DISEÑO E IMPLEMENTACIÓN DE UN CONTEXTO DE DESARROLLO PARA APLICACIONES BASADAS EN JAVA SOBRE ENTORNOS DISTRIBUIDOS (AJED –Framework)

CARLOS NORBERTO SANABRIA BERNAL
CHRISTIAN ALBERTO TORRES MANIGUA

UNIVERSIDAD DEL MAGDALENA
FACULTAD DE INGENIERÍA
PROGRAMA INGENIERÍA DE SISTEMAS
SANTA MARTA, D.T.C.H.
2004

DISEÑO E IMPLEMENTACIÓN DE UN CONTEXTO DE DESARROLLO PARA APLICACIONES BASADAS EN JAVA SOBRE ENTORNOS DISTRIBUIDOS (AJED –Framework)

CARLOS NORBERTO SANABRIA BERNAL

CHRISTIAN ALBERTO TORRES MANIGUA

Trabajo de Memoria de Grado presentado para optar al título de Ingeniero de Sistemas

Directora
EIRA ROSARIO MADERA REYES
Ingeniera de Sistemas

UNIVERSIDAD DEL MAGDALENA

FACULTAD DE INGENIERÍA

PROGRAMA INGENIERÍA DE SISTEMAS

SANTA MARTA, D.T.C.H.

2004

Nota de aceptación

_____

_____

_____

_____

Presidente del Jurado

_____

Jurado

_____

Jurado

Santa Marta, Febrero de 2004

DEDICATORIAS

A mi familia, por la paciencia y cariño que me tuvieron en todo este tiempo.

A mi futura esposa, que estuvo siempre a mi lado, y que me apoyaba en todas mis ocurrencias.

Carlos

Al señor que guía mis pasos en cada meta propuesta.

Mi Madre por inspirarme el amor al aprendizaje y una sólida ética de trabajo.

A mis hermanos que con su apoyo y consejo me dieron el impulso en momentos difíciles.

A mis sobrinitos que me dieron su alegre compañía en momentos de estudio.

Christian.

AGRADECIMIENTOS

Los autores expresan su agradecimiento a:

La Universidad del Magdalena por ofrecernos la oportunidad de enriquecer nuestros conocimientos.

Ing. Eira Rosario Madera, Directora de Admisiones, Registro y Control Académico, por su gran ayuda profesional y confianza en la elaboración de este proyecto.

Ing. Emperatriz del Socorro Zapata Zapata, por enseñarnos que el algoritmo de la vida lo resolvemos nosotros mismos.

A todos los docentes de la Universidad del Magdalena, Universidad de Antioquia, Universidad Nacional de Colombia, Universidad Industrial de Santander y Universidad de los Andes que nos brindaron sus conocimientos para la culminación de nuestros estudios.

A nuestros amigos y compañeros de trabajo (Alcides, Jaime y Jhon) que por su entusiasmo y conocimientos se han convertido en amigos para toda la vida.

CONTENIDO

PRÓLOGO

La expansión de Internet ha generado nuevas tendencia en el desarrollo de soluciones corporativas e institucionales, en gran medida orientadas a la estandarización y ágil implementación de procesos (búsqueda, inserción, actualización y borrado de datos), que aprovechando el uso extendido de navegadores Web, han encontrado en este un cliente con óptimos resultados.

Este trabajo esta concebido como una herramienta de colaboración que sigue las nuevas tendencias de cambio como un aporte más al mejoramiento de los ambientes para los nuevos desarrolladores. Hoy gracias a la colaboración y apoyo de múltiples grupos de desarrollo y compañías que apoyan el software libre hemos emprendido la tarea de iniciar este trabajo, vigilando que su contenido este acorde con las nuevas tecnologías.

Hemos tratado de usar un lenguaje acorde con el contenido del presente trabajo. Por tal motivo se han simplificado algunos términos de forma explicativa, sin dejar de lado su connotación técnica.

El realizar este trabajo nos ha permitido profundizar conocimientos vinculados a nuestro entorno laboral y profesional, satisfaciendo nuestras ganas de aprender cada día más.

El propósito de este trabajo es ofrecer una herramienta que permita a los desarrolladores generar soluciones de software en corto tiempo y sin la necesidad de escribir una sola línea de código.

Por último nos queda decir que el apoyo recibido en las empresas fue un factor motivante para el desarrollo de este proyecto, haremos nuestro mejor esfuerzo para lograr la mayor aceptación por parte de las mismas, queremos agradecerles a todos aquellos que creyeron en que esto fuese posible y a todos los que tengan la oportunidad de leerlo puedan contribuir con sugerencias y críticas a mejorarlo.

Carlos Sanabria Bernal

Christian Torres Manigua

Santa Marta, 2004

INTRODUCCIÓN

El Word Wide Web Consortium (W3C) define el término "Web" como: "el universo de información accesible a través de la red". Una aplicación Web es un sistema que permite a un usuario final acceder a una parte de información contenida en el universo al que hace referencia la anterior definición del W3C. Las aplicaciones Web son aplicaciones basadas en el muy extendido paradigma "cliente/servidor". Este paradigma consiste en un servidor que sabe cómo proporcionar un servicio y un cliente que desea acceder a éste.

En el mundo actual las redes de computadores y las nuevas tecnologías en telecomunicaciones han abierto el camino para la transmisión rápida y confiable de la información. Las actuales aplicaciones, que en su mayoría se ejecutan en un mismo punto dificultando la distribución y, sobre todo, la actualización, pueden evolucionar hacia un ambiente distribuido que les permita aprovechar las infraestructuras que las soportan. En esta evolución, las nuevas herramientas software pueden jugar un papel fundamental logrando así obtener, al final, sistemas de calidad, oportunos y altamente competitivos. Para ello es necesario diseñar y desarrollar aplicaciones informáticas complejas que contemplen diversas funcionalidades como la personalización, la seguridad, el acceso a diversas bases de datos o la escalabilidad (capacidad de gestionar un incremento del número de solicitudes sin afectar negativamente los tiempos de respuesta).

AJED, mediante la vinculación de tecnologías Java como RMI (Remote Method Invocation) integrada en EJB's (Enterprise Java Beans), JDBC (Java Database Conectivity), servidores de páginas y otras tecnologías como XML (Extensible Markup

Language), dan una respuesta a todas estas necesidades, ofreciendo un marco moderno e integrado de desarrollo de software.

1. PLANTEAMIENTO DEL PROBLEMA

1.1 DESCRIPCIÓN

La interacción con el usuario final, el manejo de grandes cantidades de datos (generalmente almacenados en bases de datos), el suministro de servicios en función de la identificación del usuario y demás aspectos que han convertido a Internet en una plataforma de ejecución de aplicaciones, hacen que el desarrollo de las mismas sea muy complejo, provocando la presencia de un gran número de puntos críticos que deben considerarse.

La seguridad de las aplicaciones debe ser una premisa que debe estar presente en toda tarea de desarrollo, teniendo siempre en cuenta que cualquier cliente que haga uso del servicio puede ser un potencial atacante para el mismo.

Es fundamental disponer de una metodología de desarrollo estructurada y con una serie de objetivos claros, como son facilitar las tareas de revisión y modificación de código, actualizaciones y ampliaciones de los servicios proporcionados, permitiendo que en todo momento se tenga el control por parte del servidor de lo acontecido en la ejecución de las aplicaciones evitando así que puedan ejecutarse acciones imprevistas que den pie a la vulneración del servicio.

Algunos de los problemas comunes en el desarrollo de aplicaciones Web son:

- *Implementaciones poco robustas de mecanismos de autenticación o de mantenimiento de sesiones.*

- *Errores en el tratamiento de entrada de datos: el incorrecto procesamiento de éstos puede llevar a una malversación por parte del atacante.*
- *Ejecución de aplicaciones con máximos privilegios sobre el sistema: como consecuencia de esto un fallo en la aplicación puede comprometer completamente a la plataforma de ejecución.*

Tratamiento incorrecto de los posibles errores y los mensajes generados al producirse éstos.

1.2 FORMULACIÓN

El presente proyecto busca proporcionar un entorno de desarrollo de software integrado y eficiente, con el fin de disminuir los tiempos de producción de los desarrolladores.

En la implementación de este proyecto se integraron variadas tecnologías para solucionar problemas comunes inherentes a la vinculación de procesos interactivos y al modelamiento de gráficos (pantallas o formas con las cuales el usuario realiza sus operaciones) para hacer transparentes estos puntos críticos y complejos a los usuario.

Se propone AJED que representa el modelo que debe cumplir el entorno de desarrollo de este tipo de aplicaciones. AJED especifica las reglas que se deben tener en cuenta a la hora de definir una implementación de este modelo. Y como implementación del modelo de AJED, nació JForms un conjunto de EJBs y Servlets que basados en la definición de AJED proporcionan un entorno de trabajo para el desarrollo de aplicaciones distribuidas que requieren proporcionar una interfaz de usuario desde la cual se pueda modificar de muchas formas la información de una base de datos previamente definida.

2 OBJETIVOS

2.1 GENERAL

Crear un contexto de desarrollo de aplicaciones (framework) que permita automatizar gran parte de la creación de aplicaciones desarrolladas en Java para entornos distribuidos.

2.2 ESPECÍFICOS

- *Desarrollar un ambiente para aplicaciones independientes de las plataformas donde se ejecuten.*
- *Separar los agentes que interactúan en el modelo del negocio de los modelos diseñados como interfaz para los usuarios.*
- *Automatizar la creación de aplicaciones orientadas a ambientes distribuidos con un costo mínimo de tiempo.*
- *Aplicar un nuevo modelo de desarrollo de software para incrementar las capacidades de aplicaciones pensadas para Internet.*

3 JUSTIFICACIÓN

La solución de gran parte de los aspectos que complican y retrasan el desarrollo de aplicaciones en entornos distribuidos, con un enfoque tendiente a normalizar la comunicación entre clientes HTML y servidores de aplicaciones.

3.1 Uniformidad de las aplicaciones.

Mediante el modelo (Cliente - Servidor de Aplicaciones - Base de Datos) y la vinculación de tecnologías adicionales para la comunicación hacen que las implementaciones desarrolladas para tal estructura sean robustas y estables.

3.2 Portabilidad y compatibilidad de aplicaciones

Los programas creados son portátiles dentro de una red. El programa se compila en un código de bytes Java que puede ejecutarse en cualquier sitio de una red, en un servidor o cliente que tenga una máquina virtual de Java. La máquina virtual de Java interpreta el código de bytes y lo transforma en código que puede correr en el equipo real del ordenador. Esto significa que las diferencias individuales de las plataformas de los ordenadores, como la longitud de las instrucciones, pueden ser reconocidas y adaptadas localmente mientras el programa se ejecuta.

Esto asegura que nuestras aplicaciones sean compatibles con cualquier sistema operativo que soporte a una Máquina Virtual Java.

3.3 Alta productividad en el desarrollo.

Por medio del desarrollo de un nivel más compresible de gráficos, aplicando esquemas a los diseños XML muy al estilo HTML de una forma metódica y estructurada, sumándole la independencia de estos con relación a la función de los negocios que debe cumplir la aplicación, hacen posible que una aplicación para este tipo de entornos pueda desarrollarse en menos tiempo del estimado.

3.4 Seguridad.

El modelo planteado hereda el entorno de seguridad de Java que se conoce como modelo del patio de juegos (Sandbox model), aludiendo a esos rectángulos con arena donde se deja jugando a los niños pequeños, de manera que puedan hacer lo que quieran dentro del mismo, pero no puedan salir al exterior.

En concreto, este modelo se implementa mediante la construcción de cuatro barreras o líneas de defensa:

- *Primera línea de defensa: Características del lenguaje/compilador*
- *Segunda línea de defensa: Verificador de código de bytes*
- *Tercera línea de defensa: Cargador de clases*
- *Cuarta línea de defensa: Gestor de Seguridad*

Adicional a esto se tiene la posibilidad de integrar al ambiente protocolos de transferencia de datos como SSL (Security Socket Layer)

3.5 Fácil mantenimiento.

La estructura por capas del modelo planteado establece claros límites para la aplicación y debido a que su integración se hace al final del proceso podremos trabajar sobre cada capa sin tener de por medio otras y gracias al entorno distribuido se garantiza que el usuario tendrá la última versión de nuestra aplicación.

3.6 Economía de escala.

Las mejoras del mismo benefician a todas las aplicaciones (pasadas y futuras).

4 MARCO TEÓRICO

4.1 INTERNET

Empezó en los Estados Unidos de América en 1969, como un proyecto puramente militar. La Agencia de Proyectos de Investigación Avanzados de Defensa (DARPA) desarrolló una red de computadoras llamada ARPANET, para no centralizar los datos, lo cual permitía que cada estación de la red pudiera comunicarse con cualquier otra por varios caminos diferentes, además presentaba una solución para cuando ocurrieran fallas técnicas que pudieran hacer que la red dejase de funcionar.

Los sitios originales que se pusieron en red eran bases militares, universidades y compañías con contratos del Departamento de Defensa. Conforme creció el tamaño de esta red experimental, lo mismo sucedió con las precauciones por la seguridad. Las mismas redes usadas por las compañías y las universidades para contratos militares se estaban volviendo cada vez más accesibles al público

Como resultado, en 1984, ARPANET se dividió en dos redes separadas pero interconectadas. El lado militar fue llamado MILNET. El lado educativo todavía era llamado técnicamente ARPANET, pero cada vez se hizo mas conocida como Internet.

En mayo de 1995, entre 35 y 45 millones de personas usaban Internet y este número fue creciendo mes a mes en un 10 a 15%. Las estimaciones actuales colocan al número de personas en Internet en enero de 1997 en 62 millones de usuarios individuales.

Podemos decir que el resultado final es que lo que comenzó como un proyecto de investigación gubernamental y educativo, ahora se ha convertido en uno de los medios de comunicación más importante de la actualidad. Nunca antes había sido posible tener acceso a tantas personas de culturas y antecedentes tan variados.

4.2 WEB

La Web es una idea que se construyo sobre la Internet. Las conexiones físicas son sobre la Internet, pero introduce una serie de ideas nuevas, heredando las ya existentes.

Empezó a principios de 1990, en Suiza en el centro de investigación CERN (centro de Estudios para la Investigación Nuclear) y la idea fue de Tim Berners-Lee, que se gestó observando una libreta que él usaba para añadir y mantener referencias de cómo funcionaban los ordenadores en el CERN.

Antes de la Web, la manera de obtener los datos por la Internet era caótica: había un sinfín de maneras posibles y con ello había que conocer múltiples programas y sistemas operativos. La Web introduce un concepto fundamental: la posibilidad de lectura universal, que consiste en que una vez que la información esté disponible, se pueda acceder a ella desde cualquier ordenador, desde cualquier país, por cualquier persona autorizada, usando un único y simple programa. Para que esto fuese posible, se utilizan una serie de conceptos, el más conocido es el hipertexto.

Con Web los usuarios novatos podrían tener un tremendo poder para hallar y tener acceso a la riqueza de información localizada en sistemas de cómputos en todo el mundo.

Este solo hecho llevó un avance tremendo de Internet, un ímpetu tan grande que en 1993 World Wide Web creció un sorprendente 341000%, tres años después, en 1996, todavía sé esta duplicando cada 50 días.

4.3 URLs

Localizador Uniforme de Recursos URL (Uniform Resource Locator) es una dirección especial usada por los navegadores Web, para tener acceso a información en Internet. El URLs especifica el ordenador en que se hospeda, el directorio, y el nombre del fichero A través de estas direcciones o URLs vamos a poder conectar los diferentes objetos (no solo texto), aunque se acceda a ellos a través de diferentes protocolos. Una cualidad de los URLs es que permiten utilizar los datos ya existentes en la Internet (Wais, Gofher, ftp) y así es como consigue la Web envolver a la Internet sencilla y eficazmente

4.4 PROGRAMACIÓN ORIENTADA A OBJETOS.

La programación Orientada a objetos (POO) es una forma especial de programar, más cercana a como expresaríamos las cosas en la vida real que otros tipos de programación.

Pensar en términos de objetos es muy parecido a cómo lo haríamos en la vida real.

Java es un lenguaje de programación completamente orientado a objetos.

4.4.1 Clases en POO

Las clases son declaraciones de objetos, también se podrían definir como abstracciones de objetos. Esto quiere decir que la definición de un objeto es la clase. Cuando programamos un objeto y definimos sus características y funcionalidades en realidad lo que estamos haciendo es programar una clase.

4.4.2 Propiedades en clases

Las propiedades o atributos son las características de los objetos. Cuando definimos una propiedad normalmente especificamos su nombre y su tipo. Nos podemos hacer a la idea de que las propiedades son algo así como variables donde almacenamos datos relacionados con los objetos.

4.4.3 Métodos en las clases

Son las funcionalidades asociadas a los objetos. Cuando estamos programando las clases las llamamos métodos. Los métodos son como funciones que están asociadas a un objeto.

4.4.4 Objetos en POO

Los objetos son ejemplares de una clase cualquiera. Cuando creamos un ejemplar tenemos que especificar la clase a partir de la cual se creará. Esta acción de crear un objeto a partir de una clase se llama instanciar (que viene de una mala traducción de la palabra instace que en inglés significa ejemplar).

4.4.5 Estados en objetos

Cuando tenemos un objeto sus propiedades toman valores. El valor concreto de una propiedad de un objeto se llama estado.

4.4.6 Mensajes en objetos

Un mensaje en un objeto es la acción de efectuar una llamada a un método.

4.5 MODELO DE OBJETOS DISTRIBUIDOS EN JAVA

Objeto Remoto:

- *Objeto cuyos métodos pueden invocarse desde otras Máquinas Virtuales*
- *Descrito por una o más Interfaces Remotas (las que implementa) en las que se declaran los métodos que pueden ser invocados por objetos desde otras VMs*

4.6 ENTORNO DISTRIBUIDO

4.6.1 Sistema distribuido

- *Modelo físico: conjunto de nodos (procesadores sin memoria ni reloj común) conectados por una red.*
- *Modelo lógico: conjunto de procesos que ejecutan se concurrentemente en uno o más computadores que colaboran y se comunican intercambiando mensajes.*

AJED es un sistema distribuido de procesos, es decir, los procesos de la aplicación no residen en una sola máquina, ya que la idea principal es delegar los procesos a los diferentes puntos que entran en juego:

- *El servidor, ejecuta todas las operaciones necesarias para la conectividad con la base de datos, y atender y devolver los resultados.*
- *El cliente, maneja toda la interactividad con el usuario, las acciones que solicita sobre los componentes de una interfaz.*

En las aplicaciones Web es necesario que además del cliente se tenga el servidor que atiende las peticiones HTTP *del cliente, y como este es el caso de JForms tenemos además de los procesos anteriores otro proceso que es el Servlet o JSP que atiende dichas peticiones, y este es el encargado de transformar estas peticiones a peticiones al servidor de JForms. Este* servidor *puede residir en la misma maquina que el servidor de AJED.*

4.6.1.1 Un proceso es un programa en ejecución.

4.6.1.2 Un protocolo es un conjunto de reglas e instrucciones que gobiernan la comunicación en un sistema distribuido, es decir, el intercambio de mensajes.

4.7 MÁQUINA VIRTUAL (JVM - Java Virtual Machine)

La solución que propone Java es que el programa no corra sobre el hardware real, sino que corra sobre una (maquina virtual). Es decir, dicha maquina no existe sino que es simulada por medio de software. Cada plataforma tendría instalada esta maquina virtual (que es un programa) y a la hora de correr los programas esta maquina virtual los

correría. La maquina virtual actuaría como una forma de interprete, el cual va leyendo el programa y ejecutándolo instrucción por instrucción sobre el hardware real.

4.8 XML (Extensible Markup Language)

XML es un metalenguaje, esto es, un lenguaje para definir lenguajes. Está basado en el anterior estándar SGML (Standard Generalized Markup Language, ISO 8879), que data de 1986, pero que empezó a gestarse desde principios de los años 70. Éste no es más que un modelo de objetos (en forma de API) que permite acceder a las diferentes partes que pueden componer un documento XML o HTML.

Los alcances de XML van más allá de su aplicación en Internet, más que eso se propone como lenguaje de bajo nivel (a nivel de aplicación, no de programación) para intercambio de información estructurada entre diferentes plataformas. Se puede usar en bases de datos, editores de texto, hojas de cálculo, y casi cualquier cosa que podamos pensar.

XML juega un papel importantísimo en AJED y por lo tanto en JForms. Sobre XML se basa primero la comunicación entre el servidor de AJED y el cliente (este cliente no necesariamente es una interfaz de usuario y no necesariamente reside en la maquina del usuario, puede tratarse de un Servlet, o de una aplicación de escritorio). Es decir que las peticiones que se hagan al servidor de AJED están en formato XML, por medio de este se dice que acción debe realizar y los parámetros que dicha acción necesita, por ejemplo si se quiere insertar un registro en alguna una tabla de la base de datos el XML que se envía lleva la información necesaria para que el servidor de AJED encuentre dicha tabla, y además lleva la información que se va a insertar. Como segundo papel del XML en AJED esta la definición de los formularios que representan la interfaz y las acciones que se

pueden ejecutar, es decir, el desarrollador de aplicaciones no tiene que escribir código para definir la interfaz del usuario, o las operaciones que se pueden ejecutar desde dicha interfaz.

4.9 XSL (Extensible Stylesheet Language)

El XSL es un lenguaje que nos permite definir una presentación o formato para un documento XML. Un mismo documento XML puede tener varias hojas de estilo XSL que lo muestren en diferentes formatos (HTML, PDF, RTF, VRML, PostScript, sonido, etc.).

Una hoja de estilo XSL es una serie de reglas que determina cómo va a ocurrir la transformación. Cada regla se compone de un patrón (pattern) y una acción o plantilla (template).

De este modo, cada regla afecta a uno o varios elementos del documento XML. El efecto de las reglas es recursivo, para que un elemento situado dentro de otro elemento pueda ser también transformado. Las hojas de estilo tienen una regla raíz que, además de ser procesada, llama a las reglas adecuadas para los elementos hijos.

Una intención inicial al momento de diseñar JForms era que se valiera de XSL para transformar los formularios definidos en XML a código Java, y así tal vez el código que se genera podría ser mas eficiente que como esta en su forma actual que es el mismo código para manejar todos los XML. Esto también tiene una desventaja y por eso no utilizamos XSL y es que si es necesario transformar los XML que definen los formularios a código Java, también es necesario que cada vez que se agregue un nuevo formulario sea necesario volver a compilar la aplicación, y cada vez se tendrían mas y mas nuevas clases. Pero esto

no es razón suficiente, así que queda par futuros trabajos sobre implementaciones de AJED.

4.10 JAVA WEB START (Tecnología Java)

Java Web Start es una solución de distribución de aplicaciones basada en tecnología Java 2TM. Es la canalización entre Internet y el sistema que permite al usuario ejecutar y gestionar aplicaciones desde la Web. Java Web Start proporciona una activación fácil y rápida de las aplicaciones con un único clic y garantiza la ejecución de la última versión de la aplicación, eliminando los complicados procesos de instalación o de modernización.

- *Funcionamiento*

  Las aplicaciones hechas para esta aplicación se encontrarán en servidores Web y podrán ejecutarse de tres formas distintas:

  - *Desde un navegador de Web, haciendo clic en un enlace.*
  - *Desde el Application Manager (Gestor de aplicaciones) incorporado en Java Web Start, que efectúa el seguimiento de las aplicaciones utilizadas recientemente y proporciona un acceso rápido a sus aplicaciones favoritas.*
  - *Desde los iconos del escritorio o del menú Inicio (sólo en Microsoft Windows).*

  Java Web Start se ejecuta sobre una máquina virtual java como una aplicación de ventanas hecha con swing.

Ejecutando la aplicación Java Web Start, se nos mostrará una lista con las aplicaciones y se nos dará la opción de ejecutarlas.

- *Seguridad*

  Java Web Start está construido sobre la plataforma Java 2, que proporciona una amplia arquitectura de seguridad. Las aplicaciones ejecutadas con Java Web Start se ejecutarán de forma predeterminada en un entorno restringido ("zona protegida") con acceso limitado a los archivos y a la red. Por tanto, la ejecución de aplicaciones mediante Java Web Start mantiene la seguridad e integridad de los sistemas.

  Una aplicación puede solicitar acceso sin restricciones al sistema. En tal caso, Java Web Start mostrará un cuadro de diálogo Advertencia de seguridad cuando se ejecute la aplicación por primera vez. La advertencia mostrará información acerca del proveedor que ha desarrollado la aplicación. Si elige confiar en dicho proveedor, la aplicación se ejecutará. La información acerca del origen de la aplicación se basa en la firma de código digital.

Java Web Start es otro posible cliente para el servidor de AJED. Pero tal vez a de pasar algún tiempo mientras se abandonan un poco las interfaces en HTML para migrarlas a interfaces Java.

4.11 RMI (Tecnología Java)

Las aplicaciones RMI normalmente comprenden dos programas separados: un servidor y un cliente. Una aplicación servidor típica crea objetos remotos, hace accesibles unas referencias a dichos objetos remotos, y espera a que los clientes llamen a estos métodos u objetos remotos. Una aplicación cliente típica obtiene una referencia remota de uno o más objetos remotos en el servidor y llama a sus métodos. RMI comunica al cliente y al servidor por medio de un mecanismo de comunicación llamado JRMP (Java Remote Method Protocol) que es un protocolo estándar de bajo nivel basado en TCP (Transfer Control Protocol).

Cuando es una aplicación algunas veces nos referimos a ella como Aplicación de Objetos Distribuidos.

RMI es el otro elemento cuyo papel es definitivo en JForms. Ya que los EJBs se basan en RMI, y como son los EJBs con los que JForms proporciona el servidor de AJED.

4.11.1 Las aplicaciones de objetos distribuidos necesitan.

- *Localización Objetos Remotos*

  Las aplicaciones pueden utilizar uno de los dos mecanismos para obtener referencias a objetos remotos. Puede registrar sus objetos remotos con la facilidad de nombrado de RMI rmiregistry (Tecnología Java). O puede pasar y devolver referencias de objetos remotos como parte de su operación normal.

- *Comunicación con Objetos Remotos*

Los detalles de la comunicación entre objetos remotos son manejados por el RMI; para el programador, la comunicación remota se parecerá a una llamada estándar a un método Java.

- *Cargar Bytecodes (Código de Bytes) para objetos que son enviados.*

  Como RMI permite al llamador pasar objetos Java a objetos remotos, RMI proporciona el mecanismo necesario para cargar el código del objeto, así como la transmisión de sus datos.

4.11.2 Ventajas de la Carga Dinámica de Código

Una de las principales y únicas características de RMI es la habilidad de descargar los bytecodes (o simplemente, código) de una clase de un objeto si la clase no está definida en la máquina virtual del recibidor. Los tipos y comportamientos de un objeto, anteriormente sólo disponibles en una sola máquina virtual, ahora pueden ser transmitidos a otra máquina virtual, posiblemente remota. RMI pasa los objetos por su tipo verdadero, por eso el comportamiento de dichos objetos no cambia cuando son enviados a otra máquina virtual. Esto permite que los nuevos tipos sean introducidos en máquinas virtuales remotas, y así extender el comportamiento de una aplicación dinámicamente.

4.11.3 Interfaces, Objetos y Métodos Remotos

Una aplicación distribuida construida utilizando RMI de Java, al igual que otras aplicaciones Java, está compuesta por interfaces y clases. Los interfaces definen métodos, mientras que las clases implementan los métodos definidos en los interfaces y, quizás, también definen algunos métodos adicionales. En una aplicación distribuida, se asume que algunas implementaciones residen en diferentes máquinas virtuales. Los objetos que tienen métodos que pueden llamarse por distintas máquinas virtuales son los objetos remotos.

Un objeto se convierte en remoto implementando un interfaz remoto, que tenga estas características.

- *Un interfaz remoto.*
- *Cada método del interfaz declara que "lanza" un controlador de error (Excepción de tipo remota) además de cualquier excepción específica de la aplicación.*

El RMI trata a un objeto remoto de forma diferente a como lo hace con los objetos no-remotos cuando el objeto es pasado desde una máquina virtual a otra. En vez de hacer una copia de la implementación del objeto en la máquina virtual que lo recibe, RMI pasa una representación del objeto remoto. El llamador invoca un método en la representación del objeto remoto local que es responsable de llevar a cabo la llamada al objeto remoto.

4.12 SERIALIZACIÓN DE OBJETOS JAVA (Tecnología Java)

La serialización de un objeto consiste en obtener una secuencia de bytes que represente el estado de dicho objeto. Esta secuencia puede utilizarse de varias maneras (puede enviarse

a través de la red, guardarse en un fichero para su uso posterior, utilizarse para recomponer el objeto original, etc.).

- *Serialización en RMI*

  En RMI, la serialización se utiliza de forma casi transparente al usuario. Concretamente, se utiliza en el paso de parámetros y retorno de valores de las invocaciones a métodos de objetos remotos.

4.13 CLIENTE / SERVIDOR

Las arquitecturas cliente/servidor y las ahora llamadas arquitecturas multinivel son la base de la mayoría de los sistemas distribuidos que existen en la actualidad, los WebServices y WebDAV son algunas de las tecnologías que soportan estos sistemas. A continuación una descripción de cada una.

4.14 WEBSERVICES

Los Web Services son aplicaciones que pueden ser accedidas mediante un URL tanto en Internet como dentro de una Intranet. Su propósito principal es facilitar la integración de diversas aplicaciones tanto dentro de la organización como fuera de ella, pueden operar de una manera bajamente acoplada, es decir, sus componentes pueden ser cambiados fácilmente en tiempo de ejecución. Gracias a que los Web Services se describen a si mismos mediante el archivo WDSL (Web Services Description Language), y son publicados

dentro de servidores de registro UDDI (Universal Discovery, Description and Integration), facilitan la tarea de encontrar, usar e integrar servicios.

Gracias a los Web Services es posible hablar en este momento de Arquitecturas SOA (Service-Oriented Architectures), en las que el desarrollo de una aplicación ya no está dado únicamente en la necesidad de solucionar un conjunto de requerimientos funcionales; ahora también se tiene que pensar en la manera en que dicha aplicación puede brindar servicios tanto dentro, como fuera de los límites de la organización.

Los Web Services usan para su comunicación un protocolo basado en XML llamado SOAP (Simple Object Access Protocol), que define de manera estándar el esquema en que se hacen los requerimientos a uno de estos servicios y la forma en que estos deben retornar los mensajes. De esta manera se logra desacoplar los lenguajes de implementación y comunicación, facilitando la tarea de integrar aplicaciones desarrolladas en diferentes plataformas y lenguajes. Debido a que existe un amplia gama de aplicaciones que no necesitan el nivel de detalle que provee la especificación SOAP, posteriormente se creó un subconjunto de la misma llamada XML-RPC, que hace más fácil la implementación de Web Services de pequeña escala.

Un Web Services (WS) es una aplicación que puede ser descripta, publicada, localizada, e invocada a través de una red, generalmente Internet. Los WS son a los efectos del consumidor componentes que se encuentran dentro de una caja negra, que pueden ser utilizados sin preocuparse de como fueron implementados, y ser accedidos utilizando XML (SOAP), generalmente sobre HTTP.

La arquitectura básica del modelo de WS describe un consumidor, un proveedor y ocasionalmente un corredor (broker) y relacionados con estos agentes está las operaciones

de publish (publicar), find (encontrar) y bind (enlazar).

Los requerimientos a la hora de desarrollar o consumir un WS son:

- *una forma estándar de representar los datos: XML*
- *un formato común y extensible de mensaje: SOAP*
- *un lenguaje común y extensible para describir los servicios: WSDL (basado en XML).*
- *una forma de descubrir los servicios en Internet: UDDI (basado en XML).*

Una vasta área de uso de XML viene dada por la necesidad de interoperabilidad, permitiendo a las aplicaciones comunicarse unas con otras de una forma estándar. En efecto XML se ha vuelto la pieza común y de mejor entendimiento de la comunidad de diseñadores de software. De aquí en adelante nos referiremos a la comunicación Cliente - Servidor. El proceso servidor, del requerimiento cliente, generalmente retornará un valor como resultado de su proceso. El cliente en este caso es la función o método que hace la llamada, el cual requiere ejecutar algún proceso en el servidor.

4.14.1 RPC

RPC permite a los desarrolladores de software hacer funciones o llamadas a métodos a través de la red, lo cual permite rudimentarias aplicaciones distribuidas sobre la red. Es un mecanismo a nivel de capa de aplicación, que comunica una aplicación con otra. La llamada de RPC es idéntica a la sintaxis de una función local. La función toma la forma de interacción Cliente - Servidor. RPC actúa con el mismo espíritu Java RMI, pero se diferencia en que:

- *no hay contexto de objeto. Funciones remotas son llamadas a través de la red como una API remota. RMI logra esto naturalmente*

- *requiere un seteo para el retorno de una llamada en ambos servidores*
- *el cliente y el servidor pueden ser implementados con independencia de lenguajes de programación*

4.14.2 Comparando protocolos XML

4.14.2.1 XMLRPC

- *Liviano, rápido y fácil para la implementación de invocaciones remotas de funcionalidad a través de HTTP tunneling*
- *HTTP tunneling es integrante de este protocolo*
- *Operaciones sin conexión no están actualmente disponibles*
- *Las especificaciones son muy fáciles de entender*
- *Es muy fácil de programar, fue la meta del diseño original*
- *El trafico del protocolo es legible y entendible, y un operador podrá descifrar el tráfico no*
- *encriptado*
- *Una conexión debe abrirse para cada nuevo canal, no es posible multiplexar canales*
- *Solamente un servidor sirviendo a un cliente, no soporta descentralización*
- *Es una instancia de protocolo fija, no tiene la habilidad de pasarse a otros protocolos*
- *No existe una seguridad de autentificación especificada*
- *Request y response son ambos documentos XML*

4.14.2.2 SOAP

- Es el protocolo clave para la nueva infraestructura de interacción entre aplicaciones

- Soporta HTTP tunneling, como un transport binding

- Soporta manejo de RPC

- Soporta operaciones de desconexión

- No es fácil entender las especificaciones, es un complejo esquema de codificación, binding para protocolos y otras para HTTP que no están especificadas

- No es muy fácil de programar en el actual estado de desarrollo

- No se necesita de una especificación para descifrar el tráfico no encriptado

- El generar protocolos eficientes no fue un criterio de diseño

- Multi-channels y multiplexing no están actualmente soportadas

- Se menciona la posibilidad de multi-agent, procesing path, sin embargo no están las especificaciones en la versión 1.1

- La habilidad de host a otros protocolos no se aplica directamente

- Seguridad y autentificación no están directamente especificadas, pero si son flexibles a la hora de implementar.

- Envelope, request y response son documentos XML.

4.14.3 Evolución de los Web Services.

El concepto central de Web Services: computación orientada a los servicios, que determinan un nuevo paradigma de procesos de creación de aplicaciones altamente distribuidas. Son aplicaciones modulares, que se auto describen, pueden ser publicadas, localizadas, e invocadas desde cualquier lugar de la Web o dentro de una Intranet.

El modelo de WS potencia el desarrollo de aplicaciones distribuidas. Por ejemplo una compañía de alquiler de coches puede asociar su sistema de reservas on-line con el sistema de aerolíneas y los hoteles. De tal manera que un viajero puede al mismo tiempo hacer una

reserva de un vuelo, una habitación de hotel y un coche de alquiler.

4.14.4 Historia de la computación distribuida.

Esta evolución tecnológica y de una búsqueda de soluciones a la computación distribuida no es un problema reciente. En los 1980 los protocolos de comunicación no era objeto de interés de los desarrolladores. Realizar aplicaciones que dentro de una misma máquina se comunicaran entre sí, era suficiente. En los 1990 alcanzaron popularidad objetos como COM (Componet Object Model) introducido por Microsoft y CORBA (Common Object Request Broker Architecture) introducido por OMG (Object Management Group).

En general, COM y CORBA son modelos para escribir y encapsular código binario. Estos son componentes que pueden ser fácilmente llamados desde cualquier aplicación que soporte COM o CORBA. Sin embargo estos modelos no son fácilmente ínter operables, de tal manera que COM puede solamente llamar a COM, y lo mismo ocurre con CORBA. Conectar una maquina a otra se transformó en una prioridad cuando las redes locales se generalizaron. Fue entonces que OMG estableció IIOP (Internet Inter-ORB Protocol) como el protocolo de comunicación para CORBA. Microsoft creo DCOM (Distributed COM). Mas tarde Sun Microsystems lanzo al mercado RMI (Remote Method Invocation). Con estos protocolos se pueden llamar componentes que se encuentren en otras computadoras a través de la red. Estas llamadas se realizan bajo la forma de RPC (Remote Procedure Call). Es necesario aclarar que estos protocolos no son interoperables. La solución esta disponible para tener comunicación aplicación a aplicación desde cualquier máquina a cualquier otra sin importar el sistema operativo, entorno de lenguajes, modelos de objetos distribuidos, y usando los estándares de Internet. Esto es un buen candidato para un protocolo de comunicación universal.

4.14.5 Conectando Aplicaciones en la Web

Los Web Services manejan la interoperabilidad mediante el protocolo estándar de XML. La necesidad de pensar en el Web site como en una función.

Web Services son aplicaciones modulares que se auto describen, que pueden ser publicadas, localizadas e invocadas o usadas desde cualquier parte en la Web o dentro de cualquier red local, basadas en los estándares abiertos de Internet.

Ellos combinan los mejores aspectos de los componentes de programación, y programación Web, y son un paquete en la forma de módulos, que pueden ser reutilizados, sin preocuparnos acerca de cómo el servicio esta implementado, o aún en que lenguaje, sistema operativo, o modelos de componentes fueron usados para crearlos, así como el consumidor y desde su punto de vista es una "caja negra". Son accesibles vía los protocolos de Internet como HTTP y SMTP, y basados en XML.

XML y SOAP son la base tecnológica de la arquitectura de los Web Services. SOAP (Simple Object Access Protocol) es un protocolo de mensajes entre computadoras. SOAP especifica el formato de mensaje que accede e invoca a los objetos, mas que un objeto en particular.

4.14.6 Arquitectura de los WS

Hay 3 conceptos básicos que son: descubrir, describir e invocar. Desde el punto de vista del consumidor, tendrá primero que encontrarlo o descubrirlo. Luego el consumidor tendrá que aprender acerca del Web Service o su descripción. Finalmente podría disponer o tener la necesidad de invocarlo, suministrando los elementos de entrada y recibiendo los elementos de salida del mismo.

Esto se amolda a las situaciones de las empresas, si por ejemplo solo se quiere crear una

comunicación de aplicación a aplicación, solo alcanza con usar SOAP, sin la necesidad de los otros componentes, y WS Framework pueden ser implementados en cualquier lenguaje de programación.

4.14.7 WSDL : Web Services Definition Language

WSDL es usado para describir servicios Web por parte de las empresas, de tal forma que los clientes puedan acceder a ellos y utilizarlos. Provee información de los distintos métodos que el WS brinda, muestra como accederlos y el formato que deben tener los mensajes. WSDL es un formato XML que describe los servicios de red como un conjunto de endpoints que procesan mensajes contenedores de información orientada tanto a documentos como a procedimientos. Las operaciones y los mensajes se describen de forma abstracta y después se enlazan a un protocolo de red y a un formato de mensaje concreto para definir un endpoint de red. Los endpoints concretos relacionados se combinan en endpoints abstractos (servicios). WSDL es extensible, lo que permite la descripción de endpoints de red y sus mensajes, independientemente de los formatos de los mensajes o protocolos de red utilizados para comunicarse.

4.14.8 UDDI : Universal Description, Discovery and Integration

UDDI es un proyecto propuesto por Ariba, Microsoft e IBM. Es un estándar para registrar y descubrir los WS. La idea es una registración en el registro de empresas UDDI, un servicio centralizado y replicado a distintos nodos en forma regular, quedando disponible para ser descubierta por otras empresas.

4.14.9 SOAP : Simple Object Access Protocol

La idea detrás de SOAP es la misma que RPC. También define un protocolo para llamadas a métodos remotos, sin embargo SOAP incluye:

- *SOAP especifica información adicional incluida en el documento XML, que describe el contenido y como podría ser procesada.*
- *no se asocia fuertemente con HTTP*
- *define la especificación de algunas estructuras en XML, tales como arrays.*
- *el modelo es descentralizado, esto significa que puede ser procesado por varios intermediarios*
- *tiene características específicas para operaciones clásicas de RPC con parámetros in/out, etc.*

4.14.9.1 SOAP y Web Services

SOAP no es el único protocolo para el uso de WS. Estos pueden ser usados a través de XML-RPC, operando simplemente sobre HTTP GET, sin que por ello tengan menos componentes. Pero SOAP ha sido aceptado por proveedores y desarrolladores independientes como un estándar, el cual es mejorado todo el tiempo.

4.14.9.2 SOAP y HTTP

SOAP especifica el binding sobre HTTP, básicamente un tunneling, según el modelo request/response. El HTTP tunneling se refiere a un wrapper del protocolo de comunicación HTTP. La habilidad de crear protocolos que se colocan en el tope del protocolo estándar HTTP se hace más importante que nunca. La mayoría de las redes comerciales optimizan el tráfico del protocolo HTTP, permitiendo cualquier protocolo con base en HTTP. Esto a su vez potencia la seguridad de las redes a través de los firewall, los cuales permiten HTTP tunneling. De otra manera se introducirían potenciales riesgos al tener que habilitar otros puertos en los firewalls.

- *el Content-Type del header de HTTP debe especificar "application/soap"*

- *HTTP POST es solamente para el tipo request*
- *SOAPAction en el HTTP header es requerida para indicar el intento de request*
- *un request SOAP exitoso es indicado por un response con status HTTP y código 2nn siendo nn un número entre 00 y 99*
- *un error es respondido con el código de status 500 y el mensaje Internal Server Error, conteniendo el Fault element.*

4.14.9.3 SOAP Message

Un mensaje SOAP consiste de un documento XML. El elemento raíz es siempre el <Envelope>. El elemento <Envelope> puede contener un elemento opcional <Header>, el cual debe ser el primer hijo inmediato. El elemento <Header> puede tener múltiples header entries. Anticipa el camino del mensaje SOAP como valor agregado a lo largo del proceso. El elemento <Body> le sigue al <Header> y es obligatorio. Es el contenido del mensaje, con lo necesario para consumir el servicio. Puede contener sus propios elementos hijos, llamado body entries. Cada uno de estos es independientemente codificado, y especificado en un atributo encodingStyle que indica su uso. SOAP define solo un tipo de body entries, llamado <Fault>, que ocurre solo una vez dentro del <Body> y se utiliza para indicar errores o excepciones. Todo esto se basa en un modelo de mensajes request/response como una forma de invocar un método y pasarle parámetros. El potencial de esto es hacerlo sobre Internet usando HTTP para realizar la comunicación entre aplicaciones.

JForms se basa hasta cierta medida en los Web Services, ya que implementa parte de su filosofía (en la que se refiere a la comunicación por medio de XML y la separación de los

servicios en diferentes maquitas). JForms utiliza los EJBs para definir los servicios que ofrece, como por ejemplo el de agregar un registro a una tabla de la base de datos.

JForms utiliza su propio formato XML y no el definido por el WSDL, pero futuros proyectos de implementación de AJED podrían basarse completamente en los Web services

4.15 WEBDAV

WebDAV (Web-based Distributed Authoring and Versioning) [5] es una extensión del protocolo HTTP desarrollada alrededor de 1996 por la IETF2, que pretende hacer de la Web un medio escribible, permitiendo que varias aplicaciones puedan interoperar a través de HTTP mediante el intercambio de diferentes tipos de recursos, que no necesariamente son archivos planos como HTML y XML.

WebDAV define un conjunto de nuevos encabezados y métodos que facilitan la navegación y el mantenimiento de sitios Web, estos nuevos métodos permiten la manipulación de Properties (Méta-información sobre los recursos almacenados en el servidor) y Collections (Conjuntos de recursos [Directorios]), controlar el acceso a recursos que se encuentren en uso y además heredar todas las ventajas de la estructura HTTP/HTTPS, como la autenticación, la encripción y la navegación a través de Firewalls y Proxies. A diferencia de los métodos tradicionales de HTTP, los métodos de WebDAV poseen un cuerpo XML, esto para darle flexibilidad al protocolo y poder hacer varias operaciones con un solo requerimiento.

Paralelo a WebDAV, la IETF ha estado desarrollando otras extensiones a HTTP como DeltaV (Manejo de versiones), ACL3 (Gestión de acceso a los recursos) y DASL4

(Mecanismo de Búsqueda), que hacen de WebDAV una especificación muy robusta que se puede usar en cualquier contexto.

La aceptación que ha tenido WebDAV en la industria es muy amplia, lo que se refleja en la gran cantidad de aplicaciones que actualmente implementan esta especificación, como Microsoft Office (por medio de sus Carpetas Web), Adobe Acrobat 5, Mac OS X, Macromedia Dreamweaver, Apache Tomcat, Apache httpd, entre otros.

4.16 AMBIENTES DE DESARROLLO

4.16.1 Eclipse

Eclipse[1] surge alrededor del año 2001 como una propuesta innovadora por parte de IBM©, OTI(Object Technology International) y otras compañías que querían desarrollar una plataforma de código abierto que permitiera la integración entre diversas herramientas de desarrollo y que sirviera como una manera de capitalizar el desarrollo de 18 ISC-2003-2-32 herramientas en el lenguaje de programación Java. "Eclipse es un IDE para nada en particular", que define una funcionalidad básica sobre la cuál es posible implementar herramientas de desarrollo para varios lenguajes como Java, C/C++, HTM, XML, entre otros. Todo esto es posible porque Eclipse está diseñado para descubrir, integrar y correr distintos módulos (plug-ins) que le dicen a la plataforma cuáles son las acciones que debe llevar a cabo ante la presencia de determinado tipo de recursos.

Eclipse gira alrededor de tres ejes: El desarrollo de un ambiente de desarrollo para Java, la integración de diferentes herramientas, y la comunidad de desarrolladores open source que trabajan en él.

Como ambiente de desarrollo en Java, Eclipse provee al desarrollador de varias utilidades que le permiten escribir código de manera más rápida y eficiente. Estas herramientas son: Un editor especializado que provee funciones de asistencia de código, marcado de sintaxis, marcadores, revisión de sintaxis, etc.; un depurador y un esquema sencillo de integración con Ant y JUnit. Todo este conjunto de características hace que la labor de codificación sea más sencilla para el desarrollador de Java, permitiéndole tener todas las herramientas para su trabajo en una sola aplicación. Como plataforma de integración, la arquitectura de Eclipse permite a los desarrolladores extender fácilmente las funcionalidades del IDE, esto a través de varios Frameworks y APIS, que hacen más fácil la interacción con la plataforma. Eclipse permite integrar varias herramientas dentro de sí, esto lo libra de la limitación de una metodología de trabajo particular, permite desarrollar fácilmente aplicaciones con Internacionalización y en algunos sistemas operativos, se integra con aplicaciones y tecnologías como el reconocimiento de voz que lo hacen una aplicación accesible a una amplia gama de usuarios.

Como comunidad de código abierto, "la misión del proyecto Eclipse es adaptar y desarrollar la tecnología de Eclipse para satisfacer las necesidades de la comunidad de desarrolladores y de usuarios". Eclipse provee varias herramientas para que la colaboración entre los distintos miembros de la comunidad (corporaciones, investigadores y desarrolladores en general) puedan desarrollar herramientas inter-operables.

El desarrollo de Eclipse está dividido en varios proyectos, cada uno de ellos con un comité propio (PMCs, Project Management Committees) que coordina su evolución y que administra las listas de correo, sistemas de control de errores y versiones que facilitan la colaboración entre sus miembros.

4.17 EJB (ENTERPRISE JAVA BEAN)

Un "Enterprise Java Bean" es un "deployable component", el término "deployable component" implica que existe un ambiente de ejecución, éste ambiente es precisamente un "EJB (Enterprise Java Bean) Container" parte de un "Java Application Server", agrupando funcionalidades para una aplicación, algunas de estas son las siguientes:

Servicios ("Middleware")

Esta posiblemente sea la mayor ventaja de un EJB. Cuando se diseña un componente de Software se deben definir varios servicios para su funcionamiento, algunos pueden ser:

- *¿Si ocurre un error que procedimiento debe ejecutarse?*
- *¿Si la base de datos especificada se encuentra desactivada, existe otra alternativa?*
- *¿No fue posible cumplir exitosamente "x" procedimiento, se deben retractar sus acciones parciales o re-invocar la transacción?*

Estos Servicios (comúnmente llamados "Middleware") por lo general son requeridos además de la lógica contenida en los componentes principales, obviamente estos servicios ("Middleware") aún deben ser diseñados, sin embargo, mediante un "EJB Container" se ofrecen estos servicios y es a través de un "Enterprise Java Bean" que es posible desarrollar los componentes principales ("lógica de negocios").

Debido a la solución que intentan ofrecer EJB ("Enterprise Java Beans") su diseño gira alrededor de procedimientos remotos.

Un EJB puede interactuar con una gran gamma de clientes desde: JSP o Servlets, bases de datos, Applets.

4.18 TIPOS DE EJB ("entreprise java bean")

4.18.1 Session EJB's

Un Session EJB permite realizar cierta lógica solicitada por un cliente ya sea un JSP/Servlet, Applet e inclusive otro EJB. Existen dos tipos de Session EJB's:

4.18.1.1 Stateless (Session) EJB's

Este tipo de EJB como su nombre lo indica no mantiene estado ("Stateless") en el "EJB Container", estos EJB's son utilizados para realizar tareas rutinarias que no requieren identificar o rastrear al cliente, algunos EJB's de este tipo son: operaciones matemáticas complejas, búsquedas generales, etc.

4.18.1.2 Statefull (Session) EJB's

A diferencia de "Stateless (Session) EJB's" este tipo de EJB's permiten mantener la sesión del cliente en el "EJB Container", de esta manera el cliente puede trabajar con cierto juego de datos especifico administrado por el "EJB Container", la aplicación ideal para este tipo de EJB es un componente de compra ("Shopping Cart") el cual puede identificar artículos e información personal del cliente a través de un lapso de tiempo extenso ("Session").

4.18.2 Entity EJB's

Un Entity Bean a diferencia de un "Session Bean" trabaja en conjunción con un deposito de información (generalmente una base de datos), esto permite que el EJB manipule información residente en sistemas ajenos al "EJB Container"; en un "Statefull (Session) EJB" si ocurre una falla en el "EJB Container" se pierde toda información, mientras si se utiliza un "Entity EJB" aún permanecerá esta información en el sistema aledaño (generalmente una base de datos). En otras palabras, un "Entity EJB" manipula una copia | reflejo de información que reside en otro sistema.

Al igual que "Session EJB's" existen dos tipos de "Entity EJB's":

4.18.2.1 (Entity) Bean Managed Persistence

Este tipo de "Entity Bean" requiere que la lógica necesaria para accesar el sistema de información (base de datos) se definida manualmente, por lo general esta lógica se encuentra en JDBC y define: como y cuando debe ser accesada|actualizada|guardada la información entre el EJB y la base de datos.

4.18.2.2 (Entity) Container Managed Persistence

Este "Entity Bean" como su nombre lo indica es manejado por el "EJB Container", a diferencia de un "Bean Managed EJB" donde se requiere definir lógica de acceso manualmente, en un "Container Managed EJB" el "EJB Container" genera toda lógica de acceso para el sistema de información (base de datos).

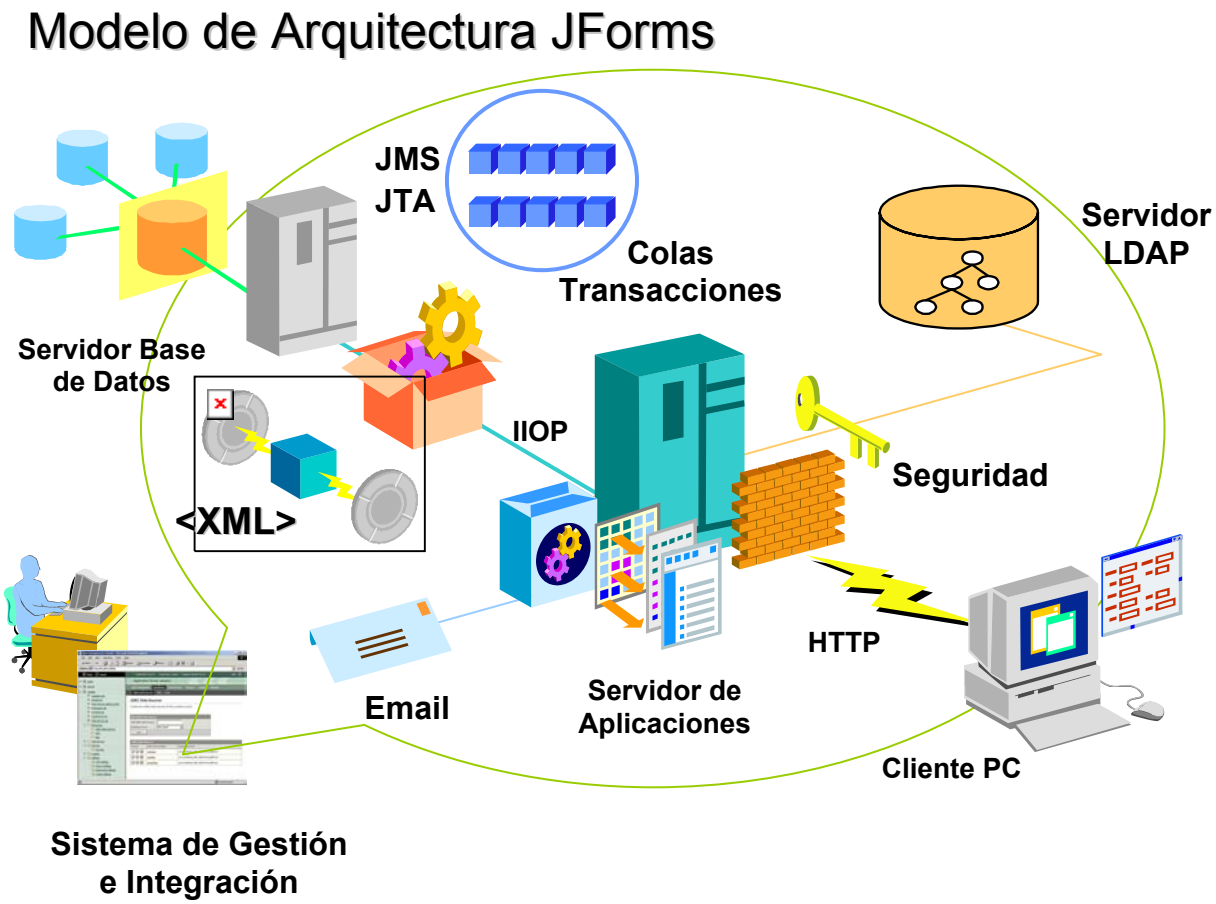
Aparentemente un "Bean Managed EJB" no tiene mucha razón de ser, sin embargo, hay casos donde es empleada lógica de acceso sumamente compleja la cual no es posible generar a través del "EJB Container", es por esto que los "Bean Managed EJB's" permanecerán en existencia a pesar de las facilidades ofrecidas por "Container Managed EJB's"

4.18.3 Messaging EJB's

Un "Messaging EJB" ofrece las funcionalidades de (valga la redundancia) sistemas "Messaging" como MQSeries de IBM o Rendez-Vous de Tibco. A muy grandes rasgos un sistema "Messaging" ofrece el funcionamiento de intermediario para recibir y publicar mensajes ("Messages"), una de las ventajas de un "Messaging System" es que opera en forma asincrónica ("asynchroynous") o "non-blocking".

5 DISEÑO DE JFORMS

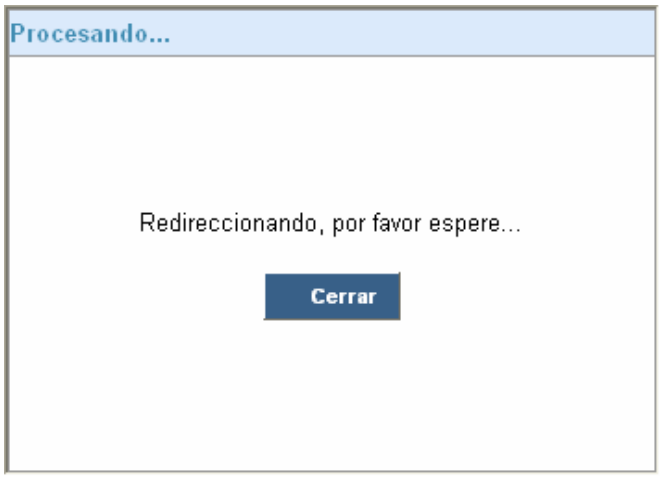
5.1 ARQUITECTURA



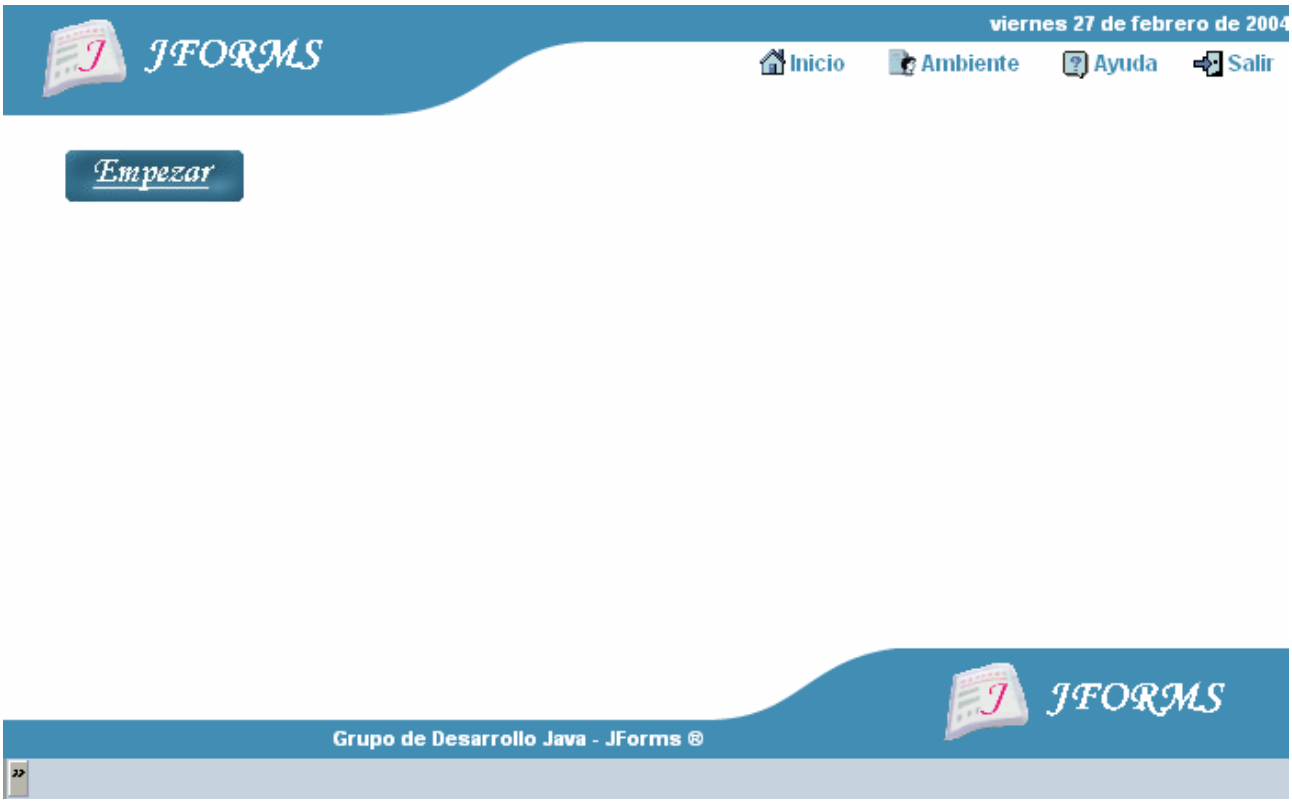
5.2 INTERFACES

5.2.1 Inicio:

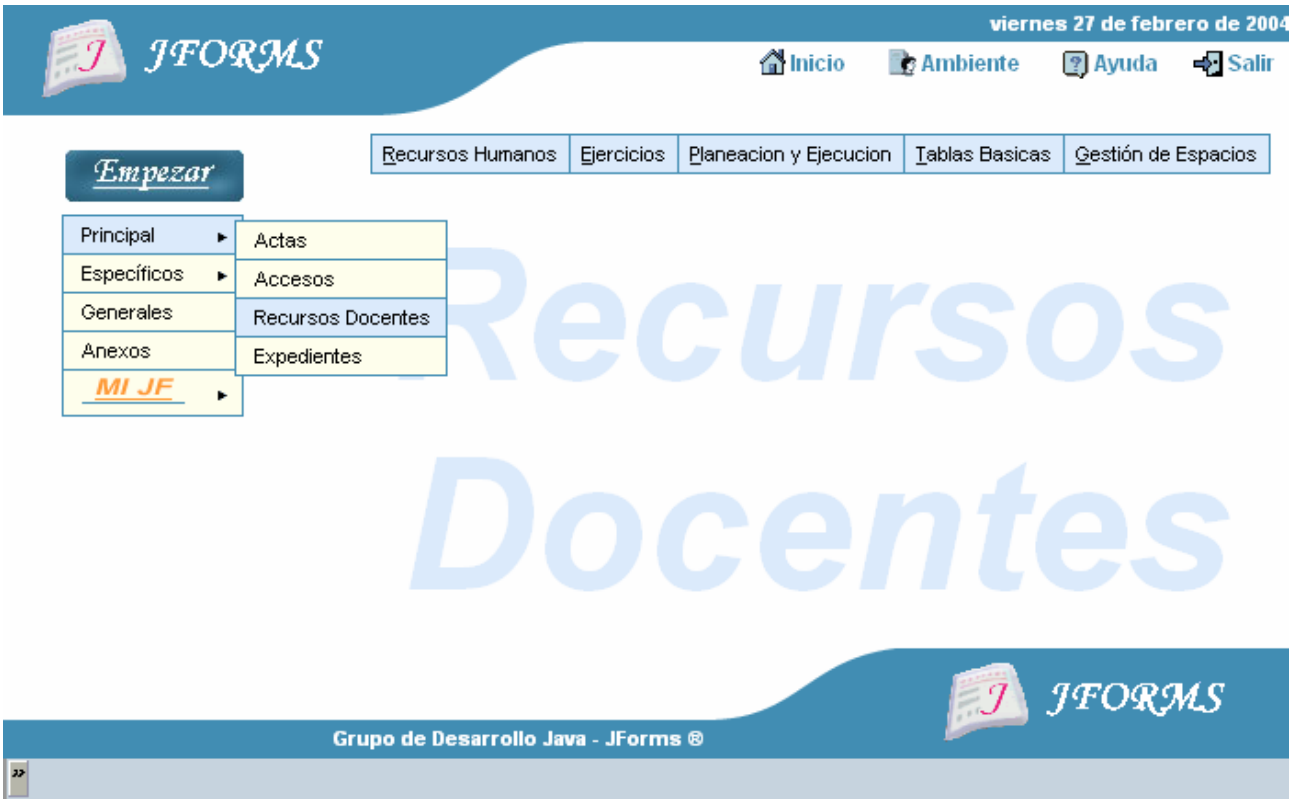
5.2.2 Procesando una petición:



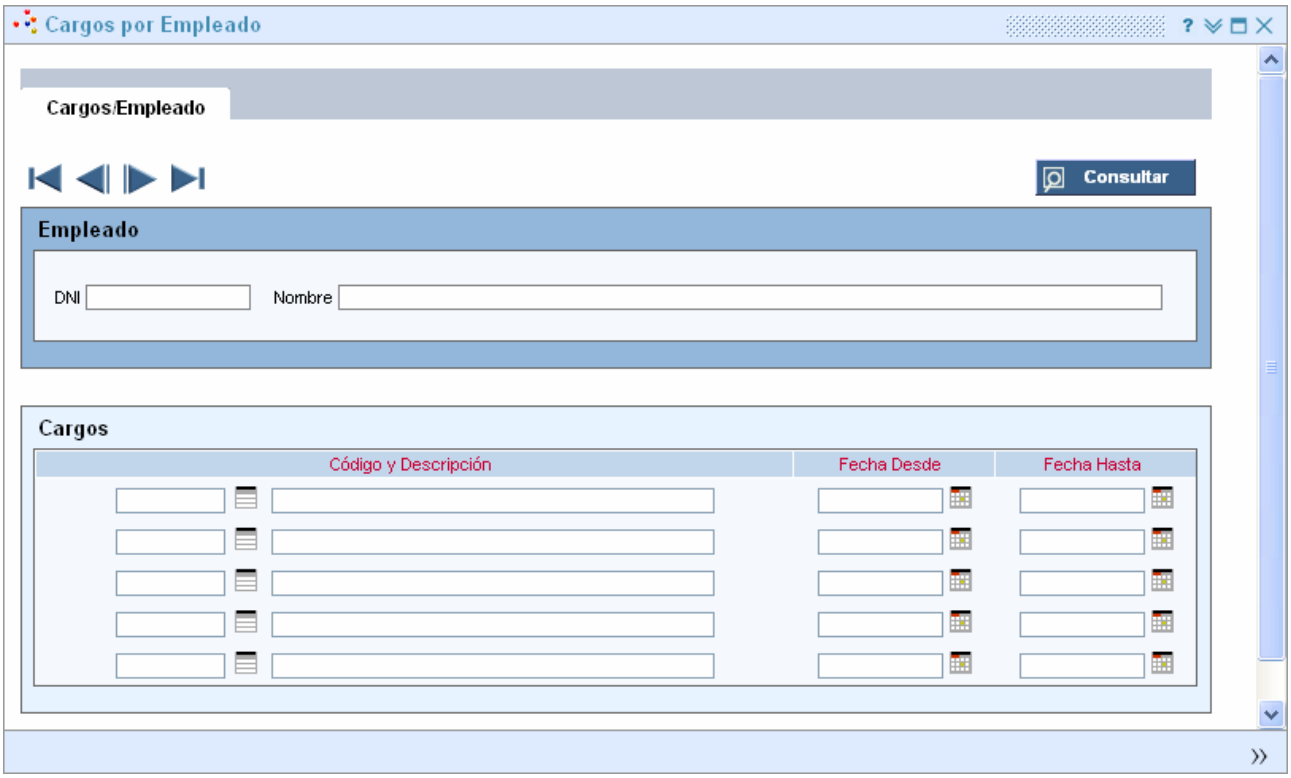
5.2.3 *Entrada en una aplicación:*



5.2.4 Menús:

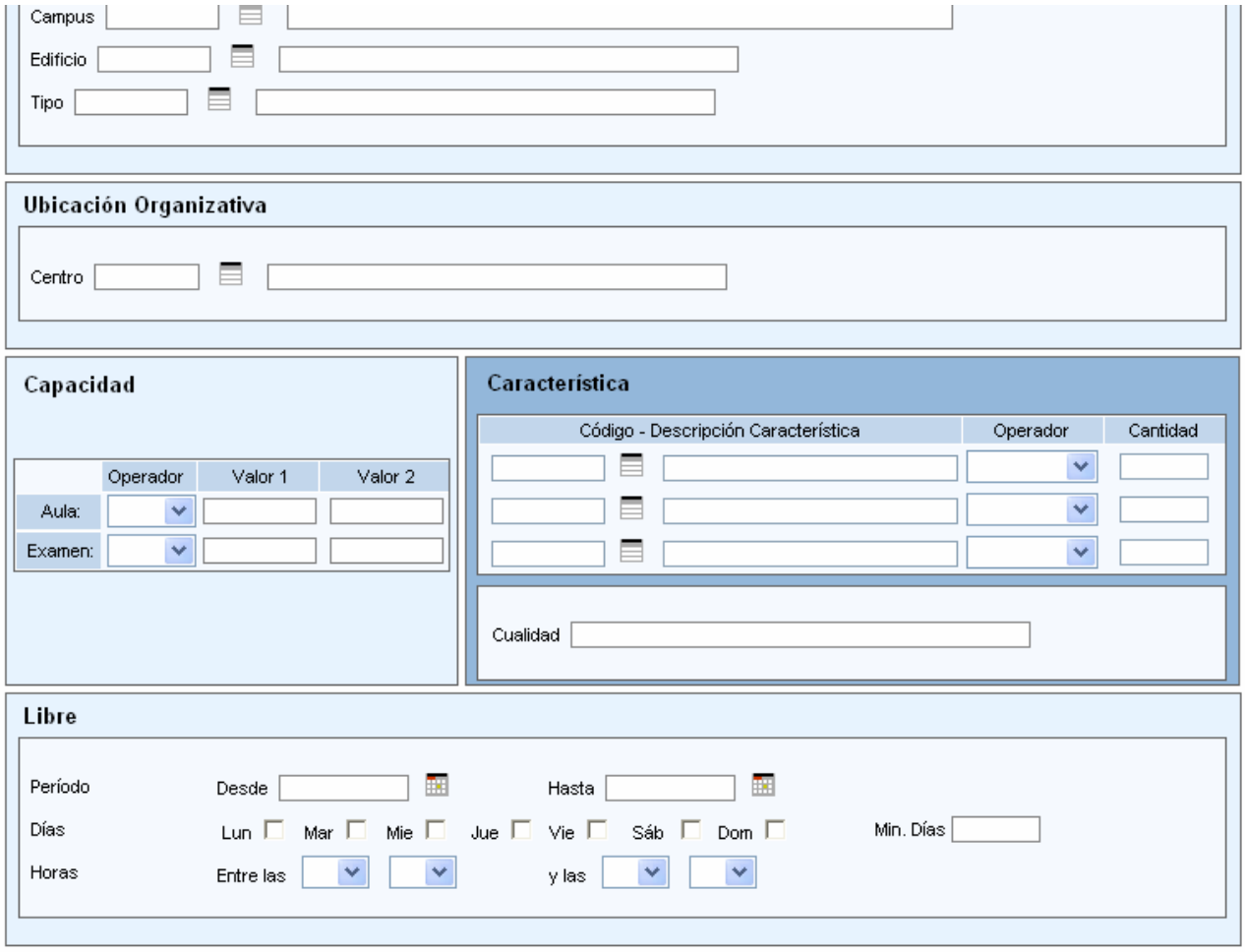

JFORMS
viernes 27 de febrero de 2004
Inicio
Ambiente
Ayuda
Salir
Empezar
Recursos Humanos
Ejercicios
Planeacion y Ejecucion
Tablas Basicas
Gestión de Espacios
Principal
Específicos
Generales
Anexos
MI JF
Actas
Accesos
Recursos Docentes
Expedientes
Recursos Docentes
JFORMS
Grupo de Desarrollo Java - JForms ®

5.2.5 *Maestro detalle:*



5.2.6 *Formas abiertas:*

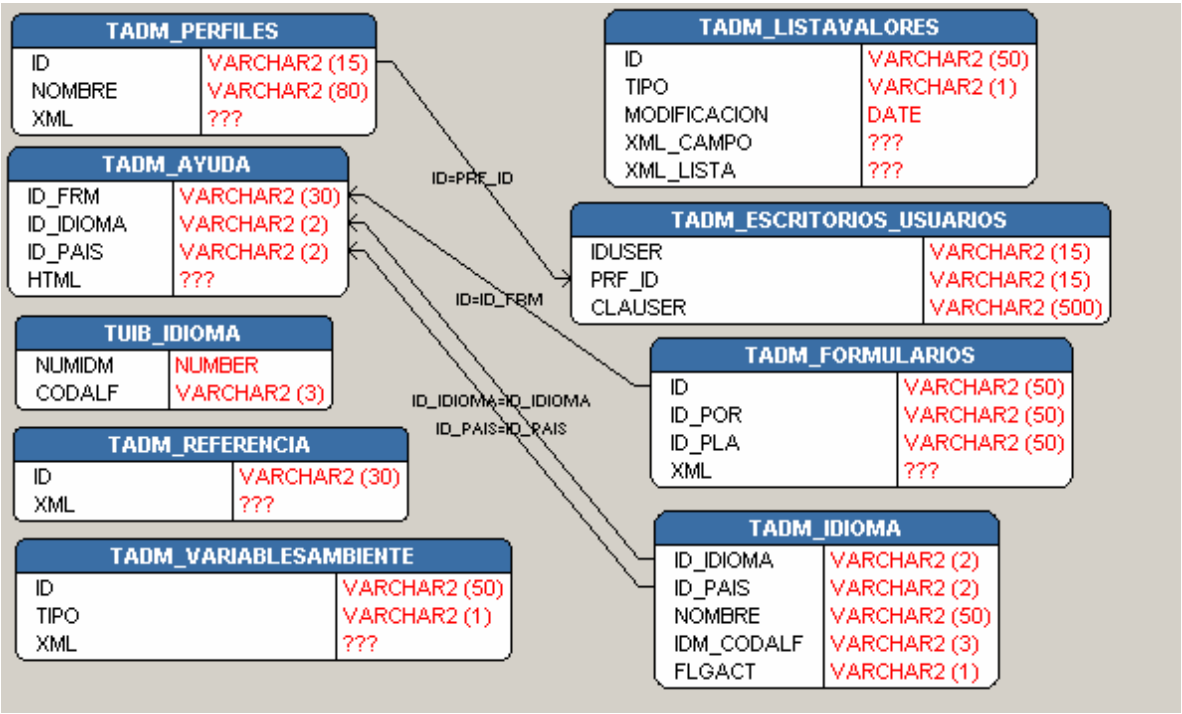
5.2.7 Múltiples regiones:



6 IMPLEMENTACIÓN DE JFORMS

JForms esta constituido por un conjunto de paquetes y tablas en la base de datos que le proporcionan conectividad y operatividad para ejecutar las peticiones realizadas por los usuarios.

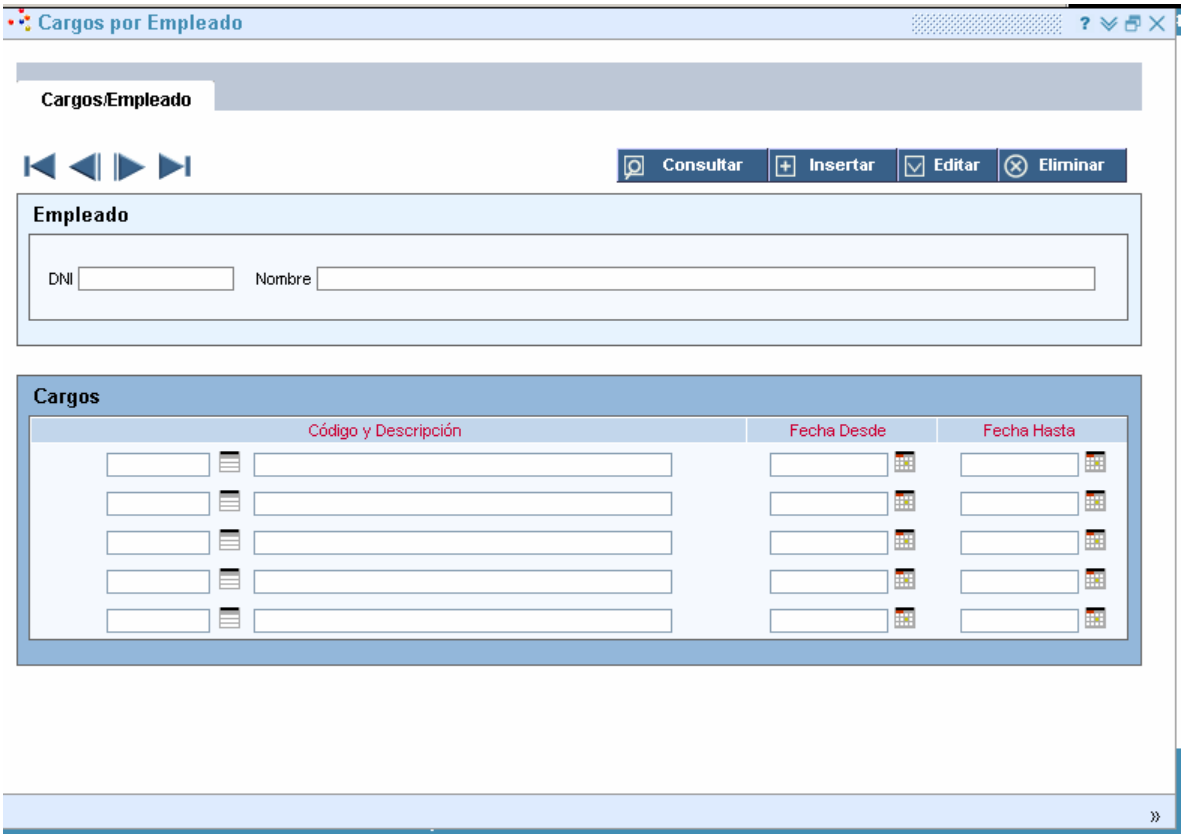
6.1 MODELO ENTIDAD-RELACIÓN



6.2 PAQUETES DE CLASES

6.2.1 Portlets

Componente dinámico del portal. Genera contenido HTML.



6.2.2 Procesadores http de Datos

Servlets que procesan los datos de captura ingresados por el usuario a través del navegador.

Convierte los datos en documentos XML y los envía a los EJB de proceso quienes devuelven una respuesta de éxito o excepción de la transacción.

6.2.3 EJB's de Proceso

Session Beans encargados del control.

Enlazan la lógica de negocio.

Provee métodos para resolver los casos de uso.

6.2.4 EJB's de Entidad

Entity Beans encargados de manejar la transaccionalidad.

Son entidades agregadoras

Se asocian con el ServicioLocalizacion para que este realice la persistencia de los datos.

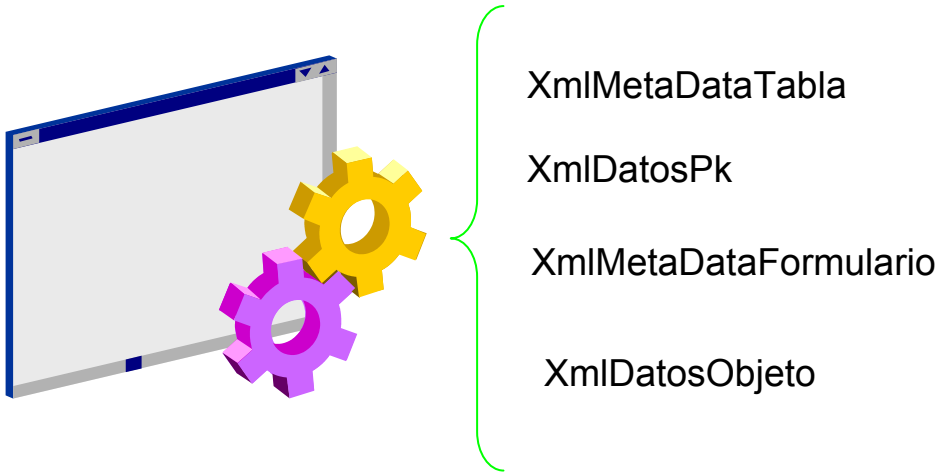
6.2.5 Servicios

Session Beans encargados de ofrecer servicios específicos.

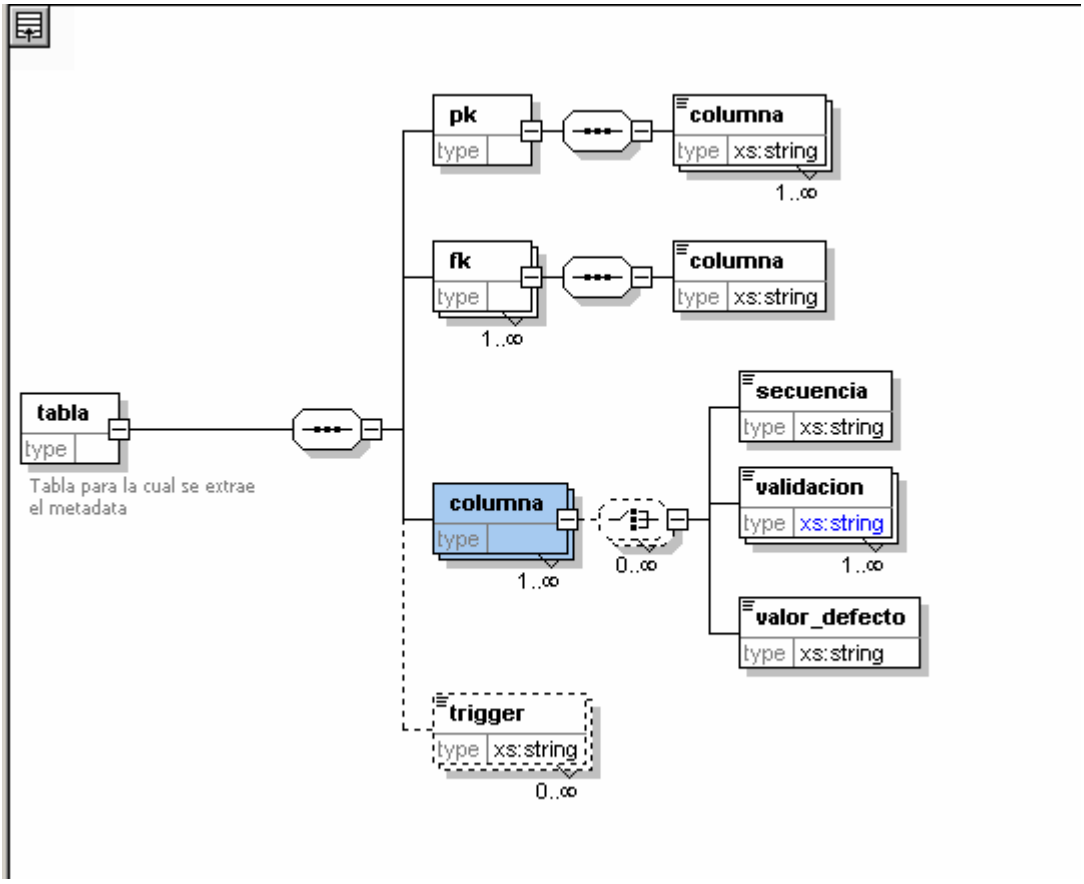
Se asocian con clases ContextoXXX que proveen la conexión al recurso.

6.2.6 XML

Documentos según el estándar (DOM – Document Object Model) que permiten el intercambio de datos y su procesamiento de manera genérica.



Ejemplo:


pk
type
columna
type xs:string
1..∞
fk
type
1..∞
columna
type xs:string
tabla
type
Tabla para la cual se extrae el metadata
secuencia
type xs:string
columna
type
1..∞
0..∞
validacion
type xs:string
1..∞
valor_defecto
type xs:string
trigger
type xs:string
0..∞

7 PRUEBAS

En el proceso de creación de una aplicación en JForms se dispone de una interfaz en la cual el usuario desarrollador basado en los XML Schema, personalizara las opciones de menú, los vínculos, imágenes y algunos textos.

Además de definir la interfaz principal, también debe definir las formas que se abrirán en la aplicación, cada una en un archivo XML independiente. Y allí deberá especificar el mapeo con la base de datos, para que JForms sepa a cuales campos y tablas debe acceder.

Se deben definir también los roles que existirán en la aplicación y cuales son los permisos de los cada uno de ellos, es decir, a que opciones y formas pueden acceder.

Las formas en JForms son el pilar para definir la aplicación, a través de ellas es que se consulta, inserta, actualiza o modifican los datos en la base de datos. Y JForms será el encargado de ejecutar las respectivas acciones y verificar que el usuario que lo esta haciendo, cuente con los privilegios requeridos.

En la definición de las formas se cuenta con muchos tipos de atributos, desde sencillos campos de texto o fechas, hasta complejos listados de información. Además se dispone de un buscador que trabaja dado unos parámetros de búsqueda para cargar datos en los campos de alguna forma.

Una vez definidos los XML que definen la aplicación y su correspondencia con los campos de la base de datos, debe cargar los XML en la misma base de datos, y esto se logra por medio de una aplicación de escritorio que viene con JForms para subir los XML creados.

Se nota que el proceso más difícil que le queda al usuario desarrollador es definir la base datos que utilizará la aplicación, ya que en la mayoría de los casos, no deberá escribir una

línea de código. Solo en los casos en que necesite una forma que se salga de las opciones definidas en JForms, deberá entonces recurrir a definir la forma a través de código JSP.

8 CONCLUSIONES

No podemos asegurar que el modelo que mostramos con AJED esta terminado, el trabajo que aun falta es tanto, como las necesidades que nacerán en un futuro no muy lejano. JForms, una de las implementaciones del modelo de AJED, es una solución para el desarrollo de muchos tipos de aplicaciones que necesiten manejar cierta información y desplegar su interfaz en un navegador de Internet, y a través de esta interfaz ofrecer las opciones al usuario de mostrar y modificar dicha información.

Para otras implementaciones de AJED podría pensarse en clientes nativos de la plataforma operativa y no solo los clientes que ofrece un navegador, esto daría mucha mas interactividad del lado del cliente. Para este objetivo podría pensarse en Java Web Start.

JForms usa como motor de base de datos a Oracle 8i, una futura implementación de AJED, podría permitir que el desarrollador utilice el motor de base de datos que mas convenga. Esto se podría lograr independizando a la aplicación de la base de datos por medio de algún patrón de diseño de software como por ejemplo el DAO (Data Access Object), un patrón para el diseño de aplicaciones en el que se separa la lógica del almacenamiento de los datos de la lógica de la aplicación, permitiendo que los datos se guarden un cualquier base de datos, o en archivos.

También podría pensarse en una implementación que permita la personalización de la apariencia de cada componente de la interfaz. XSL facilita esta tarea o tal vez XForms la

nueva especificación del w3c para definir formularios Web dentro de paginas HTM. Incluso podría pensarse en un editor de interfaces, en el cual se definan las vistas, y este editor las transforme en el correspondiente XML.

9 BIBLIOGRAFÍA

FROUFE Agustín, Java Server Pages: Manual de usuario y tutorial, Editorial Alfa Omega., Primera Edición., p. 270.

POWELL Tomas, SCHNEIDER Fritz, JavaScript: Manual de Referencia, Editorial McGraw-Hill, Primera Edición, p. 603.

http://java.sun.com/docs/books/tutorial/rmi/overview.html

http://archive.devx.com/upload/free/features/entdev/2000/03mar00/imw0300/imw0300.asp

http://java.sun.com/products/javawebstart/architecture.html

http://javahispano.org/download/articulos/serializacion.pdf

http://www.dinopolis.org/documentation/misc/theses/pzamb/node77.html

http://www.iec.csic.es/criptonomicon/java/firmas.html

http://rs6k1.csg.uwaterloo.ca/~dmg/tutorial/xml/

http://www.ulpgc.es/otros/tutoriales/xml/XSL.html

http://www.openresources.com/es/magazine/xml-tutorial/

http://www.terra.com/informatica/que-es/java.cfm

http://www.mcc.unam.mx/~cursos/Algoritmos/javaDC99-2/RMI1.html

http://www.geocities.com/santifaci/fundamentos/index.html#AEN16

10 ANEXOS

10.1 CONFIGURACIONES EN FORMATO XML

Archivo de inicio (uxxiconf.xml)

Este archivo contiene las configuraciones de los servidores que utiliza la aplicación.

```
<?xml version="1.0" encoding="ISO-8859-1"?>
<conf>
<jform idConfiguracion="portal">
 <servidorOracle>
  <driver><![CDATA[thin]]></driver>
  <servidor>
   <![CDATA[127.0.0.1]]>
  </servidor>
  <puerto><![CDATA[1521]]></puerto>
  <sid><![CDATA[basedato]]></sid>
  <protocolo><![CDATA[tcp]]></protocolo>
  <usuario><![CDATA[scott]]></usuario>
  <clave><![CDATA[123]]></clave>
 </servidorOracle>
```

Indica la máquina donde se encuentra la base de datos.

```
  <servidorServlet>
  <servidor>
   <![CDATA[127.0.0.1]]>
  </servidor>
  <puerto><![CDATA[18080]]></puerto>
  <contexto>
   <![CDATA[jforms-jf]]>
  </contexto>
  </servidorServlet>
```

Indica la máquina donde se encuentra el instalado TOMCAT.
NOTA: se recomienda colocar la IP del servidor con la cual puedan tener acceso los demás

```
 <servidorEjb>
  <fabrica>
 <![CDATA[org.jnp.interfaces.NamingContextFactory]]>
  </fabrica>
  <protocolo><![CDATA[jnp]]></protocolo>
  <servidor><![CDATA[localhost]]></servidor>
  <puerto><![CDATA[1099]]></puerto>
  <usuario><![CDATA[]]></usuario>
  <clave><![CDATA[]]></clave>
 </servidorEjb>
```

Indica la máquina donde se encuentra instalado el JBOSS (Contenedor de EJB's)

```
<servidorLdap>
<fabrica>
<![CDATA[com.sun.jndi.ldap.LdapCtxFactory]]>
</fabrica>
<protocolo><![CDATA[ldap]]></protocolo>
<servidor><![CDATA[localhost]]></servidor>
<puerto><![CDATA[389]]></puerto>
<puntoEntrada>
  <![CDATA[o=Unimag,c=CO]]>
</puntoEntrada>
<isconfig><![CDATA[true]]></isconfig>
<usuario>
  <![CDATA[cn=root,o=Unimag,c=CO]]>
</usuario>
<clave><![CDATA[secret]]></clave>
</servidorLdap>
```

Indica la máquina donde se encuentra instalado el LDAP (Servidor de Usuarios que pueden acceder al sistema).
NOTA: la etiqueta <isconfig></isconfig>, habilita o deshabilita el la autenticación de usuarios con LDAP.

Creación de archivos de configuración XML para el escritorio de un usuario.

Nota: **Para la edición de estos archivos se recomienda el uso de XMLSPY.**

Escritorios (XML)

Para configurar el escritorio de un usuario del sistema es necesario crear un archivo (escritorio.xml) basado en el esquema proporcionado con la aplicación para editar el mismo (escritorio.xsd). A continuación daremos indicaciones detalladas de cómo cargar y configurar un escritorio para un usuario determinado.

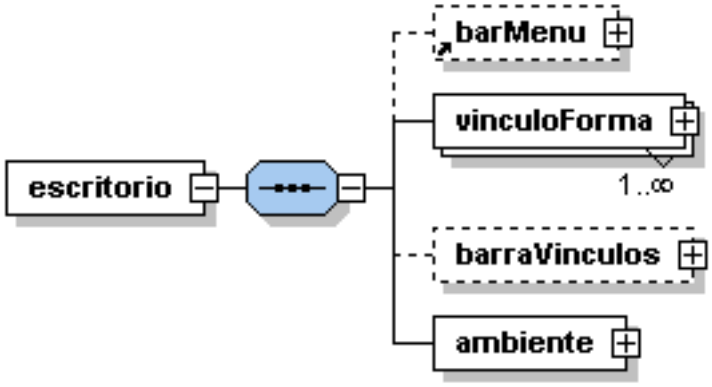
Para crear el escritorio.xml, utilizamos el XMLSPY, creamos el archivo y luego cargamos el esquema para que nos proporcione las reglas con las cuales podremos editar el archivo sin equivocaciones.

Estructura del esquema del escritorio:

El esquema es un árbol de Nodos, cada uno con propiedades.

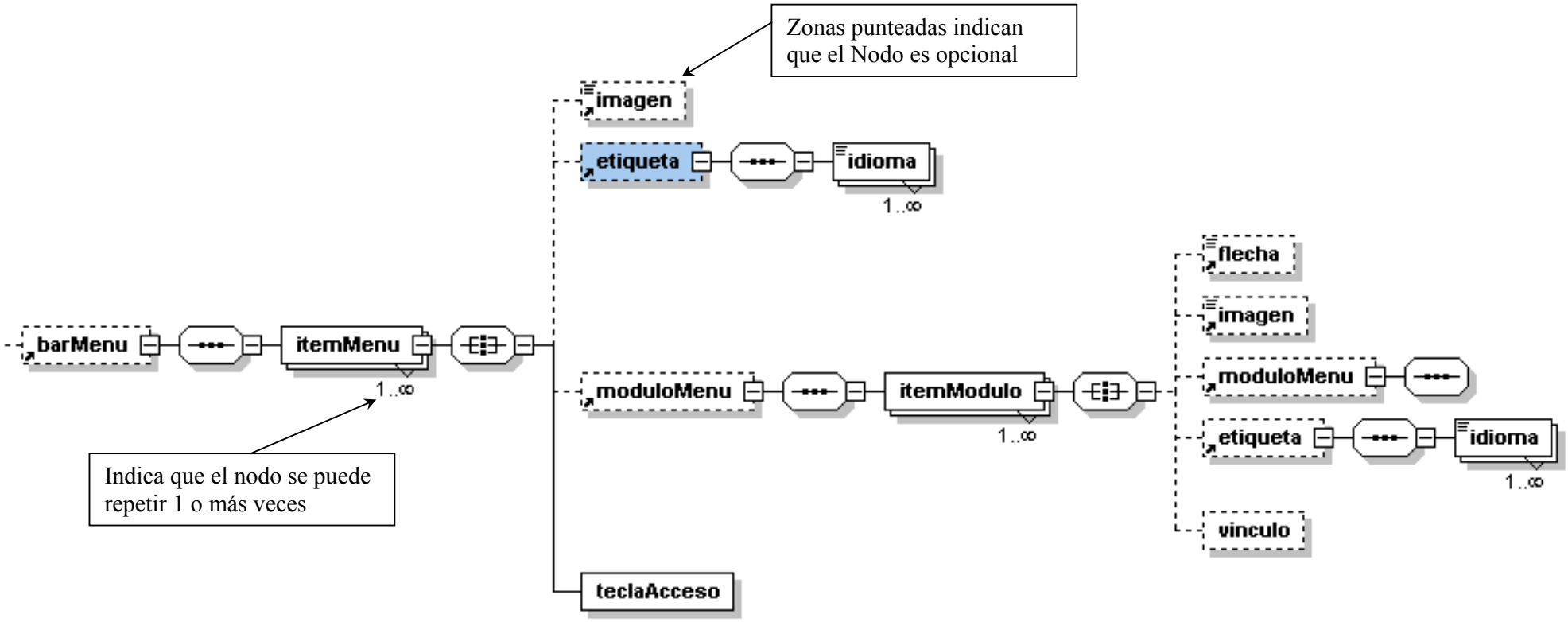
El nodo principal o raíz es **escritorio.**

A partir de este se derivan los demás:

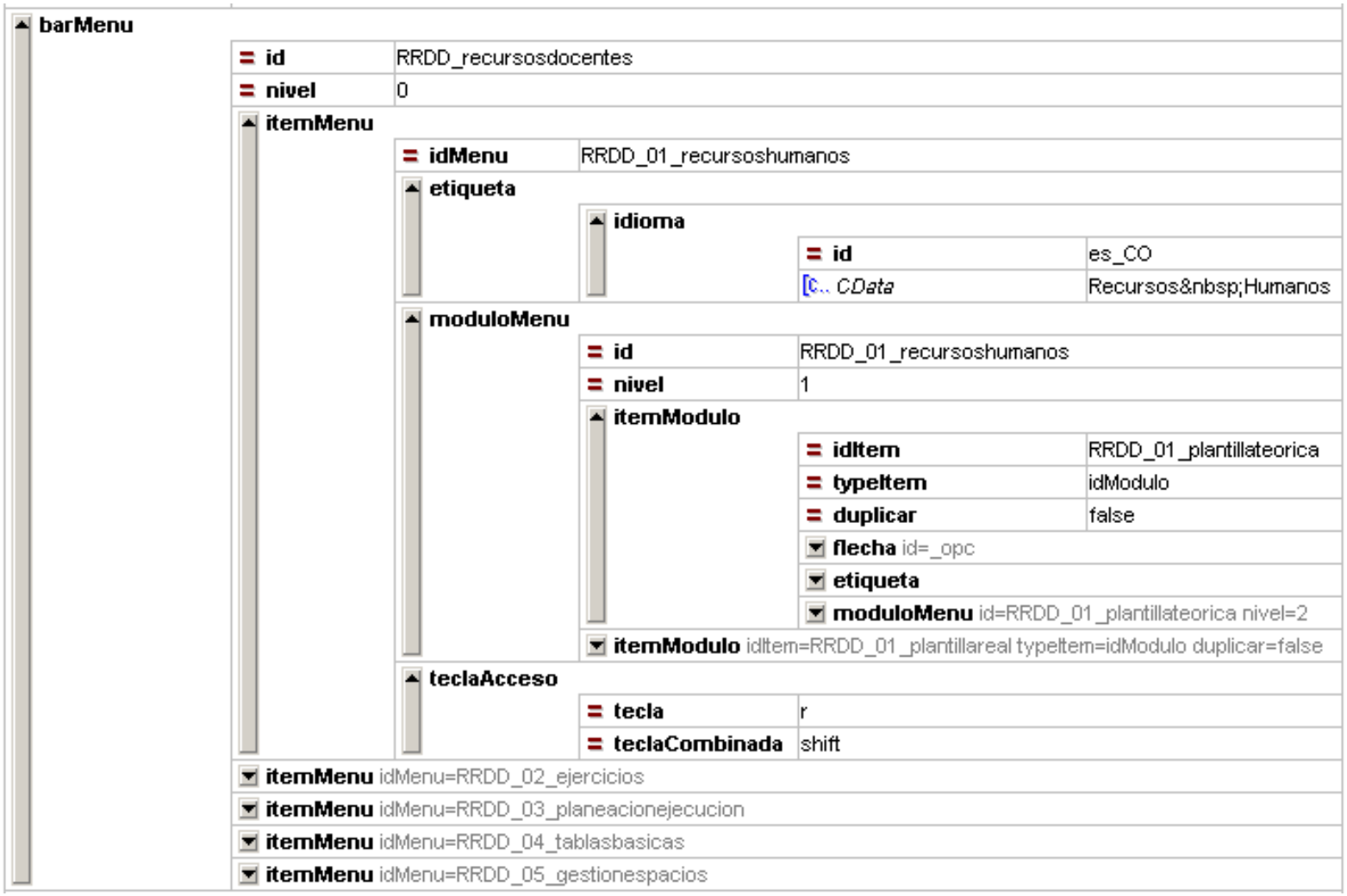

barMenu
vinculoForma
1..∞
escritorio
barraVinculos
ambiente

barMenu : Contiene el menú que va a ser desplegado para el usuario

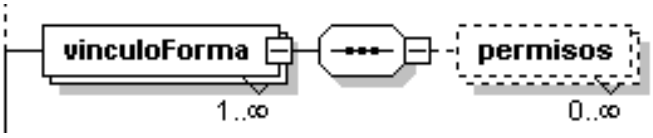
Detalle del archivo **.xsd**



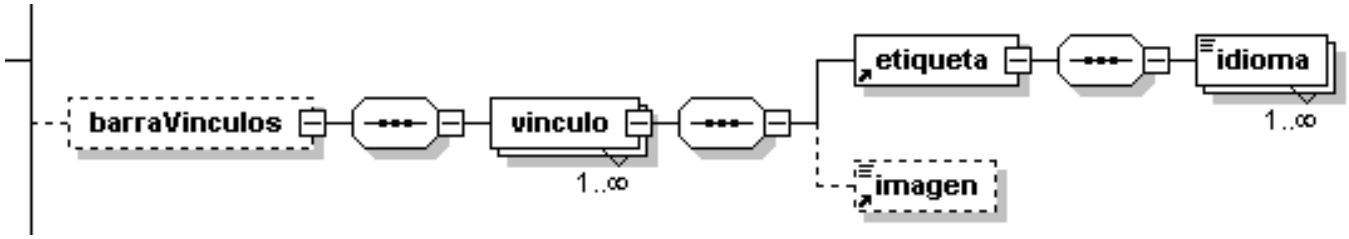
Detalle del archivo **.xml**



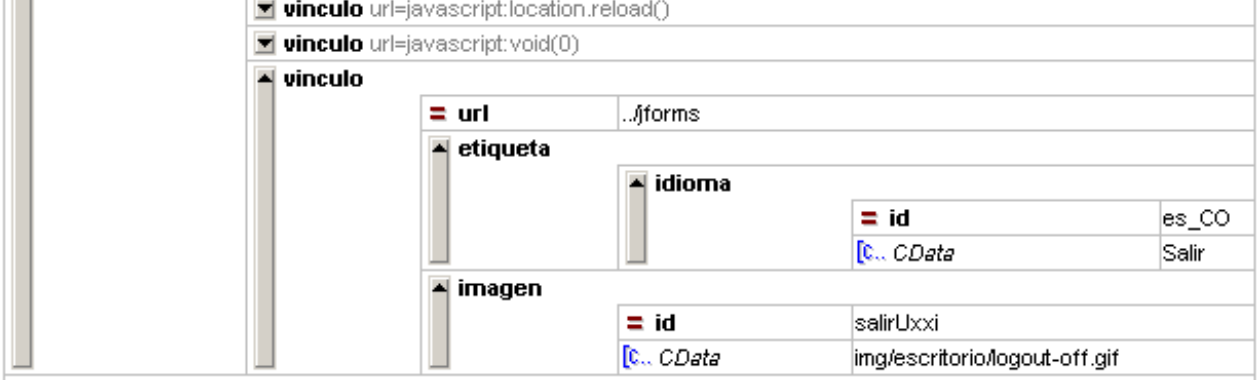
vinculoForma : Es utilizado para restringir los permisos a un usuario sobre determinada forma.



barraVinculos : **Es utilizado para crear vínculos en el escritorio del usuario, como "Salir", u otros que pueden direccionarse hacia una página en especial.**



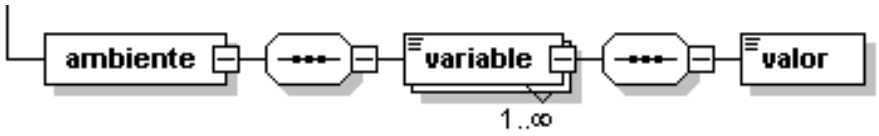
Detalle del archivo .xml

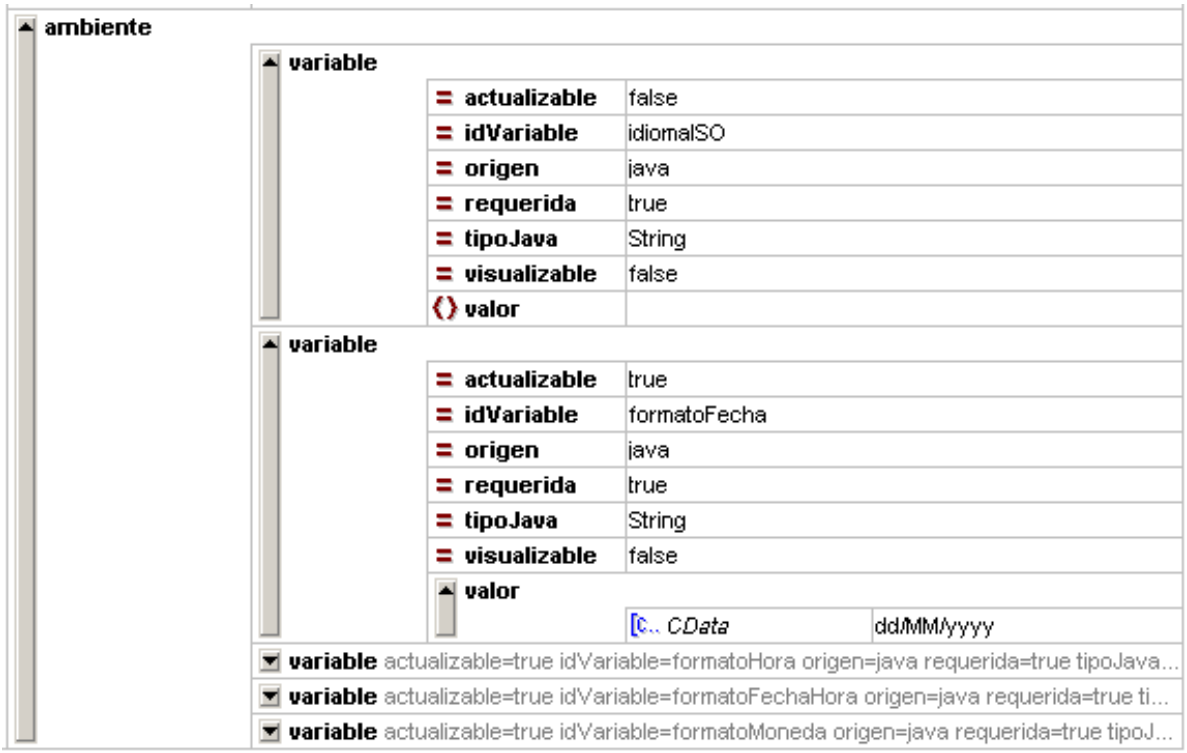


Visualización dentro de la aplicación:

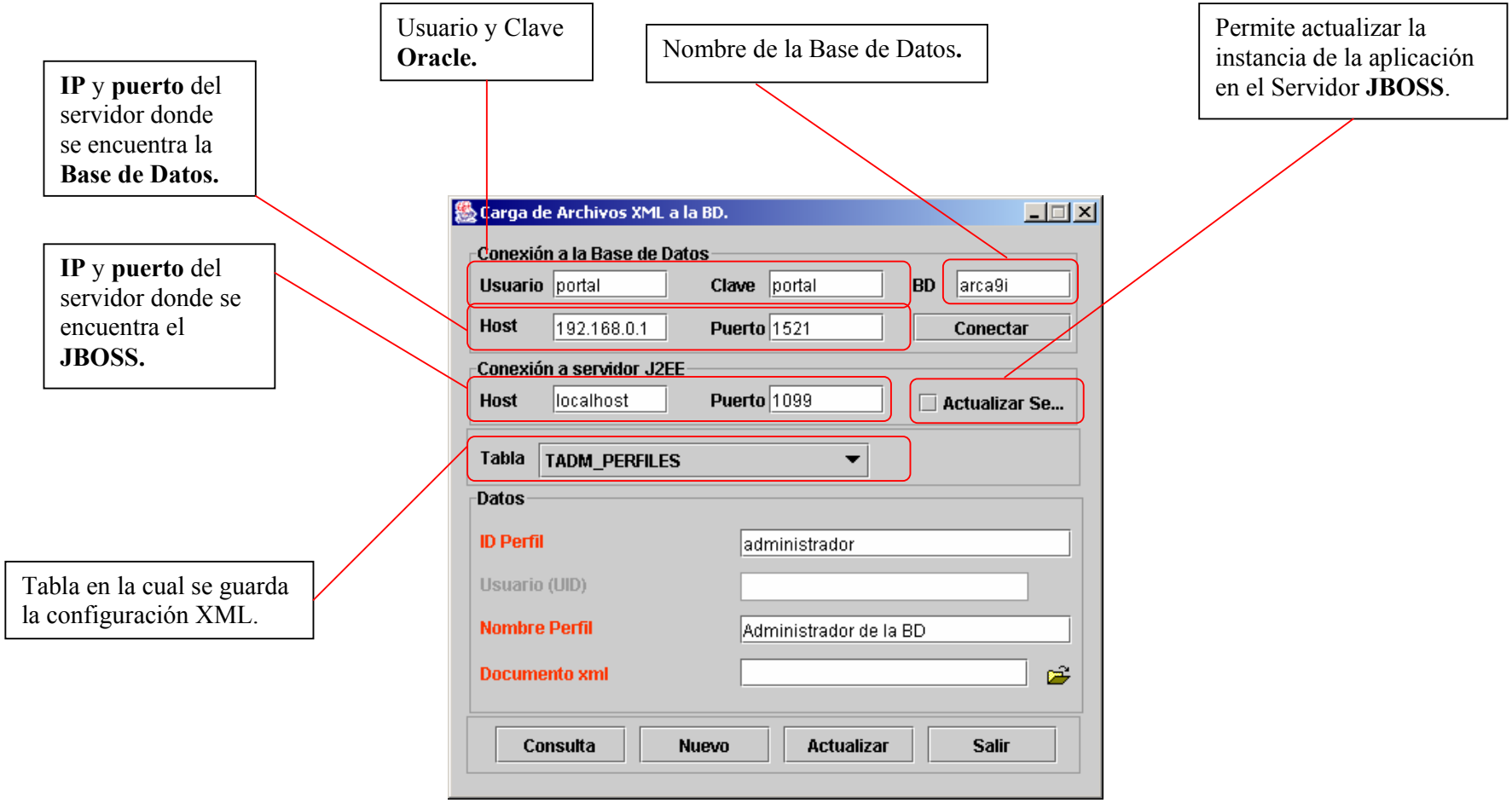
ambiente : Es utilizado para crear variables en el ambiente de ejecución de la aplicación, estas variables ya están predefinidas y son las mismas para cada entorno de usuario.

Detalle



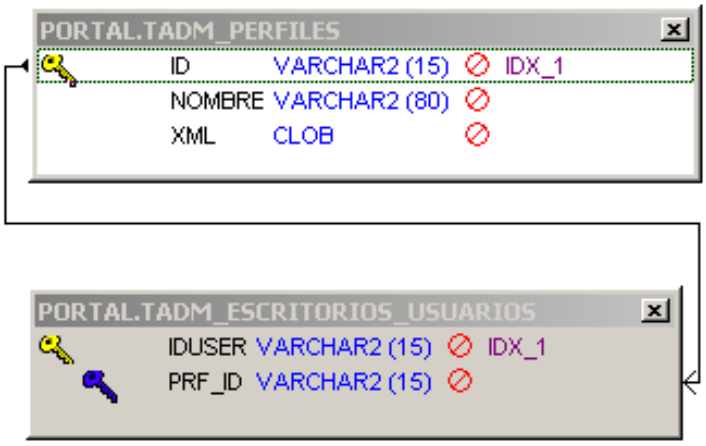


Luego de terminado el proceso de creación del archivo del escritorio pasamos a ejecutar la aplicación que se encuentra en utilidades, esta nos facilitará el trabajo de subir todo el código de configuración de archivos XML a la Base de Datos.



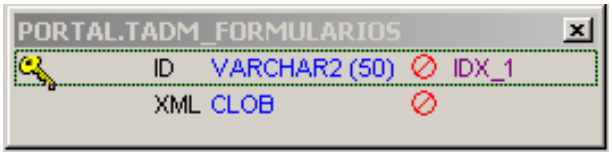
Detalle de las tablas del sistema.

TADM_PERFILES: Contiene la configuración del escritorio y los permisos que puede tener un usuario sobre determinada FORMA. Para ello se le asigna un ***ID*** al perfil, un ***NOMBRE*** (corresponde a la descripción del perfil), y el contenido XML del perfil en el campo del mismo nombre (***XML***).

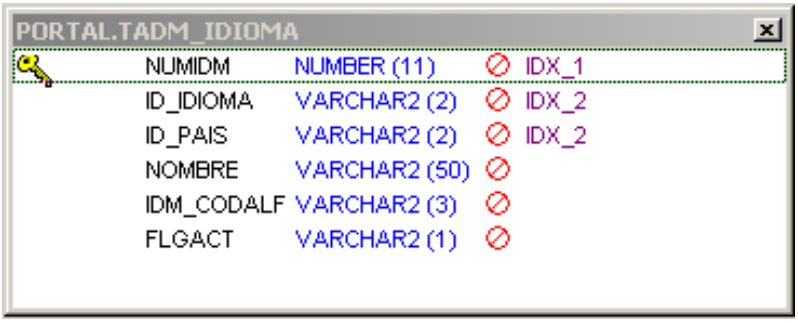


TADM_ESCRITORIOS_USUARIOS: Contiene los usuarios del sistema asociados a un perfil. Para ello se le da un nombre al usuario ***IDUSER*** y un perfil ***PRF_ID*** previamente configurado en la tabla **TADM_PERFILES**.

TADM_FORMULARIOS: Es la tabla más significativa del sistema, en ella se almacena la configuración de cada una de las formas que se desarrollan para el sistema, a las cuales se les asigna un ***ID*** (identificador de FORMA) que luego es utilizado para su llamado en el escritorio del usuario por medio del menú.



TADM_IDIOMA: El soporte de idiomas del sistema esta vinculado a esta tabla, ella contiene todos los posibles idiomas que deseen configurarse.



Para ello es necesario:

***ID_IDIOMA*:** Se especifica el idioma por medio de un identificador de 2 letras.
Ej.:

es (Español)
en (Inglés)
fr (Francés)

***ID_PAIS*:** Se especifica el país por medio de un identificador de dos letras.
Ej.:

CO (Colombia)
US (Estados Unidos)
FR (Francia)

***NOMBRE*:** Se especifica la descripción del Idioma y el país que corresponden a los identificadores antes mencionados.
Ej.:

Español – Colombia

Inglés – Estados Unidos

Francés - Francia

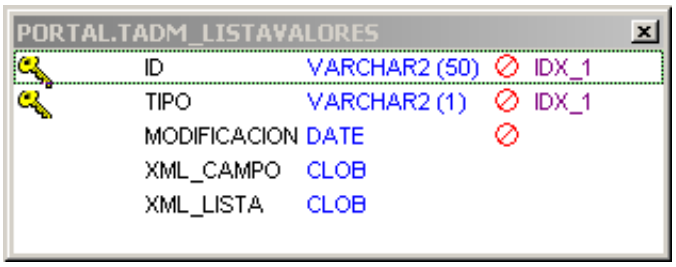
***IDM_CODALF*:** Se especifica un código de tres letras para identificar el país.

FLGACT: Se especifica si el idioma esta actualmente en uso, **1** para activar el uso del idioma y **0** para colocarlo inactivo.

TADM_LISTAVALORES: Se especifica las listas que se desplegaran en los listados de selección de la aplicación, estos pueden ser dinámicos o estáticos. Estas listas, al igual que los formularios, también son configuradas por medio de archivos XML, se componen de un ***ID*** (que junto a el ***TIPO*** de lista forman el identificador único de la tabla), se registra fecha en que fue creada la lista en el campo ***MODIFICACIÓN*** ,partiendo del ***TIPO*** (puede tener dos valores **D** = **Lista dinámica** y **E = Lista Estática**) de lista se llenan los otros campos de la siguiente manera:

XML_CAMPO: Si el valor del ***TIPO*** de lista es **D** este campo contiene el XML de configuración.

XML_LISTA: Si el valor del ***TIPO*** de lista es **E** este campo contiene el XML de configuración.

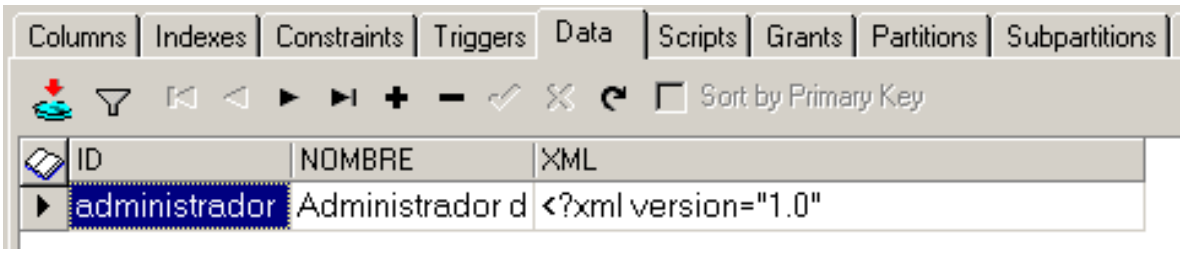


Vista dentro de la aplicación:

TADM_REFERENCIA: Esta tabla contiene todos los listados que asumimos pueden sobrepasar los 20 registros.

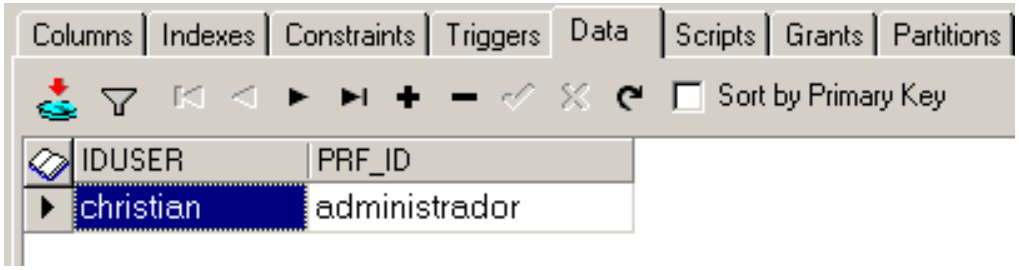
Luego de cargar en la base de datos los XML's, creamos un nuevo perfil para el usuario, en la Tabla **TADM_PERFILES,** el cual luego podremos reutilizar para otros usuarios.
Buscamos nuestro archivo **.xml** y actualizamos, si deseamos actualizar cambios sin tener que reiniciar nuestros contenedores, seleccionamos el check **Actualizar Servidor** el cual hará el trabajo por nosotros solo reiniciando el Contexto sin necesidad de reiniciar los servidores.

Detalle de la inserción de datos en la tabla (TOAD)



Luego asociamos en la Base de Datos el usuario al perfil en la tabla **TADM_ESCRITORIOS_USUARIOS.**

Detalle de la inserción de datos:

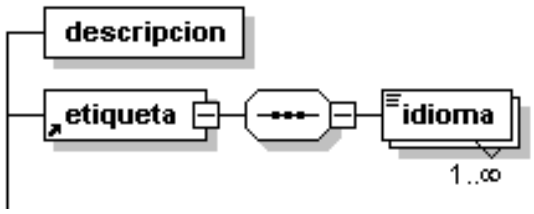


De esta forma podemos tener varios perfiles como opción excluyente para un usuario, o sea, para un usuario hay un único perfil asignado, esto es garantizado por la integridad del modelo en la Base de Datos.

Creación de archivos de configuración XML para una forma.

La configuración del un archivo XML de una "forma" comienza por conocer cada uno de los nodos y su implicación dentro del la operatividad de la misma.

El nodo raíz es **forma**



Detalle del archivo **.xml**

XML			
Comment	edited with XMLSPY v5 U (http://www.xmlspy.com) by pino (pino)		
Comment	edited with XMLSPY v5 U (http://www.xmlspy.com) by Desarrollo (JForms)		
forma			
	idFormulario	ar01_05	
	proceso	x	
	actividad	x	
	tipo	formulario	
	xmlns:*xsi*	http://www.w3.org/2001/XMLSchema-instance	
	xsi:noNamespace...	C:\cendesco\cvscendesco\produccion\xml\esquemas\forma.xsd	
	descripcion		
		CData	Tipos de Elementos
	etiqueta		
		idioma	
			codigo es_CO
			CData Forms
	presentacion		
	procesos requiereTX=false controlTX=aplicacion idProceso=ar01_02		
	envioForma		
	pivoteDatos		
	requerimientos		

Los nodos que complementan la configuración son los siguientes:

PRESENTACIÓN: En este nodo se describen los gráficos y los ID's que son utilizados para el mapeo de datos.

PROCESOS: En este nodo se describen las consultas y los paquetes para Insertar, Actualizar y Eliminar datos, se defines los ID's de los parámetros de las consultas y de los paquetes.

ENVIOFORMA: En este nodo se describen que ID's del nodo ***PRESENTACIÓN*** son mapeados hacia los ID's del nodo ***PROCESOS*** para que el motor de ejecución de consultas tenga en cuenta como llevar los datos hasta el cliente.

PIVOTEDATOS: Para efectos de hacer más rápidas las consultas debemos especificar en este nodo las llaves primarias, los campos foráneos, y otros campos con sus tipos equivalentes en Java.

REQUERIMIENTOS: Este nodo viene configurado con el idioma, ya que esta es una variable necesaria para la ejecución de las formas.