CAPÍTULO 4

IMPLEMENTACIÓN

En el presente capítulo se describen los detalles de implementación de Viajerus basados en el diseño conceptual definido anteriormente. Aquí se explica la forma en que se llevó a cabo la integración de Viajerus a la arquitectura U-DL-A, las herramientas empleadas, las tablas generadas a partir del modelo de datos entidad - relación, así como una descripción detallada de los módulos del sistema.

4.1 Integración de Viajerus a U-DL-A.

Como se mencionó en el primer capítulo de este documento, el proyecto U-DL-A pretende crear espacios personales en los que el usuario pueda accesar a diversos recursos y servicios de la biblioteca, y organizarlos de acuerdo a sus necesidades y preferencias. En este espacio se incluyen agentes personales a los que se les pueden delegar ciertas tareas y que son controlados por medio del UAD (Figura 4.1). Es a través de este módulo donde el usuario puede crear instancias de las clases de agentes con las que cuenta la biblioteca digital. Existe ya una primera implementación prototípica del concepto de espacio personal denominado "MiBiblio" [Fernández 2000], y una segunda versión del Director de Agentes (por aparecer en [Cocoletzi 2000]) a las cuales se integró este proyecto.

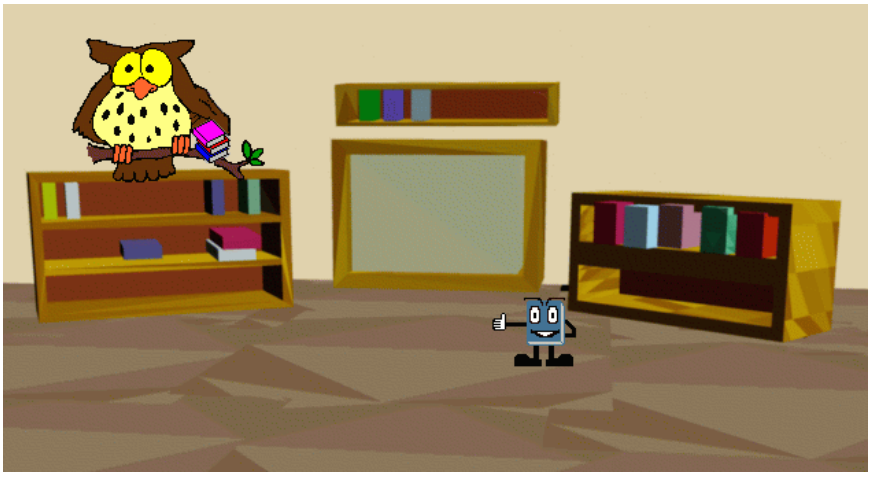

Figura 4.1 Interfaz personalizable "Mi Biblio" con agentes de usuario

(adaptada de [Fernández 2000])

Viajerus se incorporó a la arquitectura de U-DL-A al ser registrado a través del UAM, como una de las clases de agentes disponibles para ofrecer servicios recuperación al usuario. Una vez que Viajerus fue registrado, el usuario puede visualizar esta clase de agentes por medio del UAD. En caso de que se desee crear una instancia del agente Viajerus, el UAD presenta una interfaz en la que se le pide al usuario que proporcione un nombre y una descripción opcional de la tarea que va a desempeñar el agente (ver Figura 4.2) Después de esto, el UAD le pide a ALiS que registre la nueva instancia, para después informarle al usuario el estado actual y los datos relevantes del agente, y por último presentarle la interfaz inicial de Viajerus.

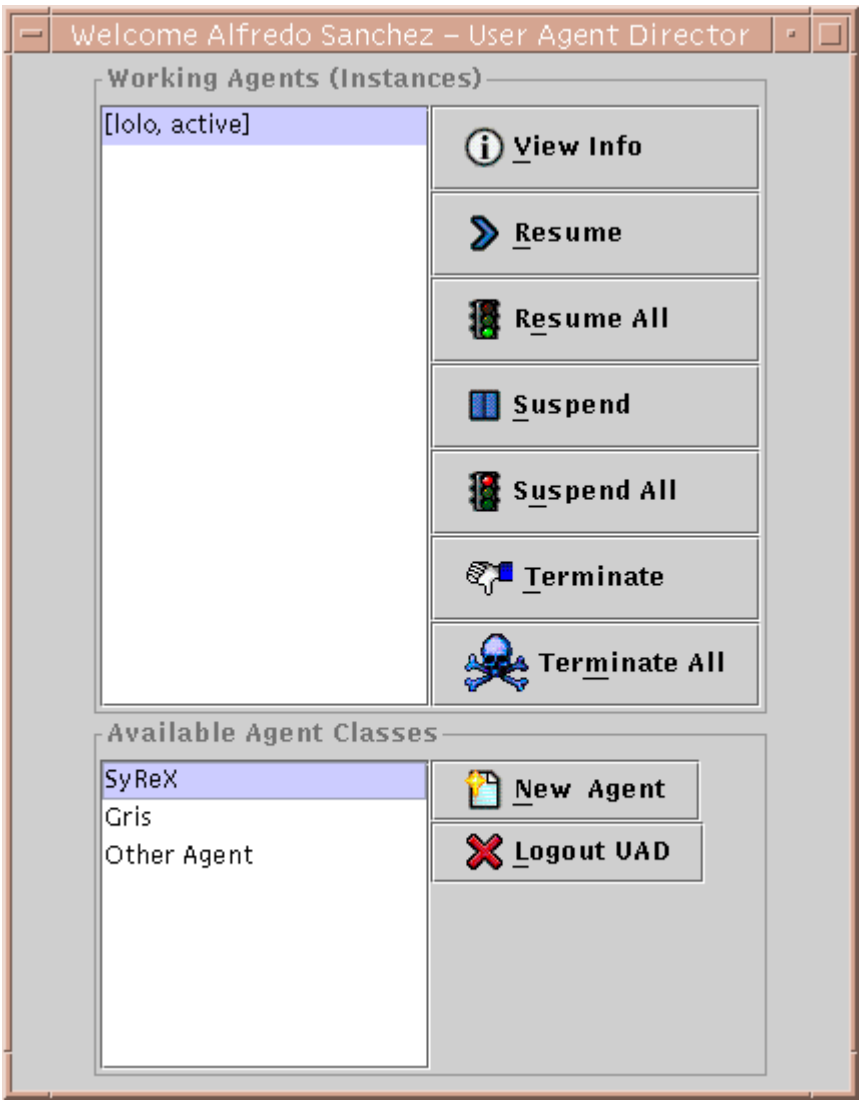

Figura 4.2 Interfaz Principal del Director de Agentes de Usuario

(adaptada de [Cocoletzi 2000])

4.2 Herramientas utilizadas

La elección de las herramientas empleadas para la implementación de Viajerus estuvo basada en la naturaleza distribuida de los agentes y de la biblioteca digital U-DL-A, en la diversidad de plataformas en la que se ofrece este servicio de recuperación, y por supuesto, en la heterogeneidad de los manejadores de bases de datos que se deseen emplear.

4.2.1 Java

Las ventajas que ofrece este lenguaje de programación han sido ampliamente consideradas para la implementación de la mayoría de los servicios que ofrece la biblioteca digital U-DL-A. El hecho de ser orientado a objetos e independiente de plataforma, de brindar una ejecución segura y de realizar un cargado dinámico de clases, son características que favorecen al desarrollo de aplicaciones Web, tal y como lo requiere el proyecto U-DL-A. En este trabajo son de principal importancia aquellas relacionadas a los mecanismos de serialización y al hecho de emplear una programación controlada por multihilos. En particular se seleccionó la plataforma Aglets Workbench (AWB) para el desarrollo de agentes móviles.

4.2.2 Aglets Workbench

Las razones por las cuales se eligió AWB para la implementación de los agentes móviles de este proyecto se fundan en que esta herramienta hereda todas las características de Java y además ofrece las siguientes bases para su desarrollo.

Modelo aglet (abstracciones y conductas de un agente). En este modelo un agente es un objeto móvil que tiene su propio hilo de ejecución, que se maneja por eventos y que se comunica por medio de mensajes. En él se definen se definen cuatro abstracciones básicas y una serie de conductas para su implementación.

Las abstracciones del modelo aglet son: *aglet* (un objeto Java móvil que visita nodos en una red), *proxy* (representante del aglet que sirve de escudo para protegerlo del acceso directo a sus métodos públicos), *contexto* (ambiente o lugar de trabajo para el aglet) e *identificador* (clave única e inmutable que identifica a cada uno de los aglets).

Así mismo define un conjunto de conductas que los aglets pueden presentar a lo largo de su ciclo de vida y son: *creación* (al nuevo aglet se le asigna un identificador, se le ubica en un contexto y se inicializa), *clonación* (se produce una copia idéntica al aglet original en el mismo contexto pero con distinto identificador), *despacho* (cuando un agente se envía de un contexto a otro), *retracción* (cuando se trae de regreso a un agente ubicado en un contexto remoto), *desactivación* y *activación* (habilidad para interrumpir temporalmente su ejecución y restaurarla en el mismo contexto) y *disposición* (cuando se interrumpe su ejecución y se elimina del contexto) [Lange y Oshima 1998].

Modelo de comunicación. AWB provee de un mecanismo para la comunicación entre aglets basado en el paso de mensajes. Estos mensajes pueden ser síncronos o asíncronos, serializados, prioritizados de acuerdo a un tipo, o bien, paralelizados.

Modelo de eventos. Este modelo permite al programador introducir manejadores de eventos en un aglet. Estos manejadores captan eventos durante el ciclo de vida del aglet permitiendo así tomar ciertas acciones. Existen tres tipos de manejadores: los de movilidad (controla eventos como el envío, llegada y retracción de los agentes), los de persistencia (manipula los eventos de activación y desactivación) y los de clonación (escucha el evento de clonación).

Mecanismo de seguridad. El marco de trabajo de los aglets provee de un mecanismo de seguridad extensible. El lenguaje Java provee la primera capa de seguridad pues el verificador del bytecode de Java revisa el formato del código. La segunda capa es proporcionada por el administrador de seguridad (una clase Java) que permite a los programadores establecer sus propios mecanismos de protección. La última capa está en manos del API de Java, ya que permite el cifrado, firmas digitales y la autentificación.

Protocolo de transferencia. Para hacer posible la transferencia de agentes, los aglets utilizan el Protocolo de Transferencia de Agentes (ATP, por sus siglas en inglés). Es un protocolo estándar a nivel aplicación diseñado para sistemas distribuidos basados en agentes. Está inspirado en Internet y en el paradigma petición-respuesta; se utiliza para manipular y transportar agentes móviles en una red independientemente del lenguaje y plataforma en el que hayan sido desarrollados [Lange 1997]. Aunque está diseñado completamente en Java, ATP posee clases altamente portables que proveen un API capaz

de crear servidores ATP, de conectar sitios ATP, y de generar respuestas a requerimientos ATP [Castellanos y Sandoval 1997].

Un aspecto definitivo para la elección de ASDK (Aglets Software Developer Kit) fue la cantidad de información y software disponible que facilitan su comprensión. Algunos sitios interesantes son los creados por el Laboratorio de Investigación de IBM, Tokyo (http://www.trl.ibm.co.jp/aglets) y por el Portal de Aglets (http://luckyspc.lboro.ac.uk/Aglet/index.html)

4.2.3 Servlets

Los Servlets brindan generosas características que facilitan la creación de aplicaciones Web, aspecto que es de considerable importancia en el presente trabajo ya que se requiere de la generación de páginas dinámicas y formas de captura para sus interfaces. Algunas ventajas de esta librería detalladas en [Hunter y Crawford 1998] se mencionan a continuación:

Extensibilidad. Los servlets extienden las capacidades del servidor.

Elegancia. El código generado en la implementación de los servlets es orientado a objetos, limpio, simple y modular.

Seguridad. Los servlets son altamente seguros ante los problemas de manejo y desperdicio de memoria, así como referencias a apuntadores inválidos.

Portables. Pueden ejecutarse en todas las plataformas que soportan Java y con la mayoría de los servidores de web disponibles.

Poderosos. Heredan el poder completo del corazón del API de Java: redes, acceso a URL´s, multi-hilado, manipulación de imágenes, compresión de datos, conexión a bases de datos por medio de JDBC, internacionalización, invocación a métodos remotos (RMI), conectividad CORBA, serialización de objetos e interacción con aglets, entre otros.

4.2.4 Informix Universal Server

El manejador de base de datos utilizado fue Informix Universal Server, esto con el objetivo de estandarizar el presente proyecto con el resto de los desarrollados en U-DL-A. Informix conserva la integridad de los datos y soporta una amplia gama de las transacciones que se llevan a cabo en la biblioteca digital. Este DBMS incluye un API (Interfaz para Programación de Aplicaciones) para comunicarse con Java por medio de JDBC (Java Data Base Connectivity). JDBC es una interfaz que sirve para acceder a bases de datos por medio de SQL (Lenguaje Estructurado de Consultas).

4.3 Tablas generadas

Las tablas generadas a partir del modelo entidad-relación (ver sección 3.3) sirven para almacenar permanentemente la información que los agentes móviles requieren para su funcionamiento, así como para saber cuáles son las bibliotecas digitales a las que se puede viajar. Estas tablas se integraron al modelo de datos definido para U-DL-A, en especial con los proyectos desarrollados por [Carballo 2000], [Cocoletzi 2000] y [Fernández 2000]. La finalidad de las tablas de Viajerus se listan a continuación, mientras que sus detalles y ejemplos pueden consultarse en el Apéndice B.

library. Posee los datos de cada una de las bibliotecas digitales con las que existe algún convenio relacionado con agentes móviles. Entre ellos se encuentran el identificador de cada de la biblioteca, su nombre completo, la descripción de su agencia bajo el protocolo ATP, el modo de despliegue que es el que ayudará a construir el URL de cada uno de los resultados obtenidos en ese acervo, y por último el tipo de contrato que tiene U-DL-A con las otras bibliotecas.

mobile_agent. Contiene la información característica de cada instancia de agente móvil. Sus campos son el identificador del agente, el de su usuario y el del aglet que le fue asignado, también almacena el nombre y la descripción de la tarea con la que el usuario lo identifica, así como el estado en el que se encuentra el agente (activo, inactivo, concluido).

result. Almacena los detalles de los resultados obtenidos por cada uno de los agentes. Sus campos incluyen el identificador del resultado; y la descripción de cada tesis recuperada, para lo que se ocupan los campos autor, año, título, carrera, universidad, grado y URL . Esto con el fin de poder ordenar los resultados a gusto del usuario.

restriction. Posee la descripción de las restricciones bajo las cuales se está realizando la búsqueda.

visit. Contiene el reporte de cada una de las visitas que realizó el agente. Contempla los campos del identificador de la visita, del agente y de la biblioteca, además de guardar el estado de la visita (exitosa, no conectada) y el número de resultados obtenidos.

4.4 Módulos Viajerus

A continuación se presenta una descripción de los detalles de implementación de los módulos que conforman Viajerus.

4.4.1 Módulo Agencia

La agencia es una entidad estática que reside en el dominio y que permite enviar y recibir agentes móviles. Para lograr este objetivo es necesario crear un servidor que sirva de anfitrión para aglets, el cual debe habilitar un puerto estándar que típicamente es el 9000. Por lo anterior, es necesario que cada nodo tenga su propia agencia o servidor que ofrezca un ambiente de ejecución en el que los agentes móviles puedan instalarse y realizar su tarea. Esta condición es un requisito ya que los aglets funcionan únicamente bajo el llamado Contexto Aglet, ambiente que garantiza que la creación, destrucción y envío de agentes se realice de una manera controlada (ver Apéndice C).

La agencia contiene a su vez dos submódulos: el *administrador de agentes móviles* y el *agente intermediario*, que en sí son dos procesos que se mantienen en continua ejecución para atender sus respectivas peticiones.

El *Administrador de Agentes Móviles* es una agente estático que se encarga de controlar los mensajes y acciones de los agentes móviles que han sido creados localmente. Cualquier instrucción sobre el cambio de estado de algún agente, o bien, algún reporte o notificación que los agentes móviles quieran comunicar, se maneja a través de mensajes manipulados por el Administrador de Agentes y los agentes viajeros involucrados. Este componente permite llevar a cabo una comunicación asíncrona.

Para manipular los servicios que brinda la agencia a los agentes móviles foráneos se creó un agente estático llamado *Agente Intermediario* que es a quien se solicita el servicio de recuperación de información. En este caso el Agente Intermediario se cerciora de que el agente que está haciendo la petición realmente provenga de algún nodo autorizado, para ello busca en la tabla *library* que el nodo origen de ese aglet esté registrado como válido. Una vez realizado este paso procede a hacer la llamada al servicio de recuperación de información. El parámetro que este servicio recibe se refiere a las características de la búsqueda, las cuales contienen las palabras clave y sus respectivas limitantes para refinarla. Este servicio regresa los resultados de su búsqueda, los cuales el Agente Intermediario se los hace llegar al móvil y éste los agrega a su equipaje para continuar con su viaje.

A groso modo el *módulo agencia* es capaz de recibir y enviar agentes, de garantizar el acceso de agentes móviles seguros al dominio y rechazar aquellos que no lo sean, de controlar las acciones de los agentes móviles locales, mediar el acceso a los recursos de información, asegurarse de que la integridad y privacía de los datos se

conserve y de atender a las peticiones de recuperación que se soliciten. Cabe mencionar que para que la agencia pueda transferir y recibir agentes es necesario manejar el protocolo ATP definido en el API de Aglets (ver sección 4.2.2).

Las dificultades que se presentaron al momento de implementar este módulo se relacionan con el hecho de crear el servidor de agentes que fuera capaz de integrarse a la arquitectura U-DL-A. Si bien la Herramienta de Desarrollo de Software Aglets (ASDK) incluye un servidor para aglets llamado Tahiti [Pérez 1998], éste presentaba una interfaz gráfica por medio de la cual el usuario interactuaba directamente con la creación, envío y eliminación de aglets; sin embargo, se tenía que habilitar un puerto distinto por cada usuario que deseara crear un agente móvil en la biblioteca. A partir de aquí es como se desarrolló un servidor general de agentes que se integró a U-DL-A como una extensión de su arquitectura, trabajando especialmente con el módulo ALiS (ver Sección 3.2). Gracias al buen desempeño de este servidor es que se decidió integrarle las funcionalidades provistas por el *Administrador de agentes* y por el *Agente Intermediario*, generando así el concepto de *agencia*.

4.4.2 Módulo agente móvil.

Está formado básicamente por dos submódulos: el de *Interfaz* y el *Agente móvil* como tal. Juntos proveen al usuario el servicio de recuperación información en forma transparente basado en el concepto de agentes.

El componente *Interfaz* provee al usuario de un ambiente accesible a través de un navegador, en el que puede delegar la tarea de recuperación sobre algún tema o personaje de interés en el contexto de colecciones de tesis digitales federadas, o bien consultar los resultados obtenidos por el agente. Una vez que se ha instanciado el agente Viajerus a través del UAD, este módulo presenta la interfaz de búsqueda sencilla (Figura 4.3), en la cual el usuario define únicamente los términos sobre los que quiere buscar y el itinerario que debe recorrer el agente. Si el usuario desea realizar una búsqueda más avanzada o refinada, se le da la opción de limitarla definiendo también sus preferencias en cuanto a fechas, autores, contenido y grado de tesis se refiere (Figura 4.4). Una vez que se ha configurado la búsqueda entonces el agente móvil inicia su recorrido, no sin antes informar al usuario sobre su estado y la forma en la que puede consultar los resultados.

El periodo de vida de un agente móvil dura hasta que termina de

recorrer los dominios especificados en su itinerario, o bien hasta que su dueño decida eliminarlo.

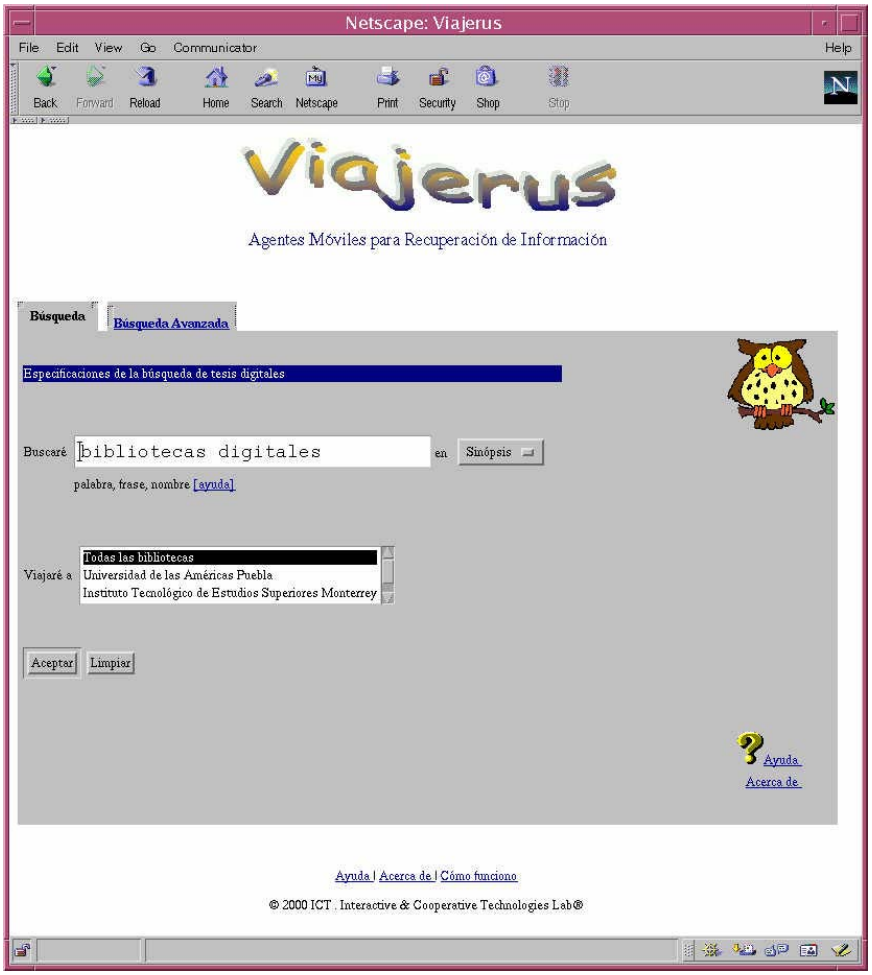

Figura 4.3 Interfaz de Viajerus. (búsqueda sencilla)

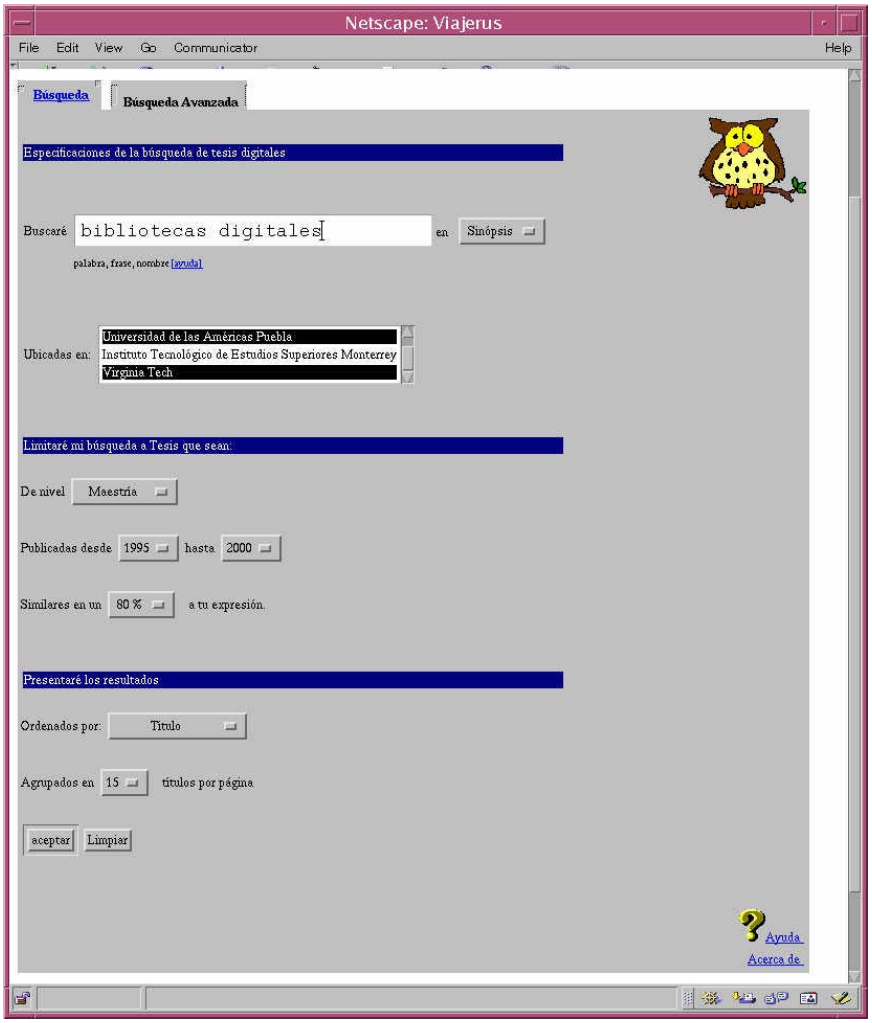

Figura 4.4 Interfaz de Viajerus. (búsqueda avanzada)

En este momento es cuando el usuario puede desconectarse, cerrar su ambiente personal o simplemente realizar otras tareas mientras que el agente está buscando la información solicitada en los diversos acervos. En cuanto el usuario desee revisar los resultados sólo tendrá que invocarlos a partir del espacio personal o bien por medio del UAD. En este momento se invoca el método correspondiente para presentar la lista de resultados en el navegador (Figura 4.5). Dichos resultados serán ligas que hagan referencia a la tesis completa, de tal forma que al ser activada alguna de las ligas de los resultados, se desplegará el documento completo que se seleccionó. Un ejemplo de cómo se visualizaría un documento completo se muestra en la Figura 4.6. Cabe mencionar que esta figura presenta un documento correspondiente a la colección de Tesis Digitales de U-DL-A, proyecto dirigido por [Fernández 2000].

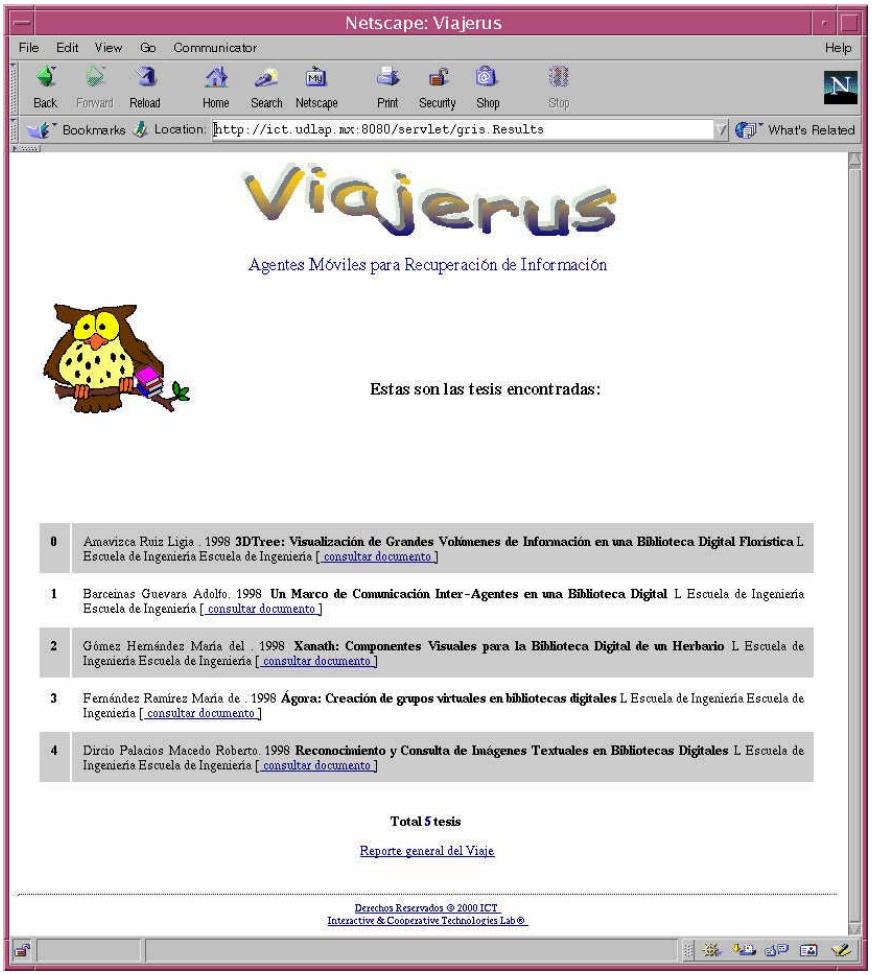

Figura 4.5 Interfaz de resultados.

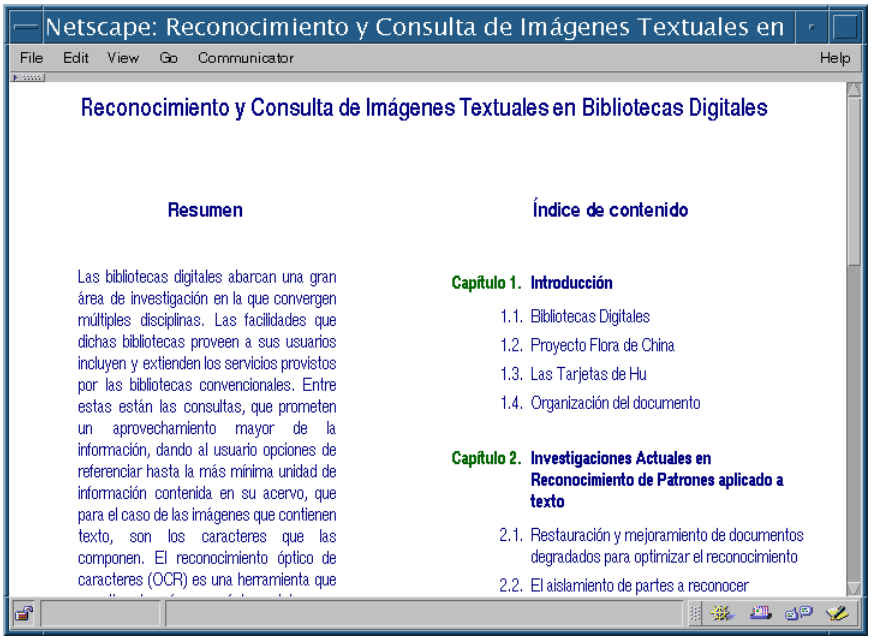

Figura 4.6 Interfaz de tesis completa (adaptada de [Fernández 2000])

Por medio de la interfaz del agente, del UAD, o bien, del espacio personal es como el agente Viajerus representa gráficamente sus estados con las siguientes imágenes. La Figura 4.7 indica que Viajerus está en un estado activo (a); que está dormido o suspendido (b); o bien bien que ha finalizado su tarea (c).

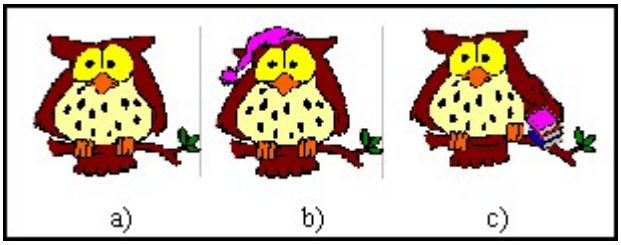

Figura 4.7 Estados del Agente Viajerus.

a) activo b) suspendido c) con resultados

Los A*gentes Móviles,* como su nombre lo indica, son los componentes que migran entre los nodos de una red. Una vez que el agente ha sido instanciado por el UAD y que le han sido configurados los parámetros de la búsqueda por medio de la interfaz de Viajerus, se procede a preparar el equipaje del agente. Este consta de su itinerario, de los detalles de la búsqueda a realizar, así como su nodo origen y tabla de resultados. Es en este momento cuando se almacenan los datos correspondientes a las tablas *mobile_agent, visit y restriction*, esto con el fin de que el usuario pueda emplear esta misma instancia de agente con las mismas restricciones para realizar búsquedas futuras o bien para consultar los resultados cuantas veces lo desee. A partir de este momento el agente inicia su viaje a través de los nodos especificados por el usuario, en caso de que alguno de ellos esté desconectado o que presente algún otro tipo de problemas el agente continuará su viaje al siguiente nodo, no sin antes anotar el problema que existió en su reporte para posteriormente presentarle al usuario los detalles generales de su viaje.

En cada llegada del agente a una agencia, éste es atendido por el Agente Intermediario, el cual valida su entrada al dominio y atiende sus peticiones. El agente móvil le proporciona al Intermediario los detalles de la búsqueda y éste, después de solicitar el servicio de recuperación de información del nodo local le entrega una lista con los resultados generados. El agente móvil los agrega a su paquete de resultados y se dirige hacia el siguiente nodo. Una vez que ha terminado de recorrer su itinerario le da a conocer al Administrador de Agentes correspondiente a su nodo origen un reporte final, que incluye un listado de resultados y anotaciones sobre cada visita (si fue exitosa, si no se pudo conectar, si no obtuvo respuesta). En cuanto el Administrador recibe este reporte, se procede a guardar tanto los resultados como las anotaciones de las visitas en la base de datos. Así,

cuando el usuario desee consultarlos sólo es necesario obtenerlos a partir de las tablas *result* y *visit*. Es en este momento cuando el agente móvil da por concluida su tarea y se dispone.

4.4.3 Módulo de recuperación de información.

Es una entidad estática que provee el servicio de recuperación de información basado en el algoritmo de espacios vectoriales. Debe existir uno de estos módulos en cada uno de los nodos, de tal forma que la biblioteca sea capaz de administrar sus propias colecciones digitales y de que el agente no requiera conocer las estructuras de datos empleadas.

El parámetro que recibe esta entidad por parte del Agente Intermediario se refiere a los detalles de la consulta, esto es, los términos o palabras clave y las restricciones de fecha, nivel o contenido de las tesis digitales. Este módulo accesa al DBMS de Informix en busca de los documentos que contengan los términos especificados. Es necesario mencionar que se recuperan todos los documentos que posean al menos alguno de los términos. Los documentos recuperados formarán una matriz como la especificada en la Figura 2.3, donde los pesos serán unos y ceros dependiendo si el término a buscar se encontró o no en cada uno de los documentos. Seguido a este paso se requiere invocar al método que aplica el modelo de recuperación de espacios vectoriales con los parámetros de la consulta y la matriz de documentos. Es aquí donde por medio de la fórmula del coseno (Figura 2.4) se calcula el ángulo de similitud entre cada documento recuperado y la consulta. Después se ordenan los resultados en forma ascendente de acuerdo al porcentaje de similitud. Posteriormente se eliminan aquellos que no cumplan con el porcentaje requerido para considerar al documento relevante, el cual fue definido por el usuario en la interfaz de búsqueda. Una vez terminada esta tarea se le hace entrega de los resultados al agente móvil para que los incluya a su equipaje y continúe su viaje.

La implementación de las clases de Viajerus se basó en el modelado UML definido en Apéndice E. La configuración de las variables de ambiente necesarias para los aglets y servlets, así como los detalles para ejecutar Viajerus se presentan en el archivo *readme.html* disponible en el paquete de este sistema.

En el capítulo siguiente se describen el estado actual de Viajerus y las pruebas realizadas durante y después de su implementación.

Chevalier Dueñas, G. A. 2000. Agentes móviles para la recuperación personalizada de información. Tesis Licenciatura. Ingeniería en Sistemas Computacionales. Departamento de Ingeniería en Sistemas Computacionales, Escuela de Ingeniería, Universidad de las Américas Puebla. Mayo.