Universidad de las Ciencias Informáticas

Facultad 2

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

Título: Herramienta para la migración entre formatos de registros bibliográficos almacenados en J-ISIS.

Autores:

Mahel Hernández Contreras

Jessica Belén Fernández Fundora

Tutor: Ing. Leandro Tabares Martín

Centro: CIGED.

La Habana, junio 2017

Si lo puedes soñar, lo puedes hacer.

Walt Disney

Declaración de Autoría:

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo. Para que así conste firmamos la presente a los __ días del mes_____de _____ del_____.

_____ _____

Jessica Belén Fernández Fundora Mahel Hernández Contreras

Firma del autor Firma del autor

_____

Ing.Leandro Tabares Martin

Firma del tutor

Datos del Contacto

Datos del Autor:

Nombre: Mahel Hernández Contreras.

Correo electrónico: mcontreras@estudiantes.uci.cu

Datos del Autor:

Nombre: Jessica Fernández Fundora.

Correo electrónico: jbfernandez@estudiantes.uci.cu

Datos del Tutor:

Nombre: Leandro Tabares Martín.

Correo electrónico: ltmartin@.uci.cu

Agradecimientos de Mahel

Han sido cinco años de mucho sacrificio y trabajo, primero que todo quiero dar gracias a Dios que me ha permitido llegar hasta aquí y ver el resultado de una larga carrera. Le doy unas infinitas gracias a mi mamà Sobeida por darme todo, por ser la mejor madre del mundo y siempre apoyarme en los buenos y malos momentos. Gracias a mi hermana Mahelis por todo el ánimo que me ha dado, por ser siempre mi consejera en mis peores momentos, por ser más que una hermana, por aguantarme siempre mis pesadeces. No puedo dejar de agradecer a mi prima Marisol que es una segunda madre para mi, gracias a ti por tus consejos, por tu ayuda, nunca se me olvidará como me ayudaste cuando muchas personas me dieron la espalda, gracias por ser una madre más para mi. Muchas gracias a todos mis amigos de la universidad por estar siempre apoyandome cuando salía mal en una prueba, cuando estaba triste, gracias a todos, en especial Yordan, Armando, Beatriz, en fin todos, muchas gracias. Gracias a Leandro mi tutor por hacer de una forma u otra que aprenda porque se que si me exigía es porque quería que aprendiera. Gracias a todas las personas que siempre confiaron y creyeron en mi. Gracias a esta Universidad por haberme permitido ingresar aquí hace cinco años atrás, y permitir que aprendiera todo lo que se, a esta universidad le debo mucho, gran parte de lo que soy se lo debo. Gracias al tribunal y al oponente, en fin muchas gracias a todos.

Agradecimientos de Jessica

A mi Dary por ser mi todo, mi ejemplo, mi guia, creer en mi y por todos los sacrificios cada año para lograr graduarme. A mi madre por su apoyo y confiar en mi siempre.

A Suárez por su amor, paciencia y apoyo incondicional.

A mi abuelo donde quiera que estes ya tienes una nieta universitaria.

A mis tatas Surlay y la Nena por estar siempre a mi lado y ayudarme en todo.

A mis tías Odalys y Sandra, mis primas Nathaly y Ailin, a mi hermana, mi sobrino, por estar presentes.

A Calvin por parecer siempre que nada le preocupa pero atento en todo.

A todos los presentes, los que creyeron, nunca me abandonaron y confiaron en mi.

Resumen

Con el surgimiento de nuevos formatos bibliográficos resulta necesario migrar los registros creados a las nuevas estructuras para garantizar su compatibilidad con los nuevos estándares. Con este propósito se desarrollan herramientas a la medida, pero tiene la desventaja de tener que implementar una herramienta nueva cada vez que surge un formato bibliográfico.

El presente trabajo tiene como objetivo desarrollar una herramienta para la migración entre formatos de registros bibliográficos almacenados en jisis. Para la realización del mismo se utilizó los requisitos definidos por el cliente. Se realiza una descripción de las herramientas, lenguaje de programación, tecnologías y metodología utilizada para el diseño y la implementación de la herramienta para la migración entre formatos de registros bibliográficos almacenados en jisis facilitando la descripción del documento. El sistema informático se desarrolló mediante el lenguaje de programación Java, el framework Spring, Eclipse como IDE, y como metodología XP. La aplicación de las pruebas posibilitó la validación de las funcionalidades de la herramienta, obteniendo asi una herramienta de fácil uso para el usuario.

Palabras Claves: bases de datos, formato bibliográfico, registro bibliográfico.

Índice de Contenido

Índice de Tabla

Índice de Figuras

Introducción

A lo largo de la historia los libros han sido el recurso más utilizado por el hombre para plasmar sus ideas con la intención de trascender en el tiempo y que las futuras generaciones puedan aprender de sus experiencias. Múltiples son los formatos bibliográficos utilizados en la actualidad para la descripción formal de los registros bibliográficos que son un conjunto de elementos informativos, organizados conforme a unas normas.

El uso de técnicas de automatización relacionadas con formatos bibliográficos fue desde sus inicios una propuesta novedosa para los sectores bibliotecarios y por lo tanto el predominio de dichos formatos en la automatización de registros, en el diseño de bases de datos y en el intercambio de información bibliográfica ha sido determinante para los medios bibliotecarios.

Con el surgimiento de nuevos formatos bibliográficos resulta necesario migrar los registros creados a las nuevas estructuras para garantizar su compatibilidad con los nuevos estándares. Con este propósito se desarrollan herramientas a la medida, pero esto tiene la desventaja de tener que implementar una herramienta nueva cada vez que surge un formato bibliográfico.

En la Universidad de las Ciencias Informáticas se encuentra el Centro de Informatización de la Gestión Documental (CIGED), un centro dedicado al desarrollo de sistemas y servicios informáticos integrales de alta calidad, en el mismo existe un sistema de gestión bibliotecaria nombrado Automatización de Bibliotecas y Centros de Documentación(ABCD). ABCD sigue la recomendación de la UNESCO sobre la utilización de bases de datos ISIS para la gestión de información bibliográfica. J-ISIS presenta una arquitectura cliente-servidor además de ser implementado completamente en Java. Entre bases de datos ISIS es posible intercambiar datos, por lo que resulta posible exportar registros almacenados en otra base de datos ISIS en J-ISIS, existiendo un componente de conexión a base de datos J-ISIS.

En el sitema de gestión bibliotecaria ABCD no existe una herramienta para la migración de formatos bibliográficos. Actualmente para almacenar datos en el sistema tiene establecido formatos bibliográficos adoptados internacionalmente. ABCD maneja registros en el formato MARC 21 teniendo en cuenta para en un futuro manejar otros formatos, pues ocurre que en muchas ocasiones cuando se va a implantar ABCD en un cliente, los datos están en otro formato bibliográfico, por lo que es necesario hacer la migración de los mismos.

Por las razones expuestas anteriormente se identifica como **problema a resolver**: ¿Cómo contribuir a la migración de registros bibliográficos almacenados en J-ISIS a diferentes formatos bibliográficos?

En este sentido el **objetivo general** que se propone alcanzar es: Desarrollar una herramienta que permita la migración entre formatos bibliográficos de registros almacenados en J-ISIS.
Se define como **objeto de estudio:** la migración entre formatos bibliográficos centrándose en el **campo de acción** la migración entre formatos bibliográficos de registros almacenados en J-ISIS.

Como **objetivos específicos** se identificaron:

1- Definir los fundamentos teóricos y metodológicos relevantes, así como las tecnologías asociadas para aplicarlas al sistema de migración entre formatos bibliográficos a desarrollar.
2- Describir la herramienta a partir de la elaboración de los artefactos definidos por la metodología seleccionada.
3- Desarrollar y realizar pruebas a la herramienta.

Para dar cumplimiento al objetivo general se definieron las siguientes **Tareas de Investigación:**

1- Análisis de los principales conceptos asociados a los formatos bibliográficos con el fin de obtener la base teórica necesaria para identificar la equivalencia entre ellos.
2- Estudio de tecnologías asociadas al gestor de bases de datos J-ISIS para determinar cómo gestionar los registros almacenados en el mismo.
3- Estudio de las tecnologías necesarias para la implementación de la herramienta a desarrollar.
4- Creación de un mecanismo genérico para la migración entre diferentes formatos bibliográficos.
5- Elaboración de los artefactos establecidos por la metodología para la descripción de la aplicación.
6- Realización de pruebas a la aplicación para verificar su correcto funcionamiento.

Métodos teóricos de Investigación:

- ✓ **Análisis y Síntesis:** Se utiliza 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.
- ✓ **Histórico-Lógico:** Se utiliza para analizar la evolución del proceso de migración, su perfeccionamiento a través del desarrollo de formatos

bibliográficos, realizando un estado crítico de trabajos anteriores para utilizarlos como punto de referencia y comparación de los resultados alcanzados.

- ✓ **Modelación:** Se utiliza para describir el funcionamiento de la aplicación a través de diagramas para una mejor comprensión de la misma.
- ✓ **Inductivo-Deductivo:** Se utiliza para adaptar a la aplicación desarrollada el conocimiento de otros desarrolladores con soluciones que utilizan mecanismos similares.
- ✓ **Sistémico:** Se utiliza para la modelación de la aplicación mediante la determinación de sus componentes, así como las relaciones entre ellos.

Métodos empíricos de Investigación:

- **Entrevista:** se empleó para conocer las necesidades del cliente, así como para recopilar información sobre las limitaciones de la aplicación para poder definir los requisitos y características de la solución propuesta.

Resultados Esperados

Con el desarrollo del sistema se pretende la:

- Creación de equivalencias entre los formatos bibliográficos MARC 21 y CEPAL.
- Creación de un mecanismo que permita migrar registros de un formato bibliográfico a otro.
- Implementación de una herramienta que utilice el mecanismo creado para la migración entre formatos de registros bibliográficos almacenados en el servidor de bases de datos J-ISIS.

Para una mejor comprensión el presente trabajo consta de introducción, tres capítulos, conclusiones, recomendaciones, referencias bibliográficas y anexos. El documento está estructurado de la siguiente forma:

Estructuración por capítulos

Capítulo 1: "Fundamentos teóricos": *S*e describen los términos más importantes asociados a la migración de formatos bibliográficos, así como elementos esenciales de la aplicación para realizar dicho proceso. Además, se realiza el estudio de los conceptos fundamentales relacionados con el tema en cuestión. Al final, se define la metodología para guiar el desarrollo de la investigación, así como las tecnologías y herramientas que serán utilizadas para la construcción de la solución.

Capítulo 2: "Propuesta de Solución": *S*e hace una descripción general de la solución propuesta, así como la definición de los requisitos funcionales y no funcionales. También se presenta la arquitectura con la que se desarrolló el mismo, y se hace referencia a los patrones de diseño empleados en la solución. Además, se muestran los artefactos definidos por la metodología de desarrollo de software seleccionada y se describe la construcción de la aplicación, donde se explican los aspectos principales de la implementación.

Capítulo 3. "Implementación y Pruebas": En él se describe el proceso de implementación de la propuesta de solución, *así como la* ejecución de pruebas sobre dicha aplicación en busca de errores relacionados con su funcionalidad, además de los principales resultados obtenidos en la etapa de pruebas, para garantizar su correcto funcionamiento y el cumplimiento con los requisitos definidos por el cliente.

Capítulo 1: Fundamentos teóricos.

1.1 Introducción

En el presente capítulo se realiza un estudio de los conceptos básicos asociados a la migración de formatos bibliográficos. De igual forma, se muestra una descripción de la herramienta, tecnologías, metodologías y lenguaje de programación utilizadas en el diseño e implementación de la herramienta para la migración entre formatos de registros bibliográficos almacenados en J-ISIS.

1.2 Conceptos asociados

1.2.1 Migración de Datos

La migración de datos consiste en la transferencia de datos de un sistema a otro y suele tener lugar en momentos de transición provocados por la llegada de una nueva aplicación, un cambio en el modo o medio de almacenamiento o las necesidades que impone el mantenimiento de la base de datos corporativa (Rizzo, 2011). Generalmente, una **migración de datos** se produce durante una actualización de hardware o la transferencia de un sistema existente a otro completamente nuevo. Algunos ejemplos son:

- Actualización de una base de datos.
- Migración hacia o desde la plataforma de hardware.
- Migración a un nuevo software.
- Fusión de dos sistemas paralelos en uno solo que se requiere cuando una empresa absorbe a otra o cuando dos negocios se fusionan.

1.2.2 Bases de Datos Documentales

Las bases de datos documentales están concebidas para el procesamiento, captura, almacenamiento, distribución y recuperación de información vinculada con la representación del conocimiento registrado en los documentos (Yunta, 2001).

Las bases de datos documentales de forma general poseen las siguientes características:

Se construyen con información no estructurada, tipo texto (documentos).

- ✓ Gestionan tipos de datos muy complejos (documentos científicos y técnicos, entre otros).
- ✓ Poseen un potente sistema de recuperación de información.

Ejemplos de bases de datos documentales son las bases de datos bibliográficas, bases de datos de prensa, bases de datos de informes de una empresa y científicos.

1.2.3 Registro Bibliográfico

Un registro bibliográfico es “un conjunto de elementos informativos, organizados conforme a unas normas, que permiten identificar a una unidad documental de manera unívoca en vistas a su localización y posterior recuperación” (Hilario, 2003).

1.2.4 Formatos Bibliográficos

Un formato bibliográfico determina el estilo en la que las referencias bibliográficas se mostrarán en un texto. De esta manera, por ejemplo, se determina si el apellido del autor tiene que ir en mayúsculas, si el título va entre comillas y si aparece el año de publicación.

Existen miles de formatos bibliográficos establecidos por editoriales e instituciones académicas como es el caso de MARC 21, CEPAL, CCF, entre otros.

Entre los formatos bibliográficos el de más amplia proyección es el MARC, que fue creado en 1965 por La Biblioteca del Congreso de Washington, quién lo ensayó hasta 1967 como proyecto piloto en colaboración de 16 bibliotecas americanas de diferente tipo, encargada de sugerir las reformas pertinentes (Betty Furrie, 2001).

1.3 Tecnologías

1.3.1 Lenguajes de Programación y de Consulta

Un lenguaje de programación es un lenguaje artificial que puede ser usado para controlar el comportamiento de una máquina, especialmente una computadora. Estos se componen de un conjunto de reglas sintácticas y semánticas que permiten expresar instrucciones que luego serán interpretadas (ALEGSA, 2010).

Java

Como lenguaje de programación para computadores, Java se introdujo a finales de 1995. Muchos expertos opinan que Java es el lenguaje ideal para aprender la informática moderna, porque incorpora conceptos de un modo estándar, mucho más sencillo y claro que otros lenguajes. Java es simple, orientado a objetos, distribuido, interpretado, robusto, seguro, de arquitectura neutra, portable, de altas prestaciones, multitarea y dinámico (Jalón, 2013). Java es un lenguaje de programación y plataforma de computación primero lanzado por Sun Microsystems en 1995. Hay muchas aplicaciones y sitios web que no funcionan a menos que haya instalado Java, y más cada día se crean (Oracle, 2016).

Se hace uso de lenguaje de programación java debido a que jisis se encuentra implementado completamente en este lenguaje, se utiliza además el componente para el acceso a datos de base de datos jisis implemetado en el centro en el propio

lenguaje siendo una garantía para el equipo de desarrollo, y las bondades del multihilo para la realización de la barra de progreso que mostrará el progreso de la migración.

1.3.2 J-ISIS

J-ISIS es un nueva multiplataforma libre y de código abierto ISIS[1], que mantiene las mismas funcionalidades y conceptos exitosos. Usa una arquitectura de Cliente/Servidor.

J-ISIS sigue los conceptos de CDS/ISIS[2] para guardar los recursos y experiencia de los usuarios, tal que los usuarios les sea familiar trabajar con la familia CDS/ISIS y recuperar los mismos conceptos. En el lado del diseñador, el objetivo principal es desarrollar una solución a largo plazo que sería de más fácil mantenimiento modular.

1.4 Herramientas

1.4.1 Entorno de Desarrollo Integrado (IDE)

Un entorno de desarrollo integrado o IDE (acrónimo en inglés de *Integrated Development Enviroment*), es un programa informático compuesto por un conjunto de herramientas de programación. Puede dedicarse en exclusiva a un solo lenguaje de programación o bien, puede utilizarse para varios. Un IDE es un entorno de programación que ha sido empaquetado como un programa de aplicación, es decir, consiste en un editor de código, un compilador, un depurador y un constructor de interfaz gráfica. Los IDEs pueden ser aplicaciones por sí solas o pueden ser parte de aplicaciones existentes (FUENTES, 2009).

NetBeans

NetBeans es un entorno de desarrollo integrado (IDE) de código abierto para el desarrollo de aplicaciones profesionales de escritorio y web, utilizando los lenguajes de programación Java, PHP, C++ entre otros (Oracle, 2017).

La plataforma NetBeans permite que las aplicaciones sean desarrolladas a partir de un conjunto de componentes de software llamados módulos. Un módulo es un archivo Java que contiene clases de java escritas para interactuar con las APIs de NetBeans y

[1] ISIS es un acrónimo de Integrated Set of Information Systems (conjunto integrado de sistemas de información). En 2003 fue establecido que este paquete era aceptado por bibliotecas en países en desarrollo como un software estándar para el desarrollo de sistemas de información.

[2] **CDS/ISIS** es un acrónimo de Computerised Documentation Service / Integrated Set of Information Systems (servicio de documentación computarizada / conjunto integrado de sistemas de información es un paquete de software para los sistemas de almacenamiento y recuperación de información no-numérica (Information Storage and Retrieval systems) desarrollado, mantenido y diseminado por UNESCO.

un archivo especial (manifest file) que lo identifica como módulo. Las aplicaciones construidas a partir de módulos pueden ser extendidas agregándole nuevos módulos. Debido a que los módulos pueden ser desarrollados independientemente, las aplicaciones basadas en la plataforma NetBeans pueden ser extendidas fácilmente por otros desarrolladores de software.

1.4.2 Marco de Trabajo (Framework)

Marco de Trabajo (*del inglés "Framework"*) se define como un conjunto de componentes físicos y lógicos estructurados de tal forma que permiten ser reutilizados en el diseño y desarrollo de nuevos sistemas de información. En una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado, a partir de una estructura software compuesta de componentes personalizables e intercambiables para el desarrollo de una aplicación. Los marcos de trabajo contienen patrones y buenas prácticas que apoyan el desarrollo de un producto y un proceso con calidad. Lo anterior permite vislumbrar la importancia de adoptar un Marco de Trabajo y cómo su selección debe ser una de las actividades relevantes al inicio de todo proceso de desarrollo software (Carlos A. Guerrero, 2014).

Spring framework

Spring ayuda a los equipos de desarrollo de todo el mundo a construir de manera simple, portátil, sistemas y aplicaciones de forma rápida y flexible basadas en una máquina virtual de Java. Permite escribir código limpio, que puede ser probado con los componentes de la infraestructura de su elección y llevar a cabo cualquier tarea. Spring proporciona un modelo de programación abierta integral, coherente, ampliamente entendida y bien soportada. Una de sus principales características es la inyección de dependencias (Pivotal, 2016).

1.5 Metodología de Desarrollo

Una metodología es un proceso formalizado o conjunto de buenas prácticas para crear software. Incluye: un conjunto de reglas a seguir, un conjunto de convenciones que la organización decide seguir y un acercamiento ingenieril y sistemático para organizar proyectos de software. En el ámbito de la Ingeniería de Software, la evolución de las metodologías de desarrollo de software ha llevado a la aparición de las denominadas metodologías ágiles, las cuales están destinadas a romper con la rigidez de las tradicionales, caracterizadas por la extensa documentación del proceso de desarrollo y por la inflexibilidad ante los cambios (Simulación de Proyectos de Software desarrollados con XP : Subsistema de Desarrollo de Tareas., 2012).

La técnica de la matriz de Boehm - Turner como resultado de aplicar el método que lleva el nombre de sus creadores, 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 (BOERAS VELÁZQUEZ, 2012).

A continuación, se muestra en una tabla, la descripción de los factores de la matriz de Boehm – Turner, con su descripción y los resultados de aplicarlos a la presente investigación.

Tabla 1Factores 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 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.	2 Integrantes
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	40 %

	puntuaciones indican una metodología ágil.	
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.	50 % 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 se muestra la matriz resultante de aplicar los cinco factores descritos por Boehm - Turner. En base al resultado observado, se decide considerar solamente metodologías ágiles para el desarrollo de la presente investigación.

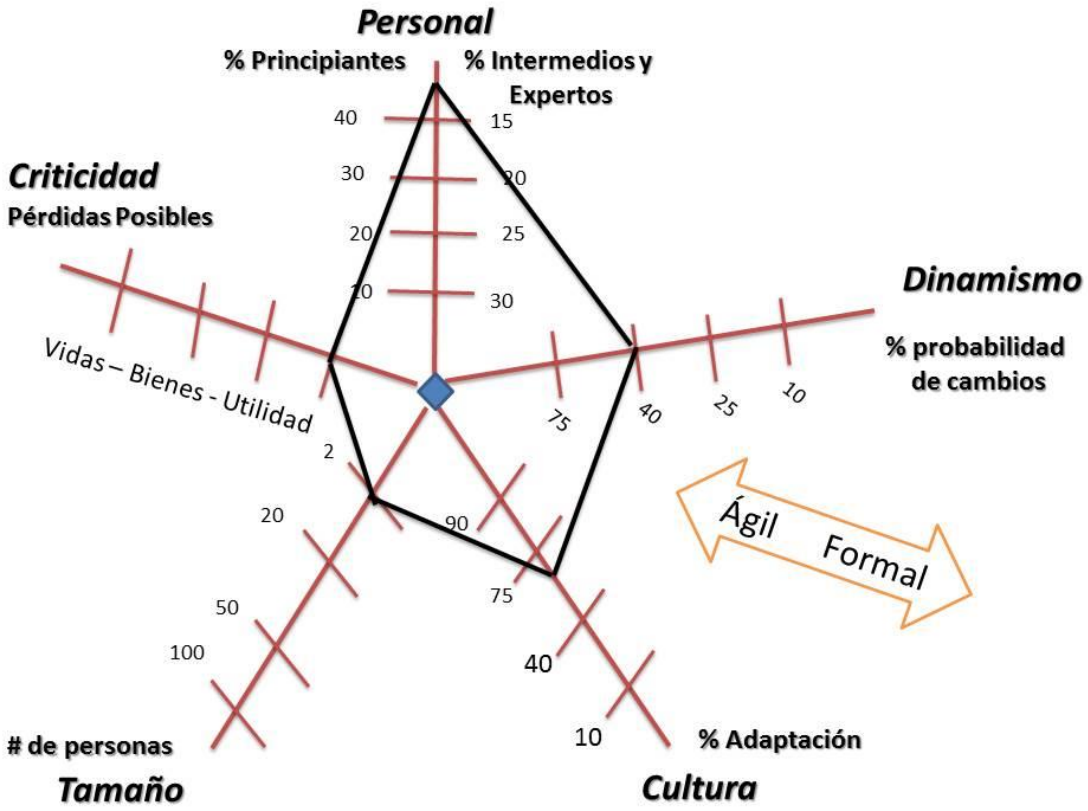


Figura 1 Matriz de Boehm -Turner aplicada a la investigación (elaboración propia).

1.5.1 SCRUM

Esta metodología centra su atención en las actividades de gerencia basándose principalmente en una planificación adaptativa y en el desarrollo incremental del software con entregas funcionales en breves períodos de tiempo. Frente a escenarios y requerimientos cambiantes, contar con una herramienta que permita simular la gestión de proyectos de desarrollo de software con SCRUM, representa una alternativa interesante para que los administradores puedan evaluar el impacto de sus decisiones sobre la gestión en el desarrollo del proyecto, sin influir o poner en riesgo el proyecto real y sus recursos (GODOY, 2014).

1.5.2 Agile Unified Process(AUP)

El AUP es un acercamiento al desarrollo del software basado en el Proceso Unificado Rational de IBM (RUP), basado en disciplinas y entregables incrementales con el tiempo. El ciclo de vida en proyectos grandes es serial mientras que en los pequeños es iterativo. Las disciplinas de AUP son (FIGUEROA, 2011):

- ✓ Modelado
- ✓ Implementación
- ✓ Prueba
- ✓ Despliegue
- ✓ Administración de la configuración
- ✓ Administración o gerencia del proyecto
- ✓ Entorno

1.5.3 Programación Extrema (XP)

La Programación Extrema (XP) reúne un conjunto de prácticas sencillas ya conocidas, pero que en este caso son llevadas a cabo conjuntamente y en forma extrema (Simulación de Proyectos de Software desarrollados con XP : Subsistema de Desarrollo de Tareas., 2012). XP propone 6 fases para el proceso de desarrollo de software, exploración, planificación, iteraiones, produción, mantenimiento y muerte del proyecto (Wesley). Bajo esta metodología, cada programador define sus pruebas cuando escribe su código de producción. Las pruebas se acoplan en el proceso de integración continua y construcción lo que rinde una plataforma altamente estable para el desarrollo futuro.

XP es una metodología ágil centrada en potenciar las relaciones interpersonales como clave para el éxito en desarrollo de software, promoviendo el trabajo en equipo, preocupándose por el aprendizaje de los desarrolladores, y propiciando un buen clima de trabajo. XP se basa en realimentación continua entre el cliente y el equipo de desarrollo, comunicación fluida entre todos los participantes, simplicidad en las

soluciones implementadas y coraje para enfrentar los cambios. XP se define como especialmente adecuada para proyectos con requisitos imprecisos y muy cambiantes, y donde existe un alto riesgo técnico (LETELIER, 2006).

Se decide utilizar la metodología XP ya que:

- ✓ No se cuenta con mucho tiempo de desarrollo, el cual debía ser aprovechado al máximo posible en el desarrollo de la aplicación y no en la redacción y construcción de numerosos productos de trabajo.
- ✓ Se ajusta a las necesidades del proyecto, al proveer técnicas sencillas para la representación de la aplicación; como también se ajusta a escenarios con requisitos cambiantes.
- ✓ Promueve la programación en dúos, lo cual conlleva ventajas implícitas como son: menor tasa de errores y mayor satisfacción de los programadores.
- ✓ Da lugar a una programación organizada.
- ✓ Facilita los cambios.
- ✓ El cliente tiene el control sobre las prioridades.

1.6 Conclusiones del Capítulo

Con la realización de este capítulo se arribaron a las siguientes conclusiones parciales:

- ✓ Se dejaron fundamentados los conceptos relacionados con la investigación que ayudaron a una mejor comprensión de la situación planteada.
- ✓ Se seleccionaron las herramientas mas adecuadas a partir del estudio realizado.
- ✓ Se escogió como lenguaje de programación Java por las funcionalidades antes mencionadas, además del soporte para crear interfaces gráficas de forma visual y la experiencia del equipo de desarrollo para el trabajo con el lenguaje.
- ✓ La metodología ágil XP para regir el proceso de desarrollo de software es la idónea para guiar el desarrollo de este proyecto ya que el equipo es pequeño y cuenta con poco tiempo.

Capítulo 2: Propuesta de Solución.

2.1 Introducción

En el presente capítulo se exponen los elementos que permiten describir la herramienta propuesta para dar solución a la situación problemática existente, se plasman los artefactos ingenieriles definidos por la metodología de desarrollo establecida y se define el estilo arquitectónico y los patrones de diseño utilizados.

2.2 Propuesta de Solución

La propuesta de solución se nombra: Herramienta para la migración entre formatos bibliográficos almacenados en J-ISIS.

La solución dispone de dos interfaces, una primera interfaz que permite seleccionar la base de datos inicio que contiene los registros que se desean migrar, y una base de datos destino donde se guardará el resultado de la migración.

Figura 2 Interfaz 1 de la aplicación

Figura 3 Interfaz 2 de la aplicación

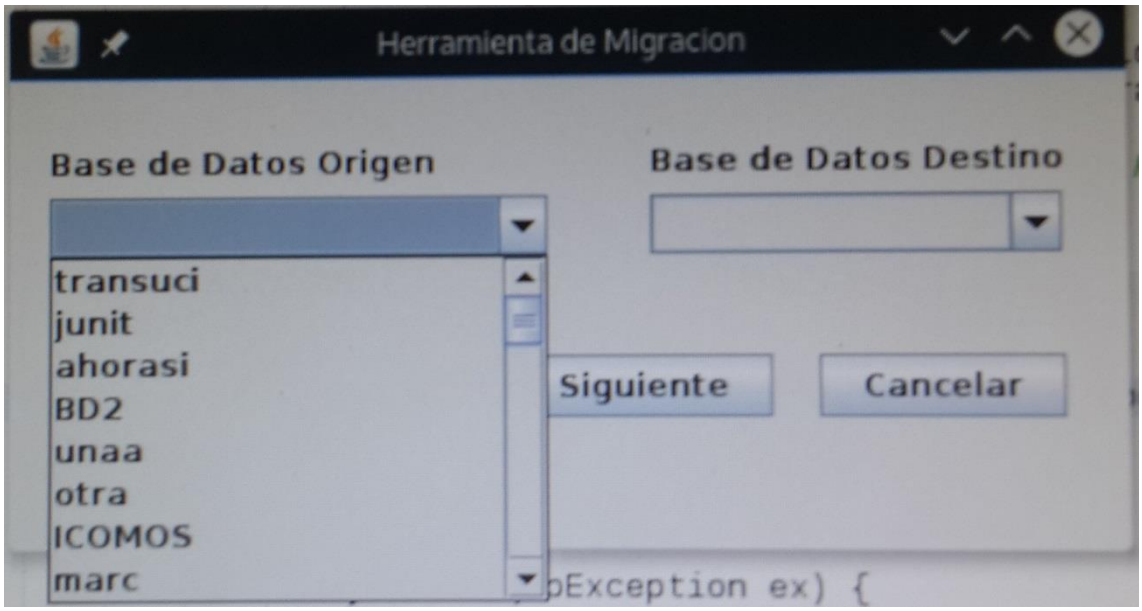


Figura 4 Selección de la base de Datos Origen

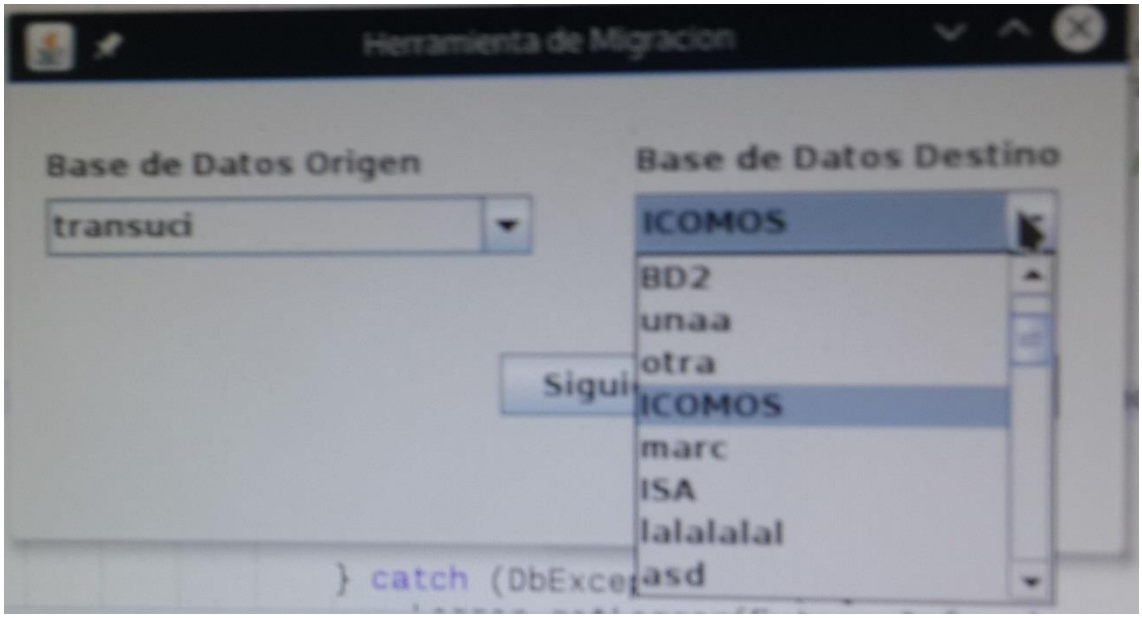


Figura 5 Selección de la Base de Datos Destino

La base de datos origen no debe ser vacía, debe contener los registros que se desean migrar, si se selecciona una base de datos que no esté llena, la aplicación lanza un mensaje de error. Al igual si la base de datos destino se selecciona llena, lanza un mensaje expresando que tiene que ser completamente vacía como se ve en la figura 6 y 7.

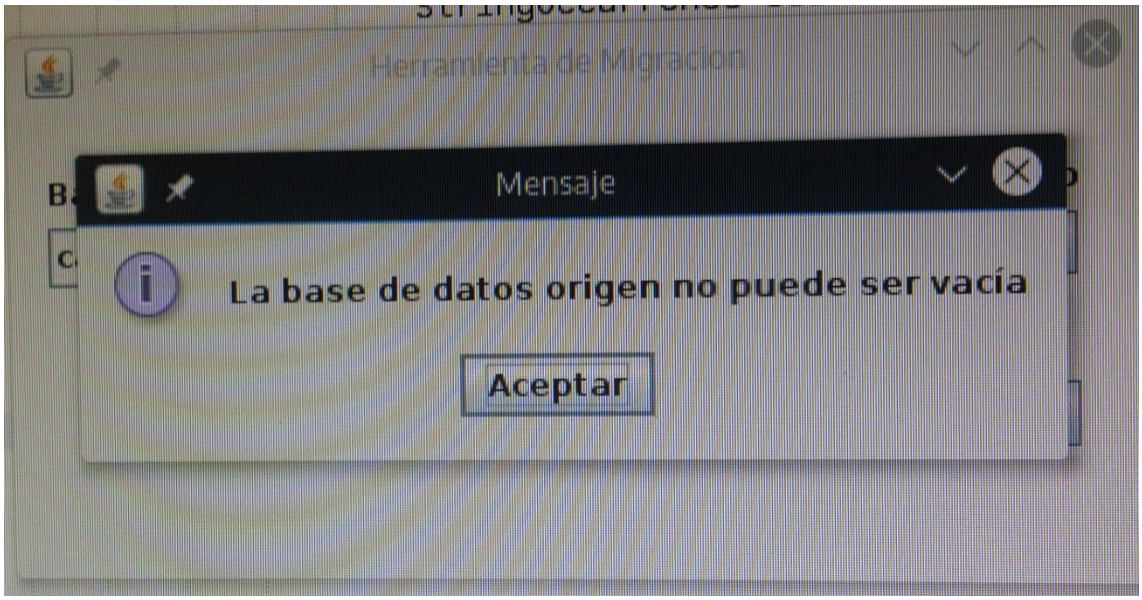


Figura 6 Mensaje de error para primera base de datos

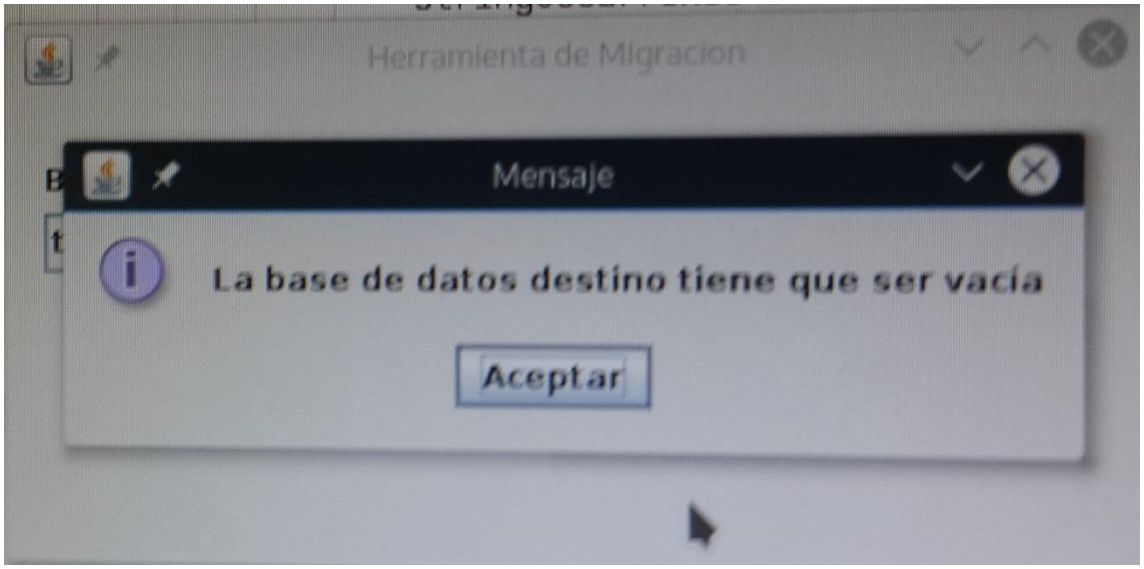


Figura 7 Mensaje de error para la base de datos destino

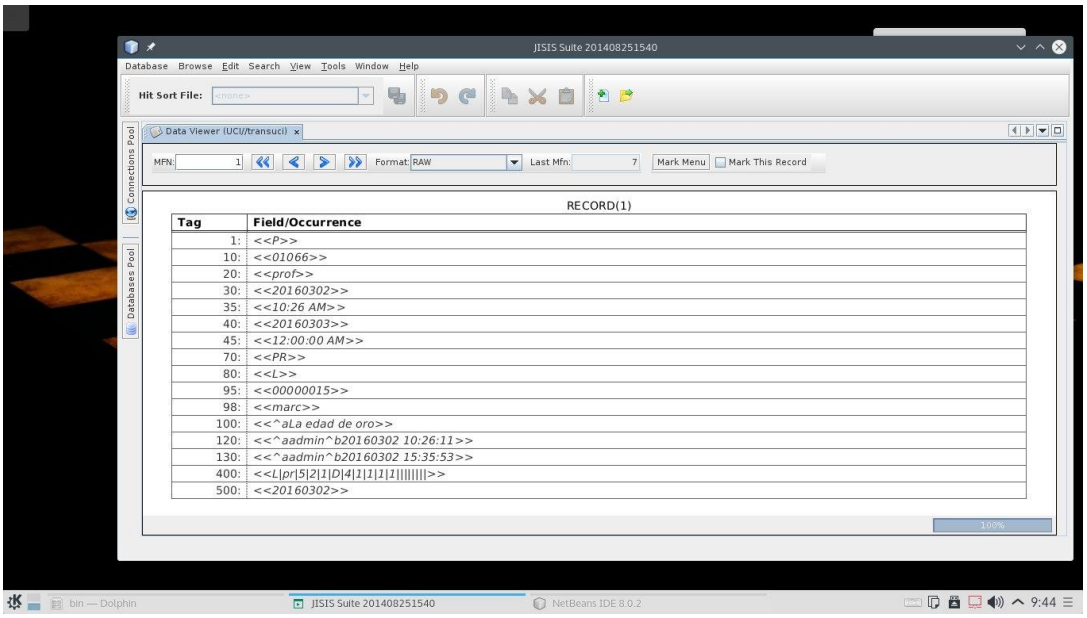


Figura 8 Ejemplo de un registro almacenado en una base de datos Jisis

Al seleccionar las bases de datos correctamente y dar clic en el botón siguiente, se pasa a la segunda interfaz, donde se debe seleccionar el formato al que se desea migrar, es decir de que formato parto y a cuál deseo llegar. Esos nombres corresponden con los nombres de los ficheros de alineación que están contenidos en una carpeta llamada ficheros en el proyecto.

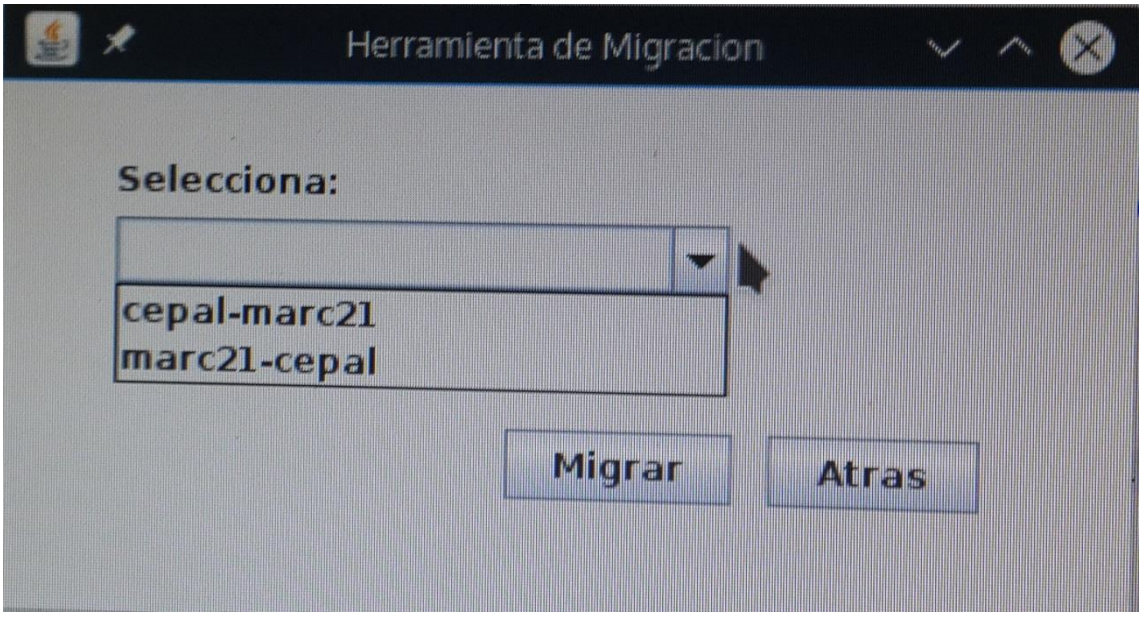


Figura 9 Selección del formato al que se desea migrar

Ya seleccionado el formato al que se desea migrar, se da clic en el botón Migrar y en la base de datos destino quedaría guardado el resultado de la migración.
La migración consiste en según lo especificado en el fichero de alineación, mover los campos de la base de datos origen hacia la base de datos destino. Un fichero de alineación puede estar estructurado de tres maneras distintas:
Ejemplo:
Campo = Campo
Campo = Campo^Subcampo
Campo^Subcampo = Campo
Como se puede observar en la figura 10.

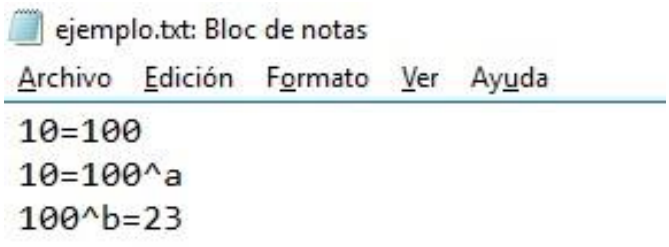


Figura 10 Ejemplo de estructura de un fichero

Los ficheros contienen las equivalencias entre los formatos bibliográficos, según el ejemplo de estrucutura de un fichero de la figura 10, quiere decir que lo que esté en el campo 10 de la base de datos origen sea movido al campo 100 de la base de datos destino.
Para la segunda línea es que lo que está en el campo 10 de la base de datos origen sea movido al campo 100 subcampo a de la base de datos destino.
Para la tercer línea significa que lo que esta en el campo 100 subcampo b, sea movido al campo 23 de la base de datos destino.
Y así es el completo funcionamiento de la aplicación Herramienta para la migración entre formatos bibliográficos almacenados en J-ISIS.

2.3 Requisitos Funcionales y no Funcionales

2.3.1. Requisitos Funcionales del sistema

Los requisitos funcionales son declaraciones de los servicios que debe proporcionar el sistema, de la manera que éste debe reaccionar a entradas particulares y de cómo se debe comportar en situaciones particulares. En algunos casos, los requisitos

funcionales de los sistemas pueden declarar explícitamente lo que el sistema no debe hacer (SOMMERVILLE, 2005).

La herramienta contará con los siguientes requisitos funcionales:

Tabla 2 Requisitos Funcionales

Número	Nombre	Descripción
RF1	RF 1: Cargar fichero de alineación.	Esta funcionalidad permite que la aplicación cargue las equivalencias entre formatos bibliográficos que posteriormente serán empleados en el proceso de la migración.
RF2	RF 2: Migrar registros entre bases de datos J-ISIS.	Esta funcionalidad consiste en el traspaso de los datos almacenados acordes a un formato bibliográfico hacia otro en correspondencia con lo descrito en el fichero de alineación.
RF3	RF 3: Mostrar el progreso de la migración.	Esta funcionalidad permite mantener informado al usuario con respecto al progreso del proceso de migración.

2.3.2. Requisitos No Funcionales del sistema

Los requisitos no funcionales son restricciones de los servicios o funciones ofrecidas por el sistema. Incluye restricciones de tiempo, sobre el proceso de desarrollo y estándares. Los requisitos no funcionales a menudo se aplican al sistema en su totalidad. Normalmente apenas se aplican a características o servicios individuales del sistema (SOMMERVILLE, 2005).

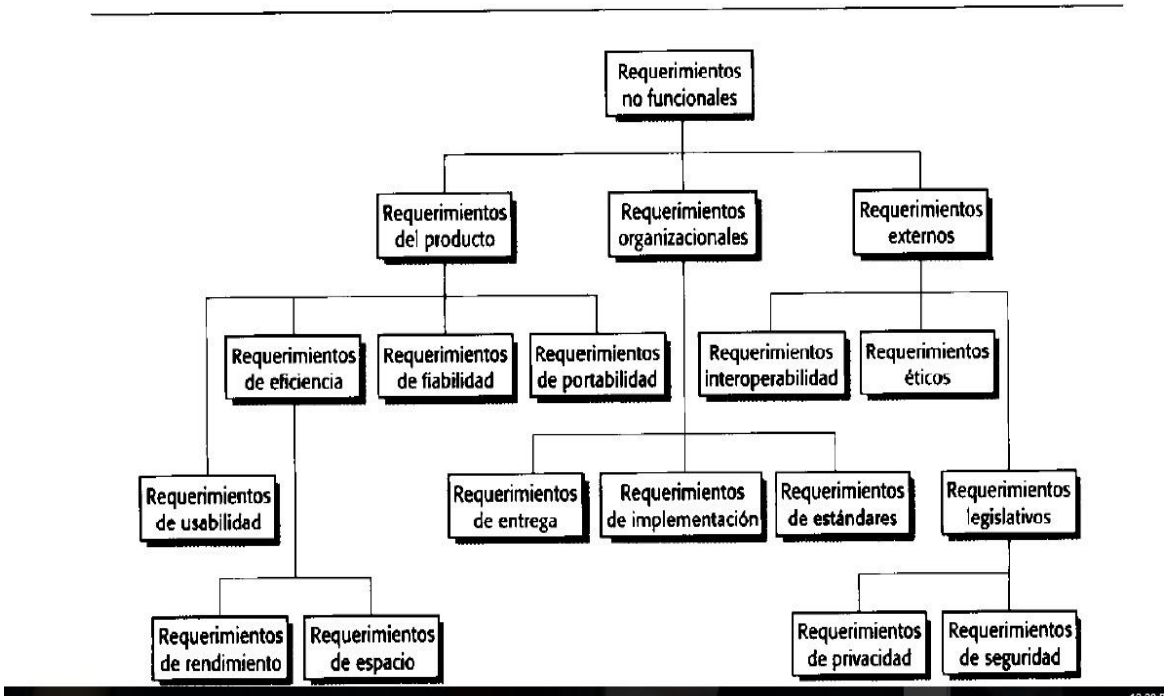


Figura 11 Tipos de requisitos no funcionales

FUENTE: SOMMERVILLE I., 2005

La herramienta contará con los siguientes requisitos no funcionales:

Requisitos de usabilidad

- ✓ La interfaz de la aplicación debe permitirle al usuario sin experiencia poder interactuar fácilmente con el sistema y así adaptarse rápidamente.

Apariencia e Interfaz

- ✓ La aplicación deberá poseer una interfaz fácil de usar por los usuarios.
- ✓ Todos los mensajes en pantalla aparecerán en idioma español.

Rendimiento

- ✓ El funcionamiento del sistema informático debe ser estable.

Requisitos de hardware

- ✓ Computador con procesador Core i3 segunda generación, 4 GB de memoria RAM.

Requisitos de software

- ✓ Para que la aplicación funcione debe estar instalado la máquina virtual de Java 7 y un Sistema Gestor de Bases de Datos(J-ISIS).

2.4 Fases del proceso de desarrollo

2.4.1 Fase de exploración

En esta fase los clientes describen las Historias de Usuario (HU) definiendo las características que van a tener cada una de ellas. Las historias de usuario son la técnica utilizada en XP para especificar los requisitos del software. Se trata de tarjetas de papel en las cuales el cliente describe brevemente las características que el sistema debe poseer, sean requisitos funcionales o no funcionales. El tratamiento de las historias de usuario es muy dinámico y flexible, en cualquier momento historias de usuario pueden romperse, reemplazarse por otras más específicas o generales, añadirse nuevas o ser modificadas. Cada historia de usuario es lo suficientemente comprensible y delimitada para que los programadores puedan implementarla en unas semanas (LETELIER, 2006).

A continuación, se explica cada uno de los datos que deben ser llenados en la HU:

Número: identificador de la HU.

Nombre: nombre que identifica a la HU.

Usuario: involucrados en la ejecución de la HU.

Iteración asignada: iteración en que se implementará la HU.

Prioridad en el negocio: prioridad de la HU respecto al resto de las HU, puede ser: alta, media o baja.

Riesgo en desarrollo: riesgo en la implementación de la HU, puede ser: alto, medio o bajo.

Puntos estimados: estima el esfuerzo asociado a la implementación de la HU. Un punto equivale a una semana ideal de programación, generalmente de 1 a 3 puntos.

Puntos reales: resultado del esfuerzo asociado a la implementación de la HU. Un punto equivale a una semana ideal de programación, generalmente de 1 a 3 puntos.

Descripción: descripción sintetizada de la HU.

Observaciones: información de interés.

A continuación, se describe la estructura de algunas Historias de Usuario con prioridad alta y media para el negocio.

Tabla 3 Historia de Usuarios Cargar Fichero de Alineación

Historia de Usuario	
Número: 1	**Nombre:** Cargar Fichero de alineación.
Usuario: Migrador	**Iteración Asignada: 1**
Prioridad en Negocio: Alta	**Puntos Estimados: 3**
Riesgo en Desarrollo: Medio	**Puntos Reales: 2**
Descripción: Permite que la aplicación cargue la información almacenada en el fichero de alineación que se encuentra estructurado acorde a lo descrito.	
Observaciones: Estos archivos de texto deben estar en la carpeta especificada del proyecto.	

Tabla 4 Historia de Usuarios Migrar Registros entre bases de datos J-ISIS

Historia de Usuario	
Número: 2	**Nombre:** Migrar registros entre bases de datos J-ISIS.
Usuario: Migrador	**Iteración Asignada: 1**
Prioridad en Negocio: Alta	**Puntos Estimados: 3**
Riesgo en Desarrollo: Alto	**Puntos Reales: 3**
Descripción: Permite exportar la información que está en un fichero de alineación hacia otra base de datos J-ISIS con el equivalente al formato escogido en primer lugar.	
Observaciones: Los formatos escogidos para realizar la migración en cada caso deben ser distintos, es decir no deben ser escogidos un mismo formato para migrar hacia ese mismo formato.	

Tabla 5 Historia de Usuarios Mostrar progreso de la migración

Historia de Usuario	
Número: 3	**Nombre:** Mostrar el progreso de la migración.
Usuario: Migrador	**Iteración Asignada: 1**
Prioridad en Negocio: Media	**Puntos Estimados: 1**
Riesgo en Desarrollo: Bajo	**Puntos Reales: 1**
Descripción: Permite mostrar en una barra el progreso de la migración.	
Observaciones:	

2.4.2 Fase de planificación

En la fase de planificación se realiza la estimación del esfuerzo que causará la implementación de cada HU, como en la metodología ágil XP las métricas son libres, puede utilizarse cualquier criterio definido para medir el desempeño del proyecto en cuestión (LETELIER, 2006).

Estimación de esfuerzo por Historias de Usuarios

Para la realización de la aplicación propuesta se efectuó una estimación de esfuerzo por cada una de las HU identificadas, donde a continuación se muestran los resultados. Los puntos de estimación están en un rango de 1 a 3.

Tabla 6 Estimación de Esfuerzo por Historias de Usuarios

Historia de Usuario	Puntos de Estimación (Semanas)
Cargar Fichero de alineación	3
Migrar registros entre bases de datos J-ISIS	3
Mostrar progreso de la migración	1

Plan de iteraciones

Luego de haber identificado las HU y de realizar una estimación del tiempo requerido para la realización de cada una, se procede a realizar el plan de iteraciones en el cual estarán comprendidas las HU según el orden en que sean realizadas por cada iteración, además del total de semanas que durará cada una de estas, se realizó la planificación estableciendo las iteraciones que necesarias antes de entregar el sistema. Teniendo en cuenta la prioridad y el riesgo de cada una de las HU se decidió dividir el proyecto en dos iteraciones, detalladas a continuación:

Iteración 1: En esta iteración se llevará a cabo el desarrollo de las HU número 1 correspondiente a las funcionalidad de cargar fichero de alineación.

Iteración 2: En esta iteración se llevará a cabo el desarrollo de las HU número 2 correspondiente a las funcionalidad migrar registros entre bases de datos J-ISIS .

Iteración 3: En esta iteración se llevará a cabo el desarrollo de las HU número 3 correspondiente a las funcionalidad mostrar progreso de la migración.

Plan de duración de las iteraciones

En el plan de duración de las iteraciones se muestran las historias de usuario que son implementadas en cada una de las iteraciones, así como la duración estimada y el orden de implementación de cada una de ellas (ESCRIBANO, 2002).

La siguiente tabla se encarga de reflejar el orden de las HU a implementar, asi como la estimación por semanas de las mismas.

Tabla 7 Plan de duración de las iteraciones

Iteración	Orden de las HU a implementar	Estimación (Semanas)
1	Cargar fichero de alineación	3
2	Migrar registros entre bases de datos J-ISIS	3
3	Mostrar el progreso de la migración	1

Plan de Entregas

En el plan de entregas se realiza un cronograma de entregas donde el cliente establece la prioridad de cada HU, cuáles serán agrupadas para conformar una

entrega y el orden de las mismas. En correspondencia, los programadores realizan una estimación del esfuerzo necesario de cada una de ellas. En la tabla se muestra la versión del componente al concluir cada iteración y si ya fue finalizada esa HU se muestra una F (LETELIER, 2006).

Tabla 8 Plan de Entregas

Historia de Usuario	Iteración 1 25/Abril/2017	Iteración 2 7/Mayo/2017	Iteración 3 18/Mayo/2017	Sistema finalizado
Cargar fichero de alineación.	V1.0	F	F	F
Migrar registros entre bases de datos J-ISIS.	-	V1.1	F	F
Mostrar el progreso de la migración.	-	-	2.0	F

2.4.3 Fase de diseño

2.5 Diseño de la arquitectura

Apunta (Cervantes, 2010) que la arquitectura de *software "constituye un modelo comprensible de cómo está estructurado el sistema y cómo trabajan juntos sus componentes"*

Se empleará la arquitectura cliente-servidor para el desarrollo del sistema informático. La herramienta para la migración entre formatos de registros bibliográficos almacenados en J-ISIS es una herramienta de escritorio que lee datos a la vez que escribe datos en un servidor de base de datos jisis por lo que puede considerarse como un cliente que encuesta a un servidor de jisis.

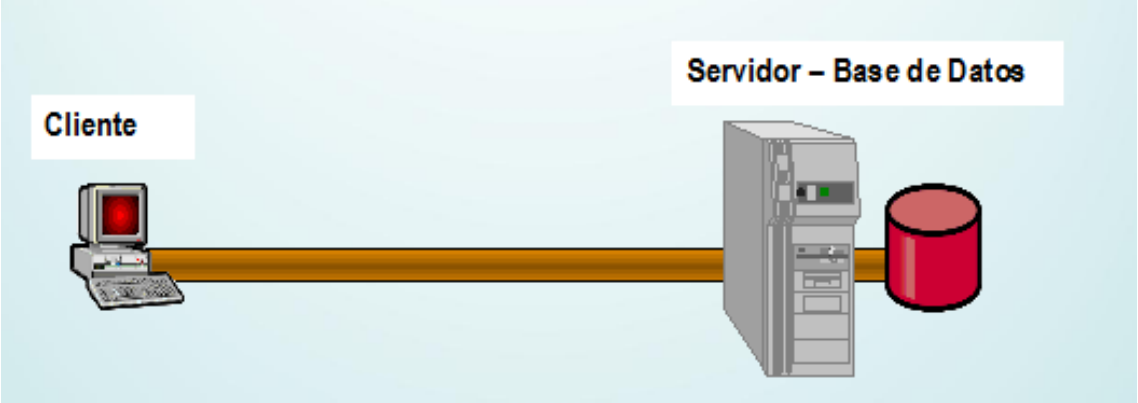


Figura 12 Arquitectura Cliente - Servidor.

Ventajas de este modelo:

- ✓ Se mantiene una conexión persistente con la base de datos.
- ✓ Se minimizan las peticiones en el servidor trasladándose la mayor parte del trabajo al cliente.
- ✓ Se gana en rendimiento gracias a la conexión directa y permanente con la base de datos. A través de una única conexión se realiza el envío y recepción de varios datos.

2.6 Patrones de Diseño

Los patrones de diseño se derivan de las ideas expuestas por Christopher Alexander, quien sugirió que había ciertos patrones comunes en la construcción del diseño que eran inherentemente efectivos. El patrón es una descripción del problema y la esencia de su solución, de modo que la solución puede ser reutilizada en diferentes entornos. El patrón no es una especificación detallada. Más bien, se puede pensar en ella como una descripción del conocimiento y la experiencia acumulada, una solución de eficacia probada a un problema común (SOMMERVILLE, 2011).

2.6.1 Patrones GRASP

- ✓ *General Responsibility Assignment Software Patterns* (GRASP, por sus siglas en inglés): describen un conjunto de principios básicos de la asignación de responsabilidades a objetos. Algunos de los patrones que conforman este grupo son: experto, creador, bajo acoplamiento, alta cohesión y controlador.

- ✓ Alta cohesión: *"es una medida de cuán relacionadas y enfocadas están las responsabilidades de una clase. Una alta cohesión caracteriza a las clases con*

responsabilidades estrechamente relacionadas que no realizan un trabajo enorme" (Pressman, 2005).

Ejemplo: Métodos de la clase extraerinfo.

- ✓ Experto: Según (Pressman, 2005), el patrón experto *"es el encargado de asignar la responsabilidad de la creación de un objeto o la implementación de un método a una clase que contenga toda la información necesaria para cumplir con dicha responsabilidad"*.

Ejemplo:

```
 */
public MainUI()  {

    initComponents();
    this.setTitle("Herramienta de Migracion");

}

@PostConstruct
private void loadValues(){
    List<String> lis;
    List<String> list;


    try {

        lis = dataProvider.getDatabaseNames("UCI");
        for (int i = 0; i < lis.size(); i++) {
          jComboBox1.addItem(lis.get(i));
          jComboBox2.addItem(lis.get(i));

        }

        jComboBox1.setSelectedIndex(-1);
        jComboBox2.setSelectedIndex(-1);
    } catch (JisisDatabaseException ex) {
        Logger.getLogger(MainUI.class.getName()).log(Level.SEVERE, null, ex);
    }
```

- ✓ Bajo acoplamiento:

  Para *Pressman el acoplamiento es una medida de la fuerza con que una clase está conectada a otras clases, con qué las conoce y con qué recurre a ellas. En tal sentido, el término bajo acoplamiento significa que una clase no depende de muchas clases"* (Pressman, 2005).

  Ejemplo: clase leerfichero con la clase extraerinfo

- ✓ Controlador: Se implementa como parte del patrón arquitectónico utilizado, sirve de intermediario entre una determinada interfaz y el algoritmo que la implementa recibiendo los datos del usuario y enviándolos a las distintas clases según el método llamado.

Ejemplo:

```
 */
@Component
@Lazy
public class extraerinfo {

    @Resource
    private IJisisDataProvider dataProvider;
    private String bdinicio;

    public void test() {
        Record rec;

        for (int i = 1; i < dataProvider.getLastMfn(bdinicio, "UCI"); i++) {
            rec = dataProvider.getRecordByMfn(i, bdinicio,"UCI");
        }


    }
public String getbdinicio(){
return bdinicio;
}

    public void setBdinicio(String bdinicio) {
        this.bdinicio = bdinicio;
    }

public void run(){
this.test();
```

2.7 Tarjetas CRC

(Clase - Responsabilidad – Colaborador)

El diseño de aplicaciones, en la metodología XP no requiere la representación del sistema mediante diagramas de clases utilizando notación UML[3], en su lugar se usan otras técnicas como las tarjetas CRC. Estas determinan responsabilidades y colaboraciones de las clases. El desarrollo de cualquier proyecto requiere de un buen diseño de sus clases para de esta forma realizarlo con la mejor calidad posible y así el cliente quede satisfecho. En la metodología XP el diseño de las clases se realiza a través de las tarjetas CRC, para de esta forma ayudar al refinamiento de las clases. De forma organizada cada tarjeta representa una clase, donde se describen las responsabilidades que tiene y las clases colaboradoras que se relacionan con la misma. Las tarjetas CRC también ayudan a diseñar el sistema en conjunto entre todo el equipo de desarrollo, aunque su principal objetivo es propiciar el enfoque

[3] Lenguaje Unificado de Modelado, es el lenguaje de modelado de sistemas de software más conocido y utilizado en la actualidad para visualizar, especificar, construir y documentar un sistema.

orientado a objetos y reducir el modo de pensar procedimental. Están diseñadas en cuatro secciones: nombre de la clase, descripción, responsabilidades y colaboradores. Una clase describe un objeto o evento de sistema, mediante sus atributos y métodos. Las responsabilidades de estas se describen por las tareas que realiza o por los métodos y los colaboradores son las demás clases con las que interactúa para cumplir con sus responsabilidades.

A continuación, se describen algunas de las tarjetas CRC diseñadas para la implementación del sistema:

Tabla 9 Tarjeta CRC extraerinfo

Nombre de la clase: extraerinfo	
Responsabilidad	**Colaboradores**
Se encarga de la migración de los registros.	• *Migrar* • *MainUI*

Tabla 10Tarjeta CRC Migrar

Nombre de la clase: Migrar	
Responsabilidad	**Colaboradores**
Se encarga de escoger los ficheros y se migran en dependencia del fichero escogido.	• Leerficheros • MainUI • extraerinfo

Tabla 11Tarjeta CRC MainUI

Nombre de la clase: MainUI	
Responsabilidad	**Colaboradores**
Se encarga de escoger la Base de Datos Origen y escoger la Base de Datos	• Migrar

destino.	• extraerinfo

Tabla 12Tarjeta CRC leerfichero

Nombre de la clase: leerfichero	
Responsabilidad	**Colaboradores**
Se encarga de leer los ficheros.	• Migrar

2.8 Conclusiones del capítulo

- ✓ La elaboración de los artefactos generados según la metodología seleccionada, garantizó la guía adecuada para el entendimiento y desarrollo de la propuesta de solución.
- ✓ La definición de requisitos funcionales facilitó la comprensión de las funcionalidades que debe cumplir la aplicación. Pues, los no funcionales permitieron identificar características necesarias para el despliegue y correcto funcionamiento del sistema.
- ✓ Spring como framework de desarrollo permitió el uso de los patrones.

Capítulo 3: Implementación y Pruebas

3.1. Introducción

El presente capítulo está orientado a las pruebas de la investigación que se llevan a cabo para dar total cumplimiento a los requisitos establecidos. Se evidencian las pruebas aplicadas a la solución con el objetivo de comprobar las funcionalidades en los diferentes escenarios y así verificar en todos los casos que los resultados sean los esperados. Para ello, se plasman las pruebas definidas por la metodología seleccionada.

3.2 Implementación

```
public void test() {
    Record rec;
    LeerDeFichero leer = new LeerDeFichero(direccion);
    leer.ObtenerLineas();
```

(Declaración de las variables que se van a utilizar en el algoritmo)

```
        for (int i = 1; i < dataProvider.getLastMfn(bdinicio,"UCI"); i++) {
```

(Ciclo que se moverá por todos los registros que contiene la base de datos que escogió el usuario como base de datos inicio)

```
            for (int a = 0; a < leer.getInicio().size(); a++) {
```

(Ciclo que se moverá por todo lo contenido en el fichero de alineación del signo igual hacia atrás, que fue guardado en una lista)

```
            rec = dataProvider.getRecordByMfn(i, bdinicio, "UCI");
```

(Se guarda en rec los registros que contiene la base de datos escogida por el usuario para realizarle la migración)

```
            try {
                if (!leer.getInicio().get(a).contains("^")) {
dato = rec.getField(Integer.parseInt(leer.getInicio().get(a))).getStringFieldValue();
```

(Se comprueba que lo que está del signo igual a la izquierda no contiene ^ , es decir que lo contenido al inicio es solo un campo, y de ser así guardo en la variable dato el valor contenido en el campo especificado en el fichero de alineación, del registro de la base de datos inicio)

```
            } else {

                String valor = leer.getInicio().get(a);
                int campo = Integer.parseInt(valor.substring(0, valor.indexOf("^")));
                String subcampo = valor.substring(valor.indexOf("^"),valor.length());
```

(si lo que esta del signo igual a la izquierda contiene ^, guardo por separado el valor del campo que sería lo que está antes de ^, y el valor del subcambo que sería lo que está después de ^).

```
                Integer indiceOcurrencia = getOcurrencia(valor);
                dato = rec.getField(campo).getSubfield(indiceOcurrencia, subcampo);
```

(Se crea una variable llamada indiceOcurrencia para contar las ocurrencias de los subcampo, y después se guarda en dato el valor del campo correspondiente).

```
            }

        } catch (DbException ex) {
            Logger.getLogger(ExtraerInfo.class.getName()).log(Level.SEVERE,     null,
ex);
        }

        IRecord record = Record.createRecord();
```

(Se crea un registro nuevo)

```
        if (!leer.getDestino().get(a).contains("^")) {

            fiel = new Field(Integer.parseInt(leer.getDestino().get(a)),
Global.FIELD_TYPE_ALPHANUMERIC);
```

(si lo que está del signo igual a la derecha no contiene ^, se crea un nuevo campo con la especificación de donde se guardará el valor)

```
            try {

                fiel.setFieldValue(dato);
```

(se gurda el dato en ese campo que se creó).

```
            } catch (DbException ex) {
                Logger.getLogger(ExtraerInfo.class.getName()).log(Level.SEVERE,
null, ex);
            }
            record.addField(fiel);
```
(se adiciona el campo al registro que se creó).

```
        } else {

            try {
                String valor1 = leer.getDestino().get(a);
                int campo1 = Integer.parseInt(valor1.substring(0, valor1.indexOf("^")));
                String subcampo1 =
valor1.substring(valor1.indexOf("^"),valor1.length());
```

(si contiene ^, guardo por separado el campo y el subcampo).

```
                ArrayList<Subfield> subfields = new ArrayList<Subfield>();
```
(se crea una lista de arreglos de subcampos).

```
                ISubfield subfield = new Subfield(subcampo1.charAt(0), dato);
                subfields.add((Subfield) subfield);
                StringOccurrence ocurrence = new StringOccurrence();

                Subfield[] subfieldsarray = subfields.toArray(new
Subfield[subfields.size()]);
                ocurrence.setSubfields(subfieldsarray);
```
(se crea un subcampo que recibe por parámetro el subcampo y el dato, después se adiciona a la lista de subcampos, esa lista se convierte a un arreglo. Al adicionarse a la lista de subcampos se crea una ocurrencia de tipo StringOcurrence donde se gurdará el subcampo).

```
                 fiel = new Field((campo1), Global.FIELD_TYPE_ALPHANUMERIC);
```
(se crea un campo nuevo con la posición según el fichero de alineación).

```
                record.addField(fiel);
```

(se adiciona el campo al registro).

```
                } catch (DbException ex) {
                    Logger.getLogger(ExtraerInfo.class.getName()).log(Level.SEVERE,
null, ex);
                }

            }
```

3.2.1 Pseudocódigo del Algoritmo

```
metodo prueba{

variable rec
CargarFicheroDeFichero leer recibe new CargarFichero(direccion);
leer.ObtenerLineas();

desde i recibe 1 hasta cantidadDeRegistros hacer{

desde  a recibe 0 hasta longitudMiembroIzquierdo{

variable rec recibe registros;

intentar{

si(miembro izquierdo no contiene ^){

variable dato recibe valor del registro en miembroIzquierdo.ObtenerEn(a)
}

sino{
varible valor recibe miembroIzquierdo.ObtenerEn(a);
variable campo recibe miembroIzquierdo.ObtenerDesde(inicio,^);
variable campo recibe miembroIzquierdo.ObtenerDesde(^,fin);

variable indiceOcurrencia recibe Ocurrencia(valor);
variable dato recibe valor del registro(campo).subcampo(indiceocurencia , subcampo);
} }
```

```
Registro record recibe crear registro;
si(miembroDerecho no contiene ^){
variable fiel = nuevo Field(miembroDerecho.ObtenerEn(a), ALPHANUMERIC);
intentar{
fiel.ModificarValor(dato);
}
record adicionar(fiel);
}
sino{
intentar{
variable valor1 recibe miembroDerecho.ObetenerEn(a);
variable campo1 recibe miembroDerecho.ObtenerDesde(inicio,^);
variale subcampo1 recibe  miembroIzquierdo.ObtenerDesde(^,final);
ListadeArreglos subfields recibe nueva ListadeArreglos();
Subcampo subfield recibe nuevo Subcampo(subcampo1 devuelve letra en(0) , dato);
subfields adicionar en((Subcampo)subfield);
CadenadeOcurrencia ocurrence recibe nueva CadenadeOcurrencia();
ArreglodeSubcampo subfieldsarray recibe subfields convertir a arreglo(nuevo
Subcampo[tamaño de subfields]);
fiel recibe nuevo Campo((campo1) , ALPHANUMERIC);
fiel modificaOcurrencia(ObtenerOcurrencia(ocurrence) , ocurrence);
record adicionarCampo(fiel);
} }
a++}
i++
} fin desde}
```

3.3 Fase de Pruebas

La fase de pruebas es la última de las fases propuestas por la Metodología XP. La verificación de los requisitos que debe cumplir aplicación a través del análisis a las posibles combinaciones de entradas y de las salidas de datos, además de la comprobación de que el software trabaje como fue diseñado, son los objetivos que se persiguen con la realización de las pruebas al sistema. Como resultado permitirá un mayor control e identificación temprana de los defectos y fallos, de esta manera se garantiza la calidad del desarrollo con una reducción notable de los costos necesarios para corregir los errores. La calidad del sistema será medida en correspondencia con el número de no conformidades que sean detectadas (VALDEZ HUARACA, 2013).

3.4 Métodos de prueba

Es de vital importancia en esta etapa definir estrategias para descubrir fallos en el sistema, mediante los métodos de pruebas, (Pressman, 2005) propone los siguientes:

3.4.1 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: 1) garanticen que todas las rutas independientes dentro de un módulo se revisaron al menos una vez, 2) revisen todas las decisiones lógicas en sus lados verdadero y falso, 3) ejecuten todos los bucles en sus fronteras y dentro de sus fronteras operativas y 4) revisen estructuras de datos internas para garantizar su validez.

3.4.2 Pruebas de caja negra

La prueba de caja negra no es una alternativa de las técnicas de prueba de la caja blanca, sino un enfoque complementario que intenta descubrir diferentes tipos de errores a los encontrados en los métodos de la caja blanca. Los campos que son estudiados desde el punto de vista de las entradas que recibe y las salidas o respuestas que produce, sin tener en cuenta su funcionamiento interno. Donde examina qué es lo que hace, pero sin dar importancia a cómo lo hace. Define las entradas y salidas, es decir, su interfaz; en cambio, no se precisa definir ni conocer los detalles internos de su funcionamiento. Es el estudio de un módulo o elemento de un sistema, desde su parte externa.

Roger Pressman en la séptima edición del libro *"Ingeniería de software Un enfoque práctico"* propone pruebas en los 4 niveles, por lo que la estrategia seguida para la realización de las pruebas a la herramienta informática son las siguientes:

Pruebas unitarias: son pruebas de caja blanca que 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.

Pruebas de integración: verifican que cada componente desarrollado no presente errores cuando se integre con los demás.

Pruebas de validación: son pruebas de caja negra que verifican las acciones que el usuario realiza en el sistema y la correcta entrada y salida de datos, pues se enfocan en la satisfacción de las necesidades del cliente.

Pruebas del sistema: son pruebas que confirman el correcto funcionamiento de las funciones desarrollas.

Las 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 de caja negra que se crean a partir de las historias de usuario. Cuando se pasa la prueba de aceptación se considera la historia de usuario finalizada. El objetivo final de estas es garantizar que los requerimientos han sido cumplidos y que el sistema es aceptable.

A continuación se detallan las distintas pruebas de Aceptación empleadas para el correcto funcionamiento de las funcionalidades previstas en cada una de las Historias de Usuario:

Prueba de Aceptación	
Código: HU1_P1	**Historia de Usuario:** Cargar Fichero de alineación
Nombre: Cargar Fichero de alineación.	
Descripción: Carga todos los ficheros de alineación que se encuentra en una ruta especificada.	
Condiciones de Ejecución: El fichero de alineación seleccionado se encuentra en la ruta especificada.	
Entrada/ Pasos de ejecución: Se procede a la ejecución del experimento.	
Resultado Esperado: El fichero de alineación es cargado satisfactoriamente.	
Evaluación de la Prueba: Satisfactoria	

Prueba de Aceptación	
Código: HU2_P1	**Historia de Usuario:** Migrar registros entre bases de datos J-ISIS.

Nombre: Migración de los registros.	
Descripción: migra los registros de la base de datos origen hacia la base de datos destino según la equivalencia del fichero de alineación escogido.	
Condiciones de Ejecución: La base de datos origen no puede encontrarse vacía y la base de datos destino no puede encontrarse llena.	
Entrada/ Pasos de ejecución: Se escoge una base de datos origen, una base de datos destino y un fichero de alineación por el cual se llevará a cabo la migración.	
Resultado Esperado: El proceso de migración el llevado a cabo satisfactoriamente.	
Evaluación de la Prueba: Satisfactoria	

Prueba de Aceptación	
Código: HU3_P1	**Historia de Usuario:** Mostrar el progreso de la migración.
Nombre: Mostrar el progreso de la migración.	
Descripción: Muestra en una barra el progreso de la migración.	
Condiciones de Ejecución: Que se lleve a cabo la migración.	
Resultado Esperado: El proceso es llevado a cabo satisfactoriamente.	
Evaluación de la Prueba: Satisfactoria	

3.5 Pruebas Unitarias

Las pruebas unitarias son establecidas antes de escribir el código y son ejecutadas constantemente ante cada modificación del sistema. Los clientes escriben las pruebas funcionales para cada historia de usuario que deba validarse. En este contexto de desarrollo evolutivo y de énfasis en pruebas constantes, la automatización para apoyar esta actividad es crucial (LETELIER, 2006).

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. JUnit se comporta de la siguiente forma con 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 (Peñarrocha, 2015).

3.5.1 Junit

JUnit es una herramienta para Java, desarrollada por Erich Gamma and Kent Beck, adoptada y apoyada por grupos partidarios de la programación extrema, la cual, entre otras cosas, sigue una política de primero probar y luego codificar. Esta y todas las herramientas descendientes de JUnit consisten de una serie de clases que auxilian en la preparación y codificación de casos de prueba y algunos mecanismos auxiliares, que en conjunto permiten ejecutar y verificar el cumplimiento de los casos de prueba. Además, provee una interfaz que permite automatizar la ejecución de grupos de casos de prueba. JUnit ha tenido mucho éxito, por lo que está incorporado en varios IDEs, tales como Eclipse y NetBeans (ÁVILA, 2013).

3.5.2 Pruebas de Ruta Básica

La prueba de ruta o trayectoria básica es una técnica de prueba de caja blanca propuesta por primera vez por Tom McCabe. El método de ruta básica permite al diseñador de casos de prueba derivar una medida de complejidad lógica de un diseño de procedimiento y usar esta medida como guía para definir un conjunto básico de rutas de ejecución. Los casos de prueba derivados para revisar el conjunto básico tienen garantía para ejecutar todo enunciado en el programa, al menos una vez durante la prueba (SOMMERVILLE, 2005).

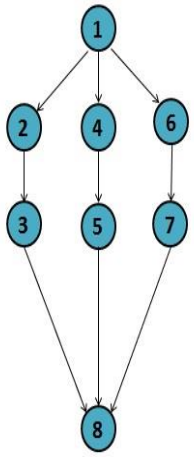
Tabla 13 Prueba de caja blanca

Prueba de caja blanca
Probado por: Mahel Hernández Contreras

Código al que se le aplica:

```
private void jButton1ActionPerformed(java.awt.event.ActionEvent evt) {
    // TODO add your handling code here:

    Migrar migrar = new Migrar(); 1
    String bd = jComboBox1.getSelectedItem().toString(); 1
    String bd2 = jComboBox2.getSelectedItem().toString(); 1

    if (dataProvider.getLastMfn(bd, "UCI") == 1) { 2
        JOptionPane.showMessageDialog(null, "La base de datos origen no puede ser vacia"); 3

        migrar.setVisible(false); 3
        this.setVisible(true); 3
        jComboBox1.setSelectedIndex(-1); 3
        jComboBox2.setSelectedIndex(-1); 3
    } else if (dataProvider.getLastMfn(bd2, "UCI") != 1) { 4
        JOptionPane.showMessageDialog(null, "La base de datos destino tiene que ser vacia"); 5
        migrar.setVisible(false); 5
        this.setVisible(true); 5
        jComboBox1.setSelectedIndex(-1); 5
        jComboBox2.setSelectedIndex(-1); 5
    } else { 6
        migrar.setVisible(true); 7
        this.setVisible(false);  7
    }

} 8
```

Representación en grafo del flujo



Complejidad Ciclomática:
V (G) = (Cantidad de Aristas – Cantidad de Nodos) + 2 = (9 - 8) + 2 = 3

Caminos independientes:

1-2-3-8
1-4-5-8

1-6-7-8

Caso de prueba para el camino básico No. 1	
Descripción: Comprueba que el mfn de la base de datos origen sea igual a 1, de ser así lanza un mensaje.	
Condición de ejecución: Toma una base de datos origen para comprobar que no se encuentra vacía.	
Procedimiento prueba automatizada	
Datos de entrada: java.awt.event.ActionEvent evt	
Evaluación del caso de prueba: Satisfactoria	
Caso de prueba para el camino básico No. 2	
Descripción: Comprueba que el mfn de la base de datos destino sea distinto de 1, de ser así lanza un mensaje.	
Condición de ejecución: Toma una base de datos destino y comprueba que no se encuentra llena.	
Procedimiento prueba automatizada.	
Datos de entrada: java.awt.event.ActionEvent evt	
Evaluación del caso de prueba: Satisfactoria	
Caso de prueba para el camino básico No. 3	
Descripción: Si no se cumple ninguno de los caminos anteriores pasa a la siguiente interfaz.	
Condición de ejecución: Toma una base de datos origen y una base de datos destino y pasa a la siguiente interfaz.	

Procedimiento prueba automatizada.	
Datos de entrada: java.awt.event.ActionEvent evt	
Evaluación del caso de prueba: Satisfactoria	

3.6 Pruebas de validación

Las pruebas de validación se realizaron a través de casos de prueba, los cuales son según **Fuente especificada no válida.***"…un conjunto de acciones con resultados y salidas previstas basadas en los requisitos de especificación del sistema"*. La partición equivalente y gráfico de prueba fueron las técnicas utilizadas para comprobar el funcionamiento del sistema.

Para la realizar estas pruebas se hizo a través de: partición equivalente que no es más que es un método 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. Un caso de prueba ideal descubre de forma inmediata una clase de errores, que de otro modo requerirían la ejecución de muchos casos antes de detectar el error genérico (SOMMERVILLE, 2011).

Tabla 14 Caso de prueba de partición equivalente del RF2 Migrar registros entre bases de datos J-ISIS.

Escenario	Descripción	Datos	Respuesta del sistema	Flujo central
EC 2.1 Migrar registros entre bases de datos J-ISIS de forma correcta.	El usuario escoge una base de datos origen llena y una base de datos destino vacía y selecciona la opción "Siguiente" , donde deberá escoger el formato al que	V Base de Datos Origen Base de Datos Destino	Se muestra el proceso de la migración.	El usuario escoge una base de datos origen y una base de datos destino y selecciona la opción "Siguiente" , donde escoge el formato al

	se desea migrar y selecciona la opción “ Migrar”.	Formato a Migrar		que se desea migrar y muestra el de progreso de la migración.
EC 2.2 Migrar registros entre bases de datos J-ISIS de forma incorrecta.	El usuario escoge una base de datos origen vacía y una base de datos destino vacía y selecciona la opción “Siguiente” , donde deberá escoger el formato al que se desea migrar y selecciona la opción “ Migrar”.	I probando BD1 MARC 21-CEPAL	El sistema muestra un mensaje ¨La base de datos origen no puede ser vacía¨ .	El usuario escoge una base de datos origen y una base de datos destino y selecciona la opción “Siguiente” , donde escoge el formato al que se desea migrar y muestra el de progreso de la migración..
EC 2.3 Migrar registros entre bases de datos J-ISIS de forma incorrecta.	El usuario escoge una base de datos origen llena y una base de datos destino llena y	I	El sistema muestra un mensaje ¨La base de datos destino tiene que ser vacía¨.	El usuario escoge una base de datos origen y una base de datos destino y

	selecciona la opción “Siguiente” , donde deberá escoger el formato al que se desea migrar y selecciona la opción“ Migrar”.	transuci		selecciona la opción “Siguiente” , donde escoge el formato al que se desea migrar y muestra el de progreso de la migración.
		MARC 21		
		MARC 21-CEPAL.		

3.7 Resultados de las pruebas

Todas las pruebas planeadas han sido ejecutadas y todos los defectos que se identificaron han sido tenidos en cuenta. Al finalizar la etapa de pruebas se obtuvo un 100% de no conformidades resueltas, lo que define que la herramienta desarrollada para la migración entre registros bibliográficos almacenados en JISIS cumple con los RF. Entre los resultados de las pruebas están los siguientes:

Tabla 15 Resultados de las pruebas

Pruebas realizadas	**Iteración**	**No conformidades**
Cargar fichero de alineación.	Iteración 1	Desarrollada satisfactoriamente.
Migrar registros entre bases de datos J-ISIS	Iteración 1	Cuando el usuario escogía una base de datos origen vacía y una base de datos destino vacía y da click en el botón “Siguiente”, el sistema no mostraba el mensaje ¨La base de datos origen no puede ser vacía¨
	Iteración 2	Cuando el usuario escogía una base

		de datos origen llena y una base de datos destino llena y da click en el botón “Siguiente” el sistema no mostraba el mensaje ¨La base de datos destino tiene que ser vacía¨.
	Iteración 3	Desarrollada satisfactoriamente.
Mostrar el progreso de la migración.	Iteración 1	Desarrollada satisfactoriamente.

3.7 Conclusiones del capítulo

- ✓ La identificación y solución de no conformidades confirmó la importancia de aplicar pruebas al software.
- ✓ La solución oportuna de las no conformidades, garantizó la calidad de la propuesta de solución.

Conclusiones Generales

Una vez culmina la presente investigación y el desarrollo de la herramienta para la migración entre formatos bibliográficos almacenados en jisis, se puede arribar a las siguientes conclusiones:

- El adecuado uso de la Metodología XP garantizó el cumplimiento de los objetivos propuestos.
- El desarrollo de la Herramienta para la migración entre formatos de registros bibliográficos almacenados en jisis facilita la equivalencia entre formatos bibliográficos.
- Las estrategias de pruebas uilizadas demostraron el correcto funcionamiento de la herramienta implementada.

Recomendaciones

Como resultado final del proceso investigativo, se obtuvo una serie de recomendaciones a tener en cuenta para desarrollos futuros:

- ✓ Realizar un Componente Visual que permita gestionar los ficheros de alineación.
- ✓ Agregar una funcionalidad que permita en la interfaz crear una base de datos en el caso que no se encuentre ninguna vacía.

Referencias Bibliográficas

ALEGSA. 2010. Diccionario de Informática y Tecnología. *Diccionario de Informática y Tecnología.* [En línea] 2010. http://www.alegsa.com.ar/Dic/lenguaje de programacion.php.

ÁVILA, A., CAMILLONI, L., MAROTTA, F., VALLESPIR, D. y APA, C. 2013. *Pruebas Unitarias en Java JUnit y TestNG.* 2013. págs. 46-48.

ÁVILA, A., CAMILLONI, L., MAROTTA, F., VALLESPIR, D. y APA,. 2013. *Pruebas Unitarias en Java JUnit y TestNG.* 2013. págs. pp. 1-7.

Betty Furrie, Departamento de Desarrollo de Bases de Datos de la Follet Software Company. 2001. Conociendo MARC Bibliográfico:Catalogación Legible por Máquina. *Conociendo MARC Bibliográfico:Catalogación Legible por Máquina.* [En línea] 2001. https://www.loc.gov/marc/umbspa/.

BOERAS VELÁZQUEZ, M., CABRERA BARROSO, L., LLANO CASTRO, E. y GONZALEZ SÁNCHEZ. 2012. Aplicando el método de Boehm y Turner vol. 5. 2012, págs. 1-12.

Carlos A. Guerrero, Jorge M. Londoño, Johanna M. Suárez, Luz E. Gutiérrez. 2014. Scielo. *Scielo.* [En línea] 2014. http://www.scielo.cl/scielo.php?script=sci_arttext&pid=S0718-07642014000200008.

Cervantes, Dr Humberto. 2010. [En línea] 2010. http://sg.com.mx/content/view/922..

Dauphin, Jean-Claude. 2014. *J-ISIS Reference Manual.* París : s.n., 2014.

ESCRIBANO, FERNÁNDEZ. 2002. *Introducción a Extreme Programming.* 2002.

FIGUEROA, R.G., SOLÍS, C.J. y CABRERA. 2011. METODOLOGÍAS TRADICIONALES VS . METODOLOGÍAS ÁGILES. 2011, págs. pp.1-9.

FUENTES, L. y TROYA, M. 2009. Lección 1 Desarrollo de Software Basado en Componentes. *Lección 1 Desarrollo de Software Basado en Componentes.* 2009, págs. p 1-22.

GODOY, D.A., BELLONI, E., KOTYNSKI, H., DOS SANTOS, H. y SOSA. 2014. Simulando Proyectos de Desarrollo de Software Administrados con Scrum. 2014, págs. p485-489.

Hilario, Ana Belén Ríos. 2003. *LA ESTRUCTURA CONCEPTUAL DEL REGISTRO BIBLIOGRÁFICO.* España : Universidad de Salamanca, 2003.

Jalón, Javier García de. 2013. Detodoprogramacion. *Detodoprogramacion.* [En línea] 2013. http://www.detodoprogramacion.com/2013.

LETELIER, P. y PENADÉS. 2006. Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP). *Métodologías ágiles para el desarrollo de software: eXtreme Programming (XP).* [En línea] 2006. http://www.cyta.com.ar/ta0502/v5n2a1.htm..

Oracle. 2016. Java. *Java.* [En línea] 2016. https://www.java.com/es/about/whatis_java.jsp.

—. 2017. NetBeans IDE. *NetBeans IDE.* [En línea] 2017. https://netbeans.org/.

Peñarrocha, Miguel. 2015. Una experiencia aplicando Test Driven Development usando una herramienta JUNIT. 2015.

Pivotal, Software. 2016. Let's build a better Enterprise. [En línea] 2016. https://spring.io/.

PRESSMAN, R.S. y MAXIM, B.R.,. 2015. *Software Engineering A practitioner`s approach 8th edition.* 2015.

Pressman, Roger S. 2005. *Ingeniería de Software.Un enfoque práctico.* Sexta . 2005.

2015. RefWorks. *RefWorks.* [En línea] Octubre de 2015. http://biblioguias.biblioteca.deusto.es/c.php?g=149257&p=982226.

Rizzo, Jennifer. 2011. Prezi. *Prezi.* [En línea] 29 de abril de 2011. https://prezi.com/og5clzmnkboc/migracion-de-datos/.

2016. Scribd. *Scribd.* [En línea] 2016. https://es.scribd.com/doc/55186864/Concepto-y-caracteristicas-de-las-bd-documentales.

Simulación de Proyectos de Software desarrollados con XP : Subsistema de Desarrollo de Tareas. **KASIAK, T. y GODOY. 2012.** 2012, págs. 572-576.

SOMMERVILLE. 2011. *Software Engineering Ninth Edition.* 2011.

SOMMERVILLE, IAN. 2005. *Ingeniería de Software.* 7ma Edición. Madrid : s.n., 2005.

Targetware. 2016-2017. Software.com.ar. *Software.com.ar.* [En línea] 2016-2017. http://www.software.com.ar/p/visual-paradigm-para-uml.

VALDEZ HUARACA, A.G., MENDOZA VALDEZ, J.L., TORRES ALARCÓN, S.J., PACHAS LAURA, C.O., MATÍAS SEBASTIAN, J. y ALFARO MEDINA. 2013. *Pruebas de Sistemas y Pruebas de Aceptación.* 2013.

Wesley, Addison. *Extreme Programming Explained.*

Yunta, Luis Rodríguez. 2001. *BASES DE DATOS DOCUMENTALES: ESTRUCTURA Y PRINCIPIOS DE USO.* Madrid : s.n., 2001.