Curso II: Sistemas distribuídos y redes.
6. Bibliotecas digitales.

Mercedes Martínez
Universidad de Valladolid
ESPAÑA

Índice General

Índice de Tablas

Índice de Figuras

Capítulo 1

Bibliotecas digitales

Existe un gran interés en las comunidades que tienen bibliotecas en diseñar e implementar sistemas de bibliotecas digitales que disimulen (oculten) las complejidades de los sistemas que se caracterizan por tener informaciones numerosas y dispares. Los usuarios se sienten frustrados en sus esfuerzos para descubrir recursos que sean relevantes, aprender a manejar interfaces diferentes, y buscar utilizando una variedad de lenguajes de consultas con sintaxis distintas y convenciones semánticas variadas. En muchas ocasiones el usuario tiene la sensación de que hay parte de información relevante que no ha encontrado. Además, la presentación de registros recuperados no siempre es la adecuada.

El concepto **biblioteca digital**[1] puede invocar significados diferentes para distintos tipos de personas o profesionales. Para algunos puede significar simplemente la digitalización de las bibliotecas tradicionales. Para un bibliotecario, puede significar además un nuevo modo de hacer las cosas: nuevas formas de adquirir datos, nuevos recursos, nuevas formas de almacenar y conservar la información, nuevos métodos de catalogación y clasificación. Para un profesional de la informática puede ser simplemente un sistema de información basado en texto distribuido, un conjunto de información interrelacionada y distribuída, con información hipermedia. Para un usuario de WWW es prácticamente esto último. Otra posible visión es la de un espacio en el cual la gente se comunica, comparte, y produce conocimiento y productos comerciales. Por último, para un profesional de la enseñanza lo más relevante es la disponibilidad de un soporte para realizar su actividad, que le permite dejar material a disposición de los alumnos, descubrir nuevas referencias e intercambiar información con otros colegas.

Aparte de la visión y significado que para cada cual pueda tener, una biblioteca digital es una colección de objetos que se encuentran a disposición de un conjunto de usuarios. En los primeros trabajos que se realizaron sobre bibliotecas digitales, se definía ésta como una colección de *documentos*, pero la aparición

[1] De aquí en adelante, se entenderá por "bibliotecas" las colecciones de documentos digitales, que pueden estar más o menos organizadas, y donde los elementos pueden consistir en información uniforme (ficheros de texto, referencias bibliograficas, *news*, páginas HTML, imágenes en diversos formatos) como una mezcla de todos o algunos de ellos.

de colecciones heterogéneas compuestas por documentos de texto, fotografías, audio, etc., y el interés en clasificar y buscar no sólo documentos "tradicionales", sino también objetos como direcciones (e-mail, URLs), información sobre los formatos, información sobre las colecciones, servidores, que no son documentos, si no información adicional (metainformación) ha hecho que la idea inicial se amplíe y actualmente se habla de *objetos* en vez de *documentos*.

Una biblioteca digital almacena materiales en formato electrónico y los manipula en grandes colecciones de forma eficiente. Además, en muchos casos una **biblioteca digital** es un grupo de colecciones distribuídas que el usuario ve como una única colección. Por ello, gran parte de la investigación en bibliotecas digitales tiene que ver con los sistemas de información en red.

Las ventajas que aporta la digitalización, que luego describiremos, junto a la aparición en Internet de amplias colecciones de objetos variados, hace de las bibliotecas digitales un campo de investigación muy activo, con numerosos proyectos destinados a ello e ideas que probar.

En el mundo de las bibliotecas digitales se ven envueltos varios grupos de personas, con intereses distintos:

- clientes, como comunidades científicas u otro tipo de usuarios cualesquiera,
- empresas comerciales relacionadas con la comercialización de bibliotecas digitales (editores, compañías de software),
- establecimientos donde se archivan datos, privados o públicos (bibliotecas públicas o privadas, agencias de administración públicas), e
- ingenieros informáticos y demás profesionales (departamentos de informática, centros de supercomputación, ...) que investigan en el campo.

1.1 Biblioteca física y biblioteca digital

Las bibliotecas digitales pueden ser, por la forma en que han surgido, de dos tipos: bibliotecas digitales que resultan de la digitalización de una biblioteca clásica o bibliotecas digitales que nacen directamente en formato digital. Aunque problemas como la indexación son comunes en ambos casos, existen algunas diferencias. En el primer tipo se encuentran proyectos como la digitalización del museo del Louvre y la red de bibliotecas nacional española. Ejemplos del segundo tipo son las colecciones de software relacionados con las bibliotecas digitales disponibles en Internet, o colecciones de *news*.

Una biblioteca tradicional está compuesta por elementos físicos (libros, revistas, ...), mientras que la biblioteca digital está formada por elementos digitales. Las bibliotecas digitales contienen los datos digitales que resultan de transformar los datos de la biblioteca física. Consecuencia de esto es que el mismo objeto (un libro, por ej.) puede ser percibido de forma distinta por el mismo usuario en su formato físico que en su formato digital. Por otro lado la *metainformación* existente en una biblioteca tradicional, como índices, catálogos, etc, también puede y es conveniente que aparezca cuando se digitaliza la biblioteca.

Ejemplos de colecciones físicas que se traducen en colecciones digitales son algunos libros, revistas (*Science, Computer*), películas. Unos aparecen en formato digital como un conjunto de imágenes resultantes de pasar hoja a hoja por el escáner , mientras que otros son directamente transformados a ficheros de texto, como en el caso de artículos de revistas.

En cualquier caso, en la transformación de una biblioteca física en una biblioteca digital siempre aparece el problema de decidir qué aspectos del original deben aparecer en su traducción digital y para cuáles no merece la pena hacer esta traducción. £Es suficiente una traducción ASCII de un libro? £O bien es mejor pasar por el scanner cada una de las páginas y que éstas aparezcan como fotografías?

Otro problema es que alguna de la metainformación que forma parte de la biblioteca, como la ubicación física de los libros, desaparece cuando estamos en un soporte informático. £Es conveniente preservarla de algún modo o es preferible que no sea así? En algunos casos la proximidad física de los libros puede servir de orientación al usuario que busca material sobre un determinado tema; proximidad que no percibe cuando trabaja con una biblioteca digital.

Por último, las tareas de adquisición, ordenación, catalogación en una biblioteca suelen hacerlas las personas que allí trabajan. Así pues, otro problema que debe resolverse en el proceso de digitalización es proporcionar a estas personas los útiles adecuados para que puedan seguir realizando estas tareas sobre la colección digital.

A todo esto, finalmente, se puede añadir que cuando se contempla la biblioteca desde el punto de vista de un usuario, hay un factor importante del que el usuario que se conecta desde un ordenador carece: el bibliotecario que le puede guiar al hacer una consulta, orientarle sobre dónde debería buscar, ...

Estos eran algunos problemas que se dan al digitalizar una biblioteca física. Sin embargo, una biblioteca digital tiene ventajas que la convierten en una opción muy interesante

- Una biblioteca tradicional suele presentar con problemas de espacio físico, a medida que adquiere más volúmenes. Estos problemas de espacio físico se traducen en la necesidad de nuevos edificios para ampliar la biblioteca existente. Estas ampliaciones suponen un gasto económico alto. Una biblioteca digital también requiere espacio físico para albergar las máquinas que contienen la colección. Sin embargo, hoy en día todo el mundo es consciente de que no son comparables el espacio necesario para albergar una enciclopedia y el ocupado por su equivalente CD-ROM, por ejemplo.
- Los problemas de presupuesto no permiten en ocasiones disponer de tantas copias de un ejemplar como se quisiera. En una biblioteca digital, una única copia puede ser consultada simultáneamente por varios lectores, que se encuentran geográficamente distantes. Por otro lado, no es necesario esperar a que el anterior usuario retorne el ejemplar a la biblioteca. Un único objeto puede ser accedido desde muchos puntos, y simultáneamente por muchos usuarios.
- Los libros de una biblioteca se van deteriorando con cada una de las con-

sultas. Un libro acaba deteriorándose con el paso de los años por el uso, y en ocasiones acaba roto. Un "libro" digital no se deteriora en cada una de las consultas.

- Los ejemplares de una biblioteca digital no se pueden robar.
- En la presentación de datos al usuario final, este puede reformatear el documento para visualizarlo de la forma que le resulte más cómoda (por ej. puede aumentar el tamaño de letra para verlo mejor).
- El almacenamiento digital permite ampliar el rango de material disponible. Por ej. las copias digitales de fotos frágiles no corren el riesgo de romperse, así como las copias digitales de sonido son más fiables y de mejor calidad que algunas casettes que puedan estar bastantes deterioradas.

La biblioteca digital resuleve entonces muchos problemas de mantenimiento y facilita el acceso de los usuarios a los recursos.

Los problemas de mantenimiento de una biblioteca digital son la caducidad del software utilizado y la necesidad de "refrescar" la base de datos. En cuanto al software, el mayor peligro es que los dispositivos lectores acaben quedando obsoletos. La necesidad de refrescar o actualizar periódicamente la base, puede plantear preguntas como £merece la pena actualizar hoy toda la información si dentro de unos años el software, formatos, pueden estar obsoletos? El riesgo de encontrarse con información obsoleta se reduce si se trabaja con estándares , como por ej. MARC, SGML.

Otra cuestión que no aparecen en una biblioteca física es la diversidad de los formatos de la información (.tex, .doc, .jpg), y la actualización de enlaces entre documentos.

Por último, uno de los puntos que más interés ha causado desde un primer momento es la protección de los derechos de *copyright*.

Estos últimos problemas son problemas comunes tanto a las bibliotecas resultantes de la digitalización de bibliotecas físicas como a las que surgen directamente en entorno digital, que no sufren los primeros mencionados.

1.2 Temas de investigación en bibliotecas digitales

El usario de una biblioteca digital busca un sistema que le permita hacer búsquedas eficientes, que le permita examinar la biblioteca de forma agradable y que, dentro de lo posible, le guíe en su tarea de búsqueda o exploración, del modo más cercano posible a la respuesta de un bibliotecario ante una consulta imprecisa.

Algunos de los campos de investigación en las bibliotecas digitales están directamente relacionados con el usuario final, que ve directamente los resultados. Otros se refieren a tareas de preprocesado de la información antes de que llegue

a él, que si bien tienen influencia sobre los resultados que recibe, no son directamente percibidos por él.

De forma general, estos campos, según definió NFS/ARPA/NASA en su *NSF Research Anouncement on Digital Libraries* , en 1993 son

1. **Sistemas para capturar información.** Inicialmente, la información se añadía manualmente. Es decir, el encargado de crear y mantener la colección se preocupa de añadir los nuevos objetos, buscando el punto en el que deben estar, y además informando de forma conveniente a los procesos que necesitan conocer estas actualizaciones, como por ejemplo, los indexadores. Actualmente, en muchos casos, la mayoría de este trabajo sigue recayendo en una persona. Es decir, es un trabajo manual, no automatizado.

   Algunas bibliotecas son estáticas. En este caso la captura de información puede ser simple. Basta con que cuando se adquiere un nuevo objeto, éste se añada a la colección ya existente.

   Si la biblioteca es bastante cambiante, los documentos pueden aparecer, desaparecer y/o cambiar su ubicación. La tarea de actualizar la biblioteca se convierte en una labor compleja y pesada para ser realizada manualmente. Así pues, se trata de automatizar la labor de integrar los nuevos objetos de forma coherente en la colección, y dejar constancia de estos cambios.

   Por ejemplo, un equipo de soporte técnico que mantiene una colección de consultas recibidas a lo largo del tiempo, junto con las respuestas enviadas. De esta forma, cuando se reciba una nueva consulta, lo primero que hará el técnico es buscar en la colección una consulta igual o parecida, para tener así todas las posibles respuestas que se han dado junto con la evolución del tratamiento. Esta es en si misma una colección muy cambiante, que se nutre de consultas recibidas originariamente por cada tecnico particular. Pero para que este conocimiento adquirido con el tiempo sea útil, las consultas deben incorporarse al sistema lo antes posible, y ser catalogadas de modo correcto.

2. **Categorización y organización de la información electrónica.** El proceso de clasificación de la información por temas, categorias, tipo de dato (documento de texto, fotografías, documento de audio, software), es principalmente manual. Sería deseable poder automatizarlo, pero una automatización completa es cuando menos difícil de conseguir.

   Un motivo de esta dificultad está en las diferentes visiones que tienen distintos usuarios, que resulta en que el mejor método de catalogación no coincide para distintos colectivos que acceden a la misma biblioteca. Por ejemplo, no son las mismas las categorías que distingue un informático o un documentalista cuando busca información sobre bibliotecas digitales. O para un fotógrafo puede que lo más interesante sea buscar todas las fotografías que aparecen en la revista hechas por un determinado autor durante un período de tiempo concreto, mientras que para un aficionado a la aventura sus intereses de búsqueda no irán por autores, sino por tipos de viaje, localización geográfica, etc..

3. **Búsqueda, filtrado, resumen de grandes cantidades de información**. Se trata de conseguir softwares que faciliten las búsquedas eficientes en las colecciones, como indexadores. También se intenta, dada una colección amplia de información, extraer algún tipo de información que sirva para describir el contenido de ésta.

4. **Visualización.** Además de poder buscar en una biblioteca, los usuarios normalmente suelen "navegar" por ella, descubriendo cómo está clasificada, adentrándose en subconjuntos que son más de su interés, y moviéndose desde unos objetos a otros relacionados. Facilitar este movimiento del usuario de forma que obtenga una visión global correcta de la información existente, que sea un proceso suficientemente rápido y grato, es el objetivo de los sistemas de catalogación y las interfaces.

5. **Protocolos de red y estándares** para asegurar la capacidad de las redes digitales para dar respuesta a las necesidades que plantean las bibliotecas digitales, y garantizar la interoperabilidad.

6. **Simplificar la utilización de uso de recursos en red** que están geográficamente distribuídos. Ya se mencionó antes que, por ejemplo, no es necesario tener más de una copia de un objeto.

7. **Aspectos individuales, sociales, y económicos** de las bibliotecas digitales. Dentro de este bloque se incluyen temas como la protección del derecho de *copyright*.

1.3 Bibliotecas distribuídas

Una biblioteca distribuída, como indica el propio nombre, es una colección de información que no se encuentra en único punto. Está repartida entre varias fuentes, pudiendo tener cada una de ellas su propio sistema de organización, indexación y consulta.

Cara al usuario final, el hecho de que la biblioteca esté distribuída debe ser transparente. El usuario la ve como una *única* biblioteca, con una única organización, índices únicos y métodos de consulta únicos. Debe percibirla como centralizada.

El hecho de que la biblioteca esté distribuída añade nuevas facetas. La primera es que por el hecho de estar en distintas máquinas, hay unos procesos de comunicación, intercambio de mensajes entre éstas. Esto supone que el software de bibliotecas digitales está sobre las capas de comunicaciones de una arquitectura en red.

La segunda es establecer una equivalencia entre los conceptos utilizados en las distintas bibliotecas y los conceptos utilizados por el usuario. Supone manejar transparentemente al usuario las variaciones en contenido y significado, lo cual lleva a la necesidad de interoperabilidad *semántica*. El modo en que se está trabajando para conseguir esta interoperabilidad es mediante la definición de

estándares, que por lo tanto sean comprensibles para sistemas distintos. Cada una de las fuentes debe entonces ocuparse de "traducir" los conceptos del estándar (que tienen significado) a los conceptos que maneja localmente.

Otra consideración es que la obtención de resultados en una biblioteca distribuída añade dos problemas que no existen en una única biblioteca. Primero, los documentos provenientes de distintas fuentes pueden tener estilos y estructuras diferentes, lo cual crea un problema en la presentación. Segundo, los esquemas de clasificación pueden ser distintos. Esto conlleva un esfuerzo para *mezclar* conjuntos de resultados individuales, ordenados según criterios particulares, en un único conjunto, consiguiendo que exista un criterio de ordenación común.

1.3.1 Arquitectura de una biblioteca digital

Las primeras bibliotecas digitales con las que se comenzó a trabajar estaban compuestas por documentos de texto. Una de las primeras necesidades que se planteó fué la de construir índices eficientes que permitiesen buscar en dichas bibliotecas. Sin embargo, desde estas bibliotecas sencillas se ha llegado a bibliotecas mucho mas complejas, con información heterogénea, procedente de fuentes diversas. E incluso, en muchos casos la biblioteca que el usuario final percibe está formada a su vez por varias bibliotecas, cada una con una organización de datos e índices propios y distintos al resto.

Por ello, actualmente cuando se plantea la creación y manejo de una biblioteca digital se diferencian tareas, que son las que van a determinar la arquitectura final de la biblioteca

1. **Recolección de la información.**
   Proceso en el cual se juntan los documentos que componen la biblioteca, se organiza y se crea, si es necesario, información adicional que pueda ser de ayuda en la posterior creación de índices y navegación. En esta etapa todo debe quedar listo para que la creación de los índices sea automática.

   Puede ser una etapa sencilla, que en bibliotecas simples se realice de modo manual. Cuanto más distribuída está la biblioteca y más heterogénea es, más complicado resulta este proceso. El caso más extremo es aquel en el cual la biblioteca se compone de información distribuída, cuya existencia además hay que descubrir; véase, por ejemplo, el caso de herramientas de búsqueda como Yahoo, Lycos, etc..

   Algunos problemas que se presentan en esta etapa son:

   - Las fuentes fueron inicialmente creadas con otros fines. Por ejemplo, puede existir ya información sobre legislación, pero ni su estructura ni su formato son óptimos para la construcción de una biblioteca que capaz de soportar consultas.
   - Las fuentes pueden dejar de estar disponibles.
   - Las fuentes pueden estar en formatos diferentes.

2. **Indexación y Recuperación.**
   Una vez que se dispone de la información, se trata de crear los índices y

conceptos que sirvan para hacer una navegación.

La indexación puede realizarse sobre todo el documento o pueden construirse índices únicamente con información relevante. Por ejemplo, títulos y palabras claves. Los objetos que se indexan pueden ser ficheros de texto o pueden ser también imágenes, audio, direcciones. La indexación de ficheros de texto es la faceta que más tiempo lleva estudiándose y por consiguiente, la más perfeccionada.

En caso de que no se indexe la totalidad de la información disponible, hay que determinar cuáles son los datos relevantes que deben ser objeto de indexación y crear estos *metadatos*.

Es conveniente que estos metadatos se representen de forma estándar, para así facilitar la interoperabilidad con otras bibliotecas.

Por lo que se refiere a la recuperación, se tiene que traducir la consulta del usuario a la consulta que entiende cada herramienta de búsqueda en particular. Esta traducción es básicamente sintáctica. Sin embargo, la utilización de estructuras como las de documentos SGML favorecerá una traducción no sólo sintáctica.

3. **Interpretación de resultados.**
   Los resultados que devuelve el software de indexación en una consulta se adaptan a una sintaxis y semántica propia. Para presentarlos al usuario estos resultados suelen pasar por una etapa de procesamiento. Además, los resultados obtenidos en una consulta pueden provenir de distintas fuentes, cada una con su propios criterios en cuanto a sintaxis y semántica. Antes de la presentación final al usuario hay que procesarlos para uniformarlos.

4. **Interacción con el usuario.**
   Tanto la etapa de realización de consulta como la posterior presentación de resultados de búsquedas al usuario final deben hacerse de forma lo más agradable posible para el usuario.

   Esto significa que por un lado la interfaz que el usuario encuentra para realizar su consulta debería ser lo más sencilla y cómoda para éste. Resulta tedioso y desagradable, en ocasiones, tener que enfrentarse a una interfaz distinta en cada consulta que se quiere hacer en bibliotecas distintas, aprendiendo además los criterios que cada sistema en concreto utiliza para leer sus consultas. Cuanto más obvio sea el criterio para el usuario, más probabilidades hay de que éste se sienta a gusto con el sistema. Esto, por supuesto, debe conseguirse sin que se pierda la capacidad de realizar consultas complejas, específicas de la biblioteca. Es decir, debe ser simple (en el manejo) y a la vez complejo (en las prestaciones).

   Una vez que el usuario ha realizado su consulta espera unos resultados. La forma de presentar estos resultados al usuario: criterios de ordenación, qué presentar (únicamente título, un pequeño resumen, ...) también es objeto de estudio.

   Ultimamente han aparecido estudios que demuestran que en una misma biblioteca, no todos los usuarios tienen las mismas expectativas cuando hacen una consulta, ni la misma experiencia. Normalmente, el encargado de una biblioteca es capaz de manejar y sacar el máximo partido en una

consulta donde se le pidan campos como: ISBN, autor, referencia, código de materia, etc.. Pero para un usuario normal, campos como el código de materia no sólo pueden carecer de significado, sino que además pueden desorientarle, en cuyo caso sería preferible que no apareciese.

Estas tareas, normalmente se asignan a distintos **agentes**, componentes de software que se encargan de una tarea específica. Si bien cada biblioteca puede tener su arquitectura específica, en la mayoría aparecen

- RECOPILADORES.
  Se encargan de recolectar la información que compondrá la biblioteca. Habitualmente, hacen un recorrido periódico por las distintas fuentes para reconocer los objetos de nueva creación.
- INDEXADORES.
  Con la base resultante de la etapa anterior, y la información sobre modificaciones, crean los índices y los actualizan.
- AGENTES DE CONSULTA.
  En las bibliotecas distribuídas, recogen la consulta del usuario y la redirigen a los índices oportunos.
- AGENTES DE VISUALIZACIÓN.
  Son los componentes de software que toman los resultados de las consultas y los muestran al usuario.

1.4 Metainformación

El concepto de **metainformación** o **metadato** ha aparecido repetidamente a lo largo de esta introducción. Resumiendo, los metadatos o metainformaciones son aquellas informaciones que, sin ser parte de la documentación original, aportan un conocimiento valioso para facilitar su mensaje.

Metainformaciones son:

- los índices,
- las palabras claves de un documento,
- informaciones sobre ubicación física de un documento, fecha de creación, última modificación, ...
- otros.

Estos metadatos son cada vez más importantes. Pueden servir para hacer índices basados en la semántica y no sólo en la sintaxis. Esto permitiría, por ejemplo, hacer búsquedas *por concepto*: buscar palabras que no aparecen en el texto, aunque los conceptos sí aparecen.

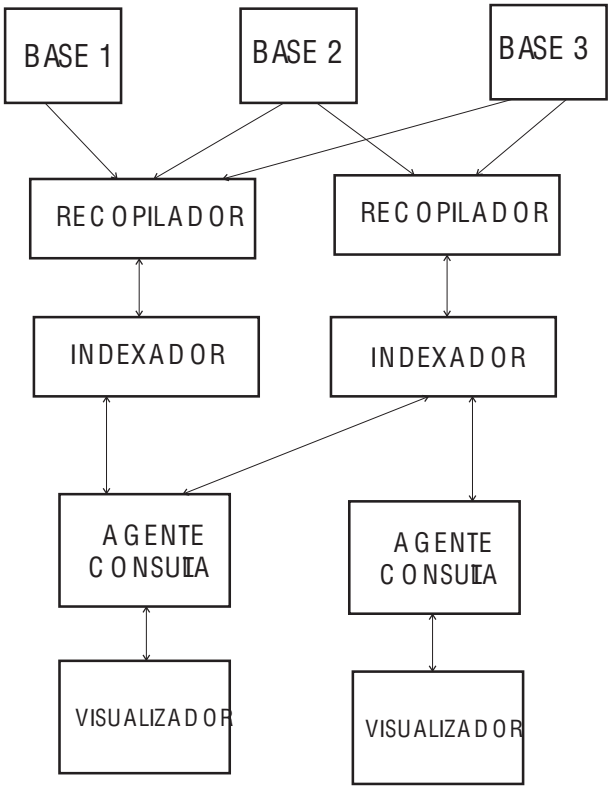

Figura 1.1: Agentes en una biblioteca digital.

También facilitan la creación de índices "diferentes", organizados por aspectos de los objetos como su procedencia, fecha. Añaden, sin embargo, un coste de almacenamiento, que puede superar en ocasiones al de los propios datos.

1.5 Estándares

En una situación ideal, clientes y proveedores de servicios, que son parte de una biblioteca digital, se crearían independientemente, atendiendo a los criterios particulares de cada implementador. Cada uno "conectaría" sus componentes en un bus de software virtual que se ocuparía de todos los problemas de interoperabilidad a nivel de protocolos. En este *bus de información*, los servicios de biblioteca traducirían de modo transparente los formatos, servicios y transacciones financieras. Si todos los servicios estuviesen conformes con un estándar, a los desarrolladores de librerías digitales les sería fácil adaptarse a esta visión. Desafortunadamente, esta convergencia de protocolos no se ha dado aún.

Beneficios que se lograrían con la utilización de estándares:

- Se facilita la comunicación o intercambio de mensajes.
- Interoperabilidad semántica. Esto significa
  - Entendimiento sobre los conceptos que maneja el usuario y las diferentes bibliotecas.
  - Entendimiento común entre los distintos softwares de recuperación que constituyen una biblioteca. Esto supone un mejor entendimiento en los *criterios de búsqueda* y en los *metadatos*.
  - Posibilidad de realizar consultas que no son simplemente sintácticas, esto es, con contenido semántico. Por ejemplo, preguntar por fechas, tipo de documento, proximidad, etc..
- Un formato de datos estándar facilita la construcción de herramientas de visualización y de consulta sobre aspectos como la estructura del documento.
- Interfaces de consultas remotas comunes. Esto evitaría al usuario la dificultad de aprender a trabajar con distintas interfaces cada vez que consulta bibliotecas distintas.

Capítulo 2

Protocolos relacionados con bibliotecas digitales

El crecimiento rápido de los recursos electrónicos, junto con la heterogeneidad de los recursos existentes dificulta encontrar recursos de interés.

Para dar servicio a los usuarios que desean acceder a la información disponible en las *bases de documentación* deben existir *interfaces gráficas* agradables para el usuario, uniformes e *intercambio de datos entre aplicaciones*.

Por debajo de esta capa visible existe otra, más técnica, transparente al usuario final. Esta capa, que debe interaccionar con la lógica de la aplicación de consulta a la biblioteca y el programa que maneja la interfaz es el **protocolo.** En el mundo de las *bibliotecas digitales* las soluciones hasta ahora desarrolladas se basan principalmente en dos protocolos: Z39.50 y HTTP, que es uno de los estándares de WWW.

La utilización de estándares, no sólo en esta capa, independiza el uso de objetos del sistema, dispositivo, o aplicación con la que se utilizan. Esto facilita en las bibliotecas digitales el intercambio de la información entre bibliotecas y fuentes diversas.

Existen estándares provenientes de campos distintos, relacionados con las bibliotecas digitales. En la comunicación y **transmisión de datos** se distinguen los protocolos relacionados con Internet y los estándares OSI. La mayoría de las bibliotecas distribuídas actuales se asientan sobre TCP/IP. Algunos de los protocolos en este campo son telnet, FTP (File Transfer Protocol), X.400 sobre TCP/IP, SMTP (Standard Mail Transfer Protocol).

Otro campo donde es muy importante el uso de estándares es el del **formato de documento**. Aquí se puede hablar de texto ASCII, formatos para imágenes: TIFF, GIF, descripción de páginas: PostScript, PDF, información estructurada: SGML, LaTeX, MARC.

Los **metadatos** que aparecen en las bibliotecas digitales también son objeto de la regulación de algunos estándars. Se normaliza la búsqueda y recuperación

sobre Internet, así como los items manejados en la comunidad bibliográfica.

Para la localización de recursos también existen estándares se emplean las URLs (Uniform Resource Locator), que están definidas para HTTP, FTP, Z39.50. Las URLs dependen de la localización geográfica del objeto, esto es, cuando el objeto cambia su ubicación, su URL cambia. Existe otro estándar para direccionar objetos independientemente de su localización, las URN (Uniform Resource Name). Más estándars dentro de este campo son WHOIS++ y CORBA.

En cuanto a los servicios de **búsqueda y recuperación**, HTTP, Z39.50, e ISO SR.

2.1 TCP/IP y OSI

Las dos arquitecturas de red más conocidas son las del modelo de referencia OSI y la del modelo TCP/IP. El motivo de que se vean aquí brevemente es que los protocolos de recuperación de información que se verán después son protocolos de la capa de aplicación, que se asientan sobre uno u otro modelo.

2.1.1 El modelo de referencia OSI

El modelo OSI está basado en una propuesta de *International Standards Organization (OSI)* para estandarizar los protocolos que se usan en distintas capas de red. Se llama **ISO OSI (Open Systems Interconnection)** porque regula la conexión de sistemas abiertos.

El modelo OSI tiene 7 capas. Los principios que guiaron su diseño son

1. Aparece una capa allí donde es necesario un nivel de abstracción diferente.
2. Cada capa realiza una función bien definida.
3. La función de cada capa se elige pensando en la definición de protocolos estándares internacionales.
4. Minimizar la información que fluye entre los interfaces de las distintas capas
5. El número de capas ha de ser suficientemente alto como para que funciones distintas no tengan que ir juntas en la misma capa innecesariamente, y suficientemente pequeño como para que la arquitectura no resulte inmanejable.

Si bien el modelo en si mismo no es una arquitectura de red ni da lugar a un estándar ISO , ISO ha sacado estándares para cada una de las capas.

A continuación se describen las 7 capas:

1. CAPA FÍSICA.
   En esta capa se realiza la transmisión de series de bits sobre un canal de

comunicación. El objetivo es evitar errores. Cuando se envía un 1, el receptor debe recibir un 1 y no un 0.

2. Capa de enlace de datos.
   Su tarea es coger un grupo de bits y transformarlo en una secuencia libre de errores de transmisión cara a la capa de red.

3. Capa de red.
   Controla la operación de la subred. Un objetivo primordial es determinar cómo encaminar los paquetes desde la fuente hasta el destino. Las rutas pueden basarse en tablas estáticas, o determinarse al comienzo del diálogo.

4. Capa de transporte.
   Su función más importante es aceptar datos de la capa de sesión, fraccionarlos en unidades más pequeñas, pasárselas a la capa de red, y asegurar que llegan correctamente al otro lado.
   También determina qué tipo de servicio dar a la capa de sesión y, en última instancia, a los usuarios de la red.

5. Capa de sesión.
   Permite que los usuarios de máquinas diferentes establezcan **sesiones** entre ellos. Una sesión sirve para que un usuario entre en un sistema remoto o para transferir ficheros entre dos máquinas.

6. Capa de presentación.
   Esta capa no se ocupa de la transferencia de bits de modo fiable de un punto a otro, sino de la sintaxis y semántica de la información.

   Un ejemplo típico de un servicio de presentación es la codificación de datos en un estándar.

7. Capa de aplicación.
   Esta capa contiene una variedad de protocolos de uso común, como los de establecimiento de un *terminal de red virtual* y la transferencia de ficheros.

2.1.2 Transmisión de datos en el modelo OSI

En el modelo OSI, el usuario envía los datos a la capa de aplicación, que le añade una cabecera (AH) al comienzo y envía el resultado a la capa de presentación. La capa de presentación realiza sus transformaciones respectivas sobre los datos y los envía a la capa de sesión. Este proceso de manipular los datos añadiendo cabeceras de control, y pasarlos a la capa inferior se repite hasta que los datos llegan a la capa física, que los envía hacia la máquina receptora. En el receptor, los datos se mueven en sentido inverso al del emisor, y en cada capa se libera a los datos de las cabeceras respectivas.

Aunque la transmisión de los datos es vertical, cada capa se programa como si fuese horizontal.

2.1.3 Modelo de referencia TCP/IP

El modelo TCP/IP es el que se utiliza en **Internet**, que a su vez aparece como una evolución de ARPANET. ARPANET surgió como un proyecto del Departamento de Defensa estadounidense, con el objetivo principal de conectar múltiples redes de modo transparente y ser capaz de reponerse de la caida de una de las partes de la red. Además, dado el contexto en que surge, con máquinas diferentes, con distintos sistemas, y softwares diferentes, la arquitectura tenía que ser flexible. Esta arquitectura con el tiempo dió lugar al modelo de referencia TCP/IP.

2.1.4 Capas del modelo TCP/IP

1. Capa Internet.
   Esta capa es responsable de que los hosts puedan enviar paquetes a una red cualquiera para que éstos viajen hasta su destino final. Se trata de una capa no orientada a conexión.

   En esta capa se define un protocolo y formato de paquetes: IP (Internet Protocol). La capa Internet es responsable del direccionamiento de paquetes. Por ello, se puede decir que esta capa es equivalente a la capa de red OSI.

2. Capa de transporte.
   Su equivalente OSI es la capa de transporte. Permite conversar a 2 entidades, en los host fuente y destino.

   Dos protocolos están definidos en esta capa: TCP (Transmission Control Protocol) y UDP (User Datagram Protocol). TCP es un protocolo fiable, orientado a conexión, que permite que una cadena de bits sea enviada desde una máquina a otra en la red sin errores. UDP es un protocolo no fiable, sin conexión, para aplicaciones que no quieren utilizar el secuenciamiento o control de flujo de TCP y prefieren utilizar control de flujo propio.

3. Capa de aplicación.
   El modelo TCP/IP no tiene capas de sesión y presentación, ya que no se consideró necesario. La capa de aplicación se encuentra sobre la capa de transporte y contiene todos los protocolos de alto nivel. Los más recientes son los de terminal virtual (TELNET), transferencia de ficheros (FTP), y correo electrónico (SMTP). A lo largo del tiempo se han añadido otros muchos, incluyendo HTTP, que permite recuperar páginas en World Wide Web.

2.1.5 Comparación de los modelos OSI y TCP

En el modelo OSI hay 3 conceptos que son imprescindibles

(a) **Servicio.**
La definición del servicio dice lo que hace la capa, no cómo acceden a ella las entidades de nivel superior ni cómo trabaja la capa.

(b) **Interfaz.**
La interfaz de una capa describe el acceso a sus servicios desde procesos de nivel superior.

(c) **Protocolo.**
Los protocolos utilizados en cada capa son propios de ella. Esto significa que dichos protocolos pueden cambiar sin que los procesos que utilizan los servicios de esta capa sean afectados.

El modelo TCP/IP no distingue claramente entre servicios, interfaz y protocolo. Por ello, los protocolos OSI son más transparente que en TCP/IP y pueden ser reemplazados de modo relativamente sencillo.

El modelo OSI se diseñó antes de que apareciesen los protocolos. En TCP/IP es justo lo contrario: los protocolos aparecieron primero, y el modelo llegó después para describir los protocolos existentes. El problema es que la descripción del modelo TCP/IP no es trasladable a ninguna otra pila de protocolos.

Otra diferencia es que el modelo OSI únicamente soporta comunicación con conexión en la capa de transporte. El modelo TCP/IP soporta comunicación con conexión y sin conexión en la capa de transporte, proporcionando así ambas opciones al usuario.

En definitiva, el modelo OSI es muy útil para entender las redes de ordenadores. Sin embargo, sus protocolos no son excesivamente populares. El modelo TCP/IP es prácticamente inexistente, pero sus protocolos son ampliamente utilizados.

2.2 WWW (HTTP)

WWW proporciona la infraestructura para acceder a documentos enlazados, desperdigados en ordenadores por toda Internet. Su enorme popularidad se debe al hecho de que proporciona una interfaz gráfica fácil de manejar para los principiantes y una enorme cantidad de información sobre casi cualquier tema.

Su desarrollo empezó en 1989 en el CERN (Centre Européen pour la Recherche Nucleaire).

Desde el punto de vista del cliente, Web consiste en una amplía colección de documentos de ámbito mundial, normalmente conocidas como **páginas**. Cada una de estas páginas puede tener enlaces a otras páginas, en cualquier punto del mundo. Enlaces que el usuario puede seguir para traer así las páginas referenciadas. El usuario visualiza las páginas con un visualizador o **browser**, como por ej. Netscape.

El servidor Web escucha en un puerto TCP, esperando conexiones de clientes (normalmente browsers), tras lo cual la conexión se ha establecido, el cliente envía una petición y el servidor le contesta. Después de esto, la conexión se cierra. El protocolo que define esta interacción es HTTP. Para cada objeto (foto, icono, etc.) de una página, el browser establece una nueva conexión TCP con el servidor para traerlo a la máquina del cliente. Es claro entonces que si

una página contiene muchos iconos, todos en el mismo servidor, realizar este proceso para todos y cada uno de los iconos no es eficiente desde el punto de vista del cliente. Sin embargo, la implementación es simple.

WWW tiene 4 conceptos que lo caracterizan, simples, pero potentes:

2.2.1 HyperText Transfer Protocol (HTTP)

Se utiliza en WWW desde 1990. HTTP es un protocolo de la capa de aplicación, adecuado para sistemas de información distribuidos e hipermedia. Es un protocolo genérico, *stateless* (sin estado), que puede emplearse para múltiples tareas. En HTTP se negocia la representación de los datos, lo cual permite que los sistemas se construyan independientemente de los datos que se transfieren. HTTP define un conjunto de reglas que tanto el cliente como el servidor deben cumplir para reconocer el tipo de los objetos con los que trabajan y el método que deben emplear para transferirlos. Además HTTP permite que los usuarios recuperen información activando enlaces hipertexto entre objetos que están almacenados en servidores distribuidos.

Se utiliza como un protocolo genérico para la comunicación entre agentes de usuario y proxies/gateways con otros sistemas Internet, incluyendo los soportados por los los protocolos SMTP, NNTP, FTP, Gopher y WAIS.

Hay 4 pasos básicos en una comunicación entre un cliente y un servidor en la arquitectura WWW: iniciación, petición, respuesta y cierre. En una comunicación típica el cliente inicia una comunicación TCP/IP y solicita un objeto, especificando la URL del objeto. Se establece entonces una comunicación con el servidor cuya dirección forma parte de la URL. La solicitud se envía con un comando GET o POST. GET pide al servidor que envíe una copia del objeto al cliente (por ej. un fichero HTML). POST sirve para que el cliente envíe datos al servidor, utilizando FORMS. Aunque es común emplear TCP para la conexión de transporte, no es un requisito del estándar.

Una vez que se ha completado la transferencia, el cliente visualiza el objeto y el servidor cierra la conexión. Esto es lo que hace que HTTP sea un protocolo *stateless*.

La respuesta del servidor consiste en

- una línea de estado, que indica la versión de HTTP que utiliza el servidor,
- un código de resultado
  - **éxito**: se ha recibido la petición, se ha entendido y es aceptada,
  - **redirección**: se pide al cliente que haga algo más para completar la solicitud,
  - **error del cliente**: la solicitud está mal formulada,
  - **error del servidor**: el servidor no ha podido completar y validar la petición,

- cabeceras opcionales. Por ejemplo, CONTENT-TYPE (tipo de objeto enviado).

2.2.2 HyperText Markup Language (HTML)

HTML es un lenguaje utilizado para crear documentos hipertexto, trasnportables de una plataforma a otra. Los documentos HTML son documentos SGML, apropiados para representar información en un rango amplio de aplicaciones. Puede representar news, mail, etc.

2.2.3 Uniform Resource Locator (URL)

Las URL sirven para direccionar doucmentos en Web. Cada URL tiene 3 partes: el protocolo (o esquema), DNS (Domain Name Server) de la página en la que se encuentra la página, y un nombre local único que identifica una página específica. Por ejemplo: *http://www.dcs.cie.uva.es/welcome.html* identfica el protocolo HTTP para la transmisión de la página, el servidor *www.dcs.cie.uva.es* y el fichero *welcome.html*.

2.2.4 Common Gateway Interface (CGI)

CGI es un estándar para que aplicaciones externas interaccionen con servidores de información HTTP. Un documento HTML es un documento estático, que siempre está en el mismo estado. Un programa CGI se ejecuta en el servidor en tiempo real, de modo que permite obtener información de forma dinámica. El servidor ejecuta programas de "modo oculto" y devuelve los resultados de la ejecución al usuario. Los programas CGI son controlados por el servidor Web. Estos programas externos se invocan a través de scripts escritos en algún lenguaje adecuado (shell scripts de UNIX, C, Perl) , y almacenados en un fichero ejecutable.

Los scripts CGI se utilizan en bibliotecas digitales para proporcionar a los usuarios diferentes servicios. Por ejemplo, en interfaces con las bibliotecas de OPACs (Online Public Access Catalogs) que permiten a los usuarios enviar consultas directamente a los catálogos, desde Web.

El usuario interacciona con el navegador Web. Escribe sus consultas en un formulario HTML. Esta consulta es transformada en una solicitud HTTP que contiene:

- método de acceso
- dirección del servidor al que se debe enviar
- la consulta propiamente dicha

Una vez que el servidor recibe la petición HTTP, ésta es ejecutada por la aplicación de recuperación de información, que es invocada por un CGI. El programa

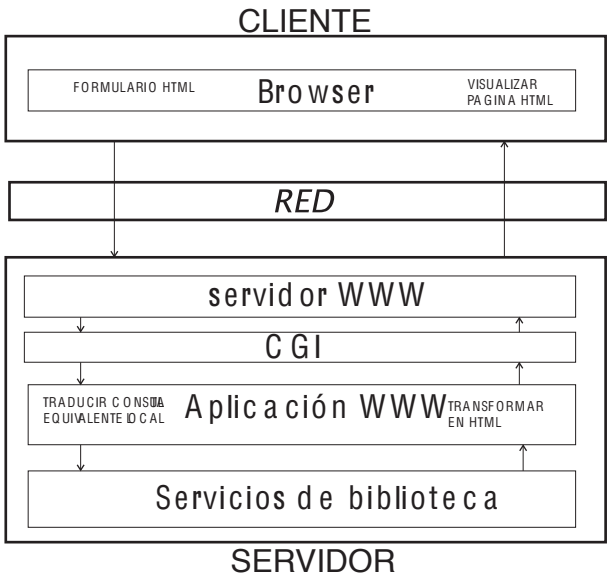

Figura 2.1: Capas de software en una biblioteca digital en WWW.

CGI recoge los resultados obtenidos por la aplicación y los devuelve al servidor Web, que se encargará de que lleguen al cliente Web dentro de su respuesta. Una vez enviada esta respuesta, el servidor cierra la comunicación. Es labor de la aplicación del cliente presentar los datos al usuario.

2.3 Protocolos de recuperación de información

Para dar servicio a los usuarios que desean acceder a la información disponible en las bases de documentación deben existir interfaces gráficas agradables para el usuario, uniformes e intercambio de datos entre aplicaciones.

Por debajo de esta capa visible existe otra, más técnica, transparente al usuario final. Esta capa, que debe interaccionar con la lógica de la aplicación de consulta a la biblioteca y el programa que maneja la interfaz es el **protocolo.** En el mundo de las *bibliotecas digitales* las soluciones hasta ahora desarrolladas se basan principalmente en dos protocolos: Z39.50 y HTTP, que es uno de los estándares de WWW.

Actualmente los dos protocolos diseñados específicamente para la recuperación de información son ISO SR y Z39.50.

2.3.1 ISO 10162/10163 (ISO SR)

ISO SR (ISO 10162) es un protocolo de la capa de aplicación, basado en el modelo cliente-servidor. El servidor proporciona acceso a una o varias bases

de información, y el cliente se conecta con el servidor para buscar en la base información de interés. Está definido como una aplicación OSI, lo cual significa que debe asentarse sobre una pila totalmente OSI.

El servidor es el encargado de hacer la correspondencia necesaria entre la consulta que recibe del cliente y el lenguaje que entiende cada base particular.

Esto es, el usuario tiene una visión uniforme de todas las bases a las que accede, aunque estas sean distintas.

SR es un protocolo en tiempo real, es decir, que la consulta que ha enviado el cliente es procesada por el servidor mientras la conexión entre el cliente y el servidor está activa.

Existen pocas implementaciones sobre este protocolo. La razón es que Z39.50-1992 (versión 2) se diseñó de forma que englobaba a ISO SR, lo cual provocó que los implementadores se centraran es Z39.50. Actualmente, en vista de esto, la intención es que ambos protocolos convergan en uno sólo.

ISO SR define 4 operaciones distintas que un cliente puede iniciar:

1. **INITIALIZE.**
   El cliente solicita al servidor el inicio de una sesión. Entre cliente y servidor se negocian los parámetros de la sesión (operaciones disponibles, tamaño máximo de registros, etc.)
2. **SEARCH.**
   El cliente busca en una o más bases disponibles desde el servidor, registros que cumplan ciertos criterios. Cuando el servidor procesa la consulta, construye un conjunto de registros resultado, al cual se le da un nombre, de modo que puede ser referenciado en posteriores operaciones PRESENT. En el resultado de la operación SEARCH, el servidor manda al cliente los registros resultado de la operación. Si la longitud de estos registros es mayor de la longitud máxima pactada durante la operación de inicialización, el cliente deberá solicitar aquellos que no haya recibido con solicitudes de tipo PRESENT.

   La sintaxis de la consulta es RPN. Los atributos que pueden utilizarse están especificados, aunque no pueden mezclarse atributos pertenecientes a conjuntos distintos en una misma consulta.

   En cuanto a los registros, el cliente puede especificar en qué formato prefiere que el servidor le envíe los registros, y si desea recibir los registros completos o no.
3. **PRESENT.**
   Esta operación sirve para que el cliente recupere registros, de uno de los conjuntos resultantes de una operación SEARCH. El cliente especifica de qué conjunto quiere registros, el primer registro que quiere y cuántos registros en total debe proporcionarle el servidor. El servidor envía los registros en la respuesta.
4. **DELETE-RESULT-SET.**
   Permite al cliente borrar uno o más conjuntos de registros resultado de búsquedas.

Capítulo 3

Z39.50

Z39.50 es un estándar ANSI/NISO, reconocido en ISO como el estándar 7498, para la recuperación de información. Surgió para resolver el problema de la recuperación en bases bibliográficas, pero ha ido ampliándose de modo que actualmente no está limitado a este campo. El protocolo es útil para la búsqueda y recuperación de información en texto plano, imágenes y multimedia. La última versión es la versión 3, de 1995.

Z39.50 es un protocolo de red: define un conjunto de reglas que precisan los formatos (tipos de datos) y procedimientos utilizados por dos ordenadores para interaccionar. El protocolo está orientado a sesión y es *stateful*, en contraste con otros protocolos Internet *stateless* como HTTP.

Es un protocolo de la capa de **aplicación**, implementado en su mayoría en entornos TCP/IP, si bien no es una exigencia del protocolo. También puede estar implementado sobre OSI (definido en el RFC1729). Z39.50 es un protocolo orientado a sesión, basado en un modelo cliente/servidor. Permite al usuario acceder a registros de bases de datos remotas especificando criterios para identificar los registros adecuados, y requerir posteriormente la transmisión de algunos de los registros identificados. Regula la interacción entre cliente y servidor. No establece ninguna norma para los datos de la base ni para el modo en que se realiza la recuperación. Tampoco dice nada sobre la interacción entre el cliente y el usuario. Especifica los atributos de los registros sobre los que se puede consultar y es muy específico en sus búsquedas.

Z39.50 unifica la sintaxis utilizada entre cliente y servidor, resuelve el problema de los interfaces de usuario diferentes, presentación de registros al usuario y aporta *inteligencia* a los clientes y servidores, que *entienden* los mensajes que intercambian.

El **cliente** u **origen**, desea consultar una (o más) bases de información para buscar y (posiblemente) recuperar información útil para el usuario. El **servidor** o **destino**, se encarga de efectuar las búsquedas y enviar al cliente las respuestas oportunas.

A través del protocolo se negocian entre el servidor y el cliente parámetros de

la comunicación antes de comenzar las consultas propiamente dichas, como

- versión del protocolo en la que van a trabajar
- tamaño máximo de los registros transferidos
- qué facilidades están disponibles durante la asociación (*scan, sort, level 1 segmentation, ...*)
- etc.

La especificación de las estructuras Z39.50 se realiza en ASN.1 (Abstract Syntax Notation 1) y la codificación de las estructuras ASN.1 en cadenas de octectos se hace en base a las reglas BER (Basic Encoding Rules).

Los mensajes enviados entre el cliente y servidor se denominan PDUs (Protocol Data Unit). El cliente construye PDUs con la información que recibe del usuario, ateniéndose a la especificación del protocolo y las envía al servidor, que a su vez recoge sus respuestas en PDUs que envía al cliente por la red.

La comunicación entre un cliente y un servidor se lleva a cabo a través de una *Z-association*. Es el cliente quien la inicia, pero pueden darla por finalizada tanto el cliente como el servidor. Durante esta asociación son válidos los parámetros negociados en la inicialización.

Implementar clientes y servidores de información con Z39.50 proporciona ventajas:

- Separa el interfaz de usuario de la información sobre servidores, herramientas y bases de datos.
- Estandariza la forma de comunicarse entre cliente y servidor.
- En el lado del servidor, no se especifican restricciones sobre el tipo de base de datos ni sobre la herramienta que efectivamente realiza la búsqueda.
- En el lado del cliente, se dialoga con un servidor *estándar*, sin preocuparse del tipo de base en la que se busca.
- Es independiente del protocolo de nivel de transporte que se encuentra por debajo.
- El usuario puede utilizar un único interfaz para formular una consulta que irá a múltiples bases o servidores, sin preocuparse del modo en que cada una de las bases maneja sus consultas.
- Interoperabilidad semántica. Cliente y servidor "entienden" la información que intercambian, lo cual les da la posibilidad de hacer transformaciones sobre los registros, previas a la presentación de datos al usuario final.

Todo esto supone en definitiva que un cliente cualquiera puede conectarse con servidores que dan acceso a bases distintas, implementadas en sistemas distintos, despreocupándose del sistema que hay en cada uno de los servidores. Cada

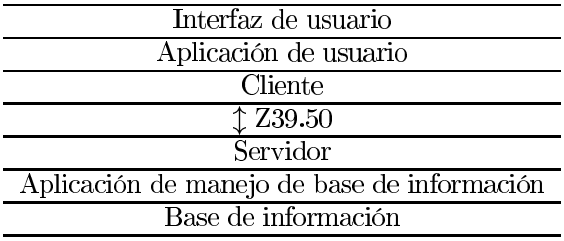

Figura 3.1: Software en aplicaciones Z39.50

cliente Z39.50, una vez recibida la información del servidor, es capaz de manipularla y presentarla al usuario del modo más conveniente en su caso particular. Esto es, si dos clientes acceden a un mismo servidor, un cliente puede presentar un interfaz al usuario tipo texto con registros completos, mientras que el segundo presenta un interfaz gráfico, con solamente parte de los registros.

3.1 Aspecto típico de una arquitectura Z39.50

Las ventajas vistas antes facilitan la división del software de recuperación realizado con Z39.50 en una serie de capas, tanto en el lado del cliente como en el del servidor.

En el **cliente**, lo primero que se encuentra es el software con el que dialoga el usuario, o **interfaz de usuario.** Es aquí donde el usuario hace su consulta, con un formato y sintaxis por él conocidos, que puede ser único para su aplicación y distinto del lenguaje de consultas Z39.50. Una vez el usuario ha introducido la consulta, la **aplicación de usuario** se encarga de procesarla y transformarla en una consulta cuyo formato y sintaxis se adapte a uno de los tipos definidos por Z39.50. Esta consulta pasa al **cliente Z39.50**, que se ocupa de establecer una conexión con el servidor, enviar la consulta e ir pidiendo los registros al servidor, todo ello ya en el "lenguaje" Z39.50 que tanto cliente como servidor entienden.

Mientras tanto, el **servidor Z39.50** espera por su parte peticiones de conexión de clientes. Una vez que recibe y acepta una conexión de un cliente, es el responsable de recoger los mensajes que éste le envía, y extraer la información que describe consultas y peticiones. Esta información es manipulada por el servidor antes de pasarla a la **aplicación de manejo de base de información,** que recibe así una consulta en un formato propio (por ej. una consulta Oracle). La aplicación procesa las peticiones que recibe del servidor Z39.50 sobre la **base de información** y devuelve los resultados al servidor Z39.50. El servidor traduce estos resultados del formato de aplicación local a alguno de los formatos reconocidos por Z39.50, y los envía en mensajes a través de la red al cliente.

Cada una de estas capas puede subdividirse a su vez en más capas. Pero esta subdivisión puede ser diferente en implementaciones distintas.

3.2 Notas sobre Z39.50

- Z39.50 no especifica la interoperación entre 2 o más servidores. Esto en principio, podría considerarse una insuficiencia, ya que obliga al usuario a conectarse separadamente con cada servidor para buscar la información cuya localización desconoce. Sin embargo, nada dice que un cierto servidor A no pueda definir una "seudo base", que en realidad representa la base de otro servidor B. Cuando el cliente se conecta con el servidor A y hace una consulta en esa base, el servidor A se conecta *transparentemente* con el servidor B (el servidor A actúa como cliente), hace la misma consulta a B, recoge el resultado y lo envía al cliente.
- Z39.50 opera en un nivel de abstracción mayor que la tecnología de bases de datos. Se trabaja con ideas abstractas como *título* o *autor*, en vez de campos concretos de una implementación. La traducción del término abstracto al correspondiente término particular se deja a cada sistema.
- Z39.50 es un protocolo para unificar no sólo los mecanismos de codificación, sino también el conocimiento semántico.

3.3 Términos importantes en el protocolo

En esta sección se describen algunos de los términos clave para trabajar con el protocolo Z39.50.

- **Punto de acceso .**
  Clave (única o no). Es el nombre de un campo (por ej. autor, título).
- **Consulta de tipo 1.**
  Uno o más puntos de acceso ligados por operadores lógicos.
- **Resultado**
  Identificación (punteros a) de los registros; no los registros en si mismos.
- **GENERIC RECORD SYNTAX (GRS).**
  Sintaxis de registros usada para transferir registros que contienen cualquier cantidad de información estructurada o no desde el servidor al cliente.
- **Abstract Syntax Notation (ASN.1).**
  Lenguaje de estructuración de datos de alto nivel (ISO 8824).
- **Protocol Data Unit (PDU).**
  Mensaje , especificado en ASN.1
- **Basic Enconding Rules (BER)**
  Reglas para traducir estructuras ASN.1 en cadenas de bytes (ISO 8825). ASN.1/BER es un método independiente de la plataforma para codificar mensajes (PDUs), que se envían entre las aplicaciones cliente y servidor. Una excepción son los registros MARC, que se transfieren según lo descrito en ISO2709 en vez de ASN.1/BER.

- **Conjunto de atributos.**
  Conjunto que define la semántica de un conjunto de atributos (términos de búsqueda) para un dominio de búsqueda (por ej. bases bibliográficas).
- **RPN.**
  Sintaxis de consulta. Subexpresiones conectadas por AND, OR y AND-NOT. Conocida como Notación Polaca Inversa por razones históricas.
- **Search Request.**
  Una o más bases de datos + consulta.
- **Access Point Clause.**
  Término + atributos.
- **Consulta de tipo-1.**
  Una o más *Access Point Clauses*, enlazadas por operadores lógicos.
- **Atributo.**
  Par: <tipo de atributo, valor de atributo>.
- **Result Set.**
  Conjunto de registros resultantes de una búsqueda, que se almacenan en el servidor y se envían al cliente cuando lo solicita.
- **OID.**
  Object IDentifier. Número que identifica un objeto dentro del protocolo. Por ej. el del protocolo Z39.50 es 1.2.840.10003 ({ ISO (1) member-body (2) US (840) ANSI-standar-Z39.50 (10003) }). Los OIDs se emplean habitualmente en los estándares OSI.
- **Elemento.**
  Parte de un registro, que se identifica con vistas a la recuperación parcial de un registro.

3.4 Etapas de un diálogo con Z39.50

Los pasos típicos de un diálogo cliente-servidor son

1. Establecer la conexión.
2. Negociar los parámetros de la conexión.
3. El usuario formula una consulta en un formato particular.
4. El cliente traduce la consulta de usuario y la pasa al servidor Z39.50. El cliente envía una petición al servidor, indicando una o más bases de datos, con una consulta y parámetros para indicar si desea recibir los registros encontrados.
5. El servidor ejecuta la búsqueda. El servidor responde con el número de registros identificados (y posiblemente alguno de los registros).
6. el cliente recupera algunos registros.
7. El cliente procesa y muestra los resultados al usuario.

3.4.1 Un ejemplo

1. Establecer la conexión.

   El cliente solicita una conexión con el servidor cuya dirección TCP es 157.88.40.189.

   El servidor acepta la conexión.

2. Negociar los parámetros de la conexión.

   El cliente propone:

   - Versión en uso del protocolo: 3
   - Servicios:
     - search (se permiten búsquedas)
     - present (se pueden recuperar registros)
   - Tamaño de los mensajes: 30*1024 bytes (tamaño máximo de un mensaje)
   - Tamaño de registro excepcional: 30*1024 bytes (no se contempla la existencia de registros de tamaño mayor al máximo propuesto)

   El servidor responde aceptando las propuestas del cliente:

   - Versión en uso del protocolo: 3
   - Servicios:
     - search
     - present
   - Tamaño de los mensajes: 30*1024 bytes
   - Tamaño de registro excepcional: 30*1024 bytes (no se contempla la existencia de registros de tamaño mayor al máximo propuesto)
   - Resultado: TRUE (acepta la conexión)

3. Formular la consulta.
   El cliente quiere encontrar apariciones del término *"Tintin"* en títulos.

4. Traducir a una consulta RPN: { Bib-1, < Use, title > } TINTIN y enviar la petición:

   - Query-type: Tipo-1
   - Databasename: "Dilib"
   - Result-set-name: "default"
   - Replace-indicator: ON
   - Small-set-upper-bound: 0
   - Large-set-upper-bound: 1
   - Medium-set-present-number: *cualquier valor*
   - query:
     - attributeSet: Bib-1 (1.2.840.10003.3.1)
     - rpn: Operando

  - attrTerm: AttributesPlusTerm
  - attributes: (< Use, title >)
  - term: TINTIN

  La base de búsqueda se llama "Dilib", la consulta es de Tipo-1 y el cliente no quiere que el servidor le mande registros en la respuesta.

5. El servidor responde con el número de registros encontrados.

   - ResultCount: 6
   - Numberofrecordsreturned: 0
   - Next-result-set-position: 1
   - Search-status: SUCCESS

6. El cliente solicita el primer registro, en formato texto.

   - Number-of-records-requested: 1
   - Result-set-start-position: 1
   - Result-set-id: "default"
   - Preferred-record-syntax: SUTRS

   El servidor se lo envía:

   - Number-of-records-returned: 1
   - Next-result-set-position: 2
   - Present-status: SUCCESS
   - Response-records:

7. El cliente muestra los resultados al usuario.

3.5 Bases de datos en Z39.50

Z39.50 define un modelo abstracto común para describir las bases de datos, de forma que cada sistema pueda hacer la correspondencia entre este modelo y el modelo local. Esto en la prática supone que no se imponen restricciones al tipo de base de información sobre la que se consulta. O sea, se puede utilizar Z39.50 con cualquier aplicación de recuperación de datos (bases de datos clásicas, herramientas de búsqueda en texto, ...).

Las bases sobre las que se realizan las búsquedas pueden contener registros compuestos tanto por información estructurada como no estructurada.

En el estándar, el término *base de datos*, se refiere a una colección de registros. Cada registro a su vez es una colección de información relacionada, tratada como una unidad. El término *registro de base* se refiere a una estructura de datos local que representa la información que hay en un registro particular.

Asociados a una base existen uno o más conjuntos de *puntos de acceso*, que pueden especificarse en una búsqueda de registros, y uno o más *elementos* que

pueden ser recuperados de un registro de la base. Un *punto de acceso* es una clave (única o no) que puede utilizarse sola o bien en combinación con otras. Un punto de acceso puede ser equivalente a un elemento, o no tener ninguna relación con ningún elemento.

La consulta se aplica a una base, especificando valores para confrontarlos con los puntos de acceso de la base.

Todo esto, en términos prácticos, significa que están definidos conjuntos de claves o puntos de acceso, por los que se puede buscar, especificando el valor que se desea encontrar en dicho campo. Una vez que el servidor ha encontrado algún resultado, puede pedirse que envíe los registros enteros o solicitar el envío de *porciones* (elementos) de registro, en vez del registro completo.

Ejemplo:

Encontrar todos los registros que tienen el valor *evangeline* en el punto de acceso *title word* AND el valor *long-fellow* en el punto de acceso *author*.

3.6 Mensajes, PDUs

Los mensajes Z39.50 intercambiados entre cliente y servidor son **PDUs (Protocol Data Units)** y se especifican en ASN.1 (Abstract Syntax Notation 1). Hay 2 partes en la creación de un mensaje:

1. Definición del contenido del mensaje en ASN.1 .
2. Codificación del mensaje en una cadena de bytes que se pueda transmitir físicamente, siguiendo las normas BER (Basic Encoding Rules).

La mayoría de las operaciones en Z39.50 se realizan mediante un diálogo consistente en mensajes de solicitud contestados con mensajes de respuesta.

Por ejemplo, en el servicio de inicialización, el cliente envía una PDU con la petición de inicialización y los parámetros, *InitRequestPDU*, y el servidor envía una PDU, *InitResponsePDU*, con la aceptación o rechazo y los parámetros efectivos en caso de que la respuesta sea positiva.

Así, siguiendo con el ejemplo de la sección 12, el diálogo se traduce en el intercambio de las siguientes PDUs:

1. *InitRequest PDU*.
2. *InitResponse PDU*.
3. *SearchRequest PDU*.
4. *SearchResponse PDU*.
5. *PresentRequest PDU*.

6. *PresentResponse PDU.*

Cada uno de los servicios descritos en la sección 18 tiene sus respectivas PDUs de inicio y respuesta, con la excepción del servicio de **cierre** de sesión, que no necesita confirmación.

Los campos que componen cada una de estas PDUs, así como sus tipos respectivos, están definidos en el estándar.

Ejemplo:

En una *SearchRequestPDU*, el cliente envía la consulta (*query*), indicando además si desea recibir o no registros en la respuesta y la longitud máxima de dichos registros (*SmallUpperBound, LargeSetLowerBound, MediumSetPresentNumber*), el nombre que debe recibir el conjunto de registros resultante de la búsqueda (*resultSetName*), la base o bases donde buscar (*databaseName*), y el formato en que prefiere recibir los registros (*preferredRecordSyntax*).

SearchRequest ::= SEQUENCE{

referenceId		ReferenceId OPTIONAL,
smallSetUpperBound	[13]	IMPLICIT INTEGER,
largeSetLowerBound	[14]	IMPLICIT INTEGER,
mediumSetPresentNumber	[15]	IMPLICIT INTEGER ,
replaceIndicator	[16]	IMPLICIT BOOLEAN,
resultSetName	[17]	IMPLICIT InternationalString,
databaseNames	[18]	IMPLICIT SEQUENCE OF DatabaseName,
smallSetElementSetNames	[100]	ElementSetNames OPTIONAL ,
mediumSetElementSetNames	[101]	ElementSetNames OPTIONAL,
preferredRecordSyntax	[104]	IMPLICIT OBJECT IDENTIFIER OPTIONAL,
query	[21]	Query,
additionalSearchInfo	[203]	IMPLICIT OtherInformation OPTIONAL,
otherInfo		OtherInformation OPTIONAL ,

Tabla 3.1: SearchRequest PDU

3.6.1 Un ejemplo

La consulta de la sección 12.1 se traduce en la siguiente secuencia de PDUs:

1. Inicialización.
   (a) InitRequest PDU

referenceId	NULL
protocolVersion	Version 3
options	Search y Present
preferredMessageSize	30*1024
exceptionalRecordSize	30*1024
implementationName	"IndexData/YAZ"
implementationId	"YAZ"

   (b) InitResponse PDU

referenceId	NULL
protocolVersion	Version 3
options	Search y Present
preferredMessageSize	30*1024
exceptionalRecordSize	30*1024
implementationName	"IndexData/YAZ"
implementationId	"YAZ"

2. Búsqueda.
   (a) SearchRequest PDU

Database-names	"Dilib"
Result-set-name	"default"
Replace-indicator	TRUE
Small-set-upper-bound	0
Large-set-lower-bound	1
Medium-set-present-number	0
Query-type	Tipo-1
Query	
attributeSet	1.2.840.10003.3.1
rpn	Operand
attrTerm	AttributesPlusTerm
attributes	(1,4)
term	TINTIN

   (b) SearchResponse PDU

Result-count	6
Number-of-records-returned	0
Next-result-set-position	1
Search-status	SUCCESS

3. Solicitud de registros.

(a) PresentRequest PDU

Number-of-records-requested	1
Result-set-start-position	1
Result-set-id	"default"
Preferred-record-syntax	1.2.840.10003.5.100 (SUTRS) (SUTRS)

(b) PresentResponse PDU

Number-of-records-returned	1
Next-result-set-position	2
Present-status	SUCCESS
records	"000000 <doc><ref>BD01 </ref><tit>Tintin au...."

3.7 Consultas

Las consultas que realiza el usuario, a través de su interfaz, deben ser traducidas a unas consultas dentro de una variedad de tipos que el estándar especifica, antes de ser incluidas en un mensaje (PDU) que el cliente enviará al servidor.

Los tipos de consultas especificadas en el estándar son

- **Tipo 0.** Unicamente se puede utilizar si existe un acuerdo previo entre el cliente y el servidor sobre la sintaxis, independiente del estándar.
- **Tipo 1.** Consulta en Notación Polaca Inversa (RPN).
- **Tipo 2.** Consulta de tipo ISO8777.
- **Tipo 100.** Consulta de tipo Z39.50, especificada en ANSI Z39.58
- **Tipo 101.** RPN extendida. Permite búsquedas por proximidad y restricciones sobre el *result set* por atributos. Si se trabaja con la versión 3 del protocolo, es idéntica a la consulta **Tipo 1.**

El estándar obliga a soportar consultas de Tipo-1, expresadas por términos de búsqueda individuales, cada uno con un conjunto de atributos, que especifican, por ejemplo, el tipo de término (tema, nombre, etc.), si está truncado y su estructura. Al igual que en el cliente es responsabilidad de la aplicación que incluye el cliente Z39.50 la *traducción* de la consulta de usuario a una consulta Z39.50, es responsabilidad del servidor hacer la correspondencia de los atributos lógicos con los de la base.

3.7.1 Consulta de TIPO-1

Una consulta de Tipo 1 es una consulta RPN (Notación Polaca Inversa). Consiste en conjunto de expresiones, posiblemente con subexpresiones, conectadas por operadores lógicos AND, OR, AND-NOT. Una expresión está formada a su vez por un conjunto de **términos de búsqueda**, cada uno de ellos con un **atributo.** Un atributo consiste en:

RPN-Query ::= Argument | Argument+Argument+Operator
Argument ::= Operand | RPN-Query
Operand ::= AttributeList + Term | ResultSetId | Restriction
Restriction ::= ResultSetId + AttributeList
operator ::= AND | OR | AND-NOT | Prox

Tabla 3.2: Gramática de consulta TIPO-1.

- conjunto de atributos al que pertenece,
- valor de atributo.

La gramática de este tipo de consulta está definida por el estándar.

3.7.2 Conjuntos de atributos

Cada uno de los atributos asociados con un término de búsqueda pertenece a un conjunto de atributos concreto, cuya definición está registrada, es decir, asignada a un único y globalmente reconocido *attribute-set-id*.

Un atributo se identifica por el OID del conjunto al que pertenece y: tipo de atributo e identificador dentro del tipo de atributo. Por ejemplo < Use, title > es un atributo perteneciente al conjunto *Bib-1*.

Z39.50 define los conjuntos de atributos siguientes:

Bib-1 Atributos bibliográficos

Exp-1 Atributos para la base Explain

Ext-1 Atributos de servicios extendidos

CCCL.1 Conjunto de atributos CCL (Common Command Language)

GILS Government Information Locator Service

STAS Scientific and Technical Attribute Set; incluye a *Bib-1*

El conjunto *Bib-1* es ampliamente utilizado, y es probablemente el más elaborado de todos.

3.7.3 Conjunto de atributos Bib-1

En el conjunto Bib-1 se distinguen **tipos de atributos**:

- Use
- Relation
- Position

- Truncation
- Structure
- Completeness

Cada tipo de atributos incluye un **rango de atributos**

- Use
  - Personal name
  - Title
  - ISBN
  - LC card number
  - etc.
- Relation
  - less than
  - less than or equal
  - equal
  - stem
  - etc.
- Truncation
  - left
  - right
  - do not truncate
  - etc.

Cada atributo Bib-1 consiste en un par *<tipo-atributo, valor>*. Por ejemplo: *<Use, Author>*.

3.7.4 Un ejemplo

Con lo visto, se puede volver sobre la consulta del ejemplo:

Buscar las apariciones del término "Tintin" en títulos.

La consulta RPN equivalente *{ Bib-1, <Use, title> } TINTIN* está compuesta en este caso por un único operando. Este operando consiste en la secuencia LISTA DE ATRIBUTOS + TERMINO. El TÉRMINO es "Tintin" y la lista de atributos la forman

1. identificador del conjunto de atributos (Bib-1)
2. un par < Tipo, valor > (< Use, title >).

3.8 Registros

El resultado de una búsqueda que ha tenido éxito es un conjunto de registros. Estos registros se conservan en el servidor, formando **conjuntos resultado** *result sets*,disponibles para que el cliente solicite todos o algunos de esos registros en el momento que desee.

Si ha habido algún problema, lo normal es que el servidor envíe registros de diagnóstico al cliente informándole sobre el problema que ha ocurrido. Así se distinguen 2 tipos de registros que un servidor puede enviar al cliente

- Registros de datos
- Registros de diagnóstico

3.8.1 Registros de datos

Los registros de datos son los registros encontrados en el servidor por la aplicación de consulta de la biblioteca. Cada consulta genera un **result set**.

Un *result set* es la identificación (punteros) de los registros. Este conjunto es local al servidor y sirve para referenciar los registros en consultas posteriores. Cuando el cliente lo solicita, el servidor le envía los registros. La estructura de un *result set* es una lista ordenada, con un nombre, donde cada elemento tiene 3 componentes:

1. un número correspondiente a la posición del triplete en la lista,
2. un nombre de base de datos,
3. un identificador único (con significación local únicamente) de un registro de la base de datos referenciada en el campo anterior.

Durante la fase de transmisión de registros, éstos pueden **segmentarse**. En la versión 2, una respuesta se limita a un único mensaje; el servidor intenta que los registros solicitados quepan dentro del mensaje, y en caso de que no sea posible, mete tantos registros como le es posible. Es decir, si el cliente quiere recuperar 10000 registros, lo que hace es pedirlos todos la primera vez, comprobar cuántos recibe en el primer envío e ir pidiéndolos en tandas del número de registros encontrados en la primera recuperación. El servidor tiene que esperar a que el cliente le envíe la petición para cada uno de los grupos de registros. En la versión 3, el servidor puede responder a una petición de recuperación de registros con múltiples mensajes consecutivos sin esperar solicitudes del cliente. Esta es segmentación de *nivel 1*.

Un problema aún más serio se da cuando *un único* registro es demasiado grande para caber en un único mensaje. La versión 3 permite segmentar un registro en varios mensajes. Esta es la segmentación de *nivel 2*.

Tanto el cliente como el servidor tienen la opción de decidir qué nivel de segmentación permiten, o si no permiten ninguna (versión 2).

Otros puntos interesantes en la recuperación de registros son:

- Un registro de una base puede incluir más de un documento. El cliente puede por tanto, recuperar un único documento en vez del registro completo.
- El cliente puede recuperar una porción de un documento, lógica o física. Por ejemplo, páginas concretas, un capítulo, todas las imágenes del documento.
- El documento puede estar disponible en múltiples formatos (SGML, texto, PostScript), en múltiples lenguajes, y otras variantes. El cliente puede interesarse en qué variantes existen de un documento, así como en la información de alguna de esas variantes (por ej., el tamaño del documento en formato PostScript). Además, puede recuperar el documento en la variante que más le interesa.
- Asociado con un documento, y para una determinada búsqueda, puede haber *hits*: punteros a términos (dentro del documento) relevantes para la búsqueda. El cliente puede solicitar estos *hits* para así localizar rápidamente las porciones más interesantes.

El protocolo soporta varios formatos de registros:

- **MARC**
  Se admiten todos los registros MARC nacionales (Danmarc, Finmarc, etc.)
- **Explain**
  Sintaxis para registros de la base Explain
- **SUTRS**
  Simple Unstructured Text Record Syntax, para registros en texto plano
- **OPAC**
  Registros OPAC (Online Public Access Catalog).
- **Summary**
  Referencias bibliográficas a nivel de sumario
- **GRS-1**
  Generic Record Syntax. Sintaxis para información no estructurada
- **Extended services**
  Sintaxis para registros de la base *Extended services*

Cada uno de los formatos está registrado con un OID, que debe acompañar a los registros devueltos por el servidor. Los registros de diagnóstico también deben estar acompañados por un OID.

TIPO TAG	VALOR	CONTENIDO
2 generica	1 (title)	"John Dee"
1 meta	16 (dateOfLastMod)	"19950325"
3 local	"Fotografía"	(NULL)

Tabla 3.3: Ejemplo. Registro GRS-1.

3.8.2 Registro GRS-1

Los registros GRS-1 (Generic Record Syntax 1) sirven para enviar registros formados por información estructurada y no estructurada.

Un registro genérico (GRS-1) está compuesto por uno o más elementos organizados jerárquicamente. Un elemento es un componente del registro. Cada elemento tiene una etiqueta (*tag*) o identificador único. Estas etiquetas pueden ser numéricas o *strings*.

Se distinguen 3 tipos de etiquetas

- **Numéricas**
  Tanto el cliente como el servidor comprenden el significado del elemento asociado
  - **Metatags**: proporcionan información sobre el registro.
  - **Genéricas**: para información general.
- **Strings**
  Proporcionan información útil para el usuario. El cliente no entiende su significado.

En el registro de la tabla 3 existen 3 campos distintos: título, metainformación sobre el registro (fecha de la última modificación) y una fotografía, que es un fichero gráfico. En este caso, el CONTENIDO del tercer elemento del registro es nulo. El fichero gráfico no se envía si es suficiente con el resto de la información. Si en base a la información recibida, el usuario tiene interés, puede solicitarlo posteriormente.

3.8.3 Registros de diagnóstico

Z39.50 registra un conjunto de registros de diagnóstico, **bib-1**, y un formato, **diag-1**.

Un registro de diagnóstico genérico, según lo define el estándar, bajo la versión 2 (y la 3 si se considera oportuno) responde al siguiente formato:

DefaultDiagFormat ::= SEQUENCE {

1.2.840.1000003.3.1	2	"Error del sistema"
(*OID de Bib-1 *)	(* temporary system error *)	

Tabla 3.4: Ejemplo. Registro de diagnóstico.

diagnosticSetId OBJECT IDENTIFIER ,
condition INTEGER,
addinfo VisibleString }

3.9 Operaciones

La versión 2 (1992) permite una serie de operaciones básicas, todas ellas soportadas también por la versión de 1995 (versión 3). El cliente puede enviar una búsqueda, indicando una o más bases donde buscar, e incluyendo una consulta así como parámetros que determinan, entre otros, si se deben enviar algunos de los registros encontrados en la respuesta a este servicio. El servidor responde con el número de éxitos y, posiblemente, con algunos registros. El cliente puede después solicitar al servidor que le envíe algunos de los registros. Los registros encontrados en una búsqueda forman un **conjunto resultado** (*result set*), de tal modo que el cliente puede referirse a los registros por su posición dentro de este conjunto.

Además de estas operaciones mínimas, hay otras posibilidades que pueden ser opcionalmente soportadas por un servidor:

- El cliente puede especificar un *element set* indicando qué elementos (componentes) quiere obtener en aquellos casos en que no desee recibir registros completos.
- El cliente puede indicar una sintaxis (*preferred syntax*) para los registros de respuesta (USMARC, texto, etc.)
- El cliente puede *nombrar* un conjunto resultados para referenciarlo posteriormente por este nombre.
- El cliente puede *borrar* un conjunto resultado de una consulta previa.
- El servidor puede imponer *control de acceso* al cliente, pidiendo identificación antes de procesar una petición.
- El servidor puede proporcionar *control de recursos* enviando un informe de estado (solicitado o no por el cliente). El servidor puede suspender el proceso y permitir al cliente decidir si continuar o no.

Todo esto se refleja en 8 operaciones reconocidas en el estándar:

1. *Init*
2. *Search*

3. *Present*
4. *Delete*
5. *Scan*
6. *Sort*
7. *Resource_report*
8. *Extended_services*

Sólo el origen puede iniciar una operación. Una petición que inicia una operación se llama *initiating request* y la respuesta que la termina *terminating response*. La operación consiste en la *initiating request* y la *terminating response*, junto con cualquier mensaje que haya en medio. Cada una de estas operaciones tiene algún servicio equivalente.

Los servicios Z39.50 se realizan mediante el intercambio de mensajes (PDUs) entre el cliente y el servidor. Los mensajes pueden ser *solicitudes* o *respuestas*. Los servicios pueden ser *confirmados*, *no confirmados*, o *condicionalmente confirmados*.

Un servicio *confirmado* es aquel donde una *solicitud* siempre va seguida por una *respuesta*. Por ejemplo, una búsqueda siempre debe recibir una respuesta. Un servicio *no confirmado* consiste en una *solicitud* que no está seguida de una respuesta (ej: *Segment*). Un servicio *condicionalmente confirmado* es aquel que puede ejecutarse bien como un servicio confirmado o no confirmado. Consiste en una solicitud que puede estar o no seguida de una respuesta (por ej. *Resource-control*).

3.10 Facilidades y servicios

Una **facilidad** es la agrupación lógica de uno o más servicios. Hay 11 facilidades. La mayoría consisten en grupos lógicos de servicios. La implementación de algunos servicios es opcional. Sin embargo, una implementación mínima debe tener por lo menos los servicios

1. *Init*
2. *Search*
3. *Present*

1. **Initialization Facility**.
   Tiene un único servicio:
   INIT SERVICE.
   Servicio iniciado por el origen para establecer una conexión. Confirmado. El cliente sugiere parámetros para la conexión como tamaño de buffer, versión del protocolo que se quiere utilizar, etc. El servidor responde con la aceptación o no de la solicitud, y los parámetros que de hecho se van a utilizar.

2. **Search Facility**.
   SEARCH SERVICE.
   Servicio iniciado por el origen para establecer una búsqueda. Confirmado. El cliente envía una consulta al servidor. El servidor crea un conjunto de resultados (*result set*) que representa los registros identificados en la consulta, que mantendrá, aunque no los envíe en la respuesta. La respuesta incluye el número de registros encontrados que cumplen los requisitos, y opcionalmente (si el cliente lo ha solicitado) algunos registros.

3. **Retrieval Facility**.
   PRESENT SERVICE.
   Servicio iniciado por el origen para comenzar una operación Present. Confirmado. El cliente solicita al servidor que le envíe algunos registros específicos, dentro de un conjunto resultado de una búsqueda anterior. Los registros deseados se especifican mediante su posición dentro del *result set*. La respuesta pueden consistir en los registros solicitados, o bien en *registros de diagnóstico*.

   SEGMENT SERVICE.
   Servicio inciado por el destino durante una operación *Present*. No confirmado. Este servicio se añadió en la versión 3. Es uno de los servicios opcionales. Permite que el cliente reciba los registros que ha solicitado al servidor en múltiples PDUs (o sea, que la respuesta se fraccione en varias PDUs). Los registros son transmitidos en sucesivas PDUs de tamaño limitado.

4. **Result-set-delete Facility**.
   DELETE SERVICE.
   Servicio iniciado por el origen para comenzar una operación *Delete*. Confirmado. El cliente solicita que se borre uno o más conjuntos resultado. El servidor manda una respuesta confirmando el estado de la operación.

5. **Browse Facility**.
   SCAN SERVICE.
   Servicio inciado por el origen para comenzar una operación *Scan*. Confirmado. Permite al cliente solicitar términos de índice. La respuesta consiste en una lista de términos que comienzan por el término que proporciona el cliente.

6. **Sort Facility**.
   SORT SERVICE.
   Servicio inciado por el origen para comenzar una operación *Sort*. Confirmado. El cliente solicita que el servidor ordene un conjunto de resultados. La respuesta consiste en el estado de la operación.

7. **Access Control Facility**.
   El servidor solicita autentificación (por ej. *password*) después de suspender la solicitud del cliente. El cliente determina en su respuesta si quiere dar por finalizada la operación.
   ACCESS-CONTROL SERVICE.
   Servicio iniciado por el destino. Confirmado. No inicia ninguna operación.

8. **Accounting/Resource Control Facility**.
   RESOURCE-CONTROL SERVICE.
   Servicio inciado por el destino. Confirmado condicionalmente. No inicia ninguna operación. El servidor solicita información sobre la utilización de recursos. El cliente determina si la operación suspendida debe darse por finalizada.

   TRIGGER-RESOURCE-CONTROL SERVICE.
   Servicio iniciado por el origen durante una operación. No confirmado. El cliente solicita que el servidor cancele la operación en marcha o que el servidor le envíe una solicitud de control de uso de recursos.

   RESOURCE-REPORT SERVICE.
   Servicio iniciado por el origen para comenzar una operación *Resource-report*. Confirmado. En la respuesta del servidor se detalla la utilización de recursos hasta el momento (por ej. dinero gastado).

9. **Explain Facility**.
   No tiene ningún servicio especial. Usa los de las facilidades *Search* y *Retrieval*. Sirve para conocer datos sobre el servidor, como por ejemplo las bases disponibles, atributos (combinaciones de atributos) soportados, sintaxis de registros. También proporciona descripción de las bases de datos. Es un servicio hasta ahora poco implementado. Alguna de la información es transparente para el cliente, pensada para ser visualizada directamente por el usuario del cliente; es la designada como *general features*.

   Si el servidor proporciona información sobre él mismo y sobre las bases de datos a las que permite acceder, lo hace a través de la facilidad *Explain*. Hace falta para ello una base de datos especial en el servidor llamada *Explain-1*. Esta base es vista por el cliente como una base más, pero sus registros pertenecen siempre al tipo *Exp-1*. El cliente puede acceder a la base *Explain-1* a través de los servicios *Search* y *Present*.

10. **Extended Services Facility**.
    Permite acceso a servicios fuera del protocolo.

    EXTENDED-SERVICES SERVICE.
    Servicio iniciado por el origen para comenzar una operación *Extended-services*. Confirmado. Algunos ejemplos específicos de operaciones que pueden solicitarse aquí son: salvar un conjunto resultado, ordenar un documento o actualizar una base de datos.

11. **Termination Facility**.
    CLOSE SERVICE.
    Iniciado por el origen o por el destino. Confirmado. No comienza ninguna operación. Permite terminar todas las operaciones que están activas.

3.11 Requerimientos mínimos de una implementación Z39.50

Cualquier sistema que implemente Z39.50 debe cumplir unos requisitos mínimos

1. El sistema debe ser, bien el *cliente*, bien el *servidor*.
2. Soportar los servicios *Init*, *Search* y *Present*.
3. Entender la sintaxis ASN.1.
4. Trabajar con consultas *Tipo-1*.
5. Implementar (como mínimo) la versión 2 del protocolo.
6. Ajustarse a las normas descritas en el protocolo.

3.12 Evolución del protocolo

Z39.50 se desarrolló a principios de los 80 como un modelo teórico. La primera versión oficial apareció en 1988. En esta versión era muy complicado hacer una implementación que pudiese interoperar con otras. Fue adoptada por WAIS (Wide Area Information Service), aunque realmente no interaccionaba con otra cosa que no fuese WAIS. En la versión 2, Z39.50-1992, se demostró que la interoperabilidad era posible.

A partir de la versión 2, la evolución de Z39.50 está fuertemente influenciada por las opiniones de un grupo de implementadores, que forman el ZIG (Z39.50 Implementor's Group).

En la versión 2 se percibe la influencia del trabajo de WAIS sobre bases textuales y la necesidad de dar cabida a proyectos como SGML.

El trabajo para conseguir una tercera versión comenzó en 1991. En 1992-1993 se lanza un programa, *Z39.50 Interoperability Testbed*, para potenciar implementaciones de Z39.50 sobre TCP/IP dado que está mucho más extendido que el modelo OSI. La versión 3 aparece finalmente como hoy se conoce en 1995. Esta versión incluye la versión 2.

En las primeras versiones el cliente tenía que pedir al servidor uno a uno los registros que éste había encontrado en su búsqueda. A partir de la versión 3 esto se ha mejorado, y el servidor puede enviar al cliente varios grupos de registros en una respuesta.

La versión 3 permite a los programadores realizar programas cliente y servidor muy versátiles. Sin embargo, aún exiten muchas aplicaciones basadas en la versión 2.

3.12.1 Desarrollo actual

Varios grupos han desarrollado **Profiles**: particularizaciones del estándar para comunidades de implementadores con necesidades comunes. Algunos ejemplos son:

- GILS (Government Information Locator System),

- *Museum Interchange Profile*,
- *Digital Collections*,
- WAIS,
- Datos geográficos.

Desde otoño de 1995 se está desarrollando la versión 4, aunque de momento sólo hay una versión en etapa de prueba, incompatible con la versión 3. Las aportaciones que hay hasta el momento son

- Con el objetivo de dar cabida a las tecnologías que ofrecen resultados clasificados por relevancia empleadas en algunas herramientas comerciales, el ZIG ha definifido una nueva clase de consulta: **Z39.50 Ranked List Query (Tipo-102)**.
- Se trabaja sobre extensiones SQL para Z39.50. Las ideas más importantes en este campo son añadir un nuevo tipo de consulta, **Tipo-SQL**, que soporte SQL3, y permitir la transmisión de registros genéricos sin necesidad de etiquetar cada campo.
- Aparece el concepto de **encapsulación**. La encapsulación supone que en cualquier PDU que inicia una operación, el cliente puede incluir, en el campo *otherInfo*, una o más PDUs, incluyendo PDUs de inicio de operación (salvo *Init*). Esta opción se negocia durante la inicialización.

3.13 Algunas implementaciones con Z39.50

- Proyecto Mann Library (Cornell). Se utiliza el protocolo Z39.50 para permitir un interfaz común de usuario para las búsquedas bibliográficas, y dar la opción a los usuarios de buscar en varias bases de datos simultáneamente.
- YAZ (Index Data). YAZ es un software para desarrollo de aplicaciones con Z39.50. Permite utilizar TCP/IP u OSI. Proporciona tres módulos al implementador
  - Módulo ASN.1,
  - Módulo de codificación ASN.1/BER,
  - Módulo para la transmisión de datos por la red.
- *U.S. National Library of Congress* . Se pueden consultar sus catálogos a través de un gateway WWW-Z39.50 para acceder a sus catálogos.
- ISITE (CNIDR). ISITE consiste en 3 módulos:
  - *Isearch*, una herramienta de indexación.
  - *Iindex*, herramienta de búsqueda.
  - *Isite*, que permite el acceso a las anteriores a través de WWW.

- ARCA (Access to Remote Catalogues by implementing SR functions). ARCA es un proyecto financiado por el *European Libraries Programme*. La finalidad es implementar un servidor Z39.50 para OPACs.
- OCLC (Online Computer Library Center). OCLC ha implementado servidores y clientes Z39.50. Proporciona funciones de libre distribución para la implementación de aplicaciones Z39.50. Actualmente tiene un proyecto, *SecondFish*, donde emplea Java para implementar Z39.50.

3.14 Organizaciones relacionadas con Z39.50

- *U.S. Library of Congress (Z39.50 maintenance agency).*
- *Z39.50 Implementor's Group (ZIG).*
- *National Information Standards Organization (NISO).*
- *American National Standards Institute (ANSI).*
- *International Standards Organization (ISO).*

3.15 Z39.50 y OSI SR

A pesar de su origen por separado, Z39.50-1992 se diseñó para englobar el protocolo ISO SR. ISO SR es un subconjunto de Z39.50-1992 sin servicio de control de recursos ni de acceso.

Existen muchas más implementaciones sobre Z39.50 que sobre ISO SR. Los esfuerzos de desarrollo están centrados actualmente en Z39.50. Las implementaciones basadas en Z39.50 son interoperables con implementaciones ISO SR. Se espera que finalmente exista un único protocolo, reconocido por OSI.

3.16 Z39.59 y WAIS

WAIS (Wide Area Information Servers) es una aplicación para realizar búsquedas en Internet. WAIS permite búsquedas en texto. Fue una de las primeras implementaciones sobre Z39.50-1988.

Los objetivos que motivaron la aparición de WAIS fueron proporcionar acceso flexible a documentos en texto, facilitando la tarea de búsqueda al usuario. Se trataba de permitir al usuario encontrar y seleccionar bases de datos entre la amplia gama existente, sin que éste necesite conocer la configuración de cada una de las bases existentes. La organización de las respuestas obtenidas queda como tarea para la máquina del usuario. Además de lo ya mencionado, WAIS proporciona una interfaz *única* al usuario. WAIS permite al usuario especificar **términos** por los cuales buscar. WAIS devuelve los resultados en orden de

relevancia. Se puede buscar en WAIS usando un cliente WAIS, via Gopher o WWW y via telnet con un cliente de libre distribución.

Se distinguen 2 tipos de implementaciones de WAIS: las comerciales y las de uso público. La distribución de uso público es conocida como *freeWAIS* y es distribuída por CNIDR (Clearinghouse for Networked Information Discovery and Retrieval).

El usuario compienza su consulta buscando en un directorio central, que es un servidor WAIS. Esta primera consulta es la que le permite localizar los servidores WAIS donde existe información sobre el tema que le interesa.

WAIS tiene las facilidades INIT y SEARCH. La búsqueda la inicia el cliente, con una *SearchRequest PDU*, y una consulta *Tipo-1*. La respuesta contiene una lista de "*WAIS-Citations*": resultados ordenados por afinidad con la consulta.

El cliente solicita el envio de registros con una *SearchRequest PDU* y una consulta *Tipo-1*.

- En Z39.50 es posible especificar que clase de término de búsqueda (autor, título, tema, etc.) se quiere emplear y qué operadores booleanos. En WAIS, un témino de búsqueda es una cadena de bits, o el conjunto de términos entre los que se puede escoger es muy limitado.
- Z39.50 permite la especificación de sintaxis de presentación de registros alternativas: distintas versiones de los formatos MARC, texto no estructurado, Postscript, etc., según especificación del usuario. Los clientes WAIS no "entienden" la estructura de los registros presentados.
- Z39.50 permite la presentación no sólo del registro completo, sino también de campos específicos, como por ej. autor o título.
- Z39.50 no impone que los resultados de una búsqueda estén ordenados.

sectionZ39.50 y JAVA

3.17 Z39.50 y WWW

Tanto World Wide Web como Z39.50 está pensados para facilitar el intercambio de información en entornos distribuidos. A su vez, también ambos se basan en un modelo cliente/servidor: un cliente solicita a un servidor cierta información.

En el entorno de bibiotecas distribuídas se propone en algunos casos la utilización de HTTP en vez de Z39.50. Esto puede causar confusión, por ejemplo, cuando uno se adentra en este tipo de problemas y se pregunta qué utilizar para desarrollar su sistema. Sin embargo, existen diferencias entre ambos que pueden ayudar a clarificar una decisión de este tipo:

- **HTTP es amistoso.**
  Proporciona al usuario páginas con un aspecto agradable, es fácil de instalar y es sencillo obtener un interfaz muy grato para el usuario.

- **HTTP es simple, Z39.50 es complejo.**
  HTTP es simple. Es un protocolo sencillo, con pocos conceptos, pero potentes. Esto hace que una vez más sea sencillo hacer implementaciones sobre HTTP. Z39.50 es un protocolo complejo, que requiere un cierto aprendizaje o familiarización para ser capaz de desarrollar algo interesante. Básicamente HTTP permite traer documentos vía una URL. Su potencia reside en la combinación de HTML y CGI, o más recientemente, también en la combinación de HTML y JAVA. Para el desarrollador de software, HTTP ofrece texto legible, que se transporta por la red, mientras que en Z39.50 el cliente y el servidor se intercambian PDUs ilegibles.

- **HTTP es general, Z39.50 es un protocolo especializado.**
  HTTP es un protocolo de propósito general, Z39.50 es un protocolo específico. Z39.50 está pensado para faciltar la búsqueda y recuperación de datos en bibliotecas digitales. De hecho, inicialmente, estaba orientado al problema de las bases bibliográficas.
  El cliente Z39.50 es "inteligente": tiene un conocimiento semántico dentro del campo de las bibliotecas. Recibe los registros en un *formato* estándar, que entiende, lo cual le sirve para manipularlos antes de entregarlos al usuario final.

- **HTTP es pasivo, Z39.50 es activo.**
  cuando se trabaja con HTTP, el *browser* es pasivo. Es decir, el cliente HTTP recibe aquello que le envía el servidor, texto, botones, gráficos, y se limita a mostrarlo al usuario tal cual lo recibe.
  El cliente HTTP tiene que "pedir" al servidor un formulario de consulta, que será el que rellene el usuario. El cliente Z39.50 uede preparar consultas complicadas sin consultar al servidor.
  Además, un cliente Z39.50 puede "decidir" cómo (formato) quiere que le envíe el servidor los resultados de una consulta.
  Por último, es el propio cliente Z39.50 quien decide cómo va a presentar el servicio al usuario y cómo va a visualizar los resultados. HTTP es un mero transporte en la comunicación.

- **Z39.50 resuelve el problema de un interfaz común.**
  con HTTP no se resuelve el problema de un interfaz común. La razón tiene que ver precisamente con el punto anterior. Si un usuario (a través de un cliente HTTP) se conecta con servidores distintos, que tienen un interfaz de consulta y muestra de resultados distintos, el cliente HTTP no puede hacer más que trasladar al usuario cada uno de los interfaces tal como los recibe de los servidores respectivos. El cliente Z39.50 entiende la información que recibe. Es capaz de presentar un interfaz de consulta común al usuario para todas las bases que consulte, y también es capaz de interpretar las respuestas recibidas por cualquier servidor Z39.50, puesto que todas se ajustan al protocolo. Esto le permite presentar las respuestas en un único interfaz de salida.

- **HTTP es *stateless*, Z39.50 es *stateful*.**
  Cuando un cliente Z39.50 inicia una asociación con un servidor Z39.50, ésta se mantiene hasta que uno de ellos expresa su deseo de darla por finalizada con un **close**. Así, durante toda la asociación se mantiene una

"memoria" de lo que ha ocurrido previamente; por ejemplo, gracias a esto se pueden mantener los *result sets* en el servidor, y el cliente puede hacer distintas operaciones sobre ellos. Sin embargo, con HTTP, una vez que el servidor ha enviado los datos resultantes de procesar una petición de un cliente, inmediatamente se cierra la conexión. Es decir, si el cliente quiere seguir trayendo información del servidor debe reiniciar nuevamente la conexión con el servidor. Además, en esta nueva conexión el servidor no "recuerda" nada sobre los resultados obtenidos anteriormente.

- **Z39.50 permite consultas acerca del servidor y la base.**
  Z39.50 permite "aprender" sobre el servidor. Puede conocer información general sobre el tipo de documentación que mantiene el servidor, cómo le permite consultarla, por qué campos puede buscar en la base. Esta exploración es previa a cualquier consulta y puede ayudar al usuario a definir mejor su consulta a tenor de la información recibida. Las facilidades mediante las cuales Z39.50 ofrece esta posibilidad son **Explain** y **scan**.

- **Z39.50 aporta contenido semántico.**
  Un browser HTTP no recibe información estructurada, sino como texto, le aporte contenido semático, útil para su manipulación antes de la visualización. Esta falta de conocimiento se "traslada" al usuario, que ve colecciones distintas como un conjunto de datos indiferenciados.

La propuesta más extendida es simultanear la utilización de HTTP y Z39.50. Existen *gateways* HTTP-Z39.50, que permiten ofrecer más de un recurso a través de la misma interfaz y que el usuario acceda a Z39.50 utilizando un browser familiar. El problema de estas herramientas es que se pierde la *estructura* de los resultados durante su transformación en documentos HTML.

Algunas de las propuestas más recientes que intentan coordinar el uso de WWW y Z39.50 se basan en la utilización de Java.

3.18 Z39.50 y JAVA

Java es un lenguaje de programación diseñado para evitar algunos de los problemas detectados en lenguajes anteriores. Según sus implementadores es: *simple, orientado a objeto, distribuído, interpretado, robusto, seguro, independiente de arquitectura, portátil, de altas prestaciones, multithreaded, dinámico, de propósito general. Java sirve para programar en Internet, mediante applets independientes de la plataforma.*

Se supone que uno de los beneficios de trabajar con Java es la reutilización del código sin necesidad de recompilarlo. Además es extensible y los *applets* que se ejecutan en las páginas Web son seguros en el sentido de que no acceden a datos importantes de la máquina en que se ejecutan.

Los programas Java pueden ejecutarse en máquinas *Java virtuales*, incluidas en aplicaciones como Netscape. Si estas máquinas virtuales se comportan del mismo modo, los programas Java son fácilmente transportables a través de Internet.

En definitiva, las virtudes atribuidas a Java son:

1. Es bueno para los programadores porque facilita la ejecución del código en máquinas distintas, seguridad y desarrollo orientado a objetos.
2. Es bueno para los desarrolladores porque permite añadir animación, interacción y variaciones fácilmente.
3. Es bueno para los mantenedores de redes porque disminuye la necesidad de aplicaciones adicionales y de actualizaciones, etc.
4. El usuario se desentiende de los tres puntos anteriores.

Java es visto también como un paso más allá de los CGIs (Common Gateway Interface) porque permite al cliente más interacción que simplemente esperar a que le lleguen datos a través de la red.

Estas cualidades han llevado a que haya surgido un interés en el mundo de las bibliotecas digitales respecto a su utilización en implementaciones realizadas hasta ahora con C y CGIs. Las aplicaciones Java pueden ejecutarse en una máquina sin un servidor Web. Si se ejecutan como *applets* dentro de una página Web, se conserva el entorno del navegador, familiar al usuario.

Algunas de estas aplicaciones (*SecondFish* de OCLC, *Willow*) proporcionan acceso a bibliotecas a través de Z39.50.

La idea es pues, implementar el cliente Z39.50 o el servidor, o ambos, utilizando Java.

Un cliente Z39.50 puede estar englobado en un *applet*, que se ocupa de proporcionar la interfaz de usuario y de ejecutar las operaciones propias del cliente. El acceso al servidor sigue siendo transparente para el usuario, sin perder tampoco el conocimiento semántico que proporciona Z39.50.

Por lo que al servidor se refiere, también es posible utilizar aquí Java. Cada una de las *z-associations* de Z39.50, que mantienen el diálogo del cliente con un servidor específico es equivalente a un *thread* de Java. Se puede mantener, por tanto, varias *z-associations* simultáneamente en un servidor.

Algunos beneficios que se pueden obtener al utilizar Java en implementaciones Z39.50 son mayor flexibilidad en el lado del cliente y liberar al servidor de parte del trabajo. O, desde otro punto de vista, liberar la máquina del cliente, dejando aquí únicamente la interfaz de usuario.

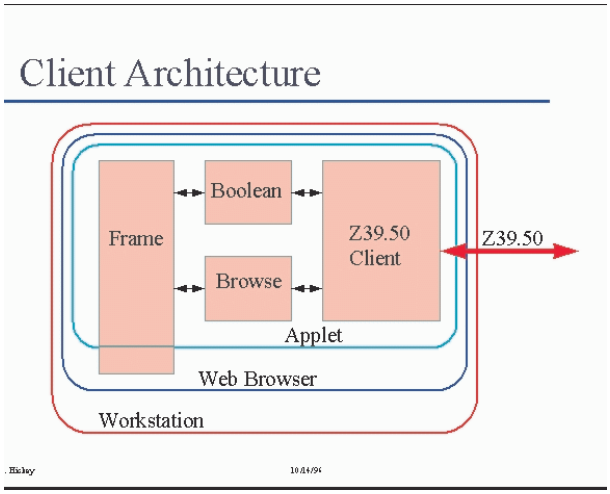

Figura 3.2: Arquitectura del cliente Java de OCLC.

Bibliografía

[1] Michael Lesk. Why digital libraries? *Follett Lecture series*, january 1995. http://www.ukolu.ac.uk/follett/prev.html.

[2] Peter J. Nürnberg, Richard Furuta, John J. Leggett, Catherine C. Marshall, and Frank M. Shipman III. Digital libraries: Issues and architectures. In *Digital Libraries 95*. Center for the Study of Digital Libraries, Texas A&M University, 1995.

[3] Special on digital libraries. *Communications of the ACM*, 38(4), april 1995.

[4] Special: Building large-scale digital libraries. *Computer*, 29(5):22–77, may 1996. http://www.computer.org/pubs/computer/dli/.

[5] Special: Digital libraries. *ERCIM news*, (27):14–30, October 1996.

[6] Andreas Paepcke. Digital libraries: Searching is not enough. *D-Lib Magazine*, May 1996.

[7] William Y. Arms, Christophe Blanchi, and Edward A. Overly. An architecture for information in digital libraries. *D-Lib Magazine*, February 1997.

[8] George H.Brett II. An integrated system for distributed information services. *D-Lib Magazine*, December 1996.

[9] World Wide Web Consortium. *Distributed indexing/searching workshop*, may 1996. http://www.w3.org/pub/WWW/Search/9605-Indexing-Workshop/.

[10] Leslie L. Daigle and Sandro Mazzucato. Distributed indexing and searching: A big picture. In *Distributed Indexing/Searching Workshop*. World Wide Web Consortium, may 1996. http://www.w3.org/pub/WWW/Search/9605-Indexing-Workshop/.

[11] An Architecture for Aggregating Distributed Data and Meta-Data Obje cts. World Wide Web Consortium, may 1996. http://www.w3.org/pub/WWW/Search/9605-Indexing-Workshop/.

[12] ANSI/NISO Z39.50-1995. *Information Retrieval (Z39.50): Application Service Definition and Protocol Specification*, July 1995. http://lcweb.loc.gov/z3950/agency.

[13] Juha Hakala. Z39.50-1995: Information retrieval protocol. http://renki.helsinki.fi/z3950/z3950pr.html, January 1996.

[14] Lorcan Dempsey, Rosemary Russell, and John Kirriemuir. Towards distributed library systems: Z39.50 in a european context. *Program. Automated library and information systems*, 30(1):1–22, 1996.

[15] Sebastian Hammer (Index Data) and John Favaro (Intecs Sistemi). Z39.50 and the World Wide Web. *D-Lib Magazine*, march 1996.

[16] Ray Denenberg. Z39.50 and navigating distributed collections. In *Distributed Indexing/Searching Workshop*, may 1996. http://lcweb.loc.gov/z3950/agency/indexing/ray.html.

[17] Maria Bruna Baldacci and John Favaro. Online public access catalogs and the World Wide Web. In *Distributed Indexing/Searching Workshop*, may 1996. http://lcweb.loc.gov/z3950/agency/indexing/ baldacci.html.

[18] Margaret St. Pierre. Z39.50 and semantic interoperability. In *Distributed Indexing/Searching Workshop*, may 1996.

[19] Dr. Chris Buckley and Peter Ryall. Z39.50 & ranked searching. In *Distributed Indexing/Searching Workshop*, may 1996.

[20] CNIDR. *Z39.50 Profile for Access to Digital Collections*. http://vinca.cnidr.org/protocols/profiles/zdl.html.

[21] Margaret St. Pierre. Z39.50 for full-text search and retrieval, may 1996.

[22] Ralph Le Van. Building a z39.50 client, 1996. ftp://ftp.rsch.oclc.org/pub/SiteSearch/z39.50_client_api/zclient.ps.

[23] Clifford A. Lynch. The z39.50 information retrieval standard. part i: A strategic view of its past, present and future. *D-Lib Magazine*, April 1997. http://www.dlib.org/dlib/april97/04lynch.html.

[24] Sandra D. Payette and Oya Y. Rieger. Z39.50. The User's Perspective. *D-Lib Magazine*, April 1997. http://www.dlib.org/dlib/april97/ cornell/04payette.html.

[25] Douglas D. Nebert and James Fullton. Use of the Isite Z39.50 Software to Search and Retrieve Spatially-referenced Data. In *Digital Libraries'95*, 1995. http://csdl.tamu.edu/DL95/papers/nebert/nebert.html.

[26] Ray Denenberg. Z39.50 and navigating distributed collections. *D-Lib Magazine*, may 1996.

[27] Paul Jones. Java and libraries. *D-Lib Magazine*, march 1997.

[28] Thomas B. Hickey. Java & z39.50. http://www.oclc.org:5047/ hickey/ presentations/lita1096/.

[29] Matt Freedman. White paper on Javafied Willow. U. Washington, october 1996.

[30] Brester Kahle Thinking Machines Corporation. An information system for corporate users: Wide Area Information Servers, 1991.

[31] WAIS over z39.50-1988, june 1994. RFC 1625.

[32] Standards related to information discovery and retrieval. http://vinca.cnidr.org/protocols/protocols.html.

[33] Z39.50 resources-a pointer page. http://ds.internic.net/z3950/z3950.html.

[34] The Library of Congress page for z39.50. http://lcweb.loc.gov:80/z3950/.

[35] CNIDR (Center for Networked Information Discovery and Retrieval). http://cnidr.org/welcome.html.

[36] Information technologie standards. http://www.ua.ac.be/MAN/TO2/toc.html.

[37] Internet searching with z39.50. http://www.markkelly.com/z3950/.

[38] Z39.50 and other standards. http://www.markkelly.com/z3950/#other.

[39] Jose Andrés González Fermoselle. Z39.50.

[40] Version 4 development. http://lcweb.loc.gov/z3950/agency/version4.html.

[41] Andrew S. Tanenbaum. *Computer Networks*. Prentice Hall, third edition, 1996.

[42] ARCA. http://www.pisa.intecs.it/projects/ARCA.

[43] Lorcan Dempsey, Anne Mumford, and Bill Tuck. Standards of relevance to networked library services. *Libraries and IT: working papers of the Information Technology Sub-comittee of the HEFCs' Libraries Review*, 1993.

[44] Lorcan Dempsey, Anne Mumford, Alan Robiette, and Chris Rusbridge. eLib standards guidelines, february 1996.