Universidad de Ciencias Informáticas

Herramienta para la migración de bases de datos ISIS a formato MARC.

Trabajo de Diploma

Para optar por el título de:

Ingeniero en Ciencias Informáticas

Autor: Luis Valdés Guerrero.
Tutor: Ing. Edisnel Carrazana Castro

Ciudad de la Habana, Cuba

Junio 2009

Declaración de autoría

Declaramos que somos los únicos autores del trabajo titulado: Herramienta para la migración de bases de datos ISIS a formato MARC.

y autorizamos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

Para que así conste firmo la presente a los ____ días del mes de _____ del año 2009.

_____ _____

Luis Valdés Guerrero Ing. Edisnel Carrazana Castro

Datos de contacto

Ing. Edisnel Carrazana Castro: Profesor graduado de Ing. en Ciencias Informáticas en el año 2007. Ha impartido asignaturas como Programación II e Inteligencia Artificial. Posee categoría docente de Instructor recién graduado; ha cursado postgrados como: Ciencia, Tecnología y Sociedad, Ideología y Política, Docencia e Innovación Universitaria, Metodología de la investigación y como parte de la maestría de Gestión de proyectos ha cursado las asignaturas Técnicas de dirección y Negociación y gestión de la contratación. Ha sido tutor de trabajos de diplomas de cursos anteriores, así como también presidente de tribunales de tesis .Es líder del grupo de producción Gestión bibliotecaria del polo Gestión de la Información y el Conocimiento, cargo que desempeña en la actualidad.

Agradecimientos

Agradezco a los que de una forma u otra han contribuido a la realización de esta tesis, principalmente a Edisnel y Sergio. Y un agradecimiento especial al movimiento de software libre por enseñarme a compartir los más preciado de la humanidad, el conocimiento.

Dedico este trabajo a mi padre por haberme mostrado el camino hacía la ciencia y el conocimiento, a mi madre por apoyarme en toda las decisiones que he tomado en mi vida, a mi hermana por ser mi ejemplo y a mi novia Marianela por siempre darme su amor y su apoyo.

Resumen

Como parte de la migración a software libre que se realiza en nuestro país, siendo la Biblioteca Nacional de Cuba "José Martí" (BNCJM) una de las principales instituciones involucradas en este proceso en el ámbito bibliotecario, pretende implantar el Sistema Integrado de Gestión Bibliotecaria (SIGB) Koha para gestionar todos los procesos bibliotecarios.

Sin embargo, en dicha institución los datos bibliográficos se gestionan con herramientas incompatibles con el SIGB Koha, por lo que se hace necesario migrar toda la información a un formato que pueda ser importado por este sistema.

Es objetivo del presente trabajo, desarrollar un herramienta que facilite el proceso de migración de bases de datos ISIS al formato MARC.

Índice general

Índice de figuras

Índice de cuadros

Introducción

Las bibliotecas son importantes centros de documentación donde se almacena gran parte del conocimiento y se brindan servicios de utilidad a usuarios con diferentes necesidades de información. Con el avance de la informática, el uso de sistemas de gestión bibliotecaria se hace cada vez más importante.

En nuestro país en las bibliotecas se utiliza mayormente un sistema de almacenamiento y recuperación de información llamado CDS/ISIS, el cual debido a al avance de la informática y la cultura de los usuarios, no satisface la demanda de información de estos últimos. Tampoco integra como un único sistema, todos los procesos de gestión bibliotecaria, por lo que el procesamiento de materiales bibliográficos no es óptimo.

Han surgido nuevas alternativas para la automatización de los procesos de gestión bibliotecaria, entre ellas los Sistemas Integrados de Gestión Bibliotecaria (SIGB), los cuales logran gestionar de forma integrada todos los procesos que se realizan en la biblioteca.

Unos de estos sistemas es el SIGB Koha, alternativa libre escogida por la Biblioteca Nacional de Cuba "José Martí" (BNCJM), para la automatización de sus procesos. Koha contiene gran cantidad de funcionalidades, incluye módulos para circulación, catalogación, adquisiciones, publicaciones seriadas, reservas, gestión de usuarios y otras.

Sin embargo para la implantación de Koha es necesario conservar los datos bibliográficos que se almacenan actualmente en bases de datos ISIS, los cuales no pueden ser importados directamente hacia el SIGB Koha, pues este solo importa archivos que contengan registros bibliográficos en formato MARC

De la situación anterior se deduce el siguiente **problema científico** de la investigación: ¿Como realizar la migración de datos de CDS/ISIS a MARC?

El **objeto de estudio** es el proceso de migración de datos bibliográficos, y su **campo de acción** será la migración de la información de CDS/ISIS a MARC.

El **objetivo general** de la investigación es crear una herramienta que automatice la migración de los datos de CDS/ISIS a MARC.

Además del objetivo general se derivan lo siguientes **objetivos específicos**:

- Investigar el modelo de datos utilizados por CDS/ISIS.
- Investigar los formatos MARC.
- Investigar como convertir entre codificaciones de caracteres.
- Investigar como crear, manipular y escribir archivos Marc.

Para llevar a cabo los objetivos se proponen las siguientes **tareas**:

- Realizar una búsqueda bibliográfica detallada sobre los formatos MARC.
- Seleccionar una herramienta que permita el acceso a bases de datos ISIS.
- Analizar las herramientas y tecnologías actuales a utilizar para el desarrollo de la herramienta.
- Implementar la herramienta.

Capítulo 1

Fundamentación teórica

Al actualizar a una nueva versión de una base de datos o de una aplicación, o al cambiar a un nuevo sistema, los datos necesitan ser preservados en este nuevo sistema. El propósito de la migración de datos es transferir datos existentes al nuevo ambiente. Necesita ser transformado a un formato conveniente para el nuevo sistema, mientras que se preserva la información presente en el viejo. [1]

¿Por qué la Migración de Datos?

Muchas circunstancias existen cuando una organización necesita migrar las aplicaciones o las bases de datos. Puede ser que sea tan simple como una mejora a una nueva versión del sistema. O puede ser que implique cambiar a una nueva base de datos o aplicación. Después de una fusión o de una adquisición, a menudo se retiran las aplicaciones redundantes, pero los datos tienen que ser preservados en el sistema de supervivencia. [1]

Procesos de migración de datos

La migración de datos está compuesta por los procesos expuestos en la siguiente figura[2].

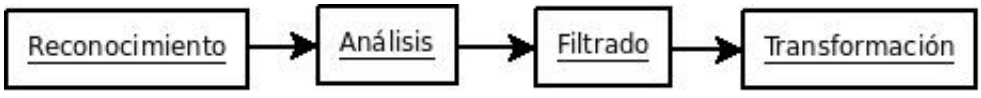

Figura 1.1: Procesos de migración de datos

- **Reconocimiento**: Proceso en el cual se identifica la estructura de de los datos en la fuente de datos.
- **Análisis**: Proceso donde se define una estructura para almacenar de forma óptima los datos de la fuente, de modo que puedan realizarse los siguientes procesos de migración de forma eficiente, usualmente este proceso se mezcla con el de reconocimiento.
- **Filtrado**: Este es un proceso opcional se realiza solo si se quiere aplicarle un filtro a los datos de alguna manera, ejemplo reducir su contenido tomando en cuenta cierto factor.
- **Transformación**: En este proceso se realizan las transformaciones requeridas para que nuestros datos tomen el formato deseado.

Para realizar el proceso de migración de datos hemos tenido en cuenta:

1. Formatos Bibliográficos.
2. Herramientas de manipulación de registros MARC.
3. Librerías de acceso a bases de datos ISIS.
4. Herramientas de desarrollo.
    - Lenguaje.
    - Librerías gráficas.
    - Metodología
    - Herramientas de automatización del proceso de desarrollo.

1.1. Formatos de registros bibliográficos

Como esta tesis se basa en la migración de datos, se debe tener conocimiento de formatos de registros bibliográficos, por lo que se realizó un estudio de los principales

formatos de registros bibliográficos, centrándose en los formatos de la familia MARC, pero antes se verá IIF que es el estándar por el cual se rige la estructura de registros de los formatos MARC y además se verá que son los registros catalográficos.

1.1.1. Registros catalográficos

Es la información que se presenta en una ficha de catálogo de biblioteca[3], estos registros pueden incluir:

- Descripción: Es la descripción bibliográfica de los materiales que son recogidas por los bibliotecarios aplicando las Reglas de Catalogación Angloamericanas, 2a. ed., revisión 2002. En estas descripciones se encuentran las secciones de: titulo, la mención de responsabilidades, la mención de edición, los detalles específicos del material, información de publicación, descripción física, la serie, las notas y los números normalizados.

- Asiento principal y asiento secundario: Son los puntos de acceso a la información del registro, estos son determinados por reglas existentes en la RCAA2 y además esta establece la forma que estos adoptaran. Los puntos de acceso son los puntos de recuperación de información en el catálogo, que los usuarios necesitaran para localizar los materiales.

- Encabezamiento de materiales: Estos no son mas que listas de encabezamiento bajo las cuales se agrupan los materiales haciendo corresponder su tema con un encabezamiento de la lista y así sera mas fácil la búsqueda por tema ya que todos quedaran agrupados.

- Signatura topográfica: La signatura topográfica de un item se selecciona utilizando el Sistema Decimal de Dewey o de La Biblioteca del Congreso(LC). Su objetivo es colocar junto en los estantes los materiales sobre un mismo tema.

1.1.2. IIF(Information Interchage Format)

Es un estándar aprobado el 13 de abril de 1994 por la American National Standards Institute, en este se describe una estructura generalizada para el intercambio

de información entre sistemas de procesamiento, principalmente fue pensado para descripciones bibliográficas de todas formas de materiales, autoridades y en fin para cualquier tipo de dato bibliográfico, este estándar no especifica el contenido de los registros, asignarle significado a cada etiqueta o identificador dependerá de la implementación que se haga del estándar; en este formato toda la información es codificada usando caracteres ASCII(no usa números binarios) y los únicos caracteres de control usados son los valores en byte 29(fin de registro), 30(fin de campo) y 31(delimitador de subcampo). En la figura 2.1 se aprecia la estructura general de IIF[4].

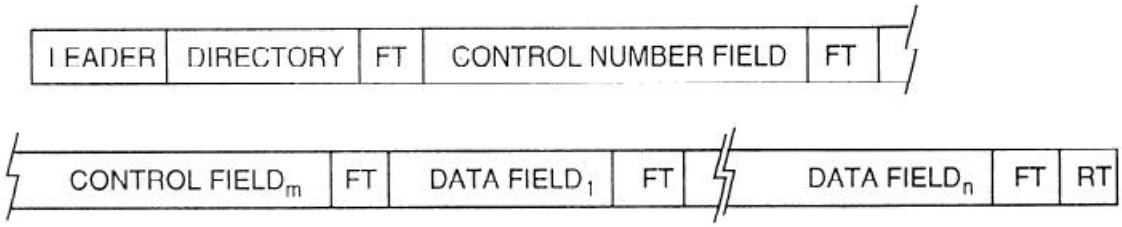

Figura 1.2: Estructura general de IIF

El líder son las primeras 24 posiciones 00-23 de cada registro y suministra información para el procesamiento del registro.

El directorio consiste en una serie de entradas, que son una referencia a cada campo de datos o control, cada entrada consiste en la etiqueta identificadora del campo, su longitud, carácter donde comienza.

FT y RT son caracteres de control usados para determinar el fin de campo y fin de registro respectivamente.

El campo de número de control es una campo que contiene un número único que identifica al registro.

Los campos de control son campos que son identificados por etiquetas que comienzan en 00 o sea serian todos los identificados por las etiquetas 00x, estos preceden a los campos de datos están compuestos solamente de datos y FT, no poseen indicadores, delimitadores.

Los campos de datos no tienen restricciones en cuanto a longitud, número o contenido del campo. La estructura de los campos de datos se puede ver en la figura 2.2.

Los indicadores cuando están definidos ocupan las primeras posiciones del campo.

Lo elementos identificadores de datos cuando están definidos van precedidos por un

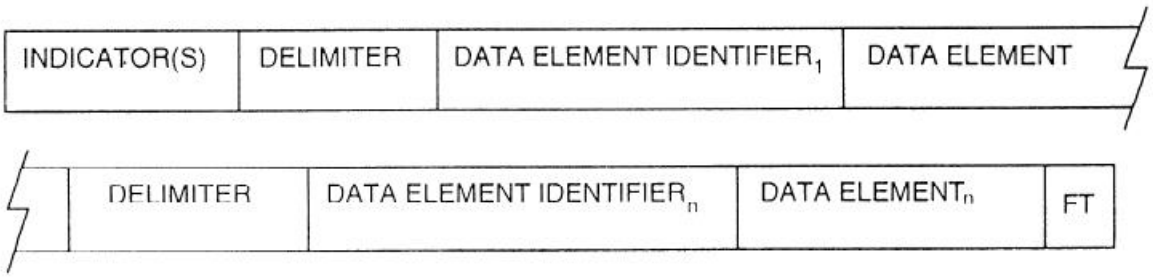

Figura 1.3: Estructura de los campos de datos

delimitador, y seguidos por los elementos de datos, los campos de datos pueden contener múltiples elementos de datos.

1.1.3. MARC

MARC(MAchine Readable Catalogue en español registro catalográfico legible por máquina) es una larga familia de estándares incompatibles, esta familia evoluciono de MARC 1 desarrollado en 1965, esta familia de formatos es incompatible en el sentido que le dan a sus etiquetas y identificadores pero la estructura de datos sigue siendo la misma para todos. Su estructura de registros es una implementación de IIF. Los registros MARC se componen de tres elementos: estructura de registros, designación de contenido y el contenido de datos de los registros. Como ya mencionamos, la estructura de registros es una implementación de IFF, la designación de contenido son los códigos y convenciones para identificar y caracterizar un elemento de datos dentro de un registro, el contenido de datos de los registros son definidos por estándares fuera del formato como International Standard Bibliographic Description (ISBD), Anglo-American Cataloguing Rules, Library of Congress Subject Headings (LCSH) u otras normas de catalogación, aunque algunos datos son definidos por MARC, ejemplo de estos son el Leader, el campo 007 y el campo 008.

MARC 21

El formato MARC 21 para datos bibliográficos contiene elementos de datos para los siguientes tipos de materiales:

- **Libros**

ítems de texto de tipo monográfico, tales como libros encuadernados, electrónicos o microformas.

- **Seriadas**
  ítems de texto con un patrón recurrente de publicación, p.e. publicaciones periódicas, diarios, anuarios.

- **Archivos de computación**
  software, datos numéricos, multimedia, sistemas o servicios en línea. Otros tipos de recursos electrónicos se codifican según su aspecto más significativo, tales como textos (libros o seriadas), cartográficos (mapas), etc.

- **Mapas**
  todo tipo de materiales cartográficos, incluyendo mapas en hojas y globos en forma impresa, manuscritos, electrónicos, y microformas.

- **Música**
  partituras musicales impresas y manuscritas.

- **Registros sonoros**
  registros sonoros no musicales, y registros sonoros musicales.

- **Material visual**
  imágenes y objetos, medios que se proyectan, películas, gráficos de dos dimensiones, objetos tridimensionales, objetos reales.

- **Materiales mixtos**
  fundamentalmente colecciones de archivo y manuscritos con material de formas mixtas.

Los campos de datos se agrupan en bloques de acuerdo con el primer carácter de la etiqueta, la función de los datos dentro del registro está determinada generalmente por el bloque al que pertenece[5].
0XX Información de control, números de identificación y clasificación, etc.
1XX Asientos principales.
2XX Títulos y párrafo del título (título, edición, pie de imprenta).

3XX Descripción física, etc.
4XX Menciones de serie.
5XX Notas.
6XX Campos de acceso temático.
7XX Asientos secundarios diferentes a los de materias y series; campos ligados.
8XX Asientos secundarios de series, existencias, etc.
9XX Reservados para implementación local.

También se da significado a los campos a partir de los dos últimos caracteres de la etiqueta, aunque existen algunas excepciones.
X00 Nombres personales
X10 Nombres corporativos
X11 Nombres de reuniones
X30 Títulos uniformes
X40 Títulos bibliográficos
X50 Términos temáticos
X51 Nombres geográficos

Unimarc

Los campos de datos se agrupan en bloques de acuerdo con el primer carácter de la etiqueta, la función de los datos dentro del registro está determinada generalmente por el bloque al que pertenece.
0XX Identification Block
1XX Coded Information Block
2XX Descriptive Information Block
3XX Notes Block
4XX Linking Entry Block
5XX Related Title Block
6XX Subject Analysis Block
7XX Intellectual Responsibility Block
8XX International Use Block

9XX National Use Block

1.1.4. Otros formatos utilizados en Cuba

En nuestro país se usan algunos formatos aparte del MARC, estos son principalmente CEPAL y FBCBUC.

CEPAL

Este formato fue creado por la Comisión Económica para América Latina y el Caribe(CEPAL) de las Naciones Unidas en la década de los 80, este ha sido muy difundido en la región de América Latina y el Caribe, donde nuestro país no se quedo al margen y este es muy usado. Este dio un impulso al desarrollo de sistemas y redes de información, que facilitó un rápido acceso a las tecnologías de las información en el área. Para la descripción bibliográficas de los documentos que utilizan este formato se utiliza Las Reglas de Catalogación Angloamericanas. Como particularidades las etiquetas identificadoras de campo de este formato tienen dos dígitos en ves de los tres, solamente el campo 100 tiene tres dígitos que utiliza MARC, otra diferencia con respecto a MARC es que no se usan subcampos[6].

FBCBUC

Es el Formato bibliográfico del catálogo de las bibliotecas universitarias cubanas, una solución desarrollada por el CENTIC-MES para lograr hacer un formato bibliográfico propio que respondiera a las necesidades de las bibliotecas universitarias cubanas, ya que el utilizados hasta el momento(CEPAL) no cumplía con los normas internacionales. También es muy conocido con el nombre de Bermello debido al MsC. Luis Bermello Crespo que fue su principal creador[7].

1.2. Herramientas de manipulación de registros MARC

Para la manipulación de de registros de tipo MARC existe herramientas muy conocidas en el ámbito bibliotecario y otros ramas dedicadas a la gestión documental,

todas estas herramientas se ejecutan sobre el sistema operativo Windows, estas aplicaciones son libres de uso pero de código cerrado. En los últimos años ha crecido una tendencia de una parte de la comunidad bibliotecaria que ha optado por soluciones libre, sobre sistemas operativos libres y SIGB libres en ves de los SIGB propietarios que los hacia muy dependiente de las empresas desarrolladoras de SIGB privativos y a un muy alto costo, estas comunidades llevan la migración de sus registros bibliotecarios a base de pequeños script desarrollados en diferentes lenguajes, fundamentalmente Perl y Python debido a su facilidad de uso para estas tareas, estos script son muy específicos para cada institución por lo que no podrían ser reutilizados por otras. En el ámbito del software existen buenos SIGB como KOHA,Emilda, OpenBiblio lo que no existe es una gama de herramientas que apoye el trabajo con registros bibliográficos y demás trabajos de migración de datos bibliográficos, principalmente desde CDS/ISIS o cualquiera de sus implementaciones, ya que este fue el principal software usado desde 1985 que fue su fecha de lanzamiento hasta que está siendo reemplazado en la actualidad por los nuevos SIGB.

Se han escogido las tres herramientas mas conocidas y utilizadas por la comunidad bibliotecaria para representar el estado del arte de las herramientas usadas para tratamiento de datos bibliográficos en formato MARC[8], estas son:

- MARCMaker
- MARCBreaker
- MARCEdit

Viendo las funcionalidades de estas herramientas, analizaremos las que pueden ser útiles para conformar la herramienta propuesta por el autor herramientas y cuales se deberían agregar para conformar una herramienta que cumpla todas las expectativas de los bibliotecarios a la hora de manejar sus datos, que les brinde facilidad de uso y los libre de todo trabajo manual.

1.2.1. MARCMaker

Es una herramienta desarrollada por la Biblioteca del Congreso, su funcionalidad es crear registros MARC a partir de un fichero texto preformateado, se ejecuta sobre

cualquier PC IBM-compatible con un micro 386 o superior con MS-DOS o Windows 95/98/ME/2000 como sistema operativo; MARCMaker acepta ficheros generados en la mayoría de editores de texto.

1.2.2. MARCBreaker

Esta herramientas funciona totalmente de forma contraria a la anterior, toma un fichero con registros en formato MARC y lo transforma en un fichero en texto plano, la salido de cada uno de ellos puede ser tomada como la entrada del otro. También es desarrollada por la Biblioteca del Congreso y se ejecuta sobre las mismas plataformas.

1.2.3. MARCEdit

Esta herramienta solo se ejecuta sobre sistemas operativos de la familia Windows, entre sus características están:

- Un editor de registros MARC.
- Una funcionalidad para generar script, para automatizar el mantenimiento de los registros MARC.
- Exportar datos de MARC a otros formatos como texto plano, XML, Dublin Core, EAD.
- Además incluye un cliente Z29.50.

1.3. ISIS y su larga familia de herramientas

Se mostrará también las características y funcionalidades que brindan la familia de herramientas ISIS[9].

1.3.1. CDS/ISIS

CDS/ISIS es un sistema generalizado de almacenamiento y recuperación, este software presenta versiones para los sistemas operativos MS-DOS y Windows.

Entre sus características podemos destacar:

- Maneja registros, campos y subcampos de longitud variable, salvando así espacio en disco.
- Maneja campos repetidos.
- Interfaz gráfica amigable.
- Permite exportar los datos a texto plano.
- Permite generar una base de datos relacional a parir de sus datos.
- Facilidad en la generación de reportes.
- Un componente de entrada de datos que permite la fácil entrada y modificación de los datos.
- FST para reformateo de campos.

1.3.2. XML2ISIS

Esta herramienta tal como su nombre indica su funcionalidad es importar datos XML para CDS/ISIS adicionando datos a una base de datos ISIS ya existente o adicionando una nueva, es distribuida bajo licencia GPL, escrita con el lenguaje Borland Delphi y se ejecuta sobre Windows.

1.3.3. GenISIS

Esta aplicación fue desarrollada por IBISCUS para la UNESCO en el lenguaje Visual Basic, se ejecuta sobre Windows. De esta herramienta existen dos versiones GenisisWeb y GenisisCD, la primera es una aplicación web que permite hacer consultas sobre una base de datos ISIS y la segunda es una interfaz para almacenamiento de la bases de datos ISIS en CDRom.

1.3.4. IsisAscii

Esta herramienta permite importar archivos ASCII a CDS/ISIS, escrito en Visual Basic y se ejecuta sobre Windows.

1.3.5. IsisMarc

Es una interfaz gráfica para CDS/ISIS pero mucho mas potente para tratar entrada de datos en formato MARC u otros que se le asemejen en estructura, fue desarrollado como un proyecto en conjunto por la Biblioteca del Congreso y el programa SIU del ministerio de educación de Argentina, escrita en C++ y de código abierto, se ejecuta sobre Windows, entre sus principales características podemos citar:

- Multiusuario.
- Soporte para diferentes idiomas.
- Permite campos con múltiples subcampos.
- Cliente Z39.50.
- Soporta copiado/pegado de registros.
- Totalmente personalizable.
- Brinda acceso al mapa de caracteres de Windows.

1.3.6. JavaIsis

Esta es una aplicación cliente-servidor para CDS/ISIS, es escrita en Java tanto el cliente como el servidor, y puede ser ejecutado sobre Windows, GNU/Linux,Unix y Macintosh. Esta aplicación brinda todas las funcionalidades de CDS/ISIS pero un ambiente cliente-servidor.

1.4. Iniciativas de migración en entornos libres

1.4.1. Verónica Lencinas. Córdoba, Argentina

Esta propone los siguientes pasos para realizar la migración de datos del modelo de datos utilizado por CDS/ISIS al formato Marc

1. Generación de un archivo de texto plano de la base de datos con OpenIsis.

2. Reformateo y recodificación del archivo de texto plano.

3. Generación de un archivo MARC21 (ISO2709).

Generación de un archivo de texto plano de la base de datos con OpenIsis.

Para realizar este paso se emplea la herramienta de lineas de consola openisis, esta se encuentra con la librería de funciones Openisis y puede descargarse del su sitio de desarrollo en Sourceforge(http://sourceforge.net/projects/isis/). Esta herramienta permite crear a partir de una base de datos CDS/ISIS un archivo en texto plano con el formato

```
numero_de_campo <caracter_tabulador> datos_del_campo
```

, donde cada registro se divide del otro mediante una linea en blanco. Su modo de uso es muy sencillo, basta con poner el siguiente comando en una terminal:

```
openisis -db baseisis > registros.txt
```

donde baseisis es la base de datos CDS/ISIS que se quiere migrar y registros.txt es el archivo de texto plano donde se quiere guardar la información.

Reformateo y recodificación del archivo de texto plano.

Para esta parte se propone que se utilice un lenguaje con las características de Perl para el buen tratamiento de textos, y codificación de caracteres. O sea se propone que uno mismo cree su propio script de reformateo de datos y codificación.

Generación de un archivo MARC21 (ISO2709).

Para esta fase su autora propone un script realizado por ella misma para generar un archivo Marc a partir del archivo de texto plano tratado el las fases anteriores(ver código del script en el Apéndice A). Este programa se denomina rec2marc.pl y puede encontrarse en la siguiente dirección http://www.koha.com.ar/utilidades.html y para utilizarlos desde una terminal tecleamos el siguiente comando ./rec2marc.pl < registros.txt > registros.mrc

1.4.2. Baiju M

Este autor muestra el procedimiento que uso para migrar una base de datos CDS/ISIS a formato Marc y lo comparte en la red para que sirva de base a los que deban acometer este tipo de tareas, siguiendo los siguientes pasos

1. Exportar el contenido de la base de datos CDS/ISIS a XML
2. Formar un archivo de texto plano formato
3. Formar el archivo Marc a partir del de texto plano obtenido

Exportar el contenido de la base de datos CDS/ISIS a XML

Exportar el contenido de la base de datos CDS/ISIS a XML Esta fase se completa utilizando una funcionalidad de CDS/ISIS para exportar sus bases de datos a XML(nota: solo esta implementada de la versión 1.4 en adelante)

Formar un archivo de texto plano formato

Esta labor se hace empleando un script en Python(ver Apéndice B) desarrollado por el autor, para que el XML obtenido en el paso anterior sea procesado por este script debe encontrarse en la localización que el script pide(../xmls/docs.xml) o modificar el script para que lo lea por otro nombre o desde consola.

Formar el archivo Marc a partir del de texto plano obtenido

Para realizar esta fase nuevamente se utiliza un script realizado por el autor, esta vez en el lenguaje Perl(ver Apéndice C), una vez obtenido el archivo Marc es importado dentro de Koha mediante la utilidad bulkmarcimport.pl.

1.4.3. isis2marc.pl

Este es un script desarrollado por el mantenedor de los modulos de ISIS en Perl para tratar de facilitar la migracio de CDS/ISIS a formato Marc(ver Apendice D). Sus funcionalidad es cojer una bases de datos CDS/ISIS y llevarlar hacia Marc teniendo en

cuenta un XML pasado como parametro donde se especifican los campos que van a ser migrados.

1.5. Conclusiones

Después de analizar estas herramientas, se obtiene como conclusión :

- Las mayoría de este tipo herramientas son libres de uso pero de código cerrado.
- Las iniciativas de migración en entornos libres llevan una curva de aprendizaje enorme, requieren muchos conocimientos de GNU/Linux y programación.
- No existen herramientas para llevar a cabo la migración de datos directamente desde una base de datos ISIS.
- La funcionalidad de reformateo de campos que brinda las FST de ISIS, tiene la limitación de no extraer información. de un campo donde sus elementos estén separados por un separador común. Ej 789;5467;5677, el no puede extraer cada uno de estos números por separados.
- Las bases de datos ISIS se encuentran codificadas en Latin1(WinISIS) y CP850(CDS/ISIS) y es necesario codificar lo registros Marc resultantes de la conversión a UTF8 para poder ser utilizados sin problemas desde KOHA. .

Estas deficiencias encontradas es lo que trataremos de dar solución con la herramienta que se propone, es decir crear una herramienta bajo licencia GPL para que pueda ser mejorada y adaptada por cualquier institución bibliotecaria, con la funcionalidad de poder migrar registros bibliográficos directamente de ISIS a los formatos Unimarc y Marc21, y que se ejecute sobre GNU/Linux.

1.6. Herramientas de desarrollado

1.6.1. Lenguaje

Como lenguaje para desarrollar la herramienta fue escogido el lenguaje Perl por las siguientes características :

- Es necesario integrar la herramienta al SIGB Koha en un futuro,y Koha esta escrita en Perl.
- La potencia que tiene en el manejo de texto debido al trabajo con expresiones regulares.
- Los módulos que tiene para el trabajo con registros MARC, llamados igualmente MARC.
- Los módulos de acceso a bases de datos ISIS que tiene(Openisis y Biblio::Isis).

1.6.2. Librerías gráficas

La librería gráfica que se escogió para el desarrollo de la herramienta fue GTK2, ya que proporciona una sencilla vinculación entre el lenguaje y dicha librería para crear una amigable y bien definida interfaz gráfica.

1.6.3. Librerías de acceso a bases de datos ISIS

Para esta funcionalidad se tuvieron en cuenta dos librerías, estas fueron Biblio::Isis y Openisis

Biblio::Isis

Es escrita totalmente en Perl, lee base de datos de cualquier familia ISIS ya sea CDS/ISIS, WinIsis y IsisMarc, entre sus funciones mas destacadas se encuentra to_ascii que convierte un registro de ISIS a texto plano y la otra mas importante es to_hash que convierte un registro de ISIS a una hash que es una estructura de datos nativa de Perl.

Openisis

Es una librería escrita en C de la que se hizo un bind con algunas funcionalidades para Perl, este es mas rápido en tiempo de ejecución que el ya visto, pero para que funcione hay que tener un compilador de C para compilar la librería de la que depende, por otra parte el bind escrito para Perl tiene un bug que radica en que cuando encuentra un campo vacío lo llena con datos aleatorios.

Visto este resumen de ambas librerías se optó por Biblio::Isis, principalmente porque estaba escrita totalmente en Perl y no se necesita de mas nada que un interprete de Perl para ejecutar la aplicación entera y otro punto que influyó es que OpenIsis tiene un bug que traería datos incorrectos y una parte importante de la aplicación es que los datos se conserven sin modificación del original.

1.7. Metodología de desarrollo

El autor de este trabajo abogó por la utilización de metodologías ágiles para su trabajo, mas específicamente una mezcla entre Scrum y XP. Estas dos metodologías acoplan muy bien juntas, Scrum es una metodología de gestión de software y Xp es una metodología de desarrollo, por tanto toda la planificación y gestión del proyecto será llevada mediante SCRUM, mientras el diseño y desarrollo se apoyará totalmente en XP.
Principales valores defendidos por las metodologías ágiles.

- Individuos e interacciones por encima de procesos y herramientas.
- Software funcional por encima de documentación abundante.
- Colaboración con los clientes por encima de negociación de contratos.
- Respuesta al cambio por encima de seguimiento a planes.

1.8. Herramientas de automatización del proceso de desarrollo

1.8.1. IDE

Como IDE utilizaremos Eclipse 3.2 con un plugin para Perl llamado EPIC, que entre sus características tiene:

- Coloreado de sintaxis.
- Detección de errores y explicación del tipo de error.

- Autocompletación de código.
- Integrado a la documentación de Perl (Perldoc).
- Permite debugear.
- Muestra módulos usados y subrutinas y variables contenidas en el fichero que se está editando.

1.8.2. Control de versiones

Para el control de versiones se usó Subversion y como clientes KdeSVN y un plugin de Eclipse llamado Subclipse. El Subclipse al estar integrado al Eclipse permite trabajar sobre las copias de trabajo como si fueran proyectos normal de Eclipse y desde hay mismo hacer checkout, commit, update y todas las operación hechas normalmente a un repositorio subversion.

1.8.3. Documentación

Para la documentación de la herramienta se utilizará POD(Plain Old Documentation) que es la forma de documentar usada en el mundo de Perl, se utiliza esta para mantener la compatibilidad en cuanto a documentación con KOHA.

Capítulo 2

Características del sistema

En el presenta capítulo se presentan las características del sistema, ya que previamente se ha tratado el estado del arte y las metodologías y herramientas a utilizar, por lo que estamos en condiciones de plantear los requisitos funcionales y no funcionales de la aplicación.

2.1. Objeto de automatización

Con la presente investigación se pretende automatizar los procesos de conversión de datos bibliográficos de ISIS a Marc, así como modificaciones automáticas sobre archivos Marc, digase unión de varios archivos en uno, operaciones de reformateo de campos y conversión de codificación de caracteres. Creando una herramienta que realice estas funcionalidades con el objetivo de facilitarle a los bibliotecarios la migración de sus catálogos de ISIS a Marc.

2.2. Información que se maneja

El contenido que manejara la aplicacíon serán las bases de datos ISIS y archivos Marc con los catálogos bibliográficos de las bibliotecas que los bibliotecarios quieran procesar, así como esquemas elaborados para ejecutar las labores de reformateo de campos, estos esquemas no son mas que el mapeo de campos utilizado en procesos de reformateo.

2.3. Propuesta del sistema

La aplicación que se propone presenta un conjunto de utilidades que facilitan el trabajo a los bibliotecarios así como al personal que trabaja con bases de datos documentales en formato Marc e ISIS, brindándole una interfaz amigable y fácil de usar, brindandoles la funcionalidad de migrar sus registros bibliográficos; también será liberada bajo licencia GPL, para que pueda ser cambiada y mejorada por quien desee para así seguir desarrollándose y pueda ser mas útil a la comunidad bibliotecaria.

2.4. Requisitos funcionales.

En la metodología XP que fue nuestra elección, los requisitos funcionales son llamados ”Historias de usuario”, para definir las historias de usuario utilizaremos la siguiente plantilla.

Historia de usuario	
Número	**Nombre**
Usuario	
Prioridad del negocio	**Riesgo de desarrollo**
Iteración asignada	
Descripción	
Observaciones	

Cuadro 2.1: Plantilla de historia de usuario

Leyenda:

Número: Número de id para las HU, será incremental en el tiempo.

Nombre: Es el nombre de la HU, sirve para identificarla fácilmente tanto para los desarrolladores coma para los clientes.

Usuario: Es el usuario del sistema que utiliza o protagoniza la historia.

Prioridad del negocio: Qué tan importante es para el cliente, se clasifica en Alta, Media y Baja.

Riesgo de desarrollo: Qué tan difícil es para el desarrollador.

Iteración asignada: Iteración a que corresponde.

Descripción: Es la descripción de la historia, detallando las operaciones del usuario y las respuestas del sistema.

Observaciones: Informaciones de interés, como glosarios, detalles del usuario, etc.

De nuestra aplicación se sacaron las siguientes tres historias de usuarios

Historia de usuario	
Número: 1	**Nombre:** Convertir a Marc
Usuario: Bibliotecario	
Prioridad del negocio: Alta	**Riesgo de desarrollo:** Medio
Iteración asignada: 1	
Descripción: El usuario escoge una base de datos en formato ISIS a través de la interfaz de usuario y el sistema convertirá la base de datos a Marc si es una base de datos ISIS valida si no se le notificará al usuario un error de conversión	
Observaciones:	

Cuadro 2.2: HU convertir a Marc

Historia de usuario	
Número: 2	**Nombre:** Unir bases de datos ISIS
Usuario: Bibliotecario	
Prioridad del negocio: Alta	**Riesgo de desarrollo:** Medio
Iteración asignada:2	
Descripción: El usuario escogerá a través de la interfaz de usuario dos bases de datos ISIS y un campo de enlace para cada uno, donde el primer archivo que fue escogido actuará como maestro, el sistema buscara cada registro del segundo archivo que contenga en su campo de enlace el contenido del registro del primero y unirá ambos registros en uno nuevo, dando como salida un nuevo archivo con la unión de los registros de ambos que cumplan que el contenido de sus campos de enlace es igual.	
Observaciones	

Cuadro 2.3: HU Unir bases de datos ISIS

Historia de usuario	
Número: 3	**Nombre:** Reformatear campos
Usuario: Bibliotecario	
Prioridad del negocio: Alta	**Riesgo de desarrollo:** Medio
Iteración asignada:3	
Descripción: El usuario definirá para los campos que tengan la información separadas por un separador común un nuevo campo así como un subcampo, para cada uno de los elementos del campo. Estas correspondencias de campos serán guardadas en ficheros dentro de una carpeta(schemes) y a la hora de realizar la migración el usuario podrá elegir un esquema de reformateo de los definidos, estos ficheros son en texto plano y de fácil edición manual de su contenido.	
Observaciones	

Cuadro 2.4: HU Reformatear campos

2.5. Requisitos no funcionales(RNF).

Los requisitos no funcionales son propiedades o cualidades que enriquecen el producto, estas cualidades son las que hacen que el producto sea mas atractivo, confiable, o usable, por ejemplo, pudiera desearse que que el producto tuviera una interfaz amigable y fácil de utilizar para un usuario con conocimientos básicos de informática.

Los requisitos no funcionales intervienen directamente en el éxito del producto final, ya que ellos influyen directamente en la aceptación del producto por parte del usuario; los requisitos no funcionales están vinculados normalmente a los requisitos funcionales, o sea una ves que se conozcan las funcionalidades del sistema, se puede determinar que elementos adicionarle para que obtenga una mayor calidad y una mayor aceptación.

2.5.1. Requerimientos de interfaz gráfica

- Interfaz sencilla, amigable y fácil de usar.

2.5.2. Requerimientos de usabilidad

- La herramienta debe ser de fácil aprendizaje.
- La herramienta podrá ser usada fácilmente por cualquier persona con conocimientos de bibliotecología.

2.5.3. Requerimientos de reusabilidad

- Debe estar implementada en capas de modo que pueda ser incorporada a KOHA su funcionalidad posteriormente

2.5.4. Requerimientos de soporte

- El código debe estar bien documentado para su desarrollo y mejoramiento futuro.

2.5.5. Requerimientos de confiabilidad

- Debe mantenerse la integridad de los datos originales en el proceso de migración.

2.5.6. Requerimientos de software

Se requiere que la maquina donde deba ejecutarse la aplicación cuente con:

- GNU/Linux
- Un interprete de Perl 5.0 o superior.
- Los módulos de Perl: Marc,Gtk,Biblio::Isis.

2.5.7. Requerimientos de hardware

- Procesador 486 o superior
- 128 Mb RAM
- Monitor tipo VGA o superior

2.6. Modelo de persistencia

2.6.1. Registros Marc

Una parte importante de la aplicación es como están representados los datos que maneja, lo fundamental es conocer como se estructura un registro Marc.
Un registro Marc está conformado por los siguientes elementos:

- Etiquetas: Son números de tres cifras.
- Indicadores: Son números de un dígito, se aplican solo a los campos que tienen valores definidos, a partir del campo 010 todos los campos tienen dos posiciones para indicadores a continuación de la etiqueta, esto daría dos indicadores por cada campo.

- Subcampo: Definen un subcampo dentro de un campo, los subcampos están precedidos por un delimitador de subcampo y un código de subcampo, el carácter utilizado de delimitador es $ y el código son es una letra en minúscula.

Una de las partes más importante de un registro es la cabecera
La cabecera tiene 24 posiciones numeradas de 00 a 23.

00-04 Longitud del registro (la computadora la determina para cada registro)
05 Estado del registro
a = incremento en el nivel de codificación
c = corregido o revisado
d = eliminado
n = nuevo
p = incremento en el nivel de codificación en la prepublicación
06 Tipo de registro
a = material no-manuscrito
c = música impresa
d = música manuscrita
e = material cartográfico
f = material cartográfico manuscrito
g = medio proyectable
i = grabación sonora no-musical
j = grabación sonora musical
k = gráfico no-proyectable bidimensional
m = archivo de computadora
o = conjunto
p = materiales mixtos
r = artefacto tridimensional u objeto natural
t = material manuscrito
07 Nivel bibliográfico
a = parte componente monográfica
b = parte componente de publicación seriada
c = colección

d = subunidad

i = recurso integrante

m = material monográfico

s = publicación seriada

08 Tipo de control

= tipo no especificado

a = archivo

09 Código del esquema de caracteres

= MARC-8

a = UCS/Unicode

10 Conteo de indicadores (siempre es ”2”)

11 Conteo del código de subcampo (siempre es ”2”)

12-16 Dirección base para los datos (la computadora la determina para cada registro)

17 Nivel de codificación

= nivel completo

1 = nivel completo, sin examinar el material

2 = nivel menos que completo, sin examinar el material

3 = nivel abreviado

4 = nivel esencial

5 = nivel parcial (preliminar)

7 = nivel mínimo

8 = nivel de prepublicación

u = desconocido

z = no aplicable

18 Forma de la catalogación descriptiva

= no-ISBD

a = RCAA2

i = ISBD

u = desconocida

19 Requisito del registro ligado

= NO se requiere un registro ligado (para procesar en forma completa este registro)

r = Se requiere un registro ligado (para procesar en forma completa este registro)

20 Longitud de la porción que da la longitud de campo (siempre es ”4”)
21 Longitud de la porción que da la posición del carácter de inicio (siempre es ”5”)
22 Longitud de la porción definida de la implementación (siempre es ”0”)
23 No definida (siempre es ”0”)

2.6.2. Esquemas de reformateo

El esquema de reformateo no es mas que un fichero en texto plano que contiene en cada linea la forma en que mapea cierto campo del nuevo archivo Marc que se quiere obtener, con uno de la base de datos ISIS origen.
Cada linea contendrá seis elementos separados por espacios, los cuales serán descritos a continuación.

1. Campo del nuevo registro marc.
2. Subcampo del nuevo registro marc.
3. Campo del viejo registro en ISIS.
4. Posición dentro del viejo campo.
5. Primer indicador.
6. Segundo indicador.

Así serán conformados los esquemas de reformateo, a continuación un ejemplo de una linea de un esquema de reformateo.

997 a 800 0 # #

Esto significa que el valor del campo 800 en la posición 0 estará contenida en el subcampo ***a*** del campo 997 en el nuevo registro Marc y que contendrá indicadores vacíos. Cuando se habla de posiciones dentro de un campo se hace refiriéndose a lo siguiente, supongamos que el campo 800 tiene el valor 45;678;1234 la posición 0 será ocupada por 45, la 1 por 678 y la 2 por 1234; o sea las posiciones están dadas por el carácter separador ”;”.

Capítulo 3

Desarrollo

3.1. Estilo arquitectónico

Un estilo arquitectónico no es mas que un modelo que define un conjunto de restricciones a los elementos de software y sus restricciones. A continuación veremos las características y justificación del estilo utilizado en nuestra solución.

3.1.1. Arquitectura en capas

Este estilo arquitectónico es uno de los más usados, porque se ha demostrado que organizar los elementos de las aplicaciones en capas independientes puede lograr una mayor eficiencia durante el tiempo de desarrollo y luego durante el mantenimiento. Una capa no es mas que una agrupación de componentes que tiene una responsabilidad bien definida, las capas forman una jerarquía de capas que cumple la siguiente regla: los componentes de una capa solo pueden utilizar componentes de su misma capa o una inferior, nunca de una superior. Con esto se minimiza la creación de redes complejas de dependencias entre módulos y es más fácil la modificación de cada capa por separado.

Ventajas

- Abstracción acerca del origen de los datos.
  Las capas solo se especializan en las funcionalidades que debe brindar sin tener en cuenta el origen de los datos procesados provenientes de otras capas

- Mejor calidad de las aplicaciones.

  Como las aplicaciones son desarrolladas en capas independientes, pueden ser testeadas independientemente y con mas detalle.

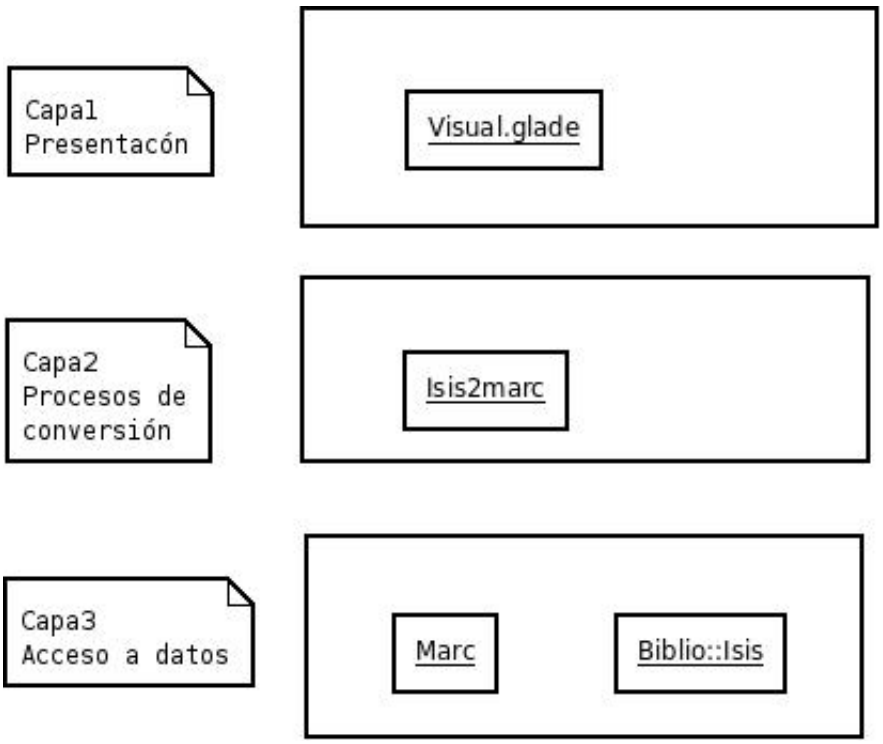

Figura 3.1: Aplicación estructurada en capas

3.2. Modelo de implementación

El modelo de implementación se centra en la organización del software en módulos, en esta vista se separan los elementos de software en unidades mas pequeñas(librerías de programas, subsistemas), así cada unidad puede ser desarrolla por un desarrollador o un pequeño equipo de desarrollo. El modelo de implementación es representado por los diagramas de componentes y subsistemas, y estos componentes están asociados con relaciones de exportación e importación. El modelo de implementación es regido por las siguientes reglas: agrupamiento, particionamiento y visibilidad. Este modelo sirve como base para la asignación de equipos de desarrollo, para la evaluación de costo

y planificación, para la monitorización del avance del proyecto y para cuestiones de reutilización, seguridad, y portabilidad.

A continuación una descripción de cada módulo utilizado

Biblio::Isis: Este módulo lee bases de datos ISIS creadas con CDS/ISIS, WinIsis o IsisMarc.

Marc: Es un módulo para leer, manipular y escribir registros bibliográficos en Marc.

Encode: Módulo para convertir entre codificación de caracteres.

Gtk2: Se usa para acceder a las librerías gráficas de Gtk.

Gtk2::GladeXML: Este módulo es para leer los archivos XML generados por Glade y generar la interfaz gráfica.

threads: Es un módulo para programación multihilo.

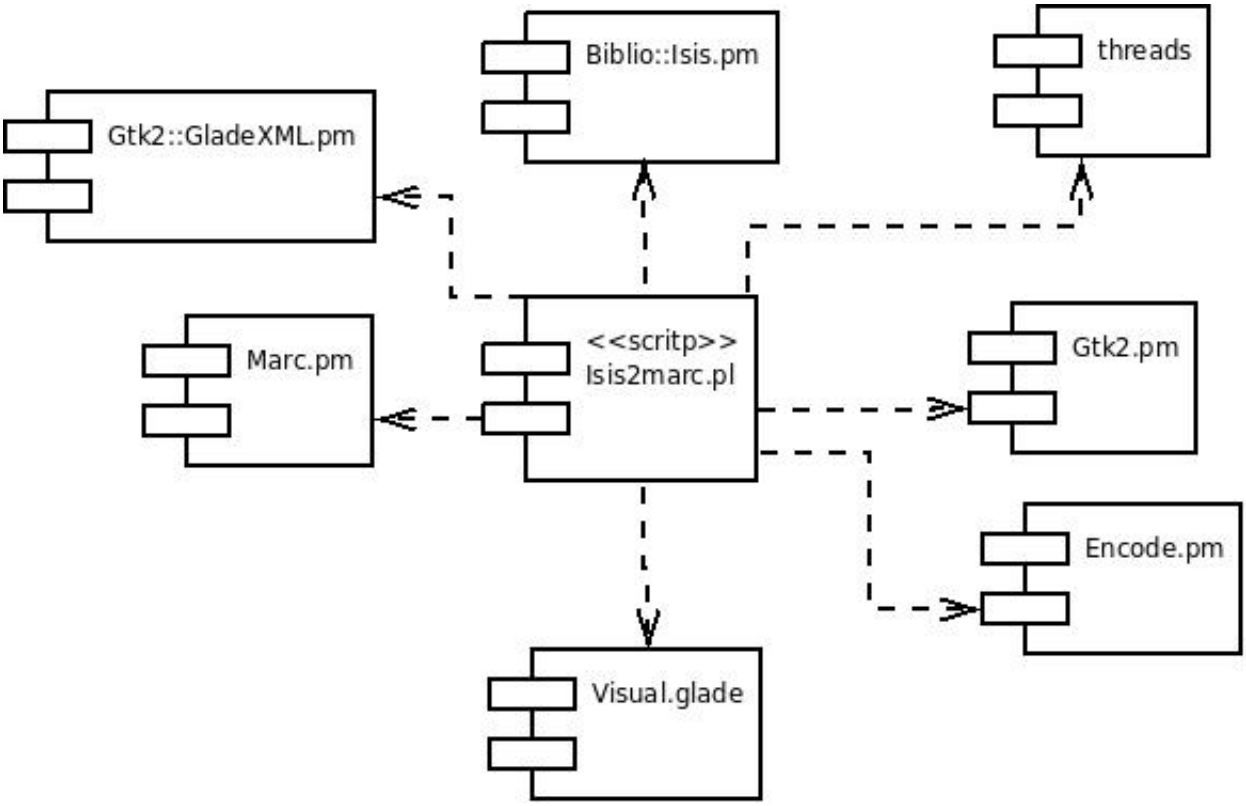

Figura 3.2: Modelo de implementación

3.3. Arquitectura

A continuación en la **figura 4.3** se muestra una vista del flujo de la información, definiendo una arquitectura de tipo filtro.

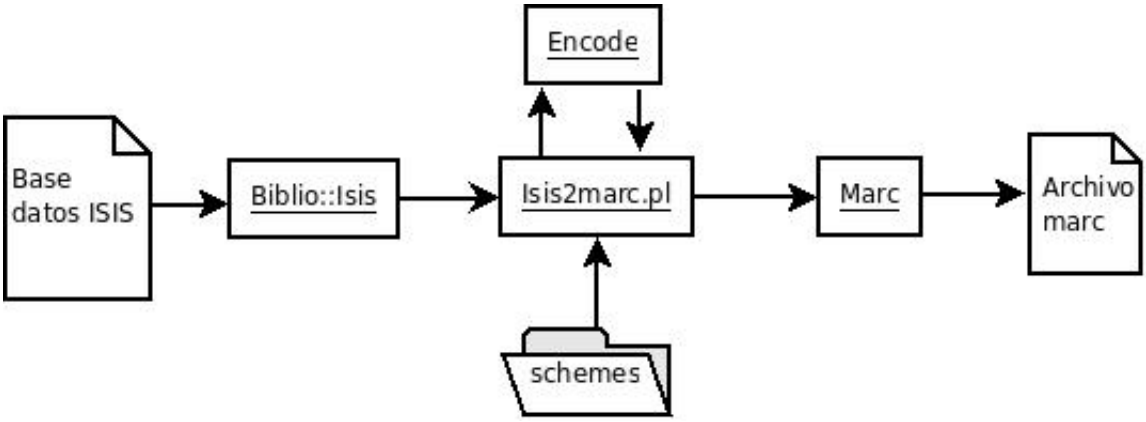

Figura 3.3: Vista de la Arquitectura

Conclusiones

La herramienta creada es una pieza muy necesaria en el proceso de migración al software libre que se está llevando a cabo en nuestro país, y principalmente para el que se quiere impulsar en las bibliotecas cubanas, que cuentan con miles de registros catalográficos que esperan por ser migrados para poder ser importados dentro de KOHA. Esta herramienta es de gran importancia para la comunidad bibliotecaria cubana que no contaba con una de su tipo y es a la ves un gran paso en la migración para el software libre que se esta llevando en nuestro país.

Recomendaciones

Basándonos en la importancia de la herramienta creada se hace las siguientes recomendaciones:

Que se siga el desarrollo de la herramienta y se implemente su incorporación al KOHA.

Que sigan desarrollándose herramientas para el apoyo en la labor de la comunidad bibliotecaria cubana.

Que por parte de proyecto Gestión Bibliotecaria la herramienta sea utilizada y se le muestre a la comunidad bibliotecaria cubana como interactuar con ella.

Glosario

RCAA : Reglas de Catalogación Angloamericanas son las normas de catalogación más completas para el procesamiento de los recursos bibliotecarios, incluyendo los más modernos formatos de los llamados metadatos.

signatura topográfica: Es un código alfanumérico que se coloca de manera visible, normalmente en el lomo del libro, y permite su localización física en la estantería de la biblioteca. La signatura agrupa los documentos por materias en las estanterías lo que te va a facilitar encontrar otros documentos del mismo tema.

Sistema Decimal de Dewey: es un sistema de clasificación de bibliotecas, desarrollado por Melvil Dewey, bibliotecario del Amherst College en Massachusetts, EE. UU., en 1876.

ASCII: El código ASCII (acrónimo inglés de American Standard Code for Information Interchange(Código Estadounidense Estándar para el Intercambio de Información), pronunciado generalmente [áski], es un código de caracteres basado en el alfabeto latino tal como se usa en inglés moderno y en otras lenguas occidentales.

latin1: es una norma de la ISO que define la codificación del alfabeto latino, incluyendo los diacríticos (como letras acentuadas, ñ, ç), y letras especiales (como ß, Ø), necesarios para la escritura de las siguientes lenguas originarias de Europa occidental: afrikaans, alemán, aragonés, asturiano, catalán, danés, escocés, español, feroés, finés, francés, gaélico, gallego, inglés, islandés, italiano, neerlandés, noruego, portugués, sueco y vasco.

cp850: es una página de códigos que se utilizó en Europa occidental, en sistemas como MS-DOS.

ISBD: es un estándar internacional que determina la forma y el contenido de la descripción bibliográfica. Surgió de la Reunión Internacional de Expertos en Catalogación, organizada por el Comité de Catalogación de la IFLA y celebrada en Copenhague en 1969.

LCSH: Library of Congress Subject Headings, describe los encabezamientos de materiales usados por la Biblioteca del Congreso

SIGB: Aplicación informática destinada a automatizar los sistemas y entornos bibliotecarios.

script: En informática, un script es un guión o conjunto de instrucciones. Permiten la automatización de tareas creando pequeñas utilidades. Es muy utilizado para la administración de sistemas UNIX. Son ejecutados por un intérprete de línea de órdenes y usualmente son archivos de texto.

FST: en cada base de datos ISIS define el contenido del archivo invertido correspondiente. Los elementos generados por esta FST, una vez almacenados en el archivo invertido, constituyen el diccionario de términos recuperables para la base de datos.

bind: es una librería para un lenguaje que enlaza este lenguaje con otro librería escrita en otro lenguaje.

Bibliografía

[1] Talend. Migración de datos [online, cited 10 Enero 2009]. Available from World Wide Web: http://es.talend.com/solutions-data-integration/data-migration.php.

[2] CROSS, DAVID. *Data Munging with Perl*. Manning Publications Co., 2001.

[3] LoC. Conociendo MARC Bibliográfico [online, cited 15 Enero 2009]. Available from World Wide Web: http://www.loc.gov/marc/umbspa/um01a06.html.

[4] NISO. ANSI/NISO Z39.2 - Information Interchange Format [online, cited 14 Diciembre 2008]. Available from World Wide Web: http://www.niso.org/kst/reports/standards?step=2\&gid=None\&project_key%3Austring%3Aiso-8859-1=fb7a107043228a342cb704973825aca7bc6ae58d.

[5] LoC. MARC 21 Formats [online, cited 16 Enero 2009]. Available from World Wide Web: http://www.loc.gov/marc/marcdocz.html.

[6] Vera Arendt, Carmen. *Sistema de información bibliográfica de la CEPAL:manual de referencia*, 2003. Available from World Wide Web: http://www.eclac.cl/publicaciones/SecretariaEjecutiva/3/LCL1963PE/lcl1963e.pdf [cited 12 Febrero 2009].

[7] Bermello Crespo, Luis. Procedimiento para la catalogación por copia de revistas en bases de datos a texto completo en Internet [online, cited 12 Enero 2009]. Available from World Wide Web: http://www.bvs.sld.cu/revistas/aci/vol12_6_04/aci03604.htm.

[8] LoC. MARC Records, Systems, and Tools (Network Development and MARC Standards Office, Library of Congress) [online, cited 14 Diciembre 2008]. Available from World Wide Web: `http://www.loc.gov/marc/marctools.html`.

[9] Unesco.
CDS/ISIS database software [online, cited 12 Diciembre 2008]. Available from World Wide Web: `http://portal.unesco.org/ci/en/ev.php-URL_ID=5330\&URL_DO=DO_TOPIC\&URL_SECTION=201.html`.

[10] Openisis. IIF, marc and z39.50 [online, cited 10 Diciembre 2008]. Available from World Wide Web: `http://malete.org/Doc/IIF`.

[11] M, Baiju. Migrating a CDS/ISIS based system to Koha [online, cited 25 Enero 2009]. Available from World Wide Web: `http://www.kohadocs.org/CDS_ISIS_to_Koha.html`.

[12] Parrella Romero, José Miguel. Migrar CDS/Isis a Koha [online, cited 27 Enero 2009]. Available from World Wide Web: `http://tech.groups.yahoo.com/group/koha-es/message/236`.

[13] Lencinas, Verónica. Cómo migrar de CDS/ISIS a Koha [online, cited 2 Febrero 2009]. Available from World Wide Web: `http://www.koha.com.ar/recursos-tiles/cmo-migrar-de-cds/isis-a-koha`.

[14] Knight, Jon. Handling MARC with Perl [online, cited 21 Febrero 2009]. Available from World Wide Web: `http://www.ariadne.ac.uk/issue7/marc/`.

[15] Equipo Gtk2-Perl. Gtk2-Perl [online, cited 24 Febrero 2009]. Available from World Wide Web: `http://gtk2-perl.sourceforge.net/doc/pod/index.html`.

[16] Bassi, Emmanuele. GTK2-Perl Tutorial [online, cited 25 Febrero 2009]. Available from World Wide Web: `http://gtk2-perl.sourceforge.net/doc/gtk2-perl-tut/`.

[17] Peñalver Romero, Gladys Marsi. MA-GMPR-UR2. Metodología ágil para proyectos de software libre. Tesis de diploma, UCI, 2008. Available from World Wide Web: `http://bibliodoc.uci.cu/TD/TD_1309_08.pdf` [cited 1 Marzo 2009].

[18] Rodríguez Villar, Malay. Introducción de procedimientos ágiles en la producción de software en la Facultad 7 de la Universidad de las Ciencias Informáticas. Tesis de diploma, UCI, 2007. Available from World Wide Web: `http://bibliodoc.uci.cu/TD/TD_0693_07.pdf` [cited 1 Marzo 2009].

Apéndice A

rec2marc.pl

```
#!/usr/bin/perl

# rec2marc.pl - script para crear registros en Marc21 a partir
# de una registro del tipo de OpenIsis (tag <tab> content)
#
# (c) 2007 Veronica Lencinas
# GPL, enjoy!

# Uso: ./rec2marc.pl < file

# TODO:
# * poder pasarle un leader y que lo incluya.
# * Doku

#$Version = 0.9;

$ino = 1;
my @tag;
my @campo;
my @lon;
$i = 0;

foreach(<STDIN>){   # lee el archivo provisto por stdin

    if ($_ eq "\n"){  # Fin de registro

@rec = split(/\n/,$rec);  # transforma el registro en un vector

foreach (@rec){
```

```
    ($t,$v) = split(/\t/,$_);
    $v =~ s/\^/\x1f/g;  # reemplazar código de subcampo
    $tag[$i] = sprintf("%03d", $t);    # carga el valor del tag
            $campo[$i] = $v;                    # carga el campo
            $lon[$i] = sprintf("%04d", length($v) + 1);   # carga longi
    $i++;
}
&do_marc;
undef(@tag);
undef(@campo);
undef(@lon);
$i = 0;
$rec = "";
    }
    else {
$rec .= $_ unless $_ eq "\n";   # Colecciona en $rec el registro
    }

}

# ------------------------------------------------

sub do_marc{   # Armar archivo marc

    $sep_cmarc = "\x1e";
    $sep_rmarc = "\x1d";
    $nl = "\n";  # Si va a usar sus registros bajo unix modifique
                   # a "\n".

my $total_campos = @campo;

# inicio_datos = 24 + (total_campos * 12) + 1
my $inicio_datos = sprintf("%05d", 24 + ($total_campos * 12) +1);

# largo_registro = $inicio_datos + suma_de_largo_campos
my $largo_registro = $inicio_datos + 1;

foreach(@lon){
    $largo_registro += $_;
}

# leader: $largo_registro . "0000022" . $inicio_datos . "0004500"
# Si quiere incluir datos dentro del leader, modifique esta linea.
#$leader = sprintf("%05d", $largo_registro) . "nam  22" . $inicio_datos
```

```
$leader = sprintf("%05d", $largo_registro) . "nam a22" . $inicio_datos


# directorio:
my $bas = 0;
undef($directorio);

for ($j=0; $j < $total_campos; $j++){
    $directorio = $directorio . $tag[$j] . $lon[$j] . sprintf("%05d", $
    $bas = $bas + $lon[$j];
}


my $iso = $leader . $directorio . $sep_cmarc;

for ($j=0; $j < $total_campos; $j++){
    $iso = $iso . $campo[$j] . $sep_cmarc;
}

$iso .= $sep_rmarc;


print "$iso$nl";


} # fin sub
#====================================================================
# Fin


FIN:
```

Apéndice B

ciddtf.py : Convert ISIS Document Database to Text File

```
## ciddtf.py : Convert ISIS Document Database to Text File

##Before Parsing :
##Change  RECORDX\xc3\x91_5\xc3\xaf\x03\x12  to RECORD
##Change & to and
##fix problems of <Tag_97> manualy.

import libxml2

record = []

def Record(i, node):
    sub_node = node.children
    while sub_node is not None:
        if sub_node.type == "element":
            record[i][sub_node.name] = sub_node.get_content()
            sub_node = sub_node.next
        else:
            sub_node = sub_node.next

# Memory debug specific
libxml2.debugMemory(1)

isis_data = libxml2.parseFile ('../xmls/docs.xml')
root = isis_data.getRootElement()
node = root.children
i = 0
```

```
while node is not None:
    if node.type != "element":
        node = node.next
        continue
    if node is None:
        break
    if node.name == "RECORD":
        ##print node.get_content()
        record.append({})
        Record(i, node)
        i = i + 1
        node = node.next
    else:
        print "unhandled node in <isis_xml>: " + node.name

isis_data.freeDoc()

# Memory debug specific
libxml2.cleanupParser()
if libxml2.debugMemory(1) == 0:
    print "OK"
else:
    print "Memory leak %d bytes" % (libxml2.debugMemory(1))
    libxml2.dumpMemory()

#####################################################################

record_file = open('../outs/docs.txt', 'w')
tag = ['Tag_4', 'Tag_7', 'Tag_11', 'Tag_12', 'Tag_13', 'Tag_18', 'Tag_

for j in record:
    if j.has_key('Tag_691'):
        record_file.write(j['Tag_691'])
##Check docs without 114 tag
##         if j.has_key('Tag_114'):
##             pass
##         else:
##             print j['Tag_691']
    else:
        record_file.write('_B_L_A_N_K_')
    for h in tag:
        if j.has_key(h):
            record_file.write('\t'+j[h])
        else:
```

```
            record_file.write('\t_B_L_A_N_K_')
    record_file.write('\n')
```

Apéndice C

Script Perl para convertir de texto plano a Marc

```
 package cmftf;
## Create MARC From Text File
## Copyright 2004 Baiju M <baijum81@hailmail.net>
## This program is licensed under GNU GPL.

use strict;
use MARC::Record;

my $input_file = "../outs/docs.txt";
my $output_file = "../outs/docs.out";
my $repeated_items = " ";
open(INFILE, $input_file) or die "Can't open $input_file: $!";
open(OUTFILE, ">>", $output_file) or die "Can't open $output_file: $!"
my $c = 0;
while (<INFILE>) {
    $c++;
    my $biblionumber_ftf_691_090_c, my $title_ftf_4_245_a, my $edition_
    ($biblionumber_ftf_691_090_c, $title_ftf_4_245_a, $edition_ftf_7_25
    if (($biblionumber_ftf_691_090_c !~ /[^\d]/)  and ($title_ftf_4_245
        $repeated_items .= " $biblionumber_ftf_691_090_c";
        my @biblionumber_array;
        open(INF, $input_file) or die "Can't open $input_file: $!";
        my $i = 1;
        while (<INF>) {
            $i++;
            if ($i >= $c) {
            my $biblionumber_ftf_691_090_c2, my $title_ftf_4_245_a2, my
```

```
        ($biblionumber_ftf_691_090_c2, $title_ftf_4_245_a2, $editio
        if (($biblionumber_ftf_691_090_c2 !~ /[^\d]/) and ($title_
            print "$biblionumber_ftf_691_090_c:$biblionumber_ftf_6
            $repeated_items .= " $biblionumber_ftf_691_090_c2";
            push @biblionumber_array, $biblionumber_ftf_691_090_c2
        }
    }
    }
    print "\n";
    close(INF);
    my $record = MARC::Record->new();

    if ($author1ee_ftf_18_100_a =~ /_B_L_A_N_K_/) {
        $author1ee_ftf_18_100_a = "";
    }
    if ($author1se_ftf_19_100_a =~ /_B_L_A_N_K_/) {
        $author1se_ftf_19_100_a = "";
    }

    my $author1_ftf = join(" ", $author1se_ftf_19_100_a, $author1ee

    if ($author1_ftf ne "") {
        my $author1 = MARC::Field->new(
                                        '100','1','',
                                        a => $author1_ftf
                                        );
        $record->add_fields($author1);
    }

    if ($title_ftf_4_245_a !~ /_B_L_A_N_K_/) {
        my $title = MARC::Field->new(
                                       '245','1','2',
                                       a => $title_ftf_4_245_a
                                       );
        $record->add_fields($title);
    }

    if ($edition_ftf_7_250_a !~ /_B_L_A_N_K_/) {
        my $edition = MARC::Field->new(
                                        '250','','',
                                        a => $edition_ftf_7_250_a
                                        );
        $record->add_fields($edition);
    }
```

```
if ($publishcountry_ftf_12_260_a !~ /_B_L_A_N_K_/) {
    my $publishcountry = MARC::Field->new(
                                          '260','','',
                                          a => $publishcountry_
                                          );
 $record->add_fields($publishcountry);
}
if ($publisher_ftf_11_260_b =~ /_B_L_A_N_K_/) {
    $publisher_ftf_11_260_b = "";
}
if ($publishyear_ftf_13_260_c =~ /_B_L_A_N_K_/) {
    $publishyear_ftf_13_260_c = "";
}
my $publisher = MARC::Field->new(
                                 '260','','',
                                 b => $publisher_ftf_11_260_b,
                                 c => $publishyear_ftf_13_260_c
                                 );
$record->add_fields($publisher);

if ($phydescr_ftf_10_300_a =~ /_B_L_A_N_K_/) {
    $phydescr_ftf_10_300_a = "";
}
my $phydescr = MARC::Field->new(
                                '300','','',
                                a => $phydescr_ftf_10_300_a,
                                f => 'BOOK'
                                );
$record->append_fields($phydescr);

if ($voldetails_ftf_110_440_v !~ /_B_L_A_N_K_/) {
    my $voldetails = MARC::Field->new(
                                      '440','','3',
                                      v => $voldetails_ftf_110_
                                      );
    $record->add_fields($voldetails);
}

if ($notes_ftf_97_500_a !~ /_B_L_A_N_K_/) {
    my $notes = MARC::Field->new(
                                 '500','','',
                                 a => $notes_ftf_97_500_a
                                 );
```

```
    $record->add_fields($notes);
}

if ($reccreated_123_508_a !~ /_B_L_A_N_K_/) {
    my $reccreated = MARC::Field->new(
                                    '508','','',
                                    a => $reccreated_123_508_
                                    );
    $record->add_fields($reccreated);
}

if ($keyword_ftf_5_520_a !~ /_B_L_A_N_K_/) {
    my $keyword = MARC::Field->new(
                                '520','','',
                                a => $keyword_ftf_5_520_a
                                );
    $record->add_fields($keyword);
}
if ($role_ftf_38_590_a =~ /_B_L_A_N_K_/) {
    $role_ftf_38_590_a = '';
}
if ($currency_153_590_b =~ /_B_L_A_N_K_/) {
    $currency_153_590_b = '';
}
my $role = MARC::Field->new(
                            '590','','',
                            a => $role_ftf_38_590_a,
                            b => $currency_153_590_b
                            );
$record->add_fields($role);

if ($author2ee_ftf_20_700_a =~ /_B_L_A_N_K_/) {
    $author2ee_ftf_20_700_a = "";
}
if ($author2se_ftf_21_700_a =~ /_B_L_A_N_K_/) {
    $author2se_ftf_21_700_a = "";
}
if ($adiauthors_ftf_24_700_a =~ /_B_L_A_N_K_/) {
    $adiauthors_ftf_24_700_a = "";
}

my $adiauthors_ftf = join(" ", $author2se_ftf_21_700_a, $autho

if ($adiauthors_ftf ne "") {
```

```
        my $adiauthors = MARC::Field->new(
                                        '700','1','',
                                        a => $adiauthors_ftf
                                        );
        $record->add_fields($adiauthors);
    }

    if ($classno_ftf_114_852_k =~ /_B_L_A_N_K_/) {
        if ($classno_ftf_115_852_h =~ /_B_L_A_N_K_/) {
            my $classno = MARC::Field->new(
                                        '852','','',
                                        k => '999.9999'
                                        );
            $record->add_fields($classno);
        }
        else {
            if ($classno_ftf_115_852_h =~ /^\d\d\d/) {
                my $classno = MARC::Field->new(
                                        '852','','',
                                        k => $classno_ftf_1
                                        );
                $record->add_fields($classno);
            }
            else {
                my $classno = MARC::Field->new(
                                        '852','','',
                                        k => '999.9998',
                                        h => $classno_ftf_1
                                        );
                $record->add_fields($classno);
            }
        }
    }
    else {
        if ($classno_ftf_114_852_k =~ /^\d\d\d/) {
            if ($classno_ftf_115_852_h =~ /_B_L_A_N_K_/) {
                $classno_ftf_115_852_h = "";
            }
            my $classno = MARC::Field->new(
                                        '852','','',
                                        k => $classno_ftf_114_8
                                        h => $classno_ftf_115_8
                                        );
            $record->add_fields($classno);
```

```
        }
        else {
            if ($classno_ftf_115_852_h =~ /^\d\d\d/) {
                my $classno = MARC::Field->new(
                                                '852','','',
                                                k => $classno_ftf_1
                                                h => $classno_ftf_1
                                                );
                $record->add_fields($classno);
            }
            else {
                if ($classno_ftf_115_852_h =~ /_B_L_A_N_K_/) {
                    $classno_ftf_115_852_h = "";
                }
                my $classno = MARC::Field->new(
                                                '852','','',
                                                k => '999.9997',
                                                h => $classno_ftf_1
                                                );
                $record->add_fields($classno);
            }
        }
    }

    if ($pubprice_151_952_r =~ /_B_L_A_N_K_/) {
        $pubprice_151_952_r = '';
    }
    if ($dateofentry_ftf_122_952_v =~ /_B_L_A_N_K_/) {
        $dateofentry_ftf_122_952_v = '';
    }
    my @biblionumber_array_tmp = sort @biblionumber_array;
    foreach (@biblionumber_array_tmp) {
        my $biblionumber = $_;
        my $barcode = MARC::Field->new(
                                        '952','','',
                                        b => "MAIN",
                                        d => "MAIN",
                                        p => $biblionumber,
                                        r => $pubprice_151_952_r,
                                        u => $biblionumber,
                                        v => $dateofentry_ftf_122_9
                                        );
        $record->add_fields($barcode);
    }
```

```
        #print "\n";
        print OUTFILE $record->as_usmarc();
    }
    else {
        #print $_;
    }
}
close(OUTFILE);
```

Apéndice D

isi2marc.pl

```
 #!/usr/bin/perl -w

# This utility will convert some (or all, depending of definition in
# configuration XMLfile) fields and subfields with remapping into MARC
# file from one or more CDS/ISIS files
#
# 2004-02-23 Dobrica Pavlinusic <dpavlin@rot13.org>
#
#
# Run without parametars for usage instructions or run without parameta
# and redirect STDOUT to file to create example configuration file like
# this:
#
# ./isis2marc.pl > config.xml
#
# If you want to create unique records, you need to define one or more
# fields as key (which will be used to produce just one record for one
# key)
#
# Keys are global for one run of script (that means for all ISIS databa
# used in one run), but you can write arbitrary values (as opposed to f
# names) inside key tag to produce unique key. For example,
#
# <key>author</key>
# <key>700$a</key>
#
# WARNING: When using <key> tag you can enter field with subfield
# (in format 700$a) just filed name (for fields which doesn't have subf
# like 005) or literal value. Fields which doesn't exist in that record
```

```
# will be skipped, and if key is empty no output record will be produce
#
# So, best way to produce just few record in output is to specify fiel
# doesn't exist at all in ISIS database for key, or just one literal va
#
#
# If ISIS databases are named same as directories in which they
# reside, you can specify just directories (so that shell globing work
# like this:
#
# ./isis2marc.pl config.xml all.marc /mnt2/*/LIBRI
#

use strict;
use OpenIsis;
use MARC;
use XML::Simple;
use Data::Dumper;

if ($#ARGV < 2) {
print STDERR "Usage: $0 config.xml marc_file.iso isis_db [isis_db ...|
print STDERR <<'_END_OF_USAGE_';

isis_db can be path to directory (if ISIS database is called
same as database) which will make shell globing work
or full path to ISIS database (without any extension)

Example configuration file will be dumped to standard output
after this, so you can just re-direct output of this script
to produce config file like this:

$ ./isis2marc.pl > config.xml

_END_OF_USAGE_

print <<'_END_OF_CONFIG_';

<?xml version="1.0" encoding="ISO-8859-2"?>
<!-- template configuration file -->
<mapping>
<record>
<key>700$a</key>
<key>700$b</key>
<field tag="700">
```

```
<indicator1>0</indicator1>
<indicator2>#</indicator2>
<subfield id="a">700$a</subfield>
<subfield id="b">700$b</subfield>
</field>
<field tag="009">
<nosubfield>900</nosubfield>
</field>
</record>

</mapping>

_END_OF_CONFIG_

exit 1;
}

my $xml = new XML::Simple();

my $config_file = shift @ARGV || die "no config file?";

my $config = $xml->XMLin($config_file,
KeyAttr => { subfield => 'id' },
ForceArray => [ 'record', 'field', 'subfield', 'nosubfield' ],
ContentKey => '-content',
) || die "can't open configuration file '$config_file': $!";

my $marc_file = shift @ARGV || die "no marc file?";

my $marc=MARC->new;

# it seems that I can't specify invalid template for 005 and prevent
# output from creating field 005
#$num->add_005s({record=>1});

select(STDOUT); $|=1;

my %stored;
my $total = 0;


foreach my $db_file (@ARGV) {

print "reading '$db_file'";
```

```
if (-d $db_file) {
$db_file =~ s,([^/]+)/*$,$1/$1,;
}

my $db = OpenIsis::open( $db_file );
my $maxmfn = OpenIsis::maxRowid( $db ) || 1;

print " [rows: $maxmfn]\n";

my $progress_len = 50;

my $step = int($maxmfn/$progress_len);
$step = 1 if ($step == 0);

my $new = 0;

for (my $mfn = 1; $mfn <= $maxmfn; $mfn++) {
print "." if ($mfn % $step == 0);
my $row = OpenIsis::read( $db, $mfn );

# unroll this field to in-memory structure data
my %data;

# delete mfn from $row because it's literal value and
# not array, so rest of code would croak
delete($row->{mfn});

foreach my $fld (keys %{$row}) {

foreach my $rec_data (@{$row->{$fld}}) {

while ($rec_data =~ s/\^(\w)([^\^]+)//) {
$data{$fld.'$'.$1} = $2;

# delete last subfield delimiter
$rec_data = "" if ($rec_data =~ /(\^\w*$|\^\w\s*$)/);
}

# record data still exist? it's field without
# subfields, then...
if ($rec_data) {
$data{$fld} = $rec_data;
}
```

```
}
}

# now, create output MARC record(s)

foreach my $cfg_rec (@{$config->{record}}) {

# do we have unique key?
my $key;
foreach (@{$cfg_rec->{key}}) {
if ($data{$_}) {
$key .= $data{$_};
} elsif (! m/^\d{3,4}(\$\w)*$/) {
$key .= $_;
} else {
$key .= "";
}
}

next if ($key && $stored{$key} || $key eq "");

$stored{$key}++ if ($key);


# this will be new record (if needed)
my $num;

# with one or more fields
foreach my $cfg_fld (@{$cfg_rec->{field}}) {

my $new_fld = $cfg_fld->{tag};

#
# first create fields without subfields
#

# with one or more subfields
foreach my $f (@{$cfg_fld->{nosubfield}}) {
next if (! $data{$f});

if (! $num) {
$num=$marc->createrecord();
$new++;
}
```

```
my $i1 = $cfg_fld->{indicator1} || ' ';
my $i2 = $cfg_fld->{indicator2} || ' ';
$marc->addfield({record=>$num,
field=>$new_fld,
i1=>$i1,
i2=>$i2,
value=>$data{$f}
});
}

#
# then create fields with subfields
#

# this will hold subfield values
my @values;

# with one or more subfields
foreach my $new_sf (keys %{$cfg_fld->{subfield}}) {
# field$subfield
my $f = $cfg_fld->{subfield}->{$new_sf};
if ($data{$f}) {
push @values, $new_sf;
push @values, $data{$f};
}
}
next if (! @values);

if (! $num) {
$num=$marc->createrecord();
$new++;
}
my $i1 = $cfg_fld->{indicator1} || ' ';
my $i2 = $cfg_fld->{indicator2} || ' ';
$marc->addfield({record=>$num,
field=>$new_fld,
i1=>$i1,
i2=>$i2,
value=>\@values}
);
}

}
}
```

```
$total += $new;
printf "\t%d (%0.2f%%) t: %d\n",$new,($new*100/$maxmfn),$total;
}

$marc->output({file=>"> $marc_file",'format'=>"usmarc"})
```