Universidad de las Ciencias Informáticas

Facultad 10

Adaptación del Módulo de Circulación del Sistema Integrado de Gestión Bibliotecaria Koha a la Biblioteca Nacional de Cuba "José Martí".

Trabajo de Diploma para optar por el título de

Ingeniero en Ciencias Informáticas.

Autor: Yusdanis Feus Pérez.

Tutores: Ing. Edisnel Carrazana Castro.

Ing. Omar Rey Lazarte.

Ciudad de La Habana, junio de 2009.

"Año del 50 Aniversario del Triunfo de la Revolución"

"Nunca consideres el estudio como una obligación, sino como una oportunidad para penetrar en el bello y maravilloso mundo del saber "

Albert Einstein.

DECLARACIÓN DE AUTORÍA

Declaro que soy el único autor de la presente tesis y reconozco 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.

_____

Yusdanis Feus Pérez

_____ _____

Ing. Edisnel Carrazana Castro Ing. Omar Rey Lazarte

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.

Ing. Omar Rey Lazarte: Profesor graduado de Ing. en Ciencias Informáticas en el año 2007. Ha impartido asignaturas como Ingeniería de Software I y II además de Práctica Profesional actualmente se desempeña como Jefe de colectivo de esta asignatura. Posee categoría docente de Instructor recién graduado; ha cursado postgrados como: Ciencia, Tecnología y Sociedad, Economía Política, RUP Básico, Ética Informática, Innovación Universitaria, Estándares Internacionales, editor de textos Látex. Ha presentado ponencias en eventos científicos como VI Forum de Ciencia y técnica y en Uciencia 2009 además ha sido tutor de trabajos de diplomas de años anteriores así como también ha sido oponente o parte del tribunal de otras tesis .Es líder de desarrollo del equipo de Bases de Datos del Proyecto Misión Portal: Solución integral para la gestión de información centralizada en los medios digitales de PDVSA.

AGRADECIMIENTOS

Quiero agradecer profundamente a todos aquellos que me ayudaron incondicionalmente para que este trabajo saliera adelante.

A mis padres por su amor y comprensión.

A mis tutores por la ayuda prestada cada vez que los necesité y las revisiones al documento.

A mis compañeros Lisvan, Adonys, Yanicet y Yadier por su soporte.

Al grupo 10504 por soportarme durante cuatro años.

A todos los profesores que tuve durante la carrera.

A la Universidad de las Ciencias Informáticas

DEDICATORIA

Dedico este trabajo especialmente a mi abuela Verónica por ser la persona que más quiero en este mundo.

A mis padres Rosaida y Juan por confiar en mí y proveerme todo lo necesario para realizar los estudios.

A mi hermana Yunaika por ser su ejemplo y estar orgullosa de mí en estos momentos.

A mi novia María Emilia por todo el amor que me ha ofrecido.

A toda mi familia porque se lo merece.

RESUMEN

Actualmente la tecnología forma parte de la vida, las bibliotecas no están exentas de este proceso; el desarrollo tecnológico ha revolucionado el trabajo propio de una biblioteca, sobre todo, como resultado de las demandas de los usuarios actuales, quienes exigen una respuesta rápida, directa y relevante a sus necesidades de información. Debido a la complejidad de los procesos que tienen lugar en un biblioteca, muchas en el mundo cuentan con un sistema automatizado, conocido como Sistema Integrado de Gestión Bibliotecaria (SIGB).

La Biblioteca Nacional de Cuba “José Martí” (BNCJM) necesita un SIGB para hacer más fácil los procesos que se realizan en la misma debido a la variedad de servicios que brinda. En el mundo se ha hecho popular en los últimos tiempos el SIGB Koha, por ser uno de los más maduros y sobre todo por estar desarrollado bajo los términos de la GPL (Licencia Pública General de GNU). Este sistema cuenta con un módulo de circulación que básicamente garantiza el control del préstamo de materiales bibliográficos, sin embargo no satisface todas las necesidades de la institución. Con este trabajo se pretende adaptar el módulo de circulación de Koha, de forma tal que la BNCJM lo pueda usar y así prestar a los usuarios un servicio de mayor calidad.

Palabras claves: biblioteca, circulación, GPL, Koha, módulo, servicio, SIGB, usuarios.

ÍNDICE DE CONTENIDO

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

INTRODUCCIÓN

El espectro de temas en el campo de las Tecnologías de la Información y las Comunicaciones (TIC) y las bibliotecas es amplio. Abarca aspectos muy especializados que van desde la recuperación de información, la aplicación de la tecnología al servicio del usuario, la descripción bibliográfica y los metadatos, las interfaces de usuarios o la preservación de los documentos electrónicos, hasta su repercusión social, el mercado y políticas de información. Incluye aspectos más generales como la automatización de bibliotecas y bibliotecas digitales.

Antes de la aparición de la informatización, ya hubo intentos de automatización en bibliotecas. Como consecuencia del elevado número de acervos bibliográficos, de usuarios y de sus diferentes relaciones (préstamos, intercambios, etc), las tareas repetitivas se multiplicaban y fueron estas las que impulsaron la mecanización. El ordenador se ha impuesto en los últimos tiempos como herramienta para realizar estás tareas.

Hoy es posible encontrar alternativas libres para la automatización de los procesos de gestión bibliotecaria, entre ellas Koha, un Sistema Integrado de Gestión de Bibliotecas, el primero de código fuente abierto en desarrollarse. Fue creado en 1999 por Katipo Communications para la Horowhenua Library Trust en Nueva Zelanda. La primera instalación se logró en enero del 2000. Koha proviene del maorí, y quiere decir obsequio, o donación.

La BNCJM es el centro rector de todas las bibliotecas en todo el país y a pesar de esto no cuenta con un sistema automatizado para prestar un mejor servicio. Anteriormente a este trabajo se había hecho el intento de automatizar la gestión bibliotecaria, primero por un grupo de desarrolladores del Instituto Politécnico José Antonio Echeverría y segundo por un equipo de desarrollo de la Universidad Central de las Villas, ninguno de los proyectos terminó.

La BNCJM por sus características se distingue de las bibliotecas convencionales y por tal motivo Koha, que está diseñado de forma estándar para el mayor número de bibliotecas posibles, no satisface todos los

requerimientos de la institución, por lo cual, hay que agregarle nuevas funcionalidades y en ocasiones cambiar la forma de realizar algunos procedimientos. Por lo anterior se identifica que el módulo de circulación de Koha no cubre las necesidades que requiere la BNCJM para ofrecer un buen servicio como situación problémica a resolver.

De la situación anterior se deduce el siguiente **problema científico de la investigación**:

¿Cómo adaptar el módulo de circulación de Koha de forma tal que permita automatizar los procesos de circulación de materiales bibliográficos en la BNCJM?

Para modificar un módulo de un SIGB que responda a las necesidades requeridas por la institución es necesario estudiar los procesos que llevan a cabo los bibliotecarios en la circulación de materiales. Por tanto el **objeto de estudio** de este trabajo se enmarca en el estudio del SIGB Koha, y el **campo de acción** se limita al módulo de de circulación de dicho sistema.

Como **objetivo general de la investigación** se propone adaptar el módulo de circulación del SIGB Koha a las necesidades de la BNCJM.

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

- ✔ Identificar los cambios que son necesarios (críticos) hacer al módulo de circulación.
- ✔ Implementar los cambios al módulo de circulación.
- ✔ Validar los cambios realizados al módulo.
- ✔ Desplegar el SIGB Koha con los cambios realizados en la BNCJM.

Con la adaptación del módulo de circulación de Koha se soluciona el problema en la BNCJM contribuyendo a que los procesos de circulación se realicen de forma más rápida y eficiente.

El desarrollo de las diferentes tareas se ordena teniendo en cuenta la modificación de un sistema que es

software libre, por lo tanto se pretende que el uso de las herramientas y los distintos programas que conlleven a su correcto funcionamiento sean libres y el producto final permita la redistribución, modificación y utilización fuera de cualquier conflicto legal.

Las **tareas** que se proponen para dar cumplimiento a los objetivos propuestos son:

- ✔ Realizar un análisis comparativo de los requisitos funcionales capturados y la funcionalidad del módulo de circulación de Koha.
- ✔ Estudiar los estándares de codificación de Koha.
- ✔ Seleccionar las herramientas a utilizar para la implementación.
- ✔ Diseñar las tablas de la base de datos asociadas al módulo de circulación.
- ✔ Implementar o modificar las librerías de Perl necesarias para garantizar el funcionamiento requerido del módulo de circulación.
- ✔ Realizar las pruebas necesarias al módulo de circulación.

Este trabajo está estructurado en 3 capítulos y anexos, que incluyen todo el trabajo investigativo sobre el SIGB Koha, así como la modificación del módulo de circulación y las pruebas que demuestren que la solución es correcta. A continuación se hace una breve descripción de cada uno de los capítulos.

Capítulo 1: Fundamentación teórica. En esté capítulo se describen las principales características y funcionamiento de Koha, se hace un estudio de instituciones que han optado por este SIGB. Además se hace una breve descripción de las técnicas y tecnologías utilizadas para la adaptación del sistema.

Capítulo 2: Descripción y análisis de la solución propuesta: Este capítulo describe la propuesta de solución al problema científico. Abarca elementos de arquitectura sobre los que se desarrollará la aplicación. Se describen los módulos de Perl y las funciones que forman parte del software, así como la base de datos y sus tablas.

Capítulo 3: Validación de la solución propuesta: En este capítulo se describen la pruebas realizadas para validar la solución. Además se muestran ejemplos de pruebas de caja blanca aplicadas al código del software y de pruebas de caja negra que se realizaron a la interfaz del mismo.

1. CAPÍTULO 1. FUNDAMENTACIÓN TEÓRICA

1.1. Introducción

Varias bibliotecas alrededor del mundo experimentan el uso de Koha, principalmente en escuelas o en bibliotecas especiales de organizaciones no lucrativas. Como muchos paquetes de SIGB de código fuente abierto, tiene el potencial de convertirse en un sistema universal siempre que despierte un interés serio en el mercado de las bibliotecas, comenzando con las bibliotecas públicas. Actualmente, su ayuda multilingüe facilita que se difunda internacionalmente con mucha facilidad. La ayuda técnica se proporciona por medio de la propia comunidad que lo emplea aunque Katipo proporciona foros y listas de correo propias del proyecto.

1.2. Características de Koha

Koha es un avanzado SIGB que contiene gran cantidad de funcionalidades. Incluye módulos para circulación, catalogación, adquisiciones, publicaciones seriadas, reservas, gestión de usuarios y relaciones entre centros de una institución y entre varias instituciones. Es un sistema de código abierto, multiplataforma, considerado además un sistema integrado, integrable y adaptable, que consta con una gran cantidad de funciones y servicios complementarios que han sido incluidas en cada una de sus versiones. Es una aplicación basada en la web lo cual constituye otra de sus características relevantes.

1.2.1. Código abierto

Koha es distribuido bajo la licencia libre GPL, lo que permite a las instituciones contar con una verdadera independencia técnica del proveedor del software, pues el usuario tiene a su disposición el código fuente del sistema. A través del código fuente es posible modificar y adaptar el sistema a partir de las necesidades de cada usuario. Es un sistema adaptable, y cada adaptación o mejora realizada en el software es posible distribuirla libremente, esto facilita la realización de versiones adaptadas localmente o para fines específicos, característica que le da una gran ventaja sobre los sistemas propietarios.

Actualmente Koha es uno de los SIGB más completos utilizado por las instituciones bibliotecarias, y su gran desarrollo se debe precisamente a la creciente comunidad de bibliotecas que colaboran para lograr sus objetivos tecnológicos gracias a su licencia.

1.2.2. Multiplataforma

Koha es un sistema multiplataforma que puede ser instalado en los sistemas operativos:.GNU/Linux, Unix, MacOS y Windows. Una vez instalado, no es necesario seguir instalando programas o componentes externos al sistema para su uso, el único requisito necesario es la existencia de un navegador (Mozilla, Internet Explorer, Ópera, entre otros) en cada PC cliente. Esto permite que las bibliotecas puedan utilizar los servicios del sistema sin importar la plataforma que se tenga, es decir, las bibliotecas no dependerán en ninguna manera de las decisiones empresariales de un proveedor determinado. Koha se podrá utilizar en bibliotecas que cuenten con computadoras con diferentes sistemas operativos, así como para los bibliotecarios será posible realizar los préstamos desde cualquier computadora de la biblioteca. Esta característica trae consigo muchas ventajas para las bibliotecas.

1.2.3. Basado en web

Koha es una aplicación basada en web. Las interfaces de los módulos se basan en tecnologías que cumplen con los estándares definidos por el World Wide Web Consortium, XHTML, CSS y Javascript, lo cual lo convierte en una solución completamente independiente de la plataforma.

La interfaz web personalizada para la administración, reporte y catalogación con la que cuenta esta aplicación permite la búsqueda simple y fácil para todos los usuarios. Posibilita a los usuarios gestionar de manera independiente su cuenta, así como la realización y gestión de reservas y de recursos on-line y físicos con la misma herramienta. Permitirá además la renovación de préstamos, así como crear y compartir listas de libros. El usuario podrá recibir a través de la funcionalidad del correo electrónico las reservas y préstamos vencidos, así como consultar las adquisiciones mediante sus lectores RSS.

1.2.4. Sistema integrado e integrable

Koha es un sistema integrado pues incluye dentro de un mismo sistema todas las funciones de gestión de una biblioteca. A través de su OPAC basado en web es posible consultar el catálogo desde cualquier ubicación una vez que un libro ha sido cargado en el sistema de catalogación así como es posible además para los usuarios informarse en el OPAC si un ejemplar está prestado o disponible.

Además el sistema es integrable, pudiéndose configurar para que se comunique con otros sistemas informáticos. Permite la búsqueda e importación de registros bibliográficos a través del protocolo Z3950 y la integración con el sistema informático de Amazon para la visualización de imágenes de libros.

1.2.5. Sistema adaptable

Koha fue creado para una biblioteca pública. Sin embargo su adaptabilidad ha permitido que el sistema fuera ampliado para el uso de todo tipo de bibliotecas. Actualmente es usado por bibliotecas académicas, especializadas, especiales, escolares y públicas.

Para el caso de bibliotecas muy grandes es posible la utilización de una base de datos adicional para acelerar la búsqueda en catálogos voluminosos. Permite además incluir colecciones que no están en la biblioteca central dentro de un mismo sistema y gestionarlos tan eficientemente como las colecciones propias, pues el sistema desde un principio ha incluido la gestión de sucursales.

1.2.6. Interfaces independientes para Intranet y OPAC

Otra de las características que hacen de Koha uno de los SIGB más completos es la posibilidad de separar el sistema en dos partes: el OPAC o catálogo al público, con el cual los usuarios de las bibliotecas interactúan, es decir, es la parte a la que tienen acceso los usuarios, y la Intranet que es el espacio de trabajo de los bibliotecarios o sea donde los bibliotecarios pueden realizar todas las gestiones características de toda biblioteca. Esta gran funcionalidad se logra a través de la configuración del servidor de aplicaciones web del sistema.

Esto es posible lograrlo de dos formas diferentes, la primera, acceder al mismo sitio a través de puertos diferentes, en este caso se especificaría en cuál puerto estaría la Intranet de la biblioteca. La segunda forma de configurar el servidor sería poniendo ambas partes del sistema en direcciones web diferentes, facilitando así, el acceso de los bibliotecarios y los usuarios.

Los usuarios de la Intranet que como ya se ha dicho son los bibliotecarios, acceden a la misma con un nombre de usuario y una contraseña. Estos mediante la Intranet podrán realizar determinadas operaciones. A cada usuario se le darán permisos para las diferentes operaciones a las cuales tendrá acceso. Los permisos serán definidos de manera muy específica, según la responsabilidad y el servicio de cada cual.

En Koha existen permisos que permiten a los bibliotecarios el uso de una función determinada, permisos que permiten el uso de un grupo de funciones y otros permisos amplios para la gestión. La asignación de permisos es aditiva, es decir, un usuario puede tener diferentes permisos dentro de la lista de estos.

1.2.7. Otras Características de Koha

Koha se caracteriza por contar con una Base de Datos dual, constituye esta un potente instrumento para lograr la organización eficaz de la información. Esta característica permite que se aprovechen las fortalezas de ambos tipos de bases de datos empleadas y asegura que el sistema sea escalable, para lograr así, que soporte el volumen de transacciones de datos de cualquier biblioteca, sin importar el tamaño de los mismos. Las bases de datos relacionales son útiles para el tratamiento de la información estructurada, como los registros catalográficos. Por su lado las bases de datos documentales se utilizan para organizar la información no estructurada como resúmenes, documentos a texto completo, etc. Estas bases de datos utilizan operadores booleanos, operadores relacionales y operadores de proximidad, que son empleados en los OPAC actuales.

Koha utiliza los estándares y protocolos empleados por las bibliotecas, asegurando la interoperabilidad de este, con otros sistemas y tecnologías.

- ✔ MARC21, UNIMARC.
- ✔ Z3950 servidor y cliente, NCIP.
- ✔ Opensearch, RSS y LDAP.
- ✔ Indizadores y analizadores de datos potentes, para optimizar la recuperación de registros.
- ✔ Herramientas de acceso a SGBD y Deep linking.
- ✔ Interfaz personalizada de administración, reporte y catalogación.
- ✔ Linux, MySQL, Zebra, Perl, Apache.

1.3. Koha en otras instituciones

Muchas instituciones y universidades a nivel mundial han adoptado Koha para la automatización de sus bibliotecas basándose en las características de este sistema. El sistema ha implantado en instituciones de países como Francia, Estados Unidos y en algunos países de América Latina, por ejemplo, la Universidad Nacional de la Plata en Argentina y la Universidad ORT de Uruguay, los cuales han obtenido resultados positivos con la implantación del sistema. Todos han encontrado en Koha una gran alternativa para la optimización de sus servicios

1.3.1. Universidad Nacional de la Plata (UNLP)

Hasta el año 2003 el Servicio de Bibliotecas de la facultad de ingeniería de la UNLP estaba compuesto por una Biblioteca Central y 8 Bibliotecas Departamentales. Ambas contaban con grandes volúmenes de datos y la informatización de los catálogos y los servicios presentaba algunas deficiencias, pues era dispar. La biblioteca utilizaba el sistema Biblo que a pesar de poseer características funcionalmente aptas, no contaba con un grado de desarrollo avanzado, por lo que no permitía la gestión integral de toda la unidad de información.

Se necesitaba de la modernización de las bibliotecas y los centros de información así como de la incorporación y adopción de las nuevas tendencias de gestión. Era evidente la necesidad de independizarse del software comercial, debido a los elevados costos de mantenimiento y de actualización de la plataforma. Se necesitaba contar con un formato estándar internacional, específicamente MARC21. Era necesario lograr la extensión de los servicios prestados por la biblioteca a la web, así como habilitar otras opciones de administración de las colecciones, particularmente desde el OPAC. Por las razones antes expuestas se decidió adoptar un sistema de código abierto, que ya hubiera sido probado e implementado por otras bibliotecas y que se adecuara a las necesidades de la institución.

Con la aplicación de sistemas y herramientas (Koha y sus dependencias) de código abierto en la biblioteca de ingeniería de la UNLP ha logrado alcanzar la total independencia del software comercial. La biblioteca ha definido su propia interfaz web para las consultas y acceso a la información por parte de los usuarios, así como para el trabajo de los bibliotecarios desde la Intranet. Se hicieron además algunas correcciones de errores por parte de los desarrolladores y la integración de Koha con el sistema de alumnos SIU-Guaraní. El sistema implantado en la UNLP ha alcanzado un importante grado de éxito, gracias a que se lograron satisfacer las necesidades requeridas por la biblioteca.

1.3.2. Universidad ORT de Uruguay

ORT es una universidad autónoma y la mayor universidad privada de este país. Posee dos bibliotecas especializadas; Centro y Pocitos, para ofrecer servicios a más de 6.000 estudiantes en 5 facultades e institutos, a docentes, graduados e investigadores. Ambas bibliotecas son especializadas y cuentan con más de 45.000 volúmenes, 200 títulos de publicaciones periódicas especializadas en soporte impreso y más de 1.400 vídeos y DVD. Incluyen además bases de datos de publicaciones en línea, las cuales permiten el acceso a artículos de texto completo de más de 5.000 títulos de revistas académicas, así como directorios y perfiles de empresa.

Inicialmente la ORT contaba con personal de informáticos y bibliotecólogos de experiencia, poseían ambas bibliotecas separadas, catálogos independientes. Para la gestión de los procesos de

almacenamiento y la recuperación de la información, más un sistema de préstamos utilizaban MicroISIS.

La universidad necesitaba para mejorar el servicio de las bibliotecas unificar el catálogo y la base de usuarios e integrar el catálogo con el proceso de circulación. Era necesario disponer de un sistema integral de gestión para disponer de un catálogo y reservas en línea y migrar sus datos a un formato que estuviera acorde con los estándares internacionales.

Para la selección del producto a implantar fueron evaluados muchos sistemas teniendo en cuenta la situación en la que se encontraba la biblioteca y las motivaciones antes expuestas, siendo seleccionado Koha por ser el sistema más cercano a los requerimientos definidos por la institución.

Actualmente la ORT cuenta con un sistema extensible del cual dispone de su código fuente, adaptado a las necesidades de la institución y que está en constante crecimiento. La ORT hizo desarrollos propios al sistema como fueron: la adaptación de las interfaces de usuario del OPAC y la Intranet del sistema. Fueron redefinidas además de acuerdo a los requerimientos propios de la universidad los módulos de búsquedas, reportes, reservas, la circulación y suspensiones, entre otros. El sistema fue puesto positivamente en marcha y actualmente la biblioteca de la ORT cuenta con un sistema integrado completo que ha cubierto las expectativas de los usuarios y especialistas.

1.4. Módulos del sistema

Koha contiene con una serie de módulos utilizados para tareas bibliográficas específicas. El sistema cuenta con un módulo de adquisición para facilitar la gestión del proceso de selección y adquisición de fuentes, un módulo de catalogación para gestionar el catálogo de la biblioteca, circulación para la gestión entre otras cosas de los préstamos y devoluciones de materiales, para todo el proceso referente a la gestión de usuarios existe el módulo de control de usuarios, cuenta además con el módulo de informes, el OPAC y el módulo de administración, este último constituye uno de los módulos más importantes debido a que es a partir del cuál se configura todo el sistema en sí.

1.4.1. Adquisición

El módulo de adquisición se encarga de la selección y adquisición de fuentes de información. A través de la selección se gestiona la información bibliográfica sobre ejemplares, editores, y otros datos, y mediante la adquisición la compra, canje o donación de materiales. Este módulo garantiza la disponibilidad de las colecciones pertinentes y la posibilidad de disponer de los depósitos de información y poner todas las fuentes y recursos de información al servicio de los usuarios.

Koha posee dos módulos de adquisición: un módulo de adquisición completo para la gestión presupuestaria, proveedores, multidivisas y un módulo simplificado que es utilizado generalmente por las bibliotecas más pequeñas.

Módulo simplificado

El módulo simplificado está implementado para permitir la adquisición de las obras y para las entradas de estas en el catálogo de manera directa. En este módulo no es posible administrar las cuestiones presupuestarias ni los comandos dirigidos a los proveedores.

Módulo completo

Por su parte el módulo completo de adquisición permite administrar operaciones tales como: los presupuestos y las partidas presupuestarias, los proveedores y los comandos por medio de las cestas de pedido o canastas. Estos comandos se encargan de realizar los procedimientos de recepción de la entrega.

Por otro lado la gestión presupuestaria se encuentra dividida en tres partes: el presupuesto disponible, que es el fondo habilitado para las adquisiciones, el presupuesto comprometido (se refiere a los comandos que se encuentren designados para alguna de las operaciones de adquisición, en dicho caso el presupuesto correspondiente está comprometido), y por último el presupuesto consumido (cuando una orden o comando se recepciona, el importe presupuestario será marcado como gasto). Es importante esclarecer que el importe de la factura que es tomado en la recepción puede ser diferente del importe

inicialmente comprometido.

Koha administra además los comandos para multidivisas, entregas de proveedores, entregas parciales, etc.

En este módulo es posible además desactivar algún proveedor, en dicho caso solo se pueden recepcionar comandos pasados. En Koha el proceso para la gestión presupuestaria no bloquea al usuario, aun cuando se encuentre en estado de rebasamiento. En caso de que el presupuesto que es inicialmente tomado sea rebasado, el cuadro de síntesis señala dicho rebasamiento por producto para el previo conocimiento de dicho estado.

1.4.2. Catalogación

Koha dentro de sus funcionalidades, permite definir diferentes modos de catalogar los materiales bibliográficos. El catalogador debe decidir primeramente qué formatos se van a utilizar, es decir si es MARC o no. Para esto se dispone la catalogación simplificada y la catalogación en MARC. Para catalogar cada uno de los tipos de materiales es posible definir una plantilla de carga de acuerdo a los campos necesarios, opción que también se incluye dentro de este módulo. El mismo dispone de una herramienta para cargar las plantillas, el editor MARC, que permite la opción simplificada y la avanzada para esta operación. Además de estas funcionalidades en este módulo están incluidas otras como: plantillas para campos codificados, lista de valores autorizados, control de autoridades, catalogación por copia, entre otros.

Catalogación simplificada

La catalogación simplificada permite aplicar Koha en una biblioteca sin personal capacitado. Esta va dirigida precisamente al personal sin experiencia en el uso de estos sistemas, los campos en esta catalogación son pocos y el único campo obligatorio es el de título.

Catalogación en MARC

En caso de que se decida por la catalogación en formato MARC, se dispone de dos tipos de formatos: MARC21 y UNIMARC. Aunque para cada uno de estos formatos, los campos de la catalogación ya vienen previamente cargados y configurados, es posible también implementar un nuevo formato MARC que no esté contemplado dentro de la catalogación del mismo nombre.

Plantillas de carga

Según las necesidades y costumbres de cada biblioteca, el editor de catalogación permite una configuración específica. Es posible definir diferentes plantillas de carga, por ejemplo, se pueden definir plantillas diferentes para libros, vídeos, DVDs, mapas, entre otros materiales, que incluyan sólo aquellos campos que se aplican al material catalogado y que responda además, a la política de la biblioteca respecto a un tipo de material específico. Para que el trabajo con las plantillas de carga sea más factible es recomendable definir solamente los campos que sean usados en forma habitual. En caso de que se requiera excepcionalmente de algún otro campo, es posible la edición del registro con la plantilla general, la cual incluye todos los campos MARC, y agregar los datos que no fueron contemplados en la plantilla particular.

Editor MARC

El editor MARC constituye una poderosa herramienta de Koha. Este editor es utilizado para cargar todos los campos que fueron definidos en las plantillas de carga. Este proceso dependiendo de la cantidad de campos definidos podrá demorarse o no. Una vez cargado el Editor de manera completa se dispondrá de todos los campos de la plantilla organizados en diferentes pestañas. Es posible configurar la pestaña donde aparecerá cada campo.

En el caso de que falte algún campo obligatorio en el registro que se quiere cargar, el sistema no permitirá que este sea agregado, pero informará de la cantidad de campos y subcampos que aún no han sido llenados y los coloreará, facilitando así la identificación de los datos faltantes para el catalogador.

Plantillas para campos codificados

Mientras que el editor MARC es utilizado por ser más cómodo para cargar los datos de longitud variable, tales como autores y títulos, los campos y subcampos codificados exigen un mayor control sobre cada una de las posiciones en el campo y las opciones disponibles para cada una de estas posiciones. Para ellos se cuenta con planillas adicionales que pueden ser llamadas desde el editor, logrando de esta manera la codificación correcta de la cabecera de MARC y que los campos pierdan su dificultad.

Lista de valores autorizados

Koha incluye un editor para crear listas de valores autorizados, tales como códigos de idiomas, países, etc. El Editor MARC se puede configurar para que en determinados subcampos sólo cargue los datos de estas listas.

Control de autoridades

Koha permite implementar el control de autoridades a través de Tesauros y a través de bases de datos de autoridades en formato MARC.

Por defecto el formato MARC21 viene con una base de datos de autoridades definida según MARC21. Para trabajar con esta base de datos es necesario primero crear plantillas para diferentes tipos de autoridades. Posteriormente se configura el Editor MARC para enlazar determinados campos de datos con la base de datos de autoridades.

Catalogación por copia

La catalogación por copia es una de las facilidades brindadas por Koha, y permite implementarla de dos maneras diferentes: desde el módulo de catalogación, donde se puede buscar el registro bibliográfico correspondiente a través de Z3950 y la segunda de las opciones posibilita subir los registros bibliográficos al repositorio.

El repositorio es un almacén de registros bibliográficos que es externo al catálogo. Para gestionar la funcionalidad de agregar un nuevo registro, Koha busca en el repositorio un registro que cumpla con las condiciones de búsqueda determinadas. Ya una vez seleccionado un registro del repositorio, este puede ser agregado directamente al catálogo, editado y además podrán ser agregados al mismo otros ejemplares.

1.4.3. Circulación

El módulo de circulación de Koha es el módulo mediante el cual se realizan los préstamos, es también el módulo a través del cual se gestionan los usuarios del sistema.

Además del procedimiento de realización de los préstamos dentro de este módulo de circulación se realizan otros procedimientos característicos de todo tipo de bibliotecas, tales como, las devoluciones, el acceso a las listas de materiales de reserva y las transferencias de materiales entre sucursales.

Tanto los préstamos como las devoluciones en Koha se efectúan de manera muy rápida. Para ambos es necesaria la introducción del número de inventario del ejemplar.

Préstamos

El procedimiento del préstamo a un usuario es preciso primeramente realizar la búsqueda del usuario que lo solicita. Esta búsqueda en Koha se puede realizar a partir de dos criterios: por el número del carné de identificación o por los apellidos del usuario. Siendo esta primera la más efectiva. Una vez seleccionado el usuario se introducirá el número del inventario del ejemplar solicitado. El sistema automáticamente calculará la fecha de devolución del ejemplar, a partir de unas reglas de préstamos que combinan la categoría de usuario y el tipo de material. Aunque es posible ingresar una fecha de devolución diferente a la generada por el sistema.

El sistema verificará la situación del usuario, en caso de que este posea materiales vencidos o de que ya tenga en préstamo la máxima cantidad de materiales permitidos, el sistema no permitirá la realización del préstamo y le avisará al bibliotecario. Aunque es posible realizar el préstamo en una segunda instancia.

Una vez realizado el préstamo se podrá imprimir un ticket o una página con toda la información de préstamos del usuario, donde se incluirán todos los ejemplares que el usuario tenga en su poder así como las fechas de devoluciones de cada ejemplar y las reservas pendientes. Es posible la configuración de Koha para la impresión automática de los tickets o la página después de realizado cada préstamo.

Devoluciones

Para las devoluciones, solo es necesario ingresar el número de inventario del ejemplar al cual se le dará baja. En caso de que este material esté pendiente de reserva, el sistema emitirá un aviso automático. Koha permite el uso de lectores de código de barras, los cuales agilizan aún más este proceso.

Lista de reservas

El módulo de circulación incluye las listas de materiales reservados a los cuales el usuario tendrá acceso en todo momento. Después de cumplido el tiempo estipulado para las reservas por la biblioteca, estas serán dadas de baja automáticamente.

Transferencias

En este módulo se administran además las transferencias de materiales entre sucursales, las cuales no son más que asignaciones permanentes que las bibliotecas pueden realizar a alguna institución o lugar determinado.

1.4.4. Usuarios

El módulo de control de usuarios no es más que el módulo a través del cual se administran todos los procesos relacionados con la gestión de los usuarios, y es conocido por otros nombres como: socios, miembros o control de usuarios. Este módulo permite la realización de búsquedas por apellidos y número de carné de identificación, controla además procesos de gestión de usuarios como el ingreso de nuevos usuarios y la baja de los mismos, así como, la modificación de los registros de usuarios y la administración de los permisos para el acceso al sistema.

Alta y modificación de los datos de los usuarios

Koha consta con un completo formulario para el ingreso y la modificación de los datos de los usuarios, así como una página de información del usuario que brinda todos los datos referentes al mismo.

Información al usuario

El sistema a partir de la página de información del usuario permite administrar otros datos y funcionalidades adicionales referentes al estado de los usuarios. Mediante esta página de información se define el nombre de usuario y la contraseña, para que este pueda tener acceso a las funcionalidades extendidas del OPAC. Permite además cargar las multas a los usuarios, así como renovar simultáneamente varios materiales, visualizar las reservas y el historial de lectura de cada usuario. Es posible agregar hijos al registro de los usuarios, modificar las claves de acceso al OPAC y la asignación de los permisos de acceso al sistema.

Multas e inhabilitaciones

En este módulo se incluyen como ya se ha dicho las multas e inhabilitaciones. Koha permite el cálculo de las multas en forma automática. El monto de estas y la forma de calcularlas es configurado en el módulo de Administración del sistema.

En cambio las inhabilitaciones deben ser gestionadas de manera manual por los bibliotecarios, estas deben ser colocadas en el registro de los usuarios y removidas después de estos. Aunque ambos procesos son efectuados manualmente es muy factible para todos crear un procedimiento automático para llevar a cabo esta tarea.

1.4.5. Reportes

Koha brinda por defecto, una serie de informes pre-configurados y un asistente para la creación de nuevos informes. Esto se realiza desde el módulo de informes, al cual también se le conoce de acuerdo con las diferentes traducciones que se le han dado al sistema como, reportes o estadísticas.

Informes pre-configurados

Koha dispone de informes tales como:

- ✔ Estadísticas de catálogo, que incluye:
  - ✗ Inventarios, que permite generar listas topográficas para el control de inventario.
  - ✗ Catálogo por tipo de material, para informar sobre la cantidad de registros bibliográficos por cada tipo de material.
- ✔ Listas “Top”, que contiene:
  - ✗ Usuarios con más préstamos.
  - ✗ Materiales más prestados.
- ✔ Inactivos.
  - ✗ Usuarios sin préstamos.
  - ✗ Materiales sin préstamo.
- ✔ Estadísticas
  - ✗ Informe del día anterior.
  - ✗ Informe del día de la fecha.
  - ✗ Materiales retrasados en su devolución.
  - ✗ Préstamos por categoría de usuario.
  - ✗ Tiempo promedio de préstamo.

Asistente para crear informes

Koha incluye un asistente para informes que viene pre-configurado para generar estadísticas sobre:

- ✔ Adquisiciones
- ✔ Usuarios o Socios
- ✔ Catálogo
- ✔ Circulación

En cada una de estas opciones se incluyen una serie de opciones para crear la consulta apropiada y obtener un informe adecuado. Los resultados pueden ser visualizados a través de la pantalla o exportados a un archivo.

1.4.6. Opac

El catálogo en línea u OPAC (On-line Public Access) es un sistema computarizado de acceso público que permite la consulta de registros bibliográficos de una biblioteca. Este es denominado en español como Catálogo Público de Acceso en Línea.

Los OPACs son sistemas interactivos ya que la comunicación entre los usuarios y estos es de forma dinámica. Los OPACs proporcionan toda la información necesaria sobre el estado de circulación de los distintos ejemplares disponibles en la biblioteca, es decir si se encuentran prestados o no, la fecha en la que estarán disponibles, etc. Permiten a los usuarios realizar operaciones tales como las reservas de los ejemplares y la realización de consultas. Estos son utilizados también para acceder a otros útiles de información bibliográfica como por ejemplo bases de datos bibliográficas tanto en discos ópticos como en línea y a los catálogos pertenecientes a otras bibliotecas.

Koha incluye entre sus módulos un OPAC completo con amplias funcionalidades. El OPAC está disponible tanto para el usuario de la biblioteca como para los bibliotecarios. El acceso al OPAC es a través de un

navegador web. Para la publicación del catálogo en la web no se necesita de un módulo adicional pues al igual que todo el sistema está basado en tecnología web.

Debido a que el OPAC de Koha es compatible con todos los estándares del consorcio de la WWW es posible visualizarlo utilizando cualquier navegador.

1.4.7. Administración

Para lograr una configuración satisfactoria del sistema es preciso que se configure primeramente el módulo de Administración ya que este módulo de Koha constituye el núcleo de la funcionalidad del sistema. Desde este módulo es posible configurar todo el sistema. Los nombres de las diferentes secciones pueden diferenciarse entre una versión del sistema y otra, a causa de las diferencias en las traducciones. A partir de él es posible definir todos los parámetros del sistema para poner en funcionamiento el mismo. El módulo de administración incluye los parámetros utilizados para la configuración del sistema.

1.5. Herramientas, lenguajes y tecnologías a utilizar

En el desarrollo de todo sistema informático es de vital importancia la selección de las herramientas, lenguajes y tecnologías a utilizar, paso que garantizará, de realizarse correctamente, un óptimo desempeño del sistema. Para la modificación del sistema al cual se refiere este trabajo, la selección se realizó teniendo en cuenta la infraestructura tecnológica de la UCI y valorando que el sistema estará orientada a funcionar sobre el sistema operativo GNU/Linux.

1.5.1. Perl

Perl, Practical Extraction and Report Language (Lenguaje Práctico para la Extracción e Informe) es un lenguaje de programación diseñado por Larry Wall y creado en 1987. Perl fue originalmente desarrollado para la manipulación de texto y en la actualidad es ampliamente utilizado en la administración de sistemas y en el desarrollo web. Perl toma características del C y del lenguaje interpretado bash. La estructura

completa de perl deriva de lenguaje C, perl es un lenguaje imperativo, con variables, expresiones, asignaciones y bloques de códigos delimitados por llaves. Perl tiene una gran potencia en la manipulación de textos debido a que incluye expresiones regulares que facilitan el trabajo con textos.[1]

Perl soporta la implementación de módulos que permiten en gran medida la separación de las funcionalidades del sistema y la reutilización de código.

1.5.2. HTML

El HTML, Hyper Text Markup Language (Lenguaje de Marcas de Hipertexto) es un lenguaje muy sencillo que permite describir hipertexto, es decir, texto presentado de forma estructurada y agradable, con enlaces que conducen a otros documentos o fuentes de información relacionadas, y con inserciones multimedia; es el lenguaje que se utiliza para presentar información en la World Wide Web (WWW).

La descripción se basa en especificar en el texto la estructura lógica del contenido (títulos, párrafos de texto normal, enumeraciones, definiciones y citas), así como los diferentes efectos que se quieren dar (cursiva, negrita, o un gráfico determinado) y dejar que luego la presentación final de dicho hipertexto se realice por un programa especializado (como Internet Explorer o Mozilla Firefox).

HTML es un lenguaje de marca, o sea, una manera de expresar la información de un documento (por ejemplo: información sobre los vínculos del hipertexto y sobre formato) en el documento mismo. Los lenguajes de marca, usan etiquetas que son marcas que se ubican dentro del texto y que brindan información de despliegue. De este modo, el Lenguaje Marca de Hipertexto es una forma específica de usar etiquetas para ofrecer información sobre un documento.

Este lenguaje indica al navegador donde colocar cada texto, cada imagen o cada video y la forma que tendrán estos al ser colocados en la página. Así que no es más que una serie de etiquetas que se utilizan para definir la forma o estilo que se quiera aplicar al documento.

1.5.3. JavaScript

Se trata de un lenguaje de programación del lado del cliente, porque es el navegador el que soporta la carga de procesamiento. Gracias a su compatibilidad con la mayoría de los navegadores modernos, es el lenguaje de programación del lado del cliente más utilizado.

Es un lenguaje de programación interpretado por el navegador que se utiliza para controlar su apariencia y manipular los eventos que ocurran en su ventana. A través del JavaScript se pueden conseguir interesantes efectos en las páginas web, validar formularios, abrir y cerrar ventanas, cambiar dinámicamente el aspecto y los contenidos de una página, realizar cálculos matemáticos sencillos.

Con este se puede crear efectos especiales en las páginas y definir interactividades con el usuario. El navegador del cliente es el encargado de interpretar las instrucciones JavaScript y ejecutarlas para realizar estos efectos e interactividades, de modo que el mayor recurso, y tal vez el único, con que cuenta este lenguaje es el propio navegador. [2]

Es un lenguaje de programación bastante sencillo y pensado para hacer las cosas con rapidez, a veces con ligereza. Incluso las personas que no tengan una experiencia previa en la programación podrán aprender este lenguaje con facilidad y utilizarlo en toda su potencia con sólo un poco de práctica.

Entre las acciones típicas que se pueden realizar con este lenguaje se tienen dos vertientes. Por un lado los efectos especiales sobre páginas web, para crear contenidos dinámicos y elementos de la página que tengan movimiento, cambien de color o cualquier otro dinamismo. Por el otro, JavaScript permite ejecutar instrucciones como respuesta a las acciones del usuario, con lo que se puede crear páginas interactivas con programas como calculadoras, agendas, o tablas de cálculo. [3]

Es un lenguaje con muchas posibilidades, permite la programación de pequeños scripts, pero también de programas más grandes, orientados a objetos, con funciones, estructuras de datos complejas. Además, pone a disposición del programador todos los elementos que forman la página web, para que éste pueda acceder a ellos y modificarlos dinámicamente. [4]

Con JavaScript el programador se convierte en el verdadero dueño y controlador de cada cosa que ocurre en la página cuando la está visualizando el cliente.

1.5.4. CGI

Interfaz común de pasarela (en inglés Common Gateway Interface, abreviado CGI) es una importante tecnología de la World Wide Web que permite a un cliente (explorador web) solicitar datos de un programa ejecutado en un servidor web. CGI especifica un estándar para transferir datos entre el cliente y el programa. Es un mecanismo de comunicación entre el servidor web y una aplicación externa cuyo resultado final de la ejecución son objetos MIME. Las aplicaciones que se ejecutan en el servidor reciben el nombre de CGIs.

Las aplicaciones CGI fueron una de las primeras maneras prácticas de crear contenido dinámico para las páginas web. En una aplicación CGI, el servidor web pasa las solicitudes del cliente a un programa externo. Este programa puede estar hecho en cualquier lenguaje que soporte el servidor, aunque por razones de portabilidad se suelen usar lenguajes de script. La salida de dicho programa es enviada al cliente en lugar del archivo estático tradicional.

CGI ha hecho posible la implementación de funciones nuevas y variadas en las páginas web, de tal manera que esta interfaz rápidamente se volvió un estándar, siendo implementada en todo tipo de servidores web.

1.5.5. Principales módulos de Perl

Un módulo es un conjunto de operaciones que pueden ser utilizadas en cualquier script en Perl simplemente llamando al módulo al que se hace referencia y las operaciones que de él nos interesa utilizar. Es evidente que dentro de un script se pueden llamar diversos módulos según lo que efectivamente le sirva a quien esté programando. En el sistema se utilizan varios módulos de Perl, unos escritos por la comunidad de Perl y certificados internacionalmente, y otros escritos por la comunidad de desarrollo de Koha.

1.5.5.1. CGI

Este módulo utiliza objetos de Perl5 para que sea más fácil llenar los formularios web y analizar su contenido. Este paquete define objetos CGI, entidades que contienen la cadena de consulta actual y otras variables de estado. Utilizando los métodos de un objeto CGI, se pueden examinar las palabras claves y los parámetros pasados a un script y crear formularios cuyos valores iniciales son tomados de la consulta actual (lo que da la preservación de la información de estado). El módulo ofrece funciones de acceso directo que producen lenguaje HTML, reduciendo la mecanografía y lo errores de codificación. También proporciona algunas de las funcionalidades más avanzadas de los CGI, incluyendo soporte a la carga de archivos, cookies, hojas de estilo en cascada y marcos.

Existen dos formas de programación con el módulo CGI, uno es el estilo orientado a objetos y el el otro es el estilo funcional u orientado a funciones.

1.5.5.2. HTML::Template

Permite hacer el uso de platillas HTML de forma simple y natural. Se extiende el HTML con algunas etiquetas nuevas como: <TMPL_VAR>, <TMPL_LOOP>, <TMPL_INCLUDE>, <TMPL_IF>, <TMPL_ELSE> y <TMPL_UNLESS>. Un archivo escrito en HTML y con estas etiquetas es llamado plantilla (en Inglés template). Generalmente se guardan separados de los scripts, que pueden ser creados por otra persona. El uso de este módulo permite rellenar los valores de las variables, los ciclos y las ramas declaradas en las plantillas. Este módulo es uno de los más importantes porque permite separar el diseño (el HTML) de los datos generados por los scripts escritos en Perl. Este módulo nos da una gran posibilidad de unirlo con el módulo CGI y facilitar en gran medida la programación web. El módulo es licenciado bajo los términos de la GPL.

1.5.5.3. DBI

DBI (Interfaz de bases de datos independiente) es un módulo de acceso a bases de datos para el lenguaje de programación Perl. Este define un conjunto de funciones, variables y convenciones que ofrecen una

interfaz de base de datos consistente e independiente de la base de datos que se esté utilizando, es decir, DBI es la interfaz independiente para base de datos de Perl, lo cual no significa que no haya otros, pero , normalmente todo lo que se puede hacer con un módulo (que no use DBI para acceso a bases de datos) , se puede hacer con este, de forma más fácil y portable.

Es importante destacar que DBI es una capa de "pegamento" entre una aplicación y uno o más módulos controladores de base de datos. Es el módulo controlador el que hace mayor parte de la verdadera labor. El DBI proporciona una interfaz estándar y el marco de los controladores a operar dentro (Ver **Anexo 1**).

1.5.5.4. DBD

DBD (Driver de Base de Datos-Data Base Driver) es el driver de la base de datos, es decir, se encarga de llevar a cabo lo que se pide en que se haga en nuestro script Perl (usando DBI) en una base de datos específica. Existe un módulo DBD para cada tipo de base de datos, dicho módulo se encarga de pasar las peticiones que realizamos en DBI a peticiones a la base de datos sobre la que se esté trabajando. Para trabajar con una base de datos determinada nos hace falta tener controlado el módulo DBD correspondiente, por ejemplo, para trabajar con la base de datos MySql hace falta el módulo DBD-MySql.

1.5.6. MySQL

MySQL es un sistema gestor de bases de datos relacionales, licenciado bajo la GPL de GNU. Su diseño multihilo le permite soportar una gran carga de forma muy eficiente. Este gestor de bases de datos es, probablemente el más usado en el mundo del software libre, debido a su gran rapidez y facilidad de uso. Esta gran aceptación es debida, en parte, a que existen infinidad de librerías y otras herramientas que permiten su uso a través de gran cantidad de lenguajes de programación, además de su fácil instalación y configuración. [5]

Las principales características de este gestor de bases de datos son las siguientes: [6]

- ✔ Aprovecha la potencia de sistemas multiprocesador, gracias a su implementación multihilo.

- ✔ Soporta gran cantidad de tipos de datos para las columnas.
- ✔ Dispone de API's en gran cantidad de lenguajes (C, C++, Java, PHP, etc).
- ✔ Gran portabilidad entre sistemas.
- ✔ Gestión de usuarios y contraseñas, manteniendo un muy buen nivel de seguridad en los datos.

Principales ventajas:

- ✔ Velocidad al realizar las operaciones, lo que le hace uno de los gestores con mejor rendimiento.
- ✔ Bajo costo en requerimientos para la elaboración de bases de datos, ya que debido a su bajo consumo puede ser ejecutado en una máquina con escasos recursos sin ningún problema.
- ✔ Facilidad de configuración e instalación.
- ✔ Soporta gran variedad de Sistemas Operativos.
- ✔ Baja probabilidad de corromper datos, incluso si los errores no se producen en el propio gestor, sino en el sistema en el que está.
- ✔ Conectividad y seguridad.

Principales desventajas:

- ✔ Un gran porcentaje de las utilidades de MySQL no están documentadas.
- ✔ No es intuitivo, como otros programas (ACCESS).

1.5.7. Apache

Apache es el servidor web hecho por excelencia, su configurabilidad, robustez y estabilidad hacen que cada vez millones de servidores reiteren su confianza en este programa. [7]

La licencia Apache es una descendiente de la licencias BSD (Berkeley Software Distribution), no es GPL (General Public License). Esta licencia permite hacer lo que se desee con el código fuente (incluso productos propietarios). [8]

Algunas razones por las que este software libre es grandemente reconocido en muchos ámbitos empresariales y tecnológicos son: [9]

- ✔ Corre en una multitud de Sistemas Operativos, lo que lo hace prácticamente universal.
- ✔ Apache es una tecnología gratuita de código fuente abierto. El hecho de ser gratuita es importante pero no tanto como que se trate de código fuente abierto. Esto le da una transparencia a este software de manera que si se quiere ver que es lo que se está instalando como servidor se puede saber, sin ningún secreto, sin ninguna puerta trasera.
- ✔ Es un servidor altamente configurable de diseño modular. Actualmente existen muchos módulos para Apache que son adaptables a este, y están ahí para que sean instalados cuando sea necesario.
- ✔ Trabaja en gran medida con Perl, PHP y otros lenguajes de script. También trabaja con Java y páginas jsp. Teniendo todo el soporte que se necesita para tener páginas dinámicas.
- ✔ Apache permite personalizar la respuesta ante los posibles errores que se puedan dar en el servidor. Es posible configurar Apache para que ejecute un determinado script cuando ocurra un error en concreto.
- ✔ Tiene una alta configurabilidad en la creación y gestión de logs. Permite la creación de ficheros de log a medida del administrador y de este modo se puede tener un mayor control sobre lo que sucede en el servidor.

1.5.8. Motor de búsqueda

Koha utiliza Zebra, el mismo, como cualquier otro motor, lo que hace es un indexar direcciones de texto estructurado. Es un motor de recuperación de alto rendimiento, de fines generales. Lee registros estructurados en una variedad de formatos de entrada (XML, MARC) y permite el acceso a ellos con expresiones booleanas exactas de la búsqueda y consultas de texto plano.

Zebra soporta grandes bases de datos, de más de diez gigabytes de datos y diez millones de registros. Soporta además actualizaciones incrementales seguras de las base de datos.

Sus desarrolladores (Index Data) permiten acceder a los datos indexados en Zebra a través de herramientas desarrolladas por ellos como yaz y php/yaz o simplemente utilizando el protocolo de comunicación Z3950. Zebra es distribuido bajo la licencia GPL por lo que puede ser utilizado libremente por cualquier persona.

1.5.9. UML

La Tecnología de Orientación a Objetos constituye la base de la reutilización de código por medio de componentes. UML (Lenguaje de Modelación Unificado o Unified Modeling Language, en inglés) es el lenguaje estándar adoptado por el OMG (Object Management Group) y mundialmente aceptado para la descripción de los "planos" de software. UML es un lenguaje gráfico para visualizar, especificar, construir y documentar cada una de las partes que comprende el desarrollo de software. UML posee formas de modelar conceptos como lo son procesos de negocio y funciones de sistema, además de aspectos concretos como lo son escribir clases en un lenguaje determinado, esquemas de base de datos y componentes de software reusables.

1.5.10. Quanta Plus

Quanta Plus es una herramienta de desarrollo web para el entorno de escritorio KDE, aunque se puede usar en otros entornos de escritorio. Está diseñado para el desarrollo web y rápidamente se está

convirtiendo en un editor maduro y con muchas características.

1.5.11. Anjuta

Anjuta es un entorno integrado de desarrollo (IDE) desarrollado por Naba Kumbar en 1999. Fue desarrollado para programar en C y C++ en sistemas GNU/Linux aunque actualmente admite lenguajes como C#, Perl y Python. Su principal objetivo es trabajar con GTK y en el escritorio GNOME, además ofrece un gran número de características avanzadas de programación. Anjuta es un software libre, liberado bajo la licencia GPL.

Anjuta incluye:

- ✔ Administrador de proyectos.
- ✔ Asistentes.
- ✔ Plantillas.
- ✔ Depurador interactivo.
- ✔ Editor de texto; que verifica y resalta las sintaxis escritas.
- ✔ Extensión para Subversion (control de versiones).
- ✔ Interprete de comandos propio.

A pesar de incluir abundantes utilidades y tener una amplia variedad de funcionalidades es un sistema bastante estable y muy ligero en el consumo de recursos. [10]

1.6. Metodología de desarrollo adoptada

Cuando se realiza proyectos de software de gran envergadura, es necesario basarse en una metodología de desarrollo de software que ayude a organizar y planificar todo el proceso para poder obtener un

producto de óptima calidad y clientes satisfechos con el resultado. En este epígrafe, se describirá las principales características de la metodología RUP (Rational Unified Process) pues es la que se va a utilizar para dar solución al problema plateado por la complejidad y el tamaño de mismo, también que previamente se había hecho el diseño del módulo de circulación utilizando esta metodología robusta.

La metodología RUP, es un proceso de desarrollo de software y junto con el Lenguaje Unificado de Modelado (UML), constituye la metodología estándar más utilizada para el análisis, implementación y documentación de sistemas orientados a objetos. El RUP divide el proceso de desarrollo en ciclos, obteniendo un producto al final de cada ciclo, cada ciclo se divide en fases que finalizan con un hito donde se debe tomar una decisión importante (Ver **Anexo 2**). Este se caracteriza por ser iterativo e incremental, estar centrado en la arquitectura y guiado por los casos de uso. Incluye artefactos (que son los productos tangibles del proceso como por ejemplo, el modelo de casos de uso, el código fuente, etc.) y roles (papel que desempeña una persona en un determinado momento, una persona puede desempeñar distintos roles a lo largo del proceso).

1.7. Conclusiones

En este capítulo se ha cumplido el objetivo principal, mostrar al lector de forma general los aspectos teóricos a tener en cuenta para la implementación de lo que se quiere. Con la intensión de ubicarlo en el contexto del SIGB Koha, del cual se da una breve explicación de su funcionamiento. Se mencionan algunas instituciones que han optado por la adaptación de Koha como solución a sus problemas. Es muy importante aclarar que ninguno de esos sistemas adaptados cumple con las necesidades de la BNCJM, primero porque son instituciones diferentes y segundo porque tienen diferencias en el modo de trabajo, pero sirven de mucha experiencia, permitiendo mediante el uso de la tecnología, el intercambio de ideas. También se describen las tecnologías y herramientas necesarias para la modificación de Koha, así como la metodología de desarrollo adoptada.

2. CAPÍTULO 2. DESCRIPCIÓN Y ANÁLISIS DE LA SOLUCIÓN PROPUESTA

2.1. Introducción

En este capítulo se tratan varios aspectos más relacionados con el desarrollo de los cambios al subsistema de circulación. Se comienza realizando una valoración del diseño propuesto por el analista según lo que necesita la BNCJM. Se hace una descripción general de la arquitectura del sistema y los requisitos funcionales. Además de la descripción de los módulos de Perl y operaciones utilizadas, así como el diseño de las base de datos y la descripción de las tablas relacionadas con el módulo de circulación.

2.2. Valoración crítica del diseño propuesto por el analista

El diseño propuesto por el analista fue considerado correcto, fácil de entender y muy específico pues permitió adquirir una comprensión de los aspectos relacionados con los requisitos funcionales y no funcionales del sistema, que sirvieron de base para el comienzo del desarrollo del mismo.

Se obtuvieron además, diagramas como los de clases de diseño que mostraron la parte estática del sistema, la representación de las clases y sus relaciones, artefactos que facilitan en gran medida la implementación de las mismas dada sus definiciones.

Unido a esto se encuentran los diagramas de interacción, que explican la colaboración que existe entre las clases y cómo son llamados los métodos y sentencias dentro de cada una, de los cuales se obtiene la información necesaria para conocer el orden de las acciones a implementar. Mientras más legible sea el diseño propuesto más fácil es la adaptación de los programadores al mismo y por ende se agiliza el proceso de traducción de clases del diseño a clases de implementación.

2.3. Concepciones generales de la arquitectura

Una Arquitectura de Software, también denominada Arquitectura Lógica, consiste en un conjunto de patrones y abstracciones coherentes que proporcionan el marco de referencia necesario para guiar la construcción del software para un sistema de información.

La Arquitectura de Software establece los fundamentos para que analistas, diseñadores y programadores trabajen en una línea común que permita alcanzar los objetivos del sistema de información.

Se selecciona y diseña con base en objetivos y restricciones. Los objetivos son aquellos prefijados para el sistema de información, pero no solamente los de tipo funcional, también otros objetivos como la mantenibilidad, auditabilidad, flexibilidad e interacción con otros sistemas de información. Las restricciones son aquellas limitaciones derivadas de las tecnologías disponibles para implementar sistemas de información. Unas arquitecturas son más recomendables de implementar con ciertas tecnologías mientras que otras tecnologías no son aptas para determinadas arquitecturas.

La arquitectura de software define, de manera abstracta, los componentes que llevan a cabo alguna tarea de computación, sus interfaces y la comunicación ente ellos. Toda arquitectura debe ser implementable en una arquitectura física, que consiste simplemente en determinar qué computadora tendrá asignada cada tarea.

La arquitectura de software, tiene que ver con el diseño y la implementación de estructuras de software de alto nivel. Es el resultado de ensamblar un cierto número de elementos arquitectónicos de forma adecuada para satisfacer la mayor funcionalidad y requerimientos de desempeño de un sistema, así como requerimientos no funcionales, como la confiabilidad, escalabilidad, portabilidad, y disponibilidad.

En fin, la Arquitectura de Software representa un alto nivel de abstracción común que la mayoría de los participantes, si no todos, pueden usar como base para crear entendimiento mutuo, formar consenso y comunicarse entre sí. En sus mejores expresiones, la descripción arquitectónica expone las restricciones de alto nivel sobre el diseño del sistema, así como la justificación de decisiones arquitectónicas

fundamentales. [11]

2.3.1. Estilos o patrones arquitectónicos presentes en la solución. Características

2.3.1.1. Modelo Cliente/Servidor

La arquitectura Cliente/Servidor es un modelo para el desarrollo de sistemas de información en el que las transacciones se dividen en procesos independientes que cooperan entre sí para intercambiar información, servicios o recursos. Se denomina cliente al proceso que inicia el diálogo o solicita los recursos y servidor al proceso que responde a las solicitudes.

Los clientes realizan generalmente funciones como:

- ✔ Manejo de la interfaz de usuario.
- ✔ Captura y validación de los datos de entrada.
- ✔ Generación de consultas e informes sobre las bases de datos.

Por su parte lo servidores realizan las siguiente funciones:

- ✔ Gestión de periféricos compartidos.
- ✔ Control de accesos concurrentes a bases de datos compartidas.
- ✔ Enlaces de comunicaciones con otras redes de área local o extensa.

Siempre que un cliente requiere un servicio lo solicita al servidor correspondiente y éste le responde proporcionándolo. Normalmente, pero no necesariamente, el cliente y el servidor están ubicados en distintos procesadores. Los clientes se suelen situar en ordenadores personales y/o estaciones de trabajo y los servidores en procesadores departamentales o de grupo.

Entre las principales características de la arquitectura Cliente/Servidor se pueden destacar las siguientes:

- ✔ El servidor presenta a todos sus clientes una interfaz única y bien definida.
- ✔ El cliente no necesita conocer la lógica del servidor, sólo su interfaz externa.
- ✔ El cliente no depende de la ubicación física del servidor, ni del tipo de equipo físico en el que se encuentra, ni de su sistema operativo.
- ✔ Los cambios en el servidor implican pocos o ningún cambio en el cliente.

En la aplicación se utiliza este modelo ya que la aplicación estará ubicada en un servidor central en la BNCJM, el cual se encargará de dar respuesta a las peticiones realizadas por los clientes que utilicen la aplicación.

2.3.1.2. Modelo Vista Controlador (MVC)

Modelo Vista Controlador es un patrón de arquitectura de software que separa los datos de una aplicación, la interfaz de usuario, y la lógica de control en tres componentes distintos. El patrón MVC se ve frecuentemente en aplicaciones web, donde la vista es la página HTML y el código que provee de datos dinámicos a la página, el modelo es el Sistema Gestor de Base de Datos y la lógica de negocio, y el controlador es el responsable de recibir los eventos de entrada desde la vista.

Modelo: El modelo administra el comportamiento y los datos del dominio de aplicación, responde a requerimientos de información sobre su estado (usualmente formulados desde la vista) y responde a instrucciones de cambiar el estado (habitualmente desde el controlador).

Vista: Esta presenta el modelo en un formato adecuado para interactuar, usualmente la interfaz de usuario.

Controlador: Este responde a eventos, usualmente acciones del usuario e invoca cambios en el modelo y probablemente en la vista.

En la **Figura 1** se muestra la interpretación gráfica del MVC según [12].

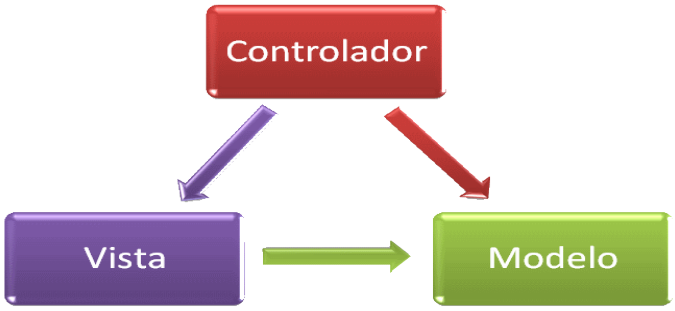


Figura 1: Modelo Vista Controlador

Tanto la vista como el controlador dependen del modelo, el cual no depende de las otras clases.

En fin, en MVC el controlador es el encargado de redirigir un procesamiento determinado para cada petición que reciba. Por tanto, el controlador debe poseer algún modo de conocer la correspondencia entre peticiones y posibles respuestas; deberá tener un mapa de peticiones y respuestas. El modelo representa la aplicación que responde una petición: los datos y las reglas de negocio. Por último, la vista define la forma de mostrar la información al usuario. En el caso del patrón MVC, el procesamiento se lleva a cabo entre sus tres componentes. El controlador recibe una petición y decide quién la lleva a cabo en la capa del modelo. Una vez que el modelo termina las operaciones pertinentes, devuelve el control de ejecución al controlador, y este envía los resultados a la capa de la vista para que muestre los resultados al usuario .

El uso de este patrón arquitectónico tiene gran ventaja ya que al separar la lógica del negocio y el acceso a datos de las interfaces de usuario, permite que cuando uno de estos se afecte, influya sobre los demás. También agrupa y clasifica los scripts según el rol que desempeñan dentro de la aplicación.

2.4. Requisitos no funcionales del sistema propuesto

Los requerimientos no funcionales son propiedades o cualidades que el producto debe tener. Debe pensarse en estas propiedades como las características que hacen al producto atractivo, usable, rápido o

confiable, por ejemplo, pudiera desearse que el sistema responda dentro de un intervalo de tiempo especificado o que obtenga los resultados de los cálculos con un nivel de precisión dado.

En muchos casos los requerimientos no funcionales son fundamentales en el éxito del producto. Normalmente están vinculados a requerimientos funcionales, es decir una vez se conozca lo que el sistema debe hacer, se puede determinar cómo ha de comportarse, qué cualidades debe tener o cuán rápido o grande debe ser.

Los requerimientos no funcionales forman una parte significativa de la especificación. Son importantes para que clientes y usuarios puedan valorar las características no funcionales del producto, pues si se conoce que el mismo cumple con la toda la funcionalidad requerida, las propiedades no funcionales, como cuán usable, seguro, conveniente y agradable, pueden marcar la diferencia entre un producto bien aceptado y uno con poca aceptación. [13]

Una buena definición de requisito no funcional es la dada por Thayer: “(...) es un requisito software que describe no lo que el software hará, sino cómo lo hará, como por ejemplo, requisitos de rendimiento. Los requisitos no funcionales son difíciles de verificar/testear, y por ello son evaluados subjetivamente.” [14]

2.4.1. Requerimientos de Apariencia o Interfaz Externa

- ✔ Se podrán distinguir colores atractivos y acordes con los recomendados por la institución.
- ✔ Debe poseer un ambiente amigable, intuitivo, sencillo y de fácil navegación, tratando así de impedir el rechazo por parte del usuario al tener que interactuar con un sistema no conocido.
- ✔ Diseño perfectamente encuadrado para resoluciones de 1024 x 768, pero preparado para verse en otras resoluciones y para los navegadores más usados.

2.4.2. Requerimientos de Usabilidad

- ✔ La aplicación debe ser flexible, fácil de usar, manteniendo un estándar de operabilidad.

- ✔ El sistema podrá ser usado por usuarios que no necesiten conocimientos avanzados de informática.

2.4.3. Requerimientos de Soporte

- ✔ El sistema estará bien documentado para garantizar futuros mantenimientos.
- ✔ Se le debe dar mantenimiento periódico a los servidores de bases de datos controlando la integridad de la información.

2.4.4. Requerimientos de Portabilidad

- ✔ El producto podrá ser usado bajo cualquier sistema operativo ya sea Unix, Linux o Windows.

2.4.5. Requerimientos de Seguridad

- ✔ Se debe restringir las funcionalidades mediante roles de usuarios y permisos garantizando que la información sea accesible al usuario autorizado.
- ✔ La información deberá ser consultada en dependencia del rol que desempeñe el usuario en la BNCJM.
- ✔ El sistema deberá estar protegido contra accesos no autorizados y las modificaciones de información .

2.4.6. Requerimientos de Software

Del lado del cliente:

- ✔ El cliente podrá tener acceso al sistema a través de cualquiera del los navegadores más conocidos.
- ✔ Cualquier sistema operativo.

Del lado del servidor:

- ✔ Sistema operativo GNU/Linux.
- ✔ Servidor Web Apache 2.x y Perl 5.x.
- ✔ Módulos de Perl necesarios.
- ✔ Servidor de Base de Datos MySQL 5.x.

2.4.7. Requerimientos de Hardware

Del lado del cliente:

- ✔ Procesador Pentium III o superior.
- ✔ 256 MB de memoria RAM o superior.
- ✔ Monitor VGA o superior.
- ✔ Tarjeta de red.

Del lado del servidor:

- ✔ Procesador basado en alguna de las familias INTEL Xeon o rendimiento superior, compatible con la arquitectura X86(IA-32) del tipo Dual Core (2 núcleos).
- ✔ 2.0 GB de memoria RAM o superior.
- ✔ Disco duro de 160 GB o superior.
- ✔ Monitor VGA o superior.
- ✔ Tarjeta de red.

2.4.8. Requerimientos de Ayuda y Documentación en línea

- ✔ Constará con un manual de usuario para hacer más fácil el aprendizaje y explotar al máximo la aplicación.
- ✔ Constará con ayuda digital que será accesible desde cualquier parte de la aplicación.

2.5. Modelo de Despliegue

Para el desarrollo de una aplicación informática es muy importante que el arquitecto de software preste especial atención a la configuración de hardware en la cual se desplegará el sistema, identificando nodos procesadores (computadoras), dispositivos y protocolos; todo lo cual en su conjunto conforma el modelo de despliegue, el cual define la arquitectura física del sistema por medio de nodos interconectados. Estos nodos son elementos de hardware sobre los cuales pueden ejecutarse los elementos software y para representar esta información se emplea el elemento de UML denominado Diagrama de Despliegue.

A continuación se presenta el Diagrama de Despliegue de SIGB Koha de forma general:

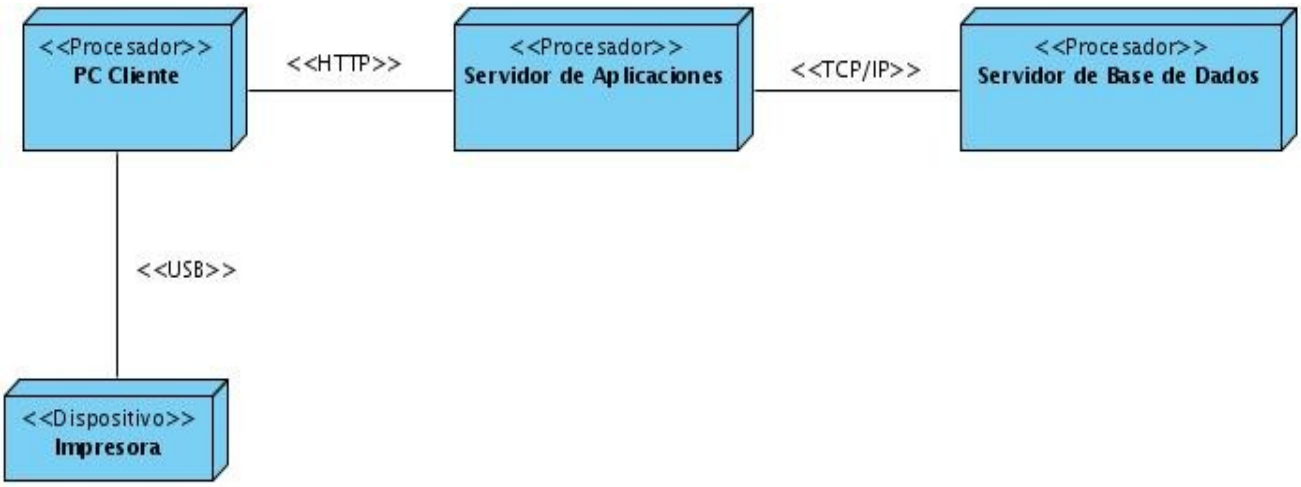


Figura 2: Diagrama de Despliegue de la aplicación

Nodo PC Cliente

Representa las computadoras que utilizaran los usuarios para interactuar con la aplicación. Establece comunicación con el servidor de aplicaciones a través del protocolo HTTP.

Nodo Impresora

Representa las impresoras, pues se requiere del uso de las mismas con el objetivo de imprimir los reportes que se necesiten. No todas las computadoras clientes necesitan de una impresora.

Nodo Servidor de Aplicaciones

En este nodo se encuentran los scripts de la aplicación.

Nodo Servidor de Base Datos

En este nodo se encuentra el Servidor de Base de datos del SIGB Koha.

2.6. Modelo de Implementación

En la implementación se empieza con el resultado del diseño y se implementa el sistema en términos de componentes, es decir, ficheros de código fuente, scripts y ejecutables, para lo cual se realiza el modelo de implementación, el cual incluye los componentes (que representan al código fuente) y la correspondencia de las clases con los componentes.

Para el desarrollo del modelo de implementación se dividió el sistema en subsistemas de implementación, que no son más que una colección de componentes y otros subsistemas de implementación usados para estructurar el modelo de implementación y dividirlos en pequeñas partes que pueden ser integradas y probadas de forma separada.

El subsistema Circulación está dividido en tres subsistemas de implementación fundamentales: el subsistema Vistas, el subsistema Controladoras y el subsistema Modelos, estructurados de modo que

agrupan los scripts según el rol que desempeñan dentro del patrón arquitectónico Modelo-Vista-Controlador.

A continuación se muestra el Diagrama de Componentes de forma global:

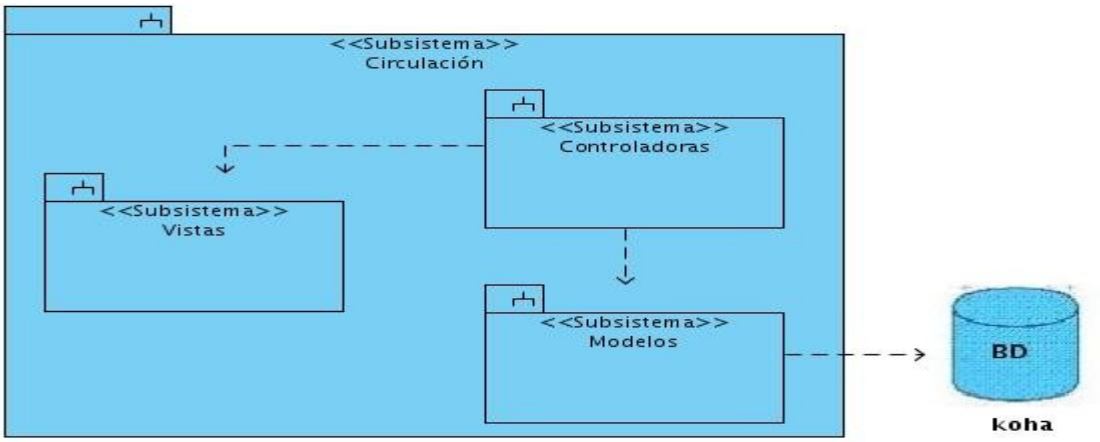


Figura 3: Diagrama de Componentes (Global)

El subsistema Vistas se divide a su vez en dos subsistemas que responden a grupos de componentes con funcionalidades comunes dentro del módulo de circulación. Aquí se agrupan los componentes con que interactúa el usuario, estos son manejados por el subsistema de Controladoras.

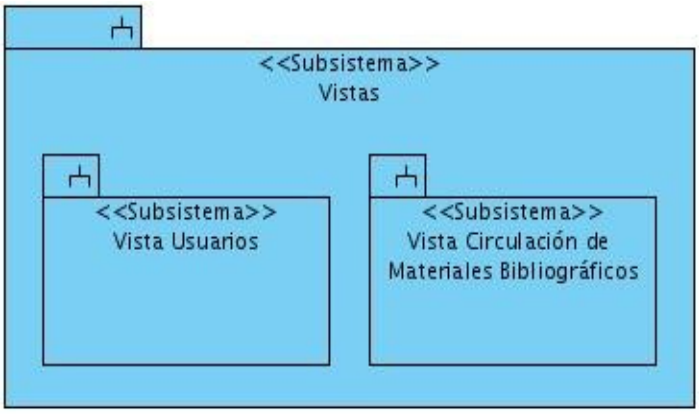


Figura 4: Subsistema Vistas

A continuación se describen detalladamente cada uno de los subsistemas que componen las vistas del módulo.

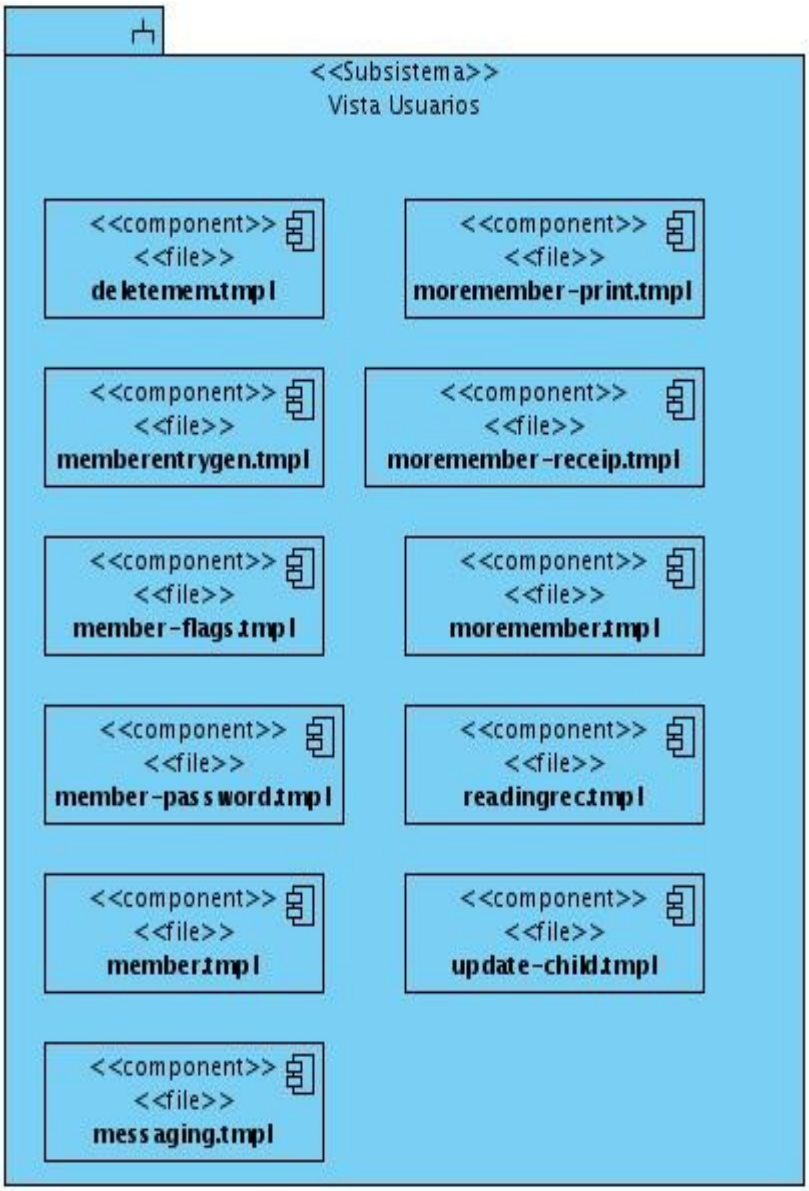


Figura 5: Subsistema Vista Usuarios

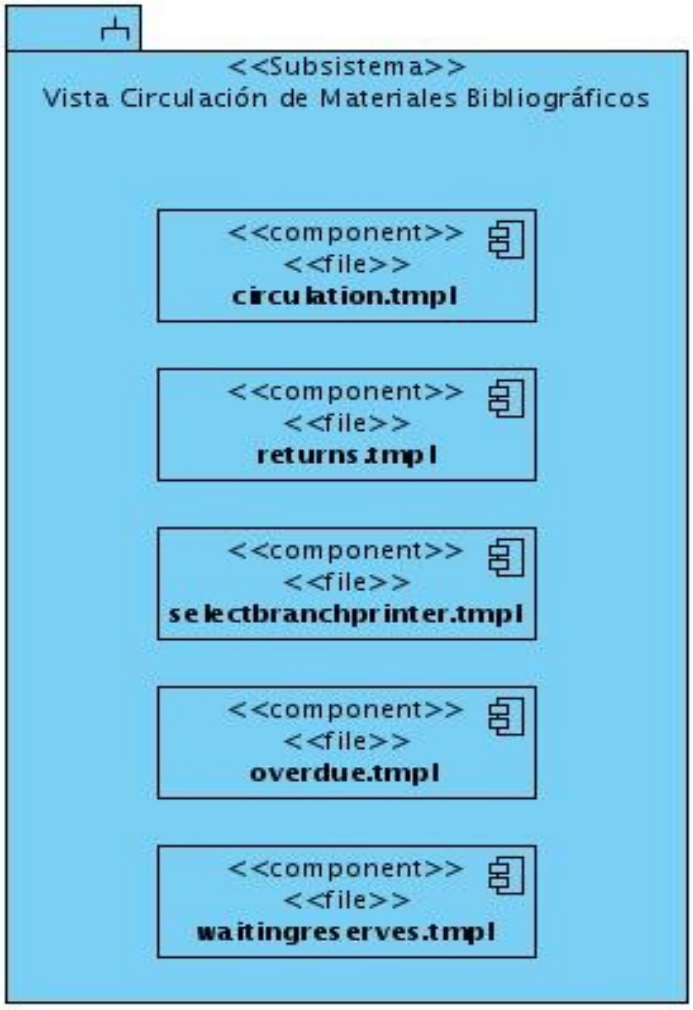


Figura 6: Subsistema Vista Circulación de Materiales Bibliográficos

El subsistema Controladoras, es el rector de las actividades de la aplicación, este contiene 16 ficheros de código fuente, los cuales interactúan con los demás subsistemas coordinando las acciones del software.

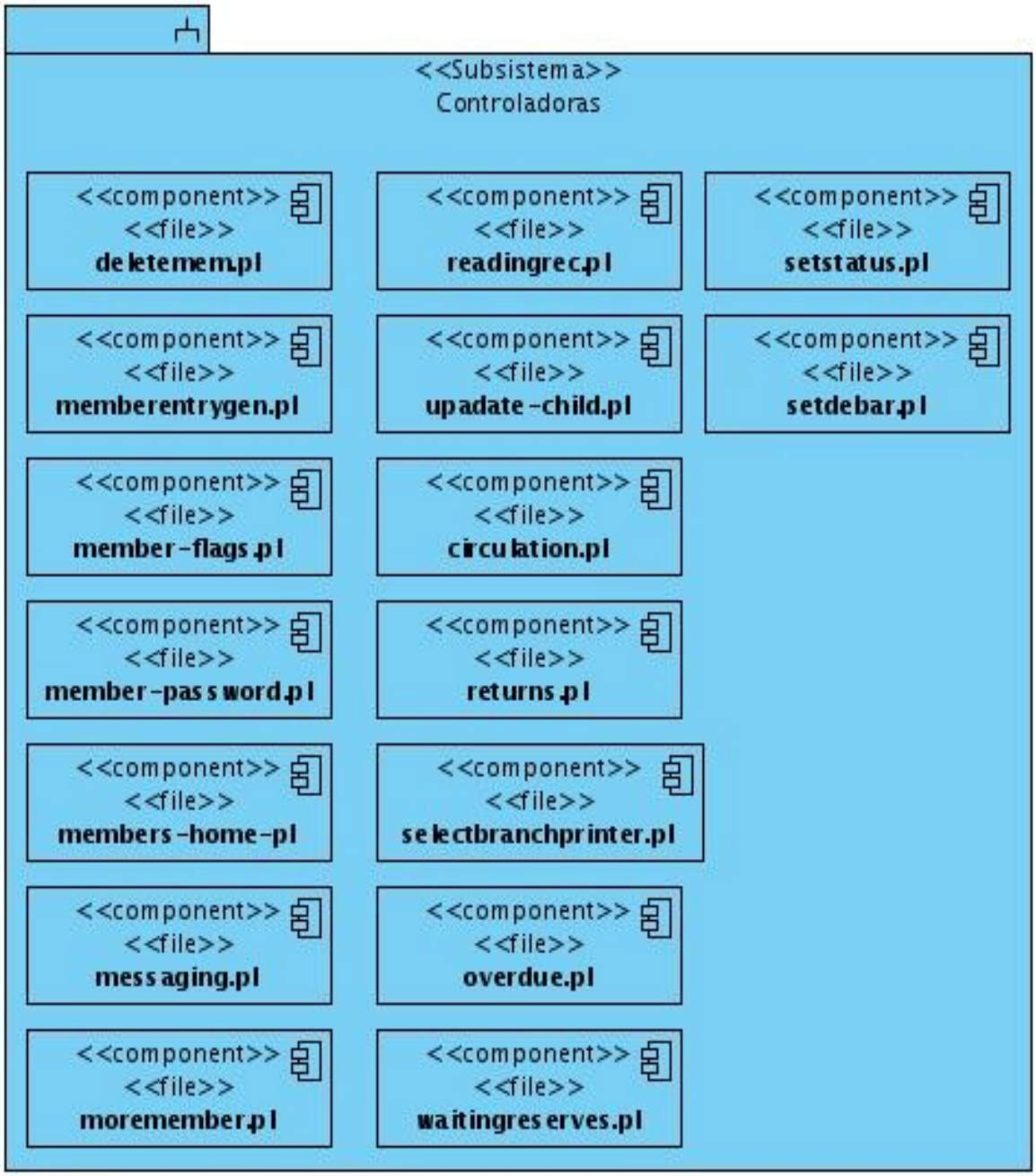


Figura 7: Subsistema Controladoras

El subsistema Modelos es el encargado de interactuar con la base de datos, este está compuesto por 7 componentes.

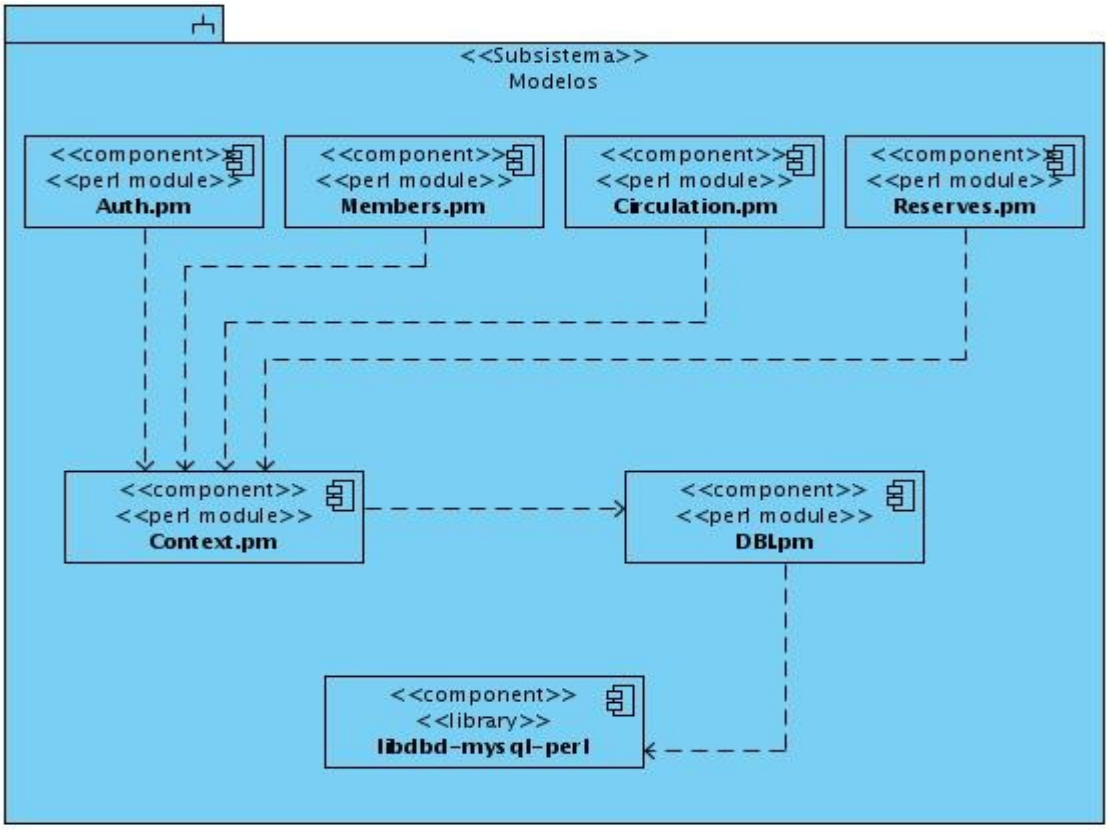


Figura 8: Subsistema Modelos

Es crucial tener bien definido el modo en que los componentes interactúan entre sí. En el caso del módulo de circulación; escrito en el lenguaje de programación Perl, todos los componentes son ficheros y módulos que agrupan funcionalidades ya que este lenguaje no es compilado sino interpretado. Las relaciones que se establecen entre estos scripts son dependencias. A continuación ilustran las relaciones entre los componentes de forma más detallada, se puede apreciar de forma práctica como se implementa el patrón arquitectónico MVC.

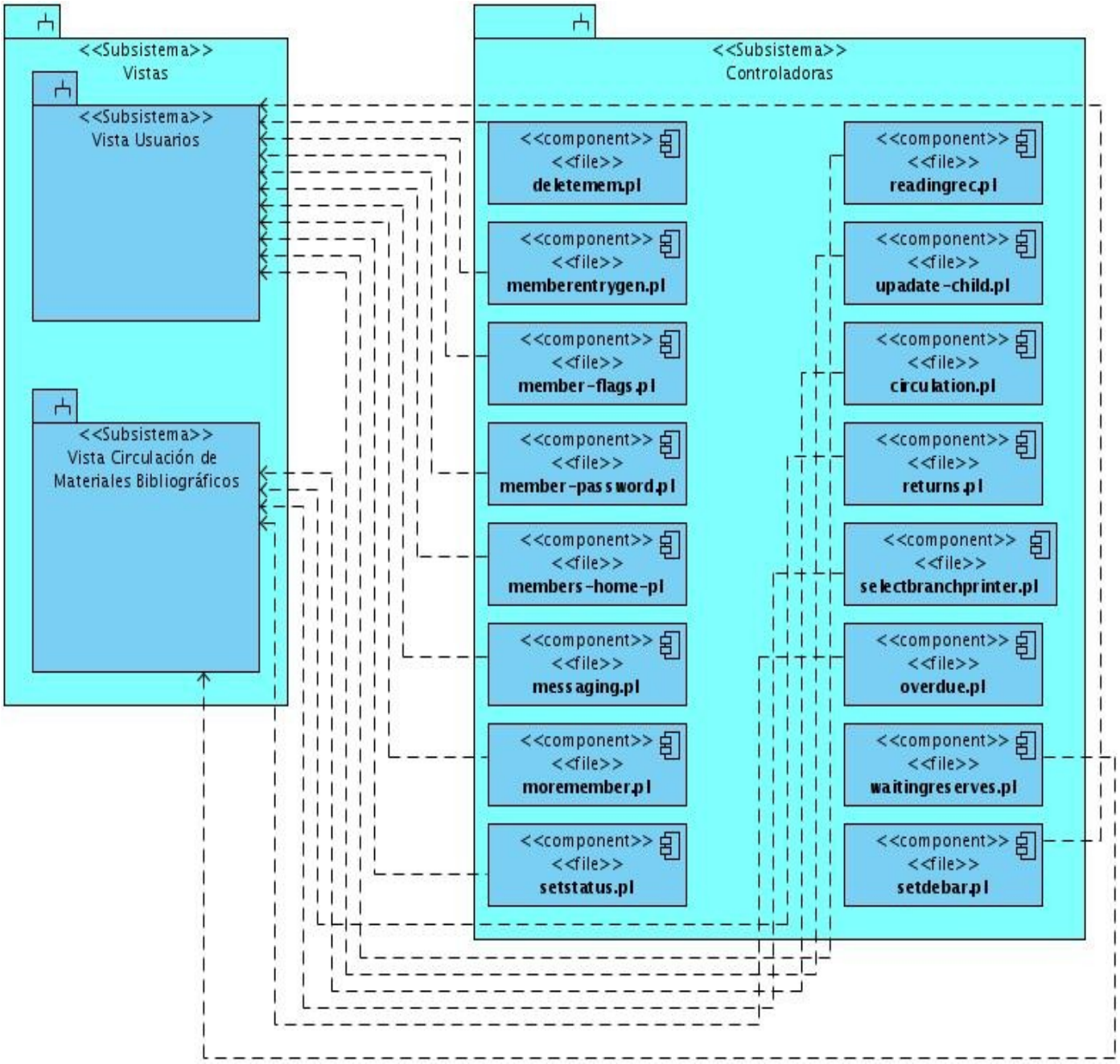


Figura 9: Relación entre el subsistema Vistas y el subsistema Controladoras

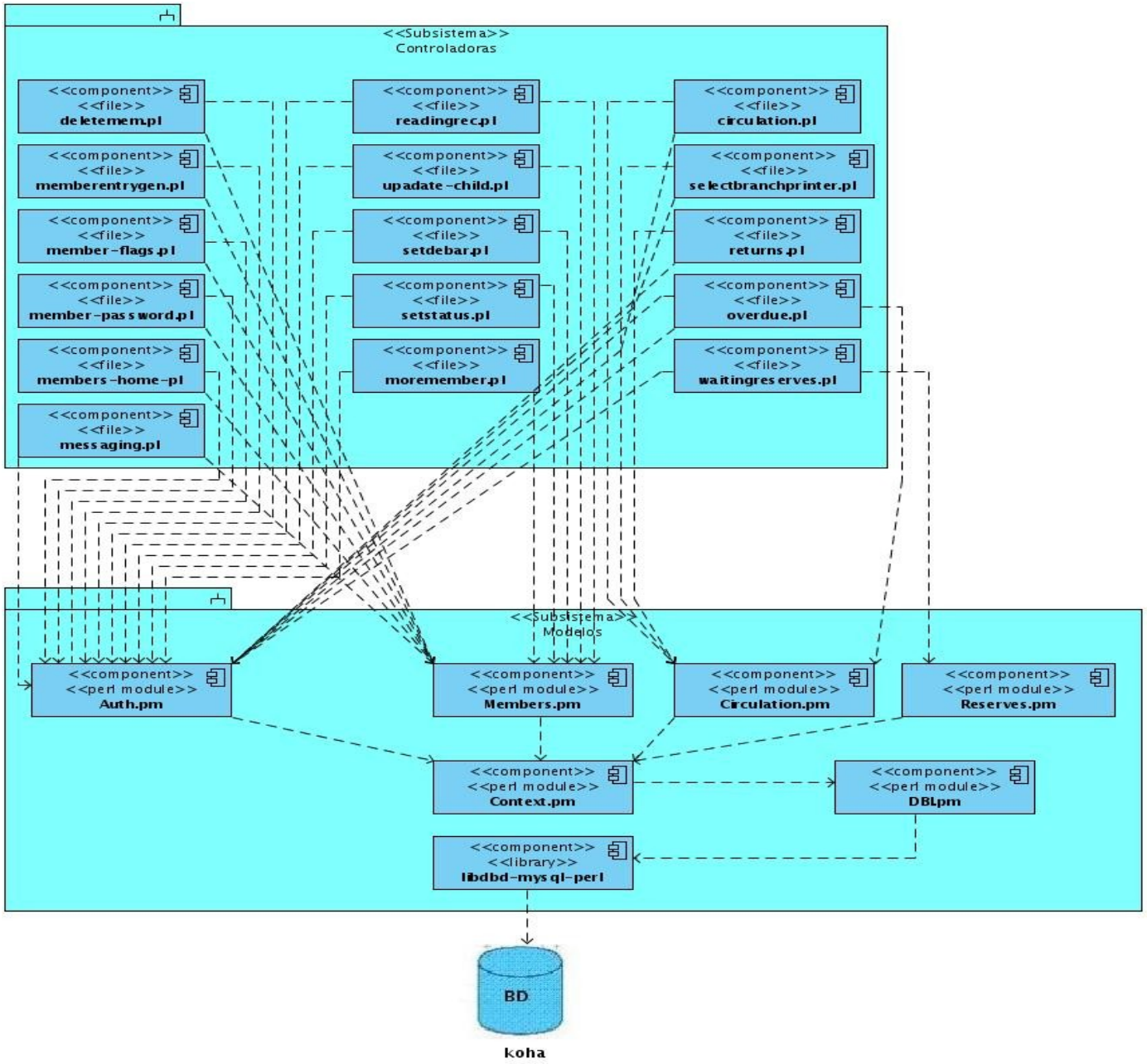


Figura 10: Relación entre el subsistema Controladoras y el subsistema Modelos

2.7. Descripción de los módulos de Perl y las funciones más importantes

A continuación se presentan las descripciones de los módulos de Perl y las subrutinas más importantes implementadas para llevar a cabo el desarrollo de la aplicación.

Nombre del módulo: Members	
Tipo: Modelo	
Descripción: Contiene las funciones necesarias para el manejo de los usuarios como adicionar, buscar, modificar y eliminar.	
Funcionalidades:	
Nombre:	*GetMemberDetails*
Descripción:	my $borrower = GetMemberDetails($borrowernumber, $cardnumber); Busca un usuario y devuelve la información de este. Si $borrowernumber es cierto (distinto de cero) busca al usuario por este número de lo contrario lo busca por el número de carné.
Nombre:	*GetMembersIssuesAndFines*
Descripción:	my ($overdue_count, $issue_count, $total_fines) = GetMemberIssuesAndFines($borrowernumber); Devuelve tres elementos. $overdue_count es el número de ítems que el prestatario tienes atrasados, es decir que no ha entregado en el tiempo establecido. $issue_count es el número de ítems que el prestatario tiene actualmente prestado. $total_fines es el total de multa impuesto.
Nombre:	*AddMember*

Descripción:	my $borrowernumber = AddMember(%borrower); Adiciona un nuevo usuario y devuelve su número, que es el consecutivo en la tabla de usuarios.
Nombre:	*ModMember*
Descripción:	my $success = ModMember(borrowernumber => $borrowernumber, [field => value]...); Modifica los datos de un usuario. Retorna verdadero en caso de ser satisfactoria la operación y falso en caso contrario.
Nombre:	*GetAllIssues*
Descripción:	my $count, $issues = GetAllIssues($borrowernumber, $sortkey, $limit); Devuelve los préstamos que tiene un prestatario.
Nombre:	*checkuniquemember*
Descripción:	my $result,$categorycode = checkuniquemember($collectivity,$surname,$firstname, $dateofbirth); Chequea si un usuario existe o no en la base de datos con los atributos pasados por parámetro.
Nombre:	*GetExpiryDate*
Descripción:	my $expirydate = GetExpiryDate($categorycode, $dateenrolled); Calcula la fecha de expiración de la cuenta de un usuario de una categoría dada a partir de la fecha de ingreso.
Nombre:	*GetBorrowercategoryList*

Descripción:	my $arrayref_hashref = GetBorrowercategoryList; Retorna una lista con todas las categorías de usuarios.
Nombre:	*GetBorrowercategory*
Descripción:	my $hashref = GetBorrowercategory($categorycode); Dada una categoría de usuario retorna los datos correspondientes a la misma.
Nombre:	*DeleteBorrower*
Descripción:	DeleteBorrower($member); Elimina el usuario pasado por parámetro y lo adiciona en la tabla de usuarios eliminados.
Nombre:	*DelMember*
Descripción:	DelMember($borrowernumber); Elimina el usuario con el número pasado por parámetro sin adicionarlo en la tabla de eliminados.
Nombre:	*GetPatronImage*
Descripción:	my ($imagedata, $dberror) = GetPatronImage($cardnumber); Devuelve la foto del usuario con el número de carné pasado por parámetro.
Nombre:	*PutPatronImage*
Descripción:	PutPatronImage($cardnumber, $mimetype, $imgfile); Almacena la foto binaria de un usuario en la base de datos. Está función también actualiza la imagen en caso de que exista otra.
Nombre:	*RmPatronImage*

Descripción:	my $dberror = RmPatronImage($cardnumber); Elimina la foto del usuario con el número de carné pasado por parámetro, de la base de datos.
Nombre:	*GetBorrowersWhoHaveNotBorrowedSince*
Descripción:	my $results = GetBorrowersWhoHaveNotBorrowedSince($date); Retorna los prestatarios a los que no se le ha prestado desde la fecha pasada por parámetro.
Nombre:	*GetBorrowersWhoHaveNeverBorrowed*
Descripción:	my $results = GetBorrowersWhoHaveNeverBorrowed; Retorna los prestatarios que nunca han pedido prestado.
Nombre:	*DebarMember*
Descripción:	my $success = DebarMember($borrowernumber); Marca como excluido al usuario con número pasado por parámetro, por lo tanto, no puede pedir prestado ningún ítem. Retorna verdadero si la operación se realiza con éxito de lo contrario retorna falso.
Nombre:	*GetAllCountryName*
Descripción:	my @result = GetAllCountryName*()*; Retorna el nombre de todos lo países que existen en la base de datos.
Nombre:	*SetDebarred*
Descripción:	SetDebarred ($status, $borrowernumber);

	Cambia el estado de un usuario de excluido a o excluido y viceversa.

Tabla 1: Descripción de las funcionalidades del módulo de Perl Members

Nombre del módulo: Circulation	
Tipo: Modelo	
Descripción: Contiene las funciones necesarias para llevar a cabo la circulación de materiales bibliográficos.	
Funcionalidades:	
Nombre:	*CanBookBeIssued*
Descripción:	my ($issuingimpossible,$needsconfirmation) = CanBookBeIssued($borrower,$barcode, $year,$month,$day); Chequea si un ítem puede ser prestado.
Nombre:	*itemissues*
Descripción:	my @issues = itemissues($biblioitemnumber, $biblio); Busca información sobre los prestatarios a quien se le ha prestado un determinado ítem.
Nombre:	*AddIssue*
Descripción:	AddIssue($borrower,$barcode,$date); Esta función es la encargada de realizar el préstamo de un ítem a un prestatario determinado.
Nombre:	*GetLoanLength*
Descripción:	my $loanlength = GetLoanLength($borrowertype,$itemtype,$branchcode);

	Función para obtener el tiempo de préstamo para un tipo de usuario, un tipo de ítem y una sucursal determinada.
Nombre:	*AddReturn*
Descripción:	my ($doreturn, $messages, $iteminformation, $borrower) = AddReturn($barcode, $branch, $exemptfine, $dropbox); Función para realizar la devolución de ítems.
Nombre:	*MarkIssueReturned*
Descripción:	MarkIssueReturned($borrowernumber, $itemnumber, $dropbox_branch); Marca un ítem como como retornado en la tabla old_ussues.
Nombre:	*GetItemIssue*
Descripción:	my $issues = GetItemIssue($itemnumber); Retorna los prestatarios que actualmente tienen prestado un determinado ítem.
Nombre:	*GetItemIssues*
Descripción:	my $issues = GetItemIssues($itemnumber, $history); Retorna todos los prestatarios a los que se le ha prestado un determinado ítem.
Nombre:	*CanBookBeRenewed*
Descripción:	my ($ok,$error) = CanBookBeRenewed($borrowernumber, $itemnumber); Retorna si un determinado ítem puede ser renovado por un prestatario.
Nombre:	*AddRenewal*
Descripción:	AddRenewal($borrowernumber, $itemnumber, $branch, $datedue);

	Renova un préstamo.

Tabla 2: Descripción de las funcionalidades del módulo de Perl Circulation

Nombre del módulo: Reserves	
Tipo: Modelo	
Descripción: Este módulo contiene las funciones necesarias para realizar las reservaciones de ítems.	
Funcionalidades:	
Nombre:	*AddReserve*
Descripción:	AddReserve($branch,$borrowernumber,$biblionumber,$constraint,$bibitems,$priority, $notes,$title,$checkitem,$found); Esta función realiza una reservación.
Nombre:	*GetReservesFromBiblionumber*
Descripción:	my @borrowerreserv=&GetReserves($biblionumber,$itemnumber,$borrowernumber); Esta función retorna una lista de reservaciones para un registro bibliográfico, un ítem o un prestatario. Solo un argumento es pasado por parámetro.
Nombre:	*GetReservesFromItemnumber*
Descripción:	my ($reservedate, $borrowernumber, $branchcode) = GetReservesFromItemnumber($itemnumber); Retorna las reservaciones para un ítem determinado.
Nombre:	*GetReserveCount*
Descripción:	my $number = GetReserveCount($borrowernumber);

	Retorna la cantidad de reservaciones que ha hecho un determinado usuario.
Nombre:	*GetReservesToBranch*
Descripción:	my @transreserv = GetReservesToBranch($frombranch); Retorna una lista de reservaciones que se han hecho en una sucursal determinada.
Nombre:	*CheckReserves*
Descripción:	my $status, $reserve = CheckReserves($itemnumber); Busca si un ítem está reservado.
Nombre:	*CancelReserve*
Descripción:	CancelReserve($biblionumber, $itemnumber, $borrowernumber); Cancela una reservación.
Nombre:	*ModReserve*
Descripción:	ModReserve($rank,$biblio,$borrower,$branch); Modifica una reservación.

Tabla 3: Descripción de las funcionalidades del módulo de Perl Reserves

Nombre del módulo: Auth	
Tipo: Modelo	
Descripción: Este módulo contiene las funciones para autenticar los usuarios y proporcionar la seguridad al sistema.	
Funcionalidades:	
Nombre:	*get_template_and_user*

Descripción:	my ($template, $borrowernumber, $cookie) = get_template_and_user({ template_name => "opac-main.tmpl", query => $query, type => "opac", authnotrequired => 1, flagsrequired => {borrow => 1, catalogue => '*', tools => 'import_patrons' }, }); Función para proporcionar la autenticación.
Nombre:	*checkauth*
Descripción:	my ($userid, $cookie, $sessionID) = checkauth($query, $noauth, $flagsrequired, $type); Verifica que un usuario tiene permisos para ejecutar determinado script.
Nombre:	*check_cookie_auth*
Descripción:	my ($status, $sessionId) = check_api_auth($cookie, $userflags); Verifica sin un identificador de sesión tiene los permisos pasados en la variable $userflags.
Nombre:	*get_session*
Descripción:	use CGI::Session; my $session = get_session($sessionID); Dado un identificador de sesión retorna el objeto CGI::Session para almacenar el estado de la misma.
Nombre:	*getuserflags*
Descripción:	my $authflags = getuserflags($flags,$dbh); Traduce un número entero ($flags) en permisos.

Tabla 4: Descripción de las funcionalidades del módulo de Perl Auth

Nombre del módulo: Context

Tipo: Modelo	
Descripción: Este módulo contiene las funciones para cargar todas las configuraciones de Koha, así como las funciones para el acceso a la base de datos.	
Funcionalidades:	
Nombre:	*dbh*
Descripción:	my $dbh = C4::Context->dbh; Retorna el manejador de la base de datos, si no existe lo crea.
Nombre:	*new_dbh*
Descripción:	my $dbh = C4::Context->new_dbh; Crea un nuevo manejador de base de datos.
Nombre:	*set_dbh*
Descripción:	$my_dbh = C4::Connect->new_dbh; C4::Connect->set_dbh($my_dbh); ... C4::Connect->restore_dbh; Cambia el manejador de base de datos.
Nombre:	*restore_dbh*
Descripción:	C4::Context->restore_dbh; Restaura el manejador de la base de datos.
Nombre:	*read_config_file*
Descripción:	Función para leer el fichero de configuración de Koha e inicializar variables de entorno.

Tabla 5: Descripción de las funcionalidades del módulo de Perl Context

2.8. Modelo de Datos

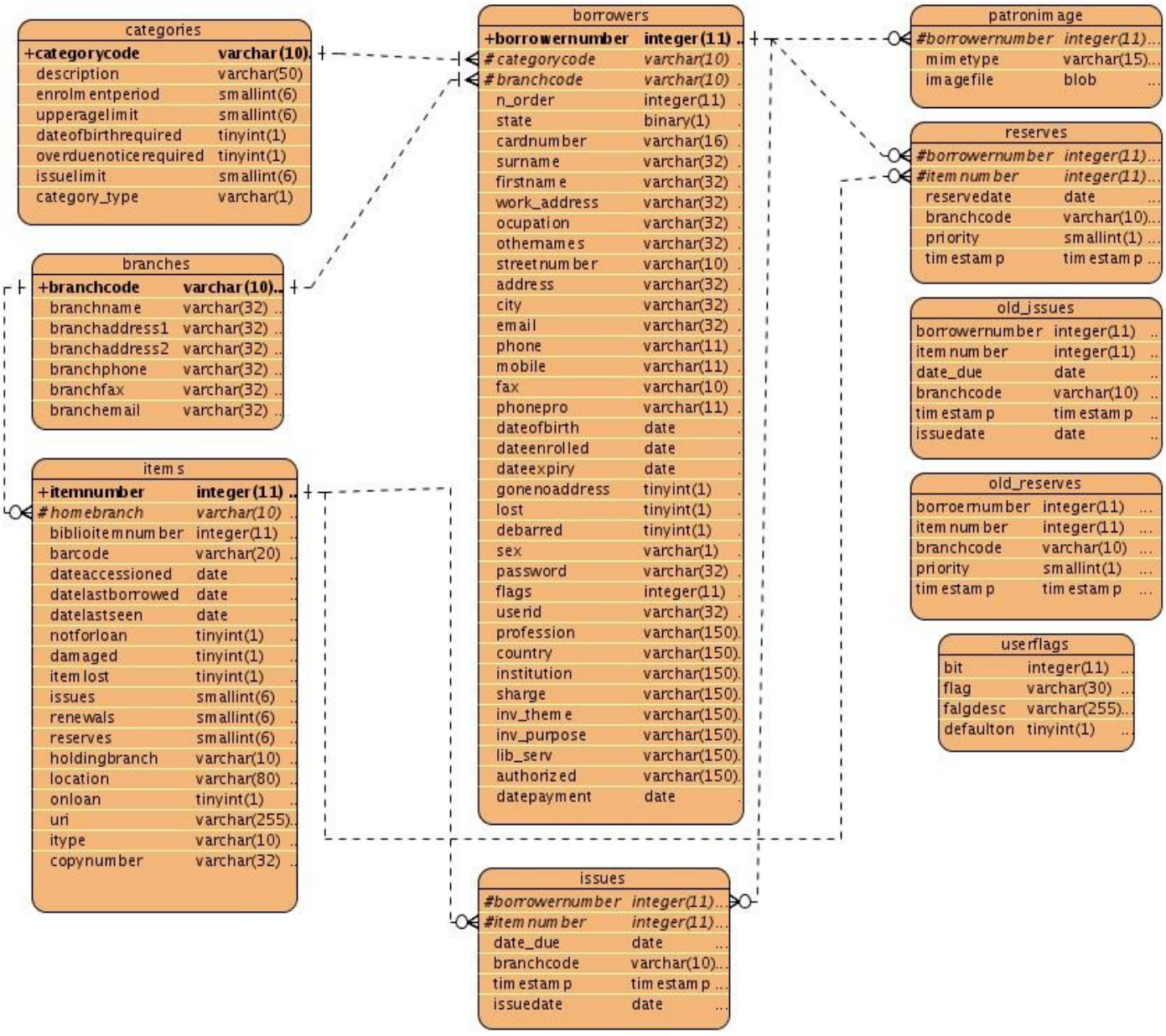


Figura 11: Modelo de Datos

2.9. Descripción de las tablas de la base de datos

Nombre: borrowers		
Descripción: Contiene la información de los usuarios.		
Atributo	**Tipo**	**Descripción**
borrowernumber	integer	Número de usuario, identificador de la tabla
categorycode	varchar	Código de la categoría a la que pertenece el usuario
branchcode	varchar	Identificador de la sucursal a la que pertenece el usuario
n_order	integer	Número de orden para la categoría de usuario
state	binary	Estado del usuario (Activo = 1, Pasivo = 0)
cardnumber	varchar	Número de carné del usuario (Único)
surname	varchar	Apellidos
fistname	varchar	Primer nombre
work_address	varchar	Dirección del centro de trabajo o estudio
ocupation	varchar	Ocupación
othernames	varchar	Segundo nombre del usuario
address	varchar	Domicilio del usuario
city	varchar	Ciudad de nacimiento del usuario
email	varchar	Dirección electrónica del usuario
phone	varchar	Teléfono particular del usuario
mobile	varchar	Teléfono celular del usuario
fax	varchar	Fax del usuario o del centro de trabajo

phonepro	varchar	Teléfono del centro de trabajo o estudio
dateenrollet	date	Fecha de inscripción del usuario
dateepiry	date	Fecha de vencimiento de la inscripción
gonenoaddress	tinyin	Especifica si el usuario tiene dirección dudosa
lost	tinyin	Especifica si el usuario tiene el carné perdido
debarred	tinyin	Especifica si el usuario está excluido
sex	varchar	Sexo del usuario
password	varchar	Contraseña del usuario
flags	integer	Número que significa los privilegios del usuario
userid	varchar	Identificador del usuario (Único)
profession	varchar	Profesión del usuario
country	varchar	País donde vive el usuario
institution	varchar	Institución donde trabaja o estudia el usuario
charge	varchar	Cargo del usuario
inv_theme	varchar	Tema sobre el que va a investigar
inv_purpose	varchar	Propósito de la investigación
lib_serv	varchar	Servicios de la biblioteca que va a visitar el usuario
authorized	varchar	Nombre del bibliotecario que autoriza la inscripción
datepayment	varchar	Fecha de pago de la inscripción

Tabla 6: Descripción de la tabla borrowers.

Nombre: items		
Descripción: Contiene la información de los materiales bibliográficos.		
Atributo	**Tipo**	**Descripción**
itemnumber	integer	Identificador del ítem
biblioitemnumber	varchar	Número de registro bibliográfico al que pertenece
barcode	varchar	Código de barras del ítem (Número de Inventario)
dateaccessioned	date	Fecha de entrada en la biblioteca
homebranch	varchar	Sucursal a la que pertenece
datelastborrowed	date	Fecha de la última ves que se prestó
datelastseen	date	Fecha de la última vez que se consultó
notforloan	tinyint	Especifica si ítem no se puede prestar
damage	tinyint	Especifica si el ítem está dañado
itemlost	tinyint	Especifica si el ítem está perdido
itemcallnumber	varchar	Clasificación de ítem
issues	integer	Cantidad de préstamos
renewals	integer	Cantidad de renovaciones
reserves	integer	Cantidad de reservaciones
holdingbranch	varchar	Sucursal donde se encuentra actualmente
location	varchar	Localización del ítem dentro de la biblioteca
onloan	date	Especifica la fecha que se prestó, si está prestado
url	varchar	URL del ítem si existe en formato digital

itype	varchar	Tipo de ítem
copynumber	integer	Número de copia

Tabla 7: Descripción de la tabla items.

Nombre: categories		
Descripción: Contiene la información de las categorías de usuarios.		
Atributo	**Tipo**	**Descripción**
categorycode	varchar	Código de la categoría
description	varchar	Descripción de la categoría
enrolmenttperiod	integer	Período de asociación (Meses)
dateofbirthrequiere	integer	Edad requerida
upperagelimit	integer	Edad máxima requerida para la categoría
overduenoticerequiered	integer	Especifica si la categoría requiere avisos de penalizaciones
issuelimit	integer	Cantidad máxima de préstamos
category_type	varchar	Tipo de categoría

Tabla 8: Descripción de la tabla categories.

Nombre: patronimage		
Descripción: Contiene las fotos de los usuarios		
Atributo	**Tipo**	**Descripción**
borrowernumber	integer	Número de usuario

mimetype	varchar	Tipo de archivo
image	blob	Foto del usuario

Tabla 9: Descripción de la tabla patronimage

Nombre: branches		
Descripción: Contiene la información de las sucursales		
Atributo	**Tipo**	**Descripción**
branchcode	varchar	Código de la sucursal
branchname	varchar	Nombre de la sucursal
branchaddress1	varchar	Dirección
branchaddress2	varchar	Dirección alternativa
branchphone	varchar	Teléfono de la sucursal
branchfax	varchar	Fax de la sucursal
branchemail	varchar	Dirección electrónica de la sucursal

Tabla 10: Descripción de la tabla branches.

Nombre: issues		
Descripción: Contiene la información de los préstamos		
Atributo	**Tipo**	**Descripción**
borrowernumber	integer	Número del usuario
itemnumber	integer	Número del ítem
date_due	date	Fecha de devolución
branchcode	varchar	Código de la sucursal que prestó el ítem

timestamp	timestamp	Fecha y hora del préstamo
issuedate	date	Fecha en que se realizó el préstamo

Tabla 11: Descripción de la tabla issues

Nombre: reserves		
Descripción: Contiene la información de las reservas de ítem.		
Atributo	**Tipo**	**Descripción**
borrowernumber	integer	Número del usuario
reservedate	date	Fecha de la reservación
itemnumber	integer	Número del ítem
branchcode	varchar	Sucursal donde se hace la reservación
priority	integer	Prioridad de la reservación
timestamp	timestamp	Fecha y hora de la reservación

Tabla 12: Descripción de la tabla reserves

Nombre: userflags		
Descripción: Contiene la información de los permisos de usuario.		
Atributo	**Tipo**	**Descripción**
bit	integer	Número del permiso
flag	varchar	Indica el permiso de usuario
flagdesc	varchar	Descripción del permiso de usuario
defaulton	varchar	Especifica si el usuario tiene el permiso por defecto

Tabla 13: Descripción de la tabla userflags

2.10. Conclusiones

En el presente capítulo se han ofrecido los elementos necesarios para la implementación de lo cambios al módulo de circulación de Koha. Se describieron los principales módulos de Perl necesarios para el correcto funcionamiento del subsistema, y sus funcionalidades, las cuales responden a las necesidades del cliente identificadas por los analistas durante en flujo de trabajo levantamiento de requisitos. Además, se realizó la descripción de la base de datos que permite comprender los mecanismos fundamentales de almacenamiento de la información.

3. CAPÍTULO 3. VALIDACIÓN DE LA SOLUCIÓN PROPUESTA

3.1. Introducción

Uno de los mayores problemas que se afrontan actualmente en la esfera de la informática es la calidad del software, por lo que el proceso de pruebas es sin dudas uno de los aspectos fundamentales para medir el estado de calidad de un sistema informático.

En el desarrollo del software, las posibilidades de errores son innumerables. Pueden darse por una mala especificación de los requisitos funcionales, uso indebido de las estructuras de datos, errores al enlazar módulos, etcétera.

La prueba y validación de los resultados no es un proceso que se realiza una vez desarrollado el software, sino que debe efectuarse en cada una de las etapas de desarrollo. Es fundamental medir la cobertura de las pruebas, es decir, la determinación de cuando se han realizado las pruebas suficientes. Si se siguen encontrando errores cada vez que se procesa el programa, las pruebas deben continuar.

Durante el mantenimiento debe existir una documentación de pruebas que incluya casos de prueba y resultados esperados. Si se producen modificaciones en el programa, habrá que probar de nuevo todas las partes del programa afectadas por las modificaciones.

Por lo dicho anteriormente, se desarrolló este capítulo, con el objetivo de especificar las pruebas que serán hechas al software y describir todo lo encontrado en las mismas.

3.2. Pruebas aplicadas

Las pruebas son la actividad en la cual un sistema o componente es ejecutado bajo condiciones o requerimientos específicos, donde los resultados son observados y registrados, y es hecha una evaluación de algún aspecto del sistema o componente.

Con la realización de estas pruebas se pretende encontrar y documentar los defectos que puedan afectar la calidad del software, validar y probar los requisitos que debe cumplir el software y a su vez que estos fueron implementados correctamente.

Es necesario analizar que las pruebas no pueden asegurar la ausencia de defectos sino que permiten demostrar que existen defectos en el software, que cada prototipo que se quiera entregar al final de una iteración debe ser probado y evaluado.

Los casos de prueba especifican la forma de probar el sistema en cuestión, incluyendo la entrada o resultado con el que se ha de probar y las condiciones bajo las que ha de probarse. Es un conjunto de entradas y resultados esperados que ponen en práctica el funcionamiento de un componente con el objetivo de causar fallas y detectar defectos. Entre los casos de prueba pueden existir todos los tipos de relaciones, pero las más importantes son las de precedencia.

Todo producto puede ser probado de las siguientes formas:

1. Conociendo el funcionamiento del producto, para poder desarrollar pruebas que aseguren que la operación interna se ajusta a las especificaciones y que todos los componentes internos se han comprobado de forma adecuada.
2. Conociendo la funcionalidad específica para la cual fue diseñado el producto, para poder llevar a cabo pruebas que demuestren que cada función es completamente operativa.

En el primer caso, las pruebas están dirigidas a la aplicación de métodos basados en caja blanca y en el segundo caso, a los métodos basados en caja negra. Los de este primer grupo permiten la comprobación de los caminos lógicos del software proponiendo casos de prueba que ejerciten conjuntos específicos de condiciones.

3.3. Pruebas de caja blanca

Las pruebas de caja blanca normalmente se denominan pruebas de cobertura o pruebas de caja transparente. Al total de pruebas de caja blanca se le llama cobertura. La cobertura es un número

porcentual que indica cuanto código del programa se ha probado.

Básicamente la idea de pruebas de cobertura consiste en diseñar un plan de pruebas en las que se vaya ejecutando sistemáticamente el código hasta que haya corrido todo o la gran mayoría de él, esto que parece complicado, pero lo es más, cuando el programa contiene código de difícil alcance, como por ejemplo manejadores de errores o "código muerto".

Es muy recomendable alcanzar una elevada cobertura de sentencias, aunque no siempre es posible por premura de tiempo o medios. La prueba de caja blanca se basa en el diseño de casos de prueba que usa la estructura de control del diseño procedimental para derivarlos. Mediante la prueba de la caja blanca se puede obtener casos de prueba que:

- ✔ Garanticen que se ejerciten por lo menos una vez todos los caminos independientes de cada módulo, programa o método.
- ✔ Ejerciten todas las decisiones lógicas en las vertientes verdadera y falsa.
- ✔ Ejecuten todos los bucles en sus límites operacionales.
- ✔ Ejerciten las estructuras internas de datos para asegurar su validez.

3.3.1. Aplicación de prueba de caja blanca

Para la realización de las pruebas de caja blanca a un determinado algoritmo es necesario efectuar anteriormente el análisis de la complejidad ciclomática del mismo. Este valor define la cantidad máxima de caminos independientes del programa y nos da el límite superior para el número de casos de prueba que se deben construir para asegurar que se ejecuta casa sentencia al menos una vez y cada condición simple se habrá ejercitado al menos una vez en sus valores verdadero y falso.

Un camino independiente es cualquier camino del programa que introduce, por lo menos, un nuevo conjunto de sentencias simples o una nueva condición. En términos del grafo de flujo de control, un camino independiente es cualquier camino que atraviesa todo el programa y que esté constituido por lo menos por una arista que no haya sido recorrido anteriormente a que ese camino haya sido definido

La porción de código que se toma para realizar la prueba es la función encargada de mostrar todos los préstamos que se le han hecho a un determinado usuario: GetAllIssues.

Porción de código:

```
sub GetAllIssues {
      my ( $borrowernumber, $order, $limit ) = @_;                                    (1)
      my $dbh   = C4::Context->dbh;                                                   (1)
      my $count = 0;                                                                  (1)
      my $query = "SELECT *,items.timestamp AS itemstimestamp FROM issues
          LEFT JOIN items on items.itemnumber=issues.itemnumber
          LEFT JOIN biblio ON items.biblionumber=biblio.biblionumber
          LEFT JOIN biblioitems ON
          items.biblioitemnumber=biblioitems.biblioitemnumber
          WHERE borrowernumber=?
          UNION ALL
          SELECT *,items.timestamp AS itemstimestamp                                  (1)
          FROM old_issues
          LEFT JOIN items on items.itemnumber=old_issues.itemnumber
          LEFT JOIN biblio ON items.biblionumber=biblio.biblionumber
          LEFT JOIN biblioitems ON
          items.biblioitemnumber=biblioitems.biblioitemnumber
          WHERE borrowernumber=?
          order by $order";
      if ( $limit != 0 ) {                                                            (2)
         $query .= " limit $limit";                                                   (3)
      }
      my $sth = $dbh->prepare($query);                                                (4)
      $sth->execute($borrowernumber, $borrowernumber);                                (4)
      my @result;                                                                     (4)
      my $i = 0;                                                                      (4)
      while ( my $data = $sth->fetchrow_hashref ) {                                   (5)
         $result[$i] = $data;                                                         (6)
         $i++;                                                                        (6)
         $count++;                                                                    (6)
      }
      if ( C4::Context->preference("ReadingHistory") ) {                              (7)
          my $query2 = "SELECT * FROM oldissues
             LEFT JOIN items ON items.itemnumber=oldissues.itemnumber
              LEFT JOIN biblio ON items.biblionumber=biblio.biblionumber
              LEFT JOIN biblioitems ON                                                (8)
              items.biblioitemnumber=biblioitems.biblioitemnumber
              WHERE borrowernumber=?
              ORDER BY $order";
          if ( $limit != 0 ) {                                                        (9)
                $limit = $limit - $count;                                             (10)
```

```
                $query2 .= " limit $limit";                                    (10)
            }
            my $sth2 = $dbh->prepare($query2);                                 (11)
            $sth2->execute($borrowernumber);                                   (11)
            while ( my $data2 = $sth2->fetchrow_hashref ) {                    (12)
            $result[$i] = $data2;                                              (13)
            $i++;                                                              (13)
            }
            $sth2->finish;                                                     (14)
        }
        $sth->finish;                                                          (15)
        return ( $i, \@result );                                               (15)
}                                                                              (16)
```

Construcción del grafo de flujo asociado al código representado anteriormente:

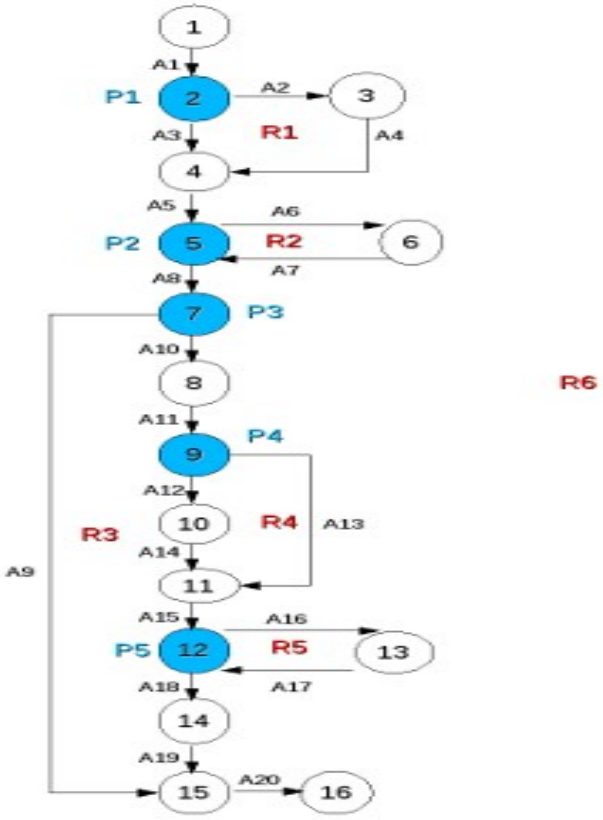


Figura 12: Grafo de flujo de la función GetAllIssues

Aplicación de las fórmulas para calcular la complejidad ciclomática. Para que el resultado sea correcto debe ser el mismo para las tres fórmulas.

1. V (G) = (A – N) + 2.
2. V (G) = P + 1.
3. V (G) = R.

Sustituyendo los valores se obtiene que:

1. V (G) = 20 aristas - 16 nodos + 2 = 6.
2. V (G) = 5 nodos predicados + 1 = 6.
3. V (G) = 6 regiones.

Por tanto la complejidad ciclomática del grafo de flujo de la **Figura 12** es igual a 6. Esto quiere decir que hay que determinar 6 caminos independientes.

Como caminos independientes se determinaron:

Camino 1: 1-2-4-5-7-8-9-10-11-12-14-15-16

Camino 2: 1-2-3-4-5-7-8-9-10-11-12-14-15-16

Camino 3: 12-3-4-5-6-5-7-8-9-10-11-12-14-15-16

Camino 4: 1-2-3-4-5-6-5-7-15-16

Camino 5: 1-2-3-5-6-5-7-8-9-11-12-14-15-16

Camino 6: 1-2-3-4-5-6-5-7-8-9-11-12-13-12-14-15-16

Este algoritmo está conformado por sentencias repetitivas (bucles) y por condicionales que se ejecutan una tras la otra, de esta manera solo es necesario probar que estos bucles y condicionales se ejecutan completamente, garantizando una cobertura total que se logra al menos una prueba para cada camino independiente.

Para el camino 6 que recorre la mayor cantidad de nodos del grafo de flujo se conformó el siguiente caso de prueba:

Caso de prueba para el Camino 6:

Descripción y asignación:

Se quiere buscar todos los préstamos de un paciente determinado, esta información se debe encontrar en la base de datos. La función debe retornar una referencia a una lista la cual en cada posición tiene la información solicitada.

Condición de ejecución:

Se especifica cada parámetro para que cumpla la condición deseada y ver el funcionamiento de la función. Todos los datos deben ser entrados.

Entrada:

borrowernumber = 1, order = 'author', limit =50.

Resultado esperado:

Al entrar todos los datos correctamente al sistema, se muestran todas los préstamos realizado al usuario, especificando su número, el orden y el límite de resultados.

Evaluación de los resultados obtenidos.

Para el caso de prueba descrito, el algoritmo recorrió todos los bucles y las condicionales, confirmando su

buen funcionamiento. Luego de aplicar el caso de prueba, se pudo comprobar que el flujo de trabajo del procedimiento está correcto ya que cumple con las condiciones necesarias que se habían planteado para esta función. Para lograr la cobertura total es necesario probar los restantes caminos independientes al menos una vez.

3.4. Pruebas de caja negra

La prueba de caja negra se centra principalmente en los requisitos funcionales del software. Estas pruebas permiten obtener un conjunto de condiciones de entrada que ejerciten completamente todos los requisitos funcionales de un programa. En ellas se ignora la estructura de control, concentrándose en los requisitos funcionales del sistema y ejercitándolos.

La prueba de caja negra no es una alternativa a 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.

Dentro del método de caja negra la técnica de la Partición de Equivalencia es una de las más efectivas pues permite examinar los valores válidos e inválidos de las entradas existentes en el software, descubre de forma inmediata una clase de errores que, de otro modo, requerirían la ejecución de muchos casos antes de detectar el error genérico.

3.4.1. Aplicación de prueba de caja negra

Para la aplicación de este tipo de prueba se utilizará el caso de uso “*Gestionar Usuarios*”

Objetivo del test:

Validar el caso de uso “*Gestionar Usuarios*”.

Las pruebas realizadas a este caso de uso son:

- ✔ Adicionar usuario

- ✔ Buscar usuario
- ✔ Modificar usuario
- ✔ Eliminar usuario

Flujo central:

Una vez que el usuario llega a la BNCJM, pide asociarse a la misma de acuerdo a la categoría de usuario que le corresponda según su nivel. El bibliotecario que está en el local de información realiza la inscripción y entrega el carné de asociado. El bibliotecario también puede realizar la búsqueda de un usuario ya inscrito, modificación de los datos del perfil y la eliminación.

Pre condiciones:

Deben estar presentes en la base de datos:

- ✔ Categorías de usuarios
- ✔ Usuarios

Sección: Adicionar usuarios					
Acción	**Clases válidas**	**Clases no válidas**	**Res. de la prueba**	**Observaciones**	**Cumplimiento**
El bibliotecario selecciona la opción “Usuarios” del menú principal	Se muestra la interfaz correctamente	No se muestra la interfaz para gestionar usuarios	Satisfactorio	Haciendo uso de la interfaz para gestionar usuarios	100 %

El bibliotecario selecciona la opción “Nuevo”	Muestra el formulario para introducir los datos del nuevo usuario	No muestra el formulario	Satisfactorio	La operación se realizó exitosamente	100 %
El bibliotecario selecciona la opción “Adicionar”	Se adiciona correctamente el usuario	No se adiciona el usuario	Satisfactorio	Si no es posible realizar la operación el sistema muestra un mensaje de error	100 %

Tabla 14: Descripción de prueba de caja negra a la interfaz Buscar usuarios

Sección: Buscar usuarios					
Acción	**Clases válidas**	**Clases no válidas**	**Resultado de la prueba**	**Observaciones**	**Cumplimiento**
El bibliotecario selección la opción para gestionar usuarios del menú principal	El sistema muestra la interfaz correctamente	El sistema no muestra la interfaz al bibliotecario	Satisfactorio	Haciendo uso de la interfaz para gestionar usuarios	100 %
El bibliotecario llena los datos	El sistema muestra aquellos	No se muestra los usuarios que	Satisfactorio	Si no existen usuarios que	100 %

para la búsqueda y presiona la opción “Buscar”	usuarios que coincidan con los datos entrados	coincidan con los datos entrados		coincidan con los datos entrados el sistema muestra un mensaje de alerta	
El bibliotecario accede a la información detallada del usuario	El sistema muestra todos los datos del perfil de usuario	El sistema no muestra la información extendida del usuario	Satisfactorio	El sistema muestra la información completa del usuario y brinda otras opciones	100 %

Tabla 15: Descripción de prueba de caja negra a la interfaz Buscar usuarios

Sección: Modificar usuarios					
Acción	**Clases válidas**	**Clases no válidas**	**Resultado de la prueba**	**Observaciones**	**Cumplimiento**
El bibliotecario selecciona la opción “Usuarios” de menú principal	El sistema muestra la interfaz correspondiente	El sistema no muestra la interfaz correcta	Satisfactorio	Haciendo uso de la interfaz para gestionar usuarios	100 %
El bibliotecario busca al usuario	El sistema muestra la	El sistema no muestra la	Satisfactorio	El sistema muestra un	100 %

que va a modificar	información del usuario	información del usuario		formulario para modificar los datos del usuario	
El bibliotecario introduce los nuevos datos y presiona la opción “Aceptar”	El sistema modifica los datos del usuario y los muestra	El sistema no modifica los datos del usuario, ni los muestra el bibliotecario	Satisfactorio	Si no es posible modificar los datos del usuario el sistema muestra un mensaje de error	100 %

Tabla 16: Descripción de prueba de caja negra a la interfaz Modificar usuarios

Sección: Eliminar usuarios					
Acción	**Clases válidas**	**Clases no válidas**	**Resultado de la prueba**	**Observaciones**	**Cumplimiento**
El bibliotecario selecciona la opción para gestionar usuarios del menú principal	Muestra la interfaz de usuarios	No se muestra la interfaz	Satisfactorio	Haciendo uso de la interfaz para gestionar usuarios	100 %
El bibliotecario realiza la	Se muestra el	No se muestra el	Satisfactorio	Si no hay ningún usuario	100 %

búsqueda del usuario	usuario buscado	usuario buscado		con los datos de la búsqueda el sistema emite un aviso	
El bibliotecario selecciona el usuario y da clic en la opción “Eliminar Usuario”	Se elimina correctamente el usuario seleccionado	No se elimina el usuario	Satisfactorio	Si no es posible eliminar el usuario el sistema muestra un mensaje error	100 %

Tabla 17: Descripción de prueba de caja negra a la interfaz Eliminar usuarios

3.5. Conclusiones

En este capítulo se describen los métodos de pruebas usados en el desarrollo de la aplicación. Además se muestran ejemplos de pruebas de caja blanca y caja negra que se le realizaron al software, demostrando su adecuado funcionamiento y reflejando la calidad con que ha sido llevada a cabo la proyección del sistema. Estas acciones realizadas durante la implementación, tienen el objetivo de asegurar que el sistema implementado tenga la calidad necesaria para lograr la satisfacción del cliente.

4. CONCLUSIONES GENERALES

Con la culminación de este trabajo:

- ✔ Se identificaron y se implementaron los cambios necesarios al módulo del circulación de Koha para que la BNCJM lo pueda usar.
- ✔ Se hicieron las pruebas necesarias que validan la solución propuesta.
- ✔ El módulo de circulación está listo para ser desplegado en la BNCJM.

Con el desarrollo de este proyecto de diploma se dio cumplimiento al objetivo general propuesto, obteniéndose un aplicación web que mejora en gran medida los procesos de circulación de materiales bibliográficos en la BNCJM.

5. RECOMENDACIONES

Se propone que:

- ✔ Se implemente una nueva funcionalidad al módulo de circulación para gestionar los pedidos de acervos bibliográficos que no se encuentren en el mismo local donde se prestan.
- ✔ Se implante Koha en las bibliotecas del país que tengan funcionamiento similar a la BNCJM y puedan mejorar su servicio.
- ✔ Se publiquen los cambios realizados para dar a conocer el trabajo realizado y con el objetivo intercambiar ideas con instituciones de Cuba y del mundo.

6. REFERENCIAS BIBLIOGRÁFICAS

[1]. **Wall, Lary.** [En línea]. [Citado el: 13 de noviembre del 2008]. Disponible en: [http://www.perl.org].

[2]. **Alvarez, Miguel Angel**. Qué es Javascript. desarrolloweb.com. [En línea]. [Citado el: 14 de noviembre de 2008.] Disponible en: [http://www.desarrolloweb.com/articulos/25.php].

[3]. Ídem referencia [2].

[4]. Ídem referencia [2].

[5]. **Pozo, Salvador**. MySQL con Clase. [En línea]. [Citado el: 13 de noviembre del 2008.] Disponible en: [http://mysql.conclase.net/curso/index.php].

[6]. Ídem referencia [5].

[7]. **Ciberaula**. Una Introducción a APACHE. Ciberaula. [En línea]. [Citado el: 14 de noviembre de 2008.] . Disponible en: [http://linux.ciberaula.com/articulo/linux_apache_intro].

[8]. Ídem referencia [7].

[9]. Ídem referencia [7].

[10]. **Anjuta Team**. [En línea]. [Citado el: 15 de noviembre del 2008]. Disponible en: [http://anjuta.org/].

[11]. **Reynoso, Carlos Billy**. Introducción a la arquitectura de software. 2004.

[12]. **Microsoft**. Model View Controller. MSDN. [En línea]. [Citado el: 20 de marzo de 2009]. Disponible en: [http://msdn.microsoft.com/en-us/library/ms978748.aspx].

[13]. **Dpto ISW UCI.** ISW1 Conf 3 Flujo de trabajo de requerimientos. Ciudad Habana. 2007.

[14]. **Doffman, M. and Thayer, R**. Standards, Guidelines and Examples on System and Software. s.l. : IEEE Computer Society Press, 1990.

7. BIBLIOGRAFÍA

BNCJM. *Proyecto Manual de Procedimientos de los servicios*. Ciudad de La Habana. 2004

BNCJM. *Reglamento de la Biblioteca Nacional.* Ciudad de La Habana. 2004

Bunce, Tim. *DBI-1.607.* [En línea]. [Consultado el: 6 de marzo de 2009]. Disponible en: [http://search.cpan.org/~timb/DBI-1.607/DBI.pm].

Engard, Nicole. Ayuda de Koha**.** *Koha 3.01.X Manual*. [En línea]. [Consultado el: 15 de marzo de 2009]. Disponible en: [https://sites.google.com/a/liblime.com/koha-manual/pdfs].

Ferraro, Joshua. *Installation Guide for Installing Koha on Debian Etch with MySQL 5.* Abril de 2008

García, J. *Manual de MySQL.* 2006.

Index Data. INDEXDATA. *Zebra.* [En línea]. [Consultado el: 20 de enero de 2009]. Disponible en: [http://www.indexdata.dk/zebra].

Koha Development Team. *Koha Venezuela*.[En línea]. [Consultado el: 10 de noviembre de 2008] Disponible en: [http://koha.org.ve/?q=node/3]. 2008

León, Casiano R. *"Perl: Fundamentos, Procesos y Lenguajes"*. 2006

Microsoft. *Model View Controller. MSDN*. [En línea]. [Consultado el: 20 de marzo de 2009]. Disponible en: [http://msdn.microsoft.com/en-us/library/ms978748.aspx].

Mobile Encyclopedia. Wiki: Koha. [En línea]. [Consultado el: 20 de noviembre de 2008]. Disponible en: [http://wapedia.mobi/es/Koha]. 2008

Reynoso, Carlos Billy. Introducción a la arquitectura de software. 2004

Pressman, **R**. Ingeniería del Software. *Un enfoque práctico.* La Habana: Félix Varela. 2005.

Rey Lazarte, O. biblioteca.uci.cu. *Diseño del módulo de Circulación para la Biblioteca Nacional José Martí.* [En línea]. [Consultado el: 12 de noviembre de 2008]. Disponible en: [http://bibliodoc.uci.cu/TD/TD_0917_07.pdf]. 2007

Stein, Lincoln D. *CGI.pm-3.43.* [En línea]. [Consultado el: 6 de marzo de 2009]. Disponible en: [http://search.cpan.org/~lds/CGI.pm-3.43/CGI.pm].

Sun Microsystems, Inc. *Java BluePrints: Model-View-Controller*. [En línea] . [Consultado el: 25 de marzo de 2008]. Disponible en: [http://java.sun.com/blueprints/patterns/MVC-detailed.html].

The Apache Software Foundation. The Apache Software Foundation. *Apache projects.* [En línea]. [Consultado el: 20 de noviembre de 2008]. Disponible en: [http://www.apache.org].

The Koha Development Team, y Katipo Communications Ltd. 2006. *Koha - Open Source ILS - Integrated.* [En línea]. [Consultado el: 25 noviembre.] Disponible en: [http://www.koha.org].

The Perl Foundation. Perl.org. *Site Information and Contacts*. [En línea]. [Consultado el: 2 de diciembre de 2008]. Disponible en: [http://www.perl.org].

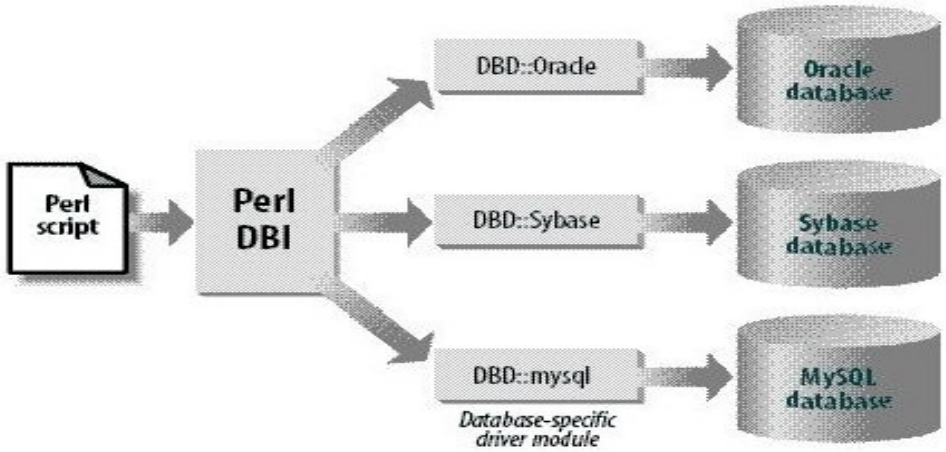
Tregar, Sam. *HTML Template-2.9.* [En línea]. [Consultado el: 5 de marzo de 2009]. Disponible en: [http://search.cpan.org/~samtregar/HTML-Template-2.9/Template.pm].

Veloz Morales, Dargel y Porras Herrera, L. Amalia. biblioteca.uci.cu. *Implantación de un Sistema Integrado de Gestión Bibliotecaria (SIGB) en la biblioteca de la UCI.* [En línea]. [Consultado el: 11 de noviembre de 2008]. Disponible en:[http://bibliodoc.uci.cu/TD/TD_1257_08.pdff]. 2008

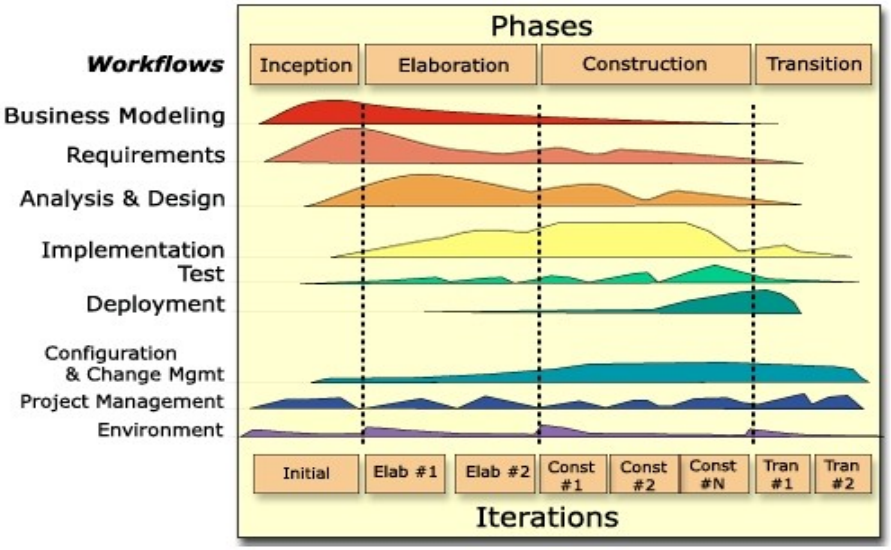
Vromans, Johan. *"Programming Perl".* 1998

8. ANEXOS

8.1. Anexo 1: Arquitectura de DBI



8.2. Anexo 2: Gráfico de la Metodología de desarrollo RUP



9. GLOSARIO DE TÉRMINOS

Cliente: Sistema que establece un intercambio de datos con un servidor.

Deep linking: conocido en español como enlace profundo, es un hipervínculo que no apunta a la página principal de un sitio, sino a una página interna específica o imagen de las que conforman un sitio web.

Framework: es una estructura de soporte definida en la cual otro proyecto de software puede ser organizado y desarrollado. Típicamente, un framework puede incluir soporte de programas, bibliotecas y un lenguaje interpretado entre otros software para ayudar a desarrollar y unir los diferentes componentes de un proyecto.

GPL: La GNU General Public License (inglés: Licencia Pública General) es una licencia creada por la Free Software Foundation y orientada principalmente a los términos de distribución, modificación y uso de software. Su propósito es declarar que el software cubierto por esta licencia es Software Libre.

IDE: Entorno de desarrollo integrado o en inglés Integrated Development Environment ('IDE'). Es un programa compuesto por un conjunto de herramientas para un programador desde el que se pueden editar programas, compilarlos y depurarlos.

Indizador: es un buscador que registra ordenadamente datos e informaciones para la elaboración de sus propios índices.

LDAP: por sus siglas en inglés Lightweight Directory Access Protocol , es un protocolo utilizado para acceder a información almacenada en un directorio de información así como usuarios, contraseñas y otras entidades en un entorno de red, funciona como una base de datos optimizada para las operaciones de lectura y búsqueda .

MARC21: Es un registro catalográfico legible por máquina (MAchine- Readable Cataloging) "Legible por máquina" significa que un tipo particular de máquina, una computadora, puede leer e interpretar los datos contenidos en un registro catalográfico.

MicroISIS: MicroISIS o Micro CDS/ISIS, (Computerized Documentation System - Integrated Set for Information System), es un sistema generalizado de almacenamiento y recuperación de información,

diseñado especialmente para el manejo de bases de datos constituidas principalmente por texto, es en general un archivo de datos relacionados generados para satisfacer los requerimientos de información de los usuarios.

NCIP: NISO Circulation Interchange Protocol, es un protocolo utilizado para normalizar las transacciones de circulación entre distintos sistemas de gestión bibliotecarios, permitiendo el préstamo directo a usuarios que pertenezcan a otras bibliotecas.

Opensearch: es un conjunto de tecnologías que permiten publicar los resultados de una búsqueda en un formato adecuado para la sindicación y agregación. Es una forma para que las páginas web y los motores de búsqueda publiquen sus resultados de forma accesible.

Registro bibliográfico: Conjunto de datos que describen y representan un documento concreto, con vistas a su recuperación dentro de un catálogo de biblioteca o en una base de datos bibliográfica. Contiene todos los datos necesarios para describir (de forma breve o completa, según el nivel de la descripción) el documento, presentados dichos datos en un formato bibliográfico específico normalizado. En la actualidad dicho formato es legible por ordenador (MARC). Un registro bibliográfico suele incluir datos sobre el autor, el título, la edición, el editor, serie, notas y números normalizados del documento.

RSS: del inglés Really Simple Syndication es un formato para la sindicación de contenidos de páginas web, es decir son utilizados para la publicación y actualización constante de información.

Servidor: computadora central de un sistema de red que provee servicios y recursos (programas, comunicaciones, archivos, etc.) a otras computadoras (clientes) conectadas a ella.

SGBD: Un Sistema Gestor o Manejador de Bases de Datos es un conjunto de programas que permite a los usuarios crear y mantener una base de datos, por lo tanto, el SGBD es un software de propósito general que facilita el proceso de definir, construir y manipular la base de datos para diversas aplicaciones. Pueden ser de propósito general o específico.

SIGB: “conjunto de recursos humanos que utilizan dispositivos y programas informáticos, adecuados a las naturaleza de los datos, para realizar procesos y facilitar los servicios que permiten alcanzar el objetivo de la biblioteca: almacenar de forma organizada el conocimiento humano contenido en todo tipo de

materiales bibliográficos para satisfacer las necesidades informativas de, formativas, recreativas y/o de investigación de los usuarios" (García Melero; García Camarero, 1999).

Software Libre: se refiere a la libertad de los usuarios para ejecutar, copiar, distribuir, estudiar, cambiar y mejorar el software.

UNIMARC: Es un formato para registros de autoridad, para registros bibliográficos y para registros de fondos y localizaciones.

WWW: (World Wide Web) Telaraña o malla mundial. Sistema de información con mecanismos de hipertexto. Los usuarios pueden crear, editar y visualizar documentos de hipertexto. También llamado W3.

XML: es un metalenguaje extensible de etiquetas desarrollado por el World Wide Web Consortium (W3C).

Z3950: Es el nombre de un estándar definido por ANSI/NISO que permite comunicar sistemas que funcionan en distinto hardware y usan distinto software. Fue diseñado para solucionar los problemas asociados a la búsqueda en múltiples bases de datos con diferentes lenguajes y procedimientos.