Mecanismo de Ayuda Contextual para Sistemas de Información Federados

Proyecto de Grado 2004
Informe Final

INCO – Facultad de Ingeniería – UDELAR
Montevideo, Junio 2005

Tutores:

Ing. Federico Piedrabuena
Dr. Ing. Raúl Ruggia

Estudiantes:

Laura González
Guillermo Roldós
Flavia Serra

RESUMEN

El objetivo de este proyecto es desarrollar utilitarios y componentes para proveer funcionalidades de ayuda en línea a aplicaciones que integran Sistemas de Información Federados a través de la Web, en particular a las aplicaciones del sistema Link-All. Dicha ayuda debe ser sensible al contexto de ejecución, el cual depende entre otras cosas, de las características del usuario y de la aplicación y funcionalidad que se están ejecutando. Entre el Sistema de Ayuda y las aplicaciones debe existir muy bajo acoplamiento, siendo esta situación totalmente transparente para los usuarios, ya que ellos podrían manipular la ayuda de todas las aplicaciones como si fuera una sola, permitiendo así que los usuarios puedan conocer y trabajar en todas las posibilidades que le ofrece el Sistema de Información.

En base al estudio de las herramientas de ayuda existentes se adquiere conocimiento acerca del tema, se identifican ventajas y desventajas de las mismas, y se realiza un primer diseño del sistema con las funcionalidades básicas. A partir de éste, se implementa un caso particular que solo contempla los requerimientos de la plataforma Link-All, que más tarde es generalizado para obtener la solución que se presenta en este trabajo.

La solución final se divide en tres grandes partes. En primer lugar, un framework, que es la parte central del sistema, ya que contiene toda la lógica del negocio, luego, una implementación de éste para el sistema Link-All, y por último, un conjunto de utilitarios que proveen una serie de servicios fundamentales para el correcto funcionamiento del framework.

El desafío fue obtener un sistema con los niveles de calidad exigidos en el contexto del sistema Link-All, pero a su vez, se buscó respetar fuertemente las características de extensibilidad, flexibilidad y adaptabilidad, logrando así un Mecanismo de Ayuda Contextual configurable también a otros Sistemas de Información Federados.

Palabras Clave — Ayuda Contextual, Interfaz de Usuario, Usabilidad, Sistema de Información, Link-All.

ÍNDICE

1 INTRODUCCIÓN

1.1 Descripción del Problema

Generalmente, cuando un usuario ejecuta una aplicación, ésta tiene una ayuda asociada. Esta ayuda almacena toda la información que se presenta al usuario en pantalla en el momento de solicitar dicha ayuda. El Mecanismo de Ayuda Contextual será el encargado de administrar estos contenidos y de mostrarlos al usuario de acuerdo al contexto en el que se encuentra. Para esto, se busca entonces satisfacer las siguientes características:

- **Contextual**
  La ayuda que se provee debe ser sensible al contexto de ejecución. Este contexto depende tanto de la aplicación como de la función que se está ejecutando, de las características del usuario, del servidor que ejecuta la ayuda y de la información que se dispone para mostrar al usuario.

- **Variedad de mecanismos de ayuda**
  La ayuda puede darse mediante diferentes mecanismos, como por ejemplo: explicación corta de las opciones disponibles, visión más completa de qué incluye la aplicación o la operación en curso, navegación hacia otros datos o funcionalidades, posicionamiento en el sistema, reporte de errores y/o consultas. En cuanto a la forma de la ayuda, puede ser HTML pura o usar herramientas con mayor capacidad multimedia.

- **Configurabilidad y parametrización**
  La herramienta de Ayuda debe ser lo más configurable y parametrizable posible, facilitando su evolutividad y evitando las referencias "hard-codeadas" en el software.

- **Eficiencia**
  El mecanismo de ayuda debe ser ágil y eficiente, de lo contrario, no resulta práctico. Esto se aplica fundamentalmente a tipos de consulta más inmediatos, pudiendo ser sacrificado para consultas más complejas. Para la eficiencia también debe tenerse en cuenta el porte del equipo, la velocidad de la conexión y la forma en que el cliente se conectará al servidor (local o remoto).

A su vez, se busca que el Mecanismo de Ayuda Contextual sea un servicio común en una plataforma que contendrá un conjunto de aplicaciones que eventualmente serán independientes entre sí, debiendo existir muy bajo acoplamiento entre dichas aplicaciones y la ayuda. Es importante que desde el punto de vista de los usuarios, las aplicaciones se presenten integradas unas a otras, de esta forma, todas ellas se podrán visualizar como si fuera una sola, permitiendo así que los usuarios puedan conocer y trabajar en todas las posibilidades que le ofrece el Sistema de Información. A continuación, se adjunta un esquema en el cuál se presenta la relación existente entre todos los componentes del sistema:

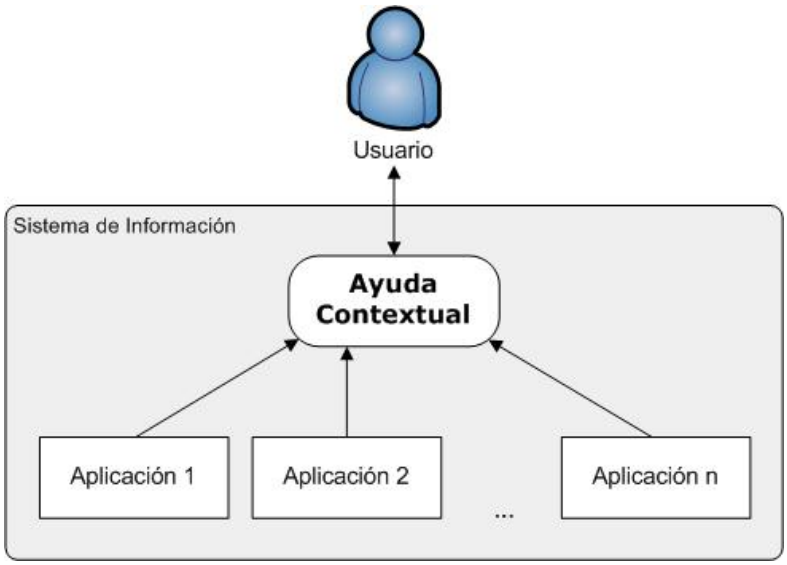


Figura 1

En la figura 1, se presenta la relación existente entre los usuarios, las aplicaciones (de un Sistema de Información), y la Ayuda Contextual. Las aplicaciones brindan al Mecanismo de Ayuda Contextual la información que desean mostrar al usuario cada vez que éste cometa un error o solicite ayuda, a su vez, el

Mecanismo de Ayuda Contextual es el encargado de presentar en pantalla todos los datos que obtiene de las aplicaciones.

Es importante destacar que este proyecto se realizo en un marco de Control de Calidad basado en la tesis de maestría del Ing. Diego Vallespir [42]. Se establece un nivel de calidad alto para todas las funcionalidades requeridas por la plataforma Link-All, mientras que para el resto de las funcionalidades se requiere un nivel de calidad[1] menos exigente.

1.2 Contexto del Proyecto

Este proyecto se encuentra en un contexto de Sistemas de Información Federados a través de la Web llamado LinkAll, cuyos componentes básicos deben ser manipulados remotamente. A continuación se presenta una visión general que pretende introducir al lector en los objetivos generales de este proyecto y una descripción de la plataforma desarrollada para éste.

1.2.1 Link-All

1.2.1.1 Visión General

El Proyecto Link-All [1] (Local-communities Insertion NetworK para América Latina) está financiado por el Programa @lis apoyado por la Oficina de Cooperación Europe-Aid de la Comisión Europea. Está dirigido a asistir en la apropiación de prácticas de tecnologías de la información (IT) innovadoras, apoyar la colaboración, promover el intercambio de experiencias, la transferencia de conocimiento y mejorar las habilidades relacionadas en tres sectores principales:

- Artesanía
- Eco-Agro Turismo
- Cultura

La plataforma Link-All incluye herramientas IT diseñadas e integradas para proveer una serie de facilidades claves que permiten la inclusión electrónica y fortalecen la integración de actividades de desarrollo local.

Los objetivos generales son:

- Proveer a comunidades locales de la oportunidad de incorporarse a la sociedad de la información a través de un conjunto de facilidades, servicios y funcionalidades especialmente diseñados para PYMEs, organizaciones y actores locales claves en los 3 sectores seleccionados.

- Promover una cadena económica integrada y localmente sustentada para comunidades locales, apoyada en una fuerte identidad local, y basada en recursos naturales, humanos y culturales locales.

- Posibilitar y facilitar la inserción exitosa de comunidades remotas en el mercado global, a través del desarrollo de una red transnacional europeo-latinoamericana de cooperación entre actores artesanales, culturales y de eco-agro turismo.

- Ofrecer a actores de los sectores objetivo acceso a un amplio espectro de buenas prácticas en Europa y América Latina y mejorar sus capacidades de colaboración conjunta para construir sinergias entre los sectores antes indicados.

1.2.1.2 Descripción de la Plataforma

El sistema Link-All [1] esta compuesto por una serie de nodos que conforman una red colaborativa de datos y servicios, cuyo principal objetivo es brindar soporte a la comunidad virtual Link-All.

El componente esencial de la red Link-All (figura 2), es el nodo Link-All. Este constituye el bloque constructor de la red, en el cual se encuentran los servicios. En la red no existe jerarquía ni distinción entre los nodos que la conforman, con excepción del “Nodo Soporte” que brinda una serie de servicios particulares.

[1] En el Anexo [C], se presenta un informe más detallado sobre el Control de Calidad y Testing del sistema.

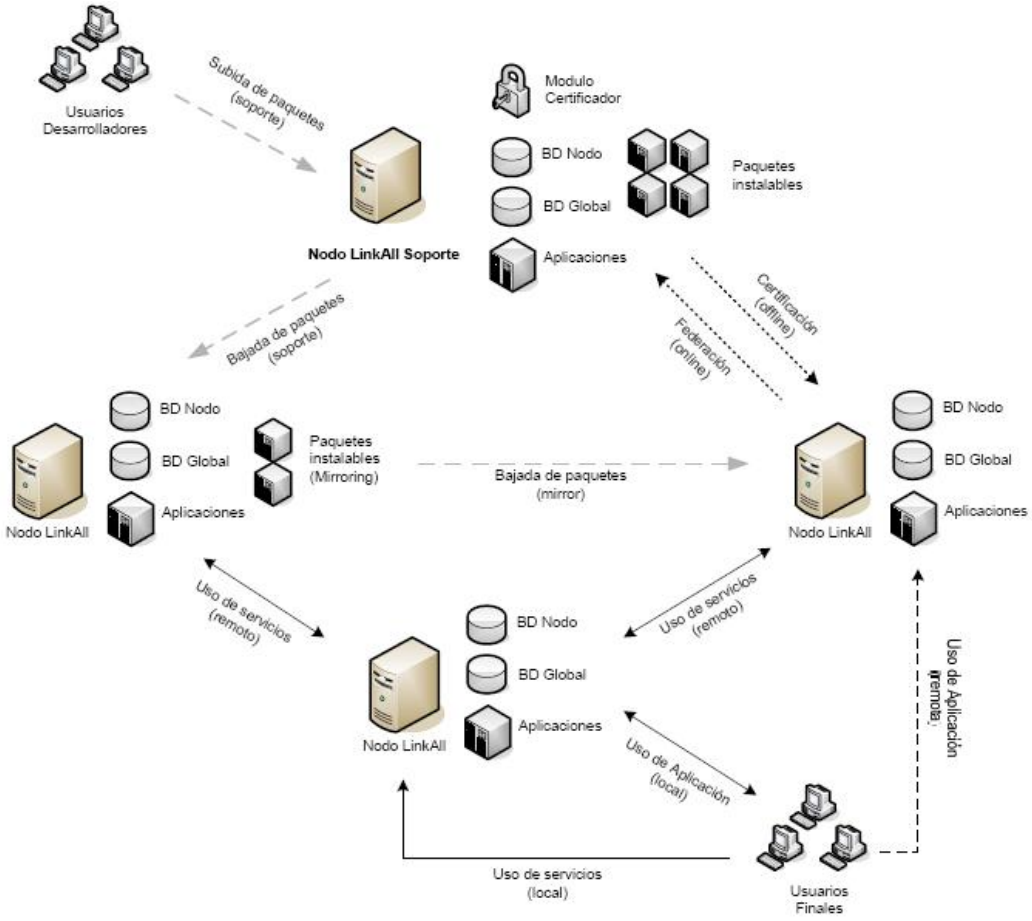


Figura 2

Dentro del nodo Link-All pueden diferenciarse dos grandes espacios o regiones. Una dedicada a brindar servicios de soporte y otra dedicada al hosting de aplicaciones utilitarias. Estas son las aplicaciones que pueden ser accedidas por los usuarios del sistema, ya sean locales o remotos (en este o en otro nodo Link-All).

Es importante destacar, que los objetivos y conceptos básicos de esta red colaborativa se aplican a cualquier Sistema de Información Federado.

1.3 Motivación

A partir de la propuesta de proveer funcionalidades de ayuda para la plataforma Link-All, se plantea desarrollar un Mecanismo de Ayuda Contextual que contemple también las necesidades de otros Sistemas de Información Federados.

1.4 Objetivos

El objetivo principal es desarrollar utilitarios y componentes para Sistemas de Información Federados (en particular para el ambiente Link-All), que provean funcionalidades de ayuda en línea para las aplicaciones que se estén ejecutando de forma local o remota.

Las funcionalidades de ayuda deben cumplir propiedades tales como:

- ejecutarse on-line en forma eficiente
- depender del contexto de operación que se este ejecutando
- depender del usuario y contexto del servidor
- ser lo más expresivas posible (utilizando herramientas multimedia y gráficas)
- incluir funciones de navegación y acceso a otros tipos de información
- ser aplicables en forma general a cualquier aplicación que se incluya
- incluir funcionalidades de orientación (tipo “mapa en el sistema”) que ayuden a ubicar la operación que se ejecuta
- permitir a los desarrolladores especificar la ayuda para sus aplicaciones

Se debe prever la ejecución de las funciones de ayuda en modalidad HTML "puro" (cliente super-fino) o con herramientas que requieran un cliente potente y comunicaciones rápidas. Aquí se deben buscar balances entre potencia de la ayuda y factibilidad de ejecución por parte de PC's de pequeño porte y comunicado con bajo ancho de banda.

1.5 Resultados Esperados

Se busca que la solución al problema contemple dos objetivos:

- Lograr un sistema que interactúe de forma eficiente con la plataforma Link-All, con un nivel de calidad adecuado para su puesta en producción.

- Obtener un producto que permita cubrir los fines académicos de este proyecto y que sea capaz de interactuar también con otros Sistemas de Información Federados.

1.6 Organización del Documento

La organización de este documento esta basada en la "Guía Informe de Proyecto de Grado" [40], también se consideró conveniente consultar informes de otros Proyectos de Grado [38] [39] que sirvieron de apoyo para entender la forma en que se debía explicar y documentar cada uno de los pasos que se han llevado a cabo a lo largo de este trabajo.

El informe se divide en nueve capítulos, cada uno de ellos agrupa los puntos más importantes del proyecto. La finalidad es presentar inicialmente una idea global del problema, para luego ir profundizando paso a paso en cada uno de los temas. La estructura de este documento es la siguiente:

- **Capítulo 2**
  Se brinda una breve descripción de los diferentes sistemas de ayuda que fueron relevados para realizar el diseño inicial del sistema, además, se presentan otras tecnologías investigadas.

- **Capítulo 3**
  Se plantea un esquema global de la evolución del proyecto, presentando los conceptos más importantes que intervienen en la solución final.

- **Capítulo 4**
  Se presenta el diseño de la solución y se describe la arquitectura.

- **Capítulo 5**
  Se presenta la implementación del framework para un Sistema de Información Federado: Link-All.

- **Capítulo 6**
  Se describen y justifican las herramientas y las API's utilizadas en la implementación.

- **Capítulo 7**
  Conclusiones y Trabajo a Futuro. Se dejan abiertas diferentes posibilidades de trabajo.

- **Capítulo 8**
  Referencias. Se agrupan por temas.

- **Capítulo 9**
  Anexos.

2 ESTADO DEL ARTE

2.1 Introducción

El estudio del Estado del Arte se centró en la búsqueda de información de los Sistemas de Ayuda existentes, conocer su funcionamiento y las diferentes herramientas que éstos ofrecen.

El resultado de esta investigación fue que en la actualidad existen una gran cantidad de productos que permiten brindar ayuda a aplicaciones. Cada uno de ellos posee su propio formato para describir la información, pero en general, todos manejan los mismos conceptos y proveen funcionalidades similares.

Usualmente, la ayuda de una aplicación está compuesta por:

- conjunto de *Contenidos*
- *Tabla de Contenido*
- *Índice*
- *Glosario* y
- *Búsqueda* de texto dentro de los contenidos.

En la figura 3 se presenta el modelo conceptual de la primera aproximación a nuestro Sistema de Ayuda Contextual.

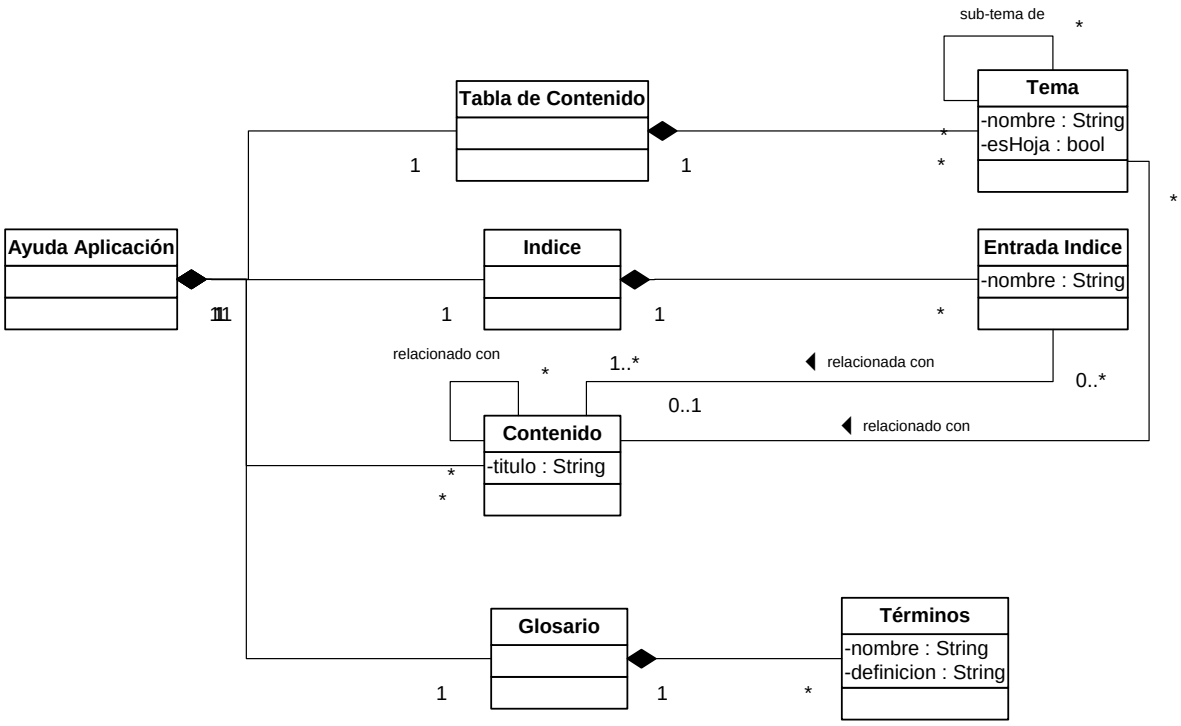


Figura 3

2.1.1 Contenidos

Los contenidos brindan a los usuarios la información de ayuda que se visualiza a través de un explorador web. Ejemplos de contenidos son páginas html, archivos pdf, documentos word, películas flash, imágenes, o cualquier elemento que genere dinámicamente algunas de las formas mencionadas anteriormente.

2.1.2 Tabla de Contenido

La tabla de contenido despliega un conjunto de nodos (temas de ayuda), generalmente, en una estructura de árbol. Cada uno de estos nodos puede tener (o no), un contenido asociado. La tabla de contenido brinda al usuario una forma sencilla de navegar a través de todos los contenidos que componen la ayuda.

2.1.3 Índice

El índice es un conjunto de palabras claves, cada una de ellas tiene un conjunto de contenidos asociados.

2.1.4 Glosario

El glosario está compuesto por un conjunto de parejas. Cada pareja esta formada por un término y una definición asociada.

2.1.5 Búsqueda
Permite realizar la búsqueda de texto dentro de los contenidos.

2.1.6 Favoritos
Conjunto de contenidos que el usuario puede conservar para, más tarde, poder acceder a ellos más rápidamente.

2.2 Sistemas de Ayuda
Todos los Sistemas de Ayuda[2] investigados fueron de gran aporte para la realización del Mecanismo de Ayuda Contextual. Conocer y entender cada una de sus funcionalidades fue de gran importancia, ya que se debía mantener sus funcionalidades básicas, intentando mejorarlas de forma transparente para el usuario final. En la búsqueda de los objetivos planteados, se realizó un análisis de las funcionalidades que debían ser ofrecidas y que no estaban presentes en las ayudas investigadas. Un ejemplo de esto es que la ayuda de una aplicación pueda ser obtenida tanto de forma local como remota.

En los siguientes puntos se pretende destacar las características más importantes de los Sistemas de Ayuda consultados:

- **WinHelp**
  Sistema de ayuda en línea de Microsoft [2]. Todas las versiones de Microsoft Windows continúan soportando ayuda en línea en este formato, sin embargo, el desarrollo de WinHelp fue discontinuado. La ayuda es creada a partir de archivos con formato RTF.

- **Microsoft HTML Help**
  Sistema de ayuda estándar [3] para la plataforma Windows. El gran cambio con respecto a WinHelp es la utilización de archivos HTML en lugar de archivos RTF.

- **JavaHelp**
  Sistema [4] extensible e independiente de la plataforma, que permite incorporar ayuda en línea en applets, componentes, aplicaciones, sistemas operativos y dispositivos. JavaHelp también puede ser utilizado para distribuir documentación en línea para la web o intranet corporativa. La característica única del sistema de ayuda JavaHelp es que está desarrollado en Java, por lo que se adapta mejor a las aplicaciones desarrolladas en ese lenguaje de programación.

- **FlashHelp**
  Nuevo formato de ayuda [5] en línea diseñado para soportar aplicaciones web, de escritorio e híbridas. Tiene la característica de presentar un mismo "look and feel" y las mismas funcionalidades en diversos exploradores web y sistemas operativos, como ser Windows, Macintosh y Linux. FlashHelp combina HTML, XML y Flash en un sistema independiente de la plataforma.

- **Tecnología Help de Oracle**
  Ofrece los medios para desarrollar y presentar sistemas de ayuda basados en HTML [6]. Los autores crean un único sistema de ayuda que puede luego ser presentado tanto en:

  - un ambiente java, utilizando “Oracle Help for Java” (OHJ)
    Conjunto de componentes java, una API java y una especificación de formato para desarrollar y presentar contenido de ayuda basado en HTML.

  - un ambiente Web, utilizando “Oracle Help for the Web” (OHW)
    Es un servlet Java y una especificación de formatos de archivo para desarrollar y distribuir contenido de ayuda basado en HTML. Con OHW todo el procesamiento toma lugar en el servidor, a través del servlet OHW, por lo que muchos usuarios tienen acceso a una misma instalación del sistema de ayuda.

2.3 Metadata
Un punto a resolver, era la forma de representar la información de la ayuda de cada una de las aplicaciones. En principio, se realizó el análisis de diferentes lenguajes orientados a metadatos [9] [36], como son:

- **XML:** describe contenido [7] [8]

[2] La descripción completa de cada uno de los Sistemas de Ayuda investigados está disponible en el Anexo [A].

- **RDF:** define un formalismo para la definición de recursos [8] [10] [11]

- **OWL:** Web Ontology Language, lenguaje de ontologías web [10] [12]

Dado que la información a representar era de escasa complejidad y los tiempos del proyecto de grado no son extensos no se profundizó en el estudio de los formatos RDF y OWL, y se consideró XML para la representación de los datos.

2.4 Tecnologías

La decisión de utilizar Java [13] como lenguaje de programación requirió del estudio de las diferentes tecnologías [18] que forman parte de este entorno:

- Plataforma J2SE: Permite desarrollar Applets o Aplicaciones, y brinda servicios para manejo de E/S, interfaz gráfica , RMI, JNDI.

- Plataforma J2EE: Especificación para desarrollar aplicativos de gran porte. Define un mecanismo estándar basado en el concepto de portabilidad de Java.

- EJBs: Enterprise Java Beans. Propuesta de una arquitectura basada en componentes para encapsular la lógica del negocio en componentes configurables y reutilizables. Definen cómo deben ser escritos los componentes del lado del servidor. Existen tres tipos de componentes en la especificación 2.0 de los EJBs: Session Beans, Entity Beans y Message-driven Beans.

- Web Services: Componente de Software que se comunica con otras aplicaciones codificando los mensajes en XML, y envía estos mensajes a través de protocolos estándares de internet (tales como HTTP).

- RMI: Remote Method Invocation. Forma de comunicación entre objetos distribuidos y Java.

- Java Servlets: Componentes orientados al request / response. Toman los pedidos de un cliente (web browser) y luego generan la respuesta al mismo.

- JSP: Componentes para la web, muy similares a los Servlets. La mayor diferencia está en que las JSPs no son código Java puro, sino que están más orientadas al diseño de las páginas web.

Si bien es importante la lógica del negocio, el diseño de la interfaz de usuario es la parte visible de un sistema que funciona en un ambiente web, por lo que se estudió la forma de obtener una interfaz amigable. Ya que Flash [19] es considerada una de las tecnologías más aceptadas para el diseño de interfaz de usuario, se buscó lograr la interacción entre “Rich Internet Applications” (RIA) [23] que puedan ser ejecutadas con Flash Player, y aplicaciones Java del lado del servidor. Existen varias opciones para esto, se consideraron las siguientes [20]:

- Flash Remoting
  A través de FlashRemoting es posible comunicar un cliente Flash con una aplicación Java que se ejecuta del lado del servidor. Para implementar esto, hay varias alternativas:

  - OpenAMF
    Es un proyecto open source, que permite la integración Flash-Java mediante Flash Remoting, pero uno de los primeros inconvenientes presentados en este caso era que no existía la posibilidad de llamar a Servlets o JSPs. Si bien en la actualidad se considera resuelta esta carencia, aún sigue siendo la opción menos desarrollada.

  - FlashORB
    FlashORB Flash Remoting es compatible con los componentes de Macromedia. Esta disponible para los ambientes J2EE y .NET, y para el caso de J2EE tiene dos ediciones, estándar y profesional. La versión estándar tiene una licencia gratuita, pero la versión profesional que es la más completa, no es gratuita ni open source.

  - Macromedia Flash Remoting MX 2004
    Es considerada la versión oficial de Flash Remoting, pero su principal desventaja es que no es gratuita ni open source.

- OpenLaszlo [22], es una plataforma open source y gratuita que permite el desarrollo y distribución de "Rich Internet Applications" (RIA's) [23]. Las aplicaciones desarrolladas con OpenLaszlo pueden ejecutarse en la mayoría de los exploradores web corriendo en los principales sistemas operativos. Para desarrollar aplicaciones con OpenLaszlo se utiliza XML y un lenguaje de scripting, a partir del cual se genera un archivo ejecutable de Flash.

3 PLANTEO DE LA SOLUCIÓN

3.1 Introducción

La búsqueda de la solución comenzó a partir de los requerimientos básicos del sistema Link-All [1]. Era necesario contemplar las necesidades del contexto que enmarca al proyecto, pero a su vez se pretendía obtener una solución lo suficientemente genérica que pudiera cubrir no solo los requerimientos del sistema Link-All, sino también los de otros Sistemas de Información.

Para lograr este objetivo, en primer lugar se consideró la interfaz de usuario. Ésta debía presentar un diseño sencillo y conocido para los usuarios, por esta razón, los primeros prototipos fueron en su totalidad para la determinación de la misma. Después de obtenida la parte visual del Sistema de Ayuda, el siguiente paso era determinar su arquitectura.

Dado que inicialmente no se tenían los conocimientos necesarios para abordar una solución genérica, se decide plantear una solución particular para Link-All, que permitiera luego, en base a la investigación realizada de las diferentes tecnologías, extenderse a una solución más completa.

Una vez logrado el Mecanismo de Ayuda Contextual para Link-All, el siguiente paso fue reutilizar la solución parcial apoyándose en los conocimientos adquiridos, para obtener el Mecanismo de Ayuda Contextual para Sistemas de Información Federados (MACSIF), un framework que permite proveer funcionalidades de ayuda en línea a aplicaciones que componen Sistemas de Información Federados a través de la Web (SIFW).

3.2 Descripción General

MACSIF es el núcleo de la solución final ya que contiene toda la lógica del negocio. Además de manejar los conceptos básicos que determinan la ayuda de una aplicación (tabla de contenido, glosario, etc.), integra conceptos que permiten establecer el contexto de ejecución de una aplicación, pudiendo de esta forma brindar ayuda efectivamente sensible al contexto. Define operaciones de alto nivel para acceder y consultar dicha ayuda, de manera tal que se independiza del formato en el que se describe la misma. A su vez, y de acuerdo a la tendencia actual a “Service Oriented Architecture” (SOA) [32], MACSIF provee un mecanismo por el cual la ayuda de una aplicación podría ser obtenida tanto de forma local como remota.

Este framework está orientado a Sistemas de Información Federados a través de la Web. Un SIFW está compuesto, a grandes rasgos, por un conjunto de aplicaciones que comparten datos y servicios. A su vez, estas aplicaciones se identifican por un código y pueden estar ubicadas en uno o varios servidores, cada una de ellas tiene asociado un conjunto de ayudas, una por cada idioma que maneje la aplicación.

El contexto de ejecución define cómo será presentada la ayuda y está compuesto principalmente por la siguiente información del usuario:

- tarea que esta realizando
- perfil

De este modo, dos usuarios que solicitan los servicios del sistema, podrían obtener ayudas diferentes, en el caso que los contextos de ejecución no fueran los mismos.

Por lo tanto, los objetivos del framework son:

- brindar ayuda en línea a los usuarios de las aplicaciones de un SIFW
- proveer un mecanismo por el cual sea posible adaptar el framework a las características de un SIFW especifico
- permitir la inclusión de utilitarios y/o servicios que asistan en la concreción de los puntos anteriores

3.2.1 Ayuda en Línea

MACSIF ofrece a los usuarios de las aplicaciones las funcionalidades básicas de ayuda, y las maneja de la forma que se describe a continuación.

La visualización de los contenidos de una ayuda es la principal funcionalidad que se le brinda al usuario. La tabla de contenido brinda una forma sencilla de navegar a través de los mismos y está representada en una

estructura de árbol. El Índice y el Glosario se manejan de la forma habitual, un conjunto de palabras claves asociadas a contenidos y términos asociados a definiciones, respectivamente.

Para la búsqueda en contenidos, MACSIF provee un mecanismo por el cual un usuario puede realizar búsquedas dentro de la ayuda de la aplicación actual o en la ayuda de otras aplicaciones, especificando un texto o un patrón. Para el caso de los favoritos permite agregar, eliminar y consultar contenidos.

3.2.2 Mecanismo de Adaptación a un SIFW

3.2.2.1 Identificador de Contexto y Mapeos

Para poder proporcionar ayuda de acuerdo a la tarea que está realizando el usuario es necesario tener una correspondencia entre los contenidos de las ayudas y las posibles tareas que ofrece una aplicación. Como forma de generalizar esta idea, MACSIF define el concepto de *contextId*, que en su forma más sencilla podría ser el nombre de una tarea.

Por lo tanto, se definen los mapeos *(mappings)*, como el conjunto de correspondencias entre los *contextId* y los contenidos de ayuda de una aplicación. Un *contextId* puede tener asociado varios contenidos, pero sólo uno de ellos se define como el contenido principal, es decir, el primero en ser mostrado (para ese *contextId*), en el momento que se solicita la ayuda. MACSIF también permite definir la estructura del *contextId*, o sea, permite especificar, para un SIFW específico, qué elementos determinan el contenido a presentar. A continuación, en la figura 4, se presentan algunos ejemplos de los mismos.

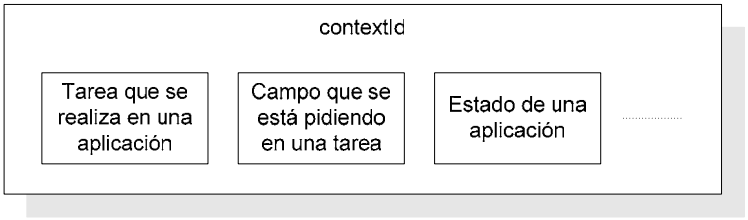


Figura 4

3.2.2.2 Obtención del Contexto

Como se mencionó anteriormente, el contexto de ejecución contiene dos elementos importantes, la tarea que está realizando el usuario y el perfil del mismo.

Dentro de la tarea que se esta realizando se distinguen:

- la aplicación que está ejecutando el usuario
- el lenguaje elegido por éste
- el *contextId* actual (*currentContextId*), que como se explicó anteriormente determina el contenido de ayuda a mostrar

Por otro lado, el perfil del usuario determina a qué información puede acceder el mismo, por ejemplo:

- todos los *contextId* accesibles para el usuario
- todas las aplicaciones en las que tiene permiso de acceso

Es importante destacar que MACSIF permite definir la forma en que se obtiene esta información, ya que es muy factible que varíe de un SIFW a otro.

3.2.3 Utilitarios

3.2.3.1 Proveedores de Ayuda

Un proveedor de ayuda (*HelpProvider*), es el encargado de proporcionar los elementos (contenidos, índice, glosario, etc.), que componen la ayuda de una aplicación en un determinado idioma. Un proveedor puede proporcionar la ayuda a varias aplicaciones en varios idiomas, a su vez, la ayuda de una aplicación en un determinado idioma podría estar disponible en varios proveedores. En la figura 5 se presenta un esquema de lo mencionado antes.

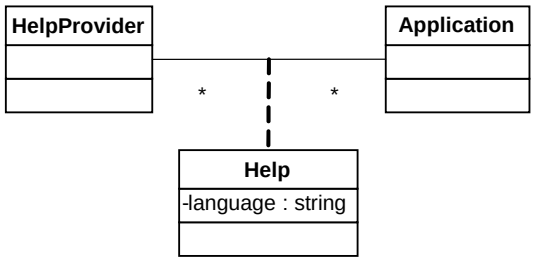


Figura 5

MACSIF puede interactuar con diversos proveedores, permitiendo así que cada aplicación seleccione la forma de proveer la ayuda de acuerdo a sus necesidades. Por lo tanto, el framework, permite especificar:

- el conjunto de proveedores con los que trabajará
- el orden en que se buscará ayuda en dichos proveedores, definiendo una prioridad
- el proveedor que una aplicación utilizará por defecto

Como se puede observar en la figura 6, para cada proveedor de ayuda debe definirse un *mapping* (conjunto de correspondencias entre los *contextId* y los contenidos de ayuda de una aplicación).

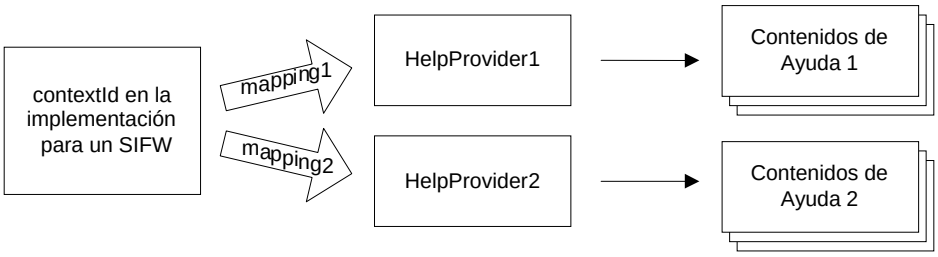


Figura 6

Es destacable el hecho de que un proveedor no está asociado a un SIFW específico, por lo que varios SIFW que utilicen MACSIF podrían hacer uso del mismo proveedor para obtener las ayudas de sus aplicaciones.

3.2.3.2 Proveedores de búsqueda

Estos componentes (*SearchProvider*), proveen funcionalidades de búsqueda de texto en los contenidos de las ayudas, tanto de una como de todas las aplicaciones. Para esto, es posible crear un motor de búsqueda o utilizar uno existente. Un proveedor de búsqueda no está asociado a un SIFW específico, por lo que varios SIFW que implementen MACSIF pueden utilizar el mismo proveedor.

3.2.4 Esquema General

En este punto se pretende fijar los conceptos planteados anteriormente, presentando en la figura 7 un esquema general de lo que sería la solución final de una implementación del framework para un SIFW.

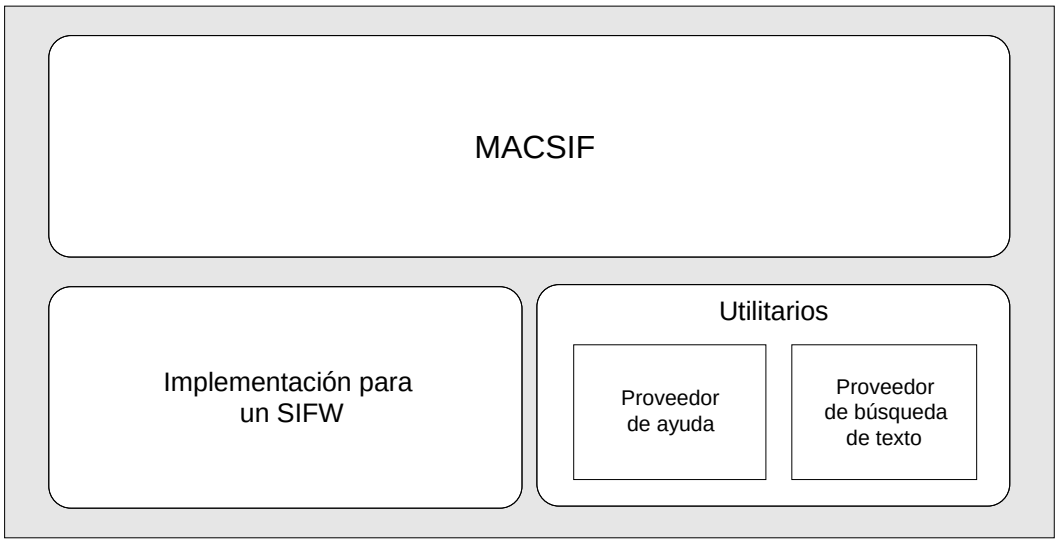


Figura 7

3.3 Cuadro Comparativo

En la tabla 1 se presenta un cuadro comparativo entre los diferentes sistemas de ayuda investigados y el Mecanismo de Ayuda Contextual.

	HTML Help	Sun Java Help	Flash Help	Help Oracle	MACSIF
Tabla de Contenido	✓	✓	✓	✓	✓
Índice	✓	✓	✓	✓	✓
Buscar	✓	✓	✓	✓	✓
Glosario	✓	✓	✓	✓	✓
Comp. de Navegación	✓	✓	✓	✓	✓
Ayuda para Aplicaciones Web				✓	✓
Favoritos	✓				✓
Manejo de distintos Formatos de Help					✓
Seguridad					✓
Orientada a SIFW					✓

Tabla 1

4 DISEÑO

4.1 Interfaz de Usuario

Dado que la ayuda debe ser presentada en un formato que le resulte conocido al usuario [37] de las aplicaciones, para el diseño de la interfaz se debió respetar los estándares utilizados por todos los sistemas de ayuda. Teniendo en cuenta que se estaba desarrollando un sistema para la web, fue necesario conocer los requisitos básicos de diseño y usabilidad [33].

En la figura 8 se presenta la pantalla inicial del Sistema de Ayuda Contextual, y a continuación se listan los conceptos que se tomaron en cuenta en el momento del diseño de la interfaz de usuario:

1. Se dedicó la mayor parte del espacio a contenido de interés para el usuario, dejando un porcentaje relativamente pequeño a los accesorios de navegación.

2. Se busco la simplicidad, tanto en el diseño como en el color de fondo (se utilizo siempre la misma tonalidad).

3. El sistema fue probado en diferentes navegadores (Internet Explorer, Mozilla, Firefox y Netscape Navigator). Con respecto al sistema operativo, se realizaron pruebas en Microsoft Windows y Linux.

4. El tiempo de respuesta es uno de los principales factores determinantes de la usabilidad de un sitio, y aunque se utilizaron componentes Flash, se buscó en todo momento obtener una página liviana.

5. Los enlaces en los nodos de la tabla de contenido aportan información acerca del tema que se presentará en el contenido de la derecha. No se utilizaron los colores estándares azul y púrpura para los nodos no visitados y visitados respectivamente, porque éstos no son enlaces a páginas web, sino que referencian contenidos de ayuda. Además, en este caso se buscó el clásico estilo de una tabla de contenido de un sistema de ayuda.

6. Se utilizaron textos cortos y justificados a la izquierda, con fuente lo suficientemente grande y sin movimiento. Además, se evito el texto todo en mayúsculas.

7. Se busco aumentar la legibilidad en aquellos textos que contienen imágenes, cuando el usuario pasa el puntero del ratón sobre uno de estos textos, el tamaño de la imagen se agranda y el texto se vuelve en negritas.

8. Dado que el Mecanismo de Ayuda es contextual la página principal puede contener cualquiera de los contenidos de ayuda que estén disponibles para el usuario, por lo que el botón *Inicio* aparece en todo momento y es el encargado de posicionar al sistema en el contenido que se presento cuando el usuario solicito la ayuda.

9. La interfaz de navegación permite responder a las preguntas “¿dónde estoy?”, “¿dónde he estado?” y “¿a dónde puedo ir?”, siempre haciendo referencias a contenidos de ayuda.

10. Se le ofrece al usuario libertad de navegación, ya que luego de solicitar la ayuda de una aplicación, podrá continuar consultando información acerca de otras aplicaciones en las cuales dicho usuario tiene permiso de acceso. Para esto solo deberá seleccionar en el combo de aplicaciones la de su interés.

Aunque es muy difícil (probablemente imposible), lograr el diseño perfecto, se busco fuertemente respetar cada uno de los conceptos de usabilidad, y se dejan abiertas todas las posibilidades de mejora.

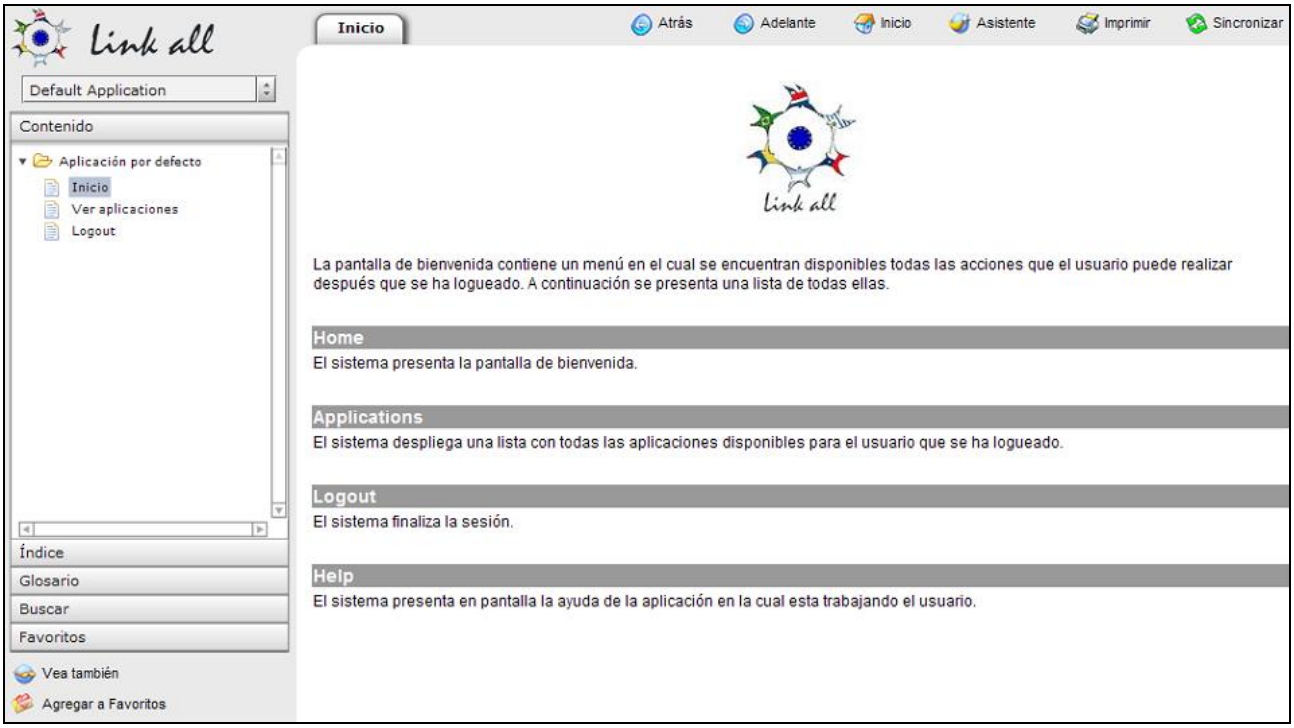


Figura 8

4.2 Arquitectura

Para el planteo de la arquitectura de la solución final[3], fueron de gran importancia los conocimientos adquiridos acerca de UML y Patrones de Diseños [29] [30], ya que la falta de los mismos, fue lo que motivó a plantear inicialmente una solución parcial que permitiera a los integrantes del grupo profundizar e investigar más sobre las diferentes posibilidades de diseño. A continuación se presenta la descripción completa de la arquitectura del Mecanismo de Ayuda Contextual, basada en el Diseño de Software con Patrones [13] [31].

El framework, MACSIF, está formado por un conjunto de componentes, y un grupo de interfaces que son la forma de comunicación entre éste, los utilitarios y el SIFW específico que hace uso de dicho framework.

En la figura 9 se puede observar que los componentes que conforman una determinada implementación de MACSIF se pueden dividir en tres grandes grupos:

- componentes de MACSIF
- componentes específicos para un SIFW
- componentes utilitarios

[3] Para mayor información de la arquitectura puede consultarse el Anexo [D]

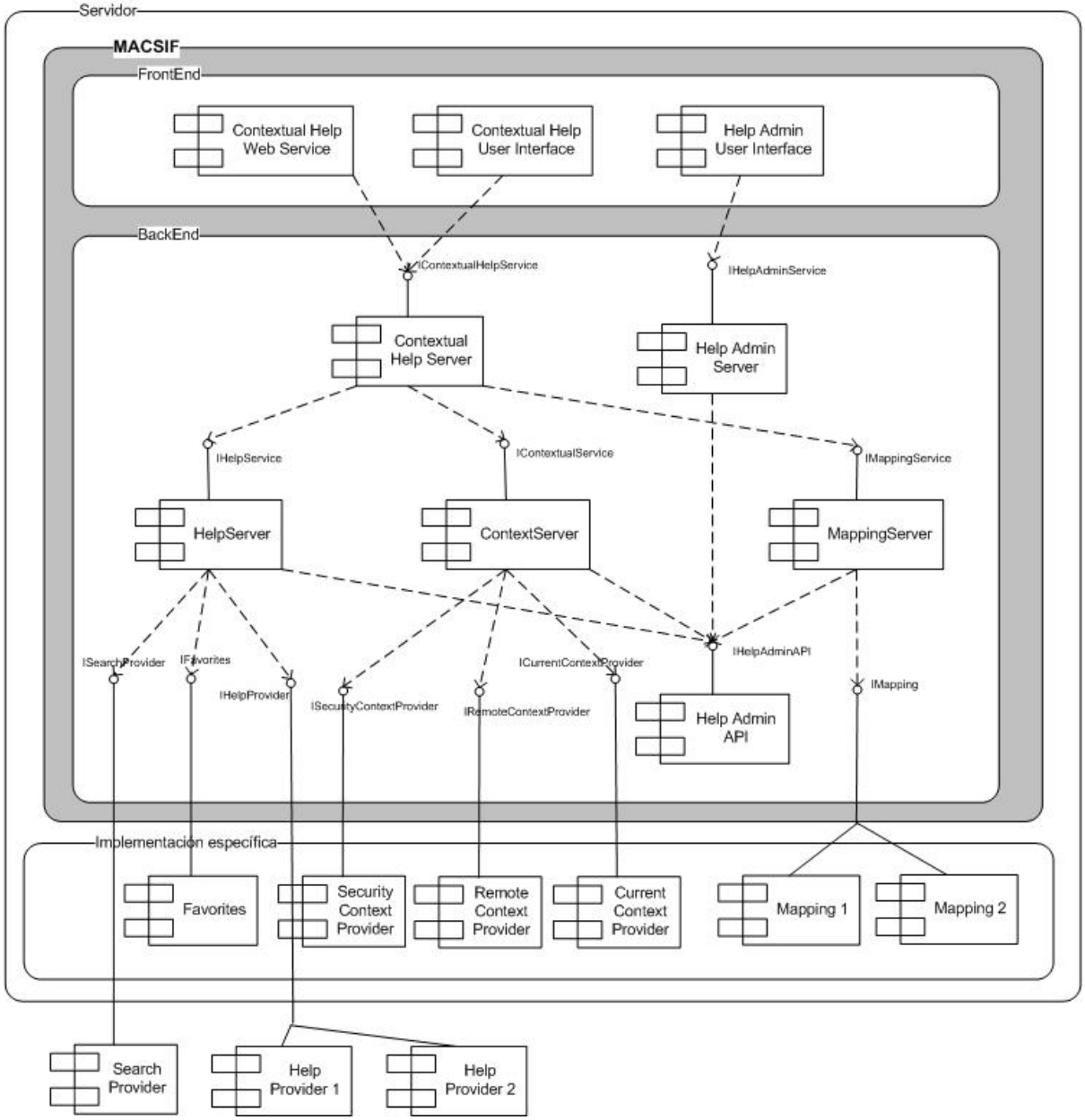


Figura 9

4.2.1 Framework

Los componentes de MACSIF se dividen en dos grandes partes, el Frontend y el Backend.

4.2.1.1 Frontend

En el frontend se encuentran los componentes encargados de atender las solicitudes no solo de los usuarios sino también de otros sistemas. A continuación se presenta una breve descripción de cada uno de dichos componentes:

- *ContextualHelpUserInterface*
  Es el componente encargado de interactuar con los usuarios de las aplicaciones. Cuando un usuario solicita la ayuda, éste la despliega en pantalla. A su vez, ofrece varias formas de navegación a través de él y provee varias funcionalidades de ayuda en línea (tabla de contenido, el índice, el glosario, favoritos, búsqueda de texto, etc.).

- *HelpAdminUserInterface*
  Es el componente responsable de interactuar con los usuarios administradores. A partir de esta interfaz de usuario es posible configurar todos los parámetros que definen el comportamiento del framework,

algunos de esos parámetros pueden ser los proveedores de ayuda con los que se trabajará o los proveedores por defecto de las aplicaciones.

- *ContextualHelpWebService*
  Este componente expone los métodos del *ContextualHelpServer* a través de un Web Service. Esto resulta muy conveniente cuando la comunicación a través de RMI no es posible, o cuando se desea interactuar con el *ContextualHelpServer* utilizando componentes que no están implementados en Java.

Los componentes *ContextualHelpUserInterface* y *HelpAdminUserInterface* implementan el patrón de diseño Model View Controller (MVC), la interfaz recolecta información que es enviada a un Controller que decide (de acuerdo a dicha información), quién es el encargado de manejarla y delega el pedido al responsable del mismo, el cual podría consultar otros componentes (modelo). Finalmente, el Controller retorna la vista adecuada.

4.2.1.2 Backend

En el backend se encuentran los componentes que contienen la lógica del negocio y el componente de acceso a datos de configuración. Una descripción del mismo se presenta a continuación:

- *ContextualHelpServer*
  Este es el principal componente del framework, en el se encuentra la mayor parte de la lógica del negocio. *ContextualHelpServer* implementa el patrón Bussiness Delegate (BD), ya que recibe pedidos del componente *ContextualHelpUserInterface*, los procesa delegando tareas particulares a otros componentes, y los responde. Cuando este componente recibe un pedido de ayuda, interactúa con el componente *ContextServer* para obtener los datos del contexto de ejecución:
  - aplicación
  - lenguaje
  - *contextId* actual
  - aplicaciones a las que tiene permiso
  - *contextId* accesibles para el usuario

  Luego de obtenida esta información, solicita al *HelpServer* la ayuda para la aplicación en el lenguaje especificado. También necesita interactuar con *MappingServer* para conocer, a partir del *currentContexId*, qué contenido debe estar seleccionado al momento de mostrar la ayuda y dados todos los posibles *contextId*, qué contenidos puede visualizar el usuario. Finalmente, el *ContextualHelpServer* modifica la ayuda obtenida por el componente *HelpServer*, eliminando los contenidos en los que no tiene permiso el usuario, y seleccionando el contenido que se corresponde con el *currentContextId*.

- *HelpServer*
  Recibe pedidos del componente *ContextualHelpServer* y se encarga de retornar la ayuda de las aplicaciones en un determinado idioma. En caso de no ser especificado el proveedor (*HelpProvider*), este componente debe consultar a *HelpAdminServer* la lista de proveedores con los que se está trabajando y buscar en cada uno de ellos la ayuda de la aplicación en el idioma especificado. Si la aplicación tiene un proveedor por defecto (información que también debe ser consultada a *HelpAdminServer*), será éste el primer *HelpProvider* en ser consultado. Además, el *HelpServer* interactúa con los componentes que gestionan los contenidos favoritos y las búsquedas de texto en los contenidos de ayuda, ya que también se encarga de proveer esta información.

- *ContextServer*
  Este componente es el responsable de proporcionar toda la información relativa al contexto de ejecución, para esto debe interactuar con los componentes que implementan *ICurrentContextProvider*, *ISecurityContextProvider* y *IRemoteContextProvider*.

- *MappingServer*
  Es el componente encargado de, a partir de un *contextId*, obtener el contenido principal (el primero en ser desplegado en pantalla en el momento que el usuario solicita la ayuda), y los contenidos relacionados de un determinado *HelpProvider*. Este componente debe interactuar con *HelpAdminServer* para conocer el *Mapping* que se corresponde con el proveedor de ayuda actual.

- *HelpAdminServer*
  Administra todos los parámetros que definen el comportamiento del framework. *HelpAdminServer* interactúa con otros componentes del backend para dar a conocer el valor de cada uno de los parámetros. A su vez, procesa y responde a los pedidos del componente *HelpAdminUserInterface*, utilizando para esto los servicios del *HelpAdminAPI.*

- *HelpAdminAPI*
  Es un componente utilitario cuya tarea es encapsular el acceso a los datos de configuración del framework.

4.2.2 Implementación específica para un SIFW

Estos componentes son los que permiten adaptar el framework, MACSIF, a las características particulares de un determinado SIFW.

4.2.2.1 Información de Contexto

Los componentes que se presentan a continuación son los responsables de obtener la información que determina el contexto de ejecución. Para poder obtener dicha información, estos componentes necesitan datos provenientes, en general, de la interfaz de usuario. Estos datos están encapsulados en un objeto que implementa la interfaz *IContext.*

- *SecurityContextProvider*
  Obtiene toda la información del contexto de ejecución relativa a los permisos del usuario. El componente *SecurityContextProvider* debe implementar la interfaz *ISecurityContextProvider* provista por MACSIF.

- *CurrentContextProvider*
  Es el componente encargado de obtener toda la información del contexto de ejecución relativa a la tarea actual del usuario, y debe implementar la interfaz *ICurrentContextProvider* que provee MACSIF.

- *RemoteContextProvider*
  Este componente es el encargado de obtener la información del contexto de ejecución remoto del usuario, y debe implementar la interfaz *IRemoteContextProvider* definida en MACSIF.

4.2.2.2 Mapeos

Como se definió anteriormente, los mapeos son un conjunto de correspondencias entre *contextId* y contenidos de ayuda. Dado que el *contextId* es específico de cada SIFW, los mapeos también lo son. Además, como los mapeos hacen referencia a los contenidos de un determinado *HelpProvider*, se deberá definir un componente *Mapping* para cada *HelpProvider*.

- *Mapping*
  Estos componentes son los responsables de conocer los contenidos asociados a un determinado *contextId*. Cada uno de estos componentes debe implementar la interfaz *IMapping* definida en MACSIF.

4.2.2.3 Favoritos

Para poder almacenar los contenidos favoritos de un usuario, es necesario identificar al mismo de alguna forma. La forma de identificación es particular del SIFW, es por esto que MACSIF define la interfaz *IUserId*. El componente que implementa esta interfaz debe incluir los elementos que identifican a un usuario en el SIFW específico.

- *Favorites*
  Es el componente encargado de administrar (agregar, consultar y eliminar), los contenidos favoritos de cada usuario, y debe implementar la interfaz *IFavorites* que provee MACSIF.

4.2.3 Utilitarios

4.2.3.1 Help Providers

Los *HelpProviders* son los componentes encargados de proveer la información que compone la ayuda de una aplicación. Todo proveedor debe implementar la interfaz *IHelpProvider*. MACSIF brinda la posibilidad de que estos componentes puedan accederse tanto de forma local como remota.

4.2.3.2 Search Providers

Los *SearchProviders* son los componentes responsables de realizar búsquedas de texto en los contenidos de ayuda que proveen los *HelpProviders*. Cada *SearchProvider* debe implementar la interfaz *ISearchProvider* provista por MACSIF.

5 CASO PARTICULAR: PLATAFORMA LINK-ALL

Las aplicaciones de Link-All comparten servicios[4] e información. Los servicios son brindados a través de componentes públicos que exponen funcionalidades a dichas aplicaciones. Estos servicios son utilizados en la implementación de MACSIF para este caso particular.

5.1 Información de Contexto

- Contexto Actual
  El contexto está determinado por el número de sesión interna, el cual es obtenido a través de una cookie grabada por la plataforma Link-All.

- Tarea actual
  Una vez determinado el identificador de sesión, se pueden utilizar los servicios brindados por el componente *SessionServer* de la plataforma Link-All, para obtener los datos que determinan el contexto actual de ejecución. En esta implementación, el contexto de ejecución (*contextId*) está determinado por el nombre de la funcionalidad de la aplicación que se está ejecutando y por la operación registrada (por ejemplo "uso de la funcionalidad", "error en la funcionalidad", etc.).

  El *contextId* (en este caso funcionalidad y operación), determina el contenido de ayuda a mostrar según los *Mappings* antes definidos. Por esta razón, para una misma funcionalidad se pueden presentar dos ayudas diferentes (figura 10), dependiendo de los siguientes casos:
  - la funcionalidad se ejecuta correctamente (y el usuario solicita ayuda)
  - el usuario comete un error en el momento de ejecutar la funcionalidad

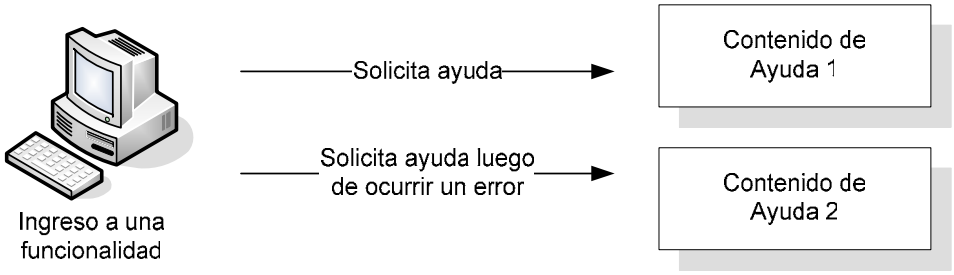


Figura 10

Un dato importante que también se obtiene del componente *SessionServer* (a través del identificador de sesión), es el perfil de la sesión, el cual proporciona el idioma seleccionado por el usuario. Esta información determina el idioma que se utilizará en el momento de presentar la interfaz de usuario. Los contenidos de la ayuda también se presentan en el idioma especificado, pero en el caso que los contenidos no se encuentren disponibles en dicho idioma, se considera uno por defecto.

- Seguridad
  Para poder desplegar en pantalla sólo aquella información en la que tiene permiso el usuario, se debe interactuar con el componente *SecurityServer* de Link-All. Los permisos están determinados por las funcionalidades a las que puede acceder dicho usuario.

5.2 Interacción con la Plataforma Link-All

En el punto 4.2.2 se introdujo al lector en los componentes que permiten la interacción entre un SIFW específico y el framework, en este ítem se pretende detallar la implementación para un caso particular: el Sistema de Información Link-All.

En la figura 11 se observa cómo los componentes que implementan las interfaces de MACSIF interactúan con la plataforma Link-All.

[4] Para una información detallada de los servicios de la plataforma Link-All, consultar la Documentación Técnica en el Anexo [E].

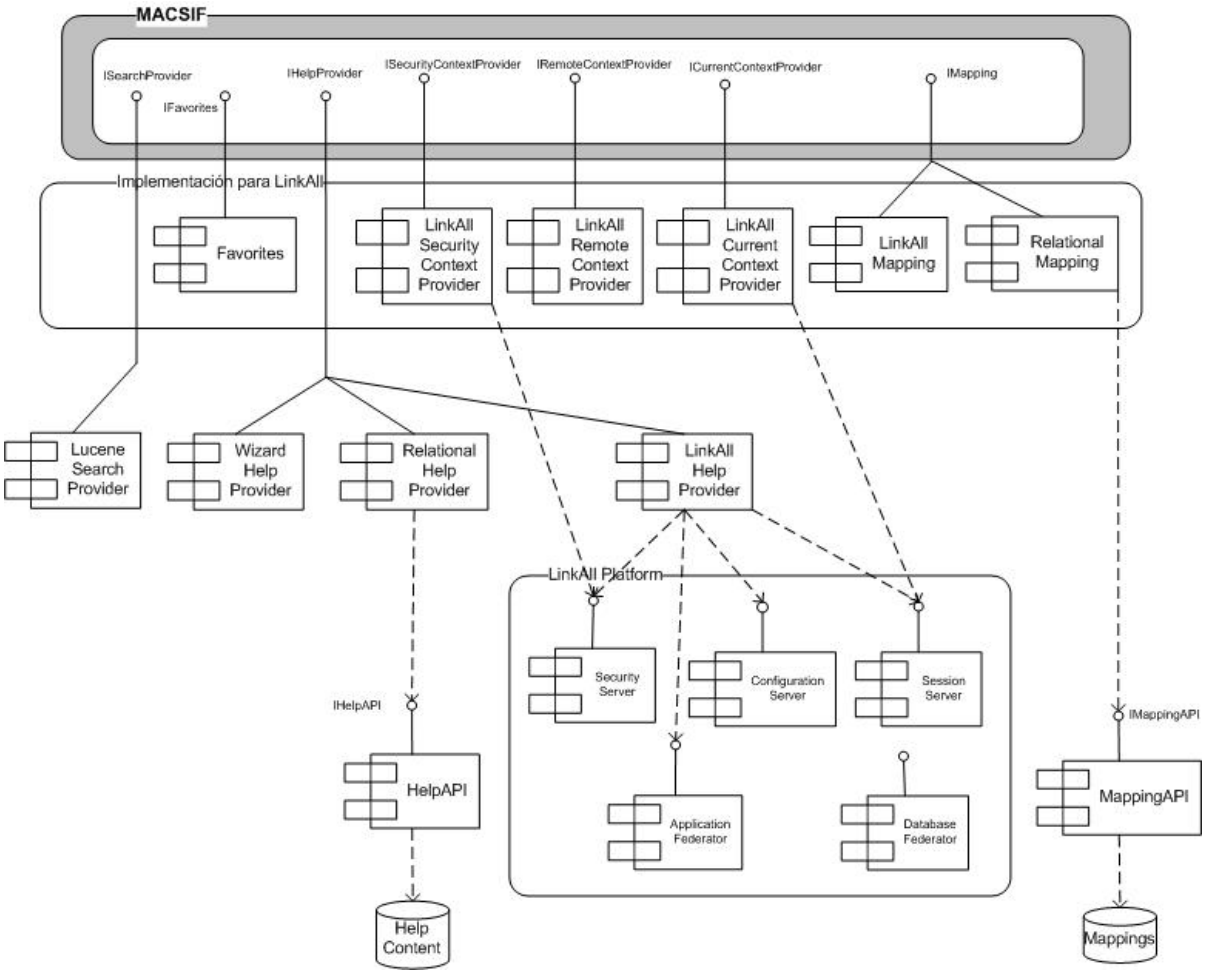


Figura 11

5.2.1 Componentes

Para el sistema Link-All se desarrollaron un grupo de componentes que implementan las interfaces del framework, a continuación se presenta una descripción de cada una de ellas:

- *LinkAllCurrentContextProvider*
  Este componente implementa la interfaz *ICurrentContextProvider*, que provee operaciones para determinar la tarea actual (en el caso del sistema Link-All, la funcionalidad que se esta ejecutando), del usuario.

- *LinkAllSecurityContextProvider*
  Es el componente que implementa la interfaz *ISecurityContextProvider*, que provee operaciones para obtener los permisos de los usuarios.

- *LinkAllRemoteContextProvider*
  Es el componente que implementa la interfaz *IRemoteContextProvider*, que provee operaciones para determinar si una aplicación se ejecuta de forma local o remota, para el último caso, obtiene los datos del contexto remoto.

- *LinkAllMapping*
  Este componente implementa la interfaz *IMapping* (para el proveedor de ayuda *LinkAllHelpProvider*), y provee los mapeos entre las tuplas funcionalidad – operación y los contenidos de ayuda asociados a cada una de ellas. La correspondencia es trivial, ya que los códigos de contenidos coinciden con los códigos de funcionalidad.

- *RelationalMapping*
  Este componente también implementa la interfaz *IMapping* (para el proveedor de ayuda *RelationalHelpProvider*). En este caso la correspondencia es más compleja, ya que la misma está

grabada en una Base de Datos, por lo que este componente debe encargarse de obtener los datos. Para esto, utiliza los servicios del componente *MappingAPI*, la cual se describe a continuación.

- *MappingAPI*
  Es un componente utilitario, cuya tarea es encapsular el acceso a la Base de Datos, en donde se graba la información de mapeos de Link-All con la ayuda provista por el *RelationalHelpProvider*. Dado que esta Base de Datos podría no estar en el mismo servidor que la plataforma Link-All, este componente es un EJB y brinda la interfaz *IMappingAPI* para exponer sus servicios.

  La Base de Datos almacena no sólo la información de los mapeos, sino también todos los datos correspondientes a las ayudas. Pero dado que los mapeos son independientes de la ayuda en sí, éstos eventualmente podrían ser guardados en otra Base de Datos, por lo que debería adaptarse el componente *MappingAPI* a estos cambios.

- *LinkAllFavorites*
  Es el componente responsable de implementar las funcionalidades para la gestión de contenidos favoritos de un usuario, implementando para esto las interfaces *IFavorites* y *IUser* de MACSIF.

5.3 Utilitarios para la Plataforma Link-All

El framework interactúa con una serie de utilitarios que debieron ser implementados para que MACSIF provea las funcionalidades de ayuda a la plataforma Link-All. A pesar de que estos utilitarios fueron desarrollados para este caso particular, son independientes del SIFW.

- *RelationalHelpProvider*
  Es el principal proveedor de ayuda de la implementación para Link-All, ya que obtiene toda la información requerida por el framework de una Base de Datos relacional. Para accederla, utiliza los servicios del componente *HelpAPI* (que se describe en el siguiente punto).

  Las aplicaciones brindan toda o parte de la información requerida por este proveedor a través de un archivo con formato XML.

- *HelpAPI*
  Es un componente utilitario, cuya tarea es encapsular el acceso a la Base de Datos en la que se almacena la información de ayuda de la plataforma Link-All (y eventualmente, de otras plataformas). Este componente es un EJB y provee la interfaz *IHelpAPI* para exponer sus servicios.

  Para encapsular el acceso a los datos, el cual depende del manejador (DBMS) que se esté utilizando, implementa los patrones de diseño Value Objects (VO) y Data Access Object (DAO).

- *LinkAllHelpProvider*
  A través de la implementación de la interfaz *IHelpProvider*, este componente provee ayuda con información que obtiene directamente de la plataforma Link-All, en la forma que se muestra a continuación:

  Este proveedor es consultado cuando no se encuentra ayuda en el *RelationalHelpProvider* para:
  - la aplicación
  - el idioma del usuario
  - el idioma por defecto

  Su objetivo es ofrecer una ayuda básica pero completa acerca de la funcionalidad que se esta ejecutando y de cómo ésta se vincula con otras funcionalidades de la aplicación. A continuación se presenta al lector cómo el componente *LinkAllHelpProvider* crea los objetos que forman la ayuda de una aplicación, utilizando los servicios provistos por el *ApplicationFederator* y por el *SecurityServer* de la plataforma Link-All:

  - Tabla de contenidos
    La raíz contiene el nombre de la aplicación, mientras que las hojas contienen los nombres de las funcionalidades de dicha aplicación.

- Contenidos
  Los contenidos que se muestran se generan dinámicamente. Cuando se selecciona en la tabla de contenido la raíz, el contenido que se presenta despliega el nombre y la descripción de la aplicación, así como su número de versión y la fecha de instalación en la plataforma Link-All, además se listan las funcionalidades que la conforman. Cuando se selecciona una de ellas (una hoja de la tabla de contenido), se muestra el nombre y la descripción de la misma.

- Índice
  Las entradas de índice se corresponden con los nombres de las funcionalidades de la aplicación, y tienen como único contenido asociado, el mismo que se presenta cuando se seleccionan dichas funcionalidades en la tabla de contenido.

- Glosario
  Los términos del glosario se corresponden con los nombres de las funcionalidades de la aplicación, y la definición que se presenta es la descripción de las mismas.

- *WizardHelpProvider*
  Su objetivo es proveer al usuario de guías o sugerencias de acción, basándose en una serie de heurísticas, como por ejemplo acciones frecuentes de otros usuarios.

- *LuceneSearchProvider*
  Este componente provee funcionalidades de búsqueda de texto en los contenidos de las ayudas, tanto de una como de todas las aplicaciones, e implementa la interfaz *ISearchProvider*. Para la plataforma Link-All, se desarrolló un utilitario basado en la API de búsqueda Lucene [25].

  La búsqueda se realiza en un índice compuesto por documentos, cada uno de ellos está formado por las duplas (nombre de campo,valor). Sobre estas duplas se realiza la búsqueda y además se puede obtener el valor de las mismas. En la figura 12 se observan los elementos que componen un índice de Lucene.

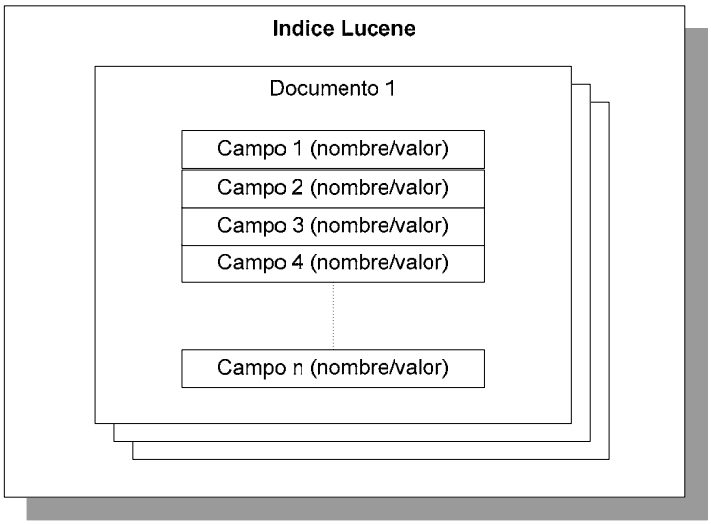


Figura 12

Antes de comenzar una búsqueda, los índices deben estar construidos, por lo que este proveedor (*LuceneSearchProvider*), brinda funcionalidades tanto para crear los índices como para realizar búsquedas en ellos. Cuando se crean los documentos que componen un índice, se especifican los campos que incluirán. En este proveedor, cada documento corresponde a un contenido de ayuda, y se definen los siguientes campos para cada uno de ellos:

- plataforma
- proveedor de ayuda
- aplicación
- lenguaje
- código de contenido
- texto asociado al contenido

La búsqueda se realiza en el campo de texto, obteniéndose un conjunto de documentos resultantes. El resto de los campos indican qué contenidos se asocian con los resultados de la búsqueda y qué *HelpProvider* los provee, de esta forma el framework obtiene todos los datos que necesita para mostrar la información.

La estructura de campos definida para los documentos del índice de búsqueda, permite además que la misma pueda realizarse sobre una aplicación o sobre todas las aplicaciones que forman parte del SIFW (sean éstas locales o remotas). Esta característica ofrece muchas posibilidades al usuario, ya que puede buscar información relacionada con un tema de su interés en cualquier aplicación de la plataforma Link-All, lo que permite, a través de MACSIF, ver a todas las aplicaciones como si fueran una sola. A su vez, se obtiene una forma de "invitar" al usuario a conocer las posibles aplicaciones en las que él podría trabajar, lo que resulta una funcionalidad muy importante para la plataforma Link-All.

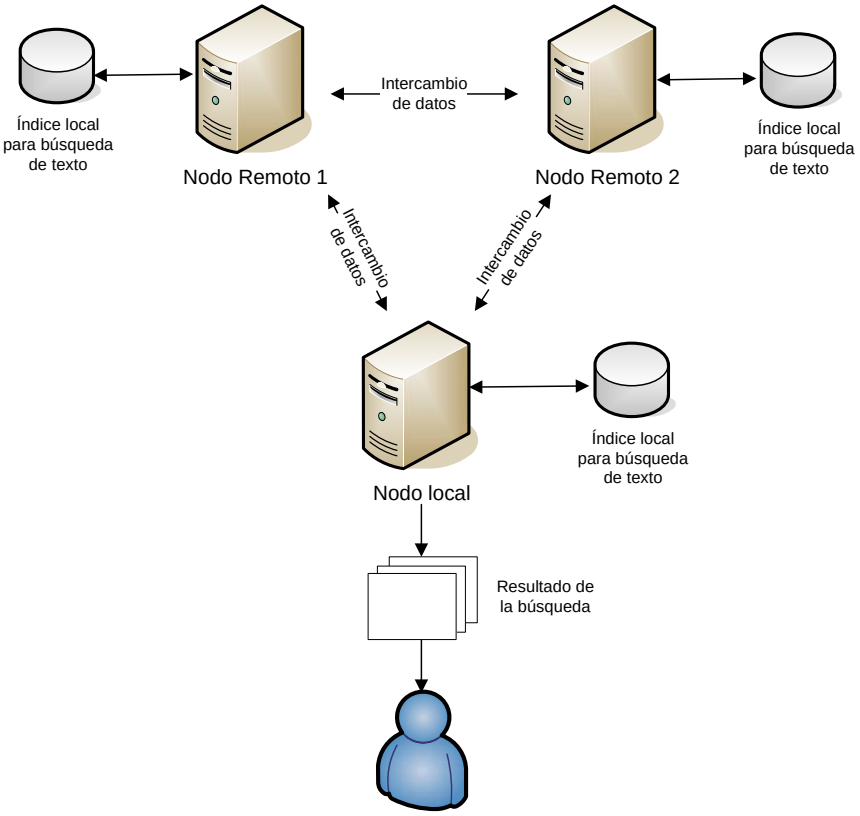


Figura 13

En la figura 13 se muestra cómo los nodos de la plataforma intercambian la información de los contenidos de ayuda de cada *HelpProvider* que gestiona cada uno de ellos. De esta forma, todos los contenidos de ayuda de las aplicaciones locales al *Nodo Remoto 1* estarán disponibles para búsquedas de texto en los otros dos nodos (*Nodo Local* y *Nodo Remoto 2*), lo mismo ocurrirá con las aplicaciones locales al *Nodo Remoto 2* y con las aplicaciones locales al *Nodo Local*.

6 TECNOLOGÍAS

El estudio de las tecnologías involucradas en los requerimientos está fuertemente ligado al contexto de los Sistemas de Información Federados a través de la Web. A continuación se describen aquellas que fueron utilizadas para la implementación del Mecanismo de Ayuda Contextual.

6.1 Java

Se analizan las distintas tecnologías y se considera Java [13] [14] el más conveniente de los lenguajes de programación, ya que éste es el lenguaje utilizado para desarrollar la plataforma Link-All. Si bien es factible la integración de Java con otras tecnologías (como pueden ser php o perl), no es la finalidad de este proyecto centrarse en estos temas de investigación.

6.2 JBoss

Se optó por JBoss [15] [16] como servidor, pero dado que éste sólo proporciona un contenedor de EJB´s, para la compaginación de servlets y JSP se utilizó en conjunción con Tomcat.
Tomcat viene incluído con JBoss, lo que resulta una ventaja a la hora de invertir tiempo de coordinación y configuración entre el contenedor de EJB´s y el contenedor web.

JBoss cuenta con varias ventajas, además de tratarse de un producto Open Source, sus requerimientos son mínimos en cuanto a memoria y disco, ya que corre muy eficazmente en una máquina con 64 megabytes de RAM. Tiene una base de datos SQL interna para permitir la persistencia de los beans, que se configura automáticamente con el seteo del JBoss.

6.3 MySQL

Como servidor de Base de Datos relacional se seleccionó MySQL [17]. Una de las características importantes es que también se trata de un producto Open Source, además su conectividad, velocidad y seguridad hacen de MySQL un servidor bastante apropiado para acceder a Bases de Datos en Internet.

6.4 OpenLaszlo

OpenLaszlo [22], es una plataforma que permite el desarrollo y distribución de aplicaciones que pueden ejecutarse en la mayoría de los exploradores web corriendo en los principales sistemas operativos. Para desarrollar aplicaciones con OpenLaszlo se utiliza XML y un lenguaje de scripting, a partir del cual se genera un archivo ejecutable de Flash.

Se investigó la herramienta, realizando diferentes pruebas a partir de distintos archivos XML. También se planteó la posibilidad de utilizar Flash [19] para la realización de la interfaz, por lo que se llevo a cabo una investigación de Flash Remotig (ver punto 2.4), como posible herramienta para la integración Flash-Java y por las siguientes razones se decidió utilizar Openlaszlo para el desarrollo de la interfaz de usuario:

- OpenLaszlo es una herramienta open source, si bien existen opciones gratuitas (OpenAMF y la versión estándar de FlashORB de Flash Remoting para J2EE), el resto de las características de OpenLaszlo fueron de mayor peso a la hora de tomar una decisión.

- Para desarrollar aplicaciones con OpenLaszlo es posible utilizar Eclipse a través de un plugin que provee IBM, lo que permite continuar en el mismo ambiente de trabajo.

- Considerando los tiempos del proyecto de grado, se estimó que sería más eficiente el poder obtener ejecutables Flash a partir de archivos XML, que realizar el desarrollo de los componentes de la interfaz de usuario con el lenguaje de programación ActionScript, de Flash MX.

6.5 API's utilizadas

6.5.1 Link-All

Para poder proveer funcionalidades de ayuda que dependan del contexto de operación que se está ejecutando así como del usuario y contexto de servidor, a las aplicaciones que forman parte del ambiente Link-All es necesario utilizar la API Link-All [24] que permite la integración entre el Sistema de Ayuda y la plataforma Link- All.

6.5.2 Lucene

Para la implementación de la búsqueda de texto en los contenidos de ayuda, se utilizó la API Lucene [25]. Lucene es un potente motor de búsqueda Open Source escrito completamente en el lenguaje de programación Java. Permite ser empleado en cualquier aplicación que requiera búsquedas en textos, especialmente multiplataforma.

Lucene provee un optimizador, a través de una lista de artículos, preposiciones, etc.; y un lenguaje que permite realizar consultas complejas. La búsqueda se realiza en un índice compuesto por documentos, cada uno de ellos formados por tuplas campo – valor. Cada índice debe ser construido antes de realizar una búsqueda, luego se especifican los campos que éstos incluirán. Un ejemplo de cómo definir estos campos se presentó en el punto 4.2.5 para el componente *LuceneSearchProvider*.

6.5.3 APIs Java para XML

Dado que la tabla de contenido, para el *RelationalHelpProvider*, se crea a partir de un string con formato XML, se utilizan las API's JAXP [26] (Java API for XML processing) y SAX [27] (Simple API for XML) para el manejo de las mismas.

6.5.4 JFreeChart

Es una API de Java [28] que representa gráficos en 2D y 3D, gráficos de barras, de líneas, gráficas de series de tiempo, etc. JFreeChart puede ser usado en aplicaciones, applets, servlets y JSP. En el caso del Mecanismo de Ayuda Contextual, fue utilizado en la realización de los contenidos que se presentan al solicitar el *Wizard*.

6.5.5 JUnit

Es un API de Java [41] que constituye un marco (framework) sencillo para la creación de pruebas de unidad de forma automática. Es importante destacar que permite reutilizar los códigos definidos para el testing.
JUnit distingue entre fallos y errores. Un fallo se produce porque hay una discrepancia entre el resultado esperado y el real, y se detecta mediante aserciones. Puede decirse que es un problema que puede anticiparse. Mientras que un error, se produce como consecuencia de un problema no anticipado, como puede ser un ArrayIndexOutOfBoundsException.

7 CONCLUSIONES Y TRABAJO A FUTURO

A medida que se avanzaba en la investigación para obtener la solución que cubriera los objetivos de este proyecto, surgieron ideas y bosquejos de cómo podría ser enriquecido el Mecanismo de Ayuda Contextual. Por razones académicas, fue importante ceñir los tiempos de desarrollo, lo que no implica dejar de plantear todas aquellas posibilidades que permitan mejorar la solución final al problema planteado.

Actualmente, MACSIF permite especificar un conjunto de parámetros en un archivo de configuración, por lo que se pretende que eventualmente estos parámetros sean manipulados de forma dinámica, obteniendo así un sistema más flexible aún.

Otra inquietud que resulta interesante plantear, es la implementación de una *Cache* en la cual se puedan almacenar datos, en particular datos remotos, permitiendo así mejorar los tiempos cada vez que sea necesario interactuar con una aplicación remota.

Un elemento importante de la interfaz de usuario de un sistema que se ejecuta a través de Internet, es su accesibilidad. El número de personas que acceden a la web cada vez es mayor [34]. Por esta razón, hacer páginas accesibles [35] tiene efectos secundarios positivos, ya que no solo podrán acceder personas con discapacidades, sino que también serán más accesibles para las diferentes tecnologías de navegación por la web, como por ejemplo los robots indexadores de los motores de búsqueda. Todas estas características hacen considerar que en el futuro el Mecanismo de Ayuda Contextual pueda contar con una interfaz de usuario que no solo respete los principios de usabilidad, sino que también sea accesible, contemplando así, las necesidades de todos los usuarios [35].

Consultando algunos sitios web surgió la inquietud de una nueva tecnología, los Portlets [21]. Un Portlet es un componente lógico de un Portal Web, aplicación web que puede proporcionar al usuario un contenido personalizado (idioma, contenidos, etc.). Los sitios más importantes presentan una tendencia al uso de los mismos, un claro ejemplo de un portal web que utiliza portlets es “my.yahoo.com”. Por esta razón, resulta sumamente interesante realizar una investigación exhaustiva sobre dicha tecnología y estudiar la posibilidad de que el Mecanismo de Ayuda Contextual sea integrable y compatible con estos diferentes tipos de sistemas.

Se considera que el modelo de proceso iterativo e incremental utilizado en el proyecto fue el adecuado, ya que permitió encarar el problema desde una perspectiva muy general, para luego ir incorporando mejoras sucesivas hasta obtener la solución final. Se esta ampliamente conforme con el producto obtenido, se logró el principal objetivo, desarrollar utilitarios y componentes de ayuda para Sistemas de Información Federados, y en particular, se cubrieron todas las especificaciones exigidas por la plataforma Link-All. Como Sistema de Ayuda, no sólo cubre todas las funcionalidades ofrecidas por los sistemas existentes, sino que también ofrece nuevas funcionalidades. Además, se considera que se han logrado los objetivos académicos, como son conocer nuevas tecnologías e incrementar los conocimientos adquiridos a lo largo de la carrera.

8 REFERENCIAS

Proyecto Link-All

[1] – Link-All
http://www.link-all.org/PublicSite/Summary.htm Junio, 2004.
http://www.fing.edu.uy/inco/proyectos/linkall/Documentos/Descripcion-LinkAll.doc Junio, 2004.

Sistemas de Ayuda

[2] – WinHelp
http://www.knopf.com/resources/help/winhelp.html Junio, 2004.

[3] – Microsoft HTML Help
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/htmlhelp/html/vsconHH1Start.asp Junio, 2004.
http://www.cherryleaf.com/stateofhelp3.htm Junio, 2004.

[4] – JavaHelp
http://java.sun.com/products/javahelp/ Setiembre, 2004.
O'Reilly - Creating Effective Javahelp

[5] – FlashHelp
http://www.macromedia.com/software/robohelp/productinfo/flash_help/ Julio, 2004.

[6] – OracleHelp, OHJ - OHW
http://www.oracle.com/technology/tech/java/help/index.html Setiembre, 2004.

Metadata

[7] – XML
http://www.ulpgc.es/otros/tutoriales/xml/ Julio, 2004.
http://www.programacion.net/html/xml/principal.htm Julio, 2004.

[8] – XML, DTD, RDF
http://www.w3schools.com Agosto, 2004.
http://es.wikipedia.org/wiki/Web_sem%C3%A1ntica Agosto, 2004

[9] – Ontologías
http://www.fing.edu.uy/~lsilva/SW.htm Octubre, 2004.

[10] – Repositorio de esquemas RDF y OWL
http://www.schemaweb.info/ Agosto, 2004.

[11] – Página oficial de la W3C sobre RDF
http://www.w3.org/RDF/ Agosto, 2004.

[12] – Página oficial de la W3C sobre OWL
http://www.w3.org/2004/OWL/ Agosto, 2004.

Tecnologías

[13] – Java
http://java.sun.com/ Julio, 2004.

[14] – Sitio oficial Eclipse
http://www.eclipse.org/ Agosto, 2004.

[15] – Sitio oficial JBoss
http://www.jboss.org/products/index Agosto, 2004.
http://www.osmosislatina.com/jboss/basico.htm Agosto, 2004.

[16] – Sitio oficial Jboss IDE
http://www.jboss.org/products/jbosside Setiembre, 2004.

[17] – MySQL
http://www.mysql-hispano.org Octubre, 2004.

[18] – J2EE, J2ME, J2SE, Java y JDBC
http://www.programacion.com/java/tutorial Agosto, 2004.

[19] – Flash
http://www.macromedia.com/software/flash/ Marzo, 2005.

[20] – Flash Remoting
http://www.macromedia.com/flashremoting Marzo, 2005.
http://www.themidnightcoders.com/flashorb/aboutFlashorb.htm Marzo, 2005.
http://sourceforge.net/projects/openamf/ Marzo, 2005.
http://www.fing.edu.uy/inco/proyectos/linkall/downloads_sp.php Febrero, 2005. "Interoperabilidad Flash-Java". Autor: Ing. Bruno Rienzi.

[21] – Portlets
http://developers.sun.com/prodtech/portalserver/reference/techart/jsr168/index.html Febrero, 2005.
http://www.atsistemas.com/portlets.html Febrero, 2005.

[22] – OpenLaszlo
http://www.laszlosystems.com Abril, 2005.
http://www.openlaszlo.org/ Mayo, 2005.

[23] – Rich Internet Applications (RIA)
http://www.laszlosystems.com Abril, 2005.
http://www.macromedia.com/resources/business/rich_internet_apps/ Mayo, 2005.
http://www.laszlosystems.com//download/products/literature/Laszlo_RIA.pdf Mayo, 2005.

API's

[24] – Link-All API Tutorial
http://www.fing.edu.uy/inco/proyectos/linkall/downloads_sp.php Marzo, 2005.

[25] – Lucene
http://jakarta.apache.org/lucene/docs/index.html Febrero, 2005.

[26] – JAXP
http://java.sun.com/xml/jaxp/index.jsp Agosto, 2004.

[27] – SAX
http://www.saxproject.org Agosto, 2004.

[28] – JFreeChart
http://www.jfree.org Abril, 2005.

Análisis y Diseño

[29] – UML y Patrones
Una introducción al análisis y diseño orientado a objetos y al proceso unificado. LARMAN, CRAIG. Segunda Edición. Ed: Prentice Hall, Inc. (2003) - ISBN: 84-205-3438-2.

[30] – Patrones de Diseños
http://msdn.microsoft.com/, Marzo 2005.

[31] – Diseño de Software con Patrones
http://www.javahispano.org/licencias/ Noviembre 2004.

[32] – Service Oriented Architecture (SOA)
http://java.sun.com/developer/technicalArticles/WebServices/soa2/ , Diciembre, 2004.
http://msdn.microsoft.com/architecture/soa/default.aspx, Diciembre, 2004.

Interfaz de Usuario

[33] – Estilos Web
http://www.webstyleguide.com/process/index.html Diciembre, 2004.
Web Style Guide (2nd. edition) - Basic Design Principles for Creating Web Sites de Lou Rosenfeld.
http://www.w3.org/2001/Talks/winwriters-20010305/all.htm Enero, 2005.

[34] –Accesibilidad
http://html.conclase.net/articulos/accesibilidad Enero, 2005.
http://www.w3.org/WAI/ Enero, 2005.

[35] – Recomendaciones de la W3C
http://www.w3.org/DOM/DOMTR Diciembre, 2004.
http://www.w3c.es/Consorcio/ Diciembre, 2004.

Cursos

[36] – Web y Bases de Datos
http://www.fing.edu.uy/inco/grupos/csi/esp/Cursos/cursos_posg/WebBD2002/ Julio, 2004.
Docentes: Regina Motz y Veronika Peralta.

[37] – Interacción persona computadora
http://www.fing.edu.uy/inco/cursos/inpercom/ Octubre, 2004.
Docentes: Eduardo Fernández y Tomás Laurenzo.

Proyectos de Grado

[38] – Mecanismo de Interoperabilidad entre Servidores de Aplicaciones.
http://www.fing.edu.uy/inco/grupos/csi/servicios/WebT5C/indexa.htm
Jorge Besil, Carla Pais, Darío Sande. Tutor: Dr. Ing. Raúl Ruggia. Instituto de Computación, Facultad de Ingeniería, UdelaR. Agosto, 2003.

[39] – Diseño de la metadata semántica para un orquestador de operaciones encapsuladas. Una Ontología para inferir cambios a nivel de Servicios.
http://www.fing.edu.uy/~pgmetsem/
Santiago Dalchiele, Ana Díaz, Daniel Gerchicov, Alvaro Rettich. Tutores: Regina Motz, Fernando Rodriguez.
Usuario Responsable: Jorge Abin. Instituto de Computación, Facultad de Ingeniería, UdelaR. Diciembre, 2004.

[40] – Guía Informe de Proyecto de Grado
http://www.fing.edu.uy/inco/cursos/proygrado/inicial.htm Marzo, 2005.

Control de Calidad y Testing

[41] – JUnit
http://jwebunit.sourceforge.net/ Diciembre, 2004.
http://www.junit.org/index.htm Diciembre, 2004.

[42] – Control de Calidad
http://www.fing.edu.uy/~dvallesp/Tesis/webActividades/index2.htm Junio, 2004.
Responsable: Ing. Diego Vallespir. Instituto de Computación, Facultad de Ingeniería, UdelaR. Marzo, 2005.

9 ANEXOS

[A] – Tecnologías de Ayuda
Se presenta un breve estudio relativo a las diferentes tecnologías de ayuda disponibles en el mercado.

[B] – Proceso de Desarrollo
Se describe detalladamente el proceso de desarrollo seguido a lo largo del proyecto.

[C] – Calidad y Testing
Se describen las pruebas realizadas y los resultados obtenidos, además se exponen las métricas de calidad de diseño.

[D] – Documento de Arquitectura
Se presenta la descripción de la arquitectura del sistema.

[E] – Documentación Técnica
Se plantea de forma muy detallada la solución presentada en este proyecto, además, se analizan algunas de las tecnologías utilizadas.

[F] – Documento de Casos de Uso
Se describen en detalle los casos de uso del Sistema.

[G] – Manual de Usuario
Se describe la forma de utilización del sistema desde la perspectiva del usuario.

[H] – Manual de Desarrollo e Instalación
Se describe el procedimiento necesario para realizar una implementación específica de MACSIF.

Mecanismo de Ayuda Contextual para Sistemas de Información Federados

Anexo
Tecnologías de Ayuda

INCO – Facultad de Ingeniería – UDELAR

Montevideo, Junio 2005

Tutores:

Ing. Federico Piedrabuena
Dr. Ing. Raúl Ruggia

Estudiantes:

Laura González
Guillermo Roldós
Flavia Serra

ÍNDICE

1 INTRODUCCIÓN

El propósito de este documento es mostrar las diferentes tecnologías disponibles en el mercado para almacenar y mostrar ayuda, realizándose un breve estudio relativo a cada una de ellas. Se busca resumir las ventajas y desventajas de cada producto, así como los aspectos comunes a todos ellos, mostrando cómo se aplicó este conocimiento en el desarrollo del producto presentado en este proyecto.

Se estudiaron las siguientes tecnologías de ayuda:

- Microsoft HTML Help
- Sun JavaHelp
- Macromedia FlashHelp
- Tecnología Help de Oracle

2 MICROSOFT HTML HELP

2.1 Descripción

Es una API [1] desarrollada por Microsoft para brindar ayuda a aplicaciones Windows, sucediendo a la tecnología anterior de Microsoft: *Winhelp*. El desarrollador de aplicaciones puede incluir ayuda "contextual" etiquetando determinados controles o ventanas, con el objetivo de pasarle este identificador a la ayuda en el momento que se solicita. *Html Help* utiliza funciones de bajo nivel del Internet Explorer, lo que le permite manejar contenidos de una gran variedad de tipos (todos los soportados por el navegador).

2.2 Características

Para almacenar los datos de ayuda, se utiliza un formato de archivo nativo (".chm"), el cual se puede adjuntar como un archivo separado de la aplicación a la cual brinda ayuda. Dado que este archivo está compilado y comprimido, se trata de la principal mejora con respecto a *Winhelp*, ya que ésta usaba el formato ".rtf" para expresar la ayuda, lo que hacía más pesada la distribución. Por otro lado, el hecho de que no exista información disponible acerca del formato de estos archivos compilados, dificulta el análisis de sus características.

Para la generación de los archivos ".chm", es posible utilizar la herramienta de autoría *Html Help Workshop* [5] (de libre distribución). Esta herramienta permite definir de una forma intuitiva los componentes de la ayuda: tabla de contenido, índice, etc., para un proyecto. Una vez finalizado el ingreso de la información, genera el archivo compilado (.chm) que se distribuye con la aplicación. Otra forma de generar contenido para ser utilizado por *Html Help*, es utilizar alguna de las varias herramientas disponibles en el mercado (*Robo Help* [9], *HelpBreeze* [7], *Doc-To-Help* [8], etc).

Para visualizar ayuda de este tipo, se debe utilizar la herramienta *Html Help Viewer* [6] (de libre distribución), la que utiliza el formato estándar para la visualización de ayudas (panel de navegación a la izquierda, panel de herramientas arriba, y panel de contenido a la derecha). En la figura 1 se muestra un ejemplo de su interfaz:

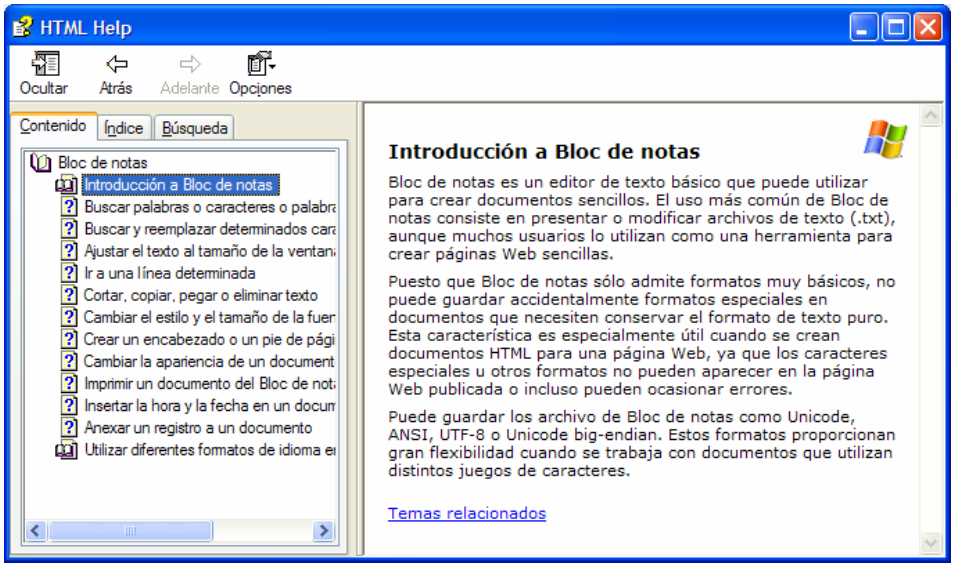


Figura 1

2.3 Ventajas

2.3.1 Flexibilidad en el tipo de archivos que accede
Los contenidos de ayuda pueden ser de muchos tipos, no solamente Html. Esto incluye todos los formatos reconocidos por Internet Explorer, dándole flexibilidad en el manejo de contenidos.

2.3.2 Estándar
Está impuesto como un formato estándar de ayuda de todas las nuevas aplicaciones Windows, aunque todavía quedan muchas aplicaciones que siguen utilizando el viejo formato "rtf" para *Winhelp*.

2.3.3 Herramienta de autoría gratuita
Se provee de forma gratuita una herramienta de autoría simple e intuitiva. De esta forma se simplifica la generación de ayuda para aplicaciones Windows.

2.4 Desventajas

2.4.1 Portabilidad
Solo funciona con aplicaciones Windows. Aunque es posible colocar un link a un archivo ".chm" en una página Html, esto solo funcionará con el Internet Explorer.

2.4.2 Extensibilidad y performance
Debido a que la información de ayuda debe encontrarse almacenada en un archivo, no es posible hacerlo en una Base de Datos relacional. Esto puede ser importante si el volumen de información es grande, ya que se pierde en velocidad de respuesta.

2.4.3 Datos estáticos
No es posible generar información de ayuda dinámica, ya que el archivo es totalmente estático. Si se desea agregar temas nuevos se debe regenerar todo el índice, para lo que se debe contar con el proyecto utilizado en primera instancia.

2.4.4 Uni-aplicación
No es posible integrar la información de ayuda de múltiples aplicaciones.

3 SUN JAVAHELP

3.1 Descripción
Es una especificación [2] provista por Sun, que brinda funcionalidades de Help a aplicaciones Java de distinto tipo (aplicaciones de escritorio, componentes Bean, applets, aplicaciones "server-based"). Se provee además una implementación de referencia basada en Swing.

3.2 Características
Visualmente, el *HelpViewer* se compone de tres "frames":

- un menú superior
- un árbol jerárquico de "tópicos" (temas de ayuda) a la izquierda, que incluye "tabs" para búsqueda, índice, y glosario
- el contenido de cada tópico a la derecha (llamado "metadata")

Internamente, este contenido se encuentra completamente en archivos de texto (Html, XML, etc.), los cuales se pueden distribuir por separado o en un único JAR. Para utilizar la ayuda, se debe crear una estructura formada de la siguiente manera:

- Una página Html que contendrá la estructura jerárquica de tópicos, indicando por cada uno: una URL que contendrá el archivo a mostrar cuando se pulse sobre ese tópico, o una lista de sub-tópicos.

- Un archivo HelpSet (.hs) que contiene toda la información para el *HelpViewer*.

- Un archivo de mapeo (.jhm) que contiene todas las referencias: identificador – página correspondiente. O sea, por cada identificador (es lo que se usa luego en los otros archivos) se mapea un link a una página Html.

- Un conjunto de archivos XML llamados "Metadata de Navegación", que indican:

- Tabla de contenido
- Índice
- Glosario

- Un conjunto de archivos creados por la propia herramienta (a través del comando *jhindexer*), que servirán para realizar búsquedas de texto. Estos archivos se ubicarán en un directorio llamado *JavaHelpSearch*.

Como en el caso de *Html Help* de Microsoft, para generar contenido para ser utilizado por *Java Help*, se puede utilizar alguna de las herramientas disponibles en el mercado. Una vez generado el contenido, será necesario distribuirlo junto con la aplicación, la cual accederá al contenido de la ayuda a través de uno de los archivos indicados antes (archivo HS, por HelpSet) utilizando las funcionalidades provistas en el paquete *javax.help*. En la figura 2 se muestra un ejemplo de la interfaz del *HelpViewer*:

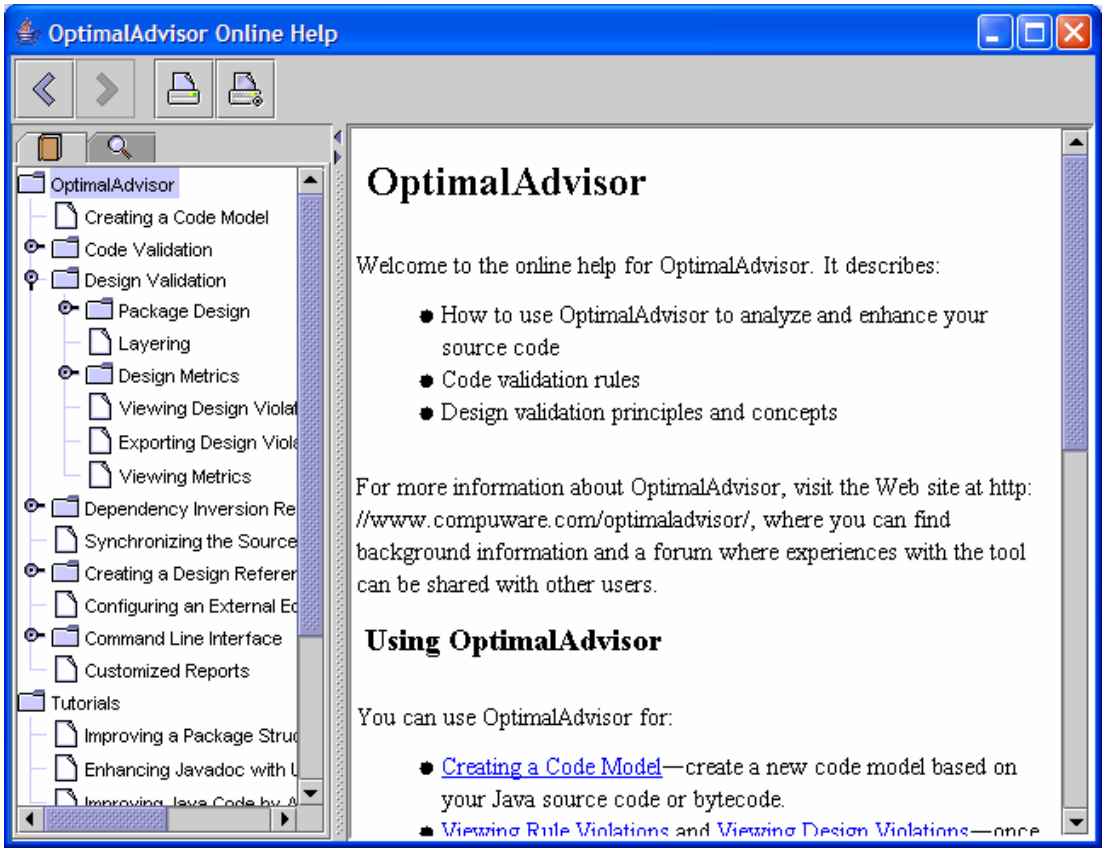


Figura 2

3.3 Ventajas

3.3.1 Portabilidad

Al ser 100% Java puro, se garantiza la portabilidad con todas las plataformas que soportan la JVM.

3.3.2 Estándar

Está impuesto como un formato estándar de ayuda, lo que hace que todas las herramientas de tipo “Help Authoring Tool” lo generen.

3.3.3 Búsqueda de texto

Se considera que el motor de búsqueda de texto dentro de la metadata prevista por la especificación está muy bien logrado. Contiene referencias a diccionarios semánticos para búsquedas más completas, etc.

3.4 Desventajas

3.4.1 Formato de la Metadata

Ya que todo el contenido de la ayuda está descrito en archivos de texto, no es posible obtener esta información de otros medios.

3.4.2 Interfaz gráfica

Se considera que la interfaz gráfica que presenta la implementación en Swing no está muy bien lograda (sobre todo en lo que tiene que ver con búsqueda temática e índice), ya que los íconos que muestra no son amigables, y no se incluye ningún texto aclaratorio.

3.4.3 Implementación incompleta del motor de búsqueda

Aunque la especificación del motor de búsqueda es bastante completa, la implementación que se presenta no incluye la mayor parte de las características propuestas.

3.4.4 Interacción con otros formatos de ayuda

Toda la información que desee mostrarse a través de *Java Help* debe estar guardada en el formato nativo, ya que no incluye facilidades para acceder contenido almacenado en otro formato.

4 MACROMEDIA FLASHHELP

4.1 Descripción

Es un formato [3] de ayuda desarrollado por Macromedia, para brindar ayuda a aplicaciones de escritorio y Web. Está basada en Flash, lo que le da un muy buen aspecto visual, que hace agradable la utilización de estas ayudas.

4.2 Características

Para la generación de contenido *Flash Help* es necesario utilizar la herramienta de autoría *Robo Help* [9] (que tiene un costo de U$S 999.- en su versión estándar). Esta herramienta permite definir de una forma intuitiva y amigable los componentes de la ayuda: tabla de contenido, índice, etc., para un proyecto. Una vez finalizado el ingreso de la información, genera una serie de archivos Flash, Javascript, Html, Xml, etc., los que compondrán la ayuda.

Es posible importar información almacenada en diferentes formatos: Html, Word, PDF, XML, etc. Una vez procesada con *Robo Help*, se puede generar también diferentes formatos, como: *Flash Help*, *WinHelp*, *Html Help*, *Oracle Help*, *Java Help*, Word, PDF, etc. Esto le da mucha versatilidad a la hora de convertir información de un formato a otro. En la figura 3 se muestra un ejemplo de la interfaz del *Flash Help*:

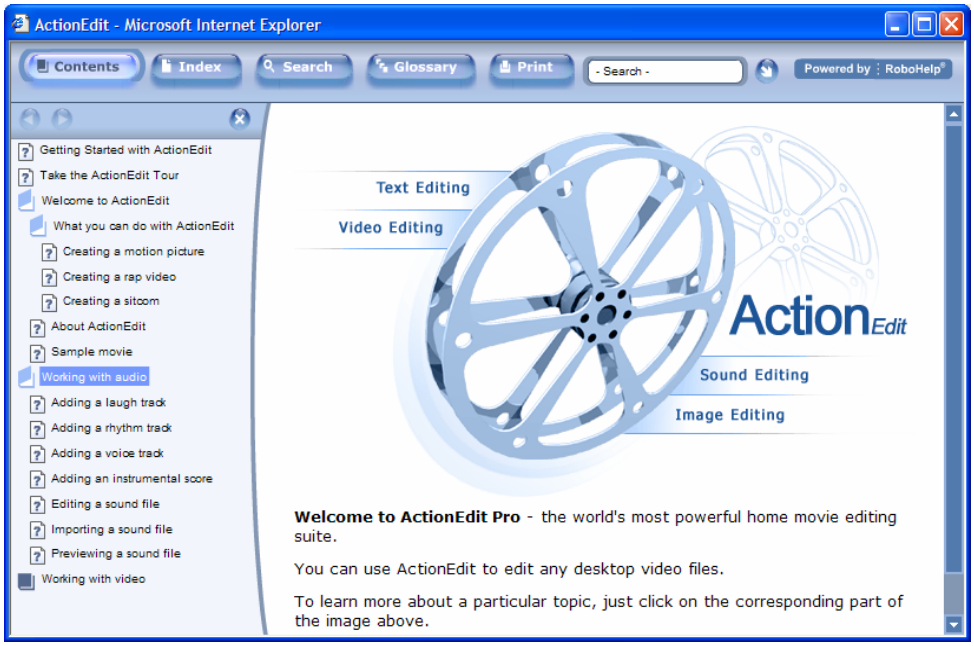


Figura 3

4.3 Ventajas

4.3.1 Independencia del navegador

El contenido de ayuda generado por *Robo Help* funciona correctamente en la gran mayoría de los navegadores, ya que además de generarse películas Flash, el código Javascript generado contempla esta posibilidad.

4.3.2 Look & Feel

El aspecto visual y capacidades multimedia de Flash hacen que las ayudas sean muy amigable. Además, es muy fácil cambiar todo el aspecto de la ayuda modificando tan solo un par de preferencias en el *Robo Help*.

4.3.3 Excelente herramienta de autoría

Si bien no es gratuita, se considera que la herramienta de generación de contenido es excelente. No es necesario conocer Flash para utilizarla, y posee gran variedad de "templates" para facilitar la tarea. Además, es capaz de generar otros formatos de ayuda que no sean *Flash Help*.

4.4 Desventajas

4.4.1 Costo de la herramienta de autoría

El hecho de que la herramienta de generación de contenido no sea gratuita imposibilita que esta tecnología se extienda.

4.4.2 Interacción con otros formatos de ayuda

Si bien *Robo Help* permite importar información almacenada en diferentes formatos (Word, PDF, etc.), toda la información que desee mostrarse a través de *Flash Help* debe estar guardada en el formato nativo.

5 TECNOLOGÍA HELP DE ORACLE

5.1 Descripción

La tecnología Help de *Oracle* [4], brinda los medios para desarrollar y presentar sistemas de ayuda basados en Html. Los autores crean un único sistema de ayuda que puede luego ser presentado tanto en un ambiente java, utilizando *Oracle Help for Java* (OHJ) así como en un ambiente Web, utilizando *Oracle Help for the Web* (OHW).

5.2 Características

OHJ es un conjunto de componentes Java, un API Java y una especificación de formato para desarrollar y presentar información de ayuda basada en Html en un ambiente Java. El *Oracle Help for Java Developer's Kit* (OHJDK) incluye la tecnología OHJ junto con herramientas y documentación para desarrollar ayuda sensible al contexto para aplicaciones y applets Java.

La interfaz de usuario de OHJ tiene dos partes principales:

- una ventana de navegación
- una ventana que despliega la información de ayuda.

Los usuarios pueden juntar las ventanas, para que aparezcan como una sola, o pueden mantener las ventanas separadas.

La ventana de navegación tiene un conjunto de "tabs" para navegar y encontrar información en el sistema de ayuda. Por defecto, se incluyen "tabs" para contenido, índice y búsqueda. Se pueden cambiar varias características de la ventana de navegación configurando algunos parámetros, por ejemplo, se pueden cambiar los textos en los "tabs", se pueden desplegar varias tablas de contenido e incluso se pueden crear "tabs" personalizados.

La ventana que despliega la información de ayuda, utiliza un componente incluido en el OHJDK para presentar el contenido Html. Sin embargo, este componente puede ser remplazado por cualquier otro componente para visualizar archivos Html. En la figura 4 se muestra un ejemplo de las ventanas antes mencionadas:

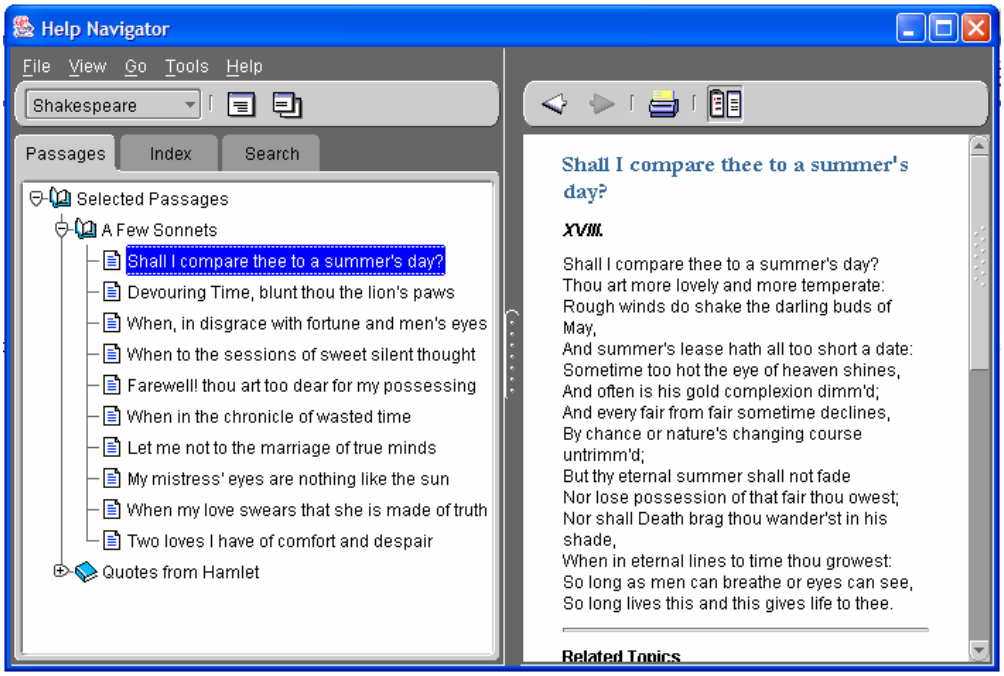


Figura 4

Por otro lado, OHW es un servlet Java y una especificación de formatos de archivo para desarrollar y distribuir contenido de ayuda basado en Html en un ambiente web. Con OHW todo el procesamiento toma lugar en el servidor, a través del servlet OHW, por lo que muchos usuarios tienen acceso a una misma instalación del sistema de ayuda.

La interfaz de usuario de OHW provee las mismas características que OHJ. Sin embargo, como OHW es una aplicación web, hay algunas diferencias en apariencia y comportamiento. En la figura 5 se muestra una imagen de dicha interfaz de usuario:

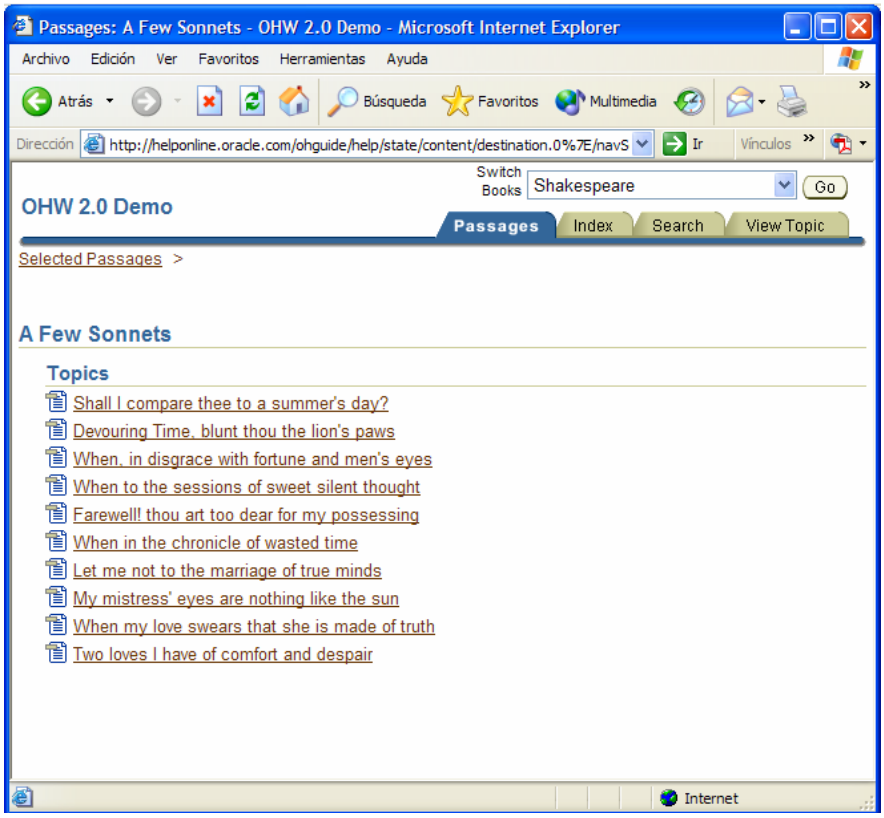


Figura 5

Junto con los archivos Html para crear la información de ayuda, OHJ utiliza varios archivos de control para crear y manejar el sistema de ayuda, por ejemplo, archivos para definir el índice y la tabla de contenido. OHJ reconoce tanto el formato de archivos XML definido por Sun para Java Help, así como el formato definido por Microsoft para *Microsoft Html Help*.

Para generar los archivos requeridos por OHJ o OHW, se pueden utilizar las siguientes herramientas: *Robo Help* [9] o *AuthorIT* [10].

5.3 Ventajas

5.3.1 Portabilidad
Al ser 100% Java puro, se garantiza la portabilidad con todas las plataformas que soportan la JVM.

5.3.2 Soporte de principales formatos
Maneja los dos formatos más utilizados para la especificación de información de ayuda, por lo que se puede utilizar la mayoría de las herramientas de tipo "Help Authoring Tool".

5.3.3 Solución para aplicaciones web
OHW da una solución para brindar ayuda a aplicaciones Web. Además, la interfaz de usuario de OHW utiliza Html básico por lo que puede visualizarse correctamente en la mayoría de los exploradores web.

5.3.4 Accesibilidad e Internacionalización
Todas las características de la interfaz de usuario son accesibles utilizando el teclado, incluyendo el componente para la visualización de Html. Además, todos los textos utilizados en la interfaz de usuario son almacenados en archivos de recursos, por lo que son fácilmente traducibles.

5.3.5 Arquitectura abierta
OHJ tienen una arquitectura que permite de una forma sencilla sustituir los componentes por defecto, como puede ser la utilidad de búsqueda, por componentes personalizados.

5.4 Desventajas

5.4.1 Look & Feel
La interfaz de usuario utilizada por OHW utiliza sólo Html estático por lo que no es tan amigable como por ejemplo la ayuda generada por *Flash Help*.

5.4.2 Herramienta de autoría
No existe una herramienta gratuita para generar el contenido y archivos de control requeridos.

6 CONCLUSIONES
En base a un análisis de las tecnologías de ayuda existentes, es posible extraer una serie de características o conceptos comunes a todas ellas, como ser:

- Los conceptos manejados son siempre los mismos: contenidos de ayuda independientes, tabla de contenido en forma de árbol jerárquico para una fácil navegación a través de los contenidos, índice con palabras clave y contenidos vinculados a ellas, glosario para términos especiales, y búsqueda de texto en los contenidos.

- Generalmente se muestra la información en 2 o 3 frames: a la izquierda los componentes de navegación y a la derecha los contenidos. Puede haber un frame superior con otras operaciones como "Imprimir", etc.

- En la mayoría de los casos se incluye la posibilidad de mantener "Favoritos" por usuario, de forma de facilitar el acceso a los contenidos de su preferencia.

Asimismo, es posible detectar una serie de carencias comunes a todas estas tecnologías de ayuda, como ser:

- Si bien algunas herramientas permitan importar y exportar, desde y hacia diferentes formatos, ninguna de las ellas permite, en tiempo de ejecución, utilizar diferentes formatos. Cada tecnología utiliza su propio formato de representación de datos, lo que hace que no sea posible mantener información en diferentes medios (como ser Bases de Datos, etc.).

- En ningún caso existe la posibilidad de restringir la visualización de contenidos de ayuda de acuerdo a los permisos de acceso del usuario que consulta.

- No existen tecnologías de ayuda orientadas a Sistemas de Información Federados, por lo que no existe la forma de:
    - Integrar la información de ayuda de las múltiples aplicaciones que forman un sistema de este tipo, ya que se orientan en todos los casos a ayudas para una aplicación.
    - Interactuar con servidores remotos para obtener ayuda correspondiente a aplicaciones que se estén ejecutando en ellos. En todos los casos, los orígenes de datos están locales al sitio donde se ejecuta la ayuda (o en el mejor de los casos un link estático a un servidor).

7 REFERENCIAS

Sistemas de Ayuda

[1] – Microsoft HTML Help
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/htmlhelp/html/vsconHH1Start.asp Junio, 2004.
http://www.cherryleaf.com/stateofhelp3.htm Junio, 2004.

[2] – JavaHelp
http://java.sun.com/products/javahelp/ Setiembre, 2004.
O'Reilly - Creating Effective Javahelp

[3] – FlashHelp
http://www.macromedia.com/software/robohelp/productinfo/flash_help/ Julio, 2004.
http://www.cherryleaf.com/news-wildfire.htm Julio, 2004.

[4] – OracleHelp, OHJ - OHW
http://www.oracle.com/technology/tech/java/help/index.html Setiembre, 2004.

Herramientas

[5] – Microsoft HTML Help Workshop
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/htmlhelp/html/vsconwhtshw.asp Junio, 2004.

[6] – Microsoft HTML Help Viewer
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/htmlhelp/html/vscontripane.asp Junio, 2004.

[7] – HelpBreeze
http://www.solutionsoft.com/hlpbrz.htm Julio, 2004.

[8] – Doc-To-Help
http://www.wextech.com/doc-to-help.html Julio, 2004.

[9] – Macromedia RoboHelp
http://www.macromedia.com/software/robohelp/ Julio, 2004.

[10] – AuthorIT
http://www.author-it.com/ Setiembre, 2004.

Mecanismo de Ayuda Contextual para Sistemas de Información Federados

Anexo
Proceso de Desarrollo

INCO – Facultad de Ingeniería – UDELAR

Montevideo, Junio 2005

Tutores:

Ing. Federico Piedrabuena
Dr. Ing. Raúl Ruggia

Estudiantes:

Laura González
Guillermo Roldós
Flavia Serra

ÍNDICE

1 INTRODUCCIÓN

1.1 Objetivo

El objetivo de este documento es mostrar en detalle el proceso seguido en el Proyecto "Mecanismo de Ayuda Contextual para Sistemas de Información Federados (Basados en Servicios)". En éste se presenta la planificación inicial, sus ajustes, y los pasos seguidos durante el mismo. Como material de apoyo, se incluyen referencias a los documentos presentados y publicados durante el transcurso de todo el proyecto.

1.2 Modelo de Proceso

El proceso se basó en el modelo iterativo incremental que se aplica en la materia Ingeniería de Software, que está basado en el Proceso Unificado de Rational [1]. En este modelo se definen 3 fases para el proceso de desarrollo, en las que el producto va incorporando funcionalidades y mejoras, obteniendo al final de cada una de estas un producto cada vez más completo. Es importante, por tanto, la definición clara (durante la fase inicial) de los objetivos que se buscan en cada fase.

1.3 Organización del Documento

Este documento contiene un capítulo para cada fase de desarrollo, en los que se presentan los objetivos de cada una de ellas, el producto obtenido en las mismas, y el testing correspondiente al producto (si corresponde). Finalmente, se incluyen algunas conclusiones sobre el modelo aplicado.

Las versiones anteriores de todos los documentos están disponibles en la página Web del proyecto [14], por lo que no se incluye una referencia a ellos.

2 FASE INICIAL

2.1 Objetivos

Los objetivos de esta fase fueron: relevar y priorizar la mayor cantidad de requerimientos posible a través de entrevistas con el cliente; realizar una planificación detallada, y producir un prototipo inicial que sea de utilidad para las siguientes fases del desarrollo. A su vez, debido a que la interfaz de usuario es un elemento importante del proyecto, está dentro de los objetivos investigar sobre usabilidad y accesibilidad.

2.2 Documentos generados

2.2.1 Requerimientos

El día 18/6/2004 se realiza con los clientes la primera reunión de requerimientos, en [6] se presenta el documento con las decisiones tomadas. A partir de esta reunión, y de varias consultas informales realizadas de forma personal o vía mail, se obtuvo la versión 1.0 del Documento de Requerimientos, el cual fue refinado en sucesivas versiones 1.1 y 1.2.

2.2.2 Casos de Uso

En los últimos días de julio se generó la versión 1.0.0 del Documento de Casos de Uso, y a partir de algunos comentarios del tutor con respecto al formato del mismo se modificó, en los primeros días de agosto, a la versión 1.0.1.

2.2.3 Alcance

Se decidió que no sería necesario dividir la Fase de Elaboración en iteraciones para que no se extendiera excesivamente en el tiempo, así como también que la Fase de Construcción se dividiría en tres iteraciones, cada una de las cuales generaría una versión del Sistema que se debería testear. El documento Alcance del Sistema en su versión 1.0.1 establece los objetivos que se persiguen en cada una de las iteraciones de la Fase de Construcción, los que se resumen a continuación:

- **Primera iteración**: se busca implementar el caso básico, en el que esté disponible toda la información de ayuda para la aplicación que se está usando, y mostrando información básica. El objetivo de esta iteración es invertir más tiempo en el armado del núcleo central del Sistema, dejando para las sucesivas iteraciones el agregado de más funcionalidades.

- **Segunda iteración**: se busca agregar las funcionalidades que son mínimamente necesarias para Link-All, como son el contexto del usuario y los temas relacionados. Con esto, se obtendría una versión del Sistema que debía poseer nivel de calidad 5 (correspondiente a un producto que sería puesto en producción), teniendo en cuenta que se dispondría de más tiempo para realizar el testing.

- **Tercera iteración**: se agregan funcionalidades extras que no son necesarias para el sistema Link-All, y que probablemente no sean puestas en producción. Por esto, en esta iteración se exige un nivel de calidad 3 (correspondiente a un prototipo), ya que se busca aportar más al proyecto de grado en sí, abstrayéndonos de los objetivos definidos para Link-All.

2.2.4 Casos de Prueba de Casos de Uso

A efectos de facilitar la tarea de testing, se produjo un documento con los casos de prueba que se debía generar para probar los casos de uso, buscando cubrir todos los escenarios posibles.

2.3 Prototipo inicial

El día 7/9/2004 se presentó el prototipo inicial del Sistema, el cual incluye solamente interfaz de usuario (sin lógica) a efectos de fijar objetivos y discutir ideas con el cliente. Esta interfaz (figura 1), fue aprobada y se dió por finalizada la Fase Inicial, ya que no era necesario realizar ningún testing.

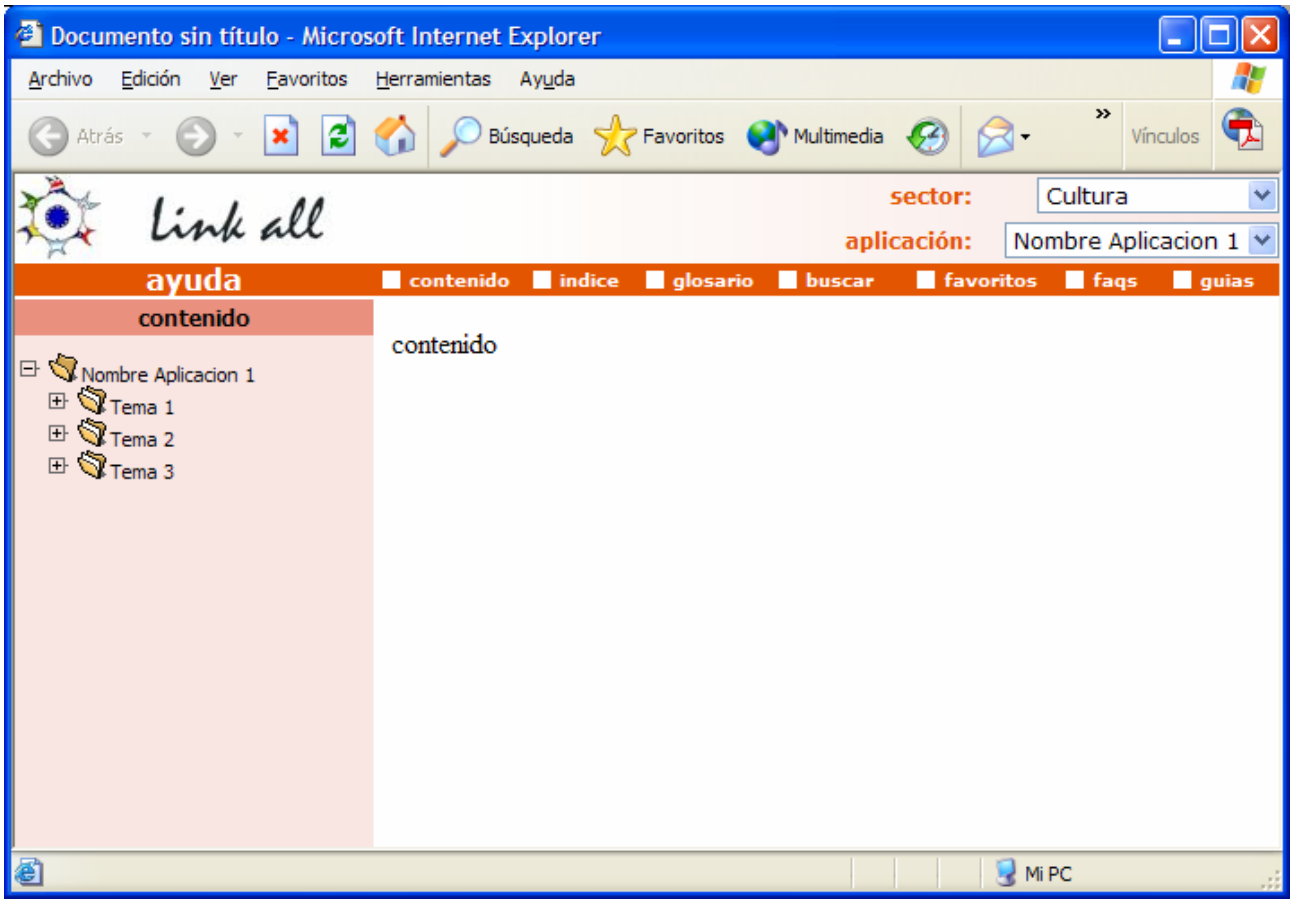


Figura 1

2.4 Conclusiones de la fase

Se cumplieron los objetivos de relevar y priorizar los requerimientos, así como realizar una planificación detallada de las siguientes etapas del desarrollo. También se definieron los principales casos de uso y se obtuvo una visión general del problema.

3 FASE DE ELABORACIÓN

3.1 Objetivos

Los objetivos de esta fase fueron: realizar una evaluación detallada de las herramientas de ayuda disponibles en el mercado, así como obtener una primera versión de la arquitectura y de la interfaz de usuario. Además, se buscó obtener versiones definitivas de los documentos generados en la fase anterior. Para realizar un buen diseño de la solución, se siguió la metodología propuesta por Craig Larman para el análisis y diseño orientado a objetos [2].

3.2 Documentos generados

3.2.1 Documentos de la Fase Inicial

A partir del conocimiento más profundo acerca de las herramientas de ayuda existentes, y de las funcionalidades que las mismas ofrecen, se refinaron todos los documentos generados en la fase anterior, lográndose los siguientes documentos en sus versiones finales:

- Documento de Requrimientos versión 2.0.0 [7]
- Documento de Casos de Uso versión 2.0.0 [5]
- Alcance del Sistema versión 2.0.0 [8]

3.2.2 Glosario

A efectos de facilitar la lectura de personas sin conocimiento de algunos términos incluidos en la documentación y para fijar conceptos entre los propios integrantes del grupo, se generó un Glosario [9].

3.2.3 Arquitectura

Se realizó la versión 1.0.0 del Documento de Descripción de la Arquitectura, y posteriormente se refinó a la versión 1.0.1.

3.2.4 Plan de Testing y modelo de Reporte de Errores

A partir de los documentos de testing presentados en la materia Ingeniería de Software, se realizó el documento de Plan de Testing 1.0.0, en el cual se determinaron los métodos de testing que serían aplicados, así como los criterios de cubrimiento. Además, se propone un modelo para el Reporte de Errores en la versión 1.0.0, y luego se mejora en la versión 1.0.1.

3.3 Prototipo

Dado que a mediados de noviembre de 2004 se realizaría un encuentro técnico del proyecto Link-All (en Florianópolis), se propone definir una interfaz de usuario para su presentación en dicho encuentro. Por este motivo, se posterga la incorporación de lógica al prototipo y se prioriza la presentación. A continuación, en la figura 2, se muestra la interfaz, que fue aceptada dando por finalizada la Fase de Elaboración.

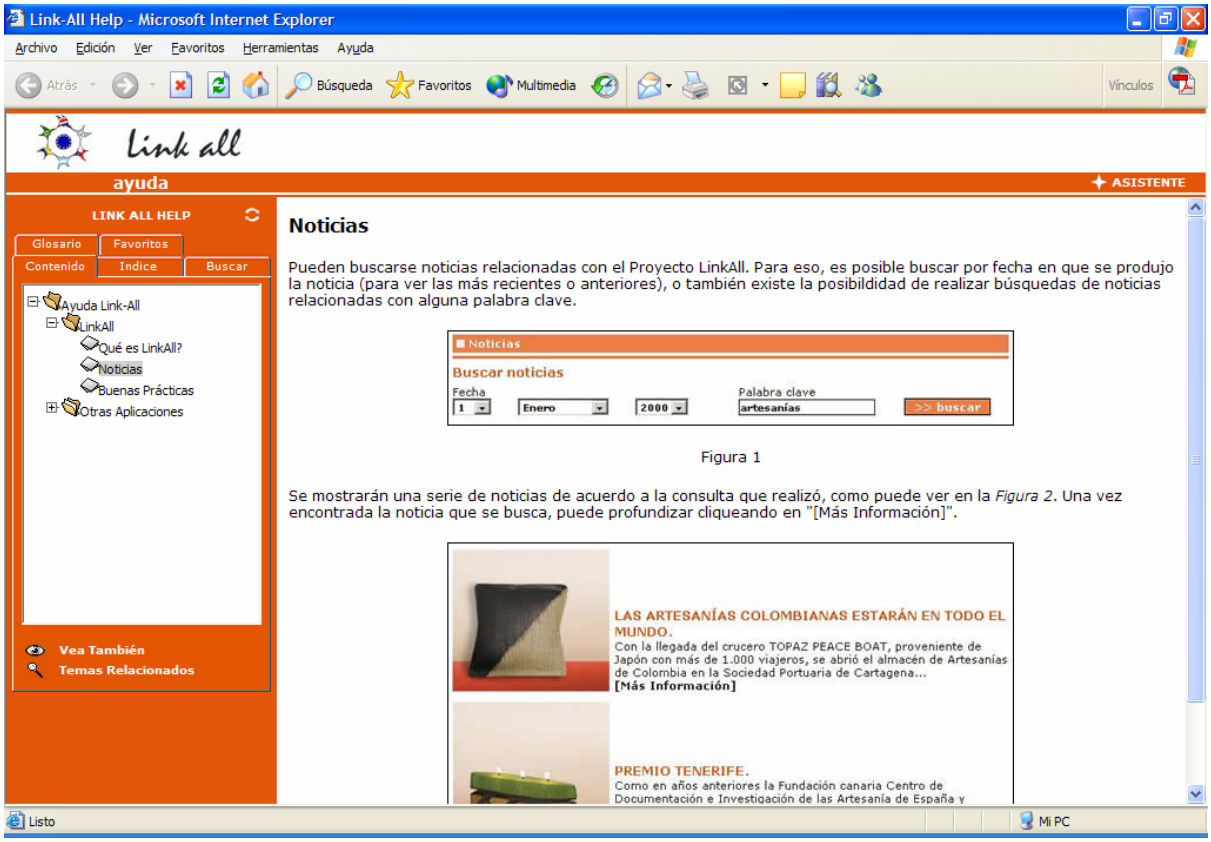


Figura 2

3.4 Conclusiones de la fase

Se cumplieron los objetivos de obtener una visión clara del Sistema a construir, la cual queda establecida en los documentos correspondientes. Esta visión surge de la investigación realizada sobre otras herramientas de ayuda. Si bien se buscaba que el prototipo resultante de esta fase incorporara algo de lógica, se considera importante el hecho de que se haya aprobado una interfaz que fue presentada con gran nivel de detalle.

4 FASE DE CONSTRUCCIÓN

4.1 Primera Iteración

La primera iteración se centró en definir e implementar el núcleo de la aplicación de ayuda, con el propósito de ir agregando funcionalidades a partir de él en las sucesivas iteraciones. Con este objetivo, se mejoró la definición de la arquitectura que se había planteado en la fase anterior (ver Documento de Arquitectura versión 2.0.0), y se creó un documento Modelo de Diseño (versión 1.0.0), el cual es una descripción detallada a nivel de componentes, paquetes, clases, contratos y operaciones. Se buscó generar la primera versión del Sistema, con el objetivo de no implementar nada hasta que todo estuviera lo suficientemente claro a nivel de diseño. Este documento fue de gran utilidad para la implementación inicial del producto (que se realizó en base a él), aunque no se mantuvo en las sucesivas etapas del proyecto debido a su complejidad.

4.1.1 Objetivos

El diseño de la arquitectura provee una serie de componentes interactuando entre ellos en capas, y accediendo a los datos en los que se guarda la información de ayuda para cada aplicación. En esta iteración, el objetivo es implementar todos los componentes con las operaciones básicas, trabajando en el contexto ideal, en el cual los datos están cargados correcta y completamente.

4.1.2 Desarrollo

La implementación se realizó en los tiempos esperados y se cerró la primera iteración el día 3/12/2004 con la versión 1.0 del producto, aunque luego se siguió en paralelo con las pruebas unitarias de esta versión y con el desarrollo de la versión siguiente (correspondiente a la segunda iteración). Esto se hizo así para ganar tiempo, debido a que se iba a utilizar JUnit para realizar el testing, fue necesario realizar una investigación acerca de esta herramienta, ya que ninguno de los integrantes del grupo tenía experiencia con la misma. El día 20/12 se finaliza el testing con los resultados que se muestran en el punto 4.1.3. Se decidió que las fallas encontradas serían corregidas en la siguiente iteración (la que ya estaba avanzada). Un punto importante es que se probaron individualmente todos los métodos de todas las clases del Sistema, que es más que lo que exige el nivel de calidad 5 (éste sólo exisge pruebas a nivel de componentes). Con esto, se pretende detectar una mayor cantidad de errores en etapas tempranas del proyecto.

Una vez finalizado el testing de la versión 1.0, se utilizaron herramientas para medir la calidad del diseño (OptimalAdvisor [3]) y el cubrimiento de las pruebas realizadas (Hansel [4]). Los resultados de estos análisis se observan en el punto 4.1.3. En particular, la medida del cubrimiento fue muy útil, ya que no se invirtió mucho tiempo en obtenerla y determinó que se había testeado cubriendo una buena cantidad de escenarios (ya que no se había utilizado ninguna herramienta para determinar los casos de prueba a utilizar).

4.1.3 Resultados

- **Pruebas unitarias**: Se probó un total de 31 clases con sus respectivos métodos. El total de pruebas realizadas fue de 359, encontrándose fallas en 32 de ellas (8,91%). Estas fallas se categorizar según la importancia en:
  - alta (3 – 0,84%)
  - media (14 – 3,90%)
  - baja (15 – 4,18%)

  Se documentó en forma detallada cada una de estas fallas, lo que puede encontrarse en el Documento de Reporte de Errores versión 1.1.0.

- **Pruebas de Integración**: Están incluidas en los totales anteriores, ya que algunas de las clases testeadas corresponden a la “fachada” que presentan los componentes.

- **Pruebas de Sistema**: Se probó informalmente, ya que se implementó solamente un caso de uso (el más complejo e importante) y observando el comportamiento ante diferentes acciones.

- **Medida del Cubrimiento**: Para probar el cubrimiento se utilizó Hansel, el cual genera una lista de casos de prueba para cada bloque de código que encuentra en las clases testeadas. Si durante las pruebas no se recorre dicho bloque de código, se reporta una falla. Tomando en cada clase el total de casos de prueba generados por Hansel (o sea, el total de bloques de código) y el total de fallas arrojadas sobre ellos, se logra un cubrimiento del 76%. Si no se toman en cuenta los métodos de actualización de datos en las clases DAO (patrón de Acceso a Datos utilizado) este porcentaje de cubrimiento asciende a 81%. Esto corresponde en su gran mayoría a excepciones ("catch") que no son atrapadas en ninguna de las pruebas.

- **Métricas de Calidad del Diseño**: Se utilizó OptimalAdvisor para analizar la calidad del diseño. Si bien los resultados arrojados fueron buenos para algunas variables (2 de 5 métricas muestran un valor óptimo de 1), no lo son tanto para otras, las que se deben mejorar. Con este fin, se decide incorporar algunos patrones de diseño (además de los que ya se utilizaban) para la siguiente iteración, como ser Singleton y Factory.

4.2 Segunda Iteración

La segunda iteración se inicia en los primeros días de diciembre de 2004, teniendo la particularidad de que se fueron mostrando avances a través de pequeñas iteraciones en las que se cerraban versiones del producto, con el objetivo de validarlas con los clientes. El motivo de esto fue que la iteración sería larga ya que se pretendía incorporar una gran cantidad de funcionalidades.

4.2.1 Objetivos

Como se definió en el Alcance del Sistema (ver punto 2.2.3), en la segunda iteración se buscó implementar el resto de las funcionalidades que son importantes y necesarias para Link-All, como ser todas las que hacen referencia al contexto (incluyendo internacionalización a través de multi-idioma en todos los mensajes generados por pantalla), manejo de Índice, Glosario, Vea También, y búsqueda de texto en las ayudas.

4.2.2 Desarrollo

Como se mencionó, esta iteración se dividió en mini-iteraciones, las que se describen a continuación.

- La primera de estas pequeñas iteraciones se cerró con la versión 1.1 el día 20/12/2004 y se presentó junto con el resultado del testing de la versión 1.0 correspondiente a la primera iteración. Las funcionalidades que se incorporaron fueron:

    - Parametrización del código a través de archivos de configuración.
    - Implementación de la funcionalidad de Índice.
    - Implementación de la funcionalidad de Glosario.

- La segunda pequeña iteración se cierra el día 20/02/2005 con la versión 2.0, la que incorporó las siguientes funcionalidades:

    - Implementación de la funcionalidad "Vea También".
    - Internacionalización a través de multi-idioma en todos los mensajes que el Sistema muestra por pantalla.
    - Se tomó en cuenta el idioma del usuario para mostrar la ayuda.
    - Mejoras en performance y escalabilidad, ya que se adoptan algunos patrones como Factory y Singleton, y los componentes que encapsulan el acceso a datos se convierten a EJB.
    - Se implementó una ayuda por defecto en caso de no existir la ayuda que el usuario requiere.
    - Se permitió que las carpetas de la tabla de contenido tuvieran un contenido asociado.

- La tercera y última pequeña iteración se cierra el día 07/03/2005 con la versión 2.1, que incorporó las siguientes funcionalidades:

    - Búsqueda de texto en los contenidos de las ayudas, utilizando la API Lucene. Se permite buscar en la ayuda de una aplicación o de todas las aplicaciones.
    - Se actualizó la versión de la API de Link-All, de la versión 1.6 a la versión 1.8.
    - Se agregó información sobre la aplicación y sus funcionalidad en las páginas por defecto.

  Una vez finalizada esta versión, se realizó un Manual de Instalación que incluye instrucciones sobre la instalación del producto y un detalle de los cambios realizados en cada versión. Este manual está disponible en [10].

- Sobre la versión 2.1 se realizaron las pruebas unitarias (se testearon los componentes, tal como exige el nivel de calidad 5 para esta versión), cuyos resultados pueden observarse en el punto 4.2.3. Luego de corregidas las fallas encontradas se generó la versión 2.2, la cual se testeó nuevamente a nivel de pruebas unitarias (no arrojó fallas) y a nivel de pruebas de casos de uso, arrojó dos fallas menores, por lo que se decidió cerrar la Segunda Iteración con la versión 2.2 el día 14/03/2005. Esta versión es la que incluye todas las funcionalidades básicas para la plataforma Link-All (y algunas otras), y satisface el nivel de calidad correspondiente.

4.2.3 Resultados

- **Pruebas unitarias**: Se realizaron sobre todas las operaciones brindadas por los componentes, cubriendo la mayor cantidad de entradas posibles (incluyendo casos de borde, etc.). Se hicieron un total de 187 casos de prueba, encontrándose fallas en 10 de ellas (5,35%). Como se destacó antes, estas fallas se categorizar según la importancia en:
  - alta (0 %)
  - media (1 – 0,53%)
  - baja (9 – 4,81%)

  Se documentó en forma detallada cada una de estas fallas, esta información puede ser consultada en el Documento de Reporte de Errores versión 1.2.0.

- **Pruebas de Sistema**: Se realizaron los casos de uso siguiendo los casos de prueba planteados en el documento correspondiente (Documento de Casos de Prueba de Casos de Uso), observándose 2 fallas de importancia baja o muy baja, que fueron reportadas en el Documento de Reporte de Errores versión 1.2.1.

- **Medida del Cubrimiento**: No se pudo medir el cubrimiento ya que la herramienta utilizada (Hansel) generó problemas para los componentes EJB's y no funcionó correctamente.

- **Métricas de Calidad del Diseño**: A través de la incorporación de patrones de diseño, se mejoraron algunas métricas de calidad, como se puede ver en el Documento de Métricas de Calidad versión 1.0.1. En particular, la métrica EP (que mide el nivel de encapsulamiento) pasó a ser óptima (valor máximo). Sin embargo, aún permanecen dos métricas con valores no óptimos, principalmente debido a un paquete utilitario que se definió con clases que debían ser vistas y utilizadas por todos los componentes. Se concluyó que si bien algunas métricas se mantenían bajas, se conocía el motivo y se consideraba que no incidían en la calidad del producto final.

4.3 Tercera Iteración

Para la tercera iteración se tomó en cuenta que ya se había finalizado una versión del producto que satisfacía las necesidades básicas de Link-All. Como se definió en el Alcance del Sistema en el punto 2.2.3, esta tercera iteración estaría dedicada a aportar funcionalidades o características avanzadas, que si bien no eran necesarias para Link-All, serían un buen aporte al Proyecto de Grado en sí. Por este motivo, no se exigió un nivel de calidad mayor a 3 para las nuevas funcionalidades.

4.3.1 Objetivos

Al iniciar la iteración se evaluó qué características debía tener el producto para satisfacer los objetivos del Proyecto de Grado (ya no del proyecto Link-All), llegándose a la conclusión de que el inconveniente más notorio era que debido al nivel de compromiso con Link-All se desarrolló un producto que no era fácilmente extensible más allá de dicha plataforma. Por tanto, se fijó el objetivo de reordenar los componentes desarrollados para que el producto final fuera más extensible, contemplando la posibilidad de poder utilizarse en cualquier contexto de Sistemas de Información Federados a través de la Web.

4.3.2 Desarrollo

Como la iteración anterior, ésta también fue dividida en pequeñas iteraciones, las que se describen a continuación.

- En la primera de estas iteraciones se generó un nuevo diseño, compuesto no solo por todos los componentes de la versión anterior, sino también por nuevos componentes, de forma de lograr mayor extensibilidad. Se reordenó el código y el día 02/04/2005 se cerró la versión 3.0, la que contiene todas las funcionalidades que se incluían en la versión anterior. Con esto, se obtiene una versión (y una nueva arquitectura), que cumple las especificaciones de Link-All, con una arquitectura más extensible.

- La segunda iteración se centró en la incorporación de funcionalidades que no estaban incluidas en versiones anteriores, ejemplos de ellas son:
    - "Favoritos"
    - control de acceso a los contenidos de ayuda de acuerdo a los permisos definidos en las aplicaciones.

  También se extendió aún más la arquitectura al incorporar componentes para administración del Sistema de Ayuda Contextual, así como se transformaron algunos componentes a EJB para mayor escalabilidad. En particular, se dio la posibilidad de incorporar diferentes motores para las búsquedas de texto, incorporando el concepto de "Proveedores de búsqueda de texto", similares a los "Proveedores de ayuda" ya incluidos en la iteración anterior. Se cerró la versión 3.1 el día 23/04/2005, sobre la cual se realizaron las pruebas unitarias (se testearon los componentes, ya que se exige un nivel de calidad 5 a esta versión con el objetivo de sustituir a la versión 2.2 mencionada en el capítulo anterior). Los resultados de estas pruebas pueden observarse en el punto 4.3.3.

- Para la tercera iteración, se corrigieron las fallas encontradas en la segunda iteración, además de incorporar la funcionalidad de "Asistente" (implementado como un proveedor de ayuda más), y una serie de mejoras en la interfaz de usuario con el objetivo de respetar los conceptos de usabilidad. Finalmente se cerró la versión 3.2 el día 22/05/2005, en la cual se probaron todos los componentes ya que éstos intervenían en casos de uso con nivel de calidad 5. No se detectó ninguna falla, por lo que se realizó una prueba de sistema. Esta es la versión final del producto, e incluye todas las funcionalidades básicas para la plataforma Link-All satisfaciendo el nivel de calidad 5. También incluye funcionalidades extras en un nivel de calidad 3.

4.3.3 Resultados

- **Pruebas unitarias**: Se realizaron sobre todas las operaciones brindadas por los componentes, cubriendo la mayor cantidad de entradas posibles (incluyendo casos de borde, etc.). Se realizaron un total de 259 casos de prueba, encontrándose para la versión 3.1, fallas en 13 de ellas (5,02%). Como se destacó antes, estas fallas se categorizan según su importancia en:
    - alta (0 %)
    - media (4 – 1,54%)
    - baja (9 – 3,47%)

  Se documentó en forma detallada cada una de estas fallas, esta información puede ser consultada en el Documento de Reporte de Errores versión 1.3.0 y en la Planilla de Errores versión 1.2.0 [11].

- **Pruebas de Sistema**: Se realizaron los casos de uso siguiendo los casos de prueba planteados en los documentos correspondientes (Documento de Casos de Prueba de Casos de Uso y Casos de Prueba de Aceptación [13]), y el resultado está disponible en el Documento de Reporte de Errores versión 1.3.1 [12].

- **Medida del Cubrimiento**: La herramienta utilizada (Hansel) genera problemas cuando los componentes testeados son EJB's, y dado que la mayoría de los componentes testeados fueron EJB's, no se pudo medir el cubrimiento.

- **Métricas de Calidad del Diseño**: Como se mencionó, se realizó un reordenamiento de los componentes con el propósito de mejorar el diseño. El resultado puede verse en el Documento de Métricas de Calidad versión 1.0.2, aunque el nivel de encapsulamiento logrado no se puede observar en los valores de las métricas ya que se analiza cada paquete por separado.

4.4 Conclusiones de la fase

Se cumplieron los objetivos fijados en el inicio del proyecto, ya que a partir de los conocimientos adquiridos en las etapas anteriores, se fue concretando cada uno de los objetivos, en versiones del producto cada vez más completas. Si bien las dos primeras iteraciones estaban orientadas a obtener una versión del producto para la plataforma Link-All, la tercera se enfocó en obtener un prototipo con más funcionalidades. Se obtuvo, entonces, un producto que cumple las especificaciones básicas definidas para Link-All, y también posee funcionalidades adicionales, cumpliendo en todas ellas con las características exigidas para el nivel de calidad 5. Por lo tanto, se logró un producto muy flexible y fácil de extender, por lo que se considera se lograron los objetivos.

5 CONCLUSIONES

Una característica importante de este proyecto, es que en el inicio, no se poseía una definición clara del producto que se deseaba construir. Esto obligó a realizar propuestas y ofrecer alternativas, basadas en el estudio de herramientas y tecnologías similares existentes en el mercado. Este proceso abarcó las dos primeras fases del proyecto, pero fueron fundamentales para obtener una definición concreta del problema y del producto a construir, a través de un prototipo de Interfaz de Usuario detallado.

A partir de esto, se puede afirmar que el modelo de proceso iterativo e incremental utilizado en el proyecto fue el adecuado, ya que permitió encarar el problema desde una perspectiva muy general y básica, para incorporar mejoras sucesivas hasta obtener la solución final.

6 REFERENCIAS

Análisis y Diseño

[1] – UML y Patrones
http://www-306.ibm.com/software/rational/ Setiembre, 2004.

[2] – UML y Patrones
Una introducción al análisis y diseño orientado a objetos y al proceso unificado. LARMAN, CRAIG. Segunda Edición. Ed: Prentice Hall, Inc. (2003) - ISBN: 84-205-3438-2.

Calidad y Testing

[3] – Métricas de Calidad de Diseño - OptimalAdvisor
http://javacentral.compuware.com/pasta/concepts/packageDesign.html Diciembre, 2004.

[4] – Hansel
http://hansel.sourceforge.net Febrero, 2005.
http://sourceforge.net/project/showfiles.php?group_id=54171 Febrero, 2005.

Proyecto de Grado

[5] – Informe Final – Anexo: Documento de Casos de Uso
"Mecanismo de Ayuda Contextual para Sistemas de Información Federados". Proyecto de Grado 2004.
Laura González, Guillermo Roldós, Flavia Serra

[6] – Reunión de Requerimientos 18_06_2004
"Mecanismo de Ayuda Contextual para Sistemas de Información Federados". Proyecto de Grado 2004.
Laura González, Guillermo Roldós, Flavia Serra

[7] – Documento de Requerimientos
"Mecanismo de Ayuda Contextual para Sistemas de Información Federados". Proyecto de Grado 2004.
Laura González, Guillermo Roldós, Flavia Serra

[8] – Alcance del Sistema
"Mecanismo de Ayuda Contextual para Sistemas de Información Federados". Proyecto de Grado 2004.
Laura González, Guillermo Roldós, Flavia Serra

[9] – Glosario
"Mecanismo de Ayuda Contextual para Sistemas de Información Federados". Proyecto de Grado 2004.
Laura González, Guillermo Roldós, Flavia Serra

[10] – Manual de Instalación
"Mecanismo de Ayuda Contextual para Sistemas de Información Federados". Proyecto de Grado 2004.
Laura González, Guillermo Roldós, Flavia Serra

[11] – Planilla de Errores
"Mecanismo de Ayuda Contextual para Sistemas de Información Federados". Proyecto de Grado 2004.
Laura González, Guillermo Roldós, Flavia Serra

[12] – Documento de Reporte de Errores
"Mecanismo de Ayuda Contextual para Sistemas de Información Federados". Proyecto de Grado 2004.
Laura González, Guillermo Roldós, Flavia Serra

[13] – Casos de Prueba de Aceptación
"Mecanismo de Ayuda Contextual para Sistemas de Información Federados". Proyecto de Grado 2004.
Laura González, Guillermo Roldós, Flavia Serra

[14] – Página web
http://www.fing.edu.uy/~pgctxhlp/ Mayo, 2005.

Mecanismo de Ayuda Contextual para Sistemas de Información Federados

Anexo
Calidad y Testing

INCO – Facultad de Ingeniería – UDELAR

Montevideo, Junio 2005

Tutores:

Ing. Federico Piedrabuena
Dr. Ing. Raúl Ruggia

Estudiantes:

Laura González
Guillermo Roldós
Flavia Serra

ÍNDICE

1 INTRODUCCIÓN

1.1 Descripción General

El proyecto de grado está dentro de un marco de Control de Calidad basado en la tesis de maestría del Ing. Diego Vallespir [1], por lo que deben cumplirse un conjunto de actividades de acuerdo al nivel de calidad exigido para cada una de las tareas.

El Sistema de Ayuda Contextual cuenta con una característica importante, es un producto que será puesto en producción y deberá cumplir con un conjunto de especificaciones impuestas por el proyecto Link-All [5], por lo que todas aquellas funcionalidades que forman parte de dicho producto deberán cumplir con un nivel de calidad alto. A su vez, forma parte de un proyecto de grado, lo que implica la investigación de otros sistemas de ayuda y de diferentes tecnologías buscando desarrollar funcionalidades que los sistemas existentes no ofrecen, que si bien son importantes como tesis de grado, no exigen un nivel de calidad alto.

1.2 Niveles de Calidad

1.2.1 Nivel de Calidad 3

En este nivel [1] se conocen las fallas del sistema y cuanta funcionalidad del mismo ha sido cubierta por los casos de prueba. De esta manera, se puede conocer qué cosas el sistema no ofrece y preparar futuras versiones sabiendo de antemano qué es lo que hay que corregir y se tiene una idea de qué falta testear. Este nivel permite tener confianza en la funcionalidad del sistema, pudiendo hacer demos y teniendo conocimiento de algunas fallas del mismo.

1.2.2 Nivel de Calidad 5

En este nivel [1] se resalta la disminución en la tasa de faltas en el sistema y la posibilidad de reuso. Es recomendable estar en este nivel si el producto se va a continuar en años subsiguientes, si se requiere un número relativamente bajo de fallas o que éstas no sean graves.

1.3 Niveles de Calidad y Casos de Uso

En esta sección se presenta la correspondencia entre los diferentes casos de uso y los niveles de calidad exigidos en el proyecto de grado.

Casos de Uso correspondientes al nivel de calidad 5:

- Ver Ayuda Contextual
- Seleccionar Tema
- Ver Contenido
- Buscar Texto
- Consultar Índice
- Consultar Glosario
- Ver Temas Relacionados

Casos de Uso correspondientes al nivel de calidad 3:

- Utilizar Asistente
- Mantener Favoritos

Para una descripción completa de cada uno de los casos de uso, puede consultarse en [7].

2 TESTING

2.1 Pruebas realizadas

La fase de construcción se dividió en tres iteraciones[1]. Al finalizar cada una de ellas, se cerró una versión del sistema y se realizó el testing correspondiente. El testing esta determinado por los requerimientos, como se mencionó anteriormente, todos aquellos que se corresponden con la plataforma Link-All exigen un nivel de calidad más alto que el resto.

En las dos primeras etapas el desarrollo se concentró en los requerimientos del sistema LinkAll, lo cual exigía mayor nivel de calidad [1], ya que el producto será puesto en producción (nivel de calidad 5). La última etapa consistió en obtener una solución que contemplara los objetivos académicos del proyecto de grado, por lo que se

[1] Para más información sobre el Proceso de Desarrollo consultar el Anexo [8].

realizó un producto extensible, flexible y adaptable. A pesar de que en esta etapa se exigió un nivel de calidad 3, se respetó estrictamente el nivel de calidad 5 para las funcionalidades exigidas por la plataforma Link-All.

2.1.1 Primera iteración

2.1.1.1 Versión 1.0

Para todas las operaciones brindadas por los componentes que intervienen en casos de uso que tienen nivel de calidad alto, se buscó cubrir la mayor cantidad de casos posibles, por lo que se realizó un testing unitario de todas las clases, lo que consiste en varias pruebas para cada método de cada clase. La herramienta que se utilizó fue la API de Java: JUnit.

2.1.2 Segunda iteración

2.1.2.1 Versión 2.1

En esta segunda iteración se testearon componentes, no se realizaron test unitarios para las clases como en la versión anterior, y dado que JUnit permite reutilizar los códigos de pruebas, sólo se generaron códigos para las nuevas operaciones. Una vez corregidas todas las fallas detectadas, se generó la versión 2.2 del sistema y se realizan nuevamente todas las pruebas unitarias de los componentes. En este caso, no se obtuvo ninguna falla. Luego, se realizó el testing de casos de uso de la versión 2.2 y se detectaron dos fallas de prioridad baja.

2.1.3 Tercera Iteración

2.1.3.1 Versión 3.1

Se realizó la prueba de todas las operaciones brindadas por los componentes ya que éstas intervienen en casos de uso con nivel de calidad 5, cubriendo la mayor cantidad de entradas posibles (incluyendo casos de borde, etc.).

2.1.3.2 Versión 3.2

Se corrigieron las fallas encontradas en la versión 3.1 y se probaron todos los componentes. No se detectó ninguna falla, por lo que se realizó una prueba de sistema.

2.2 Resultados Obtenidos

A continuación se presentan los resultados obtenidos en las pruebas realizadas al finalizar cada iteración. Se brinda un cuadro con las cantidades de casos de prueba y fallas encontradas, así como un gráfico de las tasas de error categorizadas por prioridad, alta, media y baja.

2.2.1 Primera Iteración

2.2.1.1 Versión 1.0

Para el testing de clase se realizaron 324 pruebas que lanzaron los resultados que se presentan en la tabla 1. Los datos se representan en la gráfica de la figura 13.

Número de Pruebas Realizadas	324	
Errores Reportados		
Prioridad Alta	3	(0,93%)
Prioridad Media	11	(3,40%)
Prioridad Baja	15	(4,63%)
TOTAL	29	(8,95%)

Tabla 1

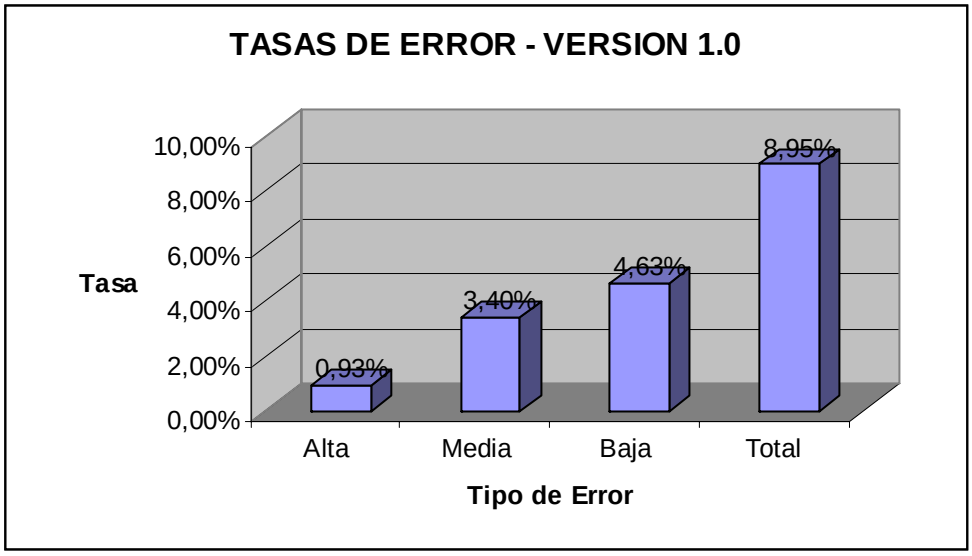


Figura 13

2.2.2 Segunda iteración

2.2.2.1 Versión 2.1

En el testing de componentes se realizaron 187 pruebas que lanzaron los resultados que se presentan en la tabla 2. Los datos de esta versión se comparan con los resultados obtenidos en la versión 1.0 del sistema, en la gráfica de la figura 14.

Número de Pruebas Realizadas	187	
Errores Reportados		
Prioridad Alta	0	(0%)
Prioridad Media	1	(0,53%)
Prioridad Baja	9	(4,81%)
TOTAL	10	(5,35%)

Tabla 2

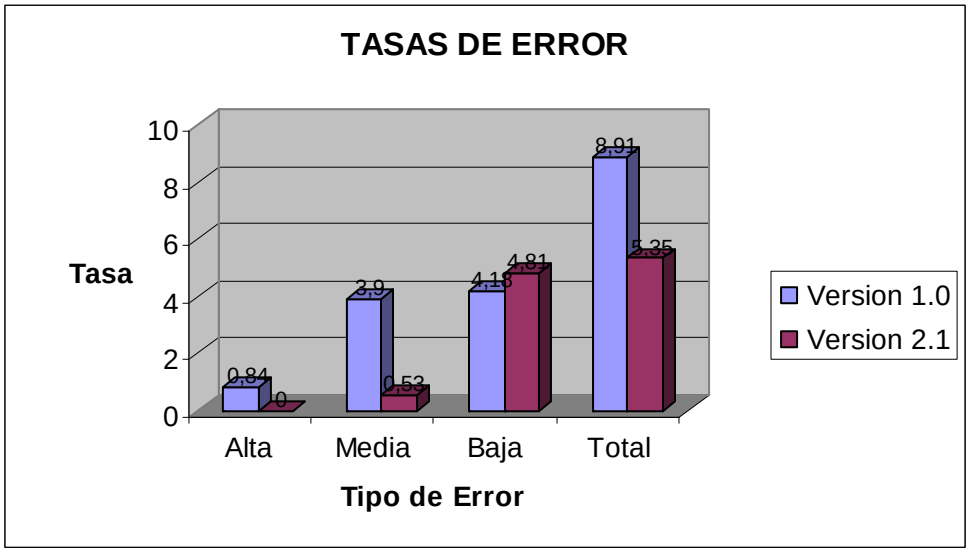


Figura 14

2.2.3 Tercera Iteración

2.2.3.1 Versión 3.1

Se realizaron un total de 259 casos de prueba. Los datos arrojados se presentan en la tabla 3.

Número de Pruebas Realizadas	259	
Errores Reportados		
Prioridad Alta	0	(0%)
Prioridad Media	4	(1,54%)
Prioridad Baja	9	(3,47%)
TOTAL	13	(5,02%)

Tabla 3

En la figura 15, se comparan los resultados obtenidos en esta iteración con los resultados obtenidos en las iteraciones anteriores.

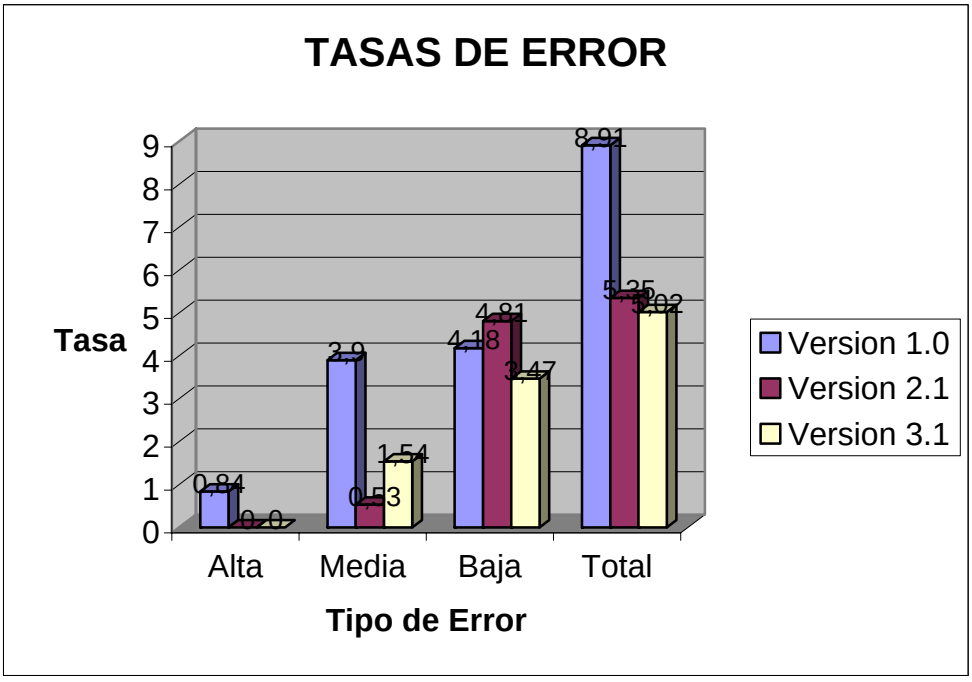


Figura 15

2.2.3.2 Versión 3.2

En la tabla 4 se presentan los resultados de la prueba de casos de uso de la versión 3.2.

Errores Reportados	
Prioridad Alta	0
Prioridad Media	1
Prioridad Baja	4
Prioridad Muy Baja	1
TOTAL	6

Tabla 4

2.3 Medida de cubrimiento

Para medir el cubrimiento de las pruebas realizadas se utilizó la herramienta Hansel [3]. Ésta es un proyecto desarrollado en el ámbito de Sourgeforge.net, con el propósito de medir el cubrimiento de las pruebas que se realizan utilizando JUnit. La finalidad de esta herramienta es determinar el porcentaje de cubrimiento del testing realizado, y para esto, genera un caso de prueba para cada bloque de código existente en las clases que se testean. Se arroja una falla para cada bloque que no sea efectivamente recorrido durante el testing.

Se llegó a la conclusión de que la medida del cubrimiento fue del 76%. Esta cifra varía si no se consideran las clases que implementan el patrón de diseño DAO, ya que en éstas no se realizó el testing de los métodos de actualización de datos (insert, update y delete), porque no son utilizados por el sistema. Por lo tanto, sin considerar dichas clases, se logra un cubrimiento del 81%, pero es importante destacar que estas cifras se corresponden con la versión 1.0 del sistema, ya que para las versiones posteriores no se pudo aplicar Hansel, ya que esta herramienta no funciona con EJB's.

3 CALIDAD DEL DISEÑO

Se utilizó la herramienta OptimalAdvisor [4] para determinar las métricas de calidad de diseño, asociadas a los componentes de software [20]. Esta herramienta analiza diferentes métricas, dando la posibilidad de realizar un análisis detallado de las dependencias existentes entre los diferentes paquetes, lo que da la posibilidad de detectar posibles mejoras en el diseño. Las métricas de calidad analizan los siguientes puntos:

- ADP: ciclos de dependencia en los paquetes.
- DIP: las clases deberían depender de clases abstractas o interfaces. Por lo tanto, se mide el porcentaje de clases implementadas que cumplen con este criterio.
- SAP: los paquetes deberían ser tan abstractos como estables.
- EP: los componentes implementados deberían ocultar los detalles de implementación a aquellos que los utilizan.
- LSP: límite máximo de elementos en un paquete. Este criterio es bastante relativo, ya que depende mucho del tamaño de la aplicación. Tanto es así, que la misma herramienta permite modificar el límite de elementos para adaptarlo a las necesidades.

3.1 Versión 1.0

Esta versión del sistema representaba una solución parcial del problema, ya que inicialmente se buscó obtener un sistema que sólo contemplara los requerimientos de la plataforma Link-All. Observando la tabla 5, que se presenta a continuación, se concluye lo siguiente acerca de las métricas de calidad:

- El indicador menos favorable es el DIP.
- Con respecto al indicador SAP, parecería que el valor más bajo se encuentra en el paquete *ctxhlp.uiHelp*. Por lo que se planteó estudiar este paquete en particular, para determinar las causas de esta medida. En cuanto a EP, dos paquetes tienen un valor nulo en este indicador (*ctxhlp.common* y *ctxhlp.services*), lo que también implicó una revisión de los mismos.

- Para los indicadores ADP y LSP se obtuvieron valores óptimos (salvo para el paquete *ctxhlp.common*).

Si bien en esta versión se tenía una solución parcial, fue necesario profundizar en las métricas que no arrojaron buenos resultados, para poder evaluar la incidencia en la calidad de la solución final del problema.

Paquete	ADP	DIP	SAP	EP	LSP
ctxhlp.common	1	0.128	0.548	0	0.6
ctxhlp.help	1	0.043	0.583	0.25	1
ctxhlp.helpAPI	1	0.005	0.467	0.4	1
ctxhlp.mappingAPI	1	0.009	0.467	0.2	1
ctxhlp.services	1	1	0.333	0	1
ctxhlp.uiHelp	1	0.511	0.188	1	1
Ctxhlp	1	0.116	0.393	0.333	1

Tabla 5

En el documento de Arquitectura [6] puede consultarse la descripción de cada uno de los paquetes listados en la tabla 5.

3.2 Versión 2.1

En esta versión del producto se continuaba trabajando sobre la solución parcial, pero de todas formas se aplicaron patrones de diseño que no habían sido utilizados en la versión anterior (Factory, Singleton), a su vez, se tomó la decisión de que dos de los componentes existentes debían ser EJB´s. Con esto, se buscaba mejorar el diseño y dar extensibilidad al producto para poder obtener una solución final totalmente genérica como se

planteó en los objetivos, pero siempre respetando las exigencias en el nivel de calidad correspondiente a cada uno de los requerimientos. Sin embargo, los resultados no mejoraron sustancialmente en las métricas DIP y SAP, aunque la métrica EP pasó de tener un valor de 0.333 a tener un valor óptimo de 1.

Los reportes provistos por la herramienta OptimalAdvisor (tabla 6), permitieron observar que la gran mayoría de las dependencias que se presentaron correspondían al paquete *Common*. Este era un paquete de utilitarios, que contenía clases que debían ser vistas por todos los componentes, lo que permite concluir que si bien algunas métricas se mantenían bajas, se conocía el motivo, por lo que su valor no se asociaba a un mal diseño, por lo tanto no incidiría en la calidad del producto final.

Paquete	**ADP**	**DIP**	**SAP**	**EP**	**LSP**
ctxhlp.common	1	0.069	0.60	0.053	0.1
ctxhlp.help	1	0.022	0.583	0.25	1
ctxhlp.helpAPI	1	0.009	0.524	0.143	1
ctxhlp.mappingAPI	1	0.017	0.524	0.286	1
ctxhlp.services	1	1	0.262	0	1
ctxhlp.uiHelp	1	0.5	0.188	1	1
Ctxhlp	1	0.127	0.347	1	1

Tabla 6

En el documento de Arquitectura [6] puede consultarse la descripción de cada uno de los paquetes listados en la tabla 6.

3.3 Versión 3.1

En esta versión del producto se redefinió totalmente la estructura de paquetes y componentes, logrando un gran nivel de encapsulamiento entre los componentes. Esto no se refleja en la métrica EP (que mide justamente eso), ya que está expresada por paquete, y no en forma global. No tiene sentido hablar de encapsulamietno dentro de cada paquete, pero a su vez, no hay nada que mida el mismo a nivel global, por lo que EP no es representativa. Las otras métricas muestran buenos valores excepto SAP, que es una métrica un tanto abstracta como para intentar concluir acerca del diseño. A continuación, en la tabla 7, se presenta en detalle las métricas de calidad arrojadas para esta versión del software.

Paquete	**ADP**	**DIP**	**SAP**	**EP**	**LSP**
Ctxhlp	0.998	0.539	0.306	0.333	1
Favorites	1	1	0.375	0	1
Linkall	1	1	0.682	0.875	1
Lucene	1	1	0.286	0	1
Mapping	1	1	0.375	0	1
relational	1	0.078	0.44	0	1

Tabla 7

En el documento de Arquitectura [6] puede consultarse la descripción de cada uno de los paquetes listados en la tabla 7.

4 CONCLUSIONES

El testing realizado fue de gran apoyo con respecto a las decisiones tomadas, ya que a la hora de realizar cambios en el diseño, se consideraron no solo los test realizados, sino también las métricas de calidad que fueron una guía constante a lo largo del proyecto.

Al comenzar el testing utilizando JUnit, fue necesario definir cada uno de los casos de pruebas, pero es importante destacar que para los sucesivos test, esta API de Java permite reutilizar cada uno de los códigos generados, y si bien insumió un tiempo considerable al principio, éste fue compensado con los resultados obtenidos.

A su vez, fue de gran utilidad el tiempo invertido en la herramienta Hansel (el cual es mínimo si se compara con el tiempo necesario para realizar las pruebas y los beneficios obtenidos), ya que a partir de la medida del cubrimiento de las pruebas realizadas utilizando JUnit, se pudo tener conocimiento de qué bloques de código no estaban siendo considerados en los test, lo que resultó una guía a la hora de definir nuevos casos de pruebas.

En cuanto a cómo medir la calidad del diseño, la herramienta OptimalAdvisor fue sumamente útil para dicho propósito, así como para detectar posibles defectos u oportunidades de mejora.

Finalmente, teniendo en cuenta la disminución de fallas totales en los test realizados en cada una de las versiones del producto, se puede concluir que a pesar del tiempo invertido (el cual fue importante considerando los tiempos del proyecto), en la implementación de los test y en el aprendizaje de las diferentes herramientas (OptimalAdvisor y Hansel), los resultados obtenidos impactan definitivamente en la calidad del producto.

5 REFERENCIAS

Control de Calidad y Testing

[1] – Control de Calidad
http://www.fing.edu.uy/~dvallesp/Tesis/webActividades/index2.htm Junio, 2004.
Responsable: Ing. Diego Vallespir. Instituto de Computación, Facultad de Ingeniería, UdelaR. Marzo, 2005.

[2] – JUnit
http://jwebunit.sourceforge.net/ Diciembre, 2004.
http://www.junit.org/index.htm Diciembre, 2004.

[3] – Hansel
http://hansel.sourceforge.net Febrero, 2005.
http://sourceforge.net/project/showfiles.php?group_id=54171 Febrero, 2005.

[4] – Métricas de Calidad de Diseño - OptimalAdvisor
http://javacentral.compuware.com/pasta/concepts/packageDesign.html Diciembre, 2004.

Proyecto Link-All

[5] – Sistemas de Información Federados a través de la Web (Link-All)
http://www.link-all.org/PublicSite/Summary.htm Junio, 2004.
http://www.fing.edu.uy/inco/proyectos/linkall/Documentos/Descripcion-LinkAll.doc Junio, 2004.

Proyecto de Grado

[6] – Informe Final – Anexo: Documento de Arquitectura
"Mecanismo de Ayuda Contextual para Sistemas de Información Federados". Proyecto de grado 2004.
Laura González, Guillermo Roldós, Flavia Serra

[7] – Informe Final – Anexo: Documento de Casos de Uso
"Mecanismo de Ayuda Contextual para Sistemas de Información Federados". Proyecto de grado 2004.
Laura González, Guillermo Roldós, Flavia Serra

[8] – Informe Final – Anexo: Proceso de Desarrollo
"Mecanismo de Ayuda Contextual para Sistemas de Información Federados". Proyecto de grado 2004.
Laura González, Guillermo Roldós, Flavia Serra

Mecanismo de Ayuda Contextual para Sistemas de Información Federados

Anexo
Documento de Arquitectura

INCO – Facultad de Ingeniería – UDELAR

Montevideo, Junio 2005

Tutores:

Ing. Federico Piedrabuena
Dr. Ing. Raúl Ruggia

Estudiantes:

Laura González
Guillermo Roldós
Flavia Serra

ÍNDICE

1 VISTA DEL MODELO DE CASOS DE USO

1.1 Diagrama de los Casos de Uso Relevantes a la Arquitectura

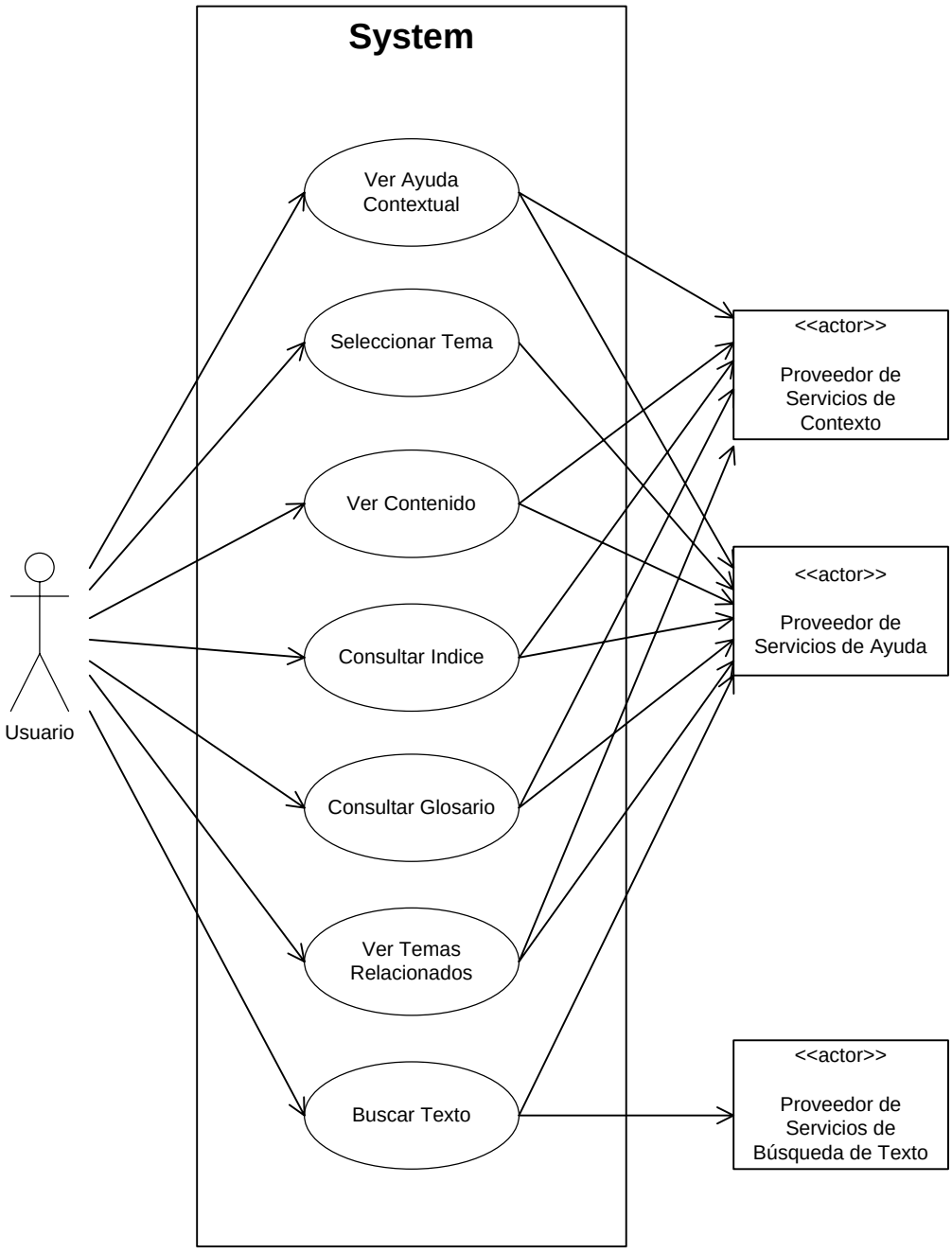


Figura 1

1.2 Casos de Uso Relevantes a la Arquitectura

La lista completa de los casos de uso está disponible en [1].

1.2.1 Caso de Uso: Ver Ayuda Contextual

Un usuario que utiliza la plataforma y posiblemente está haciendo uso de una de sus aplicaciones, hace clic sobre el link de ayuda. El sistema le brinda ayuda o información, de acuerdo a determinadas variables, como pueden ser características del usuario, aplicación y/o funcionalidad que está ejecutando, posibles errores que el usuario haya cometido, etc.

1.2.2 Caso de Uso: Seleccionar Tema

En la pantalla de ayuda, a menudo se tendrán links a temas de ayuda, ya sea porque se ha consultado el árbol de contenido, se ha consultado el índice, o se ha efectuado alguna otra acción. En esa situación, el usuario hace clic sobre el link del tema cuya información desea consultar y se presenta en pantalla dicha información.

1.2.3 Caso de Uso: Ver Contenido

El usuario hace clic sobre el link Contenido y el contenido de la ayuda es mostrado en pantalla a través de un árbol jerárquico.

1.2.4 Caso de Uso: Consultar Índice

El usuario hace clic sobre el link índice y se muestran en pantalla las entradas del índice y sus temas asociados. El usuario puede ingresar una cadena de caracteres para filtrar las entradas del índice mostradas.

1.2.5 Caso de Uso: Consultar Glosario

El usuario hace clic sobre el link glosario y se muestran en pantalla los términos del glosario. Cuando el usuario hace clic sobre uno de dichos términos se muestra en pantalla la definición del mismo.

1.2.6 Caso de Uso: Ver Temas Relacionados

El usuario hace clic sobre el link Temas Relacionados y se le presentan en pantalla una lista de temas relacionados al tema de ayuda mostrado en pantalla.

1.2.7 Caso de Uso: Buscar Texto

El usuario hace clic sobre el link buscar, e ingresa una palabra o frase en un cuadro de texto. El usuario hace clic sobre el botón buscar, y se muestran en pantalla los temas cuyo texto contienen dicha palabra o frase.

2 VISTA DEL MODELO DE ANÁLISIS

2.1 Diagrama de Paquetes

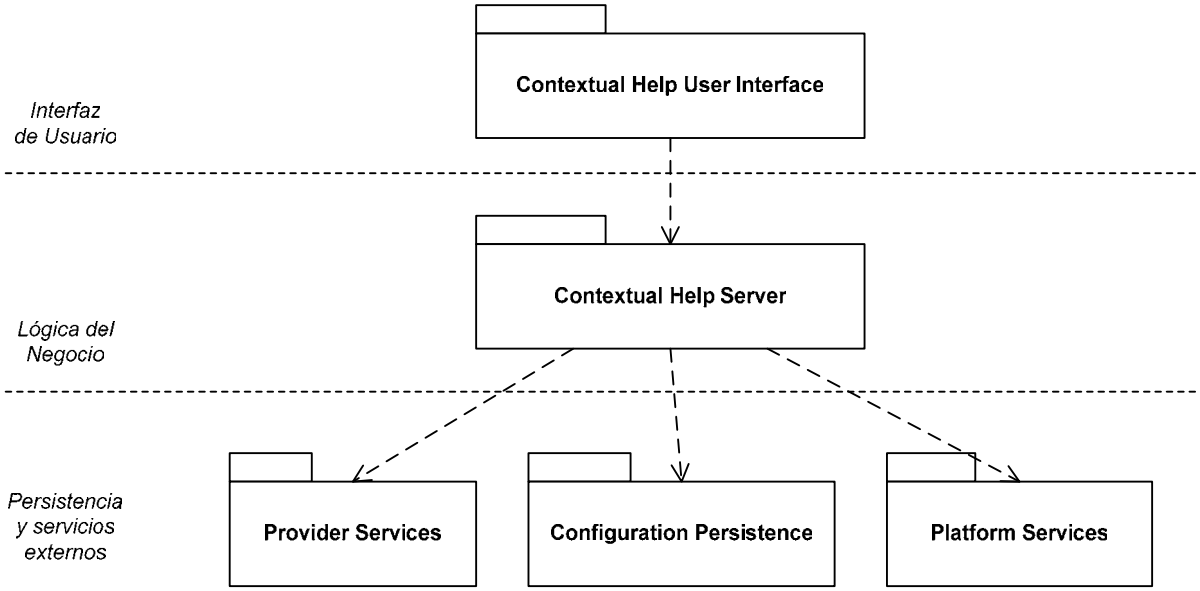


Figura 2

2.2 Análisis de Paquetes

2.2.1 Contextual Help User Inteface

Se encarga de la interacción con el usuario a través de páginas web para mostrarle la información de ayuda que obtiene de las capas inferiores.

2.2.2 Contextual Help Server

Contiene la lógica del negocio. Recibe pedidos de ayuda desde la interfaz de usuario y los resuelve utilizando los servicios provistos por la capa inferior. Para poder obtener la ayuda contextual maneja los conceptos de contexto, ayuda y mapeos entre ambos. Estos conceptos se describirán más detalladamente en los capítulos siguientes.

2.2.3 Configuration Persistence

Provee los valores de configuración del Sistema. Éstos son definidos por el Administrador de la plataforma, y permiten que el Sistema sea extensible. Ejemplos de estos parámetros pueden ser qué proveedores de servicios están disponibles, para que éstos sean utilizados por el Contextual Help Server.

2.2.4 Platform Services

Provee las interfaces para que el Sistema se integre en una determinada plataforma de aplicaciones federadas. Estas interfaces se refieren a las variables que conforman el contexto de ejecución, así como los permisos asociados a los usuarios.

2.2.5 Provider Services

Provee las interfaces para que el Sistema de Ayuda utilice los servicios de diferentes proveedores externos. Existen dos tipos de proveedores definidos:

- proveedores de ayuda
- proveedores de búsqueda de texto

Cualquier proveedor de estos servicios serán componentes externos al Sistema y deberán implementar las interfaces provistas para interactuar con el mismo. Esto brinda gran extensibilidad ya que el Contextual Help Server conocerá todos los proveedores con los que cuenta e interactuará con ellos, de forma de mostrar siempre la ayuda más conveniente en cada situación.

3 TRAZABILIDAD DESDE EL MODELO DE ANÁLISIS AL MODELO DE DISEÑO

En la figura 3 se observa la descomposición de cada uno de los paquetes detallados en el capítulo anterior, con los subsistemas que se mostrarán en el capítulo siguiente.

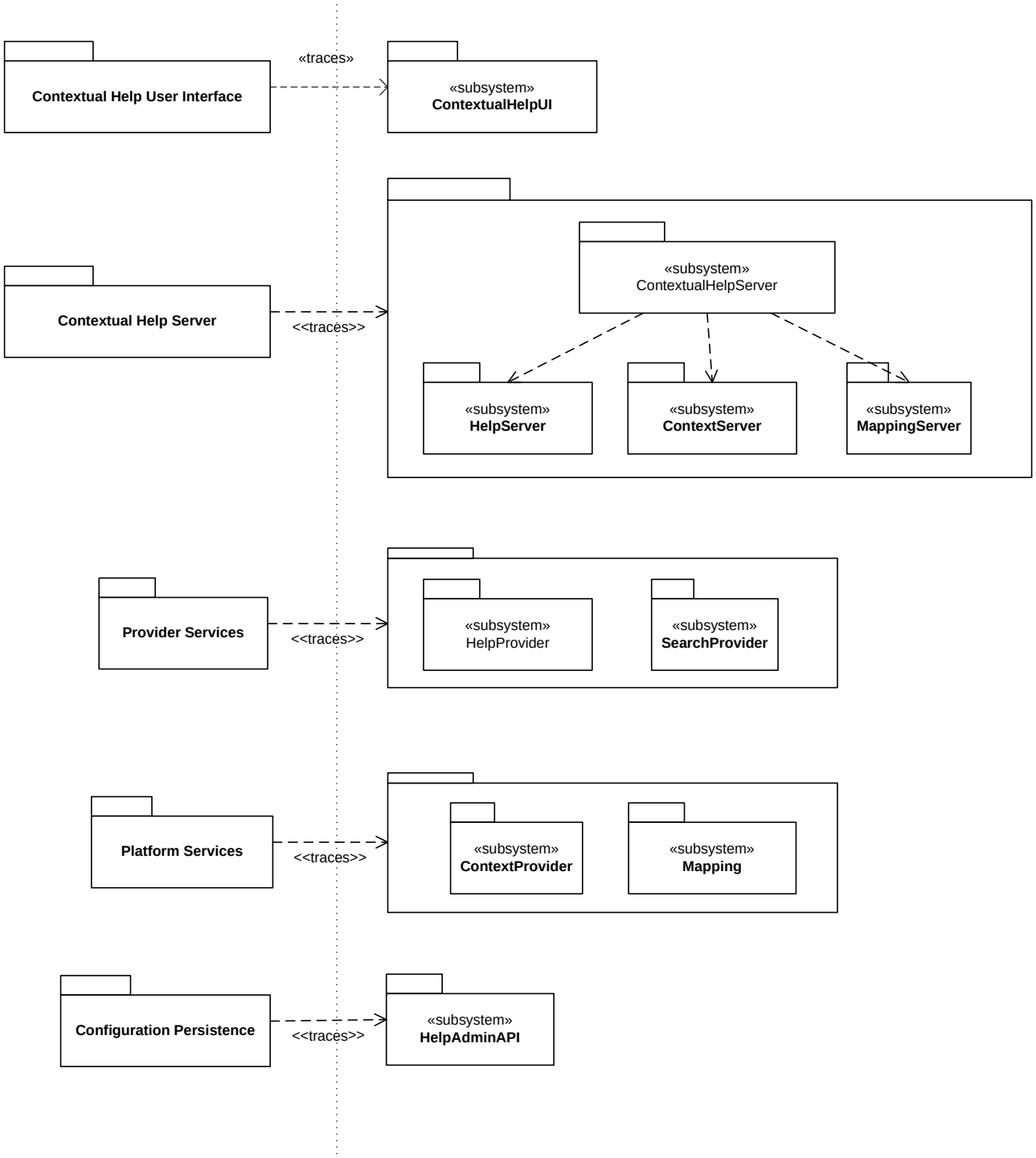


Figura 3

4 VISTA DEL MODELO DE DISEÑO

4.1 Descomposición en Subsistemas

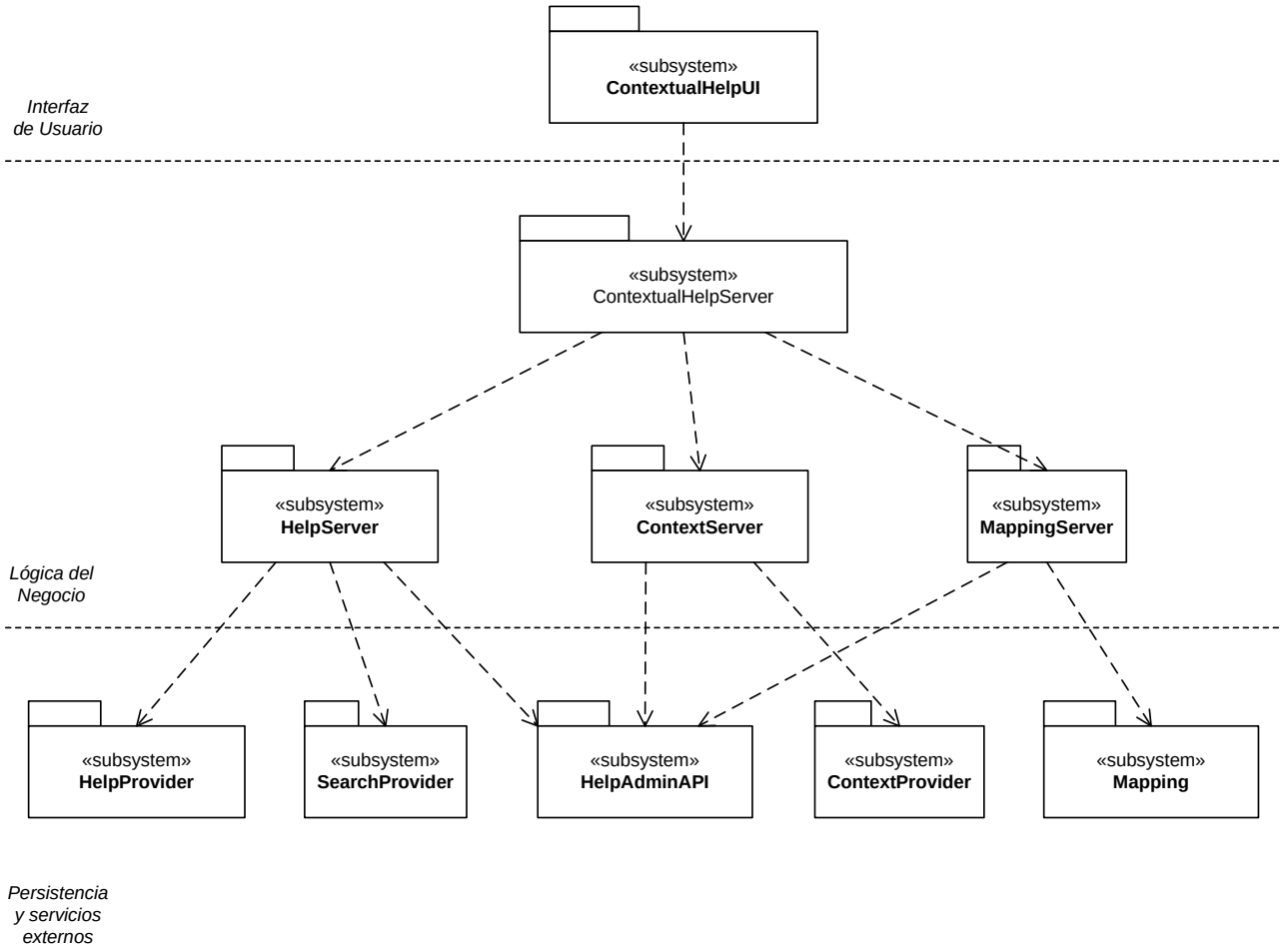


Figura 4

4.1.1 Subsistema ContextualHelpUI

Presenta la ayuda contextual cuando el usuario la requiere y a su vez brinda varias formas de navegación a través de ella, como ser la tabla de contenido, el índice, el glosario, favoritos, búsqueda de texto, etc. Este subsistema implementa el patrón de diseño Model View Controller (MVC) (más información sobre patrones de diseño en [2]), obteniendo los datos a mostrar del *ContextualHelpServer*.

4.1.2 Subsistema ContextualHelpServer

Es el subsistema más importante, ya que en él se encuentra la mayor parte de la lógica de negocio. Recibe pedidos del *ContextualHelpUI*, y los procesa y responde apoyándose en otros subsistemas de este paquete, implementando el patrón de diseño Business Delegator.

La interfaz que brinda es la *IContextualHelpService*, la cual brinda operaciones para obtener todos los datos que componen la ayuda, como se verá más adelante.

4.1.3 Subsistema HelpServer

Es responsable de encontrar ayuda para una aplicación en un determinado idioma. Recibe los pedidos de información de ayuda provenientes del *ContextualHelpServer*, y consulta al subsistema *HelpAdminAPI* para obtener la lista de proveedores de ayuda externos con los que se está trabajando. A cada uno de ellos le solicita la información que corresponda a la aplicación en el idioma especificado utilizando las operaciones provistas por la interfaz *IHelpProvider* del subsistema *HelpProvider*.

Una operación similar será realizada cuando se reciben pedidos de búsqueda de texto en los contenidos de las ayudas, obteniendo para eso la lista de proveedores de búsqueda externos del *HelpAdminAPI* y utilizando sus servicios a través de las operaciones provistas por la interfaz *ISearchProvider* del subsistema *SearchProvider*.

La interfaz que brinda este subsistema es la *IHelpService*, la cual brinda operaciones para obtener datos de ayuda puntuales, como se verá más adelante.

4.1.4 Subsistema ContextServer

Se encarga de proporcionar la información relativa al contexto de ejecución cuando se produce un pedido de ayuda. Para obtener dicha información, utiliza los servicios brindados por el subsistema *HelpAdminAPI* para conocer qué componente externo implementa las interfaces *ICurrentContextProvider*, *ISecurityContextProvider* e *IRemoteContextProvider* del subsistema *ContextProvider*. Brinda la interfaz *IContextService*.

4.1.5 Subsistema MappingServer

Se encarga de obtener los contenidos de ayuda correspondientes a un contexto de ejecución, relacionados con un determinado proveedor de ayuda. Este componente debe consultar al *HelpAdminAPI* para conocer la información de los mapeos entre contextos y contenidos de ayuda, y utiliza además los servicios de los componentes que implementan la interfaz *IMapping* del subsistema *ctxhlp.mapping*. Brinda la interfaz *IMappingService*.

4.1.6 Subsistema HelpProvider

Está formado por una sola interfaz llamada *IHelpProvider*, la que debe ser implementada por los proveedores de ayuda que deseen incorporarse a la plataforma. Estos proveedores son componentes externos al Sistema, y para integrarse deben implementar las operaciones que se definen en dicha interfaz.

4.1.7 Subsistema SearchProvider

Está formado por una sola interfaz llamada *ISearchProvider*, la que debe ser implementada por los proveedores de búsqueda de texto que deseen incorporarse a la plataforma. Estos proveedores son componentes externos al Sistema, y para integrarse deben implementar las operaciones que se definen en dicha interfaz.

4.1.8 Subsistema HelpAdminAPI

Es el encargado de obtener y devolver los valores de los parámetros que indican qué componentes externos interactúan con el sistema, así como valores por defecto, etc. De esta forma, los otros subsistemas de este paquete conocerán a quién deben solicitar determinada información. Brinda la interfaz *IMappingService*.

4.1.9 Subsistema ContextProvider

Está formado por tres interfaces llamadas *ICurrentContextProvider*, *ISecurityContextProvider* e *IRemoteContextProvider*, las que deben ser implementadas por los componentes que dan soporte para la integración con la plataforma. Estos componentes si bien son externos al Sistema, son necesarios para que el Sistema se pueda integrar de forma óptima en una plataforma determinada. Las operaciones que se definen en dichas interfaces se detallarán más adelante.

4.1.10 Subsistema Mapping

Está formado por una sola interfaz llamada *IMapping*, la que debe ser implementada por los componentes que dan soporte para la integración con la plataforma. Debe existir una implementación por cada proveedor de ayuda soportado ya que brinda funcionalidades que relacionan a los contenidos de ayuda que maneja cada proveedor con la definición del contexto.

4.2 Diseño de Casos de Uso

Los siguientes diagramas muestran cómo intervienen los subsistemas mencionados en el punto anterior en cada caso de uso.

4.2.1 Diseño de Caso de Uso: Ver Ayuda Contextual

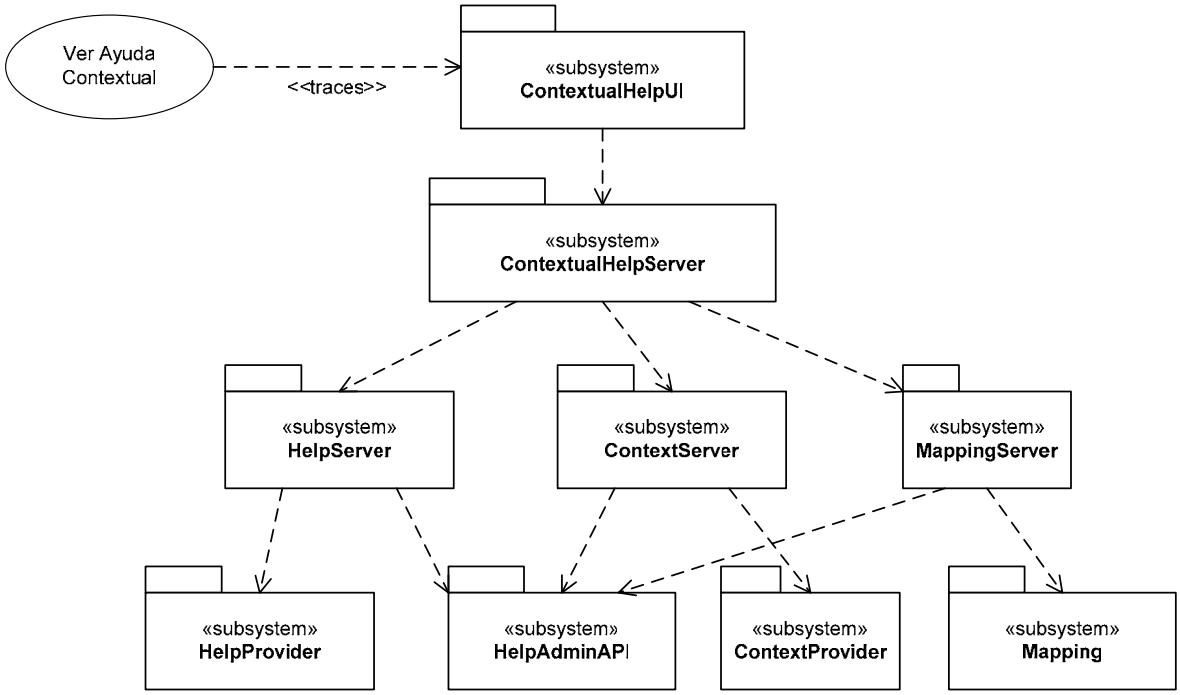


Figura 5

4.2.2 Diseño de Caso de Uso: Seleccionar Tema

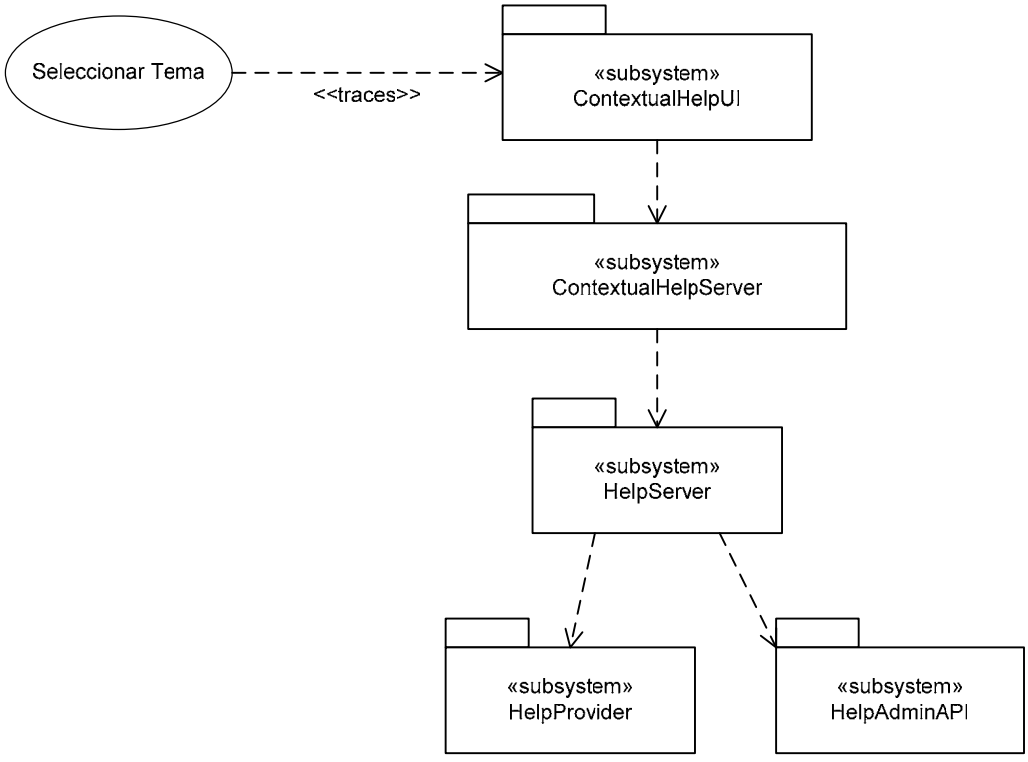


Figura 6

4.2.3 Diseño de Caso de Uso: Ver Contenido

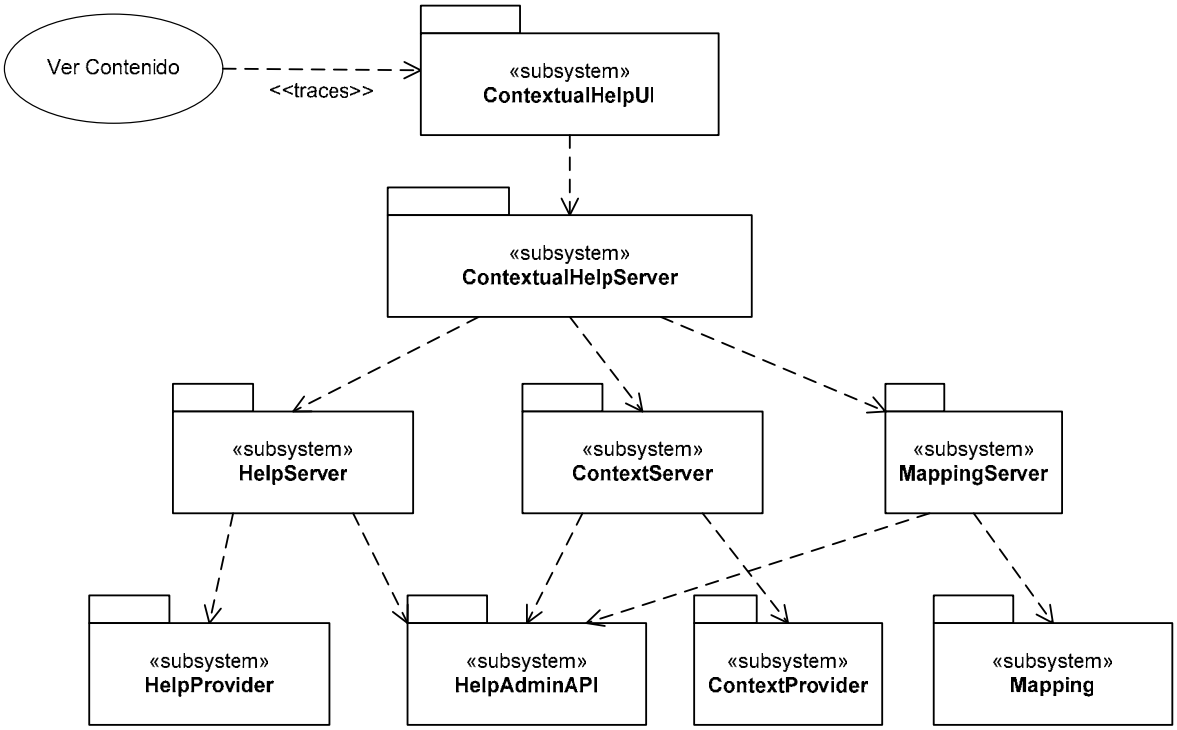


Figura 7

4.2.4 Diseño de Caso de Uso: Consultar Índice

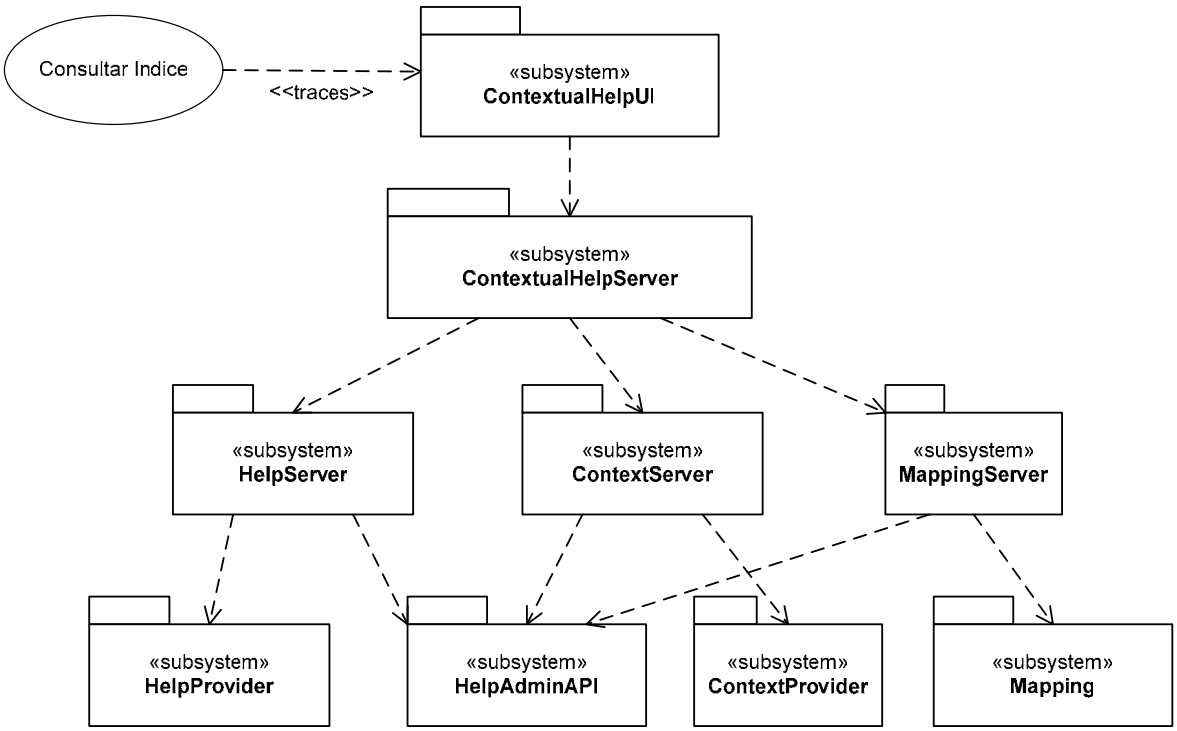


Figura 8

4.2.5 Diseño de Caso de Uso: Consultar Glosario

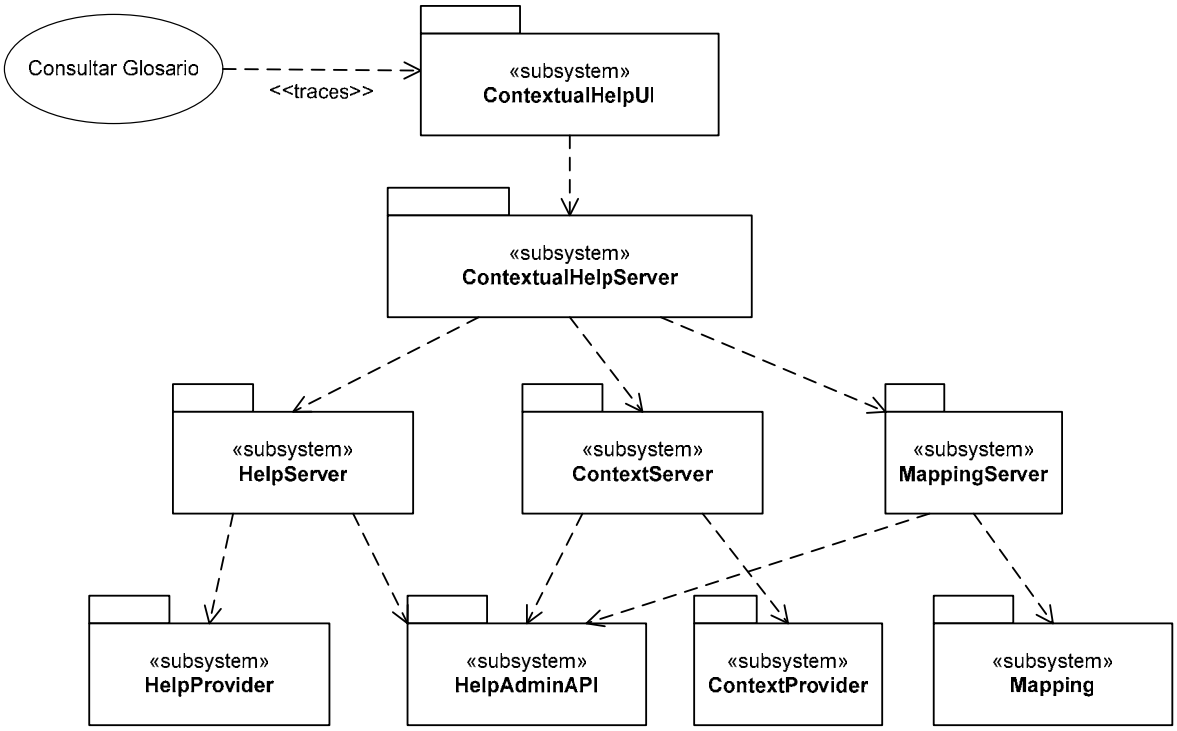


Figura 9

4.2.6 Diseño de Caso de Uso: Ver Temas Relacionados

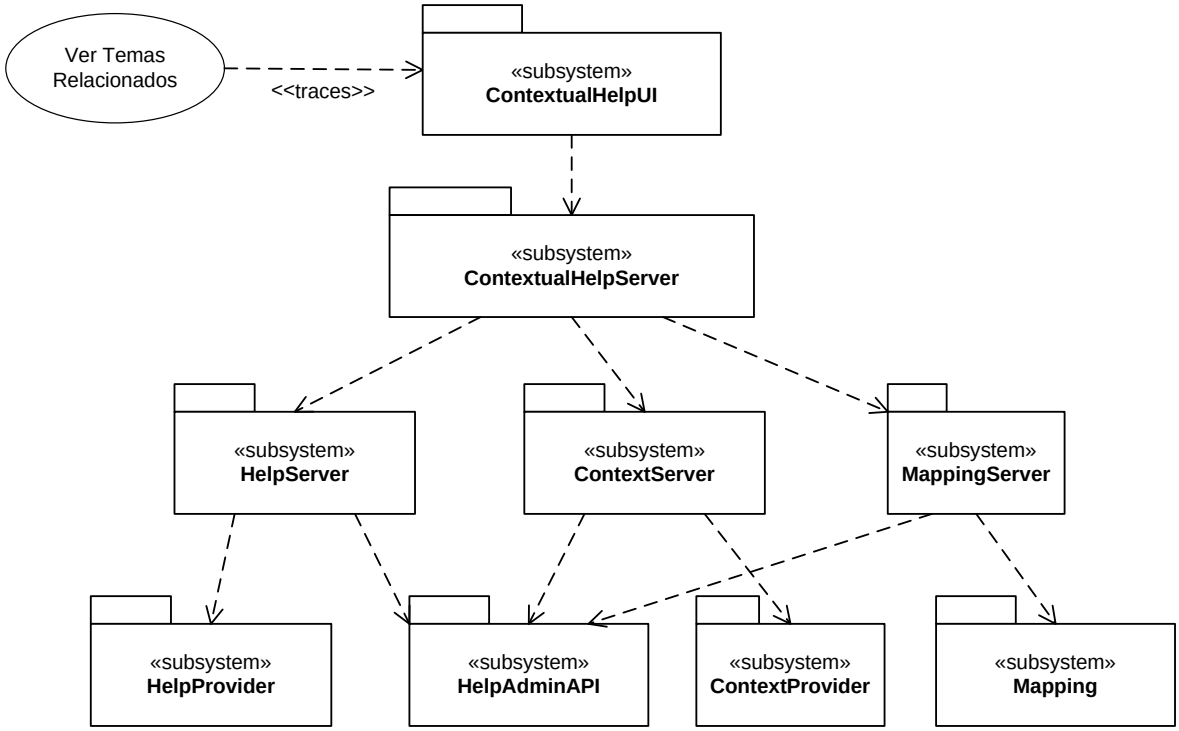


Figura 10

4.2.7 Diseño de Caso de Uso: Buscar Texto

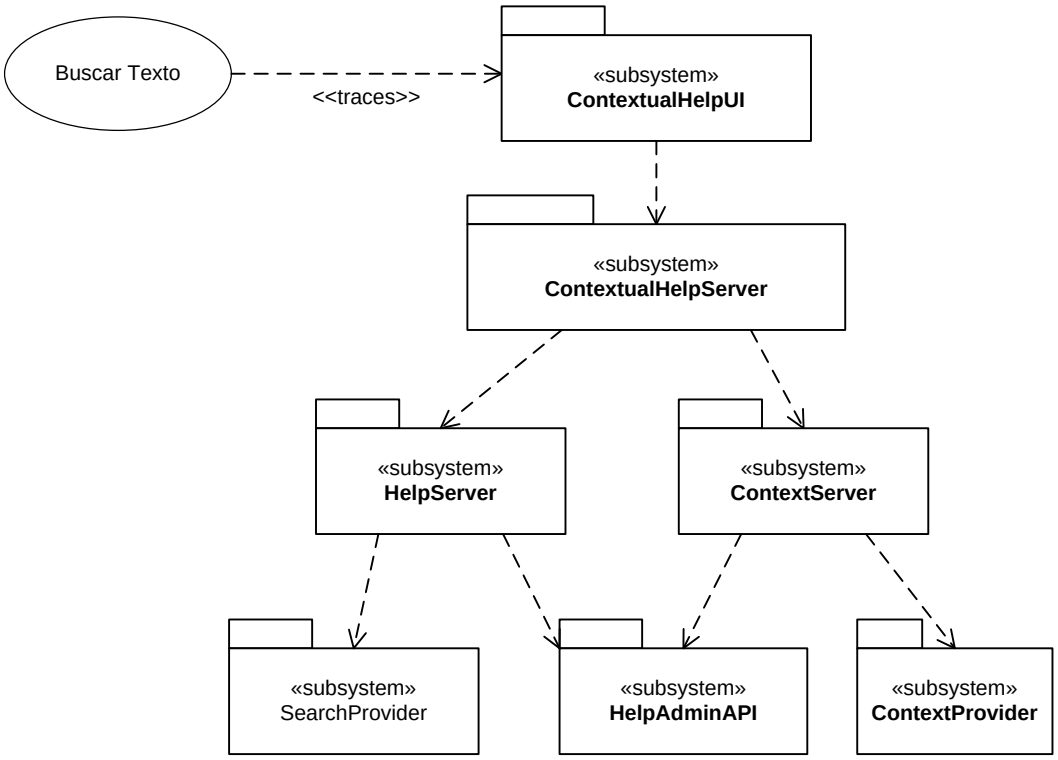


Figura 11

5 VISTA DEL MODELO DE DISTRIBUCIÓN

5.1 Diagrama de Distribución

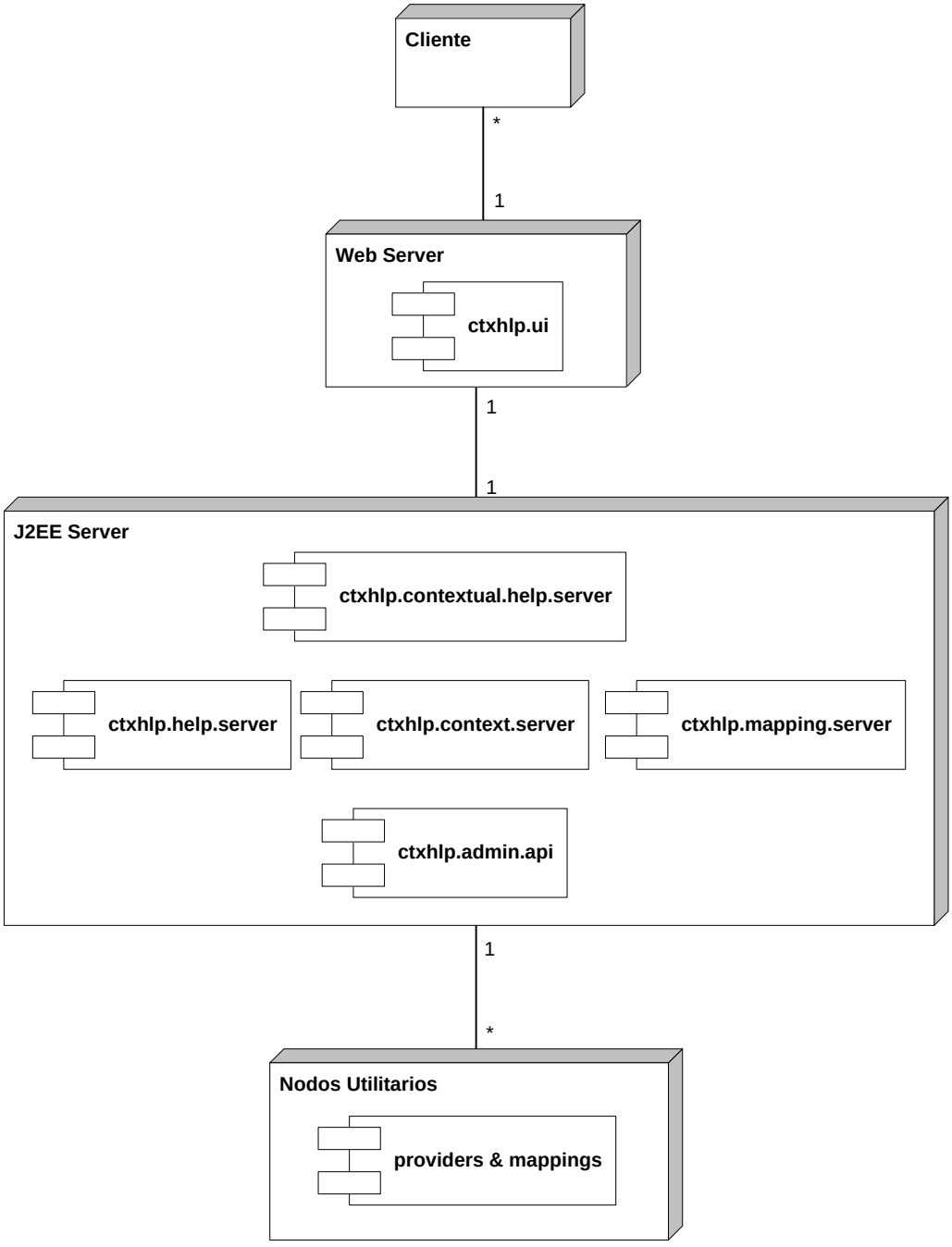


Figura 12

5.2 Nodos

5.2.1 Cliente

Es un PC capaz de ejecutar un browser con Macromedia Flash Player. No requiere grandes cantidades de memoria ni un gran procesador.

5.2.2 Web Server

En este nodo se ejecuta el Servidor Web con el que se comunican los clientes, por lo que debe existir una instalación de alguno de ellos (por ej. Tomcat). Debido a que se grabarán algunas variables de sesión (algunas de las cuales pueden llegar a ser pesadas, dependiendo de la ayuda creada por las aplicaciones), es necesario

que este nodo tenga una buena capacidad de RAM, a efectos de que no haga "swapping" a disco, con lo que el proceso se haría bastante más lento.

5.2.3 J2EE Server

En este nodo se ejecuta el contenedor J2EE con el que se comunicará el servidor Web, por lo que debe existir una instalación de alguno de ellos (por ej. JBoss). Este nodo puede ser físicamente el mismo equipo que contiene al servidor Web, ya que muchos servidores de aplicaciones contienen además un servidor Web. Debido a que se resuelve aquí prácticamente toda la lógica de las solicitudes de ayuda, es deseable que este nodo tenga la mayor capacidad posible de memoria y procesador.

5.2.4 Nodos utilitarios

Estos nodos son los que se encargarán de proveer servicios de ayuda y/o de búsqueda de texto, por lo que sus características pueden variar mucho según las necesidades de cada proveedor.

5.3 Conexiones

5.3.1 Cliente a Web Server

Esta es una conexión a Internet, que puede ser de buena o mala calidad, o del ancho de banda que se disponga. Aunque la interfaz con el usuario fue pensada para que funcionara correctamente con bajos anchos de banda o mala calidad en la conexión, cuanto mejor sea la calidad de éstos se obtendrán mejores resultados en la carga de la interfaz de usuario.

5.3.2 Web Server a J2EE Server

Esta conexión debe ser lo más rápida y confiable posible, ya que de otra manera se verá resentido el funcionamiento de todos los clientes conectados. Se debe tener en cuenta que de acuerdo a los volúmenes de datos que se transfieran (los que pueden ser grandes dependiendo del tipo de las ayudas creadas por las aplicaciones), es necesario que esta conexión esté bajo permanente estudio de carga, de modo de detectar si es necesario ampliar el ancho de banda.

5.3.3 J2EE Server a Nodos utilitarios

El ancho de banda y disponibilidad de estas conexiones depende de las características de cada proveedor. Sin embargo, dado que es posible definirse más de un proveedor de ayuda y de búsquedas de texto, si alguno de ellos no está disponible, siempre se podrá obtener información de los otros (en la medida que estén disponibles).

6 TRAZABILIDAD DESDE EL MODELO DE DISEÑO AL MODELO DE IMPLEMENTACIÓN:

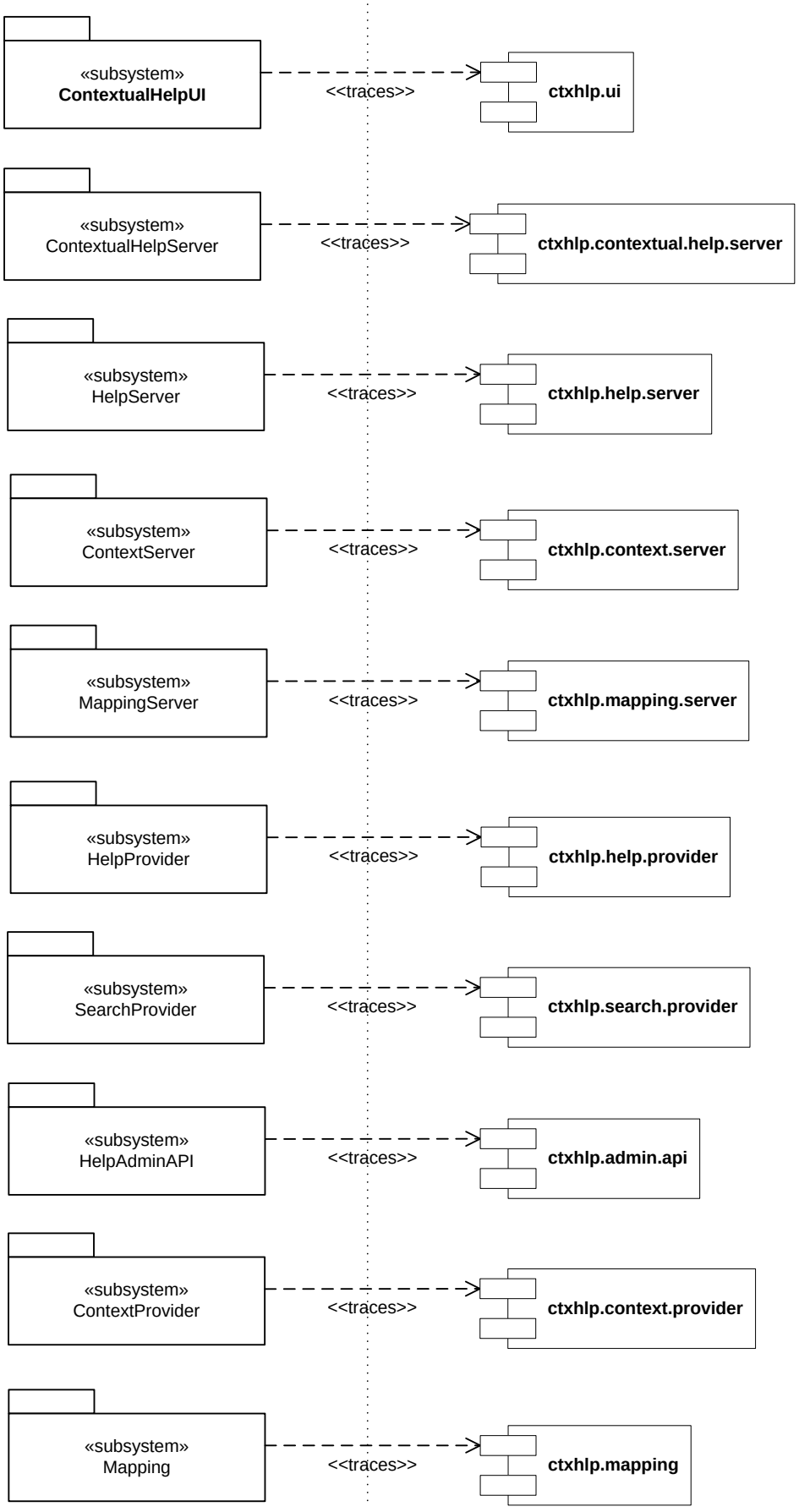


Figura 13

7 VISTA DEL MODELO DE IMPLEMENTACIÓN

7.1 Componentes

7.1.1 Componente *ctxhlp.ui*

Implementa el subsistema *ContextualHelpUI* para la interacción con el usuario. Sus principales clases son:

- un Servlet que recibe los pedidos desde el navegador del cliente
- una serie de manejadores (llamados "Handlers") que derivan la solicitud a la capa lógica
- una serie de páginas Html que se devuelven al usuario.

Estas páginas Html contienen películas Flash desarrolladas mediante la tecnología OpenLaszlo [3] [4]. La forma de comunicación entre los Handlers y los objetos Flash de las páginas Html es a través de las variables de sesión, es decir, los Handlers graban el resultado de las operaciones como variables de sesión en el Servidor Web, las que son accedidas por las JSP para formar dinámicamente el contenido de los objetos Flash.

Otra característica importante de este componente es que implementa la internacionalización de todos los mensajes que se muestran al usuario (inclusive "labels" y "caption" en botones), de forma que todos los textos que el usuario vea estén expresados en su idioma.

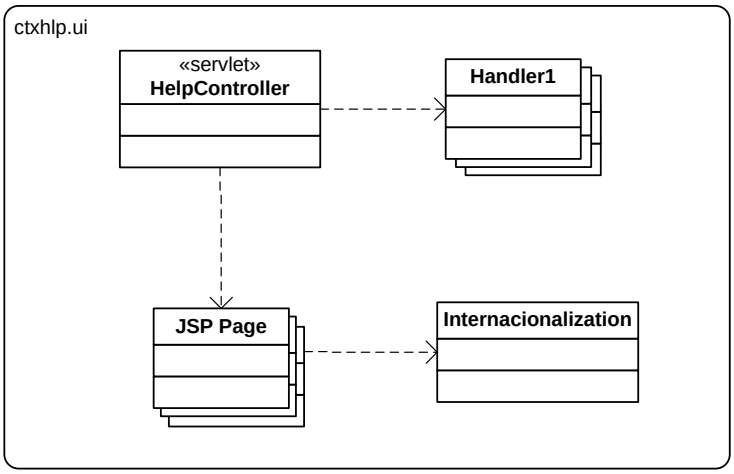


Figura 14

7.1.2 Componente *ctxhlp.contextual.help.server*

Implementa el subsistema *ContextualHelpServer*, el cual se encarga de gestionar los pedidos provenientes desde la UI. Se trata de un Enterprise Java Bean de tipo "Sessionless" [5], con el objetivo de que el servidor Web que gestiona la interfaz de usuario pueda estar corriendo en otro nodo diferente al que está corriendo el contenedor J2EE en el cual se ubica el Sistema de Ayuda.

Este EJB ofrece todas las operaciones que es capaz de resolver el Sistema, algunas de las cuales devuelven datos puntuales y otras son más complejas. En la figura 15 se observa el diagrama de la interfaz que expone este componente, la cual es la cara visible de todo el Sistema.

«interface» ctxhlp.contextual.help.server::**IContextualHelpService**
+getContextualHelp(in ctx : IContext, in providersInfo : IHelpProviderInfo[ ]) : Help *+getContextualHelp(in application : string, in language : string, in ctx : IContext, in providersInfo : IHelpProviderInfo[ ]) : Help* *+getContextualHelp(in application : string, in language : string, in provider : string, in ctx : IContext, in providersInfo : IHelpProviderInfo[ ]) : Help* *+getContentTable(in application : string, in language : string, in provider : string, in ctx : IContext, in providersInfo : IHelpProviderInfo[ ]) : IContentTable* *+getIndex(in application : string, in language : string, in provider : string, in ctx : IContext, in providersInfo : IHelpProviderInfo[ ]) : IIndex* *+getGlossary(in application : string, in language : string, in provider : string, in ctx : IContext, in providersInfo : IHelpProviderInfo[ ]) : IGlossary* *+getContent(in application : string, in language : string, in content : string, in provider : string, in ctx : IContext, in providersInfo : IHelpProviderInfo[ ]) : Content* *+getRelatedContents(in application : string, in language : string, in contentCode : string, in provider : string, in ctx : IContext, in providersInfo : IHelpProviderInfo[ ]) : Content[ ]* *+getSearchResult(in text : string, in application : string, in language : string, in ctx : IContext, in providersInfo : IHelpProviderInfo[ ]) : Content[ ]* *+getAvailableApps(in ctx : IContext, in providersInfo : IHelpProviderInfo[ ]) : Application* *+getFavorites(in ctx : IContext) : Content[ ]* *+addToFavorites(in application : string, in language : string, in contentCode : string, in provider : string, in title : string, in ctx : IContext)* *+removeFromFavorites(in application : string, in language : string, in contentCode : string, in provider : string, in ctx : IContext)*

Figura 15

7.1.3 Componente *ctxhlp.help.server*

Implementa el subsistema *HelpServer*, el cual se encarga de obtener toda la información relativa a la ayuda. Esta ayuda es independiente del contexto, en el sentido de que este componente no maneja ninguna variable de contexto, sino solamente lo relativo a la ayuda. Es importante destacar que este componente no accede directamente a la información almacenada, sino que para eso utiliza los servicios de los proveedores de ayuda (llamados *HelpProviders*) y de los proveedores de búsqueda de texto (llamados *SearchProviders*) disponibles. Estos proveedores son componentes externos al Sistema, los que brindan funcionalidades a través de servicios para que sean accedidos por el componente *ctxhlp.help.server*.

El componente puede obtener información relativa a un proveedor o de todos ellos, de acuerdo a los valores que reciba como parámetros. En el caso que se requiera información de todos los proveedores, obtiene este dato del componente *ctxhlp.admin.api*. En la figura 16 se presenta el diagrama de la interfaz que expone este componente.

«interface» ctxhlp.help.server::**IHelpService**
+getContextualHelp(in application : string, in language : string, in provider : string, in providersInfo : IHelpProviderInfo[ ]) : Help *+getContextualHelp(in application : string, in language : string, in providersInfo : IHelpProviderInfo[ ]) : Help* *+getContentTable(in application : string, in language : string, in provider : string, in providersInfo : IHelpProviderInfo[ ]) : IContentTable* *+getIndex(in application : string, in language : string, in provider : string, in providersInfo : IHelpProviderInfo[ ]) : IIndex* *+getGlossary(in application : string, in language : string, in provider : string, in providersInfo : IHelpProviderInfo[ ]) : IGlossary* *+getContent(in application : string, in language : string, in content : string, in provider : string, in providersInfo : IHelpProviderInfo[ ]) : Content* *+getRelatedContents(in application : string, in language : string, in contentCode : string, in provider : string, in providersInfo : IHelpProviderInfo[ ]) : Content[ ]* *+getSearchResult(in application : string, in language : string, in text : string) : Content[ ]* *+getAllContents(in application : string, in language : string, in provider : string, in providersInfo : IHelpProviderInfo[ ]) : IContent[ ]* *+createSearchIndex(in language : string, in providersInfo : IHelpProviderInfo[ ])* *+getFavorites(in user : IUser) : Content[ ]* *+addToFavorites(in application : string, in language : string, in contentCode : string, in provider : string, in title : string, in user : IUser)* *+removeFromFavorites(in application : string, in language : string, in contentCode : string, in provider : string, in user : IUser)*

Figura 16

7.1.4 Componente *ctxhlp.context.server*

Implementa el subsistema *ContextServer*, el cual se encarga de obtener la información relativa al contexto de ejecución. Es decir, cuando se solicita ayuda, para que ésta sea contextual deben obtenerse todas las variables del contexto que sean útiles para definir qué ayuda se debe mostrar. Aquí pueden incidir variables tales como:

- qué aplicación se está ejecutando
- qué operación dentro de ésta
- cuál es el idioma del usuario
- sus permisos, etc.

Para obtener estos datos, se utilizan los servicios de los componentes externos que brindan esta información relativa a la plataforma en la que se está ejecutando la ayuda. Estos componentes externos al Sistema deben implementar tres interfaces para que el *ctxhlp.context.server* pueda accederlas: *ICurrentContextProvider* (que se encarga de obtener el contexto de ejecución), *ISecurityContextProvider* (que se encarga de obtener los permisos de ejecución del usuario), e *IRemoteContextProvider* (que se encarga de determinar si la aplicación se ejecuta en un nodo que no es el local y obtener el contexto remoto en ese caso). En la figura 17 se presenta el diagrama de la interfaz que expone este componente.

«interface» ctxhlp.context.server::**IContextServer**
+getApplication(in context : IContext) : Application *+getLanguage(in context : IContext) : string* *+getCurrentContextId(in context : IContext) : IContextId* *+getUser(in context : IContext) : IUser* *+getAvailableApplications(in context : IContext) : Application[ ]* *+getAvailableContextsId(in application : string, in context : IContext) : IContextId[ ]* *+getNotAvailableContextIds(in application : Application, in context : IContext) : IContextId[ ]* *+validateApplication(in application : string, in context : IContext) : bool* *+isLocal(in application : string, in context : IContext) : bool* *+getRemoteContext(in application : string, in context : IContext) : IContext* *+getRemoteHost(in application : string, in context : IContext) : string*

Figura 17

7.1.5 Componente *ctxhlp.mapping.server*

Implementa el subsistema *MappingServer*, el cual se encarga de obtener la información relativa a las correspondencias (o mapeos) entre el contexto de ejecución y los contenidos de ayuda que corresponden a dichos mapeos para cada proveedor de ayuda. Es decir, para un mismo identificador de contexto pueden corresponder diferentes contenidos de ayuda, según qué proveedor maneje esta información. Por lo tanto, cuando se incorpora un *HelpProvider* se debe generar el componente de mapeos correspondiente implementando la interfaz *IMapping*.

Esta información es manejada por el *ctxhlp.mapping.server*, el cual obtiene del *ctxhlp.admin.api* la información de qué proveedores existen y qué componentes gestionan los mapeos con cada uno de ellos. En la figura 18 se presenta el diagrama de la interfaz que expone este componente.

Figura 18

7.1.6 Componente *ctxhlp.help.provider*

Implementa el subsistema *HelpProvider*, y está formado por una sola interfaz llamada *IHelpProvider*. Esta interfaz debe ser implementada por los proveedores de ayuda que se incorporan a la plataforma, y asegura la comunicación del Sistema con ellos a través de operaciones básicas, las que internamente resolverán de acuerdo a sus propias características y diseños. En la figura 19 se presenta el diagrama de la interfaz que expone este componente.

Figura 19

7.1.7 Componente *ctxhlp.search.provider*

Implementa el subsistema *SearchProvider*, y está formado por una sola interfaz llamada *ISearchProvider*. Este interfaz debe ser implementada por los proveedores de búsqueda de texto que se incorporan a la plataforma, y asegura la comunicación del Sistema con ellos a través de operaciones básicas, las que internamente resolverán de acuerdo a sus propias características y diseños. En la figura 20 se presenta el diagrama de la interfaz que expone este componente.

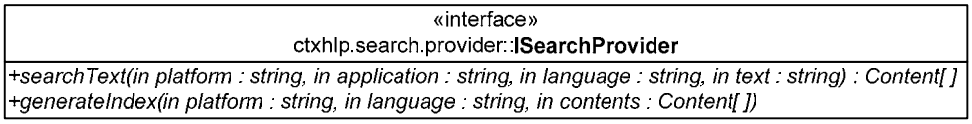


Figura 20

7.1.8 Componente *ctxhlp.admin.api*

Implementa el subsistema *HelpAdminAPI*, el cual se encarga de guardar y recuperar toda la información del entorno de ejecución del Sistema de ayuda. Este componente brinda operaciones tanto para guardar información (que será utilizada por el Administrador de la plataforma), como para recuperarla (que será usada por los subsistemas *HelpServer*, *ContextServer* y *MappingServer*, como se explicó anteriormente). Los datos que se graban a través de este componente pueden ser:

- el idioma por defecto,
- los proveedores de ayuda y de búsqueda con que se cuenta,
- quién implementa los servicios correspondientes a la plataforma, etc.

En la figura 21 se presenta el diagrama de la interfaz que expone este componente.

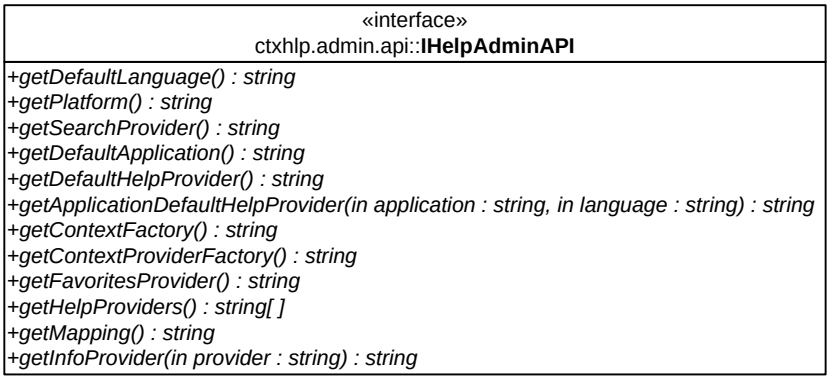


Figura 21

7.1.9 Componente *ctxhlp.context.provider*

Implementa el subsistema *ContextProvider*, y está formado por tres interfaces llamadas *ICurrentContextProvider* (permite obtener las variables que forman el contexto de ejecución de una determinada solicitud de ayuda), *ISecurityContextProvider* (permite obtener los recursos sobre los que tiene permiso el usuario que solicita la ayuda), *IRemoteContextProvider* (permite obtener el contexto remoto si la aplicación se ejecuta en un nodo que no es el local). Estas interfaces deben ser implementadas por los componentes que vinculan al Sistema con la plataforma, y aseguran la comunicación del Sistema con ellos a través de operaciones básicas, las que internamente resolverán de acuerdo a sus propias características y diseños. En las figuras 22, 23 y 24 se presentan los diagramas de las interfaces que expone este componente.

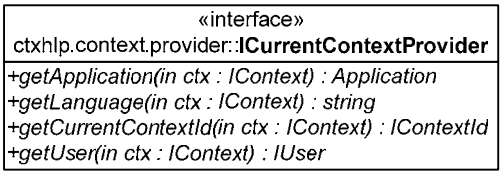


Figura 22

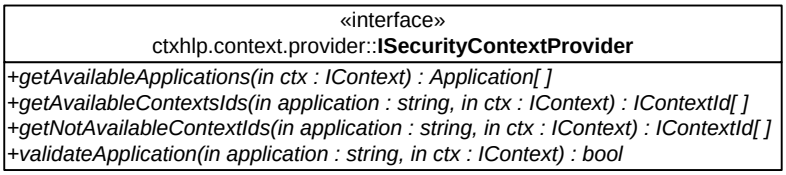


Figura 23

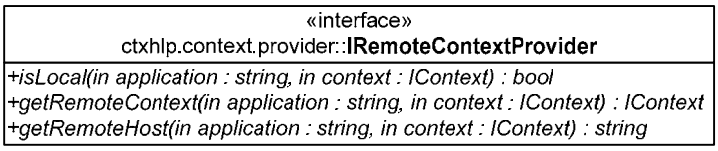


Figura 24

7.1.10 Componente *ctxhlp.mapping*

Implementa el subsistema *Mapping*, y está formado por una única interfaz llamada *IMapping*. Esta interfaz debe ser implementada para cada proveedor de ayuda que se disponga, ya que brinda los mapeos entre los identificadores de contexto de la plataforma, con los contenidos de ayuda de dicho *HelpProvider*. En la figura 25 se presenta los diagramas de las interfaces que expone este componente.

Figura 25

8 REFERENCIAS

Documentos del Proyecto

[1] – Informe Final – Anexo: Documento de Casos de Uso
"Mecanismo de Ayuda Contextual para Sistemas de Información Federados". Proyecto de Grado 2004.
Laura González, Guillermo Roldós, Flavia Serra

Patrones de diseño

[2] – UML y Patrones
Una introducción al análisis y diseño orientado a objetos y al proceso unificado. LARMAN, CRAIG. Segunda Edición. Ed: Prentice Hall, Inc. (2003) - ISBN: 84-205-3438-2.

Tecnologías

[3] – Macromedia Flash
http://www.macromedia.com/software/flash/ Julio, 2004.

[4] – OpenLaszlo
http://www.laszlosystems.com Abril, 2005.
http://www.openlaszlo.org Abril, 2005.

[5] – J2EE
http://java.sun.com/j2ee/index.jsp Agosto, 2004.

Mecanismo de Ayuda Contextual para Sistemas de Información Federados

Anexo
Documentación Técnica

INCO – Facultad de Ingeniería – UDELAR

Montevideo, Junio 2005

Tutores:

Ing. Federico Piedrabuena
Dr. Ing. Raúl Ruggia

Estudiantes:

Laura González
Guillermo Roldós
Flavia Serra

ÍNDICE

1 INTRODUCCIÓN

El Mecanismo de Ayuda Contextual para Sistemas de Información Federados (MACSIF) es un framework que permite proveer funcionalidades de ayuda en línea a aplicaciones que integran sistemas de información federados a través de la Web.

Además de manejar los conceptos básicos que conforman la ayuda de una aplicación [1] [2] [3] [4] [5], MACSIF integra conceptos que permiten establecer el contexto de ejecución de una aplicación, pudiendo de esta forma brindar ayuda efectivamente sensible al contexto.

Si bien en la actualidad existen varios formatos para describir el contenido de la ayuda de una aplicación [12], MACSIF define operaciones de alto nivel para acceder y consultar dicha ayuda, de manera que se independiza del formato en el que se describe la misma. A su vez, y de acuerdo a la tendencia actual hacia arquitecturas orientadas a servicios [6] [7], MACSIF provee un mecanismo por el cual la ayuda de una aplicación podría ser obtenida tanto de forma local como remota.

2 DESCRIPCIÓN

En esta sección se explica en qué entornos o ambientes es aplicable MACSIF, y se describen las funcionalidades y conceptos básicos utilizados por el framework.

2.1 Contexto

MACSIF está orientado a Sistemas de Información Federados a través de la Web (SIFW). A grandes rasgos un SIFW está compuesto por un conjunto de aplicaciones que comparten datos y servicios. Estas aplicaciones pueden estar distribuidas en uno o varios servidores.

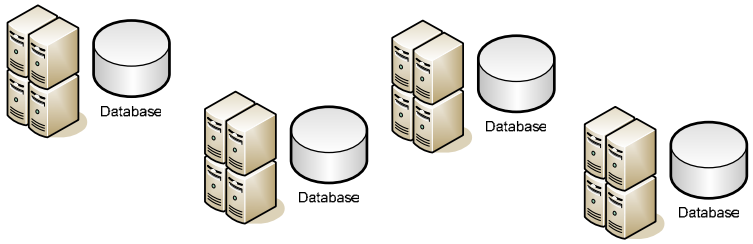


Figura 1

En este contexto, se entiende como aplicación ***(application)*** a un conjunto de elementos presentables en un explorador web (páginas html, imágenes, elementos multimedia, etc.) que permiten a usuarios realizar una serie de tareas relacionadas. Los objetivos de MACSIF son entonces brindar ayuda en línea a los usuarios de estas aplicaciones, proveer un mecanismo por el cual sea posible adaptar el framework a las características de un SIFW especifico y permitir la inclusión de utilitarios y/o servicios que ayuden a la concreción de los objetivos anteriores.

La ayuda ***(help)*** de una aplicación está compuesta por una tabla de contenido, un índice, un glosario y un conjunto de contenidos. Una aplicación puede tener asociado un conjunto de ayudas, una por cada idioma manejado por la aplicación.

El contexto ***(context)*** de ejecución define cómo será presentada la ayuda al usuario. Éste está compuesto principalmente por la siguiente información del usuario:

- tarea que está realizando
- perfil

De este modo dos usuarios que solicitan ayuda podrían obtener información distinta si sus contextos de ejecución son diferentes.

2.2 Descripción general

Con la finalidad de aclarar los conceptos, en la figura 2 se presenta un esquema general de lo que sería la solución final de una implementación de MACSIF para un SIFW.

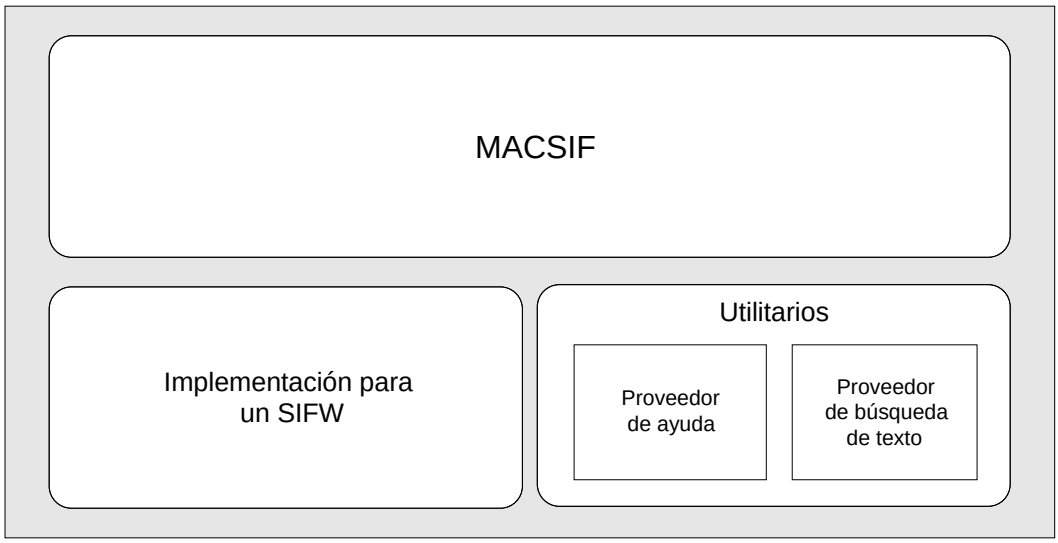


Figura 2

2.3 Ayuda en línea para usuarios de aplicaciones

El objetivo de MACSIF en relación a los usuarios de las aplicaciones es presentar ayuda sensible al contexto en un formato que les resulte conocido [8]. Para ilustrar mejor estos conceptos, en la figura 3 se muestra una copia de la pantalla inicial de un pedido de ayuda cualquiera obtenido utilizando MACSIF:

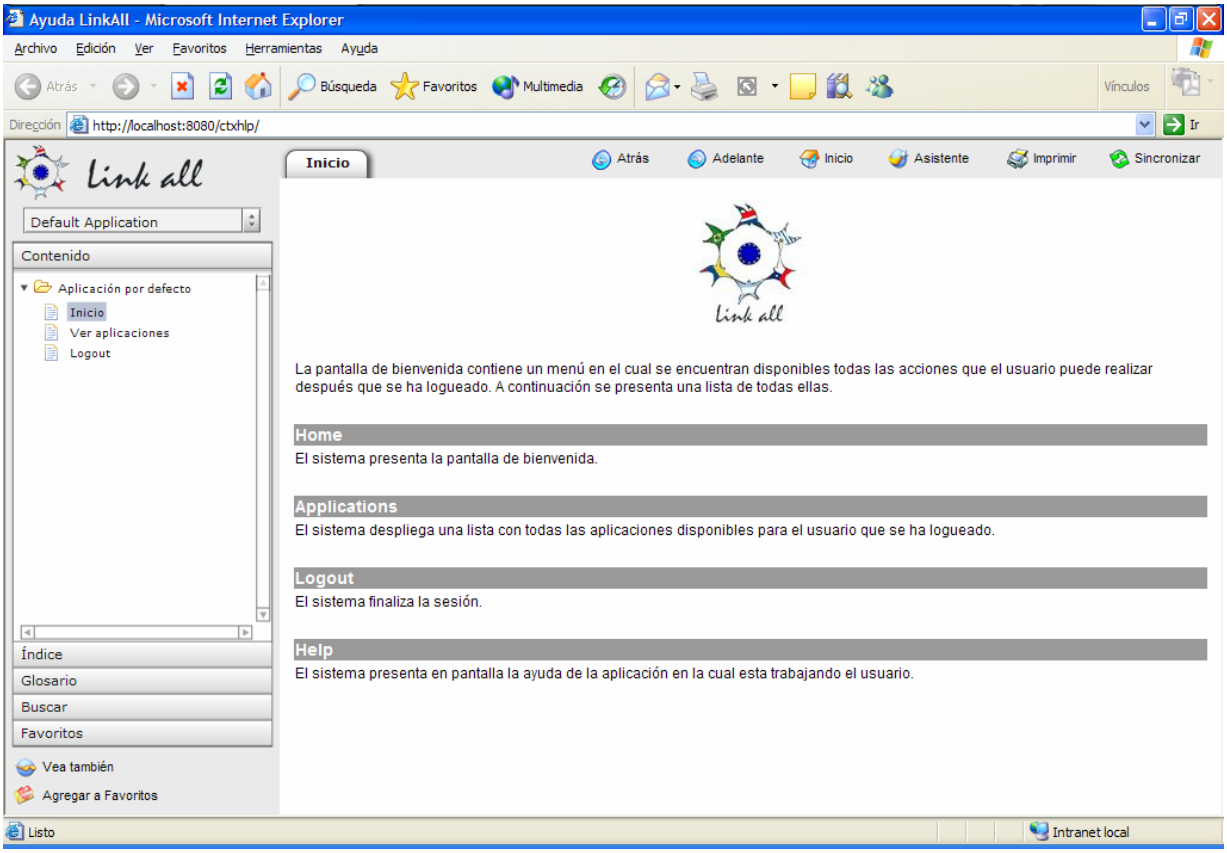


Figura 3

2.3.1 Contenidos

La principal funcionalidad que MACSIF brinda al usuario de una aplicación es la visualización de contenidos de ayuda.

Para MACSIF un contenido ***(content)*** es información que se puede visualizar a través de un explorador Web. Ejemplos de contenidos son una página Html, un archivo "pdf", un documento Word, una película Flash, una imagen, o cualquier elemento que genere dinámicamente alguno de los contenidos mencionados anteriormente.

2.3.2 Tabla de contenidos

La tabla de contenido ***(content table)*** organiza los contenidos en una estructura de árbol. Está formada por un conjunto de nodos que pueden tener asociados un contenido. La tabla de contenido brinda una forma sencilla al usuario de navegar a través de los contenidos de ayuda.

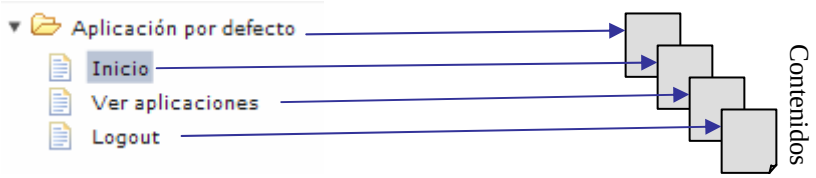


Figura 4

2.3.3 Índice

El índice ***(index)*** es un conjunto de palabras claves, cada una de las cuales tiene un conjunto de contenidos asociados. Al igual que la tabla de contenido, el índice brinda al usuario una forma sencilla de navegar a través de los contenidos de ayuda, pero en este caso agrupados de acuerdo a sus palabras clave.

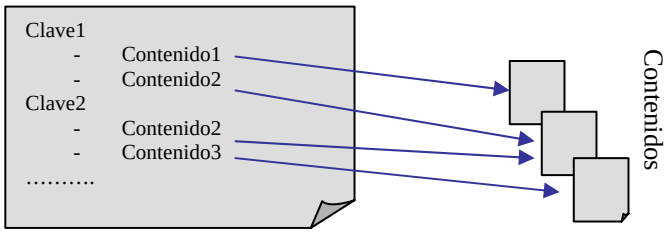


Figura 5

2.3.4 Glosario

El glosario ***(glossary)*** es un conjunto de parejas término – definición. Dado un término el usuario puede consultar la definición del mismo.

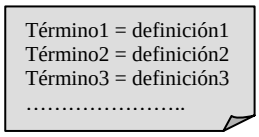


Figura 6

2.3.5 Búsqueda de contenidos

A través de un mecanismo de búsqueda de texto que se debe proveer a MACSIF, un usuario puede realizar búsquedas de contenidos en la ayuda de la aplicación actual o en la ayuda de otras aplicaciones, como se muestra en la figura 7.

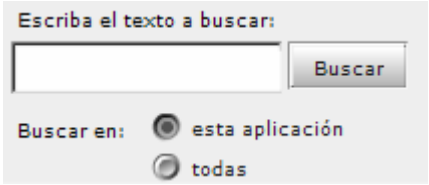


Figura 7

2.3.6 Contenidos Favoritos

Los contenidos favoritos ***(favorites)*** son un conjunto de contenidos a los cuales el usuario ha decidido guardar referencia, para poder acceder a ellos rápidamente más tarde. MACSIF permite consultar, agregar o eliminar contenidos favoritos.

2.4 Utilitarios del framework

2.4.1 Proveedores de ayuda (*Help Providers*)

Un *Help Provider* se encarga de proporcionar los elementos que componen la ayuda de una aplicación en un determinado idioma (contenidos, índice, glosario, etc.). Un *Help Provider* puede suministrar ayuda a varias aplicaciones en varios idiomas, a su vez, la ayuda de una aplicación en un determinado idioma podría estar disponible en varios *Help Providers*.

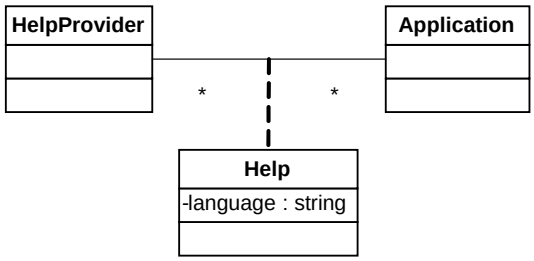


Figura 8

MACSIF permite especificar:

- el conjunto de *Help Providers* con los que trabajará
- el orden en que se buscará ayuda en dichos providers, definiendo una prioridad
- el *Help Provider* por defecto de una aplicación

Es importante destacar que un *Help Provider* no está asociado a un SIFW específico, por lo que varios SIFW que utilicen MACSIF podrían hacer uso del mismo *Help Provider* para obtener las ayudas de sus aplicaciones.

2.4.2 Proveedores de búsqueda (*Search Providers*)

Un Search Provider tiene como función la búsqueda de texto en los contenidos de las ayudas. Al igual que los *Help Providers*, un *Search Provider* no está asociado a un SIFW específico, por lo que varios SIFW que utilicen MACSIF podrían utilizar el mismo *Search Provider*.

2.5 Adaptabilidad a un SIFW específico

2.5.1 Identificador de contexto – Mapeos (*contextId – mappings*)

Para poder brindar ayuda de acuerdo a la actividad que está realizando el usuario es necesario tener una correspondencia entre los contenidos de las ayudas y las posibles actividades dentro de una aplicación. Esta correspondencia servirá, además, para identificar qué contenidos de ayuda se pueden mostrar a un usuario de acuerdo a sus permisos. Como forma de generalizar esta idea, MACSIF define el concepto de ***contextId***, que en su forma más sencilla podría ser el nombre de una actividad.

Entonces, se define mapeos ***(mappings)*** como el conjunto de correspondencias entre *contextId* y contenidos de ayuda de una aplicación. Un *contextId* puede tener asociado varios contenidos, pero uno de ellos será el contenido primario, que se mostrará inicialmente cuando se solicita ayuda sobre determinada operación. Como se ve en la figura 9 este conjunto de correspondencias tendrá que ser definido para cada *Help Provider*.

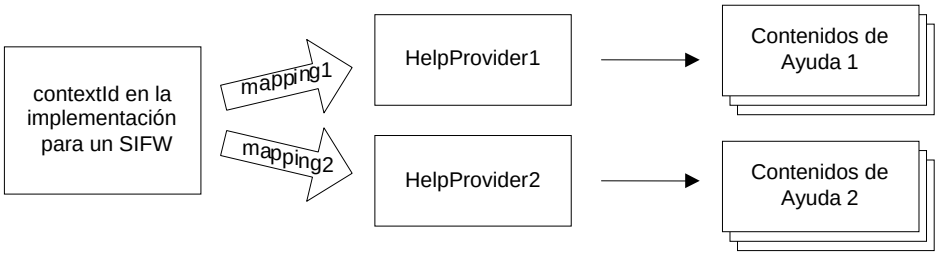


Figura 9

MACSIF permite definir la forma del *contextId*, o sea, especificar para un SIFW específico qué cosas determinan el contenido a presentar. En la figura 10 se presentan ejemplos de esto.

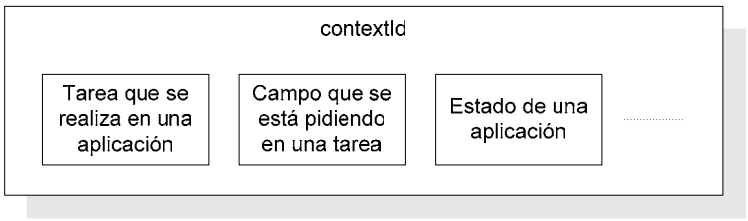


Figura 10

2.5.2 Obtención del contexto

Como se plateó anteriormente, el contexto de ejecución está formado por dos elementos principales:

- la actividad actual
- el perfil del usuario

Dentro de la actividad actual se distinguen:

- la aplicación que está ejecutando el usuario
- el lenguaje elegido por éste
- el *contextId* actual (llamado *currentContextId*), que como se explicó anteriormente determina el contenido de ayuda a mostrar.

Por otro lado, el perfil del usuario determina:

- a qué información puede acceder el mismo, por ejemplo los *contextId*
- las aplicaciones a las que tiene acceso

MACSIF permite definir la forma en que se obtiene esta información ya que es muy factible que varíe de un SIFW a otro.

3 DISEÑO Y ARQUITECTURA

3.1 Introducción

MACSIF está formado por un conjunto de componentes, donde entre otras cosas se encuentra la lógica e interfaz de usuario del framework, y de un conjunto de interfaces que son la forma de comunicación de MACSIF con los utilitarios y con el SIFW específico que hace uso del framework.

En la figura 11 se puede observar que los componentes que forman una determinada implementación de MACSIF se pueden dividir en tres grandes grupos:

- componentes de MACSIF
- componentes específicos del SIFW
- componentes utilitarios

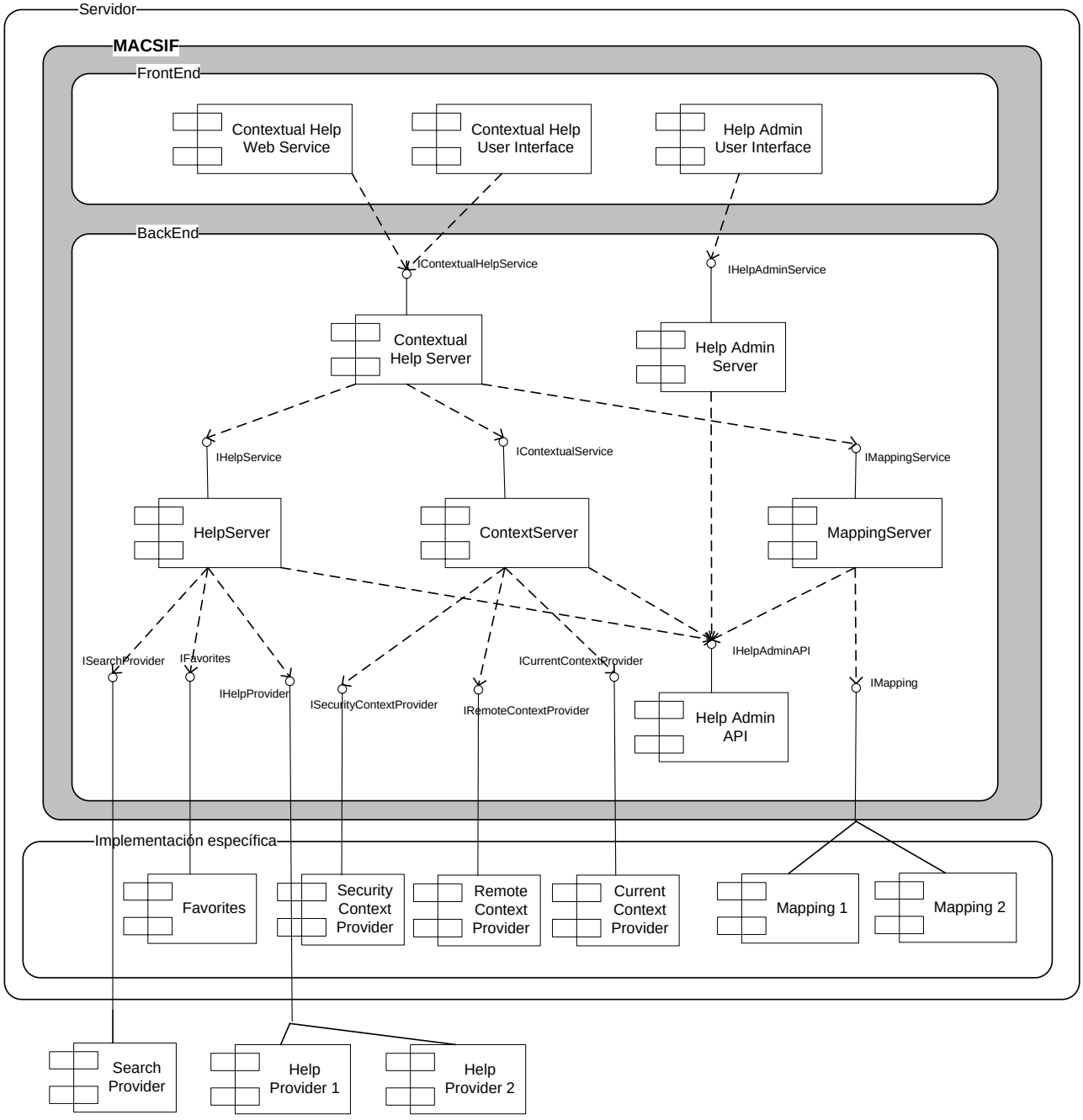


Figura 11

3.2 Componentes de MACSIF

Los componentes de MACSIF están divididos en dos grandes partes, el Backend y el Frontend. En el Backend se encuentran los componentes que contienen la lógica de negocio, como también componentes de acceso a datos de configuración. En el Frontend se encuentran los componentes encargados de atender las solicitudes de los usuarios así como también solicitudes de otros sistemas.

3.2.1 ContextualHelpUI

Es el encargado de interactuar con los usuarios de las aplicaciones, mostrando la ayuda contextual cuando el usuario la requiere, y brindando formas de navegación a través de ella. Implementa el patrón MVC (Model-View-Controller [11], ya que sus principales clases son:

- un Servlet que recibe los pedidos desde el navegador del cliente
- una serie de manejadores (llamados "Handlers") que derivan la solicitud a la capa lógica
- una serie de páginas Html que se devuelven al usuario

Estas páginas Html contienen películas Flash desarrolladas mediante la tecnología OpenLaszlo[1]. La forma de comunicación entre los *Handlers* y los objetos Flash es a través de las variables de sesión, es decir, los *Handlers* graban el resultado de las operaciones como variables de sesión en el Servidor Web, las que son accedidas por las JSP para formar dinámicamente el contenido de los objetos Flash.

Otra característica importante de este componente es que implementa la internacionalización de todos los mensajes que se muestran al usuario (inclusive etiquetas y botones), de forma que todos los textos que el usuario verá estarán expresados en el idioma que haya seleccionado.

3.2.2 HelpAdminUserInterface

Es el encargado de interactuar con los usuarios administradores. A partir de esta interfaz de usuario es posible configurar todos los parámetros que definen el comportamiento de MACSIF, como por ejemplo los proveedores de ayuda con los que se trabajará, los proveedores por defecto de las aplicaciones, etc.

3.2.3 ContextualHelpWebService

Expone los métodos del *ContextualHelpServer* a través de un Web Service. Esto resulta conveniente cuando la comunicación a través de RMI no es posible, o cuando se quiere interactuar con el *ContextualHelpServer* utilizando componentes que no estén implementados en Java.

3.2.4 ContextualHelpServer

Este es el principal componente de MACSIF, en él se encuentra la mayor parte de la lógica de negocio. El *ContextualHelpServer* recibe pedidos del componente *ContextualHelpUserInterface*, los procesa y responde apoyándose en otros componentes. Está implementado como un Enterprise Java Bean de tipo "Sessionless" con el objetivo de que el servidor Web que gestiona la interfaz de usuario pueda estar corriendo en otro servidor diferente al que está corriendo el contenedor J2EE [10] en el cual se ubica el Sistema de Ayuda.

Cuando el *ContextualHelpServer* recibe un pedido de ayuda, utiliza el *ContextServer* para obtener los datos del contexto de ejecución:

- aplicación
- lenguaje
- *contextId* actual
- aplicaciones
- *contextId* a las que tiene permiso para acceder el usuario

Con la aplicación y el lenguaje conocidos, solicita al *HelpServer* ayuda para dicha aplicación en ese lenguaje. Finalmente, el *ContextualHelpServer* modifica la ayuda obtenida eliminando los contenidos sobre los cuales el usuario no tiene permiso, y seleccionando el contenido que se corresponde con el *currentContextId*.

Implementa la interfaz *IContextualHelpService*, y en la figura 12 se pueden observar las operaciones que provee dicha interfaz, la cual es la cara visible de todo el Sistema.

[1] En el Apéndice A se presenta una descripción más completa de esta tecnología

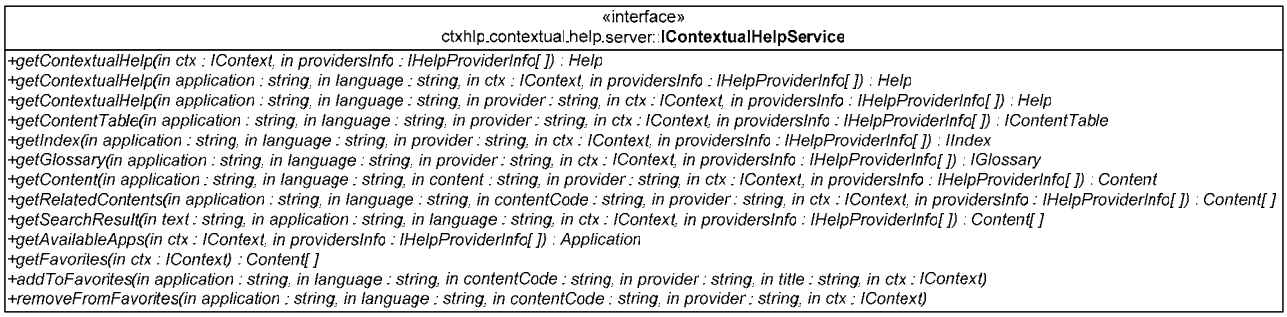


Figura 12

3.2.5 HelpServer

Es responsable de encontrar ayuda para una aplicación en un determinado idioma. El *HelpServer* recibe pedidos del *ContextualHelpServer*, que le envía la aplicación y el lenguaje de la ayuda que desea obtener. Debe entonces pedir al *HelpAdminAPI* la lista de proveedores de ayuda con los que se está trabajando y buscar en cada uno de ellos la ayuda de la aplicación en el idioma especificado. Si la aplicación tiene un proveedor por defecto, lo que también se le debe consultar al *HelpAdminAPI*, éste es el primer *HelpProvider* que se consultará.

Implementa la interfaz *IHelpService*, y en la figura 13 se pueden observar las operaciones que provee dicha interfaz.

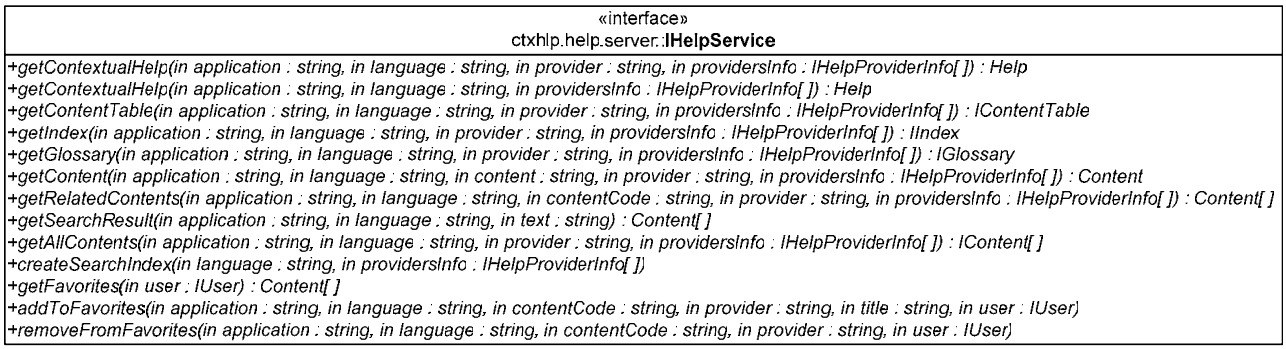


Figura 13

3.2.6 ContextServer

Es el encargado de proporcionar toda la información relativa al contexto de ejecución. Es decir, cuando se solicita ayuda, para que ésta sea contextual, deben obtenerse todas las variables del contexto que sean útiles para definir qué ayuda se debe mostrar. Aquí pueden incidir variables tales como:

- qué aplicación se está ejecutando
- qué operación dentro de ésta
- cuál es el idioma del usuario
- sus permisos, etc.

Para obtener estos datos, se utilizan los servicios de los componentes externos que brindan esta información relativa a la plataforma en la que se está ejecutando la ayuda. Estos componentes externos al Sistema deben implementar tres interfaces para que el *ContextServer* pueda accederlas: *ICurrentContextProvider* (que se encarga de obtener el contexto de ejecución), *ISecurityContextProvider* (que se encarga de obtener los permisos de ejecución del usuario), e *IRemoteContextProvider* (que se encarga de determinar si la aplicación se ejecuta en un nodo que no es el local, así como de obtener el contexto de ejecución remoto en caso de que sea así).

Implementa la interfaz *IContextService*, y en la figura 14 se pueden observar las operaciones que provee dicha interfaz.

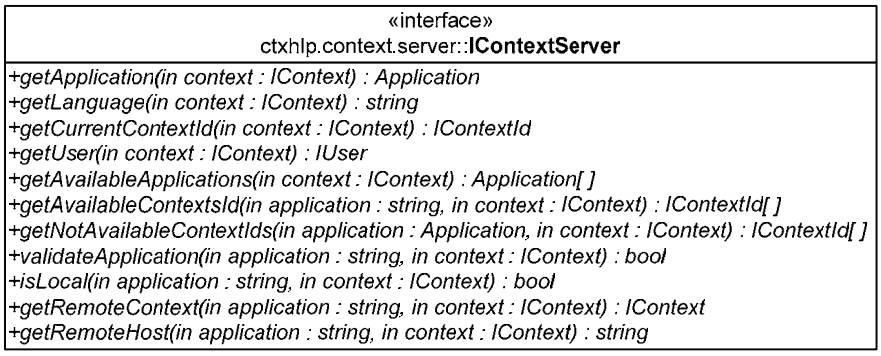


Figura 14

3.2.7 MappingServer

Se encarga de obtener la información relativa a los mapeos entre los *contextId* y los contenidos de ayuda para cada *Help Provider*. Es decir, para un mismo identificador de contexto, puede existir más de un contenido de ayuda, según qué proveedor maneje esta información. Por lo tanto, cuando se incorpora un *HelpProvider* se debe generar el componente de mapeos correspondiente implementando la interfaz *IMapping*.

Esta información es manejada por el *MappingServer*, el cual obtiene del *HelpAdminAPI* la información de qué proveedores existen y qué componentes gestionan los mapeos con cada uno de ellos. En la figura 15 se presenta el diagrama de la interfaz que expone este componente.

Figura 15

3.2.8 HelpAdminAPI

Se encarga de guardar y recuperar toda la información del entorno de ejecución de MACSIF. Este componente brinda operaciones tanto para guardar información (que será utilizada por el Administrador de la plataforma), como para recuperarla (que será usada por los subsistemas *HelpServer*, *ContextServer* y *MappingServer*, como se explicó anteriormente). Los datos que se graban a través de este componente pueden ser:

- el idioma por defecto,
- los proveedores de ayuda y de búsqueda con que se cuenta,
- quién implementa los servicios correspondientes a la plataforma, etc.

En la figura 16 se presenta el diagrama de la interfaz que expone este componente.

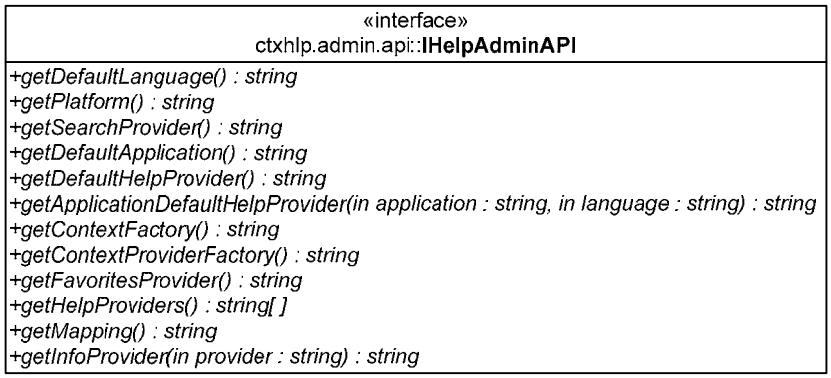


Figura 16

3.3 Componentes específicos del SIFW

Estos componentes son los que permiten adaptar a MACSIF a las características particulares de un determinado SIFW.

3.3.1 CurrentContextProvider

Es el encargado de obtener toda la información del contexto de ejecución relativa a la actividad actual del usuario. Como en el caso anterior, se obtiene información de la sesión a través del objeto que implementa la interfaz *IContext*. Un *CurrentContextProvider* debe implementar la interfaz *ICurrentContextProvider* provista por MACSIF. En la figura 17 se pueden observar las operaciones que provee dicha interfaz.

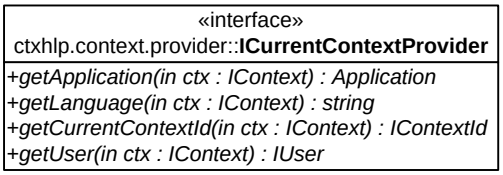


Figura 17

3.3.2 SecurityContextProvider

Es el encargado de obtener toda la información del contexto de ejecución relativa a los permisos del usuario, lo que determina los recursos a los que puede acceder. Para poder obtener dicha información estos componentes necesitan cierta información proveniente en general desde la interfaz de usuario. Esta información viene incluida en un objeto que implementa la interfaz *IContext*. Un *SecurityContextProvider* debe implementar la interfaz *ISecurityContextProvider* provista por MACSIF. En la figura 18 se pueden observar las operaciones que provee dicha interfaz.

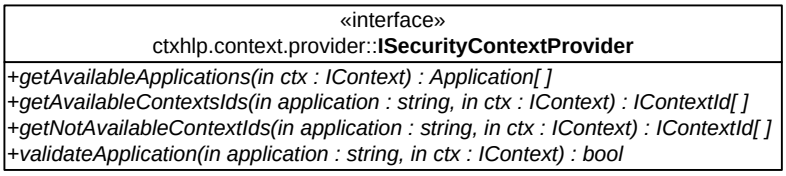


Figura 18

3.3.3 RemoteContextProvider

Es el encargado de determinar si una aplicación se ejecuta en un nodo que no es el local, así como determinar el contexto remoto en caso de que así sea. Como en los casos anteriores, se obtiene información de la sesión a través del objeto que implementa la interfaz *IContext*. Un *RemoteContextProvider* debe implementar la interfaz *IRemoteContextProvider* provista por MACSIF. En la figura 19 se pueden observar las operaciones que provee dicha interfaz.

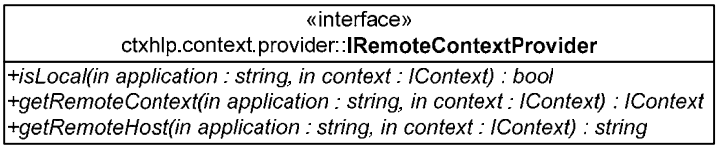


Figura 19

3.3.4 Mapping

Como se planteó anteriormente, los mapeos son un conjunto de correspondencias entre *contextId* y contenidos de ayuda. Debido a que el *contextId* es específico del SIFW en cuestión, los mapeos también lo son. Además los mapeos hacen referencia a los contenidos de un determinado *HelpProvider*, por lo que se deberá definir un componente *Mapping* para cada uno de los *HelpProviders* con los que trabaje MACSIF.

Cada componente *Mapping* debe implementar la interfaz *IMapping* definida por MACSIF, y en la figura 20 se pueden observar las operaciones que provee dicha interfaz.

Figura 20

3.3.5 Favorites

Para poder almacenar los contenidos favoritos de un usuario, es necesario identificar al mismo de alguna forma. Esta forma de identificación será particular del SIFW en cuestión, por esto MACSIF define la interfaz *IUserId*. El componente que implemente esta interfaz debe incluir los elementos que identifican a un usuario en ese SIFW.

Este componente debe implementar la interfaz *IFavorites* definida por MACSIF, y en la figura 21 se pueden observar las operaciones que provee dicha interfaz.

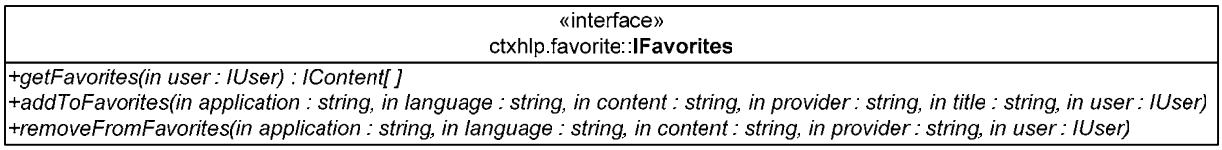


Figura 21

3.4 Componentes Utilitarios

3.4.1 *HelpProviders*

Los *HelpProviders* son los encargados de proveer la información que conforma la ayuda de una aplicación. Todo *HelpProvider* debe implementar la interfaz *IHelpProvider* definida por MACSIF. Esta interfaz debe ser implementada por los proveedores de ayuda que se incorporan a la plataforma, y asegura la comunicación del Sistema con ellos a través de operaciones básicas, las que internamente resolverán de acuerdo a sus propias características y diseños. En la figura 22 se pueden observar las operaciones que provee dicha interfaz.

Figura 22

3.4.2 *SearchProviders*

Los *SearchProviders* son componentes encargados de realizar búsquedas de texto en los contenidos de ayuda que proveen los *HelpProviders*. Por esto un *SearchProvider* debe consultar a los *HelpProviders* para obtener información de los contenidos y de esta forma poder responder a los pedidos de búsqueda, por ejemplo creando sus propios índices de búsqueda en base a dicha información.

Cada SearchProvider debe implementar la interfaz *ISearchProvider* definida por MACSIF, y en la figura 23 se pueden observar las operaciones que provee dicha interfaz.

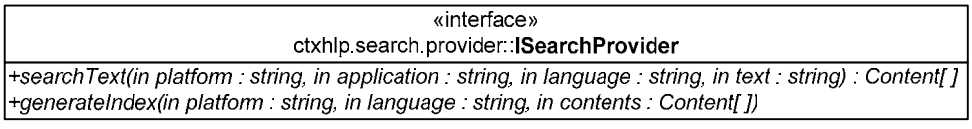


Figura 23

4 UNA APLICACIÓN DEL FRAMEWORK: LINK-ALL

Como se mencionó en el punto 2, uno de los principales objetivos de MACSIF es brindar un mecanismo de ayuda que se pueda adaptar a cualquier SIFW. Link-All es uno de estos Sistemas de Información Federados (en particular, éste es el contexto del proyecto), por lo que debemos brindar una implementación del Framework para el mismo.

Link-All [9] es una plataforma para la ejecución integrada de aplicaciones que comparten servicios e información. Estos servicios son brindados a través de componentes públicos que exponen funcionalidades a las aplicaciones. Nuestra implementación de MACSIF utilizará estos servicios, como se verá más adelante en este capítulo. También se mostrarán los utilitarios desarrollados en el contexto de este caso de estudio, como los HelpProviders y utilitarios de búsqueda de texto.

4.1 Descripción general

Se describirán las funcionalidades brindadas por la plataforma Link-All, y la forma en que fueron utilizadas esas funcionalidades para la integración tanto de MACSIF como de los diferentes utilitarios desarrollados.

4.1.1 Servicios brindados por Link-All

Si bien esta plataforma brinda gran cantidad de servicios que pueden ser útiles a las aplicaciones que se están ejecutando, esta implementación de MACSIF utilizará solamente parte de esos servicios, relativos a datos de una sesión, seguridad e información respecto de las aplicaciones y funcionalidades.

- ***SessionServer***
  Este componente se encarga de la gestión de los usuarios y sus perfiles, tanto como del registro de las tareas que ellos efectúan en las distintas aplicaciones. Para esto, dispone de un "log" en el cual se registran las operaciones que el usuario realiza dentro de la plataforma (login, logout, ingreso a una aplicación), así como las que efectúa dentro de las aplicaciones (en la medida que las aplicaciones utilicen esta facilidad). Este log se registra según un identificador de sesión interno de Link-All, de modo que todas las operaciones realizadas en el contexto de una sesión son registradas, permitiéndose obtener tanto el usuario como sus acciones dentro de la plataforma.
  También se puede obtener de la sesión todos los valores que forman el perfil del usuario, como ser su idioma, ubicación, etc. Para esto, cada vez que un usuario realiza un login a la plataforma, los datos de su perfil se copian al perfil de la sesión. De esta forma, el usuario puede modificar el perfil de la sesión sin cambiar la información de su propio perfil.

- ***SecurityServer***
  Se encarga de la gestión de los permisos de acceso a las aplicaciones, y dentro de ellas a las funcionalidades. Los permisos son manejados a nivel de grupo de usuarios, lo que permite una gestión más ordenada que si se hiciera a nivel de usuario. Es posible obtener una lista de las aplicaciones y funcionalidades a las que el usuario puede acceder.

- ***ApplicationFederator***
  Su tarea es la gestión de las aplicaciones que se ejecutan en los distintos nodos de la plataforma Link-All. Además, guarda información respecto a todas las aplicaciones, incluyendo descripción, versión, etc.

- ***ConfigurationServer***
  Guarda información de configuración de Link-All, como ser qué aplicaciones están instaladas, qué funcionalidades, parámetros, etc.

4.1.2 Implementación de MACSIF para Link-All

A continuación se mostrarán las funcionalidades brindadas por Link-All a través de los componentes vistos en el punto anterior, y la forma en que se utilizaron para la implementación de MACSIF para esta plataforma.

- ***Contexto***
  Como se mencionó en el punto 2.1, MACSIF define el concepto de contexto de una sesión, el que determina una serie de variables que utiliza para obtener la ayuda a mostrar. En la implementación para Link-All, el contexto está determinado por el número de sesión interna, el cual es obtenido a través de una cookie grabada por esta plataforma.

- ***Actividad actual***

Una vez determinado el identificador de sesión, se está en condiciones de utilizar los servicios brindados por el *SessionServer* para obtener los datos que forman el contexto actual de ejecución. En esta implementación, el contexto de ejecución (*contextId*) está dado por el nombre de la funcionalidad de la aplicación que se está ejecutando y por la operación registrada (por ejemplo "ingreso a la funcionalidad", "error en la funcionalidad", etc.). Este *contextId* (en este caso funcionalidad + operación) determinará el contenido de ayuda a mostrar de acuerdo a los mapeos definidos en el punto 2.4.1. De esta forma, se obtendrán ayudas diferentes, si se solicita ayuda trabajando normalmente en una funcionalidad y si ocurre un error trabajando con la misma.

Otro dato que se obtiene del *SessionServer* a través del identificador de sesión es su perfil, el cual nos indicará el lenguaje del usuario. Este lenguaje será utilizado tanto para mostrar todos los textos fijos, como para buscar la ayuda. En caso que no se encuentre la ayuda específica en el lenguaje del usuario, se tomará un lenguaje por defecto.

Por lo tanto, los datos que se obtienen de la plataforma son:

- el perfil del usuario (a través de las operaciones *getSessionInfo()* y *getSessionProfiles()*)
- la aplicación y funcionalidad que está ejecutando (a través de la operación *getEvents()*)

A partir del perfil del usuario se determinará el lenguaje en el que mostrará la ayuda, y la funcionalidad determinará un contenido de ayuda inicial a partir de los mapeos funcionalidad - contenido.

- ***Seguridad***
  A efectos de que MACSIF muestre solamente la información que el usuario puede acceder de acuerdo a los permisos de su grupo, se utilizaron las siguientes operaciones provistas por el *SecurityServer*:
  - *getLoginApplication()* para obtener las aplicaciones accesibles
  - *getLoginFunctionalities()*
  - *getFunctionalities()* para poder buscar los contenidos que se pueden mostrar, que son:
    - los contenidos asociados a funcionalidades sobre las que tiene permisos
    - los contenidos no asociados a ninguna funcionalidad

4.1.3 Proveedores de ayuda

Como se vio en el punto 2.4, para que la implementación de MACSIF para Link-All funcione correctamente, es necesario incorporar una serie de utilitarios, como son los proveedores de ayuda. Si bien éstos son independientes del SIFW, fue necesario implementar estos servicios para que provean las funcionalidades que MACSIF requiere.

Se implementaron dos proveedores de ayuda, los que se describen en próximos puntos de este capítulo. El objetivo que se buscó es que se disponga de un proveedor de ayuda principal, en el que se guarda toda la información que MACSIF necesita en un medio de almacenamiento que permita manejar grandes volúmenes de datos con tiempos de respuesta cortos. Complementariamente, se buscó dar siempre una respuesta al usuario, ya que si la pareja (funcionalidad, operación) sobre la que se solicitó ayuda no está disponible, se retorna una ayuda armada en base a la información disponible en Link-All. Para esto se creó el segundo proveedor de ayuda, el cual utiliza los servicios de la plataforma para brindar una información general sobre la funcionalidad que se está usando.

Además de estos proveedores de ayuda para todas las aplicaciones de la plataforma, se creó un proveedor independiente que será utilizado para el Asistente (*Wizard*). Considerando el hecho de que MACSIF permite invocar la ayuda de un proveedor directamente (sin pasar por los otros proveedores definidos), se creó éste para brindar sugerencias o guías que faciliten la navegación a aquellos usuarios que lo deseen.

4.1.3.1 RelationalHelpProvider

Este proveedor de ayuda brinda a MACSIF todas las operaciones de ayuda que requiere obteniendo los datos de una base de datos relacional. Para encapsular el acceso a los datos, el cual depende del manejador (DBMS) que se esté utilizando, implementa los patrones Value Objects (VO) y Data Access Object (DAO) [11]. Además, estos patrones están implementados en un componente externo (un EJB llamado *HelpAPI*), lo que permite que la base de datos pueda estar ubicada en otro servidor, e incluso que la misma base de datos pueda ser utilizada por otros proveedores o por aplicaciones externas a Link-All.

En la figura 24 se muestra el modelo de datos utilizado para almacenar la información que necesita, el cual se explicará con más detalle.

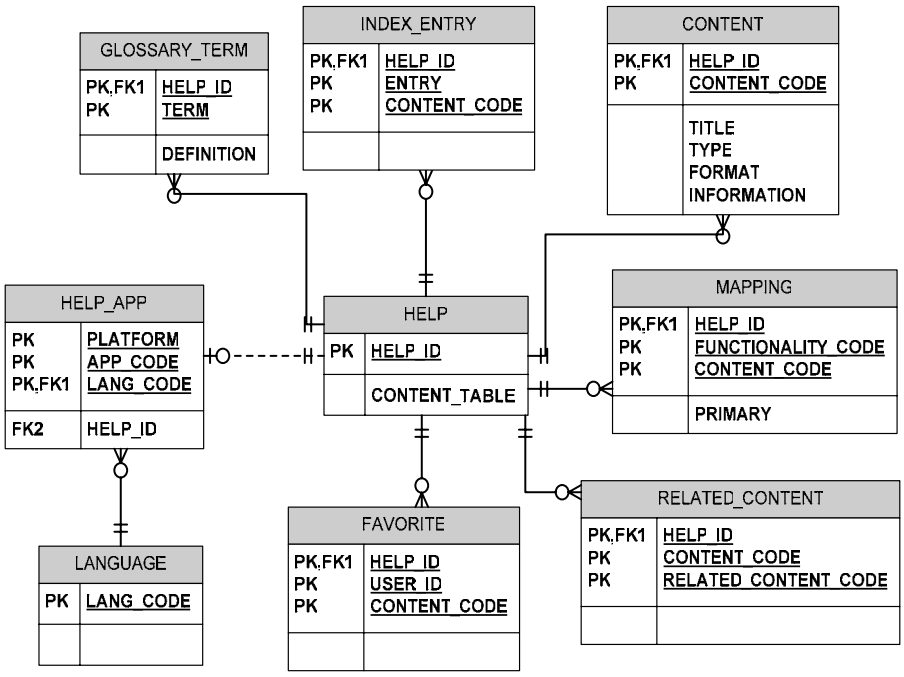


Figura 24

Cuando se solicita ayuda a través de MACSIF, se hace para una determinada aplicación y un determinado lenguaje, ya que éstos son los datos que maneja (independientemente de cómo los representa cada *HelpProvider*). Para esto, el *RelationalHelpProvider* posee la tabla *HELP_APP*, la cual contiene todas las parejas (aplicación, lenguaje) de las que se tienen datos. Si la (aplicación, lenguaje) que se solicita no está en la tabla *HELP_APP* significa que este proveedor no posee ayuda para esa aplicación en dicho lenguaje. MACSIF decidirá qué hacer en ese caso (puede buscar ayuda en otro idioma o puede dirigirse a otro proveedor).

En el caso de que se encuentre la ayuda en esa tabla se obtendrá un *Help_Id*, el cual es un identificador único del conjunto de información que forma una ayuda. Este identificador determina una única tupla en la tabla *HELP*, la que contiene además la tabla de contenido de dicha ayuda en el atributo *Content_Table*. Todas las otras tablas que contienen datos de ayuda están identificadas por el atributo *Help_Id* y poseen una restricción de integridad (Foreign Key) con la tabla *HELP*. Estas otras tablas son las siguientes:

- **Content**
  Almacena los contenidos correspondientes a una ayuda. Cada contenido está identificado por un código (*Content_Code*) que debe ser único dentro de la ayuda. Además tiene un título (*Title*) que se muestra al relacionarlo con las entradas de índice y en los resultados de búsquedas de texto, y puede estar en distintos formatos (texto, Html, Flash, Word, PDF, etc.) lo que se indica en el atributo *Format*. El atributo *Type* está relacionado con la operación sobre la que se pide ayuda, y los valores que puede tener corresponden a "ayuda", "error", etc. Por último, el atributo *Information* contiene la ayuda a mostrar o la URL del archivo que la contiene.

- **Related_content**
  Si un contenido posee algún contenido relacionado, se determinan en esta tabla. La vinculación se da a través del atributo *Content_Code*.

- **Index_entry**
  Contiene todas las palabras clave de la ayuda, así como todos los códigos de los contenidos asociados a ellas. Cada palabra clave tendrá una tupla por cada contenido asociado a ella, por lo que un mismo *Content_Code* puede estar en varias tuplas correspondientes a diferentes palabras clave.

- **Glossary_term**
  Contiene todos los términos definidos en el glosario, junto con una definición de cada uno de ellos.

4.1.3.2 LinkAllHelpProvider

Este proveedor de ayuda se utiliza cuando no se encontró una ayuda específica para la funcionalidad y operación actual, ni en el lenguaje del usuario ni en el lenguaje por defecto. Su objetivo es brindar, con los datos que puede obtener de la plataforma, una ayuda básica acerca de la funcionalidad que está utilizando y cómo ésta se vincula con otras funcionalidades de la aplicación.

A continuación se muestra cómo el *LinkAllHelpProvider* genera los objetos que forman la ayuda de una aplicación, utilizando los servicios provistos por el *ApplicationFederator* y por el *SecurityServer* de la plataforma:

- **Tabla de contenidos**
  La raíz contiene el nombre de la aplicación, y las hojas contienen los nombres de las funcionalidades de dicha aplicación.

- **Contenidos**
  Los contenidos que se muestran se generan dinámicamente. Cuando se selecciona la raíz de la tabla de contenido (correspondiente a la aplicación), el contenido muestra el nombre y la descripción de la misma, así como su número de versión y la fecha de instalación en la plataforma, además de una lista de las funcionalidades que forman la aplicación. Cuando se selecciona una hoja (funcionalidad) se muestra el nombre y la descripción de la misma.

- **Índice:**
  Las entradas de índice corresponden a los nombres de las funcionalidades de la aplicación, y como único contenido asociado tienen el mismo contenido que se arma cuando se selecciona dicha funcionalidad de la tabla de contenido.

- **Glosario**
  Los términos del glosario corresponden a los nombres de las funcionalidades de la aplicación, y como definición se muestra la descripción de las mismas.

4.1.3.3 WizardHelpProvider

Este proveedor de ayuda es utilizado como Asistente, o cuando los otros proveedores definidos no encontraron la ayuda buscada. Su objetivo es brindar guías o sugerencias de acción al usuario, basándose en una serie de heurísticas o acciones frecuentes de otros usuarios, las que pueden ser muy variadas. El hecho de que el Asistente sea un *HelpProvider* más, brinda la posibilidad de que sea muy fácil agregar o modificar heurísticas, sin tener que modificar MACSIF para eso.

4.1.4 Mapeos de los proveedores de ayuda con los contenidos

Como se explicó en el punto 2.5.1, la implementación de MACSIF para Link-All necesita realizar un mapeo o correspondencia de las parejas (funcionalidad, operación) que determinan la actividad actual para una sesión Link-All con los contenidos de ayuda, para cada *HelpProvider*.

En el caso del *RelationalHelpProvider*, esta correspondencia es guardada en la misma Base de Datos que se guarda toda la información de la ayuda, en una tabla llamada *mapping* (como se puede ver en la figura 24). Sin embargo, conceptualmente estos mapeos son independientes de la ayuda en sí, por lo que se manejan a través de un componente distinto del utilizado para el acceso a los datos de la ayuda, llamada *MappingAPI*. De esta forma, si en el futuro no se desea guardar los mapeos en la misma Base de Datos que los datos de la ayuda, se puede hacer simplemente modificando el componente *MappingAPI*.

En el caso de los mapeos con el *LinkAllHelpProvider*, esta correspondencia es trivial, debido a que los contenidos no están guardados sino que son dinámicos. Esto hace que a cada pareja (funcionalidad, operación) le corresponda un contenido armado en base a los datos de dicha pareja.

4.1.5 Proveedores de búsqueda de texto

Estos proveedores brindan funcionalidades de búsqueda de texto en los contenidos de las ayudas, tanto de una aplicación como de todas. Para esto, es posible crear un motor de búsqueda o utilizar uno ya existente. En este caso, se desarrolló un utilitario basado en una API de búsqueda de uso extendido.

4.1.5.1 LuceneSearchProvider

Este proveedor utiliza una API de búsqueda llamada Lucene. El funcionamiento de esta API se describe con más detalle en el Apéndice B, aunque podemos resumirlo diciendo que la búsqueda se realiza en un índice

compuesto por documentos, cada uno de ellos compuesto por parejas (campo, valor), sobre las cuales se puede realizar la búsqueda y de los cuales se puede obtener su valor. Esta idea se puede observar en la figura 25.

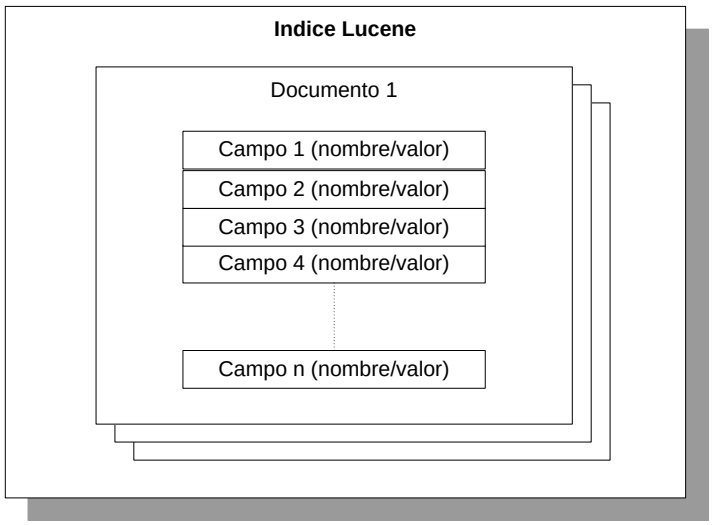


Figura 25

Antes de realizar una búsqueda, los índices deben ser construidos, por lo que este *SearchProvider* brinda funcionalidades tanto para crear el índice como para buscar en el mismo. Cuando se crean los documentos que formarán parte del índice, se especifican qué campos incluirá. En el caso de este proveedor, cada documento corresponde a un contenido de ayuda, y se definieron los siguientes campos para cada uno de ellos:

- Plataforma
- Proveedor de ayuda.
- Aplicación.
- Lenguaje.
- Código de contenido.
- Texto asociado al contenido.

La búsqueda se realiza en el campo de texto, obteniéndose un conjunto de documentos resultantes. Los otros campos de estos documentos indican qué contenidos se asocian con los resultados de la búsqueda y qué *HelpProvider* los provee, obteniendo así MACSIF todos los datos que necesita para mostrar la información.

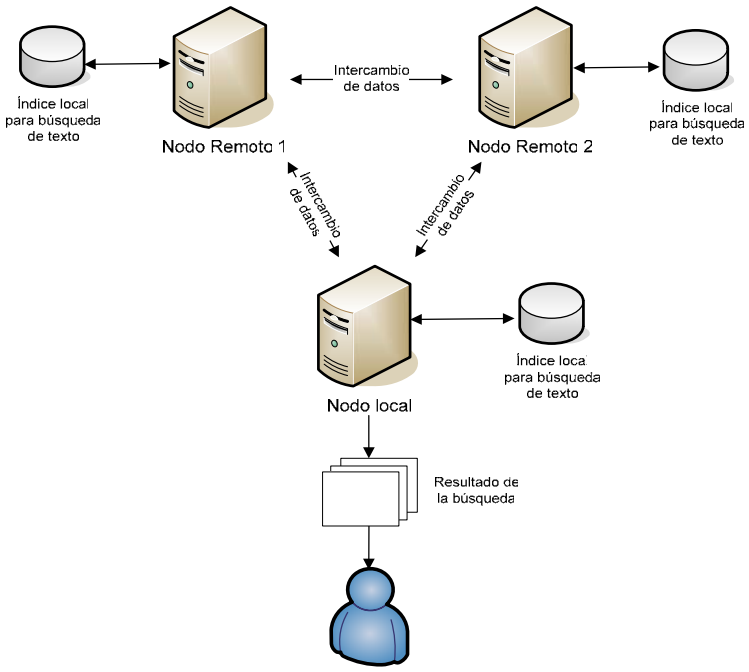


Figura 26

Como se puede ver en la figura 26, todos los nodos de la plataforma comparten entre sí la información de los contenidos de ayuda de cada *HelpProvider* que gestiona cada uno de ellos. De esta forma, todos los contenidos de ayuda de las aplicaciones locales al Nodo Remoto 1 estarán disponibles para búsquedas de texto en los otros dos nodos, y así sucesivamente. Cuando el usuario de una aplicación realiza una consulta de texto, el nodo en el cual está ejecutando la aplicación realiza una búsqueda en su índice (que incluye contenidos de ayuda correspondientes a aplicaciones locales y remotas) y devuelve el resultado. Debido a que esta consulta es local el resultado se devuelve en forma rápida, sin tener que consultar para ello a los otros nodos de la red, operación que sería costosa si se realiza por cada búsqueda de texto.

4.2 Diseño de la solución para Link-All

Como se vio en el capítulo 3, MACSIF provee una serie de interfaces que deben ser implementadas para un determinado SIFW y para los proveedores que se deseen utilizar. En este punto se mostrará cómo fue implementada cada interfaz para Link-All y para los proveedores desarrollados, tal como se ve en la figura 27.

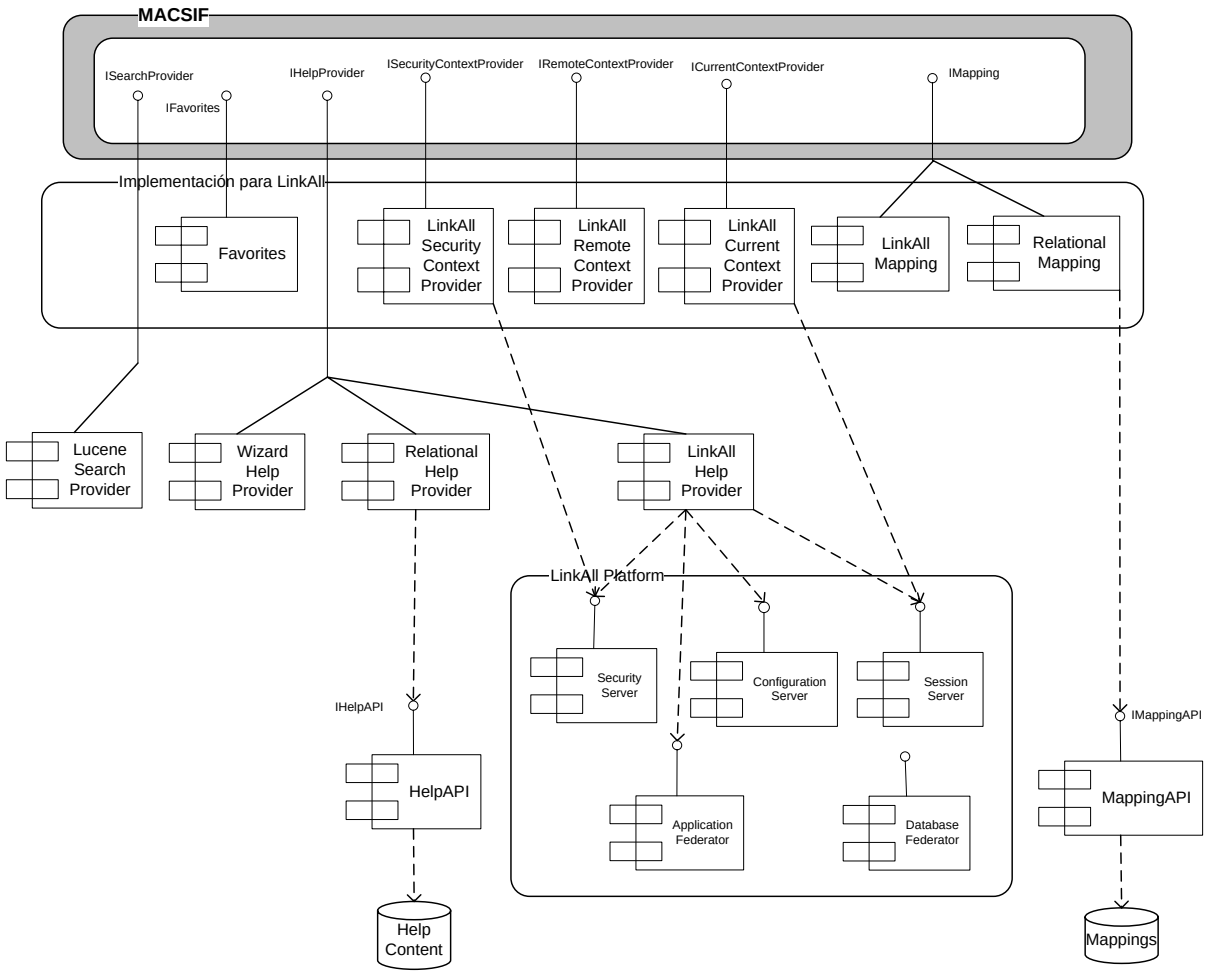


Figura 27

4.2.1 Componentes de la implementación para Link-All

A continuación se describen los componentes desarrollados para la implementación de las interfaces de MACSIF para la plataforma Link-All.

4.2.1.1 LinkAllCurrentContextProvider

Se encarga de implementar la interfaz *ICurrentContextProvider*, la cual brinda operaciones para determinar la actividad actual del usuario. Para esto, a partir del contexto formado por el identificador de sesión y utilizando los servicios del *SessionServer* de Link-All, se obtiene la aplicación que se está ejecutando, así como la funcionalidad y operación (los que determinan el contenido de ayuda a mostrarse), y el perfil de la sesión (lo que determina el lenguaje de la ayuda que se va a mostrar).

4.2.1.2 LinkAllSecurityContextProvider

Se encarga de implementar la interfaz *ISecurityContextProvider*, la cual brinda operaciones para determinar los permisos de un usuario. En el caso de Link-All, estos permisos están definidos en función de las funcionalidades a las que puede acceder de acuerdo a las definiciones del grupo al cual pertenece, información que provee Link-All a través de los servicios del *SecurityServer*.

4.2.1.3 LinkAllRemoteContextProvider

Se encarga de implementar la interfaz *IRemoteContextProvider*, la cual brinda operaciones para determinar si una aplicación se ejecuta en un nodo que no es el local, así como determinar el contexto remoto en caso de que así sea. En el caso de Link-All,

4.2.1.4 LinkAllMapping

Implementa la interfaz *IMapping* para el proveedor de ayuda *LinkAllHelpProvider*, por lo que provee los mapeos entre parejas (funcionalidad, operación) y los contenidos de ayuda asociados a éstas. En el caso de este proveedor, la correspondencia es trivial, ya que los códigos de contenidos coinciden con los códigos de funcionalidad, por lo que para cualquier operación se devuelve el código de la funcionalidad.

4.2.1.5 RelationalMapping

Es la otra implementación de la interfaz *IMapping* para la plataforma Link-All, en este caso para el proveedor de ayuda *RelationalHelpProvider*. En este caso, la correspondencia es más compleja que en el caso anterior ya que la misma está grabada en una Base de Datos, lo que hace que este componente deba encargarse de obtener los datos. Para esto, utiliza los servicios del componente *MappingAPI*, el cual se describe más abajo.

4.2.1.6 MappingAPI

Es un componente utilitario, utilizado por la *RelationalMapping*, cuya tarea es encapsular el acceso a la Base de Datos en la que se graba la información de mapeos de Link-All con la ayuda provista por el *RelationalHelpProvider*. Debido a que esta Base de Datos no tiene por qué estar en el mismo servidor que la plataforma Link-All, el componente es un EJB, y brinda la interfaz *IMappingAPI* para exponer sus servicios.

4.2.1.7 LinkAllFavorites

Se encarga de implementar las funcionalidades para la gestión de contenidos favoritos de un usuario, implementando para ello las interfaces *IFavorites* y *IUser* de MACSIF.

4.2.2 Componentes de los Utilitarios

A continuación se detallan los componentes desarrollados para cada uno de los utilitarios que forman esta aplicación de MACSIF a un caso concreto como es Link-All.

4.2.2.1 LinkAllHelpProvider

Es uno de los componentes desarrollados para proveer servicios de ayuda, a través de la implementación de la interfaz *IHelpProvider* de MACSIF. Como se mostró en el punto 4.1.3.2, la ayuda que brinda está formada por información que obtiene directamente de la plataforma Link-All.

4.2.2.2 RelationalHelpProvider

Es el proveedor de ayuda principal de la aplicación de MACSIF para Link-All, ya que contiene toda la información requerida por el Framework en una Base de Datos relacional. Para esto, utiliza los servicios del componente *HelpAPI*, el cual se describe más abajo. En el punto 4.1.3.1 se mostró el modelo de datos utilizado por este proveedor.

4.2.2.3 HelpAPI

Es un componente utilitario, utilizado por la *RelationalHelpProvider*, cuya tarea es encapsular el acceso a la Base de Datos en la que se graba la información de ayuda de Link-All y potencialmente de otras plataformas. Este componente es un EJB, y brinda la interfaz *IHelpAPI* para exponer sus servicios.

4.2.2.4 WizardHelpProvider

Fue desarrollado para proveer funcionalidades de Asistente, o sea, una ayuda extra que el usuario puede utilizar cuando no sabe bien qué es lo que desea hacer, o no está seguro de hacia donde ir. Implementa la interfaz *IHelpProvider* de MACSIF, y obtiene la información que muestra directamente de la plataforma Link-All, ya que se guarda un registro de todas las operaciones efectuadas por los usuarios.

4.2.2.5 LuceneSearchProvider

Es el único proveedor de búsquedas de texto implementado para la aplicación de MACSIF para Link-All. Este componente implementa la interfaz *ISearchProvider*, la que dispone de operaciones de búsqueda de texto y de creación del índice.

5 REFERENCIAS

Sistemas de Ayuda

[1] – WinHelp
http://www.knopf.com/resources/help/winhelp.html Junio, 2004.

[2] – Microsoft HTML Help
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/htmlhelp/html/vsconHH1Start.asp Junio, 2004.

[3] – JavaHelp
http://java.sun.com/products/javahelp/ Setiembre, 2004.
O'Reilly - Creating Effective Javahelp

[4] – FlashHelp
http://www.cherryleaf.com/news-wildfire.htm Julio, 2004.

[5] – OracleHelp, OHJ - OHW
http://www.oracle.com/technology/tech/java/help/index.html Setiembre, 2004.

Arquitecturas orientadas a servicios (SOA)

[6] – Página de Sun dedicada a estas arquitecturas
http://java.sun.com/developer/technicalArticles/WebServices/soa2/ Abril, 2005.

[7] – Página de Microsoft dedicada a estas arquitecturas
http://msdn.microsoft.com/architecture/soa/default.aspx Abril, 2005.

Usabilidad Web

[8] – Estilos Web
http://www.webstyleguide.com/process/index.html Diciembre, 2004.
Web Style Guide (2nd. edition) - Basic Design Principles for Creating Web Sites de Lou Rosenfeld.

Proyecto Link-All

[9] – Sistemas de Información Federados a través de la Web (Link-All)
http://www.link-all.org/PublicSite/Summary.htm Junio, 2004.
http://www.fing.edu.uy/inco/proyectos/linkall/Documentos/Descripcion-LinkAll.doc Junio, 2004.

Patrones de diseño

[10] – Página oficial de Sun sobre J2EE
http://java.sun.com/j2ee/index.jsp Agosto, 2004.

[11] – UML y Patrones
Una introducción al análisis y diseño orientado a objetos y al proceso unificado. LARMAN, CRAIG. Segunda Edición. Ed: Prentice Hall, Inc. (2003) - ISBN: 84-205-3438-2.

Proyecto de Grado

[12] – Informe Final – Anexo: Tecnologías de Ayuda
"Mecanismo de Ayuda Contextual para Sistemas de Información Federados". Proyecto de Grado 2004.
Laura González, Guillermo Roldós, Flavia Serra

APÉNDICES

A. OPENLASZLO

OpenLaszlo es una plataforma open source y gratuita que permite el desarrollo y distribución de "Rich Internet Applications" (RIA). La arquitectura del sistema OpenLaszlo combina el poder y usabilidad del diseño cliente-servidor con las ventajas administrativas y rentabilidad de las aplicaciones web. Las aplicaciones desarrolladas con OpenLaszlo pueden ejecutarse en la mayoría de los exploradores web corriendo en los principales sistemas operativos.

Para desarrollar aplicaciones con OpenLaszlo se utiliza XML y un lenguaje de scripting portable a través de exploradores web. Una aplicación OpenLaszlo puede ser tan pequeña como un simple archivo de código, o puede estar compuesta por múltiples archivos que definen clases y librerías reutilizables.

Kit de desarrollo de OpenLaszlo

El kit de desarrollo de OpenLaszlo consiste en un compilador desarrollado en Java, una biblioteca runtime JavaScript y un Servlet Java opcional que provee servicios adicionales para la aplicación en ejecución.

1. Compilador Java
   El compilador desarrollado en Java, compila los archivos fuente (archivos XML con extensión LZX) en archivos ejecutables en el entorno de ejecución objetivo. El entorno de ejecución objetivo actual de OpenLaszlo es el Flash Player, pero se planea poder generar archivos ejecutables para otros entornos de ejecución. El compilador provee las siguientes características:

   - Compilación de XML UI
     La descripción LZX de una interfaz de usuario, es una descripción XML declarativa de la interfaz de una aplicación. El compilador transforma esta descripción en código binario que crea la interfaz de la aplicación cuando ésta se ejecuta.

   - Compilación ECMAScript
     Los métodos y manejadores de eventos, en una interfaz de usuario LZX, son escritos en ECMAScript. ECMAScript se puede ver como la intersección de los lenguajes JavaScript y JScript. Es un lenguaje de programación creado para capturar los elementos principales comunes de ambos lenguajes. El compilador transforma el código fuente de este lenguaje en código binario optimizado.

   - Compilación de multimedia, datos, y fuentes
     El compilador también procesa archivos PNG, JPG, GIF, SWF, MP3, y archivos de fuentes TrueType y los incluye en archivos objeto de la aplicación.

   - Reportes de tamaño
     Reportes HTML del tamaño de la aplicación.

2. Servlet
   El Servlet redirige los pedidos de las aplicaciones para tipos de multimedia adicionales y para web services SOAP y XML-RPC. El servlet provee características como:

   - Codificación de una gran variedad de tipos de multimedia para que puedan ser desplegados por el Flash Player
   - Codificación de respuestas XML a un formato binario compacto
   - Cache de lo resultados de los pedidos a servidores de datos y multimedia
   - Redirección de pedidos a servidores de datos y multimedia
   - Soporte para pedidos SOAP, XML-RPC, y JavaRPC
   - Logueo y administración

3. Runtime Framework
   El Runtime Framework incluye componentes de interfaz de usuario, enlace a datos y servicios de red. Entre las características del Framework se encuentran:

   - Una rica librería de componentes de interfaz de usuario
   - Una variedad de manejadores para automáticamente posicionar y ajustar el tamaño de los componentes

- Un sistema de animación declarativo que provee animación para todos los elementos de la interfaz de usuario
- Un sistema de restricciones declarativo que automáticamente actualiza las propiedades de los atributos de la interfaz de usuario en función de otros valores o propiedades de componentes
- Actualización automática de componentes de interfaz de usuario desde datos
- Pedidos HTTP de servicios XML, SOAP, XML-RPC, y JavaRPC
- Debugging

Distribución de aplicaciones

Como se muestra en la figura A.1, OpenLaszlo ofrece dos formas para la distribución de las aplicaciones.

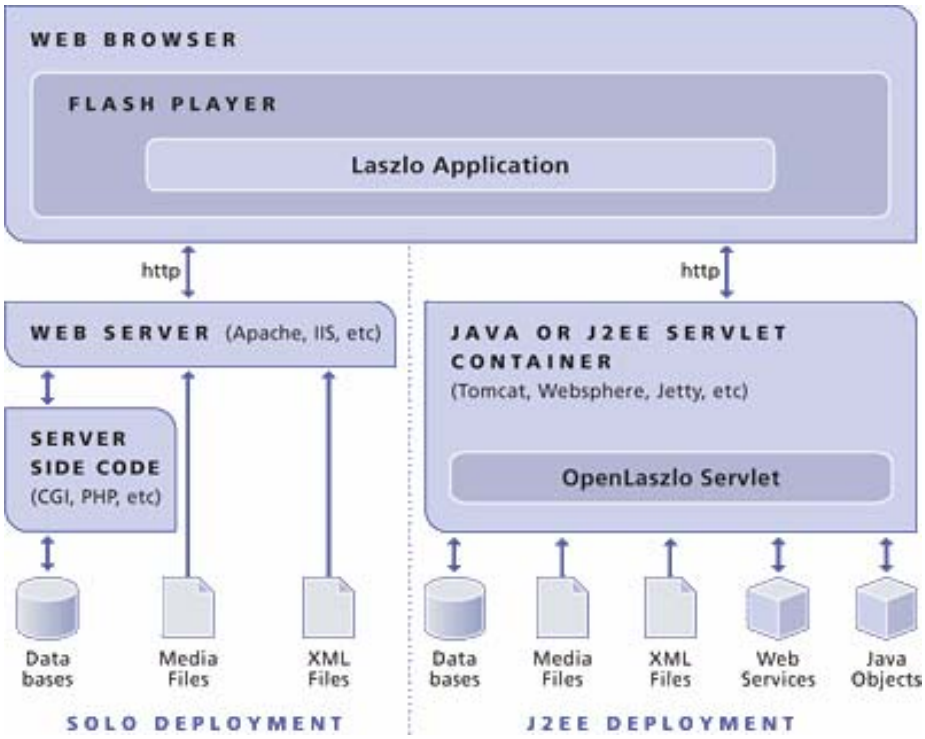


Figura A.1

SOLO (Standalone OpenLaszlo Output)

Las aplicaciones OpenLaszlo son precompiladas y distribuidas en cualquier servidor web HTTP. Este tipo de distribución soporta integración de datos vía XML sobre HTTP.

Servidor OpenLaszlo – J2EE

Las aplicaciones son compiladas y puestas en cache en tiempo de ejecución dentro de un contenedor de servlet o J2EE. Este tipo de distribución es necesario para aplicaciones que requieren integración a través de SOAP, XML-RPC o Java-RPC, o para aplicaciones que requieren conexiones persistentes o codificación de multimedia en tiempo de ejecución.

Desarrollo de aplicaciones y ejemplos

El código fuente de una aplicación OpenLaszlo es un documento XML, o una colección de documentos XML. Una aplicación OpenLaszlo está siempre incluida entre las siguientes etiquetas: <canvas> y </canvas>. OpenLaszlo provee un conjunto de componentes base para acelerar el desarrollo de RIAs. Dichos componentes tienen atributos que permiten modificar su apariencia y comportamiento.

A continuación se da una breve explicación y ejemplos de los principales componentes que están disponibles al momento de desarrollar una aplicación.

1. El siguiente ejemplo muestra la utilización de la etiqueta <text> y de los atributos bgcolor (color de fondo), fgcolor (color de primer plano) y fontsize (tamaño de letra).

```
<canvas bgcolor="blue">
     <text fgcolor="white" fontsize="14">Hola Mundo !!!</text>
</canvas>
```

2. La etiqueta <button> es otro de los componentes incluidos en la librería de componentes de OpenLaszlo.

```
<canvas>
    <button>Hello World!</button>
</canvas>
```

3. La etiqueta <window> permite fácilmente la creación de aplicaciones con múltiples ventanas dentro de un explorador web.

```
<canvas>
    <window resizable="true">
        <button>Hello World!</button>
    </window>
</canvas>
```

4. La etiqueta <view> es el elemento visible más básico de LZX. Una view puede ser utilizada como:
    a. Un componente básico
    b. Un contenedor para desplegar imágenes, audio, video, etc
    c. Un contenedor donde ubicar otros componentes
    d. El punto de partida para crear nuevos componentes

```
<canvas>
    <view width="50" height="50" bgcolor="blue"/>
</canvas>
```

5. Como la mayoría de los lenguajes de programación modernos, OpenLaszlo maneja eventos. Los componentes de la interfaz de usuario tienen un conjunto de eventos, por ejemplo: oninit, onclick, onmouseup, etc.
El siguiente código define un botón, con el texto “Hello World!”, y especifica que al hacer clic sobre ese botón el atributo texto del mismo cambiará a “Hola Mundo!”

```
<canvas>
     <button text="Hello World!">
          <method event="onclick">
              this.setAttribute('text','Hola Mundo!');
          </method>
     </button>
</canvas>
```

Se puede profundizar en los conceptos que se describen brevemente en este apéndice, consultando en la página oficial de OpenLaszlo: http://www.openlaszlo.org.

B. LUCENE

Es una API para búsqueda de texto de alta performance, multiplataforma por ser enteramente escrita en Java, y open source. Inicialmente un sub-proyecto de Apache Jakarta, el 14 de febrero de 2005 pasa a ser un proyecto principal de Apache. Se integra a cualquier aplicación con baja inversión en tiempo de investigación.

A continuación se describe el funcionamiento de esta API:

1. Inicialmente, se deben crear los índices de búsqueda. Estos índices son conjuntos de "documentos" formados por una o más parejas (nombre, valor) llamados "campos", como se muestra en la figura B.1:

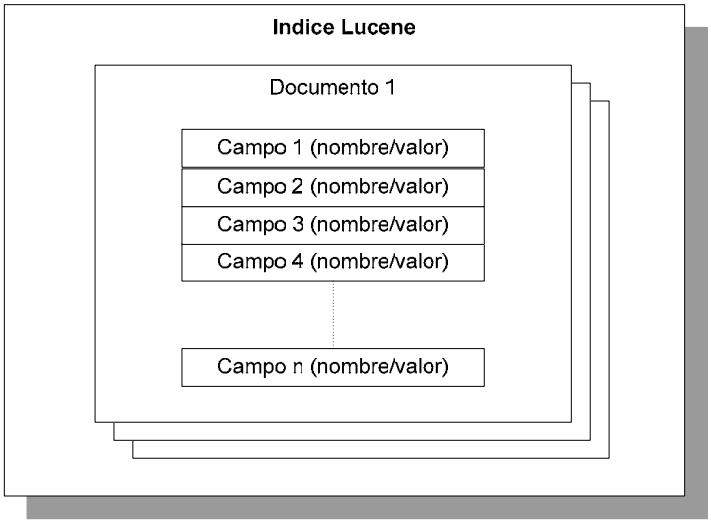


Figura B.1

Todos los documentos de un índice deben tener los mismos campos definidos, y es sobre éstos que se realiza la búsqueda. Para crear un índice, se debe utilizar la clase *IndexWriter*, como se ve en el siguiente ejemplo:

```
IndexWriter writer = new IndexWriter(FILE_NAME, new StandardAnalyzer(), true);
```

Luego se debe definir qué campos se grabarán en los documentos. Este paso es clave, ya que de ello dependerá la potencia que tendrán las búsquedas. Sin embargo, los campos que siempre deben estar son: el contenido del archivo que se indexa, y la ubicación de éste (para que sea posible ubicar el archivo una vez buscado). Una vez definidos qué valores tendrá cada campo, se debe crear cada documento (a través de la clase *Document*), cargar cada campo con el valor definido, y finalmente incorporar dicho documento al índice, como se ve en el siguiente ejemplo:

```
Document doc = new Document();
File f;
doc.add(Field.Text("path", f.getPath()+f.getName()));
doc.add(Field.Keyword("modified", DateField.timeToString(f.lastModified())));
FileInputStream is = new FileInputStream(f);
Reader reader = new BufferedReader(new InputStreamReader(is));
doc.add(Field.Text("contents", reader));
writer.addDocument(doc);
```

Es importante señalar que Lucene no indexa archivos sino documentos, por lo que para indexar el contenido de un archivo, es necesario obtener dicho contenido y cargarlo en uno de los campos del documento a indexar. Una vez incorporados todos los documentos en el índice, se debe optimizar. Es decir, Lucene provee un optimizador para evitar artículos, preposiciones, etc. En este ejemplo, se utilizó la clase *StandardAnalyzer* como optimizador, aunque es posible utilizar otra o crearse una clase Analyzer propia (simplemente se debe brindar una lista de "tokens" a evitar, que reconocerá en el texto original y los descartará). Para optimizar el texto grabado y cerrar el archivo, se ejecutan las siguientes instrucciones:

```
writer.optimize();
writer.close();
```

2. Una vez creados los índices, es posible realizar búsquedas sobre ellos, indicando siempre sobre qué campos (de los definidos) se busca. La única forma de buscar en índices Lucene es a través de un objeto *Query*, en el cual se indican los valores que se desean buscar para cada campo, como se puede ver en el siguiente ejemplo:

```
Query query = QueryParser.parse(text, "contents", new StandardAnalyzer());
```

En este caso, se busca por el campo “contents”. Notar que en este caso se debe utilizar el mismo optimizador utilizado para crear el índice. Una vez creado el objeto *Query*, se puede realizar la búsqueda, como se indica a continuación:

```
Searcher searcher = new IndexSearcher(FILE_NAME);
Hits hits = searcher.search(query);
```

El objeto *Hits* es una colección de documentos resultantes de la búsqueda, y se puede recorrer de forma de obtener cada resultado y realizar lo que se desee con ellos, como se muestra a continuación:

```
for (int i = start; i<hits.length(); i++) {
  Document doc = hits.doc(i);
  String path = doc.get("path");
  System.out.println(i + ". " + path);
}
```

Un detalle importante es que un índice Lucene puede ser accedido por distintos procesos al mismo tiempo, ya que soporta concurrencia. Además, es posible actualizar un índice en tiempo real, con procesos accediéndolo al mismo tiempo.

Como se puede ver, realizar una búsqueda en un índice Lucene puede ser muy sencillo, como en este caso, aunque también es posible realizar búsquedas más complejas. Para esto, Lucene define una sintaxis para consultas (las que son analizadas por la clase *QueryParser*). El signo “+” indica algo que se busca, el “-“ indica algo que se desea evitar, se pueden buscar palabras o frases compuestas por más de una palabra (en este caso encerradas entre comillas), es posible realizar búsquedas por determinados campos, a través del carácter “:”, e incluso utilizando operadores lógicos como AND, OR y NOT, y caracteres “wildcard” como * y ?. A continuación se ilustra este lenguaje de consulta a través de algunos ejemplos de consultas válidas, suponiendo que se busca en un índice formado por los libros de una biblioteca:

- *autor:”Asimov”*
  Devolverá todos los libros de autores cuyos nombres contengan Asimov.

- *autor:”Isaac Asimov”*
  Devolverá todos los libros del autor Isaac Asimov.

- *autor:”Isaac Asimov” AND title:”Robot*”*
  Devolverá todos los libros del autor Isaac Asimov cuyos títulos contengan palabras que comiencen con Robot.

- *autor:”Isaac Asimov” -title:”Robot”*
  Devolverá todos los libros del autor Isaac Asimov cuyos títulos no contengan la palabra Robot.

- *autor:”Isaac Asimov” AND año_publicacion:[1950 TO 1960]*
  Devolverá todos los libros del autor Isaac Asimov que hayan sido publicados entre los años 1950 y 1960.

Se puede profundizar en los conceptos que se describen brevemente en este apéndice, consultando en la página oficial de Lucene: http://lucene.apache.org.

C. JFREECHART

JFreeChart es una librería de gráficos gratuita para la plataforma Java. Está diseñada para ser utilizada en aplicaciones, applets, servlets y JSPs, y es distribuida con su código fuente completo bajo los términos de la "GNU Lesser General Public Licence".

JFreeChart puede generar una gran variedad de tipos de gráficos. A continuación, en las figuras C.1, C.2 y C.3 se presentan algunos ejemplos de las posibilidades que provee esta herramienta.

1. Gráficos tipo tortas

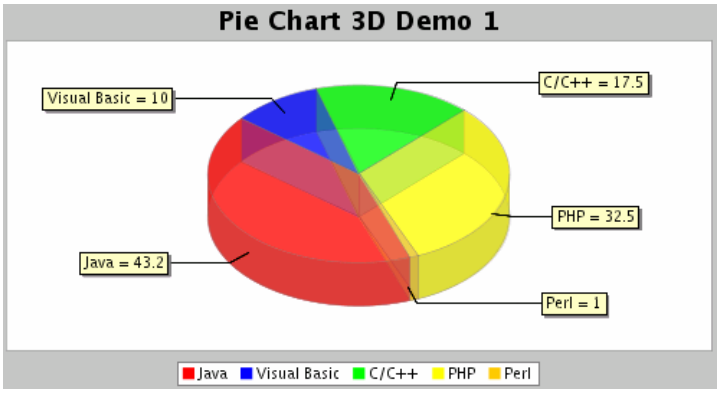


Figura C.1

2. Gráficos de barra

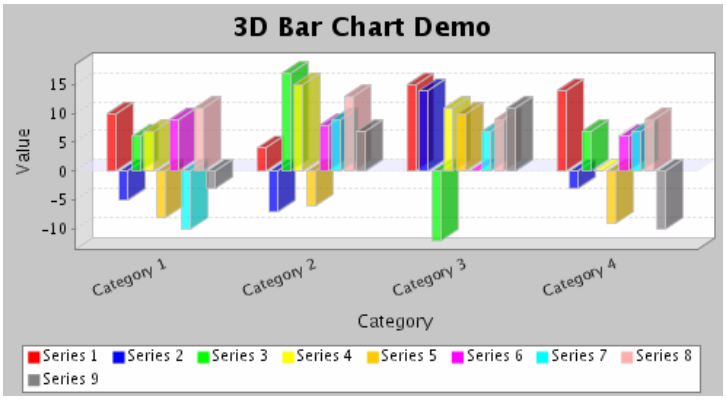


Figura C.2

3. Gráficos en tiempo

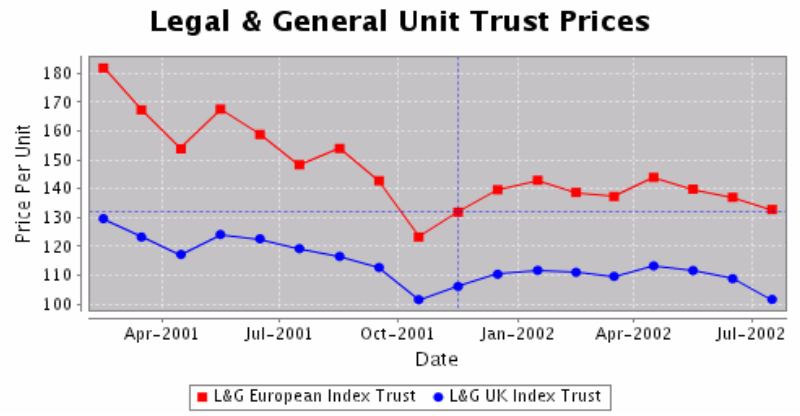


Figura C.3

Ejemplo

Se presenta a continuación los pasos a seguir y el código fuente java para generar un gráfico tipo torta.

1. Datos

   El primer paso es crear un conjunto de datos para que utilice el gráfico. Para esto se utiliza la clase DefaultPieDataset.

```
// se crea un dataset...
DefaultPieDataset data = new DefaultPieDataset();
data.setValue("Category 1", 43.2);
data.setValue("Category 2", 27.9);
data.setValue("Category 3", 79.5);
```

2. Creación del gráfico

   El siguiente paso es crear un objeto JFreeChart que pueda desplegar un gráfico utilizando el conjunto de datos definidos anteriormente.

```
// se crea el gráfico
JFreeChart chart = ChartFactory.createPieChart("Sample Pie Chart",
                                               data,
                                               true, // leyenda?
                                               true, // tooltips?
                                               false // URLs?
                                               );
```

3. Visualización del gráfico

   El paso final es desplegar el gráfico en algún lugar. JFreeChart es muy flexible en este sentido ya que hace uso de la clase Graphics2D. En este ejemplo se utilizará la clase ChartFrame que contiene todo lo necesario para desplegar el gráfico.

```
// se crea y despliega un contenedor para el gráfico
ChartFrame frame = new ChartFrame("Test", chart);
frame.pack();
frame.setVisible(true);
```

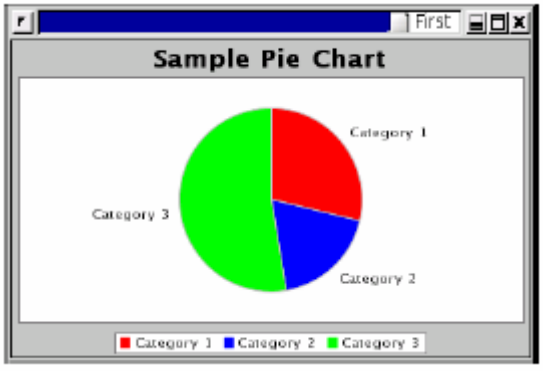


Figura C.4

Se puede profundizar en los conceptos que se describen brevemente en este apéndice, consultando en la página oficial de JFreeChart: http://www.jfree.org/jfreechart.

Mecanismo de Ayuda Contextual para Sistemas de Información Federados

Anexo
Documento de Casos de Uso

INCO – Facultad de Ingeniería – UDELAR

Montevideo, Junio 2005

Tutores:

Ing. Federico Piedrabuena
Dr. Ing. Raúl Ruggia

Estudiantes:

Laura González
Guillermo Roldós
Flavia Serra

ÍNDICE

1 ACTORES:

1.1 Usuario

Actor que representa a la persona que navega en el sitio de un Sistema de Información Federado.

1.2 Sistema

Actor que representa al Sistema de Ayuda Contextual.

1.3 Proveedor de Servicios de Contexto

Actor que representa al componente que se encargará de comunicarse con el Sistema de Información Federado.

1.4 Proveedor de Servicios de Ayuda

Actor que representa al componente que se encargará de brindar los datos de ayuda.

1.5 Proveedor de Servicios de Búsqueda de Texto

Actor que representa al componente que se encargará de realizar las búsquedas de texto.

2 DIAGRAMA DE CASOS DE USO:

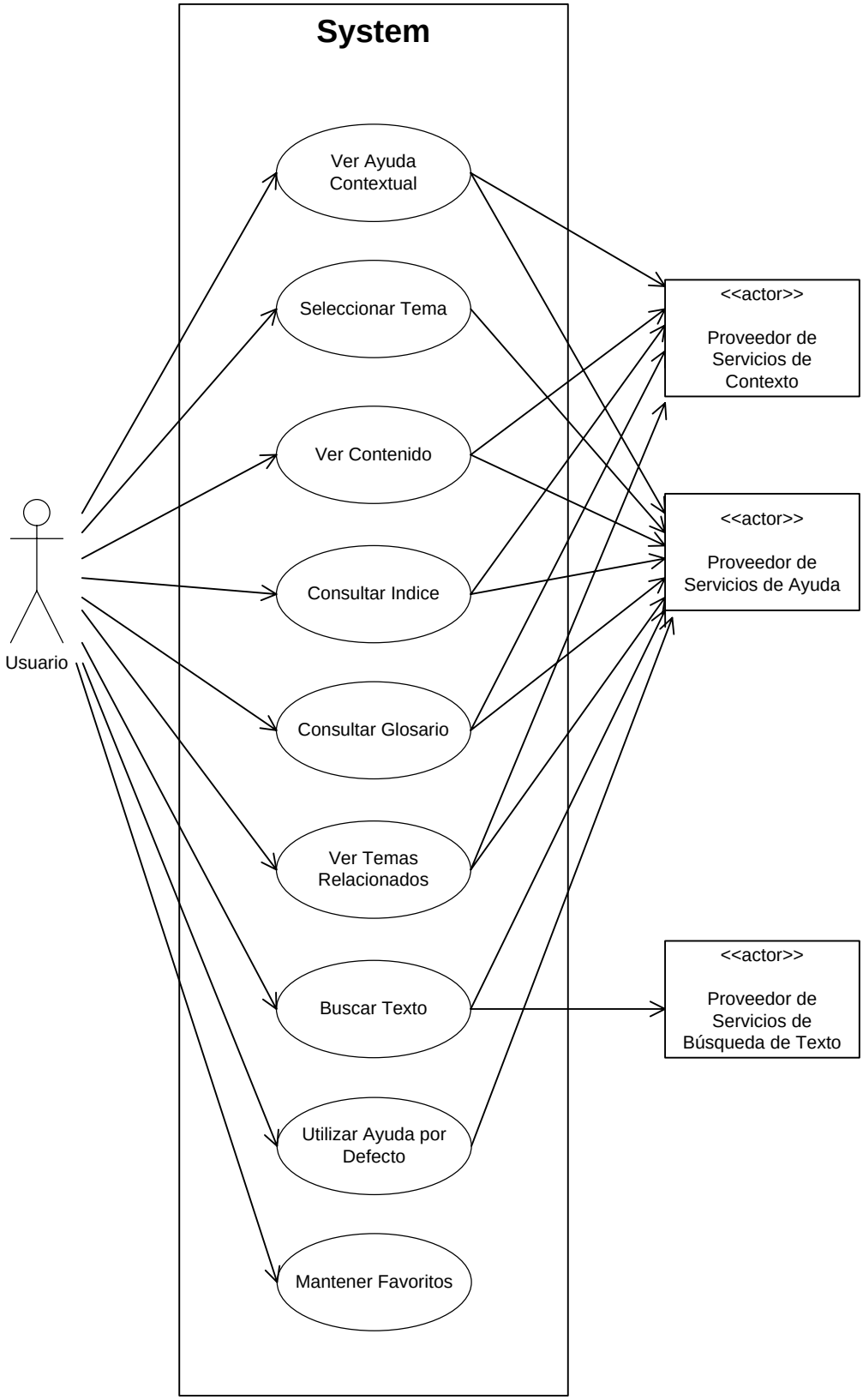


Figura 1

3 CASOS DE USO:

3.1 Caso de Uso UC01 – Ver Ayuda Contextual

3.1.1 Actores
Usuario (iniciador), Proveedor de Servicios de Contexto, Proveedor de Servicios de Ayuda.

3.1.2 Propósito
Mostrar al usuario ayuda contextual.

3.1.3 Resumen:
Un usuario que navega en el sitio web de un Sistema de Información Federado y posiblemente está haciendo uso de una aplicación, hace clic sobre el link de ayuda. El sistema le brinda ayuda o información, de acuerdo a determinadas variables, como pueden ser características del usuario, aplicación y/o funcionalidad que está ejecutando, posibles errores que el usuario haya cometido, etc.

3.1.4 Pre-Condiciones:
El usuario debe estar navegando en el sitio web del Sistema de Información Federado.

3.1.5 Tipo
Primario.

3.1.6 Flujo Normal de Eventos:

Acción del Usuario	Respuesta del Sistema
1. Hace clic sobre el link de ayuda.	
	2. Determina el contexto.
	3. Recupera la ayuda a ser mostrada de acuerdo al contexto.
	4. Muestra la pantalla de ayuda, con la ubicación actual del usuario (a través de un árbol de contenido) y la ayuda obtenida, correspondiente a la acción o funcionalidad que está ejecutando el usuario.

3.1.7 Flujo de Eventos Alternativo:
Línea 2: Si el sistema no puede obtener el contexto, utiliza uno por defecto.
Línea 3: Si el sistema no puede obtener la ayuda a ser mostrada (ya sea porque no existe o porque hubo un error al intentar obtenerla), el sistema muestra una ayuda por defecto. **(Ver 3.8 – UC08)**
Línea 4: Si el sistema detecta que el usuario ha cometido un error, le muestra al usuario información sobre el error cometido y le presenta posibles causas y soluciones al problema presentado.
Línea 4: Si el usuario no está logueado, se le muestra información de cómo loguearse.
Línea 4: Si el usuario está logueado, pero no está ejecutando ninguna aplicación, se le muestra un listado de las aplicaciones que el usuario puede ejecutar y como debe proceder para hacerlo.

3.1.8 Post-Condiciones:
Se habilitan en la pantalla de ayuda, links para que el usuario pueda consultar el contenido de la ayuda **(Ver 3.3 – UC03)**, buscar un texto **(Ver 3.4 – UC04)**, consultar el índice de la ayuda **(Ver 3.5 – UC05)**, consultar el glosario **(Ver 3.6 – UC06)**, ver contenidos relacionados **(Ver 3.7 – UC07),** y mantener sus temas favoritos **(Ver 3.9 – UC09).**

3.1.9 Requerimientos Especiales:
No hay.

3.2 Caso de Uso UC02 – Seleccionar Tema

3.2.1 Actores:
Usuario (iniciador), Proveedor de Servicios de Ayuda.

3.2.2 Propósito:
Mostrar al usuario información de un tema específico de la ayuda.

3.2.3 Resumen
En la pantalla de ayuda, a menudo se tendrán links a temas de ayuda, ya sea porque se ha consultado el árbol de contenido, se ha consultado el índice, o se ha efectuado alguna otra acción. En esa situación, el usuario hace clic sobre el link del tema cuya información desea consultar y se presenta en pantalla dicha información.

3.2.4 Pre-Condiciones:
Debe estar disponible, en la pantalla de ayuda, un link al tema de ayuda que el usuario desea consultar.

3.2.5 Tipo
Primario.

3.2.6 Flujo Normal de Eventos:

Acción del Usuario	Respuesta del Sistema
1. Hace clic sobre un link a un tema de ayuda.	
	2. Obtiene la ayuda a ser mostrada de acuerdo al contexto y al tema seleccionado por el usuario.
	3. Muestra la ayuda obtenida.

3.2.7 Flujo de Eventos Alternativo:
Línea 3: Si el sistema no puede obtener la ayuda a ser mostrada, muestra una ayuda basada en heurísticas.

3.2.8 Post-Condiciones:
Se habilita link para ver temas relacionados al seleccionado. **(Ver 3.7 – UC07)**

3.2.9 Requerimientos Especiales:
No hay.

3.3 Caso de Uso UC03 – Ver Contenido

3.3.1 Actores:
Usuario (iniciador), Proveedor de Servicios de Contexto, Proveedor de Servicios de Ayuda.

3.3.2 Propósito:
Mostrar al usuario un árbol con el contenido de la ayuda.

3.3.3 Resumen
El usuario hace clic sobre el link Contenido y el contenido de la ayuda es mostrado en pantalla a través de un árbol jerárquico.

3.3.4 Pre-Condiciones:
La pantalla de ayuda está disponible para el usuario. **(Ver 3.1 – UC01)**

3.3.5 Tipo
Primario.

3.3.6 Flujo Normal de Eventos:

Acción del Usuario	Respuesta del Sistema
1. Hace clic sobre el link Contenido	
	2. Obtiene el contenido de la ayuda de la aplicación seleccionada.
	3. Muestra el árbol de contenido obtenido, en pantalla.
	4. Selecciona el primer tema en el árbol de contenido.
	5. Muestra la información correspondiente al tema seleccionado en el árbol de contenido.

3.3.7 Flujo de Eventos Alternativo:
Línea 2: Si el sistema no puede obtener la ayuda a ser mostrada, muestra una ayuda basada en heurísticas.

3.3.8 Post-Condiciones:
Quedan disponibles links a los temas de ayuda, para que el usuario pueda seleccionar un tema específico de la ayuda. **(Ver 3.2 -UC02)**

3.3.9 Requerimientos Especiales:
No hay.

3.4 Caso de Uso UC04 – Buscar Texto

3.4.1 Actores:
Usuario (iniciador), Proveedor de Servicios de Contexto, Proveedor de Servicios de Ayuda, Proveedor de Servicios de Búsqueda de Texto.

3.4.2 Propósito:
Buscar una palabra o frase dentro del texto de los temas de la ayuda.

3.4.3 Resumen
El usuario hace clic sobre el link buscar, e ingresa una palabra o frase en un cuadro de texto. El usuario hace clic sobre el botón buscar, y se muestran en pantalla los temas cuyo texto contienen dicha palabra o frase.

3.4.4 Pre-Condiciones:
La pantalla de ayuda está disponible para el usuario. **(Ver 3.1 – UC01)**

3.4.5 Tipo
Secundario.

3.4.6 Flujo Normal de Eventos:

Acción del Usuario	Respuesta del Sistema
1. Hace clic sobre el link buscar.	
	2. Presenta en la pantalla de ayuda un cuadro de texto, un "radiobutton" para indicar el alcance de la búsqueda, y un botón con el texto buscar.
3. Ingresa la palabra o frase que desea buscar y el alcance de la búsqueda.	
	4. Busca la palabra o frase en los temas de la aplicación.
	5. Muestra los temas en los cuales encontró dicho palabra o frase.

3.4.7 Flujo de Eventos Alternativo:
Línea 4: Si el usuario selecciona todas las aplicaciones, se realiza la búsqueda en todas las aplicaciones.
Línea 5: Si el sistema no encuentra ningún tema que contenga la palabra o frase ingresada, se le informa al usuario.

3.4.8 Post-Condiciones:
Quedan disponibles links a los temas encontrados, para que el usuario pueda consultar un determinado tema de la ayuda. **(Ver 3.2 – UC02)**

3.4.9 Requerimientos Especiales:
No hay.

3.5 Caso de Uso UC05 – Consultar Índice

3.5.1 Actores:
Usuario (iniciador), Proveedor de Servicios de Contexto, Proveedor de Servicios de Ayuda.

3.5.2 Propósito:
Consultar el índice de la ayuda.

3.5.3 Resumen
El usuario hace clic sobre el link índice y se muestran en pantalla las entradas del índice y sus temas asociados. El usuario puede ingresar una cadena de caracteres para filtrar las entradas del índice mostradas.

3.5.4 Pre-Condiciones:
La pantalla de ayuda está disponible para el usuario. **(Ver 3.1 – UC01)**

3.5.5 Tipo
Secundario.

3.5.6 Flujo Normal de Eventos:

Acción del Usuario	Respuesta del Sistema
1. Hace clic sobre el link índice.	
	2. Obtiene las entradas del índice de la aplicación seleccionada.
	3. Presenta las entradas del índice obtenidas, ordenadas alfabéticamente, cada una de ellas con sus temas asociados. Además se muestra en pantalla un cuadro de texto.
4. Ingresa un texto en el cuadro de texto.	
	5. Filtra las entradas del índice mostradas en pantalla de acuerdo al texto ingresado.

3.5.7 Flujo de Eventos Alternativo:
Línea 5: Si el usuario no ingresa ningún texto no se hace el filtrado.

3.5.8 Post-Condiciones:
Quedan disponibles links a los temas asociados a las entradas de cada índice mostrado en pantalla, para que el usuario pueda seleccionar un determinado tema de la ayuda. **(Ver 3.2 – UC02)**

3.5.9 Requerimientos Especiales:
No hay.

3.6 Caso de Uso UC06 – Consultar Glosario

3.6.1 Actores:
Usuario (iniciador), Proveedor de Servicios de Contexto, Proveedor de Servicios de Ayuda.

3.6.2 Propósito:
Consultar la definición de un término del glosario.

3.6.3 Resumen
El usuario hace clic sobre el link glosario y se muestran en pantalla los términos del glosario. Cuando el usuario hace clic sobre uno de dichos términos se muestra en pantalla la definición del mismo.

3.6.4 Pre-Condiciones:
La pantalla de ayuda está disponible para el usuario. **(Ver 3.1 – UC01)**

3.6.5 Tipo
Secundario.

3.6.6 Flujo Normal de Eventos:

Acción del Usuario	Respuesta del Sistema
1. Hace clic sobre el link glosario.	
	2. Obtiene los términos de la aplicación seleccionada.
	3. Presenta, en la pantalla de ayuda, los términos encontrados .
4. Hace clic sobre un término.	
	5. Obtiene la definición del término seleccionado por el usuario.
	6. Muestra en pantalla la definición obtenida.

3.6.7 Flujo de Eventos Alternativo:
Línea 4: Si el usuario no selecciona ningún término, no se muestra ninguna definición.

3.6.8 Post-Condiciones:
No hay.

3.6.9 Requerimientos Especiales:
No hay.

3.7 Caso de Uso UC07 – Ver Temas Relacionados

3.7.1 Actores:
Usuario (iniciador), Proveedor de Servicios de Contexto, Proveedor de Servicios de Ayuda.

3.7.2 Propósito:
Buscar otros temas de ayuda relacionados al tema de ayuda mostrado en pantalla.

3.7.3 Resumen
El usuario hace clic sobre el link Vea También y se le presentan en pantalla una lista de temas relacionados al tema de ayuda mostrado en pantalla.

3.7.4 Pre-Condiciones:
La pantalla de ayuda y el link temas relacionados están disponible para el usuario. **(Ver 3.1 – UC01)**

3.7.5 Tipo
Secundario.

3.7.6 Flujo Normal de Eventos:

Acción del Usuario	Respuesta del Sistema
1. Hace clic sobre el link Temas Relacionados.	
	2. Busca temas relacionados al tema mostrado en pantalla de acuerdo a palabras clave
	3. Muestra al usuario una lista con los temas encontrados.

3.7.7 Flujo de Eventos Alternativo:
Línea 3: Si el sistema no encuentra ningún tema se le informa al usuario.

3.7.8 Post-Condiciones:
Quedan disponibles links a los temas encontrados, para que el usuario pueda consultar un determinado tema de la ayuda. **(Ver 3.2 – UC02)**

3.7.9 Requerimientos Especiales:
No hay.

3.8 Caso de Uso UC08 – Utilizar Ayuda por Defecto

3.8.1 Actores:
Usuario (iniciador), Sistema (iniciador).

3.8.2 Propósito:
Mostrar guías o asistentes al usuario basándose en heurísticas.

3.8.3 Resumen
El usuario hace clic sobre el link para ver la ayuda por defecto, o no se puede obtener la ayuda solicitada por el usuario, y se muestra en pantalla ayuda basada en heurísticas.

3.8.4 Pre-Condiciones:
No hay.

3.8.5 Tipo
Opcional.

3.8.6 Flujo Normal de Eventos:

Acción del Usuario	Respuesta del Sistema
1. Hace clic sobre el link Asistente.	
	2. Muestra en pantalla ayuda basada en heurísticas.

3.8.7 Flujo de Eventos Alternativo:
No hay.

3.8.8 Post-Condiciones:
No hay.

3.8.9 Requerimientos Especiales:
No hay.

3.9 Caso de Uso UC09 – Mantener Favoritos

3.9.1 Actores:
Usuario (iniciador).

3.9.2 Propósito:
Permitir al usuario consultar, agregar y eliminar sus temas de ayuda favoritos.

3.9.3 Resumen
El usuario hace clic en el link favoritos y se presentan en pantalla los temas favoritos del usuario. El mismo podrá entonces agregar y eliminar dichos temas favoritos.

3.9.4 Pre-Condiciones:
La pantalla de ayuda está disponible para el usuario. **(Ver 3.1 – UC01)**

3.9.5 Tipo
Opcional.

3.9.6 Flujo Normal de Eventos

Flujo Consultar Temas Favoritos

Acción del Usuario	Respuesta del Sistema
1. Hace clic sobre el link Favoritos.	
	2. Obtiene los temas de ayuda favoritos del usuario.
	3. Muestra los temas obtenidos al usuario.
	4. Se habilitan un botón para Eliminar temas favoritos.

Flujo Eliminar Tema Favorito

Acción del Usuario	Respuesta del Sistema
1. Consulta sus temas favoritos. **(Ver flujo Consultar Temas Favoritos)**	
2. Selecciona un tema favorito.	
3. Hace clic en le botón Eliminar.	
	4. Pregunta al usuario si realmente desea eliminar el tema de sus temas favoritos.
5. Confirma la eliminación del tema favorito.	
	6. Elimina el tema seleccionado por el usuario de sus temas favoritos.

Flujo Agregar Tema Favorito

Acción del Usuario	Respuesta del Sistema
1. Selecciona un tema de la ayuda. **(Ver 3.2 – UC02)**	
2. Hace clic sobre el link Agregar a Favoritos.	
	3. Agrega el tema, a los temas favoritos del usuario
	4. Muestra un mensaje informando que el tema ha sido agregado a los favoritos correctamente.

3.9.7 Flujo de Eventos Alternativo:
No hay.

3.9.8 Post-Condiciones:
No hay.

3.9.9 Requerimientos Especiales:
No hay.

4 REFERENCIAS CRUZADAS

En esta sección se muestran las referencias cruzadas entre casos de uso y los requerimientos funcionales del sistema [1], a través de una tabla.

En la primera fila de la tabla se listan los números de requerimientos funcionales (de acuerdo al documento de requerimientos) y en la primera columna de la tabla, los nombres de los casos de uso. Una cruz en el casillero i, j (donde i es la fila y j la columna de la tabla) significa que el caso de uso que se encuentra en la fila i ayuda a cumplir el requerimiento funcional que se encuentra en la columna j.

	RQ01	RQ02	RQ03	RQ04	RQ05	RQ06	RQ07	RQ08	RQ09
UC01	x	x		x			x	x	
UC02		x							
UC03		x							
UC04							x		
UC05							x		
UC06							x		
UC07			x						
UC08					x	x			x
UC09							x		

5 REFERENCIAS

Proyecto de Grado

[1] – Documento de Requerimientos
"Mecanismo de Ayuda Contextual para Sistemas de Información Federados". Proyecto de Grado 2004.
Laura González, Guillermo Roldós, Flavia Serra

Mecanismo de Ayuda Contextual para Sistemas de Información Federados

Anexo
Manual de Usuario

INCO – Facultad de Ingeniería – UDELAR

Montevideo, Junio 2005

Tutores:

Ing. Federico Piedrabuena
Dr. Ing. Raúl Ruggia

Estudiantes:

Laura González
Guillermo Roldós
Flavia Serra

ÍNDICE

1 INTRODUCCIÓN

1.1 Objetivo

El objetivo de este documento es obtener una guía de uso del Sistema de Ayuda Contextual. En principio se describe la forma de ingreso al sistema y luego cada una de sus funcionalidades.

1.1.1 Alcance

Tanto este documento como el Sistema de Ayuda Contextual están dirigidos a todo perfil de usuario. Esto significa que no se exige ningún tipo de conocimiento en informática para poder interactuar con dichas herramientas.

1.1.2 Definiciones

A continuación se presentan un conjunto de definiciones que pueden ser útiles para el lector.

- Usuario
  Persona que utiliza una aplicación para realizar alguna tarea.

- Documentación de Usuario
  Material (impreso o grabado) que provee al usuario información sobre cómo utilizar un Sistema para obtener un resultado.

- Funcionalidad
  Tarea que el usuario puede realizar, provista por una aplicación.

- Aplicación
  Software o servicio que ofrece un conjunto de tareas.

- Browser
  Software necesario para visualizar páginas Html. Es provisto por varias empresas de Software a nivel mundial, y son de libre distribución.

- Sistema de Información Federado
  Conjunto de aplicaciones que comparten datos y/o servicios.

1.2 Descripción del Sistema

El Sistema de Ayuda Contextual provee funcionalidades de ayuda en línea a las aplicaciones que integran Sistemas de Información Federados. Dicha ayuda es sensible al contexto de ejecución, el cual depende entre otras cosas, de las características del usuario y de la aplicación y funcionalidad que se están ejecutando. Los usuarios podrán visualizar las ayudas de todas las aplicaciones como si fuera una sola, pudiendo así, conocer y trabajar en todas las posibilidades que le ofrece el Sistema de Información.

1.2.1 Características Generales

Se pueden identificar como características generales del Sistema las siguientes:

- Ambiente de ejecución
  El Sistema se ejecuta a través de Internet mediante un Browser, utilizando páginas Web.

- Interfaz de Usuario
  Teniendo en cuenta que se estaba desarrollando un sistema para la Web, se tomaron en cuenta los conceptos básicos de diseño y usabilidad.

- Seguridad
  Se maneja el concepto de seguridad asociado a los contenidos que se visualizan, de modo que solamente usuarios autorizados puedan consultar ciertas pantallas de ayuda.

1.2.2 Requerimientos de Hardware y Software de base.

Los requerimientos de Hardware necesarios para el buen funcionamiento del Sistema de Ayuda se presentan en la tabla 1:

Requerimiento	Mínimo	Recomendado
Procesador	486 de 66 MHz	Pentium
Memoria RAM	16 Mb	32 Mb
Espacio libre en disco	10 Mb	20 Mb
Entrada de datos	Teclado	Teclado y ratón
Módem	Si	Si

Tabla 1

Además, debe instalarse el software Macromedia Flash Player y un Browser de acuerdo a las especificaciones que se listan en la tabla 2:

Browser	Versión Mínima
Microsoft Internet Explorer	6.0
Nestcape Navegator	7.2
Mozilla	1.7.5
Firefox	1.0

Tabla 2

En cuanto al Sistema Operativo, puede ser cualquiera en el que puedan correr las versiones de Browser anteriormente presentadas.

1.3 Organización del Documento

Este documento está compuesto por dos secciones:

- Introducción
  Brinda una idea general acerca del documento y del Sistema

- Contenido
  Presenta en detalle cada funcionalidad provista por el Sistema de Ayuda Contextual

1.3.1 Notación

Las convenciones que se tomaron en el documento en cuanto a notación son las siguientes:

- Citas textuales
  Figuran en letra *itálica*

- Resaltados
  Figuran en **negrita**

- Restricciones
  Figuran en un recuadro como sigue:

2 FUNCIONALIDADES

2.1 Ingreso al Sistema

Para iniciar el Sistema de Ayuda Contextual, debe seguir los siguientes pasos:

- Conectarse a Internet
  Como se destacó anteriormente, el Sistema se ejecuta a través de un Browser (explorador). Por lo tanto, debe ejecutar los siguientes ítems:
  Inicio / Programas / Accesorios / Comunicaciones / Acceso telefónico a redes
  Se presentará una pantalla similar a la figura 1 (dependiendo de las conexiones que el usuario tenga configuradas).

Figura 1

Debe ejecutar el ítem correspondiente a la conexión con Internet, el cual le desplegará una pantalla como la que se presenta en la figura 2.

Figura 2

Luego de realizar la conexión a Internet, aparecerá un indicador en la barra de tareas.

Atención:
Si por algún motivo no se puede realizar la conexión a Internet (línea ocupada, fallas en el servidor, etc.), el Sistema de Ayuda Contextual no podrá ser utilizado.

- Iniciar el explorador
  Una vez se haya realizado la conexión a Internet, debe iniciarse el explorador. Para esto, debe ejecutarse uno de los siguientes ítems:

  Internet Explorer: *Inicio / Programas / Internet Explorer*
  Netscape Navigator: *Inicio / Programas / Netscape / Netscape Navigator*

- Acceder al Sistema de Ayuda Contextual
  La forma de acceso al sistema dependerá del Sistema de Información Federado, por lo que se recomienda consultar el manual del Sistema de Información en el cual se está trabajando

- Página de presentación del Sistema de Ayuda Contextual
  Una vez que ingresa a la página principal, se presenta una pantalla como la imagen de la figura 3; siempre se desplegará la información (contenidos de ayuda), correspondiente al servicio (aplicación), que esta utilizando y a la tarea con la que esta trabajando el usuario.

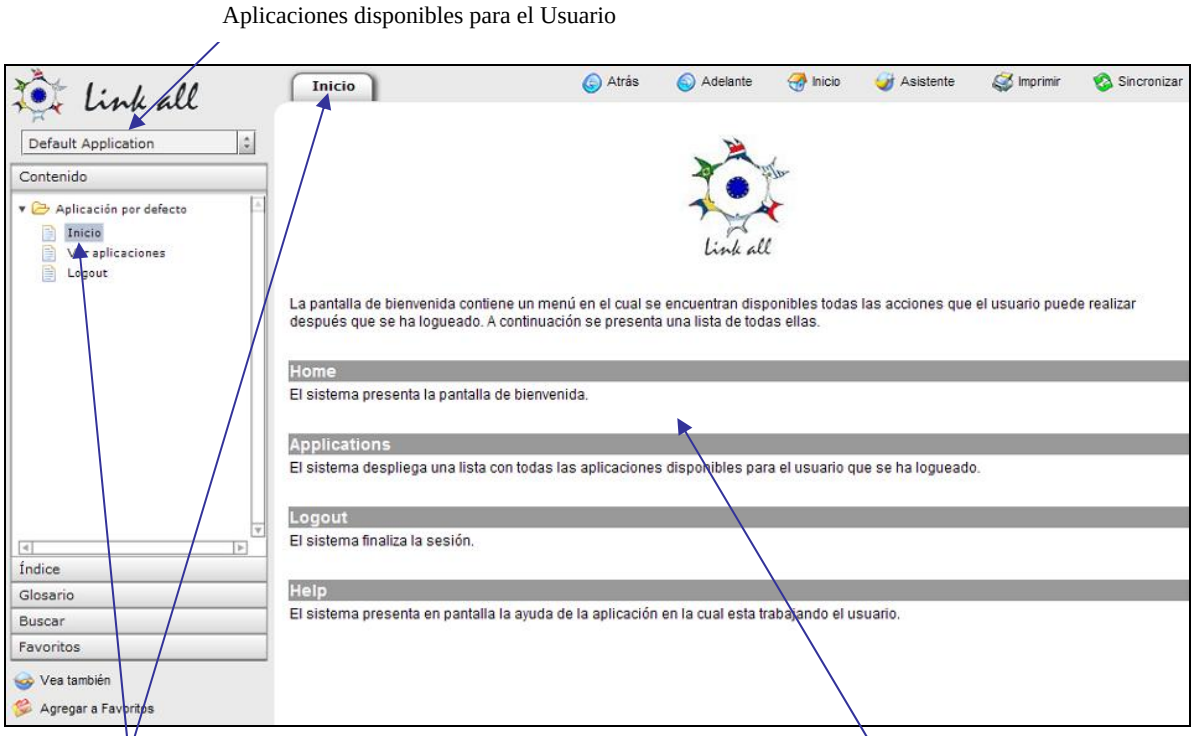


Figura 3

El contenido despliega toda la información de ayuda disponible sobre una determinada tarea. El título del mismo se presenta en la parte superior de la pantalla (figura 3) y en la *Tabla de Contenidos.* En la figura 5, se presenta una ampliación de la imagen anterior en la cual se puede apreciar la lista de opciones desplegable de aplicaciones y la *Tabla de Contenido*, donde se presenta seleccionado el título del contenido. En la figura 6, se muestra la imagen de la lista desplegable de aplicaciones cuando es seleccionado, **éste despliega sólo aquellas aplicaciones a las que puede acceder el usuario**.

La interfaz del Sistema de Ayuda Contextual **siempre** se presentará en el idioma en el cual está trabajando el usuario, en la figura 4 se muestra la imagen anterior en el idioma inglés.

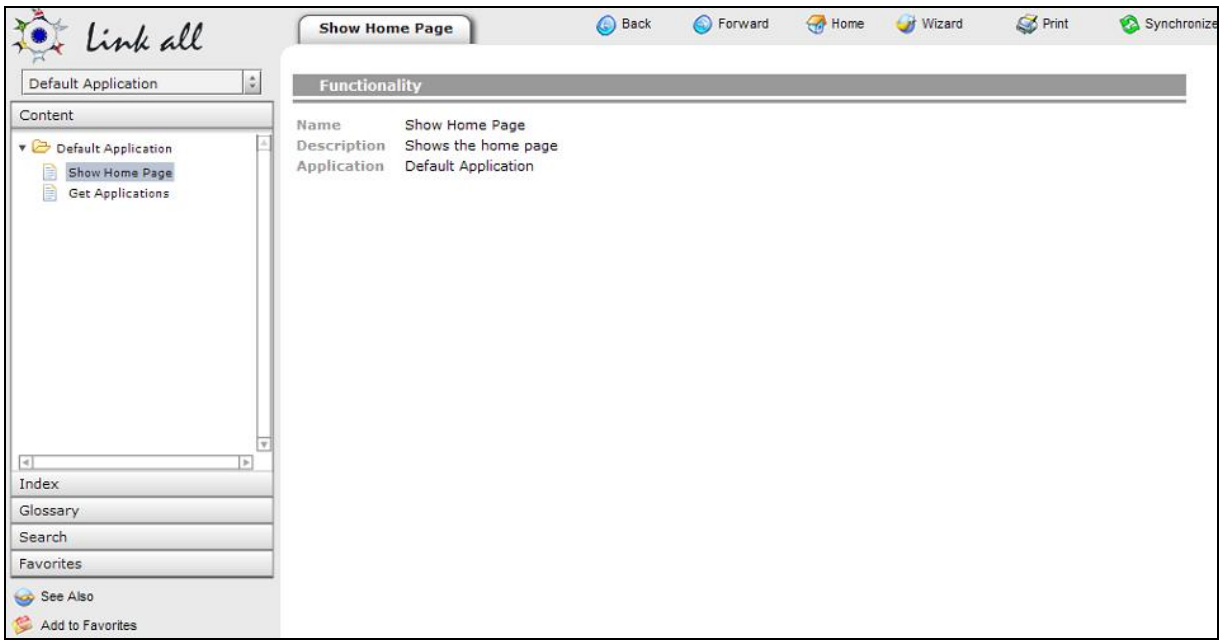


Figura 4

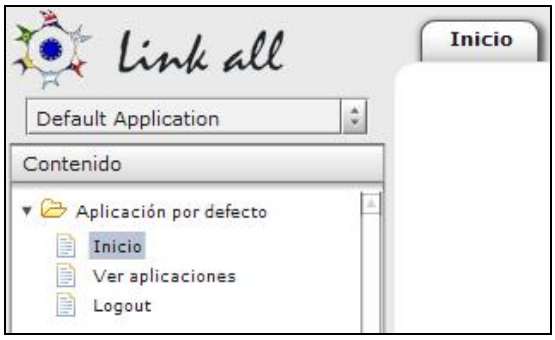


Figura 5

Inicialmente la ayuda se cargará con la información de la aplicación en la cual estaba trabajando el usuario en el momento de solicitar la ayuda. Luego, el usuario tendrá la libertad de continuar consultando la ayuda de todas las aplicaciones en las que tiene permiso.

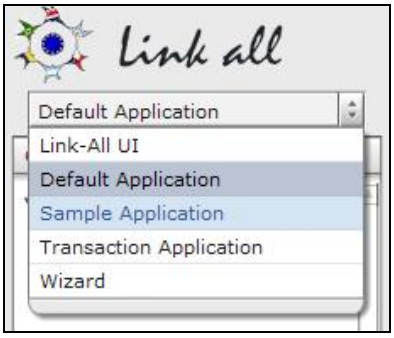


Figura 6

2.2 Tabla de Contenido

La *Tabla de Contenido* contiene una estructura de árbol y despliega una lista de nodos ordenados jerárquicamente. Como se presentó anteriormente, cada uno de los nodos contiene el nombre de un contenido (figura 7), que al ser seleccionado por el usuario desplegará en el lado derecho de la pantalla información referente al mismo. Es importante destacar, que en el árbol **sólo se presentarán aquellos contenidos en los cuales el usuario tiene permiso para consultar**.

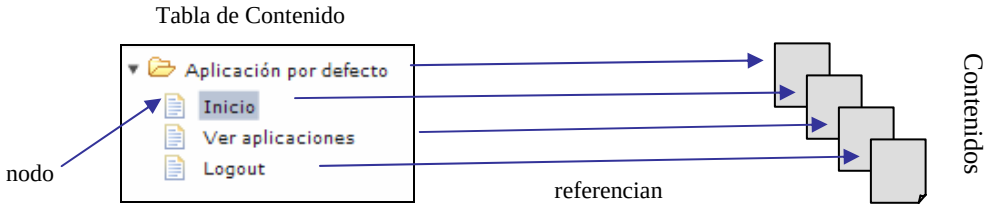


Figura 7

Para acceder a *Tabla de Contenido* debe seleccionarse el botón *Contenido* (figura 8), y la estructura con forma de "acordeón" se abrirá y presentará el conjunto de nodos disponibles para el usuario.

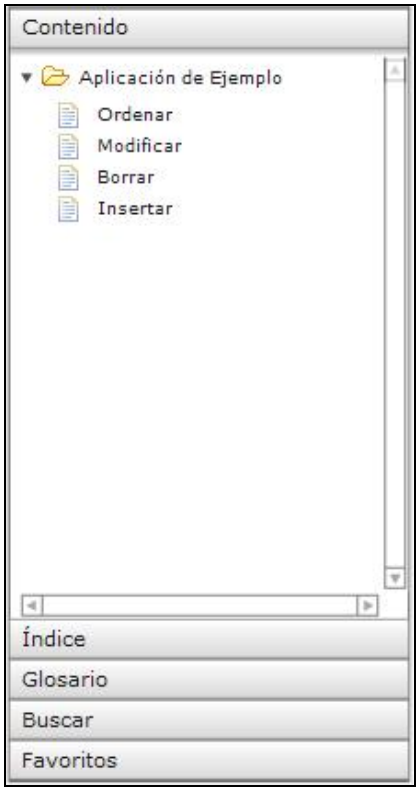


Figura 8

2.3 Índice

El *Índice* es un conjunto de palabras claves, como se muestra en la figura 9, cada una de ellas tiene un conjunto de contenidos asociados. Al igual que la *Tabla de Contenido*, éste brinda al usuario una forma sencilla de navegar a través de los contenidos de ayuda, pero en este caso agrupados de acuerdo a sus palabras clave (datos representativos del contenido de ayuda).

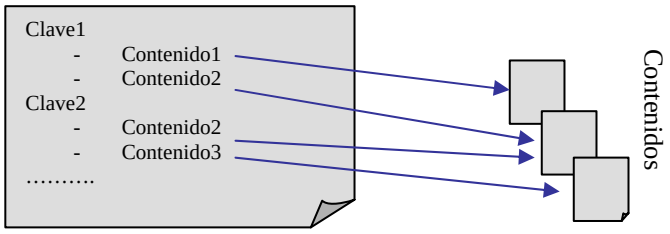


Figura 9

Para acceder, debe seleccionarse el botón *Índice*. En la figura 10 se presenta una imagen de cómo el *Índice* se despliega en pantalla.

Figura 10

A medida que el usuario ingresa el texto el sistema se posicionará en la primer palabra cuyas letras iniciales coincidan con las letras ingresadas hasta el momento por el usuario, un ejemplo de esto, se presenta en la figura 11.

Figura 11

2.4 Glosario

Como se muestra en la figura 12, el *Glosario* es un conjunto de parejas término – definición. Dado un término el usuario puede consultar la definición del mismo.

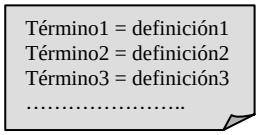


Figura 12

Para acceder, debe seleccionarse el botón *Glosario*. En la figura 13 se presenta una imagen de cómo se presenta el *Glosario*.

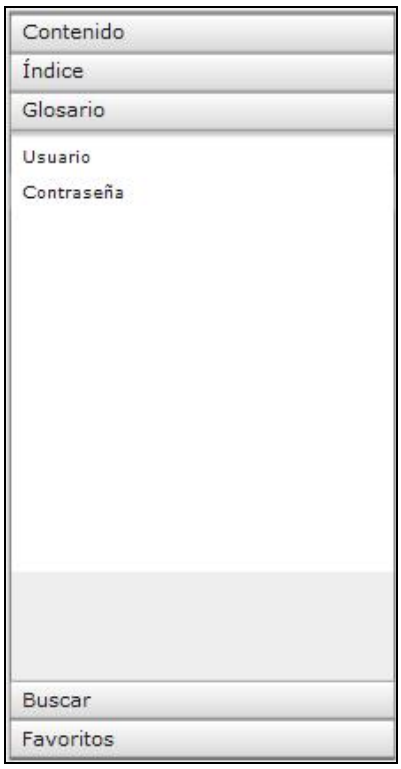


Figura 13

Cuando el usuario selecciona un término, el sistema presentará en la parte inferior la definición de mismo (figura 14).

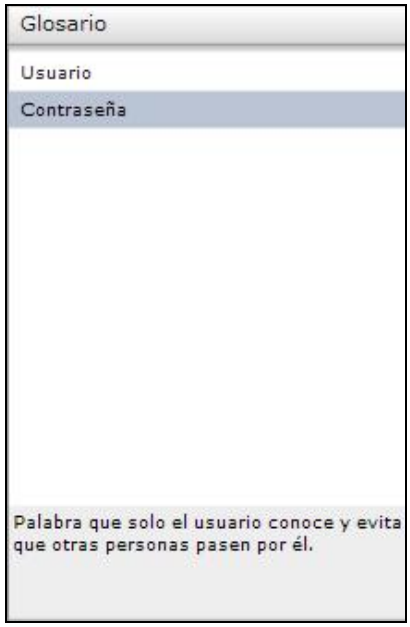


Figura 14

2.5 Buscar

Para buscar un texto dentro de los contenidos de ayuda el usuario deberá seleccionar el botón *Buscar* y se desplegará una pantalla como la figura 15.

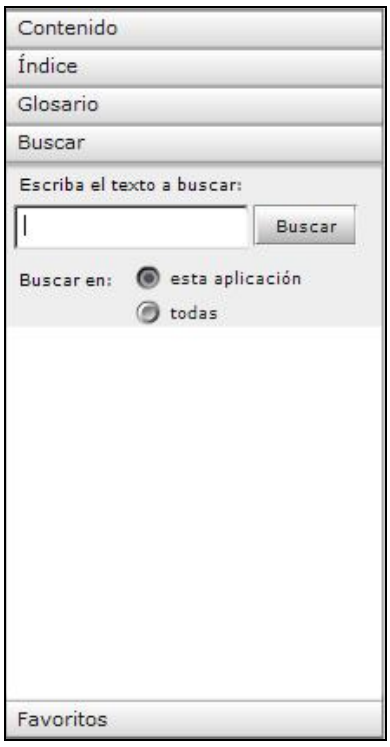


Figura 15

Para comenzar una búsqueda el usuario deberá especificar un texto y luego presionar el botón *Buscar*, como se muestra en la figura 16.

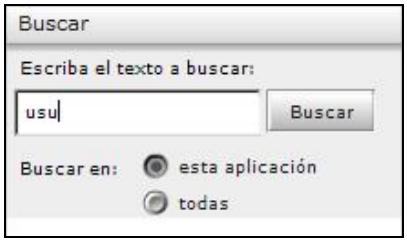


Figura 16

Finalizada la búsqueda (figura 17), el sistema presentará en pantalla todos los contenidos de la aplicación actual en el cual aparece el texto que ha ingresado. Delante del contenido se adjunta el nombre de la aplicación a la cual pertenece el mismo.

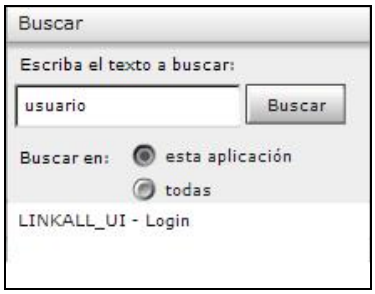


Figura 17

A su vez, el usuario tiene la opción de buscar el texto ingresado dentro de los contenidos de todas las aplicaciones que se listan en la lista desplegable de la figura 5. Seleccionando el ítem *todas* de la figura 17 y presionando nuevamente el botón *Buscar*, el sistema presentará el resultado de una nueva búsqueda, como se muestra en la figura 18.

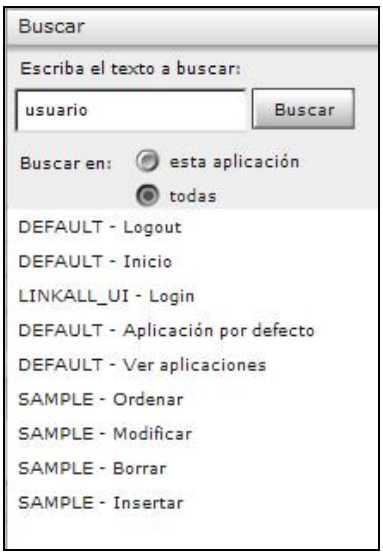


Figura 18

2.6 Favoritos

Los contenidos *Favoritos* son un conjunto de contenidos para los cuales el usuario ha decidido guardar una referencia, para poder acceder a ellos rápidamente más tarde. Para agregar un contenido a la lista de *Favoritos*, el usuario podrá seleccionar un contenido de la *Tabla de Contenido* (figura 19).

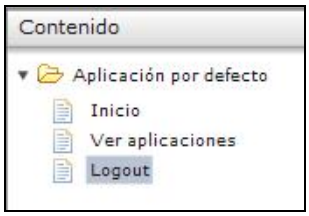


Figura 19

También podrá seleccionar un contenido de la lista de contenidos presentada en la sección *Índice* (figura 20).

Figura 20

Si lo desea, luego de realizar una búsqueda en la sección *Buscar* (figura 21), también podrá seleccionar todos aquellos contenidos de su interés para agregar a la lista de *Favoritos*.

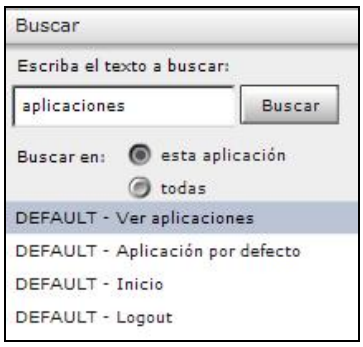


Figura 21

Luego de seleccionado un contenido, el usuario deberá presionar el botón *Agregar a Favoritos*, que se presenta en la figura 22.

Figura 22

Para acceder a la lista de contenidos favoritos, el usuario deberá seleccionar el botón *Favoritos* y se desplegará una pantalla como se muestra en la imagen de la figura 23. Cuando el usuario lo desee, podrá eliminar contenidos de la lista de favoritos, para esto, deberá seleccionar el contenido de la lista y posteriormente presionar el botón *Borrar de Favoritos*.

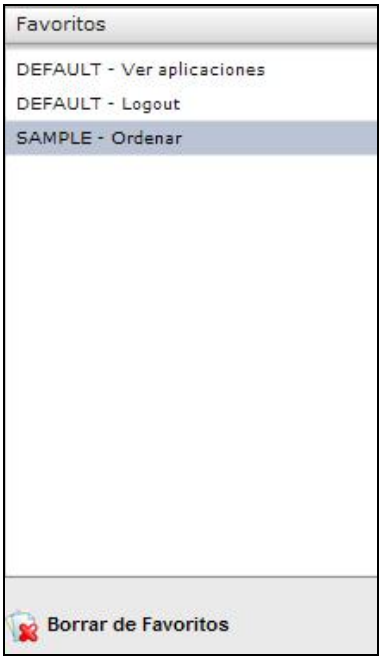


Figura 23

2.7 Vea También

Seleccionando el botón *Vea También* (figura 24), el usuario tendrá acceso a todos los contenidos relacionados con el contenido que esta visualizando. Los contenidos relacionados serán pertenecientes a la aplicación que esta seleccionada actualmente por el usuario.

Figura 24

2.8 Menú

En la parte superior de la pantalla están disponibles un conjunto de operaciones (figura 25), que le permitirán al usuario poder navegar con mayor fluidez a través de los contenidos.

Figura 25

2.8.1 Atrás y Adelante

Los botones *Atrás* y *Adelante* permiten al usuario navegar a través de todos los contenidos visitados.
Presionando el botón *Atrás* (figura 26), el usuario podrá visualizar el contenido anteriormente consultado.

Figura 26

Presionando el botón *Adelante* (figura 27), el usuario podrá volver a los contenidos posteriores.

Figura 27

Atención:
Si no se ha presionado el botón *Atrás* o se ha llegado al último contenido visitado, el botón *Adelante* no tendrá efecto.

2.8.2 Inicio

Presionando el botón *Inicio* (figura 28), el usuario podrá volver a visualizar el contenido que se cargó en pantalla en el momento de solicitar la ayuda.

Figura 28

2.8.3 Imprimir

Presionando el botón *Imprimir* (figura 29), el sistema le permitirá al usuario imprimir el contenido de ayuda que está visualizando.

Figura 29

2.8.4 Sincronizar

Presionando el botón *Sincronizar* (figura 30), el sistema se posicionará en el nodo de la *Tabla de Contenido* correspondiente al contenido de ayuda que el usuario está visualizando.

Figura 30

2.8.5 Asistente

El sistema podría contar con íconos adicionales en el menú, específicos del Sistema de Información Federado. Un ejemplo de esto es el *Asistente*. Éste sería invocado en aquellos casos en los que el usuario no encuentre la información buscada o no sepa cómo utilizarla, y se basaría en un conjunto de guías o ayudas que muestren la información ofrecida. En la figura 31 se presenta un ejemplo de los datos que podría ofrecer el *Asistente*.

Atención:
Seleccionar el *Asistente* o invocar el *Wizard* generan el mismo resultado.

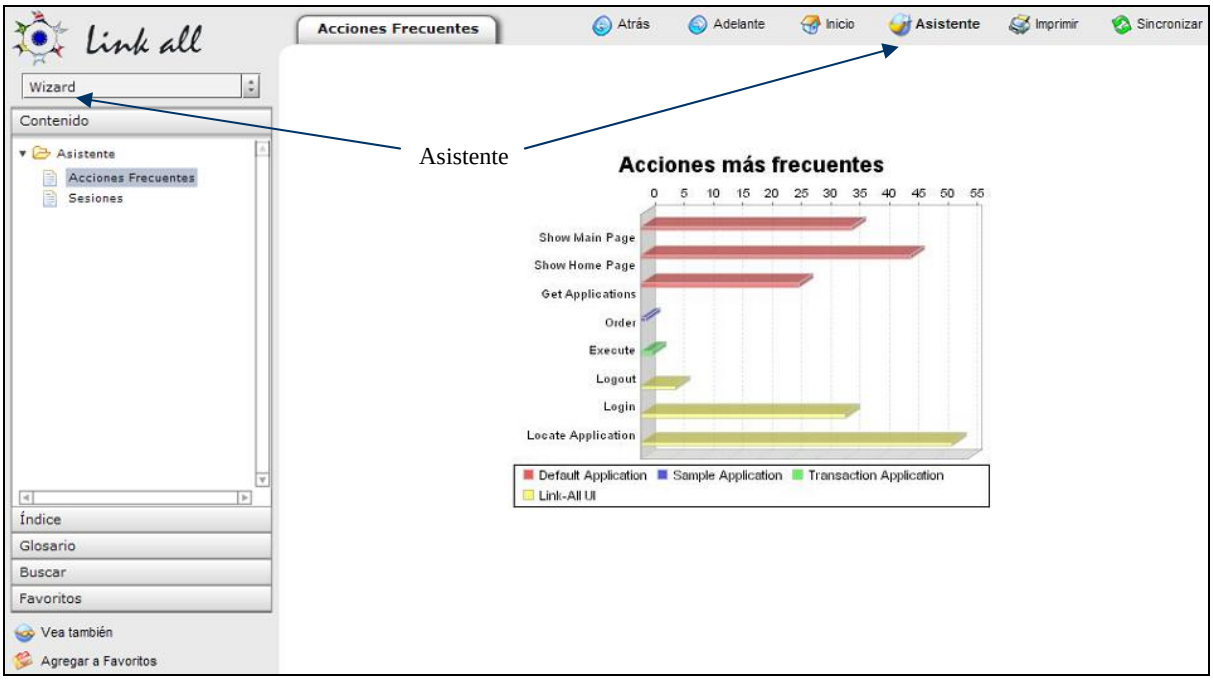


Figura 31

Mecanismo de Ayuda Contextual para Sistemas de Información Federados

Anexo
Manual de Desarrollo e Instalación

INCO – Facultad de Ingeniería – UDELAR

Montevideo, Junio 2005

Tutores:

Ing. Federico Piedrabuena
Dr. Ing. Raúl Ruggia

Estudiantes:

Laura González
Guillermo Roldós
Flavia Serra

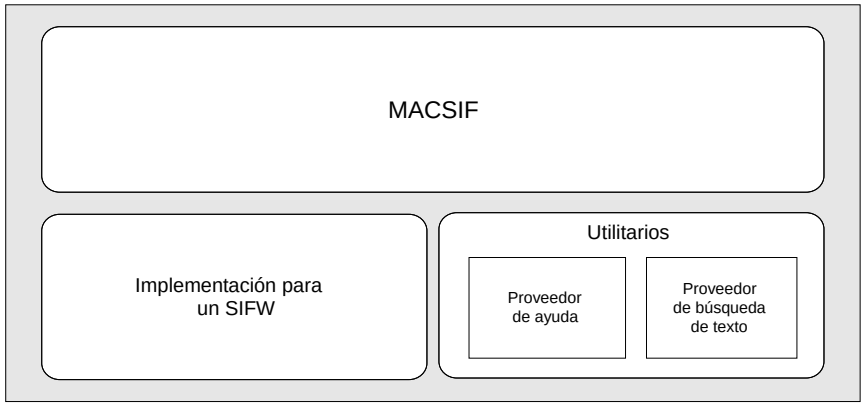
ÍNDICE

1 INTRODUCCIÓN

En este documento se indica el procedimiento necesario para realizar una implementación específica del Framework MACSIF, así como para implementar proveedores de ayuda o de búsqueda de texto. Si bien la integración del framework con la plataforma y con los proveedores no es compleja, es necesario conocer las operaciones que se deben desarrollar para que MACSIF funcione correctamente.

2 DESCRIPCIÓN GENERAL DE MACSIF

MACSIF es un framework que permite proveer funcionalidades de ayuda en línea a las aplicaciones que integran un Sistema de Información Federado a través de la Web. Si bien provee funcionalidades de ayuda de alto nivel, necesita interactuar con algunos componentes de las capas inferiores para obtener la información que va a presentar al usuario. Esta interacción con el Sistema de información federado y con los diferentes proveedores, se observa en el siguiente diagrama:



Al ser MACSIF un framework desarrollado en Java, esta integración se da a través de interfaces que deben ser implementadas y de clases abstractas que deben ser extendidas. El propósito de este documento es describir estas interfaces y clases abstractas, así como las operaciones que brindan cada una de ellas. En el Apéndice A se incluye una guía de instalación de MACSIF.

3 INTEGRACIÓN CON UN SISTEMA DE INFORMACIÓN FEDERADO

Del Sistema de información federado, es necesario obtener información acerca de los usuarios y sus permisos de acceso, de las aplicaciones que maneja, de las acciones que tienen lugar dentro de ellas, etc.

3.1 Interfaces a implementar

3.1.1 ctxhlp.context.IContext

Representa un contexto de sesión del usuario. La clase que implemente esta interfaz será creada con la primera solicitud de ayuda y enviada por cada solicitud, de forma que se pueda obtener todos los datos a partir de ella. Por tanto, debe dársele a esta clase la capacidad de conocer todos los datos de sesión necesarios para individualizarlas. La interfaz no tiene ninguna operación en particular. El desarrollador debe implementar las operaciones que crea necesarias de acuerdo a cada caso.

3.1.2 ctxhlp.context.factory.IContextFactory

Es utilizada para crear una instancia de la clase que implementa a *IContext*. Debe implementar la siguiente operación, la cual debe retornar un objeto del tipo de la clase mencionada:

```
public IContext getHelpContext(HttpServletRequest request,
                               HttpServletResponse response)
       throws ContextException;
```

3.1.3 ctxhlp.mapping.IContextId

Es utilizada para identificar la ubicación o la tarea que está ejecutando el usuario en el momento de solicitar la ayuda. Cada *contextId* identifica un contenido de ayuda principal, que se mostrará cuando el usuario solicite ayuda, y puede representar diferentes tipos de información de acuerdo a cada Sistema de Información

Federado. Sin embargo, es necesario que estos *contextId* reflejen la actividad de una aplicación, y puedan obtenerse de alguna forma cuando se solicita ayuda, ya que de eso depende qué ayuda se mostrará.

3.1.4 ctxhlp.favorite.IUser

Identifica un usuario del Sistema de Información Federado. Esta información será utilizada para gestionar los contenidos Favoritos.

3.1.5 ctxhlp.context.provider.ICurrentContextProvider

Es utilizada para obtener información de la actividad actual de la aplicación que se está ejecutando cuando el usuario solicita ayuda. Debe implementar las siguientes operaciones:

```
public Application getApplication(IContext context) throws ContextException;
```
Devuelve la aplicación que se está ejecutando en el momento de solicitar ayuda.

```
public String getLanguage(IContext context) throws ContextException;
```
Devuelve el lenguaje (o idioma) de la sesión.

```
public IContextId getContextId(IContext context) throws ContextException;
```
Devuelve el *contextId* que se está ejecutando dentro de la aplicación actual.

```
public IUser getUser(IContext context) throws ContextException;
```
Devuelve el usuario asociado a la sesión que solicitó ayuda.

3.1.6 ctxhlp.context.provider.ISecurityContextProvider

Es utilizada para obtener información acerca de los permisos de acceso que posee el usuario que solicita ayuda. De esta forma, se filtran los contenidos de ayuda que se muestran, y no se muestra ayuda sobre acciones o aplicaciones a las que el usuario no tiene acceso. Debe implementar las siguientes operaciones:

```
public Collection getAvailableApplications(IContext context)
                                throws ContextException;
```
Devuelve una colección de aplicaciones (de tipo *Application*) a las que el usuario tiene permisos de acceso.

```
public Collection getAvailableContextIds(String application, IContext context )
                                throws ContextException;
```
Devuelve una colección de *contextId* de la aplicación con el código recibido como parámetro, a las que el usuario tiene permisos de acceso.

```
public Collection getNotAvailableContextIds(String application, IContext context )
                                throws ContextException;
```
Devuelve una colección de *contextId* de la aplicación con el código recibido como parámetro, a las que el usuario no tiene permisos de acceso.

```
public boolean validateApplication(String application, IContext context)
                                throws ContextException;
```
Valida si el usuario conectado a la sesión actual tiene permisos sobre la aplicación con el código que se recibe como parámetro.

3.1.7 ctxhlp.context.provider.IRemoteContextProvider

Es utilizada para obtener información del contexto remoto en el caso de que la aplicación sobre la que se solicita ayuda no esté corriendo en el nodo local. Debe implementar las siguientes operaciones:

```
public boolean isLocal(IContext context) throws ContextException;
```
Devuelve *true* si la aplicación de la cual se solicita ayuda es local, y *false* si es remota.

```
public IContext getRemoteContext(IContext context ) throws ContextException;
```
Devuelve el contexto remoto en el caso que la aplicación no sea local.

```
public String getRemoteHost(IContext context ) throws ContextException;
```
Devuelve el nombre del Host en el cual se ejecuta la aplicación de la cual se solicita ayuda.

3.1.8 ctxhlp.context.provider.factory.IContextProviderFactory

Es utilizada para crear instancias de las clases que implementan a *ICurrentContextProvider* y *ICurrentContextProvider*. Debe implementar las siguientes operaciones:

```
public ICurrentContextProvider getCurrentContextProvider()
```

```
                                    throws ContextException;

    public ISecurityContextProvider getSecurityContextProvider()
                                    throws ContextException;

    public IRemoteContextProvider getRemoteContextProvider()
                                    throws ContextException;
```

3.1.9 ctxhlp.mapping.IMapping

Debe implementarse una vez por cada proveedor de ayuda (*HelpProvider*, ver capítulo siguiente), que se disponga en esta implementación. Permite relacionar los *contextId* con los contenidos de ayuda del *HelpProvider*. Debe implementar la siguiente operación:

```
public Collection getContentCodes(String application, String language,
                                  IContextId contentId) throws MappingException;
```

Dado un *contextId* para una determinada aplicación en un determinado idioma, devuelve una colección de los contenidos (de tipo *Content*), correspondientes.

3.1.10 ctxhlp.favorite.IFavorites

Es utilizada para gestionar los contenidos Favoritos de un usuario (esta es una funcionalidad provista por MACSIF), por lo que debe implementar las siguientes operaciones:

```
public Collection getFavorites(IUser user) throws FavoritesException;
```

Devuelve una colección de los contenidos favoritos de un usuario (de tipo *Content*). Es tarea de cada Sistema de Información Federado la gestión de estos favoritos.

```
public void addToFavorites(String application, String language, String content,
                           String provider, String title, IUser user)
                                    throws FavoritesException;
```

Agrega en la lista de favoritos de un usuario, un contenido (de tipo *Content*) con los datos recibidos como parámetro. Si el contenido ya existe, no lo agrega.

```
public void removeFromFavorites(String application, String language,
                                String content, String provider, IUser user)
                                    throws FavoritesException;
```

Borra de la lista de favoritos de un usuario, el contenido (de tipo *Content*) con los datos recibidos como parámetro. Si no existe, no borra nada.

3.2 Clases abstractas a extender

3.2.1 ctxhlp.application.Application

Representa una aplicación dentro del Sistema de información federado. Está identificada con un código único (de tipo String) y posee un nombre (de tipo String), e incluye las operaciones getter y setter correspondientes a estos atributos:

```
public final String getCode();
public final void setCode (String code);

public final String getName();
public final void setName(String name);
```

3.3 Archivo de Configuración

Para el correcto funcionamiento de MACSIF, es necesario modificar el archivo de configuración para definir los valores de ciertos parámetros:

```
DEFAULT.language=
```

Idioma por defecto de la ayuda, expresado en el estándar internacional de dos letras.

```
CONTEXT.factory=
CONTEXT.provider.factory=
```

Nombres completos de las clases que implementan las interfaces *IContextFactory* y *IContextProviderFactory* respectivamente.

```
PROVIDER.1=
```

```
PROVIDER.2=
...
```
Nombres completos de las clases que implementan el Session Bean de cada *HelpProvider*. Estos EJB son invocados vía JNDI.

```
PROVIDER.jndi.server.1=
PROVIDER.jndi.server.2=
...
```
Dirección y puerto del servidor JNDI donde se ejecuta cada *HelpProvider*.

```
PROVIDER.jndi.name.1=
PROVIDER.jndi.name.1=
...
```
Nombre JNDI de cada *HelpProvider*.

```
PROVIDER.mapping.1=
PROVIDER.mapping.2=
...
```
Nombres completos de las clases que implementan la interfaz *IMapping* para cada *HelpProvider*.

```
PROVIDER.info.1=
PROVIDER.info.2=
...
```
Nombres completos de las clases que implementan la interfaz *IHelpProviderInfo* para cada *HelpProvider*.

4 DESARROLLO DE UN PROVEEDOR DE AYUDA

De un proveedor de ayuda (*HelpProvider*), es necesario obtener todos los datos que conforman la ayuda de las aplicaciones en los diferentes idiomas. Estos *HelpProviders* deben estar implementados como Session Beans. La forma en que estos proveedores registran y obtienen esta información es transparente para MACSIF, siempre y cuando provean todas las operaciones que se detallan a continuación.

4.1 Interfaces a implementar

4.1.1 ctxhlp.help.content.table.IContentTable

Representa la tabla de contenido de la ayuda. Ésta tiene una estructura en forma de árbol jerárquico, que se mostrará de forma que el usuario navegue en los contenidos. Puede contener carpetas y sub-carpetas, las cuales pueden tener un contenido asociado o no. Las operaciones que provee son las siguientes:

```
public INode getRootNode() throws HelpException;
```
Devuelve el nodo raíz del árbol de contenido.

```
public Collection getChilds(INode node) throws HelpException;
```
Devuelve una colección de los hijos directos (de tipo *INode*), del que se recibe como parámetro.

```
public boolean isLeaf(INode node) throws HelpException;
```
Devuelve *true* si el nodo recibido es una hoja, o *false* si es una carpeta (aunque se encuentre vacía).

```
public void removeNode(INode node) throws HelpException;
```
Elimina el nodo recibido como parámetro de la estructura.

4.1.2 ctxhlp.help.content.table.INode

Representa cada uno de los nodos de la tabla de contenido. Las operaciones que provee son las siguientes:

```
public String getTitle() throws HelpException;
```
Devuelve el título del nodo (que se mostrará en el árbol).

```
public String getContent() throws HelpException;
```
Devuelve el código del contenido asociado al nodo (si existe).

4.1.3 ctxhlp.help.index.IIndex

Representa el índice de palabras clave de la ayuda. Estas palabras clave (representadas por la interfaz *IIndexEntry*), pueden tener cero o más contenidos asociados. Las operaciones que provee son las siguientes:

```
public Collection getEntries() throws HelpException;
```
Devuelve la colección de entradas de índice (objetos de tipo *IIndexEntry*) completa.

```
public Collection getContents(IIndexEntry entry) throws HelpException;
```
Devuelve la colección de los contenidos (de tipo *Content*), asociados a una palabra clave.

```
public void removeContent(IIndexEntry entry, Content content)
                          throws HelpException;
```
Elimina el contenido indicado para la palabra clave indicada.

```
public void removeContent(Content content) throws HelpException;
```
Elimina el contenido indicado para todas las palabras claves que lo contengan.

4.1.4 ctxhlp.help.index.IIndexEntry

Representa cada una de las entradas (o palabras clave) del índice de la ayuda. La única operación que provee es la siguiente:

```
public String getName() throws HelpException;
```
Devuelve el nombre de la entrada de índice, o sea, la palabra clave.

4.1.5 ctxhlp.help.glossary.IGlossary

Representa el conjunto de los términos del glosario de la ayuda. Esta funcionalidad simula un diccionario de ciertos términos que pueden resultar desconocidos para el usuario. La única operación que provee es la siguiente:

```
public Collection getTerms() throws HelpException;
```
Devuelve la colección de términos (de tipo *IGlossaryTerm*), del glosario.

```
public String getDefinition(IGlossaryTerm term) throws HelpException;
```
Devuelve la definición de un término del glosario.

4.1.6 ctxhlp.help.glossary.IGlossaryTerm

Representa cada uno de los términos del glosario de la ayuda. La única operación que provee es la siguiente:

```
public String getName() throws HelpException;
```
Devuelve la definición del término.

4.1.7 ctxhlp.help.provider.info.IHelpProviderInfo

Esta interfaz se pasa como parámetro en cada solicitud al *HelpProvider*, por lo que puede ser utilizada para conservar valores que sean necesarios. No contiene ninguna operación ya que únicamente sirve al proveedor.

4.1.8 ctxhlp.help.provider.IHelpProvider

Contiene todas las operaciones que se solicitarán al *HelpProvider* por parte de MACSIF. Las operaciones que provee son las siguientes:

```
public Help getHelp(String platform, String application, String language,
                    IHelpProviderInfo info)
             throws HelpException, HelpNotFoundException, RemoteException;
```
Es utilizada en el primer pedido de ayuda. Devuelve el objeto Help con los siguientes datos cargados: aplicación, lenguaje, proveedor, contenido actual, y titulo de la aplicación.

```
public IContentTable getContentTable(String platform, String application,
                    String language, IHelpProviderInfo info)
             throws HelpException, RemoteException;
```
Devuelve la tabla de contenido de la ayuda representada por los parámetros que recibe.

```
public IIndex getIndex(String platform, String application, String language,
                    IHelpProviderInfo info) throws HelpException, RemoteException;
```
Devuelve el índice de la ayuda representada por los parámetros que recibe.

```
public IGlossary getGlossary(String platform, String application, String language,
                    IHelpProviderInfo info) throws HelpException, RemoteException;
```
Devuelve el glosario de la ayuda representada por los parámetros que recibe.

```
public Collection getRelatedContents(String platform, String application,
                    String language, String contentCode, IHelpProviderInfo info)
              throws HelpException, RemoteException;
```

Devuelve una colección de los contenidos relacionados (de tipo *Content*) del contenido representado por los parámetros que recibe.

```
public Content getContent(String platform, String application, String language,
                    String content, IHelpProviderInfo info)
              throws HelpException, RemoteException;
```

Devuelve el contenido de la ayuda representada por los parámetros que recibe.

```
public Collection getAllContents(String platform, String application,
                    String language, IHelpProviderInfo info)
              throws HelpException, RemoteException;
```

Devuelve una colección de todos los contenidos (de tipo *Content*) de la ayuda representada por los parámetros que recibe.

```
public Collection getAllApplicationContents(String platform, String language,
                    IHelpProviderInfo info) throws HelpException, RemoteException;
```

Devuelve una colección de todos los contenidos (de tipo *Content*) de todas las aplicaciones de la plataforma para el idioma recibido como parámetro.

4.2 Clases abstractas a extender

4.2.1 ctxhlp.help.Help

Agrupa toda la información de ayuda. Los métodos getter y setter que posee son los siguientes:

```
public final String getApplication();
public final void setApplication(String application);

public final String getLanguage();
public final void setLanguage(String language);

public final String getProvider();
public final void setProvider(String provider);

public final String getTitle();
public final void setTitle(String title);

public final IContentTable getContentTable();
public final void setContentTable(IContentTable contentTable);

public final IGlossary getGlossary();
public final void setGlossary(IGlossary glossary);

public final IIndex getIndex();
public final void setIndex(IIndex index);

public final Collection getAvailableContents();
public final void setAvailableContents(Collection availableContents);

public final String getCurrentContent();
public final void setCurrentContent(String currentContent);
```

4.2.2 ctxhlp.help.content.Content

Contiene la información necesaria para gestionar un contenido de ayuda. Para MACSIF, un contenido puede estar representado en diferentes formatos: Html, XML, PDF, Word, etc., incluso puede ser generado dinámicamente. Las operaciones ya implementadas en la clase abstracta son las siguientes:

```
public final String getApplication();
public final void setApplication(String application);

public final String getCode();
public final void setCode(String code);

public final String getLanguage();
```

```
public final void setLanguage(String language);

public final String getProvider();
public final void setProvider(String provider);

public final String getTitle();
public final void setTitle(String title);

public String getTextContent();
public void setTextContent(String textContent);
```

La siguiente operación debe ser implementada por la clase del *HelpProvider* que extienda *Content*:

```
public abstract void show(HttpServletRequest request,
                          HttpServletResponse response) throws HelpException;
```

Esta operación es invocada directamente por el servidor Web, para mostrar el resultado de ella en el frame de contenido de la ayuda. Esto permite que este contenido pueda ser cualquier objeto que un browser pueda interpretar.

5 DESARROLLO DE UN PROVEEDOR DE BÚSQUEDA

De un proveedor de búsqueda de texto (*SearchProvider*), es necesario obtener los servicios necesarios de indexado de documentos y búsqueda dentro de los índices. Estos *SearchProviders* deben estar implementados como Session Beans. La forma en que estos proveedores realizan estas tareas es transparente para MACSIF, siempre y cuando provean todas las operaciones que se detallan a continuación.

5.1 Interfaces a implementar

5.1.1 ctxhlp.search.provider.ISearchProvider

Contiene las operaciones que se solicitarán al *SearchProvider* por parte de MACSIF. Las operaciones que provee son las siguientes:

```
public Collection searchText(String platform, String application, String language,
                             String text) throws SearchException, RemoteException;
```

Devuelve una colección de contenidos (de tipo *Content*) en los que encuentra el texto que se busca. La búsqueda se realiza para un determinado lenguaje, aunque la aplicación y la plataforma pueden estar vacíos (*null*). En este caso, no se toma en cuenta el parámetro (por ejemplo, si el parámetro *application* es *null*, se devuelven los contenidos de todas las aplicaciones).

```
public void generateIndex(String platform, String language, Collection contents)
                    throws SearchException, RemoteException;
```

Regenera los índices de búsqueda a partir de una colección de contenidos (de tipo Content). Esta operación será llamada periódicamente para que los índices se mantengan actualizados, por lo que es posible que algún usuario esté consultando en el momento en que se regeneran los índices. Esta capacidad debe ser tenida en cuenta por el *SearchProvider*.

APÉNDICES

A. INSTALACIÓN DE MACSIF

En esta sección se brinda la información necesaria para la instalación de MACSIF. En primer lugar se mencionan los archivos que forman parte de la distribución del producto, luego se indican los requerimientos en cuanto a software base, y finalmente se enumeran los pasos necesarios para su correcta instalación.

a. Archivos Incluidos

A continuación se enumeran y describen los archivos incluidos en la distribución:

- distribucion
  - ctxhlp.ear – Archivo de distribución del producto
  - ctxhlp.jar – Librería cliente necesaria para el desarrollo
- documentos
  - javadocs.zip – Javadocs de la librería cliente
  - guia_de_desarrollo.pdf – Manual de Desarrollo

b. Requerimientos de Software

El software necesario para ejecutar el MACSIF es el siguiente:

- Java Developer Kit (JDK) 1.4.2 o superior
  (disponible en http://java.sun.com/downloads/)

- JBoss Application Server (JBoss AS) 3.2.3 o superior
  (disponible en http://www.jboss.org/)

c. Proceso de instalación

Para la correcta instalación del MACSIF es necesario seguir los siguientes pasos:

- Verificar la correcta instalación del JDK y JBoss AS
- Modificar el archivo de configuración según lo explicado en la sección 3.3
- Copiar el archivo ctxhlp.ear a la carpeta %JBOSS_HOME%/server/default/deploy (donde JBOSS_HOME es el directorio base del JBoss AS)

Nota: Para que MACSIF funcione correctamente es necesario que las clases implementadas según lo explicado en las secciones 3.1 y 3.2, también estén instaladas correctamente en el JBoss AS.