII-16-06. El sistema de recomendación desarrollado en el presente proyecto se integrará al sistema de manejo de artículos científicos de acceso abierto [26] y formarán un solo sistema, mejorando su propio módulo de recomendación donde no se usan datos enlazados.

Las herramientas para utilizar son las siguientes:

- Aplicación web desarrollada en el framework Sails.js versión 0.12.14
- Arquitectura: Modelos-Vista-controlador.
- IDE: WebStorms.
- Entorno de desarrollo web: WampServer versión 3.0.6.
- Gestor de referencias bibliográficas wikindx v5.3.2.

Configuración Base de datos

- Nombre de las bases de datos: Investigador, wikindx5
- Base de datos: Mysql versión 5.7.14

El sistema de manejo de artículos científicos de acceso abierto utilizado es un sistema web online que facilita el proceso de investigación [26]. El sistema permite buscar artículos en varios repositorios bibliográficos de acceso abierto, almacenarlos en una biblioteca personal y generar citas bibliográficas. El sistema de artículos científicos de acceso abierto [26] consta de 6 módulos como se muestra en la **Figura 20**.

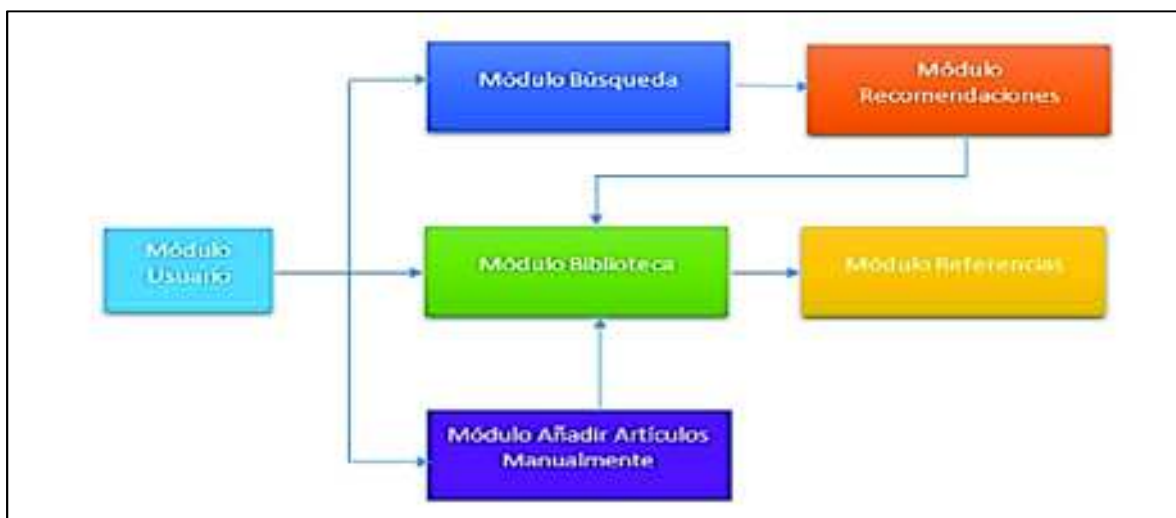


Figura 20. Módulos del sistema de manejo de artículos científicos de acceso directo

La aplicación cuenta con 6 módulos, que son los siguientes:

- **Módulo usuario:** Este módulo consta de un registro de usuario y un login.
- **Módulo buscador:** Este módulo se encarga de la extracción los datos y metadatos de los artículos buscados
- **Módulo recomendaciones:** Este módulo presenta recomendaciones de artículos relacionados a la búsqueda.
- **Módulo biblioteca:** Este módulo permite acceder a todos los datos de los artículos guardados
- **Módulo referencia:** Obtiene referencias en distintos formatos de los recursos añadidos
- **Módulo añadir artículos manualmente:** Este módulo permite añadir artículos que no se encuentran en el módulo de búsqueda y almacenarlos en la biblioteca

Para el presente proyecto se mejora el módulo de recomendación, el módulo de usuarios y el módulo de biblioteca implementando una biblioteca compartida con todos los usuarios para visualizar editar y eliminar los artículos guardados en *wikindx*. El sistema de manejo de artículos científicos de acceso directo consta de 22 funcionalidades distintas [26].

Las funcionalidades principales que se van a modificar por completo en el sistema de recomendación son:

Módulo usuario

- Registro de usuario.
- Login.

Módulo recomendaciones

- Presentar al usuario recomendaciones de artículos relacionados al tema de búsqueda.
- Visualizar los datos de los artículos recomendados.
- Guardar en la biblioteca personal los datos de los artículos recomendados que sean seleccionado.
- Visualizar las referencias de los artículos presentados en diferentes formatos.

Módulo Biblioteca

- Acceder a todos los datos de los artículos guardados las veces que se lo requiera.
- Editar datos a los artículos guardados.
- Añadir notas a los artículos guardados.
- Añadir, borrar y descargar archivos.

- Visualizar las referencias del artículo en diferentes formatos.

Las funcionalidades del sistema de manejo de artículos científicos de acceso directo que se adaptarán al sistema de recomendaciones son:

Módulo Buscador de artículos científicos de acceso abierto

- Búsqueda y recuperación de artículos alojados en repositorios de artículos científicos de acceso abierto.
- Búsqueda de artículos científicos por categoría
- Visualizar las referencias de los artículos presentados en diferentes formatos.
- Guardar en la biblioteca personal los datos de los artículos seleccionados.

Módulo Referencias:

- Obtener referencias en distintos formatos de los recursos que ha añadido.
- Exportar e importar referencias, aprovechando las funcionales de Wikindx que es un entorno virtual de investigación gratuito (un sistema bibliográfico y de redacción de citas / notas y creación de artículos en línea).

Módulo Añadir Artículos manualmente:

- Añadir manualmente artículos que no se encuentren dentro del módulo búsqueda y guardarlos en su biblioteca personal.

Arquitectura de un sistema de gestión de referencia

La arquitectura del sistema de recomendación está basada en la propuesta de [1], donde se muestra una arquitectura para mejorar un sistema de gestión de referencia con recomendaciones de datos vinculados abiertos. A continuación, la **Figura 21** presenta la arquitectura propuesta en [1].

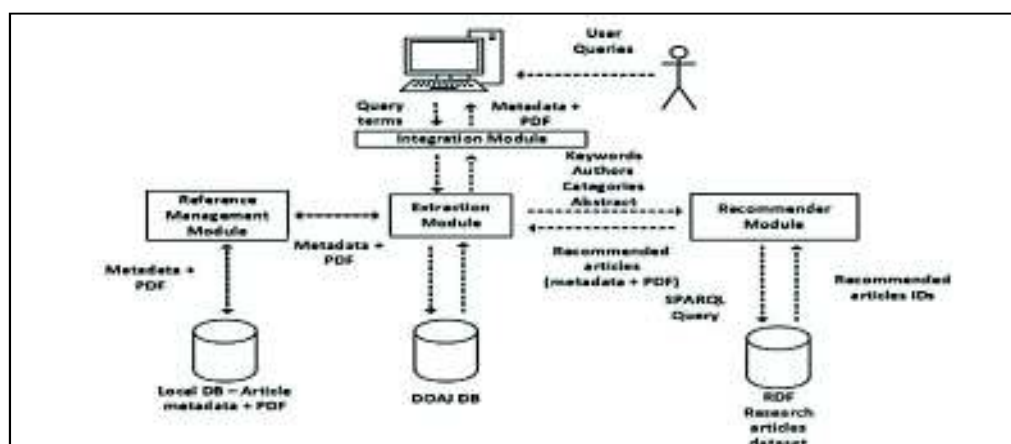


Figura 21. Arquitectura para mejorar un sistema de gestión de referencia con recomendaciones de datos vinculados abiertos [1]

Para el desarrollo de la primera iteración se adaptó el sistema de manejo de artículos científicos de acceso abierto [26] a una plantilla de aplicación web de código abierto *AdminLTE* creada por *Almsaeed Studio* [27]. Para lo cual se instaló las herramientas necesarias y así se pudo levantar la aplicación creada en Sails.js.

La **Figura 22** muestra el sistema de manejo de artículos científicos [26] con la plantilla que se utilizara en el sistema de recomendación desarrollado.



Figura 22. Sistema de manejo de artículos científicos con la plantilla.

Para la instalación y configuración del gestor de referencias *wikindx* del sistema de manejo de artículos científicos de acceso abierto se utilizó *WampServer* de versión 3.0.6, que es un entorno de desarrollo web con el fin de realizar las pruebas de conexión de la aplicación con las bases de datos “investigador” y “wikidx5”, dichas bases de datos se detallaran posteriormente.

Se utilizó al principio la aplicación *wikindx5.3.2* en cual se ubica en el directorio C:\wamp64\www. El archivo *config.php* de la aplicación *wikindx5.3.2* se adaptó con las configuraciones principales para nuestro sistema. La **Tabla 23** muestra las configuraciones que utilizo el sistema de recomendación

CONFIGURACIÓN DE LA BASE DE DATOS	
Configuración	Descripción
\$WIKINDX_DB_HOST = "localhost";	Host en el que se ejecuta el sistema de administración de base de datos relacional

Continuación de la Tabla 23

\$WIKINDX_DB = "wikindx5";	Nombre de la base de datos con la que interactúan los scripts
\$WIKINDX_DB_USER = "wikindx";	Nombre de usuario requerido para conectarse y abrir la base de datos.
\$WIKINDX_DB_PASSWORD = "wikindx";	Contraseña requerida para conectarse y abrir la base de datos.
\$ WIKINDX_DB_TYPE = "mysql";	Esta versión solo admite MySQL

Tabla 23. Descripción de la configuración de *wikindx*.

La **Figura 23** muestra el contenido del archivo *config.php* modificado para el desarrollo del sistema de recomendación

```

config.php: Bloc de notes
Archivo Edición Formato Ver Ayuda
//
// Host on which the relational db management system (i.e. the MySQL server) is running (usually localhost if
// the web files are on the same server as the ROBWS although some web hosting services may specify something like
// localhost:/tmp/mysql5.sock)
public $WIKINDX_DB_HOST = "localhost";
// name of the database which these scripts interface with:
public $WIKINDX_DB = "wikindx5";
// username and password required to connect to and open the database
// (it is strongly recommended that you change these default values):
public $WIKINDX_DB_USER = "wikindx";
public $WIKINDX_DB_PASSWORD = "wikindx";
// If using WIKINDX on a shared database, set the WIKINDX table prefix here (lowercase only)
// (do NOT change after running WIKINDX and creating the tables!):
public $WIKINDX_DB_TABLEPREFIX = 'wix_';
// This version only supports 'mysql' so do not change this.
public $WIKINDX_DB_TYPE = "mysql";
// WIKINDX uses MySQL persistent connections by default.
// Some hosting services are not configured for this: if you have problems
// connecting to your MySQL server and/or receive error messages about 'too many connections',
// set $WIKINDX_DB_PERSISTENT to FALSE
public $WIKINDX_DB_PERSISTENT = TRUE;
/*****

```

Figura 23. Archivo *config.php* (wikindx5.3.2).

Es necesario crear un administrador y configurarlo para utilizar *wikindx*. Una vez realizado la instalación y las configuraciones necesarias, la **Figura 24** muestra la pantalla principal de *wikindx* funcionando.

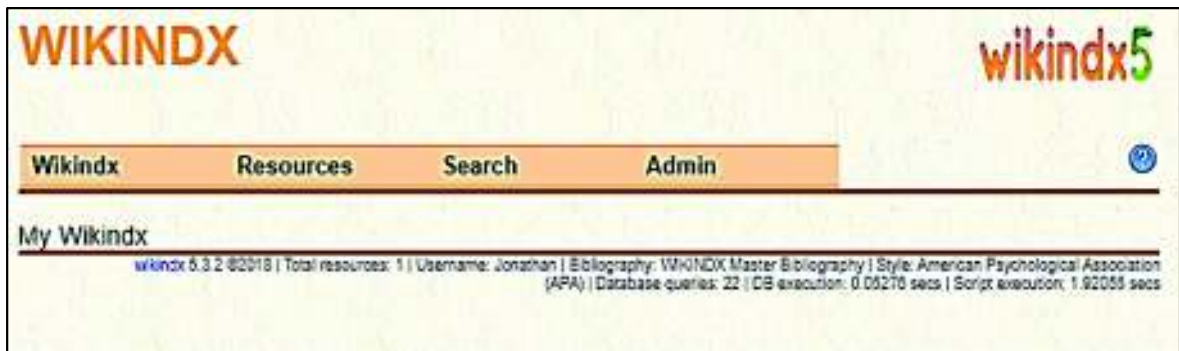


Figura 24. Pantalla principal wikindx

Una vez levantado el sistema y conectado a las bases de datos en la primera iteración se considera las tareas de elaborar prototipos de interfaces para el inicio de sesión y registro de nuevos usuarios. En la **Figuras 25** muestra la interfaz de Login.



Figura 25. interfaz login.

La **Figuras 26** muestra la interfaz de registro de usuario.



Figura 26. Prototipo de interfaz registro de usuario.

Retrospectiva de la primera iteración

El gráfico de avance o “*burndown*” es el gráfico que actualiza el equipo en las reuniones de seguimiento de la iteración, para monitorizar el ritmo de avance, y detectar de forma temprana posibles desviaciones sobre la previsión que pudieran comprometer la entrega al final de una iteración.

En la **Figura 27** debe ser actualizado diariamente, y se registra:

- En el eje Y las horas de trabajo pendiente.
- En el eje X los días de la iteración.

El equipo dispone en la pila de la iteración, de la lista de tareas que va a realizar, y cada una con la estimación del esfuerzo pendiente. Esto es: el primer día, en la pila de tareas figura para cada tarea el esfuerzo que se ha estimado, puesto que aún no se ha trabajado en ninguna de ellas.

Día a día, cada miembro del equipo actualizó en la pila del sprint el tiempo que le queda a las tareas que va desarrollando, hasta que se terminaron.

Con esta información de la pila de la iteración se actualizó el gráfico poniendo cada día el esfuerzo pendiente total de todas las tareas que aún no se han terminado. El avance ideal

de una iteración estaría representado por la diagonal que reduce el esfuerzo pendiente de forma continua y gradual hasta terminarlo en el último día de la iteración.



Figura 27. Esfuerzo realizado para la primera iteración.

La **Figura 28** dibuja la evolución de la primera iteración a través de las tareas pendientes donde:

- En el eje Y se registra las tareas pendientes.
- En el eje X se registran los días de la iteración.

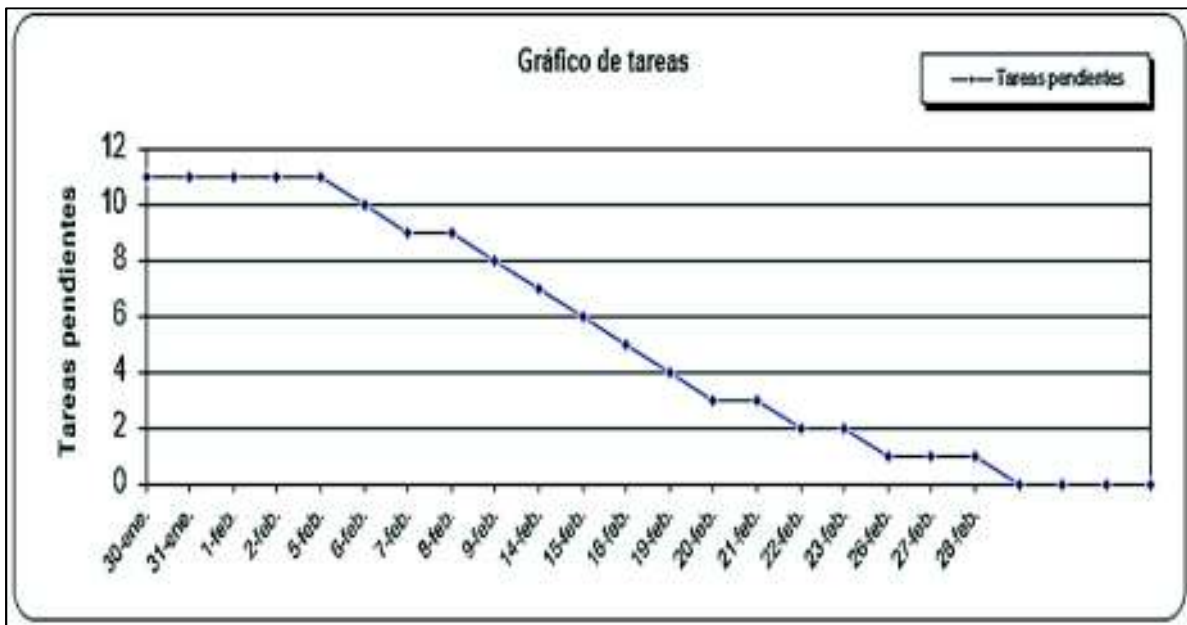


Figura 28. Tareas pendientes para la primera iteración.

2.3.2. Segunda iteración

Planificación de la segunda iteración

Los requisitos usados para la segunda iteración son RQ004 y RQ005 como lo muestra la **Tabla 24**. La primera iteración comenzará el 01 de marzo del 2018 y tendrá una duración de 22 días hasta el 30 de marzo del 2018 con una jornada de 4 horas diarias de lunes a viernes.

ID requerimientos	Nombre	Nro. De Prioridad	Descripción	Estimación
RQ004	Utilizar biblioteca.	3	El investigador tendrá acceso a una biblioteca donde podrá visualizar los artículos guardados.	2
RQ005	Visualizar citas bibliográficas.	4	El investigador podrá generar referencias bibliográficas del artículo seleccionado.	4

Tabla 24. Requerimientos para la segunda iteración

Objetivo de la segunda iteración

El objetivo para esta iteración es implementar un módulo de biblioteca donde el investigador podrá visualizar todos los artículos se hayan guardado y generar citas bibliográficas de dichos artículos.

Lista de tareas para la segunda iteración.

Las tareas que se realizaran en la segunda iteración están detalladas en la **Tabla 25**, con su identificador, el nombre de la tarea con su responsable y el tiempo estimado que va a durar para entregarla como “terminada” en el seguimiento de la segunda iteración con el formato establecido.

ID Tareas	Nombre	Responsable	Tiempo Estimado	Estimación
T-001	Elaboración de prototipo para la interfaz de biblioteca.	Equipo	4	2
T-002	Elaboración de prototipo para editar artículo.	Equipo	4	2

Continuación de la Tabla 25

T-003	Elaboración de consultas SQL para extraer metadatos de la base de datos de <i>wikindx</i> .	Equipo	8	4
T-004	Programación de funcionalidad de biblioteca.	Equipo	8	3
T-005	Implementación de paginación en resultados de biblioteca.	Equipo	8	4
T-006	Programación de funcionalidad para buscar artículos mediante un parámetro de búsqueda.	Equipo	8	3
T-007	Programación de funcionalidad de edición de artículo.	Equipo	4	3
T-008	Programación de funcionalidad de eliminación de artículo.	Equipo	4	3
T-009	Pruebas del funcionamiento de la biblioteca.	Equipo	12	3
T-010	Elaboración de prototipo para la interfaz para mostrar citas bibliográficas generadas.	Equipo	4	2
T-011	Programación de funcionalidad para generación de citas bibliográficas.	Equipo	12	4
T-012	Pruebas de generación de citas bibliográficas.	Equipo	12	3
Total, Horas			88	

Tabla 25. Tareas para la segunda iteración

Revisión de la segunda iteración

Para realizar los seguimientos de cada una de las iteraciones se usa una plantilla *scrum* **Anexo II**.

Para el seguimiento de la segunda iteración se determinará la siguiente configuración como se muestra en la **Tabla 26**.

Proyecto			
DESARROLLO DE UN SISTEMA DE RECOMENDACIÓN PARA ARTÍCULOS DE REVISTAS CIENTÍFICAS ABIERTAS UTILIZANDO TECNOLOGÍAS DE DATOS ENLAZADOS.			
Nº de sprint	Inicio	Dias	Jornada
2	1-mar.-18	22	4
TAREAS		EQUIPO	FESTIVOS
TIPOS	ESTADOS		
Análisis	Pendiente	Jonathan	
Codificación	En curso	Lenin	
Prototipado	Terminada	Equipo	
Pruebas	Eliminada		
Reunión			

Tabla 26. Datos generales de la segunda iteración

Para el seguimiento de esta iteración se muestra los detalles en la **Tabla 27**. Las tareas finalizarán cuando cambien de estado “pendiente” ha “terminado” en los tiempos definidos y por el responsable asignado.

PILA DEL SPRINT				
Backlog ID	Tarea	Tipo	Estado	Responsable
Utilizar biblioteca	Elaboración de prototipo para la interfaz de biblioteca.	Prototipado	Terminada	Equipo
	Elaboración de prototipo para editar	Prototipado	Terminada	Equipo
	Elaboración de consultas SQL para	Codificación	Terminada	Equipo
	Programación de funcionalidad de biblioteca.	Codificación	Terminada	Equipo
	Implementación de paginación en resultados de biblioteca.	Codificación	Terminada	Equipo
	Programación de funcionalidad para buscar artículos mediante un	Codificación	Terminada	Equipo
	Programación de funcionalidad de edición de artículo.	Codificación	Terminada	Equipo
	Programación de funcionalidad de el	Codificación	Terminada	Equipo
	Pruebas del funcionamiento de la biblioteca	Pruebas	Terminada	Equipo
Visualizar c	Elaboración de prototipo para la interfaz de biblioteca.	Prototipado	Terminada	Equipo
	Programación de funcionalidad para	Codificación	Terminada	Equipo
	Pruebas de generación de citas bibliográficas	Pruebas	Terminada	Equipo

Tabla 27. Tareas finalizadas de la segunda iteración

Como se definió en la cuarta historia de usuario el sistema tendrá una biblioteca donde el usuario podrá visualizar en una lista de resultados de los artículos guardados en *wikindx* en la **Figura 29** se muestra la pantalla de la biblioteca implementada.

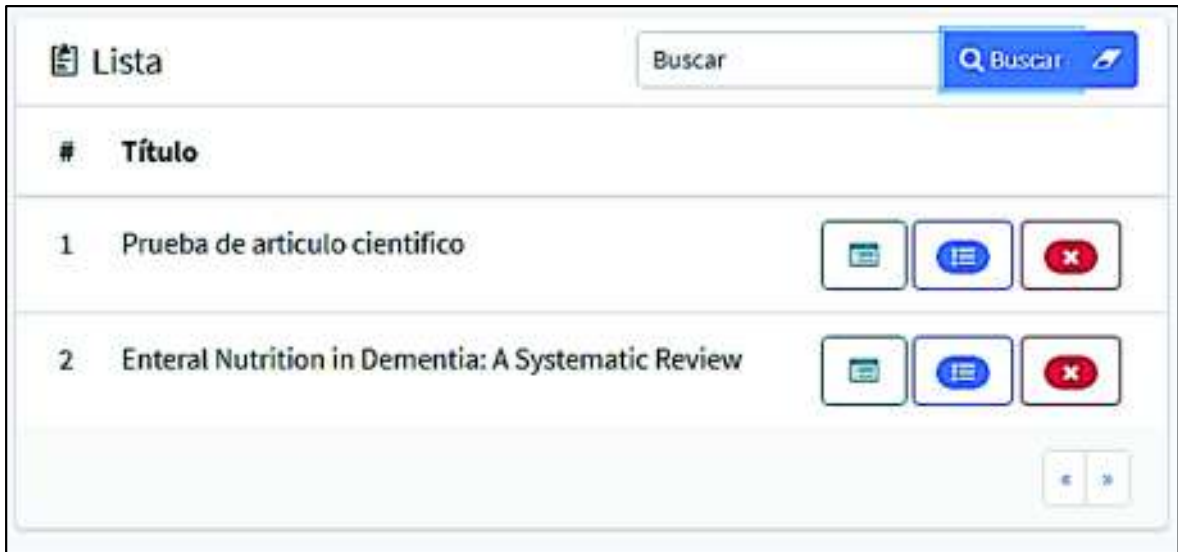


Figura 29. Pantalla de biblioteca.

En la biblioteca el investigador podrá buscar los artículos mediante el título ingresándolo en la barra de búsqueda, **Figura 30**. Además, utilizar los iconos de: editar, cita, recomendación y eliminar como se muestra en la **Figura 31**.



Figura 30. Barra de búsqueda.



Figura 31. Botones biblioteca.

Además, El investigador podrá editar y eliminar el artículo seleccionado utilizando la interface de *wikindx* mediante los botones de la biblioteca. En la **Figura 32** se muestra el artículo en la lista de recomendación.

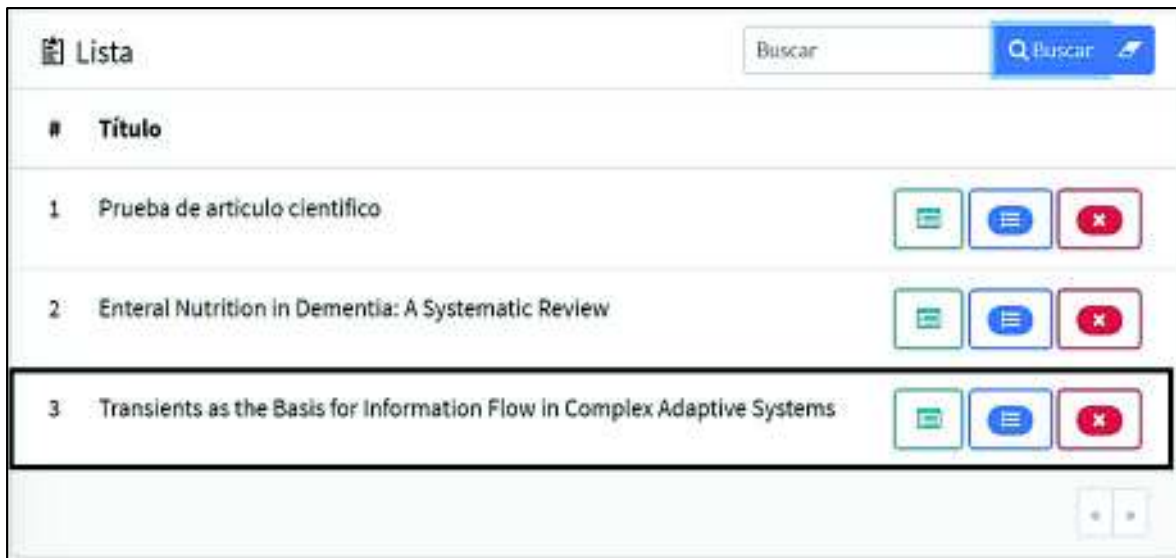


Figura 32. Artículo seleccionado en biblioteca.

En las **Figuras 33** y **34** muestran el artículo seleccionado listo para la confirmación de eliminarlo y para editarlo respectivamente con la interface de *wikindx*.

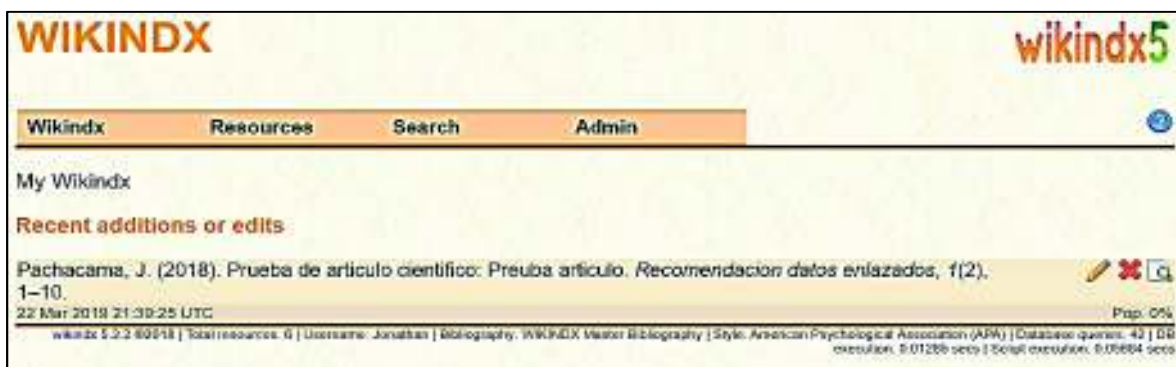


Figura 33. Eliminar artículo *wikindx*.

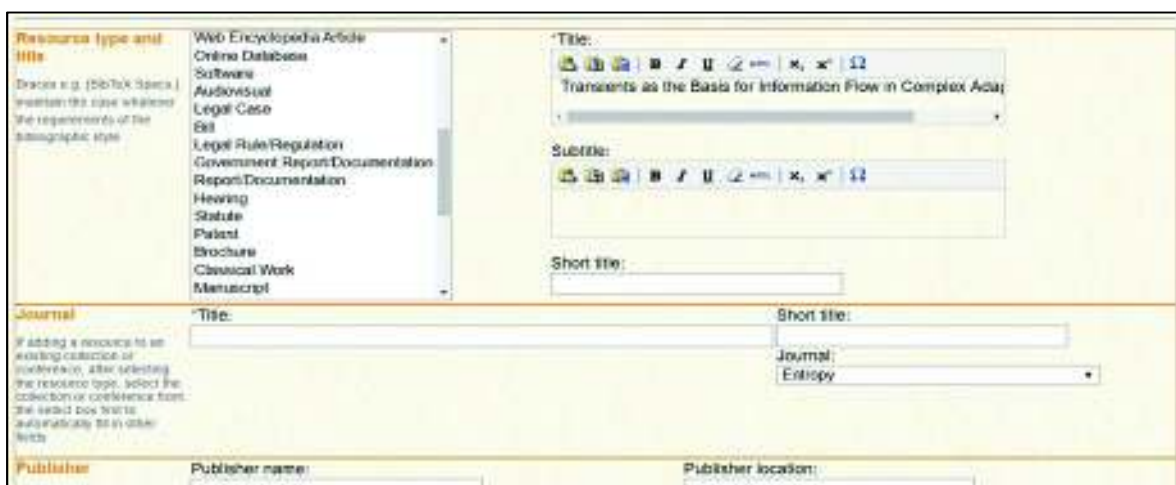


Figura 34. Editar artículo *wikindx*.

Al utilizar un gestor de referencias *wikindx* se busca dar la facilidad al investigador de generar citas bibliográficas utilizando este gestor de referencias en su versión 5 como se muestra en la **Figura 35**. La cita bibliográfica es del artículo seleccionado en la **Figura 32**.

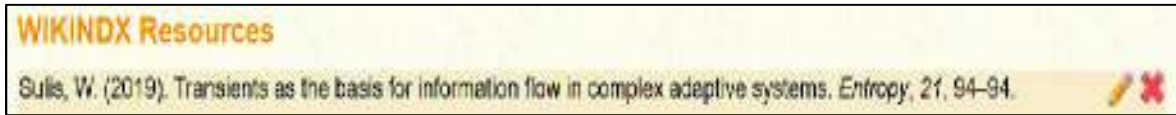


Figura 35. Cita bibliográfica *wikindx*.

Además de la cita bibliográfica *wikindx* nos presenta los metadatos que el artículo contiene como se muestra en la **Figura 36**.

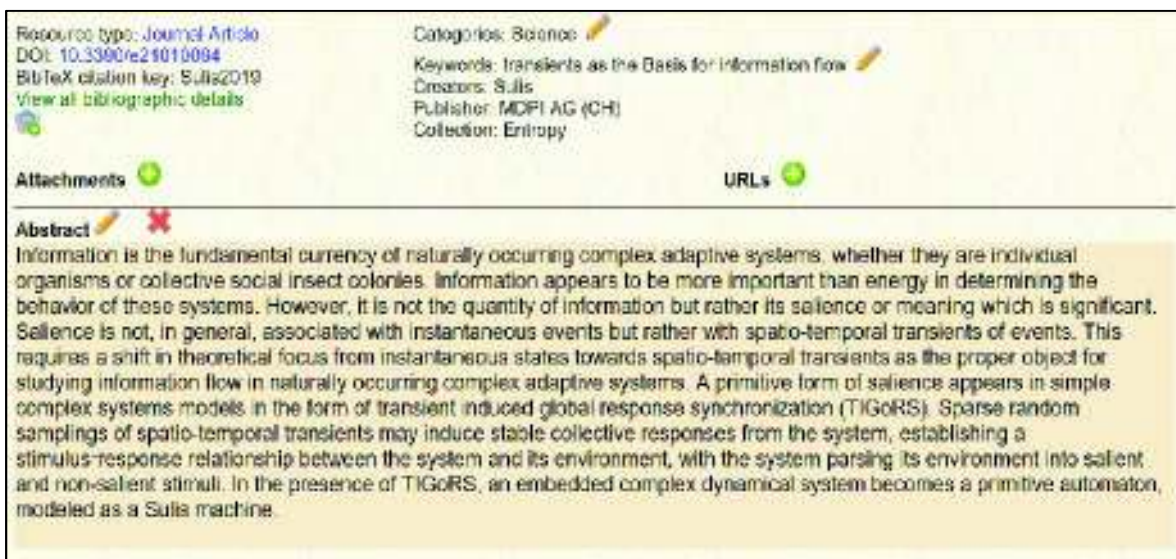


Figura 36. Metadatos del recurso en *wikindx*.

Retrospectiva de la segunda iteración

Para verificar el avance de la iteración se puede observar en la **Figura 37** el esfuerzo que se realizó para la segunda iteración.

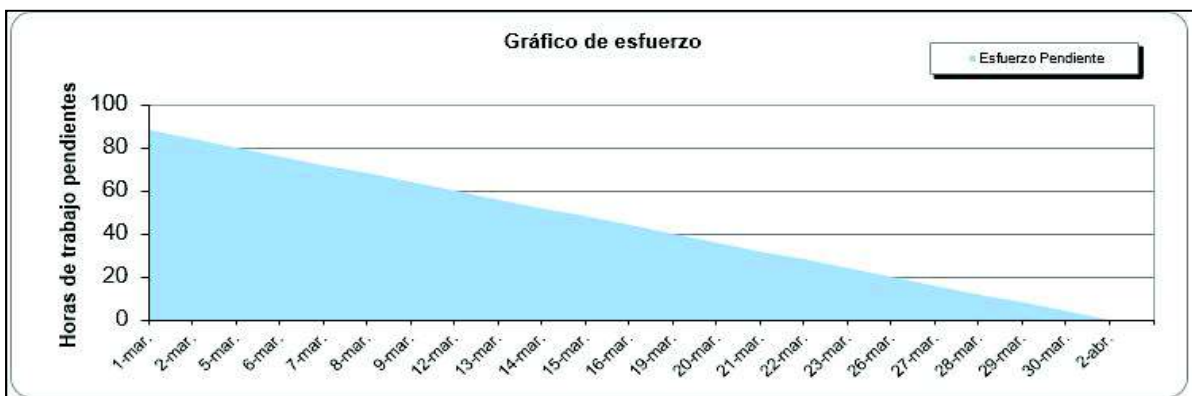


Figura 37. Esfuerzo desarrollado para la segunda iteración

En la **Figura 38** se puede observar la gráfica de tareas realizadas en los días establecidos para cada tarea.

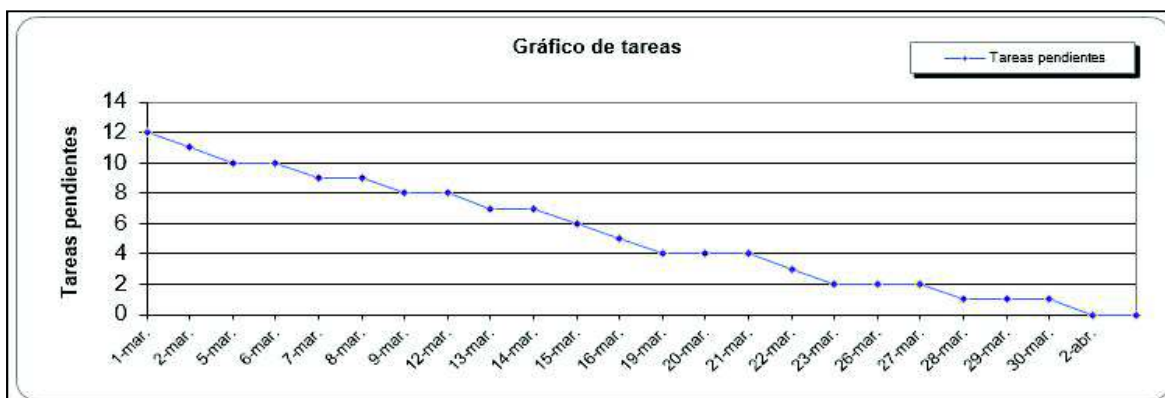


Figura 38. Tareas pendientes para el segundo sprint.

2.3.3. Tercera iteración

Planificación de la tercera iteración

Para la tercera iteración solo se tomó el requerimiento RQ006. Esta interacción se enfocó en el desarrollo del módulo de recomendación. La **Tabla 28** muestra el requerimiento en detalla para la tercera iteración.

ID requerimientos	Nombre	Nro. De Prioridad	Descripción	Estimación
RQ006	Obtener artículos relacionados.	4	El investigador podrá encontrar artículos similares de un artículo guardado previamente en <i>wikindx</i> haciendo uso del módulo de recomendación.	3

Tabla 28. Requerimiento para la tercera iteración

Objetivo de la tercera iteración

En esta iteración se hará énfasis en el algoritmo de recomendación para que el usuario pueda hacer uso de este con los datos que se encuentran en el repositorio *RKB Explorer*, esto se considera como el objetivo para la tercera iteración.

Lista de tareas para la tercera iteración.

En base al requerimiento escogido se ha hecho una lista de las siguientes tareas que se especifican en la siguiente **Tabla 29**.

ID Tareas	Nombre	Responsable	Tiempo Estimado	Estimación
T-001	Investigación de la librería <i>Apache Lucene</i> .	Lenin Montalvo	8	4
T-002	Levantamiento del entorno de desarrollo: <i>VS code y eclipse EE</i> .	Lenin Montalvo	4	2
T-003	Levantamiento de servidor Apache y Sails.js.	Lenin Montalvo	4	2
T-004	Diagrama de clases para el módulo recomendador.	Lenin Montalvo	4	3
T-005	Implementación de la clase de indexación con la librería <i>Apache Lucene</i> en JAVA.	Lenin Montalvo	3	2
T-006	Implementación de la clase de búsqueda con la librería <i>Apache Lucene</i> en JAVA.	Lenin Montalvo	3	2
T-007	Pruebas locales de indexación y búsqueda con datos quemados.	Lenin Montalvo	4	2
T-008	Implementación de solicitudes http en el lado del servidor para búsqueda de indexación.	Lenin Montalvo	4	4
T-009	Descarga de datos del repositorio digital <i>RKB Explorer</i> para almacenarlos e indexarlos en un repositorio local.	Lenin Montalvo	4	4
T-010	Implementación de peticiones http desde la vista para realizar pruebas.	Lenin Montalvo	4	3
T-011	Implementación de cabeceras para intercambio de recursos de origen cruzado desde el lado del servidor.	Lenin Montalvo	8	4

Tabla 29. Tareas para la tercera iteración (Continúa en la siguiente página)

Continuación de Tabla 29

T-012	Pruebas de resultados obtenidos con datos extraídos.	Lenin Montalvo	2	3
T-013	Implementación de método para obtener la puntuación de la búsqueda.	Lenin Montalvo	4	3
T-014	Pruebas de resultados con puntuación en los resultados.	Lenin Montalvo	4	3
T-015	Elaboración de prototipo para la interfaz de presentación de los resultados del servidor de recomendación	Jonathan Pachacama	4	3
T-016	Programación de funcionalidad de para obtener resultados del servidor de recomendación	Jonathan Pachacama	8	2
T-017	Implementación de paginación para resultados obtenidos.	Jonathan Pachacama	4	3
T-018	Pruebas de integración entre el sistema y el servidor	Equipo	4	3
Total, Horas			80	

Tabla 29. Tareas para la tercera iteración (Continuación)

Revisión de la tercera iteración

Para realizar los seguimientos de cada una de las iteraciones se usa una plantilla *Scrum Anexo III*.

Una vez ya definidas las tareas para la tercera iteración se procederá a realizar el seguimiento de dichas tareas. Primero se presentan en la **Tabla 30** los datos generales del proyecto:

Proyecto			
DESARROLLO DE UN SISTEMA DE RECOMENDACIÓN PARA			
Nº de sprint	Inicio	Días	Jornada
3	2-abr.-18	20	4
TAREAS		EQUIPO	FESTIVOS
TIPOS	ESTADOS		
Análisis	Pendiente	Jonathan	
Codificación	En curso	Lenin	
Prototipado	Terminada	Equipo	
Pruebas	Eliminada		
Reunión			

Tabla 30. Datos generales de la tercera iteración

En la **Tabla 31** se muestran todas las tareas con sus respectivas fechas y horas dedicadas a cada una de ellas.

PILA DEL SPRINT				
Backlog	Tarea	Tipo	Estado	Responsal
Obtener arti	Investigación de la librería Apache Lucene.	Análisis	Terminada	Lenin
	Levantamiento del entorno de desarrollo: VS code y eclipse EE.	Codificación	Terminada	Lenin
	Levantamiento de servidor Apache y Diagrama de clases para el módulo	Codificación	Terminada	Lenin
	Implementación de la clase de	Prototipado	Terminada	Lenin
	Implementación de la clase de	Codificación	Terminada	Lenin
	Pruebas locales de indexación y	Codificación	Terminada	Lenin
	implementación de solicitudes http	Pruebas	Terminada	Lenin
	Descarga de datos del repositorio	Codificación	Terminada	Lenin
	Implementación de peticiones http	Codificación	Terminada	Lenin
	Implementación de cabeceras para	Codificación	Terminada	Lenin
	Pruebas de resultados obtenidos	Pruebas	Terminada	Lenin
	Implementación de método para	Codificación	Terminada	Lenin
	Pruebas de resultados con	Pruebas	Terminada	Lenin
	Elaboración de prototipo para la	Codificación	Terminada	Jonathan
	Programación de funcionalidad de	Codificación	Terminada	Jonathan
	Implementación de paginación para resultados obtenidos.	Codificación	Terminada	Jonathan
	Pruebas de integración entre el sistema y el servidor.	Codificación	Terminada	Equipo

Tabla 31. Tareas finalizadas de la tercera iteración

En esta iteración se procede a la implementación del módulo de recomendación. La implementación es realizada en el lenguaje de programación Java y con el paradigma de la POO (Programación Orientada a Objetos), por lo que para una mejor comprensión de se implementan las clases se tiene un diagrama de clases.

Este diagrama de clases se encuentra representado en la **Figura 39**.

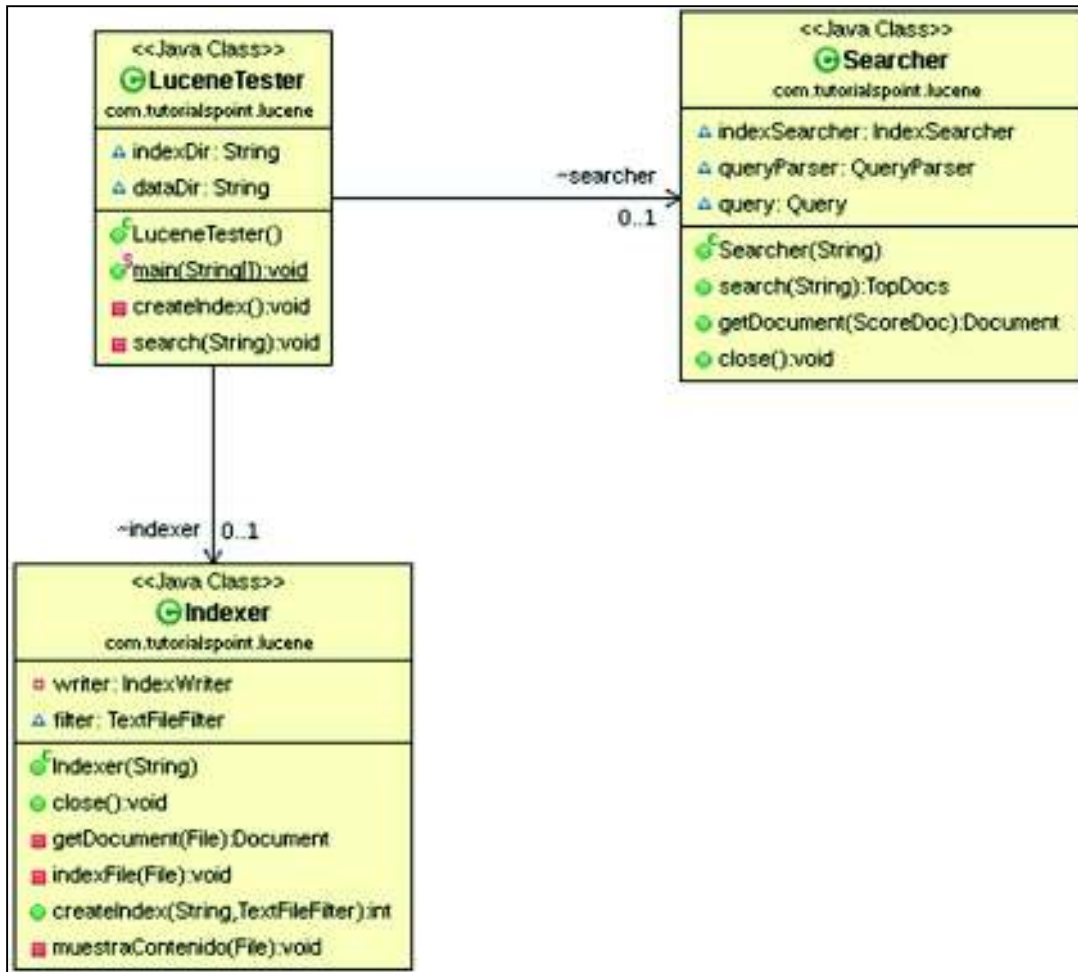


Figura 39. Diagrama de clases básico de la librería **apache lucene** en java

La implementación se la realiza en un ambiente local, por ende, las operaciones de indexación y búsqueda son realizadas en ese mismo ambiente. Para realizar este tipo de operaciones de manera remota se implementan más clases y funcionalidades lo cual se indicará más adelante.

Para la indexación se usan archivos RDF, cada archivo representa a un artículo científico. El resultado de la indexación exitosa de cuatro archivos muestra un mensaje. La **Figura 40** muestra el resultado de la indexación de archivos.

4 File indexed, time taken: 136 ms

Figura 40. Resultado de 4 archivos indexados

En la búsqueda se insertan varias palabras que contengan un título para verificar que el buscador encuentra el archivo que contiene el título con esas palabras:

Se ingresan las palabras “*Recovery Partially Decision*” del título “*Automatic Recovery Using Bounded Partially Observable Markov Decision*” que se encuentra en el tercer archivo. Se ingresaron esas palabras en el buscador y como resultado se obtiene lo que se muestra en la **Figura 41**.

```
| 1 documents found. Time :52
File: /home/lenin/Escritorio/2018A/Tesis/Recomendador/DatosRDFaRT/Data/art3.xml
```

Figura 41. Resultado de ejemplo del sistema de recomendación

Para automatizar el proceso de indexación es necesario desarrollar un programa que sea capaz de realizar todos los pasos descritos anteriormente. El programa se lo desarrolló con el lenguaje de programación *javascript* y el ambiente de ejecución *Nodejs*.

En la **Figura 42** se muestra un extracto de la principal parte del código de indexación

```
var options = {
  url: "http://localhost:8080/JavaAPI/api/articulos/indexIEEE",
  method: "POST",
  headers:{
    "Content-Type":"application/json"
  },
  body:JSON.stringify(arrayArticulos)
}

request(options, callback)
```

Figura 42. Extracción de código para indexación

Los principales elementos se encuentran:

- **Url:** Dirección a la cual se envía la petición.
- **Método (*method*):** El método HTTP, en este caso es POST.
- **Cabeceras (*headers*):** En la cabecera se envía el tipo de formato de los datos, en este caso es *JSON*.
- **Cuerpo de la solicitud (*body*):** Es donde se envían los parámetros de la solicitud POST, en este caso es el *array* con los datos de los artículos.
- **Solicitud (*request*):** Método en el cual se envían los parámetros anteriores (*options*) y una función (*callback*) la cual maneja los resultados devueltos y controla los errores.

Para extraer los datos de los repositorios *RKB* necesita una consulta *sparql* como la que se indica en la **Figura 43**:

```
SELECT * WHERE { ?articulo akt:has-title ?titulo.  
                  ?articulo akt:extn:has-abstract ?abstracto }
```

Figura 43. Consulta sparql para extraer los datos a indexar

El siguiente paso es desarrollar el prototipo de la interfaz que se usará para buscar los artículos más relevantes de una búsqueda

A continuación, se presenta las diferentes interfaces con las cuales el usuario interactúa y prueba en esta iteración:

El artículo que se ha guardado previamente es el que se presenta en la **Figura 44**.



Figura 44. Artículo guardado con éxito

Una vez que selecciona el artículo se procede a seleccionar el repositorio del cual se quiere obtener recomendaciones.

En las **Figuras 45, 46 y 47** se obtienen las diferentes listas de los diferentes repositorios:

IEEE ACM DBLP

IEEE
ieee.rkbexplorer.com

Más Información

resultados **656**

TÍTULOS Mostrar: **artículos**

1	Design Verification by Using Built-In Checkers	Metadatos
2	Efficient Algorithmic Circuit Verification Using Indexed BDDs	Metadatos
3	Improving computer security using extended static checking	Metadatos

Figura 45. Lista de recomendación de la IEEE

ACM
acm.rkbexplorer.com

Más Información

resultados **1422**

TÍTULOS Mostrar: **artículos**

1	Verification of user identity via keyboard characteristics	Metadatos
2	Identity verification and biometrics	Metadatos
3	Gait transitions	Metadatos
4	Using Nurpl for the verification and synthesis of hardware	Metadatos

Figura 46. Lista de recomendaciones de la ACM

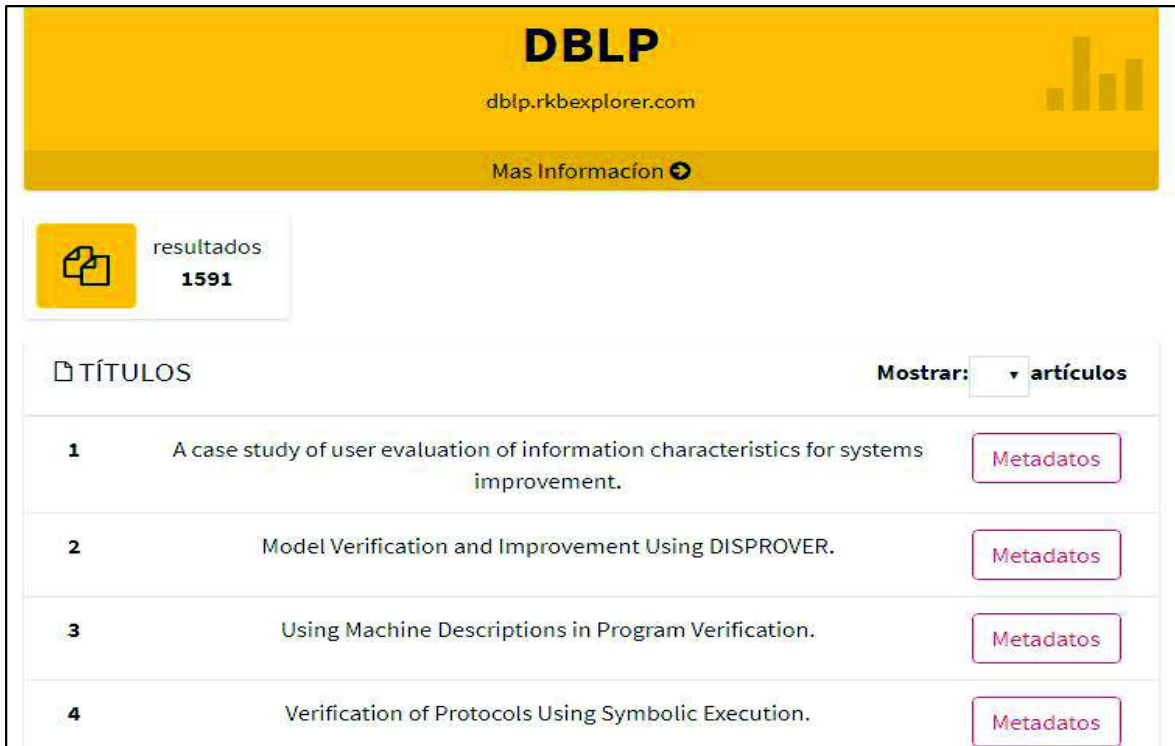


Figura 47. Lista de recomendaciones de la DBLP

Retrospectiva de la tercera iteración

A continuación, en la **Figura 48**, se visualizará el esfuerzo realizado durante todo el mes.



Figura 48. Esfuerzo desarrollado para la tercera iteración

En la **Figura 49** se expresan el avance de las tareas pendientes y los días ocupados en cada una de ellas.



Figura 49. Tareas pendientes para la tercera iteración.

2.3.4. Cuarta iteración

Planificación de la cuarta iteración

En esta iteración se hará la integración del recomendador con el sistema de citas construido previamente con *wikindx* y DOAJ. Los requerimientos para esta iteración se presentan en la **Tabla 32**:

ID requerimientos	Nombre	Nro. De Prioridad	Descripción	Estimación
RQ007	Ampliar resultados	4	El investigador podrá expandir la información de un artículo seleccionado de la lista retornada del módulo recomendador	4
RQ008	Manipular resultados	3	El investigador podrá manipular y guardar los resultados de las recomendaciones obtenidas.	4
RQ009	Redireccionar artículo	3	El investigador dirigirse a la dirección web del artículo recomendado	2

Tabla 32. Requerimientos para la cuarta iteración

Objetivo de la cuarta iteración

El objetivo para la cuarta iteración es expandir a información de los artículos guardados por medio del módulo de recomendación. Además de poder manipular los datos y redirigirse a las direcciones web de cada recurso recomendó.

Lista de tareas para la cuarta iteración.

Las tareas por realizar para el cumplimiento de los anteriores requisitos en el programa se presentan en la **Tabla 33**:

ID Tareas	Nombre	Responsable	Tiempo Estimado	Estimación
T-001	Elaboración de consultas SPARQL para la extracción de metadatos a partir de un título	Equipo	8	3
T-002	Elaboración del prototipo para la interfaz para la presentación de metadatos del artículo.	Equipo	4	2
T-003	Programación para presentar los metadatos en la interfaz.	Equipo	4	3
T-004	Pruebas para la presentación de los metadatos en la interfaz.	Equipo	12	3
T-005	Desarrollo de servidor proxy para retorno de cabeceras para intercambio de recursos de origen cruzado desde el lado del servidor <i>RKB Explorer</i> hacia el cliente	Equipo	8	4
T-006	Prueba de funcionamiento del servidor <i>proxy</i> .	Equipo	8	4
T-007	Programación de guardado de metadatos de cada recurso a la base de datos <i>wikindx</i> .	Equipo	8	4
T-008	Pruebas para el funcionamiento de edición y guardado del artículo científico	Equipo	12	3
T-009	Programación de funcionalidad para redirigirse a la página web de un artículo.	Equipo	4	2
T-010	Implementación de botón de full-text (redirección a la dirección web del artículo).	Equipo	4	2
T-011	Prueba de funcionamiento de redirección al sitio web del artículo.	Equipo	8	3
Total, Horas			80	

Tabla 33. Tareas para la cuarta iteración

Revisión de la cuarta iteración

Para realizar los seguimientos de cada una de las iteraciones se usa una plantilla *scrum* **Anexo IV**.

A continuación, En la **Tabla 34** se presentan los datos generales de la cuarta iteración:

Proyecto			
Nombre del proyecto			
Nº de sprint	Inicio	Días	Jornada
4	1-may.-18	20	4
TAREAS		EQUIPO	FESTIVOS
TIPOS	ESTADOS		
Análisis	Pendiente	Jonathan	
Codificación	En curso	Lenin	
Prototipado	Terminada	Equipo	
Pruebas	Eliminada		
Reunión			

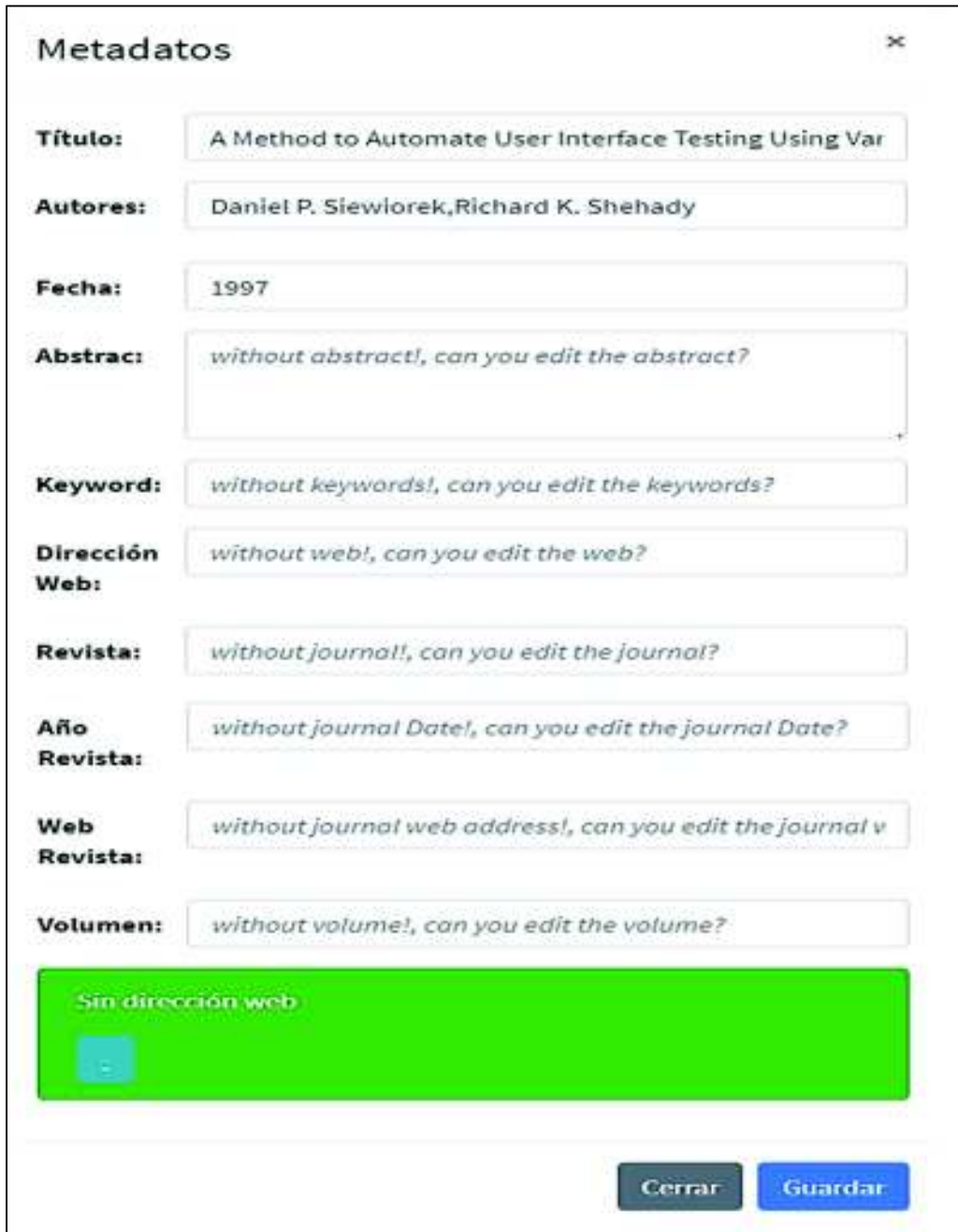
Tabla 34. Datos generales de la cuarta iteración

En la **Tabla 35** se presentan las tareas para la cuarta iteración:

SPRINT		INICIO	DURACIÓN	
4		1-may.-18	20	
Tareas pendientes Horas de trabajo pendientes				
PILA DEL SPRINT				
Backlog II	Tarea	Tipo	Estado	Responsa
Ampliar Resultados	Elaboración de consultas SPARQL para la extracción de metadatos a partir de un título	Análisis	Terminada	Equipo
	Elaboración del prototipo para la interfaz para la presentación de metadatos del artículo.	Codificación	Terminada	Equipo
	Programación para presentar los metadatos en la interfaz.	Codificación	Terminada	Equipo
	Pruebas para la presentación de los metadatos en la interfaz.	Pruebas	Terminada	Equipo
	Desarrollo de servidor proxy para retorno de cabeceras para intercambio de recursos de origen cruzado desde el lado del servidor RKBExplorer hacia el cliente	Codificación	Terminada	Equipo
	Prueba de funcionamiento del servidor proxy.	Pruebas	Terminada	Equipo
Manipulación de resultados	Programación de guardado de metadatos de cada recurso a la base de datos wikindx.	Codificación	Terminada	Equipo
Redireccionar artículo	Pruebas para el funcionamiento de edición y guardado del artículo científico	Pruebas	Terminada	Equipo
	Programación de funcionalidad para redirigirse a la página web de un artículo.	Codificación	Terminada	Equipo
	Implementación de botón de full-text (redicción a la dirección web del artículo).	Codificación	Terminada	Equipo
	Prueba de funcionamiento de redirección al sitio web del artículo.	Pruebas	Terminada	Equipo

Tabla 35. Seguimiento de tareas de la cuarta iteración

Para la presentación de los metadatos se usará una ventana, más conocida como *modal*. Algunos de los metadatos extraídos de los repositorios se presentarán vacíos, esto debido a que los metadatos no existen en el artículo. Por ello es necesario que el usuario investigador pueda manipular estos metadatos. A continuación, se presenta la interfaz usada para la presentación de los metadatos extraídos en la **Figura 50**.



The image shows a modal window titled "Metadatos" with a close button (X) in the top right corner. The modal contains several input fields for metadata, each with a label on the left and a text box on the right. The fields and their contents are:

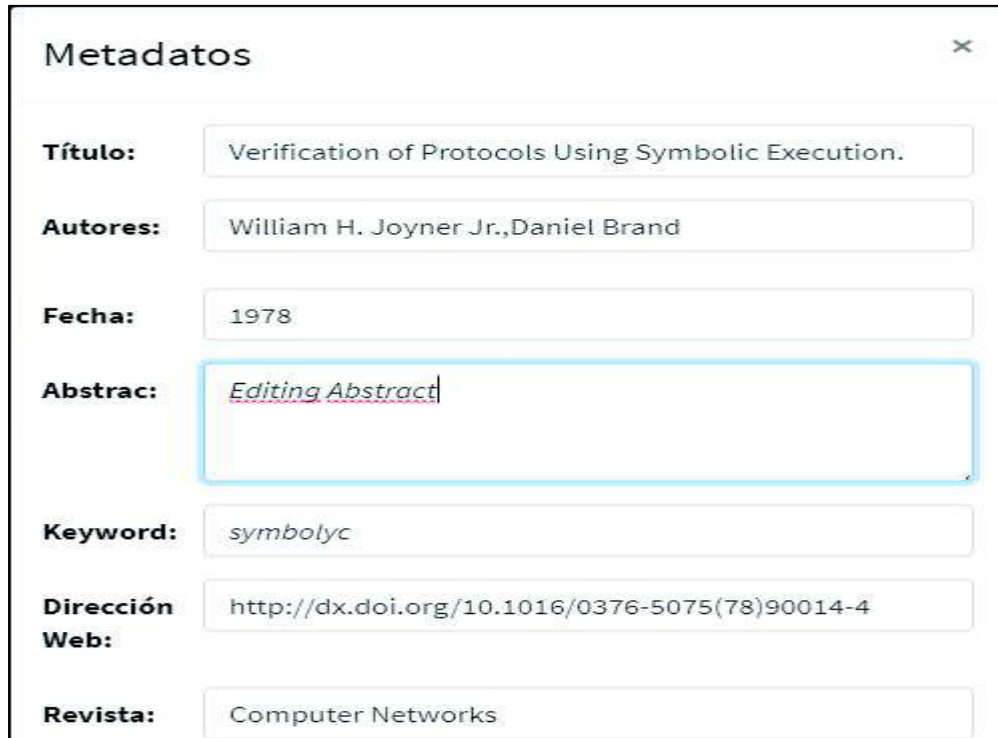
- Título:** A Method to Automate User Interface Testing Using Var
- Autores:** Daniel P. Siewlorek, Richard K. Shehady
- Fecha:** 1997
- Abstrac:** without abstract!, can you edit the abstract?
- Keyword:** without keywords!, can you edit the keywords?
- Dirección Web:** without web!, can you edit the web?
- Revista:** without journal!, can you edit the journal?
- Año Revista:** without journal Date!, can you edit the journal Date?
- Web Revista:** without journal web address!, can you edit the journal v
- Volumen:** without volume!, can you edit the volume?

At the bottom of the modal, there is a red banner with the text "Sin dirección web" and a small red icon. Below the banner are two buttons: "Cerrar" (Close) and "Guardar" (Save).

Figura 50. Modal de edición de metadatos para IEEE, DBLP y ACM.

Manipulación de resultados

Cuando se guarde el artículo al final saldrá un mensaje de éxito el cual indica que el artículo ha sido guardado exitosamente, tal como se indica en la **Figura 51**.



The image shows a web form titled "Metadatos" with a close button (X) in the top right corner. The form contains several input fields with the following data:

- Título:** Verification of Protocols Using Symbolic Execution.
- Autores:** William H. Joyner Jr., Daniel Brand
- Fecha:** 1978
- Abstrac:** Editing Abstract
- Keyword:** symbolyc
- Dirección Web:** [http://dx.doi.org/10.1016/0376-5075\(78\)90014-4](http://dx.doi.org/10.1016/0376-5075(78)90014-4)
- Revista:** Computer Networks

Figura 51. Edición de metadatos de un artículo científico

Para guardar al artículo modificado se presiona en "Guardar" y saldrá un mensaje de éxito, tal como se muestra en la **Figura 52**.



The image shows a success message with a green header and a yellow sidebar. The main content area has a green background with white text.

Artículo guardado con éxito!!!

Bien hecho!
Este es el artículo que se guardó.

Verification of Protocols Using Symbolic Execution.

Recommender module

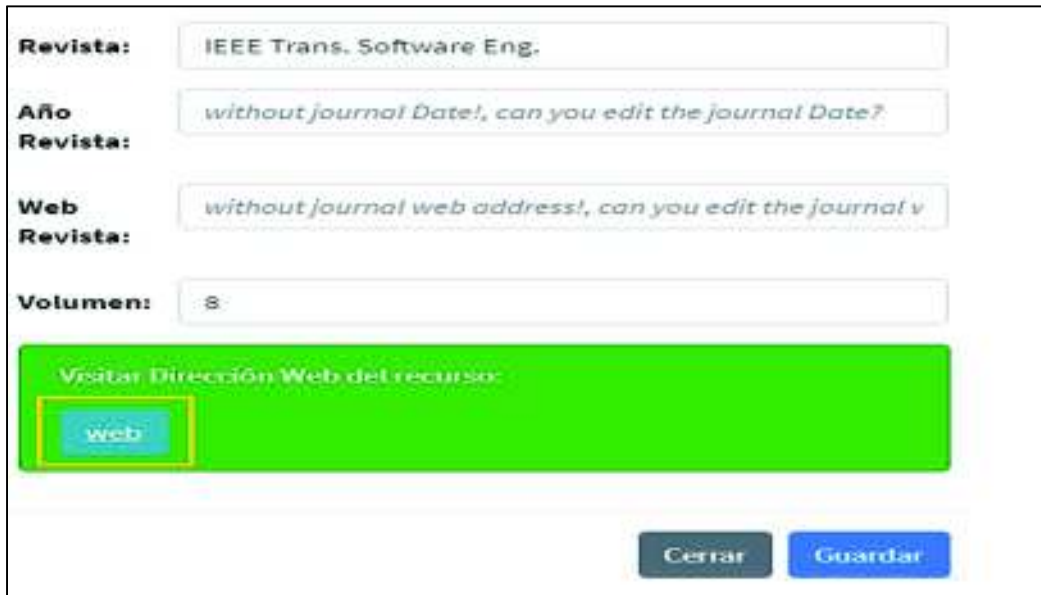
Ingresar al módulo de recomendación para encontrar artículos relacionados con su búsqueda.

Ingresar

Figura 52. Mensaje de artículo guardado

Redireccionar artículo

Para redireccionar a la página web oficial del artículo el usuario se debe presionar el botón “web” en el modal, donde se visualizan y editan los metadatos del artículo científico. En la **Figura 53** se señala al botón “web”.



Revista: IEEE Trans. Software Eng.

Año Revista: without journal Date!, can you edit the journal Date?

Web Revista: without journal web address!, can you edit the journal v

Volumen: 8

Visitar Dirección Web del recurso:

web

Cerrar Guardar

Figura 53. Artículo de la DBLP

En la **Figura 54** se muestra a la página oficial de un artículo de la IEEE, que en este caso se encontraba en el repositorio de datos de la DBLP.



Home Current Issue Past Issues Preprints Write for Us About

Download Export Citation

Home / Journals / IEEE Transactions on Software Engineering / 1982-01

Formal Program Verification Using Symbolic Execution

January 1982, pp. 43-52, vol. 8
DOI Bookmark: 10.1109/TSE.1982.234773

Keywords
Verification Conditions, Control Constructs, Program Proving, Program Verification, Rules Of Inference, Side Effects, Symbolic Execution

Authors
G.W. Ernst
R.B. Dannenberg, Department of Computer Science, Carnegie-Mellon University

Abstract
Symbolic execution provides a mechanism for formally proving programs correct. A notation is introduced which allows a concise presentation of rules of inference based on symbolic execution. Using this notation, rules of inference are developed to handle a number of language features, including loops and procedures with multiple exits. An attribute grammar is

Figura 54. Página oficial del artículo en la IEEE

Para la redirección a una página oficial de la ACM es de igual manera. En la **Figura 55** se tiene al título del artículo al cuál deseamos ir a la página oficial, se presiona ahí y en el modal presionar el botón “web”. En la **Figura 56** se muestra la página oficial del artículo.

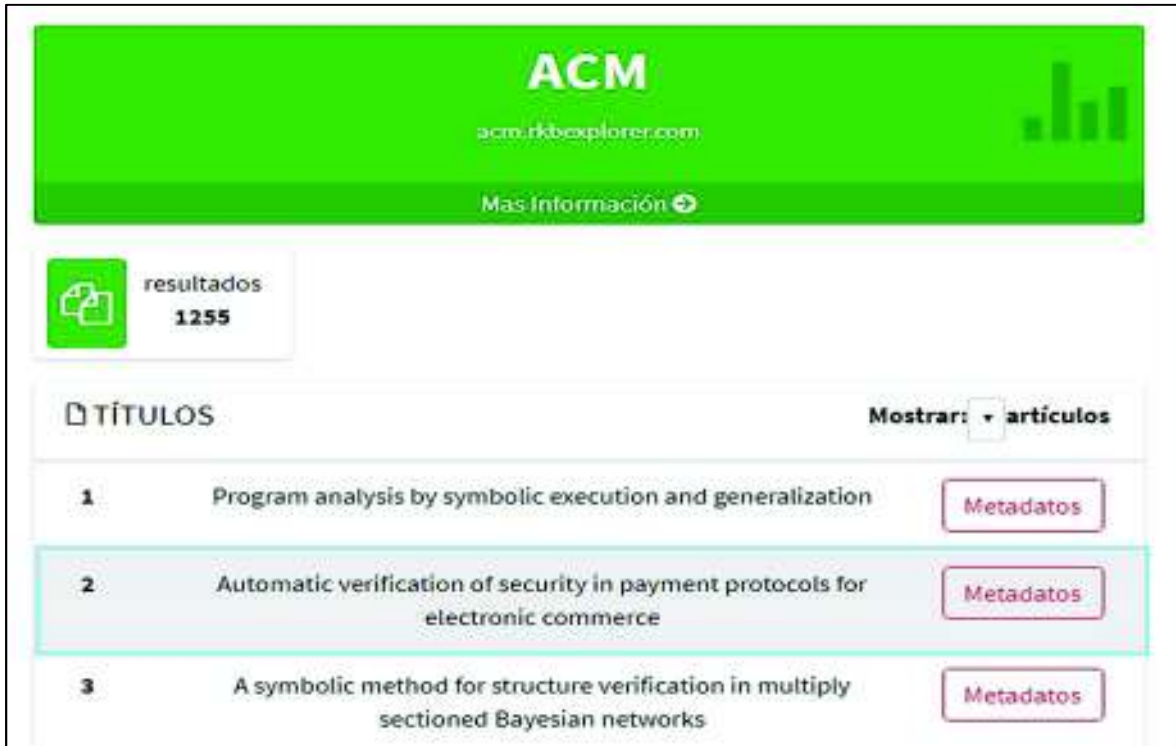


Figura 55. Artículo de la ACM



Figura 56. Página oficial del artículo en la ACM

Para la redirección a una página oficial de la IEEE es de igual manera. En la **Figura 57** se tiene al título del artículo al cuál deseamos ir a la página oficial, se presiona ahí y en el modal presionar el botón “web”. En la **Figura 58** se muestra la página oficial del artículo.



Figura 57. Metadatos de un artículo de la IEEE



Figura 58. Página oficial del artículo

Retrospectiva de la cuarta iteración

A continuación, se muestra en la **Figura 59** el esfuerzo realizado durante todo el mes de mayo.



Figura 59. Esfuerzo realizado para la cuarta iteración

En la **Figura 60** se expresa el avance de las tareas pendientes y los días ocupados en cada una de ellas.

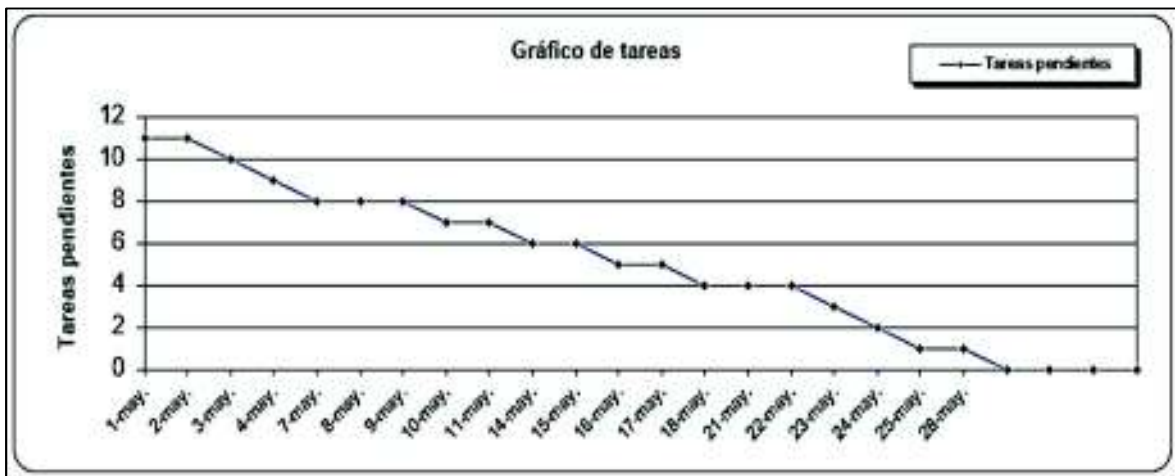


Figura 60. Tareas pendientes para la cuarta iteración.

2.4. Producto final

Arquitectura del *framework*

Patrón de Desarrollo MVC (Modelo-Vista-Controlador)

Por ser una aplicación web, el patrón el cual se ha optado para el desarrollo es el Modelo-Vista-Controlador (MVC).

Este patrón de desarrollo se encarga de separar a una aplicación en tres componentes distintos: el modelo, la vista y el controlador.

- **Modelo:** Contiene la estructura de los datos de los objetos de la aplicación, a menudo recuperan y almacenan el estado del objeto en una base de datos [28].
- **Vista:** La vista presenta los datos del modelo en una interfaz gráfica [28]. En otras palabras, se podría decir que el usuario interactúa con los datos almacenados en una base de datos mediante una vista, modificando, agregando, eliminando, actualizando, etc.
- **Controlador:** Este es el componente que se encarga de orquestar las funciones entre la vista y el modelo [28]. Escucha las acciones que se desencadenan en la vista y ejecuta las respectivas acciones hacia el modelo. En la mayoría de los casos esta acción se refleja en la vista.

Al aplicar este patrón en el desarrollo de la aplicación permitirá que las operaciones lógicas no interfieran con la presentación de los resultados en la pantalla. Esto permite una mayor flexibilidad a la hora del desarrollo del producto y al hacer mantenimiento al código fuente del programa.

En la **Figura 61** se muestra la ilustración de un diagrama MVC.

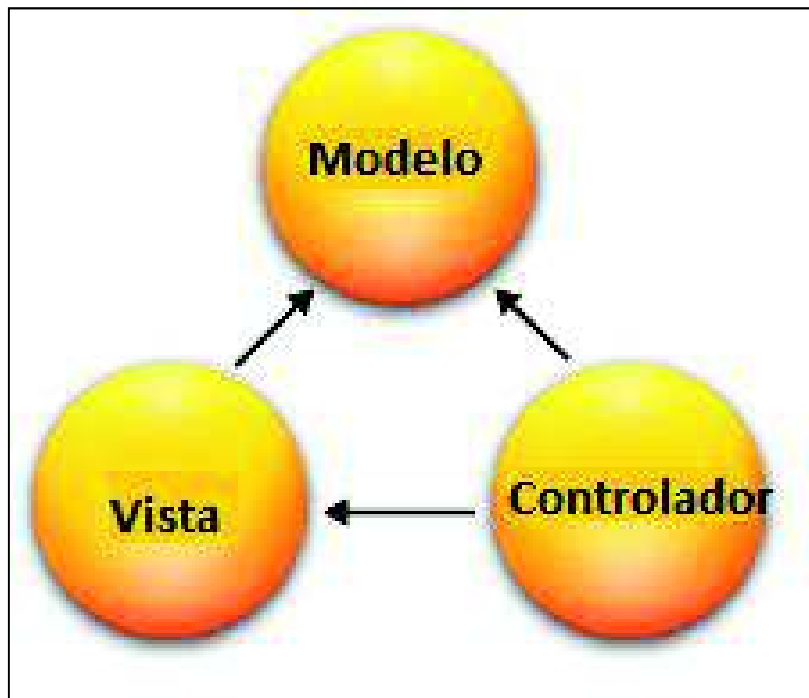


Figura 61. Modelo vista controlador [28].

En la **Figura 62** se presenta la arquitectura MVC para el sistema de recomendación de datos enlazados.

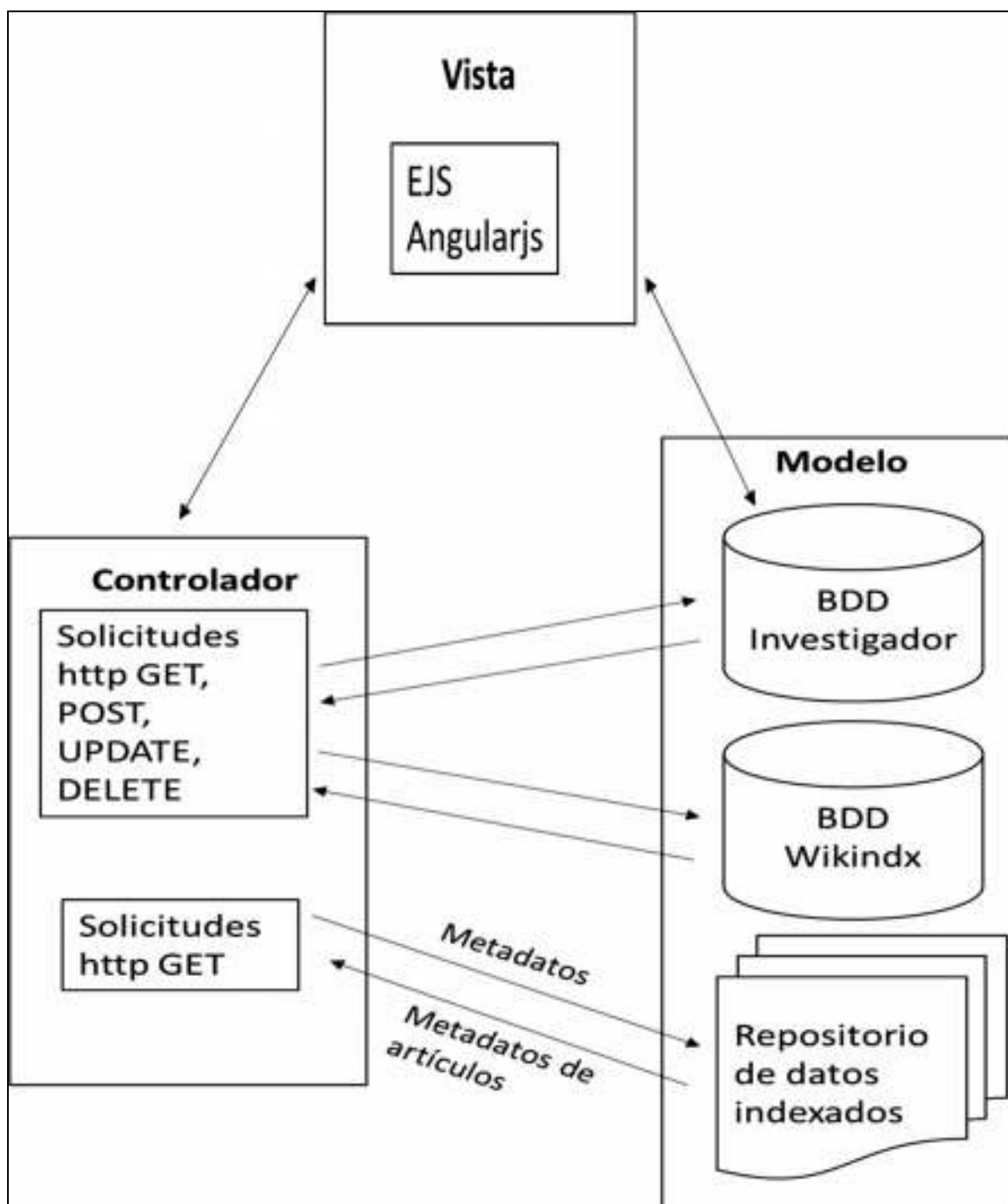


Figura 62. Arquitectura modelo vista controlador de la aplicación web

El diseño de las bases de datos se muestra en los **Figuras 63, 64, 65 y 66** donde se detalla el modelo conceptual y físico de cada base conceptualizando así las entidades y relaciones con las que interactúan.

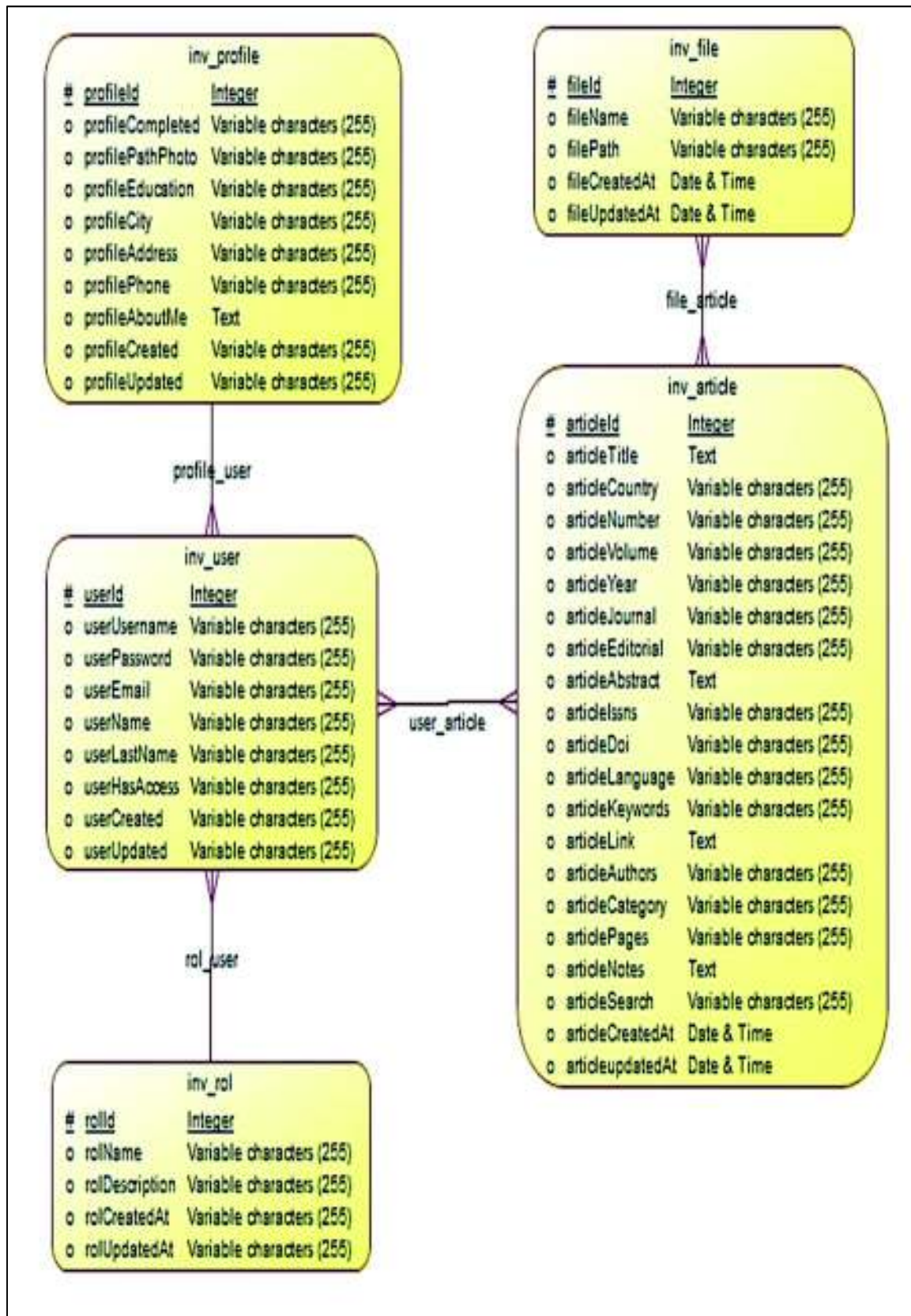


Figura 63. Modelo conceptual de la base de datos investigador utilizada

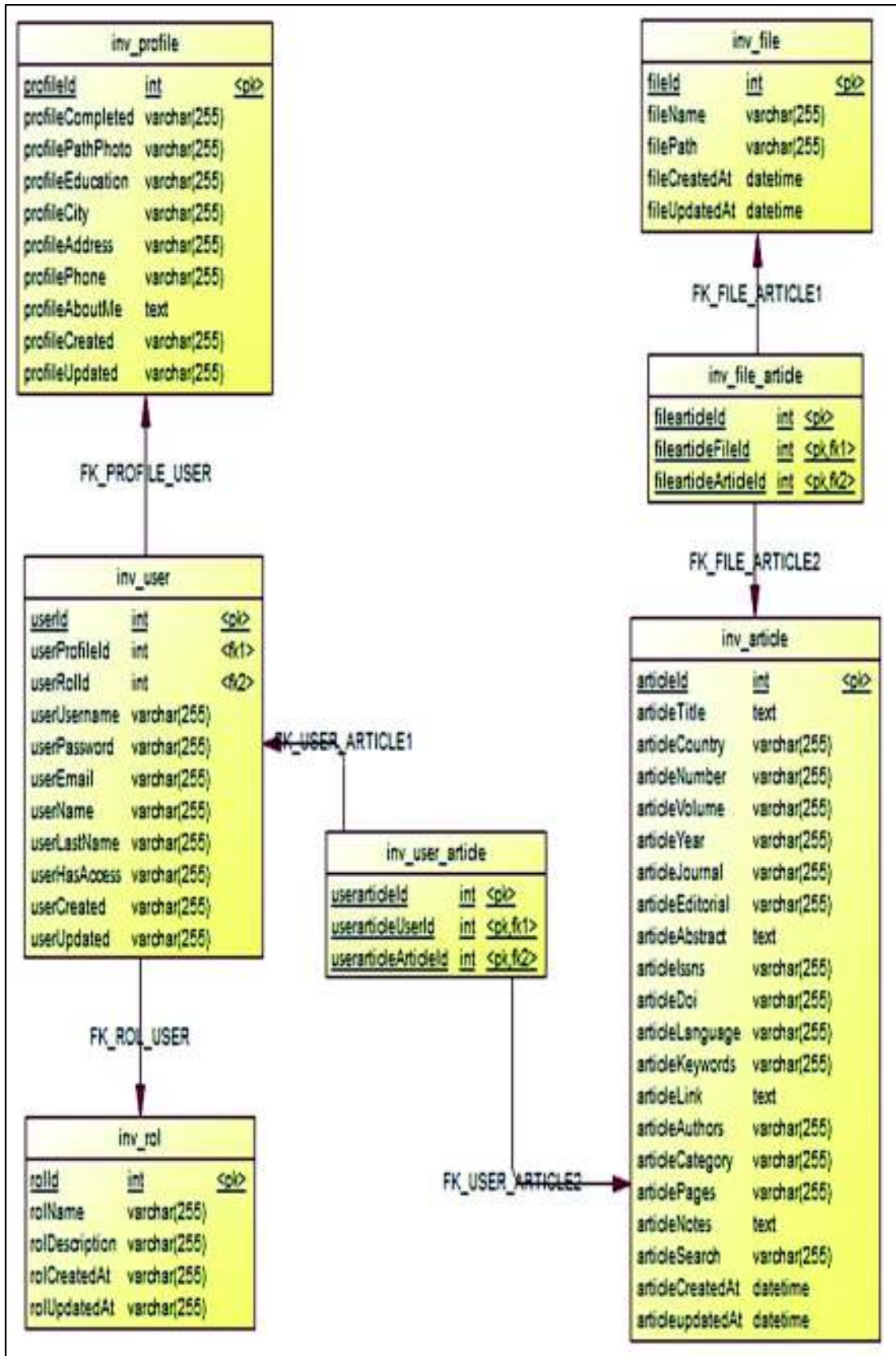


Figura 64. Modelo físico de la base de datos investigador utilizada

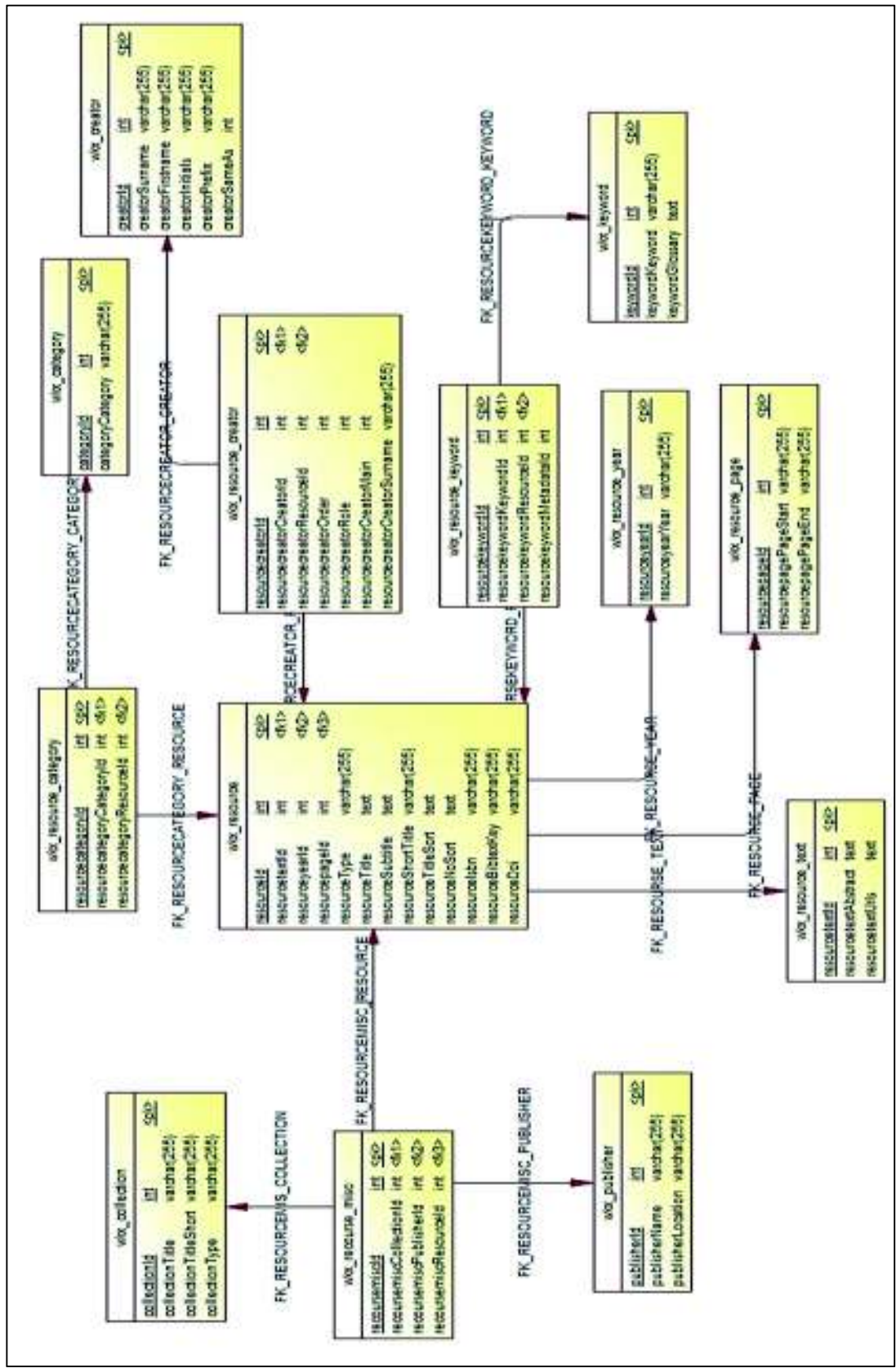


Figura 66. Modelo físico de la base de datos wikindx5 utilizada

Al haber finalizado todas las iteraciones planteadas, el sistema final consta de:

- Un módulo de registro para que el usuario pueda registrarse.
- Un módulo de inicio de sesión (*Loguin*) para que el usuario ingrese cuando ya se encuentra registrado.
- Un módulo para crear artículos y guardarlos.
- Un módulo de búsqueda.
- Un módulo biblioteca en el cual se pueden guardar los artículos buscados y creados.
- Un módulo de recomendación para encontrar artículos recomendados de la web semántica.

A continuación, se detalla a breves rasgos el funcionamiento de cada módulo:

Módulo de registro y sesión:

- Si el usuario usa por primera vez el sistema de recomendación, debe de registrarse poniendo sus datos como: nombre, apellido, email, usuario y password.
- Cuando el usuario ya se registró puede usar sus credenciales en la pantalla de inicio de sesión.
- Una vez que el usuario ya ingresó al sistema podrá hacer uso de todos los módulos.

En la **Figura 67** se muestra al esquema de registro de usuario.

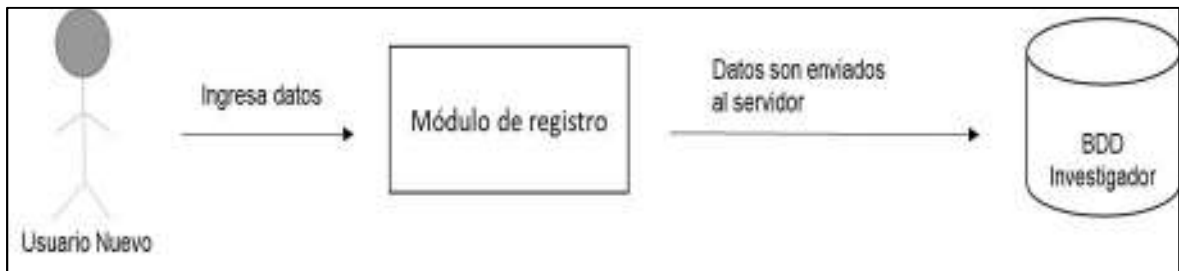


Figura 67. Registro de usuario

En la **Figura 68** se muestra el esquema de inicio de sesión de un usuario



Figura 68. Inicio de sesión

Módulo de perfil:

- En este módulo el usuario podrá subir una foto de perfil y modificar todos sus datos.

A continuación, se presenta los módulos de búsqueda, biblioteca y de recomendación funcionando de manera conjunta y se lo ilustra en la **Figura 69**.

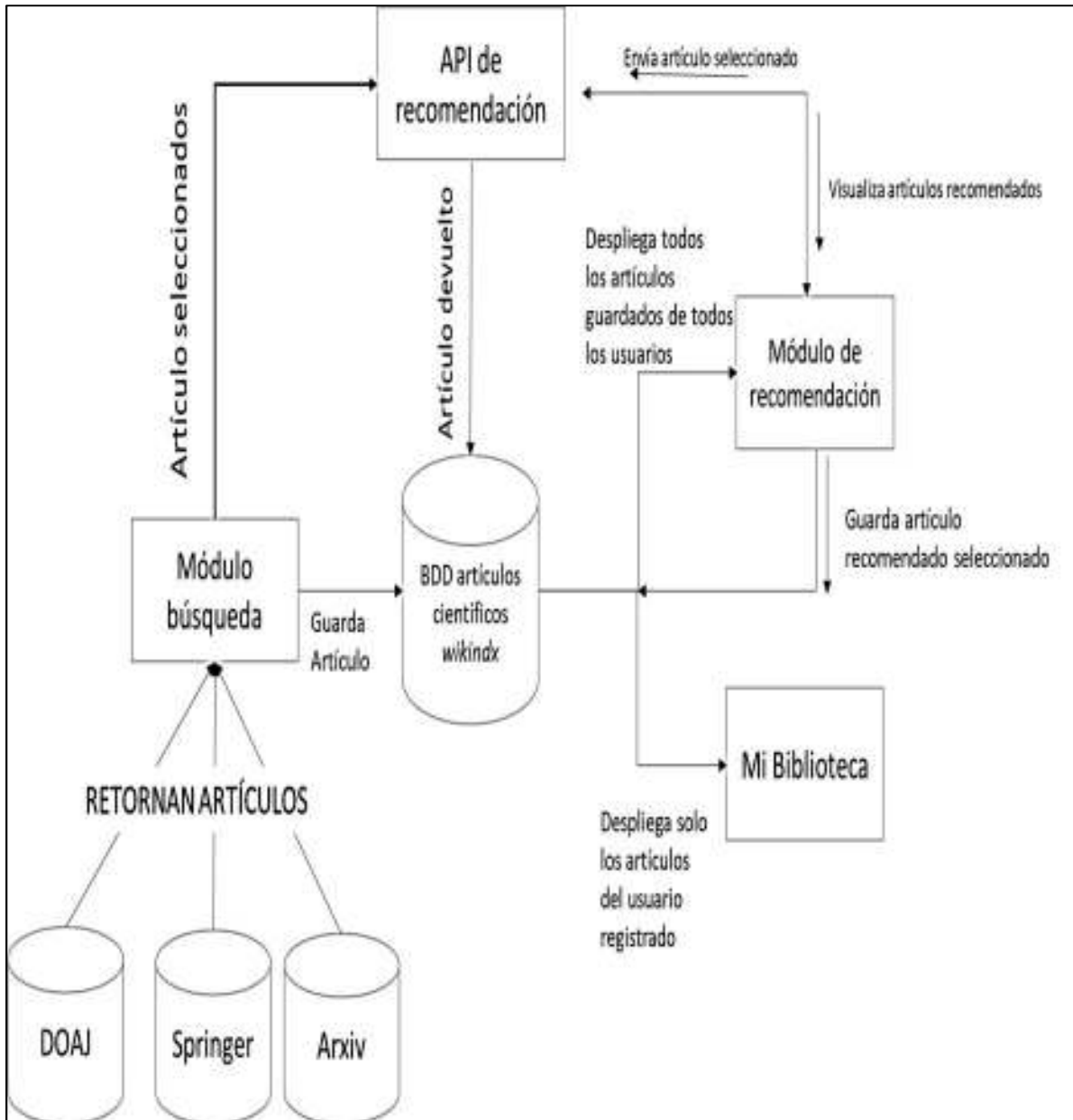


Figura 69. Esquema de funcionamiento de módulos

Módulo de búsqueda:

- El usuario ingresa las palabras de su investigación en el módulo de búsqueda.
- Este módulo despliega una variedad de artículos y el que sea de su preferencia puede guardarlo a la base de datos de *wikindx*.

- Si el usuario desea encontrar recomendaciones dadas por la IEEE, ACM y DBLP puede seleccionar alguna de las opciones que se le despliegan en la parte de abajo.
- Si algún artículo recomendado es de su interés, lo selecciona, si desea lo edita y lo guarda en la base de datos de *wikindx*.

Módulo de biblioteca:

- El usuario puede observar todos sus artículos guardados en el módulo de biblioteca.
- Si el usuario desea modificar alguno de esos artículos, lo selecciona, edita y guarda lo editado.
- En este módulo tan solo se observan lo artículos guardados por el usuario quién está iniciado sesión en el sistema.
- El usuario también tiene la posibilidad de ingresar datos de nuevos artículos y guardarlos en la base de datos de *wikindx* y de visualizarlos en su biblioteca.

Módulo de recomendación:

- Este módulo es de índole colaborativo, funciona como el módulo “mi biblioteca” con la diferencia que se van a poder visualizar todos los artículos que se han ingresado por todos los usuarios registrados en el sistema.
- El usuario tiene la posibilidad de editar los datos del artículo, visualizar la cita bibliográfica en *wikindx*, obtener recomendaciones de la IEEE, ACM y DBLP.

Para que el módulo de recomendación sea usado como un servicio se lo deberá implementar de tal forma que la aplicación pueda receptor peticiones HTTP: POST y GET. Lo que significa que actúa como el servidor de recomendación de la aplicación *web* de búsqueda de artículos científicos.

La **Figura 70** muestra el esquema UML del servicio.

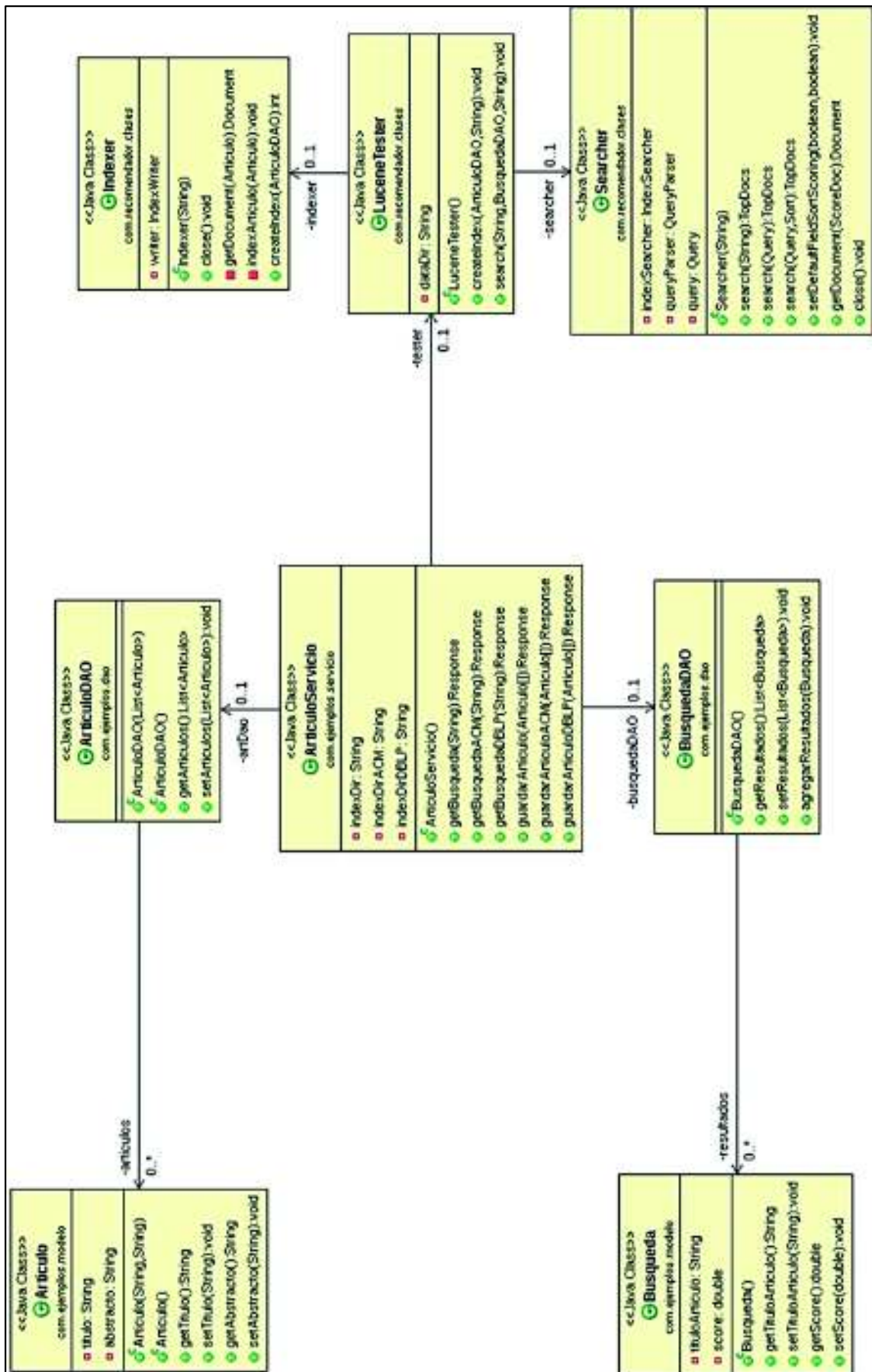


Figura 70. Diagrama de clases para el servidor de recomendación.

Proceso de indexación y búsqueda del producto final

Para la indexación y búsqueda de los artículos se realiza el proceso ilustrado en la **Figura 71**.

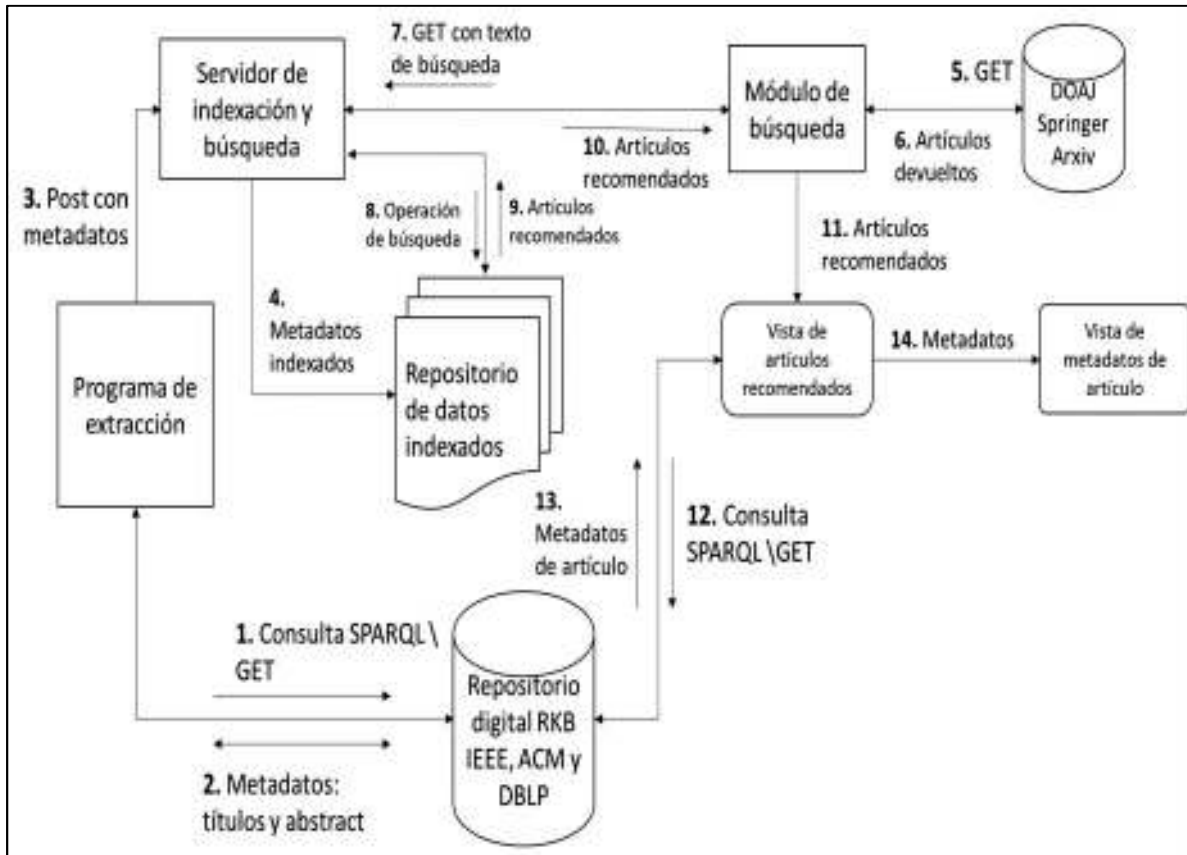


Figura 71. Proceso de indexación y búsqueda

A continuación, se explica el proceso de la **Figura 71**:

Indexación

1. El primer paso para la indexación es realizar una consulta SPARQL al repositorio digital de la RKB. Esta consulta se codifica en la *url* a la cual se realiza la solicitud HTTP GET. La solicitud GET con la consulta **SPARQL** se realiza desde el **programa de extracción**.
2. Se devuelven los metadatos de los artículos almacenados en el repositorio digital de la RKB y el programa de extracción realizará el respectivo tratamiento a los metadatos. Estos metadatos son el título y abstract.
3. Una vez hecho el tratamiento a los datos devueltos del repositorio RKB se envía con la operación POST se envía un JSON con los datos al **servidor de indexación y búsqueda**.

4. El servidor de **indexación y búsqueda** realiza el respectivo tratamiento al JSON de datos para su indexación y que sean almacenados en el **repositorio de datos indexados**.

Hasta este punto se ha completado el proceso de indexación, este proceso se lo realiza una vez o las veces en las que se requiera actualizar el repositorio. A continuación, se explica el proceso de búsqueda, en este proceso interactúan tanto el servidor de indexación y búsqueda como el sistema de recomendación de datos enlazados:

5. En el **módulo de búsqueda** del sistema de recomendación se extraen artículos de los repositorios digitales DOAJ, Arxiv o Springer enviando una solicitud GET a uno de los repositorios.
6. Los artículos son devueltos en formato JSON al módulo de búsqueda del sistema de recomendación.
7. Para obtener las recomendaciones de un artículo que se ha seleccionado previamente, se envía una solicitud GET al **servidor de indexación y búsqueda** con el texto de búsqueda, en este caso el texto viene a ser el título del artículo que se ha seleccionado previamente.
8. El **servidor de indexación y búsqueda** se encarga de realizar el proceso de búsqueda en el **repositorio de datos indexados**.
9. El **servidor de indexación y búsqueda** de extraer los artículos que tengan los datos que sean similares al texto de búsqueda ingresado anteriormente.
10. El **servidor de indexación y búsqueda** devuelve los datos de los artículos recomendados, los datos que devuelva el servidor son el título y una puntuación que califica a la similaridad con el texto de búsqueda ingresado.
11. Se presentan los artículos recomendados a la vista del usuario en forma de lista. El usuario se encarga de elegir a un artículo.
12. Cuando el usuario ha elegido el artículo que le interesa se envía una consulta SPARQL en una solicitud GET a la interfaz de consulta del repositorio digital de la RKB.
13. La interfaz de consulta de la RKB procesa la consulta **SPARQL** y devuelve los metadatos para el artículo que el usuario ha seleccionado previamente.
14. Los metadatos extraídos del repositorio digital de la RKB para el artículo seleccionado se presentan en una vista.

Servidor de recomendación:

Es el servidor donde se encuentra el motor de recomendación, las funciones principales de este motor son: la indexación y la búsqueda.

Cada función tiene sus propias funciones HTTP como *post* y *get*, así como también separadas para para los diferentes repositorios de IEEE, ACM y DBLP. Estas funciones son llamadas con una URL única, las *ur/s* definidas para la búsqueda de indexación son:

- **Búsqueda:** `http:// [nombre de repositorio]/?Búsqueda= [texto de búsqueda]`
- **Indexación:** `http:// [nombre de repositorio]/?Búsqueda= [texto de búsqueda]`

Dominio: Es la dirección del servidor donde se encuentra alojado el motor de recomendación.

Nombre de repositorio: Es el nombre del conjunto de datos del cual se desea extraer la información, esto puede ser: IEEE, DBLP y ACM.

Texto de búsqueda: Es el texto que el usuario ingresa y para el cuál se van a dirigir las recomendaciones.

Para probar la API de recomendación, se usan peticiones http a las url desde el software *postman*, como se muestra en la **Figura 72**:

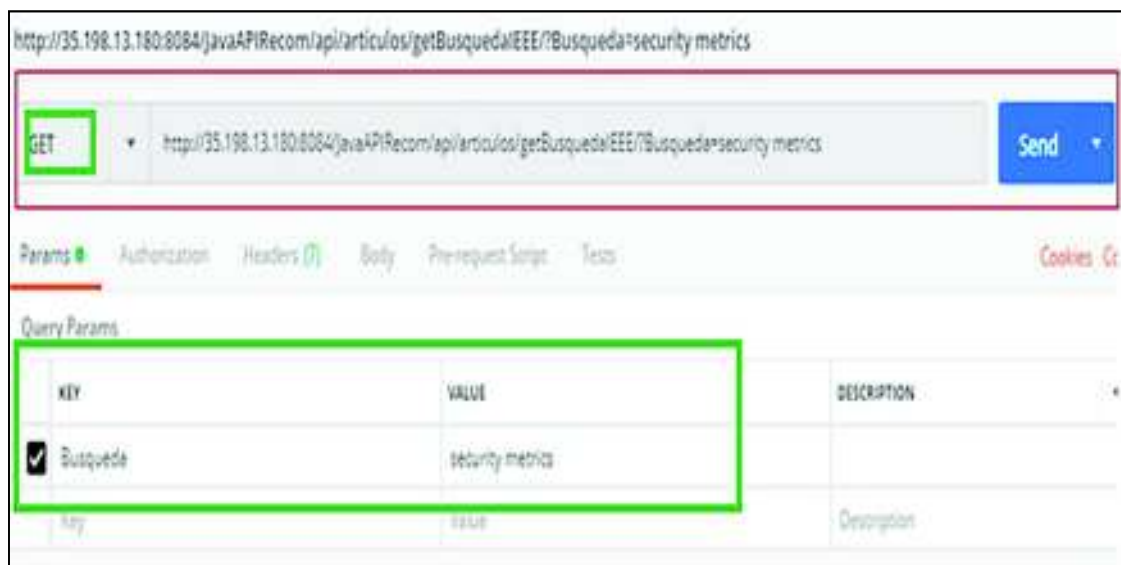


Figura 72. Método GET al servidor de recomendación

En la **Figura 76** se puede apreciar:

- **URL:** `http://35.198.13.180:8084/JavaAPIRecom/api/articulos/getBusqueda/IEEE/?Búsqueda=security metrics`
- **Método:** GET
- **Texto de búsqueda:** *Security metrics*.

Algunos de los resultados de la petición GET se muestran en la **Figura 73**:

```

{
  "score": 1.0043706893920898,
  "tituloArticulo": "Assurance Cases for Security: The Metrics Challenge"
},
{
  "score": 0.8793631196022034,
  "tituloArticulo": "The Use of Software Metrics to Assess Software Production Methods"
},
{
  "score": 0.6787670850753784,
  "tituloArticulo": "Formal development of an embedded verifier for Java Card byte code"
},
{
  "score": 0.49157893657684326,
  "tituloArticulo": "Toward acceptable metrics of authentication"
},
{
  "score": 0.409649133682251,
  "tituloArticulo": "Hierarchical computation of interval availability and related metrics"
}
}

```

Figura 73. Resultados de una solicitud GET a la API de recomendación

Sistema integrado

Cabe recalcar que el sistema final es una integración con el “Sistema de Gestión de Referencias” previamente desarrollado [26], en el **Figura 74** se ilustra esta unión.

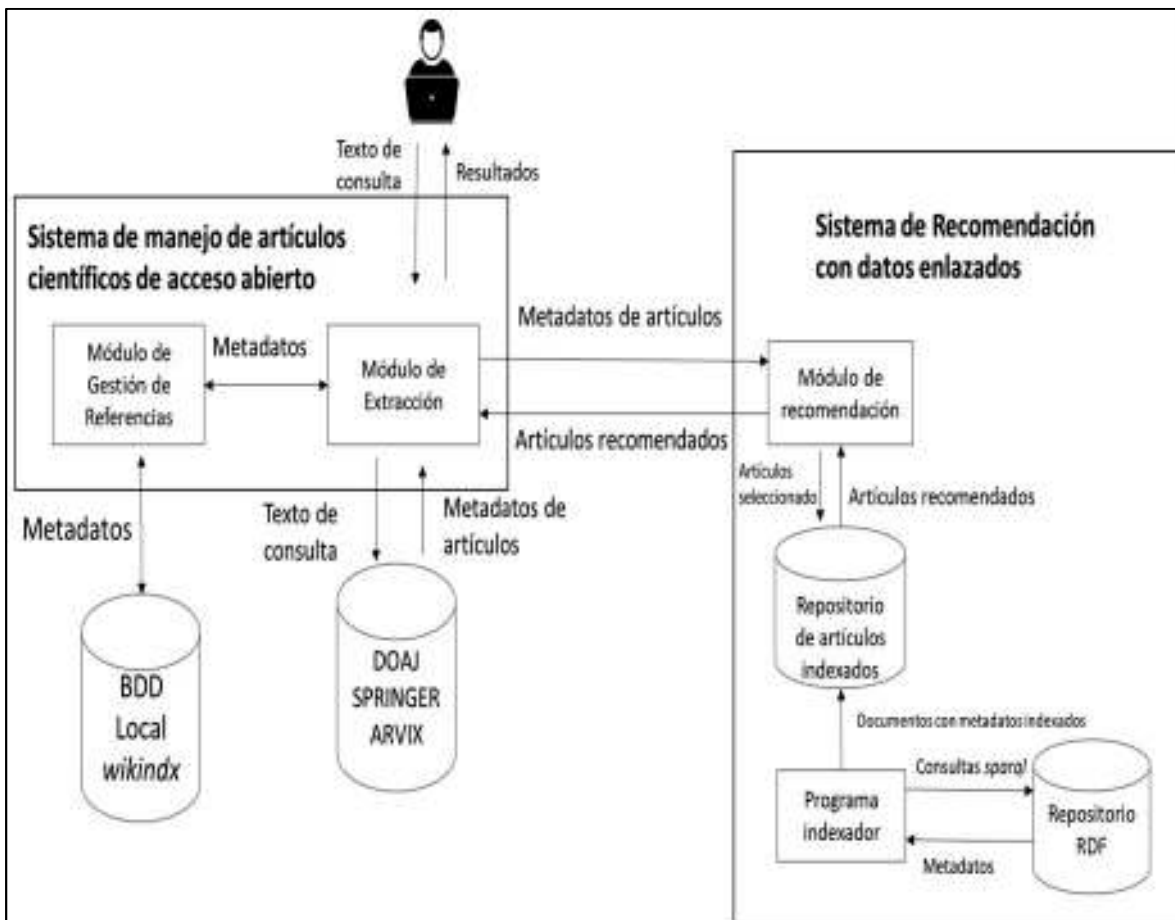


Figura 74. Arquitectura para la integración del sistema de manejo de artículos científicos de acceso abierto con el sistema de recomendación con datos enlazado

CAPITULO 3: PRUEBAS DE ACEPTACIÓN Y EVALUACIÓN DEL SISTEMA

3.1. Requisitos físicos del sistema

Espacio físico

Para la implantación del sistema se hará un solo servidor: En el servidor se encontrará el servicio *sails* que contiene el *front-end* y el *back-end* de la aplicación, y el motor de recomendación.

Este servidor será implementado en los servicios de *Google Cloud*, haciendo uso de los 300 dólares de prueba gratuito que se nos ofrece. A continuación, se presentan las características de los servidores a ser usados en la **Tabla 36**.

CARACTERÍSTICAS SERVIDOR DEL MOTOR DE RECOMENDACIÓN

Procesador	Memoria Instalada (RAM)	Tipo de Sistema	Sistema Operativo	Disco Duro
Intel(R) Xeon(R) CPU @ 2.30GHz	3 GB	64 bits	Ubuntu 18.04 LTS	500 GB

Tabla 36. Características físicas para el motor de recomendación

Las herramientas que deben ser instaladas para el funcionamiento del programa se muestran en la **Tabla 37 y 38**:

HERRAMIENTAS PARA EL SERVIDOR <i>SAI/LS</i>		
Paquete	Versión	Descripción
Nodejs	v10.15.3	Entorno de ejecución para JavaScript
Npm	6.4.1	Gestionador de paquetes para node js
Xampp	7.3.2	Sistema de gestión de bases de datos MySQL, el servidor web Apache y los intérpretes para lenguajes de script PHP y Perl.
Wikindx	V4.2.2	Gestionador de referencias.
Sails js	v0.12	Servidor de nodejs donde se va a alojar la aplicación web

Tabla 37. Herramientas para el servidor *sails*

HERRAMIENTAS PARA EL MOTOR DE RECOMENDACIÓN		
Paquete	Versión	Descripción
JVM	1.8.0_201	Máquina virtual de java
JDK	1.8.0_201	Kit de desarrollo de Java
Servidor Tomcat	8.1	Servidor para la API desarrollada en JavaEE
Apache Lucene	3.6.2	Librería para indexación y búsqueda de artículos

Tabla 38. Herramientas para el servidor de recomendación

3.2. Pruebas de aceptación

3.2.1. Pruebas de aceptación de la primera iteración

En la primera iteración se realizaron pruebas de aceptación de acuerdo con las historias de usuario que se enfocaron en la utilización del sistema de manejo de artículos científicos de acceso abierto y el gestor de referencias que utiliza dicho sistema además del registro de usuarios. En la **Tablas 39, 40, 41 y 42** se muestra los casos de cada prueba.

Caso de prueba de aceptación	
Código: 1	Historia de usuario (nro. y nombre): 1, Utilizar el sistema de manejo de artículos científicos de acceso abierto
Nombre: Prueba obtener artículos científicos de DOAJ	
Descripción: El investigador podrá utilizar el buscador del “Sistema de manejo de artículos científicos de acceso abierto”. Este buscador permite al investigador encontrar artículos de revistas científicas del repositorio digital de DOAJ.	
Condiciones de ejecución: Se debe de escoger una categoría para obtener resultados más relevantes.	
Entrada / pasos de ejecución: <ol style="list-style-type: none"> 1. En la barra lateral izquierda presionar en la pestaña de búsqueda. 2. Se presenta la pantalla de búsqueda. 3. Elegir la categoría en la cual se desea encontrar los artículos. 4. Ingresar las palabras clave de lo que se desea buscar. 5. Presionar el botón de búsqueda. 	

Continuación de la Tabla 39

<p>Resultado esperado:</p> <p>El sistema muestra una lista con los detalles de cada artículo que se ha encontrado y que concuerde con las palabras ingresadas en la caja de texto.</p>
<p>Evaluación de la prueba:</p> <p>Los resultados obtenidos de la búsqueda muestran los artículos que concuerdan con las palabras clave que se han ingresado en la caja de texto del buscador y con sus respectivos metadatos.</p>

Tabla 39. Casos de prueba de aceptación

Caso de prueba de aceptación	
Código: 2	Historia de usuario (nro. y nombre): 2, Utilizar el gestor de referencias del “Sistema de manejo de artículos científicos de acceso abierto”.
Nombre: Prueba de sincronización con el gestor de referencias wikindx.	
Descripción: El investigador podrá guardar el artículo presentado en el gestor de referencias bibliográficas <i>wikindx</i> .	
Condiciones de ejecución: Previamente se debió de realizar el proceso de búsqueda.	
Entrada / pasos de ejecución: <ol style="list-style-type: none"> 1. Elegir el artículo de preferencia en la lista presentada. 2. Presionar en el botón “Sincronizar con <i>Wikindx</i>”. 	
Resultado esperado: El sistema se redirige a una nueva página donde se muestra un mensaje que indica que el artículo se ha sido sincronizado correctamente.	
Evaluación de la prueba: El artículo guardado consta en la base de datos de <i>wikindx</i> .	

Tabla 40. Caso de prueba de aceptación 2

Caso de prueba de aceptación	
Código: 3	Historia de usuario (nro. y nombre): 3, Registrar usuarios
Nombre: Prueba registrar de datos del usuario.	
Descripción: El investigador se registrará en el sistema y la información será guardada en una base de datos.	
Condiciones de ejecución: Ingresar a la página de inicio del sistema.	
Entrada / Pasos de ejecución: <ol style="list-style-type: none"> 1. Elegir la opción de “Registro de nuevo usuario”. 2. Ingresar el nombre del usuario. 3. Ingresar el apellido del usuario. 4. Ingresar el correo electrónico. 5. Ingresar una contraseña. 6. Verificar la contraseña. 7. Presionar el botón “registrar”. 	
Resultado esperado: Si los datos fueron ingresados correctamente el sistema presentará un mensaje que diga “Usuario registrado con éxito”.	
Evaluación de la prueba: El usuario registrado se ve reflejado en la base de datos de usuario.	

Tabla 41. Caso de prueba de aceptación 3

Caso de prueba de aceptación	
Código: 4	Historia de usuario (nro. y nombre): 3, Registrar usuarios
Nombre: Prueba de funcionamiento de <i>login</i> .	
Descripción: El usuario solo podrá ingresar al sistema si está iniciada su sesión.	
Condiciones de ejecución: Registrarse en el sistema y constar en la base de datos	

Continuación de la Tabla 42

<p>Entrada / pasos de ejecución:</p> <ol style="list-style-type: none"> 1. Elegir la pestaña de “Login”. 2. Ingresar el correo electrónico. 3. Ingresar la contraseña. 4. Presionar el botón “Iniciar Sesión”
<p>Resultado esperado:</p> <p>Si los datos fueron ingresados correctamente el sistema se redirigirá a la pantalla principal del sistema.</p> <p>Si los datos no fueron ingresados correctamente se muestra un mensaje de error.</p>
<p>Evaluación de la prueba:</p> <p>El usuario ha ingresado los datos correctamente y se ha redirigido a la pantalla principal.</p> <p>El sistema muestra un mensaje de error cuando el usuario ingresó datos incorrectos.</p>

Tabla 42. Caso de prueba de aceptación 4

3.2.2. Pruebas de aceptación de la segunda iteración

En la segunda iteración se utilizó las historias de usuario relacionadas la creación de una biblioteca general donde se almacenan los artículos científicos guardados por todos los usuarios y la visualización de citas bibliográficas de cada artículo guardado. En las **Tablas 43, 44, 45 y 46** se detalla los casos de cada prueba para la segunda iteración.

Caso de prueba de aceptación	
Código: 5	Historia de usuario (nro. y nombre): 4, Usar biblioteca
Nombre: Prueba de visualización de biblioteca.	
Descripción: El usuario podrá visualizar todos los artículos guardados en la base de datos sincronizada al <i>wikindx</i> en una lista.	
Condiciones de ejecución:	
Se debe de haber guardado al menos un artículo para que sea mostrado en la lista.	

Continuación de la Tabla 43

<p>Entrada / pasos de ejecución:</p> <ol style="list-style-type: none"> 1. Elegir la pestaña de “Recomendador”. 2. Se despliega la opción de “Biblioteca”. 3. Presionar la opción de “Biblioteca”
<p>Resultado esperado:</p> <p>Se presenta la vista de la biblioteca sin resultados hasta que el usuario ingrese un parámetro en el buscador.</p>
<p>Evaluación de la prueba:</p> <p>El usuario se ha redirigido a la pantalla de biblioteca.</p>

Tabla 43. Caso de prueba de aceptación 5

Caso de prueba de aceptación	
Código: 6	Historia de usuario (nro. y nombre): 4, Usar biblioteca
Nombre: Prueba de búsqueda en biblioteca.	
Descripción: El usuario podrá buscar en la biblioteca entre todos los artículos que ha guardado previamente	
<p>Condiciones de ejecución:</p> <ul style="list-style-type: none"> - Se debe de haber guardado al menos un artículo para que sea mostrado en la lista. 	
<p>Entrada / pasos de ejecución:</p> <ol style="list-style-type: none"> 1. Se presenta la pantalla “Biblioteca”. 2. En la caja de texto ingresar un título o palabras clave. 3. Presionar el botón “Buscar”. 	
<p>Resultado esperado:</p> <ul style="list-style-type: none"> - Se presenta en la vista de la biblioteca los artículos que contienen en su título las palabras clave ingresadas. - Si no se han ingresado parámetros de búsqueda se muestran todos los artículos 	

Continuación de la Tabla 44

<p>Evaluación de la prueba:</p> <ul style="list-style-type: none"> - El usuario ha obtenido todos los resultados que contienen las palabras ingresadas. - El usuario ha obtenido todos los artículos cuando no ingresó ningún parámetro
--

Tabla 44. Caso de prueba de aceptación 6

Caso de prueba de aceptación	
Código: 7	Historia de usuario (nro. y nombre): 5, Visualizar citas bibliográficas
Nombre: Prueba de visualización de citas.	
Descripción: El usuario podrá visualizar las citas de los artículos guardados en la base de datos sincronizada con el gestor de referencias <i>wikindx</i>	
Condiciones de ejecución: Ingresar a la vista de “Biblioteca” y haber realizado la búsqueda.	
Entrada / pasos de ejecución: <ol style="list-style-type: none"> 1. Elegir el artículo buscado. 2. Presionar el botón de “Generar Cita”. 	
Resultado esperado: Se muestra en la pantalla una ventana con la información de las citas generadas.	
Evaluación de la prueba: El usuario obtiene las citas generadas del artículo seleccionado.	

Tabla 45. Caso de prueba de aceptación 7

Caso de prueba de aceptación	
Código: 8	Historia de usuario (nro. y nombre): 6, Obtener artículos relacionados
Nombre: Prueba de obtención de artículos relacionados desde la vista de Biblioteca.	
Descripción: El investigador podrá obtener recomendaciones relevantes a un artículo.	
Condiciones de ejecución: <ul style="list-style-type: none"> - El artículo del cual se va a buscar las recomendaciones debe estar guardado en el la base de datos sincronizada con el gestor de referencias <i>wikindx</i>. - Haber ingresado a la pantalla de “Biblioteca” y haber realizado la búsqueda 	
Entrada / pasos de ejecución: <ol style="list-style-type: none"> 1. Se presenta la pantalla “Biblioteca”. 2. Se muestra una lista con todos los artículos que han sido guardados en el <i>wikindx</i>. 3. Se selecciona un artículo y presionar el botón de “Recomendación”. 4. Se presenta la pantalla de “Recomendación de Artículo” con la información de los metadatos principales. 5. Se elige un repositorio presionando uno de los botones “ACM”, “DBLP” o “IEEE”. 	
Resultado esperado: Se presenta una lista de los artículos relevantes, cada lista dependerá del repositorio seleccionado.	
Evaluación de la prueba: Se observa en la pantalla una lista con todos los artículos relevantes que concuerdan con el artículo que fue seleccionado previamente.	

Tabla 46. Caso de prueba de aceptación 8

3.2.3. Pruebas de aceptación de la tercera iteración

Para la tercera iteración de realizo pruebas de aceptación para las historias de usuario referentes a obtención de artículos científicos, En esta iteración se implementó el motor de recomendación, la **Tabla 47** muestra el caso de prueba para esta iteración.

Caso de prueba de aceptación	
Código: 9	Historia de usuario (nro. y nombre): 6, Obtener artículos relacionados
Nombre: Prueba de obtención de artículos relacionados desde la vista de Búsqueda (DOAJ).	
Descripción: El investigador podrá obtener recomendaciones relevantes después de que haya sincronizado el artículo de DOAJ con el gestor de referencias <i>wikindx</i> .	
Condiciones de ejecución: <ul style="list-style-type: none">- Haber buscado un artículo en la pestaña de búsqueda.- Sincronizar artículo con el gestor de referencias <i>wikindx</i>.	
Entrada / pasos de ejecución: <ol style="list-style-type: none">1. Se presenta la pantalla de confirmación de que el artículo se ha guardado exitosamente.2. Seleccionar un repositorio presionando uno de los tres botones: "IEEE", "ACM" y "DBLP".	
Resultado esperado: <p>Se presenta una lista de los artículos relevantes, cada lista dependerá del repositorio seleccionado.</p>	
Evaluación de la prueba: <p>Se observa en la pantalla una lista con todos los artículos relevantes que concuerdan con el artículo proveniente del repositorio de DOAJ y que ha sido guardado previamente.</p>	

Tabla 47. Caso de prueba de aceptación 9

3.2.4. Pruebas de aceptación de la cuarta iteración

La cuarta iteración se enfocó en la ampliación y manipulación de resultados. Además, La redirección de artículos también abarco esta iteración. Las **Tablas 48,49,50 y 51** muestran los casos de prueba para las historias de usuarios correspondientes a la cuarta iteración.

Caso de prueba de aceptación	
Código: 10	Historia de usuario (nro. y nombre): 7, Ampliar resultados
Nombre: Prueba de obtención de metadatos de <i>RKB Explorer</i> .	
Descripción: El investigador podrá obtener del artículo seleccionado del <i>RKB Explorer</i> sus respectivos metadatos.	
Condiciones de ejecución:	
<ul style="list-style-type: none"> - Haber obtenido la recomendación de uno de los repositorios RDF. 	
Entrada / pasos de ejecución:	
<ol style="list-style-type: none"> 1. Se presenta la lista con los resultados de los artículos más relevantes. 2. Presionar el botón “Ampliar” del artículo de interés. 	
Resultado esperado:	
<ul style="list-style-type: none"> - Se presenta una ventana con los metadatos disponibles del artículo según el repositorio <i>RKB Explorer</i> seleccionado. 	
Evaluación de la prueba:	
<ul style="list-style-type: none"> - Se observa una ventana con los metadatos que están disponibles para ese artículo. 	

Tabla 48. Caso de prueba de aceptación 10

Caso de prueba de aceptación	
Código: 11	Historia de usuario (nro. y nombre): 8, Manipular resultados
Nombre: Prueba de editar metadatos del artículo seleccionado.	
Descripción: El investigador podrá ingresar datos en los campos vacíos.	
Condiciones de ejecución:	
<ul style="list-style-type: none"> - Haber obtenido la recomendación de uno de los repositorios de la <i>RKB Explorer</i>. IEEE, DBLP y ACM. - Haber seleccionado un artículo. 	

Continuación de la Tabla 49

<p>Entrada / Pasos de ejecución:</p> <ol style="list-style-type: none"> 1. Se presenta la ventana con los metadatos. 2. Seleccionar la caja de texto que se desea editar o llenar. 3. Ingresar la información requerida.
<p>Resultado esperado:</p> <ul style="list-style-type: none"> - El usuario podrá editar la información seleccionada del artículo.
<p>Evaluación de la prueba:</p> <ul style="list-style-type: none"> - El usuario puede editar sin restricción la información deseada.

Tabla 49. CASO DE PRUEBA DE ACEPTACIÓN 11

Caso de prueba de aceptación	
Código: 12	Historia de usuario (nro. y nombre): 8, Manipular resultados
Nombre: Prueba de guardar la información editada.	
Descripción: El investigador podrá guardar la información editada a la base de datos sincronizada con el <i>wikindx</i> .	
<p>Condiciones de ejecución:</p> <ul style="list-style-type: none"> - Haber editado la información deseada 	
<p>Entrada / pasos de ejecución:</p> <ol style="list-style-type: none"> 1. Se presenta la ventana con los metadatos editados. 2. Presionar el botón "Guardar". 	
<p>Resultado esperado:</p> <ul style="list-style-type: none"> - El usuario podrá guardar la información con los metadatos editados. 	
<p>Evaluación de la prueba:</p> <ul style="list-style-type: none"> - El usuario guardó los metadatos editados a la base de datos sincronizada con el <i>wikindx</i>. 	

Tabla 50. Caso de prueba de aceptación 12

Caso de prueba de aceptación	
Código: 13	Historia de usuario (nro. y nombre): 9, Redireccionar artículo
Nombre: Prueba de redireccionar a la página web del artículo recomendado.	
Descripción: El investigador podrá redirigirse a la página web del artículo seleccionado.	
Condiciones de ejecución:	
<ul style="list-style-type: none"> - Haber obtenido la información de dirección web en la visualización de metadatos 	
Entrada / pasos de ejecución:	
<ol style="list-style-type: none"> 1. Se presenta la ventana con los metadatos editados. 2. Presionar el botón “<i>web</i>”. 	
Resultado esperado:	
<ul style="list-style-type: none"> - El usuario podrá redirigirse a la página web del artículo seleccionado. 	
Evaluación de la prueba:	
<ul style="list-style-type: none"> - El usuario se redirige a la página web del artículo. - El usuario no se redirige si el artículo no tiene la información de dirección web 	

Tabla 51. Caso de prueba de aceptación 13

3.3. Pruebas de integridad

Estas pruebas ayudan a comprobar que los datos almacenados, tanto en la indexación del módulo de recomendación como en el gestor de artículos de referencia *wikindx*, sean los datos que se desean.

Para la indexación de artículos científicos existen tres repositorios de datos enlazados diferentes, estos son: IEEE, ACM y DBLP. Por lo que es necesario comprobar que dichos artículos científicos en verdad existen en los repositorios respectivos.

En el caso de la base de datos del gestor de referencia del *wikindx* se espera que los artículos que se almacenan, tanto del repositorio de revistas de acceso abierto (DOAJ) y del módulo de recomendación sean los que en verdad se seleccionó al momento de guardar.

A continuación, se presentan las **Tablas 52, 53, 54 y 55** representan las pruebas de integración que se van a realizar. Cada tabla contiene el código de la prueba, el nombre de la prueba, la descripción de la prueba, los pasos para realizar la prueba y los resultados que se esperan.

Caso de prueba de integridad
Código: 1
Nombre: Verificar datos indexados de la <i>RKB Explorer</i> IEEE en el repositorio local.
Descripción: Comprobar que los datos indexados del repositorio digital <i>RKB Explorer</i> que contiene artículos almacenados de la IEEE en verdad existan en dicho repositorio digital.
Pasos: <ol style="list-style-type: none"> 1. Ingreso de parámetros para una búsqueda de artículos relacionados. 2. Consultas <i>spargl</i> realizada al respectivo <i>endpoint</i> con el título del artículo seleccionado. 3. Comprobación de resultados.
Resultados Esperados: Los datos devueltos de la consulta <i>spargl</i> son correctos.
Resultados Obtenidos: Los datos indexados en el repositorio local se encuentran en el repositorio de la IEEE en la <i>RKB Explorer</i> .

Tabla 52. Prueba de integridad 1

Caso de prueba de integridad
Código: 2
Nombre: Verificar datos indexados de la <i>RKB Explorer</i> ACM en el repositorio local.
Descripción: Comprobar que los datos indexados del repositorio digital <i>RKB Explorer</i> que contiene artículos almacenados de la ACM en verdad existan en dicho repositorio digital.
Pasos: <ol style="list-style-type: none"> 1. Ingreso de parámetros para una búsqueda de artículos relacionados. 2. Consultas <i>spargl</i> realizada al respectivo <i>endpoint</i> con el título del artículo seleccionado. 3. Comprobación de resultados.

Continuación de Tabla 53.

Resultados Esperados: Los datos devueltos de la consulta <i>sparq</i> /son correctos.
Resultados Obtenidos: Los datos indexados en el repositorio local se encuentran en el repositorio de la <i>RKB Explorer</i> de la ACM.

Tabla 53. Prueba de integridad 2

Caso de prueba de integridad
Código: 3
Nombre: Verificar datos indexados de la <i>RKB Explorer</i> DBLP en el repositorio local.
Descripción: Comprobar que los datos indexados del repositorio digital <i>RKB Explorer</i> que contiene artículos almacenados de la DBLP en verdad existan en dicho repositorio digital.
Pasos: <ol style="list-style-type: none"> 4. Ingreso de parámetros para una búsqueda de artículos relacionados. 5. Consultas SPARQL realizada a la interfaz de consulta con el título del artículo seleccionado. 6. Comprobación de resultados.
Resultados Esperados: Los datos devueltos de la consulta <i>sparq</i> /son correctos.
Resultados Obtenidos: Los datos indexados en el repositorio local se encuentran en el repositorio de la <i>RKB Explorer</i> de la DBLP.

Tabla 54. Prueba de integridad 3

Caso de prueba de integridad
Código: 4
Nombre: Verificar que los artículos de DOAJ y del módulo de recomendación son guardados en la base de datos de <i>wikindx</i> .
Descripción: Comprobar que los artículos seleccionados del DOAJ o del módulo de recomendación sean guardados en la base de datos del gestor de referencias <i>wikindx</i> .

Continuación de Tabla 55.

Pasos <ol style="list-style-type: none">1. Sincronización de artículo con <i>wikindx</i>.2. Sentencia SQL en la base de datos de <i>wikindx</i>. 3. Comprobación de resultados.
Resultados Esperados: Los datos devueltos de la consulta SPARQL son correctos.
Resultados Obtenidos: Los resultados devueltos por las consultas SQL muestran a los artículos seleccionados del repositorio DOAJ.

Tabla 55. Prueba de integridad 4

3.4. Evaluación del software

3.4.1. Encuesta de satisfacción de los usuarios

Terminada la fase de implementación y pruebas de aceptación se procede a verificar si el sistema cumple con las expectativas del usuario.

Se realizará una encuesta en donde se calificará la facilidad de uso, la satisfacción del usuario, el tiempo de respuesta y el diseño de la aplicación.

Cada pregunta tendrá una ponderación del 1 al 4, donde: 1 es malo, 2 es regular, 3 es bueno y 4 muy bueno. Las preguntas divididas según su categoría son:

Facilidad de uso:

1. La información de los artículos encontrados como: autores, fecha de publicación, palabras clave, etc. ¿Ha sido?

Satisfacción de usuario:

2. ¿La información encontrada en su búsqueda le ha parecido?
3. La información de los artículos encontrados como: autores, fecha de publicación, palabras clave. ¿Ha sido?

Tiempo de respuesta:

4. ¿La rapidez con la que el sistema ha ejecutado sus diferentes tareas ha sido?

Diseño:

5. ¿El diseño y la distribución de los diferentes elementos de la página web le ha parecido?

3.4.2. Análisis de resultados de la encuesta

Se han encuestado a 15 personas, las cuáles han usado primero el software y han dado su calificación contestando a las preguntas presentadas.

Las respuestas varían del 1 al 4, siendo: 1 es malo, 2 es regular, 3 es bueno y 4 muy bueno. A continuación, los resultados de las encuestas:

Facilidad de uso:

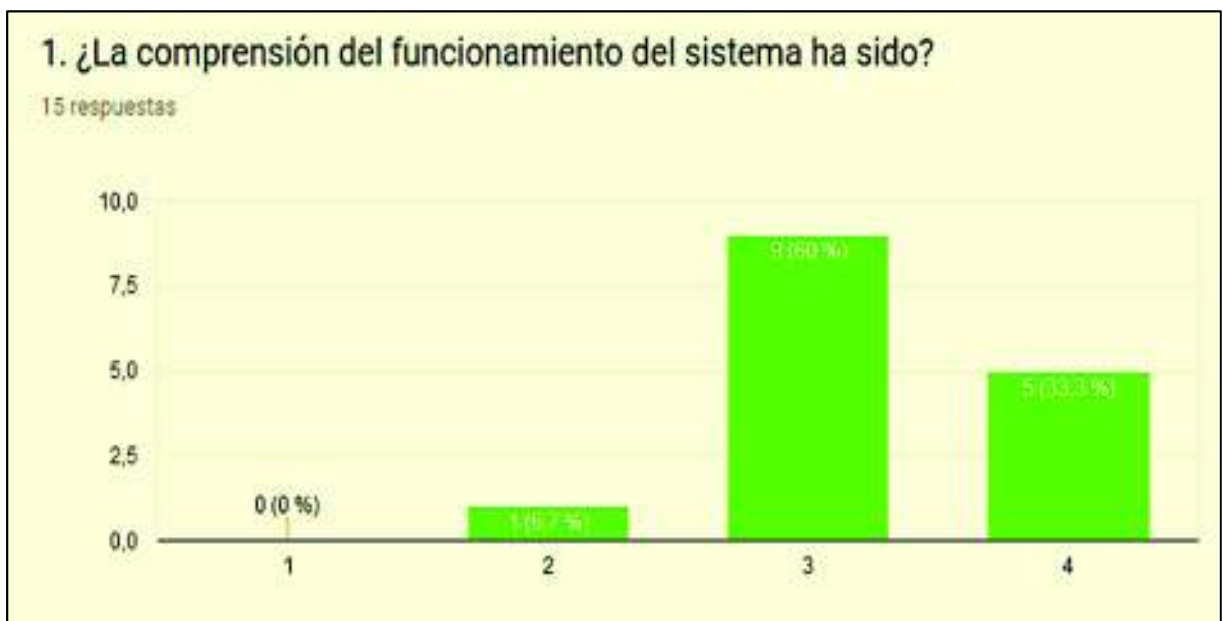


Figura 75. Resultados pregunta 1 de la encuesta

En la **Figura 75** se aprecia que del total de 15 usuarios encuestados 1 usuario consideró que es regular el comprender en un inicio el funcionamiento del software, en cambio 9 usuarios consideraron que es buena la comprensión y 5 usuarios consideraron que es muy buena.

Satisfacción de usuario:



Figura 76. Resultados pregunta 2 de la encuesta

En la **Figura 76** se aprecia que del total de 15 usuarios encuestados 4 usuarios consideraron que es buena la recomendación entregada y 11 usuarios muy buena.

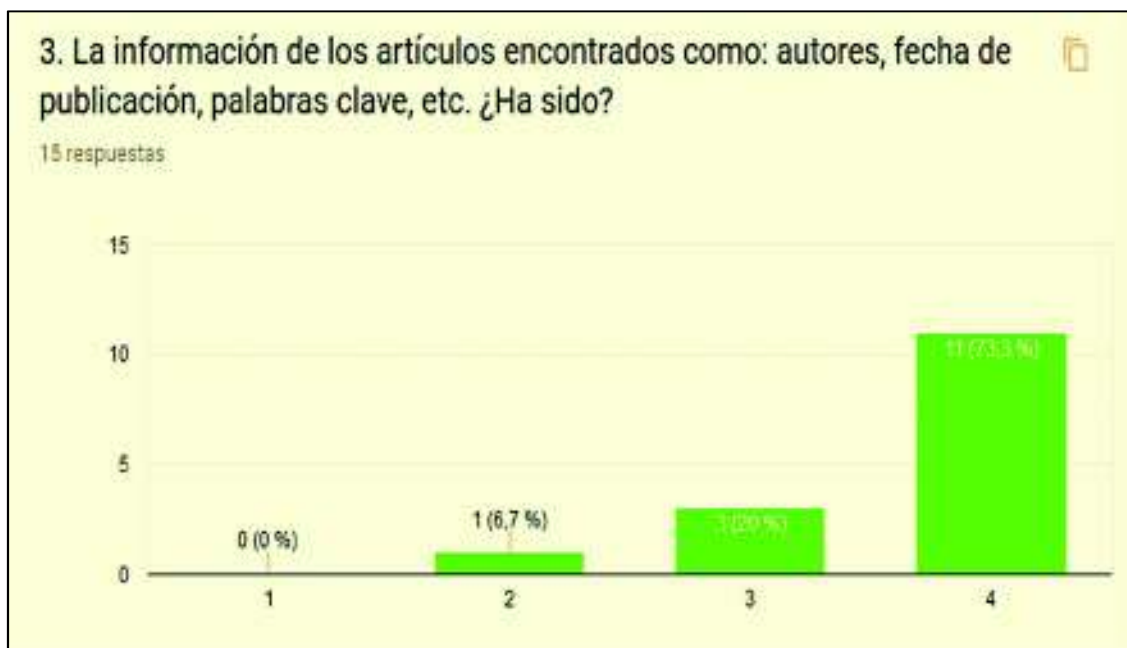


Figura 77. Resultados pregunta 3 de la encuesta

En la **Figura 77** se aprecia que del total de 15 usuarios encuestados 1 consideró que los datos presentados de los artículos ha sido regular, 3 lo consideraron buena y finalmente 11 usuarios la consideró muy buena.

Tiempo de respuesta:

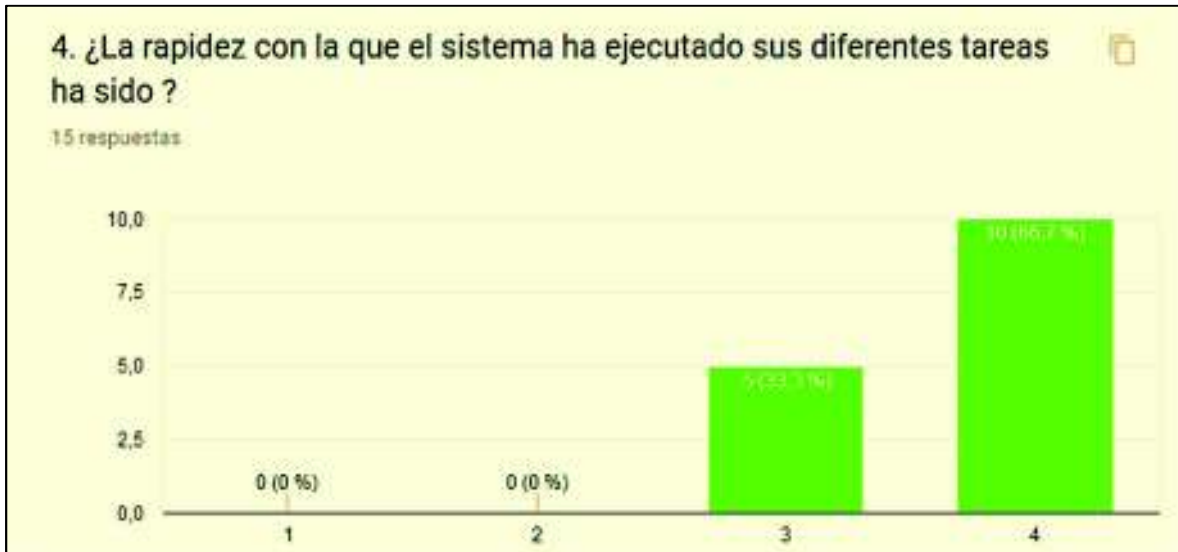


Figura 78. Resultados pregunta 4 de la encuesta

En la **Figura 78** se aprecia que de los 15 usuarios encuestados 5 consideran que la rapidez de funcionamiento del sistema es buena y el 10 la consideraron muy buena.

Diseño:



Figura 79. Resultados pregunta 5 de la encuesta

En la **Figura 79** se aprecia que de los 15 usuarios encuestados 1 consideró que el diseño de la página web es regular, 3 lo consideraron buena y 11 muy buena.

En base a los resultados obtenidos en la encuesta de del usuario, se requiere obtener una calificación promedio de cada categoría para luego obtener una calificación general sobre

la satisfacción del usuario con la aplicación web. Los promedios por pregunta se presentan en la **Tabla 56**.

Persona nro	1. ¿La comprensión del funcionamiento del sistema ha sido?	2. ¿La información encontrada en la recomendación le ha parecido?	3. La información de los artículos encontrados como: autores, fecha de publicación, palabras clave, etc. ¿Ha sido?	4. ¿La rapidez con la que el sistema ha ejecutado sus diferentes tareas ha sido ?	5. ¿El diseño de la página web le ha parecido?
1	3	4	4	4	4
2	3	4	4	4	3
3	4	4	4	4	4
4	4	4	4	4	4
5	4	4	4	4	4
6	3	3	3	4	4
7	4	4	4	4	4
8	4	4	4	3	4
9	3	4	3	3	4
10	2	3	4	3	2
11	3	4	4	4	4
12	3	3	3	3	3
13	3	4	4	4	4
14	3	3	2	3	3
15	3	4	4	4	4
Promedio	3,27	3,73	3,67	3,67	3,67

Tabla 56. Respuestas a preguntas de encuesta

Una vez obtenidos los resultados promediados por pregunta ahora se presenta el promedio por categoría y el promedio general de la calidad del sistema. Los resultados se indican en la **Tabla 57**.

Categoría	Promedio	Calificación
Facilidad de uso	3,27	Buena
Satisfacción de usuario	3,7	Muy Buena
Tiempo de respuesta	3,67	Muy Buena
Diseño	3,67	Muy Buena
Promedio General	3,58	Muy Buena

Tabla 57. Promedios por categoría y general

Como observación final se puede decir que el usuario considera que la facilidad de uso del sistema, sobre todo al principio es buena. Los resultados entregados por el sistema son muy buenos, por lo que el usuario está satisfecho. El tiempo de respuesta es muy bueno y el diseño de la aplicación web en general es muy bueno.

CAPITULO 4: CONCLUSIONES Y RECOMENDACIONES

4.1. Conclusiones

- El sistema de recomendación desarrollado utiliza la recomendación basada en contenido ya que la recomendación se realiza en base a los metadatos de un artículo científico, en específico el título y el *abstract*. Estos elementos son utilizados en la indexación para que funcionen junto a un motor de búsqueda y a un motor de recomendación.
- Las interfaces de consulta utilizadas para extraer los metadatos de los artículos científicos funcionan con la versión 1.0 del lenguaje de consulta *sparql*, por lo que no se pueden aplicar ciertas funciones o búsquedas con más de una palabra.
- Existen artículos que no tienen sus metadatos completos por lo que la extracción de metadatos de algunos de ellos se los hará de manera incompleta en varias ocasiones, por lo que el usuario podrá llenar manualmente algunos de los artículos que no existen en los repositorios de datos utilizados.
- La indexación de artículos para el repositorio de datos de la ACM y DBLP es una tarea muy pesada ya que actualmente existe una gran cantidad de artículos. Se necesita mucho tiempo y equipos potentes para indexar todos los datos a la vez.
- Los artículos almacenados en los repositorios de datos de la DBLP aparecen en otros repositorios como como la ACM y la IEEE. Teniendo en cuenta ello, lo ideal sería que exista un repositorio digital RDF unificado en donde se almacenen varios artículos científicos de diferentes repositorios.
- El acceso a los artículos completos de los repositorios IEEE, ACM y DBLP se encuentran en las paginas oficiales de estos organismos por lo que acceder a estos artículos implica un gasto.

4.2. Recomendaciones

- Es recomendable indexar el título y *abstract* de los artículos para que haya una mejor recomendación, en el caso de los artículos que no tienen *abstract* se podrá perder cierta precisión en la búsqueda.

- Se recomienda actualizar el software cada vez que haya cambios en los repositorios digitales utilizados en el sistema de recomendación ya que podría haber conflictos con los prefijos y ontologías.
- Se recomienda para un futuro tener una mejor gestión de las cuentas de los usuarios que se han registrado por motivos de seguridad y para que el usuario tenga una mejor experiencia.
- Es recomendable separar la parte de la vista y la parte de la logística del negocio de una aplicación en dos tecnologías diferentes para que la experiencia del usuario sea mejor. los recursos sean distribuidos de acuerdo con la necesidad de cada parte del negocio.
- Se recomienda la compra de un hosting o servidor en la nube para que todos los interesados en la aplicación vayan observando los diferentes cambios y aprueben o desaprueben los mismos.
- Es recomendable el uso de una herramienta de versionamiento de código para el control de los nuevos cambios que se van implementando en la aplicación y de esa manera evitar errores. Además, en caso de que la aplicación sufra cambios grandes que afecten la funcionalidad y ejecución del programa se podrá regresar a una anterior versión donde se considere que estaba correcto.
- Se recomienda, debido a la gran cantidad de datos que se extraen de los repositorios RDF, utilizar técnicas de Big Data para mejorar el rendimiento en la extracción e indexación de la información.

BIBLIOGRAFÍA:

- [1] M. Hallo y S. Luján-Mora, "An Architecture to Enhance a Reference Management System with Recommendations from Open Linked Data," 10th International Conference on Computer Supported Education, Funchai, Portugal, 2018.
- [2] F. Zarrinkalam y M. Kahani, "A multi-criteria hybrid citation recommendation system based on linked data", 2012 2nd International Conference on Computer and Knowledge Engineering (ICCKE), p. 283, 2012.
- [3] L. Peska y P. Vojtas, "Enhancing Recommender System with Linked Open Data", Flexible Query Answering Systems, p. 484, 2013.

- [4] C. Musto, P. Lops, M. de Gemmis y G. Semeraro, "Semantics-aware Recommender Systems exploiting Linked Open Data and graph-based features", Knowledge-Based Systems, vol. 136, p. 1, 2017.
- [5] The WIKINDEX Team, "About Wikindx", 2018[En línea] wikindx.sourceforge.net. Disponible en: <http://wikindx.sourceforge.net/> [Último Acceso 18 Apr. 2018].
- [6] M. Arenas y C. Buil Aranda, "La Web Semántica: Herramientas para la publicación y extracción efectiva de información en la Web," Coursera, 2016. [En línea] coursera.org. Disponible en: <https://es.coursera.org/learn/web-semantic>. [Último acceso: 11 11 2018].
- [7] M. Vallez, "La Web Semántica y las Tecnologías del Lenguaje Humano., "de Web semántica y sistemas de información documental, Gijón, Trea, 2009, pp. 155-180.
- [8] T. Berners-lee, "W3C,"2000. [En línea] w3.org. Disponible en: <https://www.w3.org/2000/Talks/1206-xml2k-tbl/slide10-0.html>. [Último acceso: 15 11 2018].
- [9] S. W. Group, "W3Consortium,"15 Enero 2008. [En línea] w3.org. Disponible en: <https://www.w3.org/TR/rdf-sparql-query/>. [Último acceso: 10 enero 2019].
- [10] L. Feigenbaum., "SPARQL Protocol and Query Language: SPARQL Frequently Asked,"8 febrero 2008. [En línea] thefigtrees.net. Disponible en: <https://bit.ly/2tFn3W4>. [Último acceso: 12 01 2019].
- [11] Wiki community, "SPARQL/Expressions and Functions," WikiBooks, Open books for an open world, [En línea] en.wikibooks.org. Disponible en: <https://bit.ly/2H4wxlQ>. [Último acceso: 12 Enero 2019].
- [12] L. Yu, "Set Up Joseki SPARQL Endpoint,"de A Developer's Guide to the Semantic Web, Springer Science & Business Media, 2011, p. 244.
- [13] ReSIST Project, "ReSIST RKB Explorer - Help,"2009. [En línea] rkbexplorer.com. Disponible en: <https://bit.ly/2Ucc962>. [Último acceso: 13 Enero 2019].
- [14] N. Kushwaha y O. P. Vyas, "Levering Bibliographic RDF Data for keyword Prediction with Association Rule Mining (ARM)," Data Science Journal, vol. 13, p. 122, 2014.
- [15] R. Meymandpour y J. Davis, "Recommendations using linked data", Proceedings of the 5th Ph.D. workshop on Information and knowledge - PIKM '12, p. 76, 2012. Disponible en: 10.1145/2389686.2389701 [Último acceso: 21 05 2019].

- [16] F. Ricci, L. Rokach y B. Shapira, "Introduction to Recommender Systems Handbook", Recommender Systems Handbook, pp. 1-35, 2010. Disponible en: 10.1007/978-0-387-85820-3_1 [Último acceso: 21 05 2019].
- [17] P. Lops, M. de Gemmis y G. Semeraro, "Content-based Recommender Systems: State of the Art and Trends", Recommender Systems Handbook, pp. 73-105, 2010. Disponible en: 10.1007/978-0-387-85820-3_3 [Último acceso: 21 05 2019].
- [18] H. Chu, Information Representation and Retrieval in the Digital Age, 1st ed. New Jersey: Information Today, Inc., 2003, 2003, pp. 103-104.
- [19] Tutorials Point, "Lucene - Indexing Classes," [En línea] tutorialspoint.com. Disponible en: <https://bit.ly/2NwXC2j>. [Último acceso: 20 01 2019].
- [20] Eernisse, M. y Riley, X. (2019). *EJS -- Embedded JavaScript templates*. [En línea] Ejs.co. Disponible en: <https://ejs.co/> [Último acceso: 7 05 2018].
- [21] Google (2019). *AngularJS*. [En línea] Docs.angularjs.org. Disponible en: <https://docs.angularjs.org/guide> [Último Acceso 17 May 2018].
- [22] The Sails Company. "Features | Sails.js," [En línea] sailsjs.com. Disponible en: <https://bit.ly/2BXGGx4>. [Último acceso: 20 05 2018].
- [23] JetBrains s.r.o, "Webstrom - The smartest JavaScript IDE," [En línea] jetbrains.com. Disponible en: <https://bit.ly/1te2zNA>. [Último acceso: 9 9 2018].
- [24] K. Shwaber y J. Sutherland, "La Guía de Scrum: Las Reglas del Juego," 2016. [En línea] scrumguides.org. Disponible en: <https://bit.ly/2yZ6kBL>. [Último acceso: 20 10 2018].
- [25] A. Menzinsky, G. López y J. Palacio, "Guía de Scrum Manager", 2016, pp. 19-47 [En línea] www.safecreative.org. Disponible en: https://www.scrummanager.net/files/sm_proyecto.pdf [Último acceso: 18 05 2019].
- [26] F. Escobar, "Desarrollo de un sistema de manejo de artículos científicos de acceso abierto", Pregrado, Escuela Politécnica Nacional, 2019.
- [27] AdminLTEStudio, "adminlte," 2014. [En línea] adminlte.io. Disponible en: <https://adminlte.io/themes/AdminLTE/documentation/index.html>. [Último acceso: 21 03 2019].
- [28] Microsoft Corp, "ASP.NET MVC Overview," Microsoft, 27 Febrero 2013. [En línea] docs.microsoft.com. Disponible en: <https://bit.ly/2H2JkoC>. [Último acceso: 09 09 2018]

GLOSARIO

ACM: Asociación de Maquinaria Computacional (*Association for Computing Machinery*), se encarga de la divulgación científica en el ámbito de la computación.

API: Interfaz de programación de aplicaciones (*Interfaz de Programación de Aplicaciones*) que cumplen muchas funciones que pueden ser usadas por otro software

Dataset: Conocido en español como conjunto de datos, es una colección de datos que generalmente está tabulada.

DBLP: *Digital Bibliography & Library Project*, es un repositorio web que contiene grandes cantidades de datos bibliográficos.

DOAJ: Directorio de Revistas de Acceso Abierto (*Directory of Open Acces Journals*)

Endpoint: Dispositivo informático remoto que se encuentra conectado a la red y es accesible para los dispositivos, de manera pública o restringida.

GET: Método http para obtener información de un recurso.

Grafo: Conjunto de nodos que están enlazados entre sí por una arista.

Html: Lenguaje de marcas de hipertexto (*Hyper text markup language*).

Http: Protocolo de transferencia de hipertexto (*Hypertext Transfer Protocol*).

IDE: Entorno de desarrollo integrado (*Integrated Development environment*).

IEEE: Instituto de ingenieros eléctricos y electrónicos (*Institute of Electrical and electronics engineers*).

JSON: Notación de objetos javascript (*JavaScript Object Notation*).

URI: Identificador de recursos uniforme (*Uniform Resource Identifier*).

Metadato: Conjunto de datos que describen a un recurso.

POST: Método http para enviar información.

Resiliencia: Capacidad de superación ante riesgos y fallos.

REST: Transferencia de estado representacional (*Representational State Transfer*).

RDF: Marco de descripción de recursos (*Resource Description Framework*).

RKB: Base de conocimiento de resiliencia (*Resilience Knowledge Base Explorer*).

SCRUM: Metodología ágil para el desarrollo de software.

SOAP: Protocolo de acceso a objetos simple (*Simple Object Acces Protocol*).

SPARQL: Protocolo y consulta de consulta RDF (*Protocol and RDF Query Language*).

SQL: Lenguaje de consulta estructurado (*Structured Query Language*).

URL: Localizador uniforme de recursos (*Uniform Resource Locator*).

XML: Lenguaje de marcado extensible (*Extensible Markup language*).

ANEXOS

Anexo I:

Plantilla *scrum* para la primera iteración, archivo plantilla_sprint1.xls.

Anexo II:

Plantilla *scrum* para la segunda iteración, archivo plantilla_sprint2.xls

Anexo III:

Plantilla *scrum* para la tercera iteración, archivo plantilla_sprint3.xls

Anexo IV:

Plantilla *scrum* para la cuarta iteración, archivo plantilla_sprint4.xls

Los **Anexos I, II, III** y **IV** se encuentran en el cd adjunto.