Universidad de las Ciencias Informáticas
Facultad 2

Herramienta web para la refactorización del mecanismo de búsqueda del Catálogo en Línea de ABCD 3.0

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

Autor: Lianet Labañino Rivera

Tutor: Ing. Leandro Tabares Martín

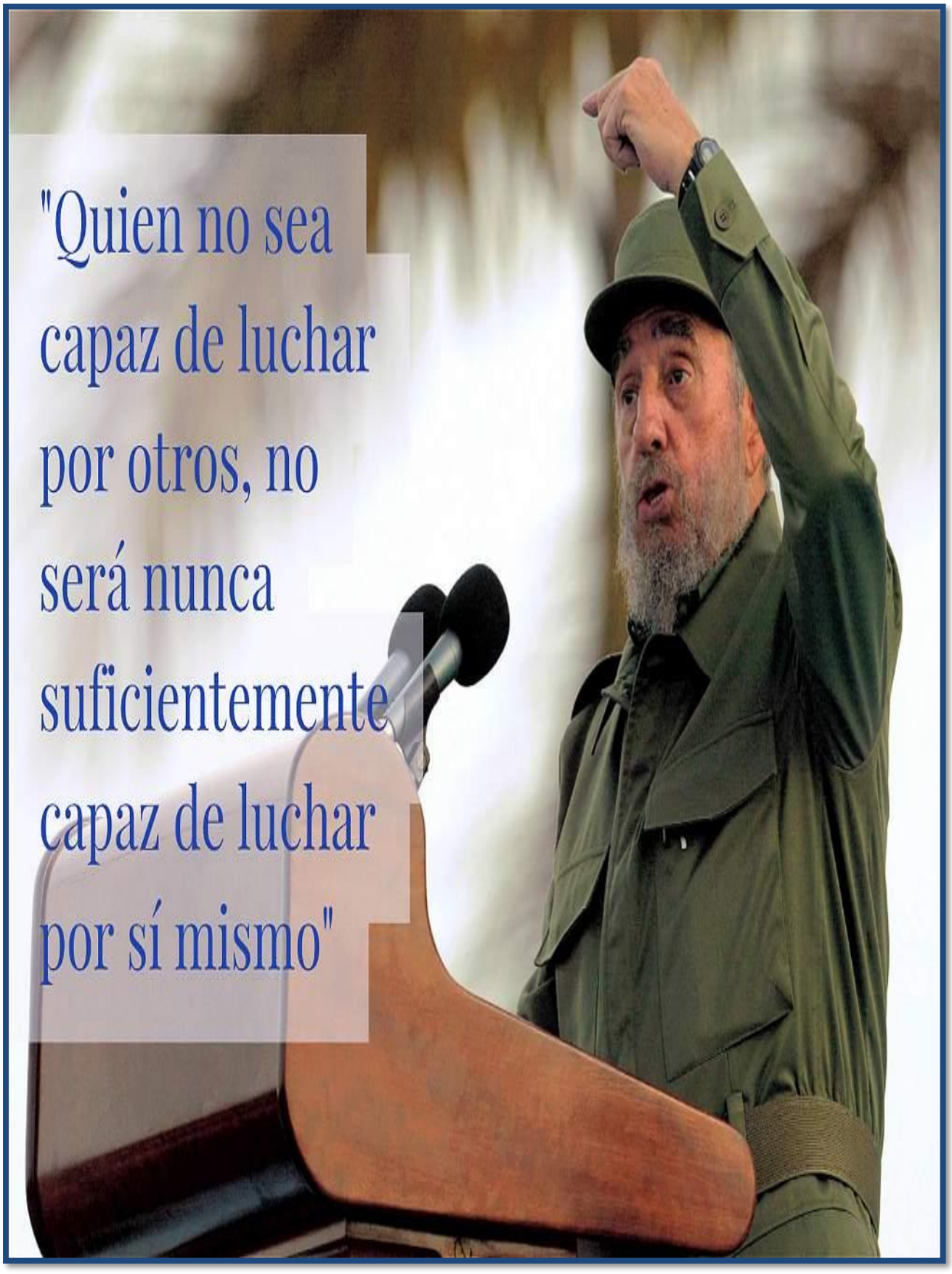
La Habana, junio de 2017
"Año 59 de la Revolución"

Declaración de autoría

Declaro ser la única autora de este trabajo y autorizo a la facultad 2 de la Universidad de las Ciencias Informáticas a hacer uso del mismo en su beneficio.

Para que así conste firmamos la presente a los 5 días del mes de julio del año 2017.

_____	_____
Lianet Labañino Rivera	Leandro Tabares Martín
Autor	Tutor


"Quien no sea
capaz de luchar
por otros, no
será nunca
suficientemente
capaz de luchar
por sí mismo"

Agradecimientos

A mi mamá y mi papá.

Dedicatoria

A mi mamá y mi papá.

Resumen

En los últimos años, la realización de la investigación sobre los catálogos en línea de acceso público se ha centrado, principalmente, en dos aspectos: el aumento de las potencialidades de la recuperación de información en las búsquedas y la mejora del diseño de las interfaces más amigables e intuitivas. El objetivo de este trabajo está enfocado a refactorizar el mecanismo de búsqueda en el Catálogo en Línea del cual disminuyendo los tiempos en la recuperación de información y realizando las búsquedas en J-ISIS por el identificador, se mejora los tiempos de respuesta de la herramienta realizada. Permitiendo un estudio de algunas contribuciones teóricas sobre este tema, seleccionando las que realmente aportan datos estructurales a la investigación permite realizar un sistema que resuelva el problema de la disponibilidad del sistema ABCDv3.0. Esto está basado en el correcto funcionamiento del mecanismo de búsqueda del sistema nombrado, Sistema para Búsqueda y Recuperación de Información, pues sus tiempos de respuesta en las búsquedas realizadas por las pruebas disminuyen con respecto a los tiempos del catálogo en línea del sistema ABCDv3.0. Se utilizó la metodología ágil XP la cual brinda una serie de artefactos que ayudan a entender la necesidad de realizar este trabajo debido al problema existente con el servidor de datos. Luego se abordó las tecnologías utilizadas como: SolR para la realización de las búsquedas y *Spring Boot* como *framework* de desarrollo. Como entorno de desarrollo la herramienta Netbeans y el lenguaje empleado para la solución es JAVA.

Palabras claves: biblioteca, búsquedas, catálogo en línea

Índice general

Índice de tablas

Índice de figuras

Introducción

Una de las principales necesidades de la humanidad desde sus inicios ha sido la acumulación y manejo de la información. Actualmente, esta se ha convertido en uno de los bienes más preciados para el crecimiento económico, social y cultural de las naciones, por lo que contar con centros de información que la almacenen es de vital importancia. Una especialización de estos centros son las bibliotecas las cuales, con el avance de las Tecnologías de Información y Comunicación (TICS) en materia de desarrollo de software, han sido automatizadas a través de los Sistemas Integrados de Gestión Bibliotecaria (SIGB).

Existe una gran gama de SIGBs entre los que se encuentra ABCD 3.0, desarrollado por el Centro de Informatización de la Gestión Documental (CIGED) que es un centro de desarrollo de la Universidad de las Ciencias Informáticas. Este está dirigido a la gestión integrada de procesos de bibliotecas y operación automatizada en línea. Su arquitectura tecnológica está basada en servicios, utilización de estándares web, protocolos abiertos, innovación e integración con nuevas metodologías y tecnologías de información y comunicación, plataforma multi-idioma, accesibilidad a usuarios y buenas prácticas de seguridad según los patrones internacionales.

El sistema ABCD 3.0 posee un Catálogo en Línea (OPAC, por sus siglas en inglés) que permite a los usuarios externos a la biblioteca consultar los materiales existentes en esta, convirtiéndose en uno módulos de ABCD de mayor concurrencia de peticiones. El catálogo en la actualidad realiza búsquedas sobre la base de datos documental del sistema, la cual está soportada sobre un servidor de bases de datos J-ISIS, es software libre y es desarrollado mayormente por una persona en el mundo. En la actualidad J-ISIS presenta problemas manejando múltiples conexiones concurrentes, esto limita su escalabilidad desde el punto de vista de la recuperación de información a la vez que hace más lentas las consultas desde el OPAC. El sistema de ABCD v3.0 al más de 9 usuarios realizarle muchas peticiones al mismo tiempo hay que reiniciar el servidor porque queda no disponible para los usuarios lo cual es un problema crítico para la disponibilidad de los datos en la biblioteca y satisfacción de los usuarios que la utilizan.

En el sistema ABCDv3.0 los módulos de adquisición, que permite el proceso de manejar la adquisición de los materiales, cuando entran libros nuevos a la biblioteca, por ejemplo y catalogación, que permite la descripción de los materiales en el formato establecido, además del catálogo en línea que permite la búsqueda de recursos de una biblioteca no quedan

disponibles cuando el gestor de bases de datos deja de estar disponible. Al ser los módulos más usados catalogación y OPAC esto afecta la disponibilidad del sistema completo.

Teniendo en cuenta los argumentos anteriormente expuestos se formula el siguiente **problema de investigación:** ¿Cómo contribuir a mejorar el tiempo en la recuperación de información en la realización de búsquedas concurrentes en el Catálogo en Línea del sistema ABCD 3.0?

Por tanto, el **objeto de estudio** de la investigación se centra en las herramientas web que implementan mecanismos de búsqueda de información en bases de datos documental y se define como **campo de acción,** herramienta web que implementa el mecanismo de búsqueda de información en el catálogo en línea del sistema ABCD 3.0.

Para solucionar el problema planteado se propone como **objetivo general**: Desarrollar una herramienta web para la refactorización del mecanismo de búsqueda de información en el Catálogo en Línea disminuyendo los tiempos en la recuperación de información.

A partir del objetivo general se derivan los siguientes **objetivos específicos**:

1. Elaborar el marco teórico referencial requerido para la investigación.
2. Diseñar la herramienta informática para la búsqueda de información en el servidor de bases datos J-ISIS del sistema ABCD 3.0.
3. Implementar la herramienta informática que mejore la búsqueda de información en el servidor de bases datos J-ISIS.
4. Validar la herramienta informática para la búsqueda de información en el servidor de bases datos J-ISIS del sistema ABCD 3.0.

Para dar cumplimiento a los objetivos específicos se proponen las siguientes **tareas de investigación**:

- Análisis de tecnologías asociadas a la implementación de las búsquedas en el OPAC de ABCD v3.0, para la Refactorización del mecanismo de búsqueda.
- Análisis de la herramienta SolR y sus posibilidades de integración con el gestor de bases de datos J-ISIS.
- Realización de la indexación de la base de datos J-ISIS con SolR para realizar búsquedas.

- Elaboración de la estrategia de prueba para la validación de las funcionalidades del sistema.

Para resolver y dar cumplimiento a los objetivos y las tareas propuestas se emplearon métodos científicos, los cuales se definen a continuación:

Métodos de investigación científica

Métodos teóricos

- Histórico-Lógico para analizar la evolución del proceso búsqueda, su perfeccionamiento a través del desarrollo de las bibliotecas y su modernización dado el impacto de las innovaciones tecnológicas; realizando un estudio crítico de trabajos anteriores para utilizarlos como punto de referencia y comparación de los resultados alcanzados.

- El análisis y síntesis al descomponer el problema de investigación en elementos por separado y profundizar en el estudio de cada uno de ellos, para luego sintetizarlos en la solución propuesta.

Método empírico

- La entrevista se utilizó para conocer los metadatos por lo que se debe indexar en el motor de búsqueda, especificar los principales metadatos de J-ISIS en el formato MARC21 y definir que parámetros de búsqueda se mostrarán en la interfaz de búsqueda.

Como **resultados** de la investigación se espera obtener:

- Aplicación de la herramienta SolR en la indexación de la base de datos J-ISIS disminuyendo los tiempos en la recuperación de información.

- Refactorización del mecanismo de búsqueda del OPAC de ABCD v3.0 aumentando su escalabilidad desde la perspectiva de búsquedas concurrentes.

Estructuración por capítulos

El trabajo se estructura de la siguiente manera: introducción, capítulo 1, capítulo 2, capítulo 3 y conclusiones, las conclusiones de la investigación general, recomendaciones, referencias bibliográficas y anexos. A continuación, una breve reseña del contenido de los capítulos:

Capítulo 1: Fundamentación teórica. Se realiza un análisis de los principales conceptos y términos asociados a la investigación, así como un estudio de las principales tecnologías, herramientas y metodología a utilizar.

Capítulo 2: Descripción de la solución. Se realiza una descripción de la propuesta solución y se determinan los requisitos funcionales y no funcionales del sistema. Se evidencia las fases de la metodología empleada como, exploración que se realiza en el comienzo de desarrollo del sistema y se extraen las historias de usuario, además la fase de planeación que resuelve en cuantas iteraciones se realizan las historias de usuario descritas que son funcionalidades del sistema. Además de la extraer los requisitos funcionales y no funcionales.

Capítulo 3: Diseño, Implementación y pruebas. En la fase de diseño de la metodología XP, se realiza una descripción del diseño, la arquitectura del sistema y se definen los patrones de diseño empleados en la realización del sistema. Además, de la relación entre las clases por medio de las tarjetas CRC y las tareas de ingeniería que son la descomposición de la historia de usuario correspondiente, se describen las pruebas realizadas y los resultados de las mismas en la fase de pruebas de la metodología XP.

Capítulo 1 Fundamentación teórica

Introducción

El presente capítulo contiene los conceptos fundamentales asociados al problema de investigación. Se describe el proceso de búsqueda de información en el catálogo en línea ABCD. Se presenta un estudio sobre diferentes herramientas informáticas de indexación de información y se aborda la metodología de desarrollo de software a utilizar y las distintas herramientas y tecnologías que serán empleadas en el desarrollo de la propuesta de solución.

1.1 Conceptos fundamentales

En el epígrafe se presentan diferentes conceptos que permiten comprender el proceso de desarrollo de la propuesta de solución:

1.1.1 Recuperación de información digital

La recuperación de información digital consiste en encontrar documentos relevantes que satisfagan la necesidad de información de un usuario, expresada en un determinado lenguaje de consulta (Bordignon 2007). Permite el acceso, búsqueda, localización de la información relevante dentro de grandes volúmenes de datos (Pinto 2004).

1.1.2 Catálogo en Línea de Acceso Público

El Catálogo en Línea de Acceso Público es la herramienta informática de acceso público que permiten la consulta y visualización de los registros que componen una biblioteca digital, le brinda al usuario la posibilidad de establecer si su necesidad de información puede o no ser satisfecha por los recursos que posee así como bibliotecas definir un perfil determinado de usuario y establecer los alcances apropiados para este (Milena, Cárdenas y Parra 2008).

1.1.3 Indexación

La indexación consiste en “registrar ordenadamente datos e informaciones, para elaborar su índice” (Real Academia Española 2017). Facilita la búsqueda de información y ayuda a seleccionar con mayor exhaustividad la información más pertinente de acuerdo con las características de los usuarios (Martínez 2012).

En el desarrollo de la presente investigación se asumió los diferentes conceptos definidos anteriormente.

1.2 Proceso de búsqueda de información en el catálogo en línea ABCD

En la actualidad el proceso de búsqueda o recuperación de información digital es complejo por los grandes volúmenes de datos que están almacenados en las bibliotecas electrónicas. En el sistema informático ABCD en su versión 3.0, la búsqueda de fuentes bibliográficas en el catálogo en línea se realiza mediante una interfaz gráfica, que le permite al usuario la introducción de diferentes criterios como: título, autor, editorial, ISBN (*International Standard Book Number,* Número Estándar Internacional de Libro), ISSN (*International Standard Serial Number*, Número Internacional Normalizado de Publicaciones Seriadas). Este catálogo realiza las búsquedas en toda la base de datos documental del sistema contenida en el servidor J-ISIS, recuperando los registros existentes que coincidan con los criterios de búsquedas insertados por el usuario. En este proceso de recuperación se utilizan las siguientes tecnologías informáticas:

1.2.1 OPAC

El sistema de Automatización de Bibliotecas y Centros de Documentación ABCD en su versión 3.0 tiene un módulo OPAC que es un catálogo en línea que permite la búsqueda de materiales y recursos existentes en la biblioteca. Este está dirigido a la satisfacción del usuario que accede a las bibliotecas en búsqueda de información digital para su interés.

1.2.2 Sistema Informático J-ISIS

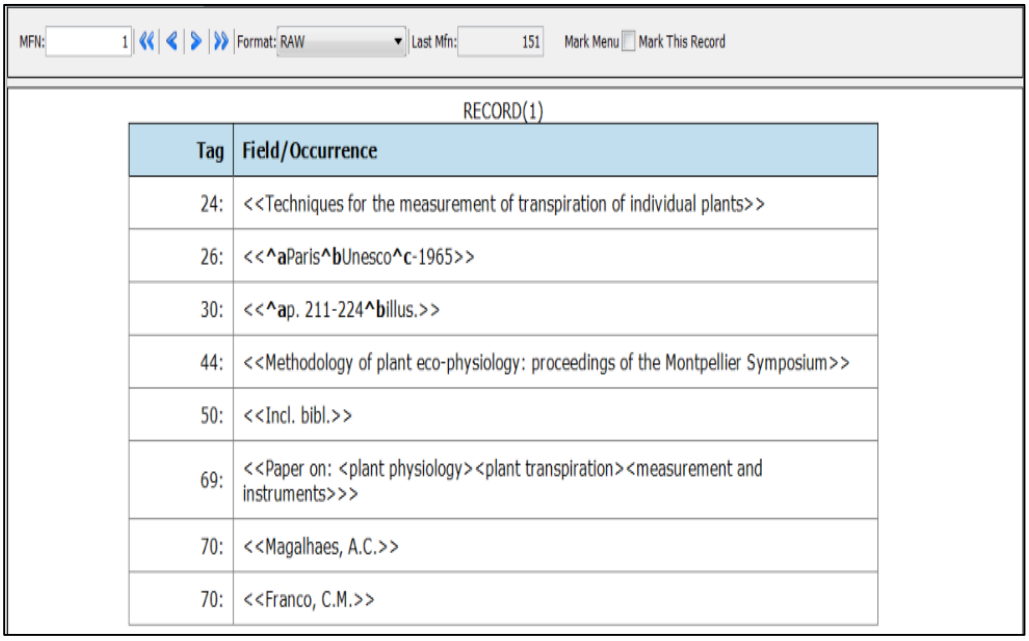
El sistema informático Java - Conjunto Integrado para Servicios de Información (J-ISIS, Java - *Integrated Set for Information Services*) es un Sistema Gestor de Bases de Datos que permite la elaboración y administración de bases de datos. Fue creado por el francés Jean-Claude Dauphin, desarrollador de software de la UNESCO (*United Nations Educational, Scientific and Cultural Organization*, Organización de las Naciones Unidas para la Educación, la Ciencia y la Cultura) en octubre del año 2008. Este servidor utiliza registros bibliográficos que son almacenados en el formato MARC21 (Dauphin 2015).

Registros bibliográficos en J-ISIS

La información que almacenan las bases de datos de J-ISIS es tratada con el término de registro bibliográfico (record, en inglés). Un registro bibliográfico es una representación de un documento mediante una serie de datos descriptivos establecidos por normas internacionales, el registro bibliográfico va a interactuar entre el usuario y el documento, informando sobre aspectos como el autor, el título, el editor o el contenido (Dauphin 2015).

Estructura de los registros bibliográficos en J-ISIS del formato MARC21

MARC (Machine Readable Cataloging, Máquina de lectura o catalogación) es un estándar digital internacional de descripción de información bibliográfica creado en el año 1965. Un registro bibliográfico se compone por un conjunto de etiquetas, ocurrencias, tipo de formato de metadato y un del Número de Archivo Maestro. Las ocurrencias (*ocurrence*) almacenan los datos, en específico, de un registro bibliográfico. Las etiquetas (*tag*) son asignaciones numéricas a cada campo u ocurrencia de un registro bibliográfico respectivamente. Los tipos de formatos de metadatos (*format)* son las distintas formas de catalogar un registro bibliográfico y el Número de Archivo Maestro (MFN, Master File Number) es un identificador numérico único de cada registro bibliográfico que permite el acceso a estos para la extracción de sus datos (Picco 2011).


MFN: 1 Format: RAW Last Mfn: 151 Mark Menu Mark This Record

RECORD(1)

Tag	Field/Occurrence
24:	<<Techniques for the measurement of transpiration of individual plants>>
26:	<<^aParis^bUnesco^c-1965>>
30:	<<^ap. 211-224^billus.>>
44:	<<Methodology of plant eco-physiology: proceedings of the Montpellier Symposium>>
50:	<<Incl. bibl.>>
69:	<<Paper on: <plant physiology><plant transpiration><measurement and instruments>>>
70:	<<Magalhaes, A.C.>>
70:	<<Franco, C.M.>>

Figura 1.1: Estructura de un registro bibliográfico en el formato MARC21.

Existen diversos tipos de metadatos que son utilizados en las bibliotecas digitales representados en la tabla 1.1 siguiente (Picco 2011):

Tabla 1.1 Tipología de los metadatos.

Tipos de Metadatos	Descripción	Ejemplos de uso
Administrativos	Metadatos usados para gestionar y administrar recursos de información.	✓ Información sobre adquisición. ✓ Seguimiento de derechos. ✓ Documentación de requerimientos de

		acceso legal. ✓ Información sobre la posición. ✓ Criterios de selección. ✓ Control de versiones.
Descriptivos	Metadatos usados para describir o identificar recursos de información.	✓ Catálogos. ✓ Índices especializados. ✓ Relaciones entre recursos de información. ✓ Anotaciones hechas por usuarios.
Preservación	Metadatos relacionados con la preservación de recursos de información.	✓ Documentación de condiciones físicas de los recursos de información. ✓ Documentación de acciones llevadas a cabo para generar versiones.
Técnicos	Metadatos relacionados con el funcionamiento de los sistemas.	✓ Documentación sobre *hardware* y *software*. ✓ Información sobre formatos, razones de comprensión, rutinas de escalado, etc… ✓ Contraseñas, llaves de encriptación.
Uso	Metadatos relacionados con el nivel de y tipología de uso de los recursos de información.	✓ Muestra de registros. ✓ Seguimiento de uso y de usuarios. ✓ Reusado de contenido. ✓ Información sobre versiones.

Según estos tipos de metadatos, dentro de los sistemas bibliotecarios digitales se hacen uso de los metadatos descriptivos pues lo que se requiere almacenar son aquellos atributos que están relacionados intrínsecamente con el documento y que se pueden diferenciar unos de otros.

1.3.1 Deficiencias del proceso de búsqueda de información en ABCD

En el sistema de ABCD cuando se realizan muchas peticiones concurrentes al catálogo en línea el servidor de bases de datos J-ISIS no queda disponible, porque presenta problemas con las peticiones simultáneas. Esta deficiencia afecta a tres módulos del sistema incluyendo dos fundamentales como: son catalogación y el propio catalogo en línea provocando afectaciones en toda la disponibilidad del sistema en cuestión.

1.3 Herramientas informáticas de indexación de información

En el proceso de búsqueda de información digital se utilizan motores de búsqueda o de recuperación. Con el objetivo de conocer las características que poseen estos motores se realiza el siguiente estudio de sistemas homólogos:

1.3.1 Lucene

Lucene que es un motor de búsqueda de alto desempeño escrito en Java. Está diseñado y desarrollado para que se pueda utilizar prácticamente en cualquier aplicación que requiera búsqueda de texto completo, especialmente aplicaciones multiplataforma. En esencia, es una biblioteca que puede ser invocada desde aplicaciones Java para realizar búsquedas. Esta biblioteca actualmente se encuentra entre los 15 mejores proyectos de código abierto y es uno de los 5 mejores proyectos de apache, con instalaciones en más de 4.000 empresas (Rowe 2015).

1.3.2 SolR

SolR es un servidor de búsqueda basado en Lucene, que provee funcionalidad de más alto nivel, como: APIs[1] para XML[2]/HTTP[3] y JSON[4], resaltar resultados, cache y replicación (Rowe 2015). Se basa en tecnologías de búsqueda de código abierto que proporciona indexación y corrección ortográfica, resaltado de golpes y capacidades avanzadas de análisis. Provee una interfaz gráfica para su administración y administrada por la Apache Software Foundation, Fundación de Software Apache (Rowe 2015).

1.3.3 Elasticsearch

Elasticsearch es un servidor de búsqueda basado en Lucene. Provee un motor de búsqueda de texto completo, distribuido y con capacidad de multi-tenencia con una interfaz web basados en REST (Representational State Transfer, Transferencia de Estado Representacional) y con documentos JSON. Elasticsearch está desarrollado en Java y está publicado como código abierto bajo las condiciones de la licencia Apache (Gormley 2015).

1.3.4 Análisis de las herramientas de indexación

[1] Application Programming Interface, Interfaz de Programación de Aplicaciones
[2] eXtensible Markup Language, Lenguaje de Marcado Extensible
[3] Hypertext Transfer Protocol, Protocolo de Transferencia de Hipertexto)
[4] JavaScript Object Notation, Notación de Objetos de JavaScript

A continuación, se presenta un análisis de las herramientas informáticas de indexación de información estudiadas anteriormente, con el objetivo de conocer si cumplen con las necesidades existentes en el mecanismo de búsqueda de información del sistema ABCD para su utilización en la refactorización de este mecanismo. En la comparación siguiente se utilizaron cuatro criterios de análisis.

Tabla 1.2 Criterio de análisis de herramientas informáticas.

Criterios de análisis	Herramientas informáticas de indexación		
	Lucene	**Elasticsearch**	**SolR**
Tecnología utilizada en el centro CIGED	No	No	Sí
Funcionalidades complejas	No	Sí	Sí
Uso del framework Spring Boot	No	No	Sí
Tecnología web	Sí	Sí	Sí

El análisis anterior permitió concluir que de las herramientas estudiadas, SolR es la que cumple con los criterios definidos, por lo que fue utilizada en el desarrollo de la propuesta de solución como motor de búsqueda.

1.4 Tecnologías para la implementación de la propuesta de solución

En el desarrollo de la propuesta de solución se utilizaron las siguientes tecnologías de implementación:

1.4.1 SolR

SolR como motor de búsqueda de información digital, presentado en el epígrafe 1.3.2.

1.4.2 JSON

JSON es un formato ligero de intercambio de datos. Leerlo y escribirlo es simple para humanos, mientras que para las máquinas es simple interpretarlo y generarlo. Es completamente independiente del lenguaje, pero utiliza convenciones de lenguajes como: C, C++, C#, Java, JavaScript, Perl y Python. Estas propiedades hacen que JSON sea un lenguaje ideal para el intercambio de datos (Jones 2015).

1.4.3 Spring Boot

Spring Boot es un framework (marco de trabajo) de código abierto basado en Spring que se utiliza para el desarrollo de aplicaciones y utiliza Java como lenguaje de programación. Proporciona configuración automática para la funcionalidad de aplicación común a muchas aplicaciones de Spring. Ofrece ayuda en la gestión de la dependencia del proyecto utilizando dependencias de inicio e indicando qué tipo de funcionalidad y de bibliotecas necesita. Contiene una interfaz de línea de comandos que permite escribir aplicaciones completas con sólo código de aplicación (Walls 2016).

1.5 Herramientas para la implementación de la propuesta de solución

En la implementación de la propuesta de solución se utilizarán las siguientes herramientas:

1.5.1 NetBeans 8.0

Netbeans es un Entorno de Desarrollo Integrado (IDE, Integrated Development Environment) para que los programadores puedan escribir, compilar, depurar y ejecutar programas. Está escrito en Java, pero puede servir para otros lenguajes de programación. Es un producto libre y gratuito sin restricciones de uso. Permite el desarrollo rápido y fácil de aplicaciones Java de escritorio, móviles y aplicaciones web, así como aplicaciones HTML5, JavaScript y CSS (Rójas 2015).

1.5.2 J-ISIS Suite v201408251540

J-ISIS es un Sistema Gestor de Bases de Datos que permite la elaboración y administración de bases de datos. J-ISIS no es un sistema de biblioteca integrado (ILS) como ABCD, es una gestión de base de datos no relacional, que utiliza los conceptos ISIS y que es particularmente adecuado para el almacenamiento de información bibliográfica (Dauphin 2015).

1.5.3 Maven v2

Maven v2 se utiliza para administrar el proceso de construcción del proyecto. Sus enfoques rígidos facilitan el seguimiento de las dependencias entre paquetes. Además del uso de una estructura plana "lógica", con paquetes que son hijos de un único padre de nivel superior. Cada paquete se configura por separado, el par sólo se utiliza para establecer algunos parámetros globales y permitir la compilación de todo el proyecto y pruebas. Esto no significa que los propios paquetes necesitan estar en una estructura de directorio plana. En su lugar, se realiza una agrupación por servicio. Por lo tanto, la dirección contiene un subdirectorio para cada proyecto; cada de los que contiene más directorios para cada paquete (Interfaz,

implementación, comunicación, despliegue). Otro hijo del directorio padre contiene todas las bibliotecas (Cooke 2007).

1.6 Metodologías de desarrollo de software

Una metodología de desarrollo de software es una colección de procedimientos, técnicas, herramientas y documentos auxiliares que ayudan a los desarrolladores de software en sus esfuerzos por implementar sistemas informáticos. Está formada por fases, etapas o disciplinas, que guiarán a los desarrolladores a elegir las técnicas más apropiadas en cada momento del proyecto para planificarlo, gestionarlo, controlarlo y evaluarlo. Se aplica dependiendo del contexto de desarrollo, tamaño del proyecto o del equipo de trabajo, cultura organizacional y características del cliente para garantizar un producto de calidad (Balaguera 2015). Para la selección de la metodología se utilizó la técnica de la matriz de Boehm – Turner explicada a continuación.

1.6.1 Aplicación de la técnica de la matriz de Boehm - Turner

La técnica de la matriz de Boehm - Turner plantea 5 criterios fundamentales mediante los que se estará valorando el proyecto estos son: tamaño del equipo, criticidad del producto, dinamismo de los cambios, cultura del equipo y personal con que se cuenta. Cada uno de esos criterios tiene elementos que lo discriminan y por tanto se tienen en cuenta a la hora de seleccionar uno u otro enfoque. Las puntuaciones hacia el centro indican un buen ajuste para un enfoque ágil, mientras que las puntuaciones hacia el exterior sugieren un enfoque más tradicional (Velásquez 2012). A continuación, se muestra en una tabla 1.3, la descripción de los factores de la matriz de Boehm – Turner, con su descripción y los resultados de aplicarlos a la presente investigación.

Tabla 1.3 Factores de la matriz de Boehm - Turner

Factores	Descripción	Valores resultantes
Tamaño (cantidad de integrantes en el equipo de desarrollo).	En presencia de equipos pequeños, los métodos ágiles son más fáciles de introducir, ejecutar y gestionar. Los equipos de menos de 10 se desempeñan mejor con un enfoque ágil dado, el cual: ➢ Facilita la co-localización física de los demás miembros. ➢ Permite la comunicación a través de los debates	1 Integrante

	cara a cara, que pueden apoyar el conocimiento no escrito (tácito) por conversaciones. A medida que crece el tamaño del equipo, si se sigue el principio ágil, se requiere de técnicas adicionales para escalar con eficacia, lo cual demanda más trabajo y habilidad.	
Criticidad (pérdidas por concepto de falla en el sistema).	Se refiere a la consecuencia de un fallo en el sistema. El enfoque ágil es más adecuado para aplicaciones triviales, donde el fallo del sistema resulta en una pérdida de conveniencia (como la pérdida de tiempo si un juego se bloquea o alguna aplicación deja de funcionar); pero no es recomendable para aplicaciones críticas para una misión o para la vida.	Utilidad
Dinamismo (% de probabilidad de cambios).	Responde a la interrogante: ¿cuán dinámico (cambiante) es el proyecto?, ¿qué porcentaje de los requisitos son propensos a cambiar durante el proyecto? Si es probable que cambien como mínimo el 50% de los requerimientos, las puntuaciones indican una metodología ágil.	30 %
Personal (habilidades del equipo de desarrollo).	Para que un proyecto ágil se desarrolle sin problemas, es recomendable una baja proporción de los desarrolladores principiantes (nivel 1) y una alta proporción de intermedios (nivel 2) y expertos (nivel 3). Si el equipo tiene un mayor porcentaje de principiantes (y, por tanto, un menor porcentaje de personal con más experiencia) entonces el enfoque robusto es el más apropiado.	100 % de principiantes.
Cultura (% de prosperidad).	Si el proyecto tiende a prosperar en una cultura donde el equipo se sienta cómodo y motivado al tener muchos grados de libertad, se recomienda una metodología ágil. En cambio, es más adecuada una robusta si prospera en una cultura donde los desarrolladores prefieren tener sus roles definidos por políticas y procedimientos claros.	70%

En la figura 1.2 se muestra la matriz resultante de aplicar los cinco factores descritos por Boehm - Turner. En base al resultado mostrado, se decide considerar solamente metodologías ágiles para el desarrollo de la presente investigación.

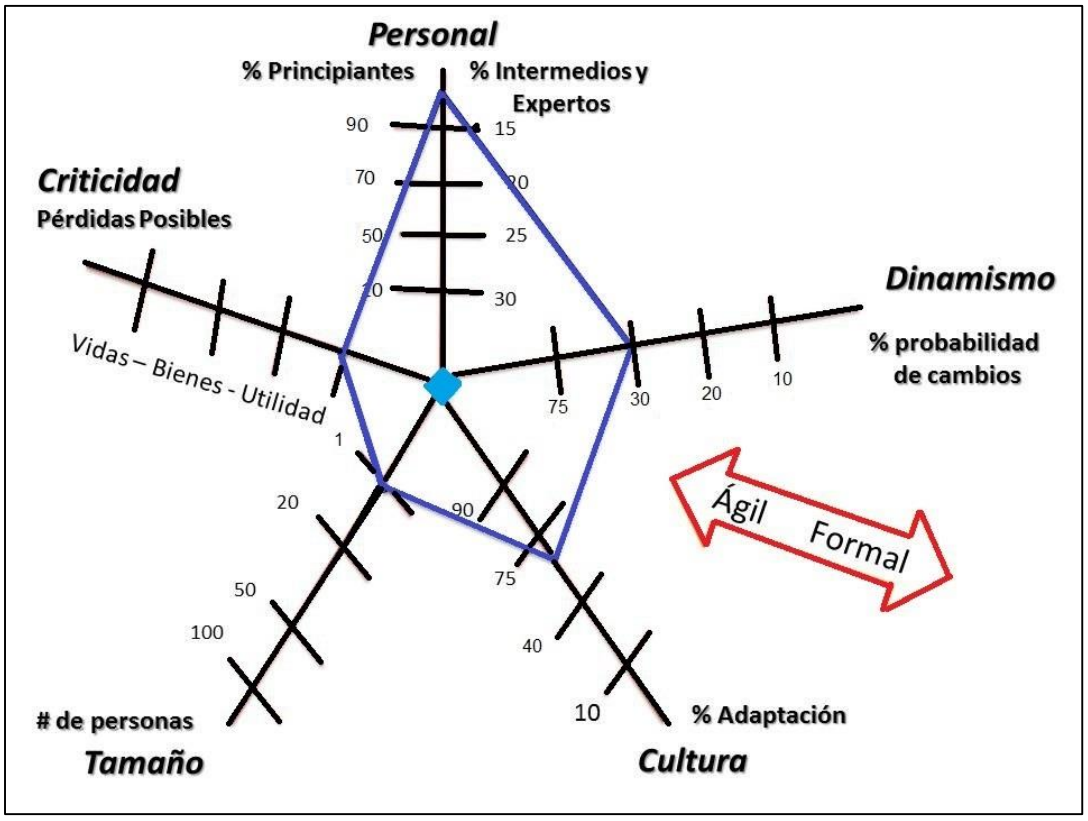


Figura 1.2 Matriz de Boehm -Turner aplicada a la investigación (elaboración propia).

1.6.2 Metodologías ágiles

Las metodologías ágiles surgieron en el 2004 y presentan un desarrollo de software iterativo e incremental, cooperativo y sencillo. Basan su fundamento en la adaptabilidad de los procesos de desarrollo y ponen de relevancia que la capacidad de respuesta a un cambio es más importante que el seguimiento estricto de un plan. Varían en sus prácticas y sus fases, sin embargo, comparten algunas características, tales como: comunicación y reducción de productos intermediarios y de la documentación extensiva (Gómez 2014).

En el desarrollo de la propuesta de solución se utilizó la metodología ágil XP (*Extreme Programming,* Programación extrema) ya que el negocio a informatizar está muy bien definido, el cliente siempre acompañó a la desarrolladora para convenir los requisitos funcionales y las características del sistema facilitando su implementación y validación; el proyecto no es extenso y demanda poca cantidad de productos de trabajo a elaborar en el proceso de desarrollo de la propuesta de solución.

1.6.3 Metodología de desarrollo XP

XP fue creada por Kent Beck a finales de 1980, promueve el trabajo en equipo, el aprendizaje de los desarrolladores y propicia un buen clima de trabajo. Se basa en la retroalimentación continua entre el cliente y el equipo de desarrollo, simplicidad en las soluciones implementadas y flexibilidad para enfrentar los cambios. Está definida especialmente para proyectos con requisitos imprecisos y muy cambiantes y donde existe un alto riesgo técnico. Se caracteriza porque planifica, analiza, diseña y prueba la solución durante todo el desarrollo del sistema. Genera poca documentación lo que hace la entrega del software menos complicada y más satisfactoria tanto para los clientes como para el equipo de proyecto (Pardo, Hurtado y Collazos 2010). La figura 1.4 (Pressman 2010) ilustra el proceso XP y resalta algunas de las ideas y tareas claves que se asocian con cada actividad estructural.

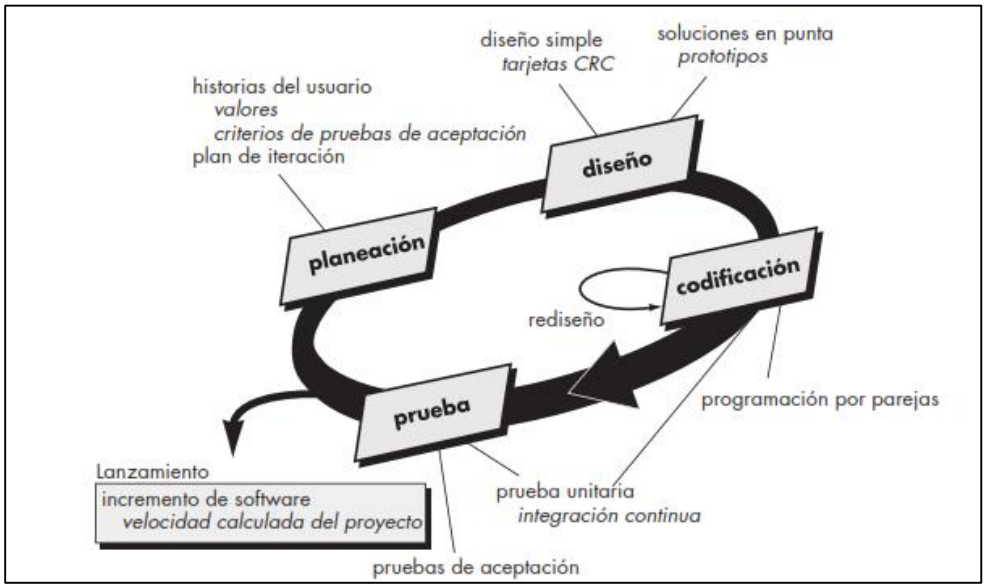


Figura 1.3 Programación Extrema.

1.7 Conclusiones del capítulo

El estudio de los principales conceptos asociados al problema planteado y de los principales referentes teóricos sobre el proceso de búsqueda de información en el catálogo en línea ABCD permitió definir las bases para el desarrollo de la investigación y conocer las características del objeto de estudio. El análisis y comparación de las diferentes herramientas informáticas para la indexación de información posibilitó la selección de SolR como motor de búsqueda a utilizar en la implementación de la propuesta de solución. Se definió además como metodología de desarrollo de software XP y herramientas y tecnologías libres para la refactorización del mecanismo de búsqueda del Catálogo en Línea de ABCD 3.0.

Capítulo 2 Análisis y diseño de la propuesta de la solución

Introducción

En el presente capítulo se describe la propuesta de solución para la creación de una aplicación de búsqueda para el catálogo en línea de ABCD. Se exponen los productos de trabajo obtenido en las fases de Exploración y Planificación que guían el proceso de implementación de la solución y se refleja las restricciones del diseño concebidas para la construcción del sistema informático.

2.1 Descripción de la propuesta de solución

La propuesta solución nombrada SBRI (Sistema para Búsqueda y Recuperación de Información) es una herramienta informática para la búsqueda de información en el servidor de base de datos J-ISIS del sistema ABCD 3.0, que posibilita solucionar los problemas de concurrencia que presenta el mismo. Al iniciar SBRI realiza la indexación en el motor de búsqueda SolR de toda la base de datos del sistema informático J-ISIS, especificándose la operación de indexación y la cantidad de registros a indexar si son necesario cambiarlos.

El proceso es dividido en dos fases, en la primera se realiza la búsqueda que tiene como entrada criterios introducidos por el usuario y como salida los resultados de esta búsqueda en correspondencia con los registros indexados por los metadatos título, autor, materia, ISBN, ISSN y editorial en el servidor SolR. En la segunda fase cuya entrada son los resultados de la búsqueda y la salida son los datos de un registro, se puede seleccionar el registro deseado por el usuario y visualizar toda su información obtenida de la base de datos J-ISIS por su identificador. Permitiendo mejorar la operación de búsqueda ya que no es realizada en toda la base de datos documental sino directamente por su identificador indexado en SolR. En la figura 2.1 se representa la propuesta de solución.

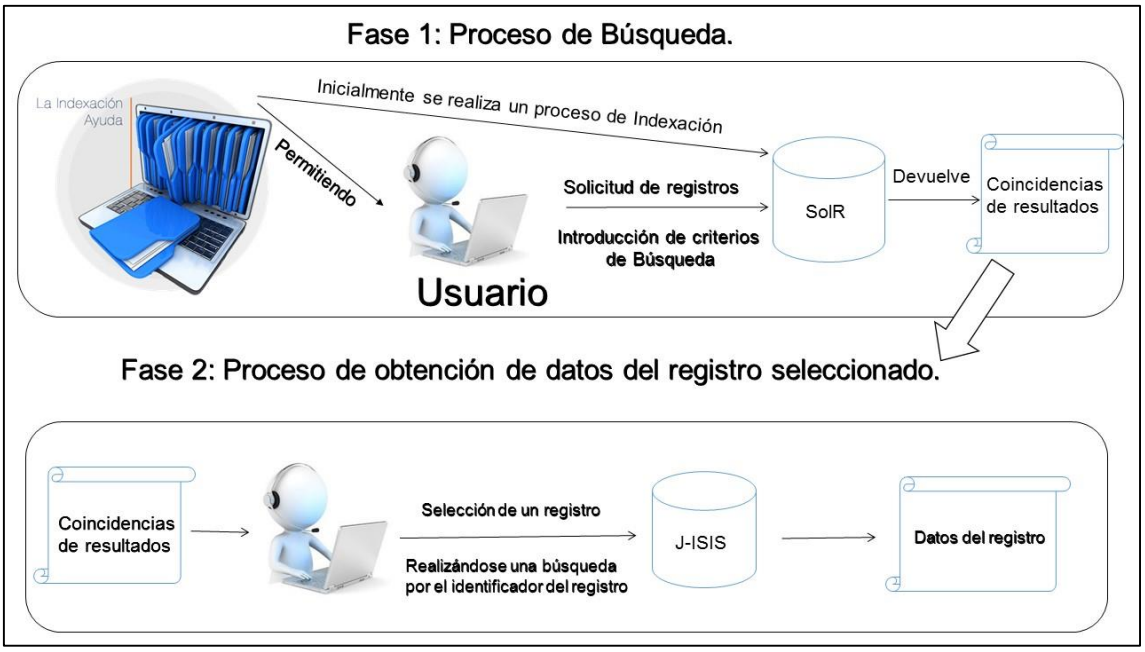


Figura 2.1 Propuesta de solución por fases.

2.2 Exploración

La fase de exploración es la primera que propone XP para comenzar el desarrollo de un producto. En esta los clientes plantean su propuesta al equipo de trabajo. Se definen las historias de usuario, que son la forma de definir los requisitos del sistema a implementar (Letelier 2008). Esta fase dura pocas semanas a pocos meses, dependiendo del tamaño y la familiaridad de los programadores con la tecnología (Rumbaugh 2000).

2.2.1 Lista de funcionalidades

La lista de funcionalidades contiene los requisitos funcionales de la propuesta de solución y su descripción correspondiente. Los mismos son mostrados en la tabla 2.1:

Tabla 2.1 Lista de funcionalidades.

No	Requisitos Funcionales	Descripción
1	Buscar registros indexados por metadatos.	Realiza una búsqueda de registros indexados por los metadatos: título, autor, materia, ISBN, ISSN y editorial en correspondencia con los criterios de búsqueda definidos por el usuario y los muestra en la interfaz del sistema.

2	Mostrar los datos del registro seleccionado.	Muestra toda la información del registro seleccionado por el usuario.
3	Paginar el listado de los resultados de la búsqueda.	Mostrar el listado de los resultados de la búsqueda en distintas páginas.

2.2.2 Características del producto

En la tabla 2.2 se listan los atributos de calidad que se tienen en cuenta en el desarrollo de la propuesta de solución.

No	Características del producto	Descripción
1	Usabilidad	• El idioma de todas las interfaces de la aplicación será español. • El sistema expondrá el menú desde su página principal. • La interfaz debe ser intuitiva para el usuario.
2	Funcionalidad	El sistema realizará las operaciones indicadas por el usuario.
3	Restricciones de software	• Para que la aplicación funcione debe estar instalado en el servidor la máquina virtual de Java 7, y la herramienta SolR en su versión 4.10.2. • Del lado del cliente es necesario tener instalado un navegador web, se recomienda Mozilla Firefox v53.0.2.
4	Restricciones de hardware	• El servidor debe contar al menos con el hardware: 4gb de memoria RAM y un procesador core-i3 de segunda generación.
5	Mantenibilidad	• La implementación del software es uniforme y las funcionalidades son comentadas para facilitar su mantenimiento.

2.2.3 Historias de usuario

Las historias de usuario tienen propósito la descripción de los requisitos funcionales del software que se implementará. Son escritas por el cliente tal y como en ellos ven las necesidades del sistema y descritas con un nivel mayor de detalles por el equipo de desarrollo. Ayudan a conducir el proceso de creación de las pruebas unitarias y de aceptación (empleados

para verificar que las historias de usuario han sido implementadas correctamente) (Valladarez 2016). En la tabla 2.2 se muestra la plantilla de historia de usuario utilizada en el desarrollo de la propuesta de solución.

Tabla 2.2 Plantilla de Historia de usuario.

Historia de usuario
Número: Permite identificar a una historia de usuario.
Nombre Historia de usuario: Describe de manera general a una historia de usuario.
Usuario: Persona que utilizará la funcionalidad del sistema descrita en la historia de usuario.
Iteración asignada: Número de iteración, en que el cliente desea que se implemente una historia de usuario.
Prioridad en negocio (Alta / Media / Baja): Grado de importancia que el cliente asigna a una historia de usuario en la clasificación Alta, Media, Baja.
Puntos estimados: Número de semanas que se necesitará para el desarrollo de una historia de usuario los cuales estarán en el rango del 1 al 3.
Riesgo en desarrollo: Hace referencia a los riesgos presentes en el desarrollo del sistema informático.
Programador responsable: Persona encargada de programar cada historia de usuario.
Descripción: Información detallada de una historia de usuario.
Observaciones: Campo opcional utilizado para aclarar, si es necesario, el requerimiento descrito de una historia de usuario.

Seguidamente se muestran las historias de usuario confeccionadas para la descripción de los requisitos de la propuesta de solución.

Tabla 2.3 Historia de usuario del requisito funcional: Buscar registros indexados por metadatos.

Historia de usuario
Número: 1
Nombre Historia de usuario: Buscar registros indexados por metadatos.
Usuario: Usuarios finales
Iteración asignada: 1.
Prioridad en negocio: Alta

Puntos estimados: 3.
Riesgo en desarrollo: Pérdida de tiempo por fallas eléctricas. Afectaciones en el rendimiento de la desarrolladora debido a la poca experiencia con las tecnologías de desarrollo.
Programador responsable: Lianet Labañino Rivera.
Descripción: Permite realizar una búsqueda de registros indexados por los metadatos: título, autor, materia, ISBN, ISSN, editorial en correspondencia con los criterios de búsqueda definidos por el usuario y los muestra en la interfaz del sistema. Los campos de búsqueda que almacena los criterios introducidos por el usuario son: • Título: es el título de la fuente bibliográfica que se desea buscar. Acepta cualquier carácter. • Autor: Es autor de la fuente bibliográfica que se desea buscar. Admite cualquier carácter. • Materia: Es la materia de la fuente bibliográfica que se desea buscar. Admite cualquier caracter. • ISBN: Es el número estándar internacional de libro de la fuente bibliográfica que se desea buscar. Admite cualquier caracter. • ISSN: Es el número internacional normalizado de publicaciones seriadas de la fuente bibliográfica que se desea buscar. Admite cualquier caracter.
Observaciones: Para que se pueda realizar una búsqueda de registros indexados por metadatos se debe primero ejecutar el servidor de J-ISIS y el de SolR.

Tabla 2.4 Historia de usuario del requisito funcional: Mostrar los datos del registro seleccionado.

Historia de usuario
Número: 2.
Nombre Historia de usuario: Mostrar los datos del registro seleccionado.
Usuario: Usuarios finales
Iteración asignada: 2.

Prioridad en negocio: Alta.
Puntos estimados: 3.
Riesgo en desarrollo: Pérdida de tiempo por fallas eléctricas. Afectaciones en el rendimiento de la desarrolladora debido a la poca experiencia con las tecnologías de desarrollo.
Programador responsable: Lianet Labañino Rivera.
Descripción: Muestra toda la información del registro seleccionado por el usuario.
Observaciones: Para poder realizar el paginado se debe realizar una búsqueda por algún criterio introducido por el usuario.

Tabla 2.5 Historia de usuario del requisito funcional: Paginar el listado de los resultados de la búsqueda.

Historia de usuario
Número: 3.
Nombre Historia de usuario: Paginar el listado de los resultados de la búsqueda.
Usuario: Usuarios finales
Iteración asignada: 3.
Prioridad en negocio: Baja.
Puntos estimados: 1.
Riesgo en desarrollo: Pérdida de tiempo por fallas eléctricas. Afectaciones en el rendimiento de la desarrolladora debido a la poca experiencia con las tecnologías de desarrollo.
Programador responsable: Lianet Labañino Rivera.
Descripción:

Se muestra el listado de los resultados de la búsqueda en distintas páginas.
Observaciones: Para que se muestren los registros debe realizarse una búsqueda y seleccionar del listado de resultados el registro deseado.

2.3 Planificación

El propósito de la planificación es que los clientes y los programadores concuerden en una fecha en la que se desarrollara el sistema en correspondencia con la prioridad establecida en las historias de usuarios elaboradas. Permite establecer un plan de entrega de las versiones del producto así como un plan de iteraciones para estimar el tiempo de duración del proyecto (Letelier 2008).

2.3.1 Plan de iteraciones

En la metodología XP, la creación del sistema se divide en iteraciones. La duración ideal de una iteración está entre una y tres semanas. Para cada una de las iteraciones el cliente establece un conjunto de historias de usuario que serán implementadas en cada iteración del proyecto (Letelier 2008). Al final de cada iteración se realizan las pruebas de aceptación y la aplicación tendrá implementada funcionalidades para dar cumplimiento a los objetivos propuestos. En la tabla 2.6 se presenta el plan de iteraciones definido para el desarrollo de la propuesta de solución.

Tabla 2.6 Tabla de Plan de iteraciones.

Iteración	Orden de las Historias de usuario	Prioridad
1	Buscar registros indexados por metadatos.	Alta
2	Mostrar los datos del registro seleccionado.	Alta
3	Paginar el listado de los resultados de la búsqueda.	Baja

2.3.2 Plan de entregas

El plan de entregas está formado por un cronograma de entregas de las diferentes versiones del producto donde el cliente con el equipo de desarrollo establecen la prioridad de cada historia de usuario. Seguidamente, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. En la tabla 2.7 se muestra las diferentes versiones obtenidas al concluir cada iteración y si ya fue finalizada la historia de usuario implicada en la iteración indicado con una F (Letelier 2008).

Tabla 2.7 Plan de entregas.

Historia de usuario	Iteración 1 Del 3 de Abril Al 21 de Abril	Iteración 2 Del 24 de Abril Al 12 de Mayo	Iteración 3 Del 15 de Mayo Al 19 de Mayo	Sistema finalizado
Buscar registros indexados por metadatos.	V1.0	F	F	F
Mostrar los datos del registro seleccionado.	-	V1.1	F	F
Paginar el listado de los resultados de la búsqueda.	-	-	V1.2	F

El plan de duración de las iteraciones se realizó luego de tener el estimado en semanas la duración de implementación de cada historia de usuario. Además se tuvo en cuenta la prioridad que el cliente y la desarrolladora le asignaron a cada historia de usuario y el nivel de complejidad que estas poseen.

2.4 Diseño

La metodología XP define un diseño “tan simple como sea posible” que permita la descripción de la estructura del sistema para guiar el proceso de desarrollo y posibilite su mantenimiento de forma fácil (Beck 1999).

2.4.1 Diseño arquitectónico

Un diseño arquitectónico define la organización estructural de un sistema, representada por sus componentes, las relaciones entre ellos y los principios que rigen su diseño y evolución. Su objetivo es tener una visión clara del software a construir y muestra las formas en que los diferentes componentes interactúan y se coordinan para alcanzar el objetivo del sistema (Pressman 2010). En el diseño arquitectónico del sistema propuesto se utiliza el patrón Modelo – Vista – Controlador (MVC), el mismo se describe a continuación.

Patrón arquitectónico MVC

Los patrones arquitectónicos son patrones de software que ofrecen soluciones a problemas de arquitectura de software como: adaptabilidad a requerimientos cambiantes, rendimiento, modularidad y acoplamiento. Especifican un conjunto de responsabilidades y recomendaciones para organizar los distintos componentes de un sistema (Pressman 2010).

El patrón MVC separa los datos de la aplicación (modelo), la interfaz de usuario (vista), y la lógica de negocio (controlador) en tres componentes distintos. En la figura 2.3 se presenta la estructura de separación que propone el patrón.

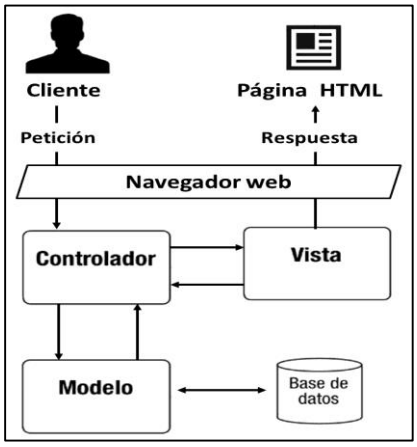


Figura 2.2 Patrón MVC.

La figura 2.4 representa la separación de las clases de la propuesta de solución según el patrón MVC en marco de trabajo Spring Boot.

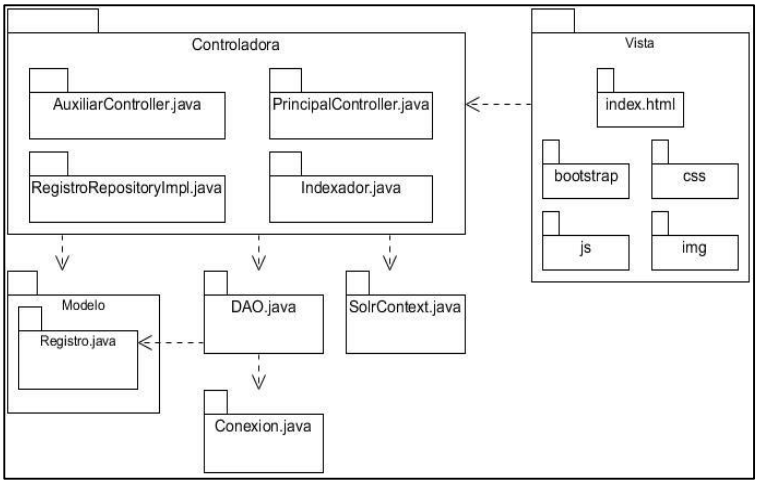


Figura 2.3 Aplicación del patrón MVC en la propuesta de solución.

En el diagrama presentado se muestran las clases controladoras de la propuesta de solución formadas por: AuxiliarController.java, PrincipalController.java, RegistroRepositoryImpl.java e Indexador.java; ellas permiten la gestión de la lógica del negocio sobre el modelo de datos y la vistas relacionadas en el modelo. En las vistas se implementa la página web principal index.html, contiene además la plantilla utilizada en el formulario de esta vista en el paquete bootstrap, así como los estilos y acciones de validación en los paquetes css, js e img; los mismos posibilitan interactuar con el usuario, mostrarle las diversas acciones disponibles y el

estado actual de los datos de la solución. En el paquete modelo se evidencia la clase entidad Registro.java facilitando la obtención de los datos. Se representan además otros componentes para el acceso a los datos en los paquetes: DAO.java, Conexion.java y SolrContext.java.

2.4.2 Patrones de diseño de software

Los patrones de diseño brindan soluciones a una serie de problemas comunes que se presentan en el desarrollo de software. Facilitan la reutilización y la capacidad de expansión del software, reducen la complejidad del código y del acoplamiento y facilitan el mantenimiento. Describe una estructura de comunicación entre componentes para resolver un problema general de diseño dentro de un contexto particular (Montenegro 2012). A continuación se describen los patrones de diseño utilizados en la propuesta de solución.

Patrones GRASP

Los patrones GRASP (General Responsibility Assignment Software Patterns, Patrones Generales de Software para la Asignación de Responsabilidades) presentan una serie de buenas prácticas para el diseño de software. El objetivo de estos patrones es describir los principios fundamentales del diseño de objetos y la asignación de responsabilidades. Los principales patrones GRASP son: Experto, Creador, Alta cohesión, Bajo acoplamiento y Controlador. Cada uno describe un problema, una solución y los beneficios de implementarlos (Pérez 2013) A continuación se relacionan los patrones GRASP utilizados en la solución:

- Patrón Experto: plantea que las responsabilidades se deben asignar al objeto experto en información. Se utiliza para modelar las entidades persistentes de la solución (Larman 2003). En la figura 2.5 se evidencia el uso de este patrón en la clase Registro.java ya que es la única clase que posee las funcionalidades para acceder a la información de los registros.

```java
@SolrDocument(solrCoreName = "marc21")
public class Registro {

    @Id
    @Field(value = "id")
    private long id;
    @Field(value = "fmn")
    private long fmn;
    @Field(value = "titulo")
    private String titulo;
    @Field(value = "autor")
    private String autor;
    @Field(value = "isbn")
    private String isbn;
    @Field(value = "issn")
    private String issn;
    @Field(value = "editorial")
    private String editorial;
    @Field(value = "materia")
    private String materia;

    public Registro(long id, long fmn, String titulo, String autor, String isbn, String issn, String editorial, String materia) {
        this.id = id;
        this.fmn = fmn;
        this.titulo = titulo;
        this.autor = autor;
        this.isbn = isbn;
        this.issn = issn;
        this.editorial = editorial;
        this.materia = materia;
    }

    public long getId() {
        return id;
    }

    public void setId(long id) {
        this.id = id;
    }
```

Figura 2.4 Utilización del patrón Experto en la clase Registro.java.

- Patrón Controlador: aumenta el potencial de reutilización, y asegura que la lógica de la aplicación no se maneja en la capa de interfaz. Un controlador es un objeto que no pertenece a la interfaz de usuario, responsable de recibir o manejar un evento del sistema y define el método para la operación de este (Larman 2003). En la figura 2.6 se muestra el uso de este patrón en la clase PrincipalController.java ya que en ella se implementan las funcionalidades para gestionar la información de la vista principal.

```java
@Controller
public class PrincipalController {

    @RequestMapping("/index")
    public String index() {
        return "index";
    }
}
```

Figura 2.5 Utilización del patrón Controlador en la clase PrincipalController.java.

- Patrón Alta Cohesión: plantea que la información almacenada en una clase debe de ser coherente y debe estar (en la medida de lo posible) relacionada con la clase (Larman 2003). En la figura 2.7 se representa el patrón mencionado en la clase SolrContext.java, que muestra la clase SolrContext la cual es solamente para la conexión con el servicio de SolR.

```
@Configuration
@EnableSolrRepositories(basePackages = {"ciged.abcd.sbri"}, multicoreSupport = true)
public class SolrContext {


    @Bean
    public SolrClient solrClient(){
        return new HttpSolrClient("http://localhost:8983/solr");
    }
```

Figura 2.6 Utilización del patrón Alta cohesión en la clase SolrContext.java.

- Patrón Bajo Acoplamiento: proporciona respuesta a la problemática de soportar bajas dependencias, bajo impacto del cambio e incremento de la reutilización. El acoplamiento es una medida de la fuerza con que un elemento está conectado a, tiene conocimiento de, confía en, otros elementos (Larman 2003). La figura 2.8 representa el patrón antes mencionado en la clase RegistroRepositoryImpl.java ya que utiliza bajas de dependencias para el cumplimiento de su función permitiendo el incremento de la reutilización de código.

```
@Repository
public class RegistroRepositoryImpl implements RegistroRepositoryCustom {

    @Resource
    private SolrTemplate solrTemplate;

    @Override
    public List<Registro> buscar_regis(Registro registro) {
        Criteria criteria = contruir_consulta(registro);
        if (criteria == null) {
            return new ArrayList<>();
        }
        Query query = new SimpleQuery(criteria);
        Page results = solrTemplate.queryForPage(query, Registro.class);
        return results.getContent();
    }
```

Figura 2.7 Utilización del patrón Bajo acoplamiento en la clase RegistroRepositoryImpl.java.

Patrón Domain Model

El patrón *Domain Model* (Modelo de dominio) consiste básicamente en usar las técnicas de diseño orientado a objetos para modelar mediante clases el dominio del problema. Un modelo

de dominio consiste en un conjunto de objetos interconectados, una gran parte de los cuales presentan estado y comportamiento, cada uno de ellos modela un concepto del problema (Fowler 2003). En la figura 2.9 se representa este patrón en la clase DAO.java donde existen anotaciones que permiten la reutilización de código.

```
@Component
public class DAO {

    private ClientDbProxy proxy;

    @Autowired
    private Conexion conexion;

    public DAO() throws DbException {
        this.conexion = new Conexion();
        this.proxy = conexion.getProxy();
    }
```

Figura 2.8 Representación del patrón Domain Model en la clase DAO.java.

Patrón DAO

El patrón DAO (*Data Access Object,* Objeto de Acceso a Datos) es un componente de *software* que ofrece una interfaz común entre una aplicación y uno o más sistemas de almacenamiento de datos, permitiendo que el software cliente se centre en los datos que necesita y se olvide de cómo se realiza el acceso a los datos o cuál es la fuente de almacenamiento (Oracle Corporation 2014). En la figura 2.10 se muestra un ejemplo de este patrón en la clase DAO.java que representa algunos objetos y métodos de la clase utilizados para realizar un puente entre J-ISIS que almacena los registros y la clase controladora que en conjunto con la clase Conexion que permiten conectarse a J-ISIS.

```java
@Component
public class DAO {

    private ClientDbProxy proxy;

    @Autowired
    private Conexion conexion;

    public DAO() throws DbException {
        this.conexion = new Conexion();
        this.proxy = conexion.getProxy();
    }

    public Conexion getConexion() {
        return conexion;
    }

    public void setConexion(Conexion conexion) {
        this.conexion = conexion;
    }

    public FieldDefinitionTable obtenerTablaDefinicionJisis() throws Exception {
        try {
            proxy.getDatabase("UCI", "marc21", Global.DATABASE_DURABILITY_WRITE);
            return proxy.getFieldDefinitionTable();
        } catch (DbException e) {
            throw new Exception(e.getMessage());
        }
    }
```

Figura 2.9 Utilización del patrón DAO en la clase DAO.java.

Patrones GOF

Los patrones de diseño GOF (The Gang of Four, La banda de los cuatro) describen soluciones simples a problemas específicos en el diseño de software orientado a objetos. Define los patrones de diseño como "combinaciones de componentes, casi siempre clases y objetos que resuelven ciertos problemas de diseño comunes" (Braude, 2003). Los patones GOF que se utilizó en la investigación son:

- **Facade** (Fachada): es un objeto "*front-end*" (términos que se refiere a la información de la capa de presentación) que es el único punto de entrada a las funcionalidades de un subsistema; la implementación y otros componentes del subsistema son privados y no pueden ser vistos por los componentes externos. Presenta una única interfaz unificada y es responsable de colaborar con los componentes del subsistema (Larman 2003). Este patrón es representado por la clase de la figura 2.11 RegistroRepositoryCustom.java la cual tiene el método buscar_regis que es implementado en la clase RegistroRepositoryImpl.java mostrada en la figura 2.12.

```java
package ciged.abcd.sbri;

import java.util.List;

public interface RegistroRepositoryCustom {
    public List<Registro> buscar_regis(Registro registro);
}
```

Figura 2.10 Utilización del patrón de Fachada en la clase RegistroRepositoryCustom.java.

```java
@Repository
public class RegistroRepositoryImpl implements RegistroRepositoryCustom {

    @Resource
    private SolrTemplate solrTemplate;

    @Override
    public List<Registro> buscar_regis(Registro registro) {
        Criteria criteria = contruir_consulta(registro);
        if (criteria == null) {
            return new ArrayList<>();
        }
        Pageable a = new PageRequest(0, 10000);
        Query query = new SimpleQuery(criteria, a);
        Page results = solrTemplate.queryForPage(query, Registro.class);
        return results.getContent();
    }
```

Figura 2.11 Utilización del patrón de Fachada en la clase RegistroRepositoryImpl.java.

- **Singleton** (instancia única) es un patrón que tiene como objetivo asegurar que una clase sólo posee una instancia y proporcionar un método de clase único que devuelva esta instancia(Larman 2003). La figura 2.13 representa el uso de este patrón en el método getConexion ya que es el que llama a la clase DAO a ClientDbProxy en cual posee una única instancia o *null* de esta clase y este método reenvía en valor del atributo de clase único devolviéndolo. Si este atributo es *null* se inicializa mediante la creación de una única instancia.

```java
public IConnection getConexion() {
    return (null == conexion) ? abrirConexion() : conexion;
}
```

Figura 2.12 Utilización del patrón Singleton en el método getConexion de la clase Conexión.java.

2.5 Conclusiones del capítulo

En el análisis y diseño de la herramienta informática para la búsqueda de información en J-ISIS del sistema ABCD 3.0 se describió la propuesta de solución que permitió tener una visión general del sistema. Se identificaron 3 funcionalidades y 5 características del producto y se

describieron 3 historias de usuario en la fase de exploración. En la planificación se definió tres iteraciones para el desarrollo de la aplicación en un período de tiempo de 7 semanas. Se elaboró el diseño arquitectónico basado en el patrón MVC y se utilizaron los patrones de diseño GRASP, Modelo de dominio, Objeto de Acceso a Datos y GOF que permitieron guiar el proceso de implementación de la solución.

Capítulo 3 Implementación y pruebas de la propuesta de solución

Introducción

El presente capítulo se enfoca en la implementación del sistema a partir las funcionalidades e historias de usuarios definidas, la planificación establecida para el desarrollo de la solución y las restricciones del diseño elaborado. Contiene las tareas de ingeniería, las tarjetas CRC y los estándares de codificación. Además, se describe la estrategia de pruebas de software utilizada en la validación del sistema y los resultados obtenidos de las mismas.

3.1 Implementación

En la implementación se definen las tareas de ingeniería que guían el proceso de implementación de las historias de usuario, se elaboran las tarjetas CRC para conocer las relaciones entre las clases y se definen los estándares de codificación para garantizar la uniformidad en el código de la propuesta de solución.

3.1.1 Tareas de ingeniería

La metodología XP propone dividir cada historia de usuario en tareas de ingeniería con el objetivo de facilitar la implementación de los programadores. las tareas de ingeniería se vinculan más al desarrollador, ya que permite tener un acercamiento con el código (Letelier 2008). En la tabla 3.1 se muestra la plantilla utilizada para describir las tareas de ingeniería de la propuesta de solución.

Tabla 3.1 Plantilla para las Tareas de Ingeniería.

Tarea de Ingeniería
Número Tarea: Permite identificar a una tarea de ingeniería.
Nro. Historia de Usuario: Número asignado de la historia correspondiente.
Nombre Tarea: Describe de manera general a una tarea de ingeniería.
Tipo de Tarea (Desarrollo / Corrección / Mejora / Otra): Tipo al que corresponde la tarea de ingeniería.
Puntos Estimados: Número de semanas que se necesitará para el desarrollo de una tarea de ingeniería los cuales estarán en el rango del 1 al 3.
Fecha Inicio: Fecha inicial de la creación de la tarea de ingeniería.

Fecha Fin: Final concluida de la tarea de ingeniería.
Programador Responsable: Persona encargada de programar la tarea de ingeniería.
Descripción: Información detallada de la tarea de ingeniería.

A continuación, se detallan las tareas de ingeniería definidas para la Historia de usuario Buscar registros indexados por metadatos. Las tareas de las historias de usuarios 2 y 3 se pueden observar el anexo 1.

Tabla 3.2 Tarea de ingeniería Configurar motor de búsqueda SolR para indexar servicio web creado.

Tarea de Ingeniería
Número Tarea: 1.
Nro. Historia de Usuario: 1.
Nombre Tarea: Configurar el motor de búsqueda SolR para indexar servicio web creado.
Tipo de Tarea: Desarrollo.
Puntos Estimados: 1.
Fecha Inicio: 3/4/2017.
Fecha Fin: 7/4/2017.
Programador Responsable: Lianet Labañino Rivera.
Descripción: Se declara el formato de indexación en el motor de búsqueda a través de la configuración del fichero schema.xml.

Tabla 3.3 Tarea de ingeniería Crear servicio web para la indexación de registros por metadatos.

Tarea de Ingeniería
Número Tarea: 2.
Nro. Historia de Usuario: 1.
Nombre Tarea: Crear servicio web para la indexación de registros por metadatos.
Tipo de Tarea: Desarrollo.

Puntos Estimados: 1.
Fecha Inicio: 10/4/2017.
Fecha Fin: 14/4/2017.
Programador Responsable: Lianet Labañino Rivera.
Descripción: Realizar una indexación de los registros almacenados en el servidor de gestión documental J-ISIS mediante la biblioteca spring.data.solr.

Tabla 3.4 Tarea de ingeniería Implementar la interfaz de usuario para realizar búsquedas sobre los índices en SolR.

Tarea de Ingeniería
Número Tarea: 3.
Nro. Historia de Usuario: 1.
Nombre Tarea: Implementar la interfaz de usuario para realizar búsquedas sobre los índices en SolR.
Tipo de Tarea: Mejora.
Puntos Estimados: 3.
Fecha Inicio: 17/4/2017.
Fecha Fin: 21/4/2017.
Programador Responsable: Lianet Labañino Rivera.
Descripción: Implementar la interfaz gráfica de usuario que permite la introducción de los criterios de búsqueda para realizar la consulta a SolR.

Se describieron 7 tareas de ingeniería que permitieron descomponer las historias de usuario en tareas más simples contribuyendo a garantizar la calidad de la implementación de las funcionalidades del sistema para cumplir con el objetivo de la investigación.

3.1.2 Tarjetas CRC

Las tarjetas CRC (Clases, Responsabilidad, Colaboración) son empleadas para la representación de las clases involucradas en el sistema definiendo las responsabilidades sobre las mismas. El formato físico de las tarjetas CRC facilita la interacción entre el cliente y el equipo de desarrollo de una forma simple y adaptable. Tiene como objetivo obtener un diseño simple y fácil de comprender por parte de los programadores (Joskowicz 2008). En desarrollo de la propuesta de solución se elaboraron 11 tarjetas CRC, un ejemplo de ellas se muestra en la tabla 3.5, las restantes podrán ser consultadas en el anexo 2.

Tabla 3.5 Tarjeta CRC de la clase RegistroRepository.

Nombre de la clase: RegistroRepository	
Responsabilidades: Es una interfaz que se encarga de exponer los métodos implementados en la clase SolrCrudRepository que es un CRUD para la indexación de datos en SolR.	**Colaboradores:** • RegistroRepository • RegistrosRepositoryCustom • SolrCrudRepository • RegistrosRepositoryImpl • Registro

3.1.3 Estándares de codificación

Un estándar de codificación es un conjunto de reglas y normas destinados a establecer uniformidad en el proceso de generación de un código. Son pautas de programación enfocadas a la estructura y apariencia física del código para facilitar su lectura, comprensión y mantenimiento (Osoria 2013).

En el desarrollo de la propuesta de solución se utilizaron los siguientes estándares de codificación:

- Notación de Pascal (Pascal Case): es un estilo de escritura que se utiliza para escribir el nombre de los identificadores, variables o componente de un proyecto colocando en mayúscula la primera letra de cada palabra que forme el nombre del elemento (Takeyas 2015). Esta notación fue empleada para declarar el nombre de las clases. En la figura 3.1 se muestra un fragmento de código en el que se evidencia el uso de este estándar.

```
@RestController
public class AuxiliarController {

    @Autowired
    private RegistroRepository repositorio;

    @Autowired
    private DAO dao;

    @RequestMapping(value = "/api/search", method = RequestMethod.POST)
    @ResponseBody
    public List<Registro> buscar(@RequestBody Registro registro) {
        return this.repositorio.buscar_regis(registro);
    }
```

Notación Pascal Case

Figura 3.1 Utilización de la notación Pascal Case.

- Notación de camello (Camel Case): es un estilo de notación que se emplea para escribir la primera letra de la identificación con minúsculas y la inicial de cada una de las palabras concatenadas se escribe con mayúscula. Existen dos tipos de Camel Case, el estilo Upper Camel Case, que define que la primera letra de cada una de las palabras que identifican a un elemento de un proyecto es mayúscula y el Lower Camel Case cuando la primera letra es minúscula (Takeyas 2015). El estilo de notación utilizado en el nombre de los métodos es el segundo.
- El nombre de las variables se definió con minúscula.
- El código es comentariado para su mejor entendimiento.

En la figura 3.2 se muestra un fragmento de código en el que se evidencia el uso de los estándares Lower Pascal Case, Declaración de variables y Comentarios:

```
// se crea la consulta con los criterios de búsqueda introducido por el usuario
    private Criteria contruirConsulta(Registro registro) {
        Criteria c = null;
        if (!registro.getTitulo().equals("")) {
            if (c == null) {
                c = new Criteria("titulo").contains(registro.getTitulo());
            } else {
                c = c.and(new Criteria("titulo").contains(registro.getTitulo()));
            }
        }
```

Comentarios

Notación Lower Camel Case

Declaración de variables

Figura 3.2 Utilización de los estilos Lower Pascal Case, Declaración de variables y Comentarios.

Los estándares de codificación definidos en la investigación permiten tener un estilo único de codificación que posibilita el estudio y entendimiento del código de la aplicación y garantiza de forma más fácil su mantenimiento.

3.2 Pruebas de software

Las pruebas de software permiten verificar la calidad del producto y su cumplimiento con los requisitos definidos. Su principal objetivo es detectar y solucionar los errores que presente la herramienta desarrollada, y así perfeccionar la solución implementada (Pressman 2010).

3.2.1 Estrategia de prueba definida

Una estrategia de prueba de software debe incluir pruebas de bajo nivel, que son necesarias para verificar los segmentos de código fuente implementados y pruebas de alto nivel, que validan las principales funciones del sistema a partir de los requerimientos del cliente (Pressman 2010). En el desarrollo de la propuesta de solución se realizaron pruebas unitarias y de aceptación.

Pruebas unitarias: son pruebas que utilizan el método de caja blanca donde se identifican errores de entrada o salida de datos y se realizan con el objetivo de detectar errores de implementación en la herramienta desarrollada (Pressman 2010).

Pruebas de aceptación: son generadas a partir de las historias de usuario elegidas para cada iteración donde el cliente verifica que lo que se está probando funcione correctamente. Las mismas son pruebas que utilizan el método de caja negra que se crean a partir de las historias de usuario. El objetivo final de estas es garantizar que los requerimientos han sido cumplidos y que el sistema es aceptable (Pressman 2010).

En el desarrollo de las pruebas se realizaron además pruebas funcionales y de rendimiento. Estas no están definidas en la metodología XP pero se utilizaron para garantizar una validación más detallada de las funcionalidades del sistema.

Pruebas funcionales: tienen como objetivo verificar que el software funciona en correspondencia a lo descrito en sus requisitos (las necesidades del cliente) (Pressman 2010). El método de prueba empleado en la validación fue la Prueba de caja negra y la técnica desarrollada fue la Partición equivalente.

Pruebas de rendimiento consisten en probar el rendimiento del software en tiempo real en su contexto de sistema (todos sus elementos plenamente integrados. Estas pruebas se desarrollan a través de la herramienta JMeter.

3.2.2 Métodos y técnica de pruebas

Los métodos de pruebas utilizados son:

Pruebas de caja blanca: estas pruebas se realizan al código fuente para asegurar que la operación interna se ajuste a las especificaciones. Plantea que, la prueba de caja blanca, en ocasiones llamada prueba de caja de vidrio, es una filosofía de diseño de casos de prueba que usa la estructura de control descrita como parte del diseño a nivel de componentes para derivar casos de prueba. Al usar los métodos de prueba de caja blanca, puede derivar casos de prueba que: garanticen que todas las rutas independientes dentro de un módulo se revisaron al menos una vez, revisen todas las decisiones lógicas en sus lados verdadero y falso, ejecuten todos los bucles en sus fronteras y dentro de sus fronteras operativas y revisen estructuras de datos internas para garantizar su validez (Pressman 2010).

Pruebas de caja negra: estas pruebas permiten demostrar que las funciones del sistema sean operativas, que la entrada se acepta de forma adecuada y que se produce una salida correcta. Las pruebas de caja negra, también llamadas pruebas de comportamiento, se enfocan en los requerimientos funcionales del software; es decir, las técnicas de prueba de caja negra le permiten derivar conjuntos de condiciones de entrada que revisarán por completo todos los requerimientos funcionales para un programa. Intentan encontrar errores en las categorías siguientes: funciones incorrectas o faltantes, errores de interfaz, errores en las estructuras de datos o en el acceso a bases de datos externas, errores de comportamiento o rendimiento y errores de inicialización y terminación (Pressman 2010).

La técnica de prueba utilizada es:

Técnica del camino básico: es una técnica de prueba de caja blanca que permite comprobar el conjunto básico de caminos de ejecución independiente en un componente o programa. Un camino es una vía por la cual procede la ejecución a través de una función desde su inicio hasta el fin (Pressman 2010).

Partición equivalente: es una técnica de prueba de caja negra que divide el campo de entrada de un programa en clases de datos de los que se pueden derivar casos de prueba. La partición equivalente busca obtener casos de prueba ideales que descubran de forma inmediata errores en el sistema. Una clase de equivalencia representa un conjunto de estados válidos o no válidos para condiciones de entrada. Una condición de entrada es un valor numérico específico, un rango de valores, un conjunto de valores relacionados o una condición lógica (Pressman 2010).

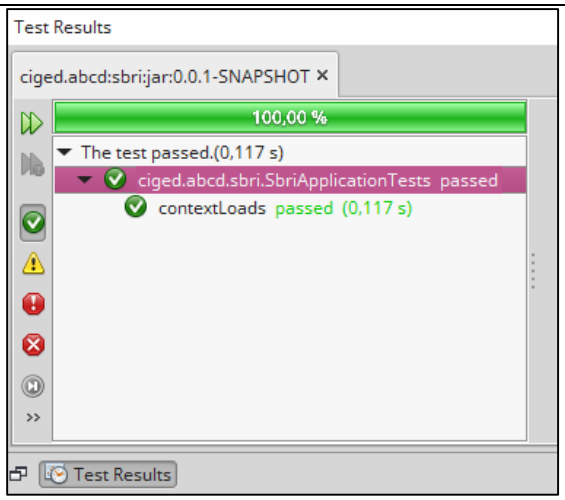
3.2.3 Aplicación y resultados de las pruebas realizadas al sistema

Pruebas unitarias

Las pruebas unitarias se desarrollaron con la herramienta JUnit v4. JUnit es un conjunto de bibliotecas que se utilizan a la hora de programar para realizar las pruebas unitarias de aplicaciones Java. Realiza ejecuciones de las clases para comprobar que el funcionamiento de éstas es totalmente correcto. Ejecuta los Test: en función del valor de entrada se evalúa el valor de retorno, por lo que, si la clase cumple con la especificación indicada dentro de dicha clase, devolverá que la prueba ha sido exitosa, en caso contrario, devolverá el fallo de dicha prueba (Planells 2015). En la tabla 3.6 se muestra la prueba unitaria aplicada al contexto del sistema.

Tabla 3.6 Prueba unitaria realizada de contexto.

Prueba Unitaria		
Nombre Prueba: Contexto.	**Tipo:** Caja blanca.	**Última Ejecución: 16-6-2017**
Ejecutado por: Lianet Labañino Rivera.	**Verificado por:** Leandro Tabares Martín.	
Descripción: Para la ejecución de esta prueba se debe tener la aplicación ejecutándose para que se pueda verificar todos los objetos creados y clase anotadas con las anotaciones de Spring Boot que permiten el correcto funcionamiento del sistema.		
Entrada: No tiene entradas solo las clases tienen que estar anotadas con la anotaciones correspondientes para cada funcionalidad.		
Criterio de Aceptación: Devolver *passed*.		
Resultado:		



Pruebas funcionales

Las pruebas funcionales fueron realizadas por la desarrolladora en tres iteraciones utilizando el método de prueba de caja negra y la técnica de Partición equivalente. Un **caso de prueba** es un conjunto de entradas de pruebas, condiciones de ejecución, resultados esperados desarrollados para cumplir un objetivo en particular o una función esperada. La entidad más simple que siempre es ejecutada como una unidad, desde el comienzo hasta el final (Pressman 2010). A continuación se muestran los casos de pruebas de las funcionalidades:

Tabla 3.7 Casos de prueba Buscar registros indexados por metadatos.

Escenario	Descripción	Texto	Respuesta esperada	Respuesta del sistema
EC 1.1 Realizar una búsqueda por algún criterio.	El usuario realiza una búsqueda llenando el campo de título escribiendo una a.	Válido	Todos los registros que tengan una a en su título.	Muestra los resultados todos los registros que coinciden con el criterio de búsqueda.
EC 1.2 Realizar una búsqueda sin criterios.	El usuario selecciona la opción buscar sin haber especificado algún criterio.	Inválido	No se encontraron resultados.	Muestra el mensaje “No se encontraron resultados”.

Tabla 3.8 Casos de prueba Mostrar los datos del registro seleccionado.

Escenario	Descripción	Texto	Respuesta esperada	Respuesta del sistema
EC 1.1 Seleccionar registro.	El usuario debe seleccionar el registro que desea visualizar su información.	Válido	Datos del registro.	Muestra los datos del registro.

Las pruebas funcionales se realizaron en dos iteraciones, en la primera se encontraron 3 no conformidades, de ellas 1 de funcionalidad y 2 de ortografía, en la segunda iteración no se encontraron no conformidades. En la figura 3.3 se presenta un gráfico con los resultados de las pruebas.

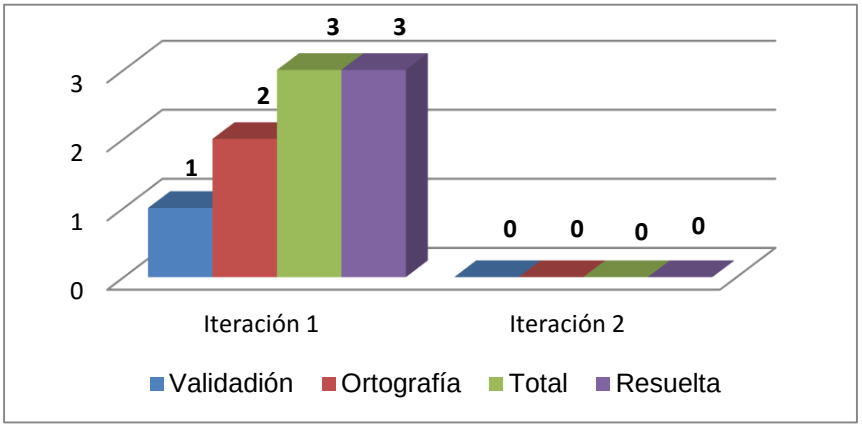


Figura 3.3 Resultados de las pruebas funcionales.

Pruebas de rendimiento

Las pruebas de rendimientos fueron desarrolladas por la autora de la investigación utilizando la herramienta JMeter. A continuación, se describe el procedimiento seguido en la realización de las pruebas y los resultados obtenidos.

Tabla 3.9 Descripción de las pruebas de Rendimiento.

Id del escenario	Escenarios de la sesión	Carga de trabajo	Descripción	Resultados esperados	Resultados de la prueba

EC 1.1	Iteración de acceso a la 1ra página del sistema.	50	El desarrolla-dor debe acceder a la página principal de la aplicación.	215.7 segundos	130.0 segundos
EC 1.2	Interacción de realización de una búsqueda.	50	El desarrollador debe realizar una búsqueda al campo título que diga a.	396.4 segundos	7.2 segundos
EC 1.3	Interacción de seleccionar el registro.	50	El desarrollador debe seleccionar un registro para verlo en detalles.	382.1 segundos	4.6 segundos

En las imágenes que se presentan a continuación se reflejan los resultados de las pruebas realizadas:

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
1 /opac	50	85	71	133	20	139	0,00%	59,0/sec	248,7
2 /opac;jses...	50	44	46	60	16	76	0,00%	64,5/sec	268,5
3 /rwt-resour...	50	3810	3830	3957	3561	3989	0,00%	11,6/sec	10906,6
4 /rwt-resour...	50	37	38	52	21	58	0,00%	131,9/sec	404,2
5 /rwt-resour...	50	182	167	221	84	233	0,00%	101,8/sec	3796,4
6 /rwt-resour...	50	226	276	289	104	294	0,00%	112,9/sec	8218,0
7 /opac	50	32	30	52	5	58	0,00%	125,9/sec	531,2
12 /rwt-reso...	50	57	52	114	6	121	0,00%	126,3/sec	3525,7
19 /rwt-reso...	50	10	6	26	2	29	0,00%	164,5/sec	153,6
22 /rwt-reso...	50	5	3	14	1	23	0,00%	151,1/sec	59,6
21 /rwt-reso...	50	7	5	21	1	33	0,00%	153,4/sec	57,5
8 /rwt-resour...	50	6	5	14	1	23	0,00%	144,5/sec	31,6
10 /rwt-reso...	50	7	4	15	2	23	0,00%	139,3/sec	34,4
9 /rwt-resour...	50	14	11	32	4	33	0,00%	140,8/sec	3713,6
18 /rwt-reso...	50	4	3	9	1	13	0,00%	150,2/sec	65,4
11 /rwt-reso...	50	4	3	8	1	12	0,00%	159,7/sec	34,3
17 /rwt-reso...	50	2	2	7	1	12	0,00%	156,7/sec	66,3
16 /rwt-reso...	50	4	4	8	1	17	0,00%	168,4/sec	150,4
15 /rwt-reso...	50	4	3	8	1	13	0,00%	167,2/sec	64,5
13 /rwt-reso...	50	3	3	7	1	13	0,00%	167,8/sec	68,5
14 /rwt-reso...	50	3	2	6	1	10	0,00%	159,2/sec	66,9
24 /rwt-reso...	50	3	2	5	1	10	0,00%	150,6/sec	61,0
25 /rwt-reso...	50	3	2	7	1	16	0,00%	143,3/sec	77,4
23 /rwt-reso...	50	3	2	5	1	14	0,00%	140,1/sec	54,2
20 /rwt-reso...	50	3	3	5	2	9	0,00%	[illegible]	567,7
Total	1250	182	6	154	1	3989	0,00%	215,7/sec	9756,8

Figura 3.4 Prueba de rendimiento del OPAC a la página principal.

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
1 /rwt-resour...	50	54	11	125	2	134	0,00%	65,9/sec	2
2 /opac?cid=...	50	14	10	30	3	35	0,00%	72,8/sec	33
4 /rwt-resour...	50	4	3	8	2	17	0,00%	67,8/sec	22
3 /opac?cid=...	50	6	6	9	3	12	0,00%	62,2/sec	28
5 /rwt-resour...	50	4	3	8	2	25	0,00%	58,9/sec	17
8 /rwt-resour...	50	4	3	9	2	17	0,00%	56,3/sec	1
6 /rwt-resour...	50	4	3	7	2	16	0,00%	53,3/sec	2
7 /rwt-resour...	50	5	4	8	2	25	0,00%	52,5/sec	4
9 /rwt-resour...	50	5	4	8	1	20	0,00%	51,7/sec	4
10 /rwt-reso...	50	4	3	7	1	26	0,00%	50,9/sec	5
11 /rwt-reso...	50	5	4	12	2	22	0,00%	50,7/sec	1
12 /rwt-reso...	50	5	5	9	2	17	0,00%	50,1/sec	3
14 /rwt-reso...	50	3	3	6	2	22	0,00%	50,7/sec	3
13 /rwt-reso...	50	7	5	16	2	20	0,00%	51,3/sec	8
15 /opac?cid...	50	11	7	30	3	34	0,00%	50,8/sec	23
16 /opac?cid...	50	6	5	11	3	18	0,00%	[illegible]	23
Total	800	9	4	16	1	134	0,0[illegible]%	396,4/3211100	79

Figura 3.5 Prueba de rendimiento del OPAC a la página donde se realiza la búsqueda.

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
42 /rwt-reso...	50	46	8	115	2	118	0,00%	63,0/sec	23,7
43 /	50	0	0	0	0	2	100,00%	67,5/sec	61,7
44 /	50	0	0	0	0	1	100,00%	68,0/sec	62,2
45 /	50	0	0	0	0	0	100,00%	63,1/sec	57,8
46 /rwt-reso...	50	4	4	7	1	11	0,00%	58,3/sec	47,3
47 /opac?cid...	50	8	6	15	2	37	0,00%	56,1/sec	282,2
49 /rwt-reso...	50	5	5	10	2	18	0,00%	55,6/sec	183,5
48 /opac?cid...	50	6	6	11	2	15	0,00%	53,2/sec	267,8
50 /rwt-reso...	50	4	4	9	1	14	0,00%	52,3/sec	53,4
52 /rwt-reso...	50	4	3	7	2	17	0,00%	52,8/sec	14,8
53 /rwt-reso...	50	3	3	7	1	11	0,00%	53,3/sec	15,2
54 /rwt-reso...	50	3	3	5	1	9	0,00%	54,6/sec	16,4
51 /rwt-reso...	50	3	3	4	2	10	0,00%	55,4/sec	16,1
55 /opac?cid...	50	4	4	7	3	12	0,00%	[illegible]	280,0
Total	700	6	3	9	0	118	21,43%	382,1/sec	668,8

Figura 3.6 Prueba de rendimiento del OPAC a la página donde se muestran los detalles del registro.

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
1 /index	50	152	116	270	6	386	0,00%	44,7/sec	186,1
6 /js/principa...	50	185	146	399	31	443	0,00%	55,2/sec	163,7
4 /webjars/b...	50	565	619	892	41	1041	0,00%	35,9/sec	1305,9
2 /webjars/b...	50	259	214	462	24	937	0,00%	33,7/sec	3993,4
5 /webjars/jq...	50	213	213	411	22	471	0,00%	28,3/sec	2373,5
3 /css/estilo....	50	92	61	176	5	435	0,00%	26,3/sec	11,(
7 /img/logo.jpg	50	79	86	129	13	151	0,00%	[illegible]7/sec	9342,3
Total	350	221	141	547	5	1041	0,00%	130,0/sec	9716,1

Figura 3.7 Prueba de rendimiento de la aplicación SBRI en la página principal.

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
24 /api/search	50	6121	6179	6555	5046	6880	0,00%	7,2/sec	3499,
Total	50	6121	6179	6555	5046	6880	0,00%	7,2/sec	3499,

Figura 3.8 Prueba de rendimiento de la aplicación SBRI a la realización de una búsqueda.

Etiqueta	# Muestras	Media	Mediana	Linea de 90%	Mín	Máx	% Error	Rendimiento	Kb/sec
16 /detalles/6	50	9263	9412	10197	5128	10818	0,00%	4,6/sec	26,0
Total	50	9263	9412	10197	5128	10818	0,00%	4,6/sec	26,0

Figura 3.9 Prueba de rendimiento del sistema SBRI para mostrar detalles de un registro.

Luego de la realización de esta prueba se puede afirmar que el sistema SBRI es una mejora a la búsqueda realizada por el OPAC, ya que se refleja en sus tiempos de respuesta, que son menores para 50 usuarios conectados al mismo tiempo. Además, se resuelve el problema de J-ISIS, pues solo se realiza la petición que verdaderamente desea el usuario. Al consumir el servicio SolR y buscar sobre lo indexado permite que los tiempos de respuesta sean satisfactorios y por esto las imágenes anteriores muestra esta comparación de los tiempos de respuesta en OPAC de ABCD v3.0 y en el sistema SBRI en distintos escenarios.

Pruebas de aceptación

Las pruebas se realizaron con el cliente, generadas a partir de las historias de usuario de las funcionalidades del sistema. A continuación, se detallan las distintas pruebas aplicadas.

Tabla 3.10 Prueba de aceptación de búsqueda de registros.

Prueba de Aceptación	
Código: HU1-P1.	**Historia de Usuario:** Buscar registros indexados por metadatos.
Nombre: Búsqueda de registros.	
Descripción: Realizar una búsqueda en SolR de los registros que coincidan con la petición realizada por el usuario en la vista.	
Condiciones de Ejecución: El servicio SolR debe estar ejecutándose.	

Entrada/ Pasos de ejecución: Se realiza una búsqueda en los campos mostrados en la vista.
Resultado Esperado: Se muestra una lista con las coincidencias.
Evaluación de la Prueba: Satisfactoria

Tabla 3.11 Prueba de aceptación Mostrar datos de los registros.

Prueba de Aceptación	
Código: HU2-P2.	**Historia de Usuario:** Mostrar los datos del registro seleccionado.
Nombre: Mostrar datos de los registros.	
Descripción: Realiza una búsqueda en SolR de los registros que coincidan con la petición realizada por el usuario en la vista y luego el usuario le da clic al registro deseado y le sale una ventana mostrando todos los metadatos del formato MARC21.	
Condiciones de Ejecución: El servidor de bases de datos J-ISIS ejecutándose.	
Entrada/ Pasos de ejecución: Se realiza una búsqueda en los campos mostrados en la vista, se da clic en el botón buscar, luego aparece una lista de resultados, se escoge el registro deseado y por último sale en una ventana todos los metadatos del formato MARC21.	
Resultado Esperado: Se muestra una lista con todos los metadatos del formato MARC21.	
Evaluación de la Prueba: Satisfactoria.	

Tabla 3.12 Prueba de aceptación Paginación de resultados.

Prueba de Aceptación	
Código: HU3-P2.	**Historia de Usuario:** Paginar el listado de los resultados de búsqueda.
Nombre: Paginación de resultados.	
Descripción: Realizar un paginado del resultado de las búsquedas.	
Condiciones de Ejecución: El servicio SolR debe estar ejecutándose y se debe realizar una búsqueda.	
Entrada/ Pasos de ejecución: Se realiza una búsqueda en los campos mostrados en la vista y se da clic en el botón buscar.	
Resultado Esperado: Se muestra una lista con las coincidencias en la vista.	
Evaluación de la Prueba: Satisfactoria.	

Las pruebas de aceptación con el cliente arrojaron resultados satisfactorios por lo que la propuesta de solución satisface las necesidades del cliente y es aceptada por el mismo.

3.3 Conclusiones del capítulo

En el desarrollo del capítulo se presentó los productos de trabajo elaborados en la implementación y las pruebas de software aplicadas. La implementación permitió obtener una herramienta web que permite agilizar el proceso el proceso de búsquedas de registros en el catálogo en línea del sistema ABCD 3.0, cumpliendo con los estándares de codificación definidos. Las pruebas unitarias comprobaron que el flujo de trabajo de las funcionalidades es correcto y los caminos se ejecutan satisfactoriamente. Las pruebas funcionales contribuyeron a identificar y corregir 3 no conformidades. Las pruebas de rendimiento demostraron que la propuesta de solución mejora la búsqueda de registros y las pruebas de aceptación evidenciaron que el sistema desarrollado satisface las necesidades del cliente.

Conclusiones generales

La investigación realizada cumple con sus objetivos planteados mediante el desarrollo de la herramienta web para la refactorización del mecanismo de búsqueda del catálogo en línea ABCD 3.0 y se arriba a las siguientes conclusiones:

- El análisis de los referentes teóricos y de los sistemas informáticos estudiados evidenció la necesidad de desarrollar una herramienta para agilizar el proceso de de búsquedas de registros en el catálogo en línea del sistema ABCD 3.0.

- Se obtuvo una herramienta web que mejora la búsqueda de registros en el servidor J-ISIS cumpliendo con los requisitos definidos por el cliente.

- La validación de la investigación se realizó a partir de la aplicación de técnicas, métricas y pruebas que garantizan el correcto funcionamiento de la aplicación y demostraron la satisfacción del cliente hacia el sistema desarrollado.

Recomendaciones

Una vez concluido el desarrollo del sistema informático para la refactorización del mecanismo de búsqueda en el Catálogo en Línea disminuyendo los tiempos en la recuperación de información se recomienda:

- Extender la herramienta al idioma inglés para que pueda ser utilizada por la comunidad internacional.

Referencias bibliográficas

BECK, K., 1999. *Extreme Programming Explained*. S.l.: s.n. ISBN 0-201-61641-6.

BORDIGNON, F., 2007. *Recuperación de información: un área de investigación en crecimiento.* [en línea]. 2007. S.l.: s.n. Disponible en: www.redalyc.org/articulo.oa?id=181414865002.

COOKE, A., 2007. *Spring, Mule, Maven: Lightweight SOA with Java (2)* [en línea]. 2007. S.l.: s.n. Disponible en: http://www.acooke.org/impl.pdf.

DAUPHIN, J.-C., 2015. *J-ISIS Reference Manual*. 23 marzo 2015. S.l.: s.n.

FOWLER, M., 2003. Patterns of Enterprise Application Architecture The Addison-Wesley Signature Series: Amazon.es: MartinFowler: Libros en idiomas extranjeros. [en línea]. [Consulta: 31 mayo 2017]. Disponible en: https://www.amazon.es/Enterprise-Application-Architecture-Addison-Wesley-Signature/dp/0321127420.

GORMLEY, C., 2015. *Elasticsearch: The Definitive Guide* [en línea]. 1ra Edición. S.l.: s.n. [Consulta: 17 junio 2017]. Disponible en: https://books.google.com.cu/books?hl=es&lr=&id=d19aBgAAQBAJ&oi=fnd&pg=PR3&dq=elasticsearch+&ots=Nyl_vPEtcI&sig=Hg74E8z7ts6cv4eNCvao7o0EyBk&redir_esc=y#v=onepage&q&f=false.

JONES, M., 2015. JSON Web Encryption. [en línea]. [Consulta: 4 abril 2017]. Disponible en: https://www.rfc-editor.org/rfc/pdfrfc/rfc7516.txt.pdf.

LARMAN, C., 2003. *UML y Patrones. Introducción al análisis y diseño orientado a objeto* [en línea]. 2003. S.l.: s.n. Disponible en: http://s3.amazonaws.com/academia.edu.documents/32421917/PREVIEW-LIBRO-9788483229279.pdf?AWSAccessKeyId=AKIAIWOWYYGZ2Y53UL3A&Expires=1496251314&Signature=ogsmUxjfGIWCGVUAAdeM3C%2BiPO0%3D&response-content-disposition=inline%3B%20filename%3DUML_y_patrones.pdf.

LETELIER, P., 2008. *Métodologías ágiles para el desarrollo de software:eXtreme Programming (XP)*. S.l.: s.n.

MARTÍNEZ, J.L., 2012. *Método para la Caracterización e Indexacion de Contenidos en la Web a Partir de Roles en un Dominio de Interés Específico". Centro de Investigación y de Estudios Avanzados* [en línea]. México: Instituto Politecnico Nacional. Disponible en: http://www.tamps.cinvestav.mx/defensa_2012_5.

MILENA, M., CÁRDENAS, L. y PARRA, Z., 2008. *Estudio sobre la visualización de la información bibliográfica en el opac del sistema de bibliotecas de la Universidad de los Andes* [en línea]. S.l.: s.n. [Consulta: 18 enero 2017]. Disponible en: https://repository.javeriana.edu.co/handle/10554/5321.

MONTENEGRO, I.J., 2012. *Uso de patrones de diseño de software: un enfoque práctico* [en línea]. S.l.: Revista Semestral de la Universidad de Costa Rica. ISBN 1409-2441. Disponible en: http://revistas.ucr.ac.cr/index.php/ingenieria/article/download/8220/pdf_7.

ORACLE CORPORATION, 2014. Core J2EE Patterns - Data Access Object. [en línea]. [Consulta: 31 mayo 2017]. Disponible en: http://www.oracle.com/technetwork/java/dataaccessobject-138824.html.

OSORIA, D.P., 2013. *Módulo para la extracción y representación de los metadatos en el Sistema de Gestión Documental de audio y video digitales TeVeo Plus V1.0* [en línea]. S.l.: Universidad de las Ciencias Informáticas. Disponible en: http://repositorio_institucional.uci.cu/jspui/bitstream/ident/8302/1/TD_06469_13.pdf.

PÉREZ, S.L.P., 2013. *Fundamentos de Ingeniería de Software. Patrones de diseño.* [en línea]. 2013. S.l.: Universidad Autónoma de México. Disponible en: http://computacion.cs.cinvestav.mx/~sperez/cursos/fis/PatronesDiseno.pdf.

PICCO, P., 2011. *Manual de catalogación automatizada*. 2011. S.l.: s.n.

PINTO, M., 2004. Búsqueda y Recuperación de Información. [en línea]. [Consulta: 22 marzo 2017]. Disponible en: http://www.mariapinto.es/e-coms/busqueda-y-recuperacion-de-informacion/.

PLANELLS, M.P., 2015. *Una experiencia aplicando Test-Driven Development (TDD) usando una herramienta JUnit* [en línea]. 2015. S.l.: s.n. Disponible en: https://riunet.upv.es/bitstream/handle/10251/76346/PE%c3%91ARROCHA%20-%20Una%20experiencia%20aplicando%20Test-Driven%20Development%20%28TDD%29%20usando%20una%20herramienta%20JUnit.pdf?sequence=2&isAllowed=y.

PRESSMAN, R.S., 2010. *Ingeniería del software UN ENFOQUE PRÁCTICO*. SÉPTIMA EDICIÓN. S.l.: s.n.

REAL ACADEMIA ESPAÑOLA, 2017. Diccionario de la Real Academia Española. [en línea]. Disponible en: http://dle.rae.es/?id=LNTIjYS.

RÓJAS, Y.I., 2015. *Módulo para la adaptación de contenido Modle*. 2015. S.l.: s.n.

ROWE, S., 2015. Apache Solr Reference Guide - Apache Solr Reference Guide - Apache Software Foundation. [en línea]. [Consulta: 18 enero 2017]. Disponible en: https://cwiki.apache.org/confluence/display/solr/Apache+Solr+Reference+Guide.

RUMBAUGH, J., 2000. *El Proceso Unificado de Desarrollo de Software*. 2000. S.l.: s.n.

TAKEYAS, B.L., 2015. *Nomenclatura sugerida para identificar los componentes de un proyecto* [en línea]. 2015. S.l.: s.n. Disponible en: http://www.itnuevolaredo.edu.mx/maestros/sis_com/takeyas/Apuntes/POO/Apuntes/02.-%20NomenclaturaComponentesProyecto.pdf.

VALLADAREZ, S., 2016. *METODOLOGIA ÁGIL DE DESARROLLO DE SOFTWARE PROGRAMACION EXTREMA.* 2016. S.l.: s.n.

WALLS, C., 2016. *Spring Boot in action*. S.l.: s.n.

Anexos

Anexo 1

Tarea de ingeniería de Buscar un registro en J-ISIS por su identificador.

Tarea de Ingeniería
Número Tarea: 4.
Nro. Historia de Usuario: 2.
Nombre Tarea: Buscar un registro en J-ISIS por su identificador.
Tipo de Tarea: Desarrollo.
Puntos Estimados: 2.
Fecha Inicio: 24/4/2017.
Fecha Fin: 5/5/2017.
Programador Responsable: Lianet Labañino Rivera.
Descripción: Se realiza una búsqueda del registro seleccionado por el usuario en el servidor de gestión documental J-ISIS por el identificador del registro indexado.

Tarea de ingeniería Mostrar los datos del registro seleccionado por el usuario

Tarea de Ingeniería
Número Tarea: 5.
Nro. Historia de Usuario: 2.
Nombre Tarea Mostrar los datos del registro seleccionado por el usuario.
Tipo de Tarea: Desarrollo.
Puntos Estimados: 1.
Fecha Inicio: 8/5/2017.
Fecha Fin: 12/5/2017.

Programador Responsable: Lianet Labañino Rivera.
Descripción: Implementar una interfaz gráfica para mostrarle al usuario los datos del registro seleccionado.

Tarea de ingeniería: Paginación de resultados.

Tarea de Ingeniería
Número Tarea: 6.
Nro. Historia de Usuario: 3.
Nombre Tarea: Paginación de resultados.
Tipo de Tarea: Desarrollo.
Puntos Estimados: 1.
Fecha Inicio: 15/5/2017.
Fecha Fin: 17/5/2017.
Programador Responsable: Lianet Labañino Rivera.
Descripción: Implementar la paginación de los resultados de la búsqueda.

Tarea de ingeniaría Implementar la interfaz gráfica de la paginación.

Tarea de Ingeniería
Número Tarea: 7.
Nro. Historia de Usuario: 3.
Nombre Tarea: Implementar la interfaz gráfica de la paginación.
Tipo de Tarea: Desarrollo.
Puntos Estimados: 1.
Fecha Inicio: 18/5/2017.
Fecha Fin: 19/5/2017.

Programador Responsable: Lianet Labañino Rivera.
Descripción: Se implementar la interfaz gráfica de usuario que visualiza la paginación de registros.

Anexo 2

Tarjeta CRC de la clase DAO.

Nombre de la clase: DAO.	
Responsabilidades:	**Colaboradores:** • Registro • Conexión

Tarjeta CRC de la clase Indexador.

Nombre de la clase: Indexador	
Responsabilidades: Obtiene y registra los datos que se van a indexar.	**Colaboradores:** • DAO • RegistroRepository

Tarjeta CRC de la clase PrincipalController.

Nombre de la clase: PrincipalController	
Responsabilidades: Recibe una petición y muestra la vista principal del sistema.	**Colaboradores:** • springboot-starter-web

Tarjeta CRC de la clase Registro.

Nombre de la clase: Registro	
Responsabilidades: Es la clase entidad que tiene los metadatos a indexar en SolR.	**Colaboradores: -**

Tarjeta CRC de la clase RegistroRepositoryCustom.

Nombre de la clase: RegistroRepositoryCustom	
Responsabilidades: Es una interfaz que expone los métodos de la clase RegistrosRepositoryImpl.	**Colaboradores:** • RegistrosRepositoryImpl • Registro

Tarjeta CRC de la clase RegistroRepositoryImpl.

Nombre de la clase: RegistroRepositoryImpl	
Responsabilidades: Implementa los métodos de la clase interfaz RegistrosRepositoryCustom.	**Colaboradores:** • Registro

Tarjeta CRC de la clase SbriApplication.

Nombre de la clase: SbriApplication	
Responsabilidades: Es la clase encargada de ejecutar la aplicación.	**Colaboradores:** • Indexador

Tarjeta CRC de la clase SolrContex

Nombre de la clase: SolrContex	
Responsabilidades: Clase que permite la conexión con el servicio SolR.	**Colaboradores:** • SolrClient (springboot-data-solr) • HttpSolrClient (springboot-data-solr) • SolrOperations (springboot-data-solr) • SolrTemplate (springboot-data-solr)

Tarjeta CRC de la clase AuxiliarController.

Nombre de la clase: Auxiliar Controller.	
Responsabilidades: La clase que recibe las peticiones AJAX.	**Colaboradores:** RegistroRepository

Tarjeta CRC de la clase Conexion.

Nombre de la clase: Conexion.	
Responsabilidades: Esta clase realiza la conexión del gestor de bases de datos J-ISIS para obtener los registros.	**Colaboradores:** La clase DAO , la Registro, SbriApplication.

Anexo 3

Formulario de Entrevista.

Formulario de Entrevista	
1	¿Cuántos metadatos considera usted que debe indexar el motor de búsqueda?
2	¿Qué metadatos considera que muestre la interfaz de la aplicación?
3	¿Cuáles son los principales metadatos de los registros en J-ISIS del formato MARC21?