Universidad de las Ciencias Informáticas

Facultad 10

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

Título: Arquitectura de Software de un Sistema Integrado para la Biblioteca Nacional "José Martí"

Autores: Edisnel Carrazana Castro / Leonardo Góngora Méndez

Tutor: Lc. Luis Guzmán Hernández

Junio, 2007

“El temor de Jehová es el principio de la sabiduría,
Y el conocimiento del Santísimo es la inteligencia.”
La Biblia (Proverbios 9:10)

DECLARACIÓN DE AUTORÍA

Declaramos ser autores de la presente tesis y reconocemos a la Universidad de las Ciencias Informáticas los derechos patrimoniales de la misma, con carácter exclusivo.

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

Leonardo Góngora Méndez	Edisnel Carrazana Castro	Luis Guzmán Hernández

DATOS DE CONTACTO

Lic. Luis Joaquín Guzmán Hernández (luisgh@uci.cu)

Graduado de Licenciatura en Ciencias de la Computación en la Universidad Central "Martha Abreu" de Las Villas en el año 2002. Trabajó durante dos años en el Centro de Información Científica de la Universidad Central como especialista en informática. Actuó como líder en varios proyectos de la rama de la gestión de información. En el año 2005 comienza a trabajar en la UCI, desempeñándose como especialista en una dirección de la infraestructura productiva. Ha participado en muchos proyectos de desarrollo de software, fundamentalmente en el desarrollo de aplicaciones para WEB. Actualmente se desempeña como Director de la dirección de producción # 2 de la Infraestructura Productiva.

AGRADECIMIENTOS

A ti Señor Jesucristo, por habernos amado antes, por salvar nuestras almas, por derramar en nuestro corazones gozo, paz, y amor; gracias Señor, por tu misericordia y tu perdón, gracias por tu fidelidad, pues ciertamente cuando todo apuntaba al fracaso, tú prometiste que este proyecto saldría adelante, y hoy te damos toda la gloria a ti, a tu nombre Jesús. Que nuestras vidas sean siempre una ofrenda grata delante de ti.

A nuestros hermanos en la fe que siempre nos apoyaron en oración para que este proyecto pudiese hoy ser realidad.

A nuestros padres, por su amor, cariño, y toda una vida de constante dedicación.

A nuestro tutor Luis Guzmán Hernández, por haber confiado en nosotros siempre, por su comprensión, ayuda, apoyo y por todo el tiempo dedicado. A nuestro comité de tesis, de manera especial a Orlando Cárdenas Fernández, Lissette Soto Pelegrín, David Leyva Leyva, y además, a todos aquellos que de una forma u otra nos han ayudado y brindado su apoyo.

A nuestros profesores, que a lo largo de toda nuestra vida estudiantil han contribuido a nuestra formación profesional.A todos los estudiantes y trabajadores de la BNJM que están vinculados al proyecto, gracias por su apoyo y sus contribuciones.

De Edisnel:
A mi hermano Leo, por su dedicación y entrega, por mantenerse firme en todo momento y por siempre tener en cuenta como se debe trabajar en unión.

De Leonardo:
A mi hermano Edi, por su paciencia y por permitirme trabajar junto a él en este proyecto. Realmente trabajamos como un verdadero equipo y sin ello, este trabajo no hubiera sido posible.

DEDICATORIA

De Edisnel:

A mis padres: Wilfredo, Nelsis y Edilberto.

A mis hermanos: Yoendris y Yolennis.

A mi familia.

De Leonardo:

A mis padres: Rafael y Aurora.

A mis abuelos: Yoel y Ana.

A mi hermanita: Doris Mirtha.

A mi familia.

RESUMEN

Uno de los sectores que se ha propuesto integrarse plenamente a la infraestructura global de la información, es el Ministerio de Cultura (MINCULT). Uno de los pasos dados en este sentido, es el programa de automatización de la gestión integral del Sistema Nacional de Bibliotecas Públicas (SNBP), con el objetivo de modernizar los servicios bibliotecarios y de información que se ofertan al pueblo y servir de soporte eficaz a los programas de la Revolución. Como parte importante del programa antes mencionado, se ha hecho el estudio de los procesos de gestión bibliotecaria de la Biblioteca Nacional José Martí (BNJM), institución rectora del SNBP, a partir de la cual se han identificado los problemas fundamentales en el sector bibliotecario del país, que plantean la tarea de cómo llevar a cabo la automatización de los principales procesos bibliotecarios mediante la construcción de un Sistema Integrado para Bibliotecas. Este trabajo propone una arquitectura de software que servirá de guía para desarrollar un Sistema Integrado para Bibliotecas (en Inglés – Integrated Library System) para la BNJM, que permitirá automatizar los principales procesos bibliotecarios, mejorando la gestión de la información y la calidad de los servicios.

Palabras claves

Sistema Integrado para Bibliotecas (ILS – Integrated Library System), arquitectura de software, cohesión, acoplamiento, modularidad, componente, escalabilidad, flexibilidad, interface, servidores de aplicaciones.

ÍNDICE

.

INTRODUCCIÓN

En nuestro país la BNJM es la encargada de compilar, publicar y difundir la bibliografía nacional cubana, entendiéndose como tal, la recepción y sistematización de la obras publicadas en el país o en el extranjero sobre Cuba y sus naturales, así como las que se publican en el extranjero por autores cubanos

La BNJM constituye un eslabón fundamental en el desarrollo sociocultural y profesional de nuestro país, y a pesar de ello, la misma carece de un ILS que permita la automatización de sus procesos internos y responda a las actuales demandadas de la institución; afectando en gran medida su buen funcionamiento, y repercutiendo fundamentalmente en lo concerniente al procesamiento de materiales y en la calidad de los servicios que se brindan en la biblioteca.

De la situación anterior se identificó el siguiente ***problema***: ¿se puede construir un Sistema Integrado para Bibliotecas que sea capaz de automatizar los procesos de núcleo de la Biblioteca Nacional José Martí?

Para crear un ILS que responda a las necesidades requeridas por la institución, es necesario estudiar los procesos bibliotecarios que se llevan a cabo en la misma. Por lo que el ***objeto de estudio*** se centra en los procesos de gestión bibliotecaria en la BNJM, y el ***campo de acción*** se limita a las actividades relacionadas con la automatización del procesamiento de materiales bibliográficos y servicios al público.

Un elemento determinante en el desarrollo de software lo constituye la arquitectura de software. Los ILS usan una arquitectura de software única para administrar los procesos de núcleo de las bibliotecas incluyendo catalogación, circulación, adquisiciones y control financiero y, con ciertos grados de variación, otros asociados, pero tal vez funciones menos críticas tales como sistemas de préstamo interbibliotecario e información de administración.

Como ***objetivo general*** se propone diseñar una arquitectura de software que permita el desarrollo de un ILS.

Para el desarrollo de la investigación nos proponemos responder a las siguientes **preguntas científicas**:

- ¿Cuáles son los principales problemas que presentan los ILS más conocidos, desarrollados sobre tecnología libre?
- ¿Cuáles son los principales subsistemas que conformarían el ILS?
- ¿Cuál es el tipo de arquitectura de software a emplear para el desarrollo del ILS?
- ¿Cuáles son los patrones de arquitectura que pueden ser aplicables al tipo de arquitectura seleccionada?
- ¿Permite la arquitectura seleccionada implementar la funcionalidad básica deseada?
- ¿Responde la arquitectura seleccionada a los atributos de calidad establecidos por los clientes?

Con el objetivo de guiar, controlar y evaluar la investigación se definieron las siguientes ***tareas***:

- Actualizar los logros y limitaciones de los ILS existentes más conocidos, enfatizando en los que están soportados sobre tecnologías libres.
- Evaluar el contenido de la información obtenida sobre los ILS analizados, establecer un diagnóstico de las tendencias actuales y tomar posición al respecto, identificando los principales subsistemas que conforman los ILS más implantados.
- Realizar un estudio detallado sobre arquitecturas de software que recoja los siguientes aspectos:
    - Modelos de arquitecturas existentes y sus aplicaciones.
    - Estilos y patrones de arquitectura.
    - Tendencias actuales.
- Escoger el tipo de arquitectura a emplear e identificar los estilos y/o patrones de arquitectura que permitan un mínimo acoplamiento y una máxima cohesión entre los componentes del sistema.
- Evaluar la arquitectura de software propuesta, verificando que la misma permita la implementación de la funcionalidad deseada y que cumpla con los atributos de calidad establecidos.

Los **métodos teóricos** que serán utilizados para darle cumplimiento a estas tareas son *Analítico-Sintético* centrándose en el análisis de las teorías, documentos, entre otros; permitiendo la extracción de los elementos más importantes de manera que se procese la información y se elaboren conclusiones; y el *Análisis Histórico-Lógico* para el estudio de la evolución y desarrollo del objeto de estudio de la investigación. Se utilizarán además los ***métodos empíricos*** de la entrevista como técnicas para la recopilación de la información del fenómeno.

Con el logro de los objetivos propuestos en el presente trabajo se obtendría una arquitectura que permitirá construir un sistema integrado para bibliotecas capaz de satisfacer las necesidades de la BNJM, al automatizar los procesos más importantes de la misma; trayendo consigo una mejora en cuanto a la calidad de los servicios a usuarios que brindan esta institución haciendo menor el tiempo de respuesta a las necesidades de los mismos; los trabajadores podrán acceder de forma eficiente a la información reduciendo el tiempo que se dedica comúnmente a las actividades laborales; el sistema debe permitir la comunicación con otras bibliotecas, lo que ahorraría tiempo de trabajo y esfuerzo en las actividades que se realizan, como la catalogación, pues la información de cualquier material que se encuentre disponible en cualquier biblioteca podrá ser accedida desde el sistema.

El presente trabajo consta de tres capítulos, las conclusiones generales, recomendaciones, referencias bibliográficas y bibliografía utilizada durante el desarrollo del trabajo, un glosario de términos y por último los anexos que complementan el cuerpo del trabajo y que son necesarios para su entendimiento.

En el primer capítulo se expone la fundamentación teórica del tema. Incluye una descripción detallada del objeto de estudio, con la finalidad de comprender el negocio. Se presenta el estado del arte de la arquitectura de software, así como los logros y limitaciones fundamentales de los ILS más usados en la actualidad, y además se hace un resumen de las tendencias y tecnologías actuales utilizadas en el desarrollo de sistemas informáticos. También se hace referencia a las metodologías, herramientas y tecnologías a utilizar en el desarrollo del sistema.

El segundo capítulo describe el propósito y funcionalidad del sistema. Se hace referencia a las restricciones de diseño; y además se documentan las vistas arquitectónicas seleccionadas para

representar la arquitectura.

En el tercer capítulo se realiza una evaluación de la arquitectura propuesta en esta tesis. La arquitectura se evalúa para verificar que cumpla con todos los requerimientos; específicamente con los atributos de calidad establecidos.

CAPÍTULO 1: Fundamentación Teórica

1.1 Objeto de estudio.

1.1.1. Descripción general.

La Biblioteca Nacional “José Martí” tiene como misión: compilar, publicar y difundir la Bibliografía Nacional Cubana, entendiéndose como tal, la recepción y sistematización de las obras publicadas en el país o en el extranjero sobre Cuba y sus naturales, así como las que se publican en el extranjero por autores cubanos. Asimismo, lleva a cabo la tarea de adquirir, procesar y conservar lo más representativo de la literatura universal. Igualmente funge como Centro Bibliográfico Nacional, depositaria de la documentación de Naciones Unidas y custodio de la papelería del Ministerio de Cultura.

En sus diferentes salas, general y especializadas, se le presta servicios a un público, en la actualidad categorizado, y cuenta además, con los departamentos Circulante y Juvenil, que ofrecen servicios típicos de biblioteca pública a todas las categorías de lectores. Por otra parte, la BNJM se ha constituido en centro difusor del arte y de la cultura nacional y universal, a través de sus galerías, cátedras, exposiciones y otras actividades de promoción.

1.1.2. Estructura interna de la BNJM.

La BNJM está organizada en varias Subdirecciones, pero solamente existen dos Subdirecciones que están directamente relacionadas con el procesamiento de materiales bibliográficos y los servicios, la *Subdirección de Procesos Técnicos* y *Subdirección de Servicio al Público*. Dentro de los objetivos de ambas subdirecciones se destacan:

Objetivos de la Subdirección de Procesos Técnicos.

- Planificar, organizar y controlar la actividad de procesamiento de la BNJM así como velar por su ejecución en el Sistema de Bibliotecas Públicas, de acuerdo con las actividades aprobadas por la

Dirección General.

- Programar, organizar y controlar la actividad de selección, adquisición y procesamiento de la Biblioteca, de acuerdo con las políticas establecidas.

- Asesorar metodológicamente el Sistema de Bibliotecas Públicas en cuanto a los procesos de formación y procesamiento de sus colecciones.

Objetivos de la Subdirección de Servicio al Público.

- Programar, organizar y controlar la prestación de servicio de la BNJM de acuerdo con los fines de ésta a través de sus diferentes áreas.
- Asesorar metodológicamente al SNBP en todo lo concerniente a la prestación de servicios a través de los diferentes departamentos y grupos especializados. En el caso de las colecciones especiales, la asesoría se extiende al procesamiento de los diferentes tipos de documentos.
- Velar por la organización y control de los documentos en los depósitos, con énfasis en las colecciones patrimoniales.

Cada Subdirección a su vez está constituida por varios departamentos y grupos:

Subdirección de Procesos Técnicos.

- Departamento de Automatización.
- Departamento de Desarrollo de Colecciones.
- Departamento de Procesos del Libro y otros Documentos.
- Departamento de Procesos de Publicaciones Seriadas.
- Departamento de Bibliographic Cubana.
- Departamento de Conservación.
- Grupo de Patrimonio.

Subdirección de Servicio al Público.

- Departamento de Servicios Generales, Catálogo Colectivo y Préstamo Interbibliotecario.
- Departamento de Información Especializada.
- Departamento de Fondos Bibliográficos.
- Departamento de Servicio Especial para Niños y Jóvenes.
- Departamento de Circulante.
- Grupo de Bibliografía de Naciones Unidas.
- Grupo de Categorización de Información.
- Grupo de Referencia.

Ambas Subdirecciones están muy relacionadas entre sí, ya que los materiales que se ponen a disposición del público, en la mayoría de los casos, primeramente han sido procesados.

1.1.3. Descripción del flujo actual de los procesos.

Las Subdirecciones de Procesos Técnicos y Servicio al Público, básicamente se dedican a procesar materiales bibliográficos y a poner los mismos a disposición de los usuarios, respectivamente. Entiéndase por materiales bibliográficos: libros, folletos, catálogos de exposición, guías turísticas, mapas, CD-ROM, DVD, publicaciones seriadas, entre otros. Estas subdirecciones se relacionan a través del flujo básico por el cual transitan los documentos. Este flujo está compuesto fundamentalmente por tres grandes procesos: Adquisición – Procesamiento – Servicio.

El procesamiento de materiales comienza con la entrada de los mismos a la Biblioteca Nacional, y es el Departamento de Desarrollo de Colecciones el encargado del proceso de *recepción*, *selección* y *adquisición* de dichos materiales. Una vez que el material ha sido seleccionado, es enviado al área donde el mismo será procesado, esta área depende del tipo de material.

El departamento de Procesos Técnicos es el encargado de procesar libros, folletos y materiales especiales (CD-ROM, cassetes, discos $3^{1/2}$, videos, DVD), y además es quien establece un asiento normalizado para cada tipo de documento de acuerdo a normas nacionales e internacionales, tanto para

el enriquecimiento del catálogo de autoridades y del vocabulario autorizado como para la función de *catalogación* de cualquier documento, garantizando la uniformidad de los accesos de todos los documentos bibliográficos de la BNJM.

El departamento de Publicaciones Seriadas realiza funciones muy similares al departamento de Procesos Técnicos, pero enfocados solamente en los materiales que se clasifican como publicaciones seriadas (revistas, periódicos, etc.). Existen además otras áreas en la institución donde se procesan documentos.

Luego de haber sido procesados los materiales, los mismos pasan a formar parte del Fondo de la BNJM, poniéndose los documentos a disposición de los clientes. A través de las Salas de Servicios al Público, se brindan servicios de lectura, referencia, préstamos, entre otros. Los usuarios pueden acceder a la información en dependencia a la categoría de usuario a la que pertenezcan. De esta manera se completa el flujo de documentos: Adquisición – Procesamiento – Servicio.

Independiente al flujo básico de los documentos, se realizan otros procesos. En el departamento de Bibliografía Nacional se lleva a cabo la actividad de compilación bibliográfica de la Bibliografía Nacional, el Índice General de Publicaciones Periódicas Cubanas, las Bibliografías especializadas y personales y los índices de colecciones cerradas de revistas cubanas; contribuyendo al desarrollo, organización y difusión de los valores esenciales de la identidad cultural.

1.1.4. Análisis crítico de la ejecución de los procesos.

Actualmente, la actualización y recuperación de una parte de la información que se genera en el proceso explicado con anterioridad, se realiza en papel de forma manual, lo que provoca que se alargue el período de actualización de la información y que se introduzcan errores humanos, esto ha traído consigo la imposibilidad de lograr, de una manera eficaz, el funcionamiento de la institución; afectando fundamentalmente lo concerniente al procesamiento de materiales y a la calidad de los servicios que se brindan en la biblioteca.

Dentro de toda la información que se genera, se encuentran los registros bibliográficos, que no son más

que los datos bibliográficos que se asocian a cada ejemplar. La BNJM se apoya en algunas herramientas de software que le facilitan la búsqueda y recuperación de información. La BNJM actualmente hace uso de un software, conocido por MicroIsis, que le permite la gestión de registros bibliográficos. MicroIsis fue creado por la UNESCO (United Nations Educational, Scientific and Cultural Organization), y aunque en sus comienzos llegó a ser una aplicación muy útil y popular entre los bibliotecarios, hoy en día se ha convertido en un software obsoleto, no pudiendo dar respuestas a las actuales demandas de la biblioteca, afectando el desarrollo de muchos de los procesos que se llevan a cabo en la institución.

Debido a las limitaciones antes mencionadas, la calidad en la gestión de la información en la biblioteca se ve afectada en gran medida, pues los mecanismos existentes para el procesamiento de materiales bibliográficos no garantizan la integridad de los datos, permitiendo duplicidad y errores en la información, además de ralentizar todo el proceso; esto incide directamente en la calidad de los servicios que se prestan al público.

1.2 Procesos de objeto de automatización.

Se desea automatizar todo el proceso por el cual transitan los documentos en la biblioteca, desde su llegada a la misma, hasta su entrega al usuario final; así como también la gestión de los servicios que presta la institución. De esto se deriva la necesidad de automatizar los procesos de selección, adquisición y catalogación de materiales, estos en lo relacionado al procesamiento de documentos; y en cuanto a los servicios, es necesario automatizar lo referente a los préstamos, gestión de usuarios, consultas y servicios de referencia y bibliografía nacional.

1.3 Sistemas automatizados existentes vinculados al campo de acción.

Aunque probablemente no lo parezca, los desafíos de la automatización de las bibliotecas de los años 70, eran menos complejos que aquellos que tenemos hoy. Para la década pasada, las colecciones de las bibliotecas han crecido constantemente para incluir diferentes tipos de contenido. Las bibliotecas de todas las clases gastan una parte principal de sus presupuestos de colección en suscripciones al contenido electrónico. Algunas concesiones menores al contenido electrónico surgieron, pero fundamentalmente los sistemas automatizados para bibliotecas están enfocados a los contenidos y flujos de trabajo

tradicionales.

En el 2003, la Biblioteca Nacional “José Martí” valoró la posibilidad de crear una aplicación que facilitara y automatizara los procesos bibliotecarios, y que además pudiese ser implantada en las bibliotecas públicas. El sistema fue contratado a la Universidad de Las Villas “Martha Abreu”, y el mismo se dio a conocer como el proyecto “QuipusNet”. Los objetivos de este sistema eran: facilitar, controlar y archivar el recorrido de cada material que se adquiriera; permitir la realización de consultas y la recepción de reportes sobre el estado de los documentos y facilitar los procesos de administración económica y estructuración virtual.

A pesar de que los comienzos del proyecto fueron muy satisfactorios, por determinadas razones el proyecto dejó de existir, por lo que la institución nunca llegó a contar con una aplicación que permitiera solucionar sus problemas.

En cuanto al ámbito internacional, se han desarrollado diversas soluciones y herramientas. Como parte de este trabajo se ha realizado un estudio de estas soluciones, específicamente las de código abierto. La mayor parte de ellas responde al modelo LAMP (Linux, Apache, MySQL y PHP), y han sido escritas en Perl o PHP. Pueden ofrecer mayores o menores prestaciones, pero básicamente dan soporte al formato MARC (Machine – Readable Cataloging), permitiendo la gestión bibliográfica de catálogos, e incorporan un OPAC (Online Public Access Catalog) avanzado. La mayoría de estas herramientas también ofrecen servidores Z39.50 (protocolo cliente – servidor para la búsqueda y recuperación de información en bases de datos remotas), y posibilidades de importar y exportar información bibliográfica a otros formatos. Sin pretender ser exhaustivo, el siguiente listado contiene los más implantados:

- OpenBiblio: http://obiblio.sourceforge.net
- Emilda: http://www.emilda.org
- Koha: http://www.koha.org
- PHPMyLibrary: www.phpmylibrary.org
- GNUteca: http://www.gnuteca.org.br
- PMB-PhpMyBibli: http://www.sigb.net

De lo presentado anteriormente, aparece claramente la disponibilidad de diversas alternativas de aplicaciones de código abierto, pero sobre todo es una realidad para el segmento de bibliotecas medianas – pequeñas.

Para adoptar un sistema que sea capaz de responder a las necesidades tanto de grandes bibliotecas (con una lógica de negocio compleja y variable, con entornos de trabajo cambiantes, con grandes volúmenes de información y complejos flujos de trabajo), como es el caso de la BNJM, así como de pequeños y medianos centros de documentación, es necesario que cuente con las características de ser modular, flexible, escalable, seguro, y que la tecnología que lo implementa le permita evolucionar y crecer a costos razonables.

Tanto los profesionales de la informática como bibliotecarios, se manifiestan muy exigentes en cuanto a las prestaciones y a la calidad con que debería contar un ILS. La complejidad que trae consigo el desarrollo de un ILS con estas características, unido a la heterogeneidad de los entornos bibliotecarios, presentan a los ILS como aplicaciones empresariales complejas de gran tamaño, específicamente en el campo de las bibliotecas y los centros de documentación. El éxito en el desarrollo de grandes aplicaciones de software, se ve influenciado en gran medida por las tecnologías empleadas.

1.4 Tendencias y tecnologías actuales.

LAMP (Linux, Apache, MySQL, Php) está considerada como una de las mejores herramientas disponibles para que cualquier organización o individuo pueda emplear un servidor Web versátil y potente. Aunque creados por separado, cada una de las tecnologías que lo forman dispone de una serie de características comunes. Especialmente interesante es el hecho que estos cuatro productos pueden funcionar en una amplia gama de hardware, con requerimientos relativamente pequeños sin perder estabilidad. Esto ha convertido a LAMP en la alternativa más adecuada para pequeñas y medianas empresas. Pero se necesita más que esto para lograr implementar un ILS que cumpla con las actuales demandas de la comunidad de usuarios. Queda fuera de los objetivos de esta tesis mostrar una comparativa entre LAMP y otras plataformas en cuanto a ventajas y desventajas en el desarrollo de software corporativo.

En la actualidad, las arquitecturas de software comercial como .NET de Microsoft y J2EE (Java 2 Platform, Enterprise Edition) de Sun MicroSystem, predominan en el mercado de las tecnologías de la información; pero existe una cuarta alternativa de software libre, una arquitectura de software formada por un trío de aplicaciones llamada: Plone, Zope y Python, que unida al sistema gestor de bases de datos PostgreSQL, forma una arquitectura de software robusta y eficiente, de código abierto y libre.

1.5 Arquitectura de Software.

La arquitectura de software es la parte de la ingeniería del software que se ocupa de la descripción y el tratamiento de la estructura de un sistema como una serie de componentes con el fin de agruparlos adecuadamente y permitir la integración de diferentes grupos de desarrollo en el mismo proyecto. Habitualmente se vincula con la fase de diseño, aunque esto no es estrictamente necesario; su principal objetivo es hacer énfasis en la importancia de la descripción estructural de los sistemas software, un aspecto bien conocido pero poco tratado.

En este capítulo se hará una introducción a los conceptos fundamentales del campo, así como un breve estudio de las alternativas disponibles para la representación y documentación de la arquitectura de software, finalmente se seleccionará una de estas alternativas, la que será utilizada a lo largo de este trabajo para representar y documentar la arquitectura de software.

1.5.1. Definición de Arquitectura de Software.

No es novedad que ninguna definición de la Arquitectura de software (AS) es respaldada unánimemente por la totalidad de los arquitectos. El número de definiciones circulantes alcanza un orden de tres dígitos, amenazando llegar a cuatro. De hecho, existen grandes compilaciones de definiciones alternativas o contrapuestas, como la colección que se encuentra en [SEI], a la que cada quien puede agregar la suya.
Una definición reconocida es la de Clements [Cle96a]:

La AS es, a grandes rasgos, una vista del sistema que incluye los componentes principales del mismo, la conducta de esos componentes según se la percibe desde el resto del sistema y las formas en que los componentes interactúan y se coordinan para alcanzar la misión del sistema. La vista arquitectónica es

una vista abstracta, aportando el más alto nivel de comprensión y la supresión o diferimiento del detalle inherente a la mayor parte de las abstracciones.

En una definición tal vez demasiado amplia, David Garlan [Gar00] establece que la AS constituye un puente entre el requerimiento y el código, ocupando el lugar que en los gráficos antiguos se reservaba para el diseño. La definición "oficial" de AS se ha acordado que sea la que brinda el documento de IEEE Std 1471-2000, que plantea:

La Arquitectura de Software es la organización fundamental de un sistema encarnada en sus componentes, las relaciones entre ellos y el ambiente y los principios que orientan su diseño y evolución.

Se adoptará en este trabajo como definición de Arquitectura de Software:

La estructura o estructuras del sistema, compuesta por elementos, las propiedades visibles de éstos, y las relaciones que mantienen dentro de ella. [Cle02].

A pesar de la abundancia de definiciones en el campo de la AS, existe en general acuerdo de que ella se refiere a la estructura a grandes rasgos del sistema, estructura consistente en componentes y relaciones entre ellos. [BCK98].

Ante el número y variedad de definiciones existentes de AS, Mary Shaw y David Garlan [SG96] proporcionaron una sistematización iluminadora, explicando las diferencias entre definiciones en función de distintas clases de modelos. Destilando las definiciones y los puntos de vista implícitos o explícitos, los autores clasifican los modelos de esta forma:

- **Modelos estructurales:** Sostienen que la AS está compuesta por componentes, conexiones entre ellos y (usualmente) otros aspectos tales como configuración, estilo, restricciones, semántica, análisis, propiedades, racionalizaciones, requerimientos, necesidades de los participantes. El trabajo en esta área está caracterizada por el desarrollo de lenguajes de descripción arquitectónica (ADLs).

- **Modelos de framework:** Son similares a la vista estructural, pero su énfasis primario radica en la estructura coherente del sistema completo (usualmente una sola), en vez de concentrarse en su composición. Los modelos de framework a menudo se refieren a dominios o clases de problemas específicos. El trabajo que ejemplifica esta variante incluye arquitecturas de software específicas de dominios, como CORBA, o modelos basados en CORBA, o repositorios de componentes específicos, como PRISM.
- **Modelos dinámicos:** Enfatizan la cualidad conductual de los sistemas. “Dinámico” puede referirse a los cambios en la configuración del sistema, o a la dinámica involucrada en el progreso de la computación, tales como valores cambiantes de datos.
- **Modelos de proceso:** Se concentran en la construcción de la arquitectura, y en los pasos o procesos involucrados en esa construcción. En esta perspectiva, la arquitectura es el resultado de seguir un argumento (script) de proceso. Esta vista se ejemplifica con el actual trabajo sobre programación de procesos para derivar arquitecturas.
- **Modelos funcionales:** Una minoría considera la arquitectura como un conjunto de componentes funcionales, organizados en capas que proporcionan servicios hacia arriba. Es tal vez útil pensar en esta visión como un framework particular.

Ninguna de estas vistas excluye a las otras, ni representa un conflicto fundamental sobre lo que es o debe ser la AS. Por el contrario, representan un espectro en la comunidad de investigación sobre distintos énfasis que pueden aplicarse a la arquitectura: sobre sus partes constituyentes, su totalidad, la forma en que se comporta una vez construida, o el proceso de su construcción. Tomadas en su conjunto, destacan más bien un consenso.

1.5.2. Estudio de alternativas para representar y documentar la arquitectura de software.

Tanto los marcos arquitectónicos como las metodologías de modelado de los organismos (ISO, IEEE, OMG) acostumbran ordenar las diferentes perspectivas de una arquitectura en términos de vistas (views). La mayoría reconoce entre tres y seis vistas.

Luego de consultar los marcos arquitectónicos y metodologías de modelado más importantes que nos brindan la academia y la industria en cuanto a selección, evaluación, representación y documentación de

la Arquitectura de Software, hemos adoptado la propuesta plasmada en el libro Documenting Software Architectures: Views and Beyond, de un colectivo de autores reconocidos en la disciplina.

Los autores analizaron muchas experiencias y extrajeron las lecciones importantes aprendidas, revisaron la literatura académica, estudiaron todos lo libros publicados, chequearon los standards y sintetizaron el conocimiento en dicho libro, en el que se expone de forma clara y precisa lo esencial que se necesita para saber definir el documento de la arquitectura de software, sirve de orientación para determinar el alcance de la misma, describe como organizarla, las técnicas, herramientas y notaciones a usar, además de brindar valiosas comparaciones, consejos y plantillas para dar comienzo al trabajo. La propuesta se integra correctamente con los frameworks y standards más utilizados en la academia y la industria del software que se analizaron. La propuesta seleccionada ha determinados los conceptos fundamentales que se utilizarán en el trabajo, así como la forma de representar y documentar sobre la Arquitectura de Software.

1.5.3. Conceptos fundamentales.

- Arquitectura de Software*.*

Es la estructura o estructuras del sistema, compuesta por elementos, las propiedades visibles de éstos, y las relaciones que mantienen dentro de ella.

- Componente.

Uno de los conceptos fundamentales de la Arquitectura de Software es el de componente. Esto se refiere en términos generales a cada una de las partes o unidades de descomposición en las que se subdivide la funcionalidad de un sistema y cuya unión da lugar al sistema completo. En su sentido más genérico puede hacer referencia a cualquier tipo de elemento estructural.

La siguiente definición ha alcanzado cierta popularidad y se considera comúnmente aceptada:

Un componente es una unidad de composición de aplicaciones software, que posee un conjunto de interfaces y un conjunto de requisitos, y que ha de poder ser desarrollado, adquirido, incorporado al sistema y compuesto con otros componentes de forma independiente, en tiempo y espacio. [Szy98]

También tendremos en cuenta la siguiente definición:

Elemento computacional principal y reserva de datos que se ejecuta en un sistema software. [Cle02]

- Conector.

En un célebre artículo [Sha94] de Mary Shaw se propuso considerar por separado las abstracciones relativas a la funcionalidad (el componente) y a su interacción (el conector). De este modo, se realiza una clara separación de intereses que permite ampliar el nivel de abstracción y aumentar la modularidad del sistema. La forma más sencilla de ver un conector es considerándolo como un protocolo de comunicación.

Se adopta como definición de conector:
Un camino de interacción en tiempo de ejecución entre dos o más componentes. [Cle02]

- Puerto.

El concepto de puerto es cercano al de conector, pero no debe ser confundido con este. Se denomina con este nombre a cada uno de los puntos de interacción por los cuales un componente puede realizar cualquier tipo de interacción.

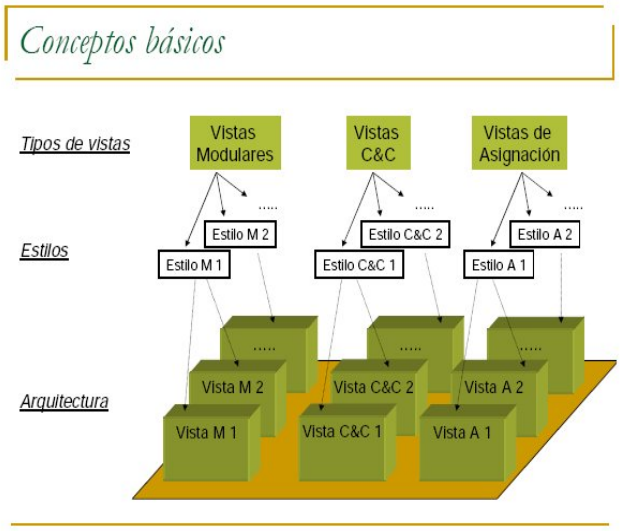

Figura 1.1 Conceptos básicos de la Arquitectura de Software

- Tipos de vistas.

Define los tipos de elementos y los tipos de relaciones usados para describir la Arquitectura del sistema desde una perspectiva particular.

- Estilo arquitectónico.

Este concepto describe y proporciona las propiedades básicas de una arquitectura. Es una especialización de tipos de elementos y tipos de relaciones, junto con un conjunto de restricciones de como pueden ser usadas. Un estilo define una familia de arquitecturas que satisfacen las restricciones. Un estilo es una especialización de un tipo de vista y refleja patrones recurrentes de interacción independientes de cualquier sistema. [Cle02]

- Vista.

Es una representación de un conjunto de elementos de sistema y las relaciones asociadas con ellos. En el contexto de tipos de vistas y estilos, una vista puede analizarse como un estilo que es limitado a un sistema particular. [Cle02].

- La arquitectura de un sistema consta de múltiples vistas, asociadas a diferentes dimensiones o estructuras del sistema.

- Las vistas se encuentran dirigidas a usuarios particulares y asociadas a requisitos no-funcionales concretos.
- Ninguna vista particular constituye la arquitectura del sistema.

1.6 Metodologías, herramientas y tecnologías.

1.6.1. Representando la arquitectura con UML.

La arquitectura, conformada por diferentes visiones del sistema, constituye un modelo de cómo está estructurado dicho sistema, sirviendo de comunicación entre las personas involucradas en el desarrollo y ayudando a realizar diversos análisis que orienten el proceso de toma de decisiones. Para que la arquitectura se convierta en una herramienta útil dentro del desarrollo y mantenimiento de los sistemas de software es necesario que se cuente con una manera precisa de representarla.

Los lenguajes desarrollados hasta el momento para representar la arquitectura presentan diferentes problemas para su utilización, por lo que se decidió utilizar un lenguaje donde estos inconvenientes pudiesen ser superados. Este lenguaje de modelamiento es UML, conocido en la industria y que además está apoyado por herramientas y metodologías de desarrollo, siendo adoptado por varias empresas como una notación estándar.

UML permite que se represente de manera semi-formal la estructura general del sistema, con la ventaja de que este mismo lenguaje puede ser usado en todas las etapas de desarrollo del sistema y su representación gráfica puede ser usada para comunicarse con los usuarios.

1.6.2. Herramientas de modelado.

Tener completa una arquitectura de software, implica obtener un proyecto de calidad. Uno de los aspectos más importantes en la arquitectura, es definir las herramientas a usar para el proyecto. Para el modelado de la arquitectura se decidió utilizar, fundamentalmente dos herramientas:

Herramienta CASE: Visual Paradigm.

Visual Paradigm utiliza UML como lenguaje de modelado. Está disponible en varias ediciones, cada una destinada a unas necesidades: Enterprise, Profesional, Community, Standard, Modeler y Personal. Es una herramienta de gran alcance, fácil de utilizar y que apoya el ciclo de vida completo del software - análisis, diseño, codificación, prueba y despliegue. Es posible dibujar diagramas UML, generar código de diagramas de la clase y viceversa, y generar la documentación. Soporta todos los diagramas de la más reciente notación de UML.

Editor de diagramas: dia.

Dia es una aplicación para la creación de diagramas técnicos. Entre sus características se incluye: soporte para distintos tipos de diagramas como diagramas entidad-relación, diagramas de red, diagramas de flujo, diagramas de estructuras estáticas de UML (Unified Modelling Language - diagramas de clase), entre otros. Permite la utilización de formas personalizadas creadas por el usuario como simples descripciones XML. Y además, brinda la posibilidad de exportar a múltiples formatos (EPS, SVG, CGM, PNG, etc.).

1.6.3. Desarrollo basado en RUP.

En este trabajo, para la selección, evaluación, representación y documentación de la Arquitectura de Software, hemos adoptado la propuesta plasmada en [Cle02], como se expuso en el Capítulo 1; sin embargo, se ha seleccionado RUP (Rational Unified Process) para guiar todo el desarrollo del software.

El Proceso Unificado de Rational (RUP, el original inglés *Rational Unified Process*) 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. [RUP].

Como todo proceso de desarrollo, su objetivo es elevar la calidad del software por todas las fases por las que pasa, a través de una mayor transparencia y control sobre el proceso. Es una forma disciplinada de asignar tareas y responsabilidades, que pretende implementar las mejores prácticas en ingeniería de Software. Es iterativo e incremental, se centra en la arquitectura y es guiado por los casos de uso. [RUP].

1.7 Servidor de aplicaciones: ZOPE.

1.7.1. Descripción general.

Zope es un servidor de aplicaciones Web escrito en el lenguaje de programación Python. Puede ser manejado casi totalmente usando una interfaz de usuario basada en páginas Web. Un sitio Web de Zope está compuesto de objetos en lugar de archivos, como es usual con la mayoría de los otros sistemas de servidores Web. Las ventajas de usar objetos en lugar de archivos son [ZTCA]:

- Combinan el comportamiento y los datos en una forma más natural que los archivos de texto plano.
- Alientan el uso de componentes estándares que se ocupan de una parte particular de las que forman una aplicación Web, permitiendo flexibilidad y buena descomposición.
- Posibilitan procesos automáticos de gestión de información.

Lo más característico de Zope es su base de datos orientada a objetos, llamada ZODB o Zope Object Database. Esta base de datos almacena objetos ordenados en un sistema similar a un sistema de ficheros, pero cada objeto tiene propiedades, métodos o puede contener a su vez otros objetos. Esta aproximación es muy diferente de las base de datos relacionales habituales. Sin embargo, Zope dispone de múltiples conectores para las diferentes bases de datos relacionales y ofrece sistemas básicos de conexión y consulta abstrayéndolos como objetos. [ZTCA].

1.7.2. Por qué usar Zope en vez de otro servidor de aplicación.

Zope puede ayudar potencialmente a crear sitios Web con menores costos y tiempos que los que se podría obtener usando a otro servidor de aplicación Web competidor. Esta reclamación es sostenida por varias características de Zope:

- Zope está libre de costos y es distribuido bajo una licencia de código abierto. Existen muchos servidores de aplicación comerciales que son relativamente caros. [Zope26].
- Zope en sí mismo es una plataforma global. Este incorpora todos los componentes necesarios

para comenzar a desarrollar una aplicación. No se tiene que licenciar software suplementario para apoyar Zope (por ejemplo: una base de datos relacional) a fin de desarrollar su aplicación. [Zope26].

- Zope permite y anima a otros desarrolladores a empaquetar y distribuir aplicaciones confeccionadas. Debido a esto, Zope tiene una amplia variedad de servicios integrados y productos de complemento disponibles para el uso inmediato. La mayor parte de estos componentes, como Zope en sí mismo, son libres y de código abierto. [Zope26].
- Las aplicaciones creadas en Zope pueden escalar casi en línea recta con la utilización del Zope Enterprise Objects (ZEO). Usando ZEO, usted puede desplegar una aplicación de Zope a través de muchas computadoras físicas sin tener que cambiar mucho de su código de aplicación. Muchos servidores de aplicación no escalan completamente de manera transparente o como era de esperar. [Zope26].
- Zope permite que desarrolladores creen aplicaciones de Web usando sólo un navegador de Web. El Internet Explorer, Mozilla, Netscape, OmniWeb, Konqueror, y navegadores de Ópera pueden ser usados para mostrar y manipular el ambiente de desarrollo de Zope (Interfaz de Administración de Zope, también conocido como el ZMI). Zope también permite que desarrolladores deleguen sin peligro deberes de desarrollo de aplicación a otros desarrolladores, utilizando el mismo interfaz a través de la Web. Muy pocos otros servidores de aplicación, si existe alguno, entregan el mismo nivel de la funcionalidad. [Zope26].
- Zope proporciona un marco de seguridad granular y extensible. Usted puede integrar fácilmente Zope con diversos sistemas de autenticación y autorización como LDAP, Windows NT, y RADIO simultáneamente, usando módulos preconstruidos. [Zope26].
- Zope permite que equipos de desarrolladores colaboren con eficacia. Los ambientes de colaboración requieren de herramientas que permitan a los usuarios trabajar sin interferir el uno con el otro, además Zope tiene Undo, Versiones, Historia y otras herramientas para ayudar las personas a trabajar sin peligro juntos, permitiendo la recuperación de errores.[Zope26].
- Zope corre en las plataformas de sistemas operativos más populares: Linux, Windows NT/2000/XP, Solaris, FreeBSD, NetBSD, OpenBSD, y Mac OS X.
- Zope puede ser ampliado usando el lenguaje interpretado Python. El Python es popular y fácil para aprender, y ello promueve el desarrollo rápido. [Zope26].

1.7.3. Zope como alternativa para el desarrollo de software bibliotecario.

Para el desarrollo de un ILS capaz de responder a las demandas actuales del software de biblioteca, y que además cuente las características deseadas tanto por los profesionales de la informática como por los bibliotecarios y especialistas relacionados con la rama, se ha decido escoger el framework Zope, como plataforma de desarrollo. Además de las características que se mencionaron anteriormente, se presentan otras que influyeron y determinaron en esta decisión.

- Contiene un lenguaje de plantillas para la creación de páginas dinámicas, facilitando la creación y el mantenimiento de la vista común de su sitio Web. [BibZ02].
- Tiene una base de datos orientada a objetos incorporada, haciendo más fácil el almacenamiento de contenido, lógica y presentación en un solo lugar. [BibZ02].
- Tiene una framework de seguridad, facilitando la delegación de mantenimiento de secciones o subsecciones del sitio a otras personas. [BibZ02].
- Los sitios en Zope son accesibles también por vía FTP y WebDAV, haciendo más fácil el uso de herramientas de escritorio para la creación de sitios. [BibZ02].
- Está escrito en Python, facilitando su extensión con sus propias personalizaciones. [BibZ02].
- Tiene una herramienta de administración de contenidos incorporada como objetos “Version” y “Undo”, los cuales hacen fácil y de forma segura, la modificación y actualización de sitios Web. [BibZ02].
- Permite la utilización de Apache como servidor Web en lugar del servidor Web proporcionado por el propio Zope. Apache es un producto con muchos años de experiencia, muy probado a todos los niveles, tanto en lo que respecta a rendimiento como seguridad.

Zope es sin duda una de las mejores plataformas para crear aplicaciones empresariales y de gestión. Su arquitectura basada en componentes le permite la construcción de aplicaciones basadas en capas, implementando las mismas a través de agrupaciones lógicas de componentes de software, ofreciendo componentes de presentación, contenido, servicios, entre otros, lo que garantiza la implementación eficiente de sistemas pequeños, medianos y complejos.

1.8 Sistemas de gestión de bases de datos.

El almacenamiento y la búsqueda y recuperación de información constituyen procesos fundamentales en los entornos bibliotecarios. Por lo que uno de los parámetros para medir la calidad de un ILS es la eficiencia con que es capaz de almacenar y recuperar la información. Es por ello que la elección de los sistemas gestores de bases de datos para una aplicación de gestión bibliotecaria, constituye un aspecto fundamental en la arquitectura de software.

1.8.1. Bases de datos ISIS.

Una base de datos ISIS consiste en filas de estructura no específica, identificada con un número único, la clave de fila. Cada fila contiene una lista de campos, con un número de campo (tag) y un valor de cadena. Dentro de una fila puede haber cero, uno o más campos con un número de campo determinado. El valor del campo generalmente es la representación textual de datos en una u otra codificación de caracteres (generalmente una de las páginas de código IBM/DOS), pero también puede contener bytes arbitrarios.

Debido a su flexibilidad, este tipo de base de datos resulta práctico para catálogos y directorios con registros altamente variables y un solo nivel de subestructura, los que actualmente se vierten en documentos XML más que en filas. El mecanismo flexible de indización combina lo mejor de búsqueda en texto completo con recuperación estructurada.

1.8.2. Bases de datos relacionales.

Una base de datos relacional es un conjunto de dos o mas tablas estructuradas en registros (líneas) y campos (columnas), que se vinculan entre si por un campo en común, en ambos casos posee las mismas características como por ejemplo el nombre de campo, tipo y longitud; a este campo generalmente de le denomina ID, identificador o clave. A esta manera de construir bases de datos se le denomina modelo relacional. [BDRE].

La mayor parte de las bases de datos que hoy se usan pertenece a la categoría de las bases de datos relacionales. Los motivos de este éxito (también comercial) hay que buscarlos en el rigor matemático y en

la potencialidad expresiva del modelo relacional en que se basan, en su facilidad de uso y, último pero no menos importante, en la disponibilidad de un lenguaje de interrogación estándar, el SQL, que, al menos potencialmente, permite que se desarrollen aplicaciones independientes del sistema gestor de base de datos concreto relacional que se use. [BDMS].

1.8.3. Integrando ISIS + RDBMS.

Si durante años los programadores de aplicaciones para bibliotecas se encontraron con programas muy pobres desde el punto de vista de la recuperación, realizadas en los paquetes comerciales estándar, también, por dar prioridad al aspecto recuperación, se llegaba a la situación inversa, en donde excelentes sistemas de recuperación, debían convivir con sistemas administrativos lentos e innecesariamente complejos.

La tendencia actual del software, y parece saludable que así sea, apunta más a la utilización de la herramienta adecuada para cada caso, y a la mayor colaboración posible entre las mismas, que a la formación de monopolios, en donde todo se realiza dentro de un único y esclavizante marco de referencia.

En la arquitectura se propone un sistema híbrido para la gestión de los datos, aprovechando las ventajas, tanto de un IRS como de un RDBMS. El mismo estará constituido por Malete, un sistema de base de datos de propósito general, el cual gestionará las bases de datos documentales, y por un RDBMS, PostgreSQL, que responderá por la gestión de los datos referentes a la administración del sistema.

1.8.4. Malete.

Malete es un sistema de base de datos de propósito general, que es muy simple a fin de ganar en rapidez, robustez y flexibilidad. Supone que el acceso a él, mayormente, se realizará mediante otro software, aunque también lo pueden hacer personas. Incluye una biblioteca que constituye el núcleo de las bases de datos, un servidor genérico y bibliotecas de acceso para varios lenguajes de programación. Malete es un marco para aplicaciones al estilo de ISIS. Está orientado a usuarios de ISIS y bibliotecarios en general. Proporciona apoyo a la conversión de una variedad de formatos de archivos conocidos como el estándar MARC, permite indexación de alto nivel, referencias (archivos de autoridades, datos cifrados), formas,

entre otras prestaciones. [MLT-O].

1.8.5. PostgreSQL.

PostgreSql es un sistema de administración de bases de datos relacional libre. Tiene más de 15 años de desarrollo activo y se ha ganado la reputación de ser confiable. Corre en la mayoría de los Sistemas Operativos más utilizados incluyendo, Linux, varias versiones de UNIX, BeOS y Windows. Cumple la prueba ACID (Atomicidad, Consistencia, Integridad, Durabilidad) y tiene soporte completo para llaves foráneas, joins, vistas, subconsultas, disparadores (triggers), y procedimientos almacenados (en varios lenguajes) Incluye la mayoría de los tipos de datos de los estándares SQL92 y SQL99. También soporta almacenamiento de objetos grandes (imágenes, sonido y video) Así, como sus propias interfaces de programación para C/C++, Java, Perl, Python, Ruby, Tcl, ODBC, entre otros, y una documentación muy completa Brinda la posibilidad de herencia de tablas y acceso concurrente multiversión (no se bloquean las tablas, ni siquiera las filas, cuando un procesador escribe). [PSQL].

1.9 Conclusiones.

En este capítulo se han abordado los temas relacionados con el objeto de estudio, el estado del arte, conceptos fundamentales, las tendencias y tecnologías actuales, las metodologías, herramientas y tecnologías a emplear como parte de la arquitectura que se propone en esta tesis; con el objetivo de profundizar en estos temas y fundamentar la necesidad de dar una solución informática, a la situación existente en la BNJM, para gestionar la información relacionada con el procesamiento de materiales bibliográficos y los servicios al público.

CAPÍTULO 2: Representando la Arquitectura

En este capítulo se documentan las vistas arquitectónicas seleccionadas para representar la arquitectura del sistema a desarrollar. A continuación se mencionan y se describen cada una de estas vistas:

Vista de Casos de Uso.

No se instancia ningún estilo, se basa en la propuesta de diagrama de casos de uso de RUP. Contiene casos de uso y el comportamiento significativo desde el punto de vista arquitectónico. Sirve como contrato entre clientes y desarrolladores, representa una entrada esencial para el análisis, diseño y pruebas.

Vista de Descomposición.

Es una instancia del estilo de descomposición del tipo de vista de módulo, el cual representa la descomposición del código en subsistemas. Es una vista jerárquica (top-down) del sistema. Es usada para dar una vista global del sistema a los stakeholders, es particularmente usada para educación y comunicación a nivel de managers, y usualmente la base para asignar trabajo.

Vista de Capas.

Es una instancia del estilo de capas del tipo de vista de módulo. Refleja la división del software en unidades, las unidades son capas, cada una de las cuales representa una máquina virtual. Hay restricciones entre las capas en la forma de relacionarse. Permite reflejar las propiedades de modificabilidad y portabilidad.

Vista de Usos.

Es una instancia del estilo de usos del tipo de vista de módulo. Se puede usar esta vista para restringir la implementación de la arquitectura. Dice a los desarrolladores que otros módulos tienen que existir para que la porción del sistema que les corresponde desarrollar trabaje correctamente. Es útil para planear desarrollo incremental, debugging y testing. Indica los efectos de un cambio específico.

Vista de Comunicación de Procesos.

Para representar esta vista se instancia el estilo Comunicación de procesos del tipo de vista Componentes & conectores. Este estilo se caracteriza por la interacción de componentes ejecutándose concurrentemente a través de varios mecanismos conectores. Proporciona las bases para el entendimiento de la organización de procesos del sistema, ilustrando la descomposición de un sistema en procesos e hilos, y puede mostrar la interacción entre los procesos. Es común en muchos sistemas y necesario en todos los sistemas distribuidos, en muchos sistemas es apropiado para entender cualquier comportamiento asociado a la concurrencia.

Vista de Implementación.

Para representar esta vista se instancia el estilo de implementación del tipo de vista de asignación. Realiza la correspondencia de módulos del tipo de vista de módulo a una infraestructura de desarrollo. Los elementos de este estilo son módulos y entidades de configuración. Este estilo es usado para describir como los módulos son mapeados a entidades dentro del sistema de gestión de configuración , así como para administrar versiones y ramas, y para coordinar el desarrollo multi-equipo.

Vista de Despliegue.

Para representar esta vista se instancia el estilo de despliegue del tipo de vista de asignación. Contiene la descripción de varios nodos físicos para las configuraciones de plataforma más típicas. Describe una o más configuraciones de red física(hardware), en las cuales el software es desplegado y corre. Es usada para análisis de rendimiento, confiabilidad y seguridad.

Vista de Asignación de Trabajo.

Para representar esta vista se instancia el estilo de asignación de trabajo del tipo de vista de asignación. Realiza la correspondencia entre los módulos del tipo de vista de módulos con los equipos de desarrollo. Los elementos de este estilo son módulos y los equipos de desarrollo. Es usado para describir cuales equipos son responsables de cuales elementos de la estructura de trabajo, así como para informar estimados de inventario y presupuesto.

2.1 Vista general del sistema.

Se desea construir un sistema que sea capaz de automatizar todo el proceso por el cual transitan los materiales bibliográficos en la BNJM, desde su llegada a la misma, hasta su entrega al usuario final, y que además permita gestionar una serie de servicios que apoyarían la toma de decisiones por

parte de la dirección de la biblioteca.

El sistema proporcionará a la institución la mayoría de las actividades que conforman los servicios básicos tradicionales de forma automatizada, definidos a través de módulos como son: adquisición, catalogación, circulación, control de series, préstamo ínter bibliotecario, catálogos en línea (OPACs), solicitudes a través de la edición de documentos impresos, control de usuarios, gestión administrativa, comunicaciones, entre otros.

2.2 Lógica y Restricciones de diseño.

A continuación se registran los aspectos de requisitos del sistema que constituyen la motivación primaria detrás de las decisiones arquitectónicas mayores. Sin lugar a dudas, existe un gran número de requisitos específicos que tienen que ser satisfechos, pero éstos son los que tienen mayor influencia arquitectónica.

2.2.1. Características deseadas para el sistema.

Desde el punto de vista arquitectónico:

- **Rendimiento aceptable**: Refiere a la *responsiveness* del sistema. Es el tiempo requerido para responder a un estímulo (evento), o número de eventos por unidad de tiempo. Por ejemplo, una posible medida podría ser cantidad de transacciones por segundo.
- **Altamente modificable**: Es la habilidad de realizar cambios al sistema en forma rápida y a bajo costo.
- **Seguro**: Es la medida de la habilidad del sistema de resistirse al uso no autorizado y negar los servicios, mientras los provee a usuarios legítimos.
- **Altamente disponible**: Es la porción de tiempo en que el sistema esta levantado y corriendo. Se

mide como el tiempo transcurrido entre fallas, así como, cuan rápido el sistema esta apto para reanudar y quedar operativo ante una falla.

- **Portable**: Es la habilidad del sistema de correr sobre diferentes ambientes. Estos ambientes pueden ser de hardware, de software o una combinación de ambos. Portability es un caso particular de modifiability.
- **Independiente del SGBD**: Permitir utilizar cualquier gestor de base de datos (PostgreSQL, MySQL, etc.). Permitiendo migrar desde un gestor a otro sin afectar la operatividad del sistema y sin ocasionar ninguna perdida de datos.

Desde el punto de vista bibliotecario:

- Debe ***gestionar datos*** tanto **locales** como **remotos**. A la información del catálogo se le suma información proveniente de páginas Web y de bases de datos remotas.
- Toda la información debe cumplir con normas establecidas, esto incluye ***normas bibliotecológicas*** y normas del ámbito de la informática. Es preferible adherir a normas internacionales abiertas que a normativas propietarias.
- El software debe ser ***altamente adaptable*** para amoldarse a las necesidades y la organización de cada biblioteca, respetando usos y métodos locales.
- Debe tener la capacidad de ***crecer y evolucionar*** con el avance tecnológico y bibliotecológico. Esto se logra con mayor facilidad, adhiriendo a estándares abiertos.
- Debe ser independiente del ***formato de catalogación***; permitir el uso de cualquier sistema de catalogación, UNIMARC, MARC21 o cualquier versión personalizada.

2.3 Documentando las vistas.

2.3.1. Vista de Casos de Uso. Subsistema de Catalogación.

2.3.1.1 Presentación primaria.

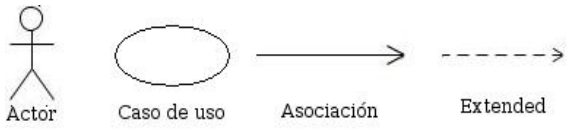

Figura 2.1 Vista de Casos de Uso.

2.3.1.2. Catálogo de elementos.

2.3.1.2.1. Elementos y sus propiedades.

- Elemento: Actores. Los actores pueden intercambiar información con el sistema.
- Propiedades:
  - Nombre: Define el nombre del actor, y su valor debe expresar un rol.
  - Responsabilidad: Define la responsabilidad que juega el actor en el sistema.
- Elemento: Casos de uso. Los casos de uso establecen un acuerdo entre clientes y desarrolladores sobre las condiciones y posibilidades (requisitos) que debe cumplir el sistema.

Caso de uso	
(1)	(2)
Propósito	(3)
Actores	
Resumen: (4)	
Referencias	(5)
Acción del actor	**Respuesta del sistema**
(6)	(7)
Flujo alternativo (8)	
Acción del actor	**Respuesta del sistema**

(9)	(10)
Puntos de extensión.	
(11)	

(1) Código del caso de uso.
(2) Nombre del caso de uso.
(3) Objetivo del caso de uso.
(4) Descripción del caso de uso.
(5) Requerimientos funcionales o casos de uso con los que se relacione el caso de uso, ya sea porque es un requerimiento que le da origen, o por ser un caso de uso que se incluye o extiende.
(6) Listado de acciones que realiza el actor.
(7) Listado de respuestas del sistema ante las acciones del actor. {En esta parte se especifica el flujo básico o curso normal de las acciones, de existir condicionantes o posibles caminos se especifican en los Flujos alternativos.}
(8) Nombre o número del flujo alternativo.
(9) Listado de acciones que realiza el actor.
(10) Listado de respuestas del sistema ante las acciones del actor.
(11) Listado de extensiones de los casos de uso. Especificar la línea del curso de acciones en que se encuentra esta extensión.

2.3.1.2.2. Relaciones y sus propiedades.

- Relaciones: Include, Extend, Generalization.
  - Include: En una forma de interacción, un caso de uso dado puede "incluir" otro. El primer caso de uso a menudo depende del resultado del caso de uso incluido. Esto es útil para extraer comportamientos verdaderamente comunes desde múltiples casos de uso a una descripción individual. La notación es una flecha rayada desde el caso de uso que lo incluye hasta el caso de uso incluido, con la etiqueta «include».
  - Extend: En otra forma de interacción, un caso de uso dado, (la extensión) puede *extender* a otro. Esta relación indica que el comportamiento del caso de uso extensión puede ser

insertado en el caso de uso extendido bajo ciertas condiciones. La notación es una flecha rayada desde el caso de uso extensión al caso de uso extendido, con la etiqueta **«**extend**»**.

- Generalization: En la tercera forma de relación entre casos de uso, existe una relación generalización/especialización. Un caso de uso dado puede estar en una forma especializada de un caso de uso existente. La notación es una línea sólida terminada en un triángulo dibujado desde el caso de uso especializado al caso de uso general.

2.3.1.2.3. Interfaces de elementos.

[Omitido]

2.3.1.2.4. Comportamiento de elementos.

No aplicable.

2.3.1.3. Un diagrama de contexto.

[Omitido]

2.3.1.4. Guía de variabilidad.

Ninguna.

2.3.1.5. Background de Arquitectura.

[Omitido]

2.3.1.6. Otra información.

[Omitido]

2.3.1.7. Paquetes de vista relacionados.

- Padre: Ninguno.
- Hijo: Ninguno.
- Hermanos:
  - Vista de Casos de Uso. Subsistema de Adquisición (no disponible en este trabajo).
  - Vista de Casos de Uso. Subsistema de Circulación (no disponible en este trabajo).
  - Vista de Casos de Uso. Subsistema de DSI (no disponible en este trabajo).
  - Vista de Casos de Uso. Subsistema de Referencias (no disponible en este trabajo).
  - Vista de Casos de Uso. Subsistema de Estadísticas (no disponible en este trabajo).

.

2.3.2. Vista de Descomposición. Sistema Integrado para Bibliotecas.

2.3.2.1. Presentación primaria.

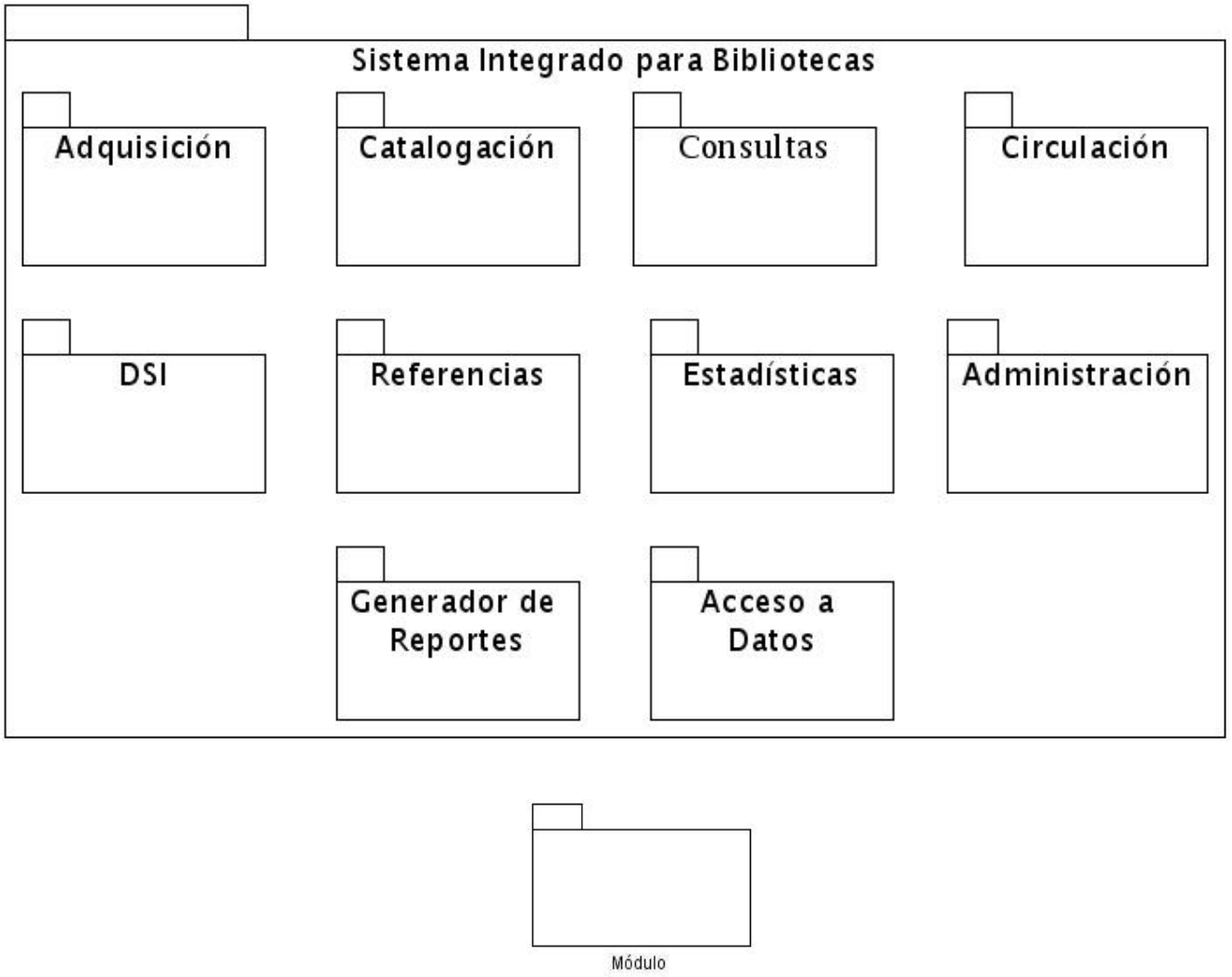

Figura 2.2 Vista de Descomposición.

2.3.2.2. Catálogo de elementos.

2.3.2.2.1. Elementos y sus propiedades.

Elementos: Módulos.

Propiedades: Nombre, Responsabilidad.

Nombre: Para referirse al módulo, debe guardar relación con la función que hace.

Responsabilidad: Es la forma de identificar el rol del módulo en el sistema completo y establecer una identidad más allá de su nombre.

Nombre de Módulo	**Responsabilidad**
Adquisición	El **módulo de Adquisición** deberá facilitar la gestión de la selección y adquisición de los nuevos ingresos de la biblioteca, así como también permitirá llevar el control del presupuesto (pedidos, proveedores, gastos, etc.). Además, permitirá hacer consultas a los catálogos.
Catalogación	El **módulo de Catalogación** es el núcleo de todo el sistema, es el que facilitará la creación de las bases de datos bibliográficos a través de las cuales se procesan (catalogar, clasificar e inscribir) los documentos y se almacenan en un sistema de gestión de bases de datos. Estos catálogos deberán cumplir con las normas internacionales, en particular con las normas ISO y formatos bibliográficos MARC. Este módulo además permitirá la gestión del Catálogo de Autoridades, así como los tesauros. Se compone de los siguientes módulos: • **Módulo de Catalogación**: encargado del procesamiento de libros y otros documentos. • **Módulo de Publicaciones Seriadas**: encargado del procesamiento de las publicaciones seriadas.
Consultas	Se compone de los siguientes módulos:

Nombre de Módulo	Responsabilidad
	• **Módulo OPAC**: permitirá el acceso en línea a los catálogos de la biblioteca, ofreciendo diferentes niveles en función del perfil de los usuarios. También contemplará las necesidades de consulta del personal bibliotecario. • **Módulo Z39.50**: permitirá realizar consultas, mediante el protocolo Z39.50, de cualquier recurso local o remoto de información bibliográfica, textual o de otro tipo, sin necesidad de abandonar el entorno habitual de trabajo. También facilitará el préstamo interbibliotecario y la catalogación cooperativa, ya que el protocolo permite la descarga de registros MARC de distintas fuentes. Este módulo además permitirá que los usuarios puedan acceder a catálogos de todo el mundo con la misma interfaz de consulta.
Circulación	El **módulo de Circulación** facilitará y agilizará las operaciones relativas al servicio de préstamo, realizando los controles (uso y circulación de documentos) adecuados para el buen funcionamiento del mismo. Permitirá la creación y modificación de una base de datos sobre los usuarios, la definición de los servicios bibliotecarios, el préstamo en diversas modalidades y la edición de productos relativos a la circulación.
DSI	El **módulo de DSI** (Diseminación Selectiva de Información) permitirá la gestión de perfiles de usuarios (datos de interés), instrumentos de avisos y diseminación de informaciones conforme al perfil del usuario, además permitirá la gestión de listas de discusión para usuarios, bibliotecarios y funcionarios de la biblioteca.
Referencias	El **módulo de Referencia** permitirá gestionar todo lo relacionado con el levantamiento bibliográfico, además de brindar a los usuarios servicios de orientación, localización e identificación de información en libros, revistas, bases de datos y recursos de multimedia que existen en la Biblioteca.

Nombre de Módulo	Responsabilidad
	Los usuarios podrán solicitar y/o recibir servicios de referencia a través de diferentes vías: • A través del propio sistema, las solicitudes se almacenarían en una base de datos, de manera tal que los especialistas atendieran dichas solicitudes, y los usuarios recibirían la información solicitada a través de email, chat, u otra vía. • A través del chat, si los especialistas se encuentran en línea, los usuarios podrán interactuar directamente con el personal especializado. El bibliotecario será responsable de registrar los datos correspondientes a cada solicitud realizada por esta vía. • A través de email, de igual forma, será el bibliotecario el responsable de registrar la información correspondiente.
Estadísticas	El **módulo de Estadísticas** permitirá analizar algunos datos asociados a los diferentes servicios, mostrando reportes (textos, tablas y gráficos) sobre las estadísticas de dichos servicios; lo que facilitará el proceso de toma de decisiones por parte de la directiva de la institución. Por ejemplo, las estadísticas del módulo de Circulación podrán brindar cuál es la temática más solicitada, que libros están en circulación, mostrar el comportamiento de los usuarios, entre otros aspectos.
Administración	Se compone de los siguientes módulos: • **Módulo de Gestión de Usuarios**: permitirá gestionar usuarios (individualmente o por grupos), dando la posibilidad de configurar los permisos de acceso a determinadas áreas y/o módulos del sistema. Esto incluye la siguiente funcionalidad: altas, bajas, modificar, buscador, motor de envío de emails automático a todos los usuarios, etc.). • **Módulo de Configuración del Sistema:** permitirá configurar los

Nombre de Módulo	Responsabilidad
	parámetros con los que trabaja el sistema, así como la configuración y/o personalización de cada uno de sus módulos.
Generador de Reportes	El **módulo Generador de Reportes** permitirá a los usuarios construir sus propios informes y estructurar sus formatos. Debe contar con soporte para bases de datos ISIS, además de bases de datos relacionales (PostgreSQL, aunque se piensa que en posteriores versiones incorpore otros servidores de bases de datos como MySQL, etc.). Debe contener estructuras básicas para el diseño de informes, como por ejemplo: encabezado de página, pie de página encabezado de reporte, sección detalles, etc. Además, debe permitir la definición de componentes como: campos relacionados a bases de datos (ISIS-PostgreSQL), inserción de líneas, gráficos, entre otros.
Acceso a Datos	El **módulo de Acceso a Datos** abarcará todo lo relacionado con el acceso a las fuentes de datos. A través de este módulo, los restantes módulos del sistema podrán acceder y/o modificar, de manera transparente, la información que radica en las bases de datos (ISIS-PostgreSQL), conteniendo los componentes necesarios para establecer dicha comunicación, separando la lógica del acceso a los datos.

2.3.2.2.2. Relaciones y sus propiedades.

Relaciones: Relación de descomposición, una forma refinada de la relación *Es-parte-de*. Cada módulo es parte del sistema. Un módulo no puede ser parte de más de un módulo.

Propiedades de las relaciones: Visibilidad, alcance para el cual la existencia del módulo es conocida. En esta vista, cada uno de los módulos es visible para los demás, es decir, cada uno puede ser usado por los otros.

2.3.2.2.3. Interfaces de elementos.

Las interfaces de los elementos se han dejado para descomposiciones subsecuentes (en etapas posteriores se representarán versiones más detalladas de esta vista).

2.3.2.2.4. Comportamiento de elementos.

No aplicable.

2.3.2.3. Un diagrama de contexto.

[Omitido]

2.3.2.4. Guía de variabilidad.

Ninguna.

2.3.2.5. Background de Arquitectura.

2.3.2.5.1. Lógica.

Para automatizar todas las tareas y procesos que se realizan en una institución es necesario realizar un estudio previo del ciclo de vida de la información. Varias han sido las definiciones dadas sobre este concepto, pero existen elementos comunes en estas definiciones y es que todos los autores coinciden en que el ciclo de vida de la información está constituido por: entradas, procesos y salidas. [BiUCI].

Las etapas del ciclo de vida de la información (CVI), no se pueden ver como elementos aislados, están estrechamente relacionadas entre sí, y cada uno de los procesos depende del que lo precede, de ahí el por qué constituyen un sistema cíclico. En cada uno de los elementos que conforman este ciclo, se definen un conjunto de tareas que deben realizar los bibliotecarios para lograr el total cumplimiento de cada etapa. Para analizar las actividades de cada fase del ciclo, se tomó como patrón la definición que da

la Escuela Soviética de Bibliotecología [Mijailov], la cual se muestra en el siguiente esquema.

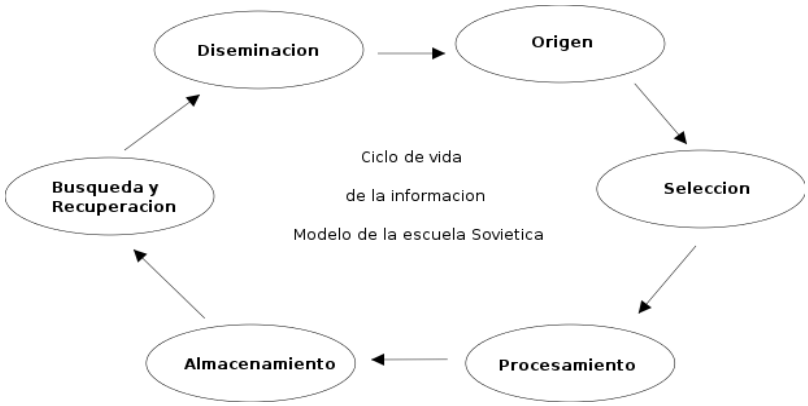

Figura. 2.3 Ciclo de vida de información.

Origen de la información: corresponde a la creación de la información como resultado de la práctica social. Incluye a los creadores intelectuales de la información (científicos, técnicos, etc.) y a los productores materiales de esta (editores, impresores, etc.). [BiUCI].

Selección: consiste en reunir exhaustivamente toda la información requerida y mantenerla actualizada, se incluyen las tareas de selección, adquisición y descarte. [BiUCI].

Procesamiento: corresponde al análisis del documento en sí, desde los puntos de vista de la forma, contenido, y la extracción de los datos registrados en el documento. Se realizan actividades como: catalogación, redacción de resúmenes y de información reseñada, clasificación, asignación del código de domicilio, integración del registro de documentos y tareas asociadas (información señal, información referativa, reseñas, etc.). [BiUCI].

Almacenamiento: conservación de los datos recolectados mediante la utilización de medios portadores que ofrezcan un largo período de almacenamiento y que puedan disponerse en un sistema ordenado, que permita la rápida y fácil recuperación de los datos relevantes. Se ejecutan las labores de organización de registros de documentos (fondo activo), manipulación física de los documentos (preparación para la

circulación, traslado a diferentes portadores materiales: por ejemplo, microfotográficos o magnéticos), la organización del fondo pasivo de búsqueda, encuadernación, conservación, restauración. [BiUCI].

Búsqueda y Recuperación: son las operaciones lógicas que garantizan la localización en una colección de datos de todos aquellos - y solamente aquellos - que proporcionan una respuesta directa a la pregunta formulada por cualquier usuario. Esta etapa está conformada por tareas como el análisis de la solicitud del usuario, elaboración de la prescripción de búsqueda, localización de modelos de búsqueda relevantes de acuerdo con la prescripción, localización de documentos y localización de datos. Se asocian otras tareas de reproducción de documentos, préstamo de documentos, servicios bibliográficos, consulta y referencia, y traducciones. [BiUCI].

Diseminación: consiste en hacer llegar a los usuarios en el período de tiempo lo más corto posible, la información sobre los documentos o los datos que ingresan en el sistema, que pueden ser de su interés. Puede originarse a partir de los resultados de procesos anteriores. Forman parte de la diseminación la información señal, los servicios referativos y de información reseñada y las bibliografías. [BiUCI].

Es tomando como base el ciclo de vida de la información, que hemos definido la estructura del sistema a desarrollar, obteniendo como resultado los módulos presentados en la sección "*Presentación primaria*".

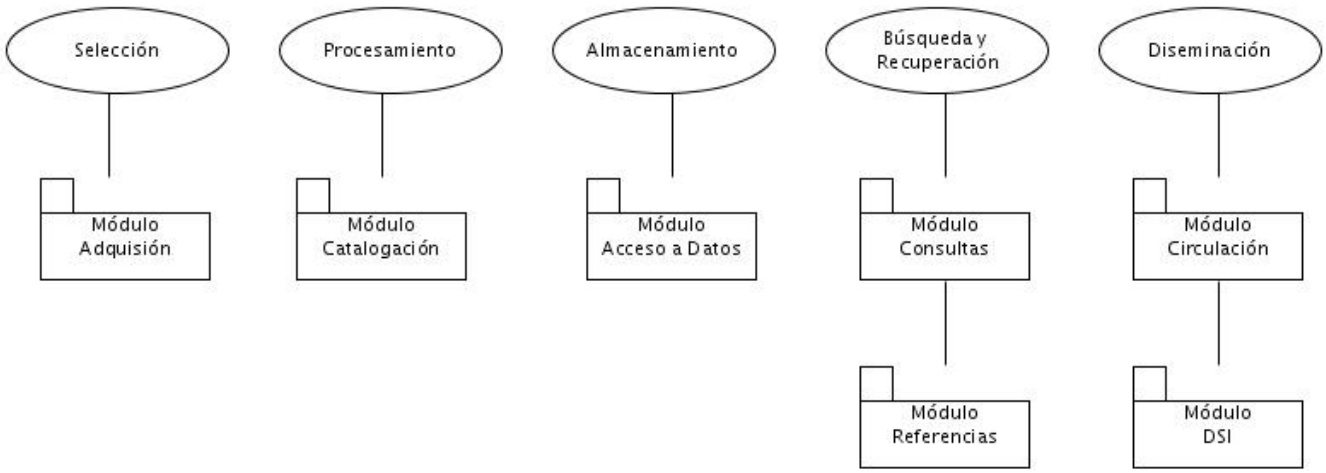

Figura 2.4 Correspondencia entre el ciclo de vida de la información y los módulos del sistema.

Además se proponen los módulos: Administración, Estadísticas y Generador de Reportes, como módulos complementarios que aportarían un valor añadido al producto.

Figura 2.5 Otros módulos del sistema.

La mayoría de los sistemas informáticos, tienen en común que todos gestionan, de una manera u otra, información asociada a los usuarios y datos propios de la configuración del sistema; es decir, que independientemente de los servicios y funcionalidad que provea un sistema, es necesario controlar el acceso y privilegios de los usuarios, modificar parámetros del sistema, entre otras tareas, es por ello que se decidió agrupar en el módulo de Administración todo lo concerniente a la gestión administrativa del sistema.

La provisión de servicios de información eficientes a un costo adecuado requiere, entre otras cosas, una producción estadística relevante en la cual los gestores puedan basar su política de toma de decisiones. Las estadísticas más valiosas son las relativas a la circulación, efectividad del sistema y aspectos relacionados con la búsqueda de información, entre otras. Los cálculos estadísticos se nutren de los datos que manipulan otros módulos del sistema, es decir, las estadísticas no interfieren en la lógica operacional del sistema, es por ello que se propone separar las estadísticas en un módulo aparte, el módulo de Estadísticas.

De manera similar al módulo de Administración, es muy común también encontrar en varios sistemas informáticos, un módulo que gestione los informes. Esta separación se debe a que los reportes son generados a partir de la información que se encuentra en las bases de datos, por lo que este módulo, aunque está muy relacionado con las demás partes del sistema, no depende de ningún otro módulo para su correcto funcionamiento. Otra de las razones por la cual se propone el módulo Generador de Reportes, es para dotar al producto de una mayor flexibilidad, ya que las salidas constituyen una parte muy importante en los sistemas informáticos, y en la mayoría de los casos, estas aplicaciones no dan la

posibilidad de que el usuario defina sus propias salidas, con este módulo el usuario podrá crear sus propios reportes según sus necesidades y requerimientos.

2.3.2.5.2. Resultados de análisis.

Ninguno.

2.3.2.5.3. Asumpciones.

- Para versiones futuras del sistema se incluirán nuevos módulos.

2.3.2.6. Otra información.

[Omitido]

2.3.2.7. Paquetes de vista relacionados.

- Padre: Ninguno.
- Hijo: Los submódulos de cada subsistema mostrado en la presentación primaria serán dados en descomposiciones subsecuentes (en etapas posteriores se representarán versiones más detalladas de esta vista).
- Hermanos: Ninguno en esta vista. Los paquetes de vista de otras vistas que expresan el mismo alcance (ámbito), es decir, que incluyen el sistema entero, se encuentran en:

    - Vista de Capas. Sistema Integrado para Bibliotecas (Sección 2.3.3).
    - Vista de Usos. Sistema Integrado para Bibliotecas (Sección 2.3.4).
    - Vista de Implementación. Sistema Integrado para Bibliotecas (Sección 2.3.6).
    - Vista de Despliegue. Sistema Integrado para Bibliotecas (Sección 2.3.7).
    - Vista de Asignación de Trabajo. Sistema Integrado para Bibliotecas (Sección 2.3.8).

2.3.3. Vista de Capas. Sistema Integrado para Bibliotecas.

2.3.3.1. Presentación primaria.

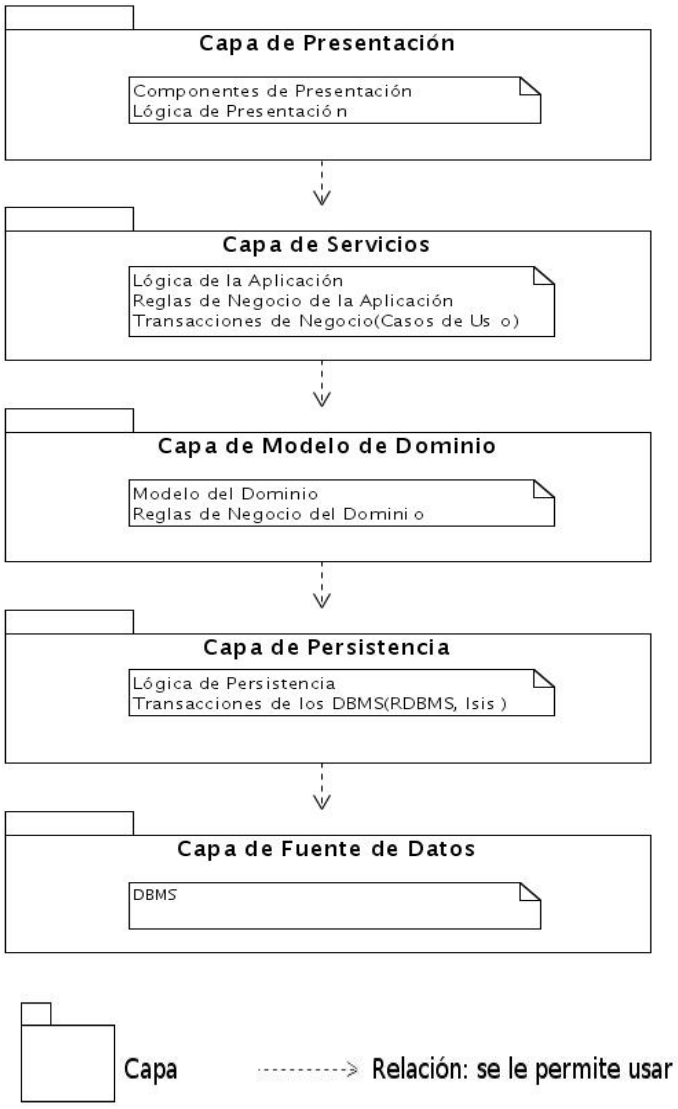

Figura 2.6 Vista de Capas.

2.3.3.2. Catálogo de elementos.

2.3.3.2.1. Elementos y sus propiedades.

- Elementos: Capas.
- Propiedades: Nombre, Responsabilidad, Contenido.
  - Nombre: Nombre de la capa.
  - Responsabilidad: Responsabilidad de la capa.
  - Contenido: Lista las unidades de software contenidas para la capa.

Nombre	**Responsabilidad**	**Contenido**
Capa de Presentación	La **capa de presentación** maneja la interacción entre el usuario y la aplicación, en ambas direcciones. El sistema estará soportado sobre un ambiente Web, donde la capa de presentación no sólo tiene que crear documentos entendibles por los usuarios, sino manejar los mensajes enviados por el browser como querystrings (cadenas de consulta) o datos de formularios.	Contiene la implementación de las interfaces de usuario de cada uno de los módulos que conforman el sistema. La capa de presentación solo busca obtener las funcionalidades o servicios que le permitan resolver la problemática de la aplicación y exponer de forma amigable y eficiente interfaces al usuario para la recolección y visualización de la información vinculada a dichos servicios.
Capa de Servicios	La **capa de servicios** es responsable de exponer los servicios necesarios en la aplicación hacia la capa de presentación. Por lo que esta capa obliga a definir una interfaz con la funcionalidad o	Contiene la implementación de los módulos que conforman el sistema, implementando los casos de uso que dan respuesta a los requisitos funcionales del sistema. En cada servicio expuesto, hará uso de los

Nombre	Responsabilidad	Contenido
	servicios que la capa de presentación necesita para cumplir con su objetivo y en cada servicio expuesto, hará uso de los objetos de la capa de dominio para la resolución del mismo.	objetos de la capa de dominio para implementar la funcionalidad necesaria. La lógica de la aplicación se resuelve completamente en esta capa.
Capa de Modelo del Dominio	La **capa de modelo del dominio** representa conceptos de negocio como reglas o estados. Lo que distingue a esta lógica por sobre el resto de la aplicación es que mantiene los conceptos centrales del proceso. Brinda la funcionalidad necesaria para que la capa de servicios implemente los servicios necesarios.	Contiene la implementación del modelo de dominio, utilizando una jerarquía de clases (propiedades y comportamiento) que implementen los conceptos fundamentales del negocio, tales como: libro, catálogo, etc.
Capa de Persistencia	La **capa de persistencia** busca que el modelo de dominio no conozca la manera en que sus datos son persistidos o almacenados en la capa de datos, ya que éste es un problema tecnológico que no tiene nada que ver con los problemas del dominio a resolver. El objetivo de la capa de persistencia es quitar del dominio la problemática asociada a este problema tecnológico que no tiene nada que ver con nuestro	Contiene dos módulos fundamentales, que se encargarán de implementar el acceso a datos, tanto a bases de datos relacionales como bibliográficas, respectivamente. Estos módulos contienen las clases necesarias para llevar a cabo la persistencia de los datos.

Nombre	Responsabilidad	Contenido
	dominio y agruparlo en esta capa. La implementación de esta capa permitirá en caso de ser necesario, el cambio de los sistemas de gestión de bases de datos sin que esto ocasione problemas a las capas superiores.	
Capa de Fuente de Datos	La **capa de fuente de datos** brinda servicios para sincronizar la capa de persistencia con un medio de almacenamiento.	En ella se encuentran las bases de datos relacionales y de registros bibliográficos, las cuales serán gestionadas a través de un sistema gestor de bases de datos relacionales y un sistema gestor de registros bibliográficos (ISIS), respectivamente.

2.3.3.2.2. Relaciones y sus propiedades.

- Relaciones: Relación *Se le permite usar.*
- Propiedades: Cada capa dependerá solamente de los elementos contenidos en ella o en las capas situadas por debajo, teniendo responsabilidades bien definidas y evitando cualquier tipo de acoplamiento con las capas superiores.

2.3.3.2.3. Interfaces de elementos.

[Omitido]

2.3.3.2.4. Comportamiento de elementos.

No aplicable.

2.3.3.3. Un diagrama de contexto.

El contexto para esta vista está establecido por el diagrama de contexto de la *Vista de Descomposición* (*Sección 2.3.2*).

2.3.3.4. Guía de variabilidad.

Ninguna.

2.3.3.5. Background de Arquitectura.

2.3.3.5.1. Lógica.

Una aplicación empresarial o también llamada software de negocio, es aquella en la cual se modela un proceso de negocios de mediana a gran escala. Esto implica en primer término un importante volumen de datos a ser persistido y accedido concurrentemente, y por otro lado una lógica basada en reglas de negocio, altamente cambiante y dependiente del dominio.

Los datos que maneja son por lo general un patrimonio muy importante de la empresa y por lo tanto la aplicación tiene como requisitos básicos garantizar su seguridad y consistencia. En algunas organizaciones esta información es consultada masivamente, por lo que otro potencial requisito es que el sistema permita un alto nivel de concurrencia en sus operaciones.

En una aplicación empresarial la lógica consiste principalmente de aspectos legales, reglas de negocio y comportamientos clásicos de sistemas que alguna vez fueron manuales. Esta lógica se la suele conocer como irregular y es propensa a continuos cambios altamente dependientes de condiciones de mercado y política.

Partiendo del criterio de que el sistema a desarrollar cumple con todas las características de una aplicación empresarial, algunas de las decisiones de diseño tomadas en el diseño de la arquitectura, son fundamentadas en modelos de desarrollo de aplicaciones empresariales.

Para organizar una aplicación empresarial, la industria del software ha convergido en una arquitectura basada en capas, dividiendo el sistema en tres capas básicas: la ***capa de presentación***, la ***capa de lógica de dominio*** y la ***capa de persistencia***. El principio detrás de esta arquitectura es que cada capa dependa sólo de los elementos contenidos en ella o en las capas situadas por debajo, teniendo responsabilidades bien definidas y evitando cualquier tipo de acoplamiento con las capas superiores [Domain – Driven Design].

Criterios de diseño.

Dado el tipo de aplicaciones seleccionado, es decir del tipo Enterprise, donde las aplicaciones poseen un dominio complejo, con lógica de negocio compleja y muchas reglas de negocio, las cuales varían con el tiempo, y van modificando a las actuales, y nutriéndose con otras nuevas, la idea central es modelar el dominio utilizando programación orientada a objetos, obteniendo así, un modelo del dominio, formado por objetos muy similares a la realidad, que se rigen según las reglas de negocio.

Para poder acompañar los cambios del negocio, actualizando así el modelo del dominio, se buscó la manera de mantener este dominio lo más aislado posible del resto de la aplicación, es decir se buscó desacoplar lo más posible al modelo de dominio del resto de la aplicación.

Para ello la arquitectura elegida es una arquitectura basada en capas lógicas (Layer Pattern), donde una de estas capas es la capa de modelo del dominio (Domain Model Layer), y ésta es la capa que buscamos que tenga el menor acoplamiento posible. Entonces partiendo de una arquitectura cliente servidor, el primer paso es quitar toda la lógica de negocio de la capa de presentación, y volcarla en la capa de modelo del dominio. Separando así muy bien todo lo que tiene que ver con obtención de información y presentación al usuario, de la lógica del dominio modelado.

Dividiendo las capas.

En una aplicación, vamos a encontrar lógica y reglas de negocio del dominio modelado, y lógica y reglas de negocio de la aplicación particular en sí, de acuerdo a como ésta hace uso del dominio. Se busca que la lógica del dominio quede en la capa de dominio, pero no la lógica de aplicación, ya que ésta no es parte del dominio sino de la aplicación que hace uso de él.

En este esquema de tres capas, la lógica de la aplicación o bien queda en la capa de presentación o queda en la capa de dominio. Sin embargo, no es parte de ninguna de las dos realmente. La idea es que el modelo de dominio, modele el dominio en general, posea las reglas inherentes a este dominio y pueda ser reutilizado en distintas aplicaciones. Cada aplicación puede tener sus propias transacciones de negocio que hacen uso del dominio de una manera particular. Si la aplicación tuviera diferentes tipos de clientes de presentación y si ellos albergaran la lógica de aplicación, ésta estaría distribuida en cada capa cliente, dificultando bastante su mantenimiento. Por todo esto se justifica el uso una capa de servicio sobre el modelo de negocio, que juega el papel de fachada. Es decir la capa de servicio se encarga de exponer los servicios necesarios en la aplicación hacia la capa de presentación. La capa de presentación solo busca obtener las funcionalidades o servicios que le permitan resolver la problemática de la aplicación y exponer de forma amigable y eficiente interfaces al usuario para la recolección y visualización de la información vinculada a dichos servicios. Por lo que esta capa obliga a definir una interfaz con la funcionalidad o servicios que la capa de presentación necesita para cumplir con su objetivo.

Esta fachada conoce al modelo y en cada servicio expuesto, hará uso de los objetos del dominio para la resolución del mismo. Si la capa de presentación corre en otro espacio físico que la capa de negocio, esta también juega el papel de interfaz remota, minimizando el tráfico entre ambas capas ya que la lógica de la aplicación se resuelve completamente en la capa de servicio.

Aún vemos que la capa del modelo del dominio sigue teniendo cierto acople con la capa de datos. Queremos evitar este conocimiento desde el modelo a la capa de datos, es decir lo que ahora buscamos es que el modelo, no conozca la manera en que sus datos son persistidos o almacenados, en la capa de datos, ya que éste es un problema tecnológico que no tiene nada que ver con los problemas del dominio a resolver, lo que nos lleva a introducir una nueva capa entre ambas, ésta capa es la capa de persistencia. El objetivo de la capa de persistencia es quitar del dominio la problemática asociada a este problema tecnológico que no tiene nada que ver con nuestro dominio y agruparlo en esta nueva capa. De esta manera se justifican cada una de las capas que se que se proponen en la arquitectura.

2.3.3.5.2. Resultados de análisis.

Ninguno.

2.3.3.5.3. Asumpciones.

Ninguna.

2.3.3.6. Otra información.

[Omitido]

2.3.3.7. Paquetes de vista relacionados.

- Padre: Ninguno.
- Hijo: Ninguno.
- Hermanos: Ninguno en esta vista. Los paquetes de vista de otras vistas que expresan el mismo alcance (ámbito), es decir, que incluyen el sistema entero, se encuentran en:

    - Vista de Descomposición. Sistema Integrado para Bibliotecas (Sección 2.3.2).
    - Vista de Capas. Sistema Integrado para Bibliotecas (Sección 2.3.3).
    - Vista de Implementación. Sistema Integrado para Bibliotecas (Sección 2.3.6).
    - Vista de Despliegue. Sistema Integrado para Bibliotecas (Sección 2.3.7).
    - Vista de Asignación de Trabajo. Sistema Integrado para Bibliotecas (Sección 2.3.8).

2.3.4. Vista de Usos. Sistema Integrado para Bibliotecas.

2.3.4.1. Presentación primaria.

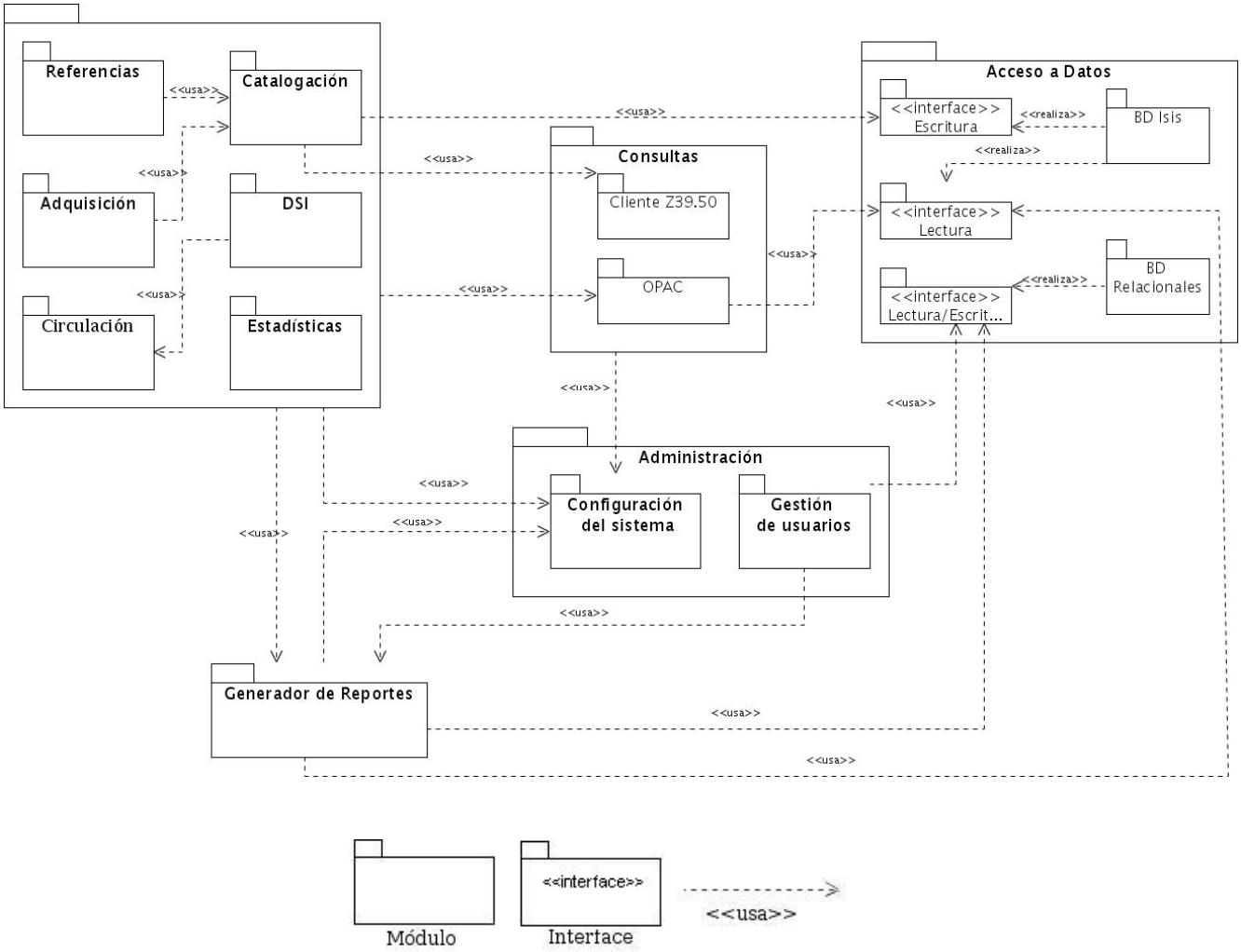

Figura 2.7 Vista de Usos.

Nota: En la figura anterior la relación “<<usa>>” tiene varios significados, según los módulos que intervengan. En versiones posteriores se explicará los tipos de relación *de uso* presentados, detallando en que consiste cada una de las relaciones.

2.3.4.2. Catálogo de elementos.

2.3.4.2.1. Elementos y sus propiedades.

- Elementos: Módulos, como se define en tipos de vistas de módulos.
- Propiedades: Nombre, Responsabilidad.
  - Nombre: Para referirse al módulo, debe guardar relación con la función que hace.
  - Responsabilidad: Es la forma de identificar el rol del módulo en el sistema completo y establecer una identidad más allá de su nombre.

Los elementos representados en la *Presentación primaria* y que no se describen a continuación, están descritos en la Vista de Descomposición (Sección 2.3.2).

Nombre del elemento	**Responsabilidad**
BD Isis	Es responsable de la funcionalidad que permitirá realizar búsqueda, recuperación y modificación en las bases de datos ISIS.
BD Relacionales	Es responsable de la funcionalidad que permitirá realizar búsqueda, recuperación y modificación en las bases de datos relacionales.

2.3.4.2.2. Relaciones y sus propiedades.

- Relaciones: La relación “usa”, la cual es una forma refinada de la relación “depende-de”. El módulo A usa el módulo B, si A depende de la presencia de un correcto funcionamiento de B para

satisfacer sus propios requerimientos.

- Propiedades: La relación “usa” puede tener una propiedad que describa en más detalle que tipos de uso puede hacer un módulo de los otros.

2.3.4.2.3. Interfaces de elementos.

Nombre del elemento	Interface	Servicios que presta
BD Isis	Lectura	Permitirá que el módulo preste servicios de acceso de lectura a las bases de datos ISIS.
	Escritura	Permitirá que el módulo preste servicios de acceso de escritura (modificación) a las bases de datos ISIS.
BD Relacionales	Lectura/Escritura	Brindará servicios de lectura/escritura a las bases de datos relacionales.
Z39.50	Interface	Expone la funcionalidad necesaria para hacer las búsquedas en catálogos locales o remotos a través del protocolo Z39.50.
OPAC	Interface	Expone la funcionalidad necesaria para hacer las búsquedas en catálogos de la biblioteca.

Las interfaces para el resto de los elementos mostrados en esta vista se especifican en la *Vista de Descomposición* (Sección 2.3.2).

2.3.4.2.3. Comportamiento de elementos.

No aplicable.

2.3.4.3. Un diagrama de contexto.

El contexto para esta vista está establecido por el diagrama de contexto de la *Vista de Descomposición* (Sección 2.3.2).

2.3.4.4. Guía de variabilidad.

Ninguna.

2.3.4.5. Background de Arquitectura.

2.3.4.5.1. Lógica.

Para una mejor comprensión de la *Presentación primaria*, se han agrupado varios módulos en otros que aparecen sin nombre, los cuales se usan solo con propósitos de agrupamiento, de tal forma que se puedan representar más fácilmente las relaciones de dependencia entre los módulos.

Los módulos que realizan accesos a bases de datos relacionales necesitan hacer accesos de lectura y escritura, por lo que el módulo BD Relacionales brinda estos servicios a través de una sola interfaz.

La mayor parte de los módulos realiza operaciones de solo lectura sobre las bases de datos Isis, solo el módulo de Catalogación podrá escribir en las mismas, por lo cual, para una mejor separación de los intereses en cuanto al manejo de los datos, el módulo BD Isis brinda dos interfaces, con accesos de lectura y escritura, esto permite además una mayor seguridad en cuanto al manejo de los datos.

2.3.4.5.2. Resultados de análisis.

Ninguno.

2.3.4.5.3. Asumpciones.

Los módulos son configurables y para ello hacen uso del módulo Administración, el cual gestiona los archivos de configuración pertenecientes a dichos módulos. Esto puede hacerse usando archivos XML (eXtensible Markup Language).

2.3.4.6. Otra información.

Hay presentes dos formas de uso entre los módulos, a través de interfaces, las cuales especifican la funcionalidad que el módulo expone y que puede ser invocada por otro; además se incluye la forma de uso en la que un módulo modifica recursos que son compartidos con otros, por ejemplo: Administración, el cual modifica los archivos de configuración pertenecientes a otros módulos.

2.3.4.7. Paquetes de vista relacionados.

- Padre: Ninguno.
- Hijo: Ninguno.
- Hermanos: Ninguno en esta vista. Los paquetes de vista de otras vistas que expresan el mismo alcance (ámbito), es decir, que incluyen el sistema entero, se encuentran en:

    - Vista de Descomposición. Sistema Integrado para Bibliotecas (Sección 2.3.2).
    - Vista de Capas. Sistema Integrado para Bibliotecas (Sección 2.3.3).
    - Vista de Implementación. Sistema Integrado para Bibliotecas (Sección 2.3.6).
    - Vista de Despliegue. Sistema Integrado para Bibliotecas (Sección 2.3.7)
    - Vista de Asignación de Trabajo. Sistema Integrado para Bibliotecas (Sección 2.3.8).

2.3.5. Vista de Comunicación de Procesos. Subsistema de Catalogación.

2.3.5.1. Presentación primaria.

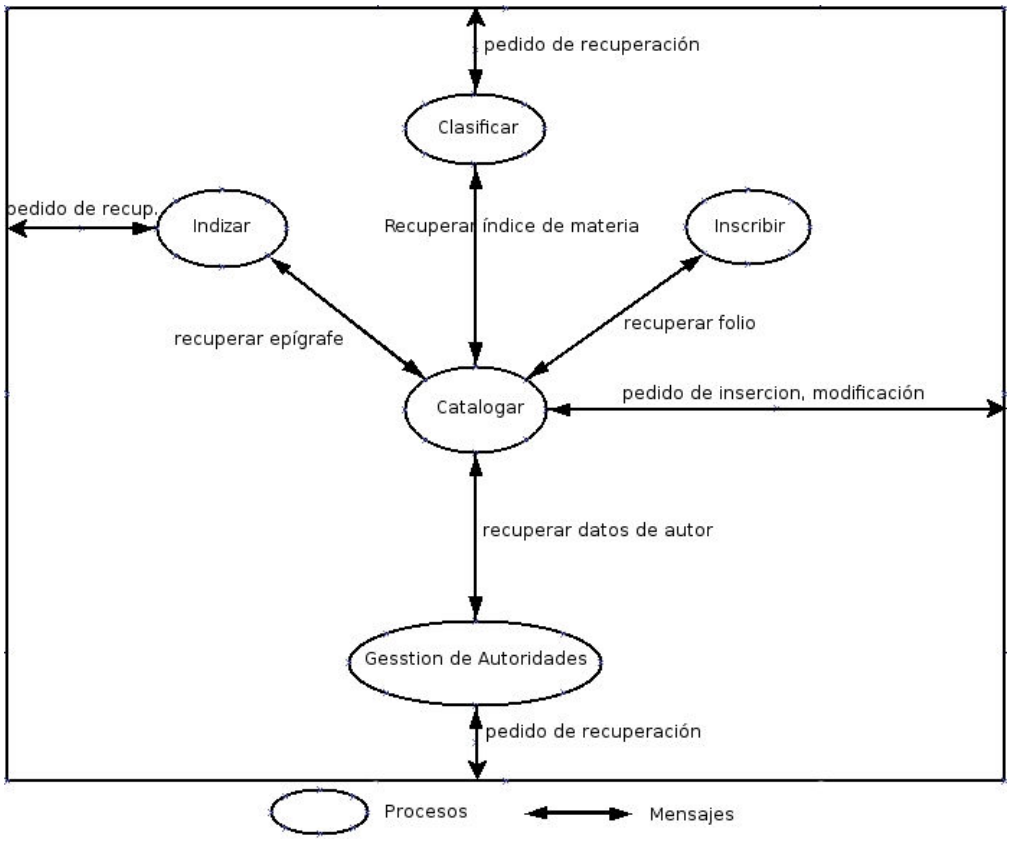

Figura 2.8 Vista de Comunicación de Procesos.

2.3.5.2. Catálogo de elementos.

2.3.5.2.1. Elementos y sus propiedades.

- Elementos: Procesos, Mensajes.
- Propiedades: Nombre, Tipo, Descripción.
  - Nombre: Nombre del elemento.
  - Tipo: Se refiere al tipo de elemento.
  - Descripción: Breve descripción sobre el rol que juega el elemento dentro del diagrama.

Nombre de elemento	Tipo	Descripción
Catalogar	Proceso	Permite a través de una interfaz de usuario creación y modificación de registros bibliográficos, los cuales contienen los datos de los documentos. Para recolectar la información necesaria se apoya en los procesos Indizar, Clasificar, Inscribir y Gestión de autoridades.
Indicar	Proceso	Permite realizar búsquedas en la base de datos que contiene el epigrafiario.
Clasificar	Proceso	Permite realizar búsquedas en la base de datos que contiene las materias con sus índices.
Inscribir	Proceso	Permite leer y actualizar el folio

Nombre de elemento	Tipo	Descripción
		disponible para los nuevos materiales que se van a catalogar. El folio se almacena en un archivos
Gestión de autoridades	Proceso	Permite recuperar, crear, modificar y eliminar datos de los autores del catálogo de autoridades.
Mensajes	Llamada(Call)	Conector para el intercambio de mensajes entre objetos. Permite hacer pedidos de inserción, modificación y recuperación.

2.3.5.2.2. Relaciones y sus propiedades.

Adjunto: muestra como los componentes y conectores están adjuntados unos a otros.

2.3.5.2.3. Interfaces de elementos.

Las interfaces de los elementos mostrados en esta vista se especifican en la *Vista de Descomposición* (Sección 2.3.2).

2.3.5.2.4. Comportamiento de elementos.

Comportamiento de elementos.

2.3.5.3. Un diagrama de contexto.

[Omitido]

2.3.5.4. Guía de variabilidad.

Ninguna.

2.3.5.5. Background de Arquitectura.

2.3.5.5.1. Lógica.

Para la representación de esta vista se ha utilizado el editor de diagramas Dia. Es un diagrama informal, para el cual no se ha utilizado UML. El cuadro dentro del cual están los elementos de la vista representa la frontera del sistema.

Los procesos representados en la presentación primaria se ejecutarán en un mismo servidor Zope, los mensajes que envía el proceso Catalogar a los demás, representan llamadas a métodos de objetos (que no se han definido aún) que estarán contenidos en los mismos, estos envían mensajes de retorno con los datos solicitados.

Los mensajes que se envían fuera de la frontera, representan peticiones al servidor de bases de datos ISIS (Malete), el cual está basado en envío de mensajes, los cuales están representados como registros; la única interface al servidor puede ser considerada como un simple función “enviar”, la cual toma un registro como parámetro y retorna un registro, este registro resultante es un mensaje válido. [MLT-P]. El envío de estos mensajes hacia el servidor, así como la respuesta del mismo hacia los procesos se hacen a través del protocolo TCP.

2.3.5.5.2. Resultados de análisis.

Ninguno.

2.3.5.5.3. Asumpciones.

Ninguna.

2.3.5.6. Otra información.

[Omitido]

2.3.5.7. Paquetes de vista relacionados.

- Padre: Ninguno.
- Hijo: Ninguno.
- Hermanos:
  - Vista de Comunicación de Procesos. Subsistema de Adquisición (no disponible en este trabajo).
  - Vista de Comunicación de Procesos. Subsistema de Circulación (no disponible en este trabajo).
  - Vista de Comunicación de Procesos. Subsistema de DSI (no disponible en este trabajo).
  - Vista de Comunicación de Procesos. Subsistema de Referencias (no disponible en este trabajo).
  - Vista de Comunicación de Procesos. Subsistema de Estadísticas (no disponible en este trabajo).

2.3.6. Vista de Implementación. Sistema Integrado para Bibliotecas.

2.3.6.1. Presentación primaria.

Subsistema	Donde se almacena	Archivos	Frecuencia de construcción	Contacto
Adquisición	URL/Adquisicion	Omitido	Omitido	dalfaro@estudiantes.uci.cu
Catalogación	URL/Catalogacion	Omitido	Omitido	gjorda@estudiantes.uci.cu
Consultas	URL/Consultas	Omitido	Omitido	kreyes@estudiantes.uci.cu
Circulación	URL/Circulacion	Omitido	Omitido	orey@estudiantes.uci.cu
DSI	URL/DSI	Omitido	Omitido	losorio@estudiantes.uci.cu
Referencias	URL/Referencia	Omitido	Omitido	lrosas@estudiantes.uci.cu
Estadísticas	URL/Estadisticas	Omitido	Omitido	yaveleira@estudiantes.uci.cu
Administración	URL/Administracion	Omitido	Omitido	?@estudiantes.uci.cu
Generador de Reportes	URL/Generador	Omitido	Omitido	?@estudiantes.uci.cu
Acceso a Datos	URL/AccesoD	Omitido	Omitido	yfeus@estudiantes.uci.cu

Nota: URL = Dirección https del servidor Subversion.

2.3.6.2. Catálogo de elementos.

2.3.6.2.1. Elementos y sus propiedades.

- Elementos de software: un módulo.

- Elemento ambiental(o de entorno): un elemento de configuración, como un archivo o un directorio (gestionados por el servidor Subversion).

En este caso los elementos aparecen en la presentación primaria, no se describen aquí para evitar redundancia, la descripción de cada módulo aparece en la *Vista de Descomposición* (Sección 2.3.2).

- Propiedades de los módulos.
  - Donde se almacena: localización en el servidor donde el subsistema está almacenado durante el desarrollo. La localización se da tomando como referencia el servicio que presta Subversion a través del protocolo https.
  - Archivos: archivos por los que está formado, incluye archivos ejecutables, de código fuente y los demás artefactos generados por el proceso de ingeniería de software. En la vista se omiten pues la etapa de desarrollo en la que se encuentra el producto no lo permite.
  - Frecuencia de construcción: ciclo de construcción, puede verse como el tiempo que dura una iteración.
  - Contacto: información de contacto del miembro del equipo responsable de proporcionar la última versión.

Además de las siguientes propiedades presentes en todos los módulos que no están en la presentación primaria:

- Plataforma de desarrollo: Servidor de Aplicaciones Zope 3.
- Lenguajes de desarrollo: Python. Se añade como excepción el módulo de Acceso a Datos, que en parte estará codificado en C.
- Intérprete de Python: Python 2.4.4

2.3.6.2.2. Relaciones y sus propiedades.

- Asignado-a: describe la asignación de un módulo a un elemento de configuración. Esta relación puede verse en la propiedad *Donde se almacena,* en la vista primaria.

2.3.6.2.3. Interfaces de elementos.

[Omitido]

2.3.6.2.4. Comportamiento de elementos.

No aplicable.

2.3.6.3. Un diagrama de contexto.

Ninguno.

2.3.6.4. Guía de variabilidad.

Ninguna.

2.3.6.5. Background de Arquitectura.

2.3.6.5.1. Lógica.

Para la gestión de la configuración existen varias alternativas en cuanto a sistemas de gestión de configuración, en este caso se opta por Subversion por las siguientes características [SubWi]:

- Se sigue la historia de los archivos y directorios a través de copias y renombrados.
- Las modificaciones (incluyendo cambios a varios archivos) son atómicas.
- El creado de ramas y etiquetas es una operación más eficiente. Tiene costo de complejidad constante (O (1)) y no lineal (O(n)) como en CVS (Concurrent Version System).
- Se envían sólo las diferencias en ambas direcciones (en CVS siempre se envían al servidor archivos completos).
- Puede ser servido, mediante Apache, sobre WebDAV/DeltaV. Esto permite que clientes WebDAV utilicen Subversion en forma transparente.
- Maneja eficientemente archivos binarios (CVS los trata internamente como si fueran de texto).

2.3.6.5.2. Resultados de análisis.

Ninguno.

2.3.6.5.3. Asumpciones.

[Omitido]

2.3.6.6. Otra información.

[Omitido]

2.3.6.7. Paquetes de vista relacionados.

- Padre: Ninguno.
- Hijo: Ninguno.
- Hermanos: Ninguno en esta vista. Los paquetes de vista de otras vistas que expresan el mismo alcance (ámbito), es decir, que incluyen el sistema entero, se encuentran en:

    - Vista de Descomposición. Sistema Integrado para Bibliotecas (Sección 2.3.2).
    - Vista de Capas. Sistema Integrado para Bibliotecas (Sección 2.3.3).
    - Vista de Implementación. Sistema Integrado para Bibliotecas (Sección 2.3.6).
    - Vista de Despliegue. Sistema Integrado para Bibliotecas (Sección 2.3.7)
    - Vista de Asignación de Trabajo. Sistema Integrado para Bibliotecas (Sección 2.3.8).

2.3.7. Vista de Despliegue. Sistema Integrado para Bibliotecas.

2.3.7.1. Presentación primaria.

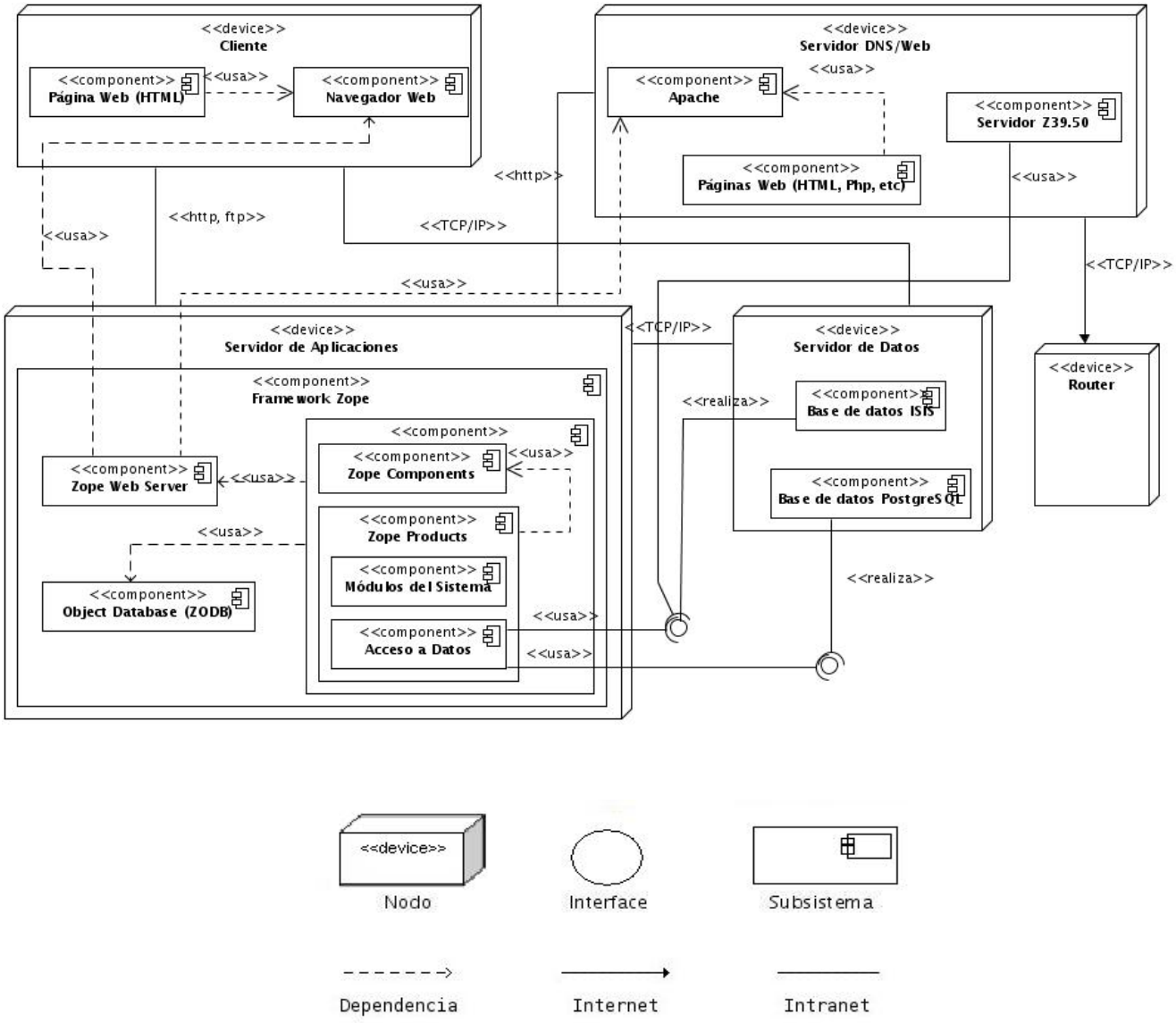

Figura 2.9 Vista de Despliegue.

2.3.7.2. Catálogo de elementos.

2.3.7.2.1. Elementos y sus propiedades.

Los elementos en el diagrama anterior son los subsistemas (representados como componentes) del Sistema Integrado para Bibliotecas (ILS), los servidores (host computer) y estaciones clientes (client computer), routers, y redes. Los subsistemas del ILS han sido definidos en la Vista de Descomposición, mientras que los otros elementos son elementos de entorno definidos en esta vista. Los elementos son descritos en la tabla siguiente; los subsistemas del ILS están por encima de la línea doble, mientras que los elementos de entorno están por debajo de la línea doble.

La columna *Velocidad de Procesamiento* define la velocidad mínima requerida para cada subsistema, y provee la velocidad de procesamiento para los elementos de entorno, en este caso, para los servidores. Las velocidades están dadas en GHz. La columna *Velocidad de Comunicación* define la velocidad de conexión de red requerida por los varios subsistemas y la velocidad que provee la red para los varios elementos de entorno. Las unidades son en megabits por segundo (MBPS). La columna *Memoria RAM* define la memoria RAM requerida por los varios subsistemas y la memoria RAM que provee cada elemento de entorno, en este caso los servidores y estaciones clientes. Las memorias están dadas en unidades de gigas (GB).

Debido a que muchos de los subsistemas del ILS estarán soportados sobre la plataforma Zope, los mismos serán agrupados en un solo subsistema, el subsistema *Framework Zope*. Esto añade claridad a información presentada, pues los requerimientos de estos subsistemas dependen, y de hecho coinciden, con los requerimientos de Zope.

Nombre del elemento	Velocidad de procesamiento (GHz)	Velocidad de comunicación (MBPS)	Memoria RAM (GB)
Framework Zope 3.x	-	-	-

Nombre del elemento	Velocidad de procesamiento (GHz)	Velocidad de comunicación (MBPS)	Memoria RAM (GB)
Apache 2.x	-	-	-
Módulo Z39.50 (servidor)	-	-	-
Gestor de BD (ISIS)	-	-	-
Gestor de BD (PostgreSQL)	-	-	-
Estación Cliente	-	-	-
Servidor de Aplicaciones	2.8	10 - 100	1.5
Servidor de BD	2.8	10 - 100	1.0
Servidor DNS/Web	2.8	10 - 100	1.0

2.3.7.2.2. Relaciones y sus propiedades.

La presentación primaria muestra como los subsistemas son *asignados a* los servidores. Algunos subsistemas se ejecutarán en servidores diferentes. Esto es en parte para proveer seguridad, rendimiento y flexibilidad en el procesamiento de datos para soportar varias fuentes de datos y varias necesidades de usuarios.

2.3.7.2.3. Interfaces de elementos.

[Omitido]

2.3.7.2.4. Comportamiento de elementos.

No aplicable.

2.3.7.3. Un diagrama de contexto.

El contexto para esta vista está establecido por el diagrama de contexto de la Vista de Descomposición (Sección 2.3.2.3).

2.3.7.4. Guía de variabilidad.

Ninguna.

2.3.7.5. Background de Arquitectura.

2.3.7.5.1. Lógica.

Aunque la presentación primaria no lo tiene en cuenta, la arquitectura contempla el uso de otra instancia de Zope (de manera transparente para el usuario) para balancear las cargas, permitiendo mejorar el tiempo de respuesta a las solicitudes hechas al sistema; se propone además el uso de dos servidores de bases de datos sincronizados, con el objetivo de mantener copias de seguridad de las bases de datos y a la vez garantizar la disponibilidad de los servicios ante posibles fallos.

2.3.7.5.2. Resultados de análisis.

Ninguno.

2.3.7.5.3. Asumpciones.

Cuando el sistema esté disponible para la Red Nacional de Bibliotecas Públicas, los servidores necesitarán mayor capacidad de procesamiento, y la red un mayor ancho de banda para garantizar el correcto funcionamiento del sistema.

2.3.7.6. Otra información.

[Omitido]

2.3.7.7. Paquetes de vista relacionados.

- Padre: Ninguno.
- Hijo: Ninguno.
- Hermanos: Ninguno en esta vista. Los paquetes de vista de otras vistas que expresan el mismo alcance (ámbito), es decir, que incluyen el sistema entero, se encuentran en:

  - Vista de Descomposición. Sistema Integrado para Bibliotecas (Sección 2.3.2).
  - Vista de Capas. Sistema Integrado para Bibliotecas (Sección 2.3.3).
  - Vista de Usos. Sistema Integrado para Bibliotecas (Sección 2.3.4).
  - Vista de Implementación. Sistema Integrado para Bibliotecas (Sección 2.3.6).
  - Vista de Asignación de Trabajo. Sistema Integrado para Bibliotecas (Sección 2.3.8).

2.3.8. Vista de Asignación de Trabajo. Sistema Integrado para Bibliotecas.

2.3.8.1. Presentación primaria.

Subsistema	**Unidad organizacional**
Adquisición	Equipo de Adquisición
Catalogación	Equipo de Catalogación
Consultas	Equipo de Consultas
Circulación	Equipo de Circulación
DSI	Equipo de DSI
Referencia	Equipo de Referencia
Estadísticas	Equipo de Estadísticas
Administración	Equipo de Administración
Generador de Reportes	Equipo de Generador de Reportes
Acceso a Datos	Equipo de Acceso a Datos

Nota: Los subsistemas están descritos en la *Vista de Descomposición* (Sección 2.3.2).

2.3.8.2. Catálogo de elementos.

2.3.8.2.1. Elementos y sus propiedades.

Elementos (Equipos)	Propiedades	
	Contacto del miembro del equipo	**Conjunto de habilidades**
Equipo de Adquisición	Dayamí Alfaro Montesino dalfaro@estudiantes.uci.cu	Las habilidades que se listan a continuación corresponden a analistas y diseñadores de sistemas. • Facilidad de comunicación, buena redacción y ortografía. Dominio de la metodología RUP y del lenguaje de modelado UML. • Modelar solución de software basada en el uso del servidor de aplicaciones Zope. • Conocimientos del lenguaje python y aplicación de patrones de diseño. • Conocimientos básicos sobre bibliotecología según el módulo que corresponda.
Equipo de Catalogación	Greidys Jorda Luegues gjorda@estudiantes.uci.cu Jaliet Barbara Rojas bjrojas@estudiantes.uci.cu Yaumara Arce Ajo yarce@estudiantes.uci.cu	

	Propiedades	
Equipo de Consultas	Kenia Reyes Hernandez (kreyes@estudiantes.uci.cu) Daylin Matos Castillo dmcastillo@estudiantes.uci.cu	
Equipo de Circulación	Omar Rey Lazarte orey@estudiantes.uci.cu	
Equipo de DSI	Lisandra Hernández Osorio losorio@estudiantes.uci.cu	
Equipo de Referencia	Lisset Rosas Moreno lrosas@estudiantes.uci.cu Yan Pavel Espinosa	
Equipo de Estadísticas	Yanicet Aveleira Rodriguez yaveleira@estudiantes.uci.cu	
Equipo de Administración	No asignado	
Equipo de Generador de Reportes	Bertha Román Martinez broman@estudiantes.uci.cu	
Equipo de Acceso a Datos	Yudanis Feus Perez yfeus@estudiantes.uci.cu (diseñador principal de bases de datos) Raydel Miranda Gomez rmiranda@estudiantes.uci.cu	• Dominio de las tecnologías y teorías de base de datos. Conocimientos sobre el modelo relacional y el manejo de herramientas para la gestión de bases de datos relacionales

	Propiedades	
	Luis Valdes Guerrero lguerrero@estudiantes.uci.cu	(PostgreSQL). • Conocimientos de formatos y reglas de catalogación, de la estructura de las bases de datos bibliográficas y el manejo de gestores de bases de datos ISIS (OpenIsis1.0, MicroIsis). • Experiencia en el uso de tecnologías de acceso a datos y su uso con los lenguajes C y Python.

2.3.8.2.2. Relaciones y sus propiedades.

- *Asignado a*: especifica la asignación de módulos o subsistemas a equipos de trabajo.

2.3.8.2.3. Interfaces de elementos.

No aplicable.

2.3.8.2.4. Comportamiento de elementos.

No aplicable.

2.3.8.3. Un diagrama de contexto.

El contexto para esta vista está establecido por el diagrama de contexto de la *Sección 2.3.2* de la Vista de Descomposición.

2.3.8.4. Guía de variabilidad.

Ninguna.

2.3.8.5. Background de Arquitectura.

2.3.8.5.1. Lógica.

En la *Presentación primaria* solo se han incluido los nombres de los analistas y diseñadores del sistema. No se ha incluido el resto de los roles al no contar el equipo con los mismos.

2.3.8.5.2. Resultados de análisis.

Ninguno.

2.3.8.5.3. Asumpciones.

[Omitido]

2.3.8.6. Otra información.

[Omitido]

2.3.8.7. Paquetes de vista relacionados.

- Padre: Ninguno.
- Hijo: Ninguno.

- Hermanos: Ninguno en esta vista. Los paquetes de vista de otras vistas que expresan el mismo alcance (ámbito), es decir, que incluyen el sistema entero, se encuentran en:

  - Vista de Descomposición. Sistema Integrado para Bibliotecas (Sección 2.3.2).
  - Vista de Capas. Sistema Integrado para Bibliotecas (Sección 2.3.3).
  - Vista de Usos. Sistema Integrado para Bibliotecas (Sección 2.3.4).
  - Vista de Despliegue. Sistema Integrado para Bibliotecas (Sección 2.3.5).
  - Vista de Implementación. Sistema Integrado para Bibliotecas (Sección 2.3.6).

2.4 Conclusiones.

La documentación presentada en este capítulo, correspondiente a las vistas de la arquitectura, representa correctamente la estructura del sistema, mostrando sus partes fundamentales en forma de módulos y las principales relaciones que se establecen entre ellos. Las vistas documentadas describen los aspectos más importantes a tener en cuenta para el desarrollo del sistema y permiten hacer un análisis de requisitos no funcionales tales como rendimiento, modificabilidad, seguridad, disponibilidad, usabilidad, entre otros. Este capítulo cubre las necesidades de información más importantes de los interesados en el desarrollo del proyecto.

CAPÍTULO 3: Evaluando la Arquitectura

La arquitectura es un artefacto decisivo en la calidad del software que se desarrolla. Su evaluación permite mitigar los diferentes riesgos asociados con el desarrollo del software. Permite mejorar la visión de los procesos críticos y validar las decisiones de diseño que se tomaron, así como tomar decisiones tempranas. Cuanto más temprano se encuentre un problema en un proyecto de software, mejor. El costo de arreglar un error durante las fases de requerimientos o diseño, es mucho menor al costo de arreglar ese mismo error en la fase de verificación. Dado que la arquitectura es un producto temprano de la fase de diseño, esta tiene un profundo efecto en el sistema y en el proyecto.

Una mala arquitectura puede llevar a un proyecto al fracaso. Todos los requerimientos de calidad pueden quedar insatisfechos. La arquitectura también determina la estructura del proyecto: configuración, agenda y presupuesto, alcance, entre otros aspectos. Es mejor cambiar la arquitectura antes que otros artefactos, que están basados en ella, se estabilicen. Realizar una evaluación de la arquitectura es la manera más económica de evitar desastres.

3.1 Evaluando una arquitectura de software.

3.1.1. ¿Cuándo una arquitectura puede ser evaluada?

Generalmente, la evaluación de la arquitectura ocurre después que esta ha sido especificada, pero antes que empiece la implementación. En un proceso iterativo y/o incremental, la evaluación se puede realizar al final de cada ciclo. Sin embargo, uno de los atractivos de la evaluación de arquitecturas es que se puede efectuar en cualquier etapa de la vida de una arquitectura. En particular, existen dos variaciones útiles: temprana y tardía. [Eval-A].

3.1.1.1. Evaluación temprana.

La evaluación no tiene porque esperar a que la arquitectura este totalmente especificada. Esta puede ser

utilizada en cualquier etapa del proceso de creación de la arquitectura, para examinar las decisiones arquitectónicas ya tomadas y decidir entre las opciones que están pendientes.

Por supuesto, la completitud y fidelidad de la evaluación es directamente proporcional a la completitud y fidelidad de la descripción de la arquitectura.

3.1.1.2. Evaluación tardía.

Esta variación toma lugar no solo cuando la arquitectura está terminada, también cuando la implementación está completa. Este caso ocurre cuando la organización hereda un sistema legado. La técnica para evaluar un sistema legado es la misma que para evaluar un sistema recién nacido. Una evaluación es útil para entender el sistema legado, y saber si este cumple con los requerimientos de calidad y comportamiento.

En general, una evaluación debe realizarse cuando hay suficiente de la arquitectura como para justificarlo. Una buena regla sería: *realizar una evaluación cuando el equipo de desarrollo empieza a tomar decisiones que dependen de la arquitectura y el costo de deshacerlas sobrepasa al costo de realizar una evaluación*.

3.1.2. ¿Quiénes están involucrados?

Hay dos grupos de personas involucrados en la evaluación de la arquitectura.

3.1.2.1. Equipo de evaluación.

Estas son las personas que conducirán la evaluación y realizaran el análisis.

3.1.2.2. Stakeholders.

Estos son los interesados en la arquitectura, y en el sistema que se construirá a partir de ella. Algunos de los stakeholders, pero no todos, serán miembros del equipo de desarrollo, como ser implementadores,

verificadores, entre otros. Un tipo especial de stakeholder es el *decision maker* (tomador de decisiones) del proyecto. Estos son los interesados en la salida de la evaluación y tienen el poder de toma de decisiones. Algunos de ellos son el arquitecto, los diseñadores y el director de proyecto. Cuando un stakeholder solicita un cambio en la arquitectura, un decision maker tiene el poder de asignar recursos para hacerlo realidad. Generalmente el cliente de la evaluación de la arquitectura es un decision maker, con un gran interés en la salida de la misma y poder sobre todo el proyecto.

En ocasiones el equipo de evaluación esta formado por integrantes del plantel del proyecto, en cuyo caso también son stakeholders. Esto no está recomendado dado que habrá pérdida de la objetividad a la hora de ver la arquitectura.

3.1.3. ¿Qué resultados produce la evaluación de una arquitectura?

En términos concretos, la evaluación de la arquitectura produce un informe, la forma y contenido del mismo varía según el método utilizado. En particular, produce repuestas a dos tipos de preguntas:

- ¿Es esta arquitectura adecuada para el sistema para la cual fue diseñada?
- ¿Cuál de dos o más arquitecturas propuestas es la más adecuada para el sistema?

Decimos que una arquitectura es adecuada cuando cumple dos criterios:

- El sistema resultante de ella cumple con los objetivos de calidad. No todas las propiedades de calidad del sistema son resultado directo de la arquitectura, pero muchas lo son. [Eval-A].
- El sistema puede ser construido con los recursos con que se cuenta: el plantel, el presupuesto, el sistema legado (si hay), entre otros. Esto es, la arquitectura es construidle. [Eval-A].

Un resultado que también produce la evaluación de una arquitectura es la captura y priorización de las metas que la arquitectura debe cumplir para poder ser considerada adecuada.

La evaluación de una arquitectura no produce resultados cuantitativos. No es de interés, por ejemplo, evaluar la performance en cantidad de transacciones por segundo a esta altura, dado que el sistema no esta construido aún. Lo que interesa, con el objetivo de mitigar los riesgos, es aprender como un atributo

de calidad es afectado por una decisión de diseño arquitectónico, para que de esta manera se pueda estudiar con cuidado dicha decisión.

Una evaluación arquitectónica dice si una arquitectura es adecuada respecto a un conjunto de metas, y problemática con respecto a otro conjunto de metas. En ocasiones, las metas pueden ser contradictorias entre ellas, o algunas ser más importantes que otras. El director de proyecto es quien deberá tomar la decisión si la arquitectura evalúa bien o mal en las distintas áreas. La evaluación ayuda a encontrar debilidades, no dirá “si” o “no”, “bien” o “mal”, “6 en 10”, dirá donde están los riesgos.

3.1.4. ¿Por qué cualidades puede ser evaluada una arquitectura?

No es del todo cierto que podamos decir que el sistema alcanzará todas sus metas de calidad con solo mirar la arquitectura. Pero muchos atributos de calidad yacen directamente en el reino de la arquitectura. Una arquitectura puede ser evaluada en base a los siguientes atributos de calidad [Eval-A]:

- *Performance.*
- *Reliability.*
- *Availability.*
- *Security.*
- *Modifiability.*
- *Portability.*
- *Functionality.*
- *Variability.*
- *Subsetability.*
- *Conceptual integrity*.

Si otro atributo de calidad, además de los mencionados antes, es importante para determinado proyecto, este también debe formar parte de la evaluación. En particular, el método de evaluación ATAM (Architecture Tradeoff Analysis Method) permite definir nuevos atributos de calidad a ser utilizados en la evaluación.

3.1.5. ¿Por qué los atributos de calidad son demasiados imprecisos para el análisis?

Los atributos de calidad son la base para a la evaluación de una arquitectura, pero por si solos no son suficientes para juzgar la adecuabilidad de la arquitectura. Generalmente, los requerimientos son escritos de la siguiente manera [Eval-A]:

- “El sistema debe ser robusto.”
- “El sistema debe ser modificable.”
- “El sistema debe ser seguro.”
- “El sistema debe tener una performance aceptable.”

Cada una de las frases anteriores está sujeta a diferentes interpretaciones y malos entendidos. Lo que uno puede considerar robusto, otro puede considerarlo apenas aceptable.

El punto es que los atributos de calidad no son cantidades absolutas, existen en un contexto con metas específicas. En particular:

- Un sistema es modificable (o no) con respecto a un tipo de cambio específico.
- Un sistema es seguro (o no) con respecto a un tipo de amenaza específica.
- Un sistema es confiable (o no) con respecto a la ocurrencia de un tipo de falta específica.
- Un sistema es performante (o no) con respecto a un criterio específico de performance.
- Una arquitectura es construible (o no) con respecto a restricciones de tiempo y presupuesto específicas.

No parece razonable, considerar que un sistema pueda alguna vez, por ejemplo, ser completamente confiable bajo toda circunstancia (pensar en problemas de energía, meteorológicos, entre otros). Dado esto, es importante que el arquitecto entienda perfectamente bajo que circunstancias un sistema debe ser confiable para ser considerado aceptable. Por lo tanto, el primer trabajo que debe realizar una evaluación de arquitectura es obtener las metas específicas de calidad ante las cuales la arquitectura será juzgada.

Si algunas de estas metas no son específicas o son ambiguas, se debe pedir a los stakeholders que

ayuden al equipo de evaluación a reescribirlas. El mecanismo a utilizar para representar estas metas es el de *escenario*. Un escenario es una pequeña descripción de la interacción de un stakeholder con el sistema. Los escenarios son parecidos a los *casos de uso*.

3.1.6. ¿Cuáles son las salidas de una evaluación arquitectónica?

3.1.6.1. Lista priorizada de los atributos de calidad requeridos.

Una evaluación arquitectónica solo puede proceder si se conoce el criterio de adecuabilidad. Es más, la obtención de los atributos de calidad requeridos contra los cuales la arquitectura será juzgada, constituye la mayor parte del trabajo. Pero ninguna arquitectura puede tener una lista interminable de atributos de calidad, y por lo tanto, se utilizan métodos de priorización concensuados. [Eval-A].

3.1.6.2. Riesgos y no riesgos.

Los riegos son decisiones arquitectónicas potencialmente problemáticas. Los no riesgos son buenas decisiones, que confían en asunciones que con frecuencia son implícitas en la arquitectura.

Documentar riesgos y no riesgos consiste en:

- Una decisión arquitectónica (o una decisión que no ha sido tomada)
- Una respuesta específica al atributo de calidad que esta siendo tratado, junto con las consecuencias del nivel predecible de la respuesta.
- Una base lógica por el efecto positivo o negativo que la decisión tuvo en alcanzar el requerimiento del atributo de calidad.

Para que un no riesgo se mantenga como tal, las asunciones no deben cambiar, o al menos si cambian, la designación como no riesgo debe ser rejustificada.

3.1.7. ¿Cuáles son los costos y beneficios de realizar una evaluación arquitectónica?

El mayor beneficio que brinda la evaluación de una arquitectura, es que descubre los problemas que si se hubiesen dejado sin descubrir, habría sido mucho más costoso corregirlos luego. En breve, una evaluación produce una mejor arquitectura. [Eval-A].

Algunos beneficios más que brinda la evaluación se presentan a continuación.

3.1.7.1. Reúne a los stakeholders.

En una evaluación arquitectónica es, comúnmente, la primera vez en que la mayoría de los stakeholders se encuentran. Es más, es la primera vez que el arquitecto se encuentra con ellos. Todos comparten un objetivo, lograr un sistema exitoso.

3.1.7.2. Fuerza una articulación en las metas específicas de calidad.

El rol del stakeholder es articular las metas de calidad que la arquitectura debería alcanzar para ser considerada exitosa. Estas metas no siempre son capturadas en algún documento de requerimientos.

3.1.7.3. Fuerza una explicación clara de la arquitectura.

El arquitecto convoca a un grupo de personas, no en privado, para explicarles la creación de la arquitectura, en detalle y sin ambigüedades. El proyecto se ve beneficiado cuanto antes se realice esta explicación.

3.1.7.4. Mejora la calidad de la documentación de la arquitectura.

Generalmente, la evaluación pide documentación que aún no esta terminada, entonces se designa alguien del plantel a terminarla. Una vez más, el proyecto se ve beneficiado porque se ingresa al desarrollo mejor preparado al tener los documentos terminados.

3.1.7.5. Descubre oportunidades de rehuso.

Los stakeholders y el equipo de evaluación provienen de afuera del proyecto de desarrollo, pero trabajan o están familiarizados con otros proyectos. Por lo tanto, ambos están en una buena posición para detectar componentes que pueden ser rehusados en otros proyectos, o conocer componentes que ya existen que pueden ser utilizados en el proyecto actual.

3.1.7.6. Resultan mejoras en las arquitecturas.

Las organizaciones que practican la evaluación arquitectónica como un estándar de su proceso de desarrollo, reportan una mejora en la calidad de las arquitecturas que son evaluadas. La evaluación de arquitecturas resulta en mejores arquitecturas no solo después de hecha, pero antes también.

Los costos de la evaluación arquitectónica son todos costos de personal y costos de oportunidad por aquel personal que participa de la evaluación en vez de hacer otra cosa. Estos costos son sencillos de calcular (están tabulados).

3.1.8. ¿Cómo puedo realizar una evaluación arquitectónica?

Dicha evaluación puede realizarse mediante técnicas cualitativas, como cuestionarios o escenarios, o a través de técnicas cuantitativas, como simulaciones o modelos matemáticos. En la literatura existen diferentes métodos de evaluación para verificar desde múltiples atributos de calidad hasta algunos en específico. Ejemplos de estos métodos de evaluación son ATAM (Architecture Trade-off Analysis Method), ARID (Active Reviews for Intermediate Designs), SAAM (Software Architecture Analysis Method), SNA (Survivable Network Analysis), RMA (Rate Monotonic Analysis), teoría de colas, teoría de confiabilidad, entre otras. [Arq-C].

Como se puede apreciar, existe una gran variedad de técnicas y/o métodos que permiten evaluar una arquitectura de software. Luego de haber realizado un estudio sobre los métodos más relevantes, teniendo en cuenta el propósito, el contexto, las debilidades y fortalezas, y además valorando que la arquitectura se encuentra en sus primeras etapas, se decidió aplicar el método ARID. De acuerdo con

[Kaz01] el método ARID es conveniente para realizar la evaluación de diseños parciales en las etapas tempranas de desarrollo.

3.2 Evaluando la arquitectura propuesta en este trabajo.

El método ARID surge de combinar las mejores cualidades de los métodos ADRs y los métodos basados en escenarios. De los ADRs nos quedamos con la participación activa de los entrevistados. Del ATAM nos quedamos con la idea de la generación de los escenarios para demostrar que el diseño propuesto cumple con los requerimientos.

La arquitectura propuesta en este trabajo se encuentra representada a un alto nivel (diseño poco detallado), esto trajo consigo la dificultad de seleccionar escenarios concretos (como por ejemplo: validación automática de la entrada de datos) que permitiesen validar si la arquitectura responde a requisitos específicos. Debido a ello, se decidió validar la arquitectura evaluando como las vistas seleccionadas responden a cada uno de los principales requerimientos y restricciones del sistema. Constituyendo nuestros escenarios los siguientes:

1. Rendimiento del sistema.
2. Modificabilidad y evolución del sistema.
3. Seguridad del sistema.
4. Disponibilidad del sistema.
5. Portabilidad del sistema.
6. Independencia entre la aplicación y los sistemas gestores de bases de datos.

Rendimiento del sistema: Se debe consultar la vista de comunicación de procesos para analizar la concurrencia en el sistema. También se debe consultar la vista de despliegue.

Modificabilidad y evolución del sistema: Para analizar el impacto de un cambio, las vistas de uso y de descomposición son muy útiles. También la vista de capas muestra como un cambio en una capa más baja puede ser ocultado detrás de sus interfaces y no impactará las capas encima de ella.

Seguridad del sistema: La vista de despliegue es usada para ver los puntos de contacto del sistema con el exterior, así como el software encargado de establecer la interacción; también muestra donde los aspectos autenticación e integridad son manejados, el servidor de aplicaciones Zope realiza esta tarea. La denegación de servicios es pérdida de rendimiento, y así el analista de seguridad querrá ver la misma información que el analista de rendimiento.

Disponibilidad del sistema: Una vista de comunicación de procesos puede ayudar a analizar los procesos de mayor concurrencia. La vista de despliegue es usada para mostrar posibles puntos de fallo. Pueden definirse números de fiabilidad para un módulo como una propiedad en la vista de descomposición.

Portabilidad del sistema: El sistema será desarrollado utilizando la plataforma de desarrollo Zope, el cual corre en las plataformas de sistemas operativos más populares: Linux, Windows NT/2000/XP, Solaris, FreeBSD, NetBSD, OpenBSD, y Mac OS X.

Independencia entre la aplicación y los sistemas gestores de bases de datos: Esto se logra con la capa de persistencia en la vista de capas. Esta capa media entre la capa de modelo del dominio y la capa de fuentes de datos, permitiendo que para la primera sea transparente la manera en que la información es persistida o almacenada en la fuente de datos. La capa de persistencia además tendrá como función fundamental, lograr que el sistema funcione correctamente independientemente de la fuente de datos seleccionada.

3.3 Conclusiones.

Debido a la ausencia de una detallada documentación, lo que hemos logrado con este capítulo es una pre-evaluación de la arquitectura, mostrando cómo cada una de las vistas documentadas en el Capítulo 2, permiten, además de implementar la funcionalidad deseada, dotar al producto de los atributos de calidad establecidos. La arquitectura que se propone es factible para construir un ILS, permitiendo además, mitigar los principales riesgos, mejorar la visión de los procesos críticos y validar las decisiones de diseño que se tomaron.

CONCLUSIONES

La investigación realizada muestra que en el campo de la arquitectura de software todavía no hay consenso generalizado en cuanto a los conceptos más importantes, ni a la forma de documentar. La propuesta seleccionada ha permitido documentar de forma correcta la arquitectura de software del sistema que se pretende construir, la documentación generada es comprendida por los miembros del equipo de desarrollo y sirve de base para la construcción del sistema.

Las vistas que se han tenido en cuenta han sido correctamente seleccionadas teniendo en consideración las necesidades de información más importantes de los interesados en el desarrollo del proyecto y han permitido una mejor comunicación entre los miembros del equipo de desarrollo y con el cliente. La evaluación de la arquitectura muestra que se ha logrado el objetivo del presente trabajo, pues se ha hecho un diseño de arquitectura que permitirá el desarrollo de un sistema integrado para la Biblioteca Nacional, que cumplirá con los requisitos de calidad que se han propuesto.

RECOMENDACIONES

Con el propósito de ampliar y mejorar la documentación de la arquitectura presentada en este trabajo, se proponen las siguientes recomendaciones:

- En etapas posteriores del proyecto incluir nuevos estilos en la documentación de la arquitectura, como el Pipe-and-Filter, para un mejor entendimiento de como serán transformados los datos en el sistema a construir. Incluir cualquier otro estilo de interés que no se haya tenido en cuenta.
- Completar la vista de comunicación de procesos correspondiente a cada uno de los módulos para un mejor entendimiento de la concurrencia en el sistema.
- En cada vista documentada, completar aquellas secciones que se han omitido por no ser objetivos de este trabajo.
- Elaborar un documento de especificación de interfaces, donde se documenten de forma detallada las interfaces de los elementos representados en las vistas de la arquitectura.
- Crear, como parte del equipo de desarrollo, un equipo encargado del estudio y aplicación de los métodos de evaluación de arquitectura de software más importantes.
- Para establecer un estándar en cuanto a documentación de la arquitectura, se recomienda que los proyectos productivos de la facultad utilicen la propuesta plasmada en [Cle02] para documentar la arquitectura, que ha sido utilizada en este trabajo.

BIBLIOGRAFÍA

- [RUP] Wikipedia. *Proceso Unificado de Rational*, [en línea]. <http://es.wikipedia.org/wiki/Proceso_Unificado_de_Rational>.[Consulta: 2 de febrero de 2007]
- [ZTCA] *¿Buscas Información sobre Zope y Plone?,* [en línea]. <http://www.zopeteca.com/> [Consulta: 3 de marzo de 2007]
- [BDRE] Wikipedia.*Modelo Relacional,* [en línea]. <http://es.wikipedia.org/wiki/Base_de_datos_relacional#Base_de_datos_relacional> [Consulta : 5 de marzo de 2007]
- [SEI] Software Engineering Institute. *How Do You Define Software Architecture?*, [en línea]. <http://www.sei.cmu.edu/architecture/definitions.html> .[Consulta : 20 de noviembre de 2006]
- [BDMS] *Web, Base de Datos y DBMS*, [en línea]. <http://www.htmlpoint.com/sql/sql_02.htm>. [Consulta: 10 de marzo de 2007]
- [MLT-O] OpenIsis Project. *Announcing Malete, the database engine powering OpenIsis 1.0,* [en línea]. <http://malete.org/Doc/OverView>. [Consulta: 4 de abril de 2007]
- [MLT-P] OpenIsis Project. *The Malete server protocol*, [en línea]. <http://malete.org/Doc/Protocol>. [Consulta: 4 de abril de 2007]
- [PSQL] *¿Qué es PostgreSQL?,* [en línea]. <http://www.openecuador.org/modules/news/article.php?storyid=31>. [Consulta: 8 de marzo de 2007]
- [SubWi] Wikipedia. *Subversion*, [en línea]. <http://es.wikipedia.org/wiki/SVN>. [Consulta: 3 de junio de 2007]
- [Arq-C] José de Jesús Suárez. *Arquitectura de Software: importancia de su ciclo de vida*, [en línea]. <http://www.enterate.unam.mx/Articulos/2006/febrero/arquitec.htm>. [Consulta: 8 de mayo de 2007]
- [Eval-A] Mauricio Dávila, Martín Germán, Diego Crutas, Andrés García. *Evaluación de Arquitecturas de Software,* [en línea]. <http://www.fing.edu.uy/inco/cursos/gestsoft/Presentaciones/Evaluacion%20de%20Arquitecturas%20-%20G10/Evaluacion%20de%20Arquitecturas.doc>. [Consulta: 3 de junio de 2007]

- [Cle02] Documenting Software Architectures: Views and Beyond
  Paul Clements (Editor), Felix Bachmann, Len Bass, David Garlan, James Ivers, Reed Little, Robert Nord, Judith Stafford, Addison Wesley Professional (2002).
- [Cle96a] Paul Clements. “A Survey of Architecture Description Languages”. Proceedings of the International Workshop on Software Specification and Design, Alemania, 1996.
- [Gar00] David Garlan. “Software Architecture: A Roadmap”. En Anthony Finkelstein (ed.), The future of software engineering, ACM Press, 2000.
- [BCK98] Len Bass, Paul Clements y Rick Kazman. Software Architecture in Practice. Reading, Addison-Wesley, 1998.
- [SG96] Mary Shaw y David Garlan. Software Architecture: Perspectives on an emerging discipline. Upper Saddle River, Prentice Hall, 1996.
- [Kaz01] Kazman, R., Clements, P., Klein, M. (2001). Evaluating Software
- Architectures: Methods and Case Studies. Addison Wesley
- [Zac87] John A. Zachman, “A Framework for Information Systems Architecture”, IBM Systems Journal, 26(3), 1987.
- [BRJ99] Grady Booch, James Rumbaugh e Ivar Jacobson. El Lenguaje Unificado de Modelado. Madrid, Addison-Wesley, 1999.
- [Kru95] Philippe Kruchten. “The 4+1 View Model of Architecture.” IEEE Software 12(6), pp. 42-50, Noviembre de 1995.
- [BMR+96] Frank Buschmann, Regine Meunier, Hans Rohnert, Peter Sommerlad y Michael Stal. Pattern-oriented software architecture – A system of patterns. John Wiley & Sons, 1996.
- [Platt02] Michael Platt. “Microsoft Architecture Overview: Executive summary”, http://msdn.microsoft.com/architecture/default.aspx?pull=/library/en- us/dnea/html/eaarchover.asp, 2002.
- [Szy98] Clemens Szyperski. Component Software: Beyond-Object Oriented Programming. Addison Wesley. Enero 1998.
- [Domain – Driven Design] Eric Evans: Domain – Driven Design: Tackling Complexity in the Heart of Software, Addison Wesley, 2003.
- [Zope26] Amos Latteier, Michael Pelletier, Chris McDonough, Peter Sabaini. The Zope Book (2.6 Edition), http://www.zope.org/Documentation/Books/ZopeBook/2_6Edition/view. New Riders, 2001

- [BibZ02] Michael R. Bernstein, Scott Robertson, and the Codeit Development Team. Zope Bible. Hungry Minds, Inc.
- [Mijailov] Mijailov, A. I.; Cherni, A. I. y Guiliarievski, R. S. Fundamentos de la Informática. Moscú: Nauka, 1973.
- [BiUCI] Rodríguez Castilla, Liuris. El sistema Integrado de Gestión Bibliotecaria: su implementación en la Biblioteca de la Universidad de las Ciencias Informáticas (BiUCI). Tesis presentada como requisito parcial para optar por el título de Bibliotecología. La Habana, 2003.
- Fowler, Martin: Enterprise Application Architecture Patterns, Addison Wesley, 2003.
- Evaluating Software Architectures: Methods and Case Studies; Paul Clements, Rick Kazman y Mark Klein; Marzo 2004.

ANEXOS

ANEXO #1. Cómo los stakeholders pueden usar la documentación.

Para seleccionar el conjunto apropiado de vistas de la arquitectura es necesario identificar los stakeholders que dependen de la documentación de la arquitectura de software, se deben entender las necesidades de información de los mismos. Se listan los roles de stakeholders de mayor importancia y como pueden usar la documentación de la arquitectura para localizar sus intereses.

Alguien nuevo en el proyecto

Leer la sección 3.1.1, para entender como está organizada la documentación, y la plantilla de vista para entender cómo están documentadas las vistas. Leer la vista general del sistema(en 3.2). Examinar las vistas de descomposición, despliegue y vista de asignación de trabajo.

Gestor de Proyecto

Para ayudar con la planeación del proyecto, concentrarse en la vista de descomposición de módulos, que ayudará a definir la asignación de trabajo. Leer la vista de despliegue para entender el ambiente de hardware que debe ser adquirido. Leer la vista de asignación de trabajo para comenzar a administrar y coordinar los equipos.

Performance Engineer (Ingeniero de Rendimiento)

Leer la vista de procesos(o communicating-processes) para entender las unidades de concurrencia potencial. Leer la vista de despliegue para entender cómo el software ha sido asignado al hardware. Leer las especificaciones de comportamiento en las vistas de Componentes&Conectores.

Analista de Seguridad

Leer la vista de despliegue para entender el ambiente físico en el cual el sistema opera.

Personal de Mantenimiento

Leer la vista de descomposición para entender las unidades de implementación que existen. Leer la vista de implementación para entender como las unidades de código son asignadas al ambiente de desarrollo.

Desarrolladores

Leer la vista de descomposición para entender las unidades básicas de software en el sistema; la vista de usos para ver como el subconjunto actual a ser desarrollado está estructurado; la vista de implementación para ver donde el software reside en el ambiente de desarrollo; la vista de capas para ver cual software les está permitido usar; y la vista de asignación de trabajo para entender las otras unidades organizacionales con las que los desarrolladores tienen que coordinar. Leer las especificaciones de interfaz para saber como usarlas.

Cliente

Leer la vista general del sistema. Consultar las vistas de descomposición, despliegue, para obtener el entendimiento de como el sistema está estructurado para llevar a cabo su misión y para obtener una apreciación del trabajo que debe ser hecho para construirlo.

Usuarios

Generalemente no son consumidores de documentación de arquitectura, pero pueden leer las especificaciones de comportamiento en las vistas Componentes&Conectores para entender como las partes del sistema se comportan.

ANEXO #2. Correspondencia con las vistas (4 + 1) de RUP.

Vista de RUP	Propuesta
Vista de casos de uso	Adoptar casos de uso para especificar comportamiento, asociado con cualquier vista.
Vista Lógica	Usar un estilo basado en módulos que muestre generalización, uso y descomposición. Usar un estilo Componente&Conector para aspectos de tiempo de ejecución
Vista de Implementación	Usar un estilo basado en módulos que contenga elementos de implementación (generalización, uso, descomposición)
Vista de procesos	Usar el estilo communicating-processes de Componente&Conector.
Vista de despliegue	Usar el estilo de despliegue del tipo de vista de asignación.

Con la propuesta que se ha aplicado en este trabajo, se pueden documentar las vistas que propone RUP, y agregar otras vistas de interés que RUP no tiene en cuenta, como son las vistas de descomposición y de capas, que son de gran importancia para describir la estructura del sistema.

GLOSARIO DE TÉRMINOS

Bajo acoplamiento: dependencia mínima entre componentes del sistema.

Catálogo: Es el mueble de madera que posee gavetas que contienen fichas ordenadas alfabéticamente por título, donde se puede chequear la existencia o no de un material en la Biblioteca Nacional.

Cohesión: Nos dice que la información que almacena un componente debe ser coherente y está en la mayor medida de lo posible relacionada con él.

Colección: Es un agrupamiento de objetos similares (libros, revistas, etc.) que se utilizan para brindar un servicio.

Fondos: Lugar donde se guardan organizadamente los materiales que se pueden utilizar para prestar servicios a los usuarios.

IEEE: Institute of Electrical and Electronics Engineers (Instituto de Ingenieros Eléctricos y Electrónicos).

Interface: un límite a través del cual dos entidades independientes se encuentran y se relacionan o comunican la una con la otra. [Cle02].

ISO: International Organization for Standardization (Organización Internacional para la Estandarización).

Máquina virtual: Dispositivo de computación abstracto; típicamente es un programa que actúa como una interface entre otro software y el hardware (u otra máquina virtual).

OMG: Object Management Group.

Sistema Integrado para Bibliotecas: en inglés ILS (Integrated Library System), se entiende por aquella solución informática capaz de gestionar las diferentes actividades que se hacen en una biblioteca: adquisición, selección y catalogación de materiales, gestión de catálogos, gestión de préstamo, etc. [falta].

Tipo de vista de asignación: Cada estilo de asignación describe la correspondencia de unidades de software con elementos de entorno (hardware, sistema de archivos o el grupo de desarrollo).

Tipo de vista componentes & conectores: Los estilos de componentes & conectores definen modelos que consisten de elementos que tienen alguna presencia en tiempo de ejecución, como son procesos, objetos, clientes, servidores y almacenes de datos; adicionalmente se incluyen como elementos los caminos de interacción, como son los enlaces de comunicación y protocolos, flujos de información y accesos a depósitos compartidos. Una vista C&C brinda una imagen de entidades en tiempo de ejecución y sus interacciones potenciales (capturadas como conectores).

Tipo de vista de módulo: En este tipo de vista, un módulo es una unidad de código que implementa un conjunto de responsabilidades. Un módulo puede ser una clase, una colección de clases, una capa o cualquier descomposición de la unidad de código. Cada módulo tiene una colección de propiedades asignadas, ejemplo: responsabilidades, visibilidad de la información y autor. Los módulos tienen relación unos con otros, ejemplo: es-parte-de, hereda-de.